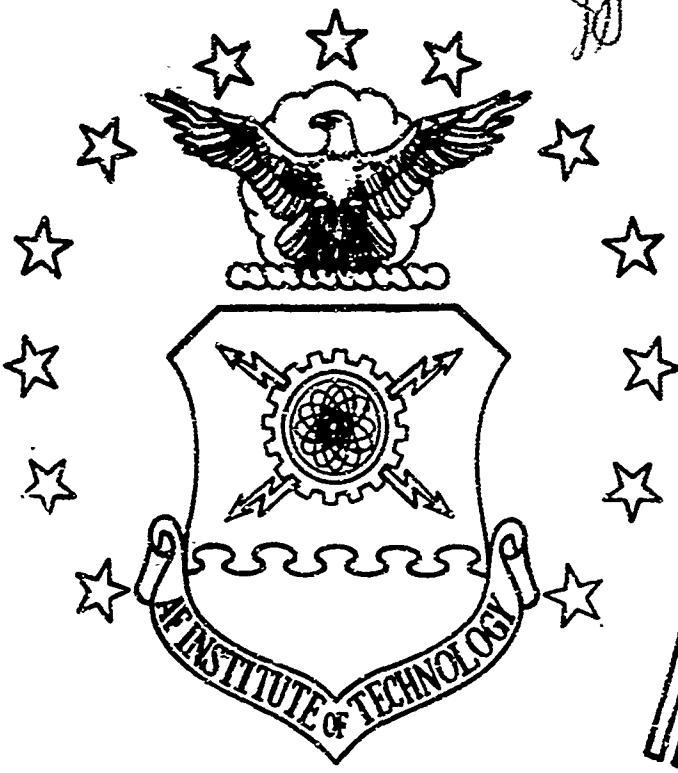
AD-760 757

REUTILIZATION OF THE MINUTEMAN GUIDANCE COMPUTER AS A NUMERICAL/PROCESS CONTROLLER

Raymond V. Cicirelli, et al

Air Force Institute of Technology
Wright-Patterson Air Force Base, Ohio

March 1973



UNITED STATES AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

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

ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433	Unclassified
	2b. GROUP

REPORT TITLE

REUTILIZATION OF THE MINUTEMAN GUIDANCE COMPUTER AS A NUMERICAL/PROCESS CONTROLLER

DESCRIPTIVE NOTES (Type of report and inclusive dates)

AFIT Thesis

AUTHOR(S) (First name, middle initial, last name)

Raymond V. Cicirelli, Lt, USCG
John M. Hill, 1/Lt, USAF

REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
March 1973	~~119~~ 148	29

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	GE/EE/73-5
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10 DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

Details of illustrations in this document may be better studied on microfiche.

Approved for public release; IAW AFR 190-17

JERRY C. HIX, Captain, USAF
Director of Information

12. SPONSORING MILITARY ACTIVITY
Air Force Institute of Technology Wright-Patterson AFB, Ohio

13 ABSTRACT

As a result of the D17B computers from the Minuteman I missiles being made available to qualifying organizations, several studies have been implemented to assess the usefulness and adaptability of the D17B in other applications. This report is directed toward their use in control applications.

The report outlines a few of the basic concepts of numerical control and process control toward the end of adapting these computers to those purposes. The primary emphasis is on numerical control with merely a small account being given of process control. The primary functional features of the computer are presented in detail for the D17B and merely mentioned in passing for the D37C computer (Minuteman II) with the expressed hope that the results of the work can be applied to the D37C.

An analog computer model of a machine positioning system was developed in order to test the control that could be exerted on a machine by the computer under program control. Various programs were developed to control the machine model in X and Y coordinates. The programs were able to control the machine model continuously over a specified trajectory. Representative paths describing arcs of circles and squares in arbitrarily selectable locations are presented as results. In addition the shaft-angle of a laboratory motor-trainer was controlled in discrete increments.

DD FORM 1 NOV 68 1473

Unclassified
Security Classification

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Minuteman Computer						
D17B Computer						
D37C Computer						
Numerical Control and D17B Computer						
Process Control and D17B Computer						
Reutilization of Minuteman Computers						

REUTILIZATION OF THE MINUTEMAN GUIDANCE COMPUTER AS A NUMERICAL/PROCESS CONTROLLER

THESIS

GE/EE/73-5

Raymond V. Cicirelli, Lt, USCG
John M. Hill, 1/Lt, USAF

REUTILIZATION OF THE MINUTEMAN GUIDANCE COMPUTER AS A NUMERICAL/PROCESS CONTROLLER

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Raymond V. Cicirelli, B.S.
Lt USCG

John M. Hill, B.S.E.E.
1/Lt USAF

Graduate Electrical Engineering
March 1973

Preface

We, the authors, must agree with the words of Professor Gary B. Lamont, "A master's thesis is an educational experience." Innumerable bits and pieces of diverse information have been garnered in the accomplishment of this work. Hopefully some of these will provide the seeds for future maturation in the field of engineering.

Professor Lamont, who supplied the original idea for this thesis project and advised us throughout, must share our grateful appreciation with Lt. Joseph Theriault who rendered invaluable assistance in learning to use the DCTB computer.

As always, our wives and families are due a substantial amount of credit for their support during this trying period.

Raymond V. Cicirelli

John M. Hill

Contents

Contents

Contents

List of Figures

List of Figures

List of Tables

REUTILIZATION OF THE MINUTEMAN GUIDANCE COMPUTER AS A NUMERICAL/PROCESS CONTROLLER

I. Introduction

The D17B and the D37C guidance and control computers are integral components of the Minuteman I and II missiles, respectively, which form a part of the United States ICBM arsenal. The Minuteman III missiles, which use D37D computers, complete the 1000 missile deployment of this system. Due to the modernization of the Minuteman I system, the United States Air Force has declared approximately 1000 outdated inertial guidance systems unserviceable (Ref 1:1). Each of these surplus guidance systems contains a D17B computer, unclassified parts of the stable platform, and power supplies. These systems are available to colleges and other qualifying agencies for the cost of shipping. In addition, as the Minuteman III missiles replace the Minuteman II missiles, the D37C computers will become available as surplus digital computers. It is expected that some D37C computers will be available as early as the fall of 1973.

The initial cost of these computers ranges from about $139,000 (D37C) to $250,000 (D17B). Since a large number of these computers have been or will be declared excess, a substantial savings could be realized if these computers could be reused in other applications. This thesis will investigate the feasibility of using these computers in the areas of numerical control and process control. This chapter will cover a brief background description of numerical control and process control, the assumptions that were used, and the organization of the thesis.

Problem Analysis

The problem that exists is to reuse surplus avionics computers to effect a savings of the money originally invested. Official Air Force policy is to discourage the procurement of computers except in certain designated application areas. Air Force Logistics Command (AFLC) is interested in the possibility of using surplus computers which are being phased out along with their parent missile systems. As mentioned previously there are a number of D17B computers available for use in areas that would benefit from the specialized features of a guidance and control computer. A thesis (Ref 15) has been written on the general feasibility of using numerical control in Air Force depot-level shops and work centers. It also presents some of the special applications in the Air Force which are particularly suited for numerical control. There are several problem areas which should be considered if surplus minicomputers are to be reused.

Among these problems is the relatively small main memory capacity of the D17B computer. Its memory is approximately the same as the smallest available memories in current minicomputers. This imposes a limitation on total program lengths and computational capabilities. In addition, the memory access time is comparatively long since it has a serial access memory (disk) rather than a random access memory. Computations take longer than desired due to this feature and to the serial operation of the computer. Data are transferred into and out of the computer serially, again imposing time constraints. A full grasp of the peculiar characteristics of the instruction set available with this computer is necessary to implement its use to maximum efficiency. The

unique instruction set on this computer could prove to be a distinct advantage in this application.

An operational D17B computer is available at AFIT and will be used for experimentation and study during this investigation. A D37C computer recently became available, but it is not operational at this time. This requires that the study be based on the D17B computer and later be extended to the D37C computer.

A final problem is the acquisition or generation of a suitable system to demonstrate the control applications of the computer. Initial investigations determined that the Air Force has surplus numerical control machines which can be obtained through DIPEC (Defense Industrial Plant Equipment Center); however, the lead time necessary to obtain such equipment, four to five months, was prohibitive in this case. In addition, charges for transportation, packing, handling, and crating would amount to several hundred dollars. Another aspect to be considered is the condition of a machine which has been stored for quite some time.

With these considerations in mind, standard laboratory and educational devices such as analog computers or motor-control trainers would be quite suitable for demonstration of the control characteristics of the computer. The use of the analog computer requires the generation of a mathematical model of the system to be simulated. A two-axis linear positioning system based on numerical control parameters will be simulated. The simulation will be developed fully in a subsequent chapter.

Numerical Control

In the last 25 years many sophisticated new techniques have been developed in the machine tool industry for the purpose of increasing the

productivity and efficiency of machine shop processes. Numerical control is one of the most outstanding of these new techniques. Numerical control (NC) is the technique of controlling a machine or process utilizing command instructions in symbolic form which are converted into automatic servomechanism control.

The first major efforts to implement a numerical control system were sponsored jointly by the U.S. Air Force and the Parsons Corporation of Traverse City, Michigan, in 1949. The project was carried out in the Servo Mechanisms Laboratory of the Massachusetts Institute of Technology. The project was not completed until 1953. A functional block diagram of the system is shown in Fig. 1. A feasibility demonstration in March 1952 sparked further work by the Air Force and private industry (Ref 25:1). This demonstration was performed using a three-axis Cincinnati Hydrotel vertical mill, a machine which shapes metal using rotary cutters.

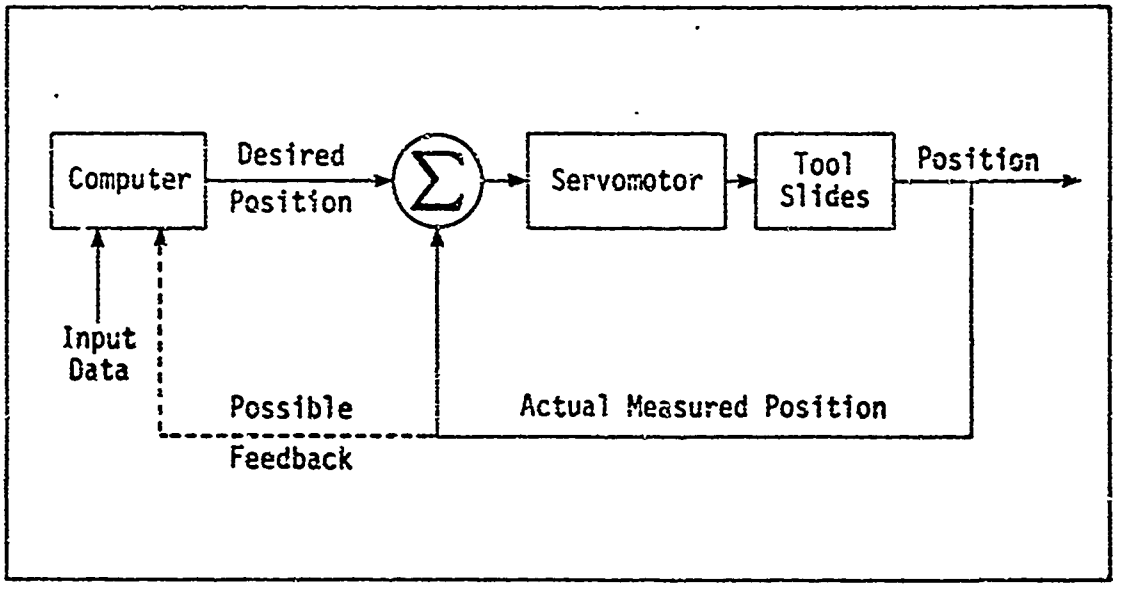


Fig. 1. Functional Block Diagram of Early NC System (Ref 25:4)

Early Applications. In the middle and late 1950's the aircraft industry was the primary user of numerical control processes. The most practical way for the aircraft industry to implement numerical control was to retrofit existing machines for numerical control. In the late 1950's, large numerical control machines were manufactured for applications such as skin milling, the removal of small layers of surface metal. At the same time multiple-tool machines were developed. In 1956 the automatic tool changer was introduced. A drill with punched tape-controlled table positioning was announced in October 1961. Because of the low price of the system, its introduction represented a major economic breakthrough. By the end of 1962 all major machine tool manufacturers were engaged in numerical control (Ref 11:13).

In the early stages of its development, numerical control required close coordination of such things as input codes and input formats. By 1958 the need for standards was critical. The Electronic Industries Association (EIA) led the efforts toward standardization. They developed a single standard (RS-244). Although this standard has been almost universally accepted in the recent past, the American Standard Code for Information Interchange (ASCII) will probably supplant it in the near future due to its more widespread acceptance in other fields, and its greater flexibility (Ref 11:13).

Early Programming Procedures. Manual programming, being the most straightforward way to obtain program tapes, was used extensively in the early days of numerical control. However, it suffers from computational complexities necessary to determine the appropriate commands required to direct the machine properly. Calculations such as surface intersections

were quite tedious to compute by hand. Early programmers needed a knowledge of analytical geometry, advanced algebra, trigonometry, and computer programming as well as machine tool operation, tooling, and machine practices (Ref 11:14).

Starting in 1956, several computer programming languages were developed which allowed the generation of complex command sequences through the input of simple statements and machining parameters. This advance was a critical step in promoting the development of continuous-contouring to its fullest potential. Continuous-contouring requires moving the cutting tool along a specified path rather than moving the tool from point-to-point without regard to the path. Most point-to-point (PTP) applications did not present such severe programming burdens, however (Ref 23:194).

Numerical control programs were the first attempts at creating a communication language between men and machines with the aid of a computer (Ref 11:15). The first language to be developed (Ref 26:193) was APT (Automatically Programmed Tools). APT was soon followed by AUTOPROMT (AUTOmatic PROgramming of Machine Tools) and then by ADAPT (Air Force Developed APT or ADaptation of APT) (Ref 26:193). The inception of the smaller minicomputers led to the development of AUTOSPOT (AUTOmatic System for POsitioning Tools), in 1962, for point-to-point applications.

The Present. Presently in use are huge, multi-processing machine tool centers which can perform many different machining operations, almost to the point of manufacturing a complete article (Ref 11:20). Two general methods, both of which eliminate the tape reader, have been developed to incorporate the use of a computer into a numerical control system. The

first method is computerized numerical control (CNC). This method involves the use of a controller (a device which converts coded commands into servo control signals) with limited information-storage capacity. Instead of feeding a punched tape directly into a numerical control machine, the program is stored in a centralized computer and fed to the controller in blocks. The controller then directs operation of the machine until it reaches the end of that particular block of instructions at which time the computer transfers in a new block of the program to the controller. The single computer can control several machines in this manner, obviously resulting in a more efficient utilization of the computer and eliminating the need for a tape reader for each machine.

The second method is DNC (Direct Numerical Control). In this method there is no tape reader or intermediate controller. Feedback and control lines go directly between the computer and the machine tool rather than to a controller. The advantages of this method are (1) costs are reduced by the absence of tape readers and controllers and by simplified electronic equipment, (2) the computer can be located remotely from the tool, (3) programs can more easily be adjusted in real-time in response to adaptive control, and (4) the computer can control more than one machine simultaneously or shift from one machine to another as the situation warrants.

Process Control

The digital computers used in process control are generally general-purpose computers similar to those used for business or scientific data processing. In addition they are provided with analog I/O systems and relatively sophisticated priority interrupts.

One of the first large-scale process control systems using a computer was at a Texaco refinery in 1954. Since that time hardware and software have changed considerably. Many larger systems now employ problem-oriented languages and on-line compilers and assemblers compared to the original efforts using machine language programming (Ref 27:84). Boiler-turbine-generators rated above 250 megawatts almost invariably use computer control for startup and shutdown (Ref 27:84).

Although the early use of computers in process control was in data logging, data analysis, and empirical model-building, the more recent attempts have been in closing the control loops through the computer itself (Ref 27:84). Even more recently the emphasis has been placed on direct digital control (DDC) (Ref 24:85). In this application the computer replaces a majority of the analog elements of the system and applies activating signals directly to the final control elements of a process.

Although analog process controllers can implement cascade, feed forward, and feedback control algorithms, they are generally not adaptive. In addition, it is difficult to implement some forms of nonlinear control and to integrate the controlling function with sequence control or optimization strategy. These functions can readily be implemented by digital computers.

In addition to the technical advantages of using computers, there are associated intangible benefits which sometimes accrue, such as more accurate process knowledge, greater safety, or better control of product quality. Better control of the product, coupled with a more accurate knowledge of the process, can result in more quality in the finished product, resulting in fewer complaints and fewer rejects. Conversely, tighter control of product quality can allow a lower average quality,

with a low reject rate, resulting in lower production costs.

Levels of Control. Figure 2 shows a division of process control into four separate levels. The first level indicates the more conventional type of control of such things as temperature, pressure, and flow. The second level includes schemes to compensate for process disturbances,

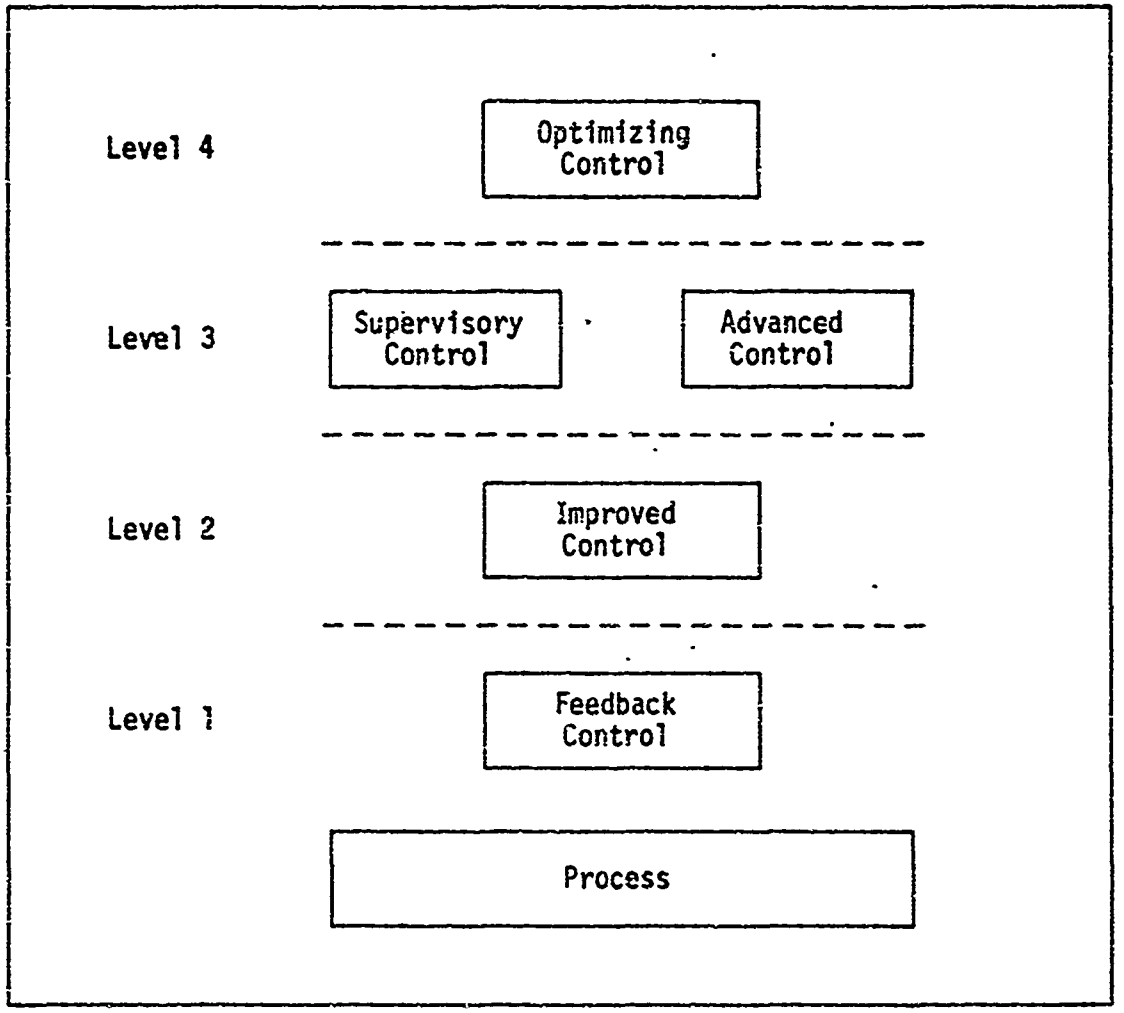


Fig. 2. Control Levels in Process Control

to eliminate the interaction of one control loop with another, and to implement various unconventional control schemes in order to improve performance by maintaining plant balances. The third level includes both

supervisory and advanced control. The supervisory function automatically changes conditions as warranted by examination of plant performance. Also included in this category are sequence control for batch operation as well as automatic startup and shutdown of continuous subsystems. The advanced control function includes adaptive control to enable automatic compensation for such things as aging catalysts in chemical reactions. The fourth level optimizes operating conditions in the plant using such factors as inventory prediction, economic scheduling, and dynamic programming. It should be noted that higher levels could be added to include the divisional and corporate level management information systems.

Figure 3 shows the relative complexity (based on memory requirement for each control level) and computer loading (based on the summation of the number of tasks times their frequency of occurrence). Computer loading increases rapidly at first but levels off later because of the

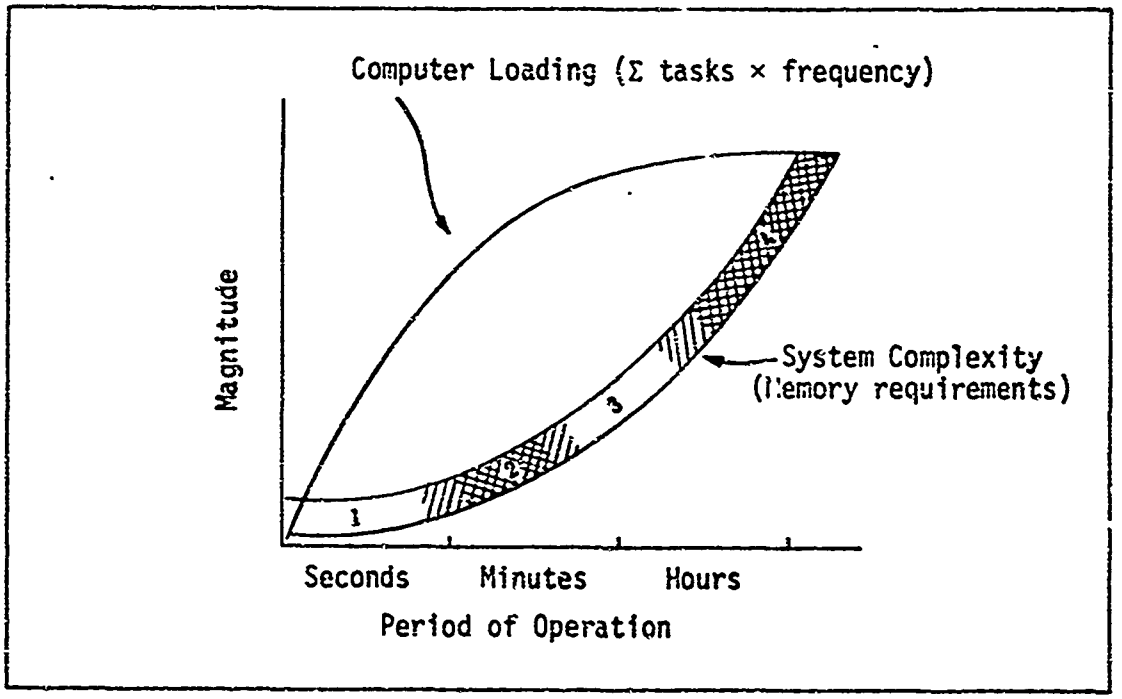


Fig. 3. Comparison of the Complexity and Loading Present in the Four Levels of Process Control (Ref 6:96)

decrease in frequency at higher levels. On the other hand, complexity increases rapidly with the control levels because of increased memory requirements.

Assumptions

This thesis is written with the assumption that the reader has a basic familiarity with computers and control engineering. It will be assumed that the D17B computer at AFIT will be available, in suitable operating condition, for experimentation during the period of the investigation. It will be further assumed that the control functions can be applied by the computer in a dedicated status, rather than being interrupted at random, to perform other tasks. Since the D37C has all of the features of the D17B plus others not available to the D17B, the final assumption will be that the operations of the D17B computer can be performed by the D37C. The additional capabilities of the D37C are discussed in Appendix A.

Thesis Outline

Chapter II of this report presents a physical description of the D17B computer with a detailed treatment of the inputs and outputs. Chapter III discusses the formulation of a model of a machine positioning system and implementation of the control system. Chapter IV presents the experimental results obtained, and Chapter V presents conclusions and recommendations for future investigations.

II. D17B Computer

To give the reader an understanding of the capabilities of the D17B computer this portion of the report will start with the physical description followed by the functional description which will include discussions of the control unit, the arithmetic unit, and the memory. Next, computer word format and programming will be described. Included in this section will be the instruction word format, full and split word formats, and the phases of operation. The next two subsections will describe the input and output functions including special features applicable to numerical control. The final subsection will describe control features usable for numerical control as well as certain specifications of the D17B which are applicable to numerical control.

The general specifications for the D17B computer are listed in Table I.

Physical Description

Size and Composition. The D17B is a guidance and control avionics computer built by Autonetics, a division of North American Rockwell, as part of the Inertial Guidance System (model NS-10Q) of the LGM 30/Minuteman ICBM Missile. The Inertial Guidance System also includes the associated stable platform and power supplies. The D17B computer occupies one half of a 12-sided, right-polygonal shell, as sketched in Fig. 4. This shell is 20 inches high, has a maximum radius of 29 inches, and is five inches in depth.

TABLE I

General Specifications of D17B Computer

Manufacturer	Autonetics
Year	1962
Type	Serial, synchronous
Number system	Binary, fixed point, sign plus 2's complement
Logic levels	False (0 volts), True (-10 volts), negative logic
Data word length	24 bits (full word) 11 bits (split word)
Instruction word length	24 bits
Number of instructions	39
Execution times	
Add	78.125 μsec
Multiply	1015.625 μsec
Divide	Software
Clock frequency	345.6 kHz
Addressing	Direct addressing Two-address (unflagged) Three-address (flagged)
Memory	Ferrous-oxide coated disk Non-destructive readout 2727 (24 bit) word capacity 78.125 μsec cycle time
Input/Output	48 digital lines (input) 26 specialized incremental inputs 28 digital lines (output) 12 analog lines (output) 3 pulse lines (output) 25,600 words/sec maximum I/O transfer rate
Physical characteristics	
Dimensions	20 in. high, 29 in. diameter, 5 in. deep
Power	28 VDC at 25 A
Circuits	DRL and DTL
Weight	62 pounds

(Adapted from Refs 4:2 and 1:4)

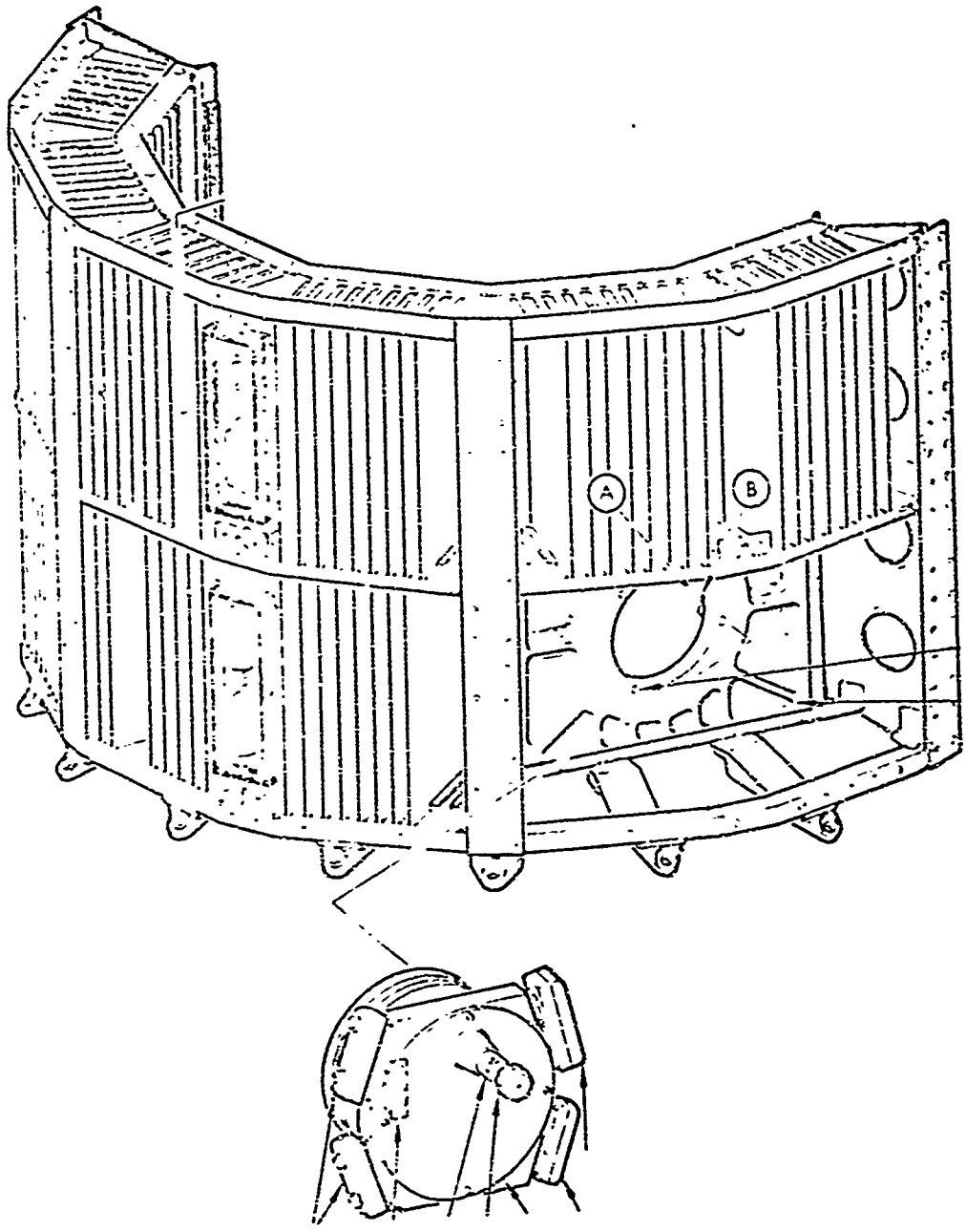


Fig. 4. Minuteman D17B Minicomputer Sketch (From Ref 4:26)

The power supply for the complete NS-10Q guidance system is contained in the other half of the shell and the stable platform is in the cavity formed by the computer and power supplies.

The D17B weighs approximately 62 pounds, contains 1521 transistors, 6282 diodes, 1116 capacitors, and 5024 resistors. These components are mounted on double copper-clad, engraved, gold-plated, glass fiber laminate circuit boards. There are 75 of these circuit boards and each one has been coated with a flexible polyurethane compound for moisture and vibration protection (Ref 18:16).

Since an airborne, computer-controlled missile only gets one chance to execute its mission, the design specifications of the D17B required very high reliability. This was achieved by using DRL (diode-resistor) logic extensively and only using DTL (diode-transistor) logic where gain was required in this fully solid-state computer. In the early 1960's when the D17B was designed, transistors were not as reliable as they are now, thus the designers used transistors only when necessary. The rotating disk memory, with non-destructive readout (NDRO), also enhanced the reliability of the computer. In actual, real-time situations involving Minuteman missiles, the mean time between failures (MTBF) was over 5.5 years.

Power Requirements. If the power supply included in the NS-10Q guidance system is used for the D17B, a 28 VDC regulated power supply capable of supplying 25 A is the only external power supply needed for operation of the computer. Other required voltages are obtained by converting 28 VDC into secondary power using solid state circuitry in the power supply section of the Inertial Guidance System. The current drawn from the 28 VDC power supply will vary from 0 to 25 A with a

steady-state value of 19 A (Ref 4:25). The computer may be operated without the associated power supply; however, it would then be necessary to supply 14 separate DC voltages as well as 1200 Hz and 400 Hz alternating current supplies. The secondary power requirements are shown in Fig. 5. Power consumption for the computer is approximately 350 watts (Refs 4:25 and 1:3).

Functional Description

The D17B computer is a general purpose, serial, binary minicomputer with the following general capabilities.

1. Sampling and processing of input data in the form of control signals, binary data, discrete signals, or incremental signals.
2. Logical decision-making, performance of arithmetic operations, and a logical AND operation using an instruction set of 39 machine language instructions.
3. Transmission of output data in the form of analog, binary, single character, or discrete signals under program control.

There are five basic functional divisions to the D17B: the Control Unit, Arithmetic Unit, Memory, Input, and Output. This functional division is shown in Fig. 6, with the Central Processing Unit (CPU) encompassing the Control Unit and Arithmetic Unit.

Terminology of the D17B. The terminology of the D17B consists of a number of basic terms which are common to the five functional divisions of the computer. To help clarify the descriptions of these divisions, the basic terms will be explained first.

Bit is the name for the amount of information that is contained in a number which can only take the value 0 or 1. The word "bit" is a contraction of "binary digit" (Ref 16:21). A word, consisting of 24 information

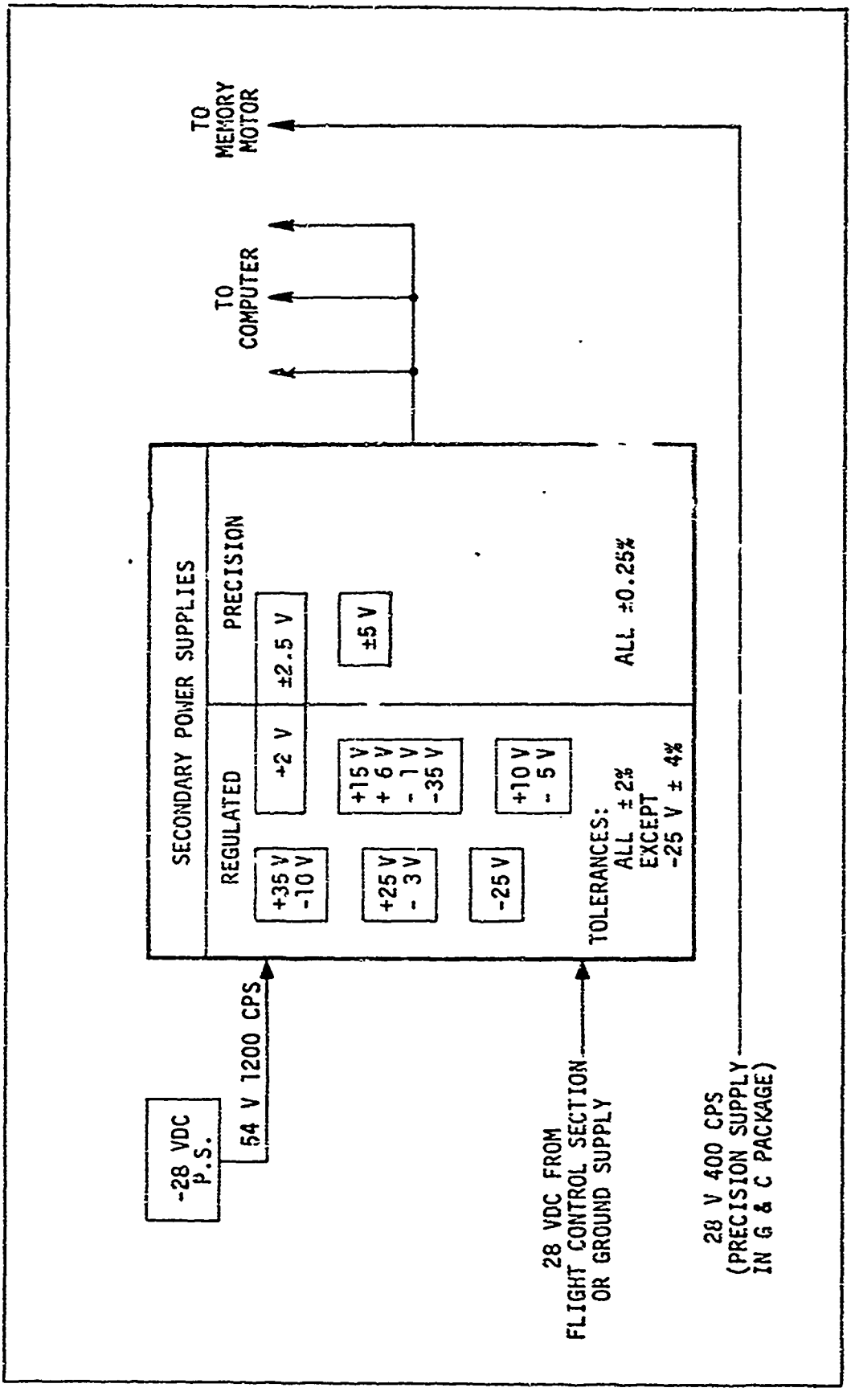


Fig. 5. Computer Power Flow

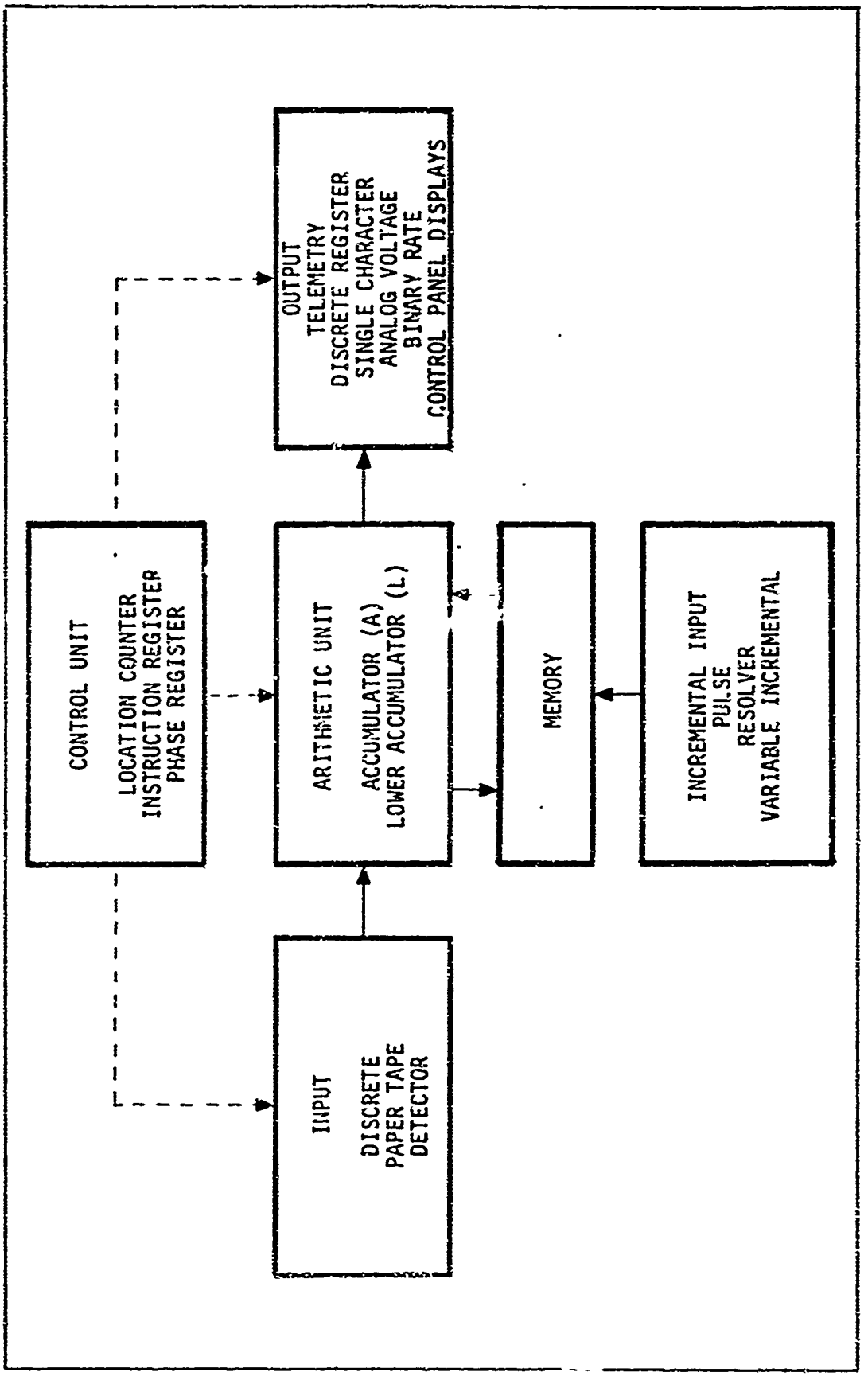


Fig. 6. Basic Functional Divisions of the D17B

bits and 3 timing bits, may be used as data (input or output), an instruction, or a control signal in the D17B.

A bit may be stored in a basic storage unit, a cell; the flip-flop is the most widely used type of cell in modern technology. A collection of cells joined together is called a register. Thus a 24-bit word can be stored in a 24-cell register (Ref 7:11). Most registers are constructed as collections of flip-flops to allow simultaneous or parallel access to all bits in the register. Since the D17B is a serial computer, simultaneous access to all the bits of a register is not necessary, thus flip-flops are not needed for the entire register. Registers or recirculating loops are composed of flip-flops and storage cells (on the magnetic disk memory) as illustrated in Fig. 7. This example of a recirculating loop is the A-loop used in the Arithmetic Unit. A_c, A_p, A_{23_w}, and A_x are flip-flops and the rest of the loop is on the memory disk. Information is read through the read flip-flop A_x, then fed to the input of the write amplifier and is rewritten on the next consecutive portion of the disk. This forms a closed loop with the information recirculating through this loop. The information first written on the disk remains on the disk as that portion of the disk moves out of the loop, then it is erased by the fixed erase heads (Ref 18:30).

The term word-time is derived from the length of time required to circulate the complete 27-bit word in a one-word loop, and is 78-1/8 μsec. A word-time can be divided into 27 bit-times since one bit is one twenty-seventh of a word (Ref 1:7).

The 24 information bits in a computer word may be subdivided into fields of various lengths. These fields may be decoded to indicate specific information such as operation codes, operand addresses, and

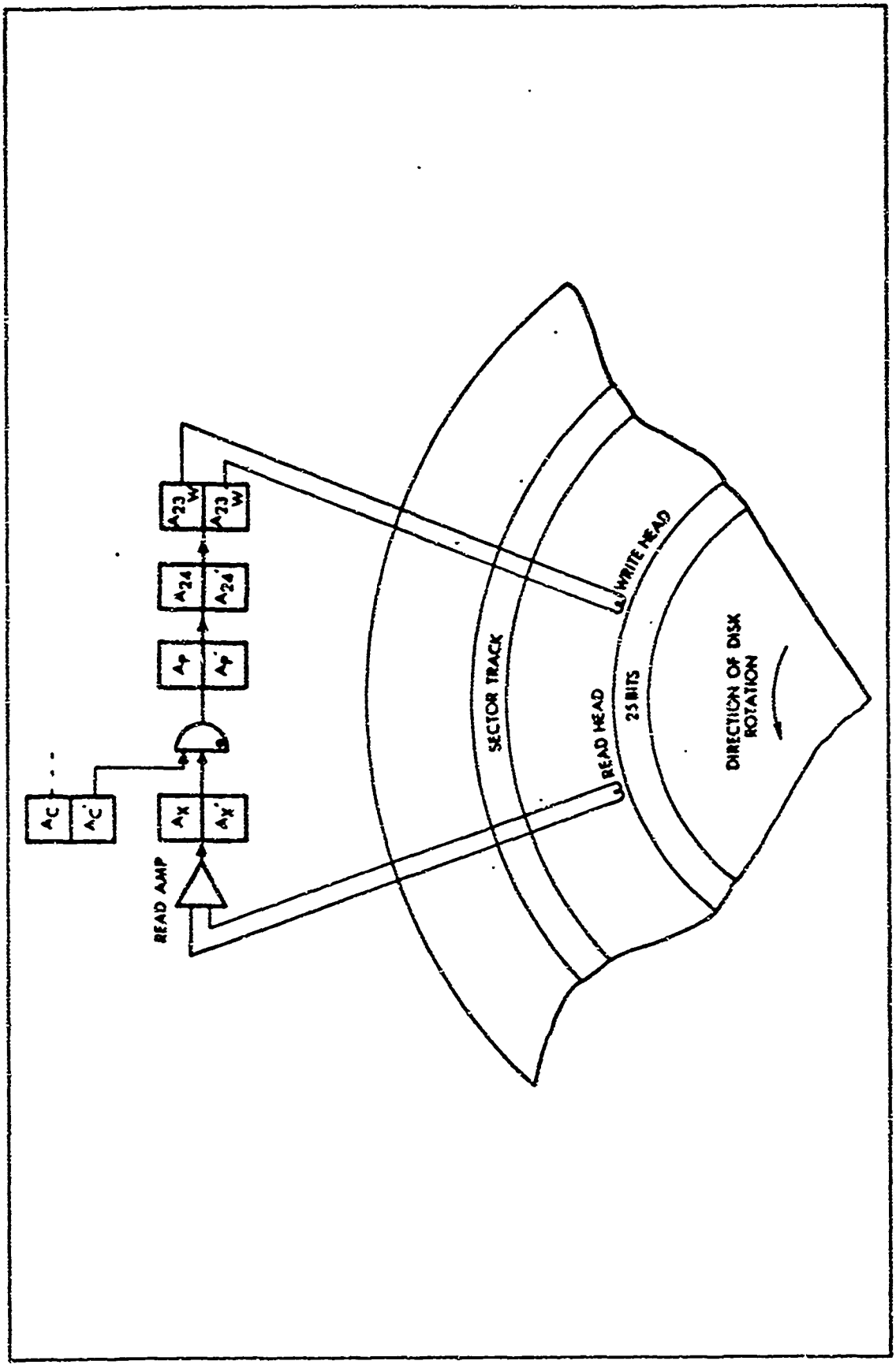


Fig. 7. Accumulator Recirculation Register

next instruction addresses. The various fields will be explained in detail in the Computer Word Format and Programming portion of this report.

Control Unit. The Control Unit processes and interprets all machine functions and controls two incremental inputs. Generally, in computers with sequential operations, the Control Unit goes through three cycles: the fetch cycle, the execute cycle, and the defer cycle. The defer cycle is used for indirect addressing which does not exist in the D17B. Computers with delay type memories like the D17B divide the fetch and execute cycles into phases. In the D17B the fetch cycle consists of the instruction search phase and the instruction read phase. The Control Unit must search for the instruction location as the disk is rotating and then transfer the instruction from memory to the Instruction Register or the I-loop. Then the execute cycle, which is sub-divided into the operand search phase, operand read phase, and the execute phase, is entered. During the execute cycle the Control Unit searches for the operand location as the disk is rotating and then transfers the operand to the Number Register or N-loop within the Control Unit. Then the instruction is executed.

The main component of the Control Unit is the Instruction Register I (ee Fig. 8). The Instruction Register or I-loop contains a 27-bit word which is composed of one delay flip-flop Ip, one write flip-flop I24w, one read flip-flop Ix, and 24 other bits written on the magnetic disk memory. New information may be entered into the I-loop when the control flip flop Ic is "one" set; otherwise, the information circulates from the magnetic disk through the flip-flops and is rewritten on the disk in a continuous loop (Ref 1:9).

Instructions are transferred from memory to the I-loop where the instruction is held for part of the interpretation process. The instruction

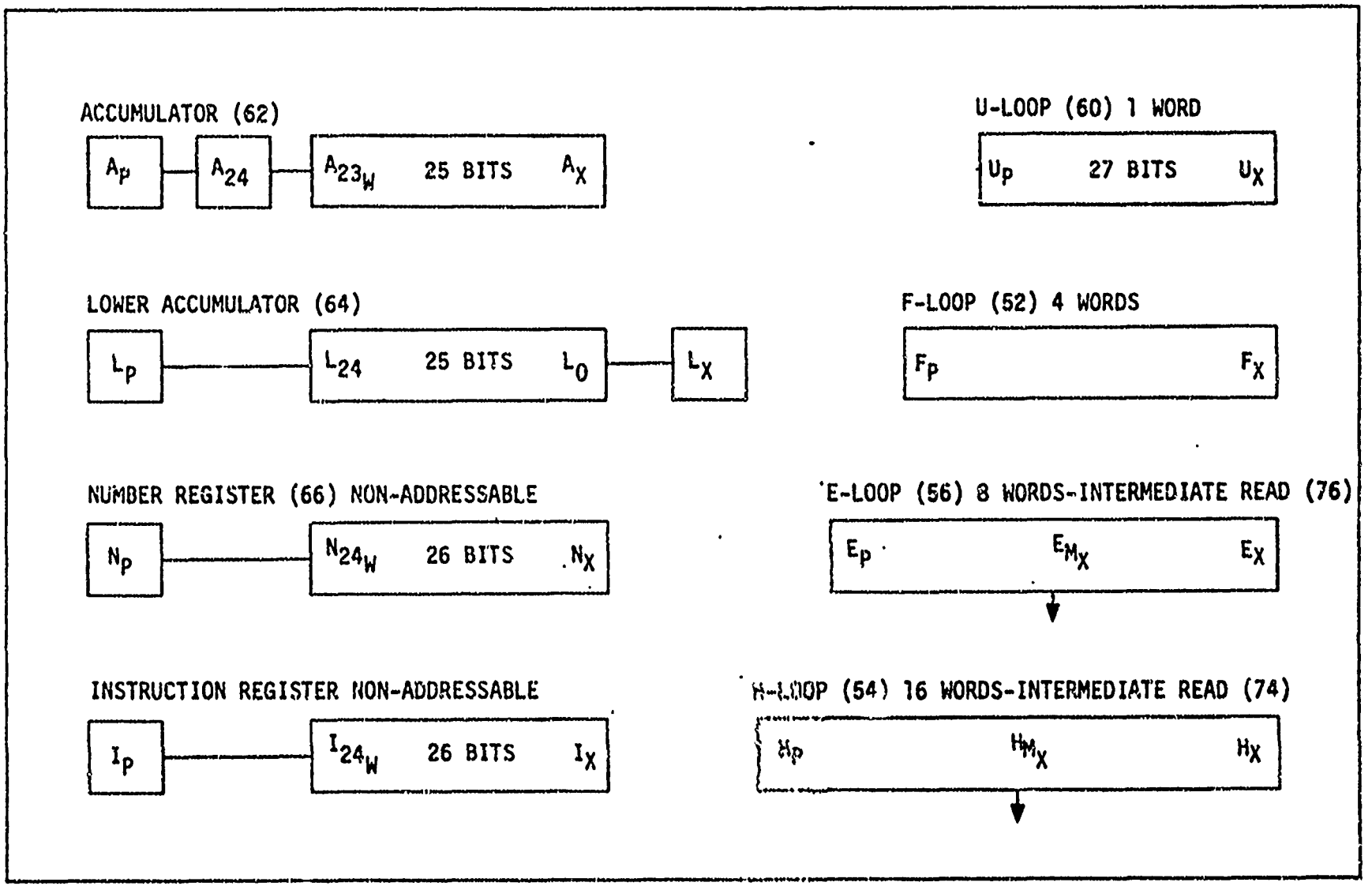


Fig. 8. Memory Registers and Loops

word is separated into fields such as the operation code, bits 21-24; and the operand channel information, bits 8-12. These bits of information are transferred to buffer registers for temporary storage and then to storage registers to be held until they are executed. Once the information is transferred to the buffer registers, the Instruction Register is free to receive the next instruction (Ref 2:16).

The operation code is transferred serially from the I-loop to the Operation Code Buffer Register (flip-flops Ip, Ob_3-Ob_1). This code is then parallel loaded into the Operation Code Storage Register (flip-flops O_3-O_1) where the operation code is held for, and during execution. In a similar manner the operand channel information is serially loaded into the Operand Channel Buffer Register (flip-flops Cb_5-Cb_1) and then parallel loaded into the Operand Channel Storage Register (flip-flops C_5-C_1). For register reference instructions which do not require operands, the channel storage flip-flops may be used as additional hardware to execute the instruction. Character output operation is an example of this application. Four bits of the accumulator are shifted into the Channel Storage Register, then output on the character output lines.

Flag storing is a special operation which allows the execution of a store operation with the initiation of other operations (such as add or multiply) during one word-time without requiring an additional instruction. Using the flag store capability saves one word of memory and one word-time because a store instruction does not have to be used. The previous contents of the Accumulator are stored in a channel specified by the code in T_{19}, T_{18}, and T_{17} which indicate bits 19, 18, and 17, respectively, in the instruction word as shown in Fig. 11, page 32. This

code is read into the Flag Code Storage Register at the first bit-time of execution. Flag storing will be explained in more detail under Computer Word and Programming.

When an operand is read from memory it is loaded into the Number Register or N-loop (see Fig. 8). The N-loop is in the Control Unit and consists of three flip-flops, N_p, N_{24_W}, N_x, and 24 bits of memory. The flip-flop N_p is used for delay, N_{24} for writing on the memory disk, and N_x for reading from the N-loop. A fourth flip-flop N_c controls the entry of new information into the N-loop (Ref 1:10).

The Phase Register consists of flip-flops P_1, P_2, and P_3. It is used to select one of four external positions for each of the three analog voltage outputs. The Phase Register can also modify the operand channel address of the multiply-modified and the split-word multiply-modified instructions. Further, the Phase Register has an input function as a selector switch for choosing one of two pairs of inputs to one of the incremental pulse-type input loops, V or R.

Other registers in the Control Unit are three voltage output registers of 8 bits each which are used as inputs to D/A converters, and a Discrete Output Register (D_5-D_1) which, together with a Discrete Output Matrix, controls the 28 Discrete Outputs. Also, there is a Binary Output Control Register which consists of three flip-flops, G_3, G_2, and G_1.

A bit counter that is controlled by the sector track of memory is used for timing control in the D17B computer. The bit counter is a set of 6 flip-flops that are used to distinguish bit-times for the serial operations of the computer. These flip-flops are designated B_1, B_2, B_3, B_4, B_5, and B_6. Three additional flip-flops, T_p, T_x, and T_0, are used with B_1-B_6 to form the logic used to count from 1 to 27 during one word-time.

B_1 is used to distinguish between even and odd bit-times and B_2 is "one" set at alternating two-word-time periods. B_3 is "one" set only during the right and left split-word bit-times. B_4 and B_5 are counting flip-flops that are used in conjunction with the other flip-flops of the bit counter. B_6 is "zero" set during the first half of the word-time and "one" set during the second half (Ref 19:23-24).

Summarizing, the major components of the Control Unit are: the Instruction Register; the Number Register; the Operand, Channel, and Flag Code Buffer and Storage Registers; the Phase Register; the Output Control Registers; and the Bit Counter.

Arithmetic Unit. The purpose of the Arithmetic Unit is to perform the calculations as directed by the Control Unit. There are two one-word registers in the Arithmetic Unit: the Accumulator and the Lower Accumulator.

The Accumulator, or A-register, accumulates the results of all the arithmetic functions and the one logical function. It also serves as an output register for voltage outputs, binary outputs, character outputs, and telemetry (Ref 18:17). The A-register as shown in Fig. 8 is composed of two delay flip-flops A_p and A_{24}, a write flip-flop A_{23_W}, a read flip-flop A_X, and 23 bits on the magnetic disk memory. When the control flip-flop A_C is "one" set, new information may be entered into the A-loop, but when A_C is "zero" set the loop recirculates and new information cannot be read in (Ref 18:30).

The Lower Accumulator or L-loop is used in conjunction with the Accumulator for certain arithmetic and logic operations. It is used in all the multiplication operations, Multiply, Split-Word Multiply, Multiply Modified, and Split-Word Multiply Modified. The multiplier which was in the Accumulator is shifted into the L-loop in reverse order (the LSB is

in the MSB position). The Logical AND to Accumulator operation logically AND's the contents of the Lower Accumulator to the Accumulator. Another function of the L-loop is to receive character inputs during loading operations and transfer these inputs to the Accumulator or the Instruction Register. The L-loop also serves as a rapid-access loop which allows data to be stored into or accessed from the L-loop in one word-time. The L-register, as shown in Fig. 8, consists of 23 bits in memory, two delay flip-flops L_X and L_D, one write flip-flop L_{24}, and a read flip-flop L_0 (Ref 18:31).

Memory. Memory in the D17B is a rotating magnetic disk using non-return-to-zero recording. The disk is driven at 6000 rpm by a 400 Hz, 3-phase hysteresis-synchronous motor. Information is transferred to the magnetic disk by 68 stationary read and write heads and remains on the disk until new data is recorded. This information is in non-volatile storage and remains stored even when power is removed from the computer. This is not true of the rapid-access loops which are partially on the disk. These loops are considered as volatile storage because the flip-flops that are a part of the loops or registers will be activated in a random state when power is restored.

The memory disk is divided into 32 concentric tracks (channels) and each channel is divided into 128 radial sectors as shown conceptually in Fig. 9. The 32 channels are numbered in even-octal progression from 00 to 72 plus the I-loop and the sector track which are not numbered. Only even numbers are used because the least significant bit of the octal number used for channel addressing is reserved for the sector address. The sector numbers are permanently recorded on a special sector track S, one number out of phase with the sector. This difference is used for

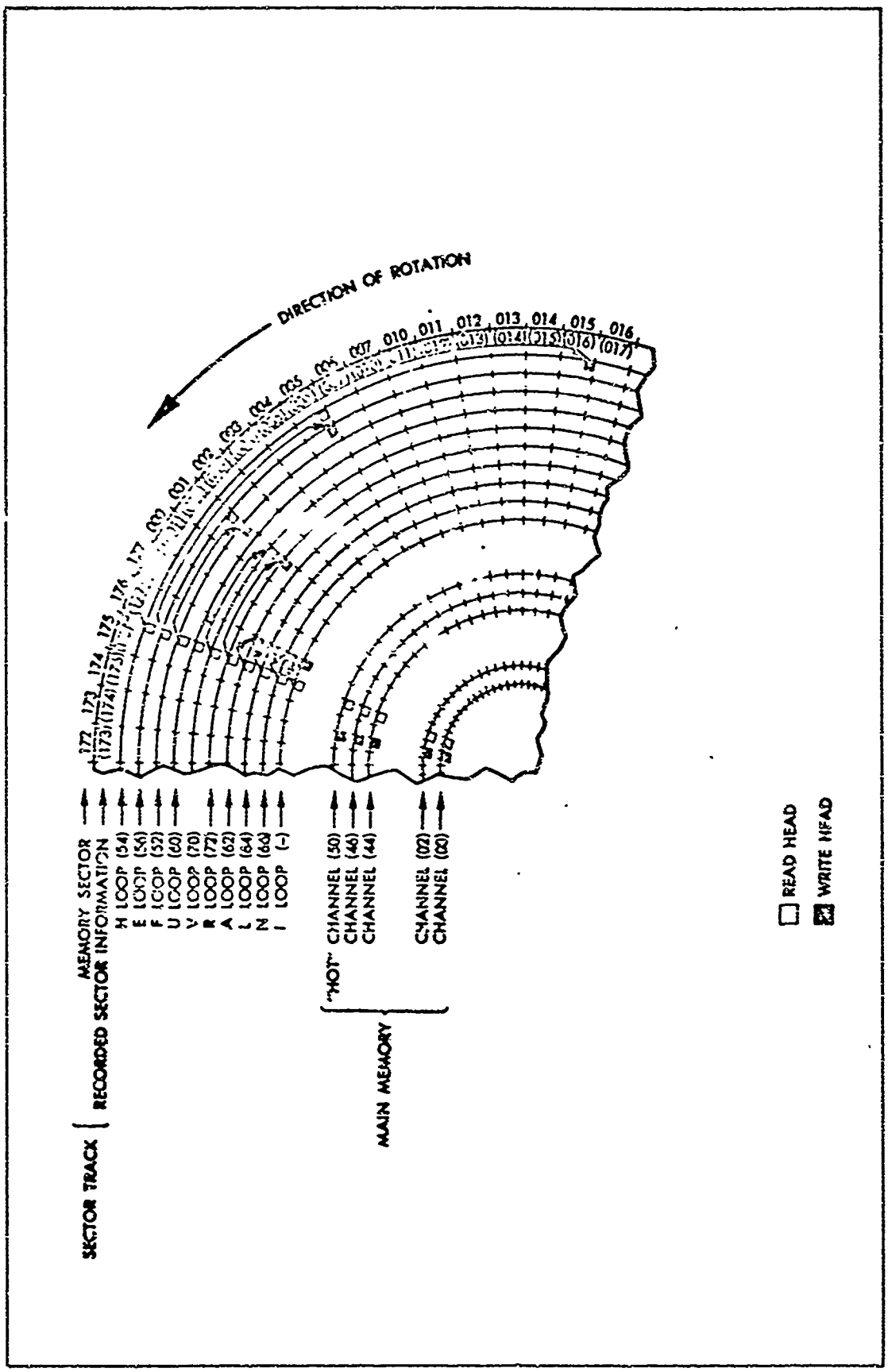
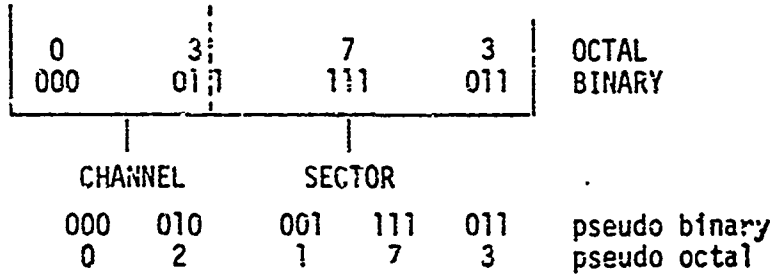


Fig. 9. D-17 Conceptual Memory Layout

timing purposes in the computer. The sectors are numbered octally from 000 to 177. Each address, 0373_8 for example, designates the channel and sector as shown below:



Thus when 0373_8 is decoded as channel 02_8 sector 173_8 it can be located in memory or in a conceptual layout of memory such as Fig. 9.

Program security or memory protection can be maintained by disabling the write heads to channels 00-46, the "cold-storage" channels, resulting in a read only memory. The various rapid-access loops and channel 50 (the "hot-storage" channel) could still be written on.

In addition to channels 00-50, there are ten recirculating loops which are used as input, arithmetic, and rapid storage operations as shown in Table II.

There are 2727 programmable words in memory (21 channels of 128 words plus 8 loops with 39 more words). The memory cycle time is 78-1/8 μsec if the memory location is coincident with a read head This is the time required to read one 24-bit serial word and is defined as one word-time. The cycle time for all one-word loops is one word-time or 78-1/8 μsec. The worst-case cycle times for the 4-, 8-, and 16-word loops are 4, 4, and 8 word-times, respectively. This is due to the intermediate read heads in the 8- and 16-word loops. The worst-case cycle time for the 21 main memory channels is 128 word-times or 10 msec (Ref 4:16).

TABLE II

Recirculation Loops in the D17B Computer

Channel	Loop	Words in Channel	Function
52	F	4	Rapid access loops
54	H	16	
56	E	8	
60	U	1	
62	A	1	One-word arithmetic and rapid-access loops
64	L	1	
66	N	1	
-	I	1	Instruction loop nonaddressable
70	V	4	Input buffer loops for incremental inputs
72	R	4	
74	H_m		Intermediate read heads*
76	E_m		

*H_m and E_m are not separate loops--they are part of the H and E loops, respectively.

Inputs and Outputs. The Inputs and Outputs of the D17B Computer will be covered separately under Inputs and Outputs because of its importance to numerical control applications.

Computer Word Format and Programming

Word size in this computer is 27 bits, but 3 bits are used for timing, resulting in a 24-bit, programmable word. This 24-bit word is presented in three basic formats: whole number, split number, and instruction. These formats are shown in Fig. 10. The two forms of the instruction word format--unflagged and flagged instructions--are shown in Fig. 11.

Unflagged Instruction. The unflagged instruction will be discussed first since it is used as a basis for the flagged instruction. The unflagged instruction is identified by the flag bit T_{20} set to zero as illustrated in Fig. 11. The unflagged instruction has five fields as shown in Table III.

TABLE III
Unflagged Instruction Fields

Bit Position	Field
$T_{24} - T_{21}$	Operation Code
T_{20}	Flag (always zero)
$T_{19} - T_{13}$	Sector of next instruction
$T_{12} - T_1$	Operand address:
$T_{12} - T_8$	Operand channel
$T_7 - T_1$	Operand sector

INSTRUCTION

T_P	T_{24} T_{23} T_{22} T_{21}	T_{20}	T_{19} T_{18} T_{17} T_{16} T_{15} T_{14} T_{13}	T_{12} T_{11} T_{10} T_9 T_8	T_7 T_6 T_5 T_4 T_3 T_2 T_1	T_0	T_X
X	OPERATION	F_{LAG}	SECTOR OF NEXT INSTR	OPERAND CHANNEL	OPERAND SECTOR	X	X

FULL NUMBER

T_P	T_{24}		T_0	T_X
X	SIGN	23 BIT BINARY FRACTION	X	X

SPLIT NUMBER

T_P	T_{24}			T_{11}		T_0	T_P
X	SIGN	10 BIT BINARY FRACTION	X X	SIGN	10 BIT BINARY FRACTION	X	X

Fig. 10. Word Formats

UNFLAGGED INSTRUCTION ($T_{20} = 0$)

T_p	T_{24} T_{23} T_{22} T_{21}	T_{20}	T_{19} T_{18} T_{17} T_{16} T_{15} T_{14} T_{13} T_{12} T_{11} T_{10} T_9 T_8	T_7 T_6 T_5 T_4 T_3 T_2 T_1	T_0 T_X
X	OPERATION CODE Op	FLAG F	NEXT INSTRUCTION SECTOR ADDRESS Sp	CHANNEL NUMBER C	SECTOR NUMBER S	X X

FLAGGED INSTRUCTION ($T_{20} = 1$)

T_p	T_{24} T_{23} T_{22} T_{21}	T_{20}	T_{19} T_{18} T_{17} T_{16} T_{15} T_{14} T_{13} T_{12} T_{11} T_{10} T_9 T_8	T_7 T_6 T_5 T_4 T_3 T_2 T_1	T_0 T_X
X	OPERATION CODE	FLAG	FLAG STORAGE LOCATION S_f	SECTOR OF NEXT INSTRUCTION S_p	CHANNEL NUMBER	SECTOR NUMBER	X X

Fig. 11. Instruction Word Format

This format is commonly called a two-address instruction although it could be called a 1-1/2 address instruction. The two addresses given in the instruction are the next instruction address and the operand address. Only half the address for the next instruction is explicitly shown; the channel is assumed to be the same channel that contains the present instruction. The unflagged instruction format may be used with all 39 instructions listed in Table B-I, Appendix B.

Flagged Instruction. The flagged instruction format is used when the programmer wants to simultaneously store the previous contents of the accumulator into a specified loop and execute another instruction, such as an ADD, MULTIPLY, or STORE. The six fields of this instruction are shown in Table IV.

TABLE IV

Flagged Instruction Fields

Bit Position	Field
$T_{24} - T_{21}$	Operation Code
T_{20}	Flag (always one)
$T_{19} - T_{17}$	Flag storage location
$T_{16} - T_{13}$	Sector of next instruction
$T_{12} - T_{1}$	Operand address:
$T_{12} - T_{6}$	Operand channel
$T_{7} - T_{1}$	Operand sector

The flag-store location refers to a 3-bit code as shown in Table V. The address of the next instruction S_p is shortened to four bits since the flag-store location code uses three bits. This limits S_p to the 16 sectors following the operand address specified in the flagged instruction. The flagged instruction is considered a three-address instruction.

TABLE V
Flag-Store Location Code

Instruction Word Bits T_{19}	T_{18}	T_{17}	Loop	Channel	Description
0	0	0	-	-	Idle
0	0	1	F	52	4-word loop
0	1	0	T	-	Telemetry
0	1	1	-	50	Hot channel
1	0	0	E	56	8-word loop
1	0	1	L	64	1-word loop
1	1	0	H	54	16-word loop
1	1	1	U	60	1-word loop

Instruction Word Format. In both the unflagged and the flagged instruction word format the operand address (channel and sector) is 12 bits, which allows direct addressing of 4096 words. Since the D17B only has a 2727 word addressable memory, all locations in memory can be addressed directly. In the unflagged instruction the next instruction address is seven bits which allows every sector (177 octal) within the channel to be addressed. A transfer instruction is needed to transfer from an instruction in one channel to an instruction in a different channel.

The operation code field in both cases is 4 bits long, which limits the D17B to 16 unique 4-bit op codes. The 13 instructions which address memory, use these 4-bit op codes and 12-bit operand addresses. Two of the remaining 4-bit op codes are used for register reference instructions such as control, logic, I/O, and shifts. The channel portion (T_{12}-T_8) of the operand address is used as an extension of the operation code in instructions which do not access memory. Op code 14 is not used, which would allow another memory reference instruction. Also, there are numerous, unused 5-bit op code extensions which could expand the instruction repertoire (Ref 4:14).

Full Word Operand. All 24 bits may be used to store one operand. Bit 2^4 is the sign bit and T_{23}-T_1 represent a 23-bit fraction in two's complement form (see Fig. 12). T_p, T_0, and T_x are timing bits used by the computer and they are not programmable. Examples of the number representation in the D17B are given in Table VI.

Split-Word Operand. Two numbers may be stored in one 24-bit word. The left half-word is formed by bits T_{24}-T_{14} and the right half-word is formed by bits T_{11}-T_1. As shown in Fig. 12, bits T_{12} and T_{13} are not used, and bits T_{24} and T_{11} represent the sign bits. Examples of split-word operands are given in Table VI.

The D17B has the capability of simultaneous execution of two identical add, subtract, or multiply instructions on the left and right half-words. This increases the speed available for a solution to a problem but with a loss in accuracy because only 11 bits are processed for each half-word rather than 24.

WHOLE NUMBER

T_p	T_{24}	T_{23}	T_{22}	T_{21}	T_{20}	T_{19}	T_{18}	T_{17}	T_{16}	T_{15}	T_{14}	T_{13}	T_{12}	T_{11}	T_{10}	T_9	T_8	T_7	T_6	T_5	T_4	T_3	T_2	T_1	T_0	T_X
X	SIGN	23 BINARY BIT FRACTION (2'S COMPLEMENT)																						X	X

SPLIT NUMBER

T_p	T_{24}	T_{23}	T_{22}	T_{21}	T_{20}	T_{19}	T_{18}	T_{17}	T_{16}	T_{15}	T_{14}	T_{13}	T_{12}	T_{11}	T_{10}	T_9	T_8	T_7	T_6	T_5	T_4	T_3	T_2	T_1	T_0	T_X
X	SIGN	10 BINARY BIT FRACTION										NOT USED		SIGN	10 BINARY BIT FRACTION										X	X

Fig. 12. Number Word Formats

TABLE VI

Examples of Number Representation Used in the D17B Computer
(Octal numbers are presented for convenience;
the computer uses binary numbers)

Type of Number	Type of Format		
	Full-Word Format	Split-Word Format	
		Left Half-Word	Right Half-Word
Maximum positive number	3777 7777	3776	1777
Maximum negative number	4000 0000	4000	2000
Minimum positive number	0000 0000	0000	0000
Minimum negative number	7777 7777	7776	3777

Phases of Operation. This computer has five phases of operation which are common to delay-type memories. These phases are instruction search (IS), instruction read (IR), operand search (OS), operand read (OR), and execute (EX) as noted earlier. The upper part of Fig. 13 illustrates how these phases would be performed in normal sequential operation. The lower part of the figure illustrates how the D17B computer can perform several of these phases simultaneously. This figure assumes minimal delay coding of instructions which require one word-time for execution. The advantage of a minimal dealy code program is that, once the program is initiated, the effective completion time of any instruction is equal to the basic execution time of that particular instruction. Minimal delay coding means placing the next instruction in the memory location which will be read next, following the execution of the present instruction.

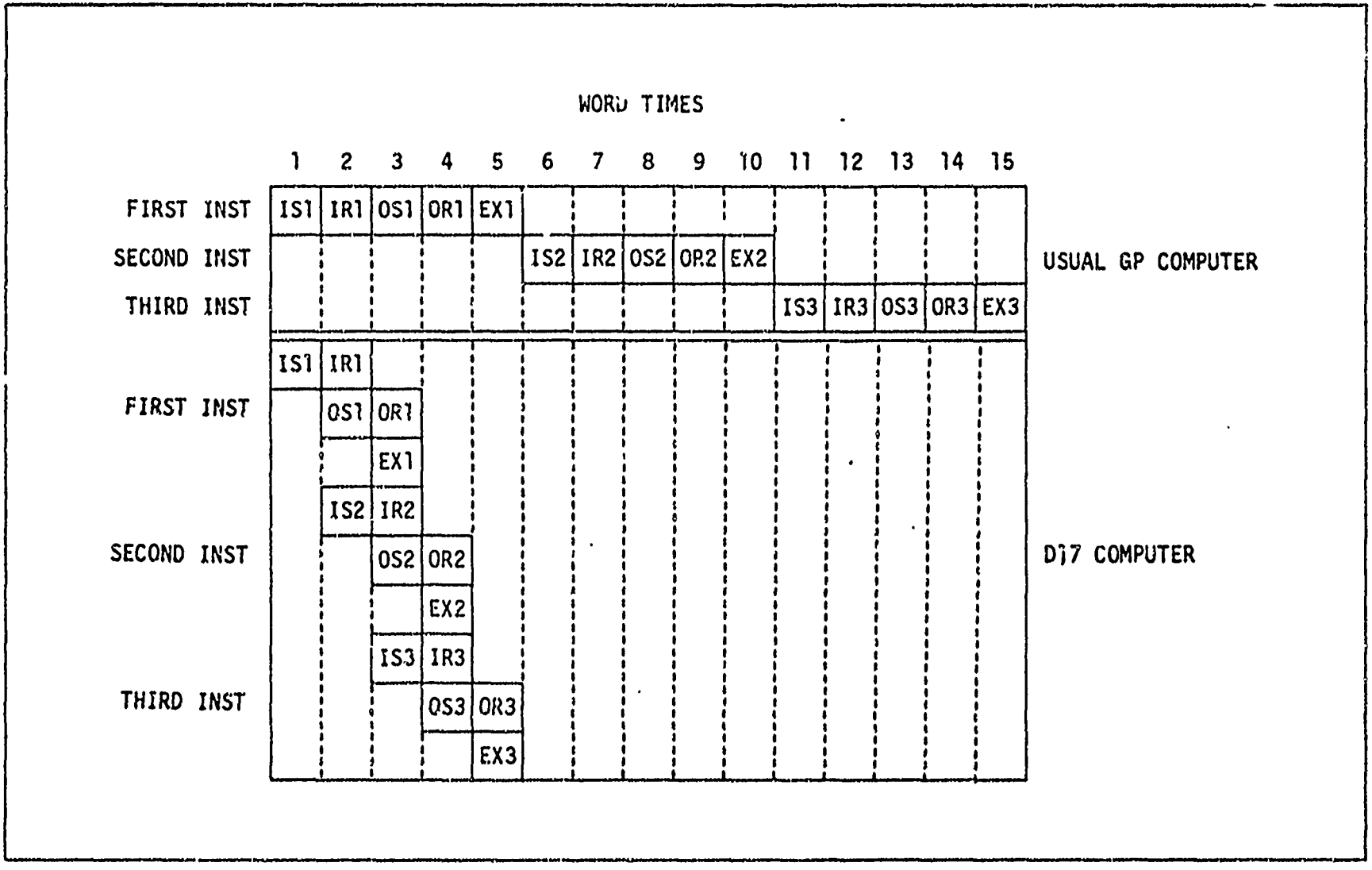


Fig. 13. Access Timing

Modes of Operation. The modes of operation can be considered as the types of operation that the computer may perform. The D17B has two basic modes, compute and non-compute. Compute mode involves the execution of instructions and the non-compute mode involves operations such as synchronization and reading instructions (Ref 1:59). These modes are divided into submodes as indicated in the Veitch Diagrams (Figs. 14 and 15). Reference 1 presents the state description and the associated register transfer equations for the modes and submodes if a more detailed presentation is desired.

Inputs

Because the Inputs and Outputs of the D17B were designed for use in the Minuteman I ICBM, many of the Inputs and Outputs are special purpose signals. Another factor to consider is that the D17B provides access to approximately 550 lines through connectors J_1-J_{11}. These lines include the Input/Output signals, control signals, power monitoring signals, and some spare lines.

The input lines as illustrated in Fig. 16 can be divided into four classes of inputs: Discrete inputs, Character inputs, Incremental inputs, and Control inputs. The Discrete and Character inputs are used for data inputs, whereas the Control inputs are usually signals from a control console. The Incremental inputs are independent of program control and are used in highly specialized applications.

Control Inputs. The Control inputs consist of eight signals: Master Reset (MR), Run (KR), Halt (KH), Single-step (KS), Initiate Load (FS), Halt Prime (IM), Enable Write (EW), and Disable Discretes (DD).

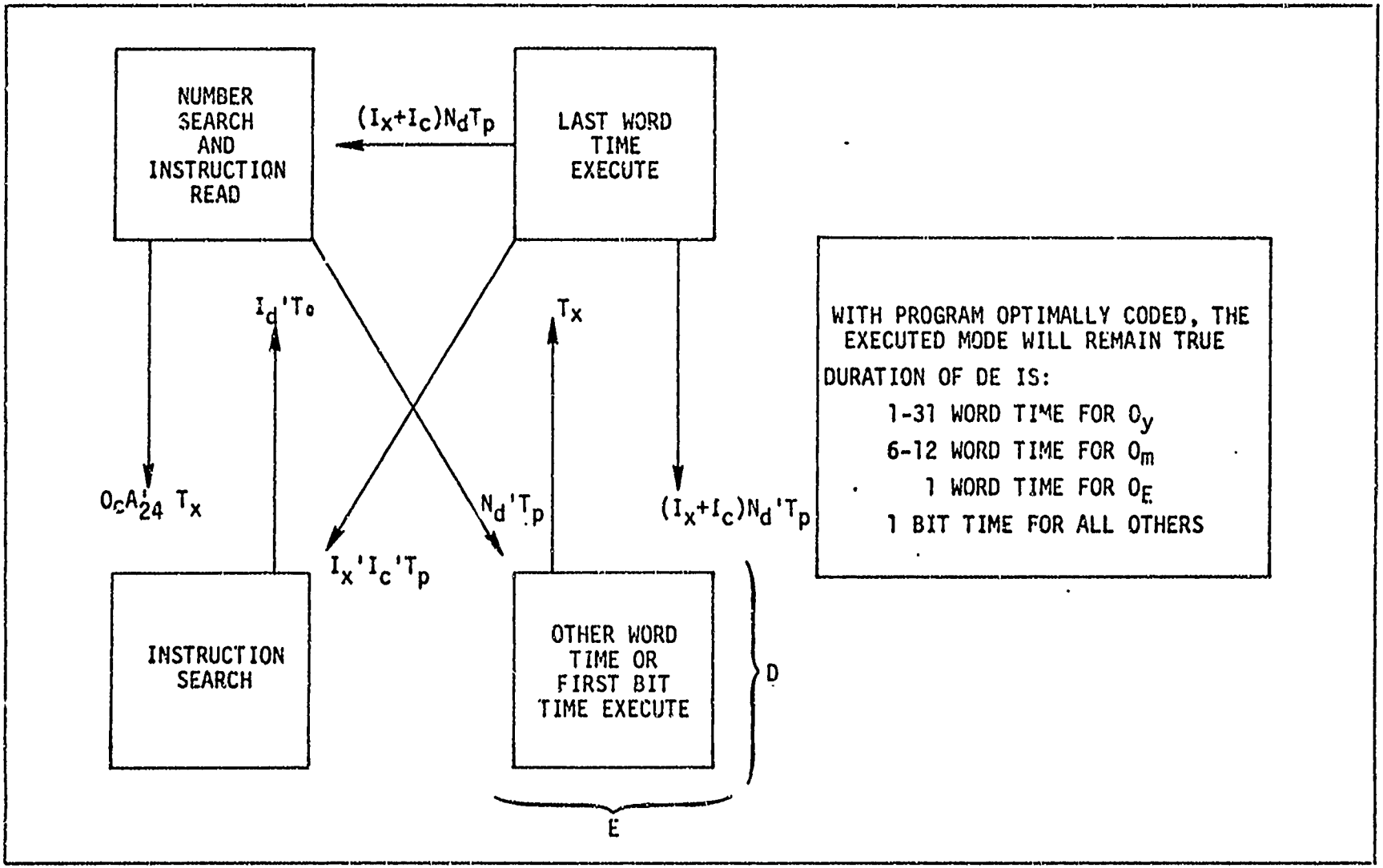


Fig. 14. Compute Mode Control (K)

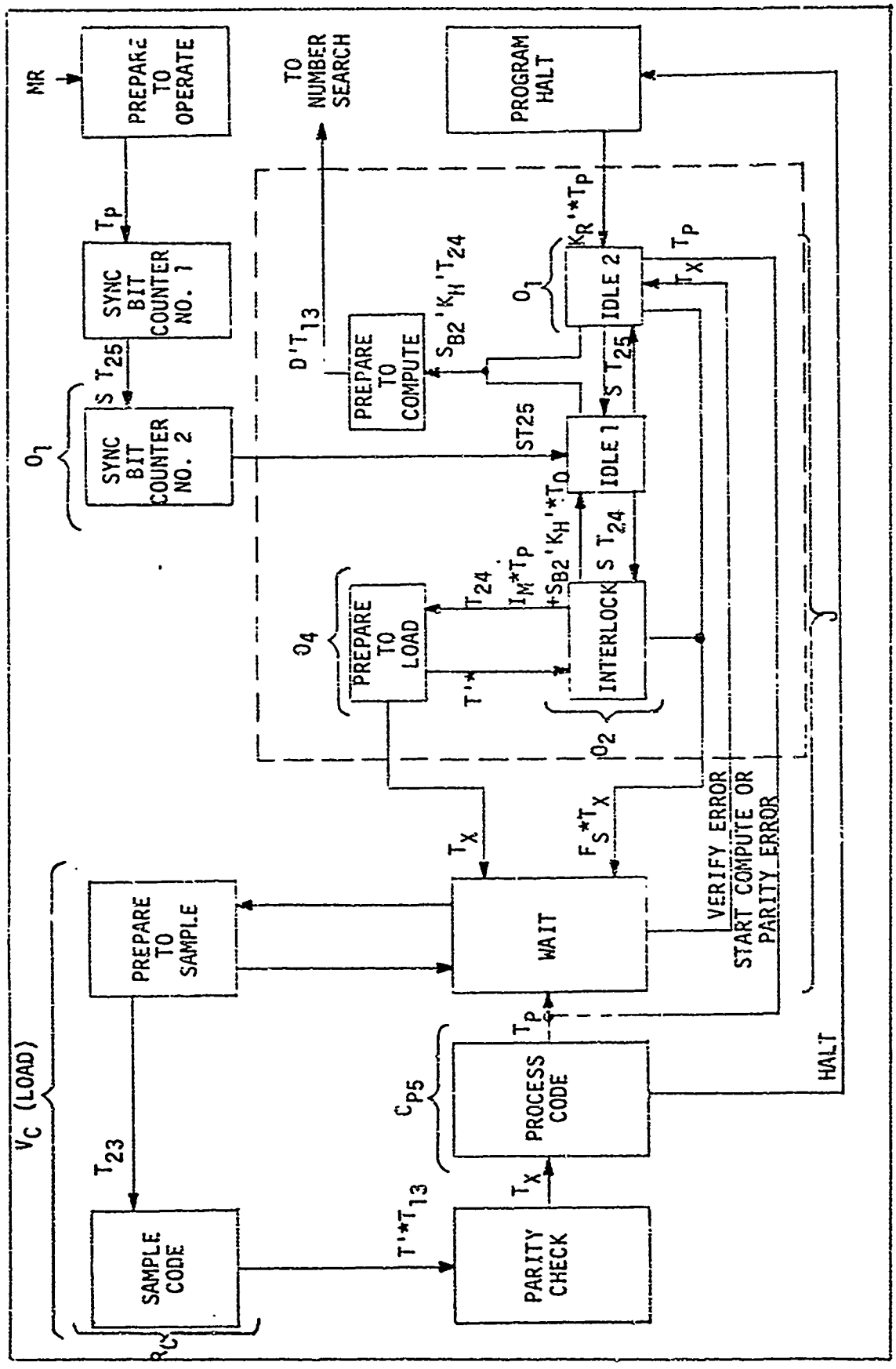


Fig. 15. Noncompute Mode Veitch Diagram

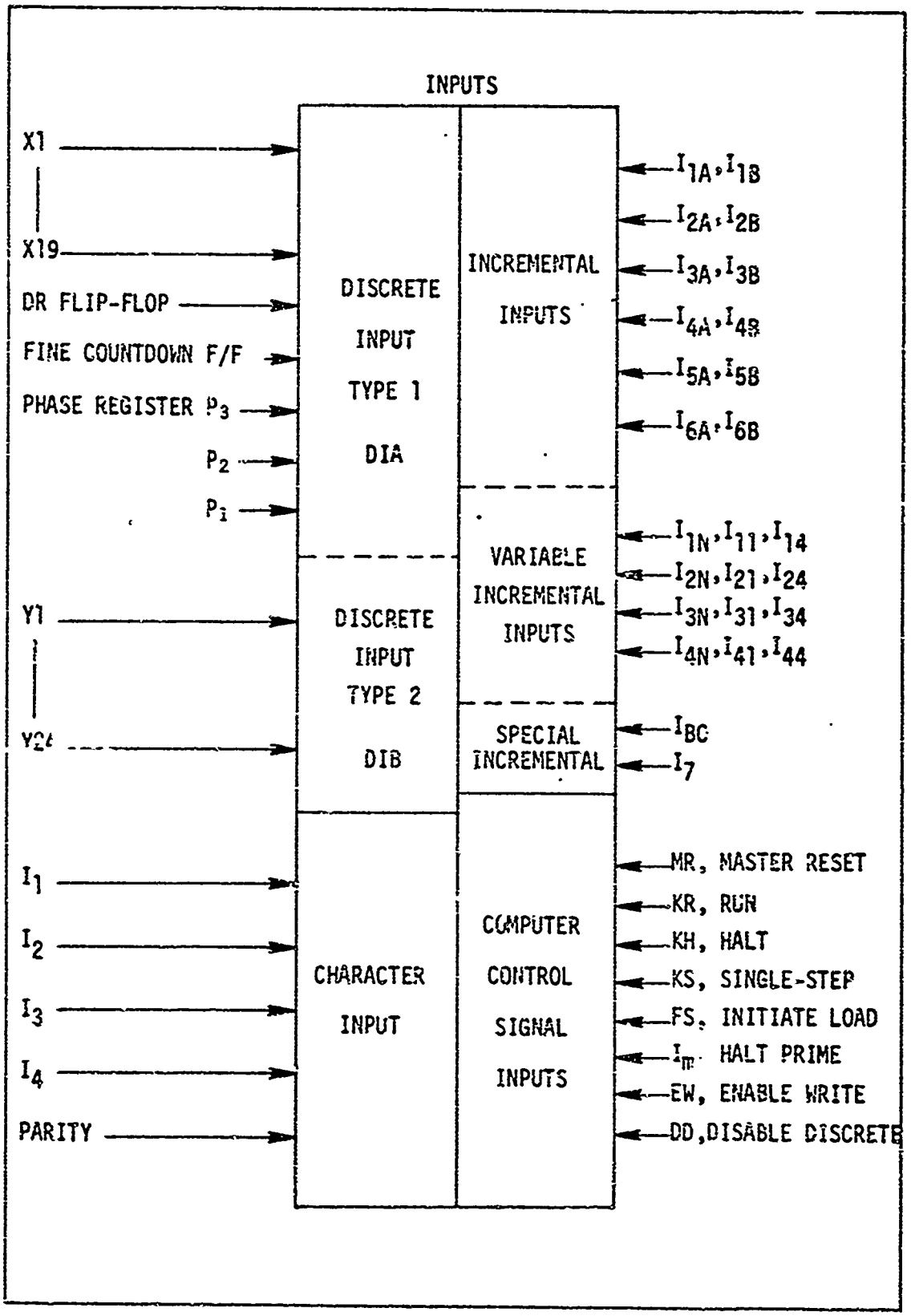


Fig. 16. D178 Computer Inputs

The Master Reset (MR) control signal primarily initializes all the major mode generators into a known mode. The flip-flops are also set in a random state during MR, thus any information in the rapid-access loops is lost. MR also causes a transition into Sync Bit Counter 1 mode and then into Sync Bit Counter 2 mode. Finally, an unconditional transfer to channel 00 sector 000, instruction 5000 000_8, is placed into the Instruction Register. This process takes approximately 30 msec, and the processor will be in the Prepare to Compute submode unless the Halt control signal had been issued. If KH (Halt) had been issued the processor would be in the Interlock mode and would remain in this mode until another control signal was issued.

The Run and Halt control signals are mutually exclusive and usually are generated from the same switch on a control console. The Run control signal is on for program execution and may be switched off to halt the computer during program execution. Switching KR (Run) off is equivalent to switching KH (Halt) on, so either terminology may be used.

KS, the Single-step control signal, is used to execute the instruction presently in the Instruction register and then return to the Manual Halt mode after the next instruction has been read into the I-register. To use KS, the Halt control signal must be on, or equivalently, KR off.

The Initiate Load (FS) control signal is used to sequence the processor into the Wait submode from the Interlock or Idle 2 submodes in those cases where the transition would not otherwise occur (Ref 29) This transition is necessary if the computer is going to execute any fill/verify operations.

If IM, Halt Prime, is on, the processor will cycle between Idle and Interlock submodes until IM is switched off or FS is switched on.

Then the computer will proceed with fill/verify operations (Ref 29).

The Enable Write control signal serves as a memory protect device for channels 00 through 46. When EW is on, these channels may be written into, but when EW is off, writing in channels 00 through 46 is disabled, although reading is still possible. The main advantage of the EW signal is to allow the computer to be turned on or off without affecting the memory. If the Enable Write control signal is on when the computer loses power, spurious signals will be written into channels 00 through 46 when power is restored, thus changing any program which may have been in those channels. During the start-up or when power is restored, the write flip-flops are enabled in random states sometimes causing unwanted signals to be written into a channel.

The final control signal is the Disable Discretes (DD) control signal. The DD signal has three functions, although the main function is to disable the discrete outputs. When DD=0, the DOA instruction changes the D-register, but all outputs are inhibited. DD=0 also inhibits writing into channel 50, although reading is still possible. This serves as a protection against spurious writing during shut-down and start-up procedures for channel 50. The third function of the Discrete Disable control signal is to disable terminal 1 of the voltage outputs. Thus, when DD=0, the outputs V_{10}, V_{20}, and V_{30} are disabled; the other voltage outputs are unaffected.

Character Inputs. The Character Inputs are used to load the memory during a cold start and also to cause the computer to transition into the proper submode to load or verify incoming octal characters. The Character Inputs are a set of five lines, I_1-I_5, where I_1 is the least significant

bit and I_5 is a parity bit as shown in Fig. 17 The four signals, I_1-I_4, comprise the load codes; all 16 combinations are used.

The Fill, Enter, and Location load codes are used most often since they are the codes involved in entering programs and data. Fill is used to sequence the computer into the Fill mode so that octal digits and other codes will be decoded by the computer. An example will probably clarify the operation of these load codes. First the Fill code is entered to ensure the Computer is ready. Next, an octal address (0000 4050) is entered, followed by the Location code. The Location code would cause 0000 4050 to be loaded into the Instruction register indicating the address in memory. Then a data word or an instruction is entered in octal, 70706161, followed by the Enter code. The Enter code causes 70706161_8 to be written into memory at channel 40, sector 50 (4050) and the Instruction register is incremented by one to $0000\ 4051_8$. Thus, if data or an instruction were to be entered in the next location in memory (4051) then the octal digits would be entered followed by the Enter code. It is not necessary to enter the location again since the I-register is incremented each time an enter code is processed. This process would be continued until the complete program had been entered.

When octal digits are entered they are shifted into the Lower Accumulator three binary digits at a time into the three low-order bit positions. At the same time the three high-order bits are shifted out and lost. The Location code transfers the octal digits to the Instruction register, where the Enter code transfers the octal digits to the Accumulator and then to memory. In both cases the octal digits remain in the Lower Accumulator.

		PARITY BIT	CHARACTERS			
		15	14	13	12	11
OCTAL DIGITS	0	1	0	0	0	0
	1	0	0	0	0	1
	2	0	0	0	1	0
	3	1	0	0	1	1
	4	0	0	1	0	0
	5	1	0	1	0	1
	6	1	0	1	1	0
	7	0	0	1	1	1
	HALT	0	1	0	0	0
	LOCATION	1	1	0	0	1
	FILL	1	1	0	1	0
	VERIFY	0	1	0	1	1
	COMPUTE	1	1	1	0	0
	ENTER	0	1	1	0	1
	CLEAR	0	1	1	1	0
	DELETE	1	1	1	1	1

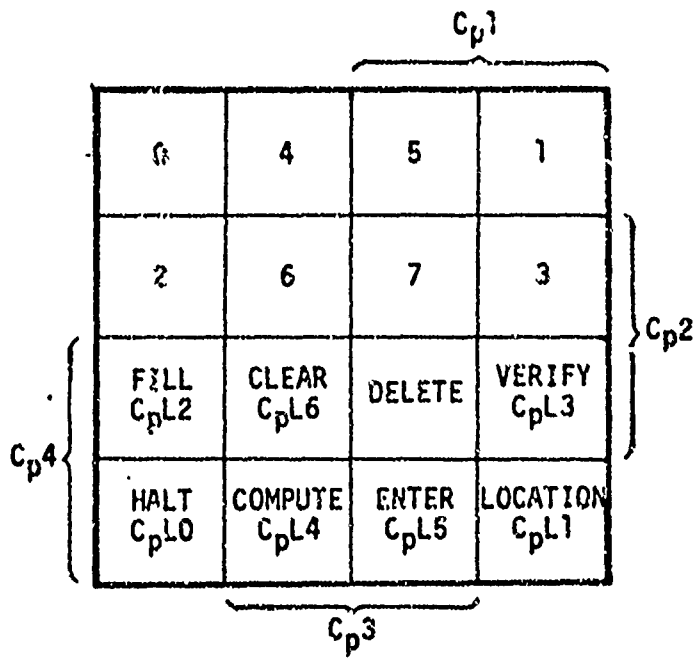


Fig. 17. Load Codes

The other load codes are used less frequently. Verify is similar to Enter except the octal digits are not entered into memory, but are compared with the octal digits in memory at the specified location and if they are not the same an error is indicated. Clear simply clears or zeroes the Lower Accumulator. The Delete load code is used as a rubout or spacer and is disregarded by the processor. The Compute load code is used to transition the processor from a Fill submode to the Idle 2 submode, where it will remain until the Halt control signal goes to zero, then it will enter the Prepare to Compute submode and then into execution (Ref 29). If a tape reader or some other automatic input device is used to input information, the Halt load code is used to stop the filling or verifying operations at a particular point in the input stream.

Discrete Inputs. The Discrete Inputs enter the D17B computer on 48 separate lines which are divided into two sets, X-inputs and Y-inputs. Only X_1-X_{19} of the X-input lines are available external to the computer. All 24 lines of the Y-inputs are available externally.

The instruction DIA transfers X_1-X_{19} to A_1-A_{19} of the Accumulator, and A_{20}-A_{24} come from monitoring signals internal to the computer. The input into A_{20} comes from the DR flip-flop which can be set externally. When DR is set (DR=1) two Discrete Outputs, D_{10} and D_{21}, are inhibited. This does not affect the operation of the computer, but it does cause an operation independent of the computer to occur. The instruction RSD will reset the DR flip-flop.

The D17B has a Fine Countdown mode which it enters with the EFC instruction and exits with the HFC instruction under program control. The Fc flip-flop monitors this condition (Fc=1 in Fine Countdown) and is the input to A_{21} when DIA is executed. Fine Countdown was used by the

computer in its original configuration to integrate velocity inputs to give position information. The phase register (P_1, P_2, P_3) provides the inputs to the remaining bits in the Accumulator when DIA is executed. P_3, P_1, and P_2 are inputs to A_{22}, A_{23}, and A_{24}, respectively.

The instruction DIB transfers Y_1-Y_{24} into A_1-A_{24} of the Accumulator which allows monitoring of the 24 external signals available to the computer. This Discrete Input could be used to input 24-bit words into the computer. Reference 29 describes a system designed and implemented by J. Theriault at AFIT which uses the Y Discrete Inputs as the primary input lines.

Incremental Inputs. As the Incremental Inputs are processed into the computer they are accumulated in two 4-word input loops, V and R. Six pairs of inputs I_{1A} and I_{1B} through I_{6A} and I_{6B} are available for the V- and R-loops as shown in Table VII. The same table shows a total of 26 possible Incremental Inputs when the special and R-variable inputs are included.

Resolver decoding is used to determine the inputs to the V-loop. A modulo-four counter, flip flops W_A and W_B, is used to determine which group of inputs will be sampled and which word of the V-loop will receive the results of the sampling. I_{1A} and I_{1B} are sampled every other word-time and I_{2A}, I_{2B} as well as I_{3A}, I_{3B} are sampled every four word-times, resulting in the sampling pattern: I_2, I_1, I_3, I_1, I_2, I_1, I_3, ... The sampling and decoding process determines if there has been a positive or a negative change in the input based on the previous sample. If there has been a positive change, the appropriate word in the V-loop (as determined by W_A and W_B) is incremented by one; a negative change would have decremented the same word by one. This decoding was done when the processor was not

TABLE VII

Incremental Input Specification

Class	Input Group	Sampling Device	Sampling Rate (samples/second)
V	I_{1AC}, I_{1BC}	V-loop, word 0 and 2	6500
V	I_{2AC}, I_{2BC}	V-loop, word 1	3250
V	I_{3AC}, I_{3BC}	V-loop, word 3	3250
R	I_{4AC}, I_{4BC}	R-loop, least significant half of word 3	3250
R	I_{5AC}, I_{5BC}	R-loop, most significant half of word 3	3250
R	I_{6AC}, I_{6AC}	R-loop, most significant half of word 2	3250
R-variable	$I_{1NC}, I_{11C}, I_{14C}$	R-loop, least significant half of word 1	3250
R-variable	$I_{3NC}, I_{31C}, I_{34C}$	R-loop, most significant half of word 1	3250
R-variable	$I_{2NC}, I_{21C}, I_{24C}$	R-loop, least significant half of word 1	3250
R-variable	$I_{4NC}, I_{41C}, I_{44C}$	R-loop, most significant half of word 1	3250
Special	I_{BC}	D_r	Continuous*
Special	I_{7C}	R-loop, least significant half of word 0	1625

*If D_r is in the "0" state.

in the Fine Countdown mode so that errors in the velocity meters could be calculated. When Fine Countdown was entered resolver decoding continued, but with a few changes. The V-loop was no longer modified by the decoder outputs. When the inputs changed, the appropriate word in the V-loop (determined by W_A and W_B) was added to or subtracted from the U-loop, a one-word loop. In this mode of operation the V- and U-loops formed a digital integrator with V as the Y register, U as the R register, and the incremental inputs as the dx input (Ref 18:64). When a solution is reached, U goes negative, D5 of the Discrete Output register is set enabling D16 if the D register had been zero set prior to the U-loop going negative.

The R-loop only processes split words, thus it has the capability of receiving eight different inputs, three resolver inputs, four variable inputs, and one pulse input. The R-incremental Inputs or resolver inputs are I_{4A}, and I_{4B} through I_{6A} and I_{6B}. Resolver decoding is identical to the V-loop except for the timing which is changed because of the split-word inputs. The results of sampling are processed the same as the V-incremental Inputs with FC=0. The second group of inputs to the R-loop are the R-variable Incremental Inputs. In general these inputs are processed the same as the V-incremental and the R-incremental inputs. The major difference is indicated in Table VII, three inputs instead of two. These three inputs are used differently also. I_{iN} (i=1-4) determines if addition or subtraction will occur while I_{i1} and I_{i4} determine the magnitude. If I_{i1} is true the operand is 1 or if I_{i4} is true the operand is 100 binary. If both I_{i1} and I_{i4} are true, the operand is 101 binary. Another difference is the use of the P_2 flip-flop of the Phase Register to select the inputs.

If P_2=1 then I_{2i} and I_{4i} (i=N, 1, 4) will be selected whereas if P_2=0, I_{1i} and I_{3i} will be selected.

The final inputs to the R-loop, I_7 and I_B, form another group of inputs called the Special Incremental Inputs. I_B is a binary signal which sets the DR flip-flop which in turn inhibits two Discrete Outputs, D10 and D21. DR can be reset using the Reset Detector (RSD) instruction. The final input, I_7, is sampled once every four word times. If I_7 is a one when it is sampled, the right split-word of word zero of the R-loop will be incremented by one. This split word, the Coarse Time Counter, can only be incremented every eight word times. When I_7 is sampled and I_7=1, then Rx is set so that four word times later when I_7 is sampled and found set, the Coarse Time Counter won't be incremented. The left half of word zero is the Fine Time Counter which is incremented every eight word times unless the Coarse Time Counter is incremented. In this case the Fine Time Counter is reset to a -1(1000000000_2) and continues counting. The Coarse Time Counter counts the number of pulses sampled and the Fine Time Counter counts the number of word times that have elapsed since the last pulse was sampled.

Outputs

The Outputs of the D17B computer are varied and specialized for the same reasons as the Input signals. The Output signals will be functionally divided into seven groups as show in Fig. 18. These signals are all processed by the I/O circuitry so that external devices may use them without overloading the internal circuitry (Ref 29).

Discrete Outputs. The Discrete Outputs are used to output 28 "on/off" signals through program control. The Discrete Output A (DOA)

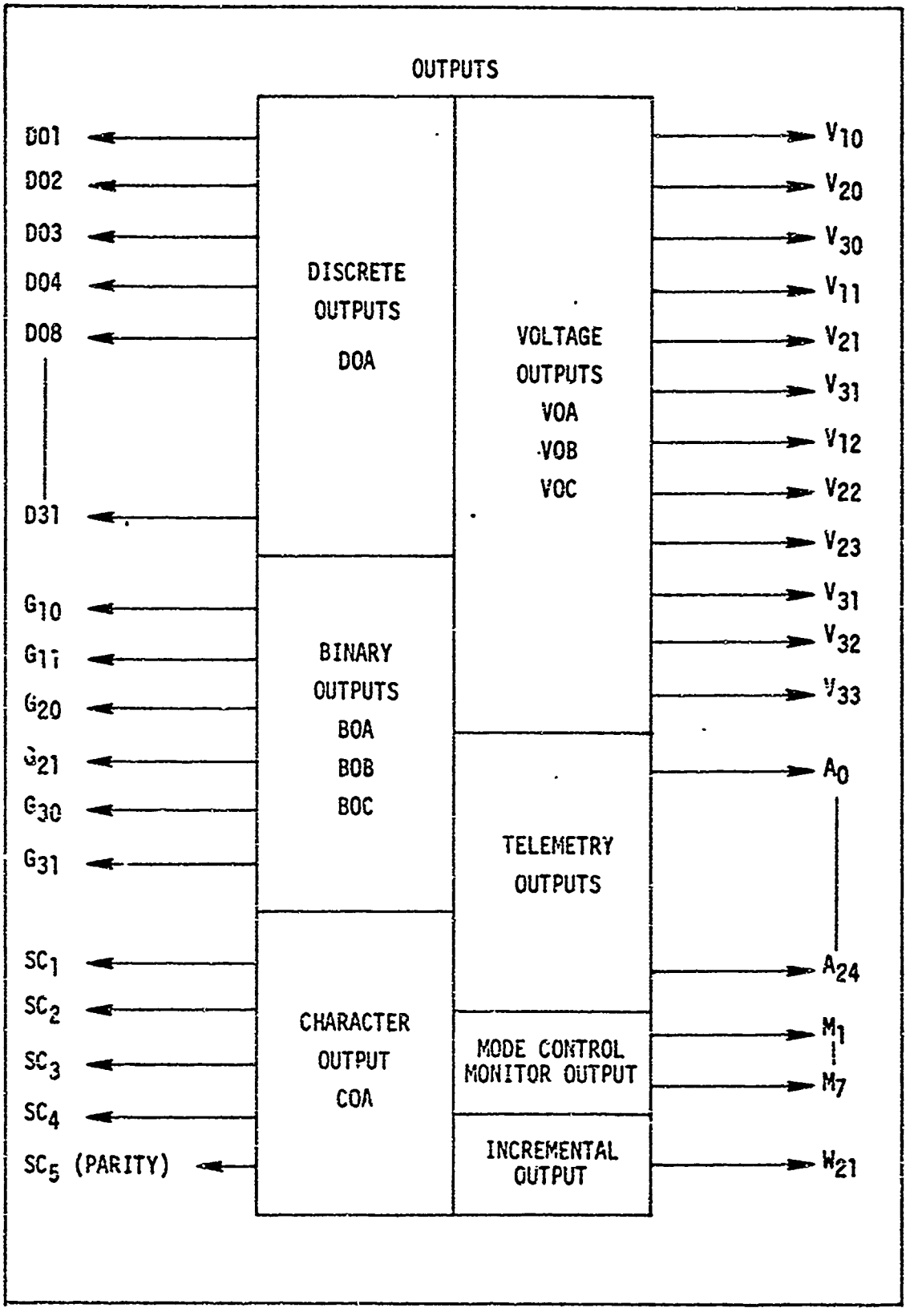


Fig. 18. D17B Computer Outputs

instruction transfers a code in the sector address portion of the instruction to the D register. This signal is then decoded and one of the 28 possible Discrete Output lines is enabled. There is one exception; D04 may be enabled at the same time as D01, D02, or D-03. Since these three combinations can be decoded externally to form D05, D06, D07, the Discrete Outputs are numbered D01-D04 then D08-D31 for a total of 28 lines.

There are several other conditions or signals which will modify the D-register outputs or the Discrete Outputs. The Disable Discrete control signal will disable all the Discrete Outputs without affecting the D-register. When the Detector flip-flop is set (DR=2), D10 and D21 are disabled, although the other outputs are not affected. Another condition which will modify the D-register, and thus the outputs, occurs during Fine Countdown. If the U-loop goes negative, D5 is set and the D-register may be changed, disabling at least half of the outputs as indicated in the Veitch diagram in Fig. 20.

Binary Outputs. The Binary Outputs consist of three signals, G_{i1} (i=1,3), and their complements, G_{i0} (i=1,3). These signals are illustrated in Fig. 19.

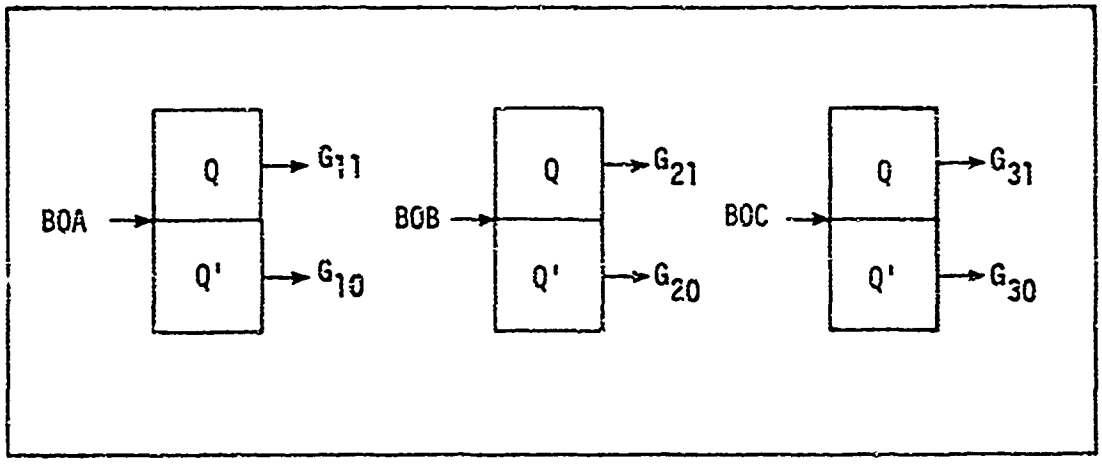


Fig. 19. Binary Outputs

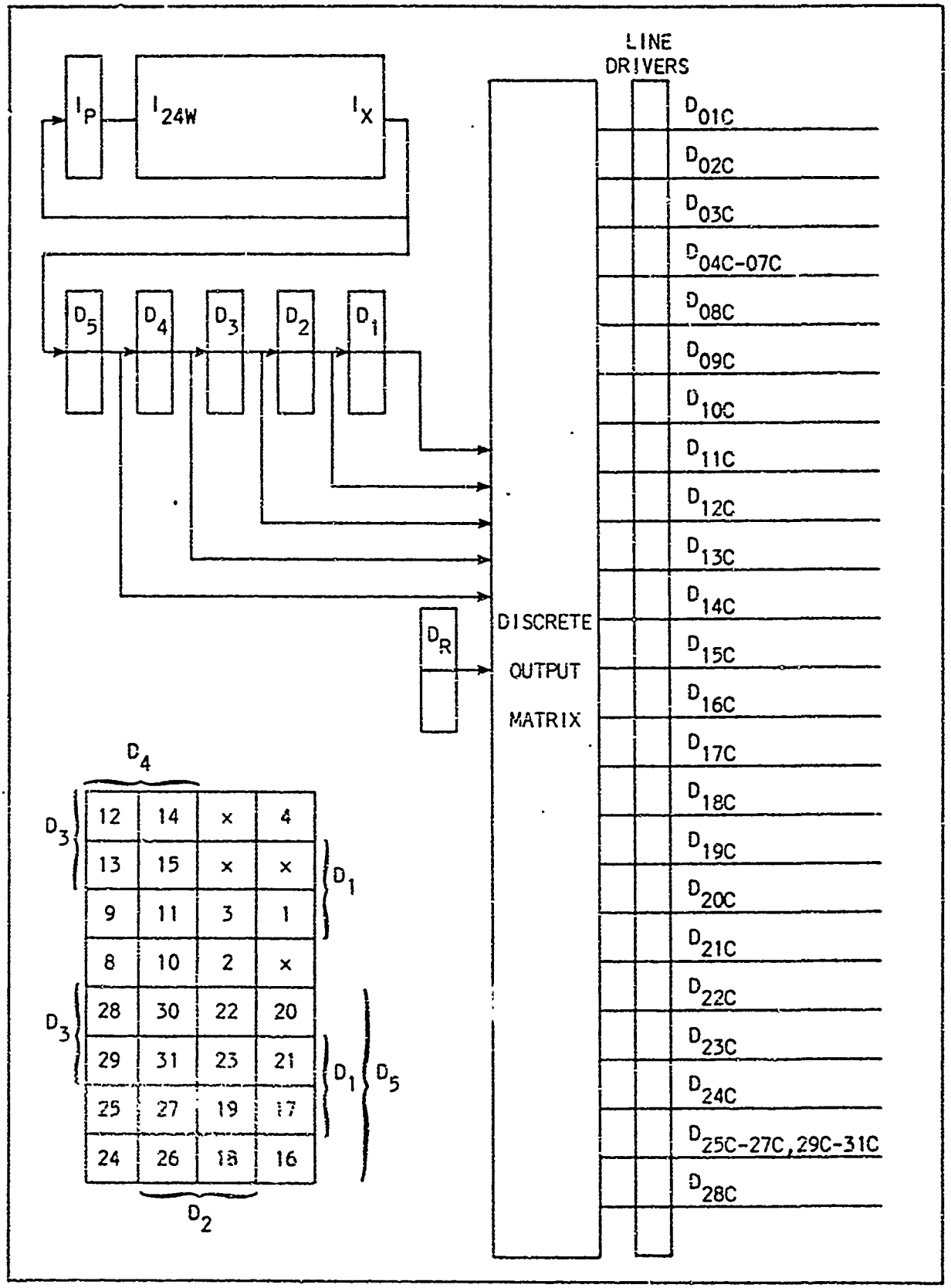


Fig. 20. Discrete Outputs

The Binary Output instructions, BOA, BOB, and BOC, are used to enable these signals. The Binary Output signal, BOA for example, will increment the contents of the Accumulator if G_{11}=1 or decrement the contents of the Accumulator if G_{10}=1. After the contents of the A-register are modified, the G_1 flip-flop is set if A<0 (G_{11}=1), or reset if A≥0(G_{10}=1). Once the respective G flip-flop is set it will remain in that state until changed by another Binary Output instruction, a Master Reset control signal, or a power loss. The last two cases result in a random setting of the flip-flops so it is assumed the signal is lost.

Character Outputs. The Character Outputs use a set of six lines, S_{c1}-S_{c5} and S_{ct}, to output 4-bit character information. Lines S_{c1}-S_{c4} determine the 4-bit character, line S_{c5} holds the parity bit, and line S_{ct} is used for timing. The Character Output A (COA) instruction is used to output one 4-bit character. When the COA instruction is executed, the Accumulator is left-shifted four bits, an odd-parity bit is generated, and S_{ct} is set for the time S_{c1}-S_{c5} is available on the output lines. The time this signal is available is determined by the low-order five bits in the COA instruction, thus the maximum time the outputs are available is 31 word times (2.4 msec).

Voltage Outputs. The Voltage Outputs consist of four sets of outputs with three signal lines for each set as shown in Fig. 21.

The Phase Register (P_1, P_2, P_3) is used to select which set of analog signals will be used. This selection is illustrated in Table VIII.

The Voltage Output instructions, VOA, VOB, and VOC, cause the high-order eight bits of the left or right split-word to be transferred into Register 1, 2, or 3, respectively, as illustrated in Fig. 21. Bit four of the instruction determines which split-word to transfer into the register.

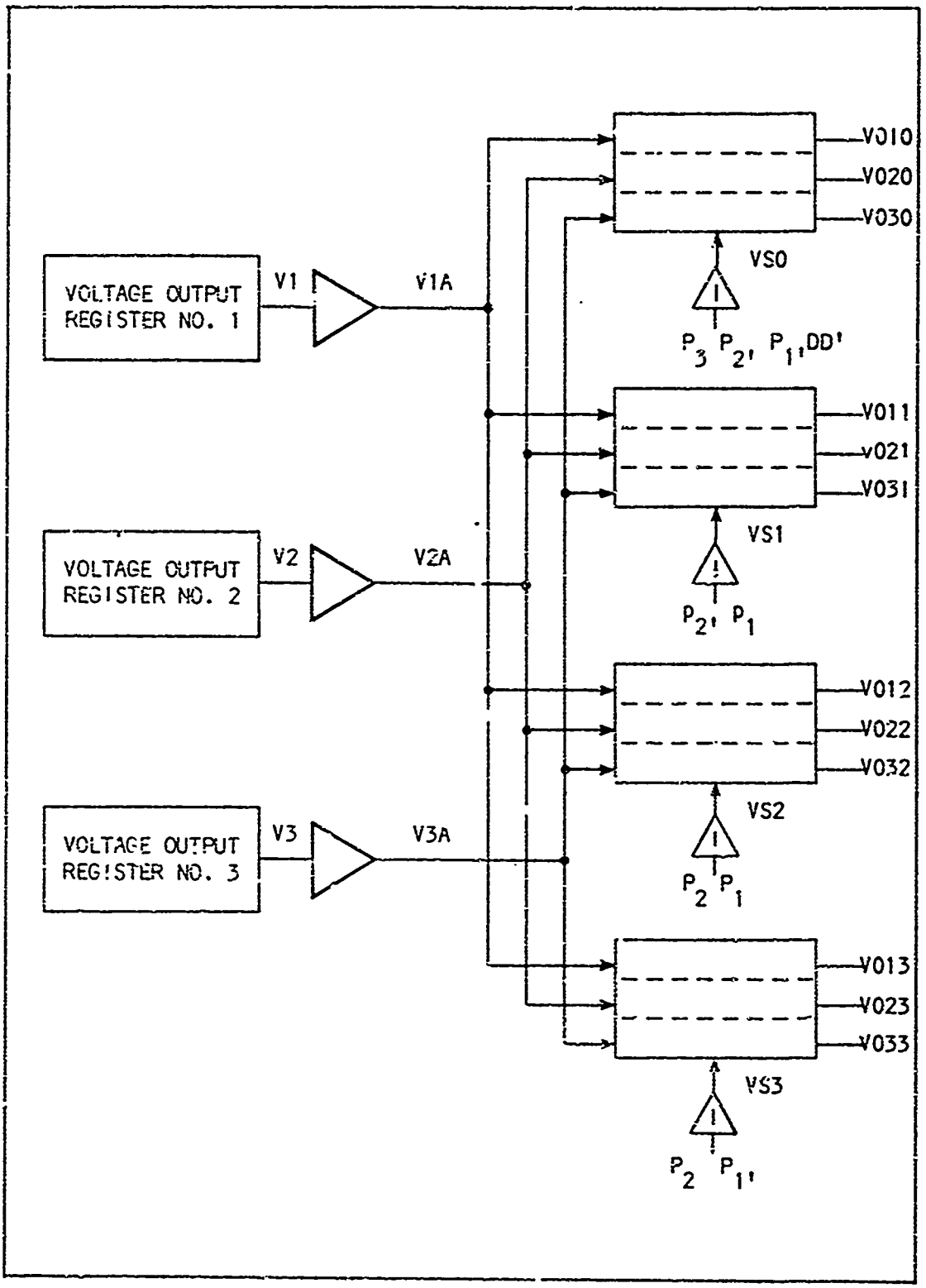


Fig. 21. Voltage Outputs

TABLE VIII

Voltage Outputs as Determined by Phase Register Settings

Voltage Outputs Enabled	Phase Register P_1	P_2	P_3[a]
V_{10}, V_{20}, V_{30}[b]	1	0	1
V_{11}, V_{21}, V_{31}	1	0	d
V_{12}, V_{22}, V_{32}	1	1	d
V_{13}, V_{23}, V_{33}	0	1	d

[a]A "d" indicates a don't care state, P_3 may be a 0 or 1.

[b]The Discrete Disable control signal will disable V_{10}, V_{20}, and V_{30} when DD=1.

The contents of Register 1, 2, or 3 are shifted into a D/A converter which outputs an analog voltage signal proportional to the digital value of the 8-bits in the register. The output of the D/A converter ranges from -5 VDC to +5 VDC and is amplified to ±20 VDC at the output lines (Ref 18:47).

Figure 21 shows that VOA enables V_{10}, V_{11}, V_{12}, or V_{13}; VOB enables V_{20}, V_{21}, V_{22}, or V_{23}; and VOC enables V_{30}, V_{31}, V_{32}, or V_{33}, depending on the Phase Register setting.

Telemetry Outputs. The Telemetry Outputs were used by the Minuteman I missile to telemeter the internal operations of the D17B to ground control stations. Telemetry can output any one of seven signals during one word time. The contents of the Accumulator can be read out continuously

or periodically under program control by flagstoring the Accumulator into the T-loop (Telemetry). The clock can be read out provided the memory disk is in position against the read-write head plate (Ref 29). The Discrete Outputs Monitor signals are specialized outputs which were originally used to monitor the progress of the missile during flight. These signals are available as Telemetry Outputs. They are simply pre-selected Discrete Output signals logically ANDed together as shown in Table IX. The Discretes Disable control signal will disable these monitor signals by disabling the inputs to the AND gates.

TABLE IX

Composition of Discrete Output Monitor Signals

Logical AND of Discrete Output Signals	Discrete Output Monitor Signal
$(D_{15})\cdot(D_{17})\cdot(D_{11})$	D_{t1}
$(D_{12})\cdot(D_{16})\cdot(D_{19})$	D_{t2}
$(D_{10})\cdot(D_{13})\cdot(D_{18})$	D_{t3}
$(D_{14})\cdot(D_{20})\cdot(D_{21})$	D_{t4}

The Memory Multiplexer flip-flop Mpx is another Telemetry Output. This flip-flop copies and outputs serially one channel of memory during one disk revolution. External logic could be used to monitor the sectors and display the output of Mpx when a specified sector is read. In this manner any word in memory could be displayed on an external device.

Another output signal is Origin Timing. When the processor is in compute mode a 2.9 μs pulse at Tp of sector 177 is issued which indicates the start of one disk revolution (Ref 29). Program Timing and Sector Channel are the final two Telemetry Outputs. Program Timing is an external signal used to indicate that the contents of the Accumulator have been flagstored into telemetry (Ref 2:68). The Sector Channel outputs are the sector numbers 000 to 177, octal, which are recorded on one channel of the disk called the Sector Channel.

Mode Control Monitor Signals. The Mode Control Monitor Signals are seven outputs which are amplified and processed for external use to monitor and display the modes of the processor (Ref 29). These seven signals are: Compute mode; Fill mode; Manual Halt mode; Parity Error; Parity or Verify Error; Program Halt mode; and Verify mode. If these signals are displayed they can help the programmer in entering or debugging a program.

Incremental Output. The Incremental Output consists of only one signal, W_{21}, which goes true once every four word times starting at sector one. This output was originally used to control the devices monitored by the Incremental Inputs (Ref 29).

Output Summary. The outputs are summarized in Fig. 18 at the beginning of this section. Briefly, the number of useful output lines available are tabulated in Table X.

TABLE X

Useful Output Lines Available

Output Signals	Lines Available
Discrete	28
Binary	6
Character	5
Voltage	12
Telemetry	24
Total	75

Specifications and Features Applicable To Control Operations

Certain specifications of the D17B computer are applicable to numerical control and process control. The most general specification or feature of the D17B which is applicable is the overlapping of several modes of operation, such as instruction search and operand search occurring during the same word-time. The 24-bit word length is advantageous in numerical control because it offers very high accuracy is a necessity for certain operations. When high accuracy is not needed the D17B can operate as a dual processor for the arithmetic operations. For this situation, the 24-bit word can be split into left and right 11-bit words, including the sign bit. Direct addressing is another desirable feature, but it is of limited value due to the small memory on disk. The split, compare, and limit instruction can be used to keep split-words from outputting values which are out of range.

Inputs. The character, discrete, and incremental inputs are applicable to control operations. The character inputs are used to enter the programs and data needed by the programs. The incremental inputs were used to integrate velocity inputs to determine position in the Minuteman I ICBM. These inputs will still be integrated when the computer is in fine countdown so that the proper inputs and constants should result in position information. Also, when a solution is reached, D5 of the Discrete Output register is enabled which could be used to stop whatever operation is in progress.

There are 43 usable discrete inputs that can be used to enter data (with added external logic), to monitor different conditions in a process, or to receive data from A/D converters. Twenty-four of these 43 inputs

can be read into the computer with one instruction and the other 19 with one more instruction. This results in an effective input rate of 3.64 μsec per discrete input.

Outputs. There are three categories of outputs that are applicable to control operations: analog, binary, and discrete. Analog or voltage outputs are usually needed in control applications, such as a valve or positioning a table on a drill press. The D17B analog outputs only have 8-bit resolution which is not accurate enough for most numerical control applications, but may be sufficient for specialized operations. There are 12 separate analog outputs, all under program control.

The binary output signals (pulse lines) are G_{11}, G_{21}, and G_{31} plus the complement for each of these binary outputs. All six signals are available for output. These signals stay on until they are turned off by the program. For example, G_{11} may be true (which makes G_{10}, the complement, false) and it will stay true until an instruction is issued to change it. Then G_{11} will be false and G_{10} will be true. By proper programming the Binary outputs can be used as a pulse output, oscillating between true and false for a specified time.

The third set of outputs, the 28 discrete outputs, could be used to turn switches on and off. As stated previously, only one discrete output can be on at any one time with the exception of D04 which can be on with D01-D03. If D011 was issued, turning switch one on, it would remain in the true state until a different discrete output was issued. Then D011 would have to be issued again to turn switch one off. To use the discrete outputs this way necessitates using a switch that requires a positive or negative going pulse to turn it on and another pulse of the

same polarity to turn the switch off. Using this type of switch requires the program to keep track of which switch is on and when.

General. In general the D17B computer has all the inputs and outputs that are used in control applications except voltage inputs. But, the slow speed, limited memory capacity, and 8-bit accuracy in the D/A converters make the use of the D17B in general control applications questionable. The D17B computer could be used to advantage in certain specialized applications such as data acquisition and preprocessing data in process control applications and simple point-to-point numerical control applications. Chapter V discusses some possible numerical control configurations using the D17B computer.

III. Model Formulation and the Implementation Of a Machine Positioning System

As mentioned previously it was not possible to obtain a machine of the desired nature in the time available. This necessitated the formulation of a model as an alternate procedure. A digital computer model of the machine was a possibility; however, an analog computer simulation was considered a more direct solution. This approach required a direct connection between the D17B and the machine model with the proper interfaces.

Servos and Drive

Several different devices are available to drive the worktable (the table which holds the part to be machined) in accordance with the desired commands. Electrohydraulic servovalves can be used to control hydraulic motors. Hydraulic power sources are frequently used when large power is required. Another method of moving the worktable is with the use of electric motors. These motors can be either armature-controlled or field-controlled.

Field-Controlled Motors. Since less power is required to control the field of the motor, this method is frequently adopted (Ref 11:47). The simplified diagram of a field-controlled dc motor is shown in Fig. 22.

For a constant armature current, i_m, the torque, T, is proportional to the flux, ϕ, which is in turn proportional to the field current, i_f. Hence the torque equation is

$$T = K_f i_f \tag{1}$$

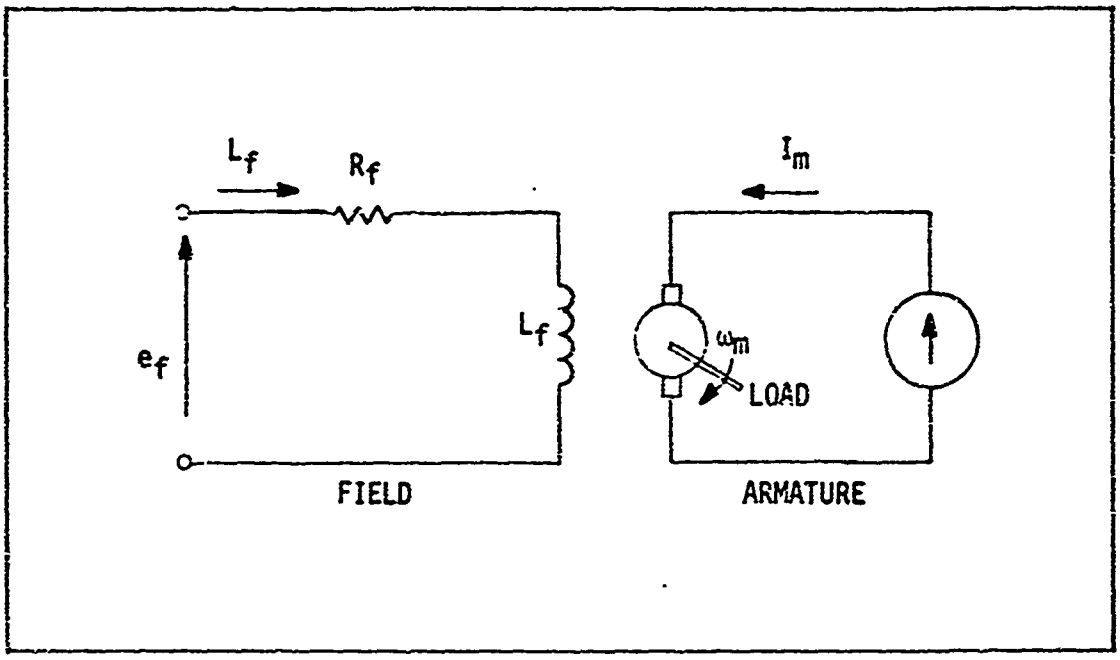


Fig. 22. Simplified Diagram of Field-Controlled dc Motor

where K_f is the proportionality constant between torque and current. The equation for the field circuit voltage, e_f, is

$$(sL_f + R_f)i_f = e_f \tag{2}$$

where s is the differential operator, L_f and R_f are field inductance and resistance, respectively. For an inertia-viscous friction load as depicted in Fig. 23, the torque equation is given by

$$(sJ + B)\omega_m = T \tag{3}$$

where J is inertia, B is the damping coefficient, and ω_m is the armature velocity.

Now the equation for motor velocity can be written as

$$(sL_f + R_f)(sJ + B)\omega_m = K_f e_f \tag{4}$$

by combining Eqs (1), (2), and (3). In practice a constant armature current is difficult to obtain. If a constant voltage is applied to the

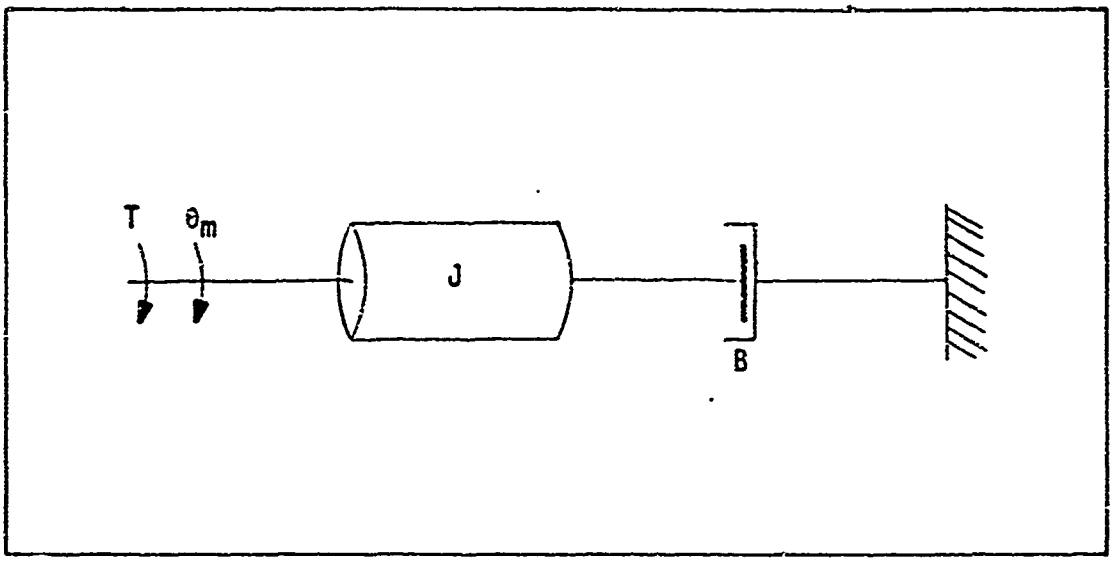


Fig. 23. Friction/Inertia Load for dc Motor

armature, the current is given by

$$i_m = \frac{E_a - e_m}{R_m} = \frac{E_a - K_2\phi\omega_m}{R_m} \tag{5}$$

where

e_m = back emf,

E_a = armature voltage,

R_m = armature resistance, and

K_2 = flux/field current conversion constant.

For $E_a >> e_m$ we can approximate i_m as

$$i_m \approx E_a/R_m \tag{6}$$

The angular rotation of the motor shaft must be transformed into linear motion to move the worktable. In actual practice very sophisticated devices are used to eliminate backlash and ensure accurate positioning. For modelling purposes a system of rotary and worm gears will suffice to effect the transformation. Figure 24 shows a simplified view of the

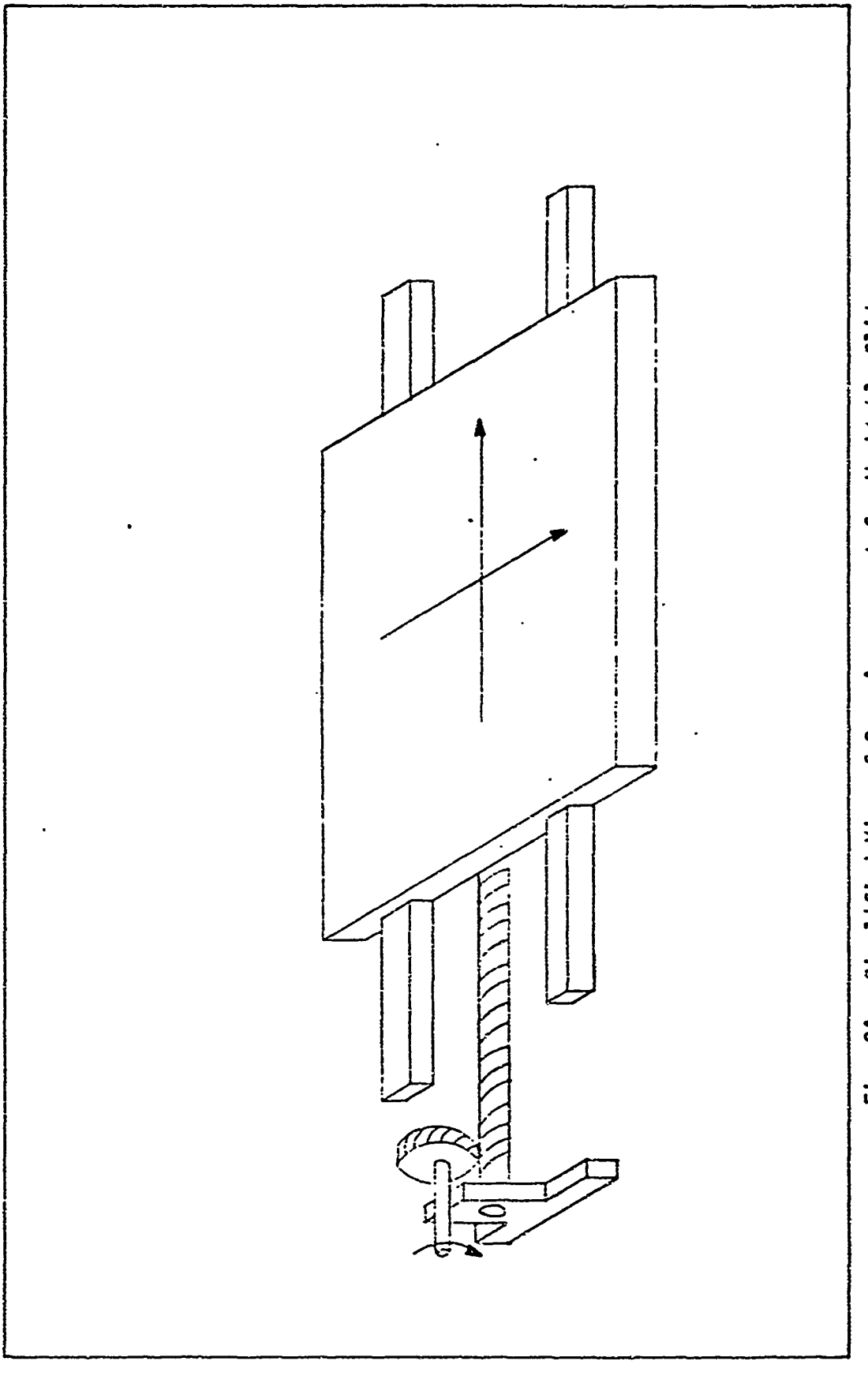


Fig. 24. Simplified View of Gear Arrangement for Worktable Slide

gearing required to transform rotary motion to linear motion. Only one axis is depicted. A similar arrangement is required for the other axis. For small loads such as a printed-circuit board, the inertia of the workpiece can be ignored. If larger loads are considered, the inertia of the table and workpiece can be included as a load on the motor (corrected by the transformation of the gearing ratio). Assuming ideal coupling by the gears, this load can be transformed into a load in parallel with the original load on the motor shaft (Ref 12:91).

Now the angular velocity, ω_m, and angular position, θ_m, are related by

$$\omega_m = s\theta_m \tag{7}$$

Similarly, assuming perfect coupling, angular position and linear position, x_1, are related by

$$x = K_g\theta_m \tag{8}$$

where K_g represents the gear-coupling constant between the motor and the table. Hence

$$\omega_m = sx/K_g \tag{9}$$

and we can rewrite Eq (4) in terms of the linear position

$$(sL_f + R_f)(sJ + B)sx = K_gK_fe_f \tag{10}$$

or, in transfer function form

$$\frac{x}{e_f} = \frac{K_gK_f}{s(sL_f + R_f)(sJ + B)} \tag{11}$$

Steady-State Response. The Final Value Theorem (Ref 12:11) states that the final steady-state response is given by

$$X(\infty) = \lim_{s\to 0} SG(S)E_f(S) \tag{12a}$$

$$X(\infty) = \lim_{s\to 0} S \frac{K_g K_f/(R_f B)}{S(T_1 S+1)(T_2 S+1)} \frac{1}{S} \tag{12b}$$

where S^{-1} is the Laplace Transform of the unit step input. From this it can be seen that X goes to infinity as time increases when a step input is applied. This can be remedied by negative feedback as depicted in Fig. 25.

Armature-Controlled Motors. Although field-controlled motors have the advantage of allowing control at a lower power level, they have the disadvantage of speeding up at lower applied voltages when one would want them to rotate quite slowly. For this reason armature-controlled motors are frequently used in applications in which it is desired to operate at very slow speeds for portions of an operational cycle.

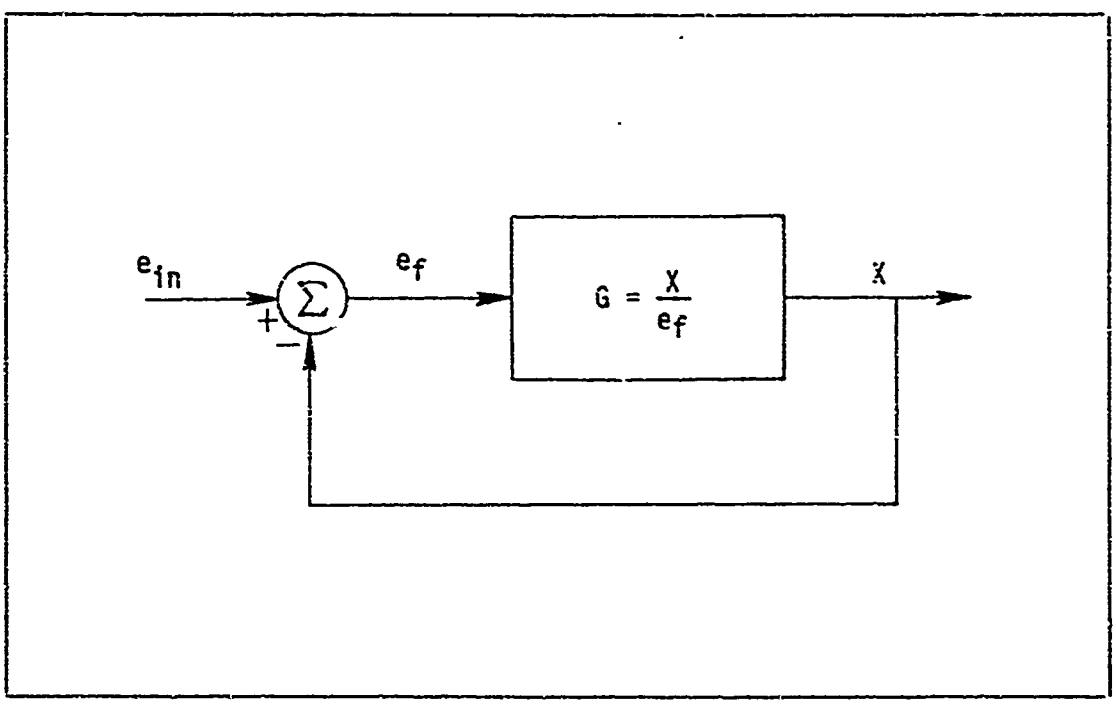


Fig. 25. Closed-Loop System

Since an armature-controlled motor is better for our application, we shall generate our model to conform to that configuration.

In the armature-controlled configuration, the torque equation is

$$T = K_3 \phi i_m \tag{13}$$

where

ϕ = field flux,

i_m = armature current, and

K_3 = a constant for a given motor.

In an armature-controlled motor the field has a fixed voltage and hence a constant flux, ϕ. Thus Eq (13) becomes

$$T = K_t i_m \tag{14}$$

where K_t is the torque constant. Figure 26 shows a simplified schematic of an armature-controlled motor.

The counter-emf, e_m, is given by

$$e_m = K_v \omega_m = K_v S \theta_m. \tag{15}$$

where K_v is the generator constant. The voltage equation in the input circuit is given by

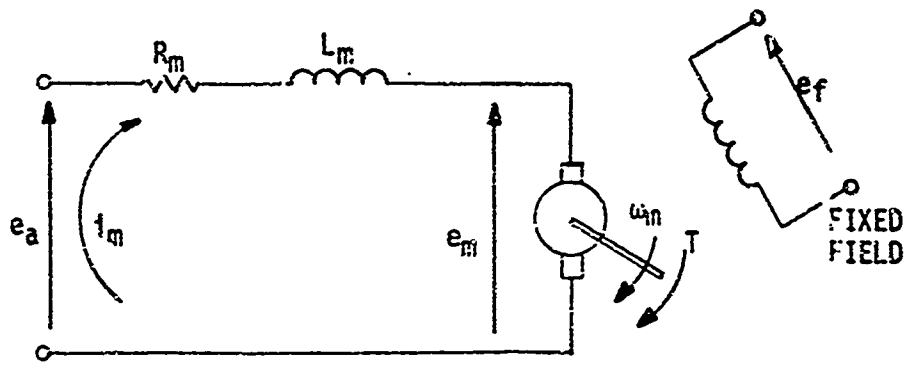
$$e_a = (L_m S + R_m) i_m + e_m \tag{16}$$

Again, assuming a damping and inertia load, Eq (3) portrays the load torque equation

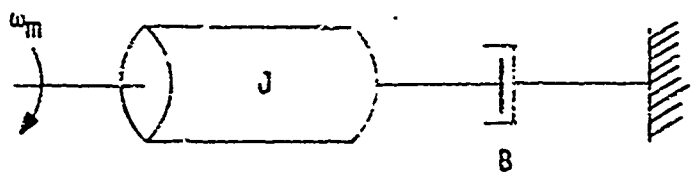
$$T = (JS + B)\omega_m \tag{3}$$

Combining Eq (3) with Eqs (14) through (16) results in

$$[(JS + B)(L_m S + R_m)/K_t + K_v]\omega_m = e_a \tag{17a}$$



a. Motor



b. Load

Fig. 26. Armature-Controlled Motor and Load

or in terms of X

$$[(S^2JL_m/K_t + S(JR_m + BL_m)/K_t + (BR_m + K_vK_t)/K_t]SX/K_g = e_a \qquad (17b)$$

A block diagram of the system incorporating unity feedback and an amplifier of gain K is shown in Fig. 27, and G is the transfer function X/e_a as defined by Eq (17b).

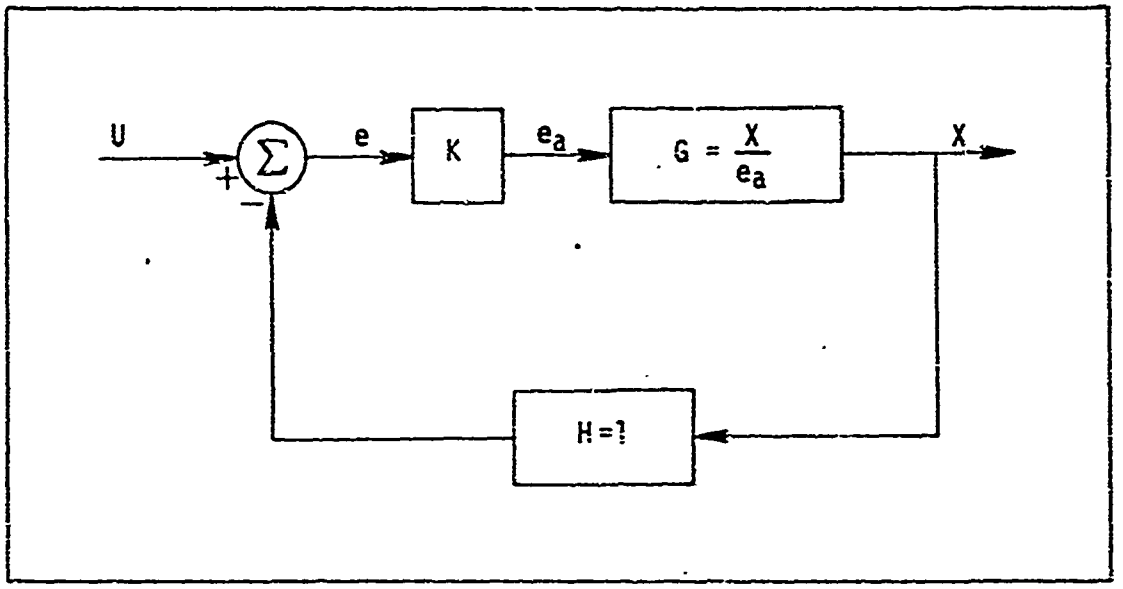


Fig. 27. Armature-Controlled Feedback System

Analog Computer Model of a Two-Axis Positioning System

Parameter Selection. Before implementing the model on the analog computer, one important step remains--that of parameter selection in accordance with some reasonable guidelines.

The first group of parameters which we might select are the ones associated with the motor itself. Selecting a five horsepower motor for its high torque capability, we could obtain a 240-volt dc, armature-controlled General Electric motor with the following parameters (Ref 23:22):

R_m = 0.615 ohms

L_m = 0.0045 henries

K_t = 0.7 lb-ft/amp

K_v = 1.21 volts/rad/sec

K_f = 150 watts = power for full field

J_m = 0.050 lb-ft sec^2

T_m = motor inertial time constant = 0.036 sec

N_r = rated speed = 1750 rpm

The only remaining parameter to consider is the gearing ratio, K_g. The constraint which is placed on this parameter is that adequately high table slew rates may be attained when moving the table from point to point. This ensures that time will not be wasted in moving long distances. Although industrial systems sometimes attain speeds of 1000 ipm in high performance applications, we will incorporate a figure of 100 ipm for modelling purposes as a figure more representative of industrial use. Considering Eq (8) and differentiating, we obtain

$$\dot{X} = K_g \dot{\theta}_m \tag{18}$$

Therefore, in consideration of the rated speed of the motor, we get

$$K_g = 0.00758 \text{ ft/rad}$$

Inserting these parameters into Eq (17b) and solving for the highest order derivative yields

$$\dddot{X} = 17.7e_a - 15.65\ddot{X} - 404\dot{X} \tag{19}$$

Assuming an amplifier gain of 12 which will produce the rated armature voltage with the maximum of 20 volts out of the D-178 computer results in a closed-loop transfer function of

$$X/U = \frac{212}{S^3 + 15.65S^2 + 404S + 212} \tag{20}$$

Figure 28 shows the root locus plot obtained with the aid of the Root Locus program at AFIT (Ref 17).

An analog computer diagram of the plant is shown in Fig. 29. The variables shown in the various portions of the circuit are amplitude scaled as indicated by the square brackets. This is necessary in order that the maximum voltage available at the output of the various amplifiers will not be exceeded (Ref 14:36).

Assuming a solution of

$$X(t) = A \sin \omega_n t \tag{21a}$$

for the second order part of the system response, we obtain

$$\dot{X}(t) = A\omega_n \cos \omega_n t \tag{21b}$$

$$\ddot{X}(t) = -A\omega_n^2 \sin \omega_n t \tag{21c}$$

$$\dddot{X}(t) = -A\omega_n^3 \cos \omega_n t \tag{21d}$$

Since $\omega_n = 20$ and $X_{max} = A = 5$ inches we can obtain conservative maximum values for X and each of its derivatives and subsequently determine each of the scale factors, α_i. As an example, $X_{max} = 5$; hence

$$\alpha_0 = \frac{\text{computer maximum voltage}}{X_{max}} = 10/5 = 2 \tag{22a}$$

Similarly,

$$\alpha_1 = 10/\dot{X}_{max} = 0.05 \tag{22b}$$

$$\alpha_2 = 10/\ddot{X}_{max} = 0.0025 \tag{22c}$$

$$\alpha_3 = 10/\dddot{X}_{max} = 0.00012 \tag{22d}$$

$$\alpha_4 = 10/e_{a_{max}} = 0.04 \tag{22e}$$

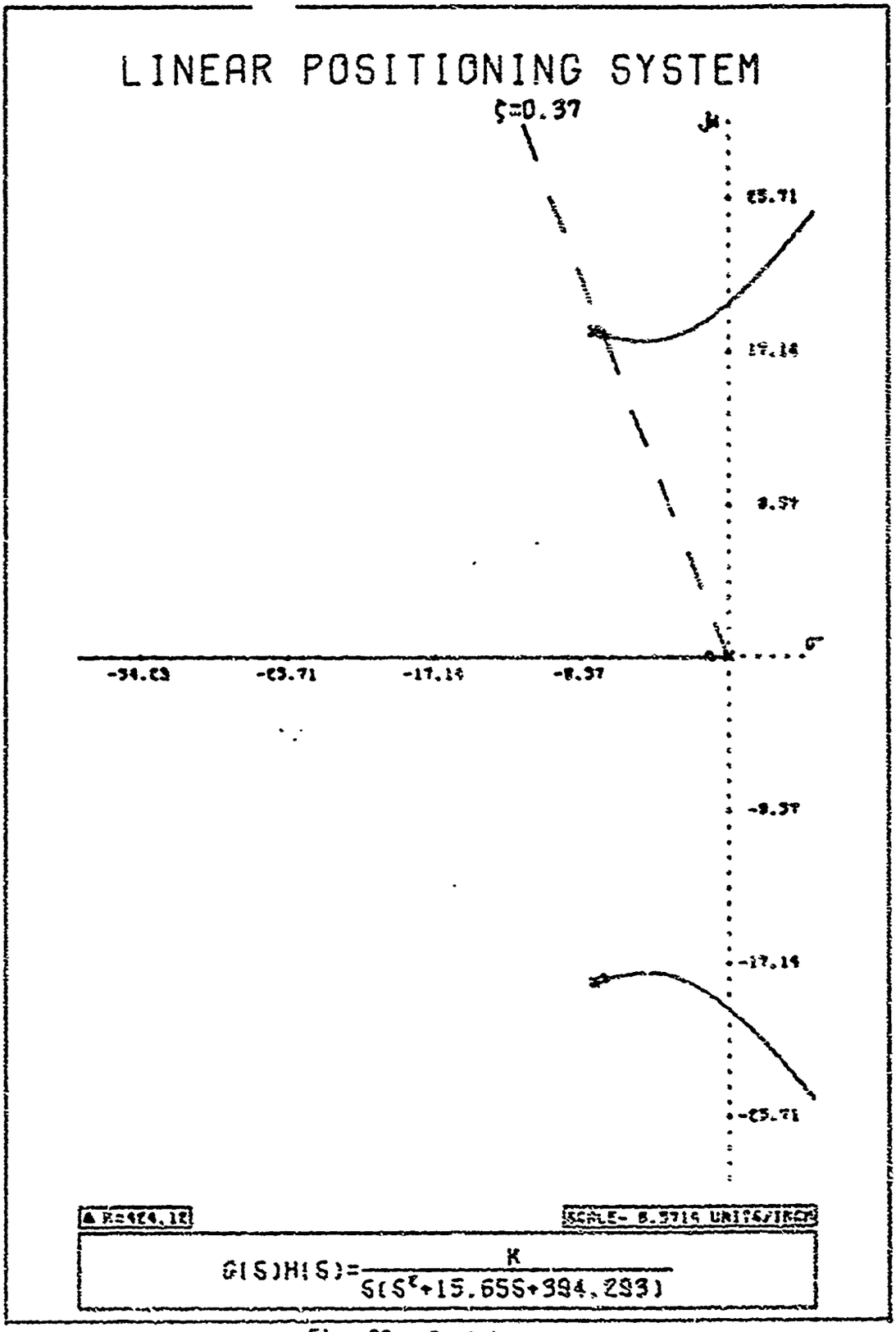


Fig. 28. Root Locus

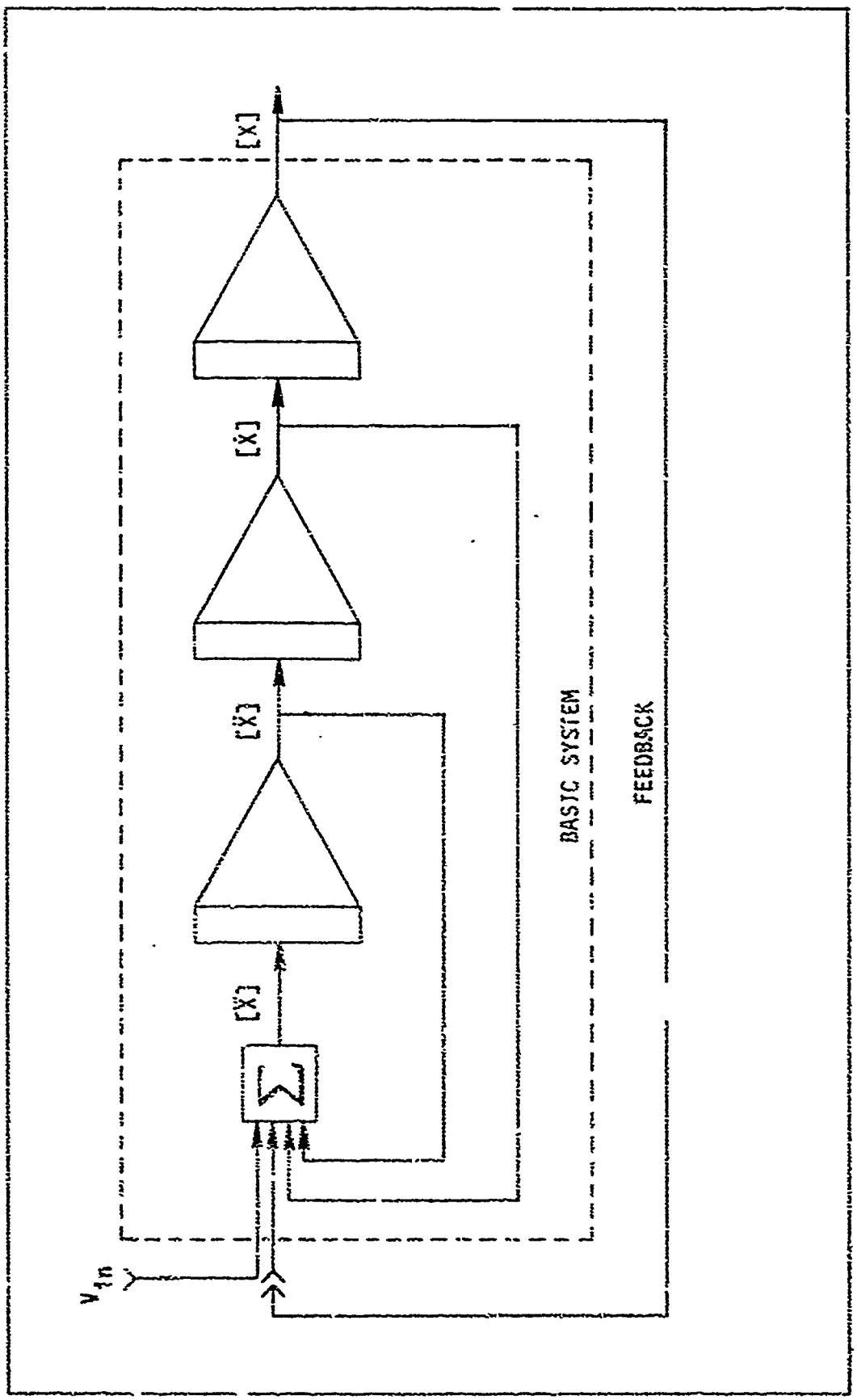


Fig. 29. Initial Computer Diagram for Armature-Controlled Motor

The final form of the positioning system was implemented on a TR-10 analog computer. Figure 30 shows selected analog computer symbols and their defining equations. The complete system diagram with feedback is shown in Fig. 31. The upper half of the figure represents the X-channel while the lower half represents the Y-channel of the system. The portions of the circuit to the right of Function Switches, FS1 and FS2, represent the inputs to the positioning system. Potentiometers 21 and 24 are the manual controls while potentiometers 3 and 13 attenuate the computer voltages (20 volts maximum magnitude) down to the scaled voltages used in the analog computer. FS1 and FS2 switch between manual control and automatic control. The inputs to potentiometers 21 and 24 are switched by means of patch cables between +10 volts and -10 volts to afford positioning in all four quadrants. Amplifiers 7 and 17 are not actually part of the basic system, but are used to precondition the signals fed to the ADC's as explained in the next section.

Most of the variables shown in the diagram are amplitude scaled as indicated by the square brackets. The voltage present in the circuit is equal to the indicated variable multiplied by its associated scaling factor, α_i. Conversely, if the voltage at a point in the circuit is known, the appropriate variable can be found by dividing it by its associated scale factor.

Computerized Control

After a general familiarization of the D17B was attained, computer programs were written to generate control signals which could be used as inputs to a positioning device. Figure 32 shows the coordinate system used as well as the computer output which will produce full-scale movement

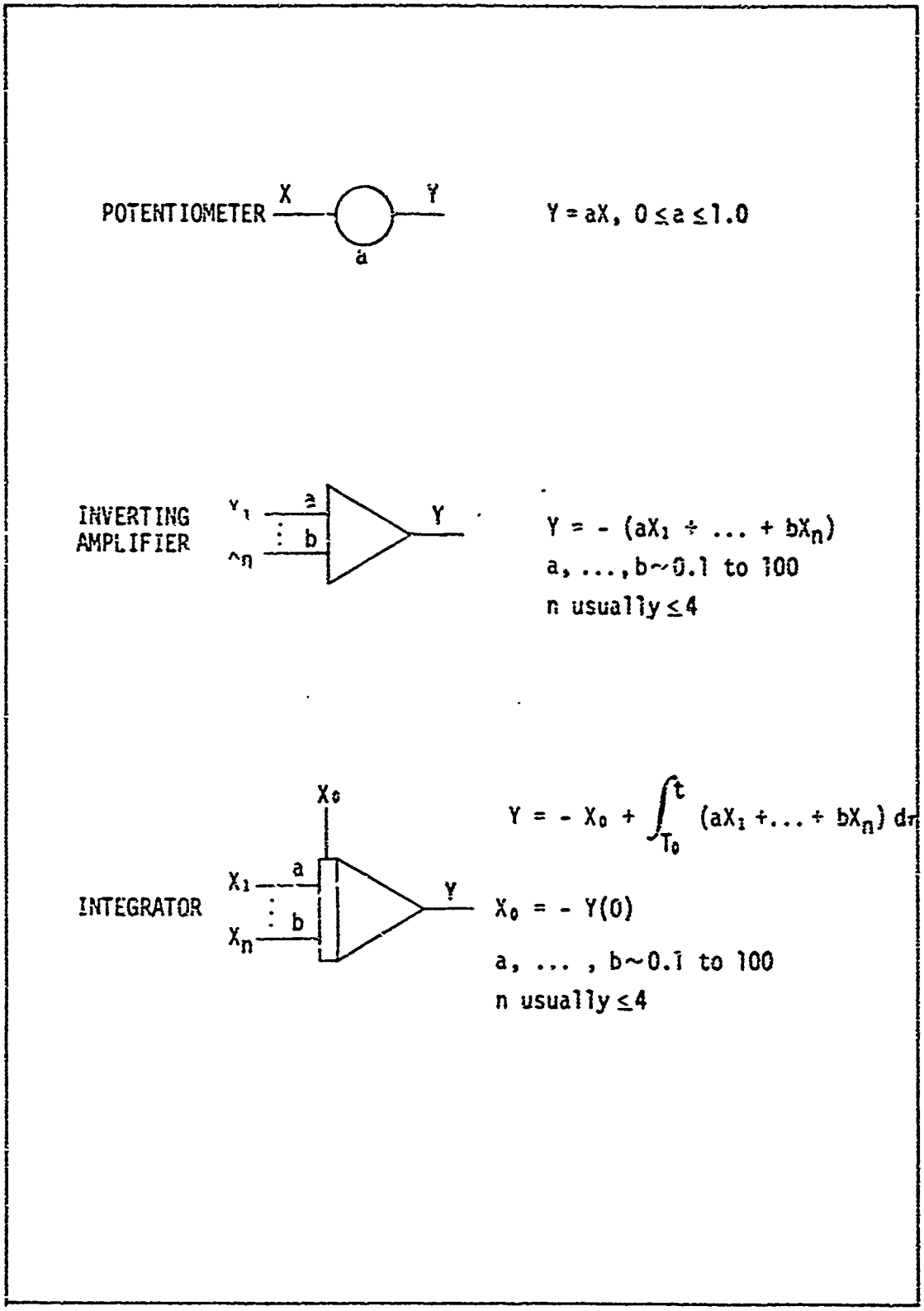


Fig. 30. Selected Analog Computer Symbols

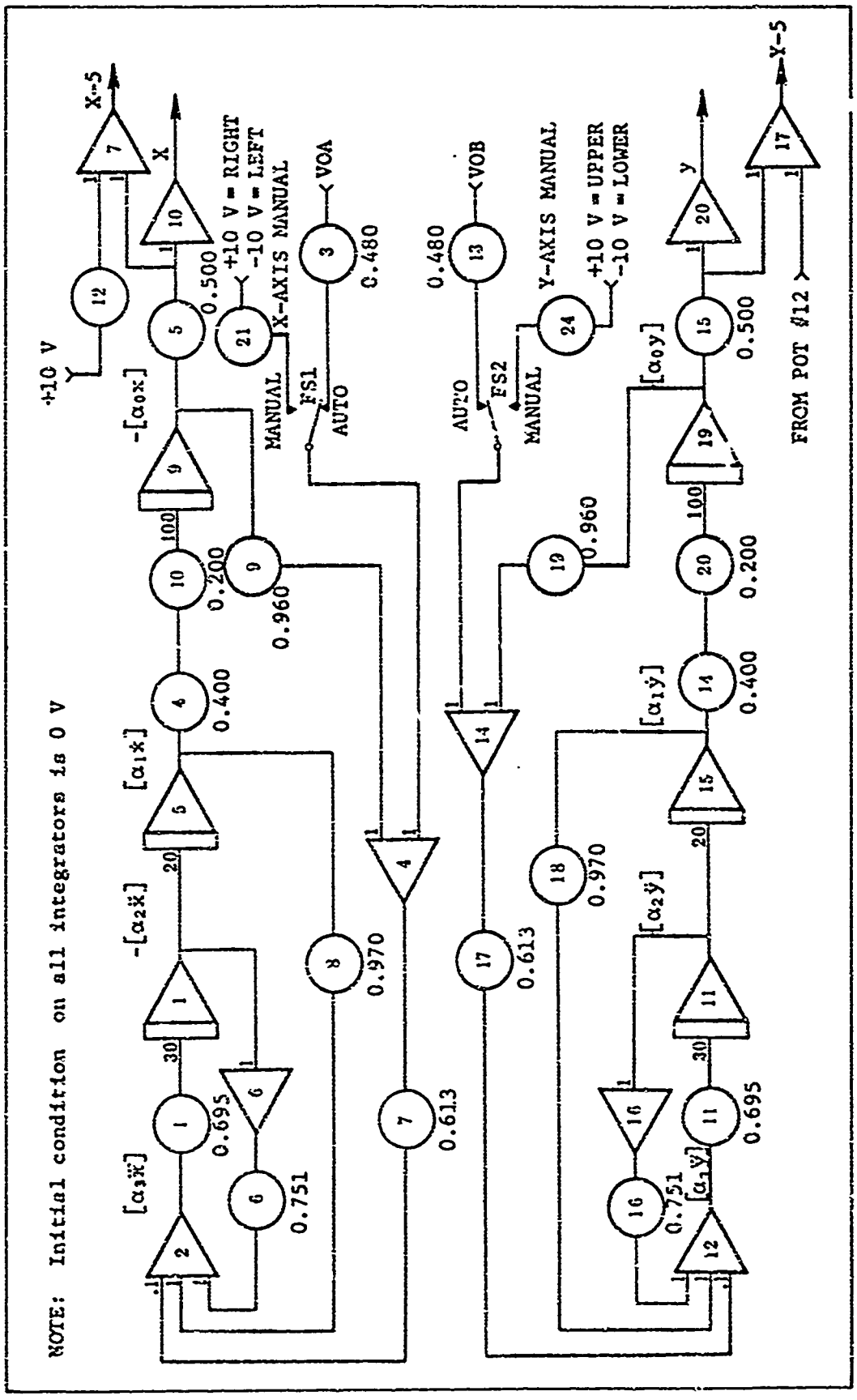
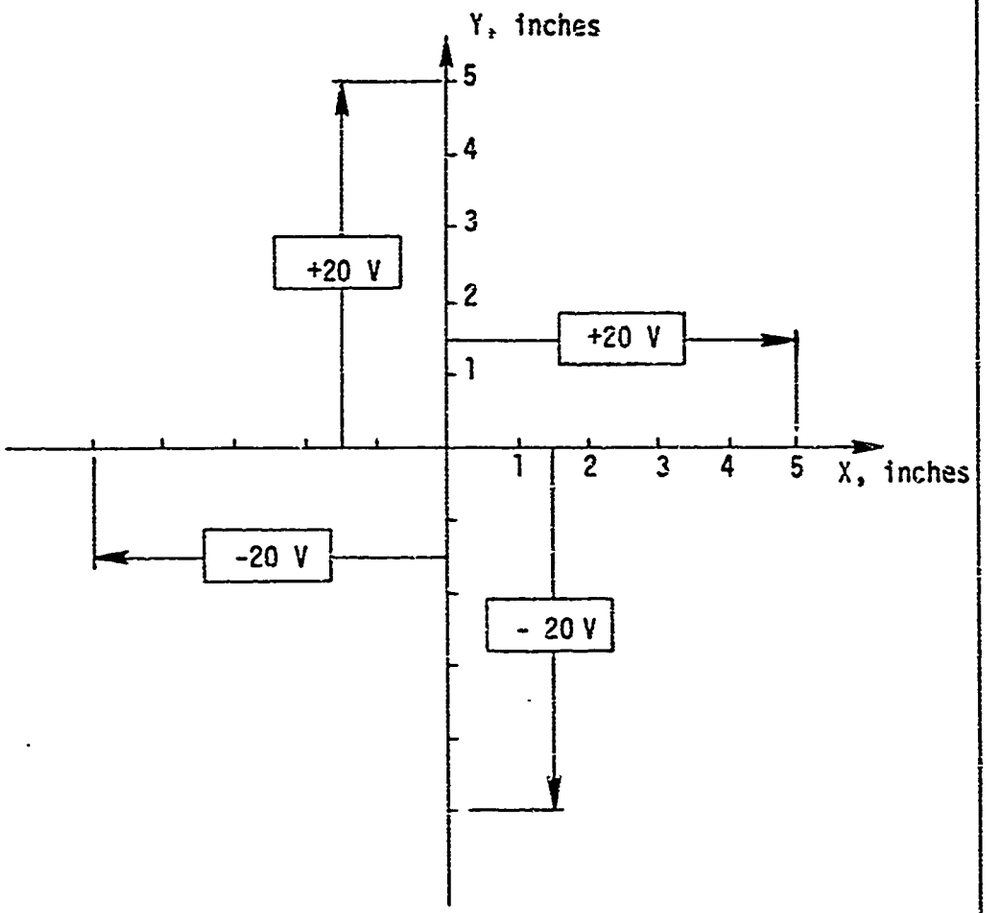


Fig. 31. Analog Computer Simulation of Machine Positioning System



NOTE: D17B output voltages required to give full-scale deflections in each direction are given in the squares.

Fig. 32. Coordinate System of Simulated Positioning System

of the positioning device. In both axes voltage outputs from the D17B will produce proportional deflections along the X and Y axes.

The initial programs produced a square in the first quadrant and a rough approximation of a circle. More sophisticated programs were developed which would generate a more precise circle and a square using either PTP or incremental positioning. These programs are presented in Appendices C and D. The algorithm used to generate the circle is also contained in Appendix D. These programs were used at first to control an X-Y plotter directly. At times the voltage outputs were displayed on an oscilloscope as a means of viewing the results of the program. In this way it could be determined whether the proper voltages were generated.

After the programs were tested in this manner for proper sequencing, the outputs of the computer were applied to the analog computer. This provided a test to determine if the programs developed the appropriate patterns at a proper speed, i.e., were the patterns generated slowly enough to be followed by the analog computer model of the machine positioning system?

Finally, feedback was provided to the computer so that the computer would have information as to whether the system responded to its commands or not. This feedback was provided by using analog-to-digital converters (ADC's). The converters used were Analog Devices, model ADC-12QZ. The specifications of these devices are given in Table XI.

The feedback system utilizing the ADC's is shown in Fig. 33. In the initial attempts feedback to the computer was used only to determine that the positioning system had responded completely to the previous command before the subsequent command was generated. The reason for this

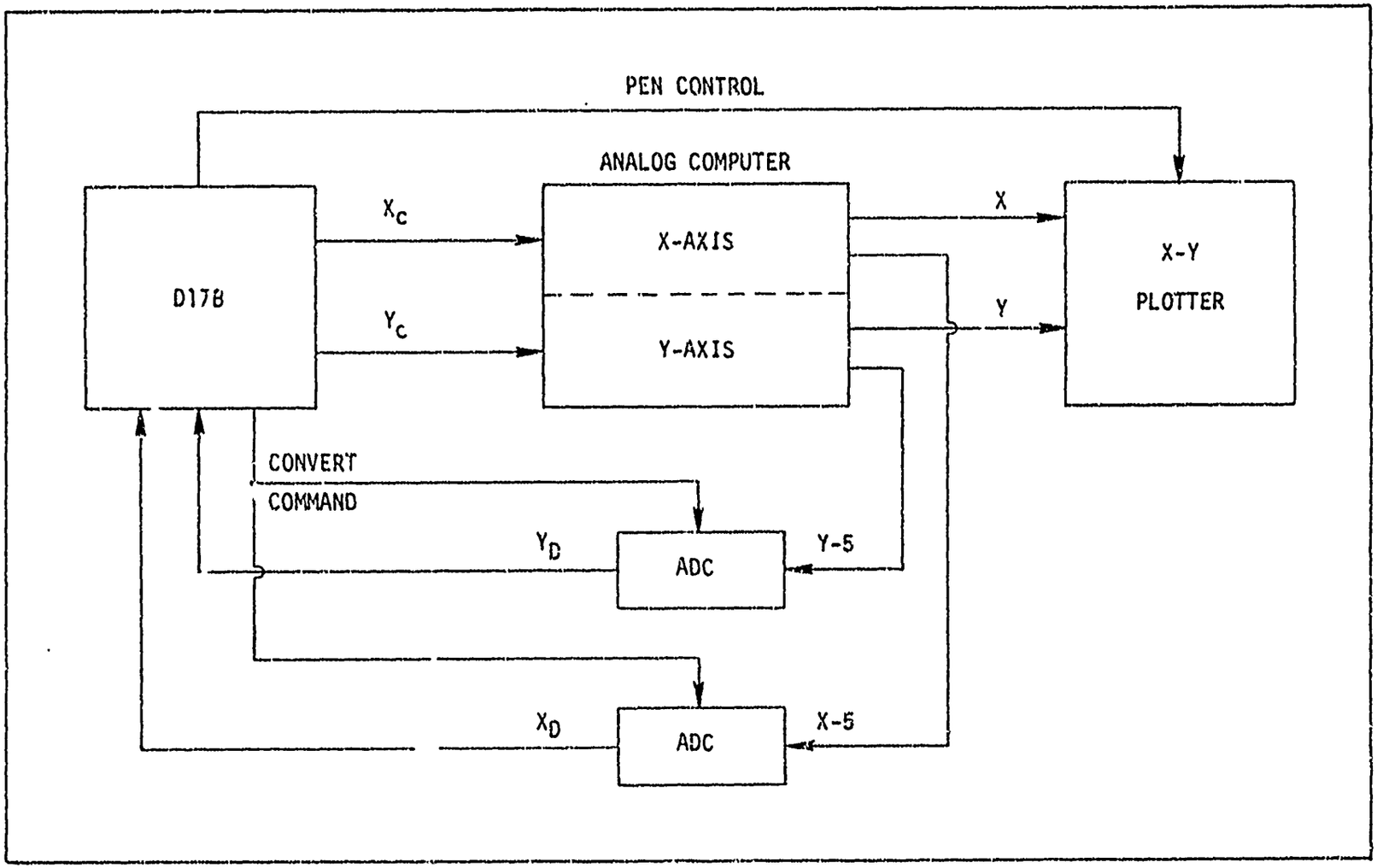


Fig. 33. Computer Positioning Control Incorporating Digital Feedback, Functional Diagram

TABLE XI

Electrical Specifications of ADC-12QZ Analog-to-Digital Converters

Resolution	12 bits
Accuracy	±1/2 LSB max
Conversion time	40 μsec, max
Input voltage ranges	±5 V, ±10 V, 0 to +5 V, 0 to +10 V
Input impedance	
10 V pp ranges	5 kΩ
20 V pp range	10 kΩ
0 to +5 V range	2.5 kΩ
Convert command	Positive pulse, 100 nsec min width; rise and fall time 1 μsec max
Output codes (positive true)	
Unipolar	
Parallel or serial	Binary
Bipolar	
Parallel	Offset binary or 2's complement
Serial	Offset binary
Logic outputs	
Status	"1" during conversion
$\overline{\text{Status}}$	"0" during conversion
Strobe	Serial data synch
Power supply	
Analog	±15 V @ 50 MA
Digital	+ 5 V @ 210 MA

sort of positive positioning control was to ensure that the positioning device went to the correct position before performing the work (such as drilling a hole) which was to be done at that location.

The next attempt at computer control involved using the D17B in the feedback path as the comparator in addition to providing the position commands to the positioning system. In this configuration the present position of the analog computer model was compared with the commanded position by the D17B and the position command signals were altered in accordance with the error signal. This corresponded to normal comparison of position information in analog positioning systems with the exception that it was done by digital means, with the aid of analog-to-digital converters and digital-to-analog converters.

Sometimes a sample-and-hold device is needed at the input of an ADC to keep the signal from changing during the conversion process. A sample-and-hold device is one for which the input/output relationship is $X_{out}(t) = X_{in}(kT)$, for $kT \leq t < (k + 1)T$, where T is the sampling interval and k is an integer.

In this application such a device is not necessary because of the low response of the signals being converted and the fast (40 μsec) conversion time of the ADC. The output of the converter was automatically held by the register wihin the encoder.

The hookup of the ADC's into the overall system is shown in Fig. 34 for the X-axis. The Y-axis hookup is identical. It can be seen that a one-shot multivibrator was used to generate a convert command pulse of seven μsec duration. Once the discrete output, DO3, is generated, it must remain true for one full wordtime (78 μsec). This would extend

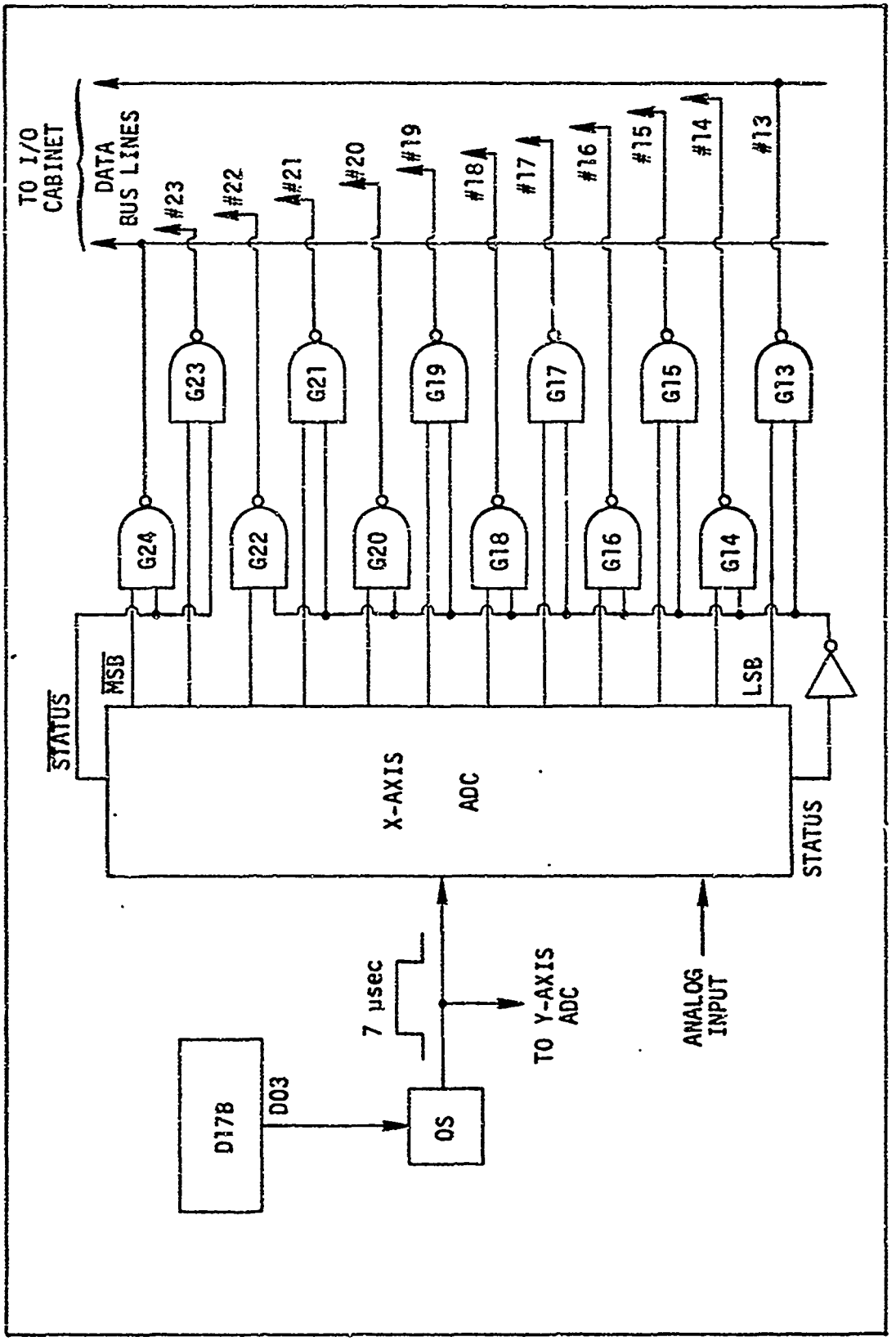


Fig. 34. ADC Hookup

beyond the 40 μsec conversion interval. The one-shot was used to avoid inadvertent subsequent triggering of the ADC. A very simple method was used to enable the NAND gates which buffered the digital output onto the data bus. The $\overline{\text{STATUS}}$ output was used to enable two of the gates directly while the remainder of the gates were driven by the inverted STATUS output since the fanout of the $\overline{\text{STATUS}}$ and STATUS signals was only five. The X and Y converter channels were treated identically except that the X outputs were put onto the 12 most significant positions of the data bus and the Y outputs were put onto the 12 least significant positions.

At this point it can also be noted that the analog inputs to the encoders are X-5 and Y-5, respectively. This was necessary due to the offset voltages used in the logic circuits to be compatible with the existing I/O cabinet and Control Console (Ref 29). The difficulty arises from the fact that the logic in the existing consoles uses 0 volts for V_{cc} on the logic modules and -5 volts for the normal ground input to the modules. These same voltages were used in the encoder logic and the analog input to the encoder had to be tied to the digital ground (actually -5 volts). Therefore, the analog output of the model had to be shifted by -5 volts to compensate for the shift in logic levels. The ground on the analog computer could not be tied to the -5 volt digital reference since the analog outputs of the computer were referenced to 0 volts.

Motomatic Hookup

As mentioned previously, the Motomatic system is a small armature-controlled dc motor which comes in modular form so as to facilitate unique connections for specific applications. This system allowed a demonstration of a "real-world" control situation with the D17B.

Essentially, the Motomatic is the same system as modeled on the analog computer except on a smaller scale. In addition the rotary shaft motion was not converted into linear motion as in the modelled system. A potentiometer connected directly to the output shaft fed back angular position information for closed-loop control.

Through standard testing procedures (Ref 12:660), the closed loop transfer function was found to be

$$\frac{\text{output angle}}{\text{input voltage}} = \frac{\theta}{V} = \frac{281}{(S + 7.5)(S + 37.5))}$$

Results for controlling the Motomatic are given in Chapter IV. The real-world application simulated was that of changing the angular position of the worktable so that a different side of the material could be shaped, stamped, etc.

IV. Results of Control Implementation

As was originally assumed, it was possible to exert control over both a modelled system and a "real-world" system (the Motomatic trainer) using the D17B computer. This chapter summarizes the results of implementing control with the D17B using three separate methods. Method A used point-to-point control to generate a square. Method B improved upon Method A in that the complete path was under control and an auxiliary work function was added. Method C implemented a closed-loop within the computer.

Method A, Point-to-Point Control

Method A was the simplest of the three methods used. Under this method the D17B was caused to compute and output an analog voltage to the modelled machine positioning system (MPS). It would then wait in a loop for a period of time required for the MPS to reach the commanded position. The MPS moved the simulated slide of a machine tool a distance proportional to the input voltage without any further assistance from the D17B. This method allowed the computer to control the MPS in a point-to-point (PTP) manner. The distance between points could be varied from zero to the maximum dimension of the MPS worktable. The restriction on this method is that the D17B must be caused to wait in a delay loop long enough for the MPS to respond to the input. The upper square in Fig. 35 was drawn on an x-y plotter using the X and Y voltage outputs of the MPS as inputs to the plotter.

Method B, Continuous Path Control

Method B added a degree of sophistication to the computer control process. Figure 35 shows a slightly modified square. Inside the square

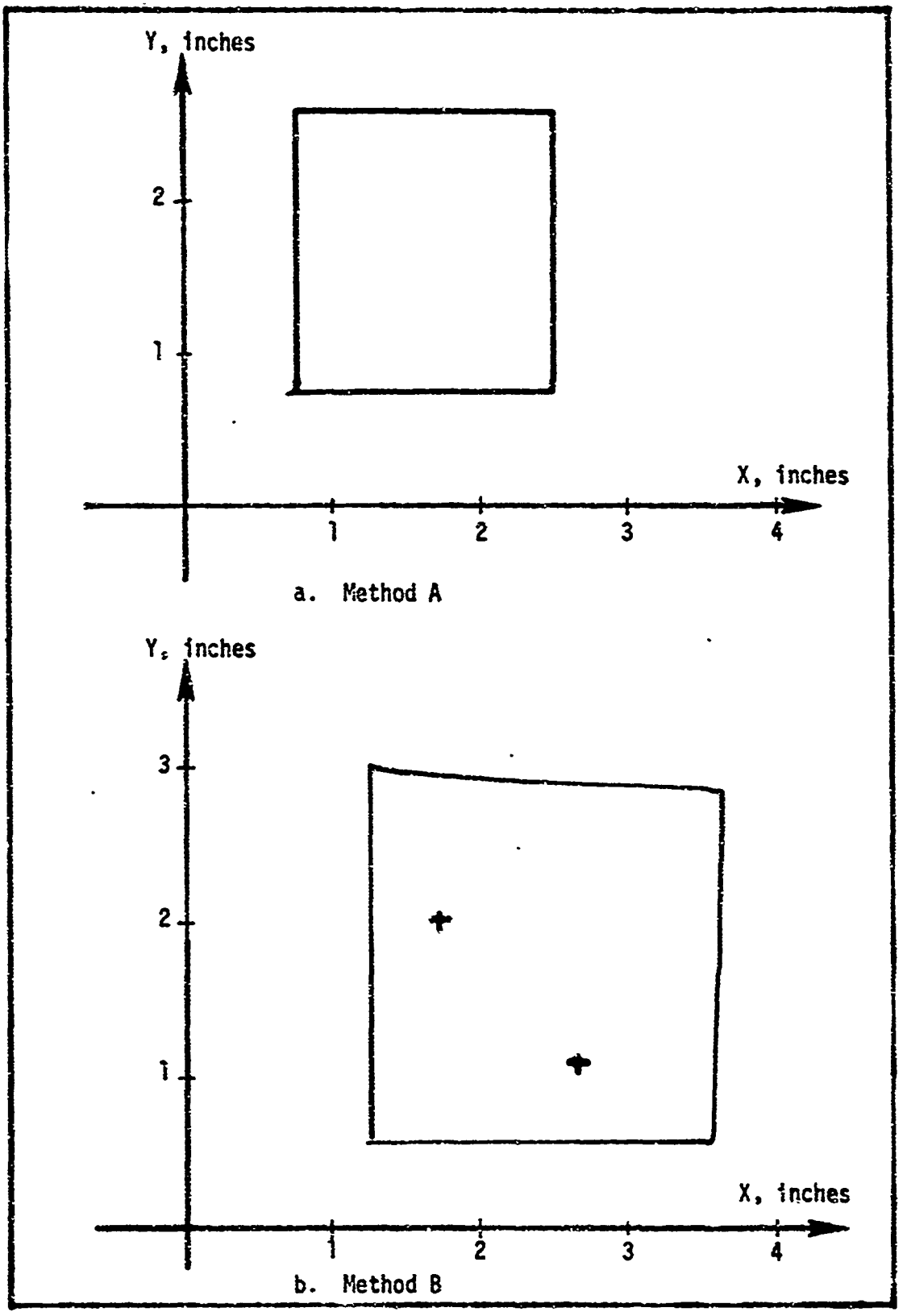


Fig. 35. Square Figures Generated by Methods A and B

are two hash marks indicating holes that might have been drilled at the prescribed locations as the MPS stopped in that position. The hash marks were also generated by the computer, which had to control the lifting of the pen on the plotter as well as implementing position commands.

In this method the computer knew at all times where the machine was positioned due to the feedback incorporated for this purpose. When the MPS was within a specified range of the desired value, the D17B calculated a new value to reposition the MPS. This method allowed faster control of the MPS since calculation of a new value could begin as soon as the MPS was close enough to the desired position. In method A the computer was required to wait a given length of time between each command.

Method B will not be as accurate as method A unless the MPS is allowed to complete the positioning before a new signal is generated. The tolerance specified in method B is dependent on the accuracy of the D/A and A/D converters, the stability of the D17B output voltages, and the capability of the MPS.

Figure 36 shows several circular arcs of arbitrary radii and arbitrary central angles. The coordinates of the points on these arcs were generated by the circular interpolation program listed in Appendix D. The plots were obtained as in method A, with the x-y plotter receiving the X and Y voltage outputs of the MPS. The initial and final points of the arcs were input to the program manually for each arc. A limitation on this arc-generation routine is that the coordinates of the initial and final points must be precalculated and be consistent points on a circle about the origin. A suitable modification in the program could undoubtedly be made to cause the circular arcs to be drawn around an arbitrary center.

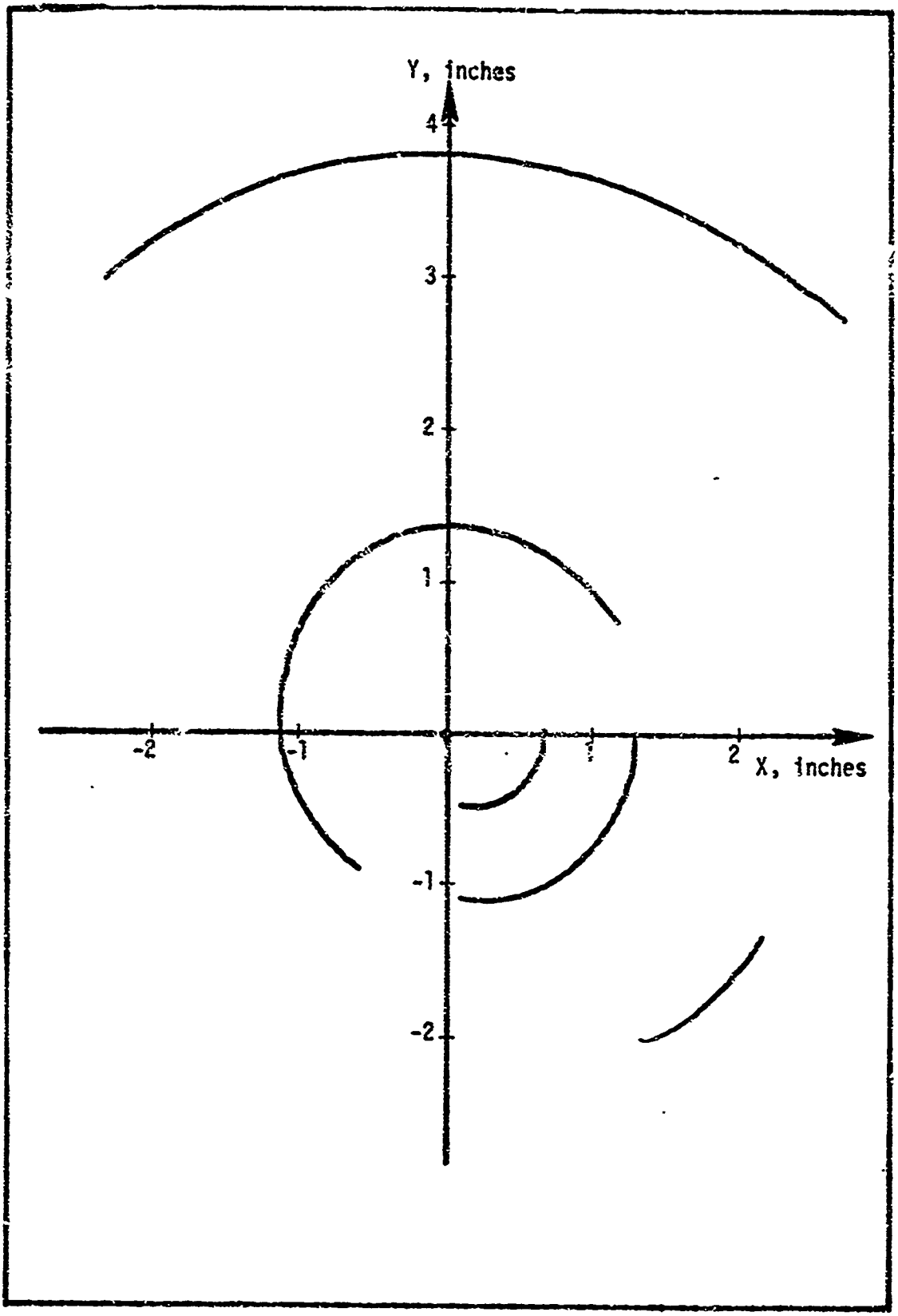


Fig. 36. Circular Arcs Generated by Method B

Method C, Closed-Loop Control

Method C is more complex than method B since the D17B was actively involved in a feedback loop. In this method the D17B checked the position of the MPS, compared it to the desired value and applied an error signal based on the difference between the desired and the actual positions. As in method B, when the position of the MPS came within a certain specified range of the desired value, the D17B calculated the next position and repeated the process.

Figure 37 shows, in quadrants one and four, an approximately semicircular arc generated by method C. On the left is an arc generated by method B.

Figure 38 shows a plot of the angular position of the shaft of the Motomatic servo system as a function of time. The position commands were generated under program control in such a manner as to effect step changes in the angular position of the motor. This procedure would correspond to the rotation of a machine tool worktable about its z-axis, holding in a given position for a length of time necessary to accomplish the desired work function, and then proceeding to a different angular position for a new task.

Error Analysis

Since the primary goal of this study was to determine the feasibility of implementing computerized control with the D17B computer rather than to design a complete, accurate control system, there were several sources of error present which were not corrected. In any practical system, these errors would have to be eliminated or compensated for in order to obtain an acceptable degree of accuracy. The sources of these

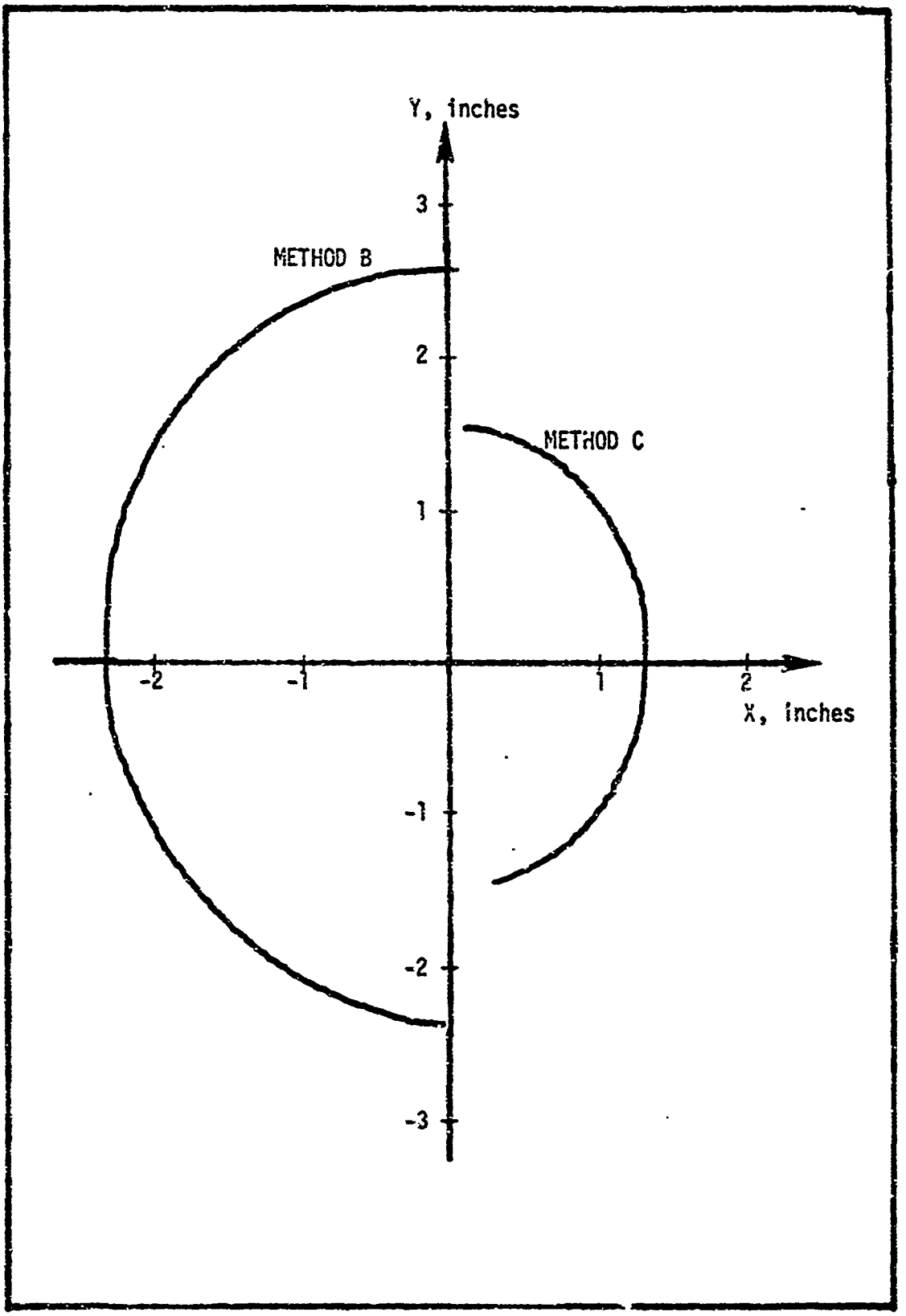


Fig. 37. Comparison of Arcs Generated by Methods B and C

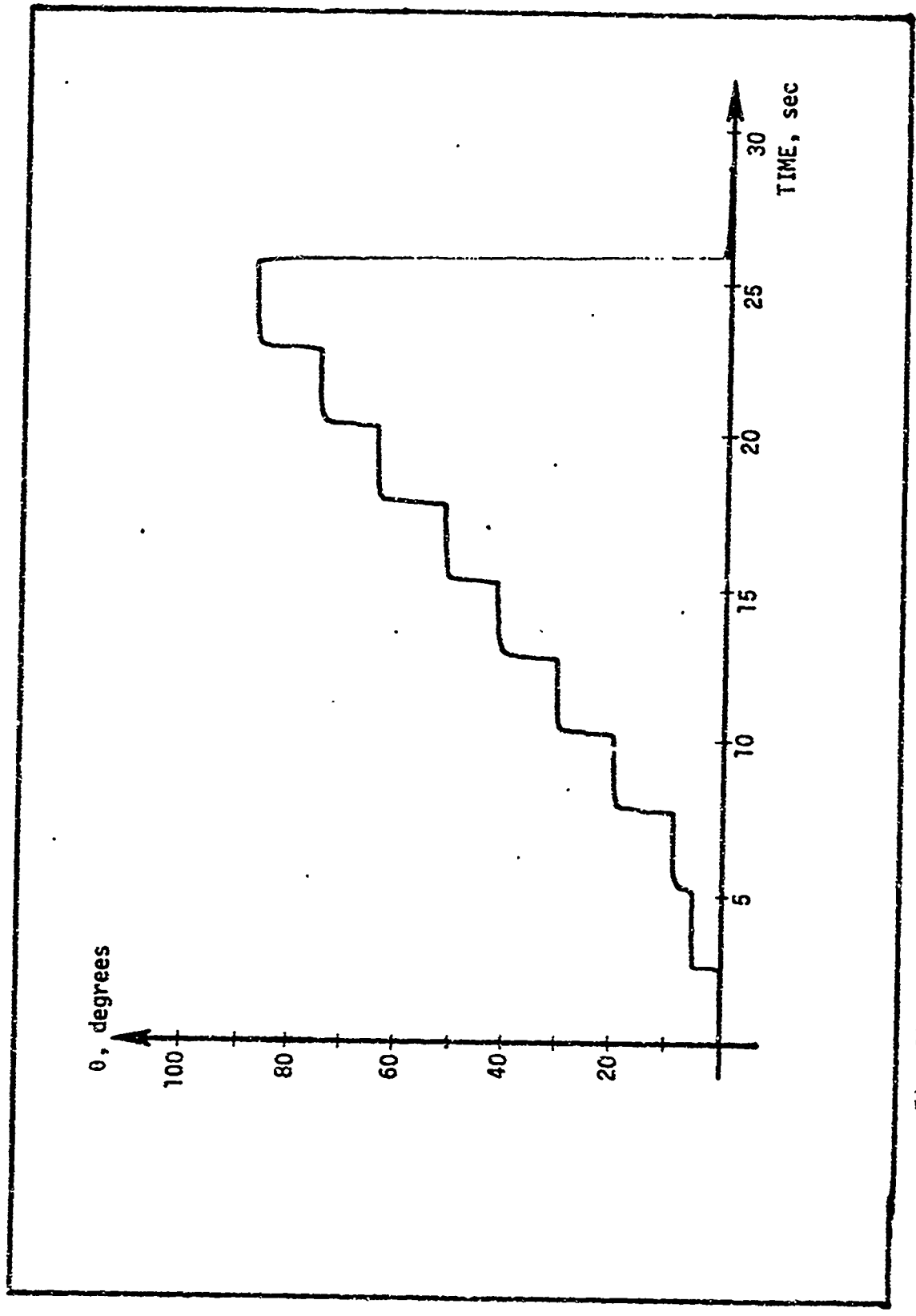


Fig. 38. Plot of Angular Position Changes Implemented by Method C

errors were the analog computer, the D17B, the A/D converters, the X-Y plotter, and the software.

During prolonged periods of operation, the analog computer tends to drift, causing variations in analog computer outputs. Similarly, the questionable accuracy of the X-Y plotter could have caused variations in the plots made of the system performance.

In the I/O interface developed at AFIT, the voltage output lines were unshielded. Because of this and the proximity of the voltage lines to many of the control signals in the control console, some noise was developed on the voltage lines. The inaccuracies in the analog signals were further compounded by a lack of calibration of the ADC's. These were not calibrated because a highly accurate voltmeter necessary for the calibration was not available.

The programs used in the demonstration of computer control were not completely debugged in all cases. The time required to completely debug programs written in machine language would not have been worth the added accuracy since the overall objectives were attained in most cases. In some instances error terms wre added to the programs to compensate for inaccuracies in the analog signals.

Although the above errors were not corrected in our demonstrations, they would have to be corrected in any practical control application.

V. D17B Software Analysis

Software for the D17B computer is practically non-existant. The Minuteman Computer Users Group (MCUG) has published a programming manual for the D17B and there are individuals in the MCUG who are writing assemblers, loaders, and subroutines for use with the D17B. Although software is being developed, there is no organization to this development and dissemination of results is poor. Most of the software developed takes advantage of the interfaces developed at each facility that has a D17B. Thus an assembler developed at AFIT might not be useful at another location because the I/O interfaces differ considerably.

Programming Development

To develop an efficient program for execution on the D17B requires a knowledge of the special features inherent in the D17B. There are several programming features which make programming the D17B computer different from other general-purpose computers.

First, there is no assembler or compiler available for the D17B so all programming has to be in machine language. Secondly, computer operation is linked to a disk memory which necessitates very careful coding of a program to insure the minimum time between instructions and between an instruction and the data it operates on. Next, there is the "flag-store" operation which permits simultaneously storing the previous contents of the Accumulator in a specified location coincident with the execution of an instruction. Finally the D17B can add, subtract, multiply, or shift two 11-bit split-words simultaneously. This aspect provides the possibility

of performing operations effectively twice as fast as normally possible if it is desired to operate on two data sets in an identical manner.

Programming Techniques

Programming is covered in detail in Ref 3, which also covers some programming techniques not covered here. One technique to consider when programming large programs is to store the instructions in the cold channels (00-46) and enter data in the hot channel (50) and the rapid-access loops. Flag-storing can be used in both channel 50 and the rapid-access loops. By storing the program in the cold channels, the program can be protected from accidental overwriting by turning the ENABLE WRITE control signal off.

Another possibility is to store small Do-loops in the rapid-access loops. For example, suppose a 15-word Do-loop was stored in the H-loop (16-word loop). For every disk revolution there would be eight and one half (126 words divided by 16) iterations of the Do-loop. If this same subroutine were stored in any channel from 00 to 50, one disk revolution would result in one iteration of the Do-loop.

Programming in Numerical Control Applications

Numerous programming languages have been developed for numerical control applications such as APT, ADAPT, and AUTOSPOT as mentioned in Chapter I (also Ref 26). These programs have been written for medium-to-large-scale computers and are extremely useful in numerical control.

Post-Processor. When an APT or similar program is used, the output must be processed by another program called a post-processor, which processes the data for an individual N/C machine. Thus, if the D17B computer were to be used with an N/C machine, a post-processor program would have

to be written for the large-scale computer which uses the APT program. Another possibility is to write a post-processor program for the D17B and execute it on the D17B. This way the D17B could do its own post-processing and free the large-scale computer for other work. A different post-processor program would have to be written for each N/C machine controlled.

Possible Configurations. Three possible configurations involving the D17B are considered in this section. The first configuration uses the computer as a controller as shown in Fig. 39.

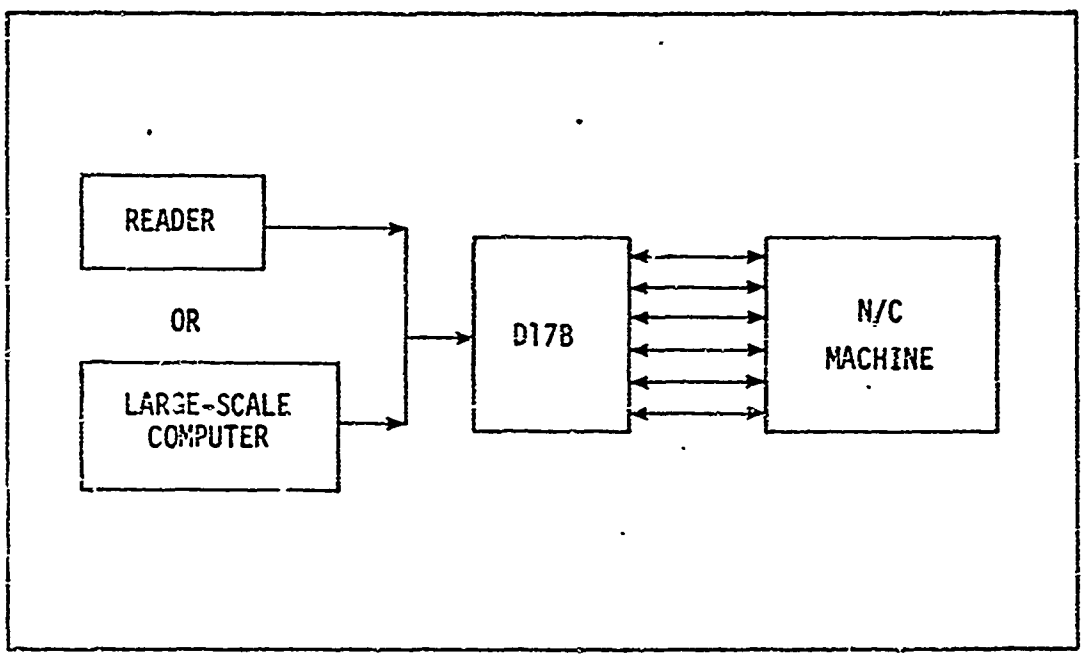


Fig. 39. D17B as a Controller

The reader could be a magnetic tape unit or a punched-tape reader for all three configurations. The D17B would receive a block of the program from the reader or another computer, store the program, and then execute it. This cycle would be continued until the program was completed. The D17B would only receive part of the program at a time because most of the memory would contain subroutines used to execute the incoming instructions.

It is doubtful that this configuration would work with the D17B because of the limited memory capacity and the slow speed of the computer. However the D37C's larger memory capacity would make it a more likely candidate for this application. This configuration is similar to method C as discussed in Chapter IV.

The second configuration, called supervisory control, is shown in Fig. 40. The program is read into and stored in the D17B and then the computer outputs blocks of information to the controller. The only

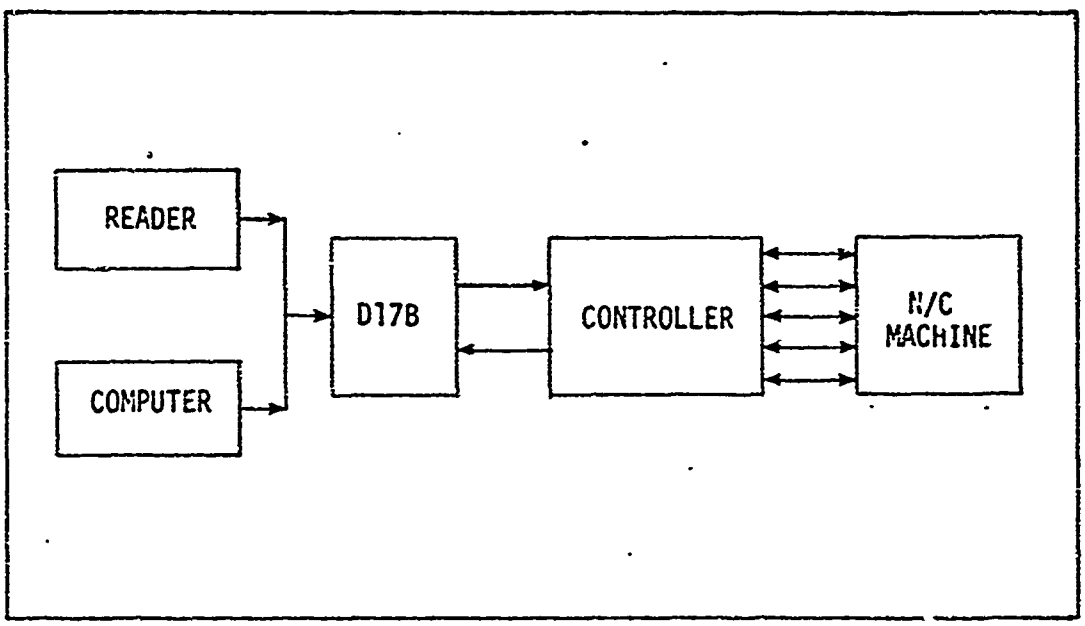


Fig. 40. Supervisory Control

difference between this configuration and a third configuration is that the third one could have another program stored in memory along with a small supervisory program. This would allow the computer to direct the N/C machine to drill 10 circuit boards of type A, then 5 circuit boards of type B, then back to type A boards and so on. Thus only a program to drill one circuit board of type A and one of type B would have to be entered along with the numbers 10 and 5 and the computer could continue the sequence indefinitely.

The third configuration, as mentioned previously, is the same as the second except for the added supervisory control. When compared to conventional numerical control, the third configuration is advantageous only if the same program is to be repeated. In this case the reader (usually a punched-tape reader) is eliminated from the reruns, thus bypassing the most susceptible part of the N/C system (Ref 9:35). If a program is to be executed only once, the third configuration merely adds to the data stream another link which does nothing. The computer would be helpful if a second execution with changed parameters was desired. In this case only a few constants would have to be changed and a new tape would not have to be made.

VI. Conclusions and Recommendations

The D17B and D37C digital computers are sturdy, reliable computers. These two characteristics would seem to be important considerations in process and numerical control applications. The previous use of the D17B and D37C as guidance and control computers in ICBM's attests to their worthiness in these respects. However, software development and documentation is extremely lacking and presents the biggest and most difficult problem to solve in implementing the D17B and D37C computers in control applications.

A detailed investigation of the software capabilities and the implementation of these abilities would be necessary before these computers could be used in controlling a process or machine. The instruction sets of the two computers are specialized for a navigational guidance and control application. These specializations should be usable in other control applications and should be investigated before using them for control purposes.

The control demonstrations presented in Chapter IV did not use these specialized instructions such as fine countdown, split-word shifts, and modified multiplication.

An investigation should be made into using these computers in control systems which use electric or electro-hydraulic stepping motors. These motors are limited to about 5000 pulses-per-second (Ref 8:82-88). This is within the theoretical capability of either computer. Theoretically one pulse could be outputted every word time or equivalently--126,000 pulses-per-second.

Our experiments with the D17B computer indicated that the computer provided better control in the method A or method B configuration. (See Chapter 4.) Methods A and B represented supervisory control in which the computer is not the feedback element, but provides the set-point for the analog controller (Ref 20:9). Thus error correction is relegated to the analog controller while overall control is maintained by the computer.

The difficulty with both of these methods is the lack of software. Programs would have to be developed for each application. Another problem would be that of interfacing the D17B to an N/C machine since the D17B uses negative logic and generally has -10 volts for a logical one and -1 volt for a logical zero. The D37C does not present this problem since its outputs are generally compatible with standard TTL logic.

A final possibility would be to put the computer in the data stream between the tape reader and the controller. In this configuration, programs could be stored in the computer and rerun as often as desired--eliminating the most susceptible item in N/C systems, the tape reader, from the reruns (Ref 10:35). The tape reader should only be used to read the program in initially and not used in the remainder of the reruns. This would also allow set-points to be changed by just entering a few points into the computer without repunching a tape.

These computers could be used in a data acquisition system (DAS) in a process control system. The primary functions of DAS are monitoring, alarm checking, and data logging. The computer could scan a set of analog inputs such as flow, temperature, position, weight, or pressure; check to see if they were within limits; and log certain values periodically (or after an alarm was issued). The hardware requirements would include analog and digital inputs; digital outputs to drive lights and other

annunciators; and analog outputs for visual display and strip charts (Ref 20:8). The D37C has all of these features while the D17B is only missing the voltage inputs. Appendix E summarizes the characteristics of the D17B, D37C, and a selected group of minicomputers.

A system with approximately 50 input points, no data reduction required, a few conversion subroutines to convert data to engineering units, and an output subroutine can be programmed on a computer with 8K of core and a 16-bit word length. This is about equal to the D37C capability, but the D17B could probably only handle about 20 points based on its smaller memory.

The D17B and D37C computers could be used in control applications, but software limitations, interface problems, slow speed, and limited memory restrict these computers to specialized applications and a feasibility study might be required for each application.

It is recommended that more extensive investigation into software for these computers be done before further attempts in control applications are tried. Because the specialized instruction sets and capabilities of these computers are not well-documented, we believe that an investigation which would explore the instruction sets, and explain methods of utilizing instructions such as split compare and limit, multiply, modified, and the fine countdown procedures would benefit any future applications-oriented study.

Bibliography

1. Allen, D. J. Laboratory Conversion and State Description of the D-17B Computer, GE/EE/72s-2. Unpublished Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, June 1972.

2. Autonetics. Minuteman D-17 Computer Training Data. Anaheim, California: Autonetics, Division of North American Rockwell, Inc., January 1960.

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

4. _____. Investigation of Minuteman D17B Computer Reutilization. New Orleans, Louisiana: Tulane University, January 1971.

5. Bergren, C. "A Simple Algorithm for Circular Interpolation." Control Engineering, 18:57-59 (September 1971).

6. Bernard, J. W. "Plan Control at the Right Level." Control Engineering, 13:95-98 (September 1966).

7. Booth, T. L. Digital Networks and Computer Systems. New York: John Wiley and Sons, Inc., 1971.

8. Budzilovich, P. N. "Use Electrohydraulic Stepping Motors for All-Digital Drives." Control Engineering, 17:82 (January 1970).

9. Cadzow, J. A. and H. R. Martens. Discrete Time and Computer Control Systems. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1970.

10. Clauss, F. J. and R. M. McKay. "Total Manufacturing Control." Automation, 18:34-40 (January 1971).

11. Danzer, R. F. and C. J. Kishel. Introduction to Numerical Control in Manufacturing. Dearborn, Michigan: American Society of Tool and Manufacturing Engineers, 1969.

12. D'Azzo, J. J. and C. H. Houpis. Feedback Control System Analysis and Synthesis. New York: McGraw-Hill, 1966.

13. Eleccion, M. "A/D and D/A Converters" IEEE Spectrum, Vol 9, No. 7:63-66 (July 1972).

14. Electronic Associates, Inc., Pace TR-10 Transistorized Analog Computer Handbook. Electronic Associates, Inc., Long Branch, New Jersey.

Bibliography

15. Eppich, S. M. Numerically-Controlled Manufacturing Equipment for USAF Depot Activities. Unpublished Thesis. Logan, Utah: Utah State University, 1971.

16. Gear, C. W. Computer Organization and Programming. New York: McGraw-Hill Book Company, 1969.

17. George, F. O. User's Manual for a Digital Computer Routine to Calculate Root Loci, AFFDL FDC TM 69-4. Unpublished report, Wright-Patterson Air Force Base, Ohio: Control Criteria Branch, Air Force Flight Dynamics Laboratory, December, 1969.

18. Hansen, D. D. and K R. Watkins. A Rigorous Logical Study-With-Lab of the D17 Digital Computer. ACC-31170P-33. Computers and Data Systems Logistics, Customer Training, 30 April 1962.

19. Henson, D. D. A Relocatable Assembler and Associated Loader for The D17B Minuteman Guidance Computer, GE/EE/72-12. Unpublished Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, March 1973.

20. Henzel, R. A. "Some Industrial Applications of Minicomputers." Computer, 4:7-21 (September/October 1971).

21. Jurgen, R. K. "Minicomputer Applications in the Seventies" in A Practical Guide to Minicomputer Applications. Edited by F. F. Coury, IEEE Press, New York, 1972.

22. Kuo, B. C. Discrete-Data Control Systems. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1970.

23. Kusko, A. Solid-State DC Motor Drives. Cambridge, Mass.: The M.I.T. Press, 1969.

24. Lapidus, G. "A Look at Minicomputer Applications" in A Practical Guide to Minicomputer Applications. Edited by F. F. Coury, IEEE Press, New York, 1972.

25. Massachusetts Institute of Technology. A Numerically Controlled Milling Machine, Servo Mechanisms Lab, MIT, Report to Air Force on Contract AF33(038)-24007.

26. Prentice, R. C. and A. D. Roberts. Programming for Numerical Control Machines. New York: McGraw-Hill Book Company, 1968.

27. Ryan, F. M. "Computers in the Plant." Control Engineering, Vol. 13, No. 9:84-86 (September 1966).

Bibliography

28. Shearer, J. L. Introduction to System Dynamics. Reading, Mass.: Addison-Wesley Publishing Co., 1967.

29. Theriault, J. Design Expansion of the D17B Computer Input/Output Facility for General-Purpose Applications, GE/EL/73-20. Unpublished Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, March 1973.

APPENDIX A

D37C Computer

The D37C computer is essentially a micro-miniaturized version of the D17B computer with extended capabilities. The basic extensions are increased memory capacity (about 2.6 times); a more varied instruction set (58 vs. 39 instructions); and the use of integrated circuits. The latter resulted in a much smaller physical package than the D17B computer. Table A-I lists the general specifications of the D37C digital computer. A similar listing for the D17B is contained in Table I, page 13. The difference in programmable memory capacity is illustrated in Table A-II.

Although integrated circuits are used, the D37C is not any faster than the D17B since both computers are limited by the magnetic disk memory. This disk rotates at 6000 rpm which results in word-time or cycle-time of 78-1/8 μsec for both computers.

The instruction set for the D37C includes 27 instructions from the D17B instruction set plus 32 new instructions for a total of 59. The D37C instruction set is listed in Table A-III. Twelve of the D17B instruction set were deleted, but six of these instructions are included in other instructions in the D37C. The remaining six instructions can be implemented by combinations of other instructions. Therefore, the D37C can perform all the operations that the D17B can; thus any functions the D17B can perform can be extended to the D37C computer.

TABLE A-I

General Specifications of D37C Computer

Manufacturer	Autonetics
Year	1964
Type	Serial, Synchronous
Number system	Binary, fixed point, sign plus 2's complement
Logic levels	False (0 volts), True (6 volts), positive logic
Data word length	24 bits (full word) 11 bits (split word)
Instruction word length	24 bits
Number of instructions	58
Execution times	
Add	78.125 μsec
Multiply	1015.625 μsec
Divide	2031.250 μsec
Clock frequency	345.6 kHz
Addressing	Direct addressing Two-address (unflagged) Three-address (flagged)
Memory	Ferrous-oxide coated disk Non-destructive readout 7222 (24 bit) word capacity 78.125 μsec cycle time
Input/Output	78 digital lines (input) 10 specialized incremental inputs 32 voltage input lines 1 cable input (maximum data rate 1600 bits/sec) 1 radio input (maximum data rate 100 bits/sec) 48 digital lines (output) 4 binary lines (output) 1 cable output 4 voltage output lines 25,600 words/sec maximum I/O transfer rate
Physical characteristics	
Dimensions	20.9×6.9×9.5 inches, 0.43 cu. ft.
Power Requirements	400 Hz, 28 VDC at 15 A
Circuits	Integrated circuits
Weight	39.9 lbs
Power Consumption	315 watts

TABLE A-II

Differences in Memory Capacities Between the D17B and the D37C

Name	Use	Channel D17B	Channel D37C	Words Per Channel
Main memory	General	27	56	128
H-loop	Rapid Access and Voltage Output[a]	1	1	16
E-loop	Rapid Access	1	1	8
F-loop	Rapid Access	1	1	4
V-loop	Velocity Input	1	1	4
R-loop	Incremental Anput	1	1	4
L-loop	Lower accumulator	1	1	1
A-loop	Accumulator	1	1	1
U-loop	Rapid Access, Fine Countdown, Cable Count[b]	1	1	1
G-loop	Incremental Output, Radio Input, Rapid Access	0	1	4
Y-loop	Conditional Load, Rapid Access	0	1	4
W-loop	Rapid Access	0	1	1

[a]Voltage output is not included in D17B

[b]Cable count is not included in D17B

TABLE A-III

Comparison of D37C and D17B Instruction Sets

Code	Description	Code	Description
	Basic Set Common to D17B and D37C		
TRA	Transfer	STO	Store Accumulator
TMI	Transfer on Minus	ADD	Add
HPR	Halt and Proceed	SAD	Split Word Add
SCL	Split Compare and Limit	SUB	Subtract
ANA	Logical AND to Accumulator	SSU	Split Word Subtract
LPR	Load Phase Register	MPY	Multiply
EFC	Enter Fine Countdown	SMP	Split Word Multiply
HFC	Halt Fine Countdown	COM	Complement
RSD	Reset Detector	MIM	Minus Magnitude
DIA	Discrete Input A	ALS	Accumulator Left Shift
DIB	Discrete Input B	ARS	Accumulator Right Shift
DOA	Discrete Output A	SAL	Split Accumulator Left Shift
COA	Character Output A	SAR	Split Accumulator Right Shift
CLA	Clear and Add		
	Added Instructions D37C Only		
DIV	Divide	SPM	Split Plus Magnitude
GBP	Generate Bit Pattern	FCL	Full Compare and Limit
GPT	Generate Parity Bit	ALC	Accumulator Left Cycle
AWC	Add Without Carry	ARC	Accumulator Right Cycle
ORA	Or to Accumulator	TZE	Transfer on Zero
MAL	Modify A and L	TSM	Trannsfer Sector on Minus
PLM	Plus Magnitude	TSZ	Transfer Sector on Zero
SFL	Set FL Flip-Flop	VIA	Voltage Input A
RFL	Reset FL Flip-Flop	VIB	Voltage Input B
SRD	Simulate Transient	VIC	Voltage Input C
DIC	Discrete Input C	VID	Voltage Input D
DOB	Discrete Output B	VIE	Voltage Input E

TABLE A-III (continued)

Code	Description	Code	Description
	Added Instructions D37C Only (cont'd)		
ECI	Enable Cable Input	VIF	Voltage Input F
ECO	Enable Cable Output	VIG	Voltage Input G
EPP	Enable Platform Power	VIH	Voltage Input H
DPP	Disable Platform Power	RIC	Radio Intercommunication
	Deleted Instructions D17B Only		
MPM	Multiply Modified	VOA	Voltage Output A
SMM	Split Word Multiply Modified	VOB	Voltage Output B
SLL	Split Left Word Left Shift	VOC	Voltage Output C
SLR	Split Left Word Right Shift	BOA	Binary Output A
SRL	Split Right Word Left Shift	BOB	Binary Output B
SRR	Split Right Word Right Shift	BOC	Binary Output C

APPENDIX B

TABLE B-I

D17B Computer Instruction Set

Code	Description	Numeric Code	Word Time
	ARITHMETIC		
CLA	CLEAR AND ADD	44 c,s	1
STO	STORE ACCUMULATOR	54 c,s	1
ADD	ADD	64 c,s	1
SAD	SPLIT WORD ADD	60 c,s	1
SUB	SUBTRACT	74 c,s	1
SSU	SPLIT WORD SUBTRACT	70 c,s	1
MPY	MULTIPLY	24 c,s	13
SMP	SPLIT WORD MULTIPLY	20 c,s	7
MPM	MULTIPLY MODIFIED	34 c,s	13
SMM	SPLIT WORD MULTIPLY MODIFIED	30 c,s	7
COM	COMPLEMENT	40 46,s	1
MIM	MINUS MAGNITUDE	40 44,s	1
	SHIFT		
ALS	ACCUMULATOR LEFT SHIFT	00 22,s	s+1
ARS	ACCUMULATOR RIGHT SHIFT	00 32,s	s+1
SAL	SPLIT ACCUMULATOR LEFT SHIFT	00 20,s	s+1
SAR	SPLIT ACCUMULATOR RIGHT SHIFT	00 30,s	s+1
SLL	SPLIT LEFT WORD LEFT SHIFT	00 24,s	s+1
SLR	SPLIT LEFT WORD RIGHT SHIFT	00 34,s	s+1
SRL	SPLIT RIGHT WORD LEFT SHIFT	00 26,s	s+1
SRR	SPLIT RIGHT WORD RIGHT SHIFT	00 36,s	s+1
	CONTROL		
TRA	TRANSFER	50 c,s	1
TMI	TRANSFER ON MINUS	10 c,s	1
HPR	HALT AND PROCEED	40 22,s	1
SCL	SPLIT COMPARE AND LIMIT	04 c,s	2
ANA	LOGICAL AND TO ACCUMULATOR	40 42,s	1
LPR	LOAD PHASE REGISTER	40 7-,s	1
EFC	ENTER FINE COUNTDOWN	40 62,s	1
HFC	HALT FINE COUNTDOWN	40 60,s	1
RSD	RESET DETECTOR	40 20,s	1

TABLE B-I (continued)

Code	Description	Numeric Code	Word Time
	INPUT/OUTPUT		
DIA	DISCRETE INPUT A	40 52,s	1
DIB	DISCRETE INPUT B	40 50,s	1
DOA	DISCRETE OUTPUT A	40 26,s	1
VOA	VOLTAGE OUTPUT A	40 30,s	1
VOB	VOLTAGE OUTPUT B	40 32,s	1
VOC	VOLTAGE OUTPUT C	40 34,s	1
BOA	BINARY OUTPUT A	40 10,s	1
BOB	BINARY OUTPUT B	40 12,s	1
BOC	BINARY OUTPUT C	40 02,s	1
COA	CHARACTER OUTPUT A	00 40,s	s+1

APPENDIX C

Square Generation Subroutine

This subroutine was written to generate a square by outputting the voltage values for the corners of the square to the positioning system simulated on the TR-10 analog computer. The program caused the

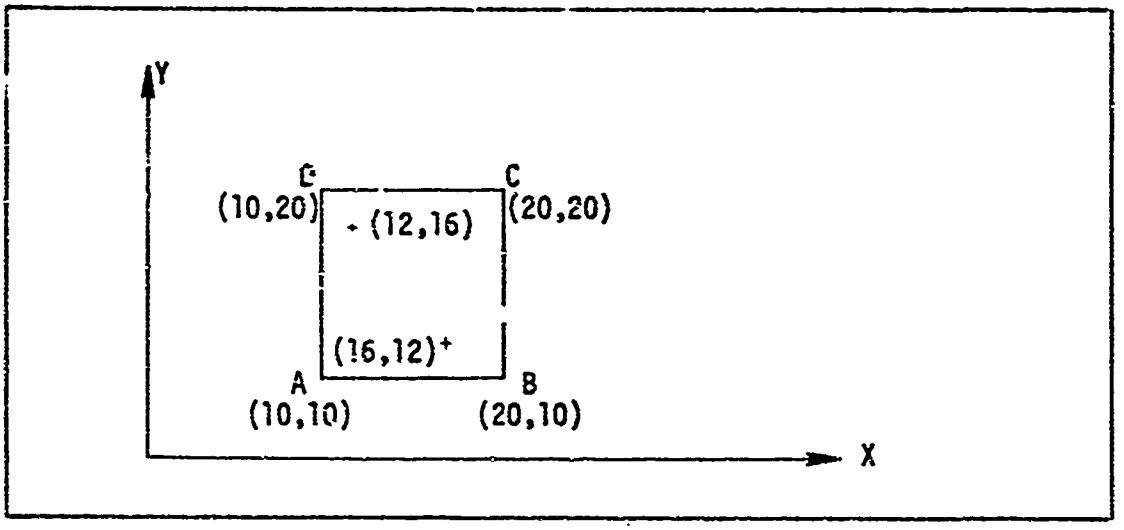


Fig. C-1. Square to be Generated

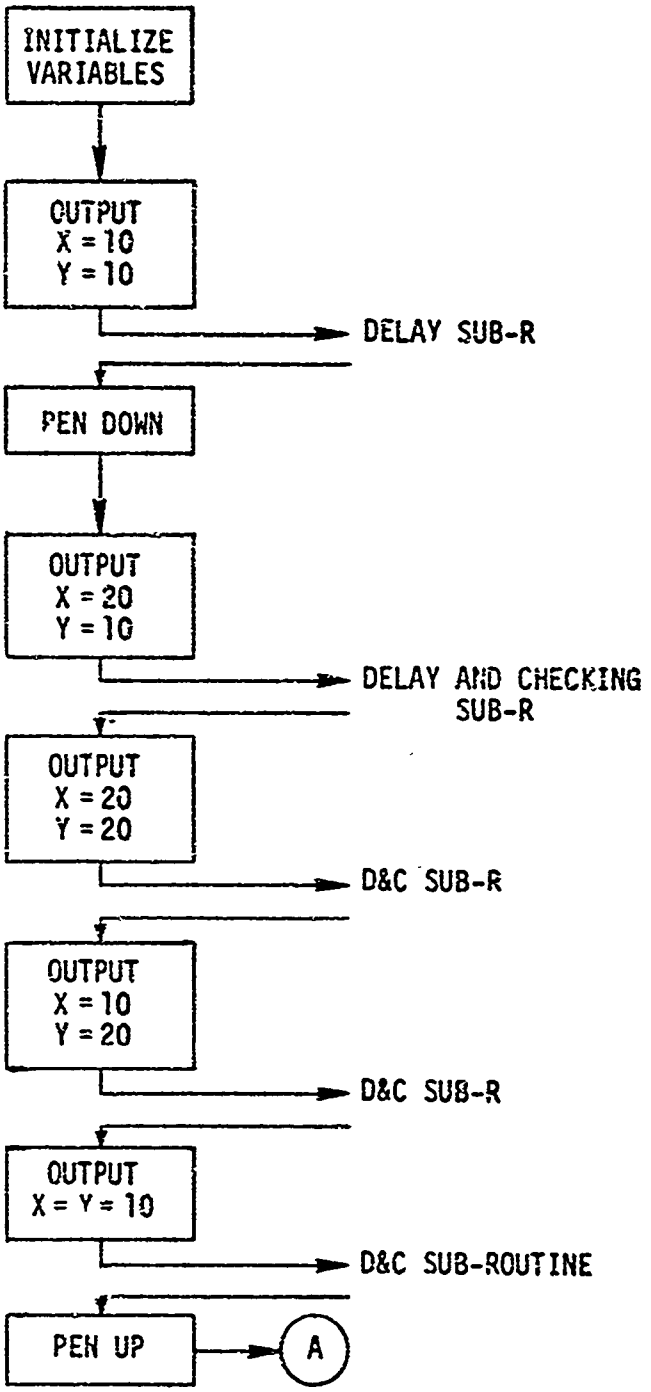
D17B to output 10 volts on both axes (point A) then go into a delay loop for approximately 10 seconds to ensure that the Positioning System (P.S.) was in the proper position. Then the coordinates for point B would be issued and the pen lowered to simulate engaging a cutting blade. The feedback from the Positioning System would be checked by the D17B every 10 msec to determine when the P.S. was within 0.5 inches of point B, then the coordinates for the next point, C, would be issued. This same procedure was followed for points D and A as well. The output of the P.S. was monitored on an X-Y plotter and a square as illustrated in Fig. 35b in Chapter IV was drawn except for the crosshatches.

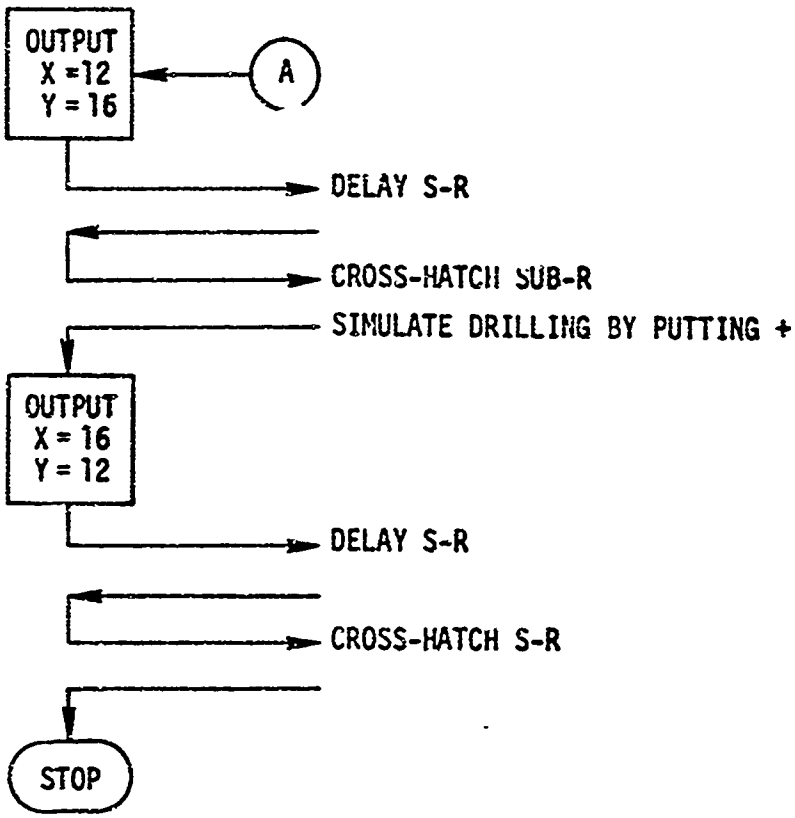
Once the P.S. returned to point A the pen was raised to simulate the halting of a cutting motion. Then the Positioning System was moved to the point (12, 16) to simulate drilling a hole at this point. Similarly the P.S. was moved to the point (16, 12) for the same reason. The cross-hatching drawn at these points simulated drilling the hole.

The actual program used is not included since there are numerous ways the D17B can be programmed to output a square. A flow chart of the program is included on the following pages.

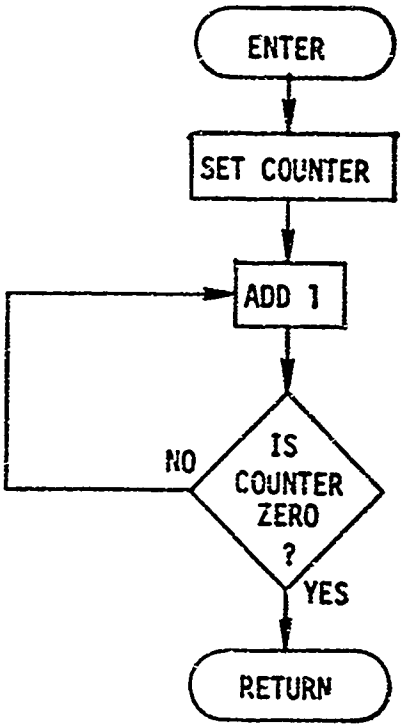
FLOW CHART OF THE

INCREMENTAL SQUARE SUBROUTINE

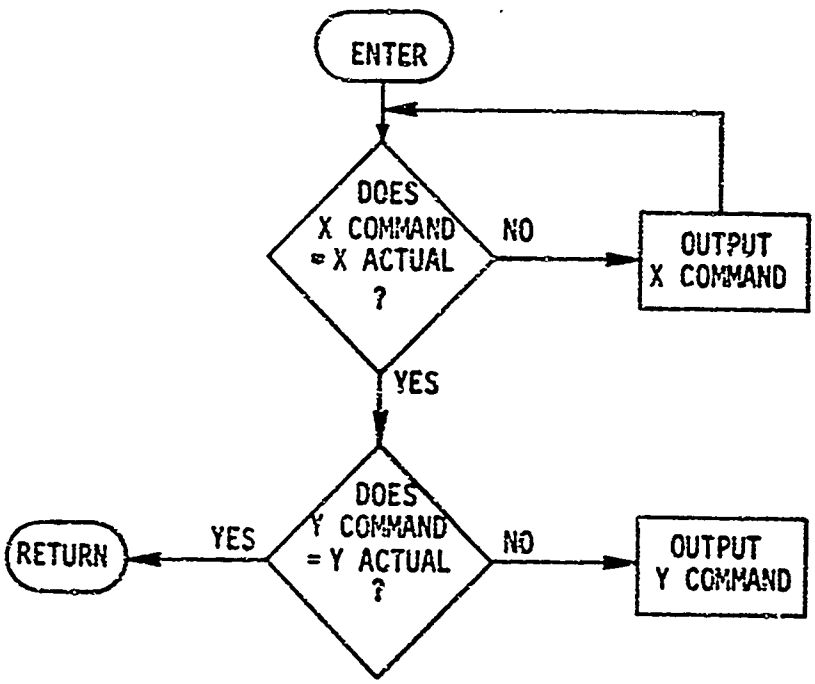



OUTPUT
X =12
Y = 16
A
DELAY S-R
CROSS-HATCH SUB-R
SIMULATE DRILLING BY PUTTING +
OUTPUT
X = 16
Y = 12
DELAY S-R
CROSS-HATCH S-R
STOP

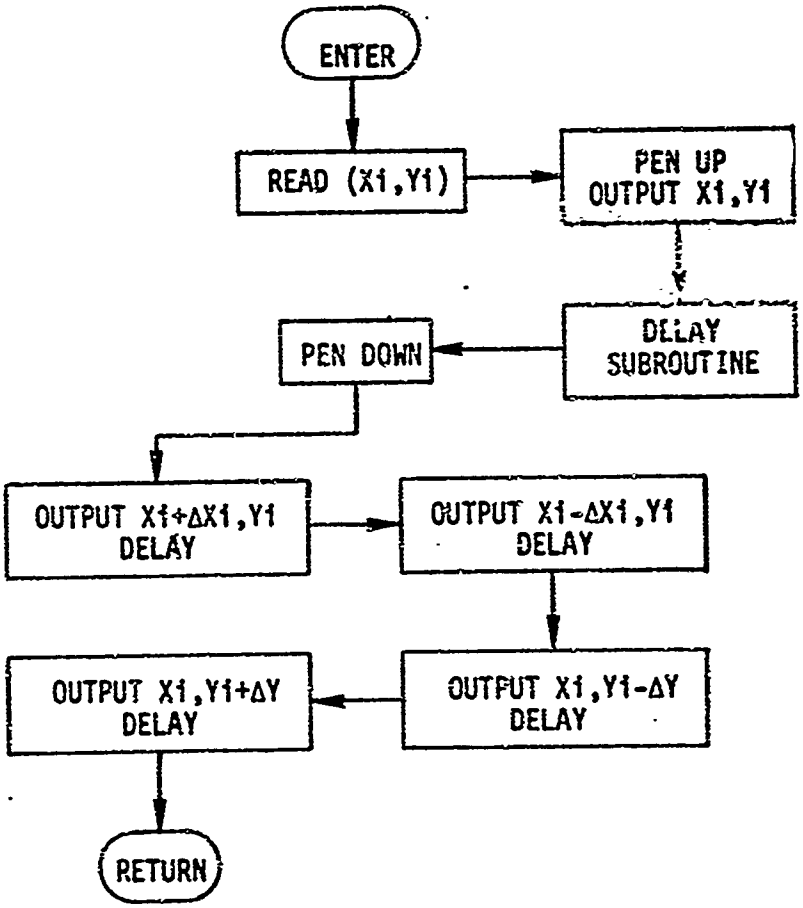
DELAY SUBROUTINE



DELAY AND CHECK SUBROUTINE



CROSS-HATCH SUBROUTINE



APPENDIX D

Circular Interpolation Subroutine

The basic algorithm used in this subroutine is derived from the geometry of a circle and the equations for rotation of coordinates. The central angle is written as $\Delta\theta$ to indicate a small angle as shown in Fig. D-1. This algorithm was developed by C. Bergren of Potter Instrument Co. (Ref 5).

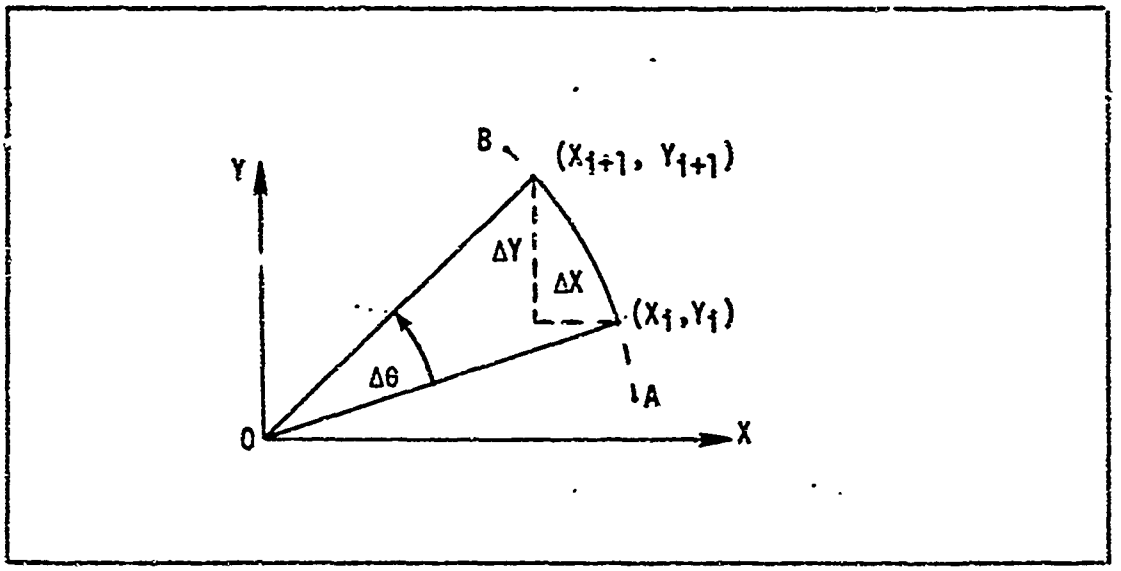


Fig. D-1. Geometric Representation of the Algorithm

Let (X_i, Y_i) be the i^{th} point on the circle, that is, the i^{th} step in the arc AB as shown in Fig. D-1. The next point (X_{i+1}, Y_{i+1}) can be obtained from this point if $\Delta\theta$ is known (Ref 5:57):

$$X_{i+1} = X_i \cos \Delta\theta - Y_i \sin \Delta\theta \qquad \text{(D-1)}$$

$$Y_{i+1} = X_i \sin \Delta\theta + Y_i \cos \Delta\theta \qquad \text{(D-2)}$$

The central angle, $\Delta\theta$ (in radians), is assumed to be a small angle and can

be approximated by a truncated Maclaurin series (Ref 5:57):

$$\sin \Delta\theta = \Delta\theta \tag{D-3}$$

$$\cos \Delta\theta = 1 - \frac{(\Delta\theta)^2}{2!} \tag{D-4}$$

Now substitute Eqs D-3 and D-4 into Eqs D-1 and D-2:

$$X_{i+1} = X_i \left(1 - \frac{(\Delta\theta)^2}{2}\right) - Y_i \, \Delta\theta \tag{D-5}$$

$$Y_{i+1} = X_i \Delta\theta + Y_i \left(1 - \frac{(\Delta\theta)^2}{2}\right) \tag{D-6}$$

Eqs D-5 and D-6 require four multiplications each, but if $\Delta\theta$ were restricted to inverse powers of two, the multiplications could be reduced to shift operations.

In going from one point to the next on the circular arc, three calculation errors are introduced. First, inaccuracies in evaluating the sine and cosine (E_r); secondly, accumulative point-to-point arc traverse error, E_{ch}; and thirdly, round-off error due to a finite number representation (Ref 5:58). The first two errors are related by

$$E_r = 2\pi \Delta\theta E_{ch} \tag{D-7}$$

Thus if $\Delta\theta < \frac{1}{2\pi}$, then $E_r < E_{ch}$ and E_{ch} is the dominating error. The value of $\Delta\theta$ can be determined by

$$E_{ch} = R \frac{(\Delta\theta)^2}{8} = \frac{1}{2^{m+3}} \tag{D-8}$$

if $\Delta\theta$ is restricted to reciprocal powers of 2 ($\Delta\theta = 2^{-m}$) (Ref 5:59). The variable m must be greater than or equal to 3 to insure $\Delta\theta < 1/2\pi$.

The circular interpolation subroutine written for the D17B uses $\Delta\theta = 2^{-6}$; thus Eqs D-5 and D-6 can be rewritten as

$$X_{i+1} = X_i - X_i/2^{13} - Y_i/2^6 \qquad \text{(D-9)}$$

$$Y_{i+1} = Y_i + X_i/2^6 - Y_i/2^{13} \qquad \text{(D-10)}$$

The solutions to Eqs D-9 and D-10 generated the values for the arc desired, but an algorithm is needed to stop the arc at the precise point desired. A simple procedure for completing a circular arc at a given point is given in Ref 5. This method assumes that the first computed step does not carry past the final step. The next to last computed point is designated (X_1, Y_1); the last computed point is designated (X_2, Y_2); and the desired final point is designated (X_f, Y_f). Then, if $X_1 - X_f$ and $X_2 - X_f$ have opposite signs and $Y_1 - Y_f$ and $Y_2 - Y_f$ have opposite signs the point (X_2, Y_2) has passed the final point. Wher this is determined (X_2, Y_2) is replaced by (X_f, Y_f).

The program implemented on the D17B used the above algorithms and the initial program was written using minimal delay coding techniques. The program on the following pages was taken from the original program, but rewritten in a manner which is easier to follow. The second program was not tested, but it was adapted from the original working program. Figures D-2, D-3 and D-4 on the pages following the program present flow charts of this program.

The special characters listed in Table D-I are used in this program. A statement in this assembly language would appear as:

```
A:  CLA X , *+1 /L ; Add X
```

or B: 47265000

In the first case, A is the symbolic location and CLA X is the mnemonic for the instruction, in this example clear and add the value in location X.

TABLE D-I

Special Characters for D17B Assembler*

Character	Definition
:	Symbol preceding is symbolic location
,	Symbol following is next instruction sector address
/	Flagstore into loop indicated by the following symbol
;	Comment follows
*	Present location
	Address Arithmetic
+	Add
-	Subtract

*Developed by D. Henson in Ref 19.

The expression "*+1" following the special character "," means the next instruction address is the present location plus one. The "/L" means to flagstore the previous contents of the accumulator into the L-loop. The rest of the statement is a comment which is ignored by the computer. The second case simply indicates that a constant value of 4726 5000 is to be assigned to symbolic location B. All numbers used in this program are in octal representation.

The following is an adaptation of the program used for circular interpolation for the D17B computer.

```
0000-0020  Setting of end points and initialization of FLAG
0000             CLA x+1, *+2
0001   X1        0700 0000              ; Starting Point X-Axis
0002             CLA *+1, *+2/L         ; L=X1
0003   Y1        7420 0000              ; Starting Point Y-Axis
0004             CLA *+1,*+2/U          ; U=Y1
0005   XF        0400 0000              ; Final Point X-Axis
0006             CLA *+1,*+2/E          ; E(7)=XF
0007   YF        0700 0000              ; Final Point Y-Axis
0010             DOA-4,*+1/E            ; E(1)=YF, Pen Down
0011             CLA*+1,*+2             ; Initialize the FLAG
0012             0000 0000
0013             STO F(2), *+1          ; Store Flag in F(2)
0014             CLA *+1, *+2           ;
0015             4000 0000
0016             CLA *+1, *+2/F         ; F(3)=4000 0000
0017             2000 0000
0020             STO F(0), Loop         ; F(0)=2000 0000
```

0021-0046 is the basic loop or group of instructions which calculate the new X_{i+1} and Y_{i+1}.

```
0021   LOOP      CLA L, *+1             ; Get Xi (old Xi+1)
0022             STO E(3), *+1          ; E(3)=Xi
0023             ARS-6, *+1             ; Shift Xi to Right 6 bits
0024             STO E(4)               ; E(4)=Xi Shifted
0025             ARS-7, *+1             ; Shift Xi to Right 7 more bits
0026             COM, *+1               ; -Xi(2^-13)
```

```
0027        ADD L, *+1          ; L=Xi - Xi(2^-13)
0030        CLA U, *+1/L        ; Get Yi
0031        STO E(0), *+1       ; E(0)=Yi
0032        ARS-6, *+1          ; Shift Yi 6 bits to Right
0033        STO E(2), *+1       ; E(2)=Yi(2^-6)
0034        ARS-7, *+1          ; Shift Yi 7 more to Right
0035        COM, *+1            ; -Yi(2^-13)
0036        ADD U, *+1          ; Yi-Yi(2^-13)
0037        ADD E(4), *+1       ; Yi-Yi(2^-13)+Xi 2^-6 =Yi+1
0040        STO E(6), *+1       ; E(6)=Yi+1 (New Computed Value)
0041        CLA L, *+1/U        ; Get Xi-Xi(2^-13)
0042        SUB E(2), *+1       ; Xi-Yi(2^-6) - Xi(2^-13)=Xi+1
0043        SUB E(7), *+1/L     ; L=Xi+1 , Xi+1-XF
0044        STO F(1), *+1       ; F(1)=Test Value Xi+1-XF
0045        CLA L, *+1
0046        STO E(5), *+1       ; E(5)=Xi+1 (New Computed Value)
```

0047-0076 is where the test variable (T) is constructed. Bits 24 and 23 are the only bits with information. Bits 22-1 are all zero. The test variable is based on $X_{i+1} - X_f$, $X_i - X_f$, $Y_{i+1} - Y_f$, and $Y_i - Y_f$.

```
0047        CLA *+1, *+2
0050        4000 0000           ; Mask
0051        TRA *+1/L           ; Mask into L
0052        CLA E(3), *+1       : Get Xi
0053        SUB E(7), *+1       ; Xi-XF
0054        ANA, *+1            ; Mask out all values except
0055        STO E(2), *+1       ; bit 24 and store in E(2)
```

```
0056        CLA F(1), *+1          ; Xi+1-XF
0057        ANA, *+1               ; Mask out all except bit 24
0060        STO F(1), *+1          ; and store in F(1)
0061        CLA *+1, *+2
0062        2000 0000              ; Mask
0063        CLA U *+1/L            ; Get Yi+1, put mask into L
0064        SUB E(1), *+1          ; Yi+1-YF
0065        ARS-1, *+1             ; Right shift one to put A24
0066        ANA, *+1               ; into A23, then mask all but A23
0067        ADD F(1), *+1          ; Combine two test variables
0070        CLA E(0), *+1/U        ; Store test value in U
0071        SUB E(1), *+1          ; Yi-YF
0072        ARS-1, *+2             ; Save bit 24 in position A23
0073        ANA, *+1
0074        ADD F(2), *+1          ; Second set of variables
0075        ADD U, *+1             ; Combine both sets into one
0076        CLA F(2)/L             ; L=Test variable (combined)
```

0077-0104 is where the flag is tested to see if X is set. If not, T is tested. If bit 24 is a 1, the flag is X-set. If bit 24 is 0, go to A.

```
0077        TMI A, *+1             ; Test Flag in F(2), if Flag=1
                                     Go to A, if not continue
0100        CLA L, *+1             ; Get test variable
0101        THI *+1, A             ; If test variable=1 continue
                                     if not go to A
0102        CLA F(2), *+1          ; Add 4000 0000 to
```

```
0103            ADD F(2), *+1          ; Flag to Set X
0104            STO F(2), *+1          ; Store modified Flag
```

0105-0115 Test flag to see if Y is set. If so, go to B; if not, check T. If bit 23 of T is 1, Y-set the flag. If not, go to INT.

```
0105  A         CLA F(2), *+1          ; Get Flag for test
0106            ALS-1, *+1             ; Prepare to test bit 23
0107            TMI B, *+1             ; If A23=1 (the Flag is set)
                                         Go to B, otherwise continue
0110            CLA L, *+1             ; Get test variable
0111            ALS-1, *+1             ; Shift to test A23
0112            TMI *+1, INT           ; If A23=1 want to set Flag,
                                         go to *+1, otherwise to INT.
0113            CLA F(2), *+1          ; Get Flag
0114            ADD F(0), *+1          ; Set Y
0115            STO F(2), *+1          ; Store Flag
```

0116-0126 Test flag. If x-set, go to C. If not X-set and not Y-set, go to INT. But if y is set, get Y_f and output X_{i+1}, Y_f.

```
0116  B         CLA F(2), *+1          ; Get Flag
0117            TMI C, X+1             ; If X is set go to C
0120            ALS-1, *+1
0121            TMI *+1, INT           ; If Y is set continue
                                         If Y is not set go to INT
0122            CLA E(1), *+1
0123            CLA E(5), *+1/U        ; Transfer YF to U-loop
0124  D         VOA, *+1/L             ; Transfer Xi+1 to L, output Xi+1
```

```
0125            CLA U, *+1
0126            VOB, *+1                ; Output YF
```

0127-0164 includes DELX and DELY which are subprograms which accept the commanded X and Y positions, respectively. The A/D converter is pulsed, then read so as to obtain the actual X and Y positions. When the actual X position equals the commanded position plus or minus the tolerance (error range), the Y position is checked in the same manner and then control is returned to the main program.

```
0127  DELX      CLS *+1, *+2            ; Mask to save 8 MSB
0130            7760 0000
0131            CLA E(5), *+1/L         ; Mask into L, Get Xi+1
0132            ANA, *+1                ; Mask all but 8 MSB
0133            TRA *+1/U
0134  L2        DOA-7, *+1              ; Send out convert command
                                          to A/D converter
0135            DIB, *+1                ; Read converted data
0136            ANA, *+1                ; Mask all but 8 MSB
0137            SUB U, *+1              ; Compare ordered position
                                          with actual position
0140            MIM, *+1                ; Check to see
0141            ADD *+1, *+2            ; if the actual position is
0142            0100 0000               ; within the error
0143            DOA-4, *+1              ; range of the commanded
0144            TMI L2, *+1             ; If it is go on, otherwise Loop
0145  DELY      CLA E(6), *+1           ; Get Yi+1
0146            ANA, *+1                ; Mask all but 8 MSB
```

```
0147            TRA *+1/U
0150   L1       DOA-7, *+1          ; Send out convert command to ADC
0151            DIB, *+1            ; Read converted data
0152            ALS-12, *+1         ; Left shift to get proper bits
                                      in 12 MSB
0153            ANA, *+1            ; Mask out all but 8 MSB
0154            SUB U, *+1          ; Compare ordered position
0155            MIM, *+1            ; Check to see if actual
0156            ADD *+1, *+2        ; position is within the
0157            0100 0000           ; error range of the
0160            DOA-4, *+1          ; commanded position.  If it is
0161            TMI L1, *+1         ; continue, if not Loop to L1
0162            CLA E(5), *+1       ; Set up L and U so
0163            CLA E(6), *+1/L     ; that they contain the new
0164            TRA LOOP/U          ; Xi and Yi (old Xi+1, Yi+1)
```

0165-0172 The INT subprogram outputs X_{i+1} and Y_{i+1} and stores these values in L and U, respectively.

```
0165   INT      CLA E(5), *+1
0166            LPR, *+1/L          ; Transfer Xi+1 into L
0167            CLA E(6), *+1       ; Transfer Yi+1 into U
0170            VOB, *+1/U          ; Output Yi+1
0171            CLA L, *+1
0172            VOA, DELX           ; Output Xi+1
```

0173-0207 If Y is set, output X_f and Y_f, then stop program. If y is not set, get X_f and Y_{i+1} ready for outputting.

```
0173   C       ALS-1, *+1
0174           TMI *+1, E              ; Check flag to see if Y is set
0175           CLA E(7), *+1           ; If Y is set output XF and YF
0176           LPR, *+1/L              ; Transfer XF into L
0177           TRA 0200                ; Transfer to another channel
0200           CLA E(1), *+1
0201           VOB, *+1                ; Output YF
0202           CLA L, *+1
0203           VOA, *+1                ; Output XF
0204           HPR, *+1                ; Halt program
0205           TRA 0000                ; Allows programmer to rerun
                                         program by issuing a Halt
                                         then Run command
0206   E       CLA E(6), *+1           ; Get Yi+1
0207           CLA E(7), D             ; Get XF
```

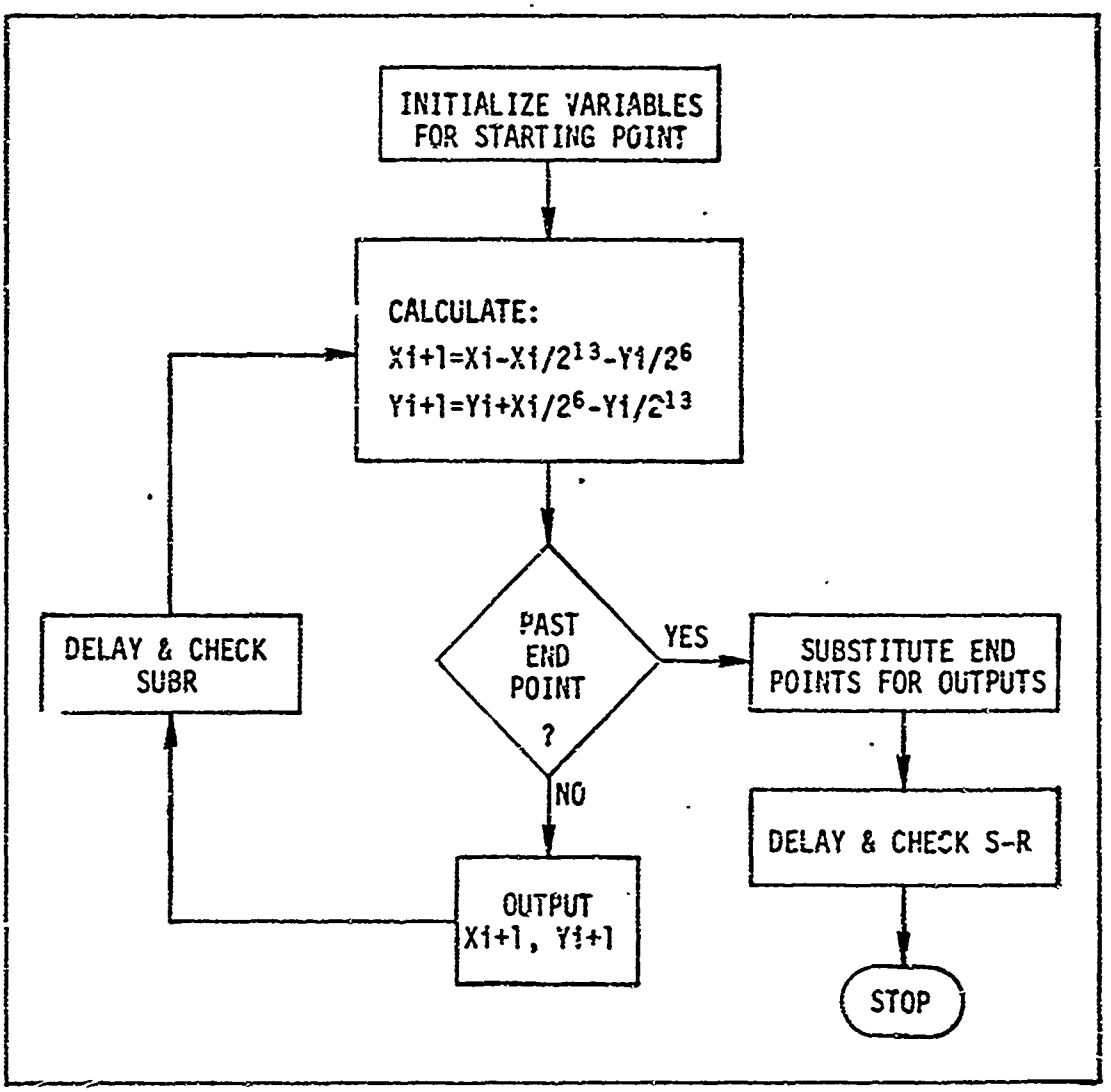


Fig. D-2. Flowchart for the Circular Interpolation Subroutine

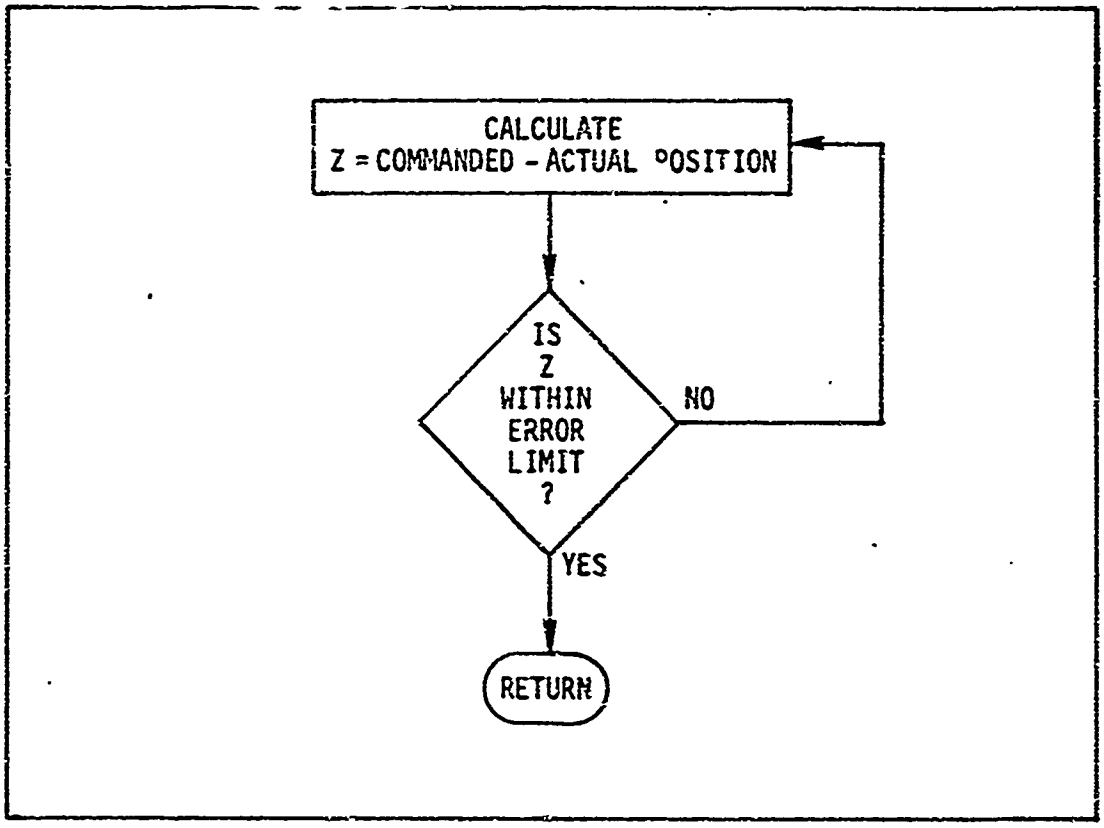


Fig. D-3. Flowchart for the Delay and Check Subroutine

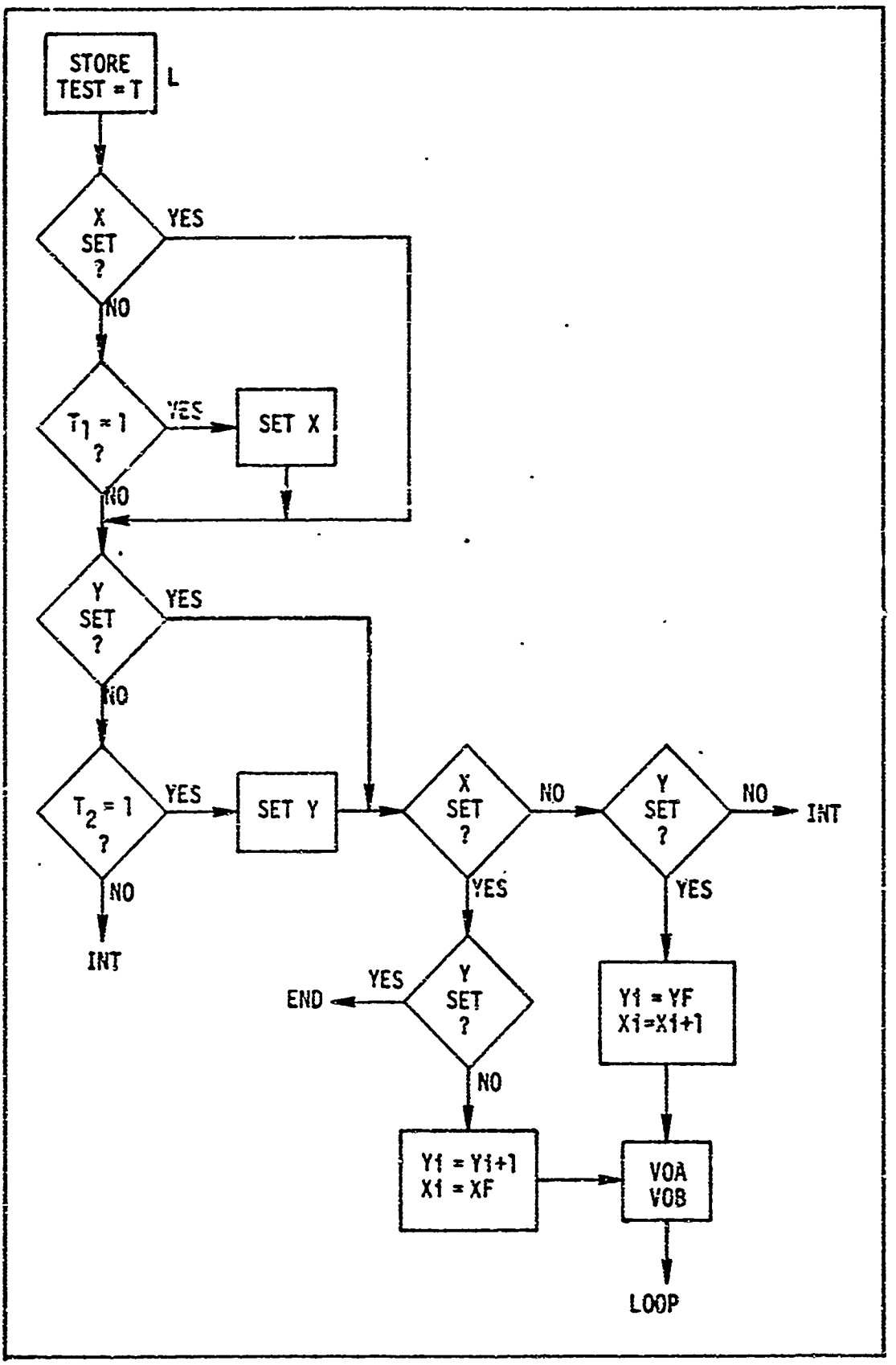


Fig. D-4. Flowchart for End Point Determination

APPENDIX E

Comparison of Minicomputers

Five minicomputers were chosen for comparison with the D17B and D37C computers as shown in Table E-I. These five minicomputers were chosen because they are presently in use in either process control or numerical control applications.

The HP-2116B minicomputer is used for nondestructive testing of coating thickness by the Norton Company. The computer also records data and then outputs a data summary at the end of each run (Ref 21:165). The PDP 8/I or 8/L computer is used in process control at Fort St. Vrain Nuclear Generating station. The PDP 8/I minicomputer controls the automatic refueling operations (Ref 21:161). The MDP-1000 is used for supervisory control of a remote power substation. It monitors the state of 65 points every 4 msec (Ref 21:164). Digital Systems, Inc. uses an SPC-12 minicomputer for numerical control of a circuit-board drilling machine (Ref 24:172).

TABLE E-I

Comparison of Minicomputers

	D17B/D37C	HP 2116B	MDP-1000	PDP 8I/PDP 8L	SPC-12
CPU Features:					
Instruction word length	24	16	12	12(24)	8/16
Accumulators	1 Memory	2	5	1	4
Registers	2 Memory/4 Memory	7	9	n/a	8
Index Registers	1	0	3	8 Memory	3
Bits for op code	4	4	8	3	8
Bits for address	12	10	12	8(15)/8(13)	12
Words directly addressable	2.7 K/7.2 K	2 K	4 K	2.56	4 K
Words indirectly addressable	none	32 K	4 K	32 K/8 K	4 K
Indirect addressing levels	none	multiple	single	single	single
Instruction Sets:					
Store time, μsec	78-1/8	3.2	6.48	3/3.2	4.2
Add time (full word), μsec	78-1/8	3.2	4.32	3/3.2	4.2
Fixed point, multiply	yes	opt	no	opt/no	no
Fixed point, divide	no/yes	opt	no	opt/no	no
Floating point, multiply	no	no	no	no	no
Floating point, divide	no	no	no	no	no
Fixed point × time (μsec)	1015-5/8	19.2	-	n/a	-
Fixed point ÷ time (μsec)	n/a / 2027-1/4	20.8	-	n/a	-
Fixed point × software time	n/a	150	n/a	360	n/a
Fixed point ÷ software time	n/a	310	n/a	460	n/a

TABLE E-I (Continued)

	D17B/D37C	HP 2116B	MDP-1000	PDP 8I/PDP 8L	SPC-12
Memory:					
Cycle time, μsec	78-1/8	1.6	2.0	1.5/1.6	2.0
Word length, bits	24	16	8	12	8
Minimum memory size, words	2.7 K/7.2 K	8 K	4 K	4 K	4 K
Maximum memory size, words	2.7 K/7.2 K	32 K	16 K	32 K/8 K	16 K
Parity checking	no	opt	no	opt	opt
Memory protect	standard	opt	no	standard	no
System Cost (CPU + 4 K Memory)	$60 (2.7 K)	$24,000 (8 K)	$8,500	$12,000/$8,500	$5,000

VITAE

John M. Hill was born on 12 February 1941 in Taylorville, Illinois. After graduating from Taylorville High School in 1959, he has served on active duty in the United States Air Force until the present time. After a period of part-time college work, he attended Oklahoma State University under the Airman Education and Commissioning Program. He received the BSEE degree in 1971 and subsequently received a Reserve commission in April of 1971 prior to attending the Air Force Institute of Technology.

Raymond V. Cicirelli was born on 17 September 1942 in Wilmington, Delaware. He graduated from high school in Hialeah, Florida, in 1960. Upon graduation from the United States Coast Guard Academy in 1965, he received a BS degree and was commissioned as an Ensign in the U.S. Coast Guard. He served on the USCGC WINNEBAGO, USCGC MACKINAW, and at a merchant marine safety office prior to his assignment to the Air Force Institute of Technology.

This thesis was typed by Mrs. Virginia Blakelock.