PROCEEDINGS OF THE FOURTH MEETING OF THE MINUTEMAN COMPUTER USERS GROUP

★ ★ ★ ★

REPORT MCUG-3-72

Meeting held

June 5—6, 1972

Silver Spring, Maryland

PROCEEDINGS* OF THE FOURTH MEETING OF THE MINUTEMAN COMPUTER USERS GROUP

EDITED
by
CHARLES H. BECK

Meeting held
June 5-6, 1972
Silver Spring, Maryland

Report MCUG-3-72

* SPONSOR:

Systems Laboratory
Department of Electrical Engineering
School of Engineering
TULANE UNIVERSITY

PREFACE

Of the nearly 1,000 reliable computers, originally costing $234,000 each, from the LGM 30 Minuteman ICBM Weapons System, approximately 400 have been declared excess by the USAF. These Minuteman D17B Computers can be classed as extremely flexible general-purpose minicomputers. Government activities, industrial contractors, universities, and other organizations have acquired these excess D17B computers for development and use in many fields of research, education, and other applications.

The Minuteman Computer Users Group (MCUG) was formed to provide for an effective information interchange and the various forms of assistance needed by the users. Those who are members of this cooperative, voluntary group assist each other by sharing results, programs, applications, interfacing techniques, maintenance procedures, and spare parts. The MCUG membership is in excess of 145 Government activities, industrial contractors, colleges, universities, and other organizations.

The fourth meeting of the MCUG was held at the Sheraton-Silver Spring Hotel in Silver Spring, Maryland on June 5-6, 1972. The registration list is included in the Appendix. The persons who attended this meeting numbered 67 and represented 46 organizations. Previous meetings have been held in Miami Beach on July 19-20, 1971, Houston on November 16, 1970, and Anaheim on June 11-12, 1970.

These *PROCEEDINGS* are a permanent record of the material presented at the meeting on June 5, 1972. This publication of the MCUG describes such topics as procurement, simulation, state description, design of a hardware divider, design of a binary display, and use of the D17B in a hybrid computer system and an automated data acquisition and waveform analysis system. The agenda also included a successful demonstration of the D17B/AutoAnalyzer

Analysis System developed in the Systems Laboratory at Tulane University under a research contract supported by the Army Medical R&D Command. This cost-effective development included an ASR35 Teletypewriter as the peripheral I/O device which provided for full alphanumeric communication with the D17B in a conversational interactive mode. The D17B/AutoAnalyzer Analysis System was delivered to the Division of Biochemistry at the Walter Reed Army Institute of Research where it is used for automated blood serum analysis. In addition to the technical sessions there was considerable exchange of information during the workshop sessions on June 6.

The assistance and encouragement of Mr. Richard F. Babler and Mr. John P. Bartell of the Defense Supply Agency are gratefully acknowledged. We also thank Mr. Billy G. Bass of WRAIR for the time and effort required to plan for demonstration of the D17B/AutoAnalyzer Analysis System at Walter Reed.

<u>Methods of Joining the MCUG</u>

1. Send a check or purchase order in the amount of $100 to the MCUG Chairman at the address given below. Specify MCUG membership and/or documentation for checkout, operation, and programming of the Minuteman D17B Computer.
2. Request an invoice for $100 to cover the items listed above.

Dr. Charles H. Beck
Professor of Electrical Engineering
Tulane University
New Orleans, Louisiana 70118

These *PROCEEDINGS* can be obtained by sending a check or purchase order for $20 to the MCUG Chairman at the address given above. MCUG members may [illegible] additional copies for $6.

The following documentation is included in the MCUG membership:

MCUG-1-71, D17B Computer Wire List and Logic Equations
MCUG-2-71, D17B Electronic Module Schematics
MCUG-3-71, Proceedings of the Third Meeting of the MCUG
MCUG-4-71, Minuteman D17B Computer Programming Manual
MCUG-1-72, D17B Power Supply Schematics
MCUG-2-72, Minuteman D17B Computer Programming Manual Supplement
MCUG-3-72, Proceedings of the Fourth Meeting of the MCUG

Charles H. Beck
Chairman, MCUG

MINUTEMAN COMPUTER USERS GROUP MEMBERSHIP

Aerospace Corporation, Los Angeles
AFCRL (LGW), Hanscom Field, Mass.
Air Force Institute of Technology, Engineering
Arizona State University, Electrical Engineering
Armstrong State College, Savannah, GA
Arnold Research Organization, Arnold AFB, TN
Augustana College, Physics, SD
Austin College, Computer Center
Ball State University, Physics
Beaver College, Psychology
Bluefield State College, Technology
Bowling Green State University, Psychology
Brigham Young University, Electrical Engineering
Buena Vista College, Electronics
Bureau of Mines, Laramie
Bureau of Mines, Pittsburgh
California Institute of Technology, Geological & Planetary Science
Center for Disease Control, DHEW
Cleveland State University, Physics
Christian Brothers College, Computer Science
Colorado State University, Atmospheric Science
Delaware River Basin Comm., Trenton
Department of State, Nassau
Des Moines Area Community College, Electronics
Dillard Univ., Mathematics & Physics
Drexel University, Electrical Engr.
Duke University, Electrical Engr.
Eastern Michigan Univ., Ypsilanti
Eastern Washington State University, Physics
Einstein College of Medicine, Radiology
Fighton, Inc., Rochester, NY
Florida State University, CAI Center
Fredericksburg Geomag. Ctr., Corbin
Ft. Belvoir, Electronics
Glastonbury High School, Physics
Goddard Institute for Space Studies
Hahnemann Medical School, Radiation
Harvard University, Physics
Haskell Indian Jr. Coll., Voc-Tech.
Heidelberg College, Physics
Hughes Aircraft Corp., Culver City
Indiana University of Pennsylvania, Physics
Johns Hopkins University, Chemistry
Kansas State College, Pittsburg, Industrial Technology
Knox College, Physics
Kutztown State College, Physical Sci.
LSU School of Medicine, Neurology
LSUNO, New Orleans, Science
Lowell Technical Inst., Lowell, Mass.
Linfield College, Research Institute
Mankato Area Voc-Tech. School, Computer Maintenance
Mass. General Hospital, Boston
McDonnell Douglas Corp., St. Louis
Medical University of South Carolina, Neurosurgery
Merchant Marine Academy, Computer Science
Merrimack College, Electrical Engr.
Michigan State University, Cyclotron Laboratory
Michigan Technological University, Electrical Engineering
Milwaukee Area Technical College, Electronics
MIT, Charles Stark Draper Laboratory
MIT, Educational Research Center
NASA, MSFC, Huntsville
Nat'l Bureau of Standards, Wash., DC
Nat'l Center for Health Statistics
Nat'l Library of Medicine, DHEW, PHS
Naval Ordnance Station, Indian Head
New Mexico State University, Electrical Engineering
New York Institute of Technology, Electronic Technology
Newark College of Engineering, Mechanical Engineering
Northwestern University, Material Science
Occidental College, Physics
Ocean Systems, Inc., Reston, Virginia
Oklahoma State University, Biochemistry

MINUTEMAN COMPUTER USERS GROUP MEMBERSHIP (Continued)

Oklahoma State University, Physics
Pennsylvania State University, Chemistry
Polytechnic Institute of Brooklyn, Electrical Engineering
Princeton University, Computer Center
Purdue University, Aeronautics, Astronautics & Engineering Science
Raytheon Corporation, Bristol, Tenn.
St. John Fisher College, Physics
San Diego State College, Biology
Singer-Kearfott, New Jersey
Sloan-Kettering Institute, Biophysics
SMU, Computing Laboratory
Southwest Minnesota State College, Electronics
Space Rad. Effects Lab., Newport News
Stanford University, Linear Accelerator Center
State University College, Brockport, Data Processing Services
State University of NY, Mat. Science
Stephen F. Austin State Univ., Physics
Stevens Institute of Technology, Electrical Engineering
Technitrol, Inc., Philadelphia
Tektronix, Inc., Atlanta
Teledyne-Ryan Aeronautical, San Diego
Tennessee Technological University, Mechanical Engineering
Texas A & M University, Physics
Tulane University, Electrical Engr.
Union Carbide (Nuclear), Oak Ridge
USDA, ARS, Ames
USDA, Research Service, Beltsville
University Computing Co., Dallas
Univ. of Akron, Electronic Tech.
University of California, Berkeley, Electrical Engineering
Univ. of Colorado, Electrical Engr.
Univ. of Arizona, Optical Sciences
Univ. of Dallas, Chemistry
Univ. of Delaware, Electrical Engr.
University of Florida, Metallurgical & Materials Engr.
University of Florida, Ophthalmology
Univ. of Houston, Electrical Engr.
Univ. of Illinois, State Water Survey
Univ. of Iowa, Neurobiology
University of Kentucky, Mechanical Engineering
University of Miami, Physics
Univ. of Michigan, Aeorspace Engr.
Univ. of Mississippl, Chemistry
University of Missouri-St. Louis, Chemistry
Univ. of Nebraska, Electrical Engr.
Univ. of Nevada, Reno, Electr. Engr.
Univ. of Nevada, Las Vegas, Physics
Univ. of New Hampshire, Electr. Engr.
Univ. of Oklahoma, Mathematics
Univ. of Pennsylvania, Geology
Univ. of Penn., Johnson Foundation
Univ. of Pittsburgh, Pharmacology
Univ. of South Florida, Physics
Univ. of Texas, Applied Research Lab.
Univ. of Trieste, Italy, Geodesy
Univ. of Virginia, Psychology
Univ. of Washington, Psychology
Univ. of Wisconsin, Computer Science
Univ. of Wisconsin, Electrical Engr.
Univ. of Wyoming, Electrical Engr.
USAFSAM, Medical Systems Division
V. A. Hospital, Lexington
V. A. Research Hospital, Chicago, Theraputic Radiology
V. A. Research Hospital, Gainesville, Nuclear Medicine
Virginia Institute of Marine Science, Gloucester Point, Virginia
Washington University, Psychology
Washington & Lee University, Physics
West Virginia University, Electrical Engineering
Wisconsin State University, Engineering Mechanics
Worcester Foundation for Experimental Biology, Shrewsbury, Mass.
Wright State University, Computer Center

TABLE OF CONTENTS

MINUTEMAN D17B COMPUTER PROCUREMENT

Approximately 800 Minuteman D17B Computers are expected to be declared excess by the USAF through 1974. The original acquisition cost per system was approximately $234,000. These computers can be acquired by qualified agencies, contractors, and grantees as the systems become available through appropriate ADPE reutilization agencies on an "as is" non-reimbursable basis as follows:

DoD Agencies

Contact respective service Hqs. for ADPE Acquisition for approval and for forwarding of Requisition Form 1419 to DARO.

DoD Agency Contractors and Grantees

Contact respective contracting officers for approval and for forwarding of Form 1419 to Defense Supply Agency, DSAH-DARO, Cameron Station, Alexandria, Virginia 22314.

Civil (Non-DoD) Agencies of the Federal Government

Contact respective Office for ADPE Acquisition for approval and for forwarding of Transfer Order Form 122 to GSA Excess Equipment Utilization Branch, Crystal Mall Bldg. 4, Washington, DC 20406.

Civil Agency Contractors and Grantees

Contact respective contracting officers for approval and for forwarding of Form 122 to GSA as listed previously.

Authorized Donees

Contact respective state surplus property offices for acquisition through DHEW Office of Surplus Property Utilization, 4452 DHEW North Bldg., Washington, DC 20201.

MINUTEMAN D17B COMPUTER DESCRIPTION

Functional Capabilities and I/O

The D17B is a small general-purpose computer. It is totally programmable and has the capabilities of: receiving and sampling analog signals, digital data, or pulse-type input signals; logical decision-making and performance of arithmetic operations using an instruction set of 39 machine language instructions; and transmission of output data in the form of analog, digital and pulse type signals under program control. Because of the extremely flexible I/O capability of the D17B, it can be quite useful in a wide variety of applications.

Central Processing Unit and Control

Since the D17B is a serial-binary computer, simultaneous access to all the bits of a memory location is not needed either for instructions or data. Hence, the arithmetic registers need not be constructed entirely of flip-flops. Instead, they are in the form of circulating loops in memory. The D17B has four double-rank arithmetic registers which are Accumulator (A), Lower Accumulator (L), Instruction Register (I), and Number Register (N). Because the L-register is addressable, it can be used as rapid-access storage in addition to performing normal arithmetic functions. There are two non-addressable arithmetic registers, the I- and N-registers, which are used without programmer control and one 3-bit pseudo-index (phase) register.

The central processing unit (CPU) has I/O access to four rapid-access memory loops of 1, 4, 8, and 16 words in addition to the main memory which is arranged in 21 channels of 128 words each. Two input buffer loops of four words each provide additional input capability to memory in the form of direct data entry. These are the V- and R-loops which can also be used as general-purpose rapid-access memory loops.

Programmed data channels cause data transfers into the arithmetic registers. All machine functions are processed and interpreted in the CPU. The memory channel address from which the next instruction is to be taken is determined by the location counter. When the CPU is ready to accept another instruction from memory, the address is specified by the channel address stored in the location counter and the sector address specified in the previous instruction.

The phase register can modify the operand address of one of the multiply instructions. This register also serves as a selector switch for choosing one of two pairs of inputs to one of the incremental pulse-type input loops and for selecting one of four external positions for each of the three D-A analog voltage outputs.

The Accumulator holds the results of all arithmetic operations and serves as an output register for parallel digital data, pulse-type signals, D-A analog voltage outputs, and telemetry data. The Lower Accumulator is involved in certain arithmetic, input, and logical operations. A real-time clock is provided by internal timing signals derived from the clock channel of the disc memory.

Specifications

The D17B is basically composed of two semi-circular sections. One half contains the power supply circuit cards which generate the various dc voltages required in the computer and a 400 Hz 3ϕ signal for providing power to the motor in the 6000 rpm disc memory. The other semi-circular section contains the discrete DRL and DTL logic components of the computer itself. Some of the detailed specifications for the D17B Computer are given in the following table. The high degree of reliability and ruggedness of the computer are evidenced by the strict requirements of the Minuteman ICBM Weapons System.

MINUTEMAN D17B COMPUTER SPECIFICATIONS

Manufacturer: Autonetics, a division of North American Rockwell

Model: D17B

Year: 1962

Type: Serial, synchronous

Number System: Binary, fixed point, 2's complement

Logic Levels: 0 or False, 0 Volts; 1 or True, -10 Volts

Data Word Length (bits): 11 or 24 (double-precision)

Instruction Word Length (bits): 24

Maximum I/O (words/s): 25,600

Number of Instructions: 39 types from a 4-bit op code by using five bits of the operand address field for instructions which do not access memory.

Execution Times:

Add (μs): 78 1/8
Multiply (μs): 546 7/8 or 1,015 5/8 (double precision)
Divide: (Software)
(Note: Parallel processing such as two simultaneous single precision operations is permitted without additional execution time.)

Clock Channel: 345.6 KHz

Addressing: Direct addressing of entire memory
Two-address (unflagged) and three-address (flagged) instructions

Memory:

Word Length (bits): 24 plus 3 timing
Type: Ferrous-oxide-coated NDRO disc
Cycle Time (μs): 78 1/8 (minimal)
Capacity (words): 5,454 or 2,727 (double precision)

Input/Output:

Input Lines:	48 digital
Output Lines:	28 digital
	12 Analog
	3 Pulse
Program:	800 5-bit characters/s

Physical Characteristics:

Dimensions:	20" high, 29" diameter
Power:	28 V dc ± 1 V at 19 A
Circuits:	DRL and DTL. Double copper clad, gold plated, glass fiber laminate, flexible polyurethane coated circuit boards.

Software: Minimal delay coding using machine language modular special-purpose subroutines.

Reliability: 5.5 years MTBF

SOFTWARE SIMULATION OF THE MINUTEMAN D17B COMPUTER

Capt Bruce Chatterton   Gary B. Lamont
Electrical Engineering Dept
Air Force Institute of Technology
Wright-Patterson Air Force Base, Ohio 45433

ABSTRACT

A software program has been developed 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 was developed at the Air Force Institute of Technology as an aid to research in the D17B Computer utilization program. The simulation program can be used as a teaching aid, for executing D17B programs, and for debugging program tapes to be run on the D17B Computer. The simulation program consists of a main program and eight subroutines. A programming language was developed for the D17B Computer Simulation Program which contains numbers and load codes, switches, and miscellaneous commands.

I. Introduction

A software simulation of the Minuteman D17B Computer has been developed at the Air Force Institute of Technology (AFIT) (Ref 2). The general development objectives and results of this simulation are presented in this paper.

The purpose for developing the D17B Computer Simulation Program was to create an aid that would be useful to the research effort of the D17B Computer utilization program. This research effort is concerned with getting a D17B Computer operational in a laboratory environment and finding useful applications.

The simulation program has shown itself to be useful in many areas. The simulation program can be used in learning the basic operations of the D17B Computer. It can also be used as backup capability for running D17B programs when the actual computer is not available. Its most important use, 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 execution-time error checking capability.

The capability for entering D17B programs from punched tape has been incorporated in the D17B Computer at AFIT. Provisions were also made in the simulation program for reading and executing the data from these same punched tapes. Therefore, the simulation program can be extremely helpful in the preparation of the program tapes which are to be read into the D17B Computer. The simulation program helps in the preparation of the program tapes by detecting and locating invalid symbols punched on the tape, by decoding the program instructions, and by detecting addressed locations in memory that are out of range of the program being executed.

<u>Problem Statement and Objectives</u>. The prime objective of the D17B Computer simulation program was to simulate the functions of the D17B Computer. To pursue this objective, the following criteria were established:

1. The simulation program was to simulate the D17B Computer at the register transfer level. A register transfer approach was used because it allowed the D17B 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. The register transfer approach also allowed for easability in tracing the information flow in the simulated computer as data is loaded and programs executed.

2. The FORTRAN Extended Language was the computer simulation language chosen for writing the simulation program. This language was chosen because of access to a computer system which contained the FORTRAN Extended Compiler. The Computer Design Language (CDL) described in reference 3 was used in

writing portions of the simulation program, but because of the nonavailability of a CDL compiler, a transformation to the FORTRAN Extended Language was made (Ref 3, Chaps 1-5). CDL is much more descriptive of computer operations than FORTRAN.

3. The simulation program was to simulate the actual computer as closely as possible. The same algorithm implemented on the D17B Computer was used in the simulation program for most functions. 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. In the areas where a close simulation could not be realized, a quasi-simulation was used. The quasi-simulation uses the same register inputs and generates the same results, but the method of obtaining the results differ.

4. The real-time control functions (Fine Countdown and Incremental Inputs) of the D17B were not to be included in the simulation program. The Fine Countdown function involves the V-loop and the U-loop forming a digital integrator which operates without program control. The Incremental inputs are inputs to the D17B which are incrementally supplied to the V-loop and R-loop without program control. The instructions associated with these functions could be added by a user who is researching the area of real-time control applications for the D17B.

The remainder of this paper will be devoted to a description of the organization and structure of the D17B Simulation Program and the D17B Computer Simulation Language.

II. <u>D17B Computer Simulation Program</u>

The organization and structure of the D17B Computer Simulation Program will be described in this section. The simulation program simulates the D17B Computer at the register transfer level and is written in the FORTRAN Extended Language to be run on the Intercom System (teletype) of a Control Data Corporation (CDC) 6600 Computer System.

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 set to the proper position 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:

1. Reading and Translation Section
2. Noncompute Mode Section
3. Compute Mode Section

Fig. 1 shows the program flow between these sections of the main program and the subroutines.

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. Input data is read as alphabetic and numeric characters. This data is then interpreted as octal or binary data, a D17B load code, a switch designation (setting), or a miscellaneous command. The miscellaneous commands are responsible for a variety of functions which include the following: register and memory display, discrete data storing, incremental data storing, and mode tracing. A transfer of operation to the noncompute mode or one of the subroutines is made to utilize this data.

The Noncompute Mode Section of the simulation program simulates the noncompute mode of the D17B computer. The noncompute mode is responsible for synchronizing, idling, preparing to load, preparing to compute, loading data into memory, and verifying the contents of memory.

The Compute Mode Section of the simulation program simulates the compute mode of the D17B Computer. The compute mode is responsible for searching, reading, and writing memory and instruction execution.

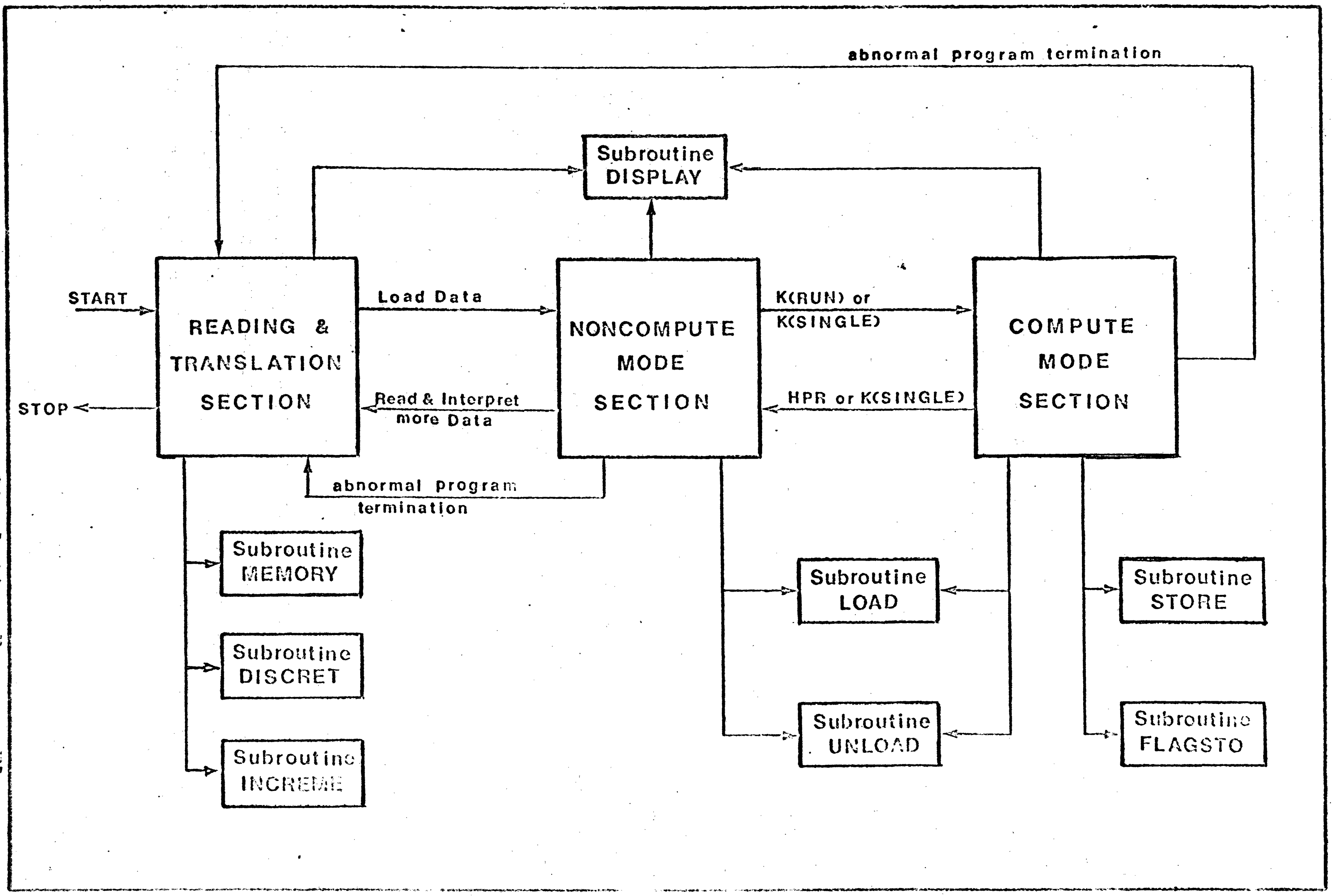


Fig. 1. D17B Computer Simulation Program Flow

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. Subroutines falling into this category are Subroutine LOAD, Subroutine UNLOAD, and Subroutine DISPLAY. Subroutine LOAD provides the function of loading the contents of the accumulator into addressed memory locations. Subroutine UNLOAD performs the function of unloading an addressed word of memory. The information unloaded is then used either as an instruction or an operand. Subroutine DISPLAY provides the simulation program with the capability of displaying the binary contents of all registers and loops which are specified by the user. The contents of a register or loop is provided as output only when the contents of that register or loop changes.

2. Those functions which are only called from one place in the main program, but which are of such importance that a separate location is beneficial to the organization of the simulation program are also subroutines. Subroutines in this category are Subroutine STORE, Subroutine FLAGSTO, and Subroutine MEMORY. Subroutine STORE implements the D17B store (STO) instruction, which stores the contents of the accumulator in the memory location specified by the instruction register. Subroutine FLAGSTO performs the function of deciphering the flag store locations bits of the instruction register. The contents of the accumulator are then stored in the deciphered channel at the sector address associated with the first wordtime of execution of the present instruction. Subroutine MEMORY provides the capability of displaying the contents of memory (channels 00 thru 50) whenever a memory command is used. Only those portions of memory that have been written into since memory was last initialized will be shown in the output listing.

3. Those functions which will not be used very frequently and could be removed from the simulation program if it was determined that they were not really needed are also subroutines. 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 INCREME. Subroutine DISCRET provides the capability of entering discrete data and storing it for use in a program using the discrete input instructions (DIA or DIB). Subroutine INCREME provides the capability for entering quasi-incremental data into the four words of the V-loop or the four words of the R-loop.

The D17B Computer Simulation Program requires approximately 35K of core memory to execute on the CDC 6600 Computer. The majority of programs require between two and five seconds of central processor time. In five seconds, approximately 1000 D17B instructions can be executed by the simulation program.

III. D17B Computer Simulation Language

The D17B Computer Simulation Language is the programming language which was developed as the input data for the D17B Computer Simulation Program. For purposes of describing this language, it has been divided into the following categories:

1. Numbers and Load Codes
2. Switches
3. Miscellaneous Inputs and Commands

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

Three different representations of the numbers and load can be specified and will be accepted by the simulation program as valid data. These three representations are Octal, Binary, and ASCII. The Octal representation represents the type of input that would be supplied from a teletype keyboard or switches on a control console. The Binary representation represents the type of input which appears on the character input lines going into the D17B

Computer. The ASCII representation represents the type of input data on a punched tape which can be entered into the D17B Computer by a tape reader.

The numbers and load codes in the three representations are as follows:

	Octal	Binary	ASCII
	Representation	Representation	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	?

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

Miscellaneous Inputs and Commands. The simulation language in this category provides many functions. The functions that will be described are listed as follows:

Register and Memory Display
Mode Tracing
Execution Specification

Register and Memory Display. The binary contents of any of the registers (A, I, L, N) or loops (U, F, E, H, V, R) 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.

To display the contents of memory (channels 00 thru 50), 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.

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

<u>Execution Specification</u>. 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 were 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.

Other miscellaneous inputs and command provide the capability of setting and clearing flipflops, initialization of the contents of memory and certain specified variables, storing of discrete input data, and storing of quasi-incremental input data.

<u>Programming Methods</u>. D17B programs are executed on the simulated computer by arranging the simulation language in a program form. D17B programming techniques are described in the Minuteman Computer User's Group Programming Manual (Ref 1). The simulation program allows data to be input without a format, so a programmer can write a continuous program with each simulation language word separated by a blank.

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 word 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.

An example program which was run on the simulated computer has been included in this paper as Appendix A.

IV. Conclusion

The software simulation program of the D17B Computer presented in this paper has been operational since November 1971. The simulation program was developed to simulate the functions of the D17B Computer. One of the objectives of this simulation was 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 to be run on the D17B Computer.

Bibliography

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

2. Chatterton, B. Software Simulation of the Minuteman D17B Computer. Master Thesis. Wright-Patterson AFB, Ohio: Air Force Institute of Technology, Department of Electrical Engineering, March 1972.

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

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

        *******************************************************
        **                                                   **
        **                   D17B COMPUTER                   **
        **                SIMULATION PROGRAM                 **
        **                                                   **
        **     DATE=  03/07/72              TIME=  14.28.58  **
        **                                                   **
       *******************************************************


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

$ ADDITION RIPPLE PROGRAM
PR(ON) MR(ON) FS(ON) EW(ON) FILL
44010002 ENTER 64010002 ENTER CLEAR 1 ENTER
MEMORY(OCTAL) MR(ON) EXECUTE(0004) SIGNAL REGISTER(A) K(RUN)
 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)


NO. OF EXECUTIONS SPECIFIED :    4
SIGNAL ON - MODES WILL BE TRACED

O(4) O(2)O(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

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=     .777  SEC
14.33.24.STOP
```

APL SIMULATION OF THE D17B

HARRY S. WARFORD, CAPT, USAF, BSC*

Introduction

A simulation of the D17B serves a broad spectrum of applications. It allows a rapid development of software by not only emulating the basic machine, but by providing an inexpensive and rapid means of providing a large array of outputs. All manner of I/O devices can be simulated for development when the actual application may be dedicated and require few, if any, I/O devices. Additionally, the simulation is useful where no D17B exists. Students can receive hands-on experience with many types of machines by merely calling on a simulation such as the one under development here. Program debugging likewise proceeds at an accelerated rate since all the powers of a large system are available with built-in tracing routines.

Program Development

This simulation is by no means complete at present but has been developed in strict accord with actual machine procedure so as to render it easily expandable to a full simulation. The serial nature of the D17B has been preserved at the word level by controlling the simulation with a sector counter advancing one sector at a time as in the rotation of the disc memory. Figure 1 is a simplified flow chart for the machine and illustrates how each phase is controlled by tests performed on the sector count.

Development from this flow chart proceeded with APL on an IBM 360 series system and later on an IBM 370 series system.** APL has proven to be an ideal language for this simulation due to its inherent capability to handle vector quantities. This was the author's first encounter with APL, hence many of the expressions are not as efficient as they could be. However, the development proceeded with few difficulties to the wide choice of APL operators.

The main program illustrated in Figure 2 was first developed with dummy instructions in place of the execution routines. Those routines not yet implemented are left in as dummy statements providing only a printed indication of proper decoding. As development continues, some of these will be deleted entirely as they produce outputs that cannot effectively be simulated or have no apparent use in a general purpose system.

*To be presented by Major M.A. Jenkin, USAF, MC
**This simulation was started during a course taken by the author at Trinity University, San Antonio, Texas.

Along with the basic program, several short routines are provided to simulate necessary panel switches to allow program loading and execution start. These are given in Figure 8 and will be discussed in greater detail following the discussion of the main routine and execution routines.

At present, twenty-two instructions have been successfully simulated. Most of these were straight-forward but for clarity all are listed here with comments as to considerations given for simulation.

CLA: Clear and Add. Present operand, now in N-register, replaces contents of A-register

ADD: Add. Contents of A-register and N-register added modulo 16777216.

SUB: Subtract. Contents of A-register and complement of N-register added modulo 16777216.

MPY: Multiply. Sign of product predetermined; contents of A-register saved in L-register; rounded product of magnitudes formed then corrected for proper sign. Sector counter advanced 12 additional counts.

SAD: Split Add. Contents of N-register and A-register decoded into split format and center bits of A-register saved. Split words added independently but simultaneously modulo 2048. A-register reassembled.

SSU: Split Subtract. Contents of N-register split and complemented. Jump to SAD1 to complete as normal split add.

SMP: Split Multiply. Middle of L-register saved; contents of A-register encoded into split word format and saved criss-cross fashion in L-register. N-register encoded into split word format and signs independently but simultaneously predetermined. Products of magnitudes formed then corrected according to each predetermined sign. During process, products are rounded. Sector counter incremented 12 additional counts.

COM: Complement. Contents of A-register complemented by subtraction.

MIM: Minus Magnitude. If contents of A-register are not negative they are forced negative by jumping to COM.

ANA: And to A. Contents of L-register and A-register are encoded into 24-bit vectors and logically anded bit by bit. REsults are decoded into 24 place binary number and left in A.

ARS: Accumulator Right Shift. Contents of A-register shifted right by division with simulated loss of rightmost bits by floor value if original A-register not negative. For negative A-register, complement of A-register is first shifted then complemented to provide for extension of sign bit. In either case, sector counter is incremented appropriate number of counts as determined by number of places shifted.

ALS: Accumulator Left Shift. Contents of A-register shifted left by multiplication and limited to 24 bits by residue modulo 16777216. Sector counter incremented appropriate number of counts as determined by number of places shifted.

SAR: Split Accumulator Right Shift. Contents of A-register encoded into split word format and middle bits saved. Each half word shifted right by scheme similar to ARS. A-register put back together and sector counter incremented appropriate number of counts.

SAL: Split Accumulator Left Shift. Contents of A-register encoded into split word format and middle bits saved. Left shift of each half word proceeds as in ALS. A-register reassembled and sector counter incremented appropriately.

SLR: Split Left Word, Right Shift. Contents of A-register encoded into split word format and middle bits and right word protected while left word shifted right. A-register reassembled and sector counter appropriately incremented.

SRR: Split Right Word, Right Shift. Contents of A-register encoded into split word format and middle bits and left word protected while right word shifted right. A-register reassembled and sector counter appropriately incremented.

SLL: Split Left Word, Left Shift. A-register encoded into split word format and middle bits saved. Left half word shifted left; jump to SAL1 to reassemble A-register and adjust sector counter.

SRL: Split Right Word, Left Shift. A-register encoded into split word format and middle bits saved. Right half word shifted left; jump to SAL1 to reassemble A-register and adjust sector counter.

TRA: Transfer. Fetch instruction specified by transfer instruction. Change active channel register.

TMI: Transfer on Minus. If contents of A-register positive continue to next instruction. If negative execute TRA.

STO: Store. Correct operand address to allow for physical placement of write head and store contents of A-register at the corrected address.

HPR: Halt and Proceed. Type out PROGRAMMED HALT and proceed only after GO has been typed into terminal.

Results of Execution Routine

Example of instructions were prepared as three-line programs with a simulated binary display providing the output. Additionally, a longer program was prepared and the accumulator monitored by a simulated octal display. The results are too lengthy to present here but have proven to be faithful copies of the machine results.

Utility Routines

As mentioned earlier the main program is supported by short simulations of pertinent panel functions. The three programs in use to date are listed in Figure 3. These programs treat the instructions and data as though they were eight place, octal, whole numbers whereas the D17B number range is approximately $\pm$ 1.

MRC simulates the master reset function and presets the I-register to TRA to channel 0, sector 0. The FILL Routine accepts the octally coded instructions and data with the exact coding used in the actual D17B system at the School of Aerospace Medicine. RUN places the computer into operation. No equivalent to halting the computer by moving the switch out of RUN has been implemented. Instead, the attention button is being used.

Conclusions and Projections

An effective simulation with considerable attention to detail has been started for the D17B.

Three tasks remain to complete the task: 1) simulate the remainder of the instructions, 2) include the flag store feature, and 3) include the rapid access loops.

Following completion of the D17B simulation, a logical next step might include an assembler to run on the D17B but designed on the simulation.

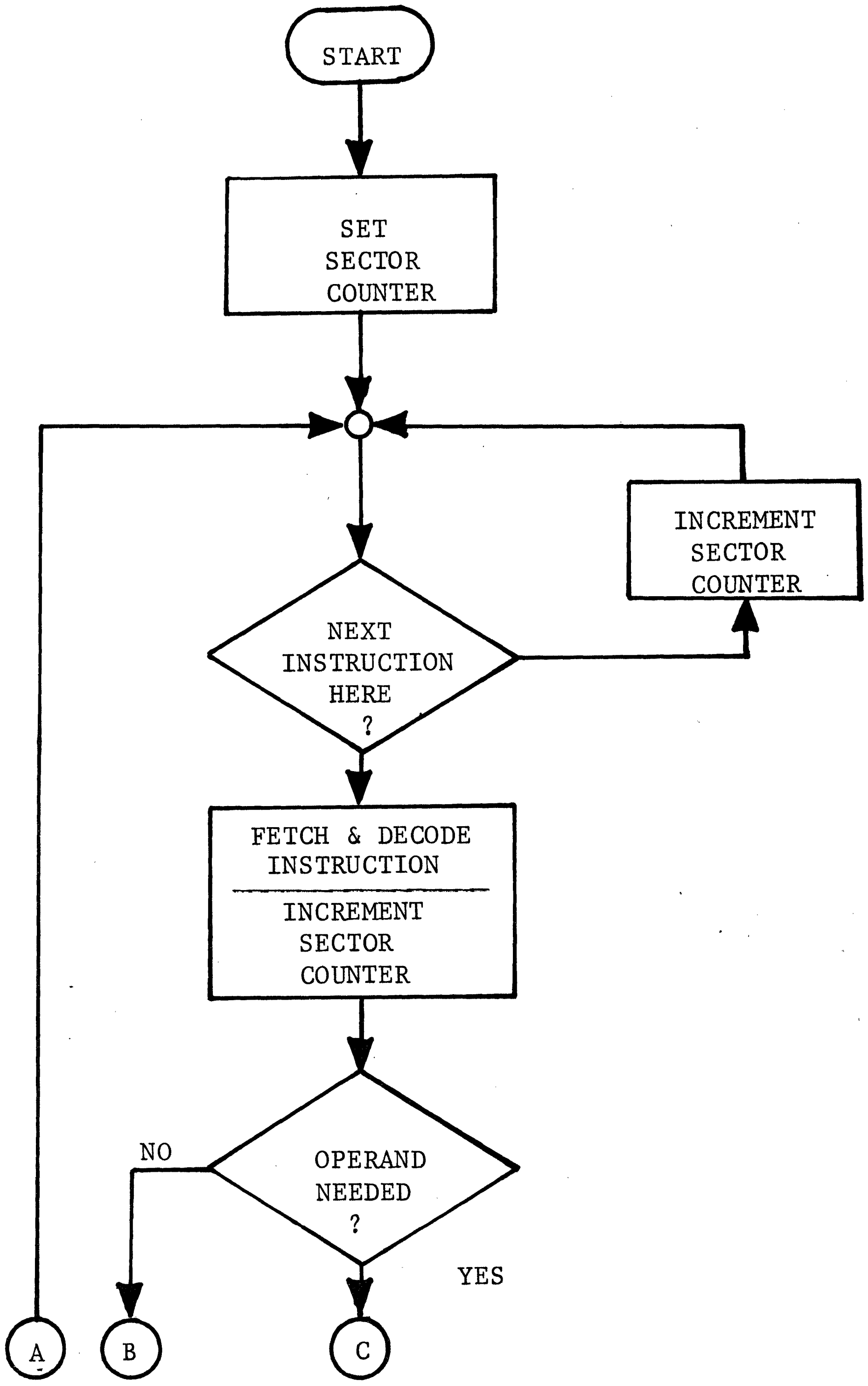


FIG. 1.a. FLOWCHART FOR D17B

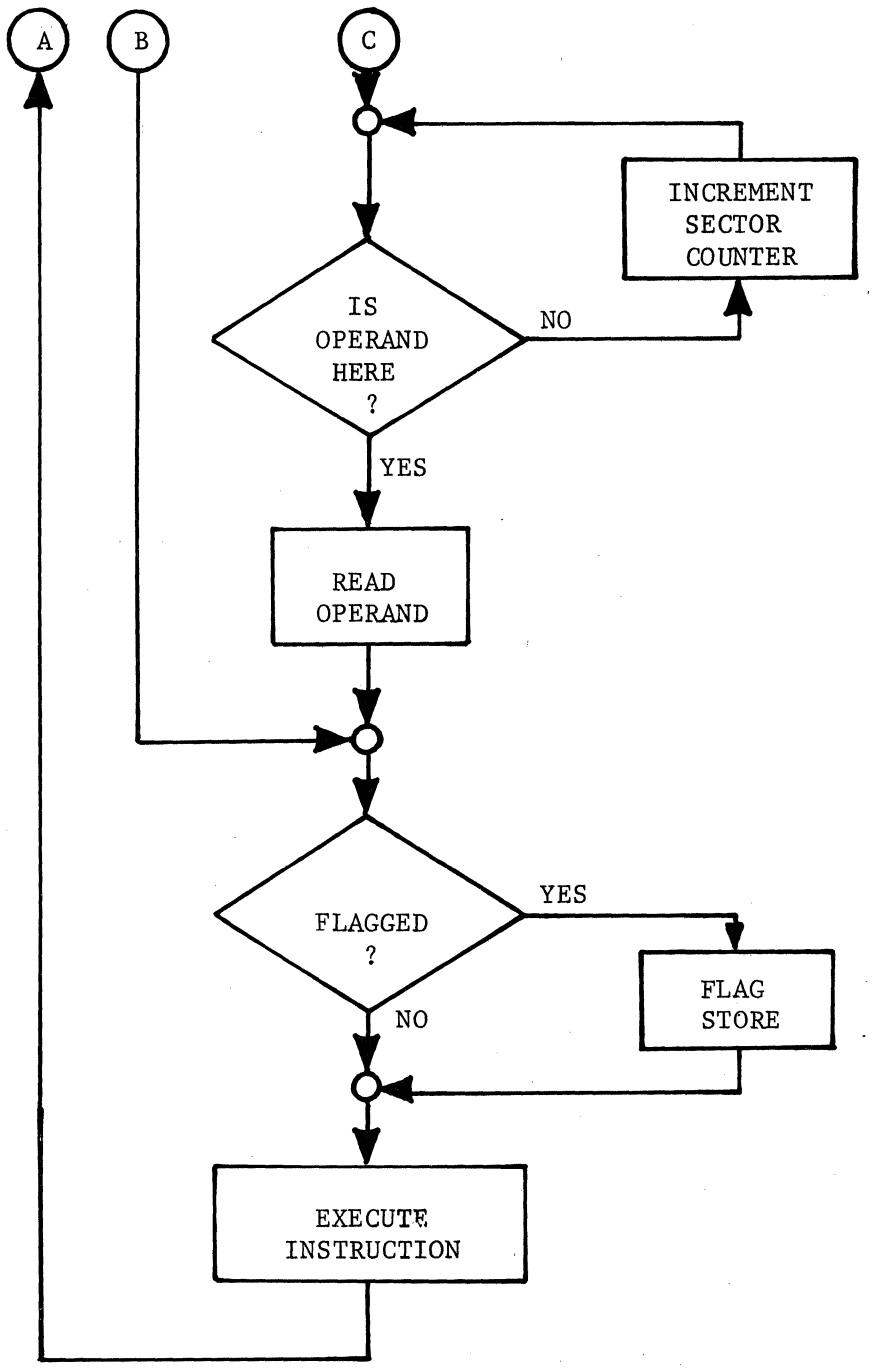


FIG. 1.b. FLOWCHART FOR D17B, CONT.

```
      ∇ MACHINE
[1]     OPZX←25ρOPERR
[2]     OPFX←32ρOPERR
[3]     OPC←ZERO,SCL,TMI,OPERR,SMP,MPY,SMM,MPM,FORTY,CLA,TRA,
        STO,SAD,ADD,SSU,SUB
[4]     SECT←1?127
[5]   IS:→(DI[2]=SECT)/DECODE
[6]     →IS,SECT←128|(SECT+1)
[7]   DECODE:DI←, 16 2 128 32 128 ⊤M[C;SECT]
[8]     ⎕←DTO AR
[9]     SECT←128|(SECT+1)
[10]    →(DI[0]∊ 0 2 8 10 11)/NOOP
[11]  OPER:SECT←128|SECT+1
[12]    →(DI[4]≠SECT)/OPER
[13]    NR←M[DI[3];DI[4]]
[14]  NOOP:→(DI[1]=0)/UNFLG
[15]    →UNFLG,ρ⎕←'A STORED'
[16]  UNFLG:→OPC[DI[0]]
[17]  ZERO:OPZX[8,9,10,11,12,13,14,15,24]←SAL,ALS,SLL,SRL,
        SAR,ARS,SLR,SRR,COA
[18]   →OPZX[DI[3]]
[19]  SAL:ARSP←, 2048 4 2048 ⊤AR
[20]    MID←ARSP[1]
[21]    ARSP←,2048|ARSP×2*DI[4]
[22]  SAL1:AR←(8192×ARSP[0])+(2048×MID)+ARSP[2]
[23]    →IS,ρSECT←128|SECT+1⌈DI[4]
[24]  ALS:AR←16777216|AR×2*DI[4]
[25]    →IS,ρSECT←128|SECT+1⌈DI[4]
[26]  SLL:ARSP←, 2048 4 2048 ⊤AR
[27]    MID←ARSP[1]
[28]    →SAL1,ρARSP[0]←2048|ARSP[0]×2*DI[4]
[29]  SRL:ARSP←, 2048 4 2048 ⊤AR
[30]    MID←ARSP[1]
[31]    →SAL1,ρARSP[2]←2048|ARSP[2]×2*DI[4]
[32]  SAR:ARSP←, 2 1024 4 2 1024 ⊤AR
[33]    SAVE←ARSP[0],ARSP[2],ARSP[3]
[34]    ARSP←⌊ARSP÷2*DI[4]
[35]    AR←ARSP[4]+(8192×ARSP[1])+(2048×SAVE[1])+(SAVE[0]×((
        2⊥(DI[4]+1)ρ1)×2*(23-DI[4])))+(SAVE[2]×((2⊥(DI[
        4]+1)ρ1)×2*(10-DI[4])))
```

Figure 2.a. Main Program

```
[36]  →IS,ρSECT←128|SECT+1⌈DI[4]
[37] ARS:→(AR>8388607)/ARS1
[38]  →ARS2,ρAR←⌊AR÷2*DI[4]
[39] ARS1:AR←16777216-⌈(16777216-AR)÷2*DI[4]
[40] ARS2:→IS,ρSECT←128|SECT+1⌈DI[4]
[41] SLR:ARSP←, 2 1024 8192 ⊤AR
[42]  SAVE←ARSP[0],ARSP[2]
[43]  ARSP←⌊ARSP÷2*DI[4]
[44]  AR←SAVE[1]+(8192×ARSP[1])+(SAVE[0]×((2⊥(DI[4]+1)ρ1)×
      2*(23-DI[4])))
[45]  →IS,ρSECT←128|SECT+1⌈DI[4]
[46] SRR:ARSP←, 8192 2 1024 ⊤AR
[47]  SAVE←ARSP[0],ARSP[1]
[48]  ARSP←⌊ARSP÷2*DI[4]
[49]  AR←ARSP[2]+(2048×SAVE[0])+(SAVE[1]×((2⊥(DI[4]+1)ρ1)×
      2*(10-DI[4])))
[50]  →IS,ρSECT←128|SECT+1⌈DI[4]
[51] COA:→IS,ρ⎕←'COA'
[52] SCL:→IS,ρ⎕←'SCL'
[53] TMI:→(AR>8388607)/TRA
[54]  →IS
[55] OPERR:→0,ρ⎕←'OPERR'
[56] SMP:LSP← 4 2048 ⊤LR
[57]  ARSP← 2048 4 2048 ⊤AR
[58]  MID←ARSP[1]
[59]  LR←ARSP[0]+(2048×LSP[0])+(8192×ARSP[2])
[60]  NRSP← 2048 4 2048 ⊤NR
[61]  SIGN←(ARSP>1023)≠(NRSP>1023)
[62]  PROD←(ARSP⌊2048-ARSP)×(NRSP⌊2048-NRSP)
[63]  LPROD←LPROD⌈SIGN[0]×2048-LPROD←(+/ 1024 2 512 ⊤PROD[0
      ])-512⊤PROD[0]
[64]  RPROD←RPROD⌈SIGN[2]×2048-RPROD←(+/ 1024 2 512 ⊤PROD[
      2])-512⊤PROD[2]
[65]  AR←(8192×LPROD)+(2048×MID)+RPROD
[66]  →IS,ρSECT←128|SECT+6
[67] MPY:SIGN←(AR>8388607)≠(NR>8388607)
[68]  LR←AR
[69]  PROD← 8388608 2 4194304 ⊤(AR⌊16777216-AR)×(NR⌊
      16777216-NR)
[70]  AR←PROD[0]+PROD[1]
```

Figure 2.b. Main Program, Cont.

```
[71]  AR←AR⌈SIGN×16777216-AR
[72]  →IS,⍴SECT←128|SECT+12
[73] SMM:→IS,⍴⎕←'SMM'
[74] MPM:→IS,⍴⎕←'MPM'
[75] FORTY:OPFX[1,4,5,8,9,11,12,13,14,17,18,19,20,
      21]←BOC,BOA,BOB,RSD,HPR,DOA,VOA,VOB,VOC,ANA,MIM,COM,
      DIB,DIA
[76]  OPFX[24,25,28,29,30,31]←HFC,EFC,LPR,LPR,LPR,LPR
[77]  →OPFX[DI[3]]
[78] BOC:→IS,⍴⎕←'BOC'
[79] BOB:→IS,⍴⎕←'BOB'
[80] BOA:→IS,⍴⎕←'BOA'
[81] RSD:→IS,⍴⎕←'RSD'
[82] HPR:'PROGRAMMED HALT'
[83] WAIT:START←⍞
[84]  →((2↑START)='GO')/IS
[85]  →WAIT
[86] DOA:→IS,⍴⎕←'DOA'
[87] VOA:→IS,⍴⎕←'VOA'
[88] VOB:→IS,⍴⎕←'VOB'
[89] VOC:→IS,⍴⎕←'VOC'
[90] ANA:→IS,⍴AR←2⊥((24⍴2)⊤AR)∧((24⍴2)⊤LR)
[91] MIM:→(AR>16777215)/IS
[92] COM:→IS,⍴AR←16777216-AR
[93] DIB:→IS,⍴⎕←'DIB'
[94] DIA:→IS,⍴⎕←'DIA'
[95] HFC:→IS,⍴⎕←'HFC'
[96] EFC:→IS,⍴⎕←'EFC'
[97] LPR:→IS,⍴⎕←'LPR'
[98] CLA:→IS,⍴AR←NR
[99] TRA:→IS,⍴(⍴C←DI[3]),(⍴DI[2]←DI[4])
[100]STO:SECT←128|SECT+1
[101] →(DI[4]≠SECT)/STO
[102] →IS,⍴M[DI[3];(128|SECT-2)]←AR
[103]SAD:NRS←, 2048 4 2048 ⊤NR
[104]SAD1:ARSP←, 2048 4 2048 ⊤AR
[105] MID←ARSP[1]
[106] ARSP←2048|ARSP+NRS
[107] →IS,⍴AR←(8192×ARSP[0])+(2048×MID)+ARSP[2]
[108]ADD:→IS,⍴AR←16777216|AR+NR
[109]SSU:→SAD1,⍴NRS←2048-(, 2048 4 2048 ⊤NR)
[110]SUB:→IS,⍴AR←16777216|AR+16777216-NR
```

Figure 2.c. Main Program, Cont.

```
        ∇FILL[⎕]∇
      ∇ FILL
[1]     'PROCEED'
[2]   READ:L←⍞
[3]     →0×⍳∧/(3⍴L)='END'
[4]     LR←OOTD(8↑¯9↑L)
[5]     →(((¯1↑L)='/'),((¯1↑L)='V'))/LOC,ENT
[6]     →READ
[7]   LOC:→READ,⍴IR←LR
[8]   ENT:ADR← 32 128 ⊤IR
[9]     M[ADR[0];ADR[1]]←AR←LR
[10]    IRS← 131072 128 ⊤IR
[11]    →READ,⍴IR←(IRS[0]×131072)+128|IRS[1]+1
      ∇
```

(a)

```
        ∇MRC[⎕]∇
      ∇ MRC
[1]     IR←10485760
      ∇
```

(b)

```
        ∇RUN[⎕]∇
      ∇ RUN
[1]   DI← 16 2 128 32 128 ⊤IR
[2]     C←DI[3]
[3]     MACHINE
      ∇
```

(c)

Figure 3. a) Fill Routine, b) Master Reset, c) Run Routine

A Hardware Divider for the D17B Guidance Computer

by

Alfred M. Williams
The Boeing Company
Houston, Texas

and

J. D. Bargainer
University of Houston
Houston, Texas 77004

The D17B Guidance Computer for the Minuteman I, ICMB is capable of performing addition, subtraction and multiplication through hardware algorithms. However, division must be performed through a software routine. A hardware division capability can be acquired by modifying the D17B operation codes and by incorporating additional hardware. This paper outlines such a modification to the D17B. The division algorithm is presented along with a description of the hardware operation. The divider is designed to perform full-word and split-word division and determine a fractional quotient and remainder. Both the quotient and remainder are accessible to the programmer through the computer registers once the operation is complete.

The Division Algorithm

A non-restoring division algorithm was desired that was capable of performing division with either positive or negative numbers in either the dividend or divisor. It was also desired that the algorithm be easy to implement on the D17B.

A divider algorithm which meets these requirements is the following fractional divide algorithm. To find $x/y = q_1 \cdot q_2 \ldots q_n$

1. Let $r_o = x$ (r_o is the first partial remainder)
2. Let
$$q_1 = \begin{cases} 1 \text{ if } r_o \text{ and } y \text{ have the same sign} \\ 0 \text{ otherwise} \end{cases}$$
3. Iteratively
$$r_i = 2r_{i-1} + (1-2q_i)y$$
$$q_i = \begin{cases} 1 \text{ if } r_{i-1} \text{ and } y \text{ have the same sign} \\ 0 \text{ otherwise} \end{cases}$$
The partial remainder r_i is therefore found by left shifting r_{i-1} and then adding y if q_i is 0 and subtracting y if q_i is 1.
4. Repeat the iteration n times or until the partial remainder is zero.
5. Add 1.000...1 to correct the quotient.
6. When $r_i = 0$ then $q_i = 1$ and $q_j = 0$ $j > i$.
7. When $r_i = 0$ then correct the quotient by adding 1.000...0.

The divider algorithm implemented on the D17B closely follows this outline; however, the last step in the process, correction of the pseudo quotient, is not performed by adding a correction factor. Instead, the sign digit is complemented and a "1" is forced into the least significant digit if the remainder is non-zero. The end result of this technique is the same as that acquired by adding the correction factor to the pseudo quotient.

The division operation designed for the D17B assumes that the dividend is stored in the accumulator. The division instruction is interpreted by the instruction processor and a divisor is loaded into the number (N) register from the specified memory address. At the end of the division operation, the quotient is stored in the accumulator and the remainder is stored in the lower accumulator. This arrangement was chosen because the quotient was desired after most division operations. In the case where the remainder is desired, it is possible to transfer it from the lower accumulator to the accumulator by loading all 1's into the accumulator and executing the ANA instruction.

It was necessary to add a delay flip flop to the accumulator to perform the left shift required by the division algorithm. A delay flip flop was also added to the lower accumulator so that the quotient digits could be stored in the proper order. Control logic was designed to do each of the following tasks: (1) force the least significant bit of the partial remainder to "0" prior to each add cycle, (2) determine whether addition or subtraction was performed during the next add cycle, (3) determine the quotient digit and store it in the lower accumulator, (4) monitor the accumulator for a zero remainder, (5) correct the pseudo quotient, (6) count the number of shifts performed to determine when the operation was completed.

The design took into account one more characteristic of the D17B. The D17B operates on two types of numbers, full-word and split-word numbers. A full-word number is composed of 24 bits, of which one bit is a sign bit. A split-word number is composed of two 11-digit numbers. As a result, two different division operation codes were required, one that would perform full-word division and one that would perform split-word division.

Divider Design

The D17B has 16 basic operation codes. All 16 codes are used. Each operation code is determined by the state of the four flip flops, 04, 03, 02, and 01. Four of the operation codes are used for multiplication, two are used for split-word and full-word "normal" multiplication and two are used for split- and full-word "modified" multiplication. The modified multiplication operation was redundant to the normal multiplication operation and was replaced by the division operation.

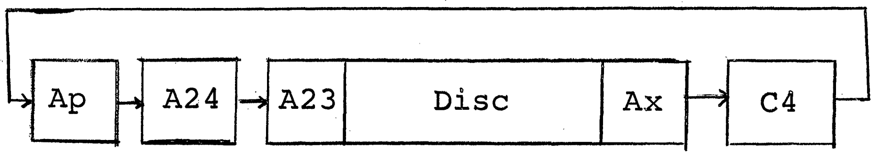
To delete the "modified" multiplication operations, it was necessary to modify the multiplication enable signals, OMO, OMF, and OM. Each of these signals enabled a period in the multiplication operation and was modified by ANDing the $\overline{02}$ signal with them. During multiplication the 02 flip flop is used as the addition/subtraction indicator to the carry/borrow flip flop, Ak. The Ak flip flop was enabled for addition if 02 was true and for subtraction if 02 was false. This function could no longer be performed by the 02 flip flop and a spare flip flop replaced it. After these modifications were completed, two operation codes were available for division. The operation code, $\overline{04}$ 03 02 01, (34), was used for full-word division and, $\overline{04}$ 03 02 $\overline{01}$ (30), was used for split-word division. The 01 flip flop was used to distinguish between full-word and split-word division.

The operation codes chosen for division maintained two of the "modified" multiplication operation characteristics. First, the number located at the specified memory address was automatically loaded into the N register during the first cycle of division. Second, the Q flip flop was one set at the end of the first word time of division. A spare flip flop in the computer, labeled the DIV flip flop, was used to designate the division operation. The first word time of division was indicated by the signal (DIV $\overline{Q}$). All remaining word times were indicated by the signal (DIV Q). The D flip flop was used to separate the middle word times from the last word time.

A word time counter is initialized during the first word time of division. The counter is composed of the CB5-CB1 flip flops. These flip flops are designed to count down at Tp time. They are initialized to (24) or $(11000)_2$ for full-word division and $(11)_{10}$ or $(01011)_2$ for split-word division. At Tp time of the first word time the accumulator recirculation control flip flop, Ac, is set. This disables normal recirculation of the accumulator flip flops and makes it possible for the accumulator to be extended by one bit. The one bit extension creates the one bit shift required by the divide algorithm. Also, at Tp time of the first word time, the lower accumulator recirculation control flip flop, Lc, is set. This disables normal recirculation of the lower accumulator flip flops and makes it possible for the lower accumulator to be extended by one bit. The lower accumulator is used to store the pseudo quotient and the one bit delay is required so that the quotient bits will be stored in the proper order. During the first word time of division, the divisor is loaded into the N register from memory. A spare flip flop designated the N2 flip flop copies the sign of the divisor loaded into the N register. The sign of the divisor is compared with the sign of the number in the accumulator. If both signs are the same the N2 flip flop is set. If the signs differ, the N2 flip flop is reset. After the N2 flip flop has completed this operation, it indicates whether addition or subtraction is performed during the next add cycle. It also indicates the proper pseudo quotient digit. This logic is operational only if a non-zero remainder exists. If a zero remainder does exist, then the N2 flip flop logic is modified so that the flip flop is set at the next compare time and reset for all remaining modify times. The C5 flip flop performs a delay so that the detected addition/subtraction operation information will be available during the add cycle. It copies the N2 flip flop at T1 time during full-word division and at Tp and T12 time during split-word division.

During the middle word times the dividend, located in the accumulator, is shifted to the left and the divisor, located in the N register, is subtracted from or added to it.

The shift of the dividend is performed by adding a delay flip flop to the accumulator. The C4 flip flop was used to provide the required one bit delay.



Extended Accumulator

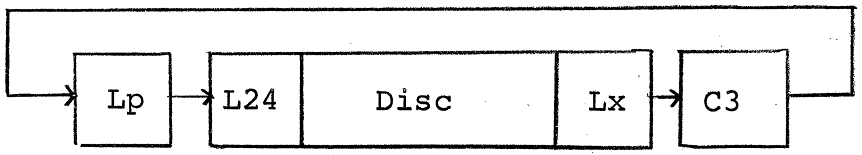
Logic for the C4 flip flop is designed so that it will copy the Ax flip flop.

The Ap flip flop is designed to copy the C4 flip flop. The A23 flip flop copies the A24 flip flop. The A24 flip flop is designed to function as the adder/subtractor flip flop during division and the Ak flip flop is used to determine the carry or borrow. Addition/Subtraction time is determined by the N1 flip flop (a spare flip flop). The add cycle is initiated at T2 time and continues through Tp time for full-word division. For split-word division the cycle occurs from T2 - T13 time and T15 - Tp time. If a zero remainder exists, the add cycle will not be initiated.

The SB3 flip flop is used to copy the adder flip flop and determine if a zero remainder exists. It is reset prior to each add cycle and set whenever the A24 flip flop is true.

The J and C2 flip flops are used to store the remainder status. Both flip flops are initially reset during the first word time. The J flip flop determines if a zero remainder exists in the least significant split word during split-word division. It copies the SB3 flip flop at T13 time, during split-word division and the C2 flip flop during full-word division.

The C2 flip flop determines if a zero remainder exists during full-word division or in the most significant split word during split-word division. It copies the SB3 flip flop at Tx time. The lower accumulator stores the pseudo quotient generated during the middle word times of division. It is extended by one bit through the addition of the C3 flip flop.



Extended Lower Accumulator

The one bit delay is required if the pseudo quotient digits are to be stored in the proper order. The loop recirculation control flip flop for the lower accumulator, the Lc flip flop, has previously been set during the first word time of division, so that the C3 flip flop can copy the Lx flip flop at the beginning of the second word time. The C3 flip flop is also designed to function as the quotient flip flop, that is, the C3 flip flop is responsible for decoding and injecting into the lower accumulator the pseudo quotient digit. This detection takes place at To time for full-word division. It is accomplished by requiring the C3 flip flop to copy the N2 flip flop.

During split-word division, the pseudo quotient is detected at T0 and T13 time. At these times, the quotient digit is stored in the C5 flip flop and it is necessary for the C3 flip flop to copy the C5 flip flop. When the C3 flip flop is not detecting a quotient digit, it copies the Lx flip flop, as previously stated.

The Lp flip flop is designed to copy the C3 flip flop during division. The L24 flip flop copies th Lp flip flop during division.

The operations just discussed are performed recursively during the middle word times of division. At the end of this period, the pseudo quotient digits are stored in the lower accumulator. The remainder is stored in the accumulator. During the last word time of division, the pseudo quotient is corrected and transferred to the accumulator and the remainder is transferred to the lower accumulator.

The remainder is transferred from the accumulator to the lower accumulator by having the Lp flip flop copy the Ax flip flop. Correction of the pseudo quotient is performed by the A24 flip flop and the C3 flip flop. The A24 flip flop is responsible for complementing the sign digit of the pseudo quotient. It complements the Ap flip flop at Tp time during full-word division and at Tp and T12 time during split-word division.

The least significant digit of the pseudo quotient is forced to "1" by the C3 flip flop if the remainder is non-zero. During the last word time of division, the C3 flip flop is set or reset at T0 time for full-word division and at T0 and T13 time for split-word division, depending on the remainder being non-zero or zero respectively. The transfer of the pseudo quotient from the lower accumulator to the accumulator is performed by the Ap flip flop. It copies the C3 flip flop during the last word time of division.

At the end of the last word time of division, it is necessary to reset the recirculation control flip flops for the accumulator, Ac, and lower accumulator, Lc and to reset the DIV flip flop.

The logic to implement the division algorithm was constructed on three cards with the same size and shape as the logic cards of the D17B. All flip-flops used in the modification were spare flip flops already in the computer and only gating logic was added. Approximately 350 diodes were used on the three cards.

Complete documentation of this modification including, wiring lists, circuit diagrams and negatives for etching the boards is available and we would be happy to send this documentation to anyone requesting it.

Bibliography

Yaohan Chu, Digital Computer Design Fundamentals, McGraw Hill Book Company Inc., 1962

Autonetics, Minuteman Computer Logical Description Engineering Manual 2065 14 January 1960

Air Force, General Purpose Digital Computer Technical Order 11G2-10-5-3-5 December 1960

Air Force, Digital Computer Electronic Modules Technical Order 11G2-10-5-3-10 15 October 1962

STATE DESCRIPTION OF D17B COMPUTER

Capt Douglas J. Allen   Gary B. Lamont
Electrical Engineering Department
Air Force Institute of Technology
Wright-Patterson Air Force Base, Ohio 45433

ABSTRACT

This report presents a state description of the D17B Computer. A set of control flipflops were chosen and from this choice the states of the computer were defined. The discussion of each state includes a set of register transfer equations that enumerate the information transfer during that state.

This approach was taken to present a compromise between a simple veitch diagram of the computer modes and a complete listing of the logic equations for the computer. Hopefully, this description will not only be a graphic study plan of the machine, but also an aid for maintenance and trouble shooting.

Introduction

A state description of the D17B Computer is a method of portraying the functional operations of the computer using the configuration of the control flipflops. A given configuration of the control flipflops is defined as a state of the computer and the paths between the states represent the functional operations. A set of register transfer equations that outline the information transfer between registers may be added to complete this description.

This approach is used for computer synthesis by Chu (Ref 2: 396-429) and is one basic method of modern computer design. As an analysis technique this method places the burden of defining which flipflops are to be considered as control flipflops on the analyzer. After this decision is made, the process is straightforward. In this presentation two primary considerations were used to choose the control flipflops. First, an effort was made to define the states so that the state description would parallel previous descriptions of the machine. Secondly, the control flipflops were chosen to make the description as simple and concise as possible.

A state description offers three advantages: (1) it presents a systematic way to study the machine, (2) the description presents a definite path to follow for maintenance checks, (3) this method presents more detail than the veitch diagram presentation.

State Description of the D17B

Operation of the D17B may be described by considering the various configurations that the control flipflops enter when the machine is executing a program. Thus, a state of the machine is defined by a particular configuration of the control flipflops. States may be represented on a diagram which depicts the various paths that the machine may cycle through during program execution. This state diagram may be used in conjunction with a description of the information exchange between registers to completely describe the machine operation.

State descriptions have the advantages of being a visual description, thus easily understood and capable of displaying large amounts of information in a concise form. Even more important, the state diagram provides a systematic approach for describing how the computer functions.

Register Transfer Notation. In order to conveniently describe how information is transferred between registers during each state, it is necessary to adopt a type of shorthand convention to condense the description. The symbols usually used in this notation are an adaptation of the system used by Chu, Ref 2: 378.

State Diagram. In this report the states of the computer have been broken into two major classes or modes, Compute (K) and Non-compute (K'). The states in these classes are represented by nodes (circles) and are numbered with an identifying number. Configurations of the major control flipflops which cause transition between states are listed beside the transition path on the diagram. Associated with each state diagram is a table which lists the state by number and name and the information transfer which occurs during that state. The Non-compute states are displayed in Figs. 1 and 2 and Compute states are shown in Figs. 3, 4, and 5. The associated register transfer notations for these state diagrams can be found in Ref 6. This reference can be obtained from the Defense Documentation Center.

Non-Compute States. Ref(3: 56) and (4: 1.1 - 2.15)

Power On Random State. When power is applied to the D17B, the controlling flipflops will become activated in a random state. Depressing the "MASTER RESET" switch causes the computer to enter a Prepare To Operate state where initialization is begun. See Fig 1.

Prepare To Operate (n1). In this state the phase register is initialized to an idle mode. F_c is turned off to prevent the computer from entering a special state called fine countdown. The Discrete output control register is initialized to prevent random discrete outputs and various other flipflops are initialized to start the synchronization of the bit counter with the sector track. Control flipflops O_2 and J are one set to allow transition to the Sync Bit Counter 1 state.

Sync Bit Counter 1 (n2). This state is the second state during which synchronization fo the Bit Counter and the Sector Track is accomplished. As shown on the state diagram, a transient master reset signal (less than one memory revolution in duration) will cause the machine to recycle through the Prepare To Compute State. The O_1 flipflop is "one" set allowing entry into the next state, Sync Bit Counter 2.

Sync Bit Counter 2 (n3). In this state the instruction register is loaded with an unconditional jump instruction to channel 0, sector 0. This instruction will be the first instruction executed unless a new instruction is loaded prior

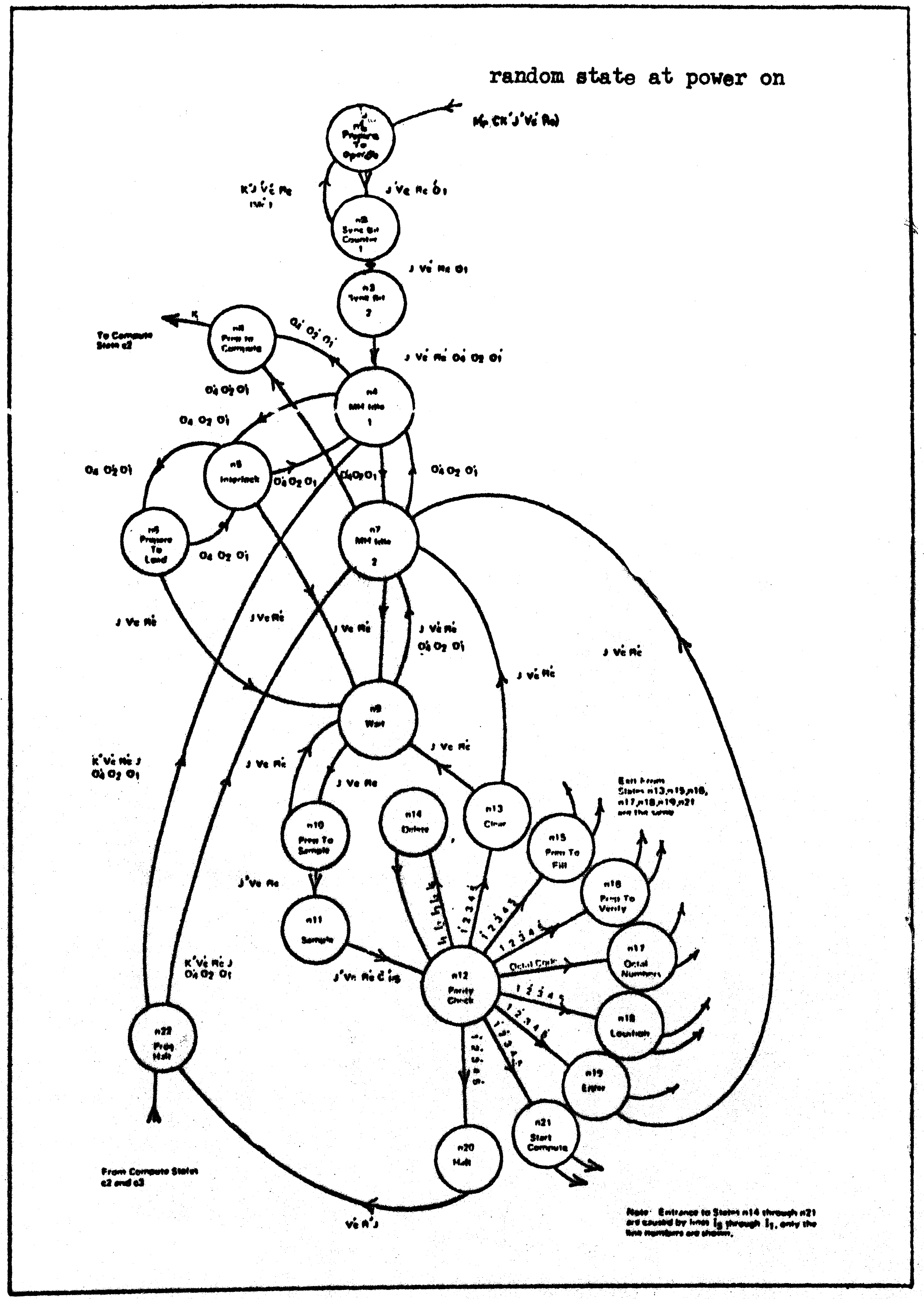


Fig. 4 D-17B Noncompute States

to the computer entering the compute mode.

After complete synchronization of the bit counter and the sector track, the R_c and O_1 flipflops are "zero" set allowing transition to the Manual Halt-Idle state.

Manual Halt-Idle 1 (n4). This state acts as a decision point for state transition. Three separate situations will cause the computer to enter the Manual Halt-Interlock state. If the previous state were n3 or n7, then state n4 was entered at a bit time corresponding to T_x of sector number 0; thus, the O_4 flipflop will be "one" set prior to the occurrence of any other state determining transition.

A third situation which could cause transition from n4 to n5 arises when the computer control switch is placed into "Halt" or "Single Step" during a compute operation. State n4 will be entered from Program Halt and transition will occur to state n5 or n7 depending on the O_1 flipflop. This state transition is not predictable since the O_1 flipflop state will be determined by the instruction that was being executed when the compute switch was placed in the Halt or Single Step. State n7 may be the next state entered if the previous state was n5. In this case n4 was entered at a bit time corresponding to T_1 of sector 177 thus allowing the O_1 flipflop to be "one" set.

State n8, Prepare To Compute, will be entered if the "Compute" switch is not in a "Halt" position and S_{b2} is zero set. S_{b2} is a flipflop that is one set as the result of a verify or parity error.

Manual Halt-Interlock (n5). If there is no Mechanical Reader Input Signal ($I^{*\prime}_m$) present or if a "Halt command is present from the "Compute" Switch or if a Sprocket timing interlock signal ($T^{*\prime}$) is present with no Fill Signal, the computer will cycle between states n6 and n5. Similarly, a cycle will exist through n7, n4, and n5 if a Mechanical Reader Input signal is present with no Fill signal (F^*_s). "Wait" state, n9, will be entered if a Fill signal is present. Thus, Manual Halt-Interlock, n5, acts as an interlock for the state transition process of the computer.

Manual Halt - Prepare to Load (n6). Prepare to Load state is entered if a device such as a photo reader is used for loading. From this state, transition will be back to n5 if a Sprocket Timing Interlock signal (T*') is present or to the Wait state, n9, if no T*' signal is present.

Manual Halt - Idle 2 (n7). The Manual Halt-Idle 2 state serves as a timing delay. From this state the computer will enter n4 if the compute switch is in the "Halt" position and/or a Parity Error has occurred. If no parity or verify errors have occurred, the next state will be n8, the Prepare to Compute State. In the event that a Fill signal (F^*_s) occurs, the next state will be n9.

Prepare to Compute (n8). In the Prepare to Compute state initialization of several flipflops is accomplished in preparation for entry in the Number Search State of Compute. J must be "one" set allowing the D flipflop to be "one" set. Then when agreement is reached between sector track and the Number Register, K is "one" set.

Wait (n9). Flipflops are initialized to receive the Input Load code in the Wait State. The computer will cycle between this state, n9, and n10, Prepare to Sample, until the Sprocket Timing Interlock signal, T*', has reached steady state. If a verify error occurs, the Idle 2 state will be reentered.

Prepare to Sample (n10). The primary purpose of the Prepare to Sample state, n10, is to allow the Sprocket Timing Interlock signal to reach steady state as described above. When this occurs, the computer will remain in the Prepare to Sample state until bit time T_{23} occurs and will then transition to the Sample state, n11.

Sample (n11). During the Sample state, the computer will load the information on Input Lines I_1* through I_5*. Note that flipflops C_{p1} through C_{p4} were "zero" set in state n9 and will be "one" set only by an I* input. At bit time T_{13} the computer will enter the Parity Check state.

Parity Check (n12). Flipflop S_{b3} will toggle on C_{p1} as C_{p1} through C_{p4} complete a circular shift. This circulation will occur on each bit time when the O_4 flipflop is "one" set. In order to insure circulation for only five bit times the O_4 flipflop is "one" set on bit time T_{20} and "zero" set on bit T_{24}. "One" setting the C_{p5} flipflop will allow a change to one of the Process Code states depending upon the contents of the Input Lines.

Process Code-Clear (n13). The clear load code causes the Lower Accumulator, L, to be filled with zeroes. "One" setting the L_c flipflop allows new information to be read into L starting with bit time T_0. Then the C_{p1} flipflop is flipflop is "zero" set preventing new information from being read into the L-loop. If a parity error is indicated by a S'_{b3} at bit time T_p the next mode will be n9; however, if no parity error occurs, the computer will bo to state n7, the Wait state.

Delete (n14). When the input lines are all "ones" no action is taken by the computer. This command can be used as a space in input tape. All "zeroes" is not used as a Delete command because the S_{b3} flipflop would indicate a parity error.

Prepare to Fill State (n15). The Prepare to Fill state is a preparation state for filling the memory. After the Fill command is processed, the succeeding Load codes will be loaded into memory until "Halt" or "Start Compute" commands are processed. In the event a parity error occurs, the next state will be n7; if no parity error occurs, n9 will be next.

Prepare to Verify (n16). The Prepare to Verify State is analogous to the Prepare to Fill State. Once the computer cycles through this state (caused by processing a load code $I_5' \ I_4 \ I_3' \ I_2 \ I_1$) the succeeding load codes will be compared with the contents of memory as specified by the Instruction Register. This actual operation will be executed as the result of an Enter command will therefore be described as part of the Enter state. Exit from this Prepare to Verify is similar to that of the Prepare to Fill state.

Octal Numbers (n17). In this state the octal numbers received from the input lines will be stored in the L register. Any number of octal codes may be loaded but only eight sets of octal digits may be stored in the Lower Accumulator at one time. Octal Numbers that are shifted out of L are lost. Exit from this state is similar to those of the other Process Code states.

Location (n18). In this state, n18, the contents of the L register is transferred to the instruction register. This information will contain the memory location, channel and sector number, that will be used to start Fill and Verify operations.

The I_c flipflop is "one" set at bit time T_0 allowing new information to be written in the I register, then it is "zero" set at bit time T_{24} after L is transferred to I.

Enter (n19). In this state, n17, the contents of Lower Accumulator will be loaded first into the accumulator, then into memory if a Prepare to Fill state had initiated a fill operation or the contents of the Accumulator and Memory will be compared if a verify operation had been initiated by the machine cycling through the Prepare to Verify state. The location of memory involved in the above operation is specified by the Instruction Register. If a parity error is detected, transition will be from n19 to n7, otherwise an error-free operation will allow the computer to go from the Enter state to the Wait state.

At this point it is necessary to define a set of four states that the computer cycles through during a Fill or Verify operation. (A Fill or Verify operation results after the computer has successfully cycled through the Prepare to Fill or Prepare to Verify states and will continue until the Halt or Start compute state is reached). These four states are called Fill-Verify Idle, fv1; Fill-Verify Number Search, fv2; Fill-Verify Wait 2 Word Times, fv3 and Fill-Verify Execute, fv4 and the computer cycles through them simultaneously as it passes through the Enter state. A state diagram of this four-state operation is depicted in Fig 2. These states will be discussed in conjunction with the Enter state since they occur simultaneously

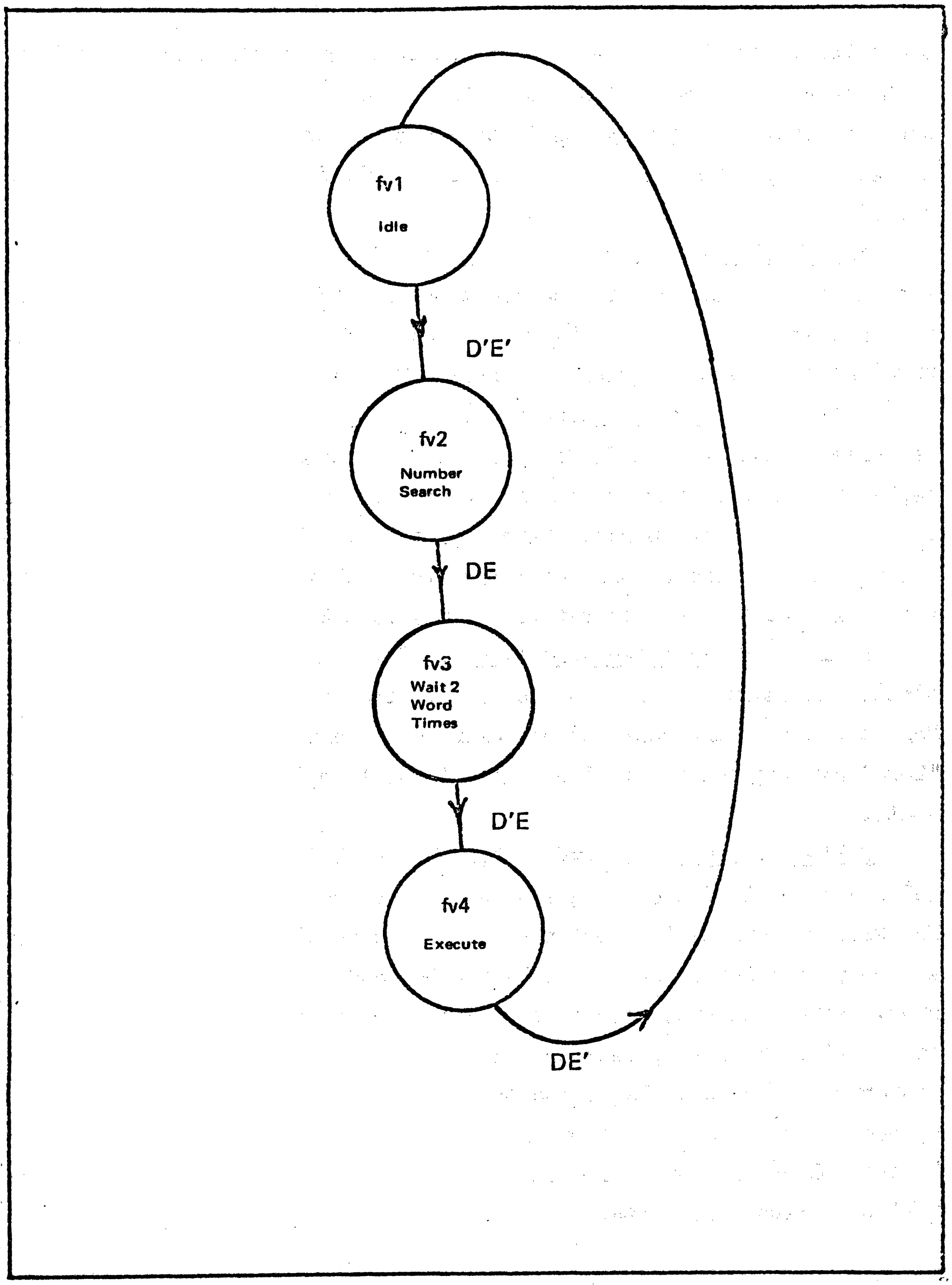


Fig. 2 Non compute Fill-Verify States

beginning in the Enter state. The action taken by the computer will vary with the part of memory that is to be filled or verified, thus it is necessary to consider not only the Enter state and the four-state cycle described above, but also the part of the memory involved in this operation must be considered.

Fill-Verify Idle (fv1). During the Fill-Verify Idle state the Lower Accumulator is copied into the accumulator. "Zero" setting the D flipflop causes transition to fv2, the Number Search State. This transition occurs simultaneously with a transition from n19 to n9 states.

Fill-Verify Number Search (fv2). During this state agreement is established between the Sector Track and the operand sector part of the I register. This comparison is made by the O_{b2} flipflop during bit times T_2 through T_7. The operand channel part of the I register is copied into the C_p register and channel agreement is established. The D and E flipflops are "one" set to cause transition to the Wait 2 Word Times state.

Fill-Verify Wait Two Word Times (fv3). During the Wait Two Word Times state, the Channel Buffer is copied into the Channel Register. The Number Register copies the contents of memory as specified by the Channel Register. "Zero" setting the D flipflop causes transition to the Fill-Verify Execute state.

Fill-Verify Execute (fv4). For both Fill Verify operations, the operand sector part of the I register will be augmented by one in this state. For Fill operations the contents of the Accumulator will be transferred to a memory location as specified by the Operand Address part of the I register. After the Fill operation, transition is made to the Fill-Verify Idle state. Verify operations are different in two ways. First, the contents of the Accumulator and the Number Register are compared. If agreement occurs S_{b2} flipflop will remain "zero" set and the next state will be fv1. Disagreement is indicated by S_{b2} "one" setting and the next state will be a Manual Halt state.

Halt (n20). When the "Halt" code is processed, the Halt state will be entered and the V_c flipflop will be "zero" set causing a transition to the Program Halt state.

Start Compute (n21). The Start Compute command when entered on the Input Lines will cause the computer to enter the Manual Halt Idel 1 state before transitioning to the Prepare to Compute and Compute states. If a parity error occurred while processing the code, the computer will not transition from the Manual Halt states.

Program Halt (n22). Four separate conditions may cause the computer to enter n22, the Program Halt state. If a "Halt" load code is successfully processed in, the computer will enter n7 before returning to Manual Halt Idle states.

Secondly, a halt instruction may be executed in the Compute mode or if the Compute Switch is not in the "Run" position when a new instruction is found the computer will return to Program Halt state from the "Last Word Time State" of Compute. Also, if during the Number Search state of compute the "Compute Switch" is not in "Run" and an instruction search is required to locate a new instruction, the computer will enter n22. In all cases the computer prepares to enter one of the Manual Halt Idle states during the Program Halt state. The actual Idle state entered depends upon the state of the O_1 flipflop which was set by the instruction being executed when state n22 was entered.

If state n22 were entered as the result of processing a Halt command during a Fill or Verify operation, the D and E flipflops will be set to cause the computer to simultaneously enter the Idle state of the Fill-Verify operation.

Compute States. Ref (3: 25) and (4: 5.1 - 6.13)

The Compute mode of the D17B is controlled by seven major control flip-flops. The K flipflop, when "one" set, indicates that the computer is in one of the "Compute" states. The various states of Compute are then

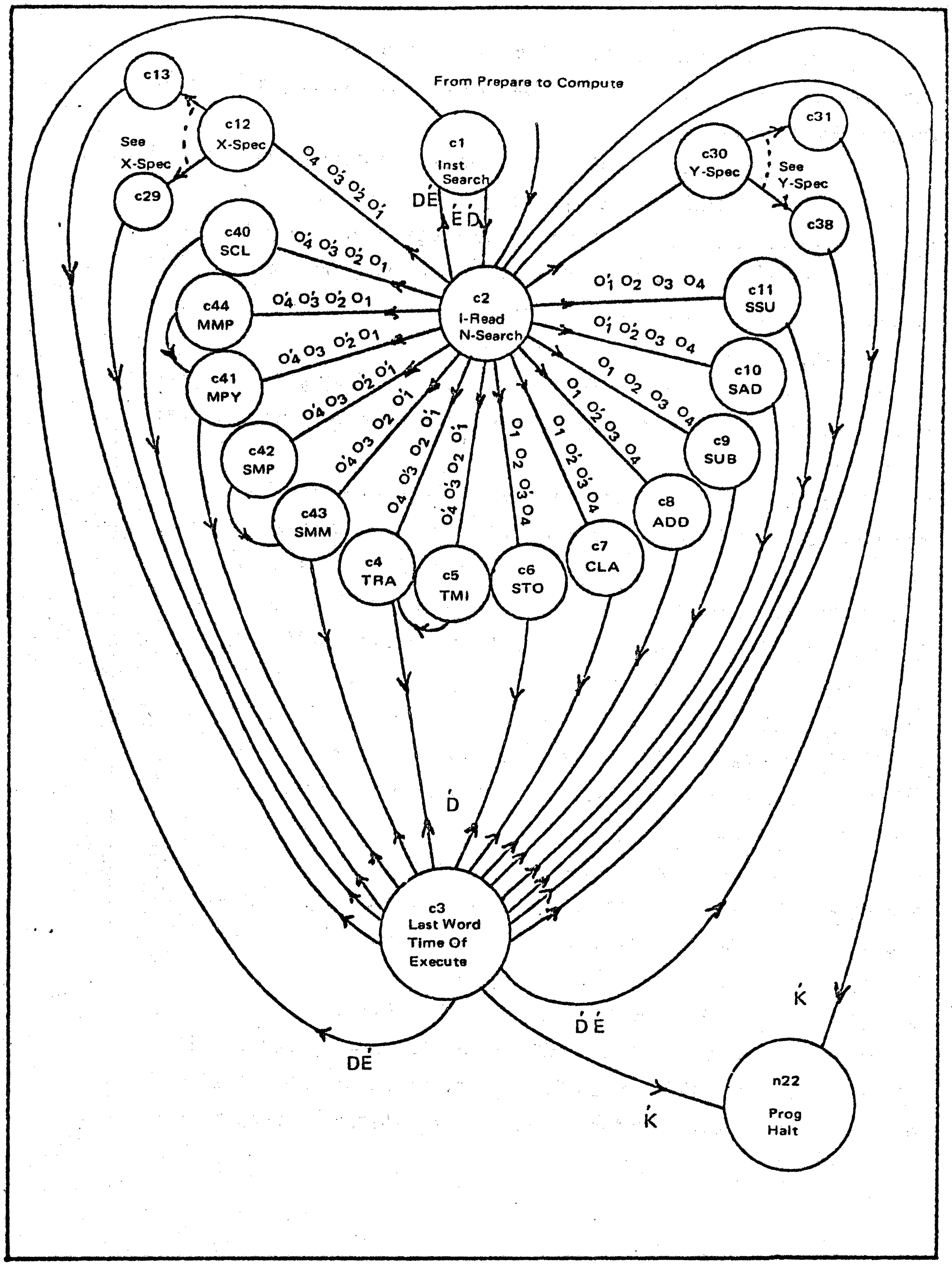


Fig.3·D-17B Compute States

controlled by the D and E flipflops. When the E flipflop is "one" set an instruction is being executed. The D flipflop, when "one" set, indicates that an instruction search is in progress and when "zero" set indicates instruction read and/or operand search is in progress. The four flipflops of the Operand Storage Register, O_4 through O_1, determine the instruction that will be executed.

<u>Instruction Search State</u> (c1). The Instruction Search State as defined in this report will be the state indicated by the flipflop settings K D E'. It is not necessary for this state to occur with the execution of every instruction.

If a program is optimally coded, a new instruction can be read into the I register during the execution of the present instruction. In this case, the instruction search operation was performed as a result of forethought of the programmer. Similarly, the Instruction Read-Number Search state may also be avoided by astute programming. In this case the computer would cycle between the two states of Execute without actually performing an instruction or operand search.

Instruction agreement occurs when the memory location addressed by "next instruction" part of I is in a position to be read by the computer. Monitoring for this condition is performed by the buffer flipflops O_{b1} and O_{b2}. These two flipflops are monitored by the I_d flipflop which controls the D flipflop. When the D flipflop becomes "zero" set, transition to state c2 occurs.

<u>Instruction Read-Number Search State</u> (c2). Instruction Read-Number Search state, c2, is a dual function state defined by DE' flipflop conditions. Like the Instruction Search State, this state may not necessarily be realized with the execution of every instruction. One-half of the dual function of the state may be exercised. For example, the next instruction may be found and read during the Execution state and the computer may cycle to state c3 for the Number Search function alone.

For number agreement the information in I_p at bit times T_2 through T_g must agree with the Sector track, S. Since the loops are effectively separate channels of 4, 8, and 16 word length, more than one flipflop is needed to check agreement for all channel elngths. Flipflop O_{b2} monitors for agreement for the 4 word loops, O_{b1} monitors for 8 word loops, S_{b1} for 16 word loops and O_{b3} monitors for the full channel length, 128 words. The N_d flipflop is the primary number agreement monitor and is changed by the above number agreement flipflops at bit time T_{13}.

Instruction Read is accomplished by setting the desired memory channel into the C_{p5} through C_{p1} flipflops. When flipflop I_d indicates Instruction agreement, the I_c flipflop is "one" set allowing the new instruction to be read into the I register. Bits I_{24} through I_{21} are read into the Operand Buffer Register, and I_{12} through I_8 are read into the channel buffer register. If the instruction is a flag-store instruction (I_{20}=1) the flag channel information, I_{19}, I_{18}, and I_{17} is read into the Flag Code Buffer Register. If the instruction is not a flage-store instruction, the Flag Code Buffer Register is loaded with "zeroes".

From this state, c2, transition will be to one of the instruction execution states or to c1 in the case of the transfer on minus instruction with a positive accumulator (see state c4 description). If the Compute Switch is not in the "Run" position when the I_c flipflop is "one" set to read a new instruction, the computer will go to Non Compute Program Halt, n22.

Last Word Time of Execute (c3). The Last Word Time of Execution, c3, will be discussed in conjunction with the execution of each of the instruction states since during this state the operation started in each of the instruction states is completed. For all one-word-time instructions (O_4=1), the instruction defining state is entered for the first bit time of execution and then the computer transitions to c3 to complete the operation.

This state acts as a decision point for the computer to exit the Compute

Mode. If the Compute Switch is not in the "Run" position and a new instruction is found, the computer will go to state n22, Non Compute Program Halt.

Unconditional Transfer (c4). The word format of the D17B makes no provision for specifying the channel of the next instruction. Thus, there must be a command to change channels of operation. The Unconditional Transfer is a "jump" instruction that is used for this purpose. In this "jump" instruction the sector of next instruction field is ignored and the complete operand address serves as the address of the next instruction. The new channel address is contained in the Operand channel portion of the transfer instruction. This information was shifted to the program channel buffer register during the instruction search operation. At bit time T_0 the program Channel Buffer Register is parallel loaded into the Program Channel Register.

Instruction agreement is controlled by the number agreement flipflop which determines the sector of the new instruction from bits I_7 through I_1 of the present instruction.

Conditional Transfer (c5). The decision for the Conditional Transfer operation is made in state c2. If bit A_{24} is zero, the accumulator is positive and the computer returns to state c1 to search for the instruction as indicated by 5p [I]. A "1" in bit position A_{24} indicates that the accumulator contains a negative number and the computer goes to state c3 and selects the new instruction as indicated by O[I].

Store Accumulator (c6). The Store state must be considered for four different situations; storing in channel 50, storing in channels 00 thru 46, storing in the loops, and flag storing.

Storing in channel 50 or "Hot storage writing" is initiated by setting the S_i flipflop to the channel 50 store code, then the Accumulator is copied directly into channel 50 and in a sector two octal-numbers less than the sector of S[I]. This two sector difference is accounted for by the fact that the write heads are separated from the read heads by two sectors.

In order to store information in channels 00 thru 46 an EWC signal must be present, enable write switch must be on. For selecting channels 00 thru 46 the computer utilizes a separate selector switch for each channel. This selection is accomplished using the contents of Channel Storage Register. The Accumulator is then stored in the memory address specified by the OP[I] minus two sector positions.

Storing in the E, F, H loops is similar to storing in channel 50 except the S_i flipflops are set by the contents of the channel buffer register.

Storing in the V and R loops may be accomplished if the computer is not in Fine Countdown mode (F_c=1) (See state c17). In this case, the contents of A is added to the incremental input at the time of execution.

A special case results when the T_{20} bit of any instruction is "1". This "flag", "1" in T_{20} is used to execute two operations with one instruction. The contents of the Accumulator will be stored in the channel indicated by the contents of bits I_{19} thru I_{17}. This means that the sector of next instruction field of the instruction being executed is limited to the four bits I_{16} thru I_{13} and the next instruction must be within the next 16 sectors. Flag storing is accomplished in the following steps: The Flag store buffer register S_b is loaded with the contents of I_{19} thru I_{17} during state c2. During the execution of the instruction the Flag Store Buffer register is parallel-loaded into the flag store register. This information is used to select the proper write heads for writing the Accumulator contents into memory.

<u>Clear and Add</u> (c7). State c7 initiates the clear and add operation, in which the contents of memory as specified by the operand address is transferred to the Accumulator. In state c7 the N_c flioflop is "one" set allowing the selected contents of memory to be read into the Number register. In state c3 the operation is completed, the selected contents of memory is read into the Accumulator.

Add (c8). State c8 initiates the add operation in which the memory contents as specified by operand address is added to the Accumulator. The sum is then stored in the Accumulator.

Subtract (c9). Subtraction is accomplished by the hardware as addition in the D17B; however, the carry operation of addition is converted to a borrow operation by a "one" in the O_2 flipflop.

Split Add (c10). During the split add operation the split word contents of the Accumulator is added to the corresponding parts of memory and the sum is stored in the split word portions of the Accumulator. At bit times T_{12} and T_{13} the A_c flipflop is "zero" set allowing the contents of A_{12} and A_{13} to remain unchanged.

Split Subtract (c11). The split subtract operation is similar to the split add operation, except that the split word contents of memory location specified by O[I] is subtracted from the contents of the Accumulator.

X Special State (c12). No action is performed in the X special state. It serves only as a decision point for the computer to enter a special set of states that require one word time to complete and do not require access to the computer memory. The Channel Storage Register contents are used to select the X special state that will be entered from c12. In this special operation the channel storage register serves as an auxiliary operation-code storage register. Since all the X special operations are one word time instructions, the specific X special state serves to define the operation and much of the actual operation is performed in state c3.

Complement (c13). The complement operation causes the 2's complement of the Accumulator to be read into the Accumulator. The Accumulator is circulated and the A_c flipflop is "one" set by the first "one" in the Accumulator. All succeeding bits of the Accumulator are complemented.

Minus Magnitude (c14). When the computer enters the Minus Magnitude state, c14, the sign of the Accumulator is tested. If the Accumulator is

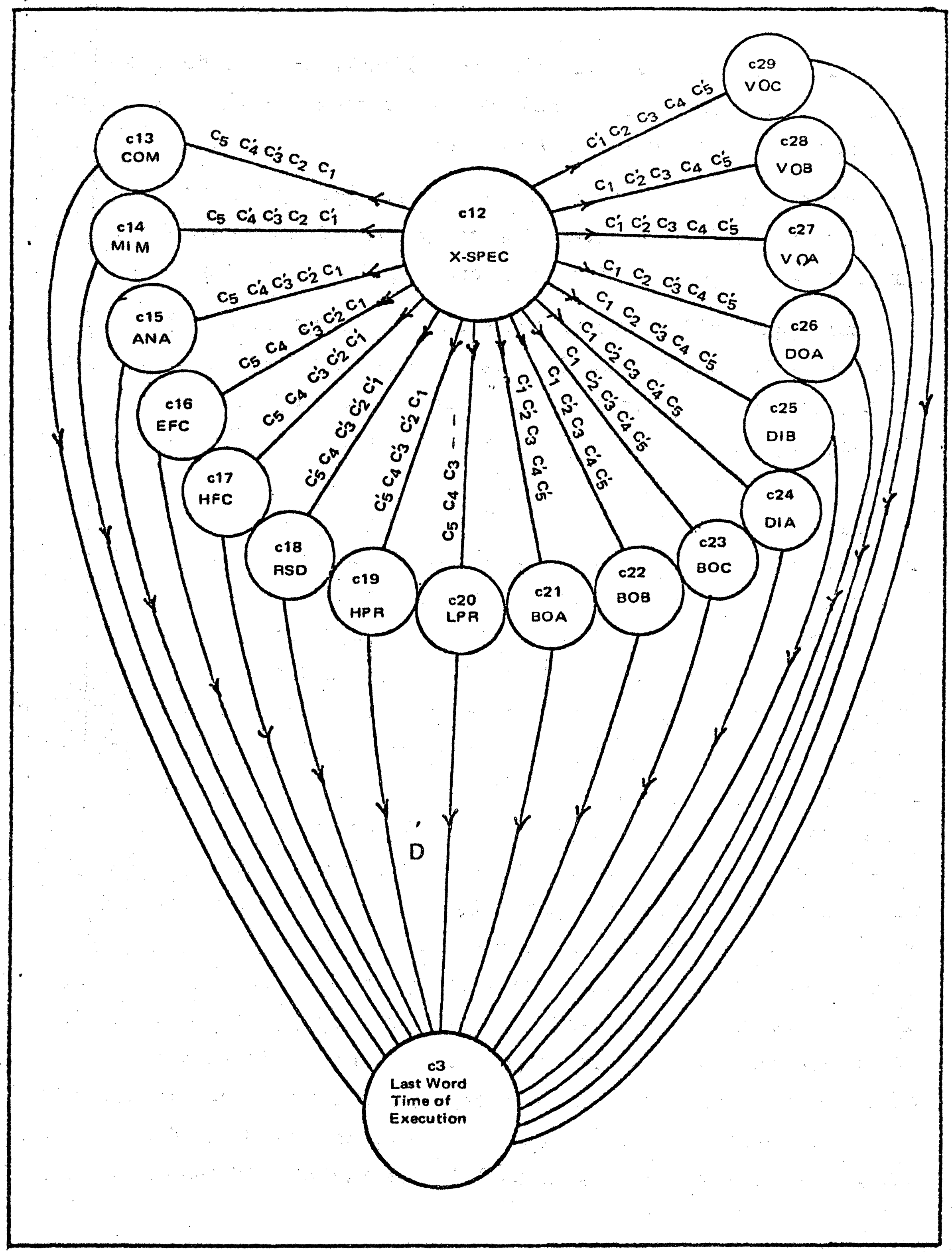


Fig. 4 X-Special Compute States

negative no action is taken; if the Accumulator is positive the C_{bl} flipflop is "one" set and copied into the C_1 flipflop, thus generating a complement instruction.

<u>Logical And to Accumulator</u> (c15). Entering state c15 causes the correspondig bits of the Accumulator and Lower Accumulator to be logically "anded".

<u>Enter Fine Countdown</u> (c16). Entering the Fine Countdown causes the F_c flipflop to be "one" set. This places the computer into a parallel operation called Fine Countdown. During Fine Countdown the V and U loops form a digital integrator. This operation will continue until the Halt Fine Countdown state is entered.

<u>Halt Fine Countdown</u> (c17). Entering the Halt Fine Countdown state, c17, causes the Fine Countdown flipflop, F_c, to be "zero" set.

<u>Reset Detector</u> (c18). When the Reset Detector state is entered, the D_r flipflop is "zero" set. The D_r flipflop is "one" set by I_b^*.

<u>Halt and Proceed</u> (c19). Entering state c19, Hand and Proceed causes the computer to enter state c3 and then state n22, Program Halt.

<u>Load Phase Register</u> (c20). The Load Phase Register special instruction causes C_2 to be loaded intp P_2 and C_1 is copied into P_2. P_3 copies the I_x flipflop at bit times T_1 through T_5. State c20 is defined by three of the C flipflops, C_5, C_4, and C_3; the remaining two C flipflops may be either "one" or "zero" set. The actual purpose in setting the Phase Register will be discussed in conjunction with state c27.

<u>Binary Output</u> (c21, c22, c23). Binary Incremental Output states may be discussed simultaneously. These states differ only in the sense that state c21 involves output flipflop G_1, c22 involves G_2, and c23 involves G_3. Only the first state, c21, will be discussed because the discussion is directly applicable to all three states by substituting the proper G_i flipflop in state c2i, where i=1, 2, or 3.

In state c21 the state of the G_1 flipflop is checked. If G_1 equals

"1" the first eight bits of A are treated as a word and +1 is added to that word. If G_1 equals "0" a 1 is subtracted from the word formed by the first eight bits of A. After one of the above operations is accomplished, the G_1 flipflop copies the sign bit of A.

Discrete Inputs (c24, c25). In both discrete input operations a set of twenty-four discrete input lines and flipflops are sampled and read into the A register. For a Discrete Input A, DIA, operation the discrete input lines X_1 through X_{19} and flipflops D_r, F_c, P_3, P_1, P_2, replace bits A_1 through A_{24} respectively.

During the operation initiated by state c25, DIB, the discrete lines Y_1 through Y_{24} replace bits A_1 through A_{24} respectively. The actual information transfer described in these states takes place in state c3; however, the states c24 and c25 serve to define the operation to be performed in state c3.

Discrete Outputs (c26). The operation initiated by state c2, Discrete Output A, causes the bits I_1 through I_5 to be loaded into the Discrete Output Register, D_1 through D_5.

Voltage Output (c27, c28, c29). The Voltage Output states are identical in concept. The function of these states varies only in the physical location of the output voltage.

Three Voltage Output Registers are loaded with the split word contents of A. If I_4 is "1", the right half of A is loaded and if I_4 is "0", the left half of A is loaded.

The states c27, c28, and c29 determine which set of Voltage-Output flipflops, V_{i1} through V_{i8}, (i=1, 2, or 3) will be loaded from A. If c27, VOA, is entered V_{11} through V_{18} will be loaded; c28, VOB, causes V_{21} through V_{28} to be loaded; and c29, VOC, causes v_{31} through V_{38} to be loaded with the proper half-word of A.

The Phase Register also affects the output location of each voltage line.

Y Special State (c30). The Y Special state, c30, serves only as a decision point for entering specific states c31 through c38. Operations initiated by the Y Special state do not require access to Memory; however, they do require more than one word time to complete.

Accumulator Left Shift (c31). A left shift operation is accomplished in the D17B by adding an extra flipflop, A_k, to the A loop for the number of word times equal to the number of shifts required. The number of shifts is specified by I_1 through I_5. This number is loaded into the Channel Buffer Register and counted down at each word time.

Accumulator Right Shift (c32). State c32 initiates a right shift of the Accumulator. To accomplish this operation, the A_p flipflop is removed from the recirculation loop of the Accumulator. The number of right sifts required is indicated by I_1 through I_5 and the A_p flipflop remains out of the A loop for that number of word times. If the Accumulator is positive, zeroes are filled into the vacated bits; however, if the Accumulator contains a negative number, 1's replace the bit positions vacated by the right shift.

Split Accumulator Left and Split Accumulator Right Shift (c33, c34). The discussion of states c31 and c32 are directly applicable to the states c33 and c34 respectively. In the split-shift states the left and right half words of the Accumulator are shifted the same number of bit positions but are treated as separate words.

Split Left Word Left Shift (c35). State c35 initiates an operation which causes the left half-word of the Accumulator to be shifted left by the number of bit positions specified in I_1 through I_5. The discussion of state c31 is applicable to this state except that bits A_{14} through A_{24} only are affected.

Split Right Word Left Shift (c36). Bits A_1 through A_{10} only are affected by the Split Right Word Left Shift operation. As implied by the state name, the rihgt half-word of the A register is shifted left.

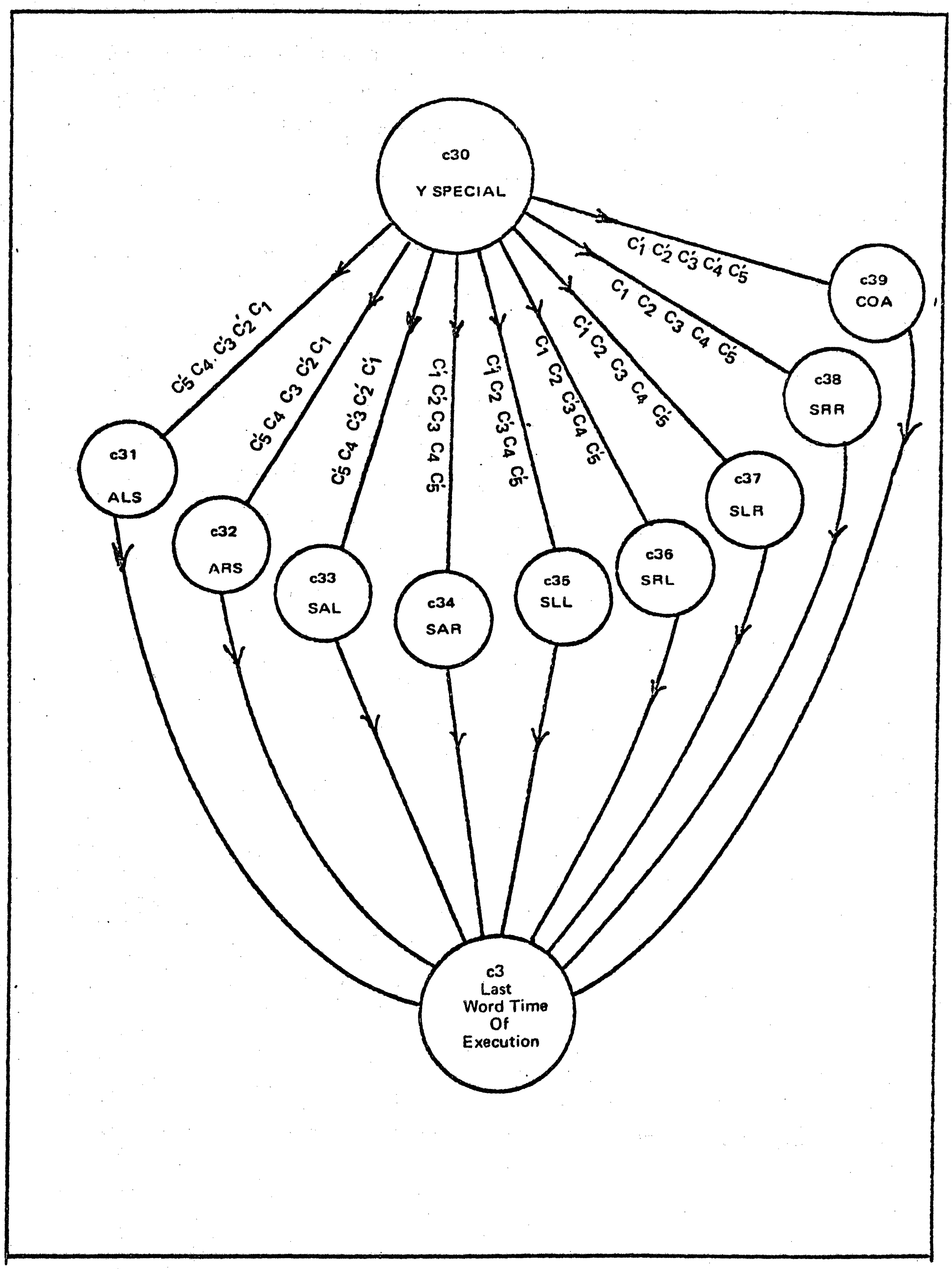


Fig. 5 Y Special Compute States

Split Left Word Right Shift (c37). State c37 initiates a right shift of the left half-word of the Accumulator. As in all right shift operations, if the half-word were positive, the bits vacated by the shifting are filled with zeroes and if the half-word were negative, 1's are filled into the vacated bit positions.

Split Right Word Right Shift (c38). State c38 initiates a right shift of the right half-word of the Accumulator. The discussion of c37 is directly applicable to this state except the right half-word is shifted.

Single Character Output (c39). The operation initiated by state c39 shifts the four most significant bits out of the Accumulator and presents them to the four character output lines. A fifth character output line is used as a parity line. This information is presented on the character output lines for the number of word times specified in s[I].

The Single Character Output operation is accomplished in the following manner. The sector portion of the instruction operand is shifted into the Operand Channel Buffer Register. Each word time this register is decreased by one, thus it is used to terminate the operation after the end of (s[I])+1 word times.

During the first word time of the Single Character Output operation, the circulation loop of the Accumulator is extended to include four flipflops of the Operand Channel Buffer Register: C_1, C_2, C_3, and C_4. This causes the four most significant bits of the Accumulator to be left shifted into these C flipflops. Parity is indicated by the J flipflop by "zero" setting it at the beginning of the operation and allowing it to toggle as each "1" is shifted into the flipflop.

The parity (J) and output (C_4, C_3, C_2, and C_1) is presented on the output lines S_{c5} through S_{c1}, respectively, with the occurrence of each S_{cT} timing pulse.

Split Compare and Limit (c40). State c40 initiates the Split Compare and Limit Operation in which the split-word contents of the Accumulator is compared with the corresponding bits of a word in memory. The memory is specified in the operand of the SCL instruction.

If the contents of the memory word is greater than that of A, no changes are made. If the split word portion of A is positive and greater than the corresponding part of the memory word, the split memory word replaces the split-word of A.

If the quantity in memory is less than the corresponding part of A and that half-word of A is negative, the two's complement of the memory half-word replaces the Accumulator half word.

Multiply (c41). The Multiply operation is initiated by state c41. The operation causes the contents of the Accumulator to be moved to the Lower Accumulator and the product of the Accumulator and memory contents specified by the MPY operand is placed in the Accumulator.

Split Multiply (c42). State c42 initiates the Split Multiply operation. This operation is similar to the Multiply operation except the left half-word of A goes into the right half-word of L and the right half-word of A goes into the left half-word of L. The split words of the Accumulator and the memory word specified by O I are multiplied and stored in the respective split words of the Accumulator.

Split Multiply Modified (c43). Split Multiply Modified is an operation which causes the three least significant bits of the Channel Buffer Register to be replaced by the "exclusive or" of those bits and the contents of the Phase register. The operation then proceeds as a Split Multiply operation. Split Multiply Modified commands allow the computer programmer to vary the effective operand channel address depending upon the Phase register contents.

Multiply Modified (c44). State c44 initiates the Multiply Modified operation which causes the three least significant bits of the Channel Buffer Register to be changed by an "exclusive or" operation with the Phase Register. After the above modification, a multiply operation is accomplished as described in state c41. It is noteworthy that this operation does not change the original multiply instruction in memory.

State Description Summary

In the above description of the D17B the various configurations of control flipflops were used to define states of the computer. These state definitions are not unique and many other sets of flipflop combinations may be used to describe the machine operation. The states described were chosen because they could be given names that correlate with other published information about the D17B. Hopefully, this type of description will be an aid not only in understanding the operations of the machine, but also in maintaining it. For example, the "state" of an inoperable machine may be determined by checking the status of the control flipflops. Once the state is identified, the malfunctioning circuit may become apparent by considering which flipflop is preventing normal state transition.

Bibliography

1. Autonetics. Part 1 Preliminary Maintenance Manual of the Minuteman D17B Computer and Associated Test Equipment, P.O. Memo 71. Anaheim, California: Autonetics, Division of North American Rockwell, Inc., January 1960.

2. Chu, Yaohan. Digital Computer Design Fundamentals. New York: McGraw-Hill Inc., 1962.

3. Hansen, D. D. and Watkins, K. R. A Rigorous Logical Study - with Lab - of the D17B Digital Computer. ACC-31170P-33. Anaheim, California: Computer and Data Systems Dept of Autonetics Division of North American Rockwell, Inc., 30 April 1962.

4. Shoryer, L. O. D17B Computer Manual. Anaheim, California: Autonetics Division of North American Rockwell, Inc., 1 July 1960.

5. USAF Technical Order. Technical Manual Overhaul and Repair General Purpose Computer (Model D17B), T.O. 11G2-10-5-3-5. Los Angeles, California: Air Force Kier Lithographic, 24 November 1964.

6. Allen, Douglas J. Laboratory Conversion and State Description of the D17B Computer, Thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1972.

Appendix A

List of Terms and Abbreviations

A_k: Carry, borrow and misc. flipflop.

A_p: "A" register extra delay flipflop.

A_x: "A" register read flipflop.

A_{24}: "A" register delay flipflop.

A_{23w}: "A" register write flipflop.

B_6, B_5, B_4, B_3, B_2, B_1: Bit time counter flipflops.

C_{b5}, C_{b4}, C_{b3}, C_{b2}, C_{b1}: Operand channel buffer register and word time counter flipflops.

C_{p5}, C_{p4}, C_{p3}, C_{p2}, C_{p1}: Program channel register.

C_5, C_4, C_3, C_2, C_1: Operand channel storage register and auxiliary operation-code storage register.

D17B: Designation of the computer used for guidance in the Minuteman I missile.

D_c: Shift control for "Discrete Output" register.

D_{dc}: Discrete disable signal from a control panel to control the discrete outputs.

D_r: Gyro malfunction indicator flipflop.

D_5, D_4, D_3, D_2, D_1: "Discrete Output" register.

D: Control flipflop.

E: Control flipflop.

E_{mx}: "E" loop intermediate read flipflop.

E_x: "E" loop end read flipflop.

E_p: "E" loop write flipflop.

E_{wc}: Enable write signal - from a control panel - enables "cold storage" write heads in memory.

F_c: Fine-countdown-mode indicator flipflop.

F_p: "F" loop write flipflop.

F_s: Also F_{sc} in some writings - signal from a control panel that directs the computer to enter the Prepare to Fill state.

F_x: "F" loop read flipflop.

G_3, G_2, G_1: Binary outputs flipflops.

H_p: "H" loop write flipflop.

H_{mx}: "H" loop intermediate read flipflop.

H_x: "H" loop end read flipflop.

I_c: "I" register interrupt control flipflop.

I_d: "Instruction Search" sector disagreement indicator flipflop.

I_i: Also I_{ic}, the i^th signal input to the computer from an external source for character input, i=1, ..., 5.

I_{mc}: Symbol for a mechanical input signal to the computer, command to enter the Wait state.

I_p: "I" register extra delay flipflop.

I_x: "I" register read flipflop.

I_{24w}: "I" register write flipflop.

J; Control flipflop.

K: Control flipflop.

K'_{hc}: Halt not or run signal from a control console - directs the computer to enter the compute states.

K'_{kr}: Run not or halt signal from a control console - directs the computer to enter the non-compute states.

L_c: "L" register interrupt control flipflops.

L_0: "L" register delay flipflop.

L_p: "L" register extra delay flipflop.

L_x: "L" register read flipflop.

L_{24w}: "L" register write flipflop.

M_{px}: Memory output buffer flipflop.

M_{rc}: Also M_r - master reset signal from a control console, initiates the computer to the Prepare to Operate state.

N_c: "N" register interrupt control flipflop.

N_d: "Number Search" sector disagreement flipflop.

N_p: "N" register extra delay flipflop.

N_x: "N" register read flipflop.

N_{24w}: "N" register write flipflop.

O_{b3}, O_{b2}, O_{b1}: Operation-Code-Buffer register.

O_4, O_3, O_2, O_1: Operation-Code-Storage register.

P_3, P_2, P_1: Phase register.

Q: Special timing flipflop.

R_c: "R" loop interrupt control and mode control flipflop.

R_p: "R" loop write flipflop.

R_x: "R" loop read flipflop.

S: Information read from the sector track of the D17B computer memory.

S_{b3}, S_{b2}, S_{b1}: "Flag-Code" buffer register.

S_3, S_2, S_1: "Flag-Code" storage register.

T_c: Sprocket timing signal; used to direct the computer to accept character inputs.

T_i: Bit times of the computer, i=1, ..., 24.

T_0: "To Time" indicator flipflop.

T_p: "T_p Time" indicator flipflop.

T_x: "T_x Time" indicator flipflop.

U_p: "U" loop write flipflop.

U_x: "U" loop read flipflop.

V_c: "V" loop interrupt control and state control flipflop..

V_p: "V" loop write flipflop.

V_x: "V" loop read flipflop.

V_{38}, V_{37}, ..., V_{31}: Voltage output register number 3.

V_{28}, V_{27}, ..., V_{21}: Voltage output register number 2.

V_{18}, V_{17}, ..., V_{11}: Voltage output register number 1.

$0^{A}1$: Symbolizes that the flipflop named A_1 is set to a logical "zero" condition or "zero set".

$1^{A}1$: Symbolizes that the flipflop named A_1 is set to a logical "one" condition or "one set".

A_1^*: The star or asterisk indicates an external signal to the computer that has been changed in voltage level but has the same logical meaning as the symbol with no asterisk.

A': Prime is used to indicate a logical "not" when A is a logical 1, A' is a logical 0.

Flipflop names and some definitions in this list were taken from Ref (1: 110-114).

USE OF THE D17B IN A HYBRID COMPUTER SYSTEM

Lansing B. Evans and Charles H. Beck
Tulane University
Department of Electrical Engineering
New Orleans, LA 70118

ABSTRACT

Now that the USAF has released a large number of Minuteman D17B Computers which were originally designed for missile guidance, other applications for these excess general-purpose computers have been undergoing a rapid evolution in many fields. This paper describes a new hybrid computing application for the Minuteman D17B Computer which makes use of a large number of the flexible capabilities of these computers. Hybrid computing system design can take full advantage of the capabilities of both analog and digital computers as well as those of special hardware that is possible to develop because of the availability of information in both continuous and discrete form. Motivation for this type of application for the D17B stems primarily from the versatile I/O capability of these machines. The purpose of this paper is to present some of the design considerations and typical applications for a D17B-TR48 hybrid computing system, to describe the present system configuration, and to outline a specific hybrid optimization modeling problem that is being solved using this system configuration. The D17B has been found to be completely satisfactory for this automated design application.

BASIC DEFINITION OF A HYBRID SYSTEM

In a broad sense the field of hybrid computation includes all computing techniques which combine some of the features of digital computation with some of the features of analog computation. The combination of digital and analog devices brings together many of the characteristic advantages of both types of hardware and software. In many cases a disadvantage of one part of the system is more than compensated for by an attribute of another part of the system. The idea of interacting advantages will become more evident by citing some of the capabilities of the two major components of the hybrid system, the analog and digital computers. There will be additional entries to the list of general capabilities which follows depending on the specific computers being used in a particular hybrid system.

Some capabilities of the analog computer include:

1. Dependent variables within the machine are treated in continuous form.
2. High-speed or real-time computation is available with computing speeds limited primarily by the bandwidth of the computing elements.
3. There exists the ability to perform efficiently such operations as addition, multiplication, integration and non-linear function generation; on the other hand, there is very limited ability to make logical decisions or store data.
4. Programming techniques involve patching together the various computing elements.

On the other side, some of the capabilities of the digital computer include the following:

1. All data within the computer is in discrete or quantized form.
2. In general only one operation can be performed at a time and many computing units must be time shared.
3. The facility exists for storing alphanumeric data indefinitely.
4. The ability exists to perform logical decisions and operations using either numerical or non-numerical data.
5. There exists the ability to modify the program extensively on the basis of any calculation.

Almost any computing system is a subset of a complete hybrid system. Whether a system is almost purely digital with only minimal analog capability, nearly all analog with a small amount of digital ability, or anywhere in between, it qualifies as a hybrid system and the principles of hybrid computing may be applied to it.

HYBRID COMPUTER APPLICATIONS

Because of the inherent flexibility of the hybrid computer, there are numerous applications for this type of system. One of the most important is modeling and parameter optimization involving dynamic systems. This particular application makes use of a true hybrid system involving an analog computer, digital computer, and appropriate interface components. As the name implies, modeling requires the use of known experimental input and output data to obtain an accurate mathematical or topological model of the system involved. With a complete hybrid system, relatively complicated and multi-variate models may be considered.

The block diagram of a typical hybrid modeling technique is shown in Figure 1. During a run the actual system or function generator representing the actual system is operated in parallel with the assumed model. The index of performance (IP), which measures the quality of the model, is formed on the analog computer by integrating the square of the error function over the time of the run. Using this technique the digital computer adjusts the model parameters after sampling the IP from the previous run and performing needed optimization calculations. The digital computer makes its decisions on the new parameter settings using an optimization method such as the Tulane Automated Hybrid Optimization (TAHO) technique.

The TAHO technique has been applied to circuits and various physical systems. Currently the TAHO technique is being used for a multi-variate model of the head and neck of a pilot in a crash situation. The data are obtained by monitoring human subjects who ride an acceleration sled along a track. A simulation of these data is used as the actual system portion of Figure 1.

Other typical applications of the hybrid computer system include:

Aerospace Simulation
Simulation of Process Control
Simulation of Man-Machine Systems
Random Process Simulation

USE OF THE MINUTEMAN D17B COMPUTER IN HYBRID APPLICATIONS

In most hybrid systems the digital computer provides control functions as well as the digital computation for the entire system. Therefore, the digital computer must have the ability to communicate not only with the usual digital peripheral equipment but also with the remainder of the hybrid system. The Minuteman D17B Computer has the needed input/output versatility and the flexibility required for a hybrid system. For its size, the D17B has a large number of digital input and output lines, pulse output lines, and analog type output lines.

The programmability of the D17B is also a significant advantage for a hybrid system. The D17B has a complete set of arithmetic, control, and input/output instructions. It is also capable of instruction modification which is an important factor in efficient software for a hybrid system. Because many of the operations required to control the hybrid interface and the analog

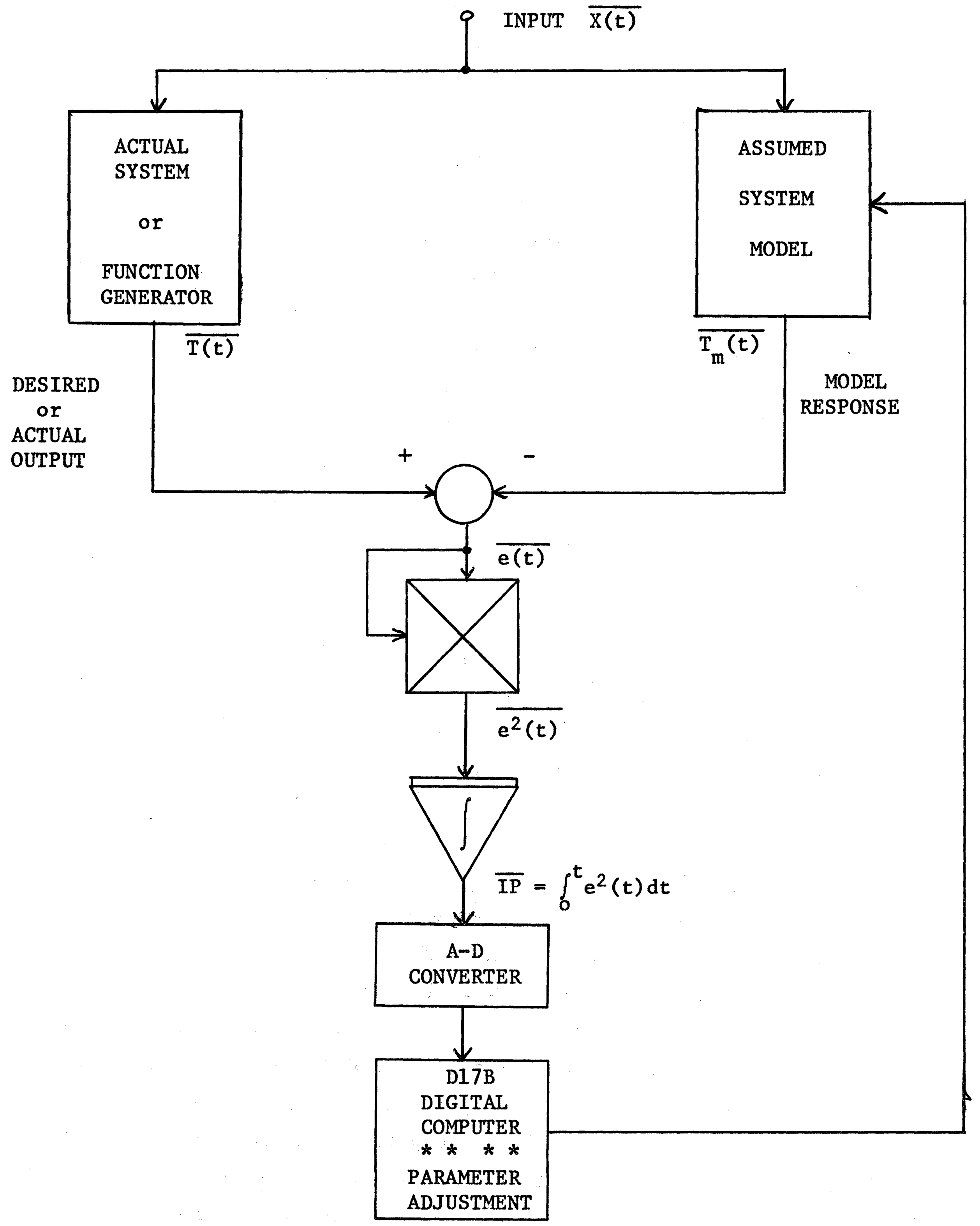


Fig. 1. Tulane Hybrid Optimization Modeling Technique.

computer are of a relatively basic bit-level nature, the machine language programming of the D17B can be far more efficient than the use of a compiler language. In addition, as will be seen in the next section, the D17B input/output instructions are very well suited for hybrid operation.

Perhaps one of the greatest advantages of the D17B hybrid system is the complete flexibility of the configuration. By the nature of the definition of a hybrid system, it may be anywhere from pure digital to pure analog. The D17B will fill the digital computer requirements for any of these systems if the memory size and speed are suitable. The D17B hybrid system described in the following section is a complete hybrid system with full analog and digital capabilities. A full system such as this allows for any operation from merely using the D17B to control the mode of the analog computer to the sampling of analog signals with the A-D converter and performing all processing digitally. This means that practically any computing application can be realized as a subset of a complete hybrid system.

MINUTEMAN D17B/TR-48 HYBRID SYSTEM CONFIGURATION

In designing the configuration of the D17B/TR-48 hybrid computer system, a careful effort was made for full and efficient use of the D17B input/output capabilities. A block diagram of the basic D17B/TR-48 hybrid computer system is shown in Figure 2. It can be seen by inspection of the diagram that this system is a complete, digitally-controlled hybrid system.

The two major paths of information flow in Figure 2 are those from the digital to the analog computer and those from the analog to the digital machine. Since many applications require multi-variate analysis, it is necessary that the major paths in both directions be multi-channel. Within reasonable limits, this presents no problem to the ability of the D17B to control the interface.

The major components of the analog-to-digital information path include a 16-channel multiplexer and an analog-to-digital (A-D) converter. The multiplexer allows 16 analog signals to time share one A-D converter. The D17B controls the operation of both the multiplexer and the A-D converter as shown in Figure 3. In order to permit one of the inputs of the multiplexer to be switched to the input of the A-D converter, a four-bit binary address, between 0 and 15 decimal, is transmitted to the multiplexer address register. The COA

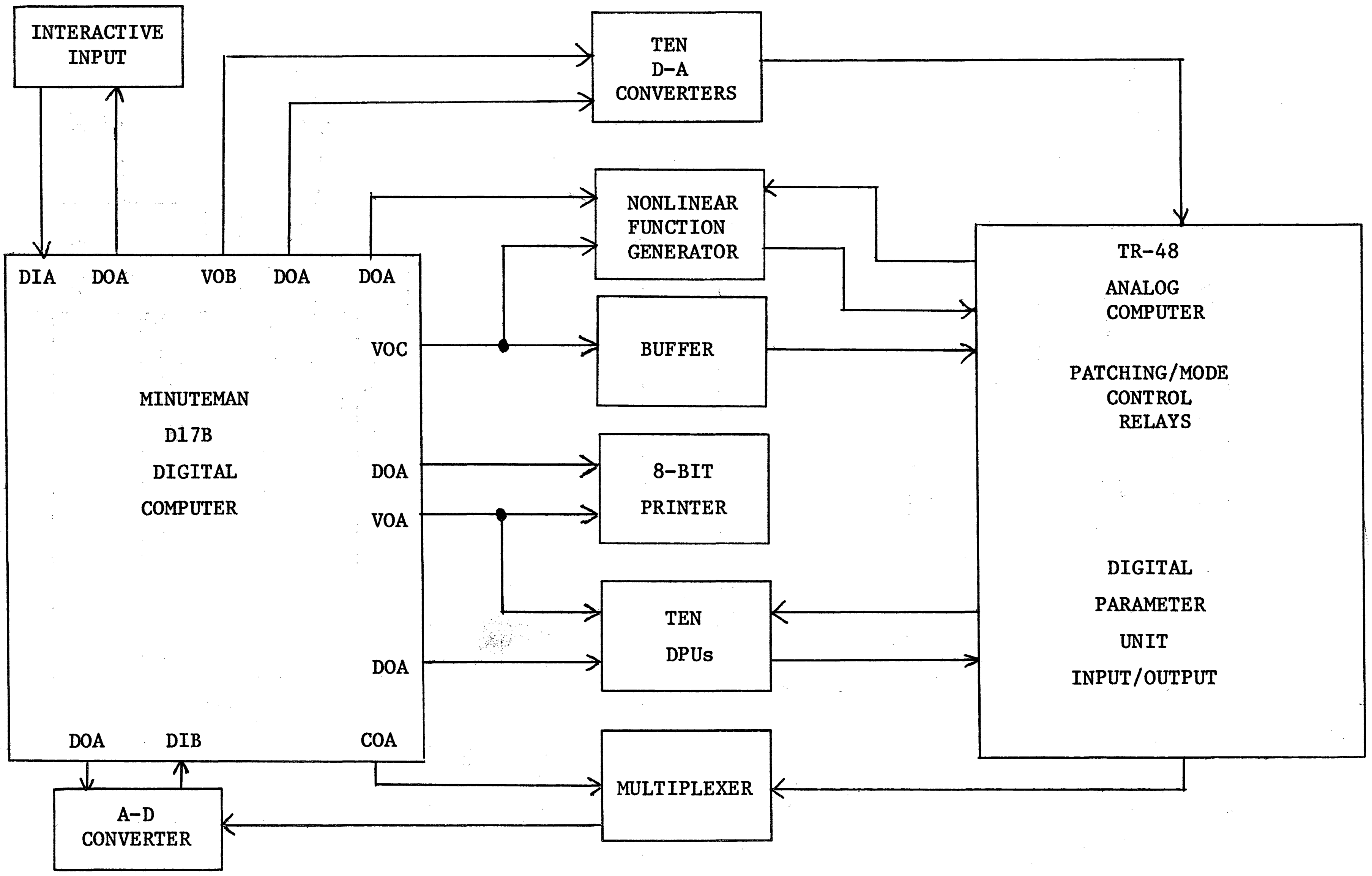


Fig. 2. Tulane D17B/TR-48 Hybrid Computer System.

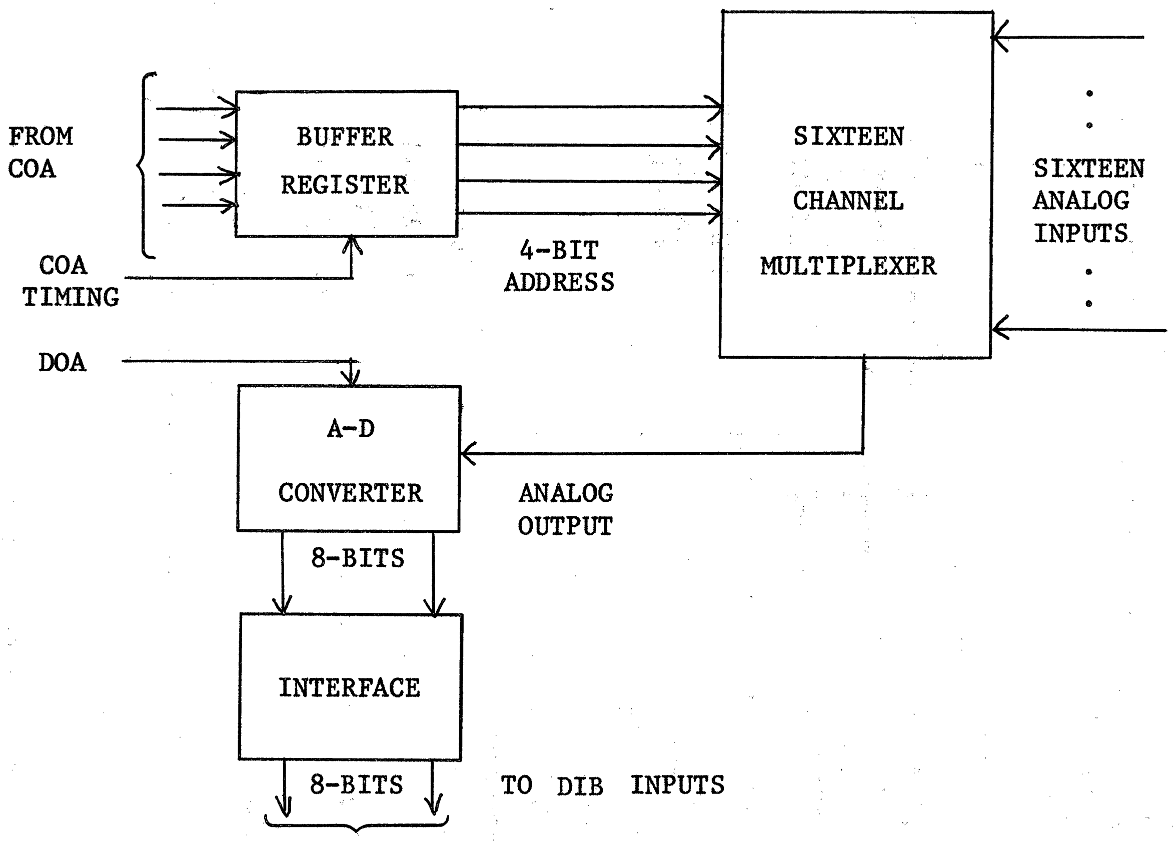


Fig. 3. Hybrid System A-D and Multiplexer.

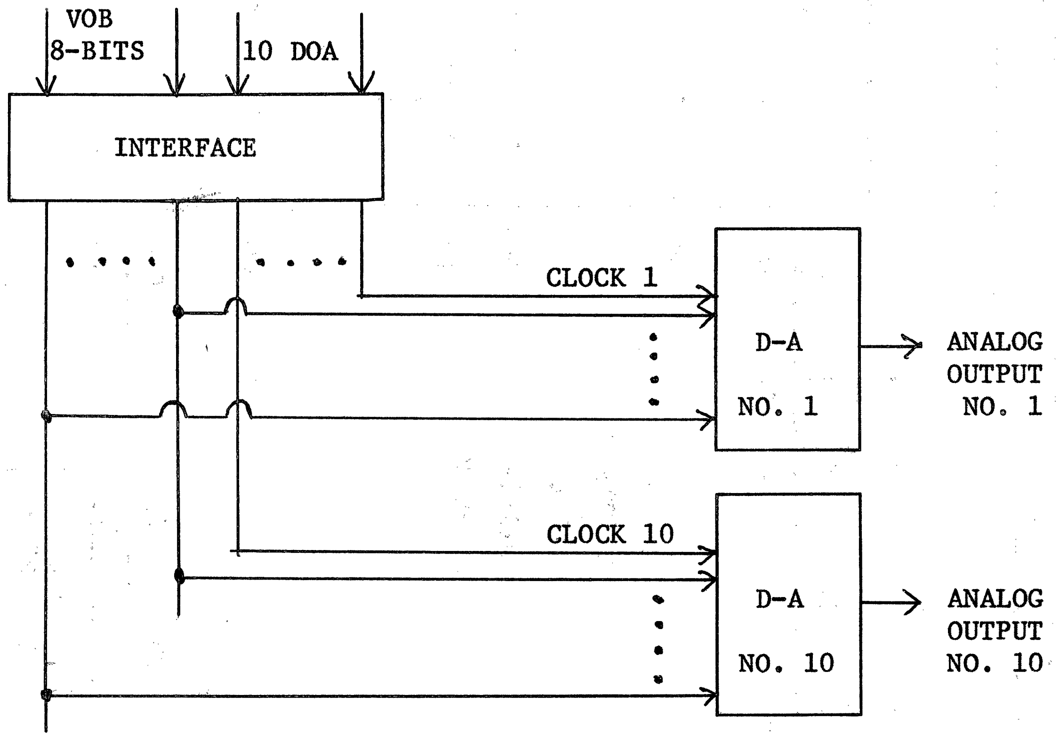


Fig. 4. Hybrid System D-A Converters.

(character output) instruction is used for this purpose because it sends out a clock pulse along with a four-bit parallel pulse type word which can easily and conveniently be interfaced to the multiplexer address register. The use of the COA instruction in this case also means that only one machine language instruction will be needed to control the multiplexer. Once the proper analog signal has been applied to the A-D converter, two discrete output (DOA) instructions are executed to cause the A-D converter to digitize the analog input. Two DOAs are used to generate a pulse as required by the A-D converter.

The ten digital-to-analog (D-A) converters shown in Figure 2 are the most important links in the flow of information from the digital computer to the analog computer. The D-A converters are also under complete control of the D17B. Figure 4 shows how the D17B loads and controls the D-As. The D-As used in the Tulane hybrid system accept an 8-bit digital input. This 8-bit input is loaded into one of the internal D-A buffer registers and converted to a proportional analog value when a clock pulse is applied to that D-A. Since no input is loaded into a D-A until a clock pulse is applied, the digital inputs of all D-As may be connected together and tied to an 8-bit digital bus.

The D17B has a voltage output (VOB) instruction which may be used for an 8-bit parallel digital output. Eight bits from the accumulator are transferred to the VOB register when a VOB instruction is executed. Once the VOB has been executed, the desired 8-bit word is applied to all D-A inputs. A pair of DOAs are then used to load the digital word into the proper one of the ten D-As. Only a few machine language instructions are needed for D-A control.

In addition to the basic A-D and D-A units, the interface contains two somewhat more sophisticated components. These are the digital parameter units (DPU) and the digitally controlled nonlinear function generator (DCNFG). These devices involve interactions between digital and analog signals rather than a conversion from one form to the other.

The DPUs provide for the digital control of the parameters in the analog computer patching. This control is performed electronically at high speed by the D17B. With this ability the parameters of the model may be changed at high speed under program control. Basically the DPU is a hybrid multiplier. It multiplies the 8-bit digital word transferred from the VOA lines by the corresponding analog signal from the TR-48 as shown in Figure 5.

The interface between the D17B and the DPUs is quite similar to the one for the D-As shown in Figure 4. The 8-bit VOA lines are fed to the inputs of

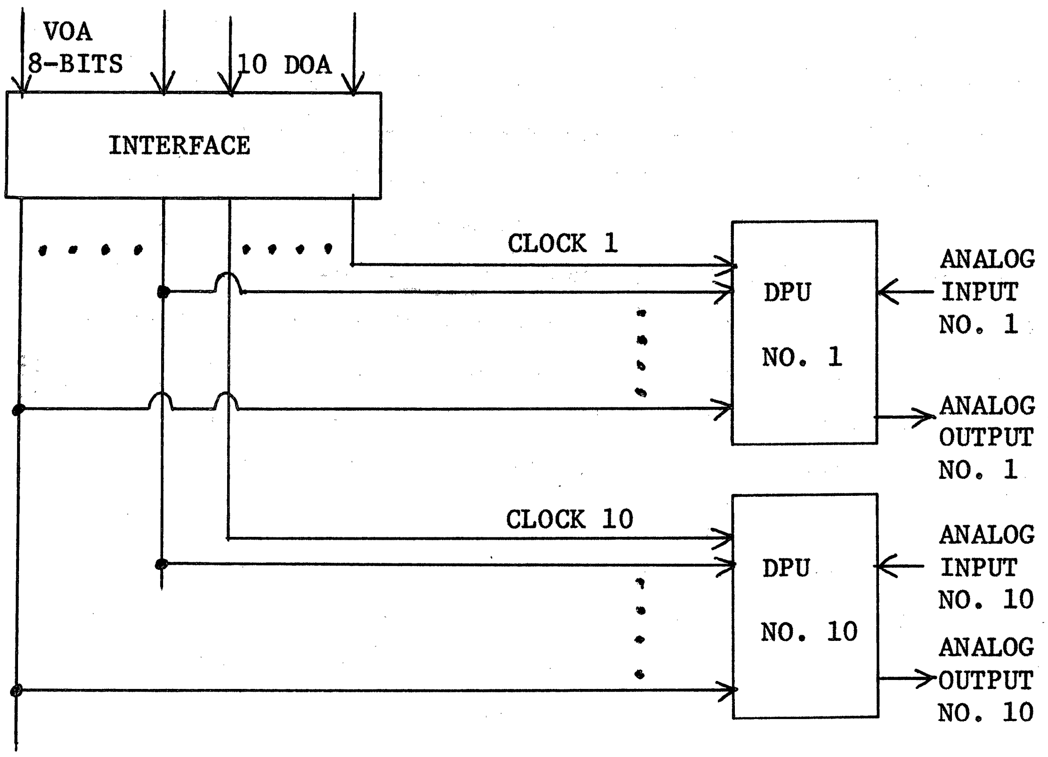


Fig. 5. Digital Parameter Units.

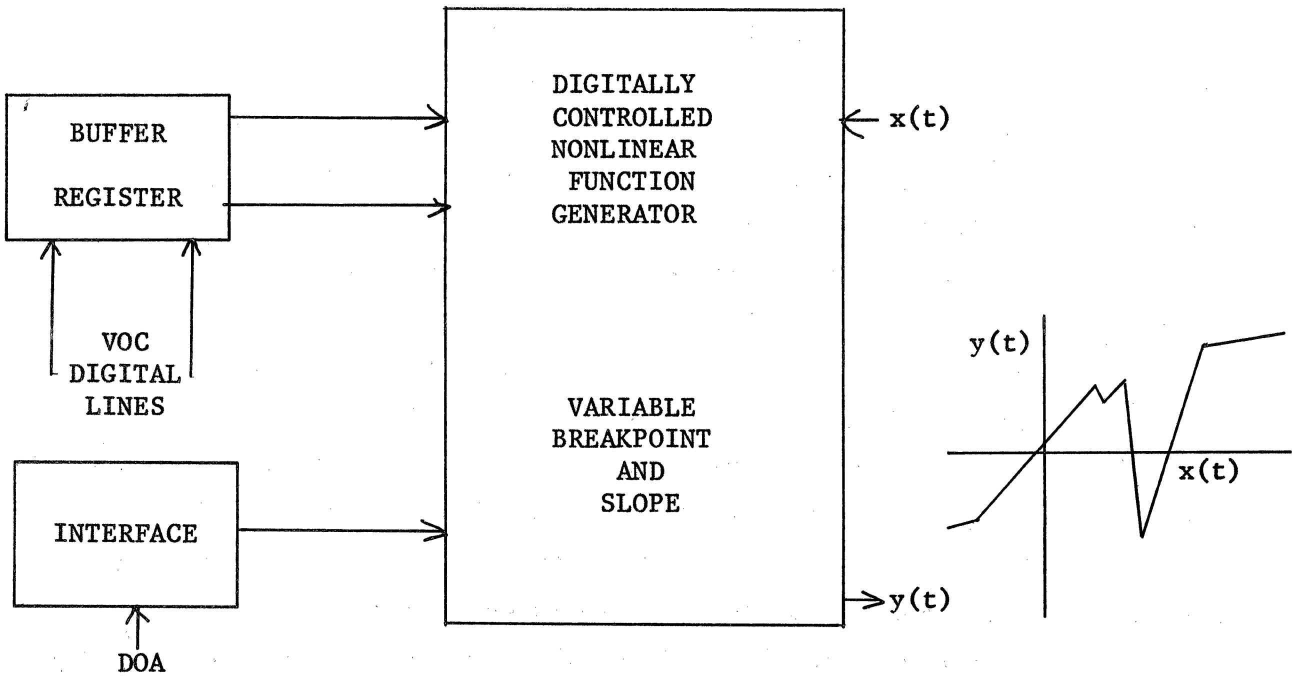


Fig. 6. Digitally Controlled Nonlinear Function Generator.

all DPUs. The digital word is then loaded using two DOA instructions as in the case of the D-A converters. The analog inputs and outputs for the DPUs are patched on the TR-48 patch board. These extremely powerful hybrid computing elements are also convenient to use with D17B computer machine language.

The nonlinear function generator is a digitally controlled variable breakpoint function generator. The function may have up to ten segments of any desired slope, and the slopes and breakpoints can be programmed into the generator under D17B program control. The slopes and breakpoints are loaded into the generator using the 8-bit VOC lines and DOA pulses. After the loading of the desired function, the output, y(t), takes on the function output for the corresponding analog input, x(t), as shown in Figure 6. The setup time for the function generator is fast and versatile as is the case for the DPUs.

The mode control on the TR-48 Analog Computer is operated from external relays controlled by the D17B Computer as shown in Figure 2. The outputs from the VOC digital lines are loaded into a buffer which drives the relays on the external patch board. The relays that are not used for mode control may be used for high-speed patching changes in the analog computer program.

CONCLUSIONS

While the D17B is a small general-purpose digital computer, the versatile input/output capability of this machine has allowed for the development of a compact, efficient hybrid computer system when used in conjunction with a TR-48 Analog Computer. The most important benefit of a computer system such as the Minuteman D17B/TR-48 hybrid system described in this paper is that it may be used in a wide spectrum of computing applications.

REFERENCES

M. H. Kuo, Automated Modeling of Dynamic Systems Using Hybrid Computer Optimimization Techniques, Doctoral Dissertation, Tulane University, New Orleans, Louisiana, 1971.

C. H. Beck, *et al*, "Direct modeling of nonlinear systems using hybrid computer optimization techniques," Conference Proceedings, Seventh Annual Allerton Conference on Circuit and System Theory, C-4, 6, Urbana, Illinois, 1969.

G. A. Bekey and W. J. Karplus, Hybrid Computation, John Wiley & Sons, Inc., New York, 1968.

DESIGN OF A BINARY DISPLAY FOR THE D17B COMPUTER

by

HARRY S. WARFORD, Capt, USAF, BSC*
DAVID S. MORAN, GS-9

INTRODUCTION

Hardware development for the D17B computing system has proceeded rather slowly as a spare time interest at the USAF School of Aerospace Medicine. As a result, the binary display technique described by this paper has not been optimized for future growth of the total system. However, it has evolved into compact and relatively inexpensive design through effective use of machine inherent characteristics. The overall design calls for the capability to monitor any register or memory track with random access to any particular sector. At present, the hardware for monitoring the one-word registers is complete and the design is complete for the remaining circuitry to randomly address memory location.

TECHNIQUE

The D17B utilizes the 24-bit full word for programming but actual word length on the disc memory is 27 bits. The 3-bit "dead time" has been used in our design to facilitate display without the need for additional holding registers while maintaining the capability to update the display each word time. During the 24-bit times representing the computer word a 24-bit serial entry shift register is filled from the D17B while the light emitting diode display is blanked. Then during the 3-bit dead time the shift register is halted and the parallel outputs drive the display. At the end of the 3-bit times, the display is blanked and the information is changed or reloaded into the register.

For random access the sector channel is to be monitored as shown in Figure 1. Sector number information will be captured in an external register and compared with the numbers selected on a set of octally coded thumbwheel switches. When the information agrees, the proper shift pulses are gated to the aforementioned 24-bit register to capture the next word of the chosen channel. Channel choice is by a second set of thumbwheel switches and the decoding internal to the D17B.

*To be presented by Michael Jenkin, Major, USAF, MC, USAF School of Aerospace Medicine, Brooks AFB Texas 78235

CIRCUITRY

Figures 2 and 3 show the circuits used to implement the basic display. The derived control signals are shown in Figure 4. It must be noted here that the logic signals were considered to be of positive sense for ease of design with commercial DTL logic. Additionally,the levels were not translated but the signals were merely attenuated to produce a five volt swing and the integrated circuits were operated with "VCC" at 0 VDC and "ground" at minus 5 VDC. Figure 5 shows the proposed control signals to accomplish random access and Figures 6 and 7 show the present design being constructed for this purpose. At present, all logic has been broken into modules representing a byte of data and implemented with commercial plug-in cards and racks.

OTHER DEVELOPMENTS

Little effort has been expended on hard-copy output thus far since the surplus Flexowriters obtained for this project rapidly deteriorated and failed early in the project. However, an extremely simple and inexpensive technique was used to provide input only by mounting a second set of leaf switches in tandem with those used to operate the punch select magnets of our remaining operable Flexowriter. This provides complete electrical isolation thus alleviating the need to modify the Flexowriter power supply and requires a single capacitor to shape the timing pulses.

Additionally, an extra tape reader has been converted to stand-alone use as depicted in Figure 8. A manual I/O panel similar to those described at earlier user's meetings provides for miscellaneous control.

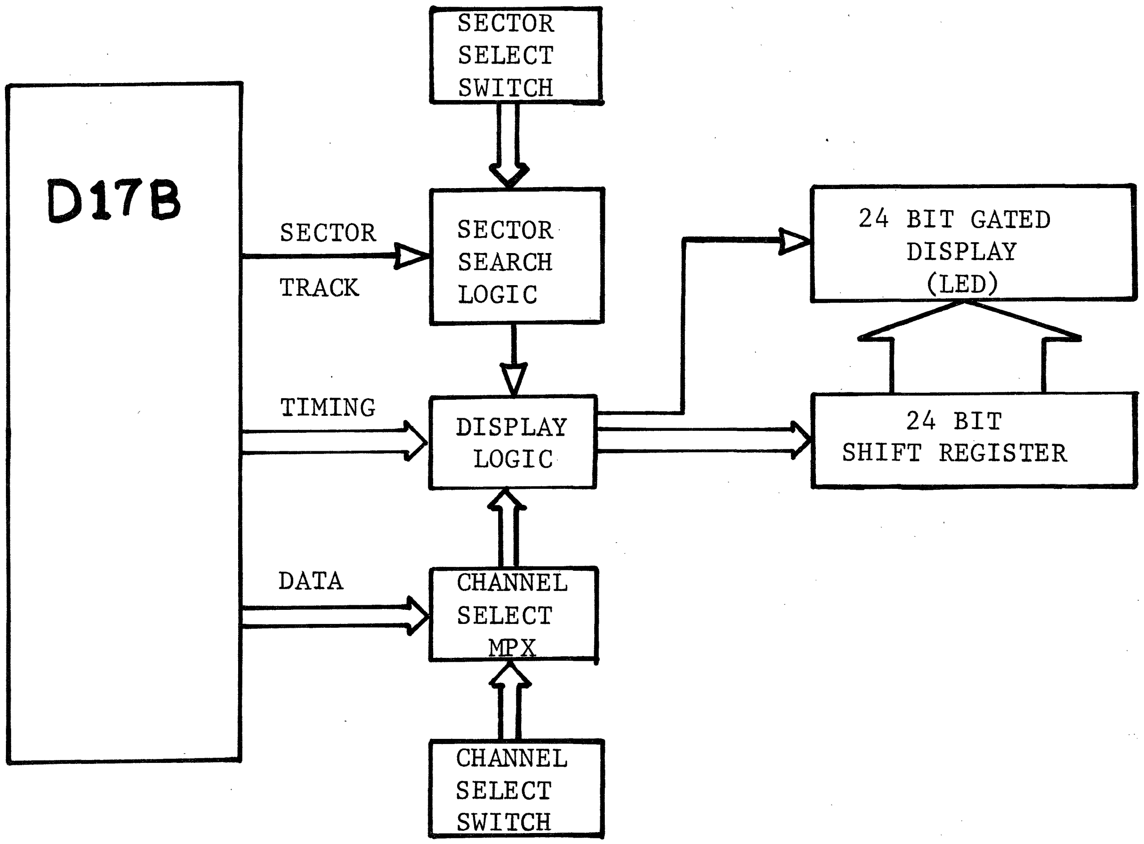


FIG. 1 BINARY DISPLAY SYSTEM

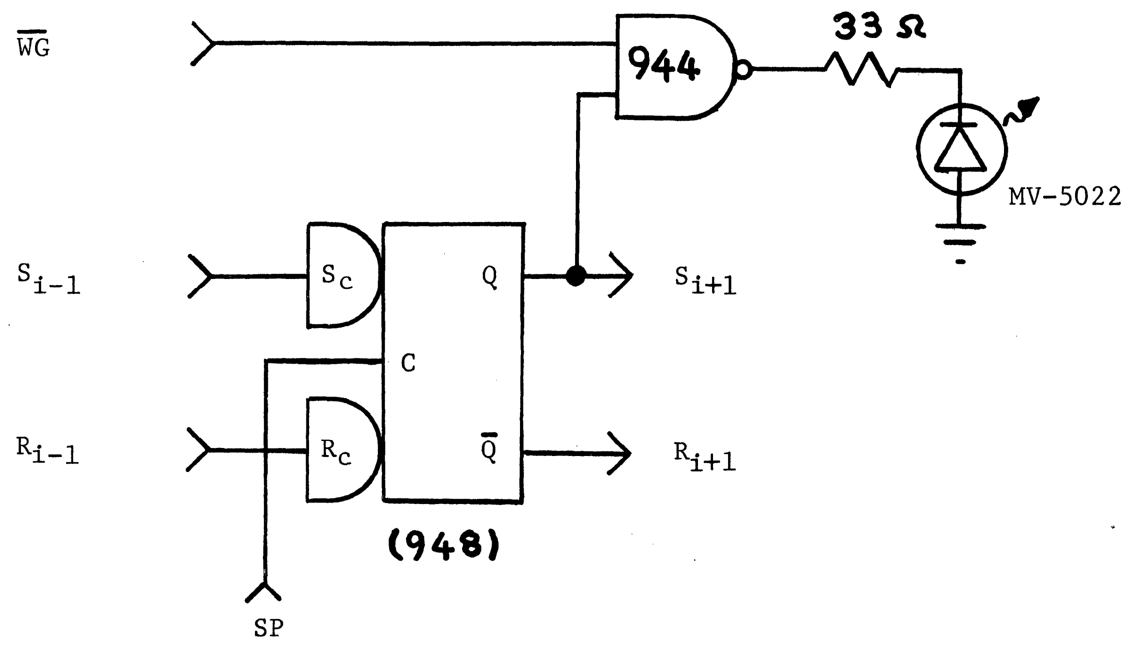
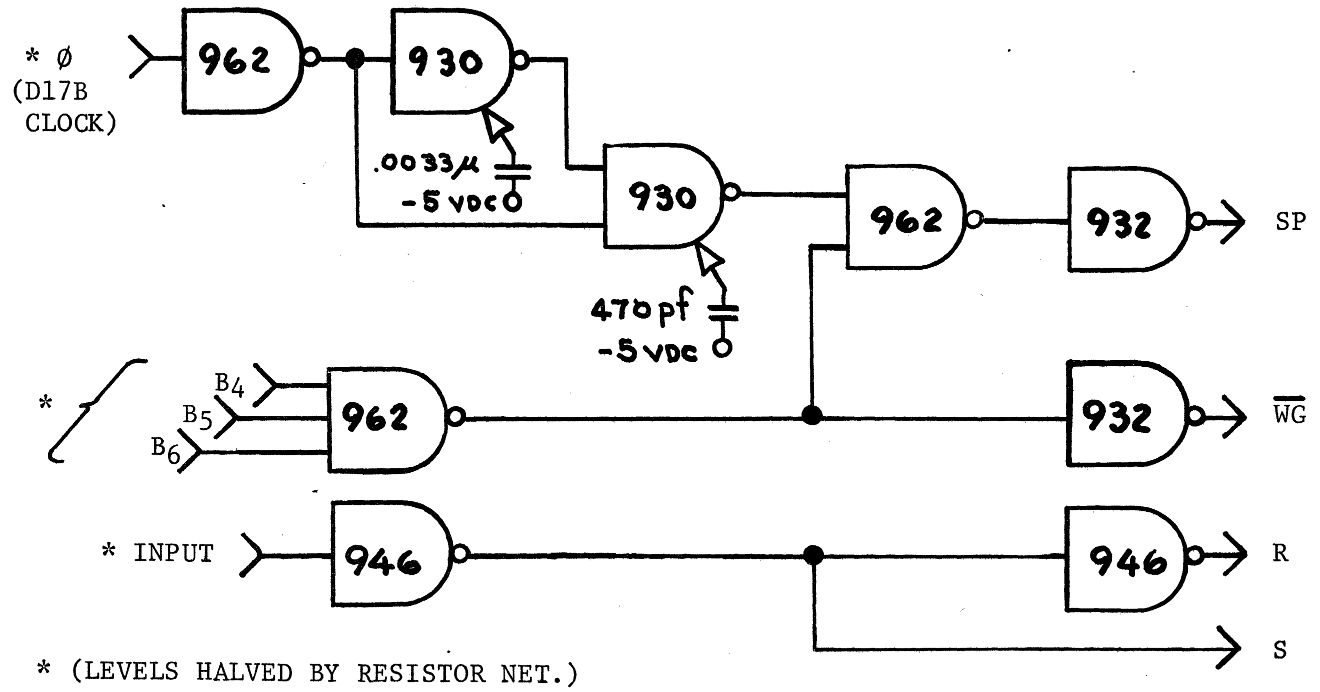


FIG. 2 ONE BIT OF DISPLAY REGISTER



* (LEVELS HALVED BY RESISTOR NET.)

FIG. 3 DISPLAY CONTROL

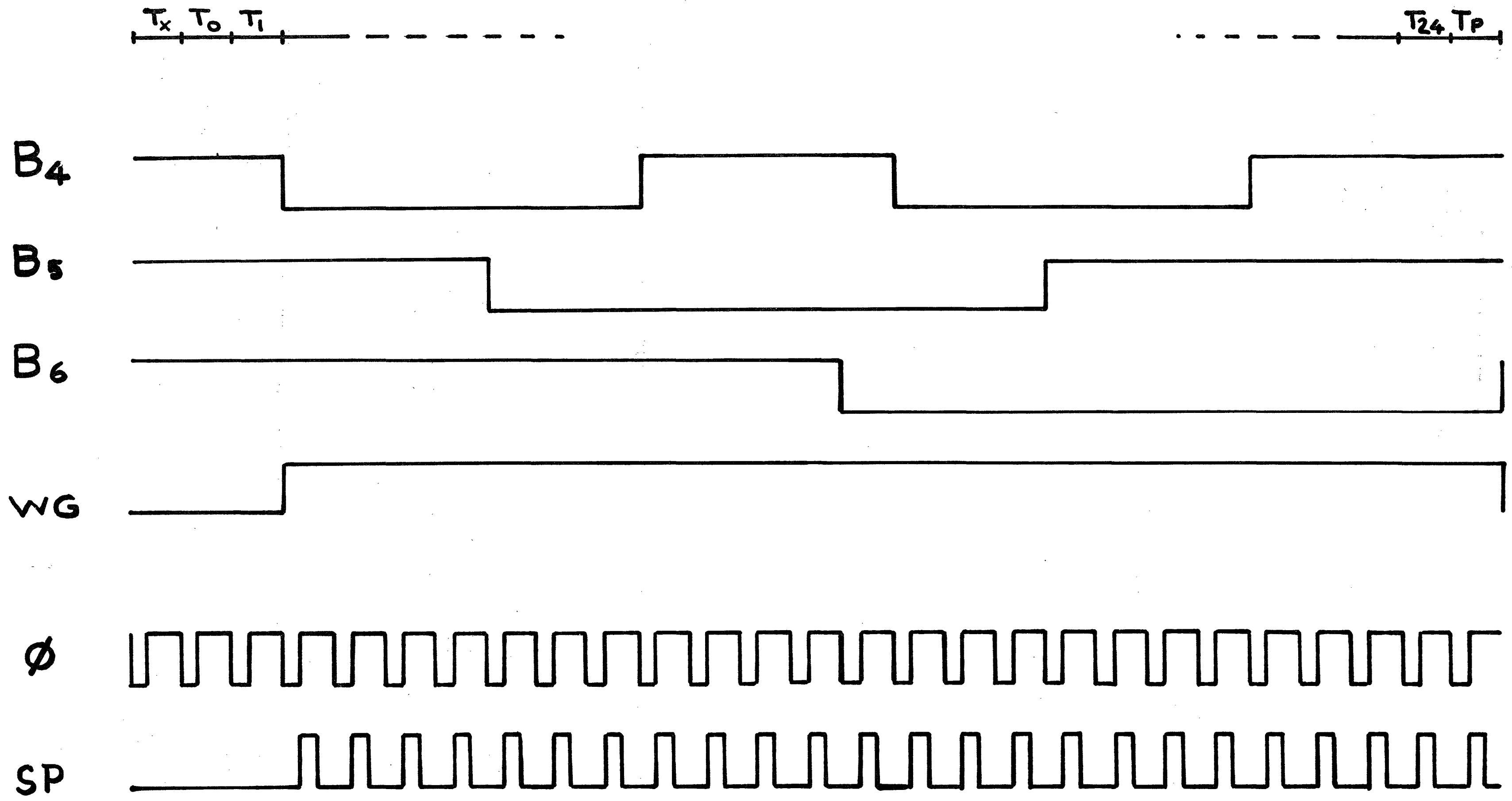


FIG. 4 TIMING FOR BASIC DISPLAY

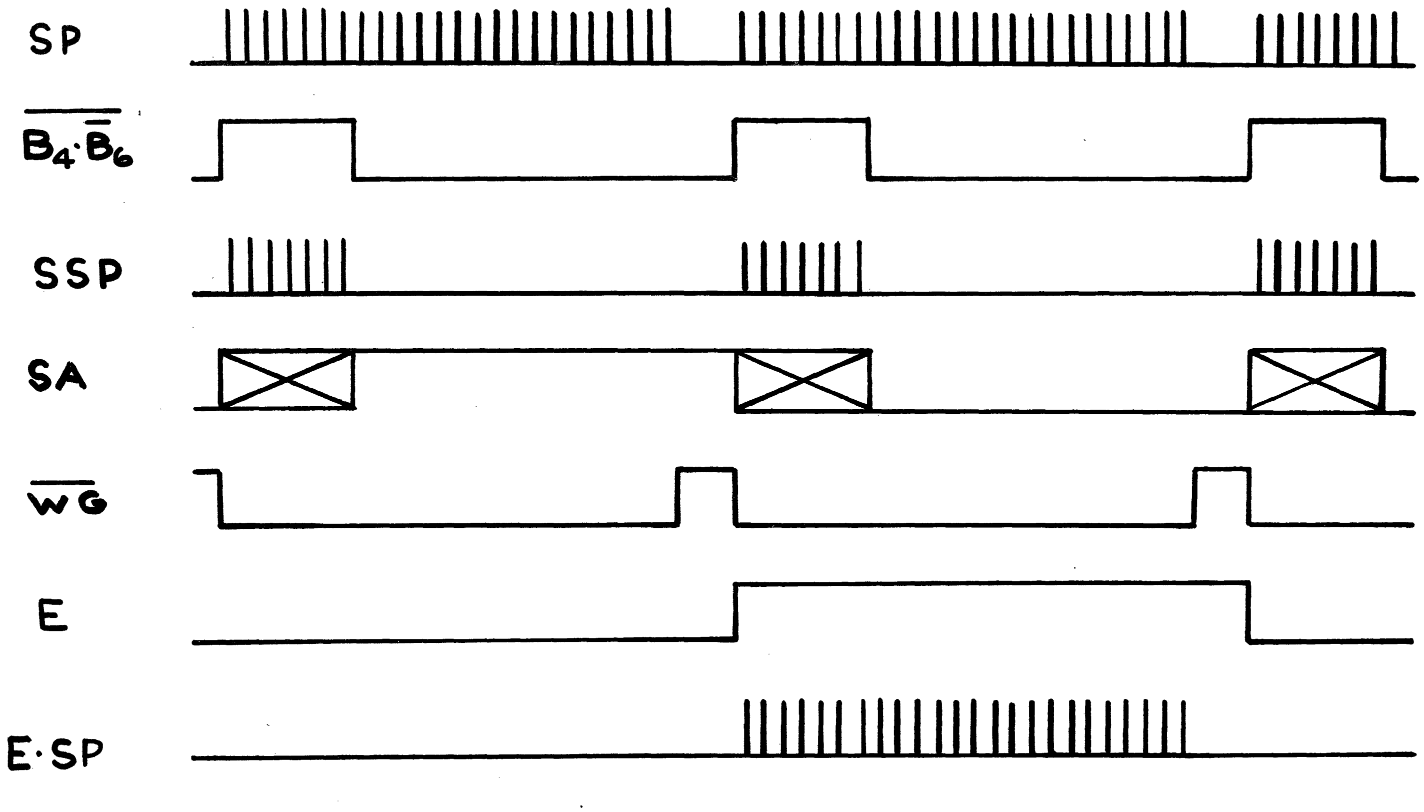


FIG. 5   TIMING FOR RANDOM ACCESS

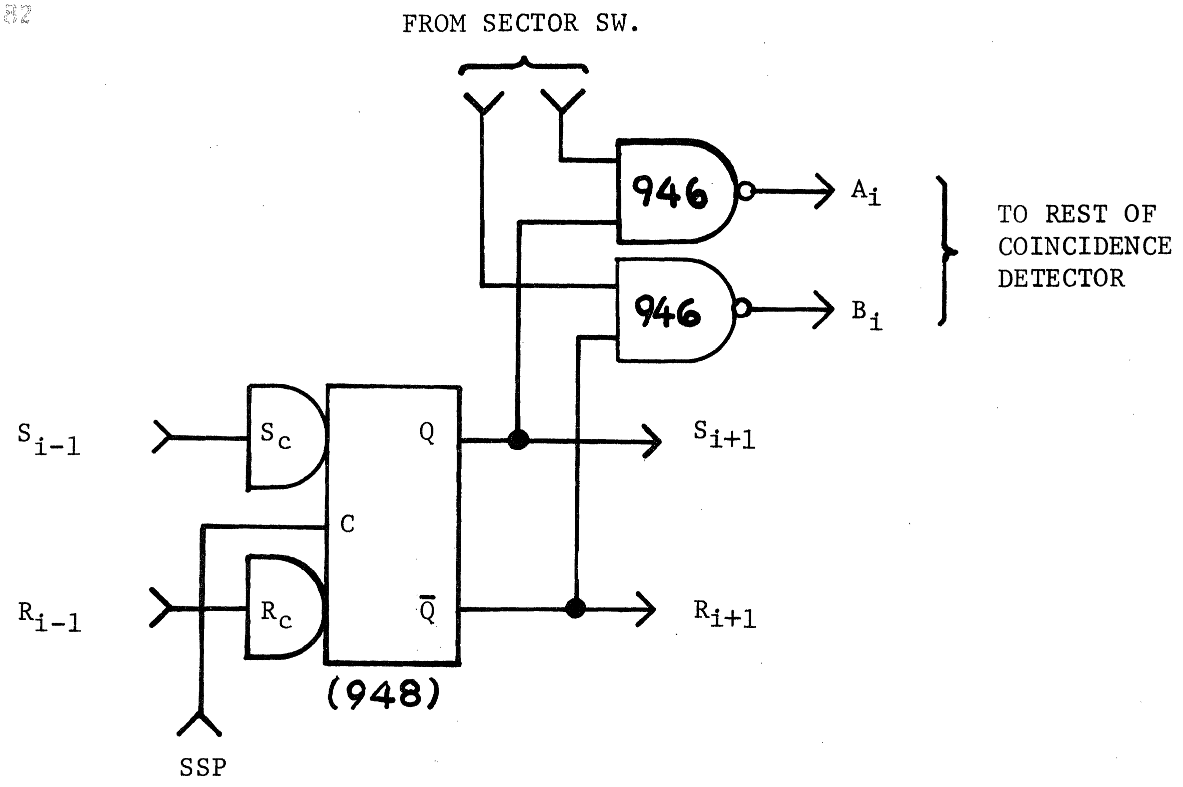


FIG. 6 ONE BIT OF SECTOR NUMBER REGISTER

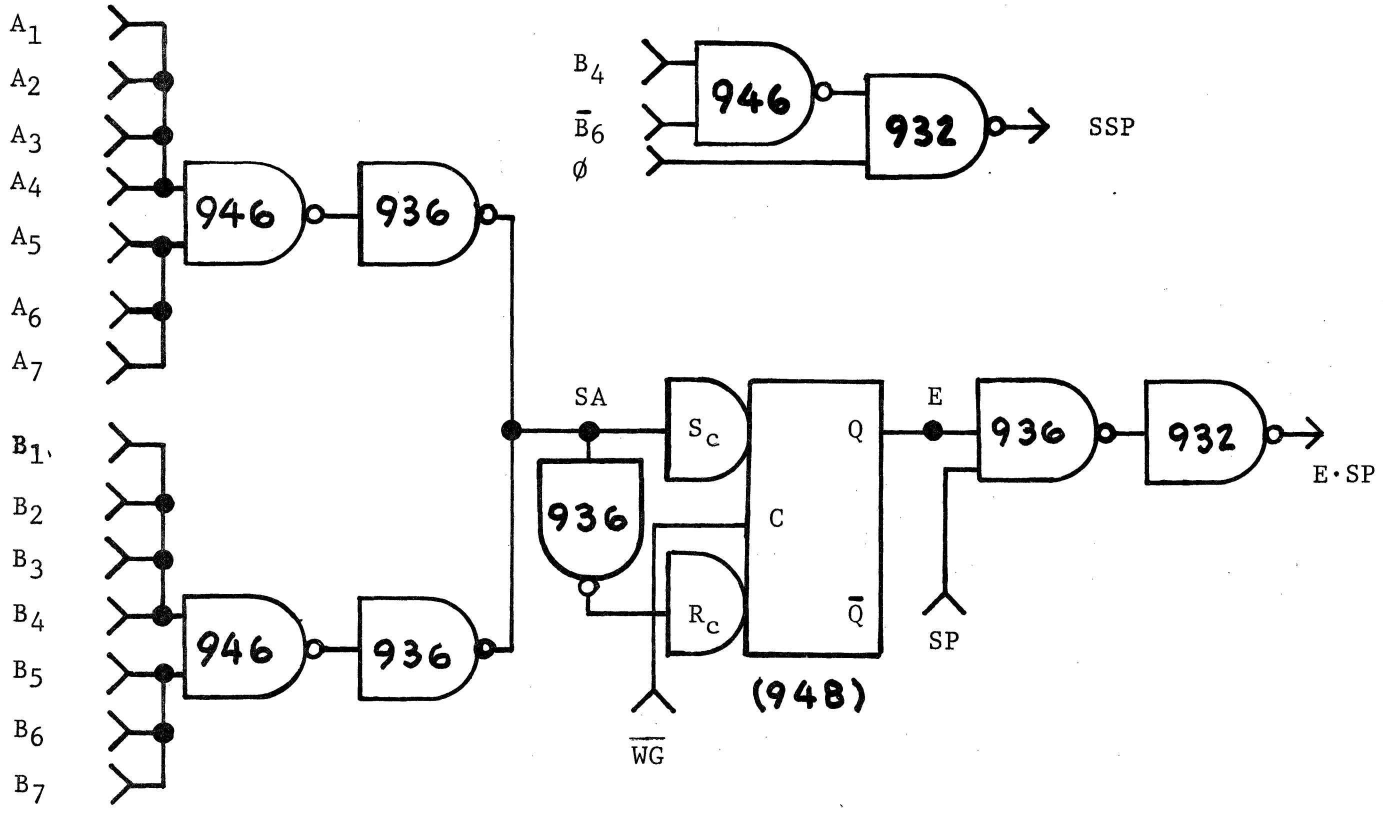


FIG. 7 CONTROL FOR SECTOR SEARCH

AUTOMATED DATA ACQUISITION AND WAVEFORM ANALYSIS USING THE MINUTEMAN D17B COMPUTER

Charles H. Beck and Yih-Young Chen
Systems Laboratory, Tulane University
New Orleans, LA 70118

SUMMARY

This paper describes the development of an automated data acquisition and waveform analysis system for a single-channel AutoAnalyzer using the Minuteman D17B Computer. This automated system is a typical example exhibiting several essential features on which other laboratory data acquisition systems can be based. The complete AutoAnalyzer Analysis System was set up in the Tulane Systems Laboratory, and total protein analysis of blood serum was executed as a representative example of the use of this system. An identical system which was also developed in the Systems Laboratory was shipped to the Walter Reed Army Medical Center for use in biochemistry research. This system was used to demonstrate the success of this development during the Fourth Meeting of the MCUG. The consistency, accuracy, reliability, and cost-effectiveness of this system have shown the usefulness of the D17B Computer in this application.

INTRODUCTION

Current advances in medical practice are making ever increasing demands on medical laboratories which result in a shortage of skilled technical staff. Medical laboratory instruments are finding increased use for clinical and also research chemical analysis. Some laboratory instruments such as AutoAnalyzers have increased the productivity that is possible for a given size staff, but there is still need for improved efficiency if the total laboratory workload is to be increased without the addition of technical staff or the purchase of more expensive equipment.

Typical data from these instruments are in the form of recorder traces consisting of a series of peaks. These peaks are conventionally interpreted by graphical evaluation performed manually. This tedious and inaccurate task of manually performing waveform analysis suggests the need for cost-effective automation.

Automation offers a potential solution as well as a different set of problems. Various automated analysis systems are available through either commercial or research organizations, but these systems represent considerable cost and offer little flexibility. Most chemical technologists can be trained to operate a variety of laboratory instruments. However, special-purpose commercial analyzers require increased hardware cost if they are to be used in more than a limited application. This additional hardware cost will be needed for each new task unless the analyzer includes a general-purpose programmable computer. This latter solution offers flexibility, but the cost may even be prohibitive for many laboratories especially if multiple units are needed.

One method of reducing the high cost of laboratory automation is to use excess DoD computer equipment such as the Minuteman D17B computers from the Minuteman I missiles. The main requirements for successful use of the D17B as a general-purpose minicomputer are software development and the interfacing of various peripheral I/O devices and instruments. Application areas will expand as the interfacing of these sensing and display devices is developed.

OPERATIONAL PRINCIPLES OF THE AUTOANALYZER

The AutoAnalyzer is a continuous-flow chemical analysis system in which individual operations are performed on a flowing stream containing specimens such as blood serum. The AutoAnalyzer makes it possible to measure the concentration of various constituents of the blood serum as individual patient samples flow through the system. The standards, controls, and samples are measured continuously against a fixed reference. The final results are traced on a chart by a recorder stylus.

It is possible to place a re-transmitting potentiometer on a shaft of the pen drive for the purpose of creating an electrical signal whose amplitude is proportional to that of the plotted waveform. The re-transmitting device can be connected to the recorder shaft of the AutoAnalyzer in a few minutes thus providing the only modification required for interfacing the AutoAnalyzer to the automated D17B data acquisition and waveform analysis system.

There are a few operational procedures that should be followed quite carefully in operating the AutoAnalyzer. The essential requirement for the correct operation of the entire system is proper flow of the various liquids and air through the pump, coils, and the flow cell. One of the most critical characteristics of the flow is the bubble pattern. It must be uniform through the system for accurate analysis. It is always desirable to assure that all tubes have clean, smooth inside walls. In general, the AutoAnalyzer has been proven to be a reliable and consistent instrument.

SET-UP OF THE AUTOMATED MINUTEMAN/AUTOANALYZER SYSTEM

A Minuteman/AutoAnalyzer Analysis System has been developed in the Tulane Systems Laboratory under a research contract supported by the Army Medical R&D Command, and total protein analysis of blood serum has been executed as one representative example of the use of this system. This automated data analysis system for use with an AutoAnalyzer is one typical example of a system which includes a Minuteman D17B Computer and has potential for cost-effective medical application in clinical and research laboratories.

It has been demonstrated that this system gives accurate, consistent, and useful results. The cost-effective use of the D17B for automated concentration analysis from an AutoAnalyzer compares favorably with commercial special-design analyzers, but the capability is similar to analyzers which offer the increased flexibility that is possible with a small general-purpose computer.

The D17B requires a 28 V DC ± 1 V power supply that is capable of providing 20 A in continuous duty. The power supply should be checked for proper output voltage under actual operating conditions before it is connected to the D17B. The D17B should be operated in a cool, dry environment to minimize any hardware failures. Therefore, an air conditioned laboratory is desirable in many locations, and a fan is necessary to cool the electronic circuitry as well as the memory. A relatively simple control panel can be constructed for the D17B which allows the operator to supply specific initializing and interactive input signals.

A Model ASR35 Teletype was used as an I/O device in this system. The interface between the D17B and the Teletype has three modes of operation: Character Input, Print Out, and Direct Input. The Character Input mode permits the operator to load data into the D17B via keyboard or tape reader. The Print Out mode is entered when the D17B is to execute a print out subroutine. The Direct Input mode is used to load data directly into the Accumulator when a discrete input instruction is executed. The interface will automatically go into the proper mode depending on the state of the D17B. The cost of the interface was minimized by time-sharing hardware for the various modes of operation.

The estimated cost for installation of an initial Minuteman/AutoAnalyzer Analysis System is approximately $440 plus 60 man-hours technician time. This cost does not include amounts for the AutoAnalyzer, Teletype, or the external power supplies and the D17B which were acquired as excess property.

AUTOANALYZER WAVEFORM ANALYSIS

In the development of the automated system, a typical waveform from the AutoAnalyzer was simulated using D17B hybrid function generation initially instead of using output directly from the laboratory instrument. D17B hybrid simulation offers the advantage of faster operation than is possible when the waveform is obtained directly from the AutoAnalyzer. The zero-order hold waveform obtained from a D-A converter was improved by filtering with a simple first-order feedback network.

Data sampling is an essential consideration in automated waveform analysis. The sampling theorem specifies the lower limit for the sampling rate, and the upper limit is determined by the signal waveform and the characteristics of the specific A-D converter that is used. The amplitude spectrum of the simulated signal was monitored using a wave analyzer for estimating the sampling rate. The minimum sampling rate required for this waveform was also investigated by using two additional approaches, namely practical considerations and explicit function representation. With the data from these analyses an optimal sampling rate of one sample per two second interval was determined. This sampling rate is practical and has been shown to be suitable for one percent (1%) accuracy.

The output waveform from the AutoAnalyzer consists of a series of peaks corresponding to the individual standards and specimen samples. By observing typical output charts obtained from clinical laboratories, potential sources of distortion in the analog signal were identified. Distortion is characterized by such effects as noise, carry-over, drift, irregularity in the crest, and early or late rise. Therefore, error and sensitivity analysis were applied in the sampling rate determination.

AUTOANALYZER ANALYSIS PROGRAM

The output of the AutoAnalyzer is a signal with a waveform consisting of a series of peaks. The height of each peak represents the concentration of a certain compound or radical in the corresponding specimen. Criteria have been established to check whether the distortion in the signal is above the maximum acceptable level for a qualified peak. A peak is qualified only when all the criteria are met. A complete program is composed of various subroutines which are linked together by a standardized subroutine linkage technique.

The flow chart for the AutoAnalyzer Analysis Program (AAP) is shown in Figure 1. The system checkout subroutine will check for proper operation of the system, both software- and hardware-wise, daily or as requested by the operator. During the execution of this subroutine, the peak values and ACCEPTABLE/NOT ACCEPTABLE messages will be recorded indicating whether or not the system checkout was satisfactory for each peak.

After the system checkout, the program determines the base-line as the zero reference level before the calibration curve is validated. After the base-line determination, there are five calibration standards to be validated. The calibration waveform is then followed by the specimen samples. A peak is identified by falling readings following rising readings. The sampling rate is under program control.

The first peak of each batch of samples is used for time-base synchronization purposes. When the first peak is detected, a counter is set so that the following peaks must arrive during specified time limits. The duration of the allowed peak time window is also under program control. A 20 second time window has been used in this development.

At the end of the calibration curve validation, a message will be printed to indicate whether the calibration curve is acceptable or not. A drift standard may be included to form the batch depending on whether drift correction is required. The peak height of the drift standard is compared to the value of stored data. If the difference is beyond a certain limit, i.e. a drift is detected, a linear correction will be applied to the previous peak heights. Sample concentration which is the most meaningful pathological data is calculated by using linear interpolation in order to agree with convention. The concentration value and the associated data will be recorded in a specific format which can be designed as preferred by modifying the program.

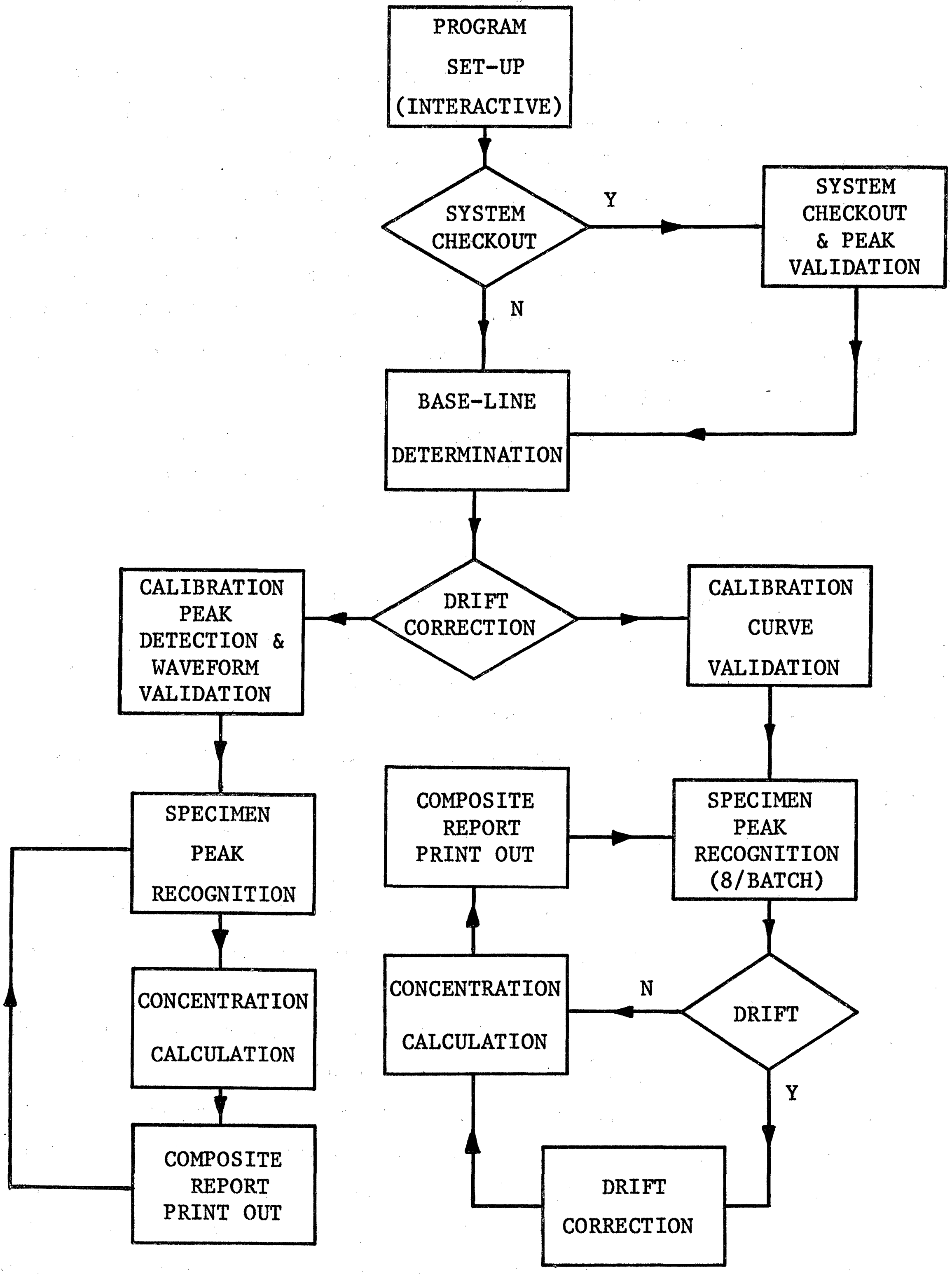

PROGRAM SET-UP (INTERACTIVE)
SYSTEM CHECKOUT
Y
N
SYSTEM CHECKOUT & PEAK VALIDATION
BASE-LINE DETERMINATION
DRIFT CORRECTION
CALIBRATION PEAK DETECTION & WAVEFORM VALIDATION
CALIBRATION CURVE VALIDATION
SPECIMEN PEAK RECOGNITION
COMPOSITE REPORT PRINT OUT
SPECIMEN PEAK RECOGNITION (8/BATCH)
CONCENTRATION CALCULATION
N
DRIFT
Y
CONCENTRATION CALCULATION
COMPOSITE REPORT PRINT OUT
DRIFT CORRECTION

Fig. 1. AutoAnalyzer Analysis Program Flow Chart.

This program is designed with the ability to detect possible errors. If an error is detected, a buzzer will be triggered by a pulse generated using a Discrete Output line from the D17B. An error message will also be printed by the Teletype. From this indication the operator can decide on the appropriate action. The entire automated system will be in the "Halt and Proceed" mode waiting for the operator to furnish further instructions.

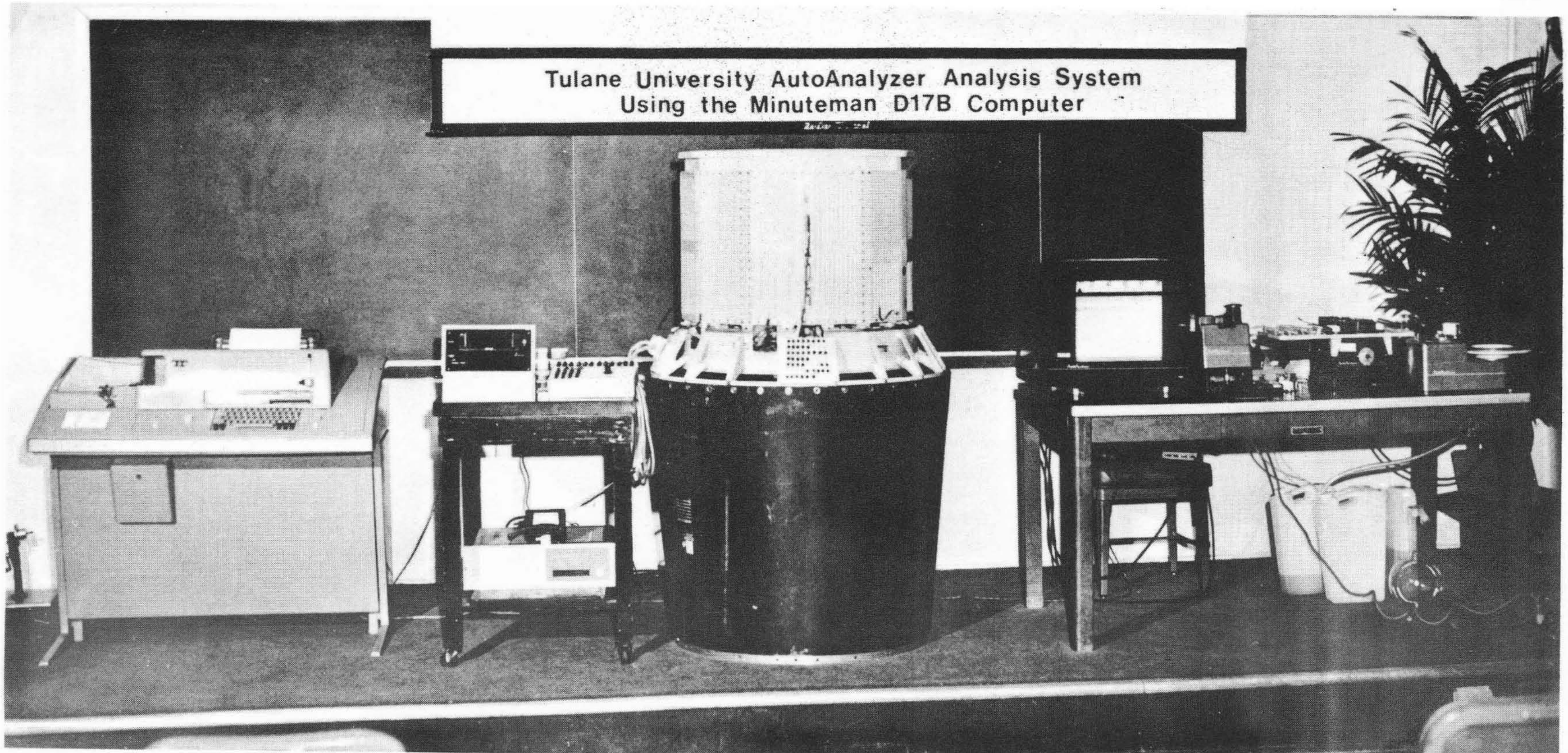
SYSTEM DEMONSTRATION AT WRAIR

A system which typified the prototype development of the Minuteman D17B/ Technicon single-channel AutoAnalyzer was shipped to Walter Reed Army Institute of Research at the Walter Reed Army Medical Center and was used to exhibit one of the many possible methodologies during demonstrations on June 5 and 7 by the Systems Laboratory staff. A U.S. Army photograph of the system appears in Figure 2. Resulting demonstrations have shown the D17B Computer to be capable, reliable and cost-effective in this application.

CONCLUSIONS

The Minuteman ICBM Weapons System currently includes approximately 300 operational Minuteman I missiles. Several hundred additional reliable D17B minicomputers from Minuteman I missiles are being declared excess by the USAF. These small general-purpose computers with extremely flexible I/O originally cost $234,000 each.

Although the D17B does not provide all the capability of large-scale computers, it does resemble them functionally, and it possesses a number of similar features. One unique feature of the D17B is the reliability factor which has been quoted by USAF to be 5.5 years MTBF for over 1,000 units. These computers can provide numerous computing requirements with application to medicine and other fields. Work that has been completed in the Systems Laboratory at Tulane has demonstrated that the modifications required to provide for the use of the D17B in a continuous duty laboratory environment are not only feasible but cost-effective. A system using the D17B has been developed for automated data acquisition, control and waveform evaluation


Tulane University AutoAnalyzer Analysis System
Using the Minuteman D17B Computer

necessary to perform automated concentration analysis from an AutoAnalyzer. Commercial equipment for such analysis is available, but these systems have high costs that are often difficult to justify especially if a number of similar systems are desired.

Construction of a complete single-channel AutoAnalyzer Analysis System which includes a D17B Computer is estimated to require 60 man-hours of technician time. The total cost for an initial unit will be approximately 50% of the current price of a special-purpose commercial unit. The cost comparison for a dual-channel unit is approximately 30% of that for a special-purpose commercial unit. These cost estimates include both time and materials for a complete system with a teletype used for I/O. The cost will be reduced very considerably if these units are mass produced. Although the capability of this system is similar to analyzers which include the versatility of small general-purpose computers, the cost effectiveness is quite advantageous when compared with commercial special-purpose analyzers.

Accuracy, consistency, simplicity, and cost-effectiveness are the main features of the AutoAnalyzer Analysis System. The methodology for total protein analysis which was demonstrated during this development is just one example of the use of this system with the AutoAnalyzer, and the AutoAnalyzer Analysis System is just one typical application for the D17B Computer. The D17B has been found to be extremely capable and reliable in this application.

This system was designed such that there would be no disruption of the usual laboratory procedure. In addition, the simplicity of the system is such that existing personnel can easily be trained to operate the automated system. This system will give laboratory technologists and professional staff more time to evaluate results and investigate specific abnormal data; i.e., effective utilization of less-skilled personnel is possible.

The AAS has full alpha-numeric capability and a proven accuracy of better than 1%. The accuracy and reliability obtained with the AAS are considerably increased compared to that obtained by manual means because of the reduction of human errors in procedure and data manipulation. It has also been shown that there will be added flexibility in the implementation of sophisticated experimental and analysis techniques. This system offers future adaptability to the analysis of signals from other laboratory instruments and to other computing applications by simply reading in different programs.

REFERENCES

1. C. H. Beck, D17B Computer Programming Manual, Minuteman Computer Users Group, Tulane University, New Orleans, 1971.

2. C. H. Beck, Proceedings of the Third Meeting of the Minuteman Computer Users Group, Minuteman Computer Users Group, Tulane University, New Orleans, 1971.

3. W. L. White, M. M. Erickson and S. C. Stevens, Practical Automation for the Clinical Laboratory, C. V. Mosby Co., St. Louis, 1968.

4. Laboratory Organization and Data Handling Familiarization, T & T Technology, Inc., 1971.

5. R. H. Laessig and P. P. Tong, "Digital Concentration Analyzer for the Single Channel AutoAnalyzer," American Laboratory, 67, September 1970.

6. L. G. Whitby and D. Simpson, "Experience with on-line computing in clinical chemistry," J. Clin. Path., vol. 22, suppl. no. 3, 107-124, 1969.

7. P. Gray and J. A. owen, "Experience with on-line computer processing of data from an AutoAnalyzer complex," Clin. Chim. Acta, vol. 24, 389-399, 1969.

8. N. P. Wilburn and I. D. Coffin, "Combination of on-line analysis with collection of multicomponent spectra in an on-line computer," IBM J. R&D, vol. 13, no. 1, 46, January 1969.

9. M. A. Evenson, et. al., "Application of an on-line data acquisition system using the LINC computer in the clinical chemistry lab," Automation in Anal. Chem., vol. 1, 137, 1968.

10. M. A. Blaivas and A. H. Mencz, "Progress report on the use of a computer in the automated clinical chemistry laboratory," Automation in Anal. Chem., vol. 1, 368-372, 1967.

ACKNOWLEDGEMENTS

This project has been supported in part by the U. S. Army Medical Research and Development Command under Contract No. DADA17-71-C-1019. The authors are particularly indebted to Capt. Timothy C. Doyle, Computer Systems Branch of the USAMRDC, and to Mr. Billy G. Bass, Division of Biochemistry at WRAIR, for their valuable assistance and contributions to the success of this project. Although the success of this development is due to the efforts of the entire staff of the Systems Laboratory, special credit is due to Mr. Lansing B. Evans for his supervision of the installation and demonstration of the system at WRAIR.

R E G I S T R A T I O N

Fourth Meeting of the Minuteman Computer Users Group

Sheraton-Silver Spring

June 5-6, 1972

Capt. Douglas J. Allen
Air Force Institute of Technology
AFIT-SE
Wright Patterson AFB, OH 45433

Mr. Richard F. Babler, Chief
Defense ADPE Reutilization Office
Cameron Station
Alexandria, VA 22314

Mr. Ned G. Barber
Psychology
City College of New York
138th St. & Convent Avenue
New York, NY 10031

Dr. James D. Bargainer
Associate Professor
Electrical Engineering
University of Houston
Houston, TX 77004

Mr. David W. Barrett
Chemistry
Johns Hopkins University
Charles & 34th Street
Baltimore, MD 21218

Mr. John P. Bartell
Computer Equipment Analyst
Defense Supply Agency
DSAH-LS (DARO)
Cameron Station
Alexandria, VA 22314

Dr. Charles H. Beck
Professor of Electrical Engineering
Tulane University
New Orleans, LA 70118

Mr. Robert A. Beken
Programmer
Air-Medic Micronesia
Box 3
Glenn Dale, MD 20769

Prof. Philip J. Best
Associate Professor, Psychology
University of Virginia
Gilmer Hall
Charlottesville, VA 22901

Maj. Robert C. Brady
USAF HQ/XOOFC
5916 Minuteman Road
Springfield, VA 22152

Dr. Sam J. Cipolla
Assistant Professor, Physics
Creighton University
Omaha, NE 68131

Mr. Richard S. Cook, Director
ADP Resource Management Div.
General Services Administration
18th & E Street
Washington, DC 20406

Mr. Charles R. Cummings
Computer Equipment Analyst
National Library of Medicine
Lister Hill Center
8600 Rockville Pike
Bethesda, MD 20014

Dr. Frederic M. Davidson
Assistant Professor
Electrical Engineering
Johns Hopkins University
Charles & 34th Street
Baltimore, MD 21218

Mr. Preston G. Davis
Systems Engineering
Tennessee Technological University
Route 3
Mt. Juliet, Tennessee 37122

Mr. Edward Dowgirth
(Address not furnished)

Mr. Antonio Duque
Design Engineer
Fighton, Inc.
65 Sullivan Street
Rochester, NY 14605

Mr. John W. Dyer
Electrical Engineering
Brigham Young University
B-34
Provo, UT 84601

Mr. John W. Ecklin
Computer Equipment Analyst
DSA
Cameron Station
Alexandria, VA 22314

Mr. Carlos J. Escude
Lowell Technical Institute
450 Aiken Street
Lowell, MA 02173

Mr. Lansing B. Evans
Electrical Engineering
Tulane University
New Orleans, LA 70]]8

Dr. Joseph F. Fennell
Space Physics Lab. 120/1813
Aerospace Corporation
P.O. Box 95085
Los Angeles, CA 90045

Mr. Wayne R. Fenner
Plasma Research Labs 120/1405
Aerospace Corporation
P.O. Box 95085
Los Angeles, CA 90045

Prof. William Fishbein
Psychology
City College of New York
]38th & Amsterdam Avenue
New York, NY 10031

Mr. Charles S. Fly
Electrical Engineering
Tennessee Technological University
Rt 7, Box 238A
Clarksville, TN 37040

Mr. David S. Frager
Systems Administrator
BUMED, NNMC
14300 Cantrell Road
Silver Spring, MD 20904

Mr. William A. Garrison
Chemistry, BH462
University of Missouri - St. Louis
8001 Natural Bridge Road
St. Louis, MO 63121

Robert M. Goldberg
Radiology
Albert Einstein College
1300 Morris Park Avenue
Bronx, NY 10461

Mr. Charles E. Greene
Spec. Assistant to Director
Division Data Processing
HSMHA, PHS, DHEW - NCHS
Rm. 10A56, 5600 Fishers Lane
Rockville, MD 20852

Dr. Robert W. Gruebel
Associate Professor of Physics
Stephen F. Austin State University
Box 3044 SFA Station
Nacogdoches, TX 75961

Mr. William W. Hart, Jr.
Property Utilization Specialist
Excess Equipment Branch
General Services Administration
Washington, DC 20406

Mr. M.A. Henry
Chemistry Department
42 Whitmore Laboratory
Penn State University
University Park, PA 16802

Mr. T.A. Holden
Chief, ADP Reutilization Branch
General Services ADM, OADMS
Washington, DC 20406

Mr. W. Lee Hunter
Systems Analyst, E-291
DHEW, PHS - Ctr. for Disease Control
1600 Clifton Road
Atlanta, GA 30333

Mr. Paul S. Jean
Tennessee Technological University
Electrical Engineering
511 N. Peachtree Ave.
Cookeville, TN 38501

Maj. Michael A. Jenkin
Dir. of Plans and Hospitilization
OIC Automation Planning Group
USAF Office of Surgeon General
HQ Usaf/S6HM Forrestal Building
Washington, DC 20013

Col. Robert L. Jones
Deputy Comptroller (Data Automation)
OASD (C) OSD
1126 Buchanan Street
McLean, VA 22101

Mr. Jay M. Kaplan
Theraputic Radiology
V.A. Research Hospital
333 E. Huron
Chicago, IL 60611

Prof. Walter S. Koski
Professor of Chemistry
Johns Hopkins University
Charles & 34th Street
Baltimore, MD 21218

Mr. Richard W. Kuberry
Supervisory Geophysicist
NOAA-ERL
Fredericksburg Geomagnitic Center
Corbin, VA 22446

Dr. Gary B. Lamont
Associate Professor
Electrical Engineering
Air Force Institute of Technology
Wright Patterson AFB, OH 45433

Mr. William A. Lee
Division of Neurosugery
Medical University of South Carolina
Charleston, SC 29401

Dr. John R. Lehmann
Program Director
Computer Systems Design
National Science Foundation
Washington, DC 20550

Mr. Fred Lichtenberger
CED, B65
Picatinny Arsenal
Dover, NJ 07801

Mr. David A. McBlain
Director of Computing Services
Austin College
Box 1265
Sherman, TX 75090

Mr. Leonard D. McGann
Director
Division of Data Processing
NCHS, PHS
P.O. Box 12214
Raleigh, NC 27612

Mr. William M. McGhee
President
Fighton, Inc.
65 Sullivan Street
Rochester, NY 14605

Mr. Thomas F. McInnis
Automatic Test Systems Dept.
Advanced Projects Lab.
Hughes Aircraft Company
Building 6 MS E165
Culver City, CA 90230

Dr. W. Gordon Monahan
Biophysics
Sloan-Kettering Inst.
425 E. 68th Street
New York, NY 10021

Mr. Cesar A. Sepulveda-Nunez
Optical Sciences Center
University of Arizona
Tucson, AZ 85721

Mr. Richard Ohran
Electrical Engineering
Brighan Young University
173 FELB
Provo, UT 84601

Mr. George E. Pidick
Physics Department
University of South Florida
4202 Fowler Avenue
Tampa, FL 33620

Professor Icarus E. Pyros
Head, Office of Computer Science
US Merchant Marine Academy
Kings Point, L.I., NY 10024

Mr. John G. Romanski
Electrical Engineering
Johns Hopkins University
205 Barton Hall
Baltimore, MD 21218

Mr. Harvey S. Schultz
Biophysics
Sloan Kettering Institute
425 E. 68th Street
New York, NY 10021

Dr. Leo H. Soderholm
Investigations Leader
USDA ARS
Rm 213 Ag. Eng.
Iowa State University
Ames, IA 50010

Mr. Edward J. Taborek
Chemical Engineer, SMEFB-EE
US Army MERDC
Fort Belvoir, VA 22060

Dr. Theodore M. Thorson
Physicist, Theraputic Radiology
V.A. Research Hospital
333 E. Huron
Chicago, IL 60611

Mr. Donald T. Torres
Bureau of Industerial Hygiene
Baltimore City Health Department
602 American Building
Baltimore & South Streets
Baltimore, MD 21202

Prof. Jack C. Towne
Chemistry
University of Dallas
Irving, TX 75060

Mr. Alfred E. Traver
Mechanical Engineering
Tennessee Technological University
Cookeville, TN 38501

Dr. Edward F. Turner, Jr.
Chairman, Physics Department
Washington and Lee University
Lexington, VA 24450

Mr. Michael W. Vannier
Mechanical Engineering
University of Kentucky
Lexington, KY 40506

Mr. John R. Van Roekel
Gas Dynamics Labs
University of Michigan
Ann Arbor, MI 48105

Dr. James J. Whalen
Assistant Professor
Electrical Engineering
State University of New York at Buffalo
Rm 2B 4232 Ridge Lea Road
Amherst, NY 14226

Mr. Edward M. Wysocki
Electrical Engineering
Johns Hopkins University
Barton Hall
Charles & 34th Streets
Baltimore, MD 21218

Mr. Samuel P. Zieske
Director of Instrumentation
Austin College
Sherman, TX 75090