AMRL-TDR-63-133

THE UDOFT FLIGHT SIMULATION SYSTEM

TECHNICAL DOCUMENTARY REPORT No. AMRL-TDR-63-133

DECEMBER 1963

BEHAVIORAL SCIENCES LABORATORY
6570th AEROSPACE MEDICAL RESEARCH LABORATORIES
AEROSPACE MEDICAL DIVISION
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO

Contract Monitor: William B. Goeckler
Project No. 6114, Task No. 611413

(Prepared under Contract No. AF 33(657)-7065 by
Sylvania Electric Products, Inc., Needham, Massachusetts)

NOTICES

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Qualified requesters may obtain copies from the Defense Documentation Center (DDC), Cameron Station, Alexandria, Virginia. Orders will be expedited if placed through the librarian or other person designated to request documents from DDC formerly ASTIA).

Do not return this copy. Retain or destroy.

Stock quantities available at Office of Technical Services, Department of Commerce, Washington 25, D. C. Price per copy is $4.00.

Change of Address

Organizations receiving reports via the 6570th Aerospace Medical Research Laboratories automatic mailing lists should submit the addressograph plate stamp on the report envelope or refer to the code number when corresponding about change of address.

700 - February 1964 - 162-27-515

FOREWORD

This report contains a review of the UDOFT program from June 1956 to December 1962. This period of approximately six and one-half years comprises system development (June 1956 to April 1960), installation (April 1960 to September 1960) and operation (September 1960 to December 1962). The work was sponsored jointly by the U.S. Air Force and the U.S. Navy under Contracts N61339-40, N61339-853 and N61339-1150. This summary report has been prepared for the Behavioral Sciences Laboratory of the Aerospace Medical Division, Wright-Patterson Air Force Base, by the Electronics Systems Division of Sylvania Electric Products, Inc., Needham, Massachusetts, under Air Force Contract No. AF33(657)-7605 and is in support of Project 6114, "Simulator Techniques for Aerospace Crew Training" and Task No. 611413, "Digital Computers." Mr. William B. Goeckler, Simulation Techniques Branch, Training Research Division, Behavioral Sciences Laboratory, 6570th Aerospace Medical Research Laboratories, served as contract monitor.

Mr. Julian Wargo, Manager of Sylvania's Digital Simulation Systems Department, was principal investigator and directed the preparation of this report. Other key Sylvania personnel who contributed include F. Kearney, K. Rago, and H. Wychorski of the Computer Laboratory; Mrs. F. MacNair and J. Prutsalis of the Programming and Analysis Laboratory, and D. Rush of the Product Support Organization.

ABSTRACT

UDOFT (Universal Digital Operational Flight Trainer) represents the first full-scale application of a high-speed, general-purpose digital computer to the real-time flight simulation problem. Through the use of the stored program digital computer, simulation of different aircraft is accomplished by changing the computer program. This flexibility is the key to the realization of the full advantages of the digital control system, as compared to the conventional analog control system, in this application. Basically a high-speed, general-purpose digital computer, the UDOFT computer represents an advancement in the design of real-time control computers. With the use of dual, 4096-word, random-access, magnetic core memories, the basic instruction time for the UDOFT computer is five microseconds. To interface with the analog environment of a flight compartment, the UDOFT computer is equipped with a special-purpose, real-time input-output capability.

Use of the computer in a simulation system demanded the preparation of programs for applying the computer to the solution of the mathematical model of the real-world system under consideration. Such programs were written for the F-100A and the F9F-2. Extensive qualification testing was performed to ensure proper and complete simulation of these aircraft.

PUBLICATION REVIEW

This technical documentary report is approved.

Walter F. Grether

WALTER F. GRETHER
Technical Director
Behavioral Sciences Laboratory

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

TABLE OF CONTENTS (Cont.)

LIST OF ILLUSTRATIONS AND TABLES

UDOFT FINAL REPORT

SECTION I

INTRODUCTION

1.1 Purpose

The purpose of this document is to summarize and evaluate the UDOFT (Universal Digital Operational Flight Trainer) program in a concise, coherent and objective manner. This report analyzes the UDOFT system covering both computer hardware and computer programming aspects of real-time digital simulation. A significant portion of this document is devoted to the numerous problems encountered during the program, the approaches taken to solve these problems, and an appraisal of solution techniques.

Based upon the experience of Sylvania personnel, test data, and performance characteristics of the UDOFT system, recommendations are proposed and documented for reference with regard to future digital flight simulation systems.

1.2 Background Information

Aircraft simulators for pilot and aircrew training have experienced widespread use and acceptance during the past fifteen years. During this period of time they have evolved from comparatively crude, unsophisticated devices to the highly complex electronic and electro-mechanical devices that now exist. Although the scope and magnitude of the simulation has been expanded greatly, the purpose of the simulator-trainer has remained the same; namely, that of providing to the senses of the pilot-trainee the illusion of actual aircraft behavior. Basically, this is accomplished by causing the controls and the instruments in a reproduction of the aircraft cockpit to govern and to indicate aircraft behavior exactly as they do in the actual aircraft. If the simulation requirement is limited to this form of illusion, instrument and control simulation is sufficient; this degree of realism is adequate for most forms of simulator training. If, however, more detailed training is required, the senses of the pilot-trainee must be influenced to a greater degree. This is effected by simulating characteristic aircraft sounds, aircraft motion, and the environment in which the real aircraft would be operating.

A simple simulator-trainer consists of a reproduction of the cockpit section of an aircraft, an instructor's station, and a computing element. The cockpit and the instructor's station provide inputs to and accept outputs from the computing element. As the pilot-trainee goes through the motions of flying, the cockpit controls, through appropriate transducers, provide input signals to the computing element. On the basis of the current positions of the controls and the past history of the mock flight, the computing element determines the current status of the aircraft's behavior (e.g., rate of climb, velocity, altitude, etc.) and feeds these computed values to instruments and indicators of pilot-trainee and instructor. At the same time, the instructor may arbitrarily introduce various conditions, such as heavy icing, rough air, or engine failures, by means of controls located at his station. These inputs are assimilated by the computing element and the resulting effects on the simulated behavior of the aircraft are produced. Thus the pilot-trainee can be trained in the command and control of a particular type of aircraft under both normal and abnormal operating conditions.

Simulator-trainers, in the same manner as the actual aircraft they simulate, have become more complex, more costly, and vastly different from their predecessors. The major reasons, in both instances, have been the rapid advancement in engineering technology and the keen competition among manufacturers in the respective industries. Thus relatively few of the same components are ever utilized in the different classes of systems, most all of them being of a special-purpose nature. In the case of the simulator-trainers, each different device that has ever been built has been a special-purpose analog system, the function of which has been to simulate one, and only one, particular type of aircraft. As a result, aircraft obsolescence has caused simulator-trainer obsolescence; the lack of flexibility has sounded the knell for the special-purpose system.

1.3 Program History

In order to minimize the obsolescence rate of the ever more costly analog simulator-trainers, the possible application of a general purpose digital computer to the flight simulator-trainer problem was taken under consideration at the U.S. Naval Training Device Center, Port Washington, New York. Since the digital computer art was relatively new in 1950, U.S.N.T.D.C., known then as the U.S. Navy Special Devices Center, awarded a study contract to the Moore School of Electrical Engineering (MSEE) at the University of Pennsylvania. The evolution of the program at MSEE can best be summarized by using the words of Morris Rubinoff, an individual deeply involved in the development of real-time digital flight simulation. (Ref. 1).

The results of the first year of this study contrasted markedly with the prevalent optimism of the digital computer field, then in its infancy, which had overestimated the capabilities of the computer of that day and underestimated the mathematical problems associated with digital real-time simulation.

The fastest computer under development at that time was Raydac, which had a 4 mc clock rate. The Moore School study estimated that even with Raydac the flight simulation problem would require 0.22 seconds to advance the computation of airplane flight by one quadrature step. Unfortunately, it could only be conjectured what step size was acceptable for numerical methods of solution of the differential equations of motion. Hand computations and intuition led to a guess that 1/8 second was the largest possible step which would avoid introducing spurious instabilities.

Thus, even the most optimistic estimates indicated that real-time digital airplane simulation was not yet feasible because computers were too slow by at least a factor of two. The Moore School then addressed itself to the two basic problems: that of discovering a mathematical criterion or criteria for predicting the stability of numerical solutions regardless of the actual flight path taken by the simulated airplane; and that of improving the logical structure of the digital computer to increase speed by about one order of magnitude. The computer improvements had to come from logical design because it was felt that only switching circuits with proven reliability could be incorporated into a demonstration operational flight trainer. An early decision was made to base all calculations on the 1 mc SEAC circuits, the latter running reliably since 1947.

A breakthrough on the mathematical problem was made by Dr. H. J. Gray, Jr., with his development of a "stability chart" for numerical solution of differential equations. The stability chart is a digital counterpart of the Nyquist diagram, and permits a mathematician to specify, in advance, a quadrature step in size for which stable simulation is assured.

The stability chart made it possible for the Moore School to find a "best" formula to use in stable real-time airplane simulation of high performance airplanes using real-time steps of 1/20 second or shorter. This provided a quantitative goal for the improvement required in computer speed. Certain preliminary computer modifications led immediately to a four-fold improvement, sufficient to imply feasibility of real-time airplane simulation although without any margin of safety. The most significant change was the use of separate high-speed memories for instructions and data, considered (at that time) to be a backward step, but one which gave a full factor-of-two improvement.

Still other logical improvements were incorporated into the computer design; the final result was a computer about 100 times the speed of SEAC, with a 5 microsecond add time and a 10 microsecond multiply time.

This computer along with all the analog and switch inputs, the analog and switch outputs and displays, the real-time clock, and the multiplexed digital-to-analog converter was christened "Universal Digital Operational Flight Trainer" (UDOFT). In 1954 the Moore School informed NTDC that it firmly believed that digital simulation of even supersonic aircraft was feasible using a computer such as UDOFT and advised them to proceed to have the simulator built.

On 30 January, 1956, the Training Device Center issued a specification for "The Development and Construction of a Digital Computer System for Actuation of Operational Flight Trainers." The purpose of the procurement intended by this specification was to

demonstrate that a digital computer system could be utilized to simulate a subsonic F9F-2 jet fighter and a supersonic F-100A jet fighter by actuating suitable cockpit reproductions and instructor control stations in real-time.

A contract to undertake this program was awarded to Sylvania Electric Products, Inc. on 29 June, 1956. The scope of the work dictated that the equipment, specifically the digital computer system, should be designed, developed, and constructed in accordance with the logical computer structure and preliminary circuit design prepared by the Moore School. Although Remington Rand Univac had conducted an evaluation study of the Moore School design, resulting in numerous recommendations, many areas of the design required further effort. These included but were not limited to:

1. Design and development of the dual five-microsecond coincident-current magnetic core memories
2. Design and development of the five phase 1.2 megacycle clock pulse system
3. Design and development of the computer operation and maintenance console
4. Development of the plug-in circuit modules
5. Integration of the computer system and the two government furnished analog simulators
6. Programming of the computer for the two aircrafts
7. Performance testing of the integrated system (computer systems aircraft simulation program aircraft cockpit mockup, and simulator instructor station)

After nearly four years of concerted effort and the expenditure of approximately two million dollars of Navy and Air Force funds, the UDOFT system was delivered to its permanent installation at the U.S. Naval Training Device Center Annex, Garden City, Long Island, New York. The formal unveiling of the system to the public occurred at the U. S. Navy - U.S. Air Force - UDOFT Conference and Demonstration on 13 September, 1960. At this time it was clearly stated that the UDOFT system would be utilized as a research tool to investigate problems encountered in the use of digital simulation techniques and to support the study of psychological engineering, and mathematical applications of simulation to military training.

These words have born fruit, for in nearly three years of operation, approximately 10,000 operating hours have been logged on the system for such applications as extensive testing and evaluation of the F-100A simulation model, simulation of the dynamics of a submarine (ref. 2), simulation of the dynamics of a surface ship (ref. 2), experimentation with the significance and the required accuracy of coefficients in aerodynamic equations of motion, investigation of the problems in the use of a digital computer for simulating a hypersonic earth orbital and re-entry vehicle, and the ever-present study of improved numerical procedures for maximizing the real-time simulation capability of a digital computer system. Awareness and availability of the UDOFT system is starting to permeate the military research organization; as a result it is expected that the research load placed on the UDOFT system will increase greatly in the future. The results of the current research projects and of the many now in the planning stages will provide invaluable information for the improvement of the real-time digital simulation art.

1.4 General Program Requirements

As prescribed by the original system specification, the simulation equipment shall consist of a digital computer with input-output devices and all components and circuits necessary for solving, in real-time, the basic equations of motion, position, and flight dynamics of either of two specific high performance jet aircraft. The equipment shall also present to the cockpit displays of the appropriate aircraft, the solutions of these equations as a function of aircraft control movements in terms of the aircraft's performance and flying qualities. The system shall be sufficiently flexible to simulate any one of several types of single-engine jet fighter aircraft. The computer system shall consist of a digital computer, a computer control console, a punched-card handling input mechanism, an

output printer, analog-digital and digital-analog converters, multiplexers, servos, and all associated equipment needed to accept pilot and instructor commands and to provide the instrument and control reactions to the mock F9F-2 and F-100A cockpits.

As the program progressed, it was realized that additional items, not considered at the time the specification was prepared, were necessary. The most important item by far was the computer program for the simulation of the F-100A aircraft; of lesser profundity but equal importance were programs for aiding operating personnel in the maintenance of the computer system. Further, and again in the area of programming, there existed the need to develop programs to aid the programming personnel to create the aircraft simulation programs. As may be evident from these examples, the art and the understanding of programming was not even as advanced as the understanding of digital computer hardware which, although these statements refer to a period of time only six short years ago, was rather limited.

And of course, as with any prototype development program, requirements existed for such supplementary items as reports, drawings, and handbooks on training, installation and maintenance.

These, in brief, are the basic requirements that guided the development of the UDOFT system. Throughout the program, minor changes and additions were made to the specification; however, the intent of the specification never changed and UDOFT was developed very much as originally planned.

1.5 Organization of this Report

The body of this document, though divided into seven sections, covers four primary topics: the hardware system particularly the computer; computer programming; testing of the system as a flight simulator; and a brief evaluation of UDOFT and its use to date.

The hardware system is discussed in three sections. Section II, System Description, presents the fundamentals of digital computers, leading into an introduction to the UDOFT computer; Section III, UDOFT Computer Description, presents in some detail the various units of the computer and their functions; Section IV, Computer Hardware Development History, treats the more prominent design considerations and problems encountered during the development of the computer.

Computer programming is discussed in two sections. Section V, Simulation Program Development, presents the programming aids that were evolved to effect efficient programs development and checkout, and the prominent factors that influenced the organization of the flight simulation programs; Section VI, Simplified Description of the F-100A Program traces briefly the organization of a flight simulation program.

Delineation of the aids, programs, and procedures that were developed to expedite testing of a complete flight simulation program is presented in Section VII, Testing Aids, Programs, and Procedures Used in Establishing the Performance and Flying Qualities of the Simulated F-100A Aircraft.

The final section of the report, Section VIII, Summary, presents a brief objective evaluation of the UDOFT system, its design and its use to date.

SECTION II

SYSTEM DESCRIPTION

2.1 Introduction to Basic Digital Computers

The material in this section has been included for providing background information on the basic aspects of digital computers, which may be beneficial to persons not familiar with the vocabulary and thought patterns of the digital computer engineer. It is hoped that this material will be sufficiently informative to the uninitiated reader so that he may read this document without being overwhelmed by the bulk of specialized language and ideas contained herein.

2.1.1 Numbering System

Since digital computers manipulate electrical signals which discretely define the numerical magnitudes of a body or quantities, some notice must be taken of the numbering system employed by them. The binary system, utilizing only the digits 1 and 0, was found far more suitable for use within the computer than the familiar decimal notation with its ten digits. The octal system is then applied, to condense the bulky form of the binary into a shape readily spoken or written.

The design of digital computers has dictated the use of binary notation. The reason for this is quite simple; the components used in digital computers are inherently binary in nature. For example, the relay used extensively in the early digital computers and still used in telephone switching computers, exhibit binary qualities. When activated, its contacts assume one state; when deactivated, its contacts can assume only one alternate state. Thus, with its characteristic of two stable states of operation, the simple relay constitutes a basic digital computer element. Examples of other so-called two-state elements are vacuum tubes and transistors, which may be maintained either saturated (fully-conducting) or cut-off (non conducting), and magnetic materials in which the magnetic field may be changed from one direction to the opposite direction. Thus it can be understood why the components dictated the use of the binary numbering system rather than the converse. Were it possible to develop inexpensive and reliable deca-state elements, digital computers would, in all probability, use the decimal system.

Some computers do operate with decimal numbers; however, the technique used is that of manipulating the quantities as binary coded decimal (BCD). The basic mechanisms of these computers are still binary in nature, and such a device is over-complex.

2.1.2 The Fundamental Computer

The fundamental computer is composed of four major elements; an Arithmetic Unit, a Control unit, a Memory Unit, and an Input-Output Unit. This arrangement is depicted in Figure 1.

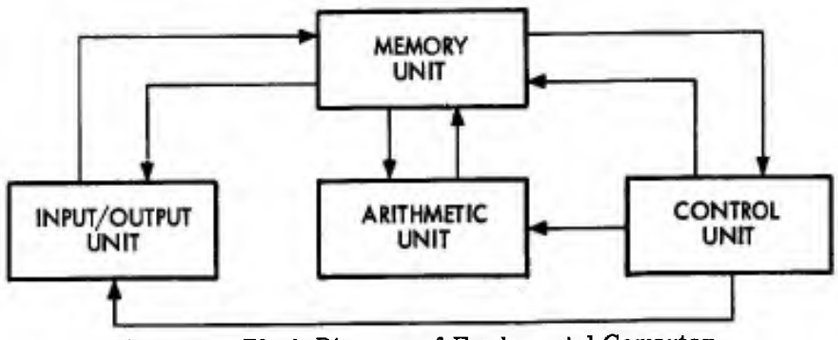


Figure 1. Block Diagram of Fundamental Computer

The Arithmetic Unit is the computational core of the computer. It is here that the basic arithmetic operations of addition, subtraction, multiplication and division are performed. In reality it is nothing more than an adding machine capable of performing multi-bit additions at extremely high speed. This limited capability is adequate, since the other

basic arithmetic operations may be performed by multi-step additions conforming to an algoritism which will achieve the desired result.

The Arithmetic Unit of a digital computer may be likened to the counting wheels in a desk calculator. The mechanism for performing the additions is present, the means of activating such a mechanism is some external influence or control, in this case the Control Unit.

The function of the Control Unit is to manipulate the Arithmetic Unit in an orderly sequence of operations, to achieve a usable result. The Control Unit may be likened to the mechanism of the desk calculator which connects the operate or control keys to the counting wheels. When the add key is depressed, the positions of the counting wheels are simply augmented by the quantity which has been entered on the calculator keyboard. When the multiply key is depressed, the operation becomes more complex. The result is a number of successive additions accompanied by an apparently erratic movement of the calculator carriage. It is the control mechanism within the calculator that determines when the proper number of additions has been performed, when the carriage should be moved, and when the operation is complete. The Control Unit of the computer performs a similar function. However, since a digital computer is capable of executing a far greater variety of commands than a desk calculator, its Control Unit is considerably more complex than its counterpart in the desk calculator.

A digital computer consisting of only an Arithmetic Unit and a Control Unit would have no greater value than the desk calculator sitting unused on a table. It is not until someone enters data in the calculator keyboard and depresses an operation key that the value of the machine is realized. So it would be also with a two-unit digital computer. It requires something to direct the control unit to add or to subtract; it requires something also to provide the quantities that are to be added or subtracted. This task falls to the third unit of the fundamental computer, namely the Memory Unit.

The primary function of the Memory Unit is precisely what the name implies; it remembers information which is inserted in it and makes this information available when requested. The basic unit of information is called a word. A word may be tersely defined as an ordered set of characters or bits, stored and transferred as a unit. The import of the information-unit takes on many forms; in one instance the memory word may represent a directive or instruction, in other instances it may represent a numerical quantity.

More precisely then, the Memory Unit stores both the instruction type of word and the operand type of word. The instruction words when withdrawn from the memory activate the Control Unit, causing a sequence of micro-operations to occur The operand words are withdrawn as called for by the instruction words and are manipulated as directed by the instructional content of the instruction word.

Using again the analogy of the desk calculator, the Memory Unit of the fundamental computer represents the human operator of the calculator. The operator processes a sequence of instructions that will cause the calculator to solve the problem; he introduces the numerical data that is to be acted upon, and performs the temporary storage of intermediate results as required, by means of his own memory or by means of some aid such as paper and pencil.

The development of the fundamental computer has now progressed to the point where it can substantially govern its own performance. However this fundamental computer, now consisting of three units, is still incomplete. Without the fourth unit, the Input-Output unit, it is comparable to a desk calculator on which the number wheels have been masked. One major function of the Input-Output Unit of the digital computer is to make available to the user of the computer the results of certain computations. This output may take any of several different forms, ranging from a single indication of satisfactory problem completion to extensive numerical print-outs or to the packaged control of a complex electro-mechanical system. Of equal importance is the input capability of the Input-Output Unit. It is by this means that the instructional program and the numerical operands are entered into the computer memory. On a higher level of utilization, the Input-Output Unit provides the means for entering new or additional data while the computer is operating. The nature of these inputs may range from magnetic tape to punched cards to real-time data describing the condition of a complex electro-mechanical system is being controlled by the computer.

The description of the fundamental four-unit computer is now complete. All digital computers may be dissected into these four identifiable units. The ensuing descriptions of the UDOFT computer are so sub-divided, in order to maintain organization and continuity.

2.1.3 Definition of Terms

However, before proceeding to a description of the UDOFT computer, it is desirable to define some of the other terms peculiar to a discussion of digital computers. The terms to be covered are register, address, program, serial, parallel and synchronous.

Regardless of the form of storage media in a digital computer, any device which is capable of storing or holding information for a period of time is referred to as a register. In the case of the Memory Unit, there may be a considerable number of registers. Each register is uniquely identified by a numerical designator. This designator is referred to as an address, because, just as a street address denotes a particular location on that street, this designator denotes the location of a particular register within the memory storage device. When information is withdrawn from or entered into a register, reference must be made to this designator or address.

Computer instructions normally serve a twofold purpose. First, the instruction word denotes a particular arithmetic or logical operation that is to be performed. Second, if an operand is involved in the operation, the instruction word denotes the address of this operand. Instruction words vary in length and complexity depending upon the particular computer. The simplest form of instruction word is the single-address type, which consists of a single operation or order and a single operand address.

Computers usually have also the capability of address modification. When address modification is specified–by a bit of the instruction word–the address of the operand is the address portion of the instruction word incremented or decremented by the contents of another register usually called an index register. Computers may have more than one index register.

The solution of a problem by a digital computer is accomplished by executing many successive instructions at a high rate of speed. The complete set of instructions devised to cause the computer to solve the problem forms a computer program. If the program is entered into and stored in the internal memory of the computer, it is referred to as a stored program. Normally, instructions are executed in sequence; however it is possible, by means of the program itself, to modify the normal instruction sequence.

Just as there are several ways of implementing an analog computer (AC, DC, etc.) there are at least two basic forms that the implementation of a digital computer may assume; these are serial and parallel. In a serial computer, data transmission between registers is effected one bit at a time, on a single transmission line. Consequently the bits of a word in a serial computer are operated upon one at a time. This results in minimum hardware but has the disadvantage of low computational speed. On the other hand, the individual bits of a word in a parallel computer are transmitted simultaneously between registers on parallel transmission lines. Consequently entire words are operated upon in a parallel computer. This results in high effective computational speed but has the disadvantage of requiring much more hardware than the serial computer to perform the identical operations.

The remaining significant term that requires definition is the word synchronous, which refers to the strict time dependency of computer operations. A computer performs synchronously when each micro-operation of an instruction is performed at a distinct instant of time during the execution of that instruction and when the sequence of micro-operations is fixed for each instruction, regardless of the configuration of the operands. If on the other hand, there is no synchronization between the execution of the micro-operations and a fixed timing cycle, the computer performs asynchronously. Asynchronous operation has the advantage of increased computational speed. This results from the fact that the computer performs simple calculations rapidly, just as a human performs simple calculations rapidly. As a case in point, it requires less time to multiply a quantity by 3 than it does to multiply the same quantity by 17, 395. However, the system logic is more complex for an asychronous computer than it is for a synchronous computer. As in the case of serial versus parallel, the tradeoff is between speed of computation and complexity of the hardware.

These basic terms having been defined, may now be used to describe briefly the form of a computer. The most common type of large-scale, general-purpose digital computer available at present is the parallel, single-address, binary synchronous computer. This same configuration of descriptors in a broad sense define the UDOFT computer.

2.1.4 Computer Programming

In the digital fundamental computer there is but a single computing element, the Arithmetic Unit, which must carry out all computations. On the other hand, an analog computer consists of many distinct computing elements, each element having one and only one function. The analog computer accepts multifarious inputs and operates upon them simultaneously. This can occur because the individual computing elements are appropriately interconnected, and function as an integrated calculating mechanism. The operation of the many elements of the analog machine have been "programmed" through the specialization of function and the purposeful interrelation of the computing elements.

The common digital computer, with only a single general-purpose computing element, demands a computer program to schedule the use of this element. This program may take a number of forms, ranging from the "wired-program" (which results in a special purpose digital computer) to the "stored program" (which is prevalent in the more popularly-known general-purpose computer. The program acts as the intermediary between the man (programmer) and the machine (computer), directing the single computational element to perform the myriad operations.

Basically, there is little difference between the preparation of a digital computer program and the planning that must precede the solution of a problem on a desk calculator. Each program or plan includes the establishment of a sequence of operations, the insertion of numerical data, the temporary storage of intermediate data, and the output of the final results. In the case of the desk calculator most of this planning is informal, since the human intelligence is in constant communication with each step of the problem. On the other hand, the required digital computer program must be a strictly formal plan, for the human intelligence is divorced from the problem during the execution of the program. Thus the digital computer program must be complete in itself before the computer can operate.

2.2 Introduction to UDOFT

The UDOFT system consists of a general purpose digital computer, two aircraft cockpit mockups (an F-100A and a F9F-2) and associated instructor consoles, computer-cockpit buffering and conversion equipment, and two real-time simulation computer programs—one for each aircraft type. A model of the physical system, as installed, is depicted in figure 2.

The function of the digital computer is to determine, by means of the stored program, the behavior of the simulated aircraft during pretended flight, and to report data reflecting this behavior to the pilot-trainee via the cockpit instrumentation. The behavior of the aircraft, which is affected by manipulation of the cockpit controls and by the instructor's supplementary controls, is described analytically by a system of equations. This system of equations relates Newton's Laws of Motion to the particular aircraft in terms of static aerodynamic coefficient data and stability derivatives. These terms describe respectively the steady state condition and the dynamic or transient responses of the aircraft.

In essence then, the digital computer in this application is a part of a closed-loop system consisting of the cockpit controls, the computer, the cockpit instrumentation, and the pilot-trainee. A basic requirement of any device in a closed-loop system is a bandwidth which will adequately pass the high frequency signals that characterize the dynamics of the system. In the analog scheme, this requirement applies directly to the bandwidth characteristic of the servos or amplifiers that constitute the computers. Bandwidth requirements, as applied to the digital device, imply problem solution rates rather than the common electrical circuit bandwidth considerations. An analogy would be the considerations that are given to the transformation of an image to an electrical signal in a television system. In television a complete image, or field, is not obtained instantanously; a finite interval of time is required to scan the image. The scanning transforms the instantaneousness of the image information into a sequence of discrete image information

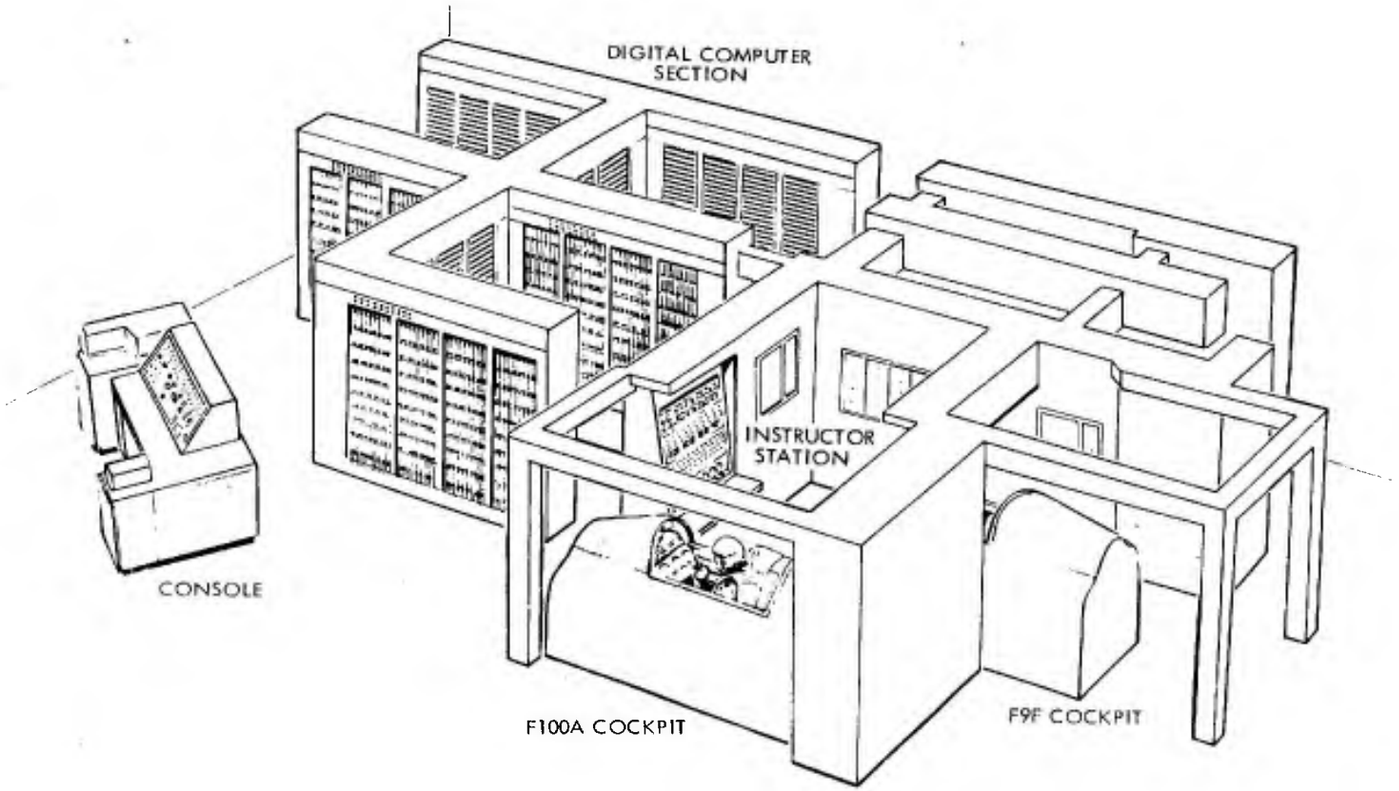


Figure 2. UDOFT System

bits. As a higher degree of resolution is required, the scanning rate must be increased; consequently the bandwidth of the system must be broadened. This is one of the most basic problems encountered when using a sampled-data device in an environment which is characteristically continuous, rather than discrete, in nature.

By analyzing the dynamic characteristics of the aircraft system, the degree of resolution required for faithful simulation, the rate at which discrete motions appear continuous to the viewer, and the characteristics of the processes that would be used to obtain numerical solutions of the differential equations, it was determined that twenty complete solutions of the instantaneous behavior of the simulated aircraft must be performed every second. With the solution rate specified it was determined that a digital computer for this application must be capable of extremely high computation speeds, namely twenty binary bit additions or subtractions in five microseconds, and multiplication in ten microseconds. These two rates are the most important because these arithmetic operations constitute most of the operations in the simulation programs. A digital computer was designed on the basis of these rates and of the role the computer would assume in the real-time closed loop system. The machine that resulted is explicitly a special-purpose, high-speed, parallel, single-address, binary, fixed point, synchronous digital computer. It is inherently a general-purpose computer and may be used as such; however, in its current application the computer lacks a certain degree of general utility, since only specialized input-output capabilities are provided.

The basic characteristics of the UDOFT computer are shown in table I. The fundamental order time of 5 microseconds enables the computer to perform complete additions, including memory accesses for the instruction and the operand, at 5 microsecond intervals. This high computation speed is accomplished through the use of two independent time-phased core memories, each with a capacity of 4096 words.

Number words are 22 binary bits in length, fixed point and fractional (i.e., the binary point is to the left of the most significant bit). The least significant 20 bits of the word are the magnitude of the number; the 21st bit is the sign of the number and the 22nd bit is an "odd-ones" parity check bit. Instruction words are only 20 binary bits in length and contain but one instruction per word. The least significant 12 bits of the instruction word designate the operand address, the next six bits designate the order type, the following bit specifies whether or not relative addressing is to be used and the last bit is again an "odd-ones" parity check bit.

Of the 64 order types that could be specified by six order-type bits, only 32 such types have been mechanized in the machine. Most of the instructions can be executed in 5 or 10 microseconds; only the divide instruction, which is infrequently used, is much longer.

The specialized UDOFT input-output system consists of five basic channels of information flow. The analog inputs, of which there are 24, are generated by 10-bit shaft-position encoders linked to the continuously variable controls in the cockpit in the cockpit mockup or at the instructor's console; examples of such controls are the throttle, the stick, the rudder pedals and the instructor's wind speed control. The analog outouts, of which there are 64, are analog voltages generated by a single 12-bit digital-to-analog converter and multiplexed to the appropriate output device located either in the cockpit mockup or at the instructor's console; examples of these devices are the altimeter, the airspeed indicator, the rate-of-climb indicator, and the control forces mechanism. The discrete inputs, 64 in all, are analogous to the sense switches of a truly general purpose digital computer. These inputs, which also originate either in the cockpit mockup or at the instructor's console, include, for example, the main battery switch, the landing gear controller, the after-burner switch, and the hydraulic system fail switch. The discrete outputs, of which there are 24, are binary outputs generated by the computer in accordance with the arithmetic sign of the number in the accumulator at the time the discrete output instruction is executed. Examples of indicators that are actuated by the discrete outputs are landing gear, up/down indicators, stall warning, and crash. The fast print facility is a 22-bit register which can be loaded under program control. The individual stages of the register control transistorized flip flops which control output devices.

Other input-output equipment which is part of the computer system consists of an IBM 514 Card Reader/Punch and an IBM Output Writer (electric typewriter). The IBM 514 provides the means for reading punched card information into the computer; the electric typewriter provides an output of printed information in non-real time.

TABLE I

BASIC CHARACTERISTICS OF THE UDOFT COMPUTER

Mode of Operation	Parallel and Synchronous
Internal Number System	Binary
Word Length	20 bits
Arithmetic System	Fixed point, fractional
Memory	Coincident current, magnetic core
Cycle Time	5.0 μsec.
Capacity	2 $\times$ 4096 words
Order Code	32 orders
Arithmetic Speeds	
Add	5.0 μsec.
Subtract	5.0 μsec.
Multiply	10.0 μsec.
Divide	105.0 μsec.
Input-Output	
Discrete	64 inputs 24 outputs
Analog	24 inputs (10 bit inverted gray code) 64 outputs (12 bit precision) converted to a voltage
Card Reader	1200 words/minute
Output Writer	20 lines/minute (short form) 6 line/minute (long form)
Fast Print Facility	22 bit output register

2.2.1 UDOFT Computer Operation–Simplified

A simplified block diagram of the UDOFT computer is shown in figure 3. The instructions of the simulation program are stored in the instruction memory; the operands or numbers specified by the address portion of these instructions are stored in the number memory. In general, the output of the instruction memory specifies both the number location in number memory and the operation to be performed. While the specified operation is being performed in the Arithmetic Unit, access is being made to the instruction memory for the next instruction to be performed. By the time the arithmetic operation indicated by the first instruction is completed, both the operation and number specified by the second instruction are available to the Arithmetic Unit. Thus fast computations are achieved since computer dead-time, normally experienced due to memory access time, is effectively reduced to zero by using separate overlapping memories. This increase in speed, necessary to perform the problem in real-time, is achieved at the cost of greater system and programming capabilities.

In the normal instruction cycle, the instruction sequence counter, a twelve-stage binary counter, contains the address of the instruction to be performed. This address is transferred to the instruction memory address register and the instruction memory read-write cycle is initiated. When the instruction is extracted from the memory, it is read into the instruction memory output register. The five bits of this word that specify the operation to be performed are transferred to the order-type decoder, which then initiates the execution of the instruction.

The address portion of the instruction may be routed to the number memory address register in the direct mode or to a modified address, called a relative address. The reason for providing this address modification feature is to permit the application of the same program sub-routine to several sets of data stored in the number memory. When using address modification, both the internal operation of the machine and the programming differ considerably from that required by the direct mode. (Refer to section 2.1.3 for discussion of address identification.)

In the direct mode of operation, the address portion of the instruction is transferred directly to the number memory address register, and the number memory read-write cycle is initiated. The number specified by the address is read into the number memory output register and then transferred to the Arithmetic Unit. When the arithmetic operation has been completed, the results are routed to the transfer register. Data in this register may then be transferred under program control to the number memory, the analog output multiplexer, the print register, the instruction memory or the tally register.

2.2.2 UDOFT System Operation–Simplified

The availability of a high-speed digital computer capable of performing the aircraft simulation program in real-time represents only a portion of the total simulation problem. The necessary adjuncts to the computer system are the "real-world" environment, consisting primarily of a replica of the aircraft cockpit, and the computer program, which provides the instructions and numerical data that will enable the computer to solve the logical and the mathematical equations representing the condition of the aircraft and its associated subsystems.

The computer system communicates with the "real world" represented by the aircraft cockpit through the computer input-output unit. The five types of intercommunications were described superficially in Section 2.2. The control of these interconnecting links is exercised by the computer through its control unit, as directed by the simulation program.

More and more, the essential importance of the computer program becomes evident. It is by means of the program that not only the immediate problem is solved but also the flexibility of the digital computer system is most readily exploited. The structure of such a program must be initially organized to allow program modification with a minimum of effort and time. The minimum extent of organization is depicted in the simplified flow diagram of the F-100A simulation program, figure 4. Each block of the diagram represents the computation of a particular set of inter-related variables; there is minimal program dependence between blocks. As a result, modifying any program block has relatively little effect on the other program blocks.

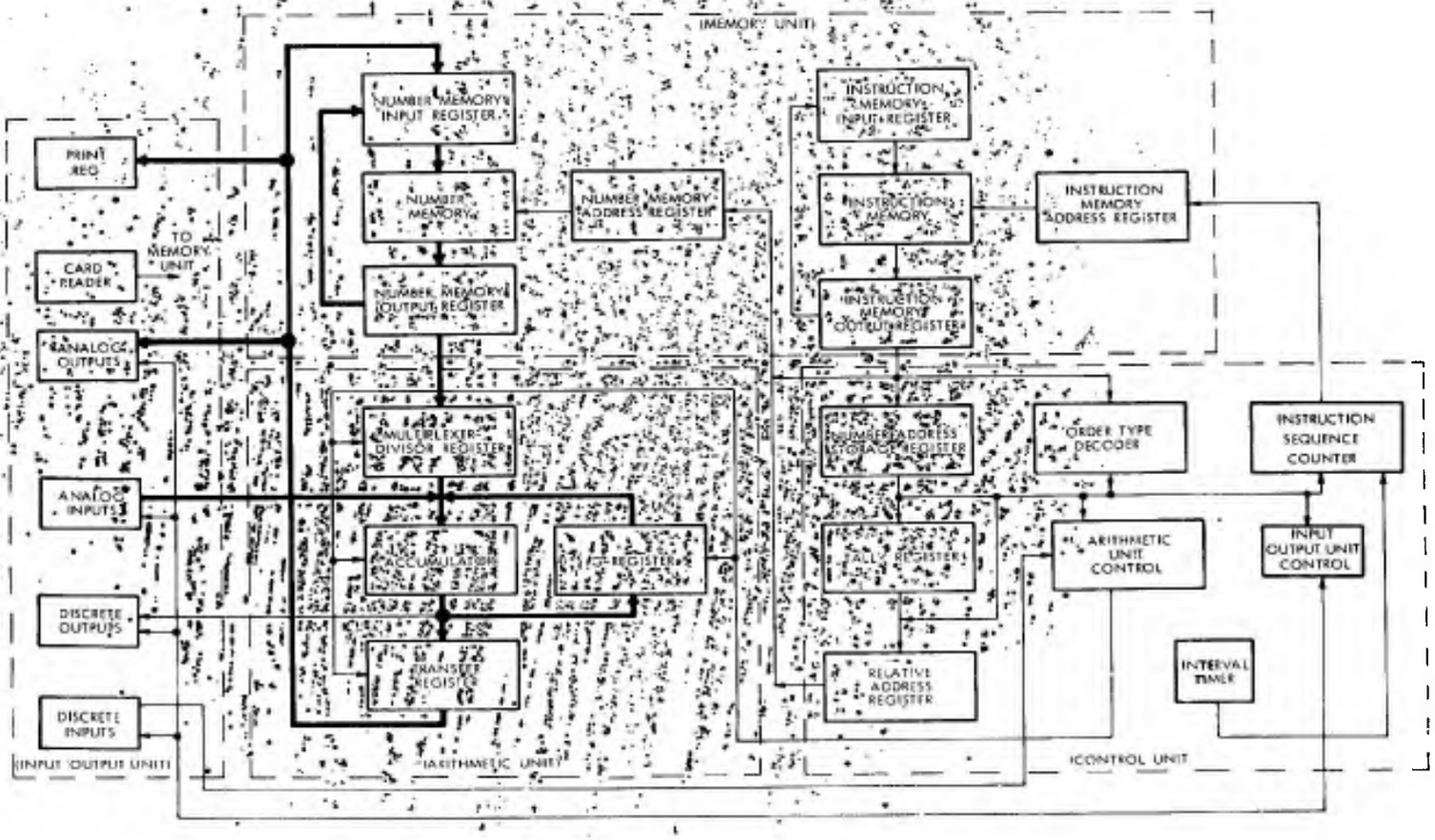


Figure 3. Block Diagram of UDOFT Computer

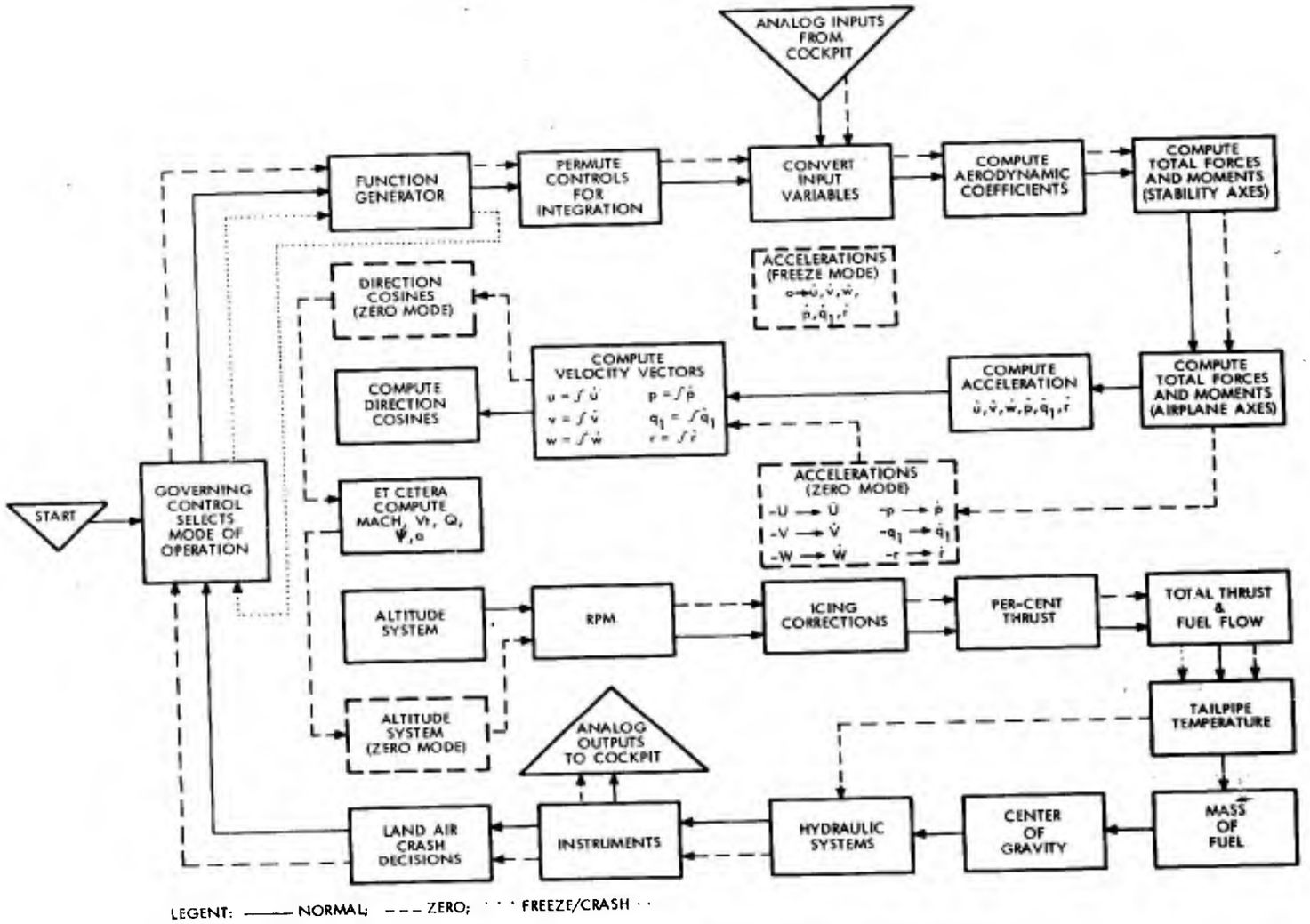


Figure 4. Flow Diagram of F-100A Aircraft Simulation Program

There are essentially three modes of program execution, as shown in figure 4: normal, freeze/crash and zero. Normal mode operation accomplishes real-time flight simulation. Freeze mode operation permits suspension of the training problem to allow the instructor to criticize a pilot-trainee's efforts. When the freeze mode is entered, the instruments are maintained at their last computed value, and simulation is suspended.

If, during a simulated flight, a trainee maneuvers in a manner which the decision routine recognizes as a crash condition, a crash is indicated and operation is automatically transferred from the normal mode to the freeze mode, thereby allowing a post mortem of the conditions that caused the crash to occur. To leave the crash mode the zero mode must be entered returning variables and instruments to their earlier values. Also, if the instructor desires to land the aircraft artificially the zero mode will effect this in a matter of seconds.

The complete simulation program for the F-100A requires approximately 3850 instructions and 3400 numbers. While operating in the normal mode, it requires approximately 35 milliseconds to obtain one complete solution of the problem. With the less complex F9F-2 aircraft, the complete solution is executed in approximately a 20-millisecond interval. In order to maintain a constant time interval, which is essential, the computer idles until the interval timer indicates that the 50 millisecond interval has been consumed, at which time the computation scheme is resumed.

SECTION III

UDOFT COMPUTER DESCRIPTION

3.1 Introduction

The UDOFT computer, though designed and developed specifically for solving the real-time aircraft simulation problem, has the same basic organization as any general purpose digital computer. The resemblance between UDOFT and a general purpose digital computer is evident from a comparison of the block diagram of the fundamental computer, figure 1, and the simplified block diagram of the UDOFT computer, figure 3.

This section of the report delineates the UDOFT computer's four functional units and the major component parts of each unit are further described in detail. The function and the implementation of these major components and their integration into the computer are presented in general terms. Before the functional units of the UDOFT computer may be discussed however, a description of the UDOFT "language" is essential.

3.2 Word Format

In the UDOFT computer the word or basic unit of information consists of twenty-two bits. Words may represent instructions or numbers.

a. Number Word

Since the computer has been organized to handle a number whose magnitude is less than unity, it is understood that the binary point is fixed at a position immediately preceding the most significant bit of the number word. Further, since the length of the number word is finite, the maximum magnitude which can be achieved is unity less the value of the least significant bit of the number word, $(1 - 2^{-20})$. In order to extend the range of the number representation and to facilitate working with quantities which are less than zero, the number is signed; thus, the total range of numbers which the UDOFT computer can understand is $(1 - 2^{-20})$ to $-(1 - 2^{-20})$. If numbers outside of this range are to be represented, they must be scaled appropriately.

The bit pattern of the number word is shown in figure 5.

P	S	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Figure 5. Number Word Format

The P position represents the parity bit; the S position, the sign bit; and positions 1 - 20, the magnitude of the quantity.

Since it is cumbersome to think of the numbers in their binary form, the binary numbers are translated into octal numbers. This translation reduces the number of descriptive "digits" from twenty to seven. The translation is simple, requiring only a grouping of the binary bits into groups of three successive bits and converting the three bit binary groups into decimal form. The method of grouping and an example of a translation is shown in figure 6.

BIT POSITION	P	S	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
BINARY REPRESENTATION	1	1	0	1	1	1	0	1	0	0	0	1	1	0	0	1	0	1	1	1	0	1
OCTAL WEIGHT FACTORS			4	2	1	4	2	1	4	2	1	4	2	1	4	2	1	4	2	1	4	2
OCTAL REPRESENTATION		-	3			5			0			6			2			7			2	

Figure 6. Number Word Translation from Binary to Octal

A "one" in the sign position indicates a negative quantity and a "zero" indicates a positive quantity. The parity bit serves as an error checking device by detecting the most common types of memory failure, gaining or losing single bits. The parity bit contributes nothing to the value of the word, but makes the word's total number of ones odd, hence the term odd parity. As each word is stored in memory, parity is formed and the appropriate value of the parity bit, "one" or "zero", is stored with it as part of the memory word. Parity is checked each time a word is used. Incorrect parity will indicate an error. In the example of figure 6, the parity bit assumed the value of "one" in order to satisfy this condition.

b. Instruction Word

The instruction word in the UDOFT computer consists of twenty binary bits (two of the twenty-two available bits are not used). The instruction word is divided into two "fields", the order type field (OT) and the number memory address field (NMAD).

BIT POSITION	P	R	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
FIELD		REL	0			T			N			M			A			D			

Figure 7. Instruction Word Format

The bit pattern and the division into field of the instruction word is shown in figure 7. The number memory address occupies the twelve low order bit positions of the word and the order type, the next six bit positions. What would normally be the nineteenth bit is defined as the relative bit. In general, the relative bit indicates whether relative addressing will be used by the next instruction. Last but not least is the parity bit, serving the same function as the parity bit for the number word.

The order type specifies any one of the thirty-two different orders that the computer can execute. Thirty-one instructions perform the basic program functions; clerical, arithmetic, control, input/output and special purpose. These functions and the individual instructions will be discussed in detail in Section 3. 3.

The number memory address field also serves a variety of purposes.

1. In clerical and arithmetic instructions, it specifies the location (address) of the operand in memory or the direction and number of places a quantity is to be shifted in the arithmetic unit.

2. In the control instructions, it specifies primarily an instruction sequence counter setting.

3. In the input/output instructions it specifies an input/output channel.

As in the case of the number words, the instruction words are also thought of in octal form rather than the binary form. The grouping of the bits and an example of the binary-to-octal translation is shown in figure 8.

BIT POSITION	P	R	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
BINARY REPRESENTATION	1	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	0	0	0
OCTAL WEIGHT FACTORS			4	2	1	4	2	1	4	2	1	4	2	1	4	2	1	4	2	1
OCTAL REPRESENTATION		R	3			4			3			0			3			0		

Figure 8. Instruction Word Translation from Binary to Octal

The instruction represented in figure 8 is $R343030_8$. The subscript eight indicates that the word is in octal form.

3.3 Instruction Repertoire

This section contains:

- a listing of UDOFT computer's thirty-two instructions categorized into groups according to their functions
- a brief description of the function of each group of instructions
- definitions of terms used in conjunction with the instructions
- a table of symbolic descriptions of each instruction, and
- a description of address modification as implemented in the UDOFT computer.

3.3.1 Arithmetic Instructions

There are nine arithmetic instructions in the UDOFT instruction repertoire. These instructions effect the four basic arithmetic operations of addition, subtraction, multiplication, and division. In addition to the basic operations there is a variation of both the addition and subtraction operations, a combination of the multiplication and addition operations, and shifting operations. The shifting operations are considered as arithmetic operations since they perform, in reality, multiplication and division by powers of two. These and the other UDOFT instructions appear in table II.

ADD	Add
ADM	Add magnitude
SUB	Subtract
SBM	Subtract Magnitude
MPY	Multiply
DIV	Divide
SHL, SHR	Shift Left, Shift Right
SHLA, SHRA	Shift Left and Add, Shift Right and Add
MAD	Multiply and Add

3.3.2 Clerical Instructions

The clerical instructions are those which cause the transfer of numerical quantities between various registers of the computer. In the majority of general purpose computers, most registers, be they memory registers or not, are addressable. In the case of the UDOFT computer, the only non-memory register which is addressable is the G-Register; any other register to which information may be transferred to is accessable only by means of special instructions. Thus, the UDOFT computer does not afford the flexibility of register accessibility found in other general purpose digital computers.

The primary avenue of data transfer in the UDOFT computer is between the Memory and the Arithmetic Unit. To facilitate these transfers, the following seven instructions are available.

CLA	Clear and Add
CLS	Clear and Subtract
CLAA	Clear and Add Magnitude

CLAS Clear and Subtract Magnitude

STO Store

TCA Transfer (store) Clear and Add

TIM Transfer to (Store in) instruction memory

Use of TIM Instruction

The use of two independent magnetic core memories, one exclusively for instruction words and the other, number words, has advantages and disadvantages. An independent Instruction Memory was expected to increase system reliability in addition to gain in computer speed despite the use of relatively low-speed memories. This conclusion is drawn from the premise that no writing of new information into the Instruction Memory is required once the program is initiated, therefore, the likelihood of damaging the computer program becomes almost non-existant. This premise is only half true since in magnetic core memory read-out of information is destructive, thereby, a rewriting of the information immediately following the read-out is necessary. Even with this fact it was still felt that loss of program control would be reduced if programmed access to the Instruction Memory were not allowed. This is the prime reason why there is no facility for program modification, other than relative addressing, in the UDOFT computer.

Without any programmed access to the Instruction Memory, the automatic testing of this memory becomes impossible. For this reason, more than any other, the TIM instruction (Transfer-to-Instruction-Memory) was introduced into the computer. The TIM instruction allows the transfer of information from the Transfer Register into the Instruction Memory. To ensure that this instruction is used only on test programs, a console switch is installed to enable the execution of this instruction. This switch must be in the "OFF" condition when simulation programs are being performed. When disabled, the TIM instruction is interpreted as a no-operation (NOP) instruction.

3. 3. 3 Control Instructions

The seven control instructions in the UDOFT computer, allow the sequence of instruction execution to be modified. By the use of these instructions it is not necessary therefore to prepare the computer program such that an unalterable sequence of instructions must be followed. In addition to easing the program preparation task, it also allows more efficient use of computer instructions and time. Were it not for these instructions, it would be virtually impossible to utilize subroutines within the total program. The utilization of such sub-routines eliminates repetitive programming which results in savings of instruction storage.

Control instructions are sub-divided into three classes; unconditional control, conditional control and programmed HALT. Unconditional control instructions exercise arbitrary control of the program. The UDOFT computer has two such control instructions; Sequence Counter Reset and Sequence Counter Reset to Content of Number Memory. Each and every time one of these instructions is encountered in the program, positive transfer of control to some other portion of the program is affected.

The conditional control instructions on the other hand provide the computer with a degree of decision-making capability. The transfer of control is dependent on or conditional to some previous action. For this reason they are differentiated from the unconditional control instructions. The UDOFT computer has four such instructions: Transfer on Minus, Transfer on Overflow, Transfer on Zero, and Sense Interval Time. Whenever these instructions are encountered in the program the transfer of program control may or may not be affected, depending upon the results of the preceding instruction.

The Programmed HALT instruction will, if enabled by a console switch, halt the computer.

The control instructions are:

SCR	Sequence Counter Reset to NMAD
SCRNM	Sequence Counter Reset to Content of Number Memory
TOV	Transfer on Overflow
TOZ	Transfer on Zero
TOM	Transfer on Minus
SENIT	Sense Interval Timer
HALT	Halt

3.3.4 Input-Output Instructions

The UDOFT input-output capability, as noted previously in Section II, is quite limited, having been designed with the intent of optimizing the transmission of data or information between the computer and the real-world represented by the mockup of the aircraft flight compartment. The real-world of the flight compartment is predominantly analog in nature as opposed to the distinctively digital nature of the computer. The input-output instructions of the UDOFT computer therefore must be capable not only of performing the basic task of data transfer but also of transforming or converting the data from one form to another.

The first form of computer communication with the external analog environment is the analog input. The analog inputs to the UDOFT computer are identical to those independent-variable inputs that are implemented with potentiometers in analog flight simulators. It would have been possible to retain this input form in the digital system by using an analog-to-digital converter to provide the necessary interface compatibility. The converter would have been a single channel device, capable of operating upon a single input at a time. At the time that UDOFT was developed, available analog-to-digital converters were both slow and costly. To circumvent these two problems it was decided that, since each input was a mechanical shaft of one form or another, digital shaft encoders would be used. Further, to attain an acceptable level of accuracy and to minimize ambiguity of reading the shaft angle, the encoders to be used would be ten bit, Gray-coded binary.

Using the encoders eliminates the need for the analog-to-digital converter; however, the Gray-coded binary encoders requires the use of a Gray-coded binary-to-conventional-binary-converter. This type of converter is relatively simple to implement; further, its resolution time is considerably short.

The Multiplex Analog Input instruction provides the means by which the computer is instructed to process the analog inputs. Since there are twenty-four kinds of analog input that may be examined, this instruction must be capable of designating a particular input in addition to enabling the converter, and transferring the converted quantity to the ten high order bit positions of the Accumulator. The designation of the particular input to be processed is contained in the low order seven bits of the address field of the instruction. Once the data has been entered into the Accumulator, it may be handled like any other number in the computer.

Just as there is the need for the capability to process real-time input data, there is the need also for processing real-time output data. The ultimate form of the outputs which represent continuously varying quantities is a D.C. voltage. This requires the conversion, within the computer, of digital data to a proportional D.C. voltage.

For reasons of economy it was decided to use a single time-shared digital-to-analog converter rather than individual converters for each of the sixty-four analog output channels. However, using a single converter, it was necessary to provide means for storing or holding each of the analog voltages during the time the remaining analog output channels were being serviced. The individual holding device is simply a capacitor in a network which allows for the rapid change of the charge on the capacitor during the time when the particular channel is being serviced, yet provides an extremely high impedance discharge path during the off-time when the other output channels are being serviced.

The function of the Multiplex Analog Output instruction is to transfer the quantity in the Transfer Register to the digital-to-analog converter and to enable the analog output channel which is specified by the six low order bits of the address field of the instruction. Only the sign bit and the eleven most significant bits of the Transfer Register are involved in the transfer of data to the converter. Thus, the numbers that may be converted to analog outputs are limited to the range from $(-7776)_8$ to $(+7776)_8$. The number $(-7776)_8$ represents an analog output which is equivalent to zero; the number $(0000)_8$ is equivalent to a mid-range output; and the number $(+7776)_8$ is equivalent to maximum possible output.

An added feature of the instruction is the ability to store the quantity in Number Memory simultaneously with the outputting process; the storage location is derived from the full address.

In addition to the processing of continuously varying data between the computer and the real-world, the computer must be able to process discrete data. Discrete data defines

such limited variables as switch positions or the status of indicator lights. Since the communication of this data is necessarily a two-way process, both inputs and outputs of this type are essential. The inputs are referred to as discrete inputs; the outputs, as discrete outputs.

Discrete inputs are handled by the computer in such a manner that a computer instruction for this purpose is unnecessary; the discrete outputs, however, require an instruction.

The function of the Multiplex Discrete Output instruction is to set the specified discrete output channel to the appropriate state. The discrete output channels are implemented with locking type relays; once the relay is closed it will remain closed until directed to open. The state which the relay is directed to assume is controlled by the sign of the number in the Accumulator at the time the MXDO instruction is performed. If the sign of the number is negative, the relay is directed to close; i.e., the discrete output is on. If the sign is positive, the relay is directed to open; i.e., the discrete output is off. The selection of the appropriate discrete output channel is controlled by the ten low order bits of the address field.

The three input-output instructions discussed up to this point are concerned primarily with the communication of information between the computer and the mechanisms representing the simulation system man-machine interface. The Print Instruction is the fourth input-output instruction in the UDOFT computer. This instruction represents the only attempt, in the design of the computer, to achieve an ability to communicate digital information to the real-world. In truly general purpose digital computers this aspect of computer operation is of vital importance. In a digital computer such as UDOFT, which is developed for an application where digital information is of minor significance compared to the other forms of information transfer, this aspect is all-too-often neglected.

Though UDOFT was developed with the intent of using the system ultimately as a training device, it has not been so and probably will not be; instead it is being used as a simulation research tool. Anytime a digital computer is used for research purposes of any form, the accumulation of digital data is paramount to the expeditious execution of a problem. The Print Instruction attempts to satisfy this need.

The Print Instruction transfers the contents of the accumulator to a special output register, the Print Register. The Print Register acts as a buffer between the computer and other output devices. The print-out mechanism as originally conceived, was to be a variation of the conventional strip recorder. Rather than using a few movable styli, the UDOFT printer would have contained twenty-three fixed styli. The marks made by activating the styli would have provided a record of the data in binary form. Due to the many disadvantages associated with using this form of output, the on-line printout scheme was discarded. However the Print Instruction and the Print Register are implemented in the computer, and are used to control an additional register wheel which in turn controls the computer's peripheral equipment. The Input-Output instructions are:

MLXI	Multiplex In analog inputs
MXLO	Multiplex Out analog output
MXDO	Multiplex discrete outputs
PRNT	Print (Fast)

In order to provide some means for printing-out program or numerical information, an output printer is provided. Although a permanent record of desired information is available in the less cumbersome octal form, the printer capability is quite limited. No instruction is implemented in the computer for this operation; the initiation and the stopping of the operation is exercised manually.

The initial loading or reading-in of the computer program is performed also by means of a limited capability mechanism. No instruction is implemented in the computer for reading information contained on punched cards into the computer; control of the operation is exercised manually.

3.3.5 Special Purpose Instructions

The preceding instructions that have been implemented in the UDOFT computer are representative of the types of instructions found in most general purpose digital computers. The five remaining instructions are SIT, TAN, TAU, NOP and NOT whose functions are listed as follows:

Instruction	Function
SIT	Set Interval Timer - Load the interval timer
TAN	Tally Number Memory Address - Load the Tally Register
TAU	Tally Arithmetic Unit - Load the Tally Register
NOP	No Operation - Mark the time
NOT	Non Existant Order Type - A spare

Although the No Operation Instruction (NOP) appears to be an inefficient instruction, it is a necessity in the UDOFT computer. It does nothing but allow five microseconds to pass before execution of the next instruction can be initiated and transfer the contents of the Accumulator to the Transfer Register. It is used primarily to over come the problem of programming a forbidden sequence of instructions, (forbidden sequences are explained in Section 3.3.6) to transfer the quantity in the Accumulator to the Transfer Register in order that it can be stored in Number Memory by a following Store Instruction. It also provides a simple means for generating a relative address for the succeeding instruction. It is apparent then that the NOP instruction provides a means to an end; it is not an end in itself.

The NOT Instruction serves no real useful purpose; it is a spare or extra instruction. Because the computer will treat this Non-Existant Order Type instruction in a manner similar to the Programmed Halt instruction, it may be used as another manually-controlled program halting mechanism.

3.3.6 UDOFT Registers and Symbolic Description of UDOFT Instructions

Symbolic Notations

C(AC)	Refers to the contents of the Accumulator
C(G)	Refers to the contents of the G-Register
C(Y)	Denotes the contents of location Y, where Y refers to some generalized location in Number Memory
MAG(Y)	Denotes the absolute value of (Y)

Subscripts will be used to denote specific bit positions in a register. For example, $C(AC)_{S,\ 1-2}$ denotes the contents of the Accumulator involving only bit positions sign, one, and two.

The negative of a number is the number with its sign reversed. Similarly, the magnitude of a number is the number with its sign made positive.

J^{th} instruction means, in general, that instruction being performed or the instruction in question.

Similarly:

$(J + 2)^{nd}$	means the second instruction following the J_{th} instruction
$(J - 1)^{st}$	means the instruction preceding the J_{th} instruction

Forbidden Sequences

One of the methods employed in UDOFT to achieve high computation speeds is the use of independent instructions and number memories which overlap in time. This results in at least portions of two successive instructions overlapping. In special cases this overlapping may result in erroneous solutions. For this reason some sequences of Instructions are forbidden.

Accumulator (AC)

The Accumulator is a register having a capacity of 20 bits plus an independent sign. It is used in all arithmetic operations.

AC Overflow

Most arithmetic operations can cause an Accumulator overflow. The Accumulator has an overflow stage which is set to the "one" condition when an overflow occurs. The overflow stage, once set to a "one", remains set until an arithmetic operation is performed, at which time it is reset to "zero."

In some cases an AC overflow is a desired result. In others it is not. The computer has a transfer on overflow instruction which is a conditional transfer of control (TOV) dependent on the state of the overflow stage. The usual method of programming is to have the TOV immediately follow the anticipated overflow. An AC overflow immediately followed by a TOV instruction will perform the transfer of control, reset the overflow stage to a "zero," and prevent the overflow indicators from being set. An overflow not immediately followed by a TOV instruction will set the overflow stage to a "one" and cause one of the three overflow indicators (DIV, SHL, ADD) to be set. A TOV instruction not immediately following an overflow will function normally if the overflow has not been cleared by an arithmetic operation. However, one of the overflow indicators will be set. In slow computation an overflow not immediately followed by a TOV instruction will halt the computer.

AC Truncation

Since it is possible to lose information on an overflow, the computer has a provision for truncating the number at all ones. For example, if the truncate an ADD OVFL switch is on and ADD OVFL occurs, the AC will be cleared and all ones will be inserted. Truncation is possible on Add, Shift Left, and Divide OVFL.

AC Positive Zero

The Accumulator is a positive zero accumulator (i.e. no matter what the method of obtaining zero is, when the accumulator is zero the sign is positive).

Transfer Register (TR)

The Transfer Register is a register having a capacity of twenty-two bits, including: a twenty bit number, an independent sign bit, and an independent parity bit.

The Transfer Register serves the following purposes:

1. A buffer between the Accumulator and the memory
2. A buffer for the Multiplexer Output Register
3. A buffer for the fast print facility
4. A buffer between the card reader and the memory
5. In conjunction with the parity former to form the correct parity to be stored with the number.

The contents of the Transfer Register at the end of the J^{th} instruction are the $C(AC)^{(J-1)st}$ instruction, unless the J^{th} instruction is an STO or an MLXO, the previous C(TR) is erased. The STO and MLXO instructions do not change the contents of the TR.

G-Register

The G-Register is a special-purpose register having a capacity of 20 bits plus sign (no parity). The G-Register is used in conjunction with the shift add, multiply add, and divide instructions. It may be used as an intermediate register for many computations since it is directly addressable. Information is stored in the G-Register by the shift add and multiply add instructions.

The contents of the G-Register Address $(1000)_8$ can be made available by the following instructions: ADD, ADM, CLA, CLAA, CLAS, CLS, MPY, MAD, SBM, and SUB. The G-Register is automatically cleared following all of these instruction with the exception of MAD and SHA. In addition, the G-Register contents may be displayed on the MD Registers as indicated by a switch on the console. There is also a G-Register Clear Switch on the console to clear the G-Register.

Due to the G-Register address being $(1000)_8$, the address $(1000)_8$ of the number memory is not available, i.e. $(1000)_8$ is reserved for the G-Register. Also, NMAD $(1000)_8$ may not be used with DIV, SCRNM, SIT, STO and TCA

Sequence Counter

The "sequence counter" is a 12-bit binary register which determines the instruction memory location from which the next instruction is to be taken. The counter functions basically as an add one binary counter which can be reset.

Tally Register

The Tally Register is a register having a capacity of 12 bits, which is used in conjunction with relative addressing. (For use of Tally Register see section on relative address.) It is addressable by the TAU and TAN instructions as well as manually from the console.

Interval Timer

The Set Interval Timer (SIT) and the Sense Interval Timer (SENIT) instruction are very special purpose instructions that serve an important function on the UDOFT computer; they exercise control over the Interval Timer. The Interval Timer is in essence a real-time clock which counts down from some specified quantity to zero. The process of counting down is performed automatically and occurs in real-time. Thus, if a quantity which is proportional to 50 milliseconds is set into the interval timer, it will require 50 milliseconds for the timer to run down to zero. The purpose of the timer is to provide a "settable" and "sensible" real-time clock which can be used to establish fixed intervals of time. The requirement for fixed time intervals is generated by the fact that the numerical integration and other time dependent functions become extremely cumbersome if a variable time interval between successive solutions (iterations) is allowed. The Set Interval Timer facilitates the "settable" function; the Sense Interval Timer facilitates the "sensible" function.

TABLE II
SYMBOLIC DESCRIPTION OF UDOFT INSTRUCTIONS

MNEMONIC CODE	OCTAL CODE	TIME (IN μSEC)	DESCRIPTION	SYMBOLIC DESCRIPTION	RESULTS ASSUMING THIS TO BE THE J^{TH} INST	NMAD (14000)$_8$ VALID	AC OVERFLOW POSSIBLE	POSSIBLE SEQUENCES ASSUMING THIS TO BE THE J^{TH} INST
ADD	30	5	ADD	REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$ ADD C(Y) TO C(AC)	TR ← C(AC)$^{U-1)^{ST}}$ AC ← C(AC)$^{U-1)^{ST}}$ + C(Y)	YES	YES	$(U+1)^{ST}$ INSTRUCTION MAY NOT BE MLXI
ADM	32	5	ADD MAGNITUDE	REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$ ADD MAG. C(Y) TO C(AC)	TR ← C(AC)$^{U-1)^{ST}}$ AC ← C(AC)$^{U-1)^{ST}}$ + MAG C(Y)	YES	YES	SAME AS ADD
CLA	34	5	CLEAR AND ADD	REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$ REPLACE C(AC) WITH C(Y)	TR ← C(AC)$^{U-1)^{ST}}$ AC ← C(Y)	YES	NO	$(U+1)^{ST}$ INSTRUCTION MAY NOT BE MLXI $(U-1)^{ST}$ INSTRUCTION MAY NOT BE MAD, SHRA, SHLA
CLAA	36	5	CLEAR AND ADD MAGNITUDE	REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$ REPLACE C(AC) WITH MAG. C(Y)	TR ← C(AC)$^{U-1)^{ST}}$ AC ← MAG. C(Y)	YES	NO	SAME AS CLA
CLAS	37	5	CLEAR AND SUBTRACT MAGNITUDE	REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$ REPLACE C(AC) WITH -MAG C(Y)	TR ← C(AC)$^{U-1)^{ST}}$ AC ← -MAG. C(Y)	YES	NO	SAME AS CLA
CLS	35	5	CLEAR AND SUBTRACT	REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$ REPLACE C(AC) WITH -C(Y)	TR ← C(AC)$^{U-1)^{ST}}$ AC ← -C(Y)	YES	NO	SAME AS CLA
DIV	20	105	DIVIDE	REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$ DIV C(AC) BY C(Y)	TR ← C(AC)$^{U-1)^{ST}}$ AC ← QUOTIENT OF $\frac{C(AC)}{C(Y)}$	NO	YES	$(U-1)^{ST}$ INST. MAY NOT BE MLXI OR TOZ $(U-1)^{ST}$ INST MAY NOT HAVE NMAD $(14000)_8$ $(U-1)^{ST}$ INST. MAY NOT BE MAD, SHRA, SHLA $(U-2)^{ND}$ INST. MAY NOT BE MAD, SHRA, SHLA
MAD	26	10	MULTIPLY AND ADD G REGISTER	REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$ MULTIPLY C(Y) BY THE C(AC) AND ADD THE C(G) TO THE PRODUCT	TR ← C(AC)$^{U-1)^{ST}}$ AC ← ϕ G ← REG ← [C(Y) × C(AC)] + CLG$^{U-1)^{ST}}$	YES	YES	$(U-1)^{ST}$ MAY NOT BE SHR, OR MLXI, CLA, CLAA, CLS, CLAS, STORE G, LXY, MLXO, TOZ, TOM, MLXI, OR ANY INSTRUCTION REFERENCING G THE $G+1)^{ND}$ MAY NOT BE MLXO, MLXI, TOZ, TOM, LXY, STO, PRNT, MLXO, TCA, STO PRNT, MLXO MAY BE THE $(U+1)^{ST}$ INST NOT THE $(U-1)^{ND}$
MLXI	12	10	MULTIPLEX IN ANALOG INPUT	REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$ CLEAR AC, ADD THE BINARY REPRESENTATION OF THE ANALOG INPUT SELECTED TO $(AC)_{1-16}$	TR ← C(AC)$^{U-1)^{ST}}$ AC ← BINARY REPRESENTATION OF THE ANALOG INPUT	YES	NO	$(U-1)^{ST}$ INST. MAY NOT BE ANY ARITHMETIC OR CLERICAL INSTRUCTION $(U-1)^{ST}$ INST MAY NOT BE MLXI $(U+1)^{ST}$ MAY NOT BE STO TCA OR MLXO
MLXO	16	5	MULTIPLEX OUT ANALOG OUTPUT	CONVERT $C(TR)_{8,11}$ TO AN ANALOG VOLTAGE AND APPLY TO THE SELECTED MULTIPLEXER CHANNEL. IF BIT 11 OF THE INSTRUCTION IS A ONE, THE $C(TR)_{5,1-20}$ WILL BE STORED IN THE LOCATION SPECIFIED BY THE NMAD	TR ← NO CHANGE AC ← NO CHANGE MULTIPLEXER OUTPUT REG. IS SET BY $C(TR)^{U-1)^{ST}}_{5,1-11}$	YES	NO	$(U+1)$ MAY NOT BE MLXI $(U-2)^{ND}$ INST. MAY NOT BE MAD, SHLA, OR SHRA
MPY	27	10	MULTIPLY	REPLACE C(TR) BY C(AC)$^{U-1)^{ST}}$ MULTIPLY C(Y) BY C(AC)	TR ← C(AC)$^{U-1)^{ST}}$ AC ← PRODUCT OF C(Y) AND C(AC)	YES	NO	$(U+1)^{ST}$ MAY NOT BE TOZ, TOM, MLXI, OR MLXO
MXDO	04	5	MULTIPLEX DISCRETE OUTPUTS	REPLACE C(TR) BY C(AC)$^{U-1)^{ST}}$ IF SIGN OF AC IS NEGATIVE, TURN ON SELECTED DISCRETE OUTPUT. IF SIGN OF AC IS POSITIVE, TURN OFF DISCRETE OUTPUT	TR ← C(AC)$^{U-1)^{ST}}$ AC ← NO CHANGE	YES	NO	SAME AS TOM
NOP	14	5	NO OPERATION	REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$	TR ← C(AC)$^{U-1)^{ST}}$ AC ← NO CHANGE	YES	NO	
PHT	00	5	PROGRAMMED HALT	REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$ HALTS THE COMPUTER IF ENABLED BY ITS STOP OR ERROR SWITCH OTHERWISE IT IS THE SAME AS NOP	TR ← C(AC)$^{U-1)^{ST}}$ AC ← NO CHANGE	YES	NO	
PRNT	05	5	PRINT	GATES C(AC)$^{U-1)^{ST}}$ INTO PRINT REGISTER BIT 18 OF THE INST. IS USED AS A MARKER. REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$ PRINT MUST BE ENABLED BY THE PRINT SWITCH, OTHERWISE IT IS THE SAME AS NOP	PRINT REG. ← $C(TR)^{U-1)^{ST}}_{8}$ TR ← C(AC)$^{U-1)^{ST}}$ AC ← NO CHANGE	YES	NO	$(U-2)^{ND}$ INST. MAY NOT BE MAD, SHRA OR SHLA
SBM	33	5	SUBTRACT MAGNITUDE	REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$ SUB. MAG. (Y) FROM C(AC)	TR ← C(AC)$^{U-1)^{ST}}$ AC ← C(AC) - MAG C(Y)	YES	YES	SAME AS ADD
SCR	03	5	SEQUENCE COUNTER RESET	REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$ RESET SEQUENCE COUNTER TO Y	TR ← C(AC)$^{U-1)^{ST}}$ AC ← NO CHANGE SEQ. COUNTER ← Y	YES	NO	
SCRNM	02	10	SEQUENCE COUNTER RESET TO CONTENT OF NUMBER MEMORY	REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$ SET SEQ. COUNTER TO $C(Y)_{4-15}$	TR ← C(AC)$^{U-1)^{ST}}$ AC ← NO CHANGE SEQ. COUNTER ← $C(Y)_{4-15}$	NO	NO	
SENIT	07	5	SENSE INTERVAL TIMER	REPLACE C(TR) WITH C(AC)$^{U-1)^{ST}}$ A) IF INTERVAL TIMER=0, PERFORM THE NEXT INSTRUCTION IN SEQUENCE. B) IF INTERVAL TIMER (IT) ≠ 0 CLEAR AC, ADD C(IT) TO $(AC)_{2-18}$ RESET SEQ. COUNTER TO (Y)	TR ← C(AC)$^{U-1)^{ST}}$ A) IF IT = 0 AC ← NO CHANGE B) IF IT ≠ 0 AC ← $C(IT)^{U-1)^{ST}}_{3-18}$ -1 SEQ. COUNTER ← Y	YES	NO	SAME AS CLA

TABLE II (Cont.)

SYMBOLIC DESCRIPTION OF UDOFT INSTRUCTIONS

SYMBOLIC DESCRIPTION OF UDOFT INSTRUCTIONS

MNEMONIC CODE	OCTAL CODE	TIME IN μSEC.	DESCRIPTION	SYMBOLIC DESCRIPTION	RESULTS ASSUMING THIS TO BE THE J^{TH} INST.	NMAD (14000)$_8$ VALID	AC OVER FLOW POSSIBLE	FORBIDDEN SEQUENCES ASSUMING THIS TO BE THE J^{TH} INST.
SHLA SHRA	24	5	SHIFT LEFT OR RIGHT DIRECTION AND NO. OF PLACES — ADD C(G) NMAD IN OCTAL RIGHT 1 6014; 2 6012; 3 6036; 4 6032; 5 6076; 6 6072; 7 6176; 8 6172 LEFT 0 6660; 1 6663; 2 6615; 3 6635; 4 6675; 5 6175	REPLACE C(TR) WITH C(AC)$^{(J-1)^{ST}}$ SHIFT AC ADD C(G) TO THE SHIFTED(AC) AND STO IN G	TR ← C(AC)$^{(J-1)^{ST}}$ AC ← 6 G ← SUM OF SHIFTED C(AC) AND C(G)$^{(J-1)^{ST}}$	YES	YES	SAME AS MAD
SHL SHR	25	5	SHIFT LEFT OR RIGHT DIRECTION AND NO. OF PLACES — NMAD IN OCTAL RIGHT 1 6016; 2 6012; 3 6036; 4 6032; 5 6076; 6 6072; 7 6136; 8 6172 LEFT 0 6660; 1 6663; 2 6615; 3 6635; 4 6675; 5 6175	REPLACE C(TR) WITH C(AC)$^{(J-1)^{ST}}$ SHIFT AC	TR ← C(AC)$^{(J-1)^{ST}}$ AC ← SHIFTED C(AC)$^{(J-1)^{ST}}$	YES	YES	$(J+1)^{ST}$ INST. MAY NOT BE MLXI, MXDD, TOM, TOZ $(J-1)^{ST}$ INST. MAY NOT BE MAD, SHRA, SHLA
SIT	13	10	SET INTERVAL TIMER	REPLACE C(TR) WITH C(AC)$^{(J-1)^{ST}}$ SIT TO C(Y)$_{1-14}$	TR ← C(AC)$^{(J-1)^{ST}}$ AC ← NO CHANGE IT ← C(Y)$_{1-14}$	NO	NO	
STO	23	5	STORE	REPLACE C(Y) WITH C(TR)	TR ← NO CHANGE AC ← NO CHANGE C(Y) ← C(TR)$_{(J-1)^{ST}}$	NO	NO	SAME AS MLXO
SUB	31	5	SUBTRACT	REPLACE C(TR) WITH C(AC)$^{(J-1)^{ST}}$ SUB. C(Y) FROM C(AC)	TR ← C(AC)$^{(J-1)^{ST}}$ AC ← C(AC) -C(Y)	YES	YES	SAME AS ADD
TAN	11	10	TALLY NUMBER ADDRESS	REPLACE C(TR) WITH C(AC)$^{(J-1)^{ST}}$ IF NEITHER TAN NOR THE $(J-1)^{ST}$ INST. IS RELATIVE, REPLACE C(TALLY REG) WITH Y IF EITHER TAN OR THE $(J-1)^{ST}$ INST. IS RELATIVE, REPLACE C(TALLY REG) BY SUM OF C(TALLY REG)$^{(J-1)^{ST}}$ AND Y. NOTE CONSULT SECTION ON RELATIVE ADDRESSING FOR FURTHER INFORMATION	TR ← C(AC)$^{(J-1)^{ST}}$ AC ← NO CHANGE IF R OF J^{TH} OR $(J-1)^{ST}$ INST. IS 0, TALLY REG. ← Y IF R OF J^{TH} OR $(J-1)^{ST}$ INST. = 1, TALLY REG. ← Y + (C(TALLY REG.)$^{(J-1)^{ST}}$	YES	NO	
TAU	01	10	TALLY ARITHMETIC UNIT	REPLACE C(TR) WITH C(AC)$^{(J-1)^{ST}}$ REPLACE C(TALLY REG) WITH C(AC)$_{4-15}$ IF BIT 11 OF THE INSTRUCTION IS A ONE, THE C(Y) ARE REPLACED BY C(AC)$_{5,1-20}$ NOTE CONSULT SECTION ON RELATIVE ADDRESSING FOR FURTHER INFORMATION	TR ← C(AC)$^{(J-1)^{ST}}$ C(AC) NO CHANGE C(TALLY REG.) ← C(AC)$^{(J-1)^{ST}}_{4-15}$ IF BIT 11 IS A ONE C(Y) ← C(AC)$^{(J-1)^{ST}}_{5,1-20}$	YES	NO	
TCA	21	10	TRANSFER CLEAR AND ADD	REPLACE C(Y) WITH C(TR)$^{(J-1)^{ST}}$ REPLACE C(TR) WITH C(AC)$^{(J-1)^{ST}}$ REPLACE C(AC) WITH C(Y)$^{(J-1)^{ST}}$	INITIAL LOCATION — FINAL LOCATION TR — Y Y — AC AC — TR	NO	NO	$(J+1)^{ST}$ MAY NOT BE MLXI, $(J-1)^{ST}$ OR $(J-2)^{ND}$ MAY NOT BE MAD, SHRA, SHLA, OR MLXI
TIM	17	10	TRANSFER TO INSTRUCTION MEMORY	TIM MUST BE ENABLED BY THE CSE TIM SWITCH, OTHERWISE IT WILL BE THE SAME AS NOP REPLACE C(Y) OF INST. MEMORY WITH C(TR)$_{2-20}$ TR BIT 2 CORRESPONDS TO THE RELATIVE BIT WHILE BITS TR$_{3-20}$ CORRESPOND TO BITS 1-18 OF THE INSTRUCTION REPLACE C(TR) WITH C(AC)$^{(J-1)^{ST}}$	TR ← C(AC)$^{(J-1)^{ST}}$ AC ← NO CHANGE INST MEMORY C(Y)$_{R,1-18}$ ← C(AC)$_{1,2-20}$	YES	NO	
TOM	22	5	TRANSFER ON MINUS	REPLACE C(TR) WITH C(AC)$^{(J-1)^{ST}}$ IF C(AC) IS NEGATIVE, RESET SEQUENCE COUNTER TO Y. IF C(AC) IS POSITIVE, PERFORM NEXT INSTRUCTION IN SEQUENCE	TR ← C(AC)$^{(J-1)^{ST}}$ AC ← NO CHANGE IF C(AC) IS NEGATIVE SEQUENCE COUNTER ← Y.	YES	NO	$(J-1)^{ST}$ MAY NOT BE MAD, MPY, SHL, SHR, SHLA, SHRA $(J-2)^{ND}$ MAY NOT BE MAD, SHLA, SHRA
TOZ	06	5	TRANSFER ON ZERO	REPLACE C(TR) WITH C(AC)$^{(J-1)^{ST}}$ IF C(AC) = 0, RESET SEQUENCE COUNTER TO Y. IF C(AC) ≠ 0, PERFORM THE NEXT INSTRUCTION IN SEQUENCE.	TR ← C(AC)$^{(J-1)^{ST}}$ AC ← NO CHANGE IF C(AC) = 0 SEQUENCE COUNTER ← Y	YES	NO	$(J-1)^{ST}$ MAY NOT BE DIV, MAD, MPY, SHL, SHR, SHLA, SHRA $(J-2)^{ND}$ MAY NOT BE MAD, SHLA, SHRA
TOV	10	10	TRANSFER ON OVERFLOW	REPLACE C(TR) WITH C(AC)$^{(J-1)^{ST}}$ IF THE ACCUMULATOR IS IN THE OVERFLOW CONDITION, THE SEQUENCE COUNTER IS RESET TO Y. IF NOT, THE NEXT INSTRUCTION IN SEQUENCE IS PERFORMED	TR ← C(AC)$^{(J-1)^{ST}}$ AC ← NO CHANGE (AC) OVERFLOW SEQUENCE COUNTER ← Y.	YES	NO	

3.3.7 Address Modification (Relative Addressing)

UDOFT makes use of an index register, called the Tally Register, to effectively modify the address of instructions. Address modification facilitates the programming of iterative programs or subroutines. A program subroutine is defined loosely as a set of instructions which is used repeatedly to solve the same problem only each time using a different set of variables. A prime example of a UDOFT program subroutine is the numerical integration subroutine. In all, this subroutine is used twelve times; to integrate the three translational accelerations ($\dot{u}$, $\dot{v}$, and $\dot{w}$) and the three rotational accelerations, ($\dot{p}$, $\dot{q}_1$ and $\dot{r}$), and to integrate the three derivatives of the direction cosines for the y-inertial axis ($\dot{l}_2$, $\dot{m}_2$, and $\dot{n}_2$) and the three derivatives of the direction cosines for the z-inertial axis ($\dot{l}_3$, $\dot{m}_3$, and $\dot{n}_3$).

It would be extremely wasteful of computer instructions (that must be stored in the Instruction Memory) if each of the twelve integrations required its own explicit program or routine. Since the only difference between the individual programs is the independent variable, a means for using the same set of instructions with the facility of modifying the address of the variable. This facility is provided in the UDOFT computer by the feature of relative addressing; in contemporary computers this feature is normally referred to as "indexing."

Address modifications is possible with all 32 UDOFT instructions. If Address modification is to occur the relative bit of the previous instruction must be a "one." When this condition is met the address used is the content of the relative address register.

The relative address register always contains the arithmetic sum of the number memory address or the previous instruction and the contents of the Tally Register at the beginning of the previous instruction.

The Tally Register is loaded by the TAN and the TAU instructions. The TAN instruction loads the Tally Register with the number memory address coded with the TAN instruction. The TAU instruction loads the Tally Register with the contents of the Accumulator bits six through seventeen.

In addition the TAN instruction has a special feature which expedites updating the Tally Register. If the relative bit coded with the TAN instruction, or the previous instruction, is a "one" the contents of the Tally Register at the beginning of the TAN instruction will be augmented by the number memory address coded with the TAN instruction.

The number memory address of the previous instruction is used in relative addressing to allow time for the Tally Register to be added to the number memory address.

3.4 Main Frame

The Arithmetic Unit, Control Unit I, and Control Unit II, taken collectively, comprise what is commonly referred to as the Main Frame, (figures 9 and 10). The function of the Main Frame is to perform all the internal operations of the computer system and to control the operation of the remaining computer units, the Memory Unit and the Input-Output Unit. Had the implementation of the arithmetic and the control functions in the UDOFT computer been more efficient, there would not have been these three separate units. Instead, a single physical unit, the Main Frame, requiring less physical volume for the hardware, would have been evolved. The main reason for considering these three units as a single operating entity is the interdependence between the units. There is more communication among these units than there is between these units and the Memory and the Input-Output Units.

Briefly, the Arithmetic Unit functions as the computational element of the Main Frame. Control Unit I performs all the control functions associated with the Arithmetic Unit. Control Unit II performs all the control functions associated with the Memory and the Input-Output Units, in addition to the supervisory control of the total computer. This section of the report will treat superficially each of the three units, delineating the major constituent elements and indicating their functions. However, in order to understand the operation and some of the idiosyncrasies of these major elements, a brief description of the master timing system of the computer is presented first.

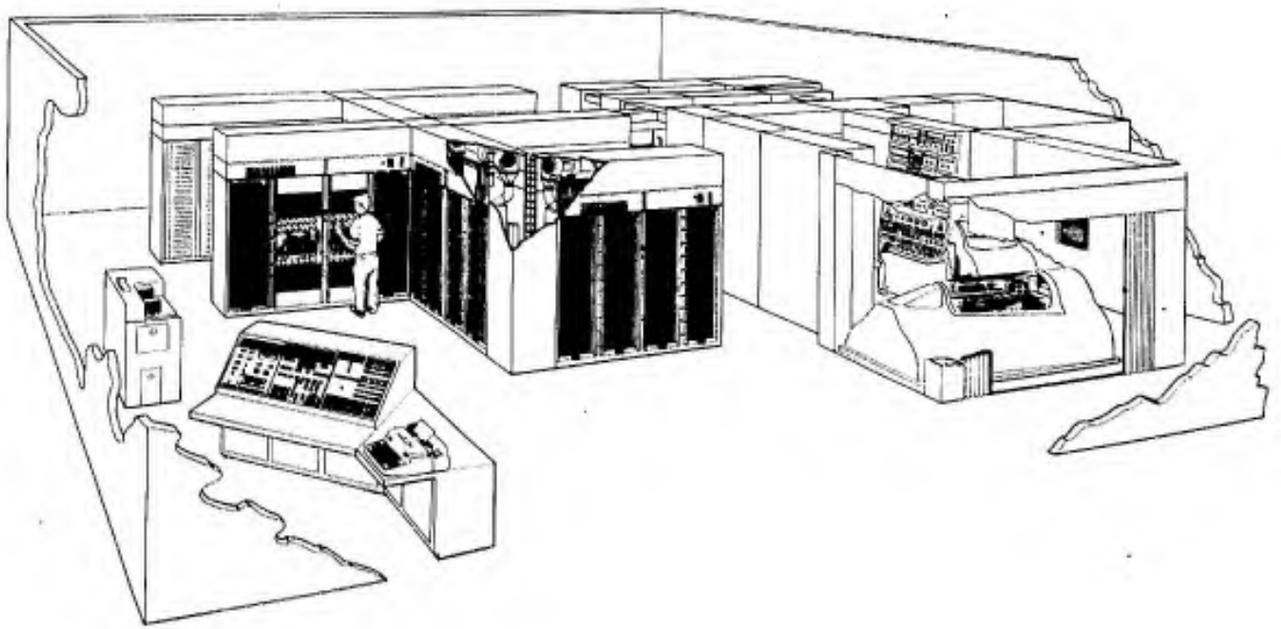


Figure 9. UDOFT Installation

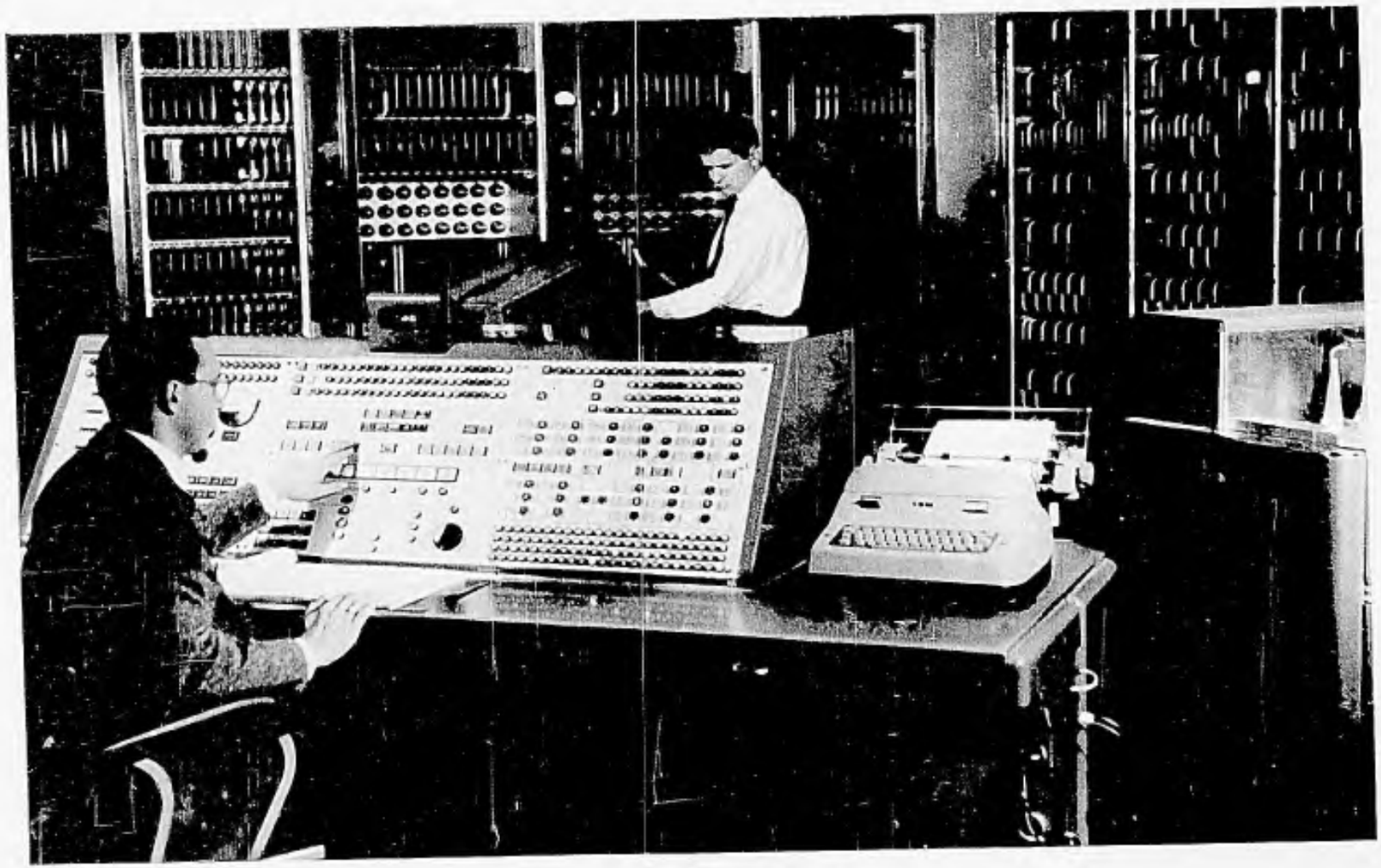


Figure 10. UDOFT Computer

3.4.1 Master Timing System

The UDOFT computer is a synchronous machine which uses dynamic logic. Nearly all large scale digital computers are synchrous, but few contemporary computers use dynamic logic. Dynamic logic enjoyed great popularity in the early 1950's, prior to the appearance of the high-frequency junction transistor. The advent of this transistor made it possible to produce large-scale computers with static logic for less cost than computers with dynamic logic which depended upon vacuum tubes with exceptional transconductance characteristics.

The UDOFT computer was designed during this period of digital computer evolution. Therefore, it might be expected that the type of logic used in the machine would be dynamic. This is further supported by the fact that the design of the computer was performed at the Moore School of Electrical Engineering of the University of Pennsylvania, at that time a practitioner of dynamic logic.

In a computer using static logic, the basic communicator of information or data is a voltage level. Usually a voltage level approximating ground potential indicates a logical zero; a voltage level which is considerably more positive or more negative than that zero state level indicates a logical one. Such signals are obtained initially from static flip-flops. The outputs of the flip-flops may then be combined logically in passive networks commonly referred to as AND and OR gates. The combining process does nothing more than exercise a form of arbitrary control over the output of the network as a function of the inputs and the structure of the network. It is through the combining process that the capabilities of the simple bi-stable elements are exploited and combined into a powerful computing mechanism.

The presence of pulse in a computer using static flip-flops and level logic is necessary to establish the time sequence by which the micro-operations that comprise the machine execution of an instruction are performed. The pulses, unlike the voltage levels, do not convey any information; their function is supervisory in nature. In the dynamic logic scheme, these same pulses are required. In addition, however, pulses rather than voltage levels are generated by the bi-stable elements. In the logical combining process pulses rather than voltage levels are combined.

An unfortunate characteristic of any physical element which is used to transmit electrical energy is that it is electrically imperfect. To cite a few examples of electrical imperfection, wires do not have zero resistance, resistors do not have zero reactance, and vacuum tubes do not exhibit perfect transfer characteristics. These deviations from the theoretical result in one characteristic common to all electrical elements, namely, time delay. Time delay is nearly always the factor that limits the speed of a digital computer. In a computer using dynamic logic, time delays are of paramount importance. This is most apparent when considering the logical combining of information pulse signals. A pulse will appear at the output of an AND gate if, and only if, all inputs are pulsed simultaneously. Some degree of intolerance is allowed on the simultaneity of the incident pulses; however, it is usually quite small.

The clock pulse or master timing system of a computer using dynamic logic performs the function of synchronizing the information pulse signals to a common time reference. Without such a slaving capability the computer would be totally dependent upon the variations in the electrical characteristics of the myriad elements that comprise the computing system, resulting, undoubtedly in extremely sporadic and unpredictable operation. In addition, a secondary source of timing pulses is required, as in the static logic machine, for controlling the starting and the stopping of the computer micro-operations. The controlling pulses are referred to as timing pulses; the synchronizing pulses, as clock pulses.

a.) Clock Pulse System

The UDOFT computer, like all dynamic machines, uses a multi-phase clock pulse system. This means that a number of timing pulse chains is derived from a single source, each pulse chain having the same frequency as the source, but being displaced in phase from it by some fraction of the pulse repetition period. In the UDOFT computer there are five phases of the 1.2 megacycle clock. The separation between successive clock pulse chains is one-fifth of the period or 167 nanoseconds (figure 11). The width of the clock pulses is approximately one-half of the period or 417 nanoseconds. The five phases are identified as 00 (phase zero), 02, 04, 06, and 08.

The original design of the computer had specified at 1.0 megacycle clock source. For this frequency, the clock pulse period is 1.0 microsecond. The phase designations were selected to indicate the number of tenths of microseconds each phase lagged the zero phase; e.g. 04 lagged 00 by 0.4 microseconds and 08 lagged 00 by 0.8 microseconds. However, when the clock rate was increased to 1.2 megacycles, the phase designations were not altered. Thus, phase 04 now lags 00 by 0.333 microseconds and 08 lags 00 by 0.667 microseconds.

A logical question that arises at this point is, why the multiphased clock? Why not a single clock pulse chain? The answer is not simple, but, basically, the use of five phases of the 1.2 megacycle clock phase produces an effective 6.0 megacycle clock. It allows the transmission or transfer of information between elements within the computer at a 6.0 megacycle rate without requiring that any single element operate at a frequency exceeding 1.2 megacycles. Had a single 6.0 megacycle clock pulse source been used in the computer, the clock pulses would have been only 83 nanoseconds in duration. The delay characteristics of the elements that were available at the time the computer was designed would not have allowed their use in such a critical application.

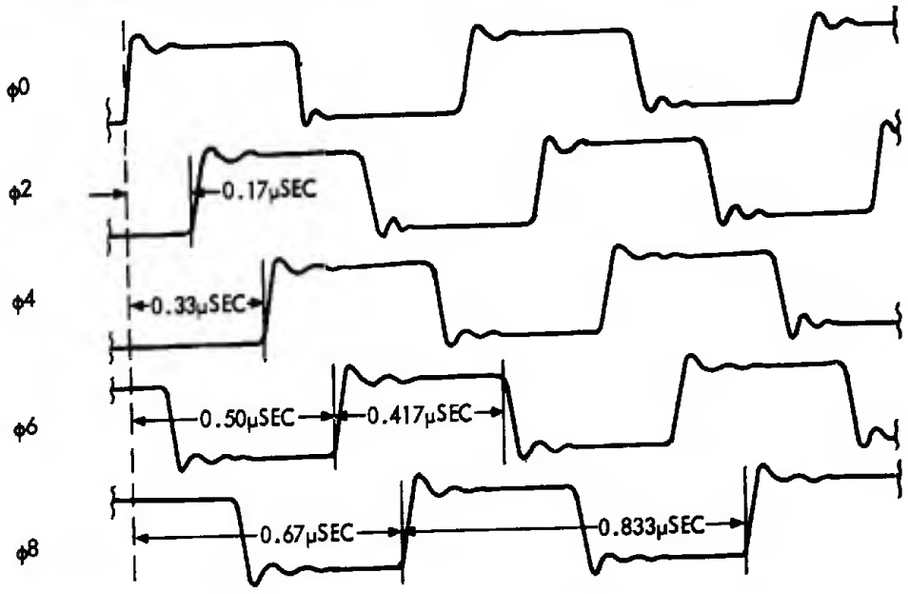


Figure 11. Five Phase Clock Pulse Characteristics

In the computer, the basic 1.2 megacycle source is a crystal controlled Colpitts oscillator which, by virtue of electron coupling, is sufficiently stable for this application. Tuning the plate tank circuit varies the amplitude of the output over a limited range without adversely affecting the frequency. In the original clock system, the 1.2 megacycle sine wave was distributed to each of the five units comprising the computer where it is amplified to a signal of approximately 100 volts peak-to-peak. The amplified signal was passed through a variable delay line inserted at this point in each cabinet. Even though care had been taken to equalize the lengths of the transmission cables between the source and the computer units, the variable delay line was still necessary to adjust for unequal delays. The 1.2 megacycle signal was then split into the various phases by a number of fixed delay lines each of which contributing 167 nanoseconds more delay than the previous one. The five phases of the clock signal, still in the form of a sine wave, were then amplified, symmetrically clipped, and clamped. The signals were then in the form of rectangular pulses. Power amplifiers were inserted at this point to allow heavy loading by the pulse amplifiers. A partial block diagram of the original clock pulse generation and distribution system is shown in figure 12.

It is vital, as has been stated previously, that the absolute and the relative timing of the clock pulse chains be exact. As the components of the clock pulse system aged, the characteristics of the clock pulses changed. The absolute timing, since it is dependent only upon the oscillator, changed very little. The relative timing, however, varied

considerably and frequently. For this reason adjustment potentiometers were inserted at the outputs of the fixed delay lines.

Under normal operating conditions (approximately 80 hours a week), the clock system required realignment every two or three months. On numerous occasions the clock pulse chains drifted sufficiently, between scheduled realignment periods, to cause operating difficulties. The operating difficulties materialize as sporadic and random failures of the computer, making it extremely difficult to determine the source of the trouble. Even when it has been determined that clock pulse timing was the cause, many hours were required to realign the clock pulse chains in each of the five cabinets. Since the clock pulse system was so critical an element and represented one of the more troublesome areas in the computer, it became necessary to redesign the system for both the generation and the distribution of the clock pulses.

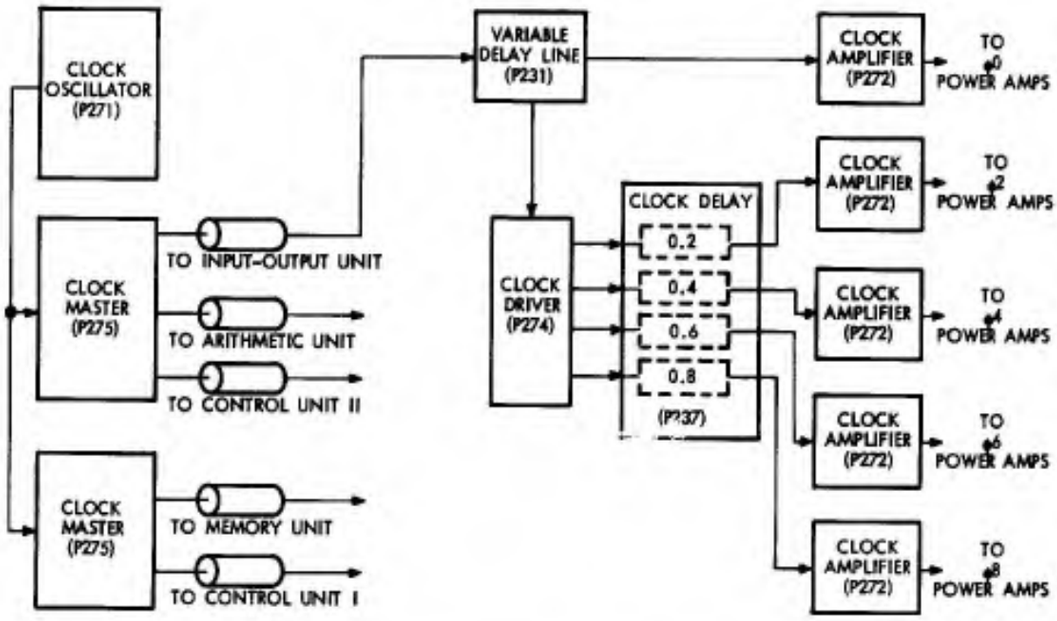


Figure 12. Block Diagram of Original Clock Pulse Generation and Distribution

The modified clock system improves the method of generating the five clock phases by elimating adjustment of phasing, minimizing adjusting of clock pulse widths and eliminating the necessity of phasing the cabinets.

The widths of the clock pulses are made independent of phase separation by using single shot multivibrators (SSMV) and well isolated fixed delay lines. As far as distortion and linearity of delay is concerned the most useful portion of the outputs of the delays is selected by proper biasing of an inverter. The leading edge of the inverted output gates a SSMV adjustable from 0.300 to 0.500 μsec. Stability and independence are achieved by using Zener diodes on each SSMV and the Inverter to derive the voltages required.

To elimate the need to phase the cabinets, the clock is generated in the Input-Output cabinet and each phase is distributed to the five cabinets on separate equal length coaxial cables. (See figure 13.)

The oscillator used is the original P271 package. Its output is clipped and clamped and applied as the clock input of a standard pulse amplifier. The pulse amplifier in conjunction with a standard signal driver is used to drive delay lines. The delay lines establish fixed time differences between phases. The signal drivers serve to isolate the phases and minimize their interdependence. The delay lines are the standard positive short delay lines. The outputs of the delay lines and the output of the fifth signal driver, the undelayed pulse, are then inverted. The inverter is biased to select the section of

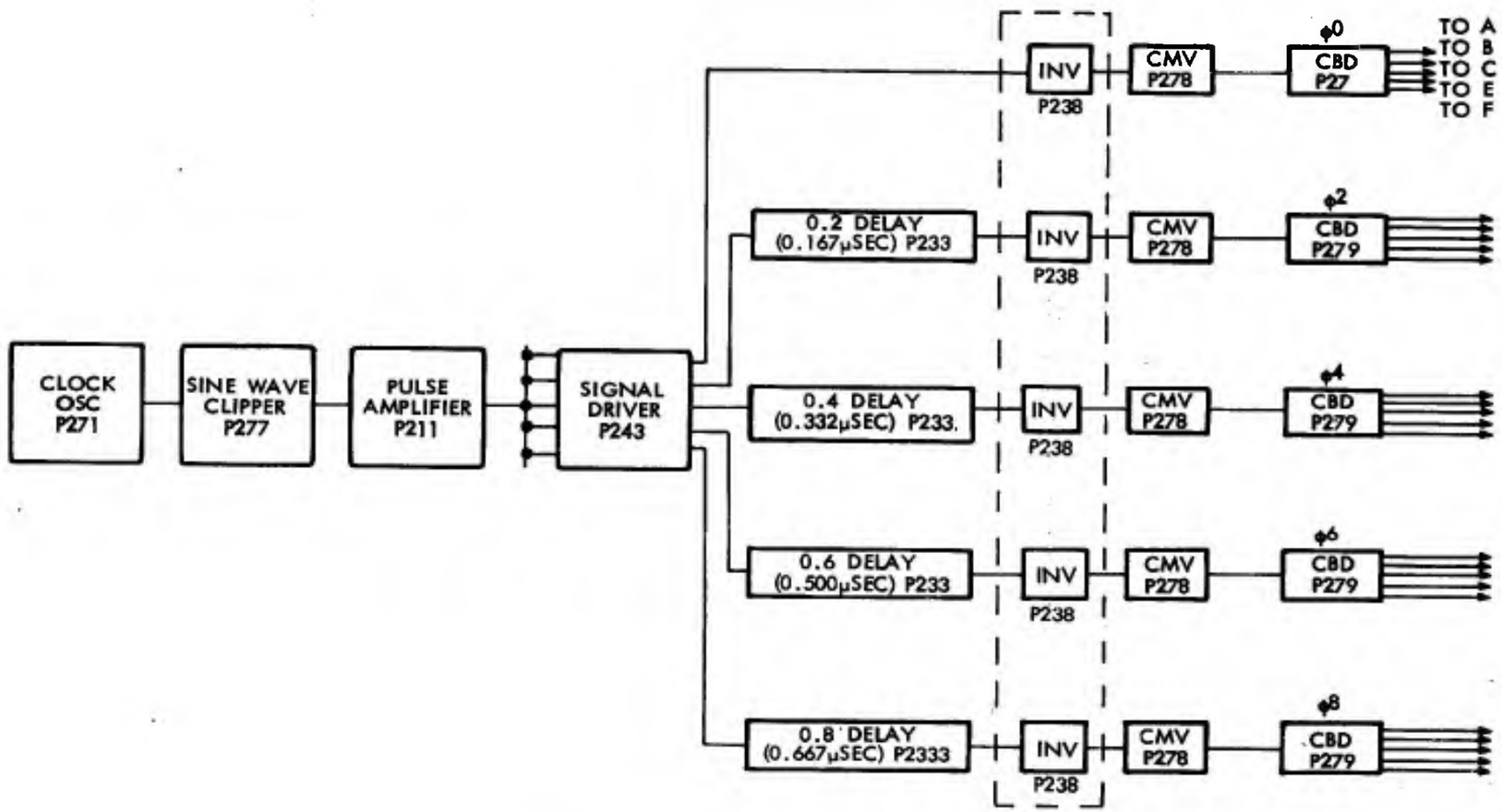


Figure 13. Modified UDOFT Clock System

the output of the delay lines which will minimize problems caused by delay line distortion. The inverted delayed pulses, which are reshaped by the inverter, gate the clock multivibrator (CMV). The input to the CMV is a ringing circuit which generates a pulse when gated by the leading edge of the inverter output. The CMV is a single shot multivibrator, adjustable from 0.300 to 0.500 μsec. The clock multivibrators drive the cable drivers which can drive up to seven coaxial cables.

The outputs of the coaxial cables drive the inputs of the clock drivers. From this point on the distribution is unchanged except for terminations needed to eliminate ringing caused by the improved rise and fall times of the clock pulses.

b.) Timing Pulse System

The computer Timer, as it is called, generates the sequences of timing pulses that indicate distinct times within a minor cycle of computer operation. A minor cycle in the UDOFT computer is 5.0 microseconds in duration, during which time thirty distinctly timed pulses are generated.

The mechanism for generating the thirty pulses is a cascaded arrangement of thirty pulse amplifiers, very much resembling a multi-tapped delay line (figure 14). Pulse amplifiers, rather than a multi-tapped delay line are used in order to insure accuracy of timing; each pulse amplifier is synchronized by means of clock pulse. The first output from the Timer is timing pulse 6.0. A pulse will appear at this output every 5.0 microseconds at a time designated as 6.0. The outputs of subsequent pulse amplifiers have designations which are 0.2 greater than the preceding output. The two-tenths figure designates an absolute separation between successive timing pulse chains identical to that used in identifying the five phases of the clock. The sequence of outputs starting with the first, is 6.0 to 6.8, then 1.0 to 5.8. Since the pulse amplifiers are arranged in a closed loop a single pulse, once injected into the loop will continue to traverse the loop until stopped manually.

3.4.2 Arithmetic Unit

The Arithmetic Unit contains the Accumulator and two registers, the Multiplicand-Divisor Register (M-D Register) and the G-Register.

a.) Accumulator

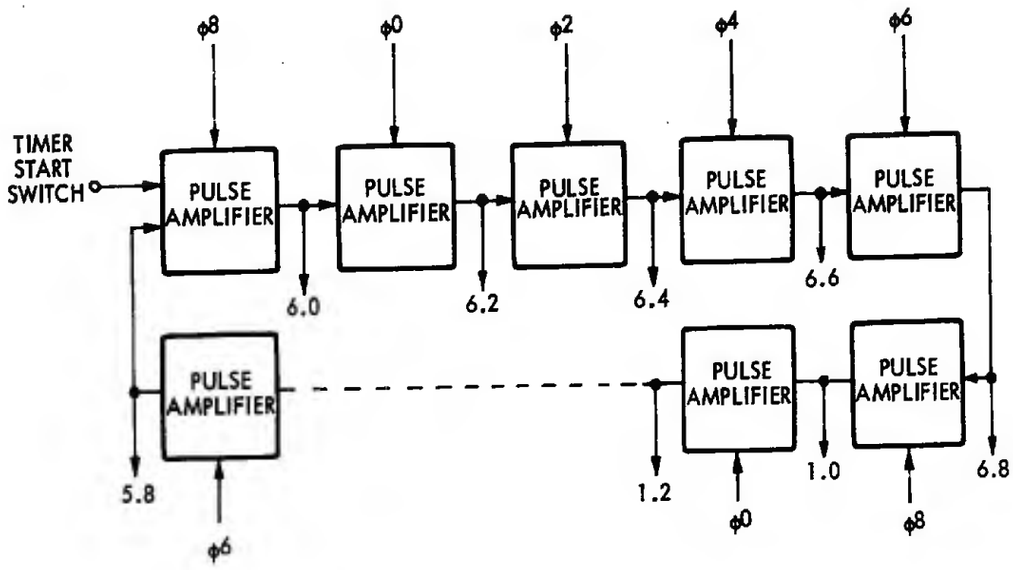
The Accumulator contains the logic necessary for performing the "add," "subtract," "shift," and "complement" operations, one or more of which are required to execute all of the arithmetic instruction. More specifically, the Accumulator consists of twenty identical stages, a partial twenty-first stage, and logic to detect carries out of the twentieth stage.

Each of the twenty stages of the accumulator functions identically in sensing the state of the corresponding bit positions of the M-D Register, the G-Register, and the Accumulator itself. The Accumulator appears to be a purely parallel unit, due to there being an individual adder, adder-subtractor, and recirculation loop for each of the twenty stages. However, the operation of the Accumulator is not purely parallel, but is better described as parallel-sequential.

Although only one bit is operated upon by each stage, and although each bit appears on an independent output line, successive stages of the Accumulator are time-separated by one clock phase (0.167 μsec). This separation in time is to allow the formation and the propagation of carries as the arithmetic operation sequentially from the least significant bits to the most significant bits of the operands.

A single Accumulator stage (figure 15) is composed of an adder, an adder-subtractor, a shift amplifier, and a recirculation amplifier. Several dispatcher lines introduce the control pulses which determine the operations to be performed by the Accumulator stage.

The heart of the Accumulator stage is the adder-subtractor. It is here where quantities from memory or the G-Register are added to or subtracted from the quantity stored in the Accumulator. The complement feature of this element is required for those instances where a true subtraction of the form (A-B), where /B/>/A/, is performed. The



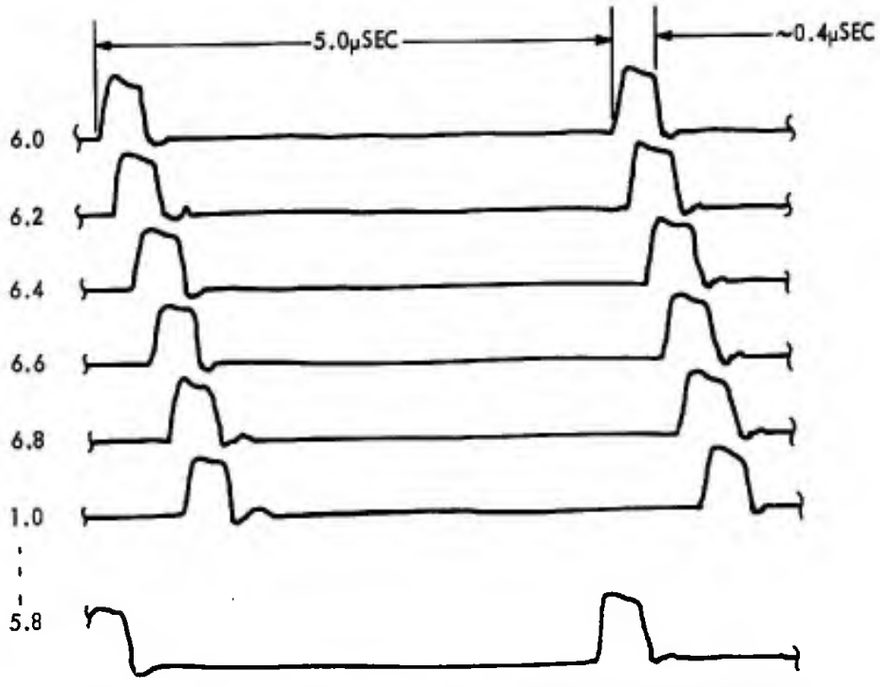


Figure 14. Block Diagram of Timing Pulse Generator Loop

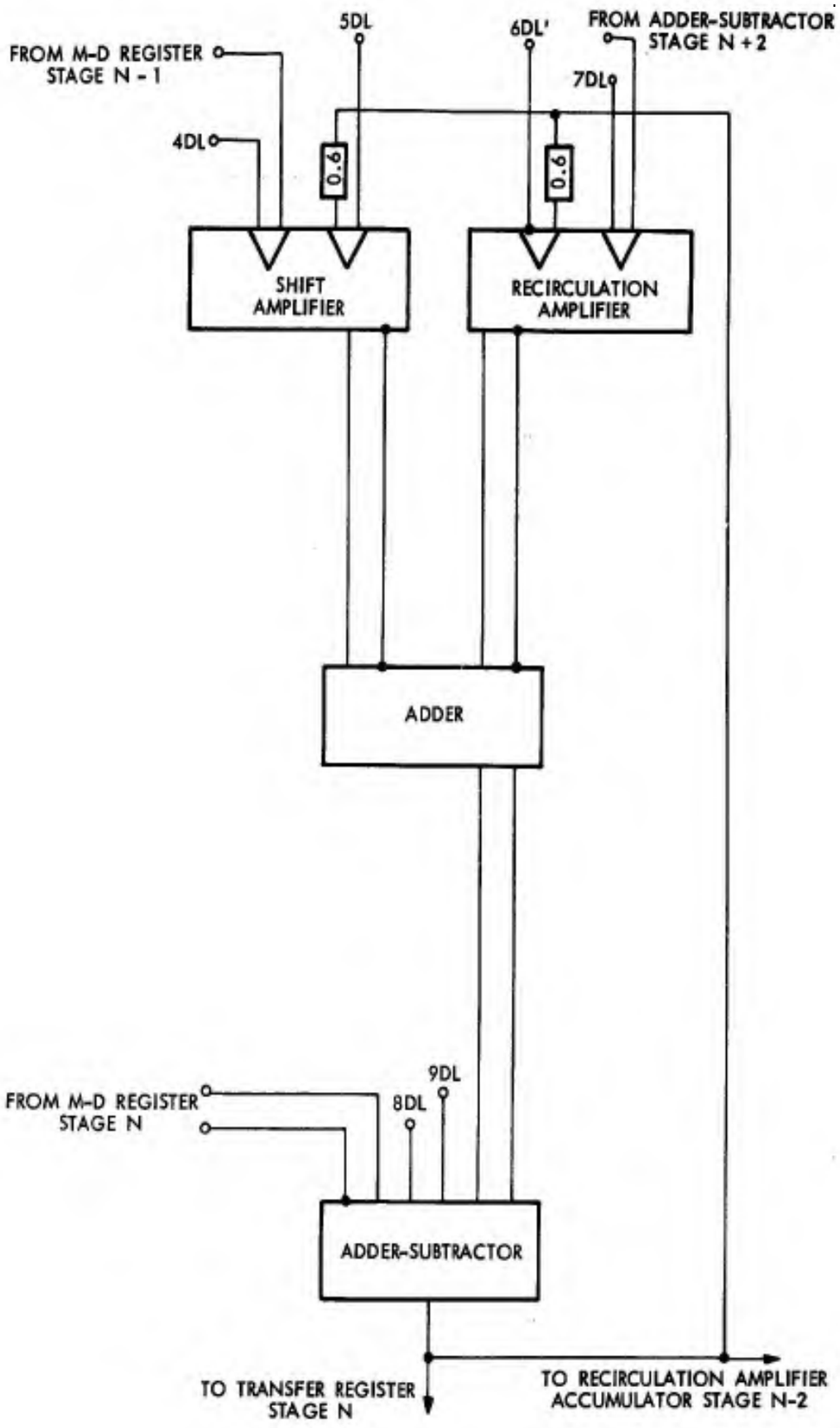


Figure 15. Block Diagram of Single Accumulator, Stage N

initial difference as determined by the Accumulator is in two's complement form, thereby necessitating a complement and a sign reversal operation.

The recirculation amplifier provides the means by which the retention of data in the Accumulator is controlled. As long as the amplifier is enabled, the single bit of data in each Accumulator stage is allowed to circulate around the major loop consisting of the adder-subtractor, the adder, and the recirculation amplifier. Inhibiting or disabling the recirculation amplifier prevents the recirculation, thereby effectively clearing the Accumulator stage.

The adder provides a second level of addition to the Accumulator stage. Shifting a number to the left in the Accumulator is performed by progressively adding the number to itself; this operation is effected by the adder. This operation could have been performed in the adder-subtractor; however, it would have resulted in increased complexity of the adder-subtractor logic. The primary reason for the inclusion of the second level adder, however, was to accelerate the execution of the multiply instruction.

Multiplication of two binary numbers in a simple digital computer consists of addition and shifting operations. Multiplication is accomplished by adding or not adding the multiplicant to the content of the accumulator depending upon the condition of the multiplier bit. An addition would occur if the multiplier bit were a 1, no addition, if the bit were a 0. Subsequent to this operation the content of the Accumulator is shifted one place to the right and the next multiplier bit is examined. This operation continues until all bits of the multiplier have been examined. The resulting sum in the accumulator is the product of the multiplication operation. In UDOFT, multiplication speed is effectively doubled by examining two multiplier bits at a time and shifting the partial sums two places to the right. In order to do this two adders are required; thus, the reason for the secondary adder.

The shift amplifier allows the quantity in the accumulator to be added to itself in the adder in order to effect a shift left of one place. Further, the shift amplifier provides entry to the adder for the multiplicand during the multiply instruction.

b.) Multiplicand-Divisor Register

The Multiplicand-Divisor Register is a 22-bit (20 data bits, sign, and parity) dynamic flip flop buffer – storage register, which acts as:

1. A buffer between the Number Memory Output Register and the Accumulator. Upon being transferred into the M-D Register from memory, the data words are checked for parity.

2. A buffer between the G-Register and the Accumulator

3. A storage register for the multiplicand during the multiplication operation.

4. A storage register for the divisor during the Division operation.

c.) G-Register

The G-Register is a special-purpose storage register with a capacity of twenty binary bits plus the sign bit. It differs from the M-D Register in that it does not consist of distinct storage flip-flops for each of the data bits; rather, it consists of five serial storage loops, similar to delay line storage loops (figure 16). Serial storage loops are used primarily to save hardware. By using one pulse amplifier and a long delay line, it is possible to store several bits serially. The opportunity to employ this technique does not occur often in UDOFT due to timing restrictions. Bits from stages of the accumulator having the same clock timing are serialized and injected into the appropriate storage loop. When access is made to the G-Register, the bits are extracted from the loops in serial-parallel fashion. In conjunction with the Multiply and Add G-Register (MAD) and the Shift and Add G-Register (SHLA, SHRA) instructions, the G-Register performs in the manner just indicated.

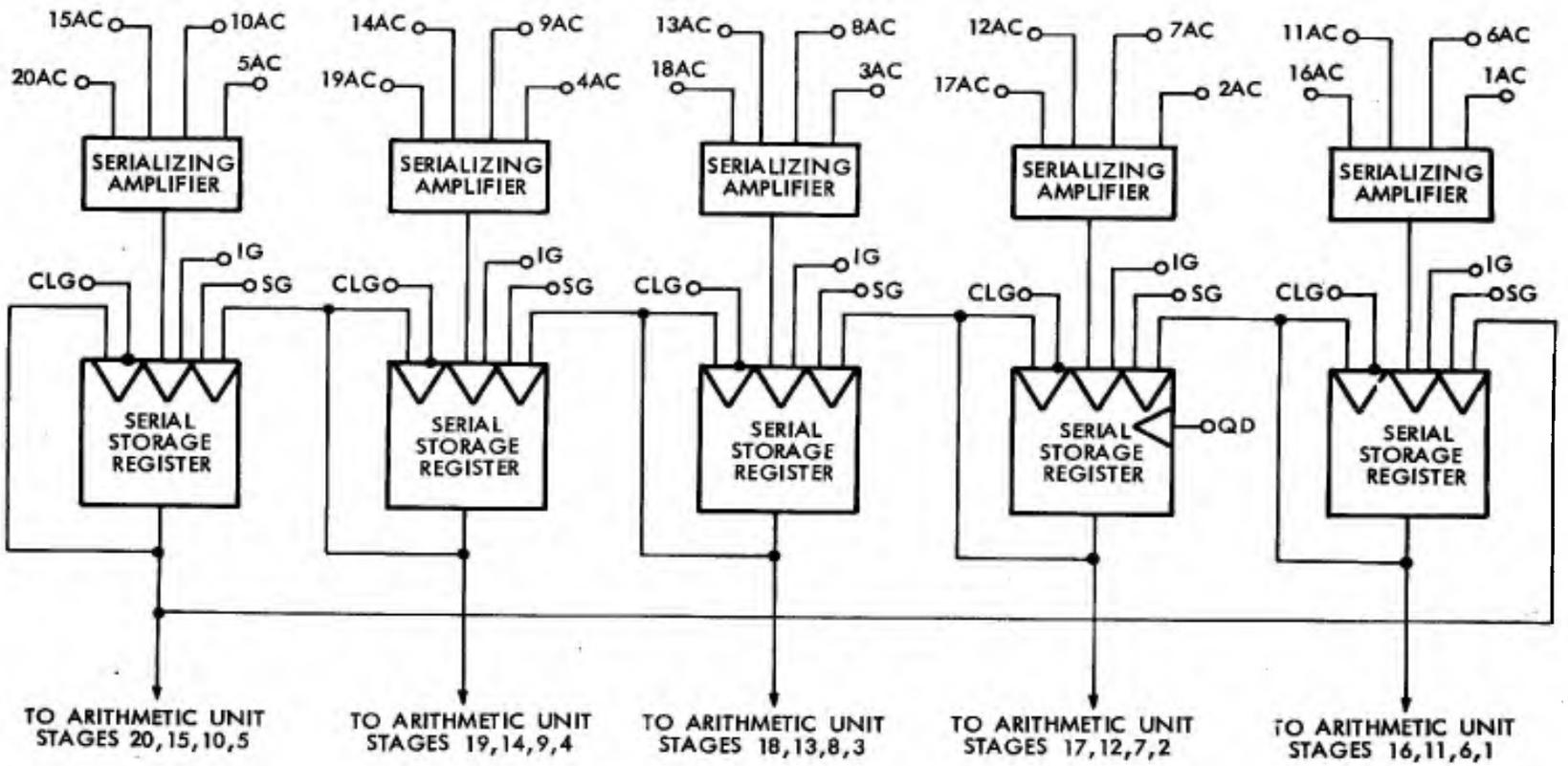


Figure 16. Block Diagram of G-Register

During the divide operation, the G-Register stores the individual quotient bits as they are formed from the operations occurring in the accumulator. Division requires twenty-one complete addition or subtraction operations in the accumulator. Since each complete arithmetic operation requires five microseconds, the Divide instruction requires 105 microseconds for its complete execution. During this period twenty quotient bits are generated. In order for the G-Register to store the quotient bits during the execution of the instruction, the capacity of the register must be increased. This is affected by opening the five serial loops and tieing them together to form one large loop which has a circulation time slightly in excess of twenty-five microseconds. A quotient bit is injected into the G-Register every five microseconds; at the end of the process another five microseconds is required to transfer the twenty quotient bits into the accumulator.

Because of timing restrictions the twentieth bit is not injected into the G-Register. Instead, it is injected in the low order stage of the accumulator directly.

3.4.3 Control Unit I

The control element of the UDOFT computer system performs three basic functions: sequence determining, arithmetic, and input-output control. The control element specifies a given instruction, identifies the operation to be performed, and identifies the operand to be used. Upon identifying the operation and the operand, the control element then provides the Arithmetic Unit with sequenced control pulses which effect the execution of the instruction.

The control element of the UDOFT computer system is subdivided, due to physical limitations, into two units, Control Unit I and Control Unit II. Control Unit I is described as follows and Control Unit II is described in 3.4.4.

Control Unit I is concerned primarily with the arithmetic control function. It is here that the five instruction word bits that specify the order type are decoded and the order type is identified. The identification of the order type initiates the generation and the propagation of the pulses that control the Arithmetic Unit. The groupings of logic elements that perform this function are defined as Dispatcher Lines.

In addition to satisfying the arithmetic control function, Control Unit I contains two registers; the Transfer Register and the Interval Timer. The Transfer Register is rightfully an integral part of the Arithmetic Unit; however, it is contained within Control Unit I simply because there was insufficient space in the Arithmetic Unit for its inclusion.

a.) Transfer Register

The Transfer Register is a 22-bit (22 data bits, sign and parity) non-addressable, dynamic flip-flop, buffer-storage register which acts as a buffer between the output of the Accumulator and various terminal units in the computer system. The particular terminal units are the Number and the Instruction Memory rewrite registers, the Multiplexer Output Register, the Tally Register, and the Print Register. During the card read-in process it acts as a buffer between the card reader and the two memories.

At the beginning of each instruction, except STO and MLXO, the Transfer Register is cleared and the contents of the Accumulator are read into the Transfer Register. Thus, the Transfer Register lags the Accumulator by one instruction. This feature imposes a programming restriction, namely, that a STO instruction will not store the results of the preceding or $(J-1)^{st}$ instruction. Rather, it will store the results of the $(J-2)^{nd}$ instruction. This feature usually causes no inconvenience to the programmer and if the programmer is careful there is no time lost due to the use of extra instructions. The problem can also be overcome by inserting a NOP instruction prior to the STO instruction, thereby allowing the proper quantity to be in the Transfer Register at the time the STO instruction is executed. However, this is not an economical method of programming.

b.) Interval Timer

The Interval Timer is provided specifically as a means for equalizing the time interval between successive iterations of the flight simulation program. More generally it is a program-controlled real-time clock.

In detail, the timer is a modified fourteen stage subtracting binary counter. It is settable, in increments of five microseconds, to a maximum value of 81.915 milliseconds by means of the SIT instruction. The count in the timer is decremented each instruction minor cycle (five microseconds) until the count is zero, after which time no change is made to the count unless a SIT instruction is performed.

As originally conceived, the SENIT instruction was capable only of sensing a zero condition in the Interval Timer. Thus, if fifteen milliseconds remained in the timer at the time the SENIT instruction was decoded, no other instruction could be performed until the timer ran down to zero. This feature seemed to be wasteful of valuable time which could be used to perform auxiliary programs. For this reason the SENIT instruction was modified to be a conditional transfer of control type instruction. If the timer is identically zero at the time the SENIT instruction is executed, the next instruction in sequence will be executed; if the timer is not identically zero, there is a transfer of control to the instruction specified by the address field of the SENIT instruction. Concurrently, the count in the timer is transferred to the Accumulator where it can be examined to determine if there is sufficient time remaining to perform the auxiliary programs. It is also useful for determining timing of routines or program segments. The original facility of the SENIT Instruction is retained simply by assigning the address of the SENIT Instruction to the address field. This forms a one instruction loop which is exited when the interval timer reaches zero.

c.) Dispatcher Lines

The ten Dispatcher Lines, identified as ODL through 9DL, generate the pulses that order the arithmetic unit of the computer to perform the arithmetic operations. A brief functional description of each Dispatcher Line follows.

1. 0DL

   Dispatcher line 0 serves two purposes. In those arithmetic operations which use the contents of the G-Register as the operand, or one of the operands, it gates the contents of the G-Register into the Multiplicand-Divisor Register. In all other instructions which check number memory parity, it isused in conjunction with 1DL to provide gating pulses for the parity checking circuitry.

2. 1DL

   Dispatcher Line 1 causes the contents of the Number Memory Output Register to be gated into the Multiplicand-Divisor Register. 1DL is pulsed therefore for all arithmetic operations involving an operand, the address of which is not that of the G-Register.

3. 2DL

   Dispatcher Line 2 causes the Multiplicand-Divisor Register to be cleared to zero. 2DL is pulsed therefore for all operations which read or transfer data into the Multiplicand-Divisor Register.

4. 3DL

   Dispatcher Line 3 causes the contents of the Multiplicand-Divisor Register to be gated into the adder-subtractors of the Accumulator. 3DL is pulsed therefore for all operations which cause the content of the Accumulator to be simply augmented or decremented. In addition to this most basic function, 3DL is pulsed sequentially as a function of the odd multiplier bits during the Multiply instruction, and is pulsed twenty-one times during the Divide instruction in order to introduce the divisor and the quotient into the Accumulator.

5. 4DL

Dispatcher Line 4 causes the contents of the Multiplicand-Divisor Register, shifted left one place, to be gated into the upper adder of each of the Accumulator stages. It functions only during the Multiply instruction, at which time it is pulsed sequentially as a function of the even multiplier bits. Dispatcher Lines 3 and 4, primarily, cause the Accumulator to perform the multiplication function; Dispatcher Line 7 contributes also to the multiplication function.

6. 5DL

Dispatcher Line 5 causes the contents of the Accumulator to be shifted to the left. Each time 5DL is pulsed, a single shift to the left of one pulse is effected. Dispatcher Line 5 functions during the right shifts as well as left shifts because single-place shifts to the right or odd numbered shifts to the right are performed by first shifting left one place and then shifting right two places at a time. It functions also during the Divide instruction in which it shifts the remainder left one place after each addition or subtraction operation.

7. 6DL

Dispatcher Line 6 causes the contents of the Accumulator to be cleared to zero. 6DL is pulsed for all instructions, such as CLA and CLAA, which explicitly require a cleared Accumulator and for other instructions, such as SHR, and DIV, which implicitly require a cleared Accumulator.

8. 7DL

Dispatcher Line 7 causes the contents of the Accumulator to be shifted two places to the right. 7DL is pulsed only during the Shift Right (SHR, SHRA) instructions and during the Multiply instruction.

9. 8DL

Dispatcher Line 8 causes the two's complement of the contents of the Accumulator to be performed. 8DL is pulsed therefore as the result of a true subtraction of the form (A-B) where $|B|>|A|$. It is pulsed also in order to establish the maximum quantity in the Accumulator after an overflow has occurred and truncation of the result is indicated. (8DL is not used during divide, since the subtract performed as a part of the divide, is not a true subtract.)

10. 9DL

Dispatcher Line 9, causes the adder-subtractors of the Accumulator to function as subtractors. 9DL is pulsed therefore whenever a true subtraction is performed. A true subtraction will occur during an Add instruction whenever the signs of the operands are different, during a Subtract Instruction whenever the signs of the operands are similar, and during the divide instruction in accordance with the division algorithm.

d.) Order Type Selectors

Control Unit I also contains the Order Type Selectors. The Order Type selectors decode the five bits of the order type of the instruction word which is read out of the Instruction Memory. These five bits uniquely define the thirty-two different instructions

which the computer is capable of executing. The results of the decoding operation is the generation of a series of pulses on one of thirty-two control lines, each line specifying one of the orders. In addition to specifying the macro-operation (Add, Subtract, Divide, etc.) that is to be executed, the control pulses, through logical combining with other conditioning signals, specify and initiate the micro-operations that constitute an instruction cycle.

e.) Sign Logic

The last major control function implemented in Control Unit I is the determination and the storage of the sign of the quantity in the Accumulator. Determination of the Accumulator sign is dependent upon the particular arithmetic operation and the signs of the operands. Therefore the signs of the quantities in the Multiplicand-Divisor Register and the G-Register are stored in Control Unit I as well as the sign of the quantity currently in the Accumulator.

3.4.4 Control Unit II

Control Unit II performs the major functions of instruction sequencing, memory control, and, to some extent, input-output control. Instruction sequencing relates specifically to the Sequence Counter and the start and halt controls; memory control relates specifically to the Number Address Storage Register, the Relative Address Register, the Tally Register, and the Sequence Counter; input-output control relates specifically to the Number Address Storage Register. It is apparent from the preceding statement that some of the registers, such as the Sequence Counter and the Number Address Storage Register, serve a multiplicity of functions. This characteristic will be exposed further in the following brief functional descriptions of each register.

a.) Sequence Counter

The Sequence Counter is properly a twelve stage, parallel, storage register. Its primary function is to specify to the Instruction Memory the address of the next instruction which is to be executed. In order to be able to specify the next instruction, the Sequence Counter must be capable of being augmented by a single count or being set to a specific count. Normally, after each instruction, the counter is simply augmented; however, after the execution of a transfer of control type instruction, the counter may be set to a specific count.

It is interesting to note that the twelve dynamic flip-flops that comprise the Sequence Counter are simply storage flip-flops; they are not interconnected to form a true binary counter. The augmenting of the count in the storage flip-flops is performed by a serial adder. The twelve bits of the register are serialized and applied, least significant bit first, to a serial adder in which one is added to the least significant bit. The augmented count then is read back into the storage register.

b.) Number Address Storage Register

The Number Address Storage Register is a twelve stage, parallel storage register. Its functions are as follows:

1. In shifting instructions it specifies the direction and number of shifts to be performed.
2. In control transfer instructions, it specifies the count to which the sequence counter may be set.
3. In the TIM instruction, it specifies an instruction memory address.
4. In the MLXI instruction, it specifies an analog input channel.
5. In conjunction with the discrete input, it specifies a discrete input.
6. In the output instructions, Multiplex Analog Output (MLXO) and and Multiplex Discrete Output (MXDO), it specifies the particular output channel to be processed.

The Number Address Storage Register actually does not address the number memory, it indicates which number memory register is being addressed.

In addition to the multiplicity of uses of the outputs of the Number Address Storage Register, there are multiple inputs to the Register. Normally the input to the register is derived from the instruction word address field as formed by the Number Address Serializing Register. However, under the condition of using relative address, the input to the Number Address Storage Register is derived from the Relative Address Register.

c.) Relative Address Register

The Relative Address Register is a twelve-bit storage register, which stores the relative address that may be used by the succeeding instruction. In reality, the register is two, six-stage, serial storage registers which operate in parallel in order to facilitate rapid access to the relative address data. The output of the Relative Address Register is applied to the Number Memory Address Register and the Number Address Storage Register; the input is always derived from the Relative Address Former.

The Relative Address Former consists of two, two-operand, serial adders with carry propagation capabilities. It is here that the serialized contents of the Tally Register are added to the serialized address field bits of the instruction words. The address field bits are serialized in the Number Address Serializing Register.

The Number Address Serializing Register consists of two six-bit serial registers. The address field of the instruction word is applied in parallel form to the input of the total register; the six even bits of the address field being applied to one six-bit register and the six odd bits being applied to the other register. The Number Address Storage Register thus converts the address field from a twelve-bit parallel word into two six-bit serial words.

The output of the Number Address Serializing Register is applied to both the Relative Address Former and the Number Address Storage Register. For those instructions for which relative address is not specified, the output of the register is gated into the Number Address Storage Register.

d.) Tally Register

The Tally Register of the UDOFT computer is comparable to an index register in the common computer. It stores the quantity by which the instruction-specified operand address is modified if relative address is specified. In keeping with the serial-parallel philosophy of the address control function, the Tally Register consists of two six-bit serial storage loops; one loop stores the six odd bits and the other, the six even bits.

The output of the Tally Register is applied only to the Relative Address Former. The inputs to the Tally Register are derived from the following sources.

1. The Transfer Register. This register provides the input during the execution of the Tally Arithmetic Unit (TAU) instruction.

2. The Relative Address Register. This register provides the input during the execution of the Relative Tally Number Address (RTAN) instructions.

3. The Number Address Serializing Register. This register provides the input during the execution of the conventional Tally Number Address (TAN) instruction.

4. Console Switches.

3.5 Memory Unit

The UDOFT computer contains two coincident-current, magnetic-core memories; the Instruction Memory for instruction words and the Number Memory for data words. The two memories are identical in design and construction; but differ slightly in operation. The similarity between the two memories is readily apparent. Each memory consists of an array of twenty-two magnetic core memory planes each of which contains 4096 ferrite cores, an address register, an output register, and a rewrite register (figure 17).

A cycle of memory operation is initiated by the transfer of the twelve bits of address information into the Memory Address Register. The twelve bits of the address are divided into four groups of three bits each. This is done in order to allow the matrix transformation of each three binary bit group into eight (2^3) distinct control functions. The sixteen distinct control signals that are derived from the two low order three-bit groups of the address are transformed into 64 (8^2) distinct control signals. These 64 control signals are defined as the selection signals for one coordinate of the memory plane array. Similarly the sixteen distinct signals, derived from the two high order three-bit groups of the address, are converted into 64 distinct control signals which are defined as the selection signals for the other coordinate of the memory plane array. Since the magnetic core memory plane is, in reality, a 64×64 matrix, pulses appearing on a single selection line of each coordinate will disturb one and only one magnetic core, namely, the core that lies at the intersection of the two lines that have been selected.

The core, when sufficiently disturbed, will exhibit a change in its magnetic field. The change in the magnetic field causes a voltage to be induced onto a third signal line or winding that passes through the magnetic core. This voltage signal is amplified and discriminated to determine if a binary one had been stored in the selected core. If a one had been sensed, a flip-flop will be set. Since there is a sense amplifier and an output register storage flip-flop associated with each bit or plane of the array, the Number Memory requires twenty-two sense amplifiers and a twenty-two stage static flip-flop storage register; for the Instruction Memory this requirement is decreased to twenty of each.

As a result of the selection process, the magnetic core loses the binary data that it was storing. It is necessary, therefore, to provide a means by which the data extracted from the disturbed cores may be returned or rewritten for future use; the rewrite register and the inhibit drivers provide this means. The selection process is reinitiated and the same core in each plane is selected; however, the polarity of the disturbing signals is reversed. The information stored in the rewrite register determines which of the selected cores will be disturbed. If a binary one had been extracted from the particular core, the core is disturbed to the ONE state; if a binary ZERO had been extracted, the core is undisturbed. The process of preventing certain cores from being disturbed, or rewritten, is controlled by the inhibit drivers. For those cores where no rewrite is necessary, a pulse will occur on a fourth line or winding which passes through the cores. The signal on this line, the inhibit winding, will be equal to, but opposite in polarity from one of the two disturbing or select signals, thereby reducing the net effect of the selection signals to the extent that the rewrite process is inhibited.

With the termination of the rewrite process one complete memory cycle is accomplished. Insofar as the related operation of the two memories is concerned, it is sufficient to state that the cyclic operation of each is identical. The only difference is the one-half memory cycle time separation between identical operations in each memory.

One major problem with the UDOFT memory scheme arises from the fact that the two memories are isolated from each other; neither can communicate directly with the other. Instruction words from the Instruction Memory cannot be introduced at all into the Number Memory. Data words from the Number Memory, however, can be introduced into the Instruction Memory, indirectly, by means of the Transfer to Instruction Memory (TIM) instruction. This limitation on intercommunication between the memories is a decided disadvantage. In the particular case of the UDOFT computer, for which each memory consists of 4096 storage registers, the instruction and data storage capability is limited to 4096 words of each type. It is impossible, therefore, to handle a total program in which the quantity of either of these types of words exceeds 4096, even though the total number of words in the program is less than 8192 words.

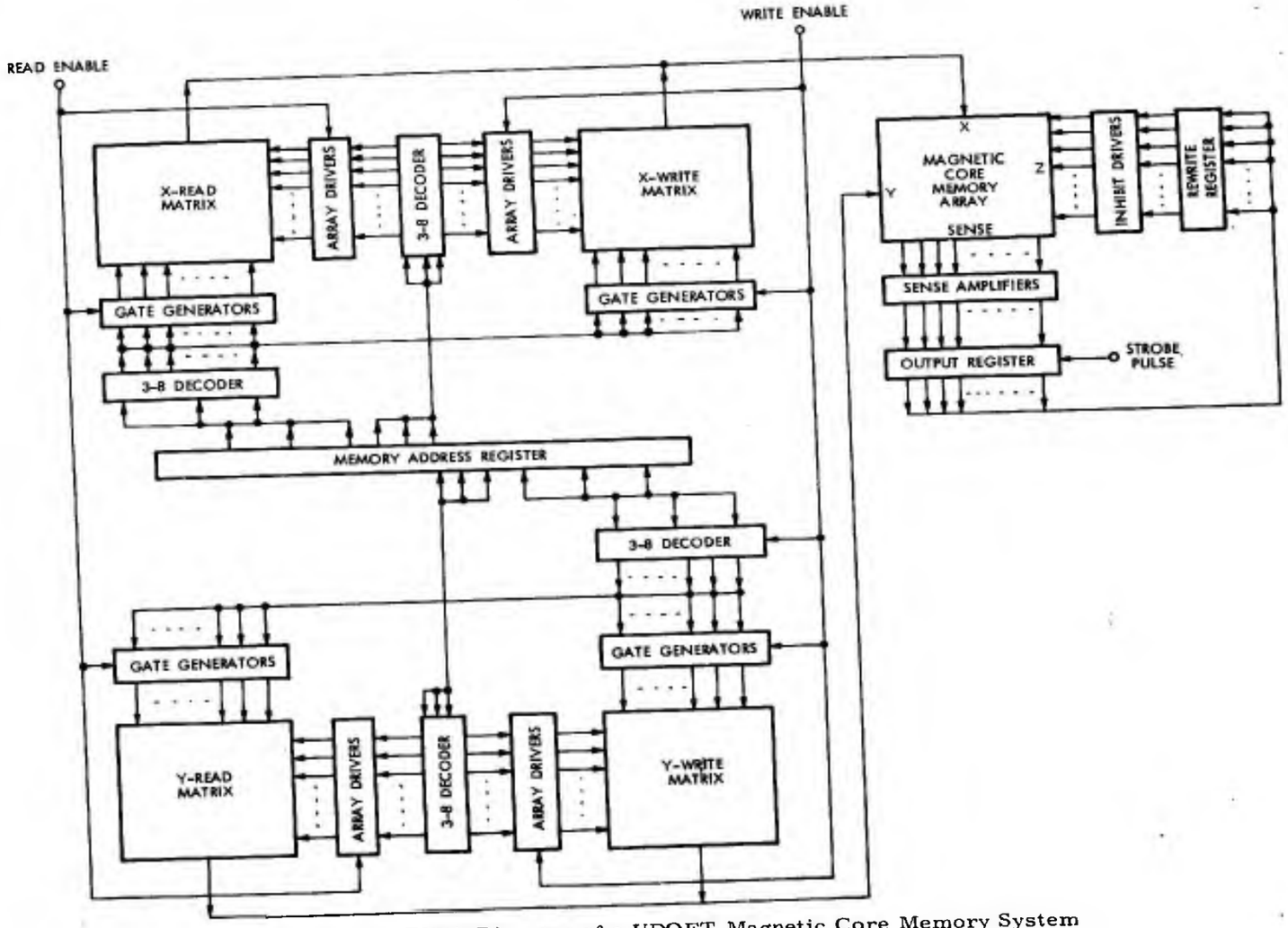


Figure 17. Block Diagram of a UDOFT Magnetic Core Memory System

3.6 Input-Output Unit

The Input-Output Unit is the buffer between the main frame of the computer and the real-world as typified by the aircraft flight compartment mockup. Fundamentally, the Input-Output Unit processes the four forms of real-time communication available to the computer system. Further, when the computer system is used for research or testing purposes, the Input-Output Unit expedites the accumulation and outputting of numerical data that is normally not required during or after the time such a system would be used as a training device.

The four forms of real-time communication were first introduced in Section 2.2 of this report. Any further description of the role of these control signals is deemed unnecessary at this point. However, a brief description of the manner in which these inputs and outputs are processed will be presented.

3.6.1 Discrete Inputs

The computer can accept control information from as many as sixty-four switches that may be situated in the synthetic flight compartment or at the simulator instructor's station. Each of the inputs generated by these sixty-four switches has a particular register in the Number Memory associated with it. The fact that a given discrete-input switch is open causes the "masking" of the contents of the associated number register. Thus, when the computer executes an instruction, the operand address of which specifies a masked register, the normal transfer of information from the addressed Number Memory Register into the Multiplicand-Divisor is suppressed. As a result nothing is read into the M-D Register which, at the initiation of the instruction, had been cleared to zero. The contents of the masked memory register are not altered in any way and can be read out once the associated discrete-input switch is closed.

The information in the controllable registers may be of a quantitative nature or a control nature. An example of quantitative data being stored in the register would be the weight of external stores. If external stores are applicable to the vehicle configuration being simulated, the controlling switch is closed and the weight of external stores is added, during the weight calculation, to the total weight of the vehicle. If external stores are not applicable, the switch is opened, and zero is added to the total weight of the vehicle. This is a very simple example and does not consider any other ramifications resulting from the introduction of external stores.

The more common use of the switch-controlled registers is program control by external means. In this case the registers would contain information of a qualitative nature. A common sequence of instructions that typifies this use would be CLA X, TOM Y. The quantity stored in X which is a discrete input controlled register, would be some negative number; its magnitude is immaterial. If the discrete-input switch is in the open condition at the time the CLA X instruction is executed, all zero's will appear in the Accumulator at the time the succeeding instruction, TOM Y, is executed. Thus, with a positive quantity (zero is always positive) in the Accumulator, no transfer of control will be performed by the TOM instruction and the succeeding instruction will be executed. If the discrete input switch had been closed, a negative quantity would appear in the Accumulator, and there would be a transfer of control to instruction Y. In this application, the discrete input function is analogous to the sense flip-flop function in a general purpose digital computer.

For the F-100A aircraft simulation program, approximately forty of the discrete-input channels are used (Table III). The majority of these discrete inputs perform program control functions.

3.6.2 Discrete Outputs

There are twenty-four discrete output channels implemented in the UDOFT computer. Each output channel consists basically of a dynamic flip-flop which controls a relay.

The discrete outputs are selected and controlled by the Multiplex Discrete Output (MXDO) instruction. The bits of the address field of the instruction determine which output flip-flop is to be selected. Once selected, the state of the flip-flop is determined by the sign of the quantity currently residing in the Accumulator. If the quantity is negative

TABLE III

DISCRETE INPUT ASSIGNMENTS FOR
F-100A SIMULATION PROGRAM

DI	NMAD	USE	DI	NMAD	USE
00	0400	NOZZLE FAIL CLOSED	40	0440	ROUGH AIR
01	0401	NOZZLE FAIL OPEN	41	0441	GUIDE VANE ANTI-ICE
02	0402	ZERO	42	0442	INCREASE ALTITUDE
03	0403	START ENGINE CRANK	43	0443	DECREASE ALTITUDE
04	0404	START ENGINE FIRE	44	0444	AUTO PILOT
05	0405	EMERGENCY FUEL REGULATOR ON	45	0445	FUEL REGULATOR FAIL
06	0406	TEMPERATURE HOT (0.934% THRUST)	46	0446	F9F-2
07	0407	DEFROST	47	0447	ALTITUDE LOCK
10	0410	DROP TANK PRESSURE	50	0450	ROLL ANGLE LOCK
11	0411	DROP TANK JETTISON	51	0451	F9F-2
12	0412	DROP TANK REFUEL	52	0452	F9F-2
13	0413	WINDSHIELD ANTI-ICE	53	0453	F9F-2
14	0414	SPEED BRAKE IN	54	0454	F9F-2
15	0415	SPEED BRAKE OUT	55	0455	F9F-2
16	0416	SPEED BRAKE DUMP	56	0456	F9F-2
17	0417	PILOT ICE	57	0457	F9F-2
20	0420	DRAG CHUTE INFLATED	60	0460	F9F-2
21	0421	CABIN PRESSURE 2.75 PSI	61	0461	F9F-2
22	0422	AFTERBURNER ON	62	0462	F9F-2
23	0423	EMERGENCY HYDRAULIC FLIGHT CONTROL SYSTEM	63	0463	F9F-2
24	0424	HYDRAULIC NUMBER 1 FAIL	64	0464	F9F-2
25	0425	HYDRAULIC NUMBER 2 FAIL	65	0465	F9F-2
26	0426	HYDRAULIC NUMBER 1 TO AO	66	0466	F9F-2
27	0427	HYDRAULIC NUMBER 2 TO AO	67	0467	F9F-2
30	0430	FREEZE	70	0470	F9F-2
31	0431	NOSEWHEEL STEERING	71	0471	F9F-2
32	0432	UTILITY HYDRAULIC FAIL	72	0472	F9F-2
33	0433	CABIN PRESSURE 5.00 PSI	73	0473	NO FUEL DEPLETION
34	0434	YAW DAMPER	74	0474	F9F-2
35	0435	MAIN TANK REFUEL	75	0475	F9F-2
36	0436	MAIN TANK DUMP	76	0476	TRUE AIRSPEED CONSTANT
37	0437	LANDING GEAR IN MOTION	77	0477	FIX CENTER OF GRAVITY

the flip-flop is set to the ONE state; the ONE state implies pulses circulating around the flip-flop storage loop. If already set to the ONE state, the flip-flop will remain set until cleared. If the quantity in the Accumulator is positive, the flip-flop will be cleared to the ZERO state.

Approximately fifteen discrete output channels are required for the F-100A problem (table IV). Had the radio aides functions been implemented in the computer, this quantity would have increased noticeably.

3.6.3 Analog Inputs

There are twenty-four analog input channels implemented in the UDOFT computer. Each input channel consists of a bit-serializing circuit. It is by means of these circuits that the ten-bit parallel output of the Gray-coded binary, shaft-angle converter is converted into serial form, the most significant bit appearing first. The output of each of the twenty-four serializing circuits forms a data input to a selection matrix. The other inputs to the selection matrix are the number memory address bits which control the selection of the appropriate data line. When an analog input channel is addressed, one control line of the matrix is activated and the serialized analog input data appearing on the associated data line is gated into the Gray code-to-binary converter.

The Gray code-to-binary converter would not be necessary had conventional binary coded shaft angle converters been used. However, at the time UDOFT was developed unambiguous read-out shaft-angle converters were very costly. Therefore, it appeared sound to use converters utilizing conventional disc and brush techniques, attaining the unambiguous read-out feature by using Gray coded discs.

As the serialized data is converted to conventional binary form, it is also transferred to the Accumulator. Although most of the communication between registers in the computer is performed in a parallel manner, the transfer of analog input data to the Accumulator is performed in a serial manner. The converted data bits are introduced serially into the eleventh stage of the Accumulator and shifted successively to the left until stages eleven through twenty contain the ten analog data bits. Since ten shifts are required, approximately nine microseconds are consumed just for transferring the data into the Accumulator. This is one of the factors which necessitates that two instruction minor cycles, ten microseconds, be provided for the execution of the Multiplex Analog Input (MLXI) instruction.

Once the quantity has been entered into the Accumulator, it can be operated upon as though it had been entered into the Accumulator from the Number Memory or the G-Register. Thus, the quantity may be processed immediately or it may be transferred to memory for later use.

Only nine analog input channels are used in the F-100A problem (table V). Of this only six inputs are obtained from the flight compartment; the three remaining inputs are control inputs established by the instructor. As in the case of both the discrete inputs and the discrete outputs, the inclusion of the radio aids functions would have a marked effect upon the utilization of the unused channels.

3.6.4 Analog Outputs

There are sixty-four analog output channels implemented in the UDOFT computer. The conversion of the binary quantities to voltage levels, and the outputting of the voltage level for each of the sixty-four channels is performed in the Input-Output Unit.

The conversion function is performed by a static flip-flop storage register and a digital-to-analog converter. The output quantity is obtained from the sign and the eleven high order stages of the Transfer Register and is transferred to the analog output storage register. The output of the register is applied to the digital-to-analog converter. The output of the converter is a D.C. voltage level, in the range of zero to ten volts, which is directly proportional to the digital quantity appearing at the input to the converter.

The converter consists of precision resistors interconnected to form a ladder network (figure 18), in which the series resistors have half the value of the shunt resistors. Each

TABLE IV

DISCRETE OUTPUT ASSIGNMENTS FOR
F-100A SIMULATION PROGRAM

DO	NMAD	USE	DO	NMAD	USE
01	1005	STABILIZER MOTION NOT O. K.	13	1101	LANDING GEAR NOT UP
02	1006	AILERON MOTION NOT O. K.	14	1102	HYDRAULIC NUMBER 2 FAIL
03	1011	CRASH	15	1104	DYNAMIC PRESSURE HIGH
04	1012	DROP TANKS	16	1110	SPARE
05	1021	SPEED BRAKES IN	17	1120	SPARE
06	1022	SPEED BRAKES OUT	18	1140	F9F-2
07	1024	SPEED BRAKES IN MOTION	19	1201	F9F-2
08	1030	HYDRAULIC NUMBER 1 FAIL	20	1202	F9F-2
09	1041	LAND INDICATOR	21	1204	F9F-2
10	1042	STALL	22	1210	F9F-2
11	1044	STALL WARNING	23	1220	F9F-2
12	1050	LANDING GEAR NOT DOWN	24	1240	F9F-1

TABLE V

ANALOG INPUT ASSIGNMENTS FOR
F-100A SIMULATION PROGRAM

AI	NMAD	USE	AI	NMAD	USE
01	0161	THROTTLE POSITION	13	0047	F9F-2
02	0151	AILERON	14	0143	F9F-2
03	0131	ELEVATOR	15	0145	F9F-2
04	0071	RUDDER	16	0146	F9F-2
05	0162	RIGHT BRAKE	17	0027	F9F-2
06	0152	LEFT BRAKE	18	0123	F9F-2
07	0132	BAROMETRIC PRESSURE	19	0125	F9F-2
08	0072	AIRPORT ELEVATION	20	0126	F9F-2
09	0164	ICING RATE	21	0017	F9F-2
10	0154	SPARE	22	0113	F9F-2
11	0134	SPARE	23	0115	F9F-2
12	0074	SPARE	24	0116	SPARE

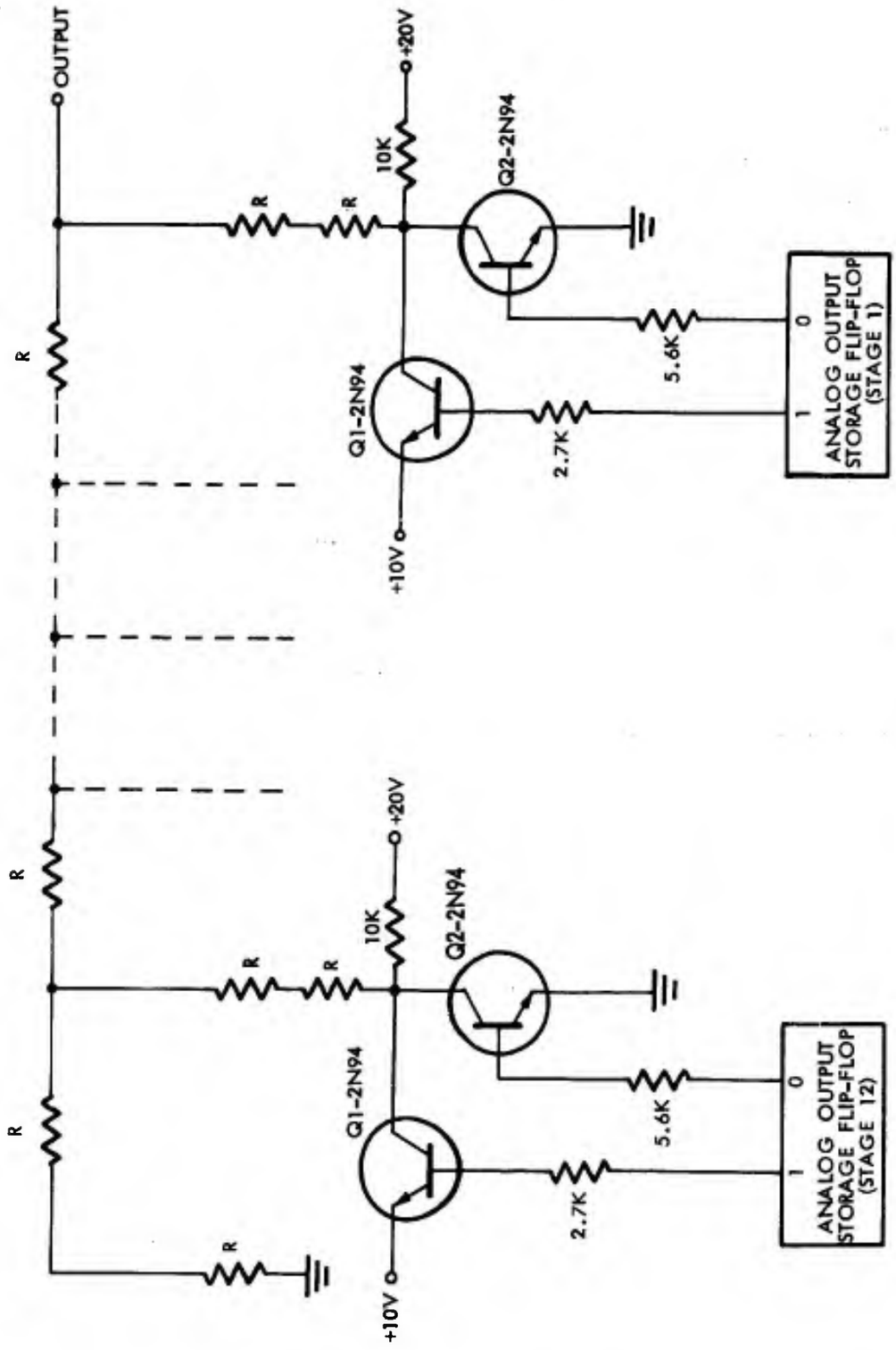


Figure 18. Schematic of Digital-to-Analog Converter

shunt resistor is returned to ground and to plus ten volts D. C. through switching transistors. The two switching transistors connected to each leg of the ladder network are controlled by the two outputs from the associated flip-flop of the output storage register. When the storage flip-flop is in the ONE state, the transistor connected to the plus ten volt supply is closed and the other transistor is open. When the flip-flop is in the ZERO state, the transistor connected to ground is closed and the other transistor is open. Thus, the shunt resistors are returned to either ground potential or ten volts depending upon the state of the driving flip-flop.

Transistor drive, rather than direct drive from the static flip-flops, was used to control the ladder network because of the near-perfect switch qualities exhibited by the transistor. When the transistor is saturated (closed) the effective resistance between the emitter and the collector is negligible; when the transistor is cut-off (open) the effective resistance between the emitter and the collector is extremely high.

The output of the converter is applied to the input of the sixty-four channel multiplexer which acts basically as 64 addressable switches. When addressed, a switch is closed, and the output of the converter, which is applied to the input of the switch, is impressed upon an analog output storage capacitor. The switch must provide a low resistance charging path for the capacitor in order that the voltage on the capacitor can be changed rapidly. Conversely, when the switch is open, a very high resistance discharge path must be provided, otherwise, the charge on the capacitor will not be maintained during the interval between successive addressings of the same output channel. Since the time interval in question is approximately fifty milliseconds, a discharge time constant of approximately five seconds is required.

The use of the addressable switches or a multiplexer eliminates the need for individual digital-to-analog converters for each output. This affords considerable savings in circuit elements, power consumption, and space requirements. A further effect of component reduction is increased reliability.

Of the sixty-four available analog output channels, approximately forty are required for the F-100A problem (table VI).

3.7 Computer Console Unit

The computer console provides manually controllable means of access to the computer and visual indication of computer status. The control function allows establishing different modes of computer operation, presetting initial conditions, and inserting data into the computer. The indication function allows examining a variety of numerical quantities within the computer in addition to indicating the status of computer performance.

The UDOFT console consists of five major sub-units, of which three are the individual console front panels that provide the control and monitoring features. The console panels were designed with the intent of grouping all controls and indicators by major function. Three major classes of functions were considered:

1. Mode selection and primary status indication.
2. Manual access and secondary status indication.
3. Maintenance aids, power control, and tertiary status indication.

The two remaining sub-units of the console unit are the (input) card reader and the (output) printer. The card reader provides the means by which bulk information is read rapidly into the computer. The printer provides the means by which bulk information is read out of the computer and converted into hard copy.

The following sections describe in detail the control and monitoring features available at the console and the two computer non-real time input-output devices, the card reader and the printer.

3.7.1 Console Panel A

The "A" panel is physically the center panel of the three console panels. It provides the computer operator with the means for establishing modes of computer operation and for monitoring status while the computer is operating (see figure 19).

TABLE VI

ANALOG OUTPUT CHANNEL ASSIGNMENTS FOR
F-100A SIMULATION PROGRAM

AO	NMAD	USE	AO	NMAD	USE
11	0000	RPM	51	0040	ROLLING RATE (F151)
12	0001	EXHAUST TEMPERATURE	52	0041	MACH NUMBER
13	0002	ACCELERATION	53	0042	CABIN ALTITUDE
14	0003	INDICATED AIRSPEED	54	0043	F9F-2
15	0004	FUEL QUANTITY	55	0044	F9F-2
16	0005	PITCH ANGLE	56	0045	ELEVATOR CONTROL LOADING
17	0006	ROLL ANGLE + SINE	57	0046	ICE QUANTITY
18	0007	ROLL ANGLE - SINE	58	0047	F9F-2
21	0010	ROLL ANGLE + COSINE	61	0050	RATE OF CLIMB (F151)
22	0011	ROLL ANGLE - COSINE	62	0051	SPARE
23	0012	RATE OF CLIMB	63	0052	SPARE
24	0013	TURNING RATE	64	0053	SPARE
25	0014	BALL ANGLE	65	0054	SPARE
26	0015	HYDRAULIC PRESSURE #1 OR #2	66	0055	SPARE
27	0016	TRUE HEADING + SINE	67	0056	SPARE
28	0017	TRUE HEADING - SINE	68	0057	SPARE
31	0020	TRUE HEADING + COSINE	71	0060	F9F-2
32	0021	TRUE HEADING - COSINE	72	0061	F9F-2
33	0022	FUEL FLOW	73	0062	F9F-2
34	0023	GROUND SPEED	74	0063	DROP TANK FUEL QUANTITY
35	0024	RUDDER CONTROL LOADING	75	0064	F9F-1
36	0025	INDICATED ALTITUDE + SINE	76	0065	SPARE
37	0026	INDICATED ALTITUDE - SINE	77	0066	SPARE
38	0027	INDICATED ALTITUDE + COSINE	78	0067	SPARE
41	0030	INDICATED ALTITUDE - COSINE	81	0070	SPARE
42	0031	TRUE AIRSPEED	82	0071	SPARE
43	0032	ALTITUDE ABOVE GROUND + SINE	83	0072	SPARE
44	0033	ALTITUDE ABOVE GROUND - SINE	84	0073	SPARE
45	0034	ALTITUDE ABOVE GROUND + COSINE	85	0074	SPARE
46	0035	ALTITUDE ABOVE GROUND - COSINE	86	0075	SPARE
47	0036	ANGLE OF ATTACK (F151)	87	0076	SPARE
48	0037	PITCHING RATE (F151)	88	0077	SPARE

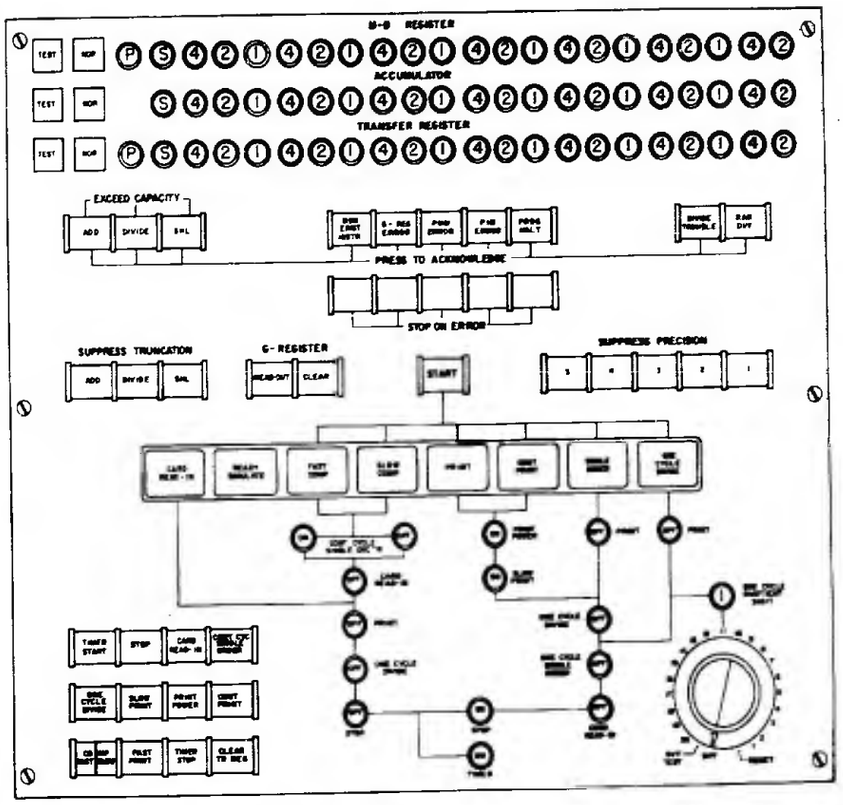


Figure 19. Computer Console Panel A

3.7.1.1 Mode Selection

The UDOFT computer has eight distinct modes of operation:

1. Fast Comp – the normal, high-speed mode of operation.

2. Slow Comp – the mode in which a fixed interval of ten microseconds is added to the normal execution time of each instruction. This mode is used primarily during program debugging because errors due to forbidden sequences of instructions are eliminated.

3. Single Order – the mode in which only a single instruction is executed. Instruction sequencing is effected manually by means of the Start switch.

4. Card Read-In – the mode which must be established in order to allow reading data from the card-reader into the computer memories. Read-in of information from the card reader is under manual control, unlike the read-in process in more conventional computers which is under program control.

5. Slow Print – the mode in which the computer is in the halted state (the program halt switch is on) and the output writer is enabled. The slow print has two modes of operation, normal mode and dump mode. In the normal mode each time the start switch is pressed the computer performs an instruction in the normal manner, and prints out a line of information. In the dump mode the computer augments the sequence counter by one, addresses both memories with the content of the sequence counter and prints a line of information each time the Start switch is pressed – no instructions are performed during this operation.

   Two formats are presently available in both the normal mode and the dump mode. The short format prints NMAD and NMAR at a rate of twenty lines per minute. The long format prints IMAD, IMR, NMAD, NMR, and AC at a rate of seven lines per minute. Both formats print the octal equivalent of the contents of the register.

   In the near future an additional format will be installed which will interpret the contents of the number memory registers as binary coded decimal and printout the decimal equivalent of the register. This format will increase the efficiency of the octal to decimal conversion routines used on the computer.

6. Continuous Print – an extension of the slow print mode. In this mode the task of pressing the start switch for each line of print is eliminated by automatically actuating the start relay at the end of each line of print. Once started the output writer will print until stopped in either mode or either format.

7. One Cycle Divide – a highly specialized mode of computer operation established to execute the multi-minor cycle divide instruction a single minor cycle or quotient bit cycle at a time. The single cycle division operation is under the control of the twenty-four position rotary switch located in the lower right corner of the panel. The quotient bit resulting from each minor cycle of the division process is displayed by the One Cycle Quotient Digit indicator. This feature is used only during periods of preventive and corrective maintenance.

8. Ready Simulate – in reality not a specialized mode of operation. This mode is essentially the fast computation mode with the added requirements so that console switches which might exert undesired influence over the computer during flight simulation must be off.

   Originally most of the console switches were included in this group. However, this proved to be too much of a restriction. At present,

due to the expanded use of the UDOFT Computer, only a few switches must be off. Examples are: stop comp, program halt, one cycle divide, and continuous cycle single order.

Once the desired mode has been selected, operation is initiated by actuating the Start switch. Operation will continue until it is stopped by any of the means available for halting computer operation. Achievement of these modes of operation is dependent upon a number of conditions. To assist in establishing the necessary conditions, a flow diagram indicating the required states of the mode control switches was devised. This was necessary, also, because there is no interlocking of the mode control switches which would have allowed direct set-up of the modes of operation.

There are eight switches that control the modes of computer operation. They are located in the lower left corner of the panel. Explicitly, they are:

1. Timer Start – a momentary contact switch which causes a pulse to be injected into the Timer loop, thus starting the Timer.

2. Stop Comp – an alternate action switch causes the operation of the computer to be halted in an orderly manner at the end of the instruction which is being executed at the time the control is actuated.

3. One Cycle Divide – an alternate action switch which establishes the necessary conditions for the one cycle divide mode of operation.

4. Card Read-In – an alternate action switch which establishes the necessary conditions for the card read-in mode of operation.

5. Cont. Cycle Single Order – an alternate action switch which establishes the final conditions for a sub-mode of computer operation not mentioned previously; namely, the continuous cycle single order mode. In this mode, established primarily for aiding corrective maintenance, the computer will execute the same instruction continuously.

6. Fast/Slow Comp. – an alternate action switch which establishes the necessary conditions for the slow computation mode of operation. Operation in the fast mode is the normal mode of operation; the abnormal mode, slow comp, is controlled by this switch.

7. Slow Print – an alternate action switch which enables the output printer and the computer mechanism which actuates the printer, and causes the computer to enter the slow print mode of operation. It is also a necessary condition, if the continuous print mode of operation is to be entered.

8. Cont. Print – an alternate action switch which establishes the necessary conditions for the continuous print mode of operation.

The four remaining switches do not directly affect the selection of operational modes. Since their functions are so varied and so singular in nature, it is not possible to classify them as a group. Therefore, the function of each switch is described separately as follows:

1. Timer Stop – a momentary contact switch which disables the computer Timer. The use of this control is so limited that there is no apparent reason for its inclusion in the system.

2. Clear TR Reg. – a momentary contact switch which causes the contents of the accumulator to be gated into the Transfer Register. If the accumulation is zero during card read-in, the switch effectively clears the transfer register by gating the accumulator (zero) into the Transfer Register. This operation is usually performed immediately prior to card read-in to ensure that the Transfer Register, which is used as the buffer between the card reader and the computer memories, will not introduce erroneous data into the first memory location addressed. The necessity for this function is eliminated when the initial

data read in from the card reader is control data which serves only to clear the Transfer Register.

3. Fast Print – an alternate action switch which enables the high-speed Printer Buffer Register. This control has been temporarily disabled, due to the extensive use of the Print Instruction and the Print Register.

4. Use TIM Inst. – an alternate action switch which, when enabled, causes the computer to execute the TIM instruction when it is decoded. Otherwise, the TIM instruction is executed as a NOP instruction.

3.7.1.2 Secondary Control

Two secondary control functions are included on this console panel. They deal primarily with the operation of the Accumulator:

1. Truncate on Overflow – three alternate action switches which, when enabled, cause the truncation of the quantity in the Accumulator if an arithmetic overflow occurred as a result of the instruction being executed. Selective control is afforded by having three switches, one for each type of instruction (Add, Divide, and Shift Left) that could cause an arithmetic overflow. The resulting truncation process causes the maximum quantity, $\pm(1 - 2^{-20})$, to be inserted into the Accumulator; the sign of the quantity being truncated is retained.

2. Suppress Precision – five alternate action switches which inhibit the normal operation of the five low order stages of the Accumulator. Although the computer operates normally with numbers whose magnitude is twenty binary bits, it is possible to reduce arbitrarily the precision of computation to as low as fifteen binary bits, by means of the five switches. This feature was incorporated for the purpose only of evaluating the fidelity of simulation as a function of word length.

3.7.1.3 Status Indication

Two distinct forms of status indication are available on the "A" panel of the console. The first form is discrete in nature and indicates the occurrence of a phenomenon. The second form of indication provides a means of monitoring the contents of the Accumulator and the three registers which communicate with it.

The discrete status indicators indicate primarily the occurrence of undesirable results. There are nine such indicators:

1. Add Overflow – indicates the occurrence of an arithmetic overflow in the Accumulator resulting from an add operation (which may occur as the result of an add, subtract, shift add or multiply add instruction).

2. Divide Overflow – indicates at the beginning of a divide operation that the dividend is larger than the divisor.

3. Shift Left Overflow – indicates the occurrence of an overflow in the Accumulator due to a shift left instruction.

4. Non Exist Instr. – indicates the decoding of the unused order type NOT.

5. G-Reg. Error – indicates that the address of the G-Register has been used with an order type which may not refer to the address of the G-Register.

6. PNM Error – indicates a number memory parity error.

7. PIM Error – indicates an instruction memory parity error.

8. Divide Trouble – indicates improper handling of the quotient digits or timing marker pulses in the G-Register. This is usually caused by failure to clear the G-Register before a divide instruction.

9. RAR Overflow – indicates an arithmetic overflow has occurred in the relative address former as a result of adding two addresses whose arithmetic sum is greater than $(7777)_8$. This is a very common occurrance, since it is often convenient to effectively decrement the address by causing an overflow.

A tenth status indicator is included with these fault indicators. However, it does not indicate the occurrence of an error in either the program or the operation of the computer. It simply indicates that the programmed halt instruction, PHT, has been decoded.

In the case of all ten of these indicators, the occurrence of a fault is acknowledged by the computer operator simply by depressing the switch-type indicator which indicated the fault.

It was possible through the design of the computer to allow the occurrence of certain of these faults to halt the computer. The faults selected were Non-Existant Instruction, G-Register Error, and Instruction and Number Memory Parity Errors. By means of the associated Stop on Error switches located immediately above the error indicators, the operator has the facility for causing the computer to halt when any of these errors are committed. A Stop on Error control, in this case a misnomer is associated also with the Programmed Halt indicator. It allows the execution of a programmed halt instruction to halt computer operation. During normal use of the computer, the Stop on Error controls are disabled; thus, the occurrence of errors, though they may interfere with the accuracy of the computed results, will not interrupt the execution of a program.

Although Stop on Error controls are not associated with the three overflow indicators, it is possible for the occurrence of an overflow to halt the execution of the program. This feature has been implemented in the computer hardware. However, a halt on overflow will occur only if the instruction immediately following the arithmetic instruction that caused the overflow is not a transfer on overflow instruction, TOV.

An overflow not followed by a transfer on overflow instruction is an invalid overflow and as such will cause the appropriate indicator to light. A divide overflow will always cause the divide overflow indicator to go on.

In slow mode every invalid overflow will halt the computer; while in fast mode the HOVW (Halt on overflow) switch must be on to cause an invalid overflow to halt the computer.

There are three banks of indicators used for displaying the contents of four registers. The four register displays are:

1. M-D Register – a bank of twenty-two indicators to display the binary contents of the Multiplicand-Divisor Register.

2. Accumulator – a bank of twenty-one indicators to display the binary contents of the Accumulator.

3. Transfer Register – a bank of twenty-two indicators to display the binary contents of the Transfer Register.

4. G-Register – the contents of the G-Register, twenty binary bits and sign, are displayed on the M-D Register indicators. Whereas the three registers mentioned previously are monitored continuously and automatically, observation of the contents of the G-Register is possible only when the computer is halted and the G-Register read-out control switch is activated.

3.7.2 Console Panel B

The "B" panel of the computer console affords the means for continuously monitoring the contents of the Instruction Memory Output Register, the Sequence Counter, and the Interval Timer; for manually inserting data into the Tally Register, the Relative Address Register, the Sequence Counter, the Interval Timer, and any register in either

the Instruction Memory or the Number Memory; and for manually calling forth data for display from any register in either the Instruction Memory or the Number Memory (see figure 20).

3.7.2.1 Status Indication

Four banks of indicators continuously indicate the status of the following computer registers:

1. Instruction Memory Output Register – a bank of twenty indicators to display the binary contents of the Instruction Memory Output Register.

2. Number Address Storage Register – a bank of twelve indicators to display the binary contents of the Number Address Storage Register.

3. Sequence Counter – a bank of twelve indicators to display the binary contents of the Sequence Counter.

4. Interval Timer – a bank of fourteen indicators to display the binary contents of the Interval Timer.

In addition there are ninety-two indicators which indicate the status of ninety-two dynamic flip-flops not otherwise monitored by any other indicator on the console. These indicators are useful to maintenance personnel by providing an additional degree of trouble-shooting capability at the console.

3.7.2.2 Control Functions

Manual control over four computer registers and the two computer memories is exercised by the switches situated in the center section of console panel B.

1. Tally and Relative Address Registers – Twelve alternate action switches are available for establishing the binary quantity to be read into either the Tally or the Relative Address Register. If the quantity is to be read into the Tally Register, the Set Tally Register switch is actuated; if into the Relative Address Register, the Set Relative Address switch is actuated. Regardless of the register being set, it is necessary first to clear the register by means of actuating the Clear Ta Rar switch. If, during operation in the single order mode, it is desired to use relative addressing in conjunction with an instruction for which relative addressing is not specified in the program, actuating the Use Relative Address switch will enable the use of relative addressing.

2. Sequence Counter – Twelve alternate action switches are available for establishing the binary quantity to be read into the Sequence Counter. The quantity is read into the Sequence Counter upon actuation of the associated Reset Switch. The reset function both clears the Sequence Counter and sets it to the desired state.

3. Interval Timer – Fourteen alternate action switches are available for establishing the binary quantity to be read into the Interval Timer. The read-in process is identical to that of the Sequence Counter as described above.

4. Instruction and Number Memory – Twelve alternate action switches are available for establishing the binary address of the memory register into or out of which information is to be read. Similarly, twenty-two alternate action switches are available for establishing the binary word that is to be read into the memory. In the case of reading into the Instruction Memory, only the first twenty of the twenty-two switches are effective. Selection of the memory to be read into or out of is controlled by the NM-IM switch. Selection of read-in or read-out is controlled by the Read In-Out switch. Initiation of the process is controlled by the Manual Oper switch. Information read into the Instruction Memory is displayed immediately upon the Instruction Memory Output Register indicators; information read into the Number Memory may be displayed

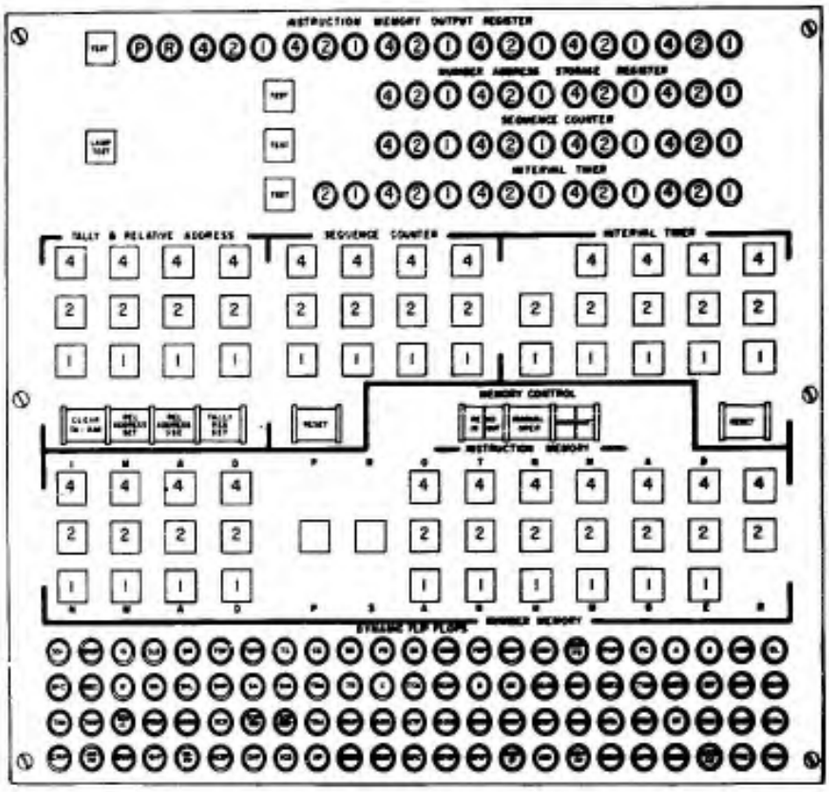


Figure 20. Computer Console Panel B

upon the M-D Register indicators by manually reading-out from the same register into which the information was stored. The same two registers display the contents of the respective memories during the manual read-out process.

It should be noted that any of the previously mentioned manual operations may be performed only when the computer is in a stopped or halted condition; otherwise, proper execution of the computer may be jeopardized.

3.7.3 Console Panel C

The "C" panel of the computer console provides the means for initiating and monitoring the application of prime power to the computer, for directing and controlling the application of the marginal checking voltage, and for minor function control and status monitoring, (see figure 21).

3.7.3.1 Prime Power Application

Large pushbutton switches are provided for connecting and for disconnecting the computer from the source of prime power. Three indicators indicate, in sequence, the application of prime power to the power supplies (Power On); the application of full filament voltage to all the vacuum tubes (Filament On); and the application of regulated D. C. power to the computer circuits (D. C. On).

3.7.3.2 Marginal Checking Voltage

Three banks of eight alternate action switches are provided to allow the selection of a particular bay of the computer to be marginally checked. Eighteen of the twenty-four switches connect the +80 marginal check voltage to the pulse amplifiers in the associated bay. Actuation of any one of these switches will illuminate the +80 MCV indicator which serves to indicate that the associated voltmeter is used to monitor the variation in the marginal checking voltage. The two forms of application of the marginal checking voltage are controlled by the MCV Selector Switch. One form is the application of a fixed increment of voltage; the other form is the application of a variable increment of voltage which is controlled by the MCV Variation Control. The remaining six bay selector switches connect the +150 marginal check voltage to the sense amplifiers and the static flip-flops in the Memory Unit. Selectors 1E1-MS and 1E4-MS select the sense amplifiers; selectors 1E1-MBR and 1E4-MBR select the Memory Output Register flip-flops and the Memory Rewrite Register flip-flops; and selectors 1E2-MAR and 1E3-MAR select the Memory Address Register flip-flops.

3.7.3.3 Status Indication

Three different forms of status indication are provided:

1. Cabinet Power or Blower Failure. At each of the major units of the computer, an indication of power failure or cooling-air blower failure for that unit is provided. Since not all of these units are visible from the computer console, it was deemed necessary to provide a single indicator at the console which would indicate a power or a blower failure anywhere in the system. The single Cabinet Blower-Power Failure indicator at the console flashes a red warning when such a condition exists. A buzzer sounds simultaneously to make the warning more prominent. The operator acknowledges the presence of a failure by disabling the audible alarm; however the red warning continues to flash as long as the failure exists.

2. Error Counters. Electro-mechanical counters are provided to maintain an account of errors that occur in the computer. Since a consistently made error could recur at a very high rate, the circuitry required for accounting for such errors would be extravagent. Therefore, the counters are simply activated by the error acknowledge switches. As a result the quantities displayed by these counters represent conservative estimates of system failures. Operating experience has shown that these error counters contribute little to improving the operation of the system.

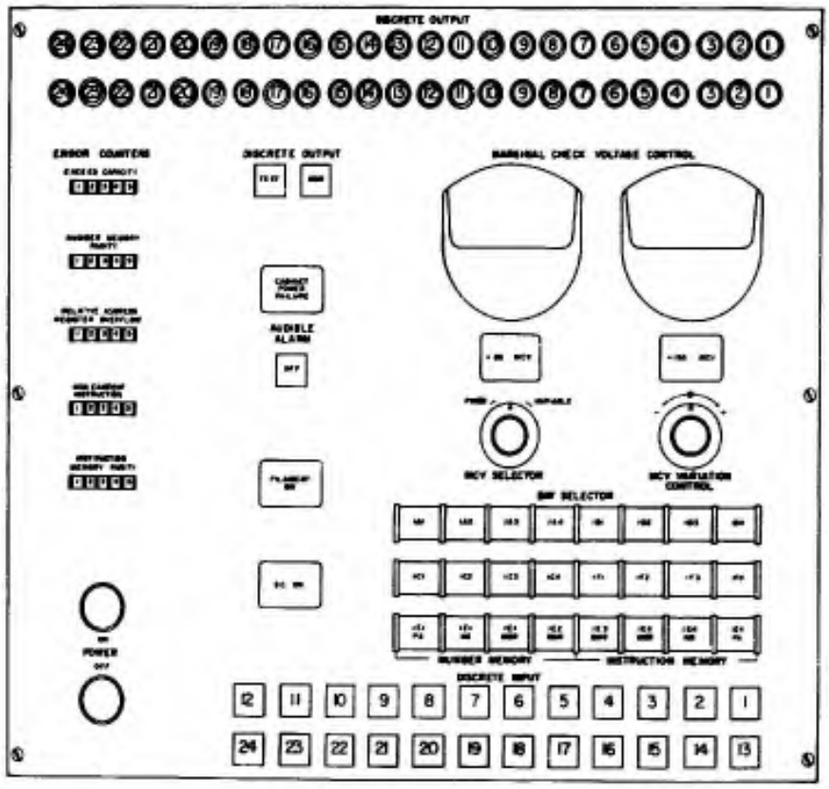


Figure 21. Computer Console Panel C

3. Discrete Output. Twenty-four pairs of indicators indicate the status of the twenty-four discrete output channels. During the execution of a simulation program, the indications are meaningless to the uninitiated. However, to the experienced programmer, they provide means, at the console, for indicating the occurrence of special events which are pertinent to the problem being solved. Also, they are used to indicate proper and improper operation during the conduct of some of the preventive maintenance programs.

3.7.3.4 Control Functions

Since the discrete inputs of the UDOFT computer provide a feature which is similar, in a small way, to the addressable flip-flop feature of the more conventional general-purpose digital computers, it was considered that control of some of these inputs from the computer console would enhance the utility of the computer. Thus, controls for twenty-four of the sixty-four discrete inputs were provided. Like the discrete outputs, the discrete inputs find their greatest use during the execution of some at the preventive maintenance programs.

3.7.4 Input Card Reader

The device used for reading punched-card information into the UDOFT computer is an IBM 514 Reproducing Punch. The reader is capable of reading cards, one line at a time, at the rate of approximately one hundred cards per minute.

Originally, the UDOFT binary cards were arranged with only five words per card. With this configuration of punched-card data, it required approximately fifteen minutes to load the F-100A simulation program into the computer. The word configuration has since been improved so that twelve words may be punched into a single card, thus decreasing read-in time for the F-100A simulation program to approximately six and a half minutes. In addition, the improved word packing density results in a 60% savings in punched cards, thereby minimizing card handling and storage problems.

In addition to its use as the input card reader, the reproducing punch is also used to reproduce and verify decks of punched cards. This feature is highly desirable since continued use of a single deck results in excessive card wear which will lead ultimately to erroneous read-in and possible fouling of the card reader mechanism.

3.7.5 Output Printer

The UDOFT computer uses an IBM electric typewriter for the output printer facility. Actuation of the printer is controlled manually by the Slow Print and Continuous Print mode selection switches located on the console "A" panel. In order to form a single line of copy for printing, the following steps are taken:

1. The Sequence Counter is reset to the address of the desired instruction.
2. The Slow Print selection switch is actuated.
3. The mode of printing, normal or dump, is selected.
4. The format short or long is selected.
5. The Start switch is actuated.

The computer will execute the single instruction and halt. Immediately upon halting, the typewriter will print a single line of copy, at the completion of which the carriage of the typewriter will be returned and all operation will cease until the Start switch is again actuated. In the Continuous Print mode of operation, the return of the carriage will automatically initiate the next cycle. This will continue so long as the computer remains in the Continuous Print mode. The form of the copy for the normal, mode, long form is shown on the following page.

7761	r340001	7041	0034640	0024640
7762	r340002	7042	0475006	0475006
7763	r340003	7043	0250560	0250560
7764	r340004	7044	-0413570	-0413570
7765	r340005	7045	0057674	0057674
7766	r340006	7046	0000000	0000000
7767	r340007	7047	0000000	0000000
7770	340000	7050	0000000	0000000
7771	037760	7760	0632700	0000000
7760	r110010	0010	0001000	0000000
7761	r340001	7051	-0267760	-0267760
⋮	⋮	⋮	⋮	⋮
7766	r340006	7056	7777776	7777776

The five columns of the print-out are, respectively, the Instruction Memory Address (IMAR), the Instruction Word (IMR), the effective Number Memory Address (NMAR), the quantity read out of the addressed Number Memory Register (NMR), and the content of the Accumulator.

When one desires to obtain only numerical results which have been stored in memory during the execution of a program, the repetitious clear-add instructions serve only to address the desired data words in memory and place them in the Number Memory Output Register(NMR) where they may be sampled by the print-out system. Since the maximum speed of the typewriter is only seven lines per minute, an excessive amount of time is consumed in obtaining the desired data which is contained in only two of the five columns of the print-out.

For this reason a short form of print-out was added, in which only the third and fourth columns (NMAR and NMR) of the long-form print-out are generated. This results in the print-out form which is shown below. (normal mode, short form)

7041	0024640
7042	0475006
7043	0250560
7044	-0413570
7046	0000000
7047	0000000
7050	0000000
7760	0632700
0010	0001000
7051	-0267760
7056	7777776
⋮	⋮

As a result of the reduction in the amount of data required to be printed out, the effective print-out speed was tripled, (about twenty lines of copy per minute). This improvement has recently been carried one step further by adding a dump mode. In the dump mode the contents of the memories can be displayed without performing instructions. During the dump mode in either short or the long format, the sequence counter is augmented by one at the end of each line of print and the contents of the sequence counter is forced into both the Number Memory Address Register and the Instruction Memory Address Register making the information in the addressed registers available to the printer.

SECTION IV

COMPUTER HARDWARE DEVELOPMENT HISTORY

4.1 Introduction

As stated previously, the design of UDOFT was conceived at the Moore School of Electrical Engineering of the University of Pennsylvania. This design resulted from a study of the feasibility of a digital flight trainer.

An evaluation study of the MSEE computer design and programming approach was performed by Remington Rand Univac, Division of the Sperry Rand Corporation, for the U.S. Naval Training Device Center at Port Washington, New York, under contract number 1743(00).

The purpose of this study was to determine whether the computer logic, circuitry, and general approach to the simulation problem as proposed by MSEE were adequate for digitally simulating the performance of an aircraft in real-time.

The following sections, though only of a summary nature, presented factually some of the modifications and recommendations made before and during the computer development stage. Reasons for the modifications and important considerations involved are evaluated objectively.

4.2 Logic Design Problems

The following account discusses the more prominent changes and improvements implemented in the logic design of the UDOFT computer. These modifications resulted from the Remington Rand Univac study and from the efforts of design engineers at Sylvania and at MSEE.

4.2.1 Number Memory

As a result of the programming activity at MSEE, it was soon realized that the originally planned number memory, with capacity for only 1024 words, could not accommodate the figures and constants required by a complete flight simulation program. Since the coincident-current magnetic core memories had not at that time been developed for the computer, it was later a simple task to alter the capacity requirement for the number memory to 4096 words, thereby making it identical to the instruction memory. The change eased the memory design problem to some extent, calling for the development of only a 4096 word memory, rather than two memories with different capacities.

The necessity for a larger memory was proved when the complete F-100A simulation program, without any consideration for temporary storage or test data, was found to use approximately 3350 number words.

The increased capacity of the number memory affected the instruction memory. For a number memory of 1024 words, the operand address field was ten bits: for 4096 words, this field had to be increased to twelve bits.

Although the design of core-memories was not included in the initial computer design formulated at MSEE, the memory for both instruction and number memory control functions had been determined. A part of the memory control function controls the memory cycle. After access to a memory register, the initial design indicated, the memory would continue to cycle at memory address $(0000)_8$ until a new address was presented. It was found desirable, primarily for maintenance purposes, to allow this to take place; maintenance personnel can readily examine memory drive current waveforms without having the computer execute an instruction.

The memory logic, as well as the logic of the rest of the computer, was developed with the intent to design a powerful computer to meet the demanding application. The programmer, ultimate user of the computer, was as a result forgotten in many cases. A significant area where this occurred was the manually controlled access to the memories; no provision had been made originally for manually writing into or reading out of either memory. This was remedied by adding the manual read-write provision, whereby access to any memory storage register is available to the programmer by means of computer console switches.

4.2.2 Parity Formation – Card Reader Input

Originally it was necessary, when preparing punched cards for the entry of information into the computer memories, to enter the parity data into the punched cards. It would have been difficult enough to punch a very large amount of binary input cards without the added burden of determining parity. However, a striking relief from this burden was obtained by having the computer determine parity during the card read-in process. By utilizing the modified Number Parity Former, parity is formed automatically for a word written into either memory from the punched card input equipment.

In retrospect, a superior approach would have been a comparison check of the punched card parity bit with the automatically formed parity of the inserted word. This feature would maintain a constant check on the validity of the information being read into the computer from the card reader. As things are, it is quite possible for erroneous information to be read into the computer unnoticed. The requirement for parity data on the punched cards, the original reason for modifying the Number Parity Former, is no longer a problem; the assembly program, UD3, is capable of determining parity and can be modified to cause the punching of the parity bit in addition to the currently required information.

A bonus feature would be the inclusion of automatic parity formation for data inserted manually from the computer console. At present, parity must be determined and inserted manually.

4.2.3 Interval Timer

The performance of an iterative real-time program requires the availability of a real-time clock or some other means by which arbitrary intervals of real time can be established. In drum-type computers there is little need for such a device, since the feature is part and parcel of the rotating magnetic drum's mode of operation. However, in a random-access core memory digital computer, there is no device which is cyclic with respect to relatively long intervals of real time. To fill this need, the Interval Timer was introduced. Like any timer, this device must be set for some arbitrary interval of time and must indicate that the fixed interval has passed. Thus the Interval Timer required the addition of two computer instructions, SIT and SENIT.

The SIT instruction retains its original function, namely, to set the Interval Timer to some arbitrary count. The SENIT instruction, on the other hand, has been modified. Originally the SENIT instruction was a conditional transfer of control instruction which functioned only when the Interval Timer had run down to zero. If the count in the timer were not zero when the SENIT instruction was decoded, the computer would remain in an idle state until the timer had run down to zero. This seemed wasteful of valuable time that could be devoted to some worthwhile, though not mandatory, computing function. Therefore the SENIT instruction was modified to its current form.

4.2.4 Additional Instructions – SCRNM and TIM

The use of two memories, one for instruction words and the other for number words, not only effectively doubled the computer speed but also isolated instruction words from number words. This isolation was carried through the computer design to such an extent that no means was provided for the modification of instructions. The only program-modification facility available in the computer was the cumbersome relative-addressing. This was thought initially to be sufficient. Since the computer was to handle only one class of problems, involving real-time flight simulation, there seemed no need to incorporate such a feature, even though it is found in nearly all contemporary digital computers. Further, it was felt that the lack of instruction modification enhanced the reliability of the computer. This conclusion was based on the premise that, with no modification of instructions, the instruction program is inviolate; that is, the probability of disrupting the instruction program in memory is decreased greatly because nothing may be done, either correctly or incorrectly, to disturb the identity of any instruction word once it has been read into the memory.

This barrier was relaxed, though never removed, to allow the addition of two program-modification instructions, SCRNM and TIM. The SCRNM instruction is not a program-modification instruction, but in a limited way provides some of the features of

a complete form of program-modification. It contributes a degree of flexibility to program formulation that otherwise could not exist, with the two memories isolated as they are.

The TIM instruction provides a form of program modification to the extent that, during the execution of a program, it is able to store new information (instruction words) in any location in the instruction memory.

Until recently, the use of this feature had been limited to maintenance and test program; real-time simulation programs did not utilize it. However, in some of the simulation programs larger instruction memory capacity was needed and effectively obtained by using the TIM instruction. The underlying doubt as to the advisibility of this feature has all but disappeared.

4.2.5 Sequence Counter

The primary function of this device is to provide a sequence of addresses for the instruction memory. A parallel register, all stages having the same clock phase and having the information available at the same time is required for this task; it must be capable of being augmented by one in five microseconds. These requirements led to the development of a register producing a non-standard count; e.g., the sequence followed by the first two stages was 00, 01, 11, 10, 00. Although assembly programs could be written to use such a count, this placed an unnecessary burden on computer programmers and users by imposing the use of conversion tables. To eliminate this burden a standard binary counter with improved control logic was developed. The technique employed was the serializing of the parallel register's contents in two groups, odd bits and even bits, and the setting of the count two bits, one odd and one even, at a time. This effectively halved the normal counting time. The counter actually interrogates the bits two at a time, beginning with the least significant two, and changing all ONEs to ZEROs until the first ZERO is encountered. The first ZERO is changed to a ONE and all the remaining bits are left unchanged, thereby effectively adding one to the contents of the register.

4.2.6 General Purpose Computation

The GPC, or slow mode of computation, was established to allow the complete execution of an instruction prior to the initiation of the next instruction. In the fast mode of computation, it is possible for three successive instructions to be in various phases of execution simultaneously. This precludes rapid computer maintenance; hence the selection of the slow mode. Originally, five microseconds was the extension to each instruction when executed in the slow mode. Five microseconds was found to be insufficient because some instructions, such as MAD, extend their advertised execution times beyond that time period. The resultant modification increased the five microsecond extension to ten microseconds.

4.2.7 Non-Existent Instruction

Before the SIT, SENIT, SCRNM, TIM, TOZ, and MOP instructions were added to the computer repertoire, there were available a number of unused order codes. With expansion in mind, the computer designers implemented the order-type decoders for the unused order codes. Since the order codes were unusable, the decoding of such an order code would indicate some type of memory malfunction. Therefore the outputs of the unused decoders were combined to signal the occurrence of a non-existent instruction, and if desired, to halt the computation thereon. Subsequently added instructions depleted the number of unused order-type decoders until only one remains. It is still referred to as NOT, Non-existent Order Type. Since it functions identically to the programmed halt instruction, the progrmmers use it as another form of controllable program halt; the maintenance engineers use it to validate the loading of instruction memory from punched cards.

A great many modification have been made and are still being made to the UDOFT computer. To record them would require a voluminous report and would lend nothing to the state-of-the-art of computer logic design. To reiterate a previous statement, the details of the logic are secondary to the function of the logic. Since many of the logic modifications involve the details of computer logic and are concerned with only the UDOFT computer, the discussion of logic design problems will be terminated at this point.

4.3 Circuits

This section gives information on the more complex circuits used in the UDOFT computer. Theory of operation, problems encountered, and means used to solve these problems are briefly discussed for each circuit. Each circuit is categorized under the computer section where its function is most significant. For instance, the pulse amplifier appears under the heading of Main Frame Circuitry; the sense amplifier appears under the heading of Memory Circuitry.

4.3.1 Main Frame Circuitry

The pulse amplifier is the basic circuit used to implement the computer logic; pulse amplifier assemblies comprise approximately 40% of the total number of plug-in assemblies in the computer system.

A careful evaluation of the original MSEE design was undertaken (see figure 22). Several breadboard models were used in this study. Although the models functioned adequately, modifications were deemed necessary to increase the operational tolerances.

4.3.1.1 Pulse Amplifier Modifications

The 1N118A diode, the universal logic diode used for such purposes in computers as AND gates, OR gates, isolation and clamping, was replaced by the S403G diode. This diode's characteristics, such as forward voltage drop, are superior to those of the 1N118A. Furthermore, groups of these diodes showed more uniform characteristics than did the 1N118A when tested in groups.

A design tolerance of ±10% was required on bias voltages to account for variation in decoupling drop, power supply regulation, transients on the input lines, noise pickup, and prime power variation. This tolerance required changing the -2.1 volts and -3.0 volts supplies to -3.0 volts and -4.5 volts respectively. The ±18.9 volt supplies were changed to ±20 volts for convenience only. The -1.3 volt biasing level was found to be high. The +45 volt plate supply was changed to +80 volts, to reduce screen dissipation and to provide larger output pulses for driving the revised gating structures.

Discrete output and console indicators were to be controlled by relays in the plate circuit of the pulse amplifiers. This method was discarded because the average pulse current could not be maintained sufficiently constant from tube to tube. A low-frequency transistor, connected as an inverter, was used instead of the relay. The output of the pulse transformer drives the base of the transistor which, because of its poor characteristics and the lack of a speed-up capacitor, averages the pulses and stays saturated. Relays or indicator lamps may then be placed in series with the transistor's collector. The transistor adds only a negligible load to the pulse amplifier.

The pulse transformer design was changed because attempts to manufacture satisfactory transformers by the MSEE method ended in 60 percent failures.

4.3.1.2 Operation of Pulse Amplifier

The pulse amplifier is a "logic" package, having one or more input AND gates of varying configurations. Such an AND gate, when coincidentally presented with a clock pulse and the proper input signals, will produce an output pulse of standard dimensions. The output pulse is a replica of the input clock pulse, delayed a specified time by the pulse amplifier circuit. This delay is compatible with the overall timing of the computer; it is measured with respect to the input clock pulse, and is considerably less than one clock phase (0.167 μsec). Thus the output can be applied to other logic circuits, and will arrive at the load before the start of the next clock pulse. The design of the pulse amplifier permits some timing discrepancy between the input pulses, with the exception of the clock pulse. The only requirements are that the input pulses arrive before the clock pulse and that they are present for at least a portion of the clock pulse.

The following four types of output are provided:

a. Positive Clamped – Most commonly used output; used to drive other pulse amplifier packages in the same cabinet

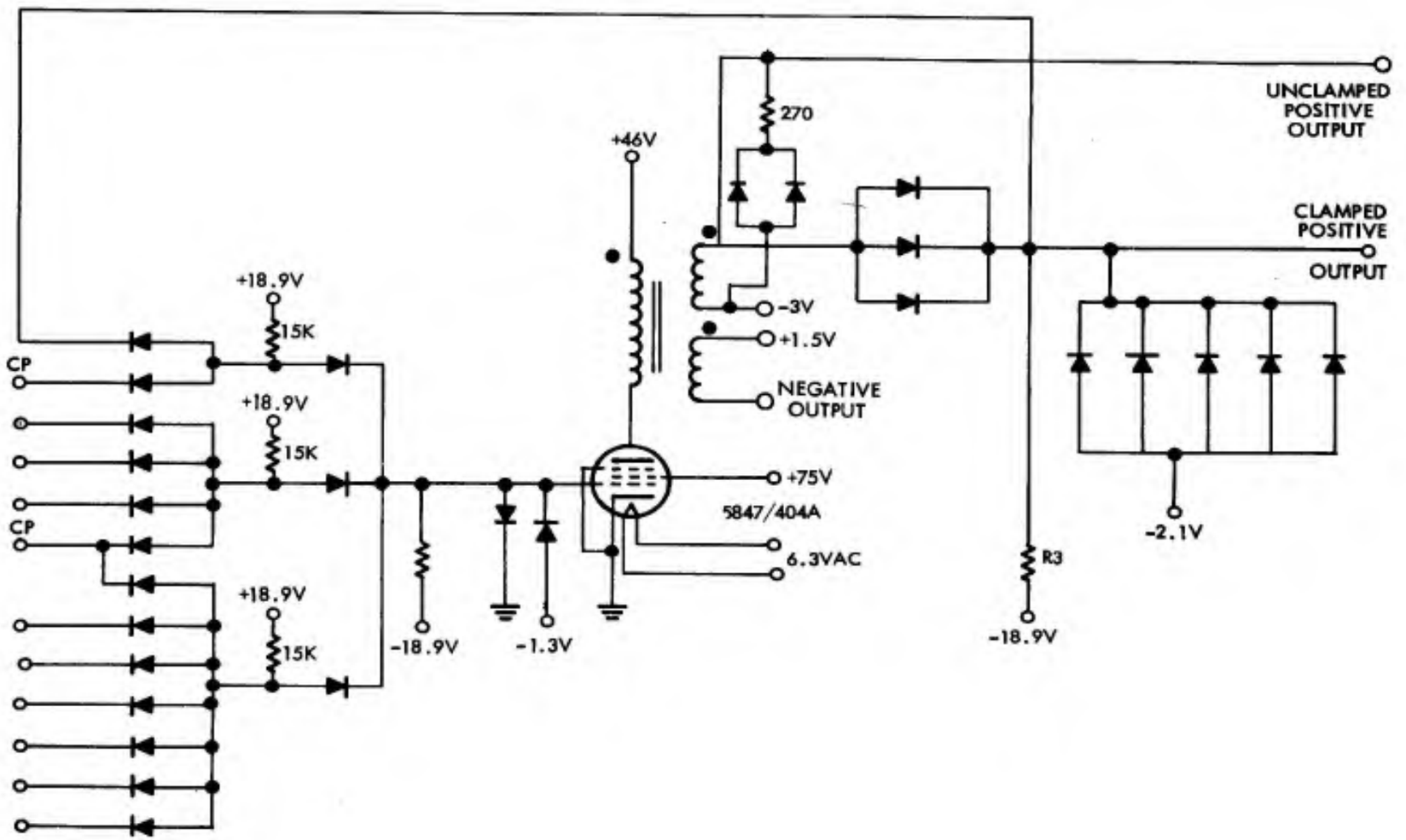


Figure 22. MSEE Pulse Amplifier Circuit

b. Positive Unclamped – Larger amplitude than positive clamped output; required to combat the effect of signal losses in OR gates, coaxial cables, and delay lines

c. Double Amplitude – Used to drive long delay lines where pulse attenuation becomes an important factor

d. Negative Unclamped – Used as an inhibit pulse to prevent an AND gate from being turned on

The pulse amplifier circuit (figure 23) contains one or more input AND gates and a recirculation AND gate, which, through an OR gate, controls the grid of a vacuum tube amplifier. When coincidence occurs at all the inputs to any AND gate, (figure 24) the tube changes from a state near cutoff to a state of high conduction. The latter state is maintained as long as one of the input AND gates or the recirculation AND gate is ON.

All pulse amplifier circuits contain the recirculation gate, which is a two-diode AND circuit. One of the inputs to this gate is the clock pulse; the other is pulse amplifier output feedback from the output transformer shown in figure 24. The recirculation circuit helps shape the output pulse by making it approximately the same duration as the clock pulse.

The output pulses are developed across the two secondary windings of the pulse transformer in the plate circuit of the tube. The manner in which the secondary windings are connected and the clamping applied determines the types of output provided. In addition, some of the pulse amplifier circuits are provided with a transistor in the output circuit to provide a D. C. output when the circuit is used as a dynamic flip-flop. The transistor averages the output pulse current and provides a D. C. voltage level to control external circuits.

When the logic requires inhibiting an input to a particular AND gate, an additional input to that gate must be supplied. This input is the clock phase preceding the normal clock phase. An examination of figure 25 explains the reason for the added clock pulse. The inhibit pulse is approximately the same width as the normal clock pulse and, since the inhibit pulse arrives first, there is an unblanked portion of the clock pulse that could allow the pulse amplifier to produce an output. To remove this possibility, the clock pulse of the previous phase is fed to the gate. When the inhibit pulse goes positive, the gate is held negative by the added input. Addition of this clock pulse input in no way alters the normal operation of the gate.

The pertinent timing characteristics of the pulse amplifier circuit are:

Circuit delay (at 1.6-volt level)	0.115 μsec max
Pulse width (between 1.6-volt points)	0.317 μsec max
Rise time (to 90% of amplitude)	0.05 μsec max
Fall time	0.05 μsec max

4.3.1.3 Pulse Amplifier in Dynamic Flip-Flop Configuration

To satisfy the logic requirements of the computer, it is often necessary that a chain of pulses be provided. The pulse amplifier is designed to perform this function, i.e., that of a dynamic flip-flop. By definition, when the pulse amplifier in the dynamic flip-flop configuration is in the one state, it produces an output pulse for every clock pulse period (0.833 micro-second); when in the zero state it has no output. The dynamic flip-flop has several advantages over the static type; faster operation, less critical circuit values, no resetting, availability of varied outputs, lower power requirements, and increased reliability.

Figure 26 depicts a pulse amplifier in the dynamic flip-flop configuration. The circuit is started normally, with pulses present at inputs A and B of Gate No. 1 coincidental with the appearance of a clock pulse. The output pulse is available 0.115 μsec later; it is then delayed an additional 0.8 of a clock period, or 0.667 μsec, before appearing at one of the inputs to Gate No. 2. Since the total delay encountered is

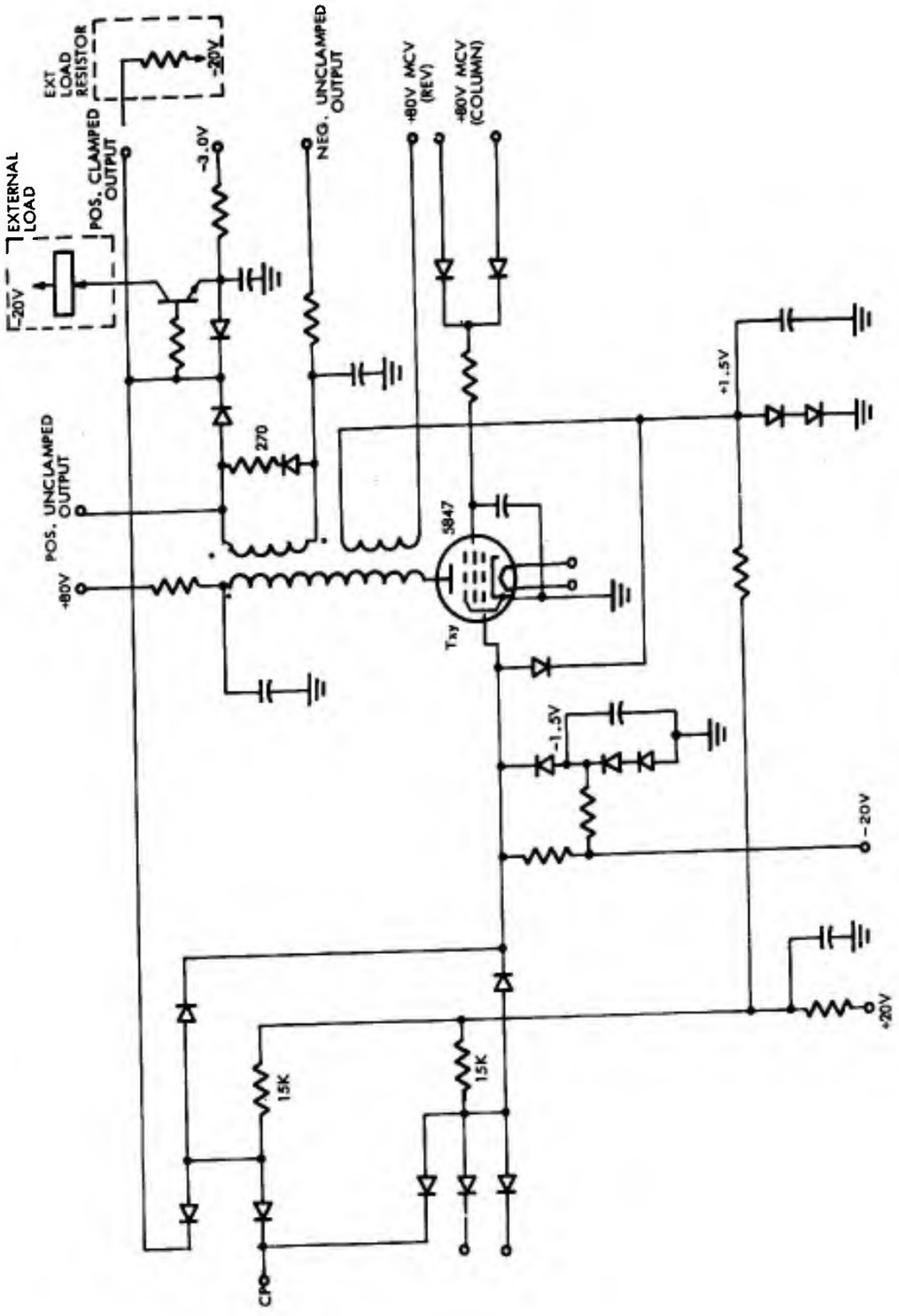
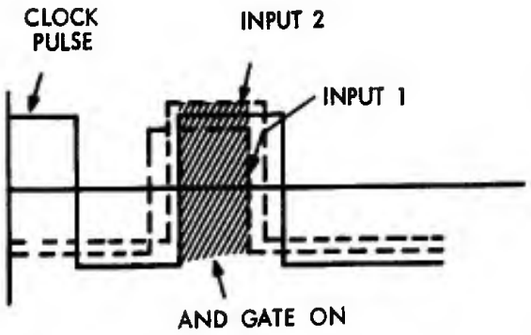
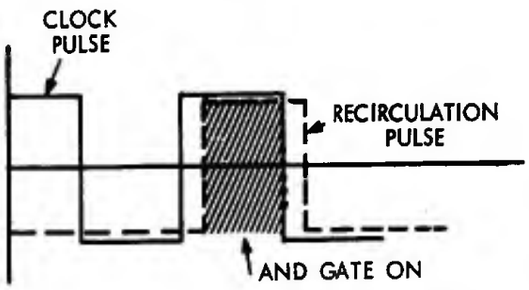


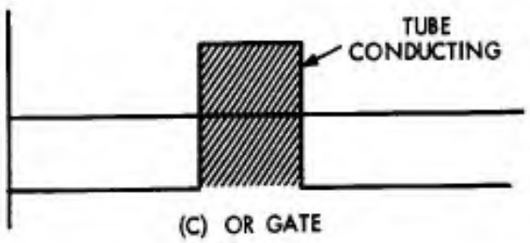
Figure 23. Revised Pulse Amplifier Circuit



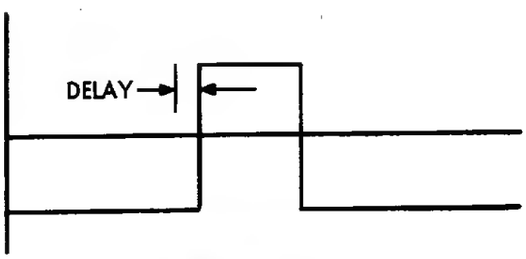
(A) INPUT AND GATE



(B) RECIRCULATION AND GATE

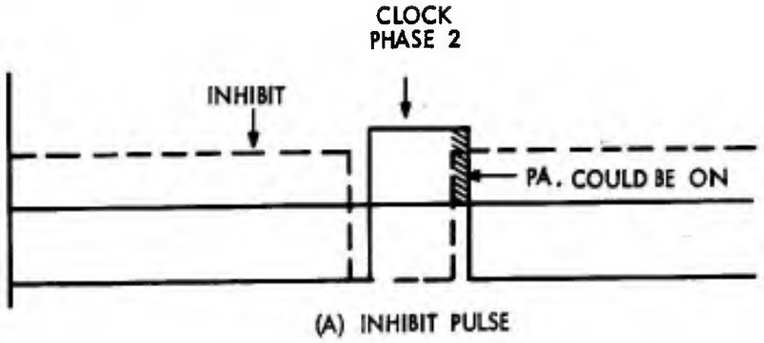


(C) OR GATE

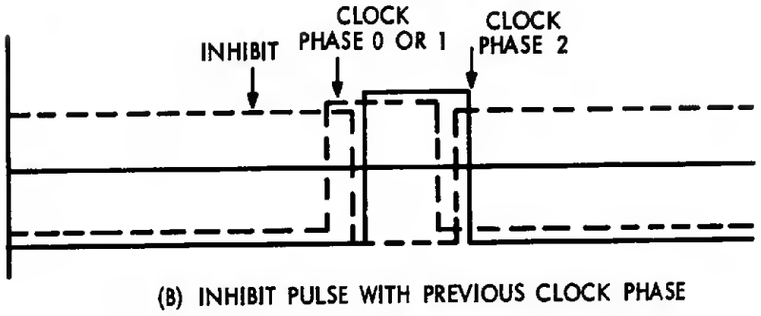


(D) OUTPUT

Figure 24. Pulse Amplifier Timing



(A) INHIBIT PULSE



(B) INHIBIT PULSE WITH PREVIOUS CLOCK PHASE

Figure 25. Inhibit Pulse Configuration

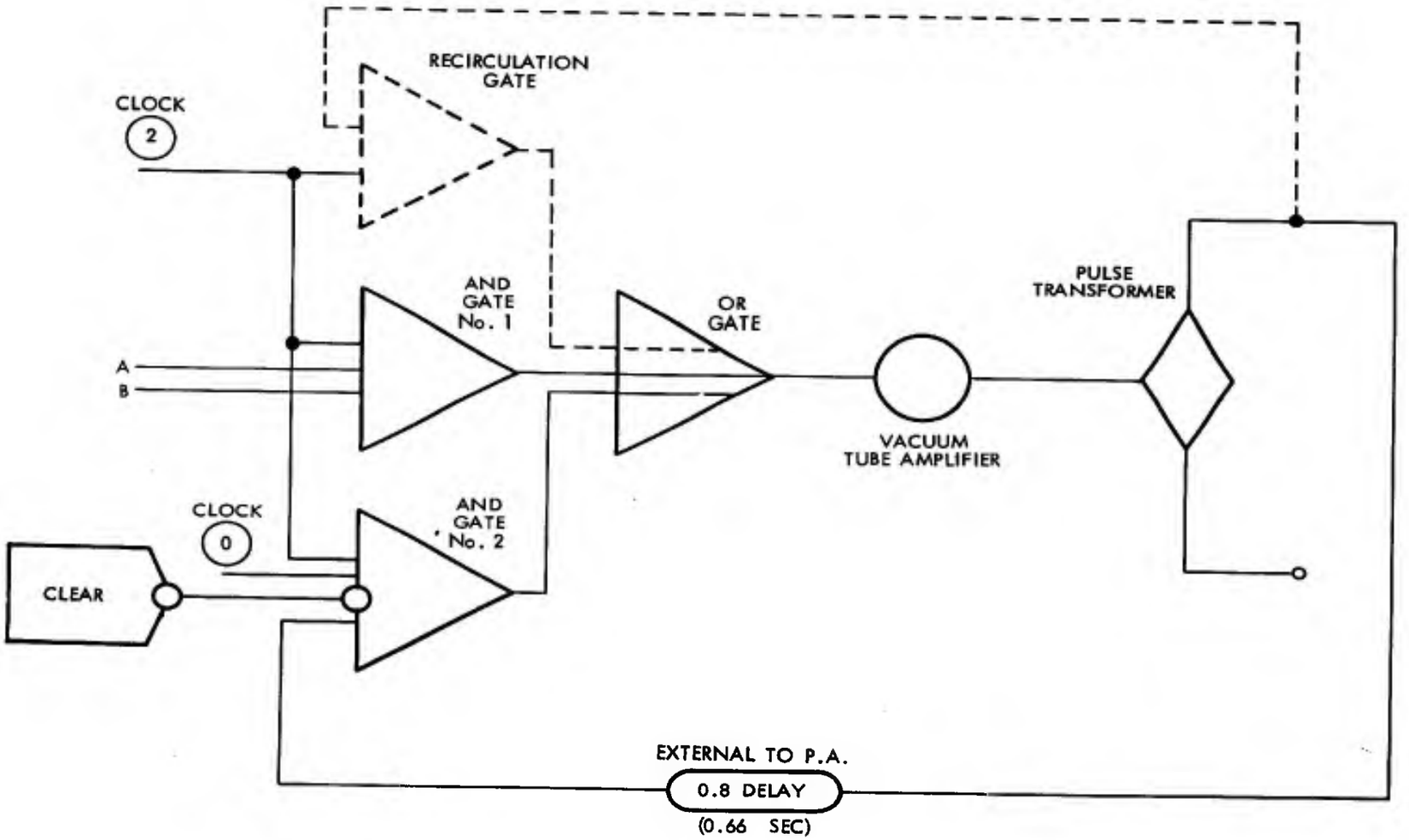


Figure 26. Logic Configuration of Dynamic Flip-Flop

equal to the time between two successive clock pulses, the output arrives at Gate No. 2 coincidently with the second clock pulse. The second input to Gate No. 2 is held positive by the reference voltage of the inhibit pulse, and the third diode is held positive by the previous clock phase. The always-present normal clock phase is also an input to Gate No. 2. Thus, the second AND gate is ON every time the clock pulse arrives, once the circuit is started by Gate No. 1. The circuit is stopped by applying an inhibit pulse to Gate No. 2.

4.3.2 Input-Output Circuitry

The method of implementing the digital-to-analog conversion, the information sampling rate, and the distribution of this information in a prescribed time interval to a control system are important factors in accomplishing successful design for digital flight simulation. A detailed description of the problems involved in the development of the digital-to-analog conversion system used in UDOFT is accordingly presented.

This analog output system for UDOFT is shown in block diagram form in figure 27. The system includes a 12-bit number storage register, a digital-to-analog converter, a 16-bit address selection register, and 64 multiplexer channels. The converter consists of switching transistors and a precision resistance ladder network whose output becomes the voltage input for each multiplexer channel. Basically, each of these channels consists of selection diodes, a diode bridge, a storage capacitor, and two cathode followers.

The flip-flops of the selection matrix provide inputs to the selection diodes which enable the diode bridge. This bridge provides isolation of the storage capacitor from the input in the OFF condition, and allows the storage capacitor to charge or discharge when the bridge is enabled. The cathode followers isolate the storage capacitor from the load and provide drive capability.

When a Multiplex Analog Output (MLXO) instruction is decoded, the number register sets the converter to the new value of the variable and the address selection register enables the multiplexer channel assigned to that address. Each channel output is stored capacitively, eliminating the need for an individual digital-to-analog converter for each output. The time-sharing of the one converter among the 64 multiplexer channels is determined by the program. The only restriction imposed upon the program is that each multiplexer channel used must be re-established every 50 milliseconds and each channel must be addressed for a minimum of 100 microseconds to insure complete charging of the storage capacitor.

4.3.2.1 Analog Output Multiplexer

The sixty-four multiplexer channel circuits provide the means by which the output of the digital-to-analog converter is applied to one of the sixty-four external loads specified by the address stored in the analog output selection register (see figure 28). The channel circuits are arranged in an 8×8 array, and have eight flip-flops associated with each co-ordinate. The outputs of the flip-flops are connected to the gate diodes of the multiplexer channels so that when one flip-flop along each co-ordinate is in the one state, the corresponding multiplexer channel is selected.

The multiplexer and the associated reference packages were primarily designed to control servo-mechanisms, although in more recent applications they have driven a variety of devices without using the reference packages.

Each multiplexer channel circuit must satisfy four basic requirements:

The output must follow in linear fashion the input from the digital-to-analog converter. (The linearity of the multiplexer is practically perfect, since all nonlinearities are second order. Unbalance in the currents I_1 and I_2 (figure 29) causes a current to flow in the input, which adds a linear inaccuracy term. Discharge (or charge) of the capacitor adds a linear inaccuracy term, but causes no non-linearity. Unbalance in the diode characteristics causes an offset but no non-linearity.)

The output must be offset from the input as little as possible.

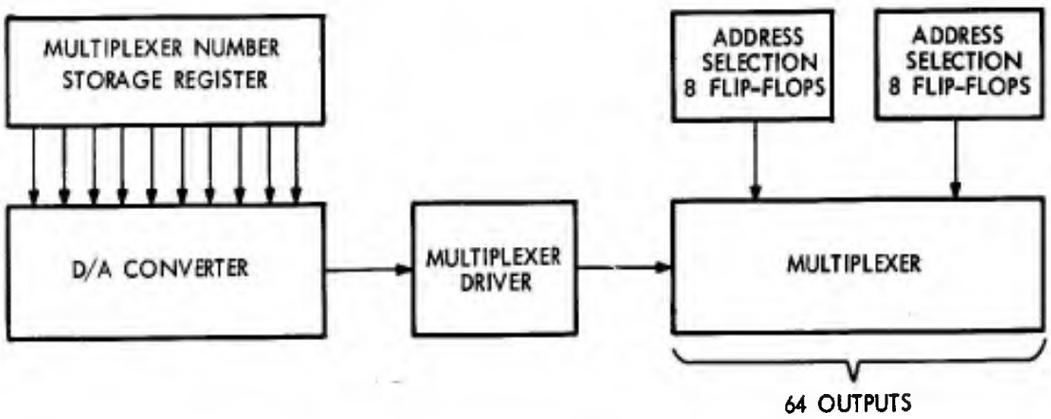


Figure 27. Block Diagram of Analog Output System

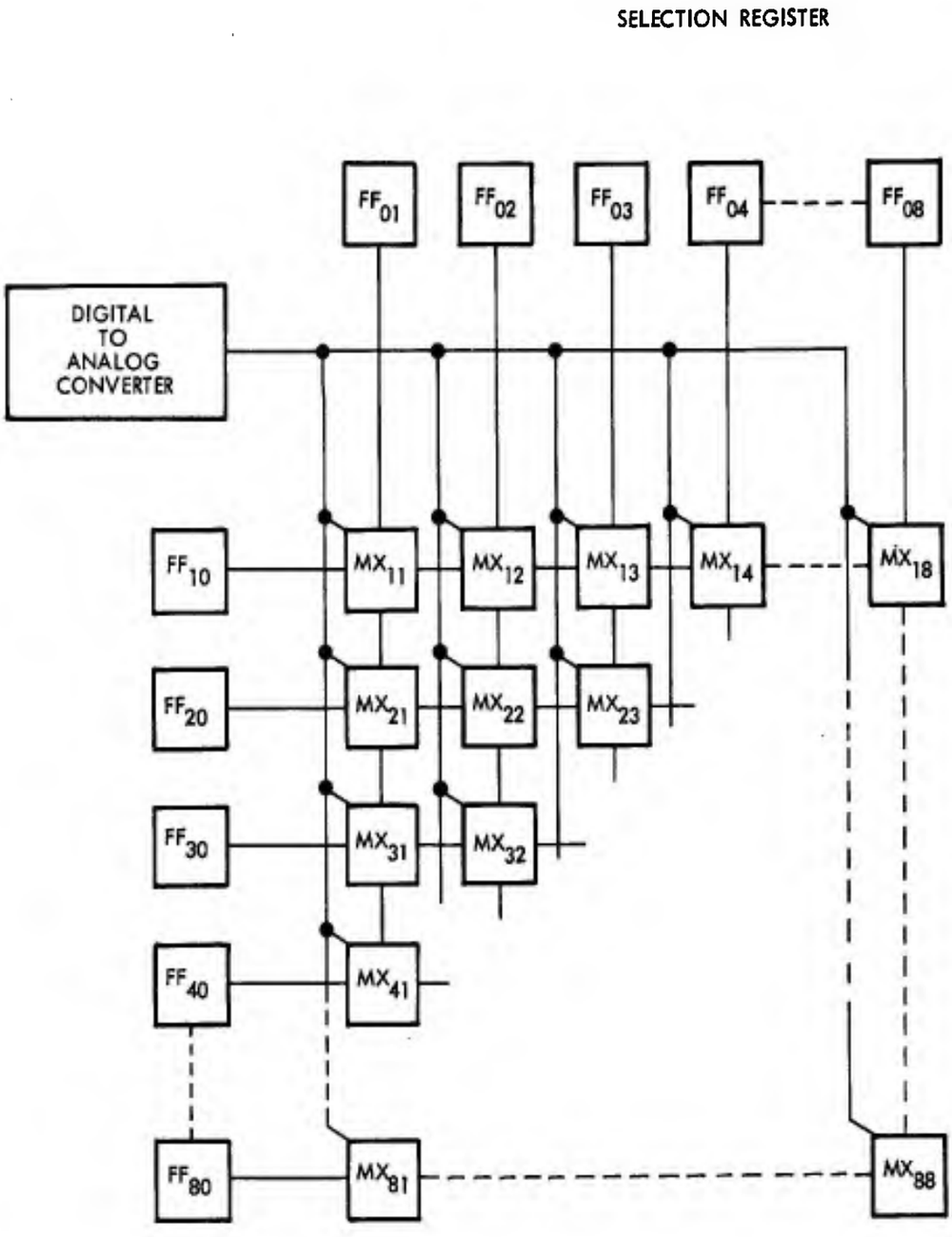


Figure 28. Block Diagram of Multiplexer Arrangement

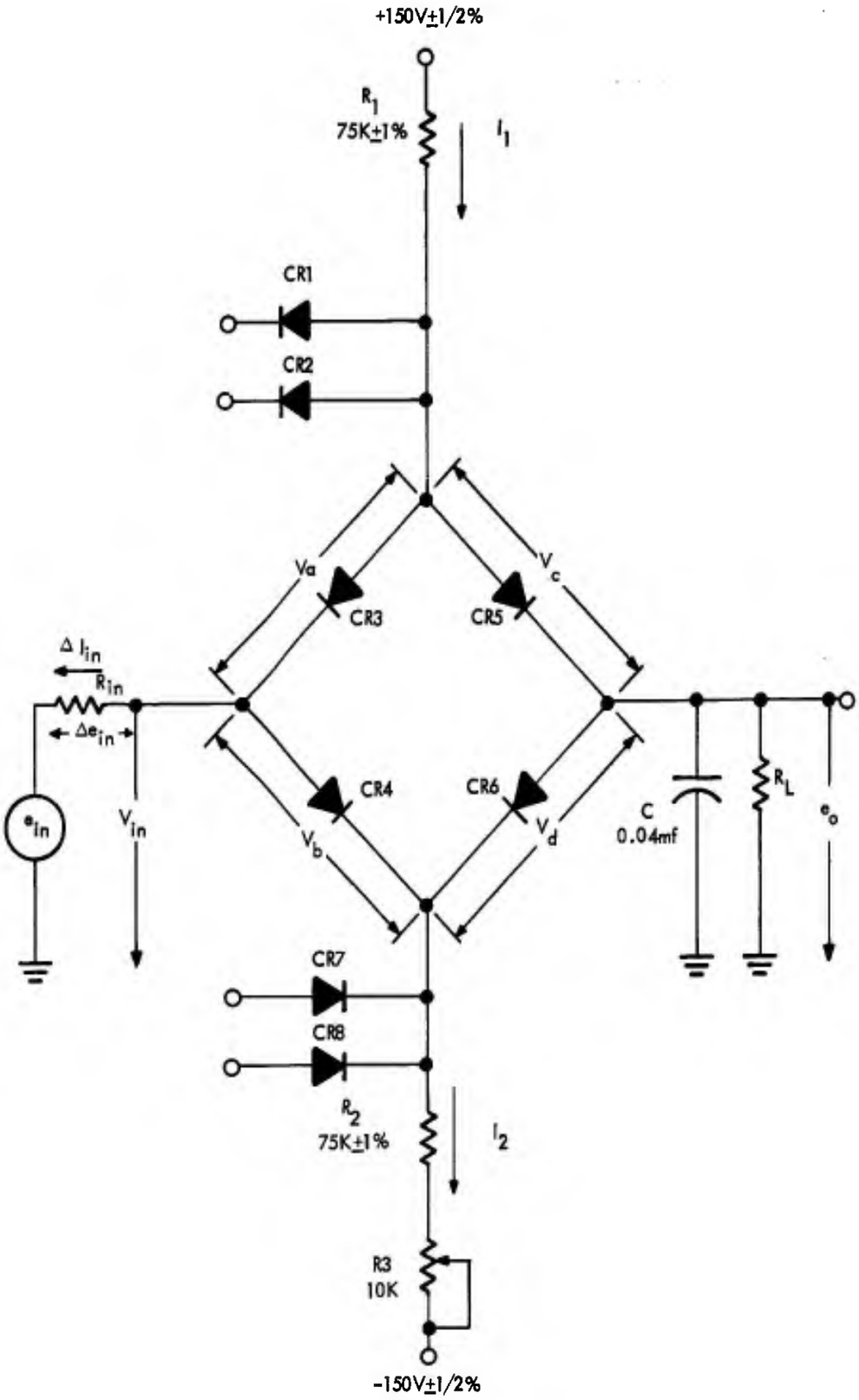


Figure 29. Schematic of Multiplexer Bridge Circuit

The change in charge on the output storage capacitor, or droop, must be less than one percent during the 50-millisecond discharge period.

The drift in the output must not exceed one percent over an eight-hour period.

a. Current Unbalance. Currents I_1 and I_2 are initially equalized by means of R_3 (figure 29) for $e_{in} = e_{io}$, where e_{io} is any particular voltage within the voltage range of e_{in}. Any other input causes an unbalance in the currents that flow in the input circuit, resulting in a ΔI_{in}.

Flow of the current ΔI_{in} will create a linear inaccuracy of

$$\Delta e_o \approx \Delta e_{in} \approx 2R_{in}(e_{io} - e_{in})/R_1 \qquad (1)$$

assuming the transformation from V_{in} to e_o is linear, and $R_1 \approx R_2 + R_3 \ll$ back resistance of the two gating diodes, CR_1 and CR_2.

As an example,

$$0 \text{ volts} \leq e_{in} \leq +10 \text{ volts}$$

$R_{in} \approx 100$ ohms (output impedance of d-a converter cathode follower)

$$R_1 = 75K$$

and let e_{io}, the voltage for e_{in} at the time of calibration, be 0 volts. Then when $e_{in} = e_{in/max} = 10.0$ volts

$$\Delta e_o \approx \Delta e_{in} \approx \frac{(2)(100)(0 - 10)}{75 \times 10^3}$$

$$\Delta e_o \approx -25 \text{ mv}$$

or a maximum linear error term of approximately +0.25%.

Since Δe_o is a linear function of R_{in}, the error can be reduced by minimizing the output impedance of the d-a converter cathode follower. The output impedance of 100 Ω which has been postulated is readily attainable if a tube with a transconductance of 10,000 micromhos or higher is used. Since the JW5847 vacuum tube has an average transconductance of 13,500 micromhos, a minimum output impedance was attained.

b. Capacitor Discharge — A serious cause of output droop is the discharge (or charge) of the capacitor through the silicon diodes when the multiplexer is OFF. The discharge (or charge) path consists of a resistive component and a constant-current component. The former causes only a linear error; the latter is the result of leakage current in the diodes. The leakage current will be opposite in diodes CR5 and CR6; it is the difference between them which causes the discharge (or charge) of the capacitor. Tests performed on these diodes (1N138B) indicated that the leakage currents ranged from 0.001 to 0.008 microamperes with 20 volts across the diode in the reverse direction. Thus, the average difference was less than 0.007 microamperes, and the droop for a 0.04 μf capacitor with the current flowing for 50 msec would be less than 10 millivolts.

The linear inaccuracy arising from the change in charge on the capacitor, due to the loading of the storage capacitor C by the load R_L (see figure 29), is described by

$$\Delta e_o = e_o\left(1 - \epsilon^{-t/R_L C}\right) \tag{2}$$

From this expression, Δe_o will be minimal when R_L is very large. The resistance R_L is the leakage of the capacitor and the input impedance of the load. The use of high quality capacitors will maximize leakage resistance, and the use of a cathode follower with extremely good grid current characteristics will maximize the load impedance. A cathode follower has good grid characteristics when it draws a minimum amount of positive grid current over the range of the input. Experimentation determined that the inverse grid current drawn by the cathode follower was more troublesome than the positive grid current.

Using the expression

$$\Delta v_c = \frac{I}{C}\,\Delta_t \tag{3}$$

where:

Δv_c = change in voltage of storage capacitor

C = capacity of storage capacitor = 0.04 μf

I = inverse grid current of the cathode follower

Δt = time interval in which storage capacitor voltage will change; = 50 msec.

yields:

$$\frac{I}{\Delta v_c} = \frac{C}{\Delta t} = \frac{0.04 \times 10^{-6}}{50 \times 10^{-3}} = 8 \times 10^{-7} \text{ amps/volt} = 8 \times 10^{-10} \text{ amps/millivolt} \tag{4}$$

Thus a tube with an inverse grid current of 8×10^{-9} amperes flowing for 50 msec will cause the storage capacitor to charge by 10 millivolts. Ordinary tubes such as the 5814, 5965, 2C51, etc. can be operated with inverse grid currents of 1×10^{-7} to 1×10^{-8} amperes. The JW5847, on the other hand, can be operated with inverse grid current approaching 1×10^{-10} amperes. The amount of capacitor discharge or droop in the D.C. voltage that the servos can tolerate was determined by experiment to be approximately 30 millivolts; thus it is certain that the servos would not function properly with the hundreds of millivolts of droop that would result from using ordinary vacuum tubes. Thus the JW5847 was selected as the one tube, other than an electrometer tube, that would satisfy the droop or discharge requirements. An electrometer was not selected because, although it possesses favorable grid characteristics, it cannot provide the signal power required to drive the servos. An additional disadvantage to an electrometer tube is that it requires closely regulated low voltage D.C. for the filament.

c. Diode Unbalance. The most serious factor contributing to an offset between the input and the output of the multiplexer is diode unbalance.

The diodes are checked for forward voltage drop when the current through them is one milliampere, and then ranked in order of voltage drop. Any group of four successively ranked diodes may be used to make up one multiplexer circuit, provided that the overall

spread of drops is no greater than 20 millivolts, and no difference between two adjacent drops is greater than 10 millivolts. It proved best to use successively ranked diodes for CR3, CR5, CR4, and CR6, in that order. The resulting offset will be limited to less than 10 millivolts. In an actual test, V_a was 610 millivolts; V_b, 618 millivolts; V_c, 622 millivolts; V_d, 630 millivolts. The difference, $|e_o - V_{in}|$, was 4.7 millivolts, with the diodes arranged as suggested.

d. Power Supply Drifts. The third cause of offset is small shifts, in the same direction, in the +150 and the -150 volt power supplies. This offset is described by:

$$\Delta e_o = (\Delta V_1 + \Delta V_2)(R_{in}/R_1) \qquad (5)$$

For an R_{in} of 100 ohms and a supply unbalance of 1/2%, the offset will be 2 millivolts.

The drift requirement for the analog outputs was specified to be less than one percent, or 100 millivolts. As mentioned previously, the converter is cathode-follower-coupled to the multiplexer, and the multiplexer is cathode-follower-coupled to the external load. Each cathode follower will introduce drift equivalent to approximately $1/\mu$ of the drift in the power supplies from which it draws current. The cathode followers are supplied from the +150 volt and the -150 volt supplies, which are regulated to one half of one percent; 60 millivolts of drift can be introduced by the cathode followers alone. Additional drift from the power supplies is introduced through the converter, the multiplexer, and the level adjustment of the storage capacitor output cathode follower.

Intolerable drift in the output, due to power supply drift, was eliminated in the overall system by supplying the answer potentiometers of the positioning servos with reference voltages derived from the converter supply voltage, in exactly the same manner in which the analog output signal is derived (figure 30). As a result, this form of drift will not affect the error signal, which is the difference between the input voltage applied to the chopper-amplifier and the voltage generated by the answer pot of the positioning servo, and which causes rotation of the servo.

The two reference voltage generators can supply a number of answer pots, the number depending upon the resistance of the potentiometers. It was determined that at least five answer pots can be supplied from one pair of reference generators. With approximately 20 answer pots requiring reference supplies in the system, a total of five pairs of reference voltage generators was sufficient.

To compensate further for drift, it was decided that two cathode followers in series would be used to read out from the storage capacitor. The combination of the two would provide the effective high input impedance required by the capacitor, and would allow the incorporation of a gain and a level adjustment to compensate for emission variation in the vacuum tubes. The gain and level adjustments would also permit matching all outputs, even though the cathode follower tubes do not have identical characteristics.

e. Vacuum Tube Drift. The one source of drift which appeared only after considerable testing time had been accumulated was the random average velocity change, present in all high-vacuum electron tubes. This drift is difficult to describe, since it varies considerably from tube to tube, and sometimes is not present over a long interval of time. Measurement made on a sample of new tubes showed a variation of several hundred millivolts (figure 31). As the tube aged, however, the variations become less eratic (figure 32). Although the average drift of the older tubes over a 200-hour interval

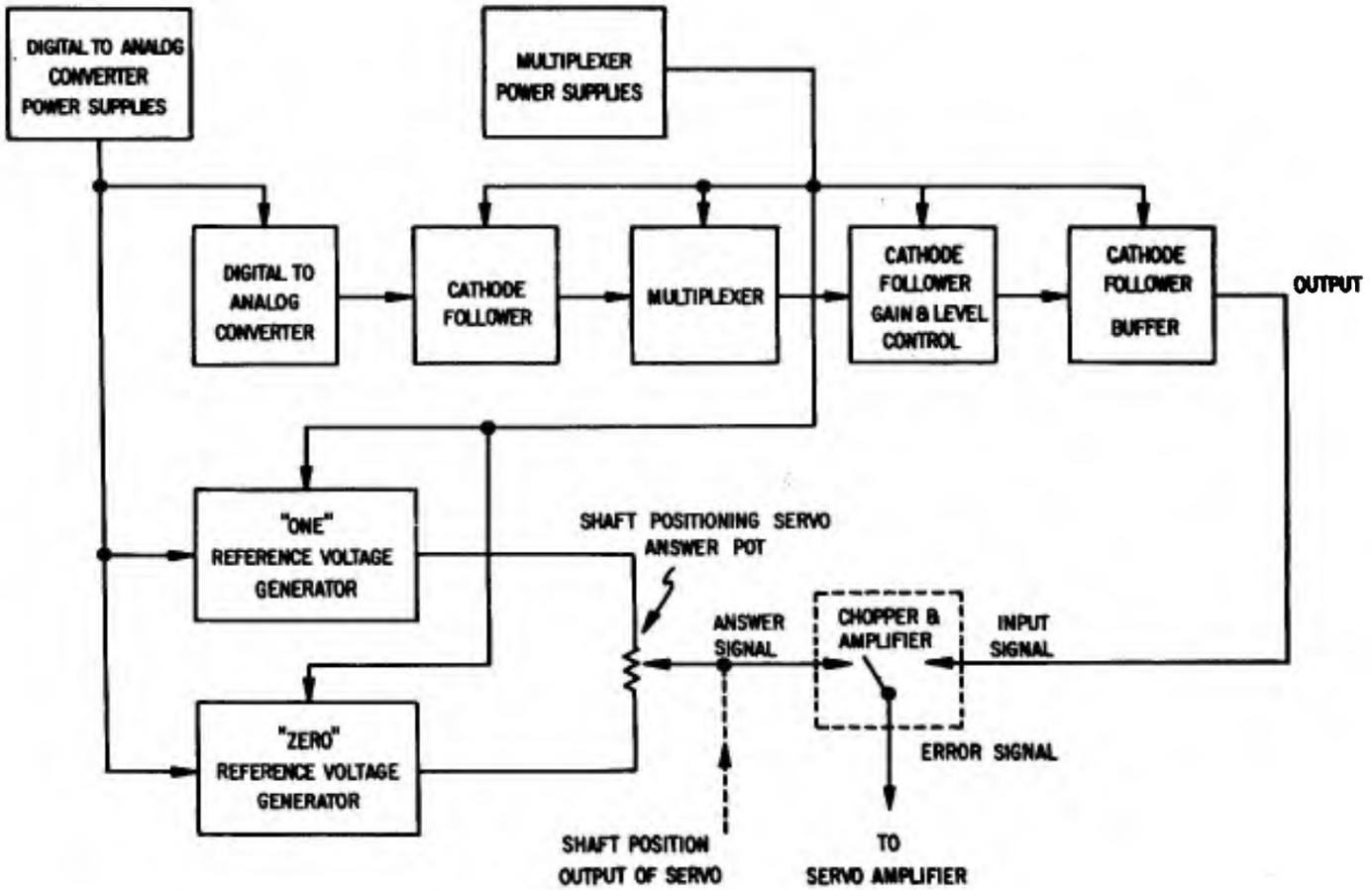


Figure 30. System for Eliminating Effect of Power Supply Drift

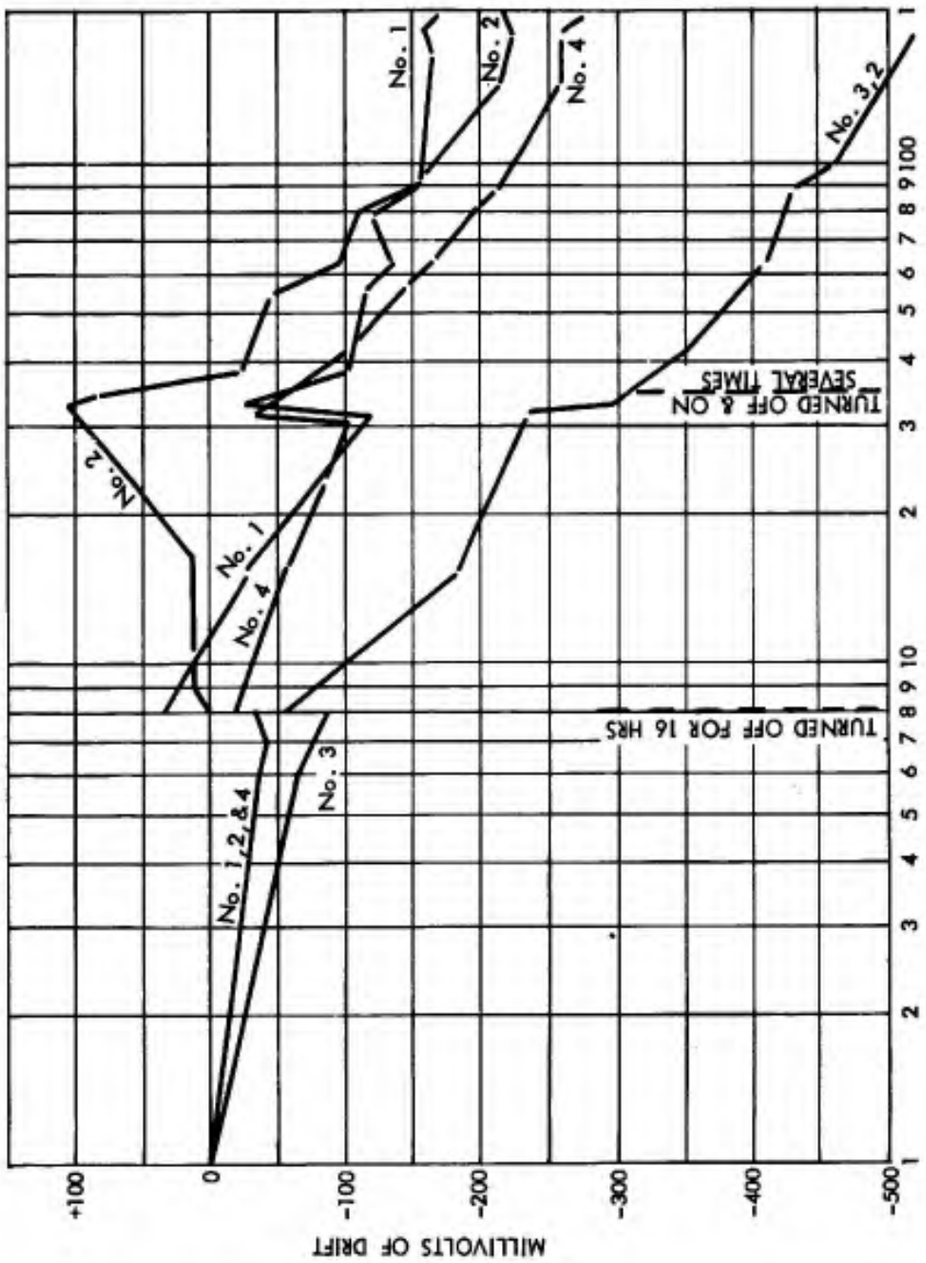


Figure 31. Drift Characteristics of New JW5847 Tubes

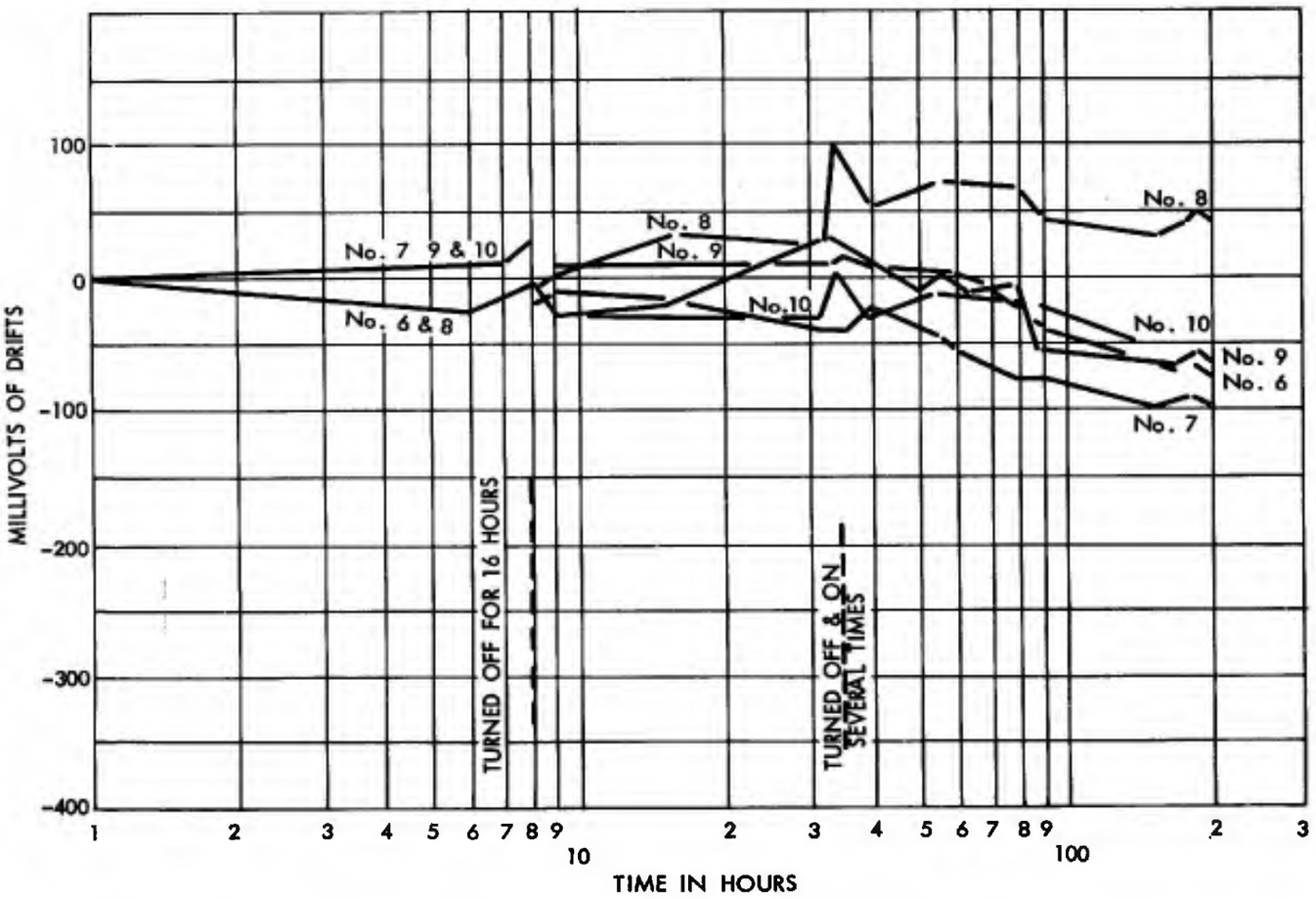


Figure 32. Drift Characteristics of Aged JW5847 Tubes

can be expected to be less than one millivolt per hour, the short term drift is variable; thus it is virtually impossible to guarantee less than 20 millivolts (two tenths of one percent) drift over any reasonable period of time.

f. Final Circuit Configuration. A complete multiplexer circuit is illustrated in figure 33. A brief description of its operation follows.

When CR1 and CR2 are at +20 volts and CR7 and CR8 are at -4.5 volts, the capacitor C charges to the voltage at point A. When diodes CR1 and CR2 are at -4.5 volts and CR7 and CR8 are at +20 volts, the voltage on C will change at a rate determined by the difference in the reverse currents of CR5 and CR6 and by the grid current of vacuum tube V1. The 10-K adjustable resistor previously in series with resistor R2 was eliminated because the shift in level for which it compensated can be adjusted in the cathode follower. The main requirements of the bridge portion of the circuit are that diodes CR5 and CR6 have less than 10 millimicroamperes of reverse current at 20 volts and that CR5 and CR6 exhibit fast recovery, less than 1 microsecond, or be matched for recovery time. Tube V1 is a triode-connected JW5847 and was selected because of its low reverse grid current, less than 1 millimicroampere. Resistor R6, in the cathode of V1, provides a 6-volt level adjustment and is necessary for the adjusiment of all outputs to the same zero level. Resistor R8 provides the necessary gain adjustment to bring all channels to the same gain. The zener diode CR9 across R6 reduces by a factor of five the gain change caused by varying R6. This reduction is necessary since the gain adjustment causes a level shift; without the diode, the interaction of the two adjustments would have made the procedure a time-consuming process. With the diode, the level adjustment alone may be used to compensate for tube drift, once the gain has been set. The particular zener diode used is a 1N429. This diode has a temperature coefficient of 0.0006 percent per degree C; ordinary zener diodes have temperature coefficients ranging up to 0.05 percent and would introduce a drift due to temperature outweighing their usefulness in reducing the gain change caused by level adjustment. The two-section, resistor-capacitor, low pass filter (160K resistors and 0.1 μmfd capacitors) was added to filter the multiplexed signal. Experimentation showed that a good instrumentation servo would follow the incremental steps in the multiplexed signal. The simple filter has a response which is down 3 db at 4 cps, 14 db at 10 cps, and 18 db at 20 cps. Tube V2A is used to provide a low impedance output to the instrumentation circuits, and diode CR10 limits the output voltage in the event of tube failure.

4.3.2.2 Static Flip-Flop

The static flip-flop is used in both the Memory Unit and the Input-Output Unit. Since the original design was intended for use in the multiplexer address register and in the analog-to-digital converter storage register, it is being discussed as a circuit pertinent to the input-output circuitry.

The schematic diagram of the static flip-flop breadboarded initially for evaluation was obtained from the MSEE (figure 34). Considerable effort was expended evaluating this flip-flop and several disadvantages of the design were indicated. Since the requirements imposed upon the flip-flop had meanwhile changed as a result of the development of the memory, a new design possessing characteristics needed for UDOFT was adopted (figure 35). The problems associated with the original static flip-flop were complementing, marginal screen grid operation, slow rise and fall times, and poor drive capability of the output cathode followers.

a. Complementing. The flip-flop was found to complement on positive input pulses, due to the reverse recovery time of the input diode. When a positive pulse was applied to the input of the conducting half of the flip-flop, a large grid current was drawn through the diode. When the pulse ended, the grid was forced

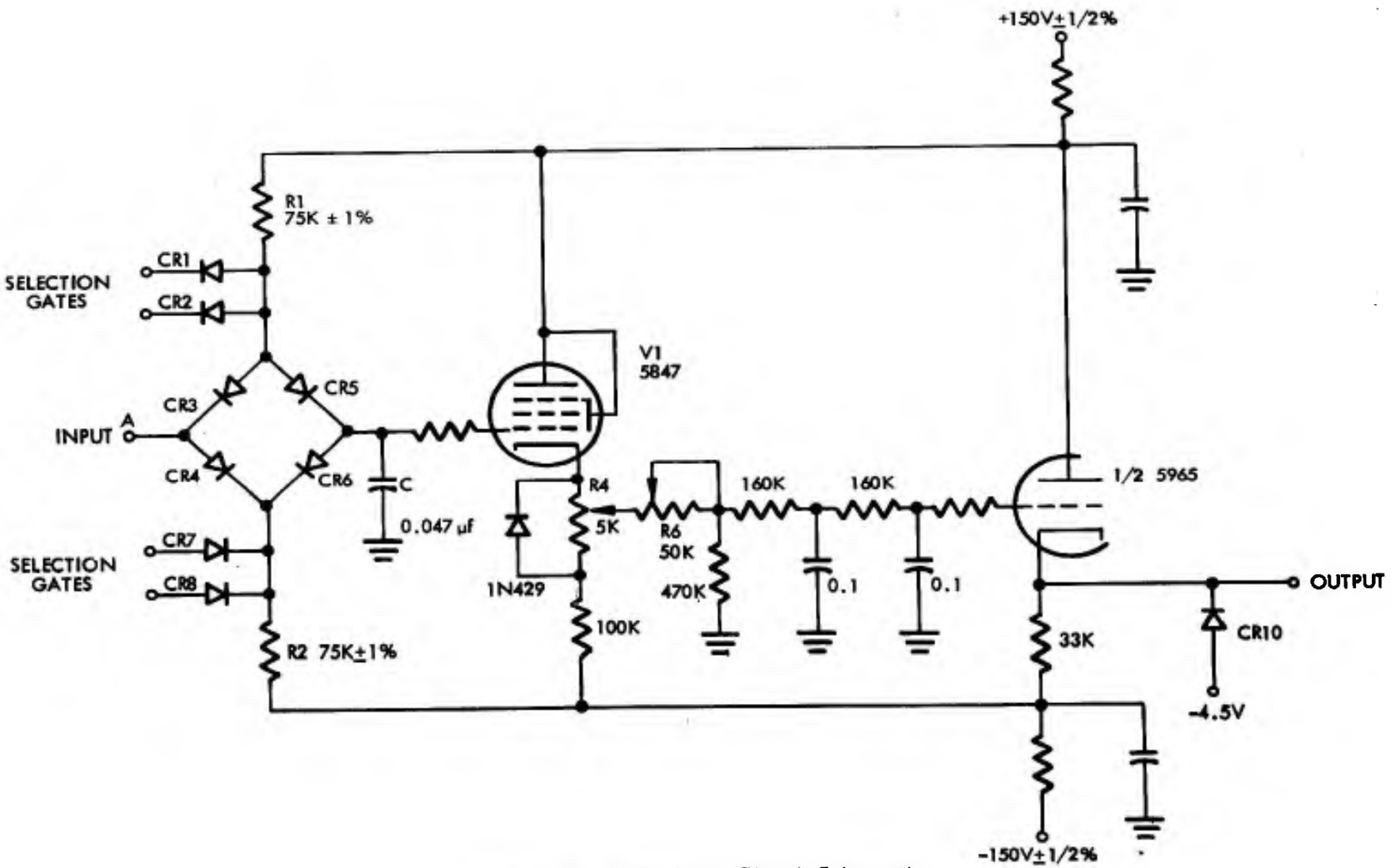


Figure 33. Multiplexer Circuit Schematic

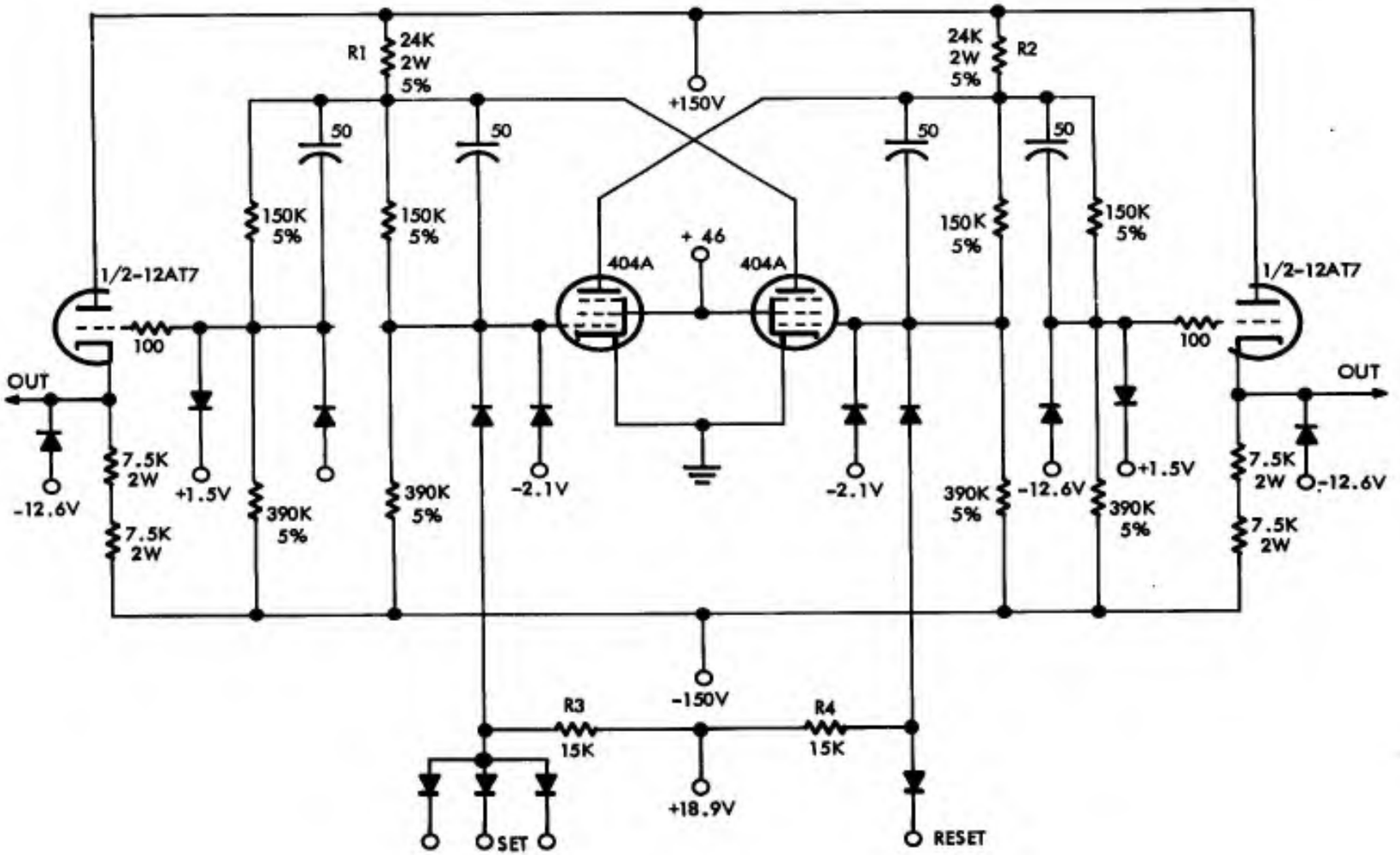


Figure 34. Schematic of Moore School Static Flip-Flop

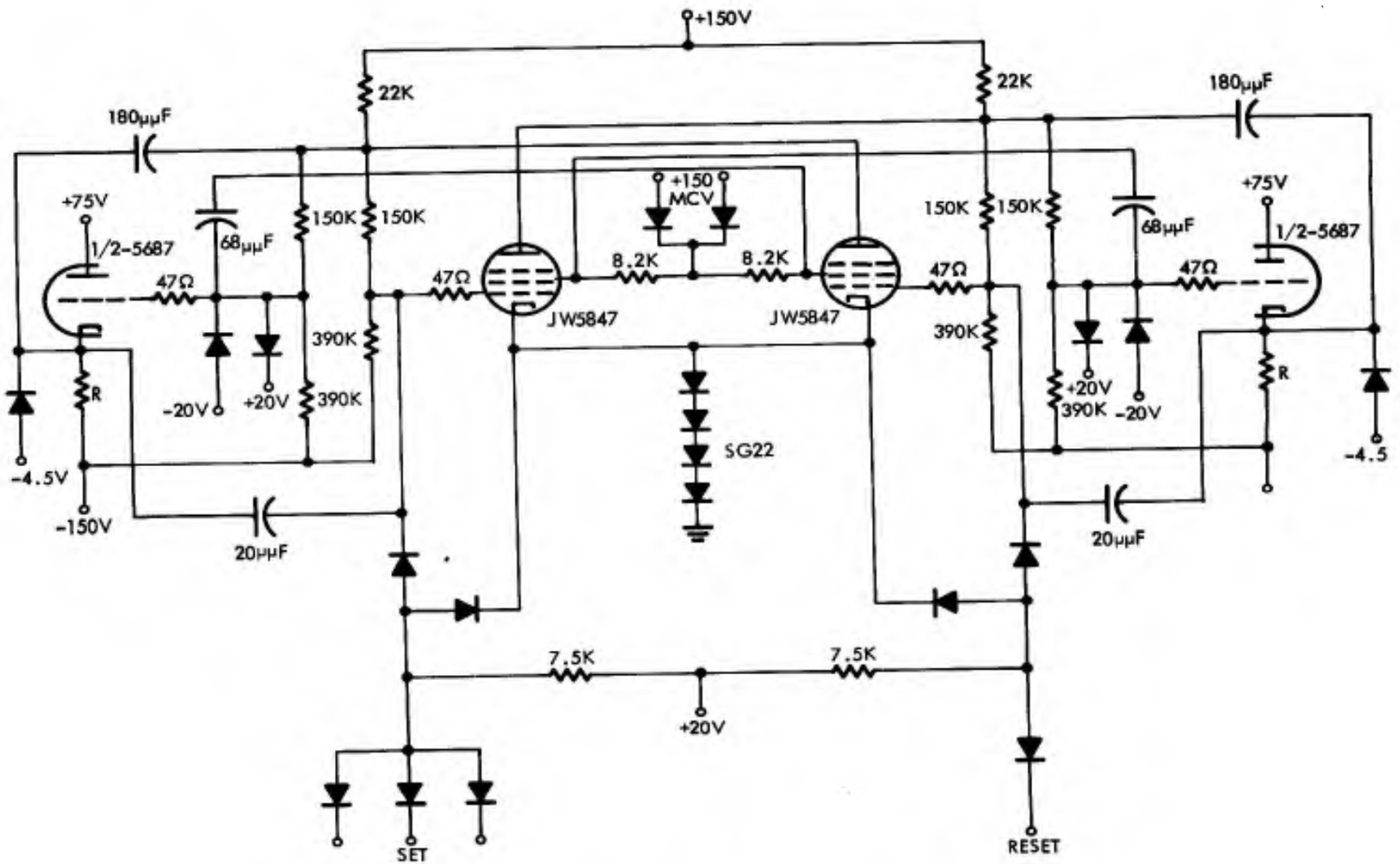


Figure 35. Schematic of Redesigned Static Flip-Flop

negative by the diode until the diode had recovered. The effect was essentially that of a capacitor across the diode. The flip-flop then triggered on a small negative signal at the ON grid, due to the capacitive coupling from the ON plate to the OFF grid, and triggered readily since the OFF tube was not completely off, being clamped to only -2.1 volts.

Revision of the flip-flop eliminated this problem, since the AND gate inputs to the flip-flop are clamped by diodes to the cathodes of the flip-flop tubes. This arrangement also limits the grid current.

b. Screen Grid Operation. It was considered bad practice to have the screen grids of the flip-flop tubes connected to +46 volts. A small sample of tubes measured under this condition indicated dissipations ranging from 65 to 80 percent of maximum rating.

Operation of the screen grid was improved by returning it to +150 volts through a selected resistor with value such that the screen grid dissipates approximately 0.6 watts. As a result, the maximum dissipation cannot be exceeded by any tube. The revised design also uses the screen to provide dynamic coupling to the grid of the cathode follower, to improve the output rise and fall times.

c. Rise and Fall Times. The rise time for a 12-volt output signal was measured to be 0.25 μsec; a 25-volt output signal, which was now needed, would have a rise time of 0.50 μsec. This rise time did not satisfy the UDOFT requirements. The fall time in the original circuit is determined by the cathode follower, the cathode resistance and the capacitive loading. Again a fast fall time was necessary, particularly for generating the read and write signals in the memory.

In the revised circuit, the rise time is determined by the screen resistance and the input capacity of the cathode follower. This occurs because the coupling to the grids of the flip-flop is taken from the output of the cathode follower.

The fall time of the positive pulse was shortened and made more independent of capacitive loading by capacitively coupling the flip-flop plates to the respective cathode follower grids.

d. Cathode Follower. It was discovered that the cathode follower output was incapable of accepting or supplying the current required for either the digital-to-analog converter or the multiplexer. To relieve this situation the cathode follower was changed from a 12AT7 to a 5687. In order to limit the output signal at +20 volts, the grid is clamped to +20 volts. The output is clamped to -4.5 volts to limit the output signal at -4.5 volts.

e. Final Specifications. The specifications of the redesigned flip-flop are:

Trigger Amplitude	4 volts (-3 volts to +1 volts)
Output Signals	-4.5 volts to +20 volts
Rise Time	120 nanoseconds
Fall Time	50 nanoseconds (unloaded) 200 nanoseconds (with 100 picofarads of loading)
Delay	100 nanoseconds
Load Capabilities	-12 ma at -4.5 volts 20 ma at +20 volts

4.4 Logic Circuit Packaging

In addition to conceiving the logic design of the UDOFT computer, MSEE studied circuit packaging methods best suited to space limitations, the logic layout, and the logic circuits. These consisted of packaging the basic pulse amplifier with various two-level AND-OR gate input configurations, packaging OR gates with various input configurations, and packaging delay lines of diverse delays to be cascaded for additional delay times.

Sylvania conducted an analysis of the proposed packaging techniques developed by MSEE in an attempt to consolidate the proposed package types into fewer and more flexible packages. The following sections discuss the results of the analysis.

4.4.1 Pulse Amplifier Plug-in Package Assemblies

The first column of Table VII indicates the seven originally proposed pulse amplifier packages with the AND-OR input configurations of each; the second column indicates the five-pulse amplifier packages that were finally used.

TABLE VII

COMPARISON OF PROPOSED AND FINAL
PULSE AMPLIFIER PACKAGE CONFIGURATION

Proposed	Final (figure 36)
Type A (2)	Type 1 (2)
Type B (5)	Type 2 (6 + 3)
Type C (3 + 2)	
Type D (6 + 3)	
Type E (4 + 3 + 3)	Type 3 (4 + 2 + 2 + 2)
Type F (4 + 4 + 3)	Type 4 (5 + 5 + 4 + 3)
Type G (5 + 5 + 5 + 4)	Type 5 (5 + 4 + 3 + 2)

Pulse amplifiers Types 1 through 4 are capable of producing the three output pulses: clamped positive, unclamped positive, and negative. Pulse amplifier Type 5 is capable of producing the three output pulses: clamped positive, unclamped positive, and double amplitude positive. Pulse amplifier Type 5 is used only in the eleven cases where drive is required for long delay lines.

The following are the explicit reasons for the modified pulse amplifier package configurations:

a. To reduce the number of package types, in order to minimize the package replacement problem, parts identification, pre-fabrication preparation, and fabrication time.

b. 1. To allow the use of a 32-pin printed circuit connector in order to minimize card breadth and allow better space utilization

2. To decrease the cost from that of a 36-pin connector which would have been necessary had the proposed G-type pulse amplifier package been used

4.4.2 OR Gate Plug-in Package Assemblies

The first column of Table VIII indicates the five originally proposed basic OR Gate packages with the input configuration of each; the second column indicates the four basic packages that replaced them.

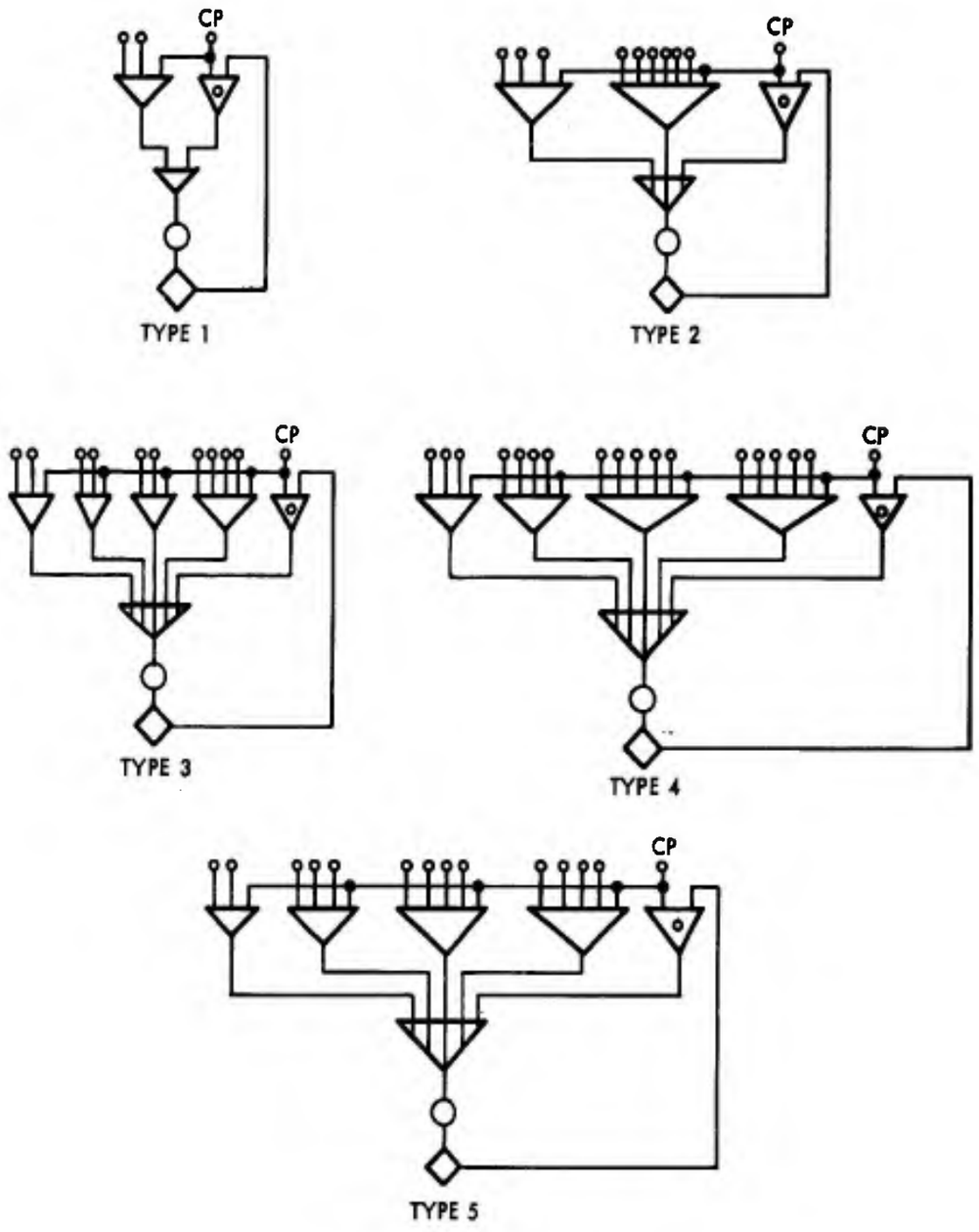


Figure 36. Logic Diagrams of Five Pulse Amplifier Package Types

TABLE VIII

COMPARISON OF PROPOSED AND FINAL
OR GATE PACKAGE CONFIGURATIONS

Proposed	Final (figure 37)
Type A (5) Type B (2) Type C (2 - 2 - 2)	Type 1 (2 - 2 - 2 - 2)
Type D (5 - 5 - 5 - 5)	Type 4 (5 - 5 - 5 - 5)
Type E (2 - 2 - 2 - 2 - 2)	Type 2 (2 - 2 - 2 - 2 - 2 - 2 - 2 - 2)
	Type 3 (2 - 2 - 3 - 5 - 5)

OR Gate Type 3 is used in those places where Type 2 is inadequate (number of inputs per gate), or Type 4 is inadequate (number of gates).

The primary reason for adopting these four configurations was to allow a more economical use of the diodes, as indicated by the flexibility of the logic. (figure 37.)

4. 4. 3 Delay Line Plug-in Package Assemblies

The first column of Table IX indicates the four originally proposed basic delay line packages; the second column indicates the five basic packages that replaced them.

TABLE IX

COMPARISON OF PROPOSED AND FINAL
DELAY LINE PACKAGE CONFIGURATION

Proposed	Final (figure 38)
Positive Delay, Single Input (short/long)	Type DP-1 Delay Line-Positive (Short only)
	Type DP-2 Delay Line-Long (5. 6)
	Type DP-3 P235 Delay Line-Long (5. 0)
	Type DP-4 Delay Line-Long (3. 0 and 2. 0)
Positive Delay, Multiple Input	Type DP-1 Delay Line-Positive and Type 1 OR Gate
Negative Delay, Single Input	Type DN-1 Delay Line-Negative
Negative Delay, Multiple Input	Type DN-1

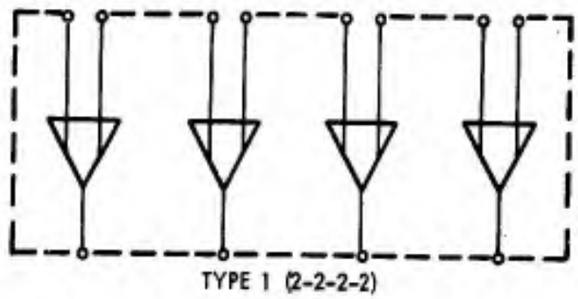
Due to the different physical requirements of the long delay lines, it was not possible to utilize the same basic configuration used for the short positive delay lines. Since there are only eleven long delay lines in the system, the exception is minor.

The following are the explicit reasons for the modified delay line configurations:

a. To increase the flexibility of the package by allowing the package to accept a number of different delays.

b. To make all short positive delay line packages and all short negative delay line packages identical, resulting in "location insensitive" delay line packages; i. e., any such package may be used in any delay line location in the computer without considering the values required at that location.

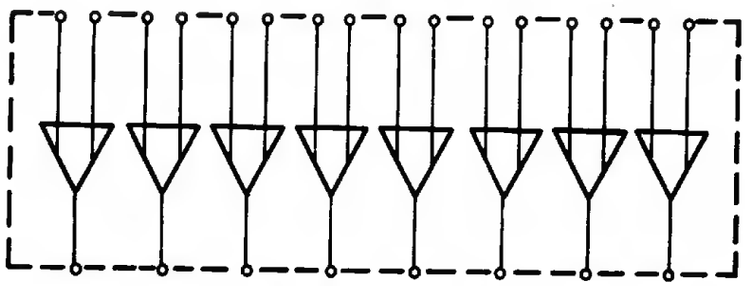
c. To reduce the cost of delay line fabrication by accepting a large number of one type.



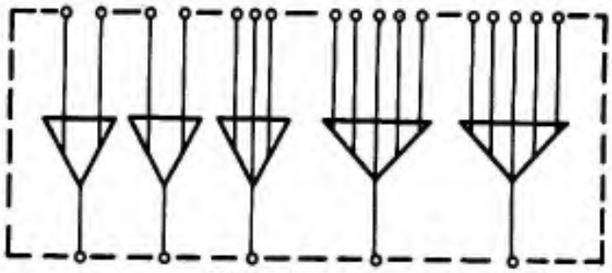
TYPE 1 (2-2-2-2)

OTHER POSSIBLE CONFIGURATIONS

2-2-4
4-4
2-6

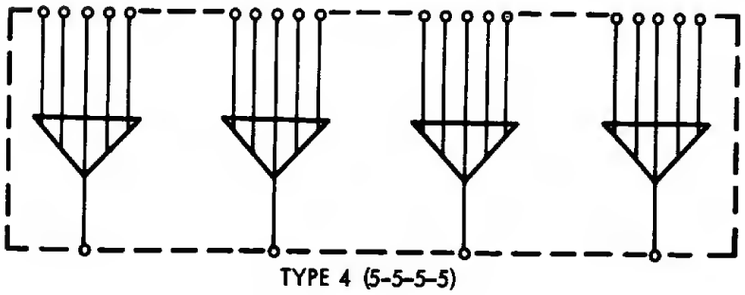


TYPE 2 (2-2-2-2-2-2-2-2)



TYPE 3 (2-2-3-5-5)

2-2-2-2-9
2-2-2-5-6
2-2-2-3-8



TYPE 4 (5-5-5-5)

2-2-8-8
2-5-5-8

Figure 37. Logic Diagrams of Four OR-Gate Package Types

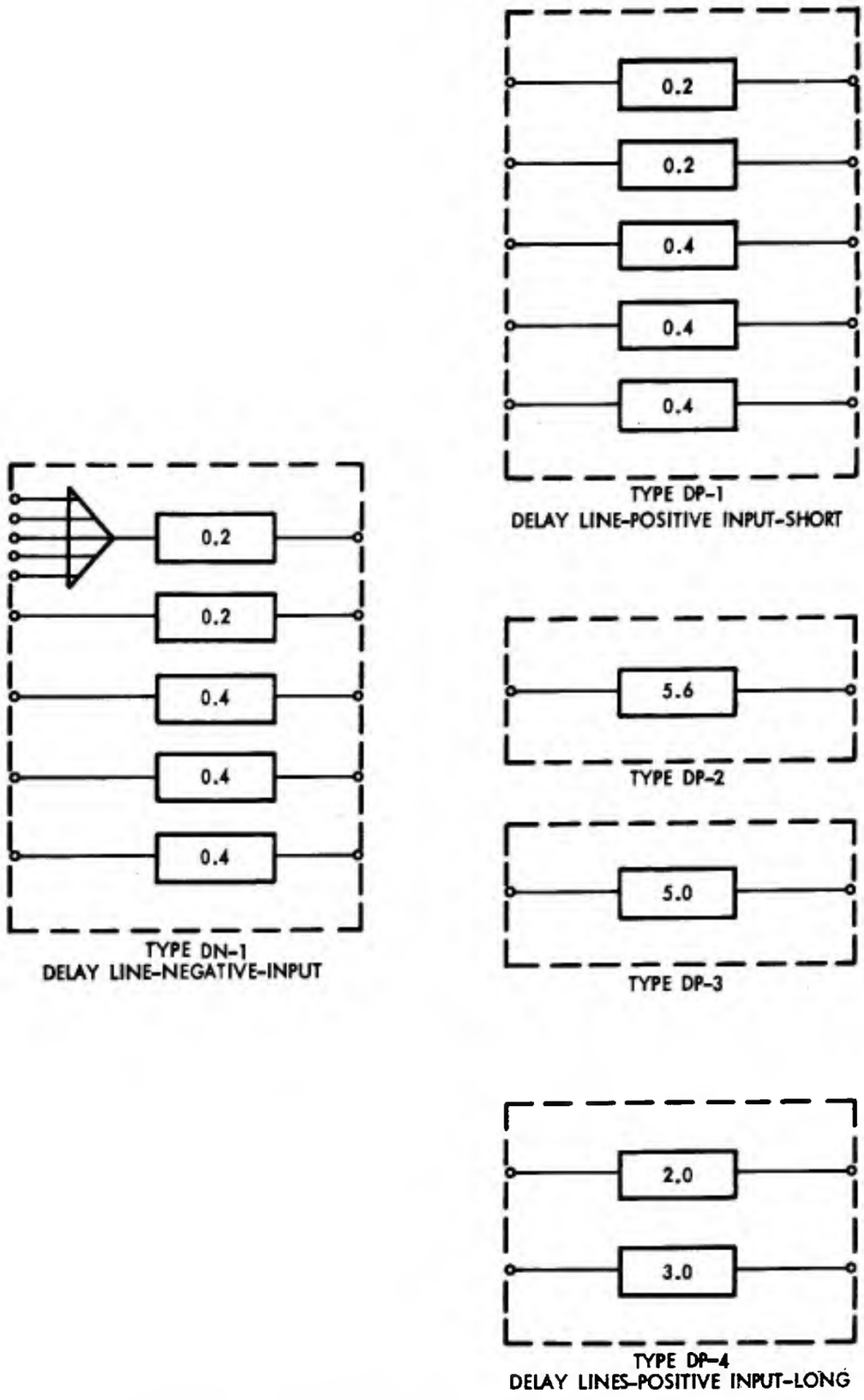


Figure 38. Logic Diagrams of Six Delay-Line Package Types

d. To take advantage of the 32-pin connector required by the pulse amplifier packages, and to pack maximum delay into one delay line package (thereby arriving at the delays of 0. 2 - 0. 2 - 0. 2 - 0. 4 - 0. 4 pulse periods of delay per assembly).

4. 4. 4 Plug-in Package Fabrication Problems

As is typical in most computers, the main frame circuit packaging technique establishes the circuit packaging plan for the rest of the computer system. A typical UDOFT main frame plug-in package assembly is shown in figure 39.

Two major problems were encountered during the fabrication of the printed circuit plug-in assemblies; board warp and copper delamination. The board warp occurred because the cloth-base phenolic material yielded under the high temperatures encountered and the forces exerted on the large board during the dip-soldering operation. By changing to glass-base epoxy material, by improving the jigging of the card for dip-soldering, and by improving the solder dipping techniques, the problem of board warpage was solved. Copper delamination, or the parting of the copper from the base material, was the second problem. When the change to the glass base epoxy was made, the problem all but disappeared, leaving only minor delaminations which occurred from caustic cleaning agents used in preparing the etched copper tabs for plating.

4. 4. 5 Classification of Printed Circuit Plug-in Package Assemblies

Almost all the digital circuitry of the UDOFT Computer is mounted on printed circuit plug-in packages, a design concept with many obvious advantages including economy of fabrication and ease of maintenance. Table X, a listing of the package types, indicates the number of each type used in the various units that comprise the computer.

4. 5 Main Frame Development

4. 5. 1 Main Frame Cabinets

The three cabinets of the main frame are identical in size, shape, and construction. Each cabinet is composed of four bays (figure 40), and each bay contains a package rack assembly accommodating ten rows of twelve packages each. This provides a maximum capacity of 120 packages per bay or 480 per cabinet.

The interconnect section is in the uppermost portion of the cabinet; it is the junction point for all power and signal cables, except coaxial cables, entering and leaving the cabinet. Barrier strips, power circuit breakers, fuses, and the marginal check voltage control chassis are located in this section of the cabinet.

Above the interconnect section is the removable blower section. This area houses the blowers which supply cooling air.

4. 5. 2 Package Racks

The package racks are structural assemblies that provide mechanical support and cooling air for the printed circuit plug-in packages, and sustain the printed-circuit connectors to which all electrical connections are made (see figure 41). The three sections of a rack are the vertical air duct, the horizontal shelves, and the connector panels.

The vertical air duct channels the cooling air from the blower into the hollow shelves. The shelf is a prime functional part of the rack structure; it is hollow, and forms the final section of the ducting system for directing air to the individual packages. Holes through the upper surface of the shelf direct cooling air to the packages. The shelf carries a slot which guides the package over its entire length during insertion and withdrawal. A metal block in this guide slot, in conjunction with the polarizing feature built into the package, prevents a package from being inserted upside down (figure 42).

The connector panels which hold the plug-in packages are fabricated from high-grade linen phenolic. The openings for the printed circuit connectors are accurately punched in the panels to assure proper alignment of connector and package in their complete assembly.

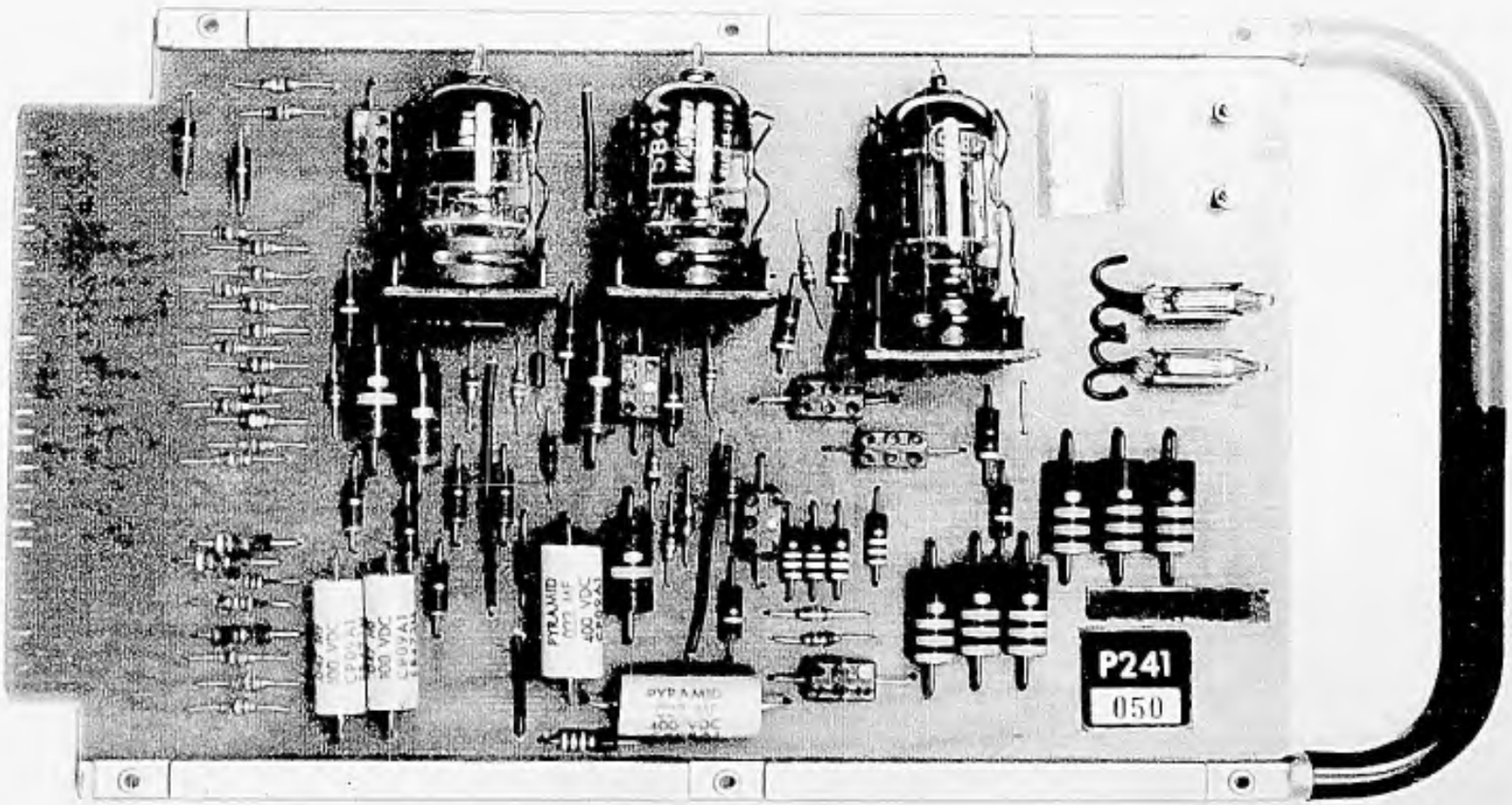


Figure 39. Representative Printed Circuit Plug-in Package

TABLE X

PACKAGE TYPES USED IN THE UDOFT COMPUTER

Package No.	Description	Requirements AU	CI	CII	MEM	I-O	Spare	Total	Handle Color Code Upper	Lower
P211	Pulse Amplifier Type 1	66	63	80		67	38	314	Red	Brown
P212	Pulse Amplifier Type 2	82	86	67		40	27	302	Red	Red
P213	Pulse Amplifier Type 3	7	42	28	24		14	115	Red	Orange
P214	Pulse Amplifier Type 4	83	53	31		63	4	234	Red	Yellow
P215	Pulse Amplifier Type 5	6	3	2			4	15	Red	Green
P221	OR Gate Type 1	15	9	10		2	5	41	Green	Brown
P222	OR Gate Type 2	8		6	2	32	5	53	Green	Red
P223	OR Gate Type 3	1	4	3			2	10	Green	Orange
P224	OR Gate Type 4		1	1		8	2	12	Green	Yellow
P231	Variable Delay	1	1	1	1	1	4	9	Blue	White
P232	Negative Delay	22	33	18		3	10	86	Yellow	Brown
P233	Positive Delay Type 1	100	114	87		34	27	362	Yellow	Red
P234	Positive Delay Type 2	2		2			1	5	Yellow	Orange
P235	Positive Delay Type 3	2					1	3	Yellow	Yellow
P236	Positive Delay Type 4	2	1				0	3	Yellow	Green
P237	Clock Pulse Delay	1	1	1	1	1	2	7	Blue	Green
P241	Static Flip-Flop Type 1				88	28	10	126	Orange	Brown
P242	Static Flip-Flop Type 2				32		5	37	Orange	Red
P243	Signal Driver	12	20	20		1	9	62	Black	Brown
P244	Crutch Card						10	10		
P251	Gate Generator Amp.				32		3	35	Gray	Brown
P252	Array Driver Amp.				32		5	37	Gray	Red
P253	Inhibit Driver Amp.				44		4	48	Gray	Orange
P255	Sense Amplifier				22		4	26	Gray	Yellow
P256	Diode Matrix				8		2	10	Gray	Green
P257	Memory Address Driver				8		2	10	Gray	White
P258	Memory Diode				72		7	79	Gray	Clear
P261	Discrete Output					12	3	15	White	Orange
P262	Digital-Analog Converter					6	2	8	White	Brown
P263	Multiplexer					32	4	36	White	Red
P264	Slow Speed Print					3	1	4	White	White
P265	Multiplexer-Driver					1	1	2	White	Green
P266	Multiplexer Reference					8	1	9	White	Yellow
P271	Clock Oscillator					1	2	3	Blue	Brown
P272	Clock Amplifier	2	2	2	2	2	5	15	Blue	Red
P273	Clock Repeater	30	34	25	18	27	20	154	Blue	Orange
P274	Clock Driver	1	1	1	1	1	2	7	Blue	Yellow
P275	Clock Master					3	1	4	Blue	Clear
P281	-3.0 Volt Sink (VR-1)	4	4	4	2	3	2	19	Brown	Brown
P282	-4.5 Volt Sink (VR-2)	2	2	2	1	2	3	12	Brown	Red
	Totals	449	474	391	390	381	254	2339		
							Less Spares	254		
							TOTAL UDOFT PACKAGES	2085		

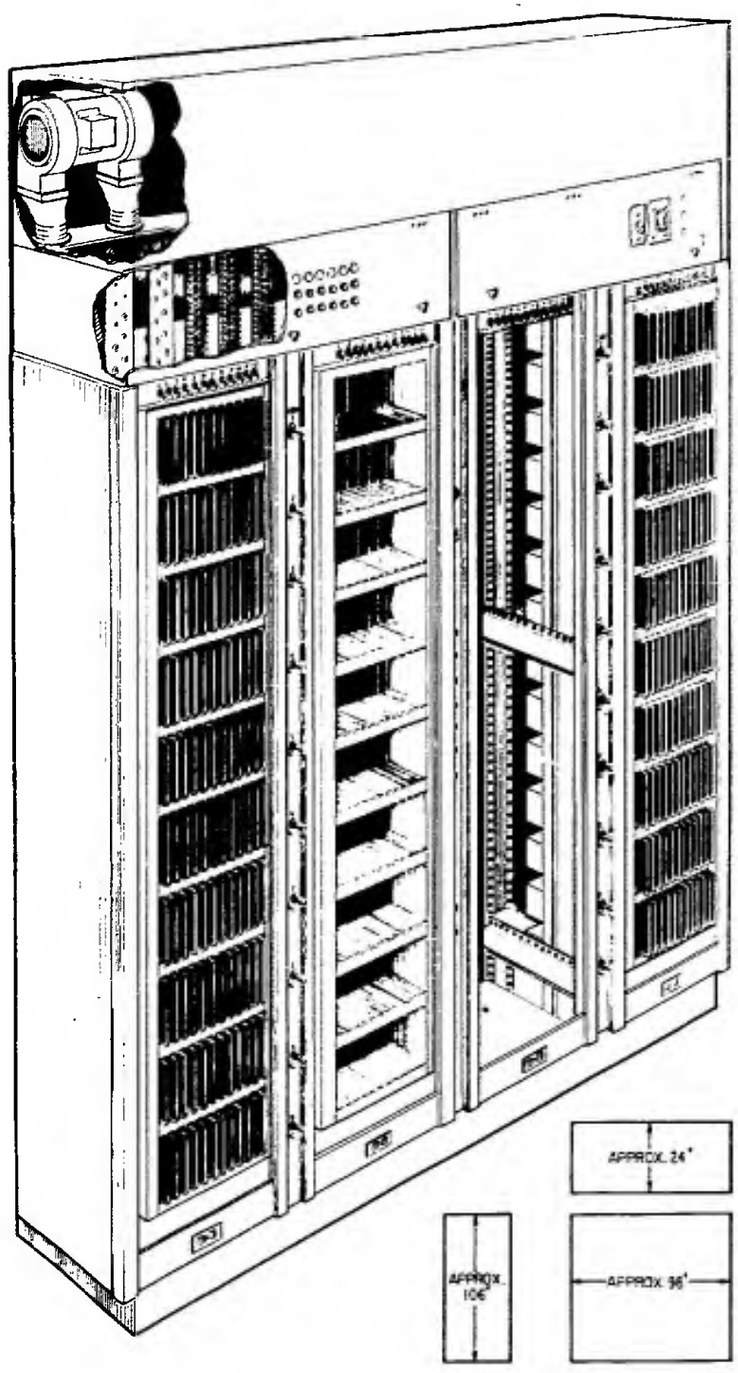


Figure 40. Typical Main Frame Cabinet

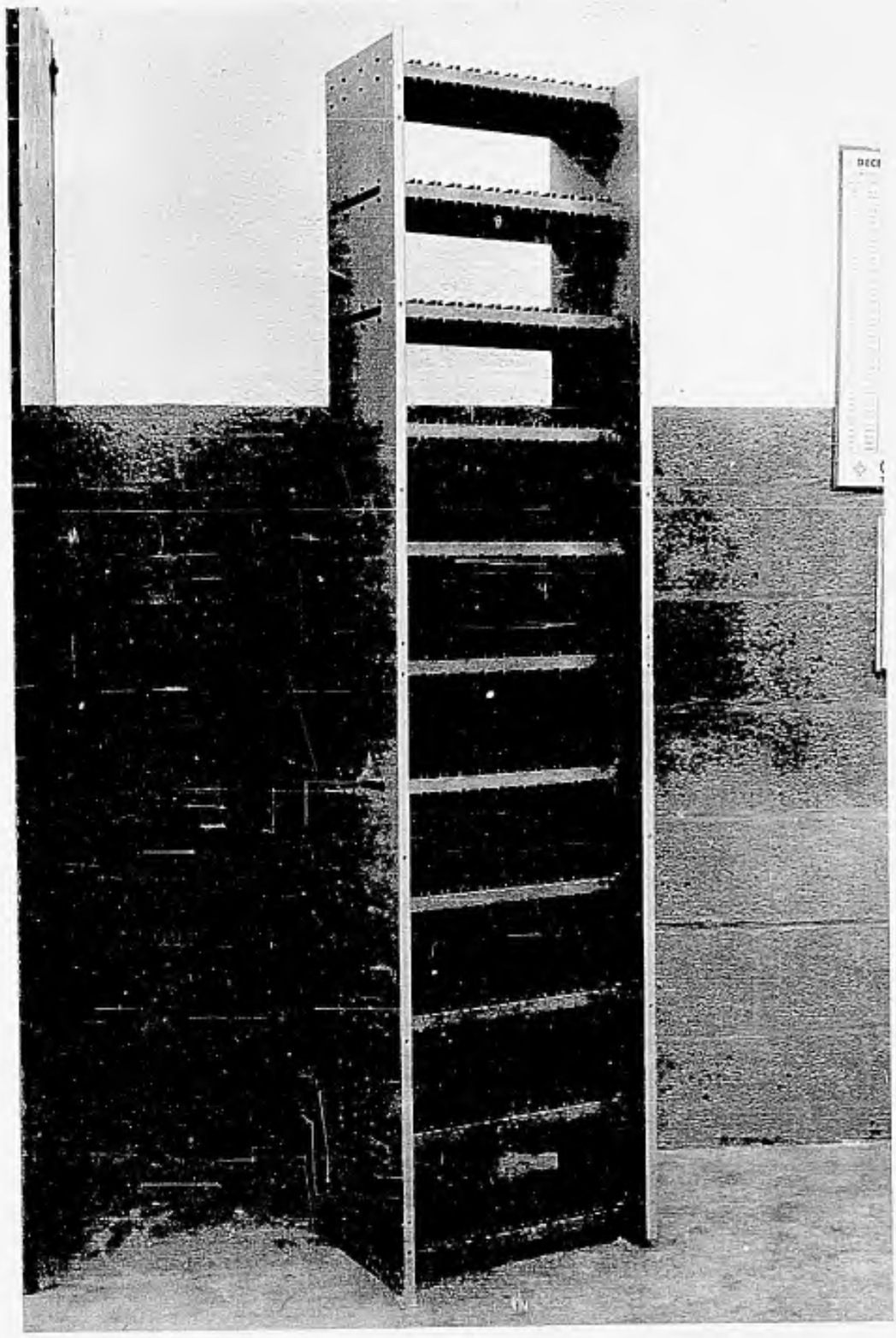


Figure 41. Card Rack with Shelves in Place

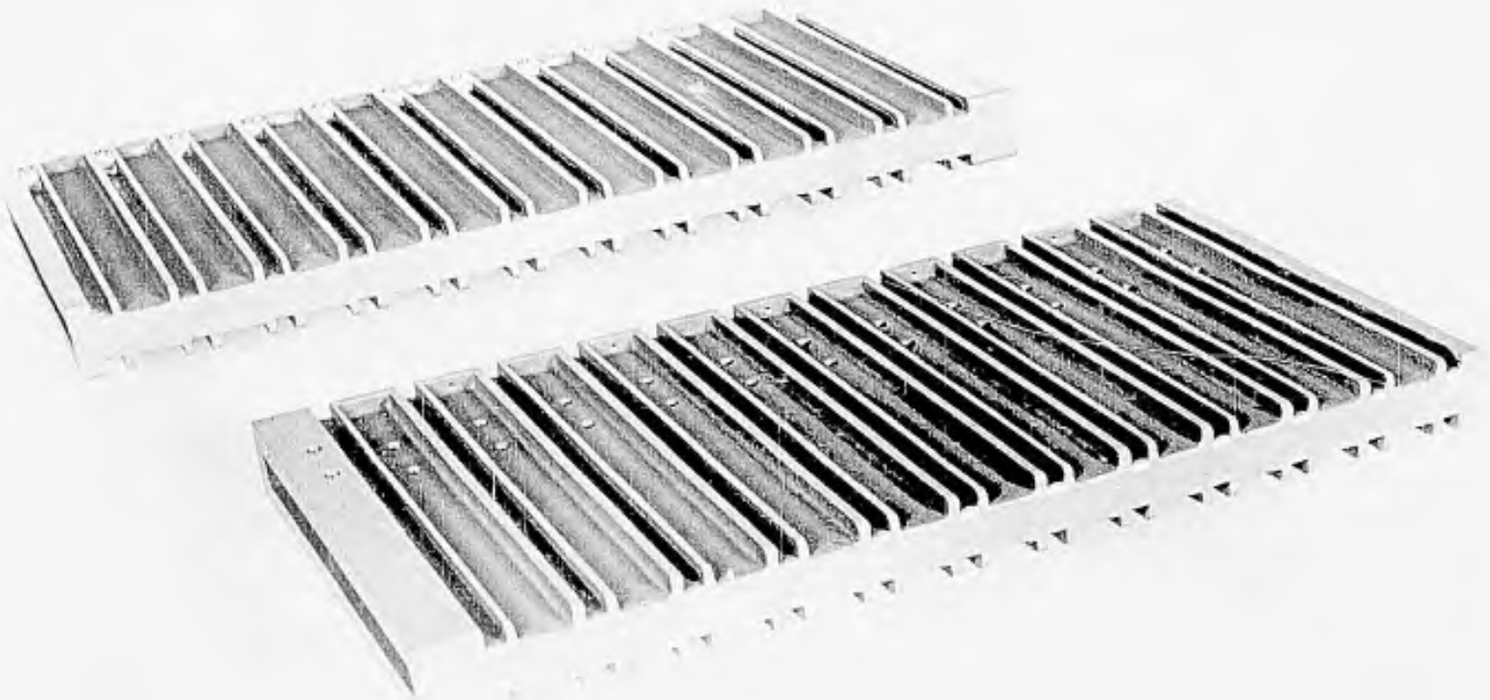


Figure 42. Finished Shelf

4. 5. 3 Rack Layout

The main frame logic layout was prepared by both development engineers and design engineers. The latter were used extensively for determining the logical groupings of the various registers and control logic. Since three cabinets constitute the main frame, an extensive analysis was conducted to effect a layout that would minimize the signal connections between them. Basic rules were established, such as laying out the registers wherever possible with the most significant digit on the left and the least significant digit on the right. An attempt was made also to group the pulse amplifiers in certain columns of the bays, to reduce the time needed for marginal checking of the pulse amplifier packages.

In laying out the logic in the cabinets, great consideration was given to maintainability, reliability, and maximum utilization of package space.

The following is a list of decisions and compromises in view of the problems involved:

a. A random wiring pattern (or point-to-point wiring) technique would be used to reduce problems arising from cross-talk and capacitive loading.

b. Coaxial cable would be used for the transmission of high frequency pulses between cabinets, and in some cases between bays in the same cabinet.

c. Signal drivers, or transistor emitter followers, would be used to drive the coaxial cables, to alleviate loading of the pulse amplifiers.

d. Clock pulse signal leads would be terminated, to reduce ringing due to excessively long leads.

e. OR Gate and Delay Line packages would be loaded as close as possible to the pulse amplifiers they feed. A maximum spacing of three horizontal spaces or one diagonal space was maintained between these critical units and their associated amplifiers.

f. The cabinets of the computer system would be arranged to eliminate excessively long coaxial cables, since even coaxial cables introduce delay and cause pulse deterioration.

The physical arrangement of the computer cabinets was an important factor in the logical organization of each cabinet. Control Unit II is adjacent to the Memory Unit. Since Control Unit I contains the Timer, it is strategically located for the distribution of timing pulses to all other units. (Figures 43, 44, 45 and 46.)

4. 5. 4 Test System

Early in the development of UDOFT, a breadboard evaluation bay was constructed to test and study various packaging techniques. The prime areas of interest were:

a. Signal lead lengths, such as coaxial, package to package, and the interaction between them

b. Methods of wiring

c. Distribution of cooling air

d. Fabrication problems

e. Marginal checking

f. Actual packaged circuit operation

g. Components, such as diodes, pulse transformers, tubes, and delay lines

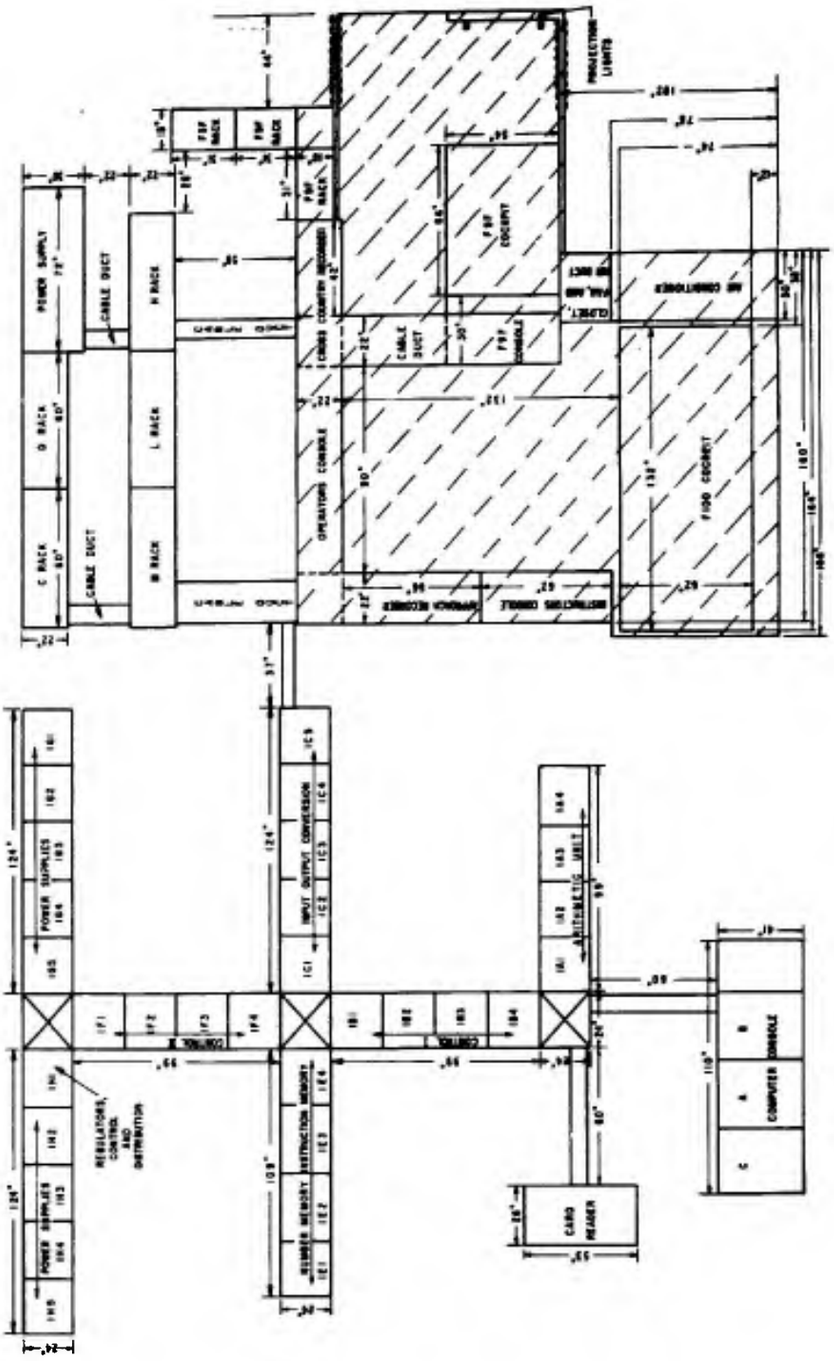


Figure 43. UDOFT System Layout

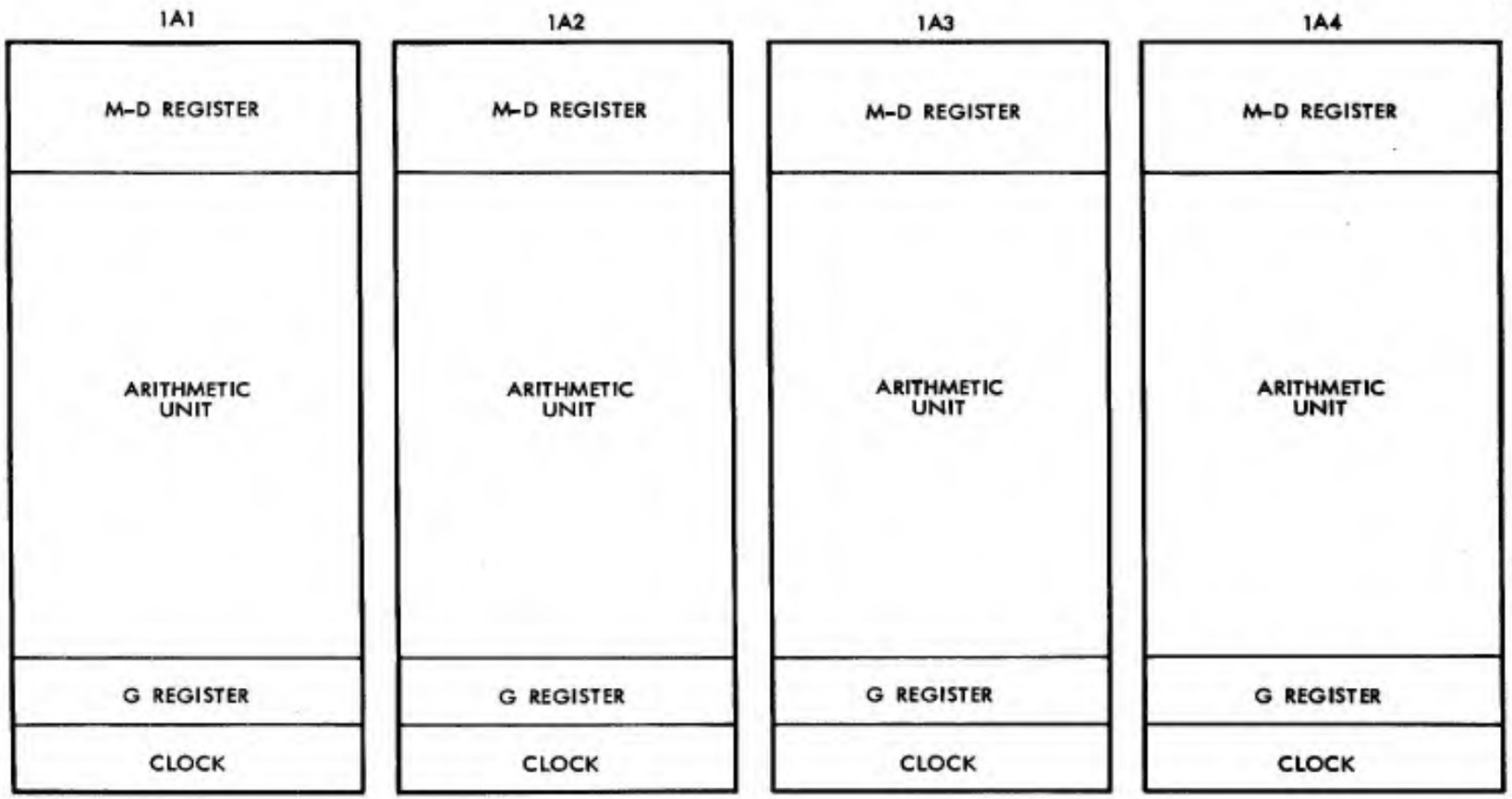


Figure 44. Layout of Arithmetic Unit Cabinet

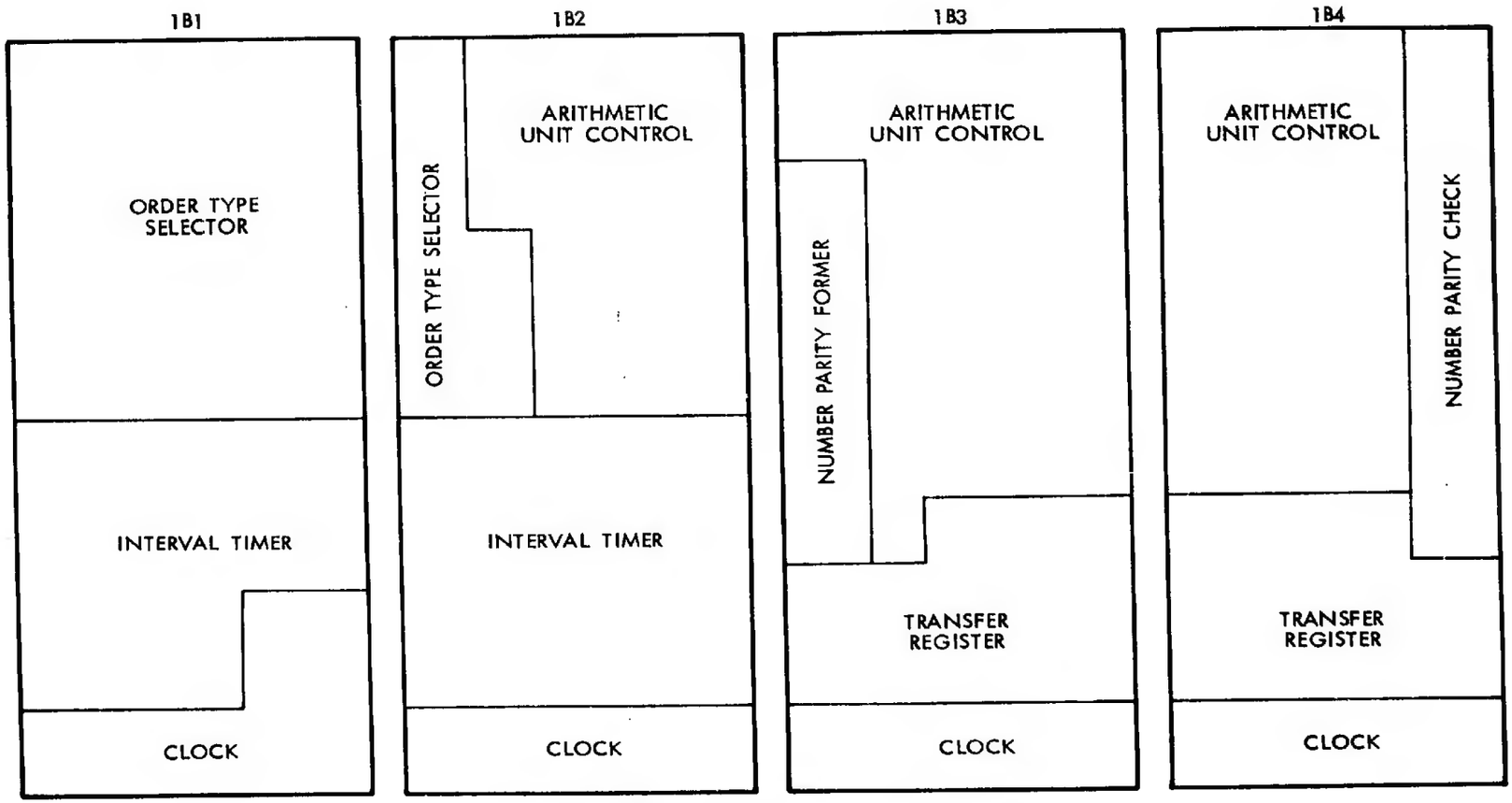


Figure 45. Layout of Control Unit I Cabinet

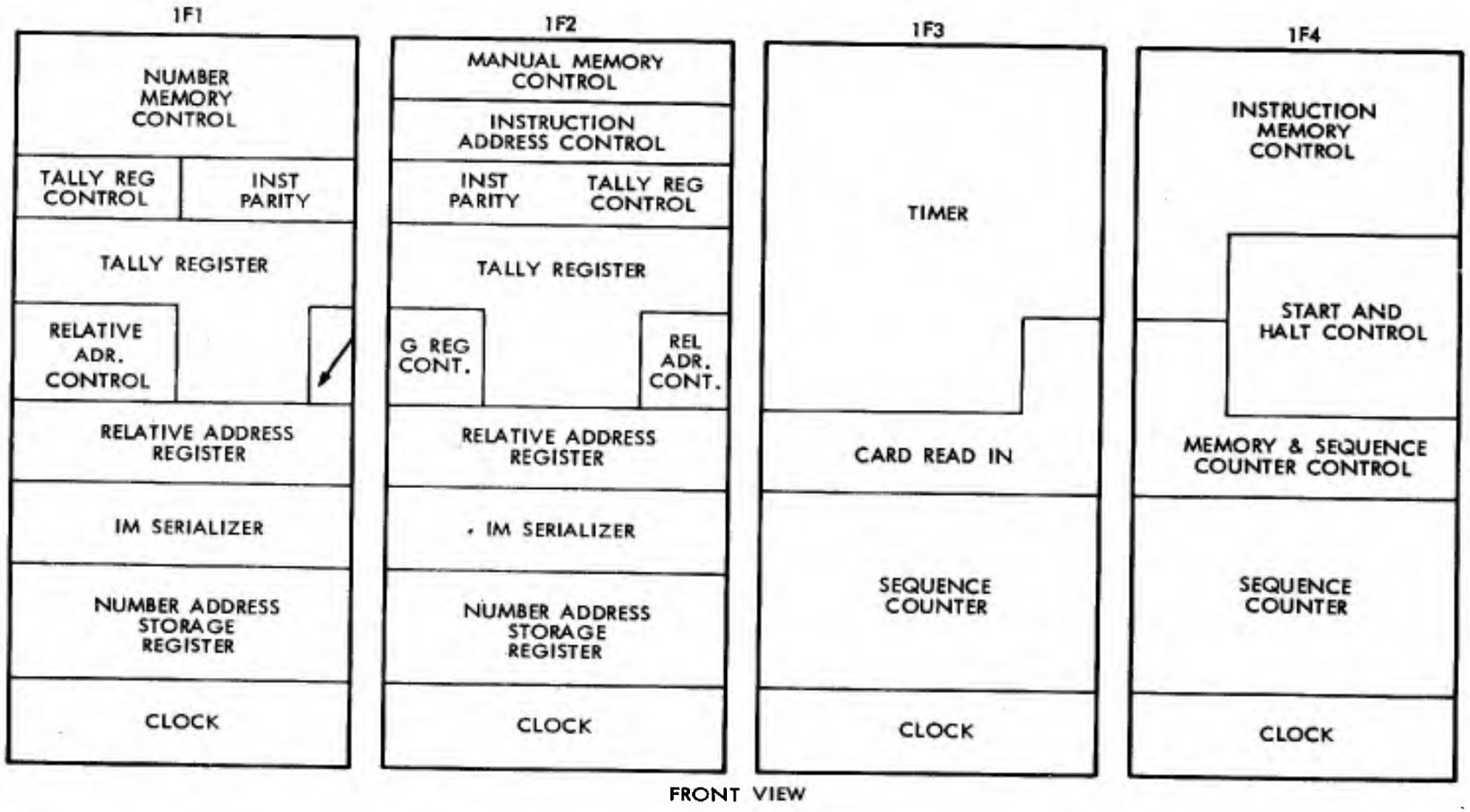


Figure 46. Layout of Control Unit II Cabinet

In summary, the test system consisted of 38 printed circuit packages arranged in a small, compact, logical system. The delay lines used were pieces of Millen delay-line cable cut to the appropriate lengths. The clock system vaguely resembled the system presently in the computer. The power supply consisted of nine laboratory-type d-c supplies with varying regulation characteristics.

After assembly of the system, approximately five weeks was consumed putting it into operating condition. Effort was applied primarily to correcting the logical timing errors, altering the clock system, discovering faulty diodes, evaluating various pulse transformers to select one for the pulse amplifier circuits, and checking hot spot temperatures.

The test procedure established consisted of routine daily checks on the different logical subsystems while varying the marginal checking voltage. During the period between 6 May 1957 and 12 December 1957, 4030 hours of breadboard operation were recorded. In this length of time there were 18 system failures, described as follows:

Logical system	3 (catastrophic) 1 (marginal)
Power supplies	7
Clock system	7

None of the failures involved the JW5847 vacuum tube, used as pulse amplifier tube, which was being evaluated. As can be seen, only four out of eighteen (approximately 22 percent) of the failures occurred in the logical system. Primarily because of the large percentage of power supply failures and the increasing number of failures that were occurring in the obsolete clock system, it was necessary to terminate the breadboard life test after 4030 hours of operation, although it was intended to conduct a 10,000 hour life test.

Although the life test ended in apparent failure, much had been gained from the breadboard.

a. The positive clamped output of the pulse amplifier was found critical when driving a remote load by means of a single long wire. When the lead length approaches twenty feet, the pulse is so badly distorted that it is unusable.

b. The marginal check voltage limits for the various packaged circuits were established.

c. It was evident that the clock system had to be revised; passing a square wave through a delay line resulted in a pulse distorted beyond recognition. The system was revised and a sine wave, rather than a square wave, was delayed then amplified and squared for use as a phase of the clock.

d. The initial ground rules for circuit packaging and plug-in package interwiring were substantiated.

4.6 Memory Development

As part of the UDOFT design, MSEE specified a random-access magnetic-core type of memory. At the time of the logic design, magnetic core memories were just becoming an acceptable method of storage. MSEE felt that by the time the initial UDOFT computer design was developed into a working system, the use of magnetic core storage, having a 5 microsecond cycle time, would be feasible. MSEE therefore specified only the timing operations, the address register and rewrite register logic, and suggested an approach to the development of this new type of memory. Circuit design and core specifications were the responsibility of the development contractor.

4. 6. 1 Memory Design and Development

The short time allowed for memory development necessitated the use of proven techniques wherever possible. Originally, it was proposed by MSEE to use a magnetic switch core matrix for the current driving devices. This approach was viewed unfavorably, due to the memory speed requirement (5 μsec) and the many disadvantages inherent in the magnetic switch core. The decision was made to use pulse transformers and to improve upon a system developed at M. I. T.

4. 6. 1. 1 Introduction to Coincident-Current Memories

In a coincident-current, magnetic-core memory of the type pioneered by M. I. T. Lincoln Laboratory, it is necessary to pulse two lines to a single core simultaneously to affect a read-out of the information stored in the core. The information, in the binary form of ONEs and ZEROs is contained in the remanent magnetic state of the square-loop-characteristic ferrite material of the core. A half-amplitude current pulse will not cause the core to change state, and the resultant voltage due to the disturb pulse is of a few millivolts' amplitude. Two half-read pulses produce sufficient magnetomotive force to drive the core into saturation. When the contents of the core (before the pulse) is a ZERO, the pulse causes only a 15 or 20 millivolt output from the selected core; when the contents of the core is a ONE, the output voltage is considerably larger, in the order of 100 or 200 millivolts.

Information read-out is a destructive process; i. e., at the end of a full read, regardless whether a ONE or a ZERO was contained in the core, the core has been driven into the ZERO remanent state. A full write, which normally follows every full read, then drives the core back into the ONE state. Means must be provided to prevent or inhibit the full write from occurring when it is required to write a ZERO into the core. The inhibit pulse is, therefore, a half-read overlapping the two coincident half-writes, resulting in a net half-write applied to the selected core.

4. 6. 1. 2 Description of the UDOFT Memory

The UDOFT memory is composed of two identical units, one for number storage and one for instruction storage. Each memory unit has 4096 registers; each register contains 22 bits. Parallel readout of a register is affected by pulsing 22 cores simultaneously, thus reading-out the entire contents of that register. Each memory is comprised of 22 planes; each plane contains 64 X 64 cores for a total of 4096 cores per plane. The X and Y lines are those 64 lines running horizontally and vertically, respectively, in each plane. Each core has one X and one Y line through its center, in addition to a Z inhibit and a Sense or read-out wire. Each of the 64 X lines is continued through the entire stack of 22 planes by external connections between each plane and the immediately adjoining plane; the 64 Y lines are similarly joined. Word selection is accomplished by simultaneously pulsing the X and Y lines which intersect at the location of the desired word.

4. 6. 1. 3 X and Y Drive Requirement

The core load on the X and Y Driver consists of 22 lines of 64 cores each, or a total of 1408 cores. Of the 1408 cores, 1386 receive a half-read and are not switched during the read-out. The other 22 cores simultaneously receive a half-read from the X-Driver and a half-read from the Y Driver. This action switches these fully-selected cores into saturation during the pulse; they return to the ZERO remanent state after the pulse. The back voltage seen by the X-Y Driver then is the sum of 1386 disturb voltages, which, because of their reversible nature, occur during the rise time of the drive current pulse. An average of 20 millivolts for 1386 cores would therefore cause a peak of approximately 27. 7 volts. That amplitude actually was observed. The 22 fully selected cores will contribute a back voltage varying anywhere between the extremes which occur when either 22 ZERO or 22 ONES are selected. The former extreme would contribute about one-half volt, mostly during the rise time; the latter, about 22 times 220 millivolts, or 4. 4 volts, occurring later than the rise time by about half the nominal switching time of the cores. For the UDOFT cores, the ONES peak occurs about 0. 6 microsecond after the start of the drive current pulse.

Within five microseconds the X-Y Driver must be addressed, drive a pulse of fixed amplitude and width into its load, and closely follow that pulse with one of opposite

polarity but equal amplitude. Furthermore, changes in loading caused by different remanent states of the 1408 cores receiving the pulses, especially those 22 cores addressed for read-out, must not cause the pulse to change appreciably in rise time or amplitude. Excessive reduction of amplitude results in a reduced ONE output with a longer switching time; an increase in switching time must be avoided in order to have a reliable strobe time. The latter is extremely important since it must occur at the right moment to exclude a maximum ZERO and include a minimum ONE.

4.6.1.4 Word Selection

A Memory Address Register, MAR, consisting of twelve standard static flip-flops, is addressed by the computer to select any one of 4096 words; two raised to the twelfth power equals 4096. Earlier memories of the same capacity employed 64 current driver tubes (twin triode 5998's) and 64 miniature tubes as Driver Amplifiers for each coordinate, X and Y, (figure 47). Word selection was accomplished by lowering the grid of one amplifier of the 64 in the X line and one amplifier of the 64 in the Y line through X and Y diode matrices, each of which had an input of six input pairs (from the six MAR flip-flops addressing that coordinate). In addition to the 128 tubes per coordinate listed above, two Gate Generators for read and write pulsing were required, each consisting of two 5998 tubes and two smaller tubes. An alternative to this method was suggested by Papian:[1] to group the grids and cathodes of the driver tubes in an 8-by-8 matrix arrangement so that one Driver Amplifier could control eight 5998's. (Figure 48.) But in reducing the number of Driver Amplifier cathodes by 112, involving 56 tubes, the Gate Generator cathode count increases by 98 for an overall saving of 2 times 14 (112 minus 98), or 28 cathodes, involving 14 tubes, per memory. The 56 tubes eliminated are of the miniature variety; of those added per memory, 28 tubes are "giants" (5998), 14 are pentodes, and 14 are miniatures. Such an overall reduction is inconspicuous.

4.6.1.5 Evolution of the Transformer Matrix

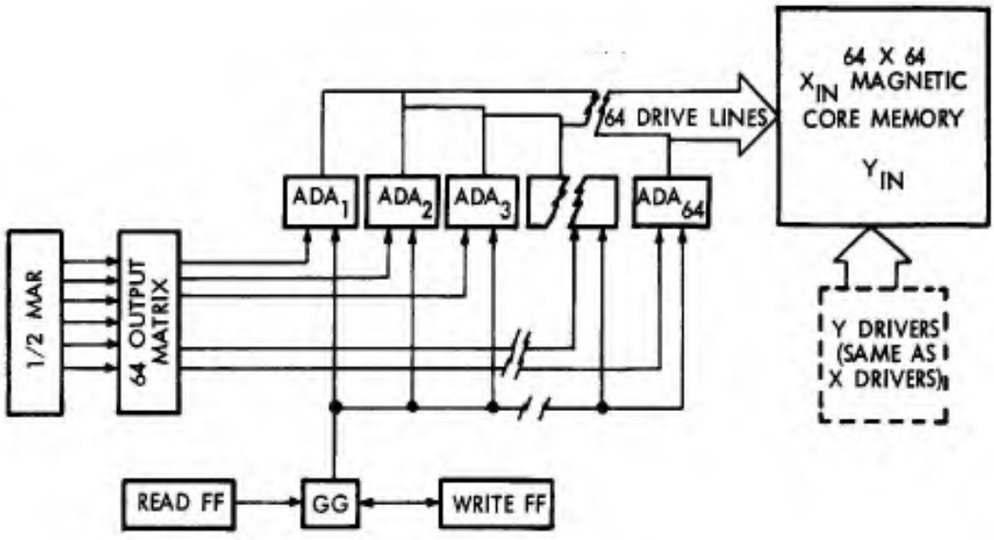
A step toward a more promising reduction in tubes was the simplification of the Gate Generator, which in Papian's scheme contains two 5998 tubes in each read and each write half, a total of four tubes, conducting about 40 ma each in the steady state and interrupted only when current is to flow in the transformer. Considerably less average power would be consumed in the Gate Generator tubes if current flowed only during the read or write pulses, respectively. Such a thought leads naturally to connecting the Array Driver in series with the Gate Generator. The circuit shown in figure 49 was a step in that direction, reducing the number of 5998 tubes in each Gate Generator from 2 to 1.

Current regulation is accomplished by the large resistance between the Gate Generator cathodes and -150 volts. Under quiescent conditions all grids are below cutoff and current flows only in the diodes and 1.2K resistor. When pulses are applied coincidentally to an Array Driver and a Gate Generator, current flows simultaneously into the 1.2K resistor from the diodes and the tube cathodes. When the sum of these two currents exceeds the quiescent diode current, the cathode voltage rises slightly, the diodes cut off, and the resultant degenerative feedback prevents the plate current from increasing appreciably above that level.

A disadvantage still in evidence was the use of 64 large tubes for the Array Drivers. This seemed particularly wasteful because only one tube of the 64 is conducting at any one time. The driving requirement on each Array Driver Amplifier is rather severe also, since it must enforce a considerable swing on the stubborn capacitance of 16 grids in parallel, each grid presenting approximately 15 picofarads.

It was suggested that transformer primaries be connected in an 8 by 8 matrix between the Array Driver cathodes and the Gate Generator plates, so that one out of eight Array Driver tubes and one out of eight Gate Generator tubes would uniquely select one out of 64 transformers and hence energize one line in the memory core array. A requirement for this circuitry is a transformer with two primaries not electrically connected, i.e., no center-tapped primary. After examination and breadboarding this system was adopted.

1. Papian, W.N., "New Ferrite Core Memory Uses Pulse Transformers", Electronics, March 1955, p. 194.

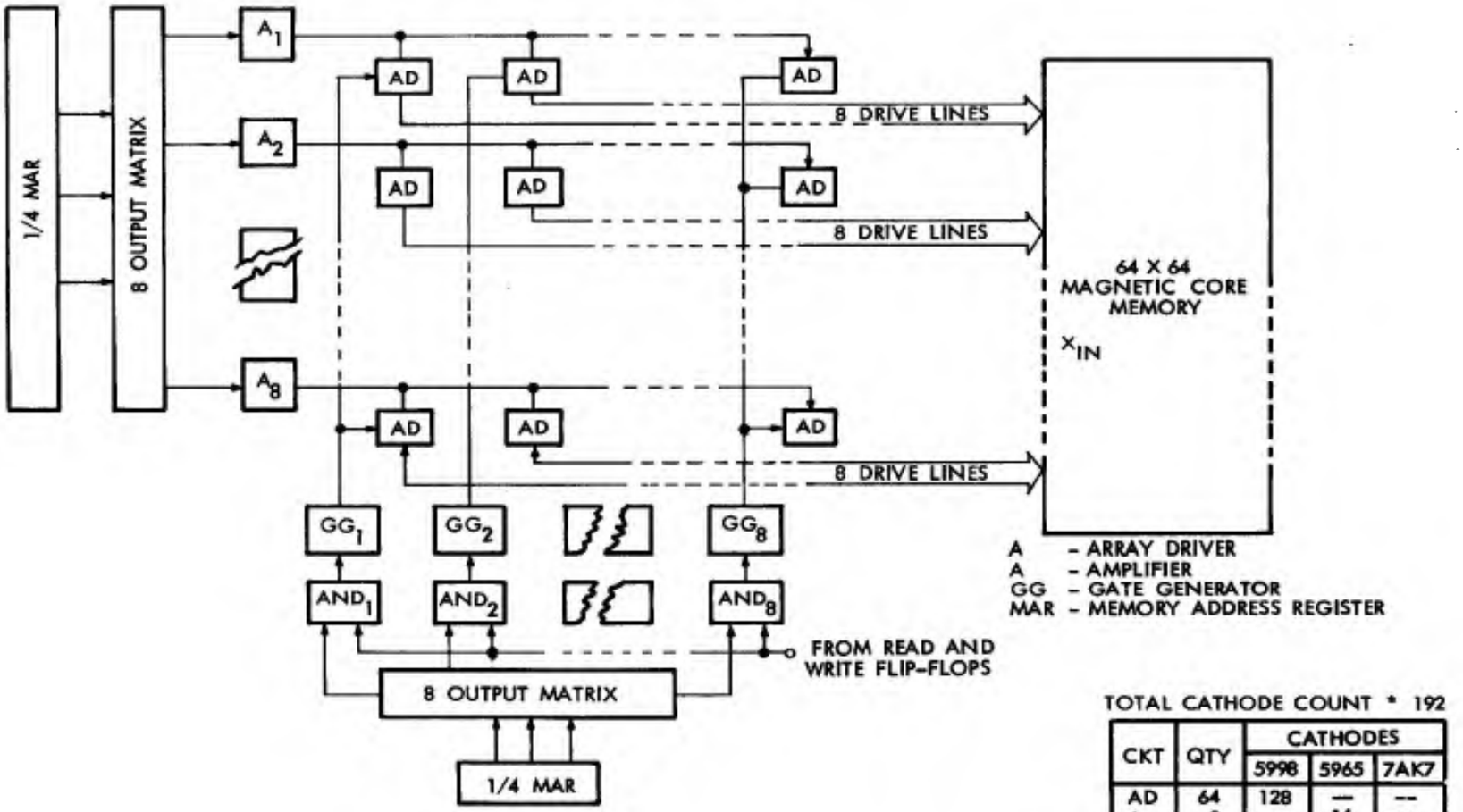


TOTAL CATHODE COUNT = 270

CKT	QTY	CATHODES		
		5998	5965	7AK7
ADA	64	128	128	-
GG	1	8	4	2
TOTAL		136	132	2

ADA - ARRAY DRIVER WITH AMPLIFIER
GG - GATE GENERATOR
MAR - MEMORY ADDRESS REGISTER

Figure 47. Common Coordinate Driving Technique



TOTAL CATHODE COUNT * 192

CKT	QTY	CATHODES		
		5998	5965	7AK7
AD	64	128	—	—
A	8	—	16	—
GG	8	16	16	16
TOTAL		144	32	16

Figure 48. Alternate Coordinate Driving Technique (Papian)

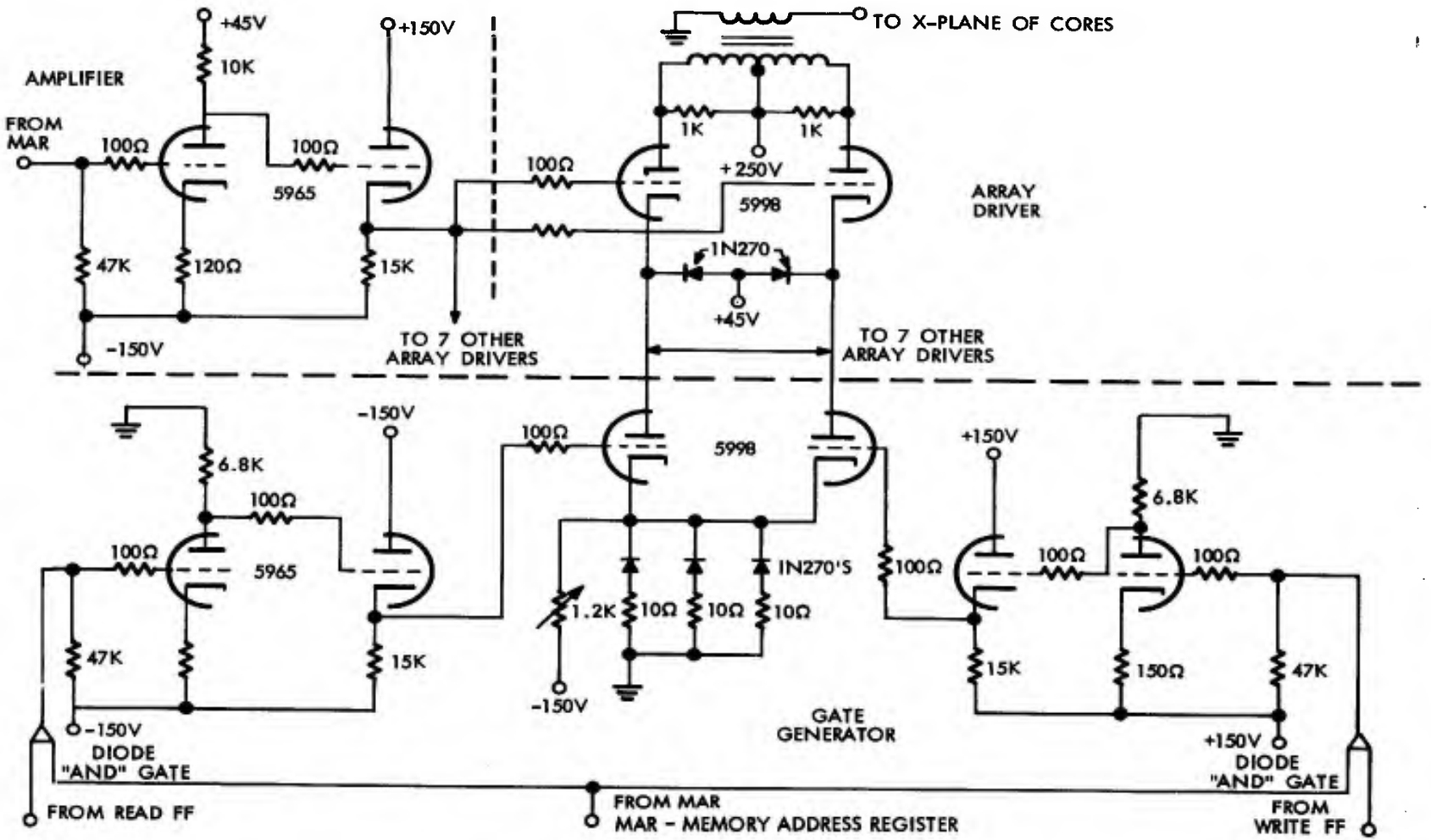


Figure 49. Schematic of a Preliminary Coordinate Driver Circuit

4. 6. 1. 6 Transformer Matrix

The block diagram of figure 50 shows the application of the transformer matrix in the present X-coordinate driver system. Only the read section is shown; an identical arrangement is necessary for write. Series diodes are necessary to block sneak paths. Two diodes instead of one are used so as to minimize the capacitive loading of the transformers.

4. 6. 1. 7 Circuit Description

The quiescent bias levels of the Array Drivers and Gate Generators must be sufficiently negative to insure cutoff at all times, particularly during the time the selected pair is pulsed. For the 7236 tube, which has replaced the 5998, this means a bias of about -60 volts on the Gate Generator; -40 volts is sufficient for the Array Driver, since the pulse on the selected Array Driver tends to raise all Array Driver cathodes. Figure 51 shows the circuit for an Array Driver and Gate Generator and their respective amplifiers. The operation is as follows:

The Address Register has been set to one of the 64 addresses for X and one for Y. Only one coordinate needs to be discussed, since the operation of X and of Y is identical. Of the eight outputs from each diode matrix, seven are at +20 volts and one is at -4. 5 volts. Each of the eight outputs is routed through two diode AND gates to its read and write amplifiers. The other inputs to the AND gates are static flip-flop outputs which determine the read and write pulse widths. The -4. 5 volt level selects the amplifiers to receive the flip-flop output pulses, which are negative.

The Gate Generator Amplifier consists of the two sections of a 5965 tube, one section as voltage amplifier and the other as cathode follower. The amplifier section is normally conducting, so that a negative 60 volts is applied to the grid of the Gate Generator 7236 tube. The negative input pulse cuts off the amplifier, causing its plate to rise to ground; the peaking coil shortens the rise time and the delay time, and causes overshoot of about ten volts. The overshoot is useful in turning on the Gate Generator, since lowest plate voltage occurs at the beginning of plate current, indicating the need for a more positive grid bias at that time.

Simultaneously with the turning-on of the Gate Generator, an Array Driver must become active if any current is to flow in the selected transformer. The Array Driver Amplifier operation is similar to that of the Gate Generator Amplifier. The quiescent output voltage is about -40 volts, but during pulsing this must increase to +150 volts to bring both the Array Driver and Gate Generator tubes to the proper operating point. The larger voltage swing ruled out the use of a 5965 tube, because of power dissipation limitations; therefore a 5687 was used. At first it was thought feasible to connect the grids of the read Array Driver and the write Array Driver and turn on both for the duration of the five microsecond minor cycle, pulsing only the Gate Generators separately for read and write. However, this proved impracticable because of the coupling between the two primaries of the transformer, which resulted in unacceptable current waveforms.

4. 6. 1. 8 Sense Amplifier

The Sense Amplifier is a device designed to sense and amplify the information signals derived from an interrogated magnetic-core memory plane. The output of this amplifier, when used in conjunction with a strobing pulse, is sufficient to trigger a static flip-flop. Thus the amplifier must be able to distinguish between ZERO and ONE signals induced on the sense winding, to eliminate all common mode disturbances, and to shape the information signal for strobing. The Sense Amplifier circuit consists of three sub-circuits whose functions are differentiation, discrimination and pulse shaping. (See figure 52.)

a. Differentiation. The differential circuit consists of that portion encompassed by the input terminals and transformer T_1. It is the function of this circuit to amplify difference inputs, and to eliminate common mode signals. Common mode signals are disturbances that originate due to coupling between the drive lines and the sense winding of the memory plane. Such inputs raise both ends of the sense winding simultaneously and cause

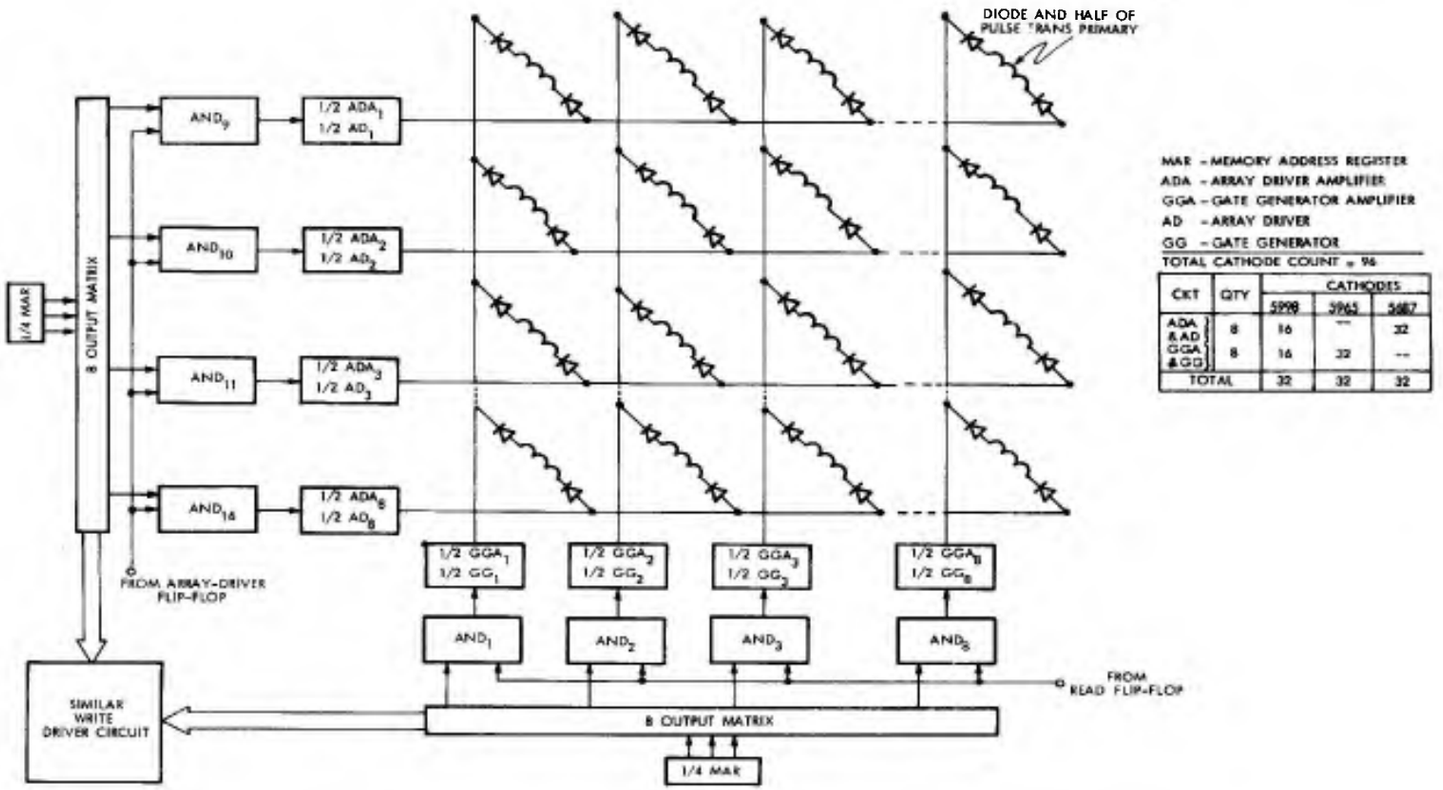


Figure 50. Simplified Block Diagram of Present X-Coordinate Driving Technique

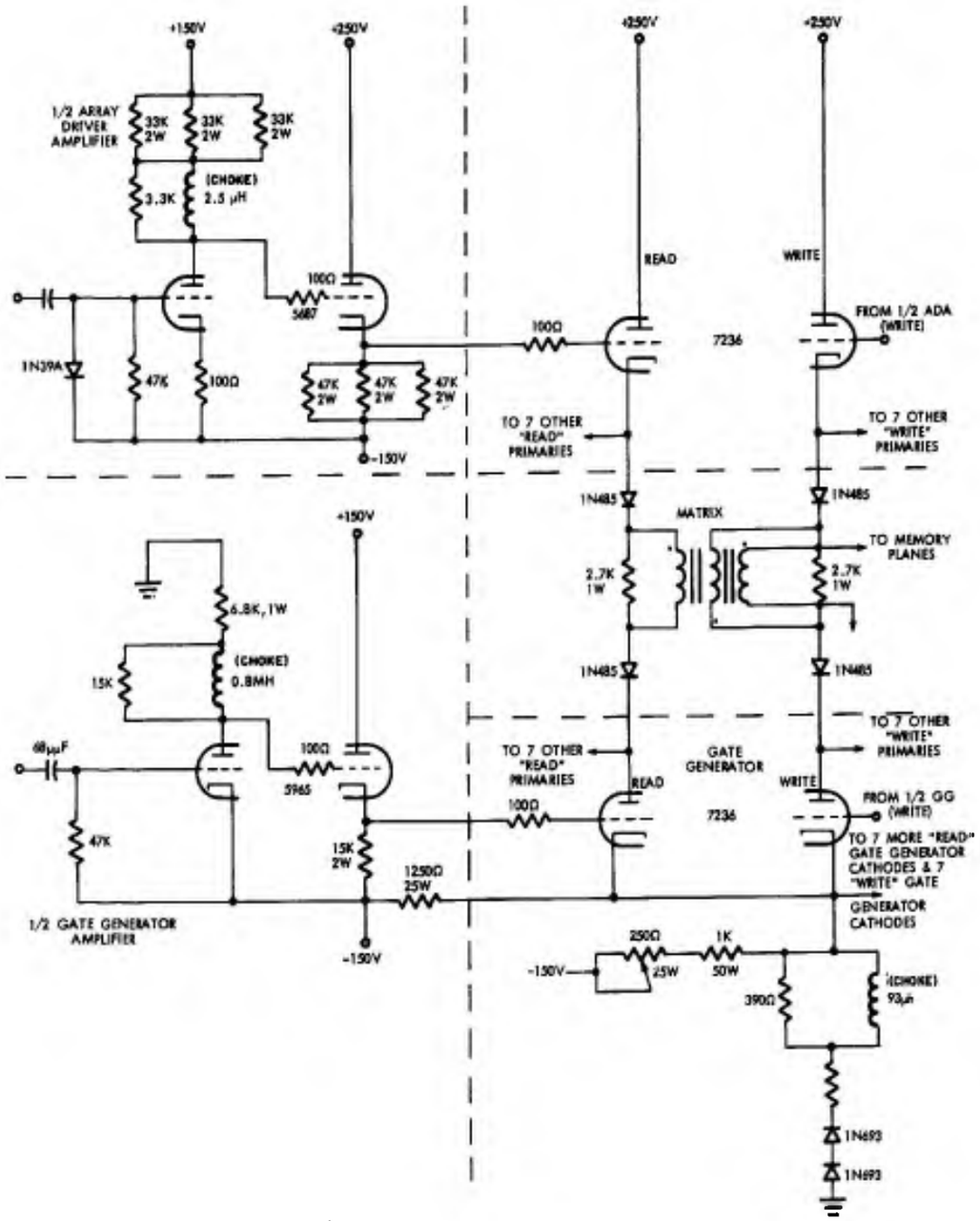


Figure 51. Schematic of X-Coordinate Driver Circuit

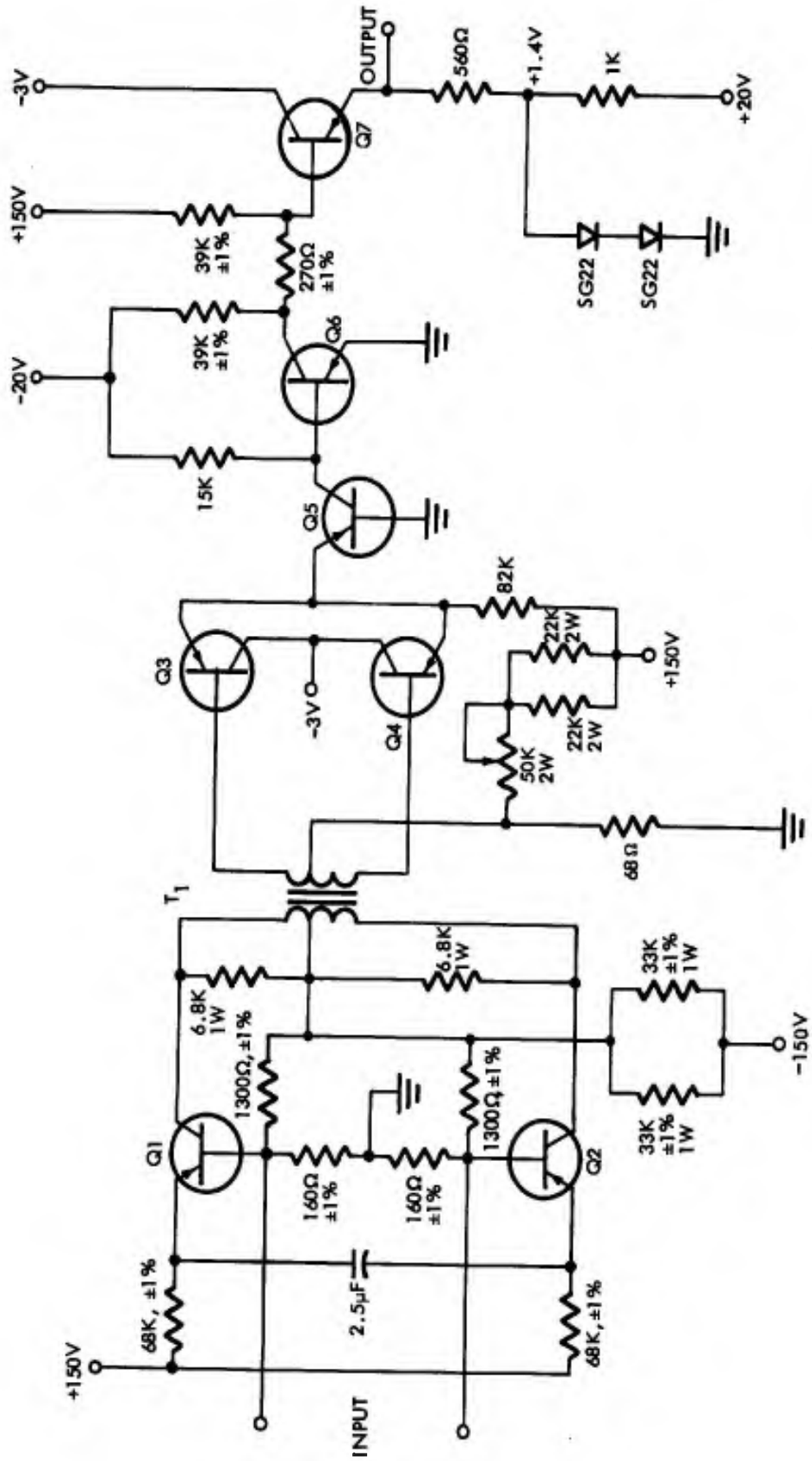


Figure 52. Schematic of Sense Amplifier Circuit

transistors Q_1 and Q_2 to conduct equally. Since the transformer primaries associated with these transistors pass current in opposite directions, complete cancellation of the unwanted signal is accomplished. The ability of this circuit to reject completely these common mode signals depends on the careful selection of components. For this reason, all resistors involved here have tolerances of ±1 percent.

When presented with a ZERO, ONE, or Inhibit induced output from the memory plane, the conduction of one transistor, Q_1 or Q_2, is increased while that of the other is decreased, resulting in an output from the secondary of T_1. ONE inputs to the amplifier have maximum amplitudes of 200 mv or so, while Inhibit inputs may be in the order of several volts. The secondary outputs of T_1, however, show none of this wide variation, since either Q_1 or Q_2 is driven into saturation by the stronger inputs.

b. Discrimination. A variable voltage divider placed in the center-tapped secondary of T_1 places a positive bias voltage on the bases of transistors Q_3 and Q_4. With Q_5 normally conducting, the emitters of Q_3 and Q_4 have a quiescent level of about +0.28 volts. Any positive voltage above +0.28 volts will then keep these transistors in the OFF condition. Since the voltage levels of ONE signals are greater than those of ZERO signals, the dc level placed on the bases of Q_3 and Q_4 can be set so that only the voltage levels of ONE signals will cause them to conduct. This is the method of rejecting ZERO inputs. The signal appearing across the secondary tends to make one base more negative and the other more positive. If the signal is large enough to overcome the dc bias on either base, it will cause that transistor to conduct. As an example, if the total gain preceding Q_3 and Q_4 is 10, then a setting of one volt on the divider will barely allow 100 millivolt signals to pass and will reject all signals of lesser magnitudes. If this bias were made smaller, the 100 mv signal would still pass through, but now would result in a wider output signal since conduction of either Q_3 or Q_4 begins before the 100 mv level.

c. Pulse Shaping. The remaining circuitry shapes the signal from Q_3 or Q_4 so that the output signal is at least 0.4 micro-seconds wide and 6 volts in amplitude. The quiescent level of the pulse is -3.0 volts, so that the maximum positive point that it reaches is 3 volts.

4.6.1.9 Inhibit Driver and Amplifier

The function of the inhibit circuitry is to produce a half-read current pulse during the memory write cycle time, in order to write a ZERO (i.e., to prevent the writing of a ONE) into a selected core within a given memory plane. For example, to write a one into a core it is necessary to have currents $1/2 I_{y_{write}}$ and $1/2 I_{x_{write}}$ occurring simultaneously; either current alone is insufficient to switch the core. When present, the inhibit current will cancel either the $1/2 I_y$ or the $1/2 I_x$ current, since it is equal and opposite. Since $1/2 I_y$ or $1/2 I_x$ alone is insufficient to switch the core, a zero is written into the core.

The inhibit driver power amplifier, a 7236 tube, was designed to provide an inhibit current pulse of 410 milliamperes peak amplitude with rise and fall times of 0.4 microseconds and 0.7 microseconds respectively. (Figure 53.) The amplitude of the output from the inhibit driver is stabilized by the large amount of cathode degeneration in the 5965 feedback amplifier tube and by negative feedback from the cathode of the 7236 current driver tube. Since decreasing transconductance in the 7236 current driver tends to be counteracted by decreasing transconductance in the 5965 feedback amplifier, the effect of tube aging on the inhibit driver output amplitude is substantially minimized.

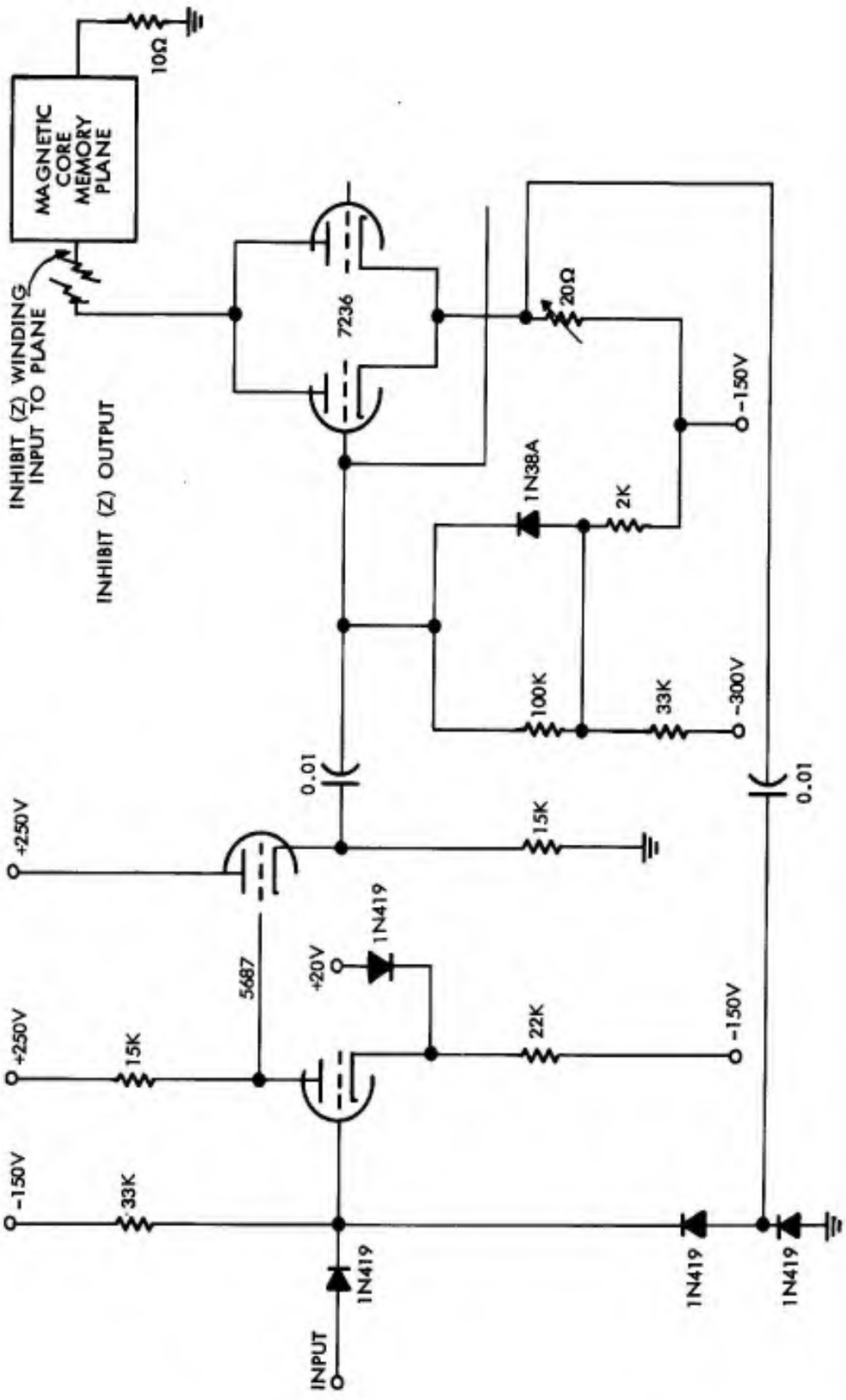


Figure 53. Schematic of Memory Z (Inhibit) Drive Circuit

4. 6. 1. 10 Breadboard System

A breadboard of a partial memory system was constructed to evaluate doubtful components such as the matrix transformers and the magnetic-core memory planes, for verifying the design of the drive circuits and for determining the optimum timing of the various control functions that occur during a 5 μsec memory cycle. Since it would have been economically unwise to use twenty-two costly magnetic-core memory planes in the breadboard system, the apparent loads carried by the current pulse generators (Array Drivers and Gate Generators) had to be simulated.

The loads for a single X-coordinate and a single Y-coordinate were simulated by means of two additional memory planes, one for each coordinate, wired to provide the same effect as an additional 21 planes (figure 54). Memory Plane 1 contains the fully selected core which may be sensed and written into as in the ultimate memory. Memory planes 2 and 3 simulate the loads for the X-driver and the Y-driver, respectively. Twenty drive lines of each simulated load plane were connected in series (figure 54); those used were lines 1 through 10 and 12 through 21. Line 11 is omitted, so that half of the cores along the fully selected line receive bucking current pulses and the other half receive aiding current pulses. If all 20 cores were to receive aiding current pulses, the back voltage seen by the transformer due to switching 20 ONES would have been twice that actually encountered in the memory, since each fully selected core would have effectively two lines (one X-line and one Y-line) in which the back voltage would be induced.

Variations in loading between switching 22 ONEs and 22 ZEROs for an ideal coordinate driver should produce no observable difference in output current. In the actual coordinate driver, however, the ALL-ONEs load caused a slight dip immediately after the drive current pulse had peaked; conversely, the ALL-ZERO's load caused an overshoot on the leading edge of the drive current pulse. It was found that the insertion of approximately 50 microhenries of inductance in the cathode of each Gate Generator helped to regulate the overshoot and the dip in the two extremes of load conditions, so that switching and peaking times remained nearly constant regardless of the load configuration. The inductance was later relocated to the constant current sources which are connected to the cathodes of the Gate Generator tubes for the X and Y coordinates. (figure 51).

As breadboard testing advanced, it became important to be able to switch between two cores, rather than being limited to a single selectable core. The switching was accomplished by selecting one of two X-coordinate lines, a single Y-coordinate line being pulsed simultaneously with either X-coordinate line. Since each coordinate line had its simulated loading distributed in another memory plane, a fourth memory plane was required.

The reasons for this development were:

a. To observe further the writing-in and reading-out of the memory cores in 5 microseconds.

b. To observe and rectify any transients that might occur when switching from one coordinate to another.

c. To determine whether inhibit noise or other transients would affect core read-out.

Results of extensive breadboard testing indicated that a memory could be addressed to read and write within 5. 0 microseconds. As a matter of fact, satisfactory operation was obtained with a memory cycle of 4. 6 microseconds. No transients or any other difficulties due to stray capacitive or inductive effects in or among the wires in the memory planes were encountered.

4. 6. 2 Memory Planes

Memory planes were procured for the UDOFT memory in accordance with a carefully prepared specification; both electrical and mechanical requirements were specified. No electrical problems were encountered with the planes accepted from the Vendor. However, the planes could not satisfy the specified shock and vibration

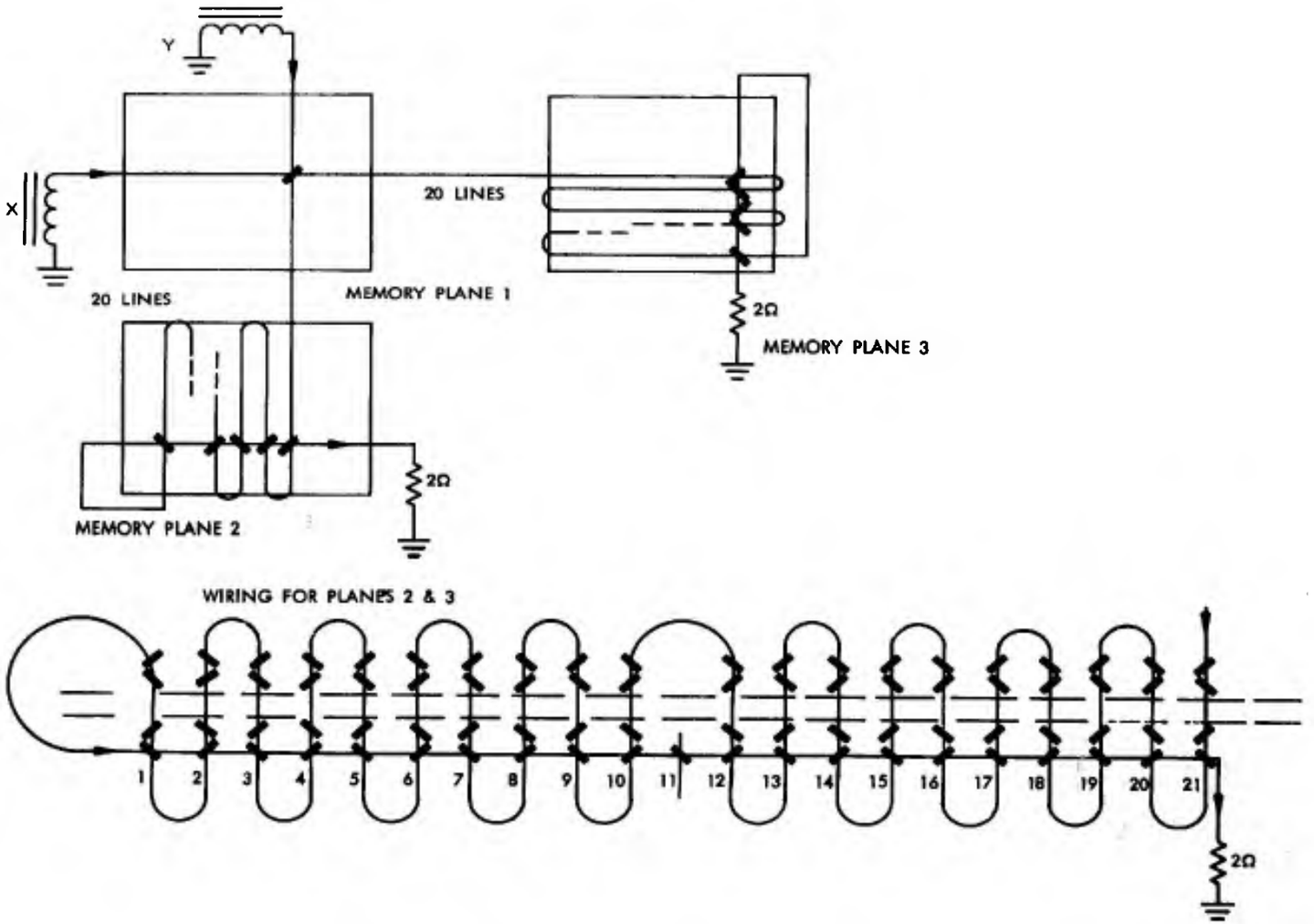


Figure 54. Simulated Load for Coordinate Drivers

requirements. Under these test conditions, the X and Y drive lines tended to break at the point where the line was soldered to lug on the frame of the memory plane. Also, the cores would rotate about the four wires through them, abrading the insulation of these wires. If this had been allowed to continue, the wires themselves could very well have been cut. To remedy the problem, each memory plane was backed or reinforced with a layer of glass cloth held in place by a coating of Polyweld.

Since temperature is a critical factor to be considered in the design of a magnetic-core memory system, temperature tests were conducted using the memory breadboard. The results indicated that satisfactory operation could be achieved for ±5° C variation about +20° C. However, it was still believed that an absolute maximum of 1° C difference in temperature should be maintained between any two cores in the memory array. Experience with the memory system during phases of total system operation proved these stringent temperature requirements were desirable and indeed necessary.

4. 6. 3 Memory Unit Cabinet

Great care was taken in laying out the Memory Unit and in packaging the memory planes, to determine the optimum design from both electro-mechanical and environmental standpoints and to utilize standard hardware wherever possible. It was decided that the Memory Unit Cabinet would consist of four bays similar to the Arithmetic Unit Cabinet, except that the two inside bays should have a 24-inch width rather than the standard 19-inch width. (Figure 55).

The memory plane arrays and associated components, of which there are 512 per memory plane array, had to be packaged physically as separate units, yet located close together to maintain the short lead lengths dictated by the minimum tolerable stray capacity. The memory plane assembly is secured in a plenum chamber which supplies the necessary air for maintaining the operating temperature of the magnetic cores and for cooling the components and core-driving vacuum tubes. The complete assembly is located midway up the 24-inch-wide bay, to allow maximum access to the unit.

The two outside bays, each of which is the standard 19 inches wide, contain the remainder of the control circuits, standard plug-in packages for which lead length is not critical.

The memory plane assembly and the driver tube chassis, located above and below the assembly, constitute the only non-standard items in the Memory Unit Cabinet. These non-standard parts must be used because of the nature of the planes, the necessity for short leads between drivers and planes, and the different cooling problems.

From the electrical design and the characteristics of the Type 5998 tubes used for the core drivers, it was estimated that each of the two 24-inch wide bays of the Memory Unit Cabinet would dissipate approximately 2300 watts. To maintain the specified operating temperatures with this amount of heat, it was necessary to divide the bay into two isolated sections for the purpose of cooling because for one end of the duplex blower could not handle the entire bay. One section, served by one end of the blower, contains the memory plane assembly and the driver tubes. This section dissipates approximately 1200 watts. The remainder of the bay, which contains the standard packages and is located above and below the memory plane and driver tube assembly, is cooled by the other end of the blower, dissipating the rest of the heat generated.

Information gathered from the operation of the memory breadboard helped establish ground rules for the layout of the logic packages in the Memory Unit Cabinet. Each memory was so packaged as to satisfy the critical requirements imposed upon the lead lengths of the X and Y coordinates. Since the X-coordinate circuits are identical to the Y-coordinate circuits, the package layout for the center bays of the cabinet is symmetrical about the magnetic-core memory arrays. Further, since the Number Memory circuits are identical to the Instruction Memory circuits, the layout of the two outside bays of the cabinet is symmetrical about the centerline of the cabinet. (Figure 56).

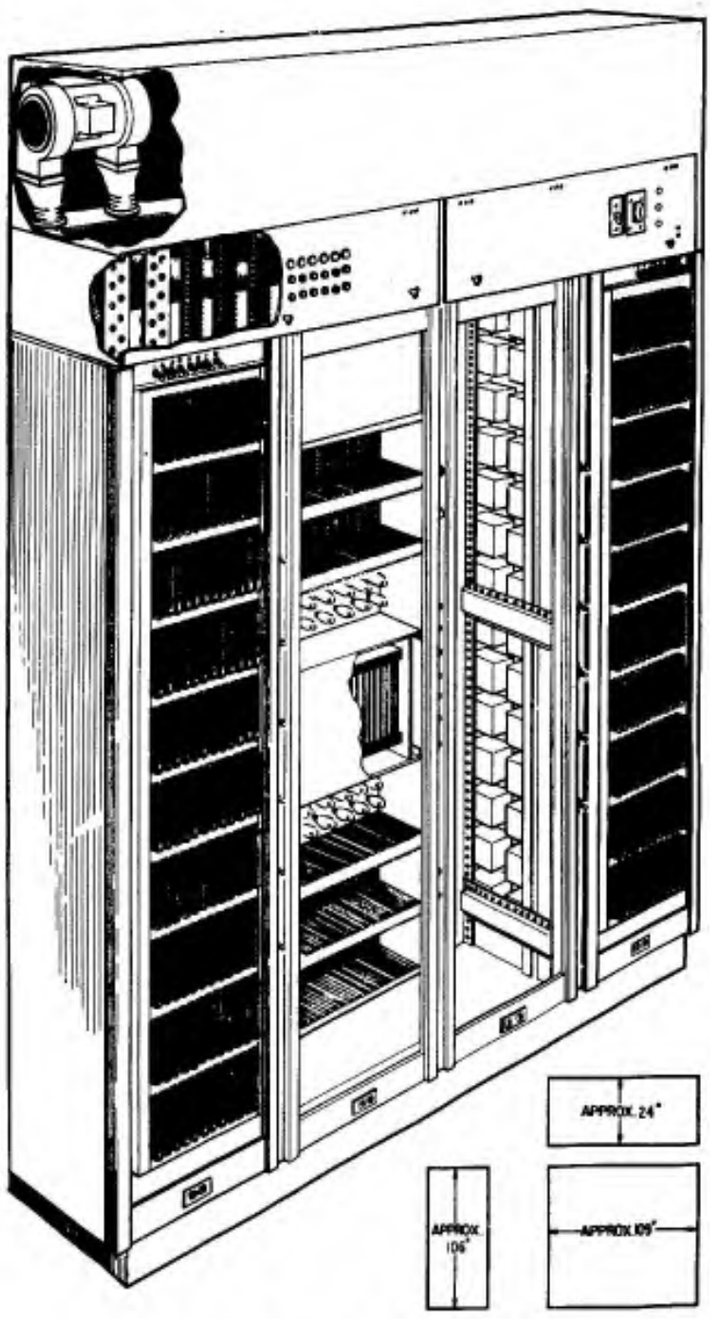


Figure 55. Memory Cabinet

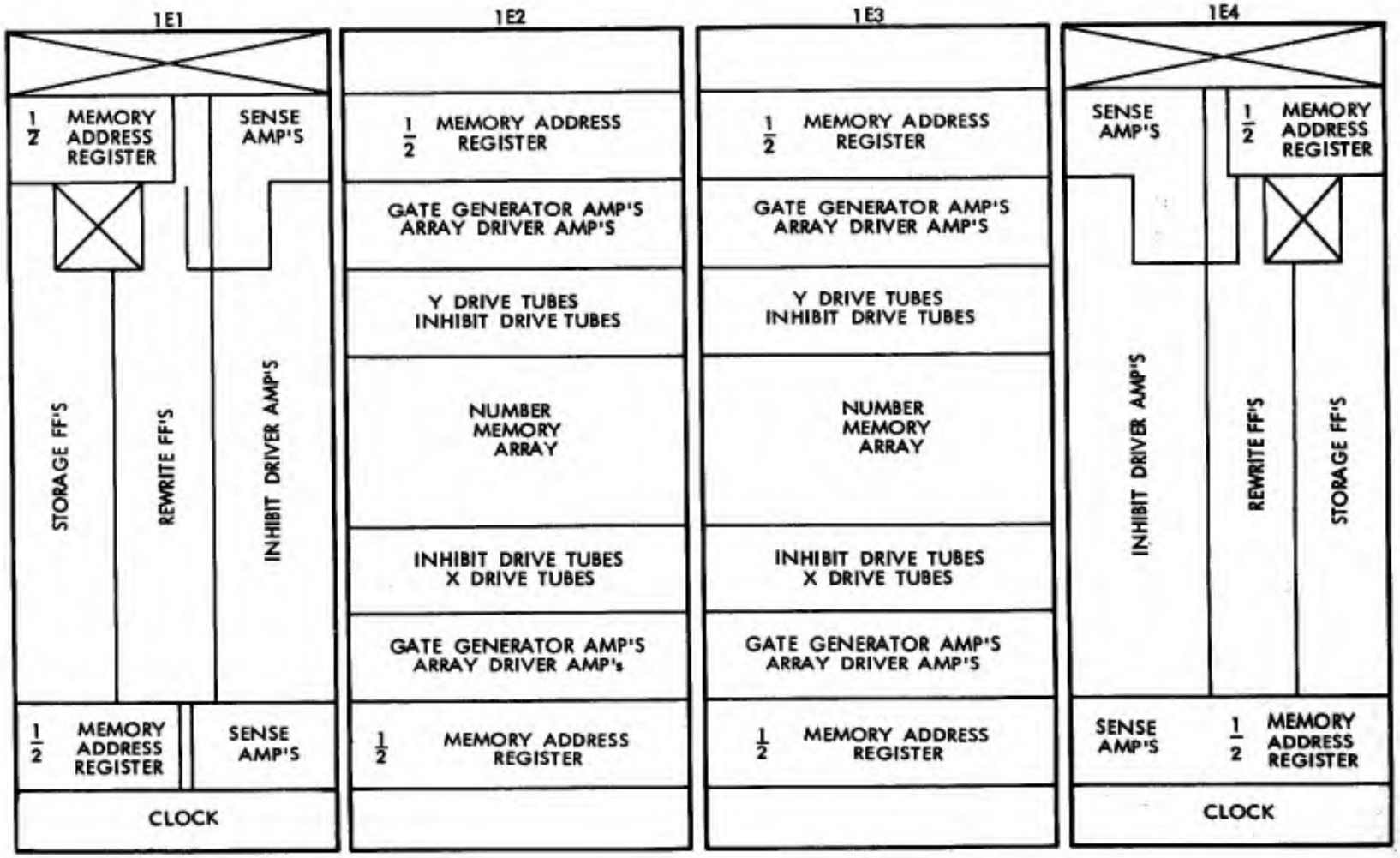


Figure 56. Layout of Memory Cabinet

4.7 Input-Output Development

4.7.1 Input-Output Unit Cabinet

The Input-Output Unit Cabinet consists of five bays. Four of these contain the control circuitry, logic, and circuits required for processing the discrete and analog outputs and inputs. The fifth bay houses the stepping switches and control relays for the output printer, or electric typewriter. This bay also contains a large number of spare package slots for any additional logic that may be added in the future. (Figure 57.)

The ground rules for emplacing the packaged circuits in this cabinet were identical to those established for the Memory Unit cabinet.

4.7.2 Computer Console

4.7.2.1 Console Cabinet

The console cabinet as originally planned was made of three commercial console sections permanently joined to form one unit approximately 64 inches long. A side elevation view of this cabinet design is shown in figure 58. After much consideration it was decided that the writing surface was too shallow, being only 10 inches deep; the well in the writing surface was disadvantageous, as it might become a receptacle for pencils, cigarettes, et cetera; the working areas of the control panels were below the writing surface as a result of the well, making the controls located at the bottom of the panels more difficult to operate and leaving them unprotected against accidental operation by notebooks or other items sliding across the writing surface; the slope of the panel was too shallow; controls and indicators at the tops of the panels were at an uncomfortable distance from the operator; and esthetically, the balance between the sections above and below the writing surface was poor.

These major objections were overcome by redesigning the upper section of the console. A side elevation of the redesigned cabinet also appears in figure 58. The writing surface was increased by covering the well; the control panels were raised above the writing surface; the slope of the control panel was increased to provide better vision of controls and the esthetic balance was improved by revising the upper section.

In addition, it was necessary to add to the console a structure to accommodate the output printer. Numerous schemes were considered; the best and most acceptable was the addition of a section, similar to the lower section of the commercial console units, at the right end of the console. This addition provided a broad surface to receive the electric typewriter, and the continuity of design and appearance was retained. A pictorial view of the finished console unit is shown in figure 59.

4.7.2.2 Console Panels

The details of the layout of each of the three console panels were discussed in Section 3.7. However, one item not discussed is the apparently misguided arrangement of register status indicators and register read-in switches.

The layout of these indicators and switches was made with regard for the programmer and the maintenance engineer. Since programmers prefer to use octal notation rather than the cumbersome binary notation, this group of status indicators and read-in switches were grouped and labeled to facilitate working with the octal form.

If the indicators are arranged as in Part a of figure 60, the transcription of the binary number is not too difficult, but there is a high probability of error, especially if the number contains many bits. Arrangement as in Part b facilitates reading them directly in octal form, by the use of yellow and green indicators to group the binary bits into octal characters. A further improvement is made in Part c, by providing a visual indication of the octal weight of each binary bit and thereby increasing the speed and reliability of data translation and transcription by the human mind.

A similar logical argument has been formed for the arrangement of the register read-in switches, of which there may be as few as twelve or as many as twenty-two.

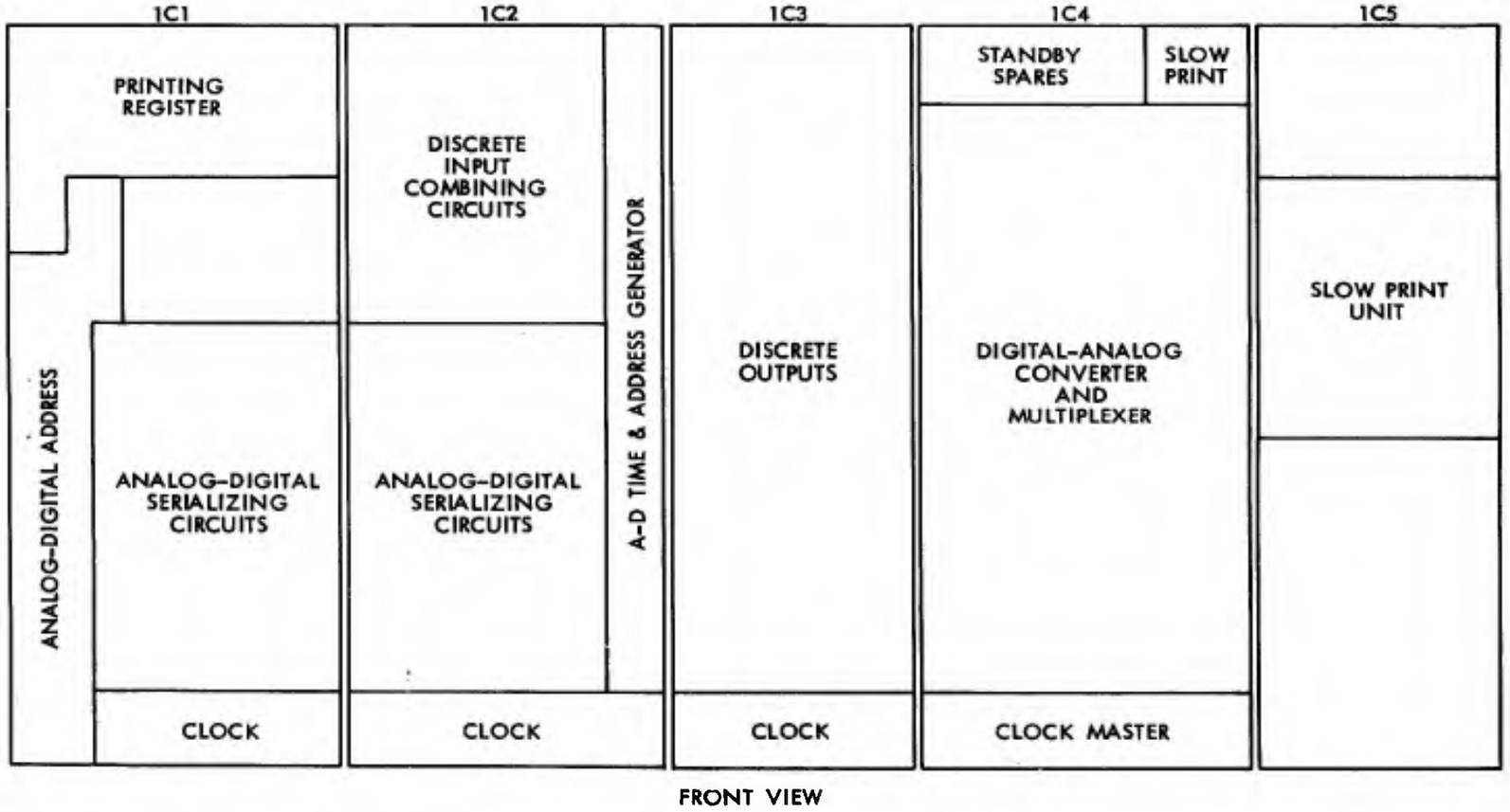


Figure 57. Layout of Input-Output Cabinet

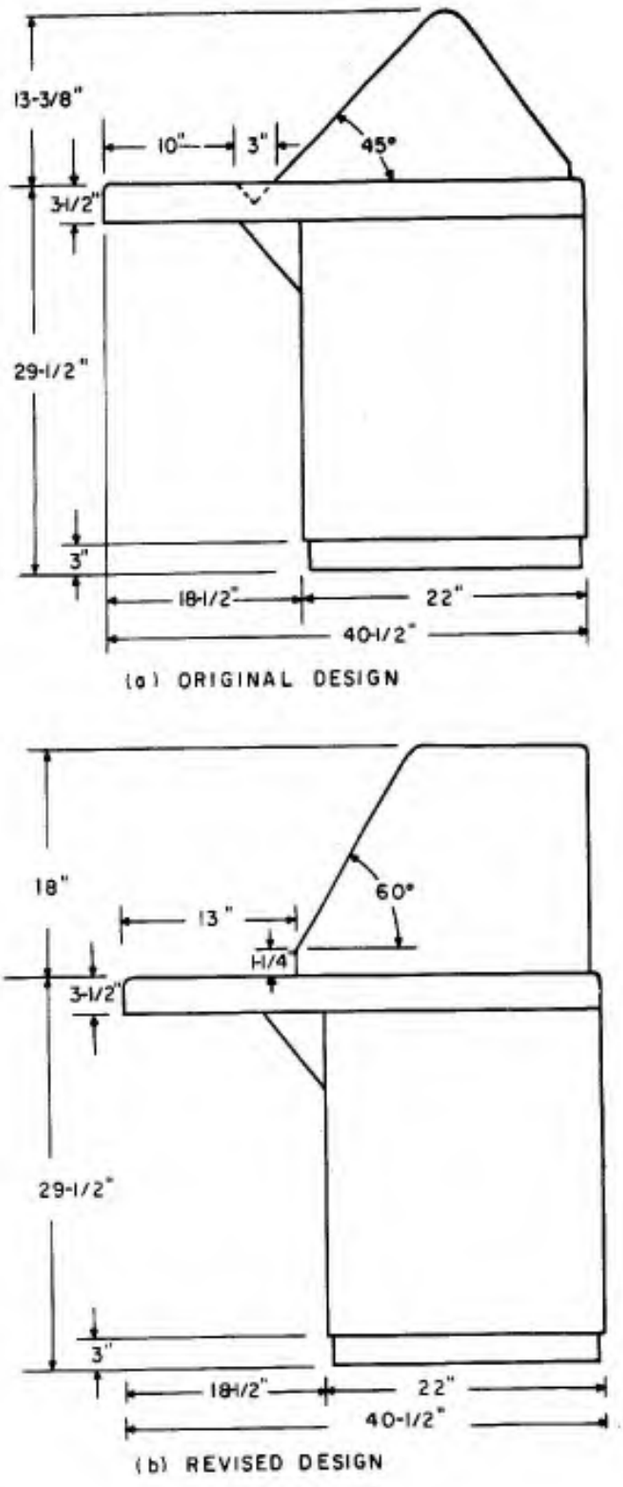


Figure 58. Original and Revised Computer Console Designs

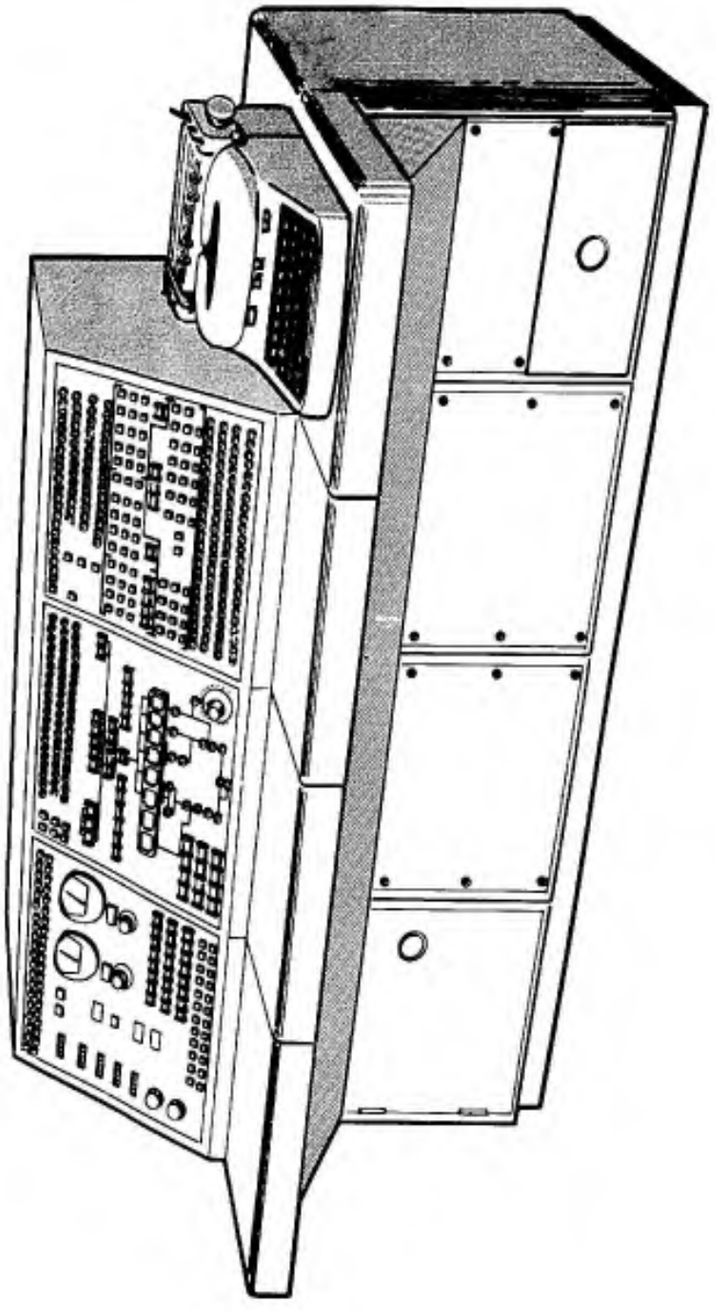


Figure 59. Computer Console

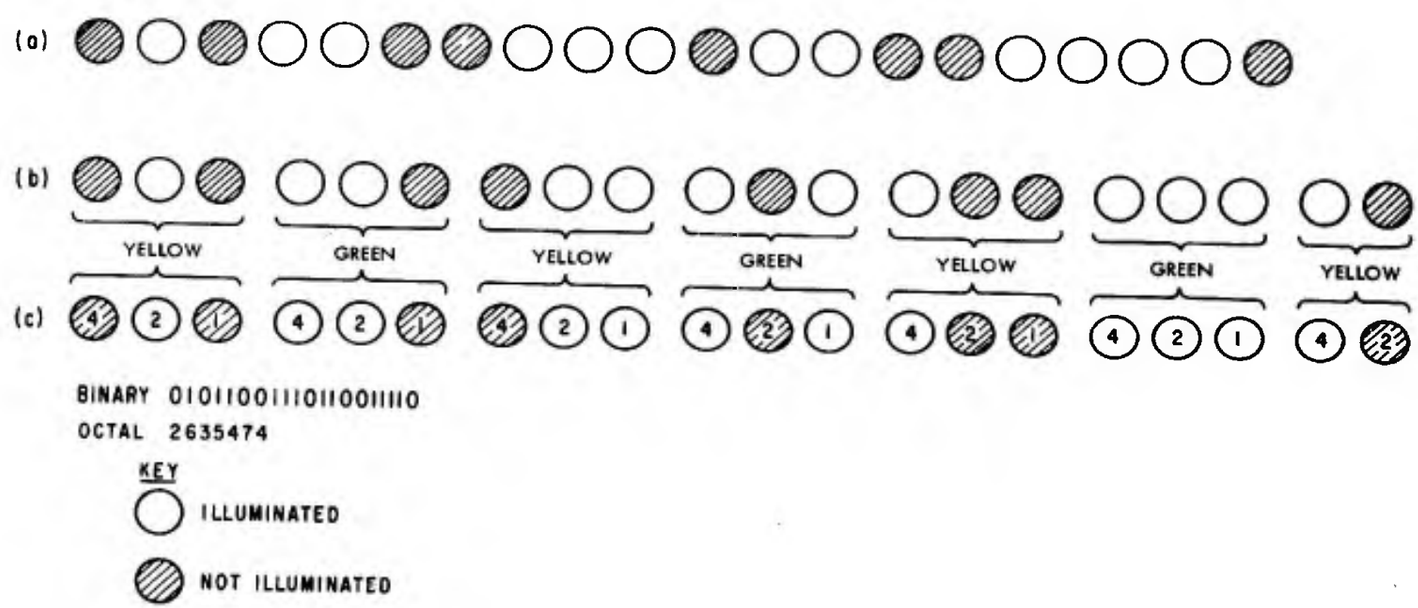


Figure 60. Indicator Lamp Arrangements

Graphic examples of this argument are shown in figure 61. From Part a, it can be seen that translating the octal information into the appropriate switch positions is probably more difficult than using the binary representation of the same data, since there is no aid provided for grouping the binary switches into octal groups. The representation of Part b is an improvement, but still lacks clarity. The representation of Part c, as was true for the indicator lamps, is the best; however, one unfavorable characteristic common also to Parts a and b remains: the switches are placed in a long horizontal row, making manipulation more difficult. To overcome this difficulty, the octal groups or triads were placed vertically as in Part d, thereby reducing the length of the register switch bank.

All switches are of the pushbutton type, whether maintained contact or momentary contact, with lighted buttons to indicate the position of the switch. Because of the large number of switches on the panel, it was determined that lighted pushbutton switches would be more desirable than toggle switches for the following reasons:

a. They are easier to operate, requiring a force in only one direction.

b. They provide a relatively large control surface, allowing markings to be placed right on the switch.

c. Illumination and color coding simplify identification and operation.

d. The switches used for setting up new information can be arranged in adding-machine keyboard fashion.

4.8 Power Supplies and Power Control

4.8.1 A-C Power

The input power to the computer is 208/120-volts, 3-phase, 60-cycle, 4-wire Y. This a-c power is subdivided into four types according to use. (Figure 62).

4.8.1.1 Utility Power

This portion of the a-c line is distributed to the various cabinets to provide 120-volt single-phase utility power at the convenience outlets.

4.8.1.2 Unregulated Power

This portion of the a-c line is distributed to the various cabinets to provide 208-volt three-phase power for the blower motors. Each cabinet has a circuit breaker ahead of the blower to protect the line in case of motor-winding short circuits.

4.8.1.3 Regulated Power

This portion of the a-c line is regulated by three single-phase a-c line regulators to provide three single phases of regulated 120 volts to the filament transformers in the cabinets. The filament power is applied in two steps when the computer is first turned on, by using a line-dropping resistor in each line of the regulated 120-volt lines, and shorting them out after a time delay of approximately 40 seconds. This somewhat gradual application of the filament power is expected to increase the life of the vacuum tubes by removing the thermal shock on each tube if full filament voltage were applied instantaneously.

4.8.1.4 Delayed Power

This portion of the a-c line is applied to the d-c power supplies as 208/120 volts, 3 phase, 4-wire Y. It is delayed in order to allow all vacuum tubes in the computer sufficient time to warm up before d-c power is applied.

4.8.2 D-C Power

D-C power is supplied independently to each cabinet from the power supply. The prime reason for these independent cables was to allow the use of smaller gauge wire without increasing the voltage drop between the supply and the load, due to cable

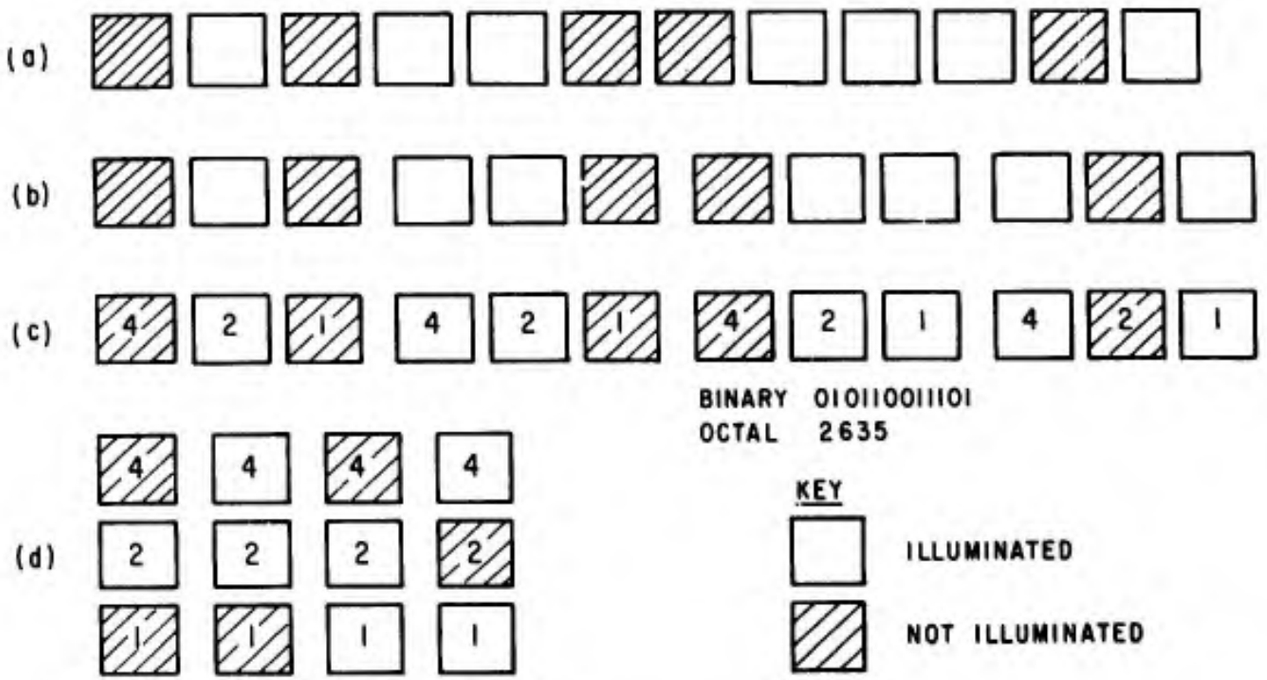


Figure 61. Read-in Switch Arrangements

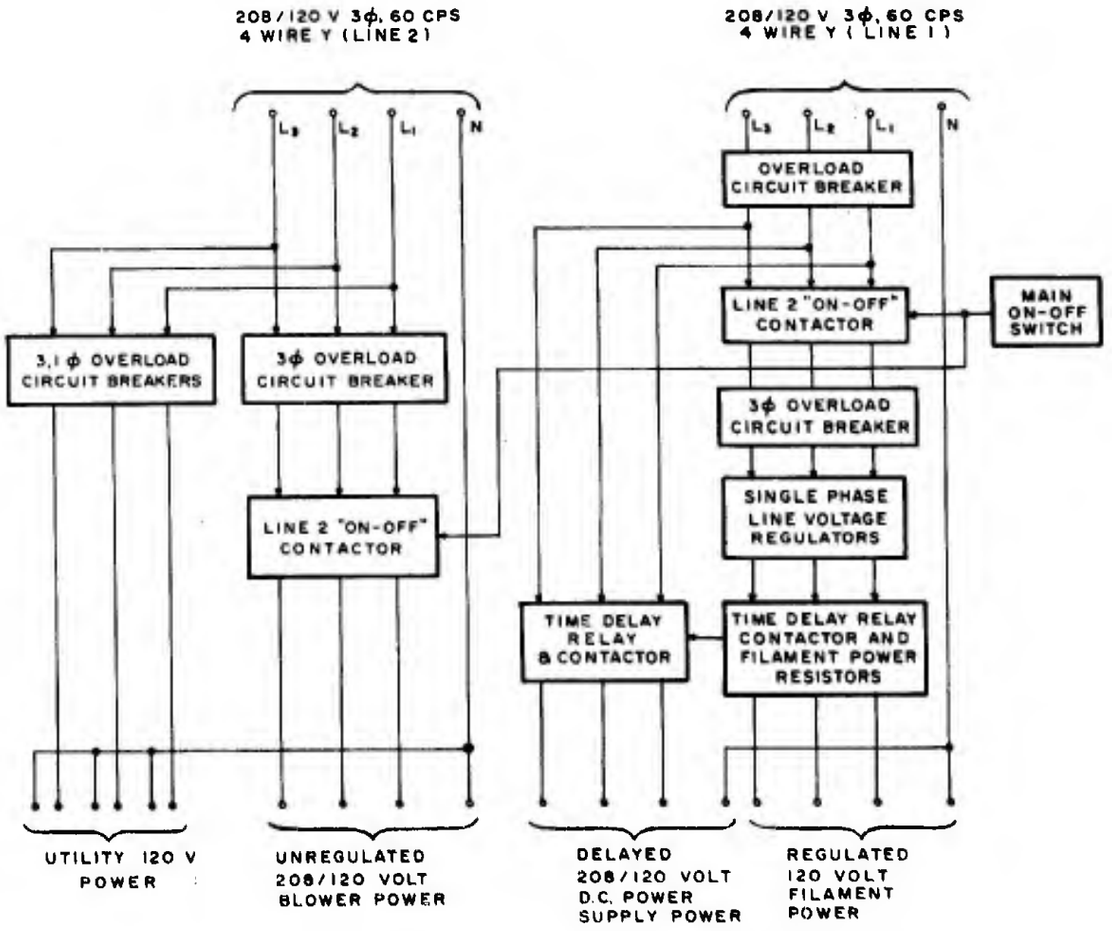


Figure 62. Block Diagram of A-C Power System

resistance and high currents that must be supplied. Similarly, to minimize the fluctuations of ground potential due to heavy transient currents that may occur during computer operation, the d-c power supply grounds or returns are not commonly joined. Separate ground wires for each supply are carried as far as the distribution point, where they are grouped together to form three independent grounds. These independent lines are carried to each cabinet, where they are commonly joined; cabinet ground is made at this same point. In addition, the cabinets of the system are interconnected to form one continuous cabinet ground. Through this rather complex scheme, it was hoped, a clean ground system would exist in the computer.

As d-c power control, a large contactor is located in each cabinet. If for any reason any of the d-c or filament supply voltages should be missing, this contactor opens, automatically disconnecting all d-c power from the cabinet.

In case of any power failure, a-c or d-c, major or minor, an alarm is sounded.

4.8.3 D-C Power Supplies

Seventeen magnetically regulated d-c power supplies supply all d-c voltages required by the computer (Table XI). A simplified block diagram of a typical three-phase magnetically regulated supply is shown in figure 63. An item of interest is the use of a fuse shunted by an indicator in series with each bank of storage capacitors. If a capacitor shorts, the fuse blows and the indicator lamp indicates the blown fuse; however, the power supply does not shut down. It continues to operate, but with higher ripple and poorer regulation. Thus a shorted capacitor will not cause unscheduled computer downtime.

4.9 Computer Unit Testing

4.9.1 Arithmetic Unit

The first phase of unit testing consisted of checking the wiring of the unit, which required two men for approximately three weeks. During the next phase, which lasted one to two weeks, corrections were made of the wiring mistakes detected during the first phase. The third and final phases consisted of the actual operational testing of the unit.

For testing purposes, the Arithmetic Unit was subdivided into a number of logical sections: the Multiplicand-Divisor register, the stages of the Accumulator, the Control Dispatcher lines, and the Clock Pulse system. Each subsystem was thoroughly checked before any packages were inserted for the next stage of operations. During the dynamic testing a number of problems were encountered.

4.9.1.1 Clock Repeater Oscillation

When all Clock Repeater packages were inserted for distribution of the clock pulses, severe oscillations resulted. After eliminating the ground system, power supplies, lead dress, supply voltage decoupling networks, and cross-coupling between the etched conductors on the Clock Repeater package as possible causes, it was determined that the parasitic suppressor grid resistors of the repeater cathode followers were inadequate.

4.9.1.2 Clock Pulse Distribution

Ringing appeared on the clock pulses at the ends of the distribution lines from the load-driving Clock Repeater packages. Investigation showed that the length of these lines (10 to 18 feet) was the cause. By altering the distribution of the clock pulses so that no clock line was longer than ten feet, and by terminating the end of each line with a resistor-diode network, the ringing was eliminated and the transmission delays along the lines were decreased.

4.9.1.3 Short Delay Lines

It was observed during the testing of the stages of the Accumulator, that the output signals from the short delay lines in many instances were marginal: that is, distortion and attenuation of the pulses passing through the delay lines were so great

TABLE XI

MAXIMUM D-C POWER REQUIREMENTS FOR UDOFT COMPUTER

Nominal Supply Voltage (Volts)	Current (Amperes)	Total Regulation*	
		20% To Full Load	0 To 20%
-4.5	25	2.5%	5.0%
-10	15	2.5%	5.0%
-20	115	1.0%	2.0%
-150	1.0	0.5%	1.0%
-150	7.0	1.0%	2.0%
-300	0.75	2.5%	5.0%
+6	25	2.5%	5.0%
+10	0.25	0.5%	1.0%
+20	40	2.5%	5.0%
+48	2.5	2.5%	5.0%
+80	25	2.5%	5.0%
+150	1.0	0.5%	1.0%
+150	25	2.5%	5.0%
+250	3.0	2.5%	5.0%
+80MCV ±25V	1.0	2.5%	2.5%
+150MCV ±25V	0.7	2.5%	2.5%
-150MCV +60V, -10	17	2.5%	5.0%

*Including peak-to-peak ripple and noise

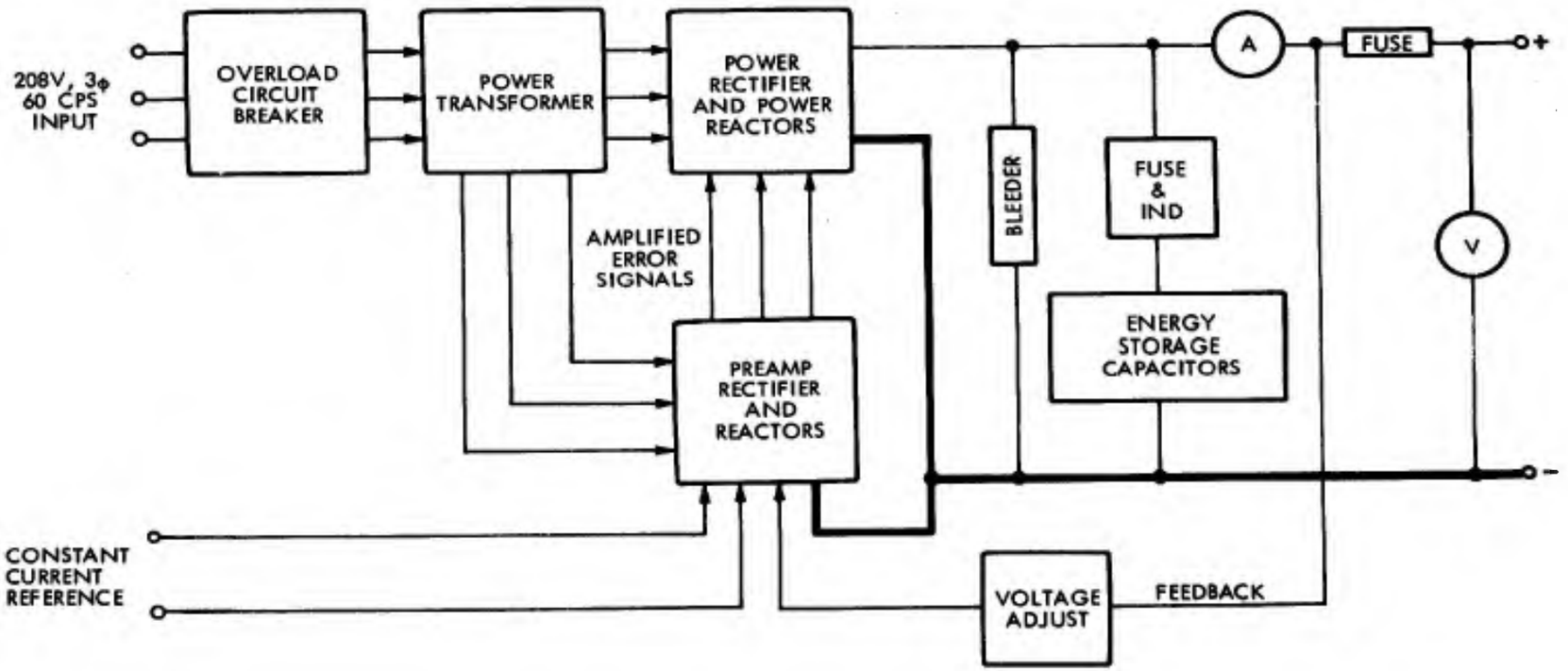


Figure 63. Block Diagram of Magnetically Regulated Power Supply

that the output pulse was barely able to control the succeeding circuit. The problem was the result of reflections at the end of the delay line.

The characteristics of the delay lines were investigated; some exhibit capacitive characteristics, while others exhibited inductive characteristics. The difference in the reactive characteristics of these lines was found to depend on dressing of the ground braid of the distributed delay line at the terminating end. If the braid extended over the helically wound center conductor and was taped tightly, the delay section was capacitively terminated; if the braid did not extend over the center conductor, the delay section was inductively terminated.

It would have been highly inefficient to correct this by reactively trimming each delay line section; the problem had to be solved by accepting the physical quirks of the lines and attempting to overcome their effects. This was accomplished by increasing the pulse amplifier plate and screen voltages from +75 volts to +80 volts, to supply larger driving pulses and to decrease the relative amount of delay line distortion, and by providing a form of termination between the two 0. 4 delay line sections that comprise the 0. 8 delay line.

4. 9. 1. 4 Long Delay Lines

During the testing of the G-Register stages, it was found that the 5. 0 clock period and the 5. 6 clock period delay lines, specified to be 4. 8 and 5. 4 clock periods long respectively, could not be trimmed by the addition of 0. 2 clock period delay lines. Pulse distortion and attenuation in the long delay lines were too great to allow connecting the delays in series. The addition of repeater pulse amplifiers compensated for the difference of 0. 2 clock period and reshaped the distorted pulse to one with the desired characteristics.

4. 9. 2 Control Unit II

This was the second computer unit to be delivered for test. Testing followed the same general plan established for the Arithmetic Unit: namely, wiring check, wiring correction, and unit test. Because improvements had been made affecting this unit as a result of the extensive dynamic testing conducted on the Arithmetic Unit, no significant problems were encountered.

4. 9. 3 Control Unit I

This was the third computer unit to be delivered for test. Again, no significant problems appeared. However, Control Unit I could not be completely tested, dynamically, without the thirty timing pulses of the five-microsecond minor cycle. Since the timing pulses are generated in Control Unit II, the first major unit interconnection was made between the two control units. When they had been tested together, all the necessary controlling functions for the Arithmetic Unit were available; the Arithmetic Unit was then connected to the Control Unit I – Control Unit II combination.

4. 9. 4 Main Frame Test

The next step was the dynamic testing of the Main Frame, consisting of the Arithmetic Unit, Control Unit II, and Control Unit I. Since the Computer Console and the Memory Unit were not available, it was necessary to substitute for these two units a simulator panel which contained all the functions otherwise normally available. As a result, the execution of the following instructions were checked on a single instruction basis: Add, Absolute Add, Clear Add, Clear Absolute Add, Subtract, Absolute Subtract, Clear Subtract, Clear Absolute Subtract, Multiply, Multiply Add, Divide, Shift (Left and Right), and Shift Add. Testing the remaining instructions which involve Program Control, Storage, and Input-Output, had to be deferred until both the Memory Unit and the Console were available.

4. 9. 5 Memory Unit Test

After the Memory Unit wiring had been checked and corrected, dynamic testing was begun in very limited form. Enough packages were inserted to allow the operation of a complete read-write cycle for a single core in a single memory plane. The results were encouraging in that the operation of the single core was wholly adequate.

Eventually packages were inserted to allow the testing of a single core in each of the other memory planes. Finally the packages for the memory addressing function were inserted, thereby allowing the testing of whole words anywhere in either memory.

At this stage considerable difficulty was encountered with the operation of the number memory. With the Sense Amplifier and the Inhibit Driver Amplifiers adjusted properly for reading and rewriting the contents of a particular number-memory storage register, it was not possible to read out of or write into another storage location without making minor adjustments to the two amplifiers. Also, within the same storage location, variation in the ONE and ZERO content of the stored number affected the reading out of the number. It was quite certain that a wrong temperature did not cause the hindrance from location to location or within the same location, since the air conditioner was functioning properly. This avenue had been investigated because the first diagnosis indicated that the cores were switching much too rapidly. There was also a noticeable increase in the noise level on the sense winding, caused by partially selected cores and by selected cores storing ZEROs.

Examination of the X-Drive and the Y-Drive lines threading the planes revealed drive current variations as the pattern of ONEs and ZEROs in the stored word was changed. The drive current is affected by the reactive load on the drive line, and this reactive load varies proportionately with the number of ONEs being stored. Thus the driver has a high impedance load when a majority of ONEs is being stored, and a low impedance load when a majority of ZEROs is being stored. As a result, the variable load on the driver causes variation in the drive current waveform generated by the driver. When the impedance is low, the driver output is essentially underdamped, allowing considerable overshoot on the leading edge of the current waveform. If the overshoot is great enough, there is sufficient drive in this half-select signal to cause only partially selected cores to switch, and to cause greater noise output from cores storing ZEROs. In addition, it causes the one wholly selected core to switch much more rapidly. The overshoot under worst load conditions was found to be in the order of 60 percent.

The addition of a shunt RL circuit in the supply line between the constant current sources and the cathodes of the Gate Generator Tubes (type 5998) corrected the overshoot. The effect was that of introducing a fixed reactive load without causing current waveform deterioration, such that variations in the reactive load of the cores became a less significant load determinant. Effectively, the driver impedance was increased to make it more nearly a true constant-current source. Though overshoot could not be eliminated completely, it was reduced to approximately 10 percent in the case of the worst load; this reduction appeared adequate for proper memory operation.

A similar change was incorporated in the Instruction Memory: no difficulties were encountered during its unit-testing.

4.9.6 Input-Output Unit

Delivery to test of the last major computer section, the Input-Output Unit, was delayed a number of months beyond the scheduled date, due to more pressing items associated directly with the Main Frame. To prevent this lateness from harming the overall program, the unit test phase was accelerated by eliminating the dynamic test portion. Rather than to check the unit independently of the rest of the system, it was tested dynamically as an integral part of the computer system.

Testing of the Discrete Outputs revealed that, although ample signals were routed to the console for driving the discrete output indicator lamps, the d-c bias level was too positive. This problem was resolved by introducing voltage dividers at the inputs to the transistors driving the console indicator lamps.

The main problem in the input-output area concerned the actuation of the slow-speed printer. The operation of the steering switch which selects the stages of the registers to be printed out was inadequate and required modification. In addition, actuation of the stepping switch and the binary-to-octal converter relays introduced transients into the system sufficient to affect some of the registers. The judicious use of decoupling networks and the terminating of the register outputs at the stepping switch resolved these problems.

4. 9. 7 Computer Console Unit

Only a minimum amount of testing, other than checking the wiring, could be performed on the Console Unit by itself. Therefore the unit was connected into the system.

Immediately apparent were problems caused by the contact bounce of the Start switch, the Manual Memory switch, the Sequence Counter Reset switch, and the Interval Timer Reset switch. The problem caused by the latter two switches was solved by utilizing a relay with sequential transfer contacts. In this way, the clear signal would be removed prior to removal of the set signal, thereby insuring proper resetting. The problem caused by the Start and Manual Memory switches differed radically in that, in each, one switch contact actuated two different pulse amplifiers, and unless the input and triggering characteristics of the two amplifiers were identical, switch bounce could cause improper operation. The solution required both changes in the computer logic and the addition of relays with mercury-wetted contacts.

Another problem area, somewhat more general in scope, concerned undesirable interaction between console switches and the computer circuitry which drives the console indicator lamps. The solution was the separation of switch voltage buses and indicator voltage buses, and sufficient decoupling of each to minimize both the transients that occur as a result of switches being operated and the effect of these transients on the driving sources in the central computer.

4. 10 Computer System Testing

The first system test was that of the manual control of the two core memories from the console. After this had been carried out successfully, the instructions that constitute the computer instruction repertoire were checked on a single-instruction basis. Other than a major logic problem concerning the Transfer on Overflow (TOV) instruction, no serious problems were encountered.

The next stage of testing involved loading an entire program and test computer operation in the fast mode. The first difficulty encountered during this phase was incorrect program read-in. The causes for improper card read-in were traced to capacitive coupling between the high impedance card reader input lines and, again, switch contact bounce in the card reader. The problem was resolved by introducing low impedance terminations to minimize the effect of capacitive coupling and decoupling networks to filter out the high frequency aspect of the card reader input signals which gave rise to the large signals being coupled into physically adjacent signal lines. After the two types of networks had been added, no trouble was encountered in obtaining reliable read-in from the card reader. However, a simple program could not be made to run. The problem lay still in the operation of the memories, primarily the Number Memory. The sources of trouble were traced to excessive noise on the ground bus, causing spurious triggering of the rewrite flip-flops; insufficient sensitivity of the output flip-flop to the strobe pulse combined with the output of the sense amplifier; and the rapid development of cathode interface impedance in the type 5998 core driver tubes. The problems were eliminated by rewiring the Memory Unit ground bus system, decreasing the flip-flop cathode bias, and replacing the 5998 tubes with type 7236 tubes.

Finally, on 22 January 1959, successful operation of the computer system under program control was attained. The control program was the Number Memory Test Program. Although the program consists of only forty or fifty instructions, it is executed for each number memory register. Thus, the gross program length is approximately 200, 000 instructions. Since the instructions are predominantly those requiring only five microseconds, the program was executed in approximately one second.

The next step was to attempt the Computer Diagnostic Program. During the time operation of this program was attempted, less obvious and logic errors were discovered. The following are some of the malfunction or improper conditions which occurred in the computer as the result of wiring errors:

a. Incorrect Accumulator sign generated for product if multiplicand negative

b. NOP order decoded improperly

c. Number Memory parity formation incorrect when 02TRR = 1 (The second least significant bit of the transfer register was a 1)

d. Incorrection determination of NSN, which is essential for MXDO instructions; NSN is the amplifier which anticipates the next sign of the accumulator

e. G-Register Error indicator failing indicate errors

f. Two discrete input switches controlling the same number-memory register

g. Number parity error incorrectly determined

On 23 April, 1959, the computer executed successfully the complete Computer Diagnostic Program. Ironically, the last problem was a loose plug-in pulse amplifier package in the Arithmetic Unit. This pulse amplifier was the last stage of Dispatcher Line 5, which controls the shift-left operations. Once this fault had been cleared, the Diagnostic Program was run for two and one half error-free hours, before operation was terminated arbitrarily.

Due to the noticeably unreliable operation of the Number Memory, it had been decided to revise the Number Memory Test Program in order to yield a more stringent test of the memory. The revised program checks each number memory location for its ability to store forty different bit configurations; twenty configurations of a single ONE, and twenty configurations of nineteen ONEs. The program consumes approximately nineteen seconds of operating time.

The program is quite simple and was so prepared in order to eliminate as many programming problems as possible. The brute-force aspect of the program is borne out by the fact that it required but two hours to debug the program completely and to make it operational. During the latter part of April, after the successes with the Diagnostic Program and the revised Number Memory Test Program, the analog output system was checked by means of the Analog Output Precision Test Program. The program causes a test servo to assume ten discrete positions, 32° apart. The position of the shaft (encoder) is read into the computer via an analog input channel, and the difference between the actual shaft position and the desired shaft position is determined and stored for print-out when the routine is complete. A number of output channels were checked and adjusted as necessary. Due to the element of time, not all output channels were checked prior to the commencement of aircraft simulation program check-out on 5 May 1959. As time was available the remaining output channels were checked.

During the month of August 1959, at a time when the programmers were involved with an F-100A simulation program problem, the complex Instruction Memory Test Program was attempted. After a few program modifications, it was found that the computer could execute the program flawlessly, stopping only at IMAD 7171 due to a bad core in the seventeenth plane. This situation was remedied by replacing the defective plane.

As a result of the successful execution of the Instruction Memory Test Program, four service programs (Number Memory Test, Diagnostic, Number Memory Test – Revised, and Instruction Memory Test) were now available for periodic confidence checking of the computer. As time permitted, effort was applied to the debugging of the Number Memory Checkerboard Program. Ultimately the checkerboard program was debugged and added to the group of available computer service programs.

4.11 Trainer Modification and Static Test

To minimize the costs of developing the digital flight simulator system, the government provided to Sylvania two flight simulators: an F-100A Simulator Trainer, Type MB-3, and an F9F-2 Operational Flight Trainer, Device 2-F-13. Further, the government directed Sylvania to make use of the analog simulators, the use being limited only to the extent that all aerodynamic, powerplant, and aircraft systems calculations be performed by the digital computer. Thus extensive use was made of such items as the trainer cockpits, the instructor's stations, the operator's stations, the integrating servos and the radio-navigational aids.

The trainer cockpits were modified to allow their use with the digital computer; potentiometer transducers were replaced with shaft-position-to-digital encoders and a number of cockpit switches and indicators were disconnected from the analog computer and rewired to the digital computer. Similar modifications were made to the instructor's stations and the operator's stations. The analog computer integrating servos were modified to function as conventional positioning servos; however, a synchro transmitter was added to each servo which was used to position similar instruments in each cockpit. The radio-navigational aids sections of the analog devices were left unmodified. The majority of the parts and assemblies which found use in the final UDOFT system were obtained from the F-100A analog simulator.

Initial testing of the modified trainer cockpits was limited to a continuity check of the lines of communication between the trainer and the computer, and a static operating check of the transducers and the indicators with the trainer cockpits. To facilitate the check-out, a small test panel was developed which ultimately became an integral part of the trainer. (Figure 64.) The 24 toggle switches grouped in two rows of 12 switches each, at the top of the panel, simulate the discrete outputs (relay closures) from the computer; the ten indicators immediately below the discrete output switches and to the left of center are used to indicate the state of the ten-bit analog input derived within the trainer; the ten indicators to the right of center are used to indicate the states of single discrete inputs derived within the trainer. The test leads, the two rows of test jacks, and the two AN connector receptacles provide the means for connecting the test panel to the trainer. The 20 test leads are associated with the ten analog input indicators and the ten discrete input indicators. No test leads are associated with the 24 simulated discrete inputs; the reason will be apparent from the material that follows. The test points are connected permanently to the left-hand AN connector receptacle. This allows the "patching" of the indicator to the appropriate signal lead within the trainer cable assembly. The right-hand AN connector receptacle is not connected to the test jacks; it is wired permanently to the twenty-four discrete output switches. This was done because there are only 24 discrete output lines from the computer to the trainer all of them grouped within a single cable assembly. Therefore the right-hand receptacle is used only for checking trainer components actuated by discrete outputs from the computer.

In order to check the output from any shaft-position-to-digital encoder or trainer discrete input switch, the trainer cable assembly carrying these signals is disconnected from the computer's Input/Output unit and connected to the left-hand receptacle on the test panel. After determining, from a posted reference list, the designation of the connector pins carrying the desired signal, the appropriate status indicator is connected, via the test jacks to the connector.

The only items not checked directly by the test panel are the computer analog outputs. The instrument-positioning servos, which are activated by those outputs, may be checked either by connecting a variable d-c voltage source to the appropriate cable assembly connector pins or by using the manual servo-positioning facility which was retained when the F-100A simulator-integrating servos were modified to simple positioning servos.

4.12 Review

This documentary report cannot relate the pertinent facts of the development of the UDOFT system in a single logical series of report sections, because the development of the UDOFT system involved the concurrent development and integration of two major items, system hardware and system software. The three preceding sections have related the prominent aspects of hardware development, namely, the UDOFT digital computer. The two sections that follow relate the prominent aspects of software, or computer programming, development. At the conclusion of these sections, the two major items of consideration, system hardware and software are integrated and the remainder of the report is devoted to various aspects of the UDOFT system as a whole.

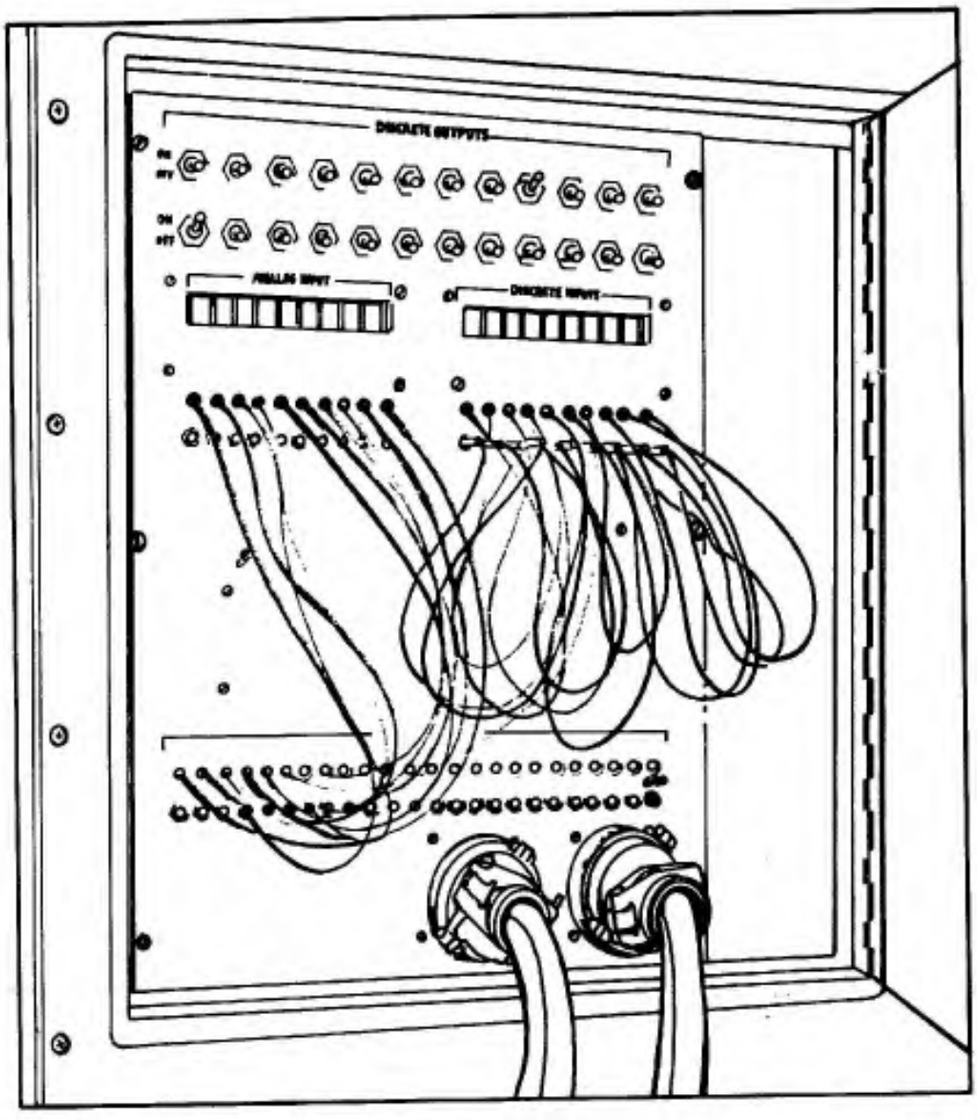


Figure 64. Trainer Static Test Panel

SECTION V

SIMULATION PROGRAM DEVELOPMENT

A review in the year 1962 of the development of the computer programs for the UDOFT computer is in many respects a study of the ancient history of the computer industry. The UDOFT programming tasks were initiated in 1956, comparatively early for attempting the real-time simulation of a dynamic system on a digital computer. Many of the problems encountered in preparing a satisfactory flight simulation program have no similarity to those encountered in the more usual applications of digital computers; a discussion of these problem areas is necessary if one is to appreciate their implications.

The following sections delineates the history of the programming task for the UDOFT computer. Detailed descriptions of selected aspects of the simulation program for the F-100A aircraft are presented in Section VI, RESULTANT PROGRAM.

5.1 Simulation of the UDOFT Computer

At the outset of the UDOFT project, the need for a readily available digital computer to simulate the UDOFT computer was apparent, because the operational flight-simulation programs were scheduled to be prepared and debugged shortly after the computer was made available. This meant that initial check-out and trial verification runs had to be conducted on another digital computer. Further, UDOFT computer test and exercise programs were being prepared to facilitate the verification of computer operation. Confidence had to be established in both the computer and the associated programs before they could be of use.

Accordingly, a program to simulate the UDOFT computer on an IBM 704 Computer was prepared. The IBM 704 was selected primarily because it was probably the most popular general-purpose digital computer readily available throughout the country at the time the UDOFT simulator program was undertaken. This simulator program, known as PSEUDOFT (PSEUDO-UDOFT) is a control program that instructs the IBM 704 to execute a program written for the UDOFT computer in UDOFT language.

The original PSEUDOFT program was divided into three main sections: the loader, the simulator, and the reporter. The loader controls the insertion of UDOFT instruction and number words from magnetic tape into the 704 magnetic core memory. The simulator, which forms the bulk of the PSEUDOFT program, translates each UDOFT instruction into 704 instructions which then cause the desired operation to be performed. Finally, the reporter interrogates each instruction to determine if any output is required, and if so, causes an on-line print-out of information on magnetic tape. The rest of the data is printed off-line.

The intent of the program was to simulate all operations of interest to the programmer, rather than all pulse-by-pulse actions of the UDOFT computer. Simulating the UDOFT instructions required subroutines for the 704 which varied from two to thirty-five 704 instructions, those for the more basic UDOFT instructions and for the UDOFT Shift and Add instruction. As a result of this and the speed disparity between the 704 and UDOFT, the time ratio for simulation was approximately 150:1 when using PSEUDOFT to run a UDOFT routine. This ratio increased even more when output was required from PSEUDOFT.

In preparing the PSEUDOFT program the problem was not so much in deciding how to do a certain thing but rather what needed to be done. The preparation of the first version of PSEUDOFT was hampered by the limited 8000 word memory capacity of the 704. This imposed heavy restrictions on the programs to be run on PSEUDOFT, since the full 8,000 words of the 704 memory would not be available to simulate the UDOFT computer memory. Furthermore, the memory limitation complicated the coding of PSEUDOFT, because care had to be taken to conserve the 704 memory in order to minimize the restrictions on the UDOFT programs. Efficiency of the PSEUDOFT program was critical also, since many passes through PSEUDOFT would be required to effect a few seconds of real-time simulation. This requirement became more important when it was realized that the UDOFT routines had to be run several times on PSEUDOFT before they were considered adequately debugged.

When the 709 replaced the 704, the same PSEUDOFT program continued in use. However, it was not as efficient because programs written for the 704 were not directly applicable to the 709. Since many users of the 709's were former 704 users, an input compatability program was made available by means of which 704 programs could be read into the 709. The resultant inefficiency was tolerable. However, the use of the 704-PSEUDOFT program on the 709 would have continued had it not been for a desire to augment the PSEUDOFT program with other program preparation and checkout aids. Since these programs were to be written for the 709, it was decided to rewrite PSEUDOFT for the 709 to achieve maximum efficiency.

The 704 PSEUDOFT program has not been discarded. It was used recently by a user of the UDOFT computer system who had a 704. The use of PSEUDOFT allowed him to debug his UDOFT programs rapidly because PSEUDOFT was able to provide him with data concerning the progress of problem solution.

If the computer to be used in a system is the first of a kind, a simulator program such as PSEUDOFT is essential. Also, if the computer is highly specialized and its input-output facility is severely limited, as is the case with the UDOFT computer, a simulator program is highly desirable.

The use of PSEUDOFT had several side effects which are mentioned in a succeeding section, "Checkout and Test." During the course of preparing these programs, various additions were made to the PSEUDOFT program to increase its capability. The finished PSEUDOFT program offers tracing, timing, variable output, overflow indication, and programming-error detection capabilities.

5.2 Use of Automatic Programming Techniques

The use of automatic programming techniques was essential to the successful completion of the operational simulation programs for UDOFT. It would have been virtually impossible to complete the programs without the help of an assembly program. This point is emphasized because, although the desirability of an assembly program is self-evident now, it was by no means obvious at the time the work was begun. A brief history of this aspect of the programming task follows.

At the beginning of the programming task it was assumed that all programs were to be written in binary. Thus, if an instruction read "clear and add the contents of number memory register four," the instruction would be entered onto a UDOFT coding sheet as 340004. This required a knowledge of the binary (or octal) representations of the UDOFT order code. It required also that the numerical data, such as might be stored in "number memory register four" would have to be entered in binary (or octal) form. The task of converting decimal data to binary would be done manually. Accordingly, PSEUDOFT was written to accept program and numerical data in octal form.

During the summer of 1958, the programming task became overpowering. Valuable time was lost to converting decimal data to octal form. As a recourse, the standard IBM 704 Assembly Program (SAP) was modified to convert numbers from decimal to fixed point binary, using a scale factor selected by the program. As an example, consider the conversion of Sin 4° to octal. Without any scale factor indication, Sin 4° = $(0.069756)_{10}$ = $(0.043556)_8$. With a scale factor of B1, the octal representation will be shifted one binary place to the left (equivalent to shifting the radix point one binary place to the right) resulting in $(0.0216670)_8$. Thus, $(0.069756)_{10}$ B1 is transformed into $(0.0216670)_8$.

In this way, all numerical data was assembled. The modified assembly program was used for this purpose only. PSEUDOFT was, in turn, modified to accept the binary cards processed by the assembly program.

Up to this time no thought had been given to the preparation of binary input cards for the UDOFT computer. The UDOFT punched card format is unique, and as such posed a problem. This problem was alleviated by preparing a program that would manipulate bits, as they appeared in 704 core memory, and punch them out on cards suitable for entry into the UDOFT computer. This meant that anything that had been used as input to PSEUDOFT, whether numbers assembled by the modified assembly program or program instructions coded in octal, could be punched on cards in accordance with the UDOFT punched card format.

These modest innovations were a great advance, but still did not alleviate the problem of writing instructions; instructions still were coded in binary form. In the spring of 1959, it became apparent that it was possible, with very little effort, to further modify SAP, the 704 Assembly Program, so that it would assemble all parts of the UDOFT program. From input consisting of mnemonic coding using symbolic addresses, the modified SAP could produce absolute coded programs in accordance with the UDOFT format, suitable for running on PSEUDOFT and for subsequent punching on UDOFT format cards. Writing a working assembly program was accomplished in a few weeks. This first assembly program known more commonly as UDAP (UDOFT Assembly Program), provided two forms of output; a 704 binary deck suitable for use with PSEUDOFT and an assembly listing in UDOFT language.

The conversion of 704 PSEUDOFT binary cards to UDOFT binary cards required the use of the punch program mentioned previously in this section. In order to eliminate this extra step, UDAP was modified again to punch UDOFT binary cards suitable for direct entry into the UDOFT computer. In addition, the assembly was modified further to write a magnetic tape, an equivalent of the 704 PSEUDOFT binary deck. This modification eliminated the time-consuming task of on-line punching of cards for PSEUDOFT; further, the resultant tape could be read into the 704 more rapidly than the punched cards. A final modification was the incorporation of parity determination for the assembled UDOFT instruction and number words and the subsequent print-out of parity as part of the UDOFT assembly.

The designation of the program, since it had been modified to such an extent, was changed to UD2 (considering the original UDAP as UD1).

Shortly after UD2 had been prepared and was being used, the decision to rewrite PSEUDOFT for the IBM 709 was made. At this time a review of the many utility programs was undertaken. Since the assembly program had become such a powerful tool, and the need for PSEUDOFT was declining, it was decided to revamp UD2. However, now that the 709 had replaced the 704, the 709 Assembly Program (9AP) was to be modified to cater to the needs of UDOFT, and continued modifications of the 704 Assembly Program (SAP) were terminated. The resultant UDOFT assembly program became UD3. Aside from using the 709 more efficiently, UD3 incorporated the forbidden sequence tester which until this time had been a separate program used in conjunction with PSEUDOFT. A still later version UD4 is now in use. UD4 is used with the 7090 computer, and prepares the 12-word per card binary cards.

A sample print-out of a UDOFT routine assembled by UD3 is presented in figure 65; the example selected is the Governing Control Program for the F-100A simulation program. The information appearing in four of the first five columns (IMAD, R, OT, NMAD) is punched onto UDOFT binary cards; the remaining columns are printed on the assembly listing to facilitate visualization of the assembled program.

The use of an assembly program to aid in program-preparation for future simulators is strongly recommended. A compiler for flight-simulation programs may even be more useful; it is well known that the computer industry as a whole is moving in the direction of compilers. However, in making the decision on compilers, the following aspects must be considered:

1. The validity and efficiency of the object programs produced by a compiler.

2. The ratio of time spent in writing a compiler to time saved in writing flight-simulation programs.

There is no question in the mind of most people in the computer field with regards to the usefulness of compilers in general. Most experienced people say that compilers cut programming time to about one quarter of the time required for symbolic coding of the type suitable for assembly programs. On the other hand, for a large system, compilers have an inherent disadvantage in that it is difficult to insert temporary check points or to make small changes to a completed program. It is also difficult to minimize running time, or even simply to keep track of running time. These disadvantages become more pronounced when preparing flight-simulation programs; however, they may not be intolerable.

The time required to write a compiler is another disadvantage. There is no way to simply and easily modify one compiler so that it can be used on a different machine

```
                    0001                ORG     1
0001  P       07    0001    GCENT       SENIT   *
0002          13    7667                SIT     GCN7
0003          37    0402                CLAS    LWT02   ZERO MODE DISCRETE INPUT
0004  P       22    0041                TOM     GCRT2   ROUTER NO. 2
0005  P       37    0430                CLAS    LWT30   FREEZE DISCRETE INPUT
0006  P       22    0063                TOM     GCRT3   ROUTER NO. 3
0007  P       34    0051                CLA     CRASH   CRASH FLAG
0010  P       22    0063                TOM     GCRT4   ROUTER NO. 4
                                                        ROUTER ONE FOR GOVERNING CONTROL PROGRAM
0011          36    7666    GCRT1       CLAA    GCN6
0012  P       24    0000                SHLA    0       USE GREG FOR COUNTER
0013          30    7660                ADD     GCN0    TEST FLAG TO SEE IF ROUTE
0014          31    7661                SUB     GCN1    IS ALREADY SET
0015          06    0037                TOZ     GCEX1   EXIT IF IT IS
0016  P       11    0000                TAN     0
0017          37    7661    GCLP        CLAS    GCN1    STEP DOWN GREG AND TEST
0020          30    1000                ADD     GREG    TO SEE IF ROUTING FINISHED
0021          06    0031                TOZ     GCEX2   EXIT
0022  P       24    0000                SHLA    0       PUT COUNT IN GREG
0023      R   14    7741                NOP*    GCTBL+1 COPY TABLE 1 INTO
0024          34    0000                CLA     0       TABLE 2
0025      R   14    7700                NOP*    GCEX
0026  P   R   23    0000                STO*    0
0027          11    0001                TAN     1       STEP TALLY BY 1
0030  P       03    0017                SCR     GCLP    GO BACK
0031  P       34    7740    GCEX2       CLA     GCTBL   SET RETURN FROM LAST
0032      R   14    7700                NOP*    GCEX    SUBROUTINE BACK TO GOVERNING
0033          23    0000                STO     0       CONTROL ROUTINE
0034          34    7661                CLA     GCN1    SET FLAG FOR THIS ROUTER
0035  P       11    0000                TAN     0       CLEAR TALLY REGISTER
0036  P       23    7660                STO     GCN0
0037  P       34    1000    GCEX1       CLA     GREG    CLEAR G REGISTER
0040          02    7700                SCRNM   GCEX    EXIT FROM GOVERNING CONTROL
```

NOTE: The letter P indicates the presence of a Parity Check bit, and the letter R indicates the presence of a Relative Address bit.

Figure 65. Sample Print-out of Assembled UDOFT Program

whose organization and order code are substantially different; no such minor operation as was performed on SAP to create UDAP would make FORTRAN into a compiler for another computer.

In the long run it might be useful to write a compiler for flight-simulation programs, perhaps a specially-written compiler. It seems preferable, however, to wait until generally acceptable techniques of digital flight-simulation have been evolved before the use of a compiler is attempted. This conclusion is based partly on the time required to produce a compiler, partly on the feeling that simulation programs should be written carefully at first, and that from this process, guidelines for a special-purpose compiler can be expected to emerge.

5.3 Checkout and Test

Checkout of the operational flight-simulation programs was performed in two stages; first, on the 704 (or the 709) using PSEUDOFT and some 704 (or 709) programs which will be described, and second, on the UDOFT computer itself, essentially without any programming aids. The primary debugging tool was PSEUDOFT, but it was not a universal tool because of costliness and the simulator program's inability to handle interface problems. The absence of programming aids for checkout on the UDOFT impeded the meeting of the development schedule. Also the input-output facilities of the UDOFT computer, while adequate for flight simulation programs which are already operational, are inadequate for the debugging process.

Certain data was checked out on straight 704 programs in order to conserve expensive computer time. The assembled data for function generation was checked out by calculating the values of the functions on a straight 704 program. This checkout method was possible because the card input used for the 704 programs was exactly the same as that used on UDOFT. Using the same card input was essential because the major source of trouble was clerical errors. With PSEUDOFT checking out the workings of the function generator subroutine, the expensive use of this simulator program for function data checkout was thus eliminated.

Following the checkout of function data, individual program routines and groupings of routines were test-run on PSEUDOFT. Where applicable, these were run at constant altitude, constant Mach number, etc., in order to by-pass the function generator for functions of these variables and thus conserve computer time.

5.3.1 Trace Facility

The facility most regularly used for first-phase checkout of program units was the trace which was provided by PSEUDOFT. By properly indicating on the program input cards, the programmer could obtain a printout of the contents of all important registers following a given instruction. By designating all instructions for this treatment, a complete trace of the program could be obtained, thus getting the maximum possible amount of information from a single run on PSEUDOFT.

An example of a UDOFT routine checked by PSEUDOFT is the short computation of the sine and the cosine of the angular position for the altitude indicator. It had been determined that linear approximations of the sine and the cosine of the angular position would provide indicator positioning to within the required accuracy. Thus, the routine computes the "linearized" sine and the "linearized" cosine of the angle to which the indicator should position.

Prior to entry into this routine, altitude has been computed module 1000 feet. Since the altitude indicator is a multi-turn device calibrated for 1000 feet per revolution, it is necessary only to determine where within this range of 1000 feet the variable is located and then to compute the "linearized" sine and cosine (figure 66).

The flow diagram of the routine (figure 67) illustrates the logical process by which the UDOFT computer evaluates the sine and the cosine. The letters A, B, and C indicate points in the routine that are attained by means of conditional transfers. The four-digit numbers in each block indicate the instructions, recorded on the program coding sheets (figure 68), that cause the computer to perform the macro-operations contained within that block.

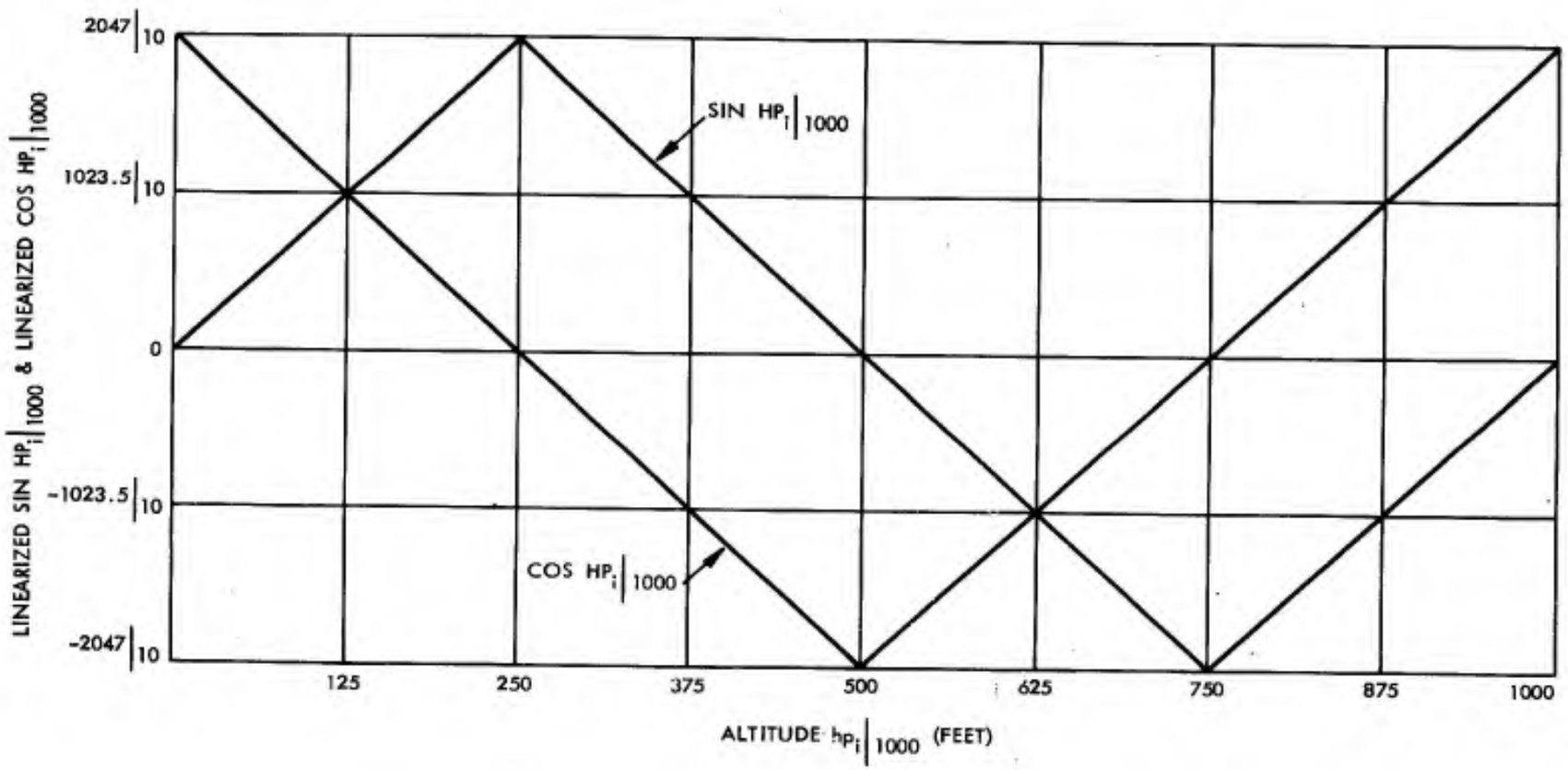


Figure 66. Curves of Linearized Sine hp_i and Cosine hp_i

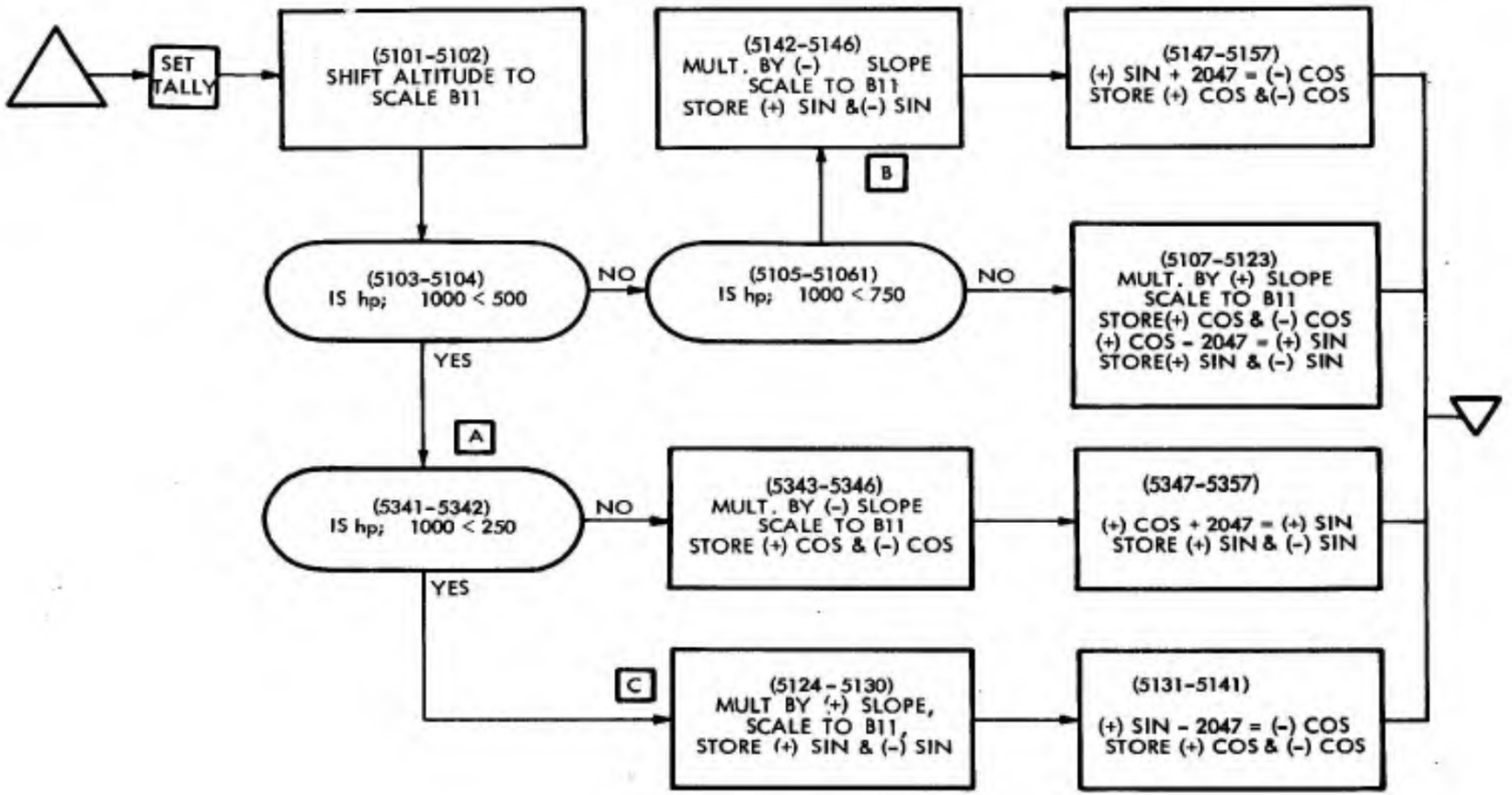


Figure 67. Diagram of Routine for Computing Linearized Sine hp_i and Cosine hp_i

UDOFT CODING FORM (INSTRUCTION MEMORY)

PROBLEM

CODER | DATE | PAGE _____ OF _____

PRELIMINARY CODE	I	M	A	D	R	I	O	T	N	M	A	D	REMARKS
STO; EXIT	5	1	0	0		I	2	3	4	7	6	0	
CLA; SI				1		I	3	4	0	1	0	1	Altitude Modulo 1000 feet B16
SH; left 5				2		I	2	5	0	1	7	5	Scale to B11
S; 500 (B11)				3		I	3	1	4	7	6	3	Test for range of variable
TOM; A				4		I	2	2	5	3	4	1	$0<hp_i<500$ transfer to A
S; 250 (B11)				5		I	3	1	4	7	6	1	Test for range of variable ($500<hp_i<1000$)
TOM; B				6		I	2	2	5	1	4	2	$500<hp_i<750$ transfer to B
M; 8.188 (B4)				7		I	2	7	4	7	6	5	$750<hp_i<1000$ multiply by slope
SH; left 4	5	1	1	0		I	2	5	0	0	7	5	Scale to B11
NOP;				1	R	I	1	4	0	0	0	2	
STO; COS				2		I	2	3	0	0	0	0	
S; 2047 (B11)				3		I	3	1	4	7	6	2	Subtract 2047 to get sine values
NOP;				4	R	I	1	4	0	0	0		Store values in designated locations
STO; SIN				5	R	I	2	3	0	0	0	0	
CLS; SIN				6	R	I	3	5	0	0	0	2	
CLS; COS				7	R	I	3	5	0	0	0	1	
STO; SIN	5	1	2	0		I	2	3	0	0	0	0	
NOP:				1	R	I	1	4	0	0	0	3	
STO; COS				2		I	2	3	0	0	0	0	
SCRNM				3		I	0	2	4	7	6	0	Exit
C ADD; 250				4		I	3	0	4	7	6	4	Restore Value of h
M; 8.188				5		I	2	7	4	7	6	5	$0<hp_i<250$; multiply by slope
SH; left 4				6		I	2	5	0	0	7	5	Scale to B11
NOP; SIN				7	R	I	1	4	0	0	0	0	

Rev. 1-9-61

Figure 68. Absolute Coding Sheets for Linearized Sine hp_i and Cosine hp_i Routines

UDOFT CODING FORM (INSTRUCTION MEMORY)

PROBLEM

CODER | DATE | PAGE ____ OF ____

PRELIMINARY CODE	I	M	A	D	R	I	O	T	N	M	A	D	REMARKS
STO; SIN	5	1	3	0		1	2	3	0	0	0	0	
S; 2047	5	1	3	1		1	3	1	4	7	6	7	Subtract 2047 to get (-) cosine value
NOP;				2	R	1	1	4	0	0	0	3	Store values in designated locations
STO; -COS				3	R	1	2	3	0	0	0	0	
CLS; SIN				4	R	1	3	5	0	0	0	3	
CLS; -COS				5	R	1	3	5	0	0	0	1	
STO; -SIN				6		1	2	3	0	0	0	0	
NOP; COS				7	R	1	1	4	0	0	0	2	
STO;	5	1	4	0		1	2	3	0	0	0	0	
SCRNM	5	1	4	1		1	0	2	4	7	6	0	Exit
B ADD; 250				2		1	3	0	4	7	6	4	
M: 8.188				3		1	2	7	4	7	6	6	$500 < hp_i < 750$; multiply by (-) slope
SH; left 4				4		1	2	5	0	0	7	5	Scale to B11
NOP; SIN				5	R	1	1	4	0	0	0	0	
STO; 0				6		1	2	3	0	0	0	0	
ADD; 2047				7		1	3	0	4	7	6	7	Add 2047 to get cosine values
NOP; -COS	5	1	5	0	R	1	1	4	0	0	0	3	Store values in designated locations
STO; -COS				1	R	1	2	3	0	0	0	0	
CLS; SIN				2	R	1	3	5	0	0	0	3	
CLS; -COS				3	R	1	3	5	0	0	0	1	
STO; -SIN				4		1	2	3	0	0	0	0	
NOP;				5	R	1	1	4	0	0	0	2	
STO; COS				6		1	2	3	0	0	0	0	
SCRNM	5	1	5	7		1	0	2	4	7	6	0	Exit

Rev. 1-9-61

Figure 68. Absolute Coding Sheets for Linearized Sine hp_i and Cosine hp_i Routine (Cont.)

UDOFT CODING FORM (INSTRUCTION MEMORY)

PROBLEM

CODER | DATE | PAGE _____ OF _____

PRELIMINARY CODE	I	M	A	D	R	I	O	T	N	M	A	D	REMARKS
A A; 250	5	3	4	1		I	3	0	4	7	8	4	
TOM; C	5	3	4	2		I	2	2	5	1	2	4	When $0 < hp_i < 250$ transfer to "C"
M; -8.188	5	3	4	3		I	2	7	4	7	6	5	$250 < hp_i < 500$; multiply by (-) slope
SH; left 4				4		I	2	5	0	0	7	5	Scale to B11
NOP;				5	R	I	1	4	0	0	0	2	
STO; COS				6		I	2	3	0	0	0	0	
A; 2047	5	3	4	7		I	3	0	4	7	6	7	Add 2047 to get sine values
NOP;	5	3	5	0	R	I	1	4	0	0	0	0	Store values in designated locations
STO; SIN				1	R	I	2	3	0	0	0	2	
CLS; COS				2	R	I	3	5	0	0	0	0	
CLS; SIN				3	R	I	3	5	0	0	0	3	
STO; -COS				4		I	2	3	0	0	0	0	
NOP;				5	R	I	1	4	0	0	0	1	
STO; -SIN						I	2	3	0	0	0	0	
SCRNM	5	3	5	7		I	0	2	4	7	6	0	Exit
SENIT	5	3	6	0		I	0	7					

Rev. 1-9-61

Figure 68. Absolute Coding Sheets for Linearized Sine hp_i and Cosine hp_i Routine (Cont.)

UDOFT CODING FORM (NUMBER MEMORY)

PROBLEM

CODER | DATE | | PAGE _____ OF _____

REMARKS	IDENTIFICATION	N	M	A	D	+/-	A	N	U	M	B	E	R
	Exit	4	7	6	0	+	0	0	0	5	3	6	0
	Constant, 500'	4	7	6	3	+	0	7	6	4	0	0	0
	Constant, 250'	4	7	6	4	+	0	3	7	2	0	0	0
	Slope, +8.188/10	4	7	6	6	-	2	0	3	0	0	4	0
	Y Intercept, +2047/10	4	7	6	7	+	3	7	7	7	0	0	0
Computed Indicated Altitude	hp_i/1000	0	1	0	1	X	X	X	X	X	X	X	X

Rev. 1-9-61

Figure 68. Absolute Coding Sheets for Linearized Sine hp_i and Cosine hp_i Routine (Cont.)

The program coding sheets (figure 69) contain all the program information that must be entered into PSEUDOFT for this program. This information includes instruction words, print-out requirements (I), and number words. Where possible a brief description of each instruction, or a group of instructions, is given. These coding sheets are indicative of the effort that must be expended to prepare a program in absolute form. The expenditure of effort is reduced greatly when the programs are prepared in symbolic form and the assembly program is used to prepare the assembly listings which will provide data comparable to that contained on the absolute coding sheets. An assembly listing of this same routine (figure 69), assembled as a part of the total F-100A program, provides a realistic indication of the extent to which an assembly program relieves the programmer of an inordinate amount of clerical work.

As has been stated previously, the purpose of PSEUDOFT was to make possible a certain amount of checkout of the operational flight programs prior to the availability of UDOFT computer. A simulator should provide exactly the same output as the computer being simulated. Fortunately, however, PSEUDOFT provided much more output than the UDOFT computer, specifically, it provided a means for tracing the exact operation of any program run on it. During the course of checking out the aircraft simulation programs, it became obvious that a selective trace facility was of the greatest value in debugging programs. (Since the completion of the UDOFT project the simulator-tracer technique of program checkout has been successfully used with a large data processing program.)

In both these cases, the full trace facility was used because it came as an adjunct to the simulator, and the simulator was deemed to be absolutely necessary. Most computing facilities do not provide full tracers as part of their debugging program package; it is generally thought to be a waste of money. The usual line of argument is that a full trace program means an interpreter*; an interpreter is expensive to write and to run, and the advantages do not justify the expense. It is true that a real tracing program requires an interpreter. It is not true that an interpreter is expensive to write; an interpreter can be written in about four man-weeks for a medium size (32 instruction) computer. It is true that an interpreter is expensive to use; it is more expensive to run a program on an interpreter than to free-running it on a computer. Note, however, that interpreters are being considered for debugging runs, not production runs. Therefore, one should not calculate cost per run, but cost per piece of information, since meeting a schedule and documentation of test results are also prime considerations. The criterion for an effective debugging run drifts even further away from cost per run. When these other factors are taken into consideration, the utility of debugging runs using a trace increases even more.

For future applications of digital computers to the real-time simulation problem, it is strongly recommended that an interpretive routine be written which allows tracing to be implemented on the final operational computer, especially if the computer is available at the beginning of the project.

* A simulator is one form of interpretive program.

SINE-COSINE ROUTINE

```
5150 P        03   5151   SCENT   SCR    * + 1
5151          23   4744   SC      STO    EXIT    SET EXIT LOCATION
5152          34   0101          CLA    S1      IS VARIABLE LESS THAN 500
5153          25   0175          SHL    5       X
5154 P        31   4745          SUB    SCN01   X
5155 P        22   5175          TOM    SCA     IF YES, GO TEST AGAIN
5156 P        31   4746          SUB    SCN02   NO, IS VARIABLE LESS THAN 750
5157          22   5214          TOM    SCB     IF YES, GO COMPUTE SINE AND COSINE
5160 P        27   4747          MPY    SCN03   NO, COMPUTE COSINE
5161 P        25   0075          SHL    4       X
5162 P   R    14   0002          NOP*   COS     X
5163          23   0000          STO    0       X
5164 P        31   4751          SUB    SCN05   SUBTRACT 2047 TO GET SINE VALUES
5165     R    14   0000          NOP*   SIN     STORE VALUES IN DESIGNATED LOCATIONS
5166     R    23   0002          STO*   COS     X
5167     R    35   0000          CLS*   SIN     X
5170     R    35   0003          CLS*   MCOS    X
5171          23   0000          STO    0       X
5172 P   R    14   0001          NOP*   MSIN    X
5173          23   0000          STO    0       X
5174          02   4744          SCRNM  EXIT    EXIT

5175          30   4746   SCA     ADD    SCN02   IS VARIABLE LESS THAN 250
5176 P        22   5232          TOM    SCC     IF YES, WHEN OLT X LT 250 TRANSFER TO SCC
5177 P        27   4750          MPY    SCN04   NO, VARIABLE LIES BETWEEN 250 AND 500
5200 P        25   0075          SHL    4       COMPUTE COSINE
5201 P   R    14   0002          NOP*   COS     X
5202          23   0000          STO    0       X
5203          30   4751          ADD    SCN05   ADD 2047 TO GET SINE VALUES
5204     R    14   0000          NOP*   SIN     STORE VALUES IN DESIGNATED LOCATIONS
5205     R    23   0002          STO*   COS     X
5206     R    35   0000          CLS*   SIN     X
5207          35   0003          CLS*   MCOS    X
5210          23   0000          STO    0       X
5211 P   R    14   0001          NOP*   MSIN    X
5212          23   0000          STO    0       X
5213          02   4744          SCRNM  EXIT    EXIT
```

Figure 69. Assembly Listing of Linearized Sine hp_i and Cosine hp_i Routine

```
5214                 30  4746  SCB   ADD   SCN02  VARIABLE LIES BETWEEN 500 AND 750
5215 P               27  4750        MPY   SCN04  COMPUTE SINE
5216 P               25  0075        SHL   4      X
5217       R         14  0000        NOP*  SIN    X
5220                 23  0000        STO   0      X
5221                 30  4751        ADD   SCN05  ADD 2047 TO GET COSINE VALUES
5222       R         14  0003        NOP*  MCOS   STORE VALUES IN DESIGNATED LOCATIONS
5223 P     R         23  0000        STO*  SIN    X
5224       R         35  0003        CLS*  MCOS   X
5225 P     R         35  0001        CLS*  MSIN   X
5226                 23  0000        STO   0      X
5227 P     R         14  0002        NOP*  COS    X
5230                 23  0000        STO   0      X
5231                 02  4744        SCRNM EXIT   EXIT

5232                 30  4746  SCC   ADD   SCN02  VARIABLE LIES BETWEEN 0 AND 250
5233 P               27  4747        MPY   SCN03  COMPUTE COSINE
5234 P               25  0075        SHL   4      X
5235       R         14  0000        NOP*  SIN    X
5236                 23  0000        STO   0      X
5237 P               31  4751        SUB   SCN05  SUBTRACT 2047 TO GET COSINE VALUES
5240       R         14  0003        NOP*  MCOS   STORE VALUES IN DESIGNATED LOCATIONS
5241 P     R         23  0000        STO*  SIN    X
5242       R         35  0003        CLS*  MCOS   X
5243 P     R         35  0001        CLS*  MSIN   X
5244                 23  0000        STO   0      X
5245 P     R         14  0002        NOP*  COS    X
5246                 23  0000        STO   0      X
5247                 02  4744        SCRNM EXIT   EXIT

                     0000  SIN   EQU   0
                     0001  MSIN  EQU   1
                     0002  COS   EQU   2
                     0003  MCOS  EQU   3
```

Figure 69. Assembly Listing of Linearized Sine hp_i and Cosine hp_i Routine (Cont.)

CONSTANTS FOR SINE – COSINE

```
4744 P  +0000000  EXIT   DEC   0           EXIT FROM SINE-COSINE                    B11
4745 P  +1750000  SCN01  DEC   500811      TO TEST RANGE OF VARIABLE                B11
4746 P  +0764000  SCN02  DEC   250811      TO TEST RANGE OF VARIABLE                B4
4747 P  +4060100  SCN03  DEC   8.18884     SLOPE                                    B4
4750    -4060100  SCN04  DEC   -8.18884    SLOPE                                    B11
4751    +7776000  SCN05  DEC   2047811     TO GET SINE VALUES
4752 P  +0000000         DEC   0                                                    B2
4753 P  +0000000  NOGSL  DEC   0           L3 SQUARED                               B1
4754    +4000000  NOGS1  DEC   181         TO COMPUTE GROUND SPEED                  B2
4755    +7451556  NOGS2  DEC   3.790740782 PREPARE GROUND SPEED TO MULTIPLEX OUT.   B14
4756    +0777600  NOGS3  DEC   2047814
4757 P  +0000000  NORCO  DEC   0
```

Figure 69. Assembly Listing of Linearized Sine hp_i and Cosine hp_i Routine (Cont.)

5.3.2 Dump Facility

A routine was written to dump certain sections of the 709 core memory in a form compatible with UDOFT notation. It was designed to assist the programmer in debugging a UDOFT program being test-run with the PSEUDOFT program. The contents of special registers in the 709 head the dump listing followed by the contents of special registers used by the UDOFT computer and simulated by PSEUDOFT (figure 70).

The first line printed out by this program indicates the contents of the 709 Accumulator (AC), the three Index Registers (X REGS A, B, C), and the status of the four Sense Lights. The second line of the printout contains the status of the six 709 Sense Switches; the Accumulator and the MQ Register Overflow lights; and the Divide Check and Tape Check flip-flops. The third line of print-out indicates the address of the last UDOFT instruction that was simulated, the instruction, and the contents of the Tally Register and the Interval Timer. The fourth line indicates the status of the UDOFT Accumulator (AU), Transfer Register, and the G-Register. The fifth line indicates the status of the sixty-four Discrete Inputs, which, even in the simulator, are accessible to the programmer. The non zero contents of those registers which simulate the UDOFT Analog Inputs are printed and identified on successive lines. In the case of the example of figure 70, only one analog input channel, A109, is nonzero.

Following the aforementioned print-out of special registers, the contents of the locations reserved for the "cockpit" program (a program to control input parameters to the UDOFT program) are printed out in 709 format. The 709 control program for the example of figure 70 occupies memory registers 24000 - 24040 (the reader is reminded again that octal notation is used throughout). UDOFT instruction memory registers (10000_8 - 20000_8, 709; 0000_8 -7777_8, UDOFT) are then dumped in UDOFT format, bypassing eight or more successive registers whose content is identically ZERO. At the completion of the instruction memory dump, the number memory is dumped in the same manner.

The sample dump shown in figure 70 should provide a clear picture of the form and the content of a dump from PSEUDOFT. The particular example is a preliminary program for F9F-2 ice quantity. It should be noted that the number memory dump is not duplicated in its entirety; it has been terminated arbitrarily at number memory location 2030 in order to facilitate its inclusion in this report. Normally, the entire number memory is dumped and printed.

It is the majority opinion of the programmers that although the dump facility should be included, it should not be substituted for tracing in order to "save computer time," since, in general, it is the trace which saves running time over the long run. It is felt also that a more convenient method of ordering a trace should be incorporated in the simulator package; the method of indicating "printout desired" on the same cards as the program input offers an obvious disadvantage in that the program input cards themselves must be changed from run to run. In contrast, for a recent simulator implemented at Sylvania, a trace is commanded by console switch settings, a more desirable method. A further refinement is to indicate that certain segments of memory only are to be traced, or that the program is to be traced from the time the Program Counter contains a_1 to the time it contains a_2. Either approach would be simple to implement, as long as an interpretive program is available.

5.3.3 Other Checkout Aids

As more and more programs were assembled and dynamically tested on PSEUDOFT it became apparent that a truly flexible means of control should be provided in addition to the fixed control program incorporated in PSEUDOFT. This control was provided by coding extra control orders with the UDOFT instruction words which cause PSEUDOFT to break off the simulation and transfer control to one of seven predetermined 704 addresses (for the 709 version, this was reduced to four addresses). At this address the programmer can place a 704 (709) program to accomplish the desired control. After the execution of the control program, control is transferred back to PSEUDOFT and the simulation continues from the point where it was interrupted by the initial transfer of control.

```
AC      000000000007      X REGS      A 00001      B 16740      C 14000      SENSE LITES 0000
SWITCHES ON 020000  AC OVERFLO OFF  MQ OVERFLO OFF  DIVIDE CHECK OFF  TAPE CHECK OFF
LAST ORDER SIMULATED    6002 07 0000          TALLY READS 000000          I.T. READS 00000
(UDOFT AU)    +0000662    TRANSFER REG    +0000662    G-REGISTER    +0000000
DISCRETE INPUTS   00000000 10 00000000 20 00000000 30 00000000 40 00000000 50 00000000 60 00000000 70 00000000 00
ANALOGUE INPUTS
 AI 09 + 7770000
0

704 CONTROL PROGRAM

24000                +007400 434631   +000000 000002   +006432 500014   +312343 226260
24004                +050000 124040   +060100 020030   +200001 124016   +050000 024023
24010                +002000 020425   +000000 006140   +000001 000000   +000000 000000
24014                +000600 000000   +000600 000000   +000000 000000   +377000 000000
24020                +377400 000000   +076100 000000   +076100 000000   +050200 024070
24024                +076100 000000   +076100 000000   +076100 000000   +060000 020014
24030                +076100 000000   +076100 000000   +076100 000000   +000000 000000
24034                -000310 000000   -377700 000000   -300000 000000   +310000 000000
24040                +000100 001000   +000000 000000   +000000 000000   +000000 000000
0
UDOFT ORDER MEMORY DUMP
0000   33 0000      00 0000      00 0000      00 0000      00 0000      00 0000      00 0000      00 0000
6000   34 6422      34 6432      07 0000      00 0000      00 0000      00 0000      00 0000      00 0000
6140   03 6141      03 6142      12 0164      25 0005      10 6174      31 6420      22 6172      31 6421
6150   22 6165      31 6421      22 6160      34 6422      30 6423      10 6223      23 6422      03 6226
6160   34 6422      30 6424      10 6223      23 6422      03 6226      34 6422      30 6425      10 6223
6170   23 6422      03 6226      34 6422      03 6226      31 6426      22 6212      31 6421      22 6205
6200   34 6422      31 6425      10 6226      23 6422      03 6226      34 6442      31 6424      10 6217
6210   23 6422      03 6226      34 6422      31 6423      10 6217      23 6422      03 6226      35 6427
6220   14 0000      23 6422      03 6226      34 6427      14 0000      23 6422      23 0246      27 6430
6230   30 6431      14 0000      23 6432      03 6000      07 0000      00 0000      00 0000      00 0000
0
UDOFT NUMBER MEMORY DUMP
0000   0000000      0000002      0000004      0000006      0000010      0000012      0000014      0000016
0010   0000020      0000022      0000024      0000026      0000030      0000000      0000000      0000000
0050   4000000      0000000      0000000      0000000      4652506      1000000      1000000      0000000
0060   4000000      4000000      0000000      0000000      0000000      4000000      0000000      0000000
0070   0000000      4000000      0000000      0000000      0000000      0000000      0000000      0000000
0100   0000000      0000000      0000000      0000000      7777776      0000000      7776000      0000000
0110   0000240      1000000      1400000      3600000      0006200      0024000      0651630      1022040
0240   0000000      0000000      0000000      0000000      0000000      0000000      -7773750     0000000
0440   0000000      0000000      0001440      0001440      0004000      0000000      0372000      2000000
0450   0000000      -0000002     -3330000     -5552000     4340000      0000000      0000000      0000000
0470   -0006200     -0006200     0000000      0000000      0000000      0000000      0000000      0000000
0500   0000000      0000560      1651400      0000000      0540000      5300406      4600000      0076400
0510   0741216      5050356      0146314      0006200      0025750      0125604      2400000      0470400
0520   1714630      0546314      2352224      0062000      1200000      3453004      0646314      0724600
0530   4143366      0000000      0571370      0663146      0152200      5341216      0300400      0000000
0730   0000000      0000000      -0054500     0000000      0776030      0000000      1000000      0000000
1010   0000000      0000000      2400000      1200000      0620000      5000000      3032400      5500000
1020   532017Z      3100000      4663016      3314630      0204700      0136152      5320172      4300000
1030   -0660422     6727262      0372000      0705430      1320712      6034332      3431462      4246456
1040   2663146      0605074      0266314      7020000      0030242      0144000      0024760      5174264
1050   2231462      3600000      4540000      0175000      3100000      5643504      0050752      1034530
1060   4631462      2663146      1700000      7652632      -5347564     1463146      4472166      1063470
1070   1250000      5174264      1463146      0764000      2631462      1212000      7400000      0706314
1100   3220000      0372000      0000000      0000000      0000000      0000000      0000000      0000000
2000   0000000      7776000      3376756      -0365604     0075340      1031462      0431462      -4314630
2010   0160506      6662200      -0277272     2643656      0111564      2143222      5300406      4773716
2020   1324772      -0254020     4463146      0314630      4543000      3231462      -0162020     0165754
2030   0207034      5204742      4061114      1463146      7776000      3376756      0006200      5174264
```

Figure 70. Sample of UDOFT Dump

The simulation of a SENIT instruction (Sense Interval Timer) results also in the automatic transfer of control to a predetermined 704 (709) address. This feature facilitates the changing of aircraft parameters at the end of the 50 millisecond iteration cycle. If no control program is encountered at the address to which program control has been transferred, PSEUDOFT continues to the next UDOFT instruction.

Usually, the control program, to which control is transferred by the SENIT instruction, contains a special output program in addition to, or in the place of, the parameter-changing program. The basic output routine designated 9 out, was prepared to set-up and print on-line (72 or 120 column print-out), or to output a complete line of information to a specified magnetic tape, or both. Any desired format may be used, and conversion from floating binary to fixed decimal, floating binary to floating decimal, or fixed binary to fixed decimal can be performed as desired.

A sample of the form of print-out which may be obtained through the use of this routine is depicted in figure 71; the program undergoing test was the F9F-2 stick force computations. The print-out control program required that, in addition to the count in the Interval Timer, the following parameters be printed out in decimal; dynamic pressure, Mach number, aileron control loading, elevator control loading, rudder control loading, aileron deflection, elevator deflection, and rudder deflection.

From the preceding discussions, it must be apparent that there were adequate automatic programming aids for the debugging tasks which were performed on the 704 and 709. When it came time for final integration and system checkout using the UDOFT computer, the complexion changed. The great deterrent to expeditious flight-simulator system checkout was the UDOFT input-output capability, which had not been designed with program checkout needs in mind.

The only printed output available from the UDOFT computer was obtained by means of the IBM electric typewriter, which was able to print out the contents of selected registers in octal notation and under program control. However, it was necessary to halt the simulation program in order that this process could occur. This feature did provide a trace facility on the UDOFT computer, but did not provide decimal output. The use of strip recorders actuated by the analog outputs provided yet another output facility. This feature was most helpful during acceptance testing of the total system as a flight simulator but was inadequate during the program debugging phase of the project.

In retrospect it can be stated that the checkout and the testing of program directly on the UDOFT computer were the least successful tasks of the total programming task. With due respect for computer design engineers, it is apparent that this shortcoming was due to lack of foresight in designing the UDOFT computer, particularly the computer's non-real-time input-output facility.

5.4 Operational Program Considerations

The development of the UDOFT system offered an opportunity to improve the fidelity of flight simulation. If used properly, the UDOFT computer can provide a simulation of a dynamic system which is superior to an analog simulation of the same system in the following respects:

1. Improved static and dynamic accuracy.
2. Improved small-signal response.
3. Freedom from drift.
4. Flexibility of "implementation" allowing rapid change incorporation.

The major aspects of the program which most noticeably affect these characteristics are:

1. The method of data reduction and subsequently the form of function generation used within the operational simulation program.
2. The mathematics for describing the position and orientation of the aircraft.

```
F9F STICK FORCE COMPUTATION
OMAD OT NMAD (NMAD) TALY
I.T. READS      +00223 OCTAL
0               Q
0               43.54688
0               DE
0                0.00000
I.T. READS      +00224 OCTAL
0               Q
0               43.54688
0               DE
0              -15.00000
I.T. READS      +00224 OCTAL
0               Q
0               43.54688
0               DE
0              -10.00000
I.T. READS      +00224 OCTAL
0               Q
0               43.54688
0               DE
0              - 6.00000
I.T. READS      +00223 OCTAL
0               Q
0               43.54688
0               DE
0              - 4.00000
I.T. READS      +00223 OCTAL
0               Q
0               43.54688
0               DE
0              - 2.00000
I.T. READS      +00223 OCTAL
0               Q
0               43.54688
0               DE
0                0.00000
I.T. READS      +00223 OCTAL
0               Q
0               43.54688
0               DE
0                2.00000

(AC)
AT   6435
     MACH        AIL         ELEV        RUD         DSA
      0.13400     0.00000    29.59375     0.03906     0.00000
     DR
      0.00000
AT   6435
     MACH        AIL         ELEV        RUD         DSA
      0.13400    -1.43750    90.87500   -51.47656   -10.00000
     DR
    -20.00000
AT   6435
     MACH        AIL         ELEV        RUD         DSA
      0.13400    -1.15625    70.44531   -35.49219   - 8.00000
     DR
    -15.00000
AT   6435
     MACH        AIL         ELEV        RUD         DSA
      0.13400    -0.85938    54.10156   -19.48438   - 6.00000
     DR
    -10.00000
AT   6435
     MACH        AIL         ELEV        RUD         DSA
      0.13400    -0.57813    45.93750   - 5.09375   - 4.00000
     DR
    - 5.00000
AT   6435
     MACH        AIL         ELEV        RUD         DSA
      0.13400    -0.28125    37.75781   - 2.00781   - 2.00000
     DR
    - 2.00000
AT   6435
     MACH        AIL         ELEV        RUD         DSA
      0.13400     0.00000    29.59375     0.03906     0.00000
     DR
      0.00000
AT   6435
     MACH        AIL         ELEV        RUD         DSA
      0.13400     0.28125    21.42969     2.10938     2.00000
     DR
      2.00000
```

Figure 71. Print-out of PSEUDOFT Checkout of F9F-2 Stick Force Computation

```
F9F STICK FORCE COMPUTATION
OMAD OT NMAD (NMAD) TALY (AC)
 I. T. READS      +00223 OCTAL      AT   6435
 0                Q                      MACH          AIL           ELEV          RUD           DSA
 0                43.54688                0.13400       0.57813       13.2500       5.19531       4.00000
 0                DE                     DR
 0                 4.00000                5.00000
 I. T. READS      +00222 OCTAL      AT   6435
 0                Q                      MACH          AIL           ELEV          RUD           DSA
 0                43.54688                0.13400       0.85938       5.06594       19.57813      6.00000
 0                DE                     DR
 0                 6.00000                10.00000
 I. T. READS      +00222 OCTAL      AT   6435
 0                Q                      MACH          AIL           ELEV          RUD           DSA
 0                43.54688                0.13400       1.15625       -11.25781     35.58594      8.00000
 0                DE                     DR
 0                10.00000                15.00000
 I. T. READS      +00222 OCTAL      AT   6435
 0                Q                      MACH          AIL           ELEV          RUD           DSA
 0                43.54688                0.13400       1.43750       -31.68750     51.57813      10.00000
```

Figure 71. Print-out of PSEUDOFT Checkout of F9F-2 Stick Force Computation (Cont.)

3. The solution rate; i.e. the rate at which the input parameters are sampled and the mathematics describing the response of the aircraft are solved.

4. The form of numerical integration.

5. The critical use of real-time.

6. The modularity of the organization of the flight-simulation program.

The following sections are devoted to a discussion of these aspects and an exposition of the associated problems that developed during the formulation of the flight-simulation programs for the UDOFT system.

5.4.1 Data Reduction and Function Generation

Function generation is a subject of considerable importance for two reasons; first, it constitutes the single largest portion of a total flight-simulation program; and second, the accuracy of the simulation depends upon the fidelity with which the shape of the measured data is represented in the computer program.*

There are two basic methods by means of which single variable functions may be generated:

1. Use of piecewise-linear approximations to the nonlinear functions with interpolation between stored data points which describe the breakpoints of the piecewise-linear functions.

2. Use of polynomial approximations to the nonlinear functions.

For the former method there are at least two approaches to the selection of the stored data points; namely, the use of breakpoints which are freely chosen to best suit each function, and the use of standard or fixed sets of breakpoints for each independent variable.

a.) Freely Chosen Breakpoints

Consider first the program to interpolate between stored data points, using a different set of breakpoints for each function, (the UDOFT function generator program performs just this function). Since the function generator must deal with essentially random breakpoints in the functions, the program must first determine the appropriate segment of the piecewise-linear function within which the independent variable lies. This process is referred to as table look-up or level-selection.** Table look-up is a relatively simple process; however; if a large number of tables must be searched, and each table contains many breakpoints, much computation time can be consumed. In order to minimize look-up time, the UDOFT table searching routine was written in such a way that searching each table from its beginning for every function was unnecessary. Rather, each table search commenced at the point in the table where a previous function had been evaluated during the cycle. Since it is probable that a function will change (only a small increment) in one program iteration cycle, it is therefore probable that the function will still be contained in the same straight-line segment or, at worst, in either of the two adjoining segments.

Further, the function data was stored in number memory in such a manner that if the correct segment for the function f_m was segment n, it would be highly probable that the correct segment for the next function to be evaluated f_{m+1} would be segment n also. It was no simple task to arrange the functions in this manner. Had it not been done however, it is quite likely that function generation alone would have consumed nearly all of the available computation cycle time.

---

*For further discussion of function generation in digital computers refer to MRL-TDR-62-271, Submicrosecond Simulation Computer Study Program, May 1962.

**L. M. Krasny, The Functional Design of a Special Purpose Digital Computer for Real-Time Flight Simulation, Massachusetts Institute of Technology, Report ESL-R-118, August 1961.

To further minimize computation time, the function generation program relies on a stored value of the slope of each line segment at each breakpoint. Even though the value of the slope represents redundant information, the storage of this data eliminated the need for the computer to evaluate $\Delta y/\Delta x$ for each line segment. This was essential because the divide process in the UDOFT computer is extremely slow (105 microseconds).

Thus, once the appropriate segment has been located, slope, breakpoint, and intercept data are extracted from memory for use in the simple computation.

$$y = m_b(x - x_b) + b_b \qquad (6)$$

where b_b is the ordinate of the function at the breakpoint, not the value of the y-intercept if the straight-line segment had been extended to the y-axis (x = o).

b.) Fixed Breakpoints

A much less time-consuming computation scheme would result if fixed or standardized breakpoints were selected for functions of the same variable. This would result in a considerable reduction in the time required for breakpoint searching. Since this approach was not considered for the UDOFT flight-simulation programs, it is not known what problems might arise from its use; Krashy* concludes there are no real problems associated with the approach and that the use of standardized breakpoints will not detract from the accuracy of the overall simulation. Even though these conclusions, which appear valid on the surface, have not been verified experimentally, there is insufficient reason to deny them. However, one area of doubt does prevail; namely, will the overall simulation remain stable when the independent variables are identical to the breakpoints, thereby resulting in all functions of these variables becoming discontinuous simultaneously.

c.) Polynomials in One Variable

Current studies indicate that fourth order polynomial approximations are adequate for describing the majority of single-variable functions that are encountered in the flight-simulation problem. The advantages of using polynomials are the greatly reduced data storage requirement and the smoother approximation of functions, thereby eliminating the discontinuities that are an inevitable result of the table look-up scheme.

Early in the UDOFT program, attempts were made to derive polynomial approximations for some of the F-100A aerodynamic functions. The results were so poor that this approach was dropped; subsequently, the decision was made to use piecewise-linear approximations throughout the program.

d.) Conclusions

A summary comparison of the single-variable methods of function generation reveals that the freely-chosen breakpoint scheme requires more computation time than does either the fixed-breakpoints per independent variable (provided the number of breakpoints is held to a reasonable number) or the polynomial scheme. All of these methods share in the difficulty in data reduction that is encounterd when using combinations of functions of one variable to approximate a bivariant function. In addition, the use of polynomial approximations requires increased effort for data reductions. However, since computer programs for obtaining least squares polynomial fits are commonplace, this task may not be so overwhelming.

Of the two variable methods (freely chosen breakpoints and polynomial fits) one cannot say apriori which is the better, since the ultimate selection depends to a great extent on the behavior of the data to be used. The surface fit provides a more nearly continuous approximation than does the table-lookup scheme, but it is not always successful in producing a polynomial of reasonable degree which adequately represents the data.

---

*L.M. Krashy, The Functional Design of a Special Purpose Digital Computer for Real-Time Flight Simulation, Massachusetts Institute of Technology, Report ESL-R-118, August 1961.

The profusion of possible methods, together with the many possible trade-offs between time, data storage, and accuracy leads to the final conclusion that, in the ultimate flight simulation program developed for execution on a general-purpose type digital computer, there can, and in all likelihood will be, several methods of function generation. In the case of the F-100A flight-simulation program however, there was no freedom in the selection of the optimum function generation scheme. The data provided had already been reduced to a form suited for analog computation. Since the UDOFT project might be classified as a final phase feasibility program, neither time nor money was expended to rederive program data from airframe data. As a result, the F-100A program followed, in general, the plan devised by Melpar for the MB-3 Flight Simulator. The functional data, organized on the basis of freely-chosen breakpoints for best fit, dictated that the F-100A program adhere to the same scheme.

Up to this point, no comments have been made regarding the generation of functions of two variables, due to the absence of bivariant functions in the F-100A program (since the data had been reduced to monovariant functions). Bivariant function generation is useful if:

1) It reduces the complexity of data reduction.

2) It reduces the total number of functions to be evaluated.

3) It reduces the number of arithmetic operations which must be performed in evaluating the functions.

The above statements are in agreement with Krashy. It has been estimated that in the time it takes to evaluate three single-variable functions using the freely chosen breakpoint scheme and to combine them in the form $f = (f_1)(f_2)(f_3)$, a single two-variable function, which might be the equivalent of the combination of six single variable functions, may be evaluated. Since it is commonly agreed that most airframe data is bivariant in nature, reduction to a number of monovariant functions may be impractical.

5.4.2 Method of Describing Position and Orientation

The equations of motion of a rotating rigid body are usually simplified by using a set of axes attached to the rigid body. The simplification results from the fact that in this axis system, the body axes, the cross products of inertia are negligible. The body axes form a right-hand set of axes, with the origin at the vehicle center of gravity: the x-axis points forward along the center line of the vehicle; the y-axis points sideward; the z-axis points downward.

Since the body is capable of rotating in space, a means for describing the angular position of the vehicle with respect to the earth is necessary. Orientation of the vehicle is specified by the three Euler angles θ (pitch), ϕ (roll), and ψ (heading). The three angles describe the orientation of the vehicle body axes with respect to a reference plane which is parallel to the earth's surface (assuming a flat earth) or, more precisely, normal to the radius vector to the center of the earth, and passes through the vehicle at its center of gravity.

If the angular velocity of the vehicle about the body axes x, y, and z is p, q_1, and r respectively, the following three differential equations indicate the interrelations between the Euler angles and the components of angular velocity:

$$\dot{\theta} = q_1 \cos\phi - r \sin\phi \tag{7}$$

$$\dot{\phi} = \frac{1}{\cos\phi} (p \cos\theta + q_1 \sin\theta \sin\phi + r \sin\theta \cos\phi) \tag{8}$$

$$\dot{\psi} = \frac{1}{\cos\theta} (q_1 \sin\phi + r \cos\phi) \tag{9}$$

It is obvious that when $\theta = 90°$ ($\cos\theta = 0$) two of these equations are indeterminate. Since a continuous solution of the equations for all values of θ and ϕ is desirable without resorting to trickery, a different approach to the problem is necessary. As indicated by

Howe*, and the University of Pennsylvania**, this problem is completely eliminated by the use of direction cosines for relating the body axes to the earth axes. If the direction cosines are identified as $l_{1, 2, 3}$, $m_{1, 2, 3}$, and $n_{1, 2, 3}$, the expressions for the direction cosines as functions of θ, ϕ and ψ are:

$$l_1 = \cos\theta \cos\psi \tag{10}$$

$$l_2 = \cos\theta \sin\psi \tag{11}$$

$$l_3 = -\sin\theta \tag{12}$$

$$m_1 = \sin\theta \sin\phi \cos\psi - \cos\phi \sin\psi \tag{13}$$

$$m_2 = \sin\theta \sin\phi \sin\psi + \cos\phi \cos\psi \tag{14}$$

$$m_3 = \cos\theta \sin\phi \tag{15}$$

$$n_1 = \sin\theta \cos\phi \cos\psi + \sin\phi \sin\psi \tag{16}$$

$$n_2 = \sin\theta \cos\phi \sin\psi - \sin\phi \cos\psi \tag{17}$$

$$n_3 = \cos\theta \cos\phi \tag{18}$$

Since the angular velocities, p, q_1 and r are readily available from the calculations of the total forces and moments acting about the vehicle body axes, it is desirable to use this information in determining the aircraft orientation in terms of θ, ϕ and ψ. Relating the three angular velocities to the direction cosines produces nine differential equations of the following form:

$$\dot{l}_{1, 2, 3} = m_{1, 2, 3}\, r - n_{1, 2, 3}\, q_1 \tag{19}$$

$$\dot{m}_{1, 2, 3} = n_{1, 2, 3}\, p - l_{1, 2, 3}\, r \tag{20}$$

$$\dot{n}_{1, 2, 3} = l_{1, 2, 3}\, q_1 - m_{1, 2, 3}\, p \tag{21}$$

These differential equations have no indeterminate points; they have other desirable properties also. The direction cosines form a redundant system, using nine quantities where, in general, only three are required, and therefore a program using these differential equations may apply certain corrections to minimize drift due to truncation of the finite computation. *** The program to normalize and orthogonalize these quantities is described in Section VI, Simplified Description of the F-100A Simulation Program.

a.) Angular Position

Aside from their use in the classical equations of motion, the Euler angles θ, ϕ, and ψ are displayed intimately on cockpit indicators. The use of direction cosines, rather than Euler angles, as the descriptors of vehicle orientation, leads to problems in the actuation of the gyro-horizon and the heading indicators. A simplified diagram of the implementation of those indicators is shown in figure 72. The voltages V_1 and V_2 are analog outputs whose values are the sine and cosine, respectively, of the quantity to be read on the instrument. P_1 and P_2 are consinusoidally and simusoidally wired potentiometers, respectively. The output of the motor amplifier is

*R.M. Howe, Coordinate Systems for Solving the Three Dimensional Flight Equations, WADC Tech Note 55-747, June, 1956.

**Flight Trainer Digital Computer Study, University of Pennsylvania, Moore School of Electrical Engineering Research Division Report 51-28, 21 March, 1951.

***Flight Trainer Digital Computer Study, University of Pennsylvania, Moore School of Electrical Engineering Research Division Report 51-28, 1957.

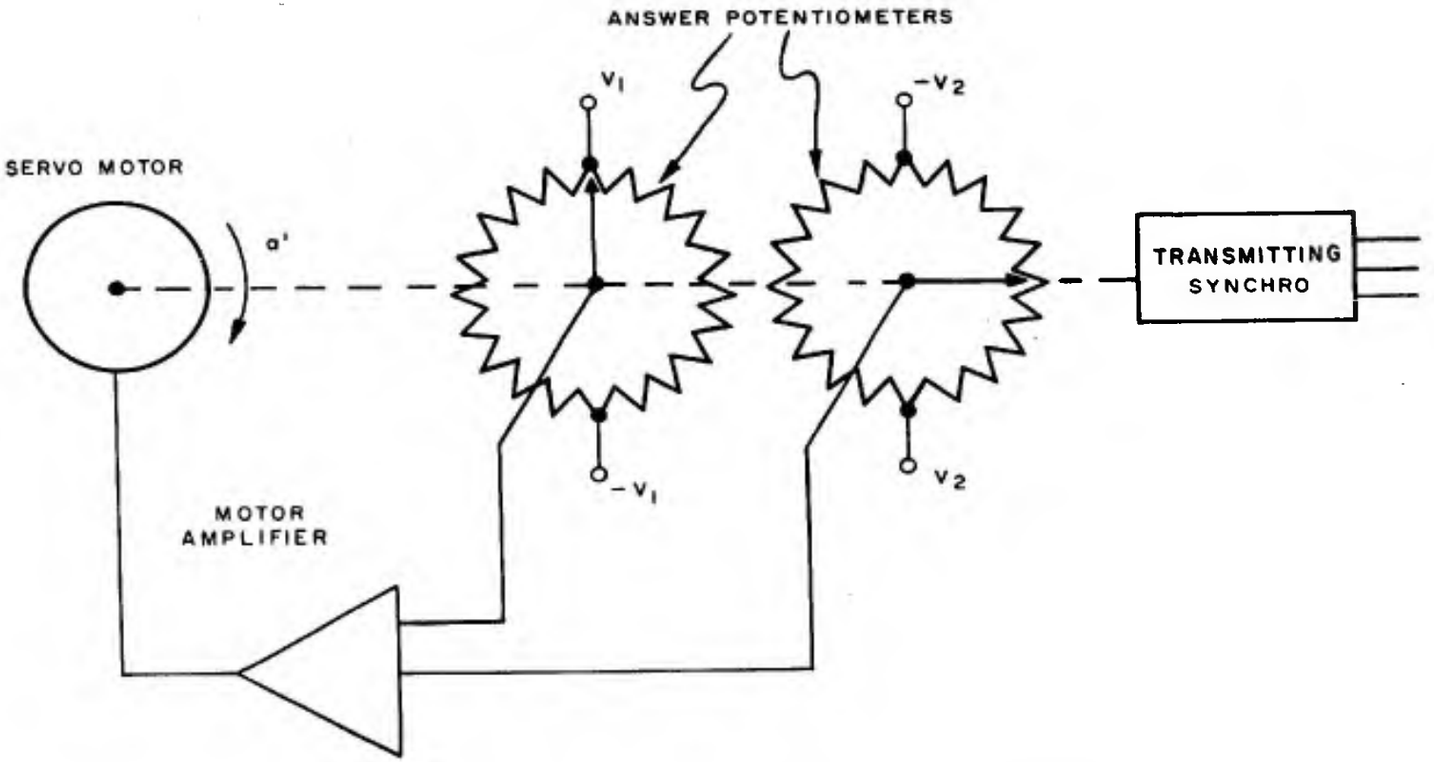


Figure 72. Diagram of Servo System Controlling Full Rotational Indicator, V-27A

$$V_1 \cos a' - V_2 \sin a'$$

However,

$$V_1 = E \sin a \text{ and } V_2 = E \cos a$$

Therefore,

$$V_1 \cos a' - V_2 \sin a' = E [\sin a \cos a' - \cos a \sin a'] \quad (22)$$

$$= E \sin (a - a')$$

where a is the quantity to be read on the instrument and a' is the angular position of the servo motor. It is clear that the servo will seek a position that causes the output of the motor amplifier to be zero. Thus, a' = a, providing the desired indication on the instrument. Therefore, for the form of analog implementation used in UDOFT, it is sometimes necessary to generate ± sine and ± cosine of the angular quantities that are to be displayed. The heading indicator and the gyro-horizon therefore may be actuated by:

$$\left.\begin{aligned} \pm l_1 &= \pm \cos\theta \cos\psi \\ \pm l_2 &= \pm \cos\theta \sin\psi \end{aligned}\right\} \quad \text{Heading} \quad (23)$$

$$\left.\begin{aligned} \pm m_3 &= \pm \cos\theta \sin\phi \\ \pm n_3 &= \pm \cos\theta \cos\phi \end{aligned}\right\} \quad \text{Roll angle} \quad (24)$$

$$l_3 = -\sin\theta \quad \text{Pitch angle} \quad (25)$$

Since the pitch attitude indicator does not rotate more than ±90°, cos θ is not required; therefore pitch angle is the direct function of $-l_3$.

It is apparent then that the problem arises in the heading and the roll angle outputs. as θ approaches 90°, l_1, l_2, m_3, and n_3 approach zero, thereby reducing the effective cos ψ, sin ψ, sin ϕ and cos ϕ signals to the instrument positioning servos. However, by shifting the quantities l_1, l_2, m_3 and n_3 to the left, while they are still in binary form, as l_3 approaches unity, these quantities are increased proportionally by a factor of two for each left shift, thereby maintaining the ratio of the sines to the cosines and compensating for the signal degeneration resulted from cos θ diminishing to zero.

The compensation is accomplished in the program by inspecting the absolute value of l_3, determining the number of single phase left-shifts required to raise l_1, l_2, m_3, and n_3 to significant values, and finally, performing the required number on l_1, l_2, m_3 and n_3.

The range of values of l_1 for which the quantities l_1, l_2, m_3, and n_3 are adjusted is:

$0.000000 \leq \lvert l_3 \rvert < 0.866578$	shift left 0
$0.866578 \leq \lvert l_3 \rvert < 0.968691$	shift left 1
$0.968691 \leq \lvert l_3 \rvert < 0.992272$	shift left 2
$0.992272 \leq \lvert l_3 \rvert < 0.998103$	shift left 3
$0.998103 \leq \lvert l_3 \rvert < 0.99541$	shift left 4
$0.999541 \leq \lvert l_3 \rvert < 1.00000$	shift left 5

b.) Ground Position

In the F-100A MB-3 analog flight simulator, it is necessary only to provide ground speed and heading information to the cross-country and approach recorders (plotting boards). Ground speed is resolved by means of the associated recorder electronics into north and east components as a function of heading. The resultant velocities are integrated in the plotting board drive mechanism.

Since heading data is obtained from the heading instrument servo, only ground speed had to be computed:

$$\text{Ground Speed} = u\cos\theta + w\sin\theta \tag{26}$$

However, since cos θ was not available readily, it was obtained from a Taylor series expansion in powers of sin θ. Thus,

$$\cos\theta \approx 1 - 1/2\ \sin^2\theta + 1/8\ \sin^4\theta \tag{27}$$

Since sin θ is available as -1_3, the final form of the ground speed calculation becomes:

$$\text{Ground Speed} \approx u\left(1 - 1/2\ 1_3^2 + 1/8\ 1_3^4\right) - w1_3 \tag{28}$$

The approximation for cos θ is satisfactory for $\theta \leq 50°$; beyond this point, the error in the resulting ground track is intolerable. However, if it is unlikely that flight will be sustained at these relatively high pitch angles, the long term error resulting from the short term inaccuracy of the approximation will be negligible.

A much improved record of ground track would be possible by using a conventional positioning-type recorder rather than an integrating-type recorder, and requiring the computer to calculate the coordinates of ground position. Since the direction cosines are available, the calculations would be simply

$$S_x = \int V_x dt = \int (u1_1 + vm_1 + wn_1)\, dt \tag{29}$$

$$S_y = \int V_y dt = \int (u1_2 + vm_2 + wn_2)\, dt \tag{30}$$

5.4.3 Solution Rate

The most desirable program iteration rate for the solution of the various aircraft parameters would be infinitely high.

An infinitely high solution rate, to be effective, requires infinite precision. Since this is not easy to achieve without using an infinitely fast and large computer, some compromise in the solution rate must be made. The solution rate may be reduced until one or more of the following appears:

1. An inordinate amount of phase lag occurs between successive iterations of the simulation program, resulting in a simulation which does not adequately approximate the dynamic characteristics of the vehicle.

2. The inaccuracy of the numerical integration method becomes intolerable.

3. Rapid changes in inputs are lost due to the low input data sampling rate.

4. Instrument movement is noticeably discontinuous due to low output multiplexing rate.

The solution rate of twenty program iterations per second selected by the Moore School has proven to be adequate for the simulation of the F-100A aircraft. (It appears that the selection of this solution rate resulted from the initial studies concerned with the determination of the most suitable method of numerical integration*). This is not

*Flight Trainer Digital Computer Study, University of Pennsylvania, Moore School of Electrical Engineering Research Division Report 54-08, 1 July 1963.

meant to imply that a selection of a twenty cycles per second solution rate is unimpeachable. For the simulation of models characterized by higher or lower natural resonant frequencies, this iteration rate may be much too low (e.g. simulation of the X-20 adaptive flight control system indicates the necessity for a solution rate as high as 320 iterations per second) or much too high (e.g. simulation of a submarine indicates a solution rate of four iterations per second is acceptable). Even for the F-100A program it was not mandatory to recompute all variables every twentieth of a second. The power plant program could just as well have been executed at a considerably slower rate, such as five solutions per second. However as the solution rate is changed, the integrity of the numerical integration method must be retained. This may well require a modification to the form of numerical integration in order to make it applicable, within tolerance, at the different solution rate.

Thus far, the comments regarding solution rate have concerned themselves with the problems associated with a low solution rate, tacitly assuming that as the solution rate is increased toward infinity the problems are minimized. Unfortunately this has not been found to be true. The effect of the iteration interval on reaching the proper velocity for steady-state straight and level flight equilibrium is a case in point. It is conceivable that with a very high solution rate (i.e. Δt very small) the cumulative effect of a very small forward acceleration, $\dot{u}$, would be lost long before the steady-state velocity was attained. The problem arises not only from the short iteration interval but also from the fact that the binary number word length in the computer is finite, in the case of UDOFT, twenty binary bits. Therefore, if a very high solution rate is required (because of the high natural resonant frequencies of the physical system being simulated), consideration must be given to increasing precision and therefore the length of the numbers that are used in the calculations.

5.4.4 Method of Numerical Integration

In the numerical integration of a set of differential equations, the values of the dependent variables, x_i are obtained in a step-by-step manner. Assume that the past values of x_i and their first derivatives with respect to time, $\frac{dx_i}{dt}$ or more simply $\dot{x}_i$, are known for all instances of time t up to and including the instant of time t_n. These values may be used to "guess" at the values of x_i one time interval later, i.e. at t_{n+1} which is equal to $t_n + \Delta t$. The formula used for "guessing" is referred to as an "open quadrature formula," symbolized by the letter "O."

The "guessed" value of x_i thus obtained is ued to obtain an approximate value of the derivative $\dot{x}_i$, at the time t_{n+1}. This approximate derivative is then used to obtain a better approximation of the true value of x_i at t_{n+1} using a "closed quadrative formula," symbolized by letter "C." By recalculating the derivative of the improved approximation, and using it in the closed quadrative formula, a second improved approximation is obtained. This short looping process is usually repeated until successive values of x_i at time t_{n+1} remain unchanged; these values should then constitute the solution to be differential equations for time t_{n+1}.

For example, consider the case of the differential equation

$$\frac{dx}{dt} = \dot{x} = f(x, t) \tag{31}$$

whose solution is known for $t \leq t_n$. Let the following open quadrature formula 0_{31}, using three past values of the ordinate x and one past value of the derivative $\dot{x}$, be employed to guess the value of x at time $t_{n+1} = t_n + \Delta t = t_n + h$:

$$x'_{n+1} = a_1 x_n + a_2 x_{n-1} + a_3 x_{n-2} + h\left(b_1 \dot{x}_n\right) \tag{32}$$

The prime (') indicates that x'_{n+1} is only approximate. The first guess at the derivative is then the corresponding value.

$$\dot{x}'_{n+1} = f\left(x'_{n+1}, t\right) \tag{33}$$

Next, let the following closed quadrative formula C_{21}, using two past values of x and the derivative just computed be employed to refine the approximation of x_{n+1} derived from the open quadrature formula.

$$x''_{n+1} = c_1 x_n + c_2 x_{n-1} + h\left(d_1 \dot{x}'_{n+1}\right) \tag{34}$$

The double prime (") indicates that x''_{n+1} is an improved but not necessarily the final accepted value. The two latter equations are then used repeatedly until successive values of x''_{n+1} are identical.

From the example it can be seen that in the symbolic designation of an open quadrature formula x_{mn}, the subscript m indicates the number of past values of the ordinate that are to be summed and the subscript n indicates the number of past values of the derivative that are to be summed. In the symbolic designation of the closed quadrature formula, C_{pq}, the subscript p, like the subscript n, indicates the number of past values of the ordinate, and the subscript q indicates the number of past values of the derivative, including the present estimated value of the derivative. Thus, $O_{41}\ C_{41}$ would use four ordinates and one derivative in both the open and the closed quadrature formulas, while $O_{33}\ C_{33}$ would use three ordinates and three derivatives in each quadrature formula.

The mixed quadrature formula $O_{mn}\ C_{pq}$ consists of the open formula O_{mn} followed by a single application of the closed formula C_{pq}, the result of this being the accepted value of x_{n+1}. All the ordinates, i.e. the x_{n-j}'s ($j = 0, 1, 2, \ldots n$), in both the open and the closed parts of this mixed formula are the final values computed from the closed quadrature formula at previous instants of time, and all the derivatives, i.e. the $\dot{x}_{n-j}$'s, are computed using the values of the ordinates derived from the open quadrature formula.

In connection with their design and feasibility studies of UDOFT, the Moore School investigators developed a technique which enabled them to choose a numerical integration method, with a compatible integration interval, which would produce the desired solution. This technique employed a graphical method for displaying the characteristics of any specified integration formula in a "stability" chart, * from which the accuracy and the stability of the numerical solution obtained by using the formula could be estimated in advance. By means of the stability charts and a shifting technique, ** two formulas, having the same stability chart, were obtained from the classical mixed quadrature formula $O_{12}\ C_{12}$; namely the open formula, O_{33} mod Gurk, and the mixed formula, $O_{30}\ C_{31}$ mod Gurk. Although the open formula is preferable from the standpoint of computational simplicity, it causes the actual solution to lag slightly behind the desired solution. Apparently the lag was not appreciable enough to cause any serious concern; therefore, the quadrature formula O_{33}, mod Gurk was established as the numerical integration method for UDOFT.

The operational F-100A flight simulation program using O_{33} mod Gurk with an integration interval of 50 milliseconds provides adequate simulation for aircraft frequencies ranging from 1 cps (snap roll) to one cycle in two minutes (phugoid). A current study, utilizing the UDOFT F-100A simulation as the model, conducted by the Moore School tends to uphold the applicability of O_{33} mod Gurk to the F-100A simulation problem. However some doubt has been cast by Kase*** on the basis that the results using O_{33} mod Gurk with different quadrature intervals were not compared with another independent solution not using O_{33} mod Gurk. Regardless of what has been said and what is being said about the validity of O_{33} mod Gurk as an applicable numerical integration method for real-time flight simulation, no operating problems have arisen from its use to date.

5.4.5 Use of Time in the Operational Program

Of all the aspects of the simulation programming task, the use of computer time is probably the most critical; it is also difficult to control. This difficulty, as experienced during the formulation of the F-100A program, was increased by the absence of not only pre-established programming techniques, but also a clear concept (from the beginning of the task) of the proportions of the overall program. The brute-force but effective approach was to minimize time through the program at all costs (i.e. at the cost of computer storage, the cost of clarity or organization, and the cost of program test points) until it was apparent that the simulation program could be executed within the iteration interval.

---

* MSEE Report 54-09

** MSEE Report 54-25

***MSEE Report 61-19

Since it represents the single largest portion of the total program, prime attention was given to the function generator. The approach, already presented in Section 4.5.1, though cumbersome, reduced function generation time to a minimum. Aside from this very important segment of the program, no spectacular efforts were made to speed-up the program. At the beginning of program formulation, instructions were used judiciously; toward the end, as it became evident that the running time would not exceed the 50 millisecond interval and as the deadline for acceptance testing was approached, less care was taken to minimize the program. The penurious use of time, though necessary (since this type of program had never been attempted before), had its repercussions. Had some of the program units been allowed to consume more running time, they could have been programmed in a more straightforward manner; also, more intermediate results could have been stored. As a result, checkout would have proceeded at a faster rate; perhaps as much time as a month would have been saved. However, there was no way of knowing in the beginning how much, if any, unused computation time would be available after the complete program had been assembled.

The following conclusions may be drawn from this experience:

1. When simulating one computer on another computer for the purpose of checking out a program written for the simulated computer, provision should be made in the simulation computer program to record elapsed time for the simulated computer.

2. Special attention should be given to optimizing (time-wise) the subroutines which are used repeatedly, to arrangement of tables, etc.

3. When a choice must be made between saving a few microseconds of computation time and storing an intermediate result which might be useful, the decision should favor the latter. These "unnecessary" instructions may be so identified on the assembly listing to facilitate their deletion if and when necessary.

4. Time estimates of the various program units should be made before any programming is undertaken. The critical program units should be written first to allow ample time for improvements resulting from the programmer's increased knowledge of the problem.

5. Time estimated and time consumed should be monitored constantly at the highest level, both to ensure the formulation of a computer program which can do all that it must do in the required time interval and to allow improvements in the program if running time permits.

As a result of the programmers being extremely time-conscious, the F-100A program is executed in less than 50 milliseconds. The actual running time varies between thirty and thirty-five milliseconds. As a result, however, there is very little excess capacity in either the instruction memory or the number memory. This has made it difficult to include automatic testing and monitoring programs in the flight simulation program. Usually, the F-100A icing routine, which is unimportant when research testing rather than training is being conducted, is removed to allow the insertion of these programs.

5.4.6 Control of Precision in the Operational Program

The basic precision problem is the same for all digital computers and all computational programs written for them: if p, a large number, is to be added to k, a small number, then the ratio $|p|/|k|$ must be less than 2^n where n is the number of bits in the binary word (strictly speaking, $|p|/|k|$ must be less than $2^n - 1$). Otherwise, k will appear to be equal to zero.

The difference in the precision problem between fixed point and floating point (whether built-in or programmed) is this: in fixed point, if p above refers, for example, to altitude, and k to some increment of altitude, the maximum allowable value of p must be determined before the program writing commences; this in turn establishes the minimum increment k which can affect the calculation. On the other hand, in floating point, although there is still a restriction on the maximum allowable value of p, the minimum increment k is a function of the current value of p, not the maximum allowable value of p; that is, when p is equal to zero, the minimum value of k which affects the computation in floating point (normalized mode) is $0.5 \times 1/(2^{2^{m/2}})$, where m is the number of bits in the exponent of the floating point word.

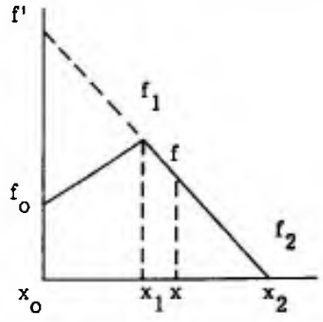
As p approaches its maximum value, the minimum value of k which can affect the computation approaches $p_{max}/2^n$, where n is the number of bits in the fractional part of the floating point word. In contrast, using fixed point computation, the minimum value of k which can ever affect the computation is $p_{max}/2^n$ (actually a somewhat larger number since the numerator is not truly p_{max}, but the next greater power of two).

These considerations affect programming in several ways, and the way in which they are handled determines, within the natural limits of the computer, the precision which is ultimately obtained from the system as a whole (assuming that the mathematical model is adequate). For example, suppose the problem is to add a series of numbers of widely varying magnitudes. In floating point, the program would be organized to add the small numbers first, otherwise they would have no effect, even though their sum is relatively large. In straight fixed point computation, the same order of computation will not have the same result since the scaling is predetermined; the best solution is a shift in scale factor (really a species of programmed floating point). Another method of control available to the programmer is to assure that no partial sums will be larger than the final sum; otherwise an artifically high scaling factor must be chosen and an unnecessarily high minimum increment will be the result. The magnitude of the partial sums is, of course, affected by the order of the computation.

Insofar as this affects the generation of the aerodynamic coefficients and similar parameters, the controlling factor is the accuracy with which the programmer knows the maximum values of these parameters and the intermediate values in their calculation. On the basis of the estimated maximum values, the scale factors are chosen, and this choice in turn determines the minimum increment which will affect the calculation. The order in which, for example, the forces due to different control surfaces are summed determines the maximum intermediate value of this addition.

This problem was handled successfully in the F-100A operational program; it is disappointing to note, however, that no methodology emerged for expeditious handling of the required analysis in the future.

Occasionally, a choice of methods of computation presents itself. For example, given the representation:



There is a choice of interpolating between f_1 and f_2 to determine f, or one can extend the line f_1 f_2 to the y-axis and interpolate between f' and f_2. It is possible that this procedure would be faster given a certain computer instruction repertoire. However, it would, in general, result in decreased precision in the result since a wider range of numbers is involved in the computation. Accordingly, the first method was used in the UDOFT programs.

The integration interval affects precision also. Reconsider the general example of Section 5.4.3 where it was desirable to calculate $u + \Delta t\dot{u}$. For a given scale factor for u and a given $\dot{u}$, the value of Δt will determine whether $\dot{u}$ will ever affect the computation. Thus, the selection of a very small Δt imposes even greater precision problems. A simple acceptance test like attaining a steady-state true airspeed for a given thrust may not be possible if $|\dot{u}|$ decreases rapidly and Δt is very small, even though all parameters involved--lift, drag, and thrust--are computed with extreme precision.

5.5 Simulation Program Organization

In preparing a large-scale real-time simulation program, the designer (programmer) must exercise care to insure that:

1. Program debugging is facilitated.
2. Programs (or sub-programs) are handled as supplements to the total program.
3. Forseen changes (and most unforeseen ones) are simple to implement.
4. Computational time is used efficiently.

Some of these exemplary qualities have been discussed in preceding sections. However, they have been discussed in the context of individual areas of the total simulation problem; i.e., integration, function generation, vehicle orientation, etc. No matter how coherent the mathematics and the logic of the individual parts of the simulation program may be, the interplay between these parts must receive the same level of consideration, otherwise, the ultimate program will lack the integrity demanded of it.

The following general description of program design is given to indicate an approach to developing real-time simulation programs.

5.5.1 General Simulation Program Formulation

For ease in analyzing the requirements for a complete simulation program, the physical systems comprising a manned weapon system can be considered as five independent computational tasks:

1. Vehicle dynamics
2. Propulsion system
3. Vehicle subsystem
4. Guidance and control systems
5. Airborne systems

It is recognized immediately that none of the above systems is capable of operation without an interchange of data with one or more of the other systems. However, for the purpose of analysis and programming, each of these systems can be examined separately and their programs developed independently. Ultimately, the parts must be assembled into a whole program. The overall program configuration is depicted in figure 73. Associated with each of these main program sections is a control program designated sub-control. The function of these sub-control programs is to make all decisions concerning the manner in which their respective computational blocks are used and to supply each computational block with the necessary input data. Also shown in figure 73 is a program control block which encompasses the highest level of control within the simulation program. Its basic function is to control the sequencing of the other blocks within the program and to provide the link between the computer programs and the real-world.

The division of the simulation program into separate computational tasks results in:

1. Comparable programs – the manner in which one basic block is programmed does not affect another.
2. Simplified debugging – the programs can be debugged in sections.
3. Flexibility – changes can be made in the computation of quantities within any one block without affecting the other blocks.

The concept of dividing the control program into sub-control programs results in advantages for control programs similar to those listed above for computational programs. In addition, the sub-control concept of program control results in more efficient use of the

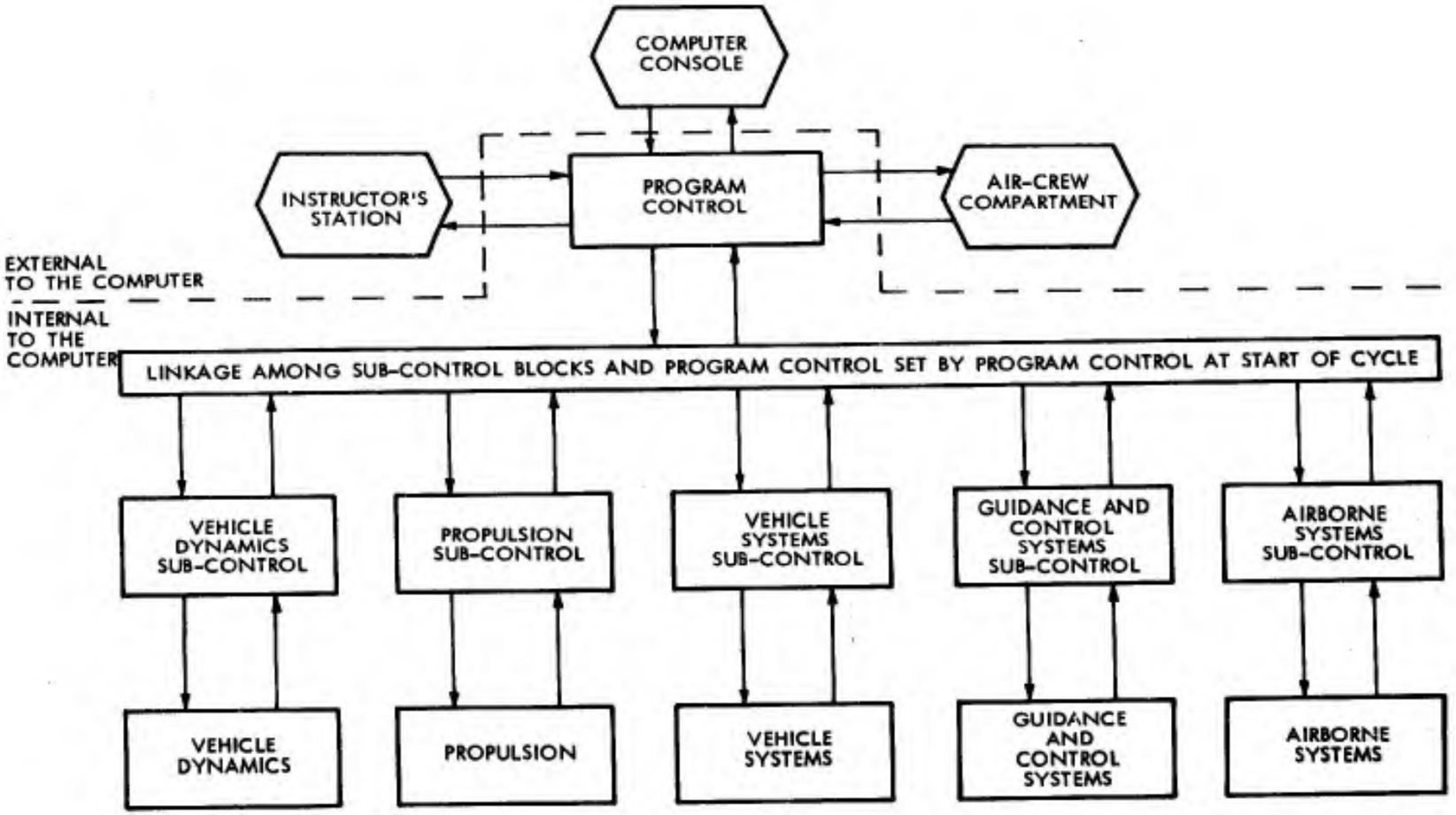


Figure 73. Block Diagram of Flight Simulation Program

available computation time (program cycle time), because the sequence and the extent of the computations can be varied with the existing conditions.

5.5.2 Programming Procedures

The simulation program must be capable of reproducing the actions of the vehicle and all its subsystems under abnormal as well as normal operating conditions. Accordingly, each system of the vehicle must be analyzed on an individual basis. The purpose of the analysis is to describe, by means of mathematical and logical equations, the normal response of the system to each of the pertinent independent variables. The effect of malfunctions at various levels within the system must then be analyzed in order to allow the incorporation of these effects into the descriptive equations. The resulting equations reflect the operation of the particular system under all conditions. These same equations form the starting point for the preparation of the subroutine.

Once the equations are known, a subroutine flow chart is prepared. The flow chart represents diagramatically the program, or flow, of mathematical and logic operations that are necessary in order to solve the descriptive equations. The next step is to fix this flowchart in the overall system organization flow charts.

At this point the program preparation task takes on a more important aspect, that of of program integration. The task concerns itself primarily with the development of the program-control routine. Implicit, however, in the development of this control program is a thorough understanding of the inter-relationship of the descriptive equations and of the problems that may arise due to improper use of the numerical integration formula. Therefore, caution must be exercised during the program integration phase to insure that the descriptive equations are solved in the proper order and that proper values of the variables are used in the solution of the individual equations. An example of a problem which occurred due to improper ordering of subroutines is presented in Section 6.2, Program Control.

Notwithstanding these subtle considerations, the task of preparing the program-control routine is somewhat mechanical in nature. The purpose of the routine is singular and its preparation is straightforward. However, if programming problems are to be minimized, consideration should be given to more than just the development of a single computer routine. By approaching the preparation of this routine with an appreciation for the task of overall program preparation and integration, standards may be derived to facilitate:

a.) Program coding.

b.) Program debugging.

c.) Communication between programmers who are assigned to individual parts of the total simulation program.

d.) Communication between computer development engineers and programmers.

Concurrent with the development of the program-control routine, the individual computational subroutines are coded by the programmers using only the descriptive equations and the flow charts.

The next step is subroutine checkout, using either the ultimate computer, or, as in the case of the UDOFT program, a general-purpose digital computer simulation of the ultimate computer. Upon completion of subroutine checkout, the programmer is required to incorporate any changes made during the coding and checkout phases which did not appear in the initial subroutine flow chart. This material is then incorporated into a detailed subroutine write-up, which when completed, contains the following:

1. Subroutine name.

2. Purpose.

3. Detailed coding flow chart.

4. Calling sequence (entry and exit designations used by the program-control routine).

5. Inputs and outputs (including required scaling).

6. Outside constants to be used within the subroutine.

7. Outside library routines necessary for successful subroutine operation.

8. Simulation mathematics and logic.

9. Subroutine description (narrative).

10. Checkout results (to avoid inefficient use of computer time during the debugging or checkout phase, all subroutines should be hand-traced before being run on the computer. The same inputs used in the hand-trace are used in the machine checkout. Results of both methods must be consistent. Both sets of results should be listed in the program write-up).

11. Symbolic program listing.

Some techniques and procedures presented in the preceding discussion were not used during the development of the UDOFT simulation programs. The material represents those precepts that extensive experience with digital real-time simulation program development has fostered. Had this approach been utilized at the beginning of the analysis and programming task, the task undoubtedly would have been executed more effectively and efficiently.

Initially, the analysis and programming task was not approached systematically. Each of the subroutines was prepared in vacuo, i.e., each subroutine was considered as a separate entity. As such, the routines were checked out on an individual basis; little attention was paid to the checkout of interrelated groups of routines. It was apparent, midway through the task, that control and checkout programs were needed to facilitate such an operation. This deficiency was rectified by the development of the program-control concept. However, during the latter stages of the programming task, it could not right all the wrongs that had been committed.

One other major aspect of the program organization which was not approached in a systematic manner and could not be changed because of the disruptive effect it would have had, was the separation of the logical expressions and the computations within the subroutines. Consequently, the checkout task was more difficult than necessary. If the logic and the computational tasks had been separated, a more flexible program could have been developed, and the debugging and checkout procedures would have been simplified. This is the reason for the inclusion of program sub-control routines, as well as the more general program-control routine, in the idealized approach to the development of real-time digital simulation programs.

SECTION VI

SIMPLIFIED DESCRIPTION OF THE F-100A SIMULATION PROGRAM

The discussion of the UDOFT system thus far has been limited to the development of the digital computer and to fundamental real-time simulation program problems. This section treats aspects of the computer's application to a specific real-time simulation problem, the simulation of the F-100A.

Valid real-time digital simulation requires a program that will accept the real-world commands of the pilot and introduce these inputs into a system of nonlinear differential equations. The coefficients ot these equations are a function of the basic aerodynamic parameters (α, ψ, Mach, h_p, etc.) which are the result of past computational history. The integration of these differential equations establishes new parameters from which the coefficients for the next cycle and the data for the cockpit displays are generated. Thus, pilot control movement results in changes in the linear and angular accelerations which are, in turn, integrated to yield new cockpit indications of airspeed, attitude, etc. In brief, this is what the aerodynamics portion of the simulation program accomplishes. In addition, the simulation includes the engine and a number of other flight systems (flight control, hydraulic systems, landing gear, speed brakes, and flaps).

6.1 Aerodynamic Equations of Motion (Logitudinal Plane)

The simulation of the aerodynamic properties of an aircraft describes the motion and the orientation of the aircraft in terms of the translation velocities, u, v, and w; the angular velocities, p, q_1, and r, which are a result of the forces acting upon the physical aircraft structure; and the parameters which describe the structure. The aerodynamic forces and moments are defined in terms of dimensionless coefficients which are functions of such basic parameters as Mach number, angle of attack, and yaw angle.

As a case in point, consider the equations for $\dot{u}$, $\dot{w}$, and $\dot{q}_1$ which describe the motion of the aircraft in the longitudinal plane.

6.1.1 Longitudinal Acceleration ($\dot{u}$)

The acceleration along the longitudinal axis of the aircraft, $\dot{u}$, is defined as:

$$\dot{u} = \frac{X_a}{M_i} - g \operatorname{Sin} \Theta - w_{q_1} + vr \tag{35}$$

where

X_a = Total forces along the airplane X axis

$$= X_s \operatorname{Cos} \alpha - Z_s \operatorname{Sin} \alpha + T - D_{wn} \tag{36}$$

where

X_s = Total drag force along the stability X axis

$$= 376q[C_D' + C_{D_{dt}} + C_{D_{\delta J}} + C_{D_{dc}} + C_{D_{L.G}}] \tag{37}$$

The equation for $\dot{u}$ describes the longitudinal acceleration of the aircraft relative to the earth in terms of body axis parameters. However, since the coefficient data for the forces are given in terms of the airplane stability axis parameters, it is necessary to transform the total forces from the stability to the body axis.

The coefficients utilized in the equations represent the simulator manufacturer's interpretation of the airframe manufacturer's estimated behavior of the particular coefficient as a function of the basic parameters. The interpretation, in the case of the F100

simulation, resulted in piecewise linear approximations which yielded the coefficient when evaluated for the basic parameters. The coefficients given in the equation for X_s are:

C'_D = basic airframe drag force coefficient

$$= \frac{f_9(Ma) + C'_L f_{10}(Ma) + C'^2_L f_{12}(Ma)}{1 - C'_L f_{11}(Ma)} \tag{38}$$

$C_{D_{dt}}$ = drag coefficient due to drop tanks

$$= C_{D_{dt}}(Ma) \tag{39}$$

$C_{D_{\delta J}}$ = drag coefficient due to speed brakes

$$= C_{D_{\delta J}}(Ma) \tag{40}$$

$C_{D_{dc}}$ = drag coefficient due to inflated drag chute

$$= 0.30585, \ \delta_{dc} = 1 \text{ or } 0 \tag{41}$$

$C_{D_{L.G}}$ = drag coefficient due to landing gear

$$= 0.0278 \tag{42}$$

6.1.2 Normal Acceleration ($\dot{w}$)

The normal acceleration, $\dot{w}$, is defined as:

$$\dot{w} = -\frac{Z_a}{M_i} + g \cos \Theta \cos \Phi - vp + u_{q_1} \tag{43}$$

where

Z_a = total forces along the airplane Y Axis

$$= Z_s \cos \alpha + X_s \sin \alpha - 0.053T \tag{44}$$

where

Z_s = total lift force along the stability Y axis

$$= 376q[C_{L_W} + C_{L_H} + C_{L_{\delta J}} + C_{L_{dt}}] \tag{45}$$

The coefficients for this equation are:

C_{L_W} = coefficient of lift due to wings

$$= [C_{L_{W_{1.1}}}(Ma) - f_1(\alpha_{WR}) f_5(Ma)] f_4(Ma) + C_{L_{W_0}} \tag{46}$$

C_{L_H} = Coefficient of lift due to stabilizer

$$= f_1(\alpha_{HR}) f_8(Ma) \tag{47}$$

$C_{L_{\delta J}}$ = Coefficient of lift due to speed brakes

$$= C_{L_{\delta J}}(Ma) \tag{48}$$

$C_{L_{dt}}$ = Coefficient of lift due drop tanks

$$= f_1(\alpha) f_8(Ma) \tag{49}$$

The basic problem with this equation for Z_s is that neither C_{L_W} nor C_{L_H} is stated explicitly in terms of Mach number and angle of attack. Rather, they are stated in terms of rigid wing angle of attack (α_{WR}) and rigid stabilizer angle of attack (α_{HR}). The solution of α_{WR} and α_{HR} is further complicated in that the equations implicitly define the dependent variables:

α_{WR} = rigid wing angle of attack

$$= \left\{\alpha - [C_{L_{W_{1.1}}}(Ma) - f_1(\alpha_{WR})] f_4(Ma) f_4(q) f_3(Ma)\right\} f_2(Ma) \tag{50}$$

α_{HR} = rigid stabilizer angle of attack

$$= \alpha + \delta_H - f_2(\alpha_{WR}) f_6(Ma) - [0.00233q + f_2(q) f_7(Ma)] C_{L_H} \tag{51}$$

In order to overcome the problems associated with the real-time digital solution of implicitly stated functions (Section 7.5.3, Short Period Longitudinal Response), the equations for α_{WR} and α_{HR} are restated in terms of Mach and the slopes and intercepts of the function of α_{WR} and α_{HR}.

$$\alpha_{WR} = \frac{\left\{\alpha_{n-1} - [b_1 - b_2 f_2(Ma)] f_4(Ma) f_1(q) f_3(Ma)\right\} f_2(Ma)}{\left\{1 - f_4(Ma) f_1(q) f_3(Ma) f_2(Ma)[a_1 - a_2 f_5(Ma)]\right\}} \tag{52}$$

$$\alpha_{HR} = \frac{\alpha_{n-1} - \delta_H - f_2(\alpha_{WR}) f_2(Ma) - [f_8(Ma)\left\{0.00233q + f_2(q) f_7(Ma)\right\}] b_3}{[1 + f_8(Ma)\left\{0.00233q + f_2(q) f_7(Ma)\right\} a_3]} \tag{53}$$

This requires a program which solves for the variables α_{WR} and α_{HR} in the various regions of the functions $C_{L_{W_{1.1}}}(\alpha_{WR})$, $f_1(\alpha_{WR})$, and $f_1(\alpha_{HR})$. The solution is valid only when the values computed for α_{WR} and α_{HR} are within the interval for which the data was substituted. The new coefficients a and b are the slopes and the intercepts, respectively, of the functions $C_{L_{W_{1.1}}}(\alpha_{WR})$, $f_1(\alpha_{WR})$, and $f_1(\alpha_{HR})$. The intercept data for these functions, unlike the intercept data for interpolation purposes in the function generator, are the

values of the piecewise linear functions of α_{WR} and α_{HR} when extended to the ordinate. The data for the above functions are:

$C_{L_{W_{1,1}}}(\alpha_{WR})$ $\qquad$ $f_1(\alpha_{WR})$

Range of α_{WR}	a_1	b_1	a_2	b_2
$-24 \leq \alpha_{WR} < -18$	000000	-1.2450	0.000000	0.00000
$-18 \leq \alpha_{WR} < -6$	0.461666	-0.4140012	-0.1583333	-0.2849998
$-6 \leq \alpha_{WR} < +8.25$	0.1151578	-0.0000580	0.3298245	0.00789470
$8.25 \leq \alpha_{WR} < 12.25$	0.0600000	+0.45500	-0.025000	0.48625
$12.25 \leq \alpha_{WR} < 18.00$	0.0095652	1.0728698	-0.03130434	0.56347817

Range of α_{HR}	a_3	b_3
$-40 \leq \alpha_{HR} < -20$	0.0078	-0.05800
$-20 \leq \alpha_{HR} < +14$	0.05570588	0.0001176
$-14 \leq \alpha_{HR} < 20$	0.033333333	0.31333338
$20 \leq \alpha_{HR} < 40$	0.0145	0.690000

The breakpoints were purposely modified from the original Melpar functions to minimize the number of decisions required by the simulation program to establish the value of α_{WR} and α_{HR}.

6.1.3 Pitching Acceleration ($\dot{q}_1$)

The remaining equation necessary to describe the motion of the aircraft in the longitudinal plane is the angular acceleration about the Y axis, $\dot{q}_1$:

$$\dot{q}_1 = \frac{Ma + 54200\, rp}{I_y} \tag{54}$$

where

Ma = total pitching moment about the airplane Y axis

$$= M_s + Z_S d - 1.33T \tag{55}$$

where

M_s = total pitching moment about the stability Y axis

$$= 376\, cq \left\{ C_{M_{\alpha_F}} + \frac{C}{2V_t} C_{M_{q_1}} q_1 + \frac{C}{2V_t} C_{M_{\dot{\alpha}}} \dot{\alpha} + C_{M_{W_a}} f_1(\alpha_{WR}) + C_{M_{dt}} \delta_{dt} + C_{M_{L.G}} \delta_{L.G} \right\} + 99 q \ell_H C_{L_H} + 115 q \ell p. \tag{56}$$

where α = throttle angle

The coefficients for this equation are:

$C_{M_{\alpha_F}}$ = basic flexible pitching moment coefficient

$$= C_{M_{\alpha_R}} - f_{14}(Ma)f_1(hp)f_3(\alpha_{WR}) \tag{57}$$

where

$C_{M_{\alpha_R}}$ = rigid pitching moment coefficient

The equations describing $C_{M_{\alpha_R}}$ takes on different forms for different intervals of α_{WR} and is given as follows:

$$-24 \leq \alpha_{WR} < -18 \qquad C_{M_{\alpha_R}} = f_{18}(M_s) \tag{58}$$

$$-18 \leq \alpha_{WR} < -6 \qquad = f_{17}(Ma) - f_{18}(Ma)\left(\frac{\alpha_{WR} + 18}{12}\right) + f_{18}(Ma)$$

$$-6 \leq \alpha_{WR} < 5 \qquad = f_{16}(Ma) - f_{17}(Ma)\left(\frac{\alpha_{WR} + 6}{11}\right) + f_{17}(Ma)$$

$$5 \leq \alpha_{WR} < 8 \qquad = f_{15}(Ma) - f_{16}(Ma)\left(\frac{\alpha_{WR} - 5}{3}\right) + f_{16}(Ma)$$

$$8 \leq \alpha_{WR} < 12 \qquad = f_{14}(Ma) - f_{15}(Ma)\left(\frac{\alpha_{WR} - 8}{4}\right) + f_{15}(Ma)$$

$$12 \leq \alpha_{WR} < 18 \qquad = f_{13}(Ma) - f_{14}(Ma)\left(\frac{\alpha_{WR} - 12}{6}\right) + f_{14}(Ma)$$

$$18 \leq \alpha_{WR} < 24 \qquad = f_{13}(Ma)$$

$C_{M_{q_1}}$ = pitching moment coefficient due to pitching rate

$$= f_{25}(Ma)f_2(hp) - f_{26}(Ma) \tag{59}$$

$C_{M_{\dot{\alpha}}}$ = pitching moment coefficient due to rate of change of angle of attack

$$f_{27}(Ma) - \frac{62,000 - hp}{64,000} = f_{28}(Ma) \tag{60}$$

$C_{M_{W_a}}$ = pitching moment coefficient due to air entering engine inlet duct

$$= C_{M_{W_a}}(Ma)\alpha \tag{61}$$

$C_{M_{dt}}$ = pitching moment coefficient due to drop tanks

The equation for pitching moment due to drop tanks takes on different forms for different ranges of angle of attack as follows.

$$-40° \leq \alpha < 0° \qquad C_{M_{dt}} = f_{22}(Ma) \tag{62}$$

$$0° \leq \alpha < 10° \qquad = f_{23}(Ma)\frac{\alpha}{10} + f_{22}(Ma)$$

$$10° \leq \alpha < 20° \qquad = [f_{24}(Ma) - f_{23}(Ma)] \frac{\alpha - 10}{10} + f_{23}(Ma) + f_{22}(Ma)$$

$$20° \leq \alpha < 40° \qquad = f_{24}(Ma) + f_{22}(Ma)$$

$C_{M_{\delta_J}}$ = pitching moment coefficient due to speed brakes.

$$= f_{20}(Ma) + f_{21}(Ma)f_2(\alpha) \tag{63}$$

$C_{M_{L.G}}$ = pitching moment coefficient due to landing gear

$$= C_{M_{L.G}}(\alpha_{WR}) \tag{64}$$

M_H = pitching moment due to horizontal stabilizer

$$= C_{L_H} \ell_H(Ma) \tag{65}$$

The pitching moment due to the horizontal stabilizer is given as a function of lift due to the horizontal stabilizer, C_{L_H}, and the variation of the stabilizer moment arm, ℓ_H. ℓ_H is defined as the distance from the airplane center of gravity to the horizontal stabilizer center of pressure. This length varies as a function of Mach number. Stabilizer deflection, δ_H, is implied in the generation of C_{L_H} and has been considered in the solution of the α_{HR} equations.

The remaining moment to be described in the pitching moment equation is the influence of a deployed drag chute:

M_{dc} = pitching moment due to inflated drag chute

$$= 115q\,\ell_p(\alpha) \tag{66}$$

The generation of the coefficients for the moments about the airplane Y stability axis is straightforward. The only coefficients which require special consideration are $C_{M_{\alpha_R}}$, equation 58, and $C_{M_{dt}}$, equation 62. In either case, the evaluation of the coefficient varies in accordance with the interval of the present value of α_{WR} and α_{HR}, respectively. Under program control the proper equation is selected and the coefficients are evaluated. In the case of $C_{M_{\alpha_R}}$, only the Mach functions corresponding to the proper interval equation are solved. This requires that $f_{13}(Ma)$ through $f_{18}(Ma)$ be evaluated outside of the function generator program. Thus, a maximum of only two functions of Mach are ever calculated rather than six.

6.2 Program Control

The three basic longitudinal equations described in the preceding section form one of many intermediate steps to the attainment of useful output data which requires, somewhere in the flow of calculations, the integration of these parameters. The numerical integration process in the simulation program requires strict adherence to a fixed program time interval. Under no conditions, therefore, may the extent of the combined calculations of aircraft aerodynamics, propulsion, and flight systems exceed the 50 millisecond program cycle. Every cycle of computation is fixed to 50 milliseconds; however, since the extent of the computation is contingent upon conditions established by the real-world input from the cockpit, the length of actual computation performed per cycle will vary from cycle to cycle. Only under steady-state conditions will the computation time per cycle remain fixed. (The number of decisions and the extent of branching will not vary.) These comments apply to the Normal mode of execution of the flight simulation program. The simulation program is characterized, however, by Freeze and Initialize (or Zero) modes of execution. Operation in the Normal, Freeze and Zero modes implies that the execution of the flight program is governed by some means which places the flight program in one of the three modes. This function is accomplished by the Governing Control Program. The Governing Control Program has the dual purpose of maintaining a constant computation interval and selecting and routing to the mode of computation for the next cycle.

In order to maintain the cycle time fixed at 50 milliseconds, the Governing Control Program is not initiated until the count in the Interval Timer has decreased to zero, at which time the timer is reset for the next 50 millisecond interval. (It is assumed that for any of the three operating modes, the time required for the computation is less than 50 milliseconds.)

The control aspect of the Governing Control Program selects the mode of operation for the next computation cycle. (Mode changing is performed only at the initiation of a new program cycle.) The modes, in order of priority, are Zero, Freeze, Crash, and Normal. (A priority is assigned to the modes since it is possible to specify the Zero, Freeze, and Crash modes simultaneously.) The selection of the mode is accomplished by inspecting three flags corresponding to the inputs for Zero, Freeze, and Crash. The Zero and Freeze inputs are manually controlled discrete inputs, while the Crash input is established by the Decision Routine. The program enters the Normal mode only when the above three flags are off.

Once the mode of operation is chosen, the chain of computational subroutines is connected, each subroutine being a link in the chain. A subroutine router sets the sequence of operations to be carried out for any given mode. This is achieved by using the following ground rules when preparing the individual subroutines:

1. During any one cycle there is only one entrance to and one exit from a subroutine.

2. Exit from a subroutine is accomplished by a SCRNM instruction. (This instruction resets the Sequence Counter to the contents of the addressed number memory register.)

3. The exit from one subroutine is the direct entrance to the next subroutine; that is, the contents of the register addressed by the SCRNM instruction is the effective address of the entrance to the next subroutine.

4. At the conclusion of the desired sequence of subroutines the last SCRNM instruction will address the number memory register which refers back to governing control.

The list of starting addresses of all subroutines to be used is stored twice. The two individual lists are in consecutive locations. One list is fixed, and is to act merely as a reference or library list to the other which is variable. Changes are made in the variable table to effect different routing. Variable table addresses are set by extracting the address data for the required sequence of operations from the fixed table.

It is highly desirable to initially order the fixed table addresses to the sequence of subroutines used in that mode in which the computer will operate most frequently. The

complete list of addresses can then be transferred to the variable table, thereby economizing on instructions (since this procedure can be coded in loop fashion). Figure 74 exemplifies the relation of the fixed and the variable tables to the subroutines.

To illustrate the operation of a subroutine router, assume that the sequence of subroutines to be followed in the next cycle is C, D, A, B (figure 74). The following would be the subroutine router to effect this sequence:

```
CLA  2002 }
NOP  0000 } Place C as first subroutine
STO  1002 }

CLA  2003 }
NOP  0000 } Place D as second subroutine
STO  1003 }

CLA  2000 }
NOP  0000 } Place A as third subroutine
STO  1004 }

CLA  2001 }
NOP  0000 } Place B as fourth subroutine
STO  1005 }

CLA  2005 }
NOP  0000 } Place return to Governing Control
STO  1006 }
```

This table of variables then becomes:

Variables Table

NMAD	Contents
1002	0011700
1003	0014600
1004	0005000
1005	0006700
1006	Address of G. C. Entry

In the case of the F-100A simulation program, the Normal mode of operation represents not only the most used mode but also the mode specifying the maximum number of available subroutines. Thus the fixed table for the Governing Control Program contains the entrance addresses of all subroutines used in the Normal mode of operation. In this mode, the subroutines and the order in which they are performed are:

Governing Control
Function Generator
Permute
Convert Input Variables
Aerodynamic Coefficients
Moments and Forces–Stability Axes
Moments and Forces–Airplane Axes
Accelerations
Velocity Vectors
Direction Cosines
Etcetera
Altitude
RPM
Icing
Percent Thrust
Total Thrust and Fuel Flow
Tailpipe Temperature
Mass of Fuel
Mass, Moments of Inertia, Center of Gravity
Hydraulic Pressure
Instruments
Decisions (Land, Air, Crash)

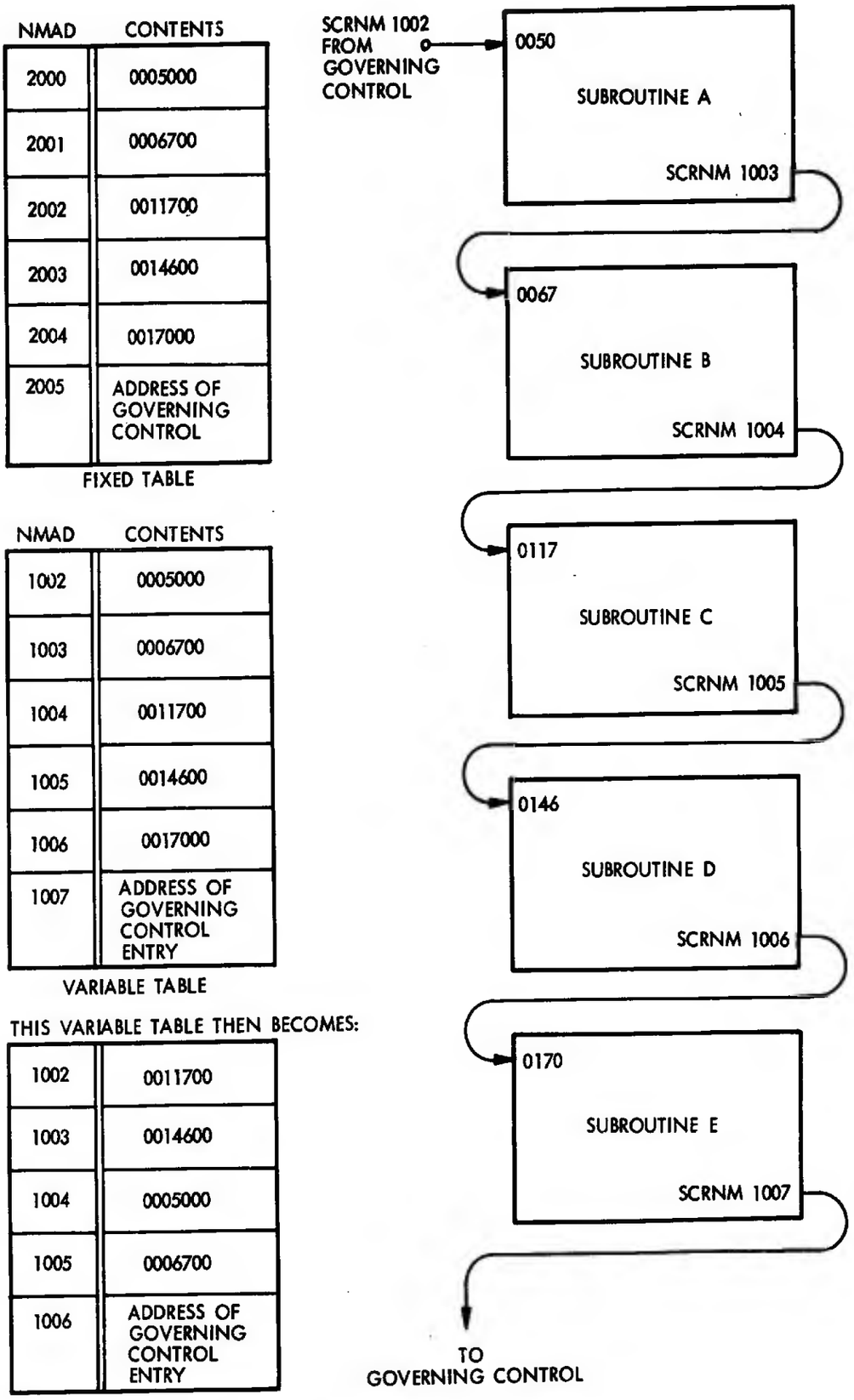


Figure 74. Diagram of Simplified Governing Control Program

In the Zero mode the specific order in which subroutines are performed is:

Governing Control
Function Generator
Permute
Convert Input Variables
Aerodynamic Coefficients
Moments and Forces-Stability Axes
Moments and Forces-Airplane Axes
Accelerations (under Zero Control)
Velocity Vectors
Direction Cosines (under Zero Control)
Etcetera
Altitude (under Zero Control)
RPM
Icing
Percent Thrust
Thrust and Fuel Flow
Tailpipe Temperature
Hydraulic Pressure
Instruments
Decisions (Land, Air, Crash)

In the Freeze or Crash mode the specific order in which the subroutines are performed is:

Governing Control
Function Generator

From the preceding description of Governing Control, it is apparent that the flight simulation program is sub-divided into a number of subroutines whose order of execution is controlled by the Governing Control Program (figure 75). The order of subroutine execution is not arbitrary, but is consistent with the flow of calculations which comprise the differential equations.

The order in which the equations are solved (the dependent variables are computed) and the resultant parameters are introduced into other equations has a significant effect upon the fidelity of the digital simulation. It was found that a delay as short as 50 milliseconds (one iteration cycle), introduced by improper ordering of subroutes, could introduce enough phase lag to cause the system of equations to become unstable in regions where the stability phase margin of the actual system of equations was quite small.

An instance of this arose during the testing of the simulation program for an operational, high performance aircraft (not the F-100A). The scheduled checkout of this simulation program was delayed seriously due to the presence of vehicle longitudinal instability at high supersonic speeds. Based on a detailed examination of the available computer output, familiarity with the computer program, and knowledge of the aircraft dynamics, it was hypothesized that the instability was due to a 50 millisecond time delay introduced into the program by the particular ordering of subroutines that was employed. This time delay would appear as a phase lag to the vehicle dynamics and could cause instability at the high Mach numbers where the aircraft stability phase margin is quite small. At low Mach numbers the aircraft is normally quite stable with a relatively large stability phase margin, and the added phase lag would not have any serious effect. An analysis of the vehicle longitudinal mode was performed to confirm the hypothesis.

As a first step in testing this hypothesis it was necessary to examine the ordering of subroutines in the program to determine the particular order of computation. Figure 76 indicates the original subroutine sequence. The block titles refer to the subroutine titles (similar but not identical to the subroutines of the F-100A simulation program which are described later in Section VI). The computations performed within each subroutine which pertain to the longitudinal instability problem are shown within the appropriate blocks.

Examination of the ordering of subroutines indicated that the current value of α is computed early in the cycle (Velocity Vectors Subroutine); however, the value of the lift coefficient, C_L corresponding to the current α is not computed until the end of the cycle

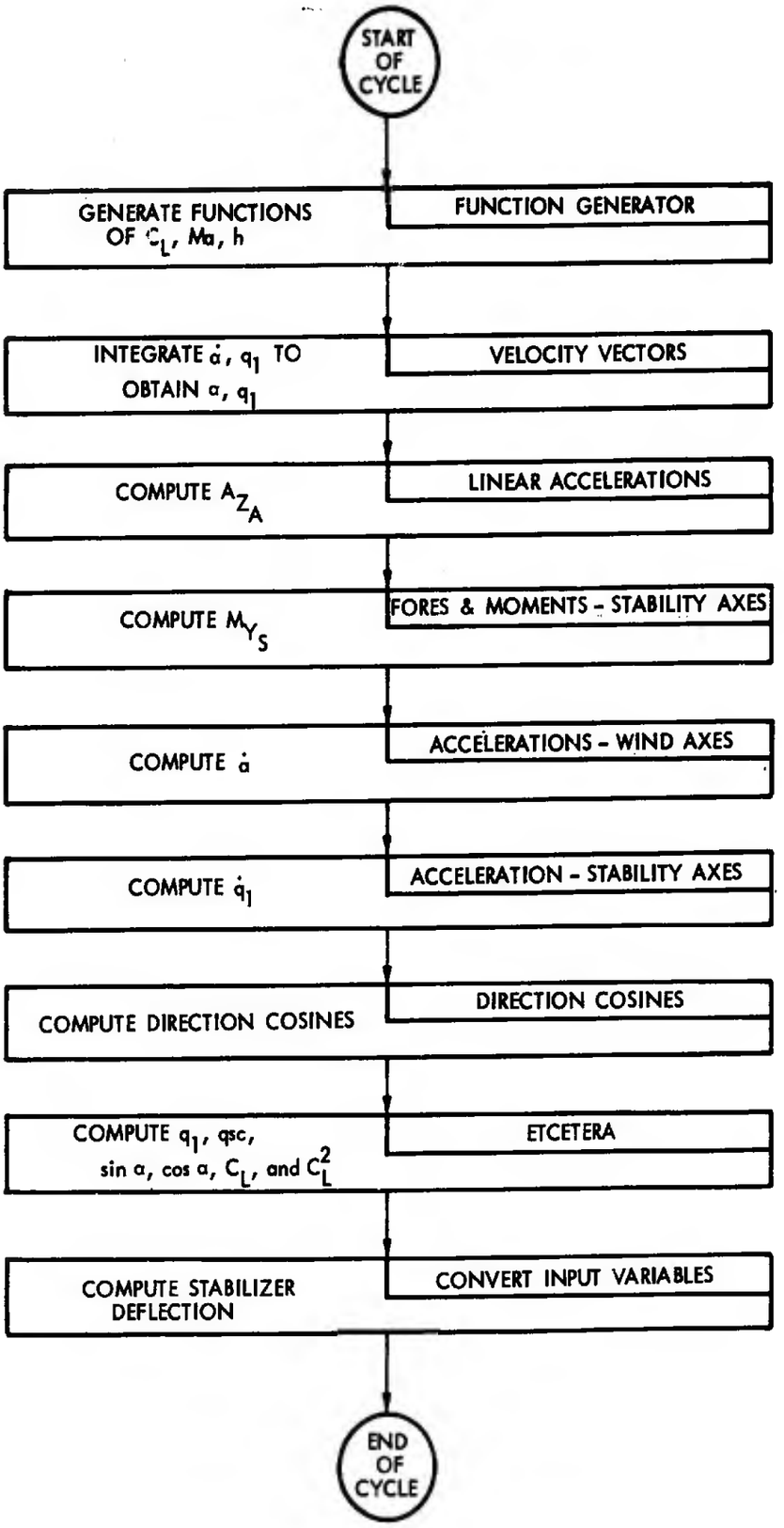


Figure 75. Governing Control Flow Diagram

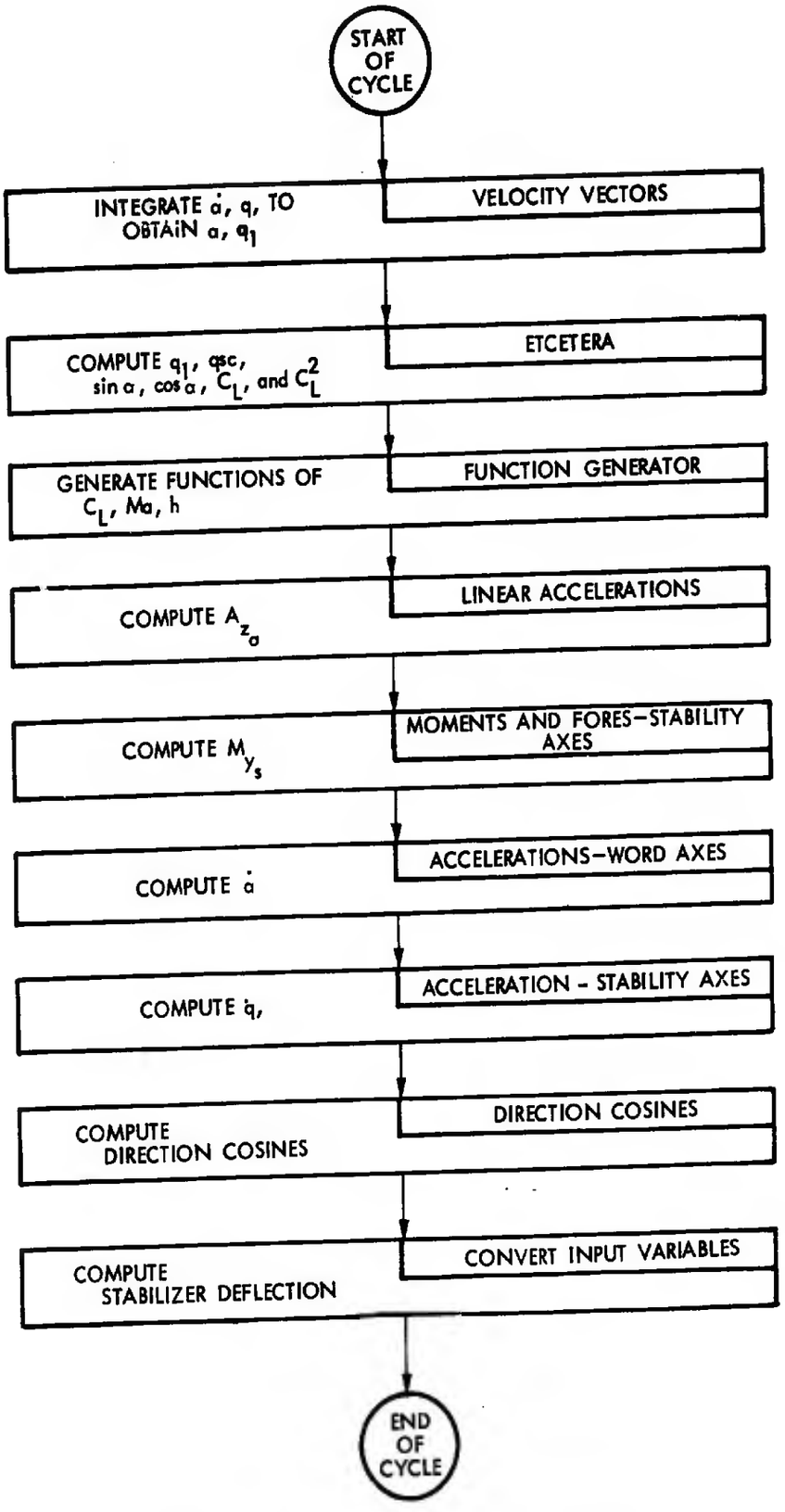


Figure 76. Initial Ordering of Subroutines

(Etcetera Subroutine). Consequently, the computation of $\dot{\alpha}$ and $\dot{q}_1$ (Angular Accelerations-Wind Axis - Subroutine and Angular Accelerations-Stability Axis - Subroutine, respectively) use and old value of C_L rather than the present value. The problem is compounded further by recognizing that the functions of C_L which are computed at the beginning of the cycle (Function Generator Subroutine) are based on the past value of C_L rather than the current value of C_L for that cycle.

On the basis of this examination it appeared that the value of C_L had been delayed by one iteration cycle of 50 milliseconds before being used in the computations of $\dot{\alpha}$ and $\dot{q}_1$. In order to determine the effects of this delay on the aircraft dynamic response and stability, a detailed analysis of the system of equations for the longitudinal mode was performed.

The equations that were examined were considered to be solved continuously, rather than periodically, in order to simplify the analysis. The next step was the formulation of a signal flow graph of the longitudinal dynamics which, in turn, was reduced to its simplest form. Conventional servo analysis procedures were used to determine that the open-loop gain and phase shift of the closed-loop system of equations were a function of frequency. The result was simply that the system of equations, without consideration of the 50 millisecond time delay introduced by C_L, was stable for all conditions. However, when the delay factor was considered, the analytical results were identical to the experimental results obtained from UDOFT. It was reasonable to conclude, therefore, that the instability problem was caused by the unwanted inclusion of a time lag in the simulation. This lag could be due only to the particular order in which the equations were solved.

Consequently, a new ordering of the subroutines was recommended (figure 77). In the revised sequence, the value of C_L corresponding to the current value of α is computed early in the cycle and is then used for all subsequent computations during that cycle. In addition, the functions of C_L are now based on the current value rather than the past value of C_L.

This recommended ordering of subroutines was implemented on UDOFT. As expected, the simulated vehicle is now stable over the entire flight regime and faithfully reproduces the desired longitudinal behavior.

6.3 Function Generator Subroutine

The primary purpose of the Function Generator subroutines is to evaluate the piecewise linear functions by solving the equation:

$$y_n = m_b(x_n - x_b) + y_b \qquad (67)$$

where: x_n is the independent variable

x_b is the breakpoint nearest to but smaller than the independent variable.

m_b is the slope of the segment

y_b is the value of the ordinate at the breakpoint

For the example of figure 78:

$$x_n = q$$

$$x_b = q_1$$

$$m_b = \frac{b_2 - b_1}{q_2 - q_1}$$

$$y_b = b_1$$

Breakpoints, slopes, and intercepts for each function are stored in a block in the following format.

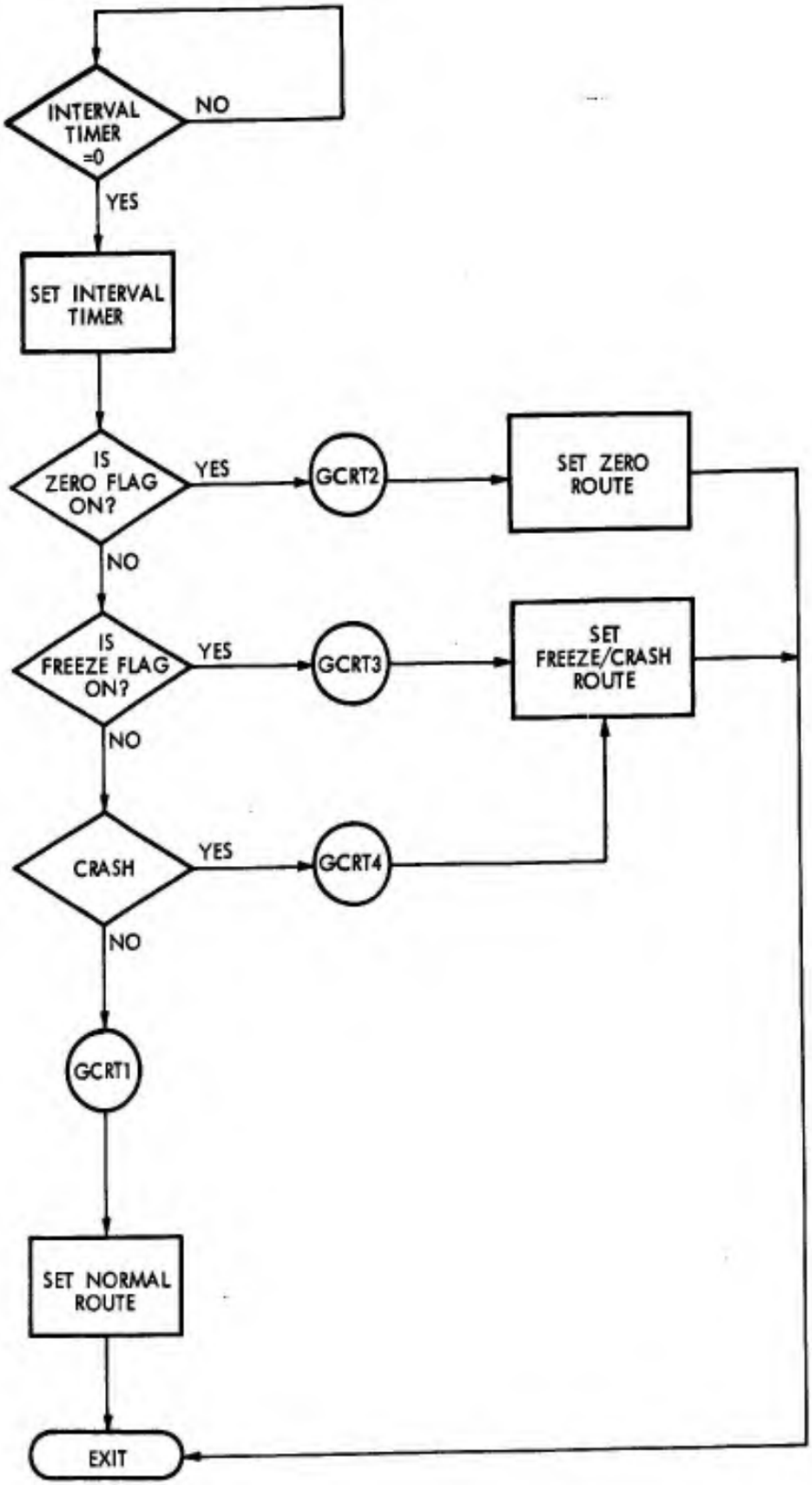


Figure 77. Recommended Ordering of Subroutines

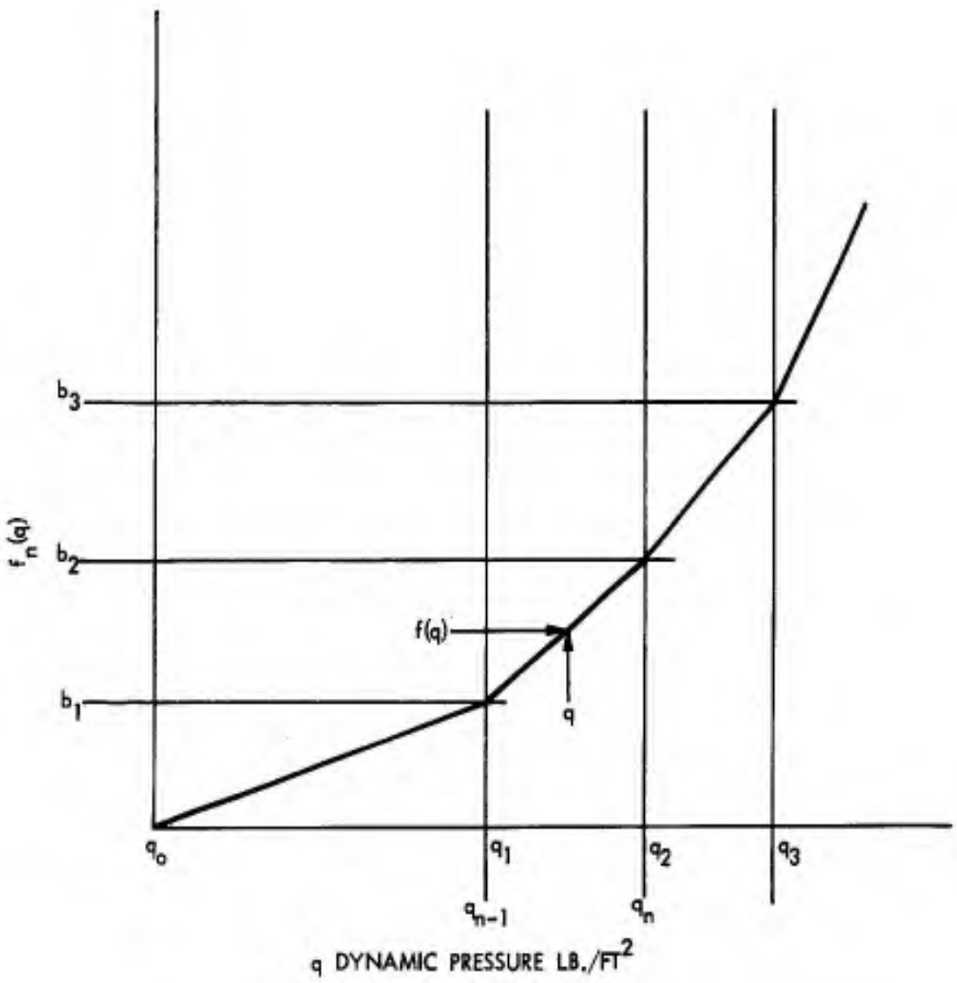


Figure 78. Plot of Typical Piecewise Linear Function Approximation

x_0 minimum x

m_0

y_0

x_1

m_1

y_1

.

.

.

x_{n-1}

m_{n-1}

y_{n-1}

x_n maximum x

As indicated, the data block will contain 3n + 1 data words for a function consisting of n straight-line segments. The number of straight-line segments in the functions that were used in the F-100A simulation program ranges from one to nine; thus, the data block lengths would vary from four to twenty-eight data words. If separate subroutines were used to evaluate each group of functions that consisted of a different number of segments, nine subroutines would be required. In order to simplify the program the following three data word block lengths were selected: 10 words, 16 words, and 28 words for functions with three, five, and nine straight-line segments respectively. Therefore, only three subroutines are required. If a particular function does not consist of three, five or nine segments exactly, the data words for that function must be introduced into the next higher data word block with the unused data words being made identically zero. As an example, a function with four segments would be placed at the beginning of the section of number memory reserved for functions with five segments (16 data words) in which the fourteenth, fifteenth, and sixteenth data words are made identically zero. An examination of the portion of number memory reserved for function data words will show that the data words for the functions of each independent variable are stored sequentially in order of increasing number of segments. This is essential if the function generator routine, as conceived and developed for the F-100A simulation program, is to maintain control of the program. This will become apparent as the discussion proceeds.

The function generator routine consists basically of five major operations (figure 79).

1. FGENT – entry into the program and initialization of Tally Register and return address, FGEND, for evaluation of functions of Mach.

2. FG2, FG7, and FG8 – setting-up Tally Register and return address, FGEND for cyclic evaluation of functions with five, three, and nine straight-line segments respectively.

3. FG3 – search of function data block for appropriate line segment, extraction of line segment data, and evaluation of function.

4. FGEND – storage of ordinate of evaluated function and supplementary operations if desired.

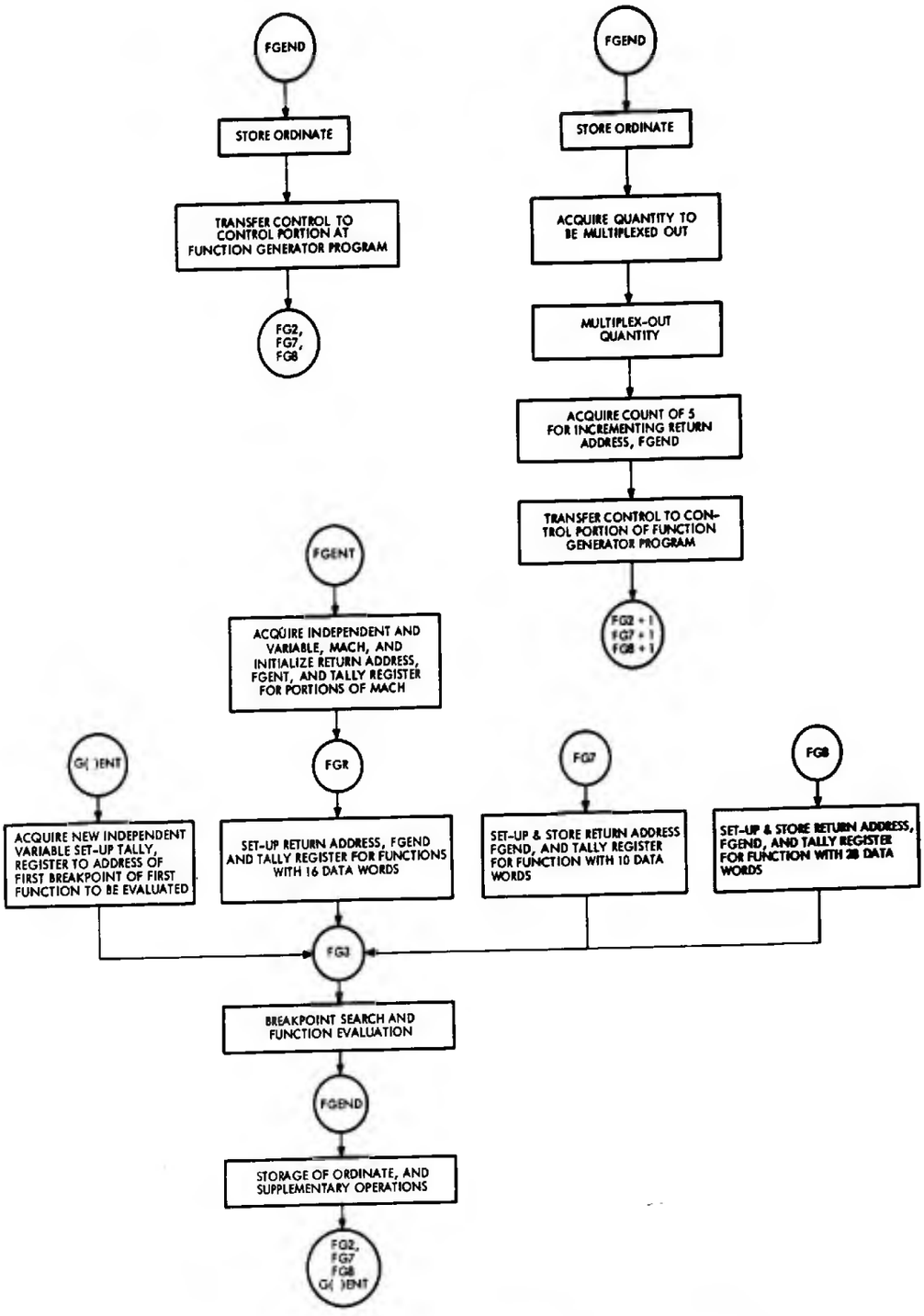


Figure 79. Function Generator Control Flow Diagram

5. G( )ENT–reentry to program for initialization of control for evaluation of functions of remaining independent variables.

A more detailed flow diagram is shown in figure 80. A better understanding of the operation of the function generator routine may be gained from an example showing the process by which a function is evaluated.

The program is entered initially at FGENT. The first operation is to acquire the independent variable for which functions are to be evaluated. Since the first functions to be evaluated are functions of Mach, the most current value of a Mach is extracted from its normal storage location in number memory and stored in the independent variable working location, FGIV. The next operation is to set the Tally Register equal to the address of the first function of Mach which will be evaluated. However, because the routine which follows increments the Tally Register by 16, the Tally Register is set initially to the first breakpoint address less 16. The return address, FGEND, by means of which the cyclic program can jump out to a multitude of two and five instruction routines, is likewise established. Normally, the short routines just mentioned are two instructions in length, containing both a store instruction (510) and an unconditional transfer of control instruction (SCR). The store instruction stores the ordinate of the evaluated function in the proper address in number memory, and the transfer of control instruction returns program control to either FG2, FG7, or FG8 for evaluation of the next function of the same independent variable. As with the setting of the Tally Register, the return address is set to an address which differs from the desired return address. The difference between addresses is a count of two, because the following routine, FG2, increments the return address by a count of two. At this point, the Tally Register is incremented by 16 and the return address, FGEND, is incremented by two. This apparent inefficiency of operation occurs because the normal incrementing of the Tally Register and the return address is performed by the routine entered at FG2 which is used for evaluation of functions with four and five breakpoints, respectively. (Considering the program as a whole, greater consistency of program design would have been maintained had the initialization process for functions of Mach been made similar to the initialization routine for the evaluation of functions of the remaining independent variables, which routine enters the main flow at FG3 rather than FG2.) Entry is made initially to the control program (FG2) which is used for evaluating functions with four or five breakpoints rather than to the control program for evaluating functions with three or fewer breakpoints (FG7), because there are no functions of Mach with fewer than three breakpoints. (A minor exception is one linear function of Mach which is evaluated immediately after entry is made into the function generator and the independent variable is acquired.)

With the establishment of the address of the first breakpoint of the first function (of Mach) to be evaluated, the search for the straight-line segment within which the independent variable (Mach) lies is initiated. The search is initiated with a test to determine if the independent variable is less than the breakpoint initially extracted from the function data block. If it is not, the Tally Register is incremented by three, thereby establishing the address of the next breakpoint of the function. The testing is continued until the first breakpoint greater than the independent variable is found. At this point, the next lower breakpoint is extracted from the function data block and the variable is again tested to determine if it is greater than or equal to this breakpoint. For the first search of the first function, only one of these tests will be performed. When the condition of the second test is satisfied, the search terminates. At this point, the Tally Register conveniently contains the address of the breakpoint that defines one parameter of the straight-line segment within which the independent variable lies. By modifying this address by factors of one and two the applicable slope and intercept data, respectively, are readily extracted from the function data block and the function is evaluated. This terminates the breakpoint search and function evaluation routine.

In the preceding discussion it was indicated that two breakpoint–independent variable tests were performed. Undoubtedly, some question may arise as to the need for the second test. (Is the independent variable greater than or equal to the breakpoint preceding that one which passed the first test?) The second test is unnecessary, as already pointed out, when the first function of each independent variable is evaluated. However, it may be necessary when evaluating all other functions. This will become apparent from the following discussion.

Assume that the first function, $f_1(Ma)$, of the independent variable has been evaluated. Further, assume that the function consists of five straight-line segments and that

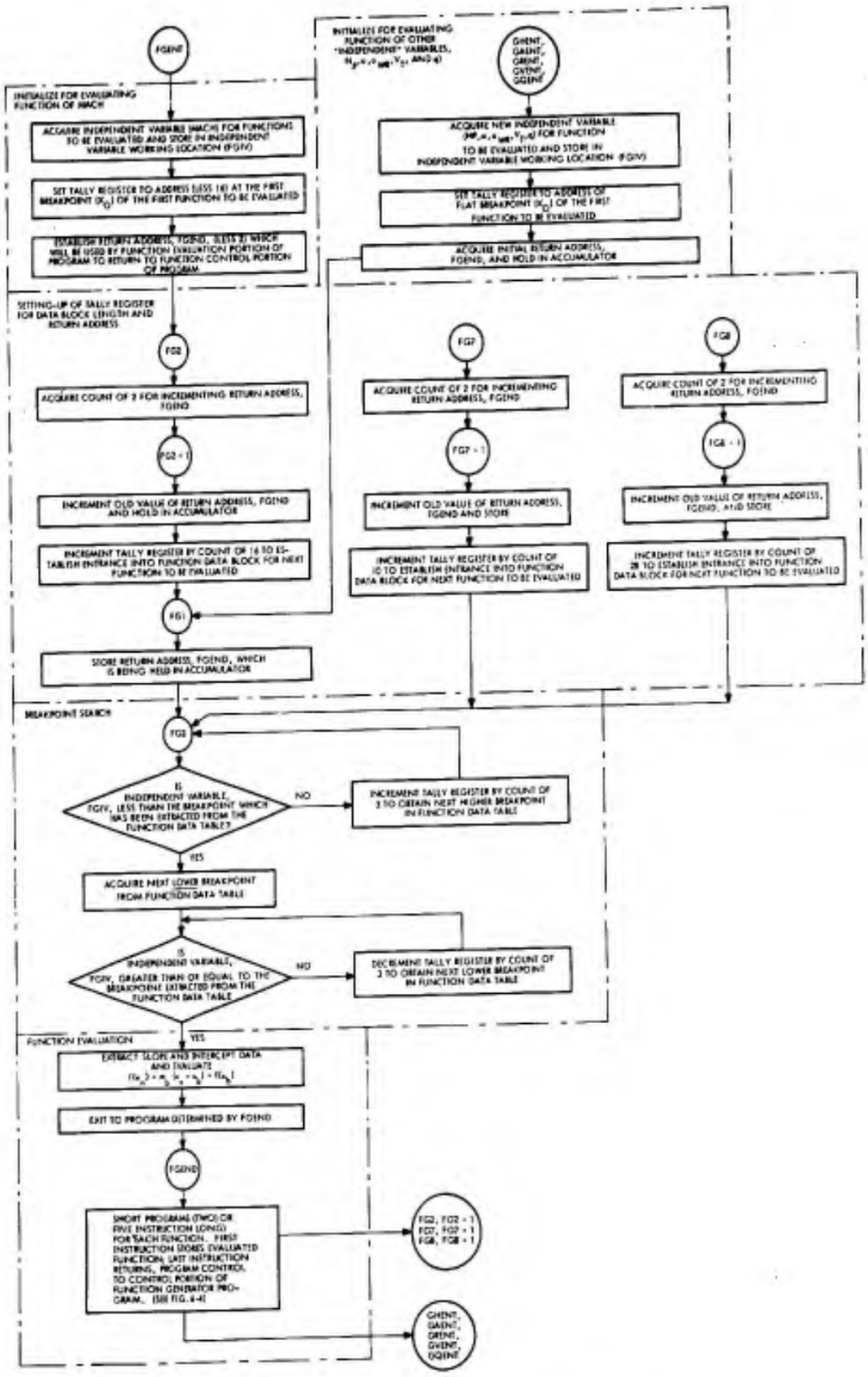


Figure 80. Function Generator Control Flow Diagram

the independent variable (FGIV) lies within the fourth segment (figure 81). When the evaluation of the first function is complete and program control is restored to FG2 or FG2 + 1 (because functions with five breakpoints are being evaluated), the Tally Register, which contains the address of the breakpoint, d_1, defining the straight-line segment for the previous function, is incremented by 16; thereby establishing the address of the fourth breakpoint, d, of the next function, $f_2(X)$, to be evaluated. In the example of figure 81, it is apparent that the independent variable, FGIV, does not lie within the fourth segment of $f_2(X)$, but lies within the first segment. As a result, the first test of the search process (the independent variable less than the breakpoint) will be satisfied. However, if it were tacitly assumed then that the next lower breakpoint is less than the independent variable, the function will be computed using the wrong segment. In these cases, testing of the lower breakpoint ensures the proper evaluation of the functions. These cases do not occur frequently, but there are a few cases where there is considerable disparity in the ordinate of the breakpoints of consecutive functions.

Function data points are grouped according to the number of breakpoints. (There is not necessarily any correlation between the ordinate of the breakpoint.) Grouping by order of breakpoint is necessary due to the substitution of zero for breakpoints which do not exist; for example, in a function having 4 breakpoints, the fifth breakpoint is made equal to zero since there is no group of four breakpoint functions. If the previous function had five breakpoints and was being evaluated in the fifth segment the fifth breakpoint, zero, would be tested against the value of the independent variable. If this zero value were less than the value of the independent variable, the Tally Register would automatically be incremented, and the next data point—an erroneous one—would be tested. Under these circumstances the program could lose control.

The main program associated with the Function Generator subroutine serves as a convenient device to space MLXO instructions. A time delay of 400/3 μsec. is necessary between any two multiplex-out commands. It was determined that the minimum time to generate any two functions would satisfy the demands of this time delay. Thus, every second time the Function Generator returns control to the main program, an MLXO operation is effected. In order to compensate for the additional instructions used in this operation, the main program causes the Accumulator to be set to the count of five and returns to the Function Generator at the desired entrance location plus one. At this point, the Accumulator will be added to the previous return address, and the result will be the proper return address for the next function.

6.3.1 Extra Function Generator

When the independent variable ranges from minus to plus, the Extra Function Generator Subroutine (figure 82) is used to compensate for the possibility of an overflow occurring within the breakpoint search routine. The overflow occurs when the independent variable is at or near the maximum negative value, and positive breakpoints are subtracted from it during the search for the proper segment. This routine functions the same as the Function Generator Subroutine with the following exceptions:

1. The input variable must be shifted one place to the right and stored in EFGIV.

2. The symbolic entrances to the routine are preceded by an E as follows: EFG1, EFG2, etc.

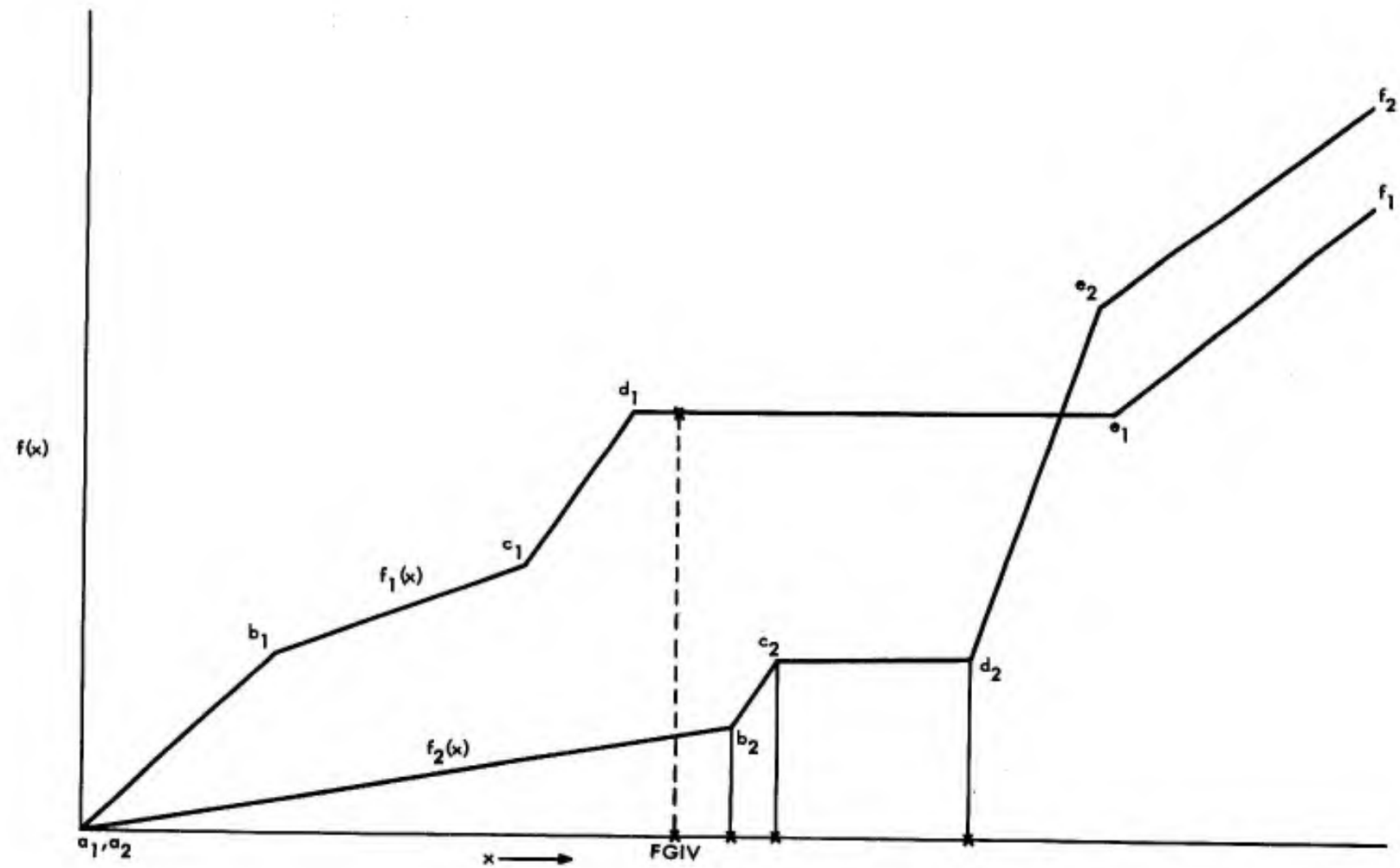


Figure 81. Two, Five-Break-Point Functions

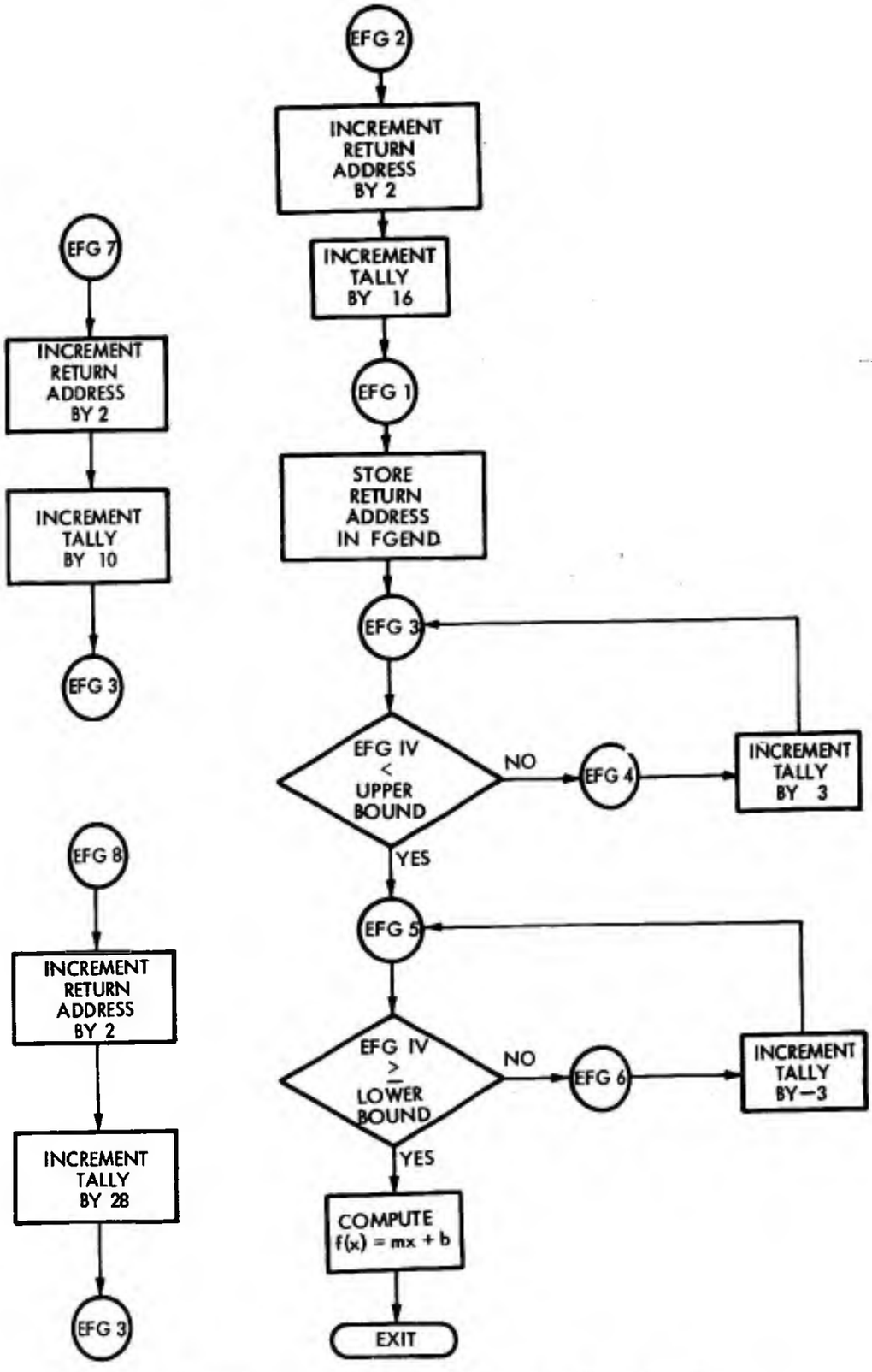


Figure 82. Extra Function Generator Flow Diagram

6.3.2 Calculation of α_{WR} and α_{HR}

The subroutines for obtaining explicit solutions of α_{WR} and α_{HR} are an adjunct to the function generator. Immediately preceding the generation of function of α_{WR} and α_{HR}, control is transferred from the Function Generator program to the α_{WR} program (figure 83).

The program selects an arbitrary interval of the independent variable and evaluates the variable using slopes and intercepts corresponding to this interval. The solution is checked to determine if the calculated result is within the interval. The program calculation continues until the independent variables and interval correspond.

The program performs the α_{WR} calculation first, since α_{HR} is a function of both α_{WR} and α_{HR}. When the program has established a legitimate value for α_{WR}, all functions of α_{WR} are calculated before the program proceeds to the α_{HR} computation. The same procedure is employed for the α_{HR} computation (figure 84).

6.3.3 Accuracy of Function Generation

Using the method of interpolation

$$y_n = m_b(x_n - x_b) + y_b \tag{68}$$

rather than the method

$$y_n = m_b(x_n) + y_{b_o} \tag{69}$$

where y_{b_o} is the Y-axis intercept of the segment b which is characterized by the slope m_b, produces a more accurate interpolation. This results because the interval interpolation is added to the value of the function at the preceding breakpoint, which acts to limit the error in the interpolation. The problems in scaling of the function are, in general, controlled by the value of the largest slope. This is especially true of Mach functions which vary considerably in the transonic region. This is a significant problem in the F-100A simulation program, so the functions of Mach were expanded in the transonic region. By expanding the functions in this manner, the values of the slopes were reduced, thereby offsetting the loss of accuracy due to the use of large scaling factors. For functions of Mach, the interpolation is in terms of Map, as follows:

$$0.0 \leq \text{Mach} < 0.9 \quad \text{Map} = \text{Mach} \tag{70}$$

$$0.9 \leq \text{Mach} < 1.1 \quad \text{Map} = 0.9 + 3\,(\text{Mach} - 0.9) \tag{71}$$

$$1.1 \leq \text{Mach} < 2.133 \quad \text{Map} = \text{Mach} + 0.4 \tag{72}$$

6.4 Convert Input Variables Subroutine

From the description of the longitudinal equations, it is evident that the present-cycle solution of the differential equations requires the coefficients to be interpreted in terms of past-cycle computations of the independent variables (Mach, α, h_p, etc.) and the present cycle longitudinal control disturbances. These control disturbances could be interpreted as a new flight control position, a gradual change in the longitudinal equation as influenced by a change in the center of gravity resulting from fuel consumption, a thrust change, deployment of speed brakes, etc. The status of the external controls are represented by numbers stored in memory.

It is apparent that, as the computation is continued indefinitely, the coefficients will be modified accordingly and the resultant computation will reproduce the short and long term response of the longitudinal equations. The computations for the present cycle must, therefore, reflect recent changes caused by external means before the computation can be allowed to continue. The subrouting which updates the flight program to the influences of the real-world is identified as Convert Input Variables. This subroutine converts analog inputs to the polarity and scaling required by the F-100 program; makes decisions relating to the utility hydraulic system, the landing gear, the speedbrakes, etc; and stores appropriate numbers in memory.

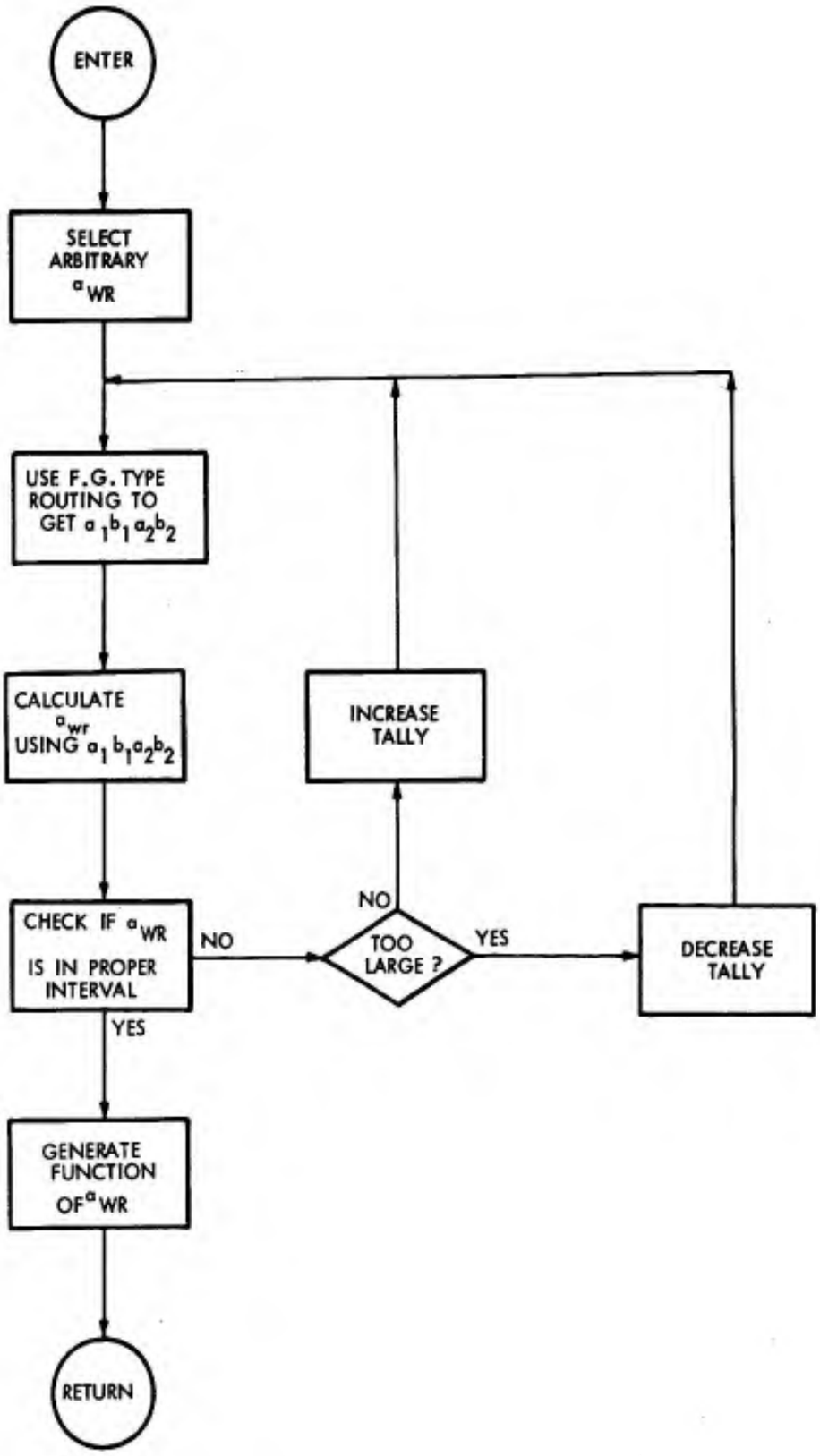


Figure 83. a_{WR} Calculation Flow Diagram

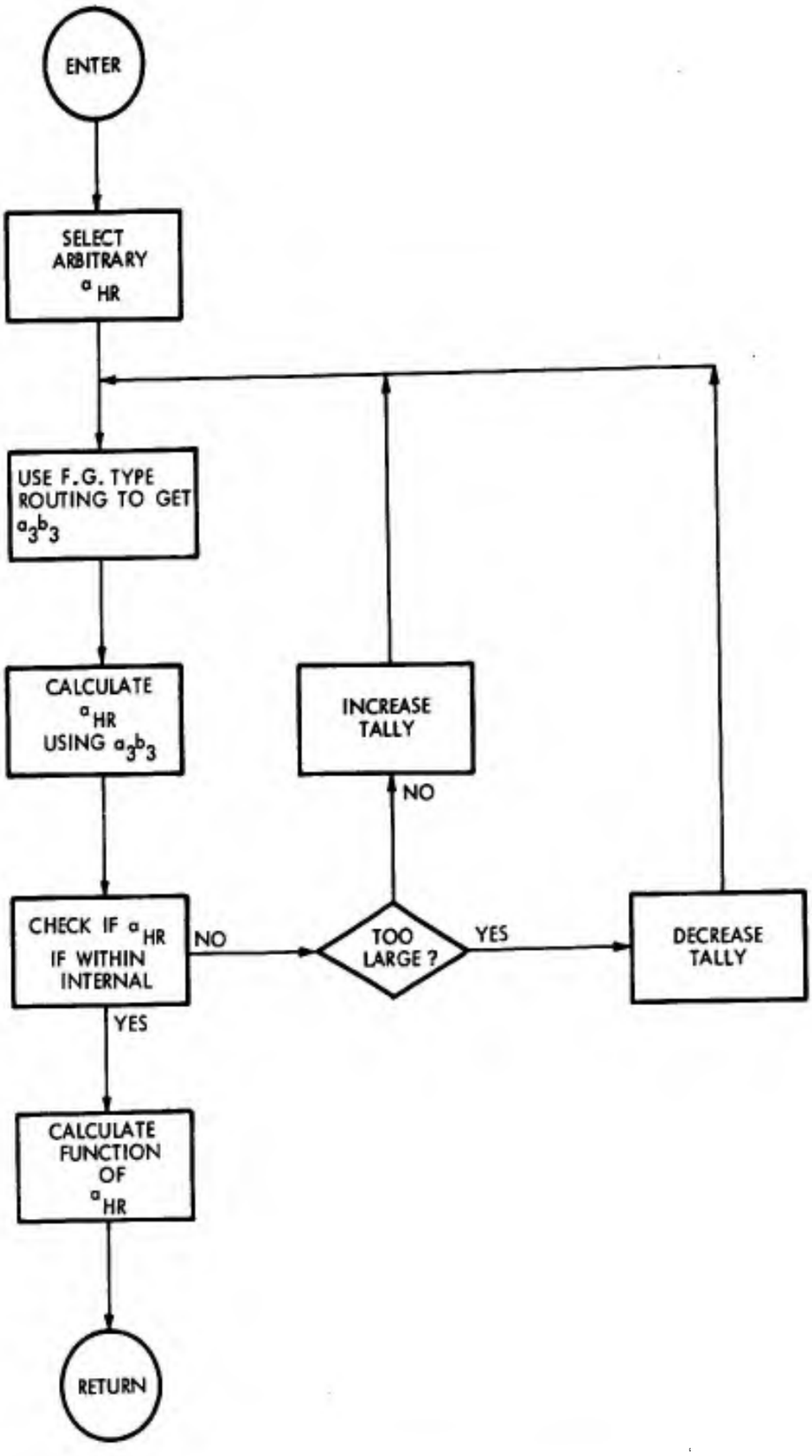


Figure 84. α_{HR} Calculation Flow Diagram

a) Analog Inputs

The encoded quantities representing the positions of the flight controls are multiplexed-in as binary numbers. A typical example of such an input is the longitudinal flight control. The multiplexed-in quantity is converted to stabilizer position and stored as some deflection. The conversion of flight control displacement to stabilizer deflection is accomplished by subtracting the binary quantity representing zero stabilizer position from the multiplexed-in value. The resultant quantity is converted to stabilizer deflection in degrees. A positive δ_H indicates forward displacement of the flight control. Similar processes are applied to the other analog inputs.

b) Discrete Inputs

The use of the discrete input is as varied as the number of times it is used. The use of a discrete input is typified by the landing gear program (figure 35). The landing gear program within the convert input variable program actually simulates the delay in the landing gear being extended or retracted. It does this by checking the status of a discrete input, LANDING GEAR IN MOTION, which is activated whenever the landing gear handle is in a position other than that shown by the indicator. The real-time delay is effected by keeping track of the number of program cycles completed before giving an indication of gear up/gear down. To minimize the added computational burden the counter is modified only every fourth, 50 millisecond cycle. With the gear in the down position the various aerodynamic coefficients related to the extended landing gear are calculated.

With the completion of the Convert Input Variables program, the flight program is updated with new input from the real-world and the equations making up the coefficient can be evaluated.

6.5 Aerodynamic Coefficients Subroutine

The function of the Aerodynamic Coefficient Subroutine is the evaluation of the coefficient equations which are given in terms of the functions computed in the Function Generator Program.

With the functional data and the real-world input defined it becomes a simple task to calculate and to store the various coefficients employed in the flight program. The coefficients are grouped together such that all coefficients due to the same real-world input are either computed or not computed as a function of the state of the input. Examples of this are drop tanks, speed brakes, and landing gear. Thus, the coefficients $C_{L_{dt}}$, $C_{D_{dt}}$, $C_{Y_{dt}}$, and $C_{M_{dt}}$ are computed when the drop tanks are on, but are ignored when the drop tanks are off. The other coefficients which are not so influenced are computed each cycle.

The program also performs a number of decision-making functions such as selecting the proper aerodynamic coefficient equations in cases where these equations vary as a function of some independent variable. An example of an aerodynamic coefficient so manipulated is the rigid pitching moment coefficient, $C_{M_{\alpha_R}}$ (equation 58, section 6.1.3).

The computation continues with the summation of the coefficients to establish the forces and the moments along and about the airplane stability axis. For the longitudinal equations, this summation is limited to X_s, Z_s, and M_{ys}.

6.6 Total Forces and Moments – Stability Axes-Subroutine

As the name implies, this program collects the stored aerodynamic coefficients and sums them in accordance with the equations for X_s, Y_s, Z_s, M_{x_s}, M_{y_s}, and M_{z_s}.

The coefficients related to the forces and moments are summed and multiplied by the proper factors to convert the coefficients both to forces in pounds and to moments in foot-pounds, respectively. The coefficients comprising the forces are multiplied by the wing area and dynamic pressure. Dynamic pressure is the result of past cycle calculations. The coefficients for the moment equations are likewise converted by employing a multiplication factor. In this case, however, the factor contains not only the dynamic pressure and wing area but also the proper moment arm. Moment arm C is the mean aerodynamic chord, and b is the wing span.

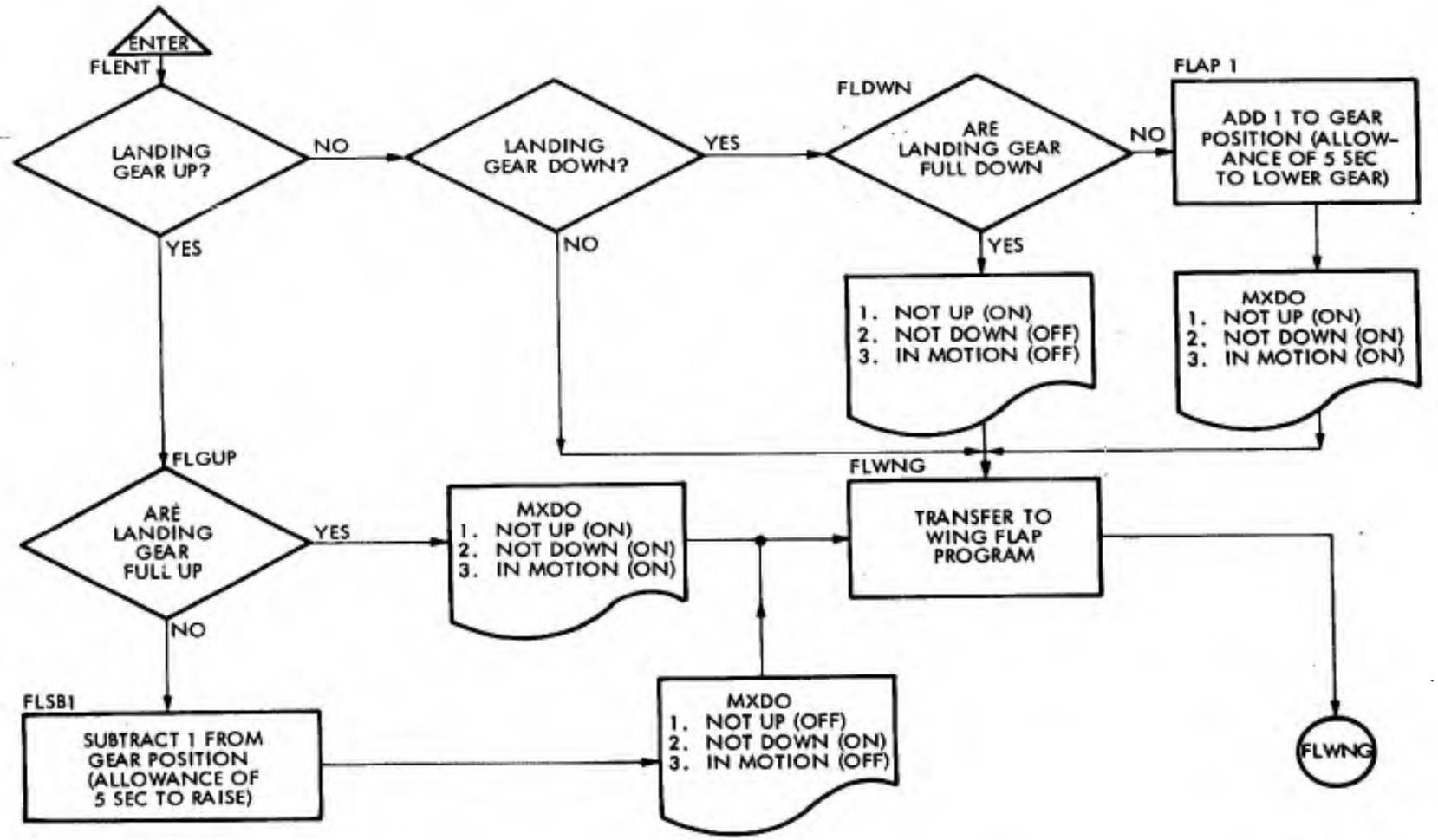


Figure 85. Landing Gear Subroutine Flow Diagram

6.7 Total Forces and Moments – Airplane Axes – Subroutine

The next step in the computation is the development of the forces and moments as a function of the airplane axes. This involves the conversion of the forces and moments as related to the stability axes into forces and moments related to the airplane axes. It also involves the addition of engine thrust which is fixed at an angle relative to some fuselage reference line.

This subroutine takes the results obtained from the summation of forces and moments of the stability axes and sums them relative to the airplane axes. The quantities are stored for the Acceleration Routine.

6.8 Accelerations Subroutine

The description of subroutines thus far describes only the forces and moments acting on the airplane. The next routine yields the resultant linear and angular accelerations along and about the airplane axes as influenced by the computed forces and moments, respectively. The linear accelerations along the airplane axes are simply derived from $F = Ma$. Longitudinal equations X_a/M_i and Z_a/M_i, where M_i is the instantaneous mass, are the simple translational accelerations. However, the component of acceleration due to gravity interpreted along the particular axis and also the sum of a number of accelerations resulting from the cross products of a linear and angular velocity modify the simple accelerations to produce the actual accelerations. The angular accelerations are summed in a similar manner with the exception that the terms are multiplied by the products of the moments of inertia.

The routine computes the six accelerations involved in the six differential equations describing $\dot{u}$, $\dot{v}$, $\dot{w}$, and $\dot{p}$, $\dot{q}_1$, $\dot{r}$. The routine also performs a number of tests to determine whether or not a discrete input is activated. The discrete inputs checked are: True Airspeed Lock, Roll Angle Lock, and Autopilot. (These functions and their use are described in section 7.1, Special Test Controls). A test is made also for land/air conditions. Under landed conditions the acceleration equations are modified extensively.

6.9 Velocity Vectors Subroutine

The generation of the accelerations along and about the airplane axes concludes the aerodynamic computations that must be performed using data that defines the behavior of the aircraft to describe the aircraft in space. The succeeding program, Velocity Vectors, computes the linear and angular velocities by integrating the accelerations.

Since integration is required also for the Direction Cosines Subroutine, it was more convenient to establish the integration process as a separate subroutine which could be entered freely. The Velocity Vectors Subroutine exists only to the extent that an entry is made into the Integration Subroutine immediately following the Accelerations Subroutine for the purpose of computing the linear and angular velocities (for the analog outputs).

6.9.1 Integration Subroutine

Numerical integration in the UDOFT real-time simulation programs is performed by applying the mixed quadrature formula, 0_{33} modGURK. This process requires that a past history of the accelerations that are being integrated and the resultant velocities be maintained for the three most recent computation cycles. This is apparent from the integration formula:

$$x_n = 2\left[Ax_{n-1} + Bx_{n-2} + Cx_{n-3} + D\dot{x}_{n-1} + E\dot{x}_{n-2} + F\dot{x}_{n-3}\right] \tag{73}$$

where the coefficients A, B, C, D, E, and F are the modified 0_{33} modGURK coefficients a, b, c, d, e, and f:

$A = 1/2a = 0.5731042$ $\quad D = 1/2\,\Delta h \cdot d = 0.0410397$

$B = 1/2b = -0.1005435$ $\quad E = 1/2\,\Delta h \cdot e = 0.0252003$

$C = 1/2c = 0.0274394$ $\quad F = 1/2\,\Delta h \cdot f = 0.0068774$

The modGURK Integration Subroutine selects the table of past values (x_i and $\dot{x}_i$) for the specified variable and performs the integration in accordance with the quadrature formula (figure 86). The integrated result is stored over the current cycle x_{n-3} value. The table of past values is never permuted; thus, for four successive computation cycles, the table of past values would be:

$$
\begin{array}{cccc}
x_{n-1} \longrightarrow & x_{n-2} \longrightarrow & x_{n-3} & x_{n-1} \\
x_{n-2} \longrightarrow & x_{n-3} & x_{n-1} \longrightarrow & x_{n-2} \\
x_{n-3} & x_{n-1} \longrightarrow & x_{n-2} \longrightarrow & x_{n-3} \\
\dot{x}_{n-1} \longrightarrow & \dot{x}_{n-2} \longrightarrow & \dot{x}_{n-3} & \dot{x}_{n-1} \\
\dot{x}_{n-2} \longrightarrow & \dot{x}_{n-3} & \dot{x}_{n-1} \longrightarrow & \dot{x}_{n-2} \\
\dot{x}_{n-3} & \dot{x}_{n-1} \longrightarrow & \dot{x}_{n-2} \longrightarrow & \dot{x}_{n-3}
\end{array}
$$

Both the determination of the oldest past values that can be discarded and the permuting of the coefficients are performed by the Permute Subroutine.

6.9.2 Permute Subroutine

The Permute Subroutine (figure 87) rotates the coefficients A, B, C, and D, E, and F so that in every cycle, coefficient A is multiplied by the n-1st value of the variable, B by the n-2nd, and C by the n-3rd, etc. A control word in the permute routine is modified each cycle so that the new values of the variable and its derivative are stored over the oldest value.

To illustrate, consider the initial storage of the table of past values and coefficients to be:

Coefficients		Past Values	
L(1)	A	L(7)	x_{n-1}
L(2)	B	L(8)	x_{n-2}
L(3)	C	L(9)	x_{n-3}
L(4)	D	L(10)	$\dot{x}_{n-1}$
L(5)	E	L(11)	$\dot{x}_{n-2}$
L(6)	F	L(12)	$\dot{x}_{n-3}$

where L(1) is defined to be storage register 1.

The modGURK program then computes:

$$L(1) \cdot L(7) + L(2) \cdot L(8) + L(3) \cdot L(9) + L(4) \cdot L(10) + L(5) \cdot L(11) + L(6) \cdot L(12)$$

The result is:

$$A(x_{n-1}) + B(x_{n-2}) + C(x_{n-3}) + D(\dot{x}_{n-1}) + E(\dot{x}_{n-2}) + F(\dot{x}_{n-3}) = x_n \qquad (74)$$

x_n is stored in L(9) and the new derivative which is computed by the Acceleration Subroutine is stored in L(12). The permute routine then rotates the coefficient by storing C[L(3)] in L(2), C[L(2)]in L(1), and C[L(1)] in L(3); C[L(4)] in L(6), C[L(6)] in L(5), and C[L(5)] in L(4). (C[L(3)] is defined to the contents of storage register 3). The two tables now appear as:

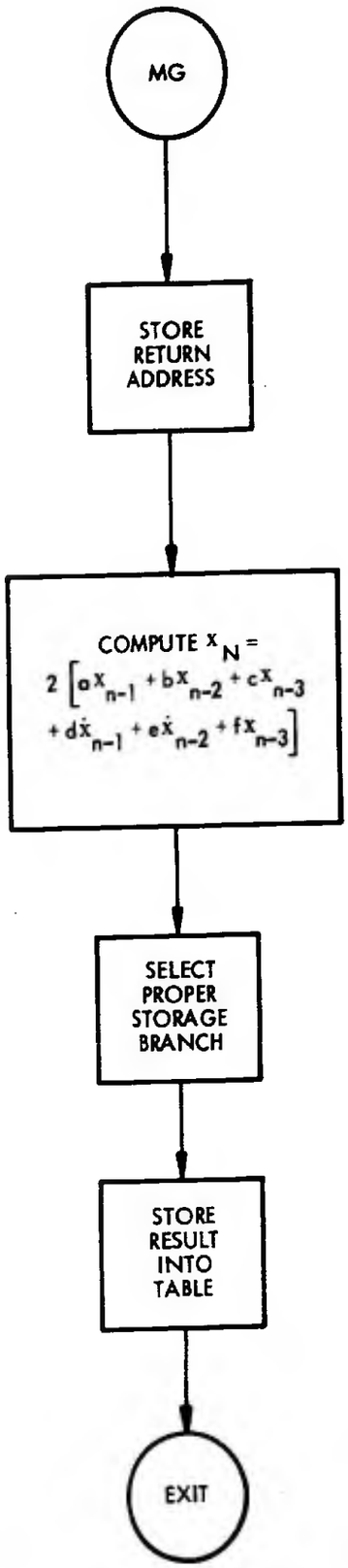


Figure 86. Modgurk Integration Formula Flow Diagram

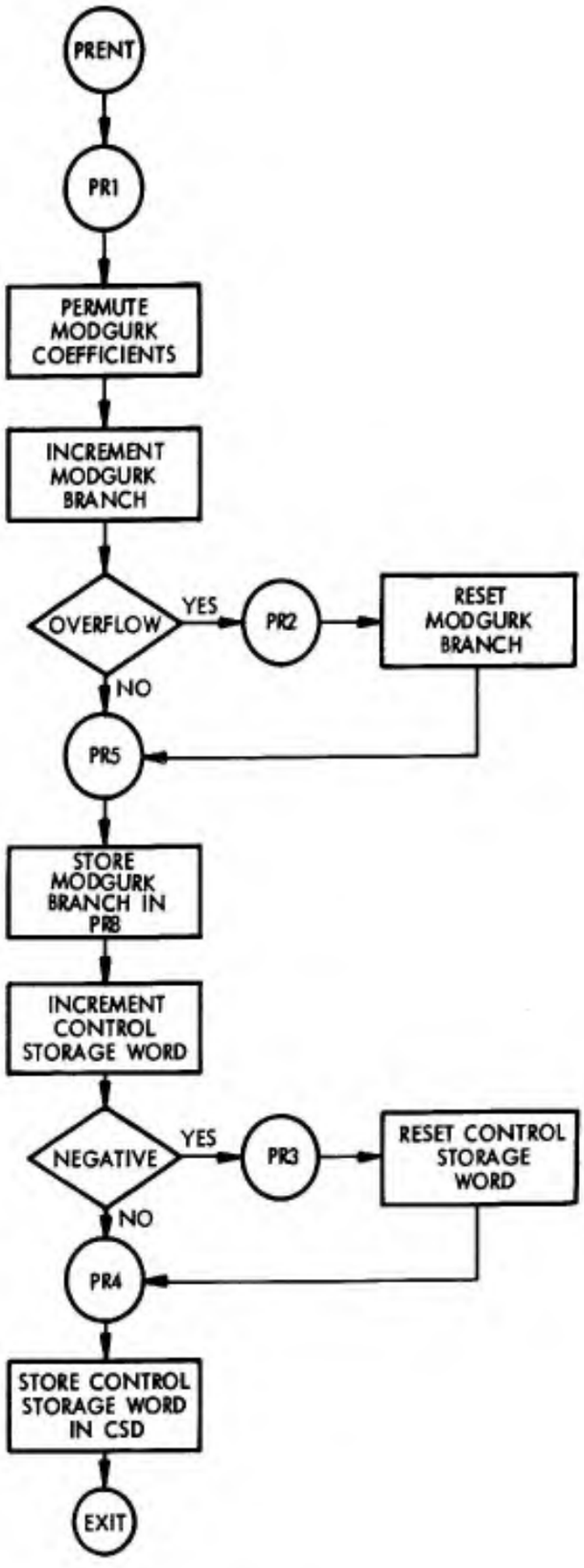


Figure 87. Permute Subroutine Flow Diagram

Coefficients		Past Values	
L(1)	B	L(7)	x_{n-2}
L(2)	C	L(8)	x_{n-3}
L(3)	A	L(9)	x_{n-1}
L(4)	E	L(10)	$\dot{x}_{n-2}$
L(5)	F	L(11)	$\dot{x}_{n-3}$
L(6)	D	L(12)	$\dot{x}_{n-1}$

Applying the Integration Subroutine, the result is

$$B(x_{n-2}) + C(x_{n-3}) + A(x_{n-1}) + E(\dot{x}_{n-2}) + F(\dot{x}_{n-3}) + D(\dot{x}_{n-1}) = x_n \tag{75}$$

The reason the coefficients are permuted each cycle rather than the past values is the fact that there exists twelve sets of variable past values which are integrated each cycle (six sets to obtain velocities and six sets in the Direction Cosines Subroutine) and each one of the twelve sets would have to be permuted each cycle.

6.10 Direction Cosines Subroutine

The next step in the computation cycle is the generation of the Euler angles. As previously indicated (section 5.4.2), direction cosines are employed to determine the angular position of the aircraft with respect to the ground. The program associated with establishing these values is called the Direction Cosines Routine.

The program (figure 88) computes six of the nine kinematic differential equations for the direction cosines and performs the integration. (The equations are the result of the projection of cross products of the rate vectors of one axes system onto another system of orthogonal coordinates.) The nine differential equations are as follows:

$$\dot{l}_1 = m_1 r = n_1 q_1 \tag{76}$$

$$\dot{l}_2 = m_1 r - n_1 q_1 \tag{77}$$

$$\dot{l}_3 = m_3 r - n_3 q_1 \tag{78}$$

$$\dot{m}_1 = n_1 p - l_1 r \tag{79}$$

$$\dot{m}_2 = n_2 p - l_2 r \tag{80}$$

$$\dot{m}_3 = n_3 p - l_3 r \tag{81}$$

$$\dot{n}_1 = l_1 q_1 - m_1 p \tag{82}$$

$$\dot{n}_2 = l_2 q_1 - m_2 p \tag{83}$$

$$\dot{n}_3 = l_3 q - m_3 p \tag{84}$$

The program integrates $\dot{l}_2$, $\dot{m}_2$, $\dot{n}_2$, and $\dot{l}_3$, $\dot{m}_3$, $\dot{n}_3$ (sets two and three using 0_{33} Mod Gurk) to obtain the direction cosines (only six equations are necessary with this method of orthogonalizing the integrated direction cosines). However, since these integrated values may not necessarily be orthogonal, orthogonality is obtained by using an iterative process in conjunction with six of the twenty-one direction cosine identities. The identities used are the following six equations: 85, 86, 90, 91, 92, and 93.

$$\text{The identity } 1 = l_3^2 + m_3^2 + n_3^2 \tag{85}$$

is used to describe the l_3, m_3, and n_3 direction cosines of set three.

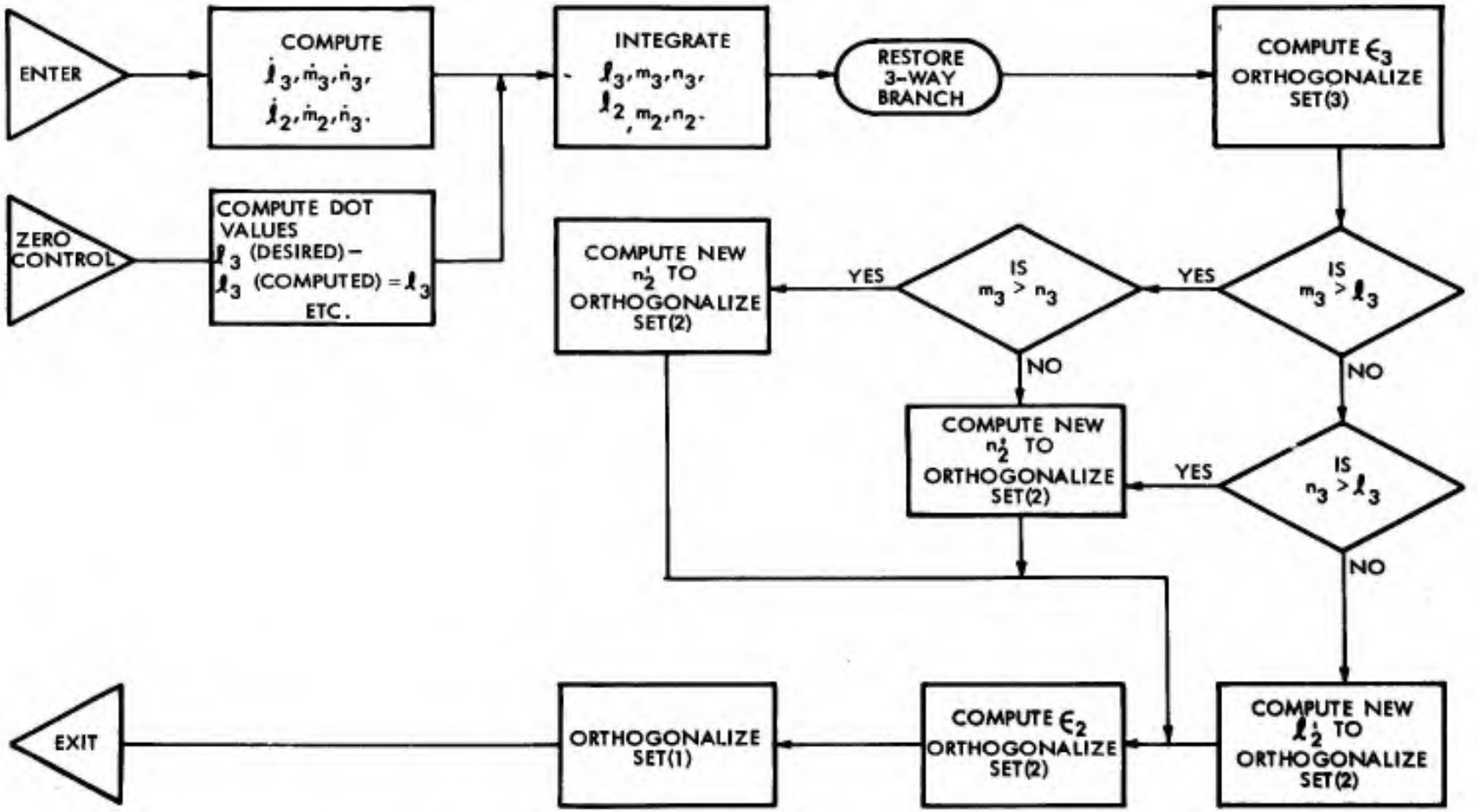


Figure 88. Direction Cosines Subroutine Flow Diagram

The identity $0 = l_2 l_3 + m_2 m_3 + n_2 n_3$ (86)

is used to obtain one component of set two, perpendicular to set three. The method assumes that a minimum of error exists in the other two direction cosines of set two. Equation 86 is solved for one of the three direction cosines using that equation which has the largest component of the set three in the denominator:

$$n'_2 = \frac{-l_2 l'_3 - m_2 m'_3}{n'_3} \tag{87}$$

for $n'_3 > l'_3$ or m'_3

$$l'_2 = \frac{-m_2 m'_3 - l_2 l'_3}{l'_3} \tag{88}$$

for $l'_3 > m'_3$ or n'_3

$$m'_2 = \frac{-n_2 n'_3 - l_2 l'_3}{m'_3} \tag{89}$$

for $m'_3 > l'_3$ or n'_3

The primes denote that the component of the direction cosine matrix has been orthogonalized with respect to one of the identities. The program then uses the same process as in set three using the following identity:

$$1 = l_2^2 + m_2^2 + n_2^2 \tag{90}$$

where the equation employs one of the direction cosines found by the use of either equation 87, 88, or 89.

The orthogonalized direction cosines of sets two and three are used in equations 91, 92, and 93 to obtain the orthogonalized direction cosines of set one.

$$l'_1 = m'_2 n'_3 - m'_3 n'_2 \tag{91}$$

$$m'_1 = n'_2 l'_3 - n'_3 l'_2 \tag{92}$$

$$n' = l'_2 m'_3 - l'_3 m'_2 \tag{93}$$

The process of orthogonalizing the integrated values is based on the following derivation. The integrated direction cosines of set three are substituted in equation 85 which yields the following:

$$l_3^2 + m_3^2 + n_3^2 = 1 + \epsilon \tag{94}$$

where ϵ indicates the lack of orthogonality.

Normalizing each term of equation 94 with respect to $(1 + \epsilon)$,

$$\frac{l_3^2}{1+\epsilon} + \frac{m_3^2}{1+\epsilon} + \frac{n_3^2}{1+\epsilon} = \frac{1+\epsilon}{1+\epsilon} = 1 \tag{95}$$

where now the orthogonalized direction cosines for set three are:

$$1'_3 = \ell_3/\sqrt{1+\epsilon} \tag{96}$$

$$m'_3 = m_3/\sqrt{1+\epsilon} \tag{97}$$

$$n'_3 = n_2/\sqrt{1+\epsilon} \tag{98}$$

Another form which is more suited to digital computer programming is utilized for UDOFT (figure 88). The derivation of this form follows. The term $1/\sqrt{1+\epsilon}$ can can be determined with sufficient accuracy by means of the first two terms of the binomial expansion:

$$1/\sqrt{1+\epsilon} \cong 1 - \frac{\epsilon}{2} \tag{99}$$

Let

$$\epsilon'_3 = 1 - \epsilon/2$$

then

$$\epsilon = 2 - 2\epsilon'_3$$

Let

$$\epsilon_3 = 2\epsilon'_3$$

then

$$\epsilon = 2 - \epsilon_3 \tag{100}$$

Substituting equation 100 into equation 94 gives the following orthogonalizing expression which is used in the program:

$$\epsilon_3 = 3 - (1_3^2 + m_3^2 + n_3^2) \tag{101}$$

The orthogonalized direction cosines in terms of ϵ_3 can be derived by substituting equations 99 and 100 into the expressions for the orthogonalized direction cosines, equations 96, 97, and 98. The resultant equations are as follows:

$$1'_3 = \frac{\epsilon_3}{2}\ 1_3 \tag{102}$$

$$m'_3 = \frac{\epsilon_3}{2}\ m_3 \tag{103}$$

$$n'_3 = \frac{\epsilon_3}{2}\ n_3 \tag{104}$$

Therefore, orthogonalizing set three becomes a simple matter of using the integrated values of 1_3, m_3, and n_3; substituting these values into equation 101 to solve for ϵ_3; and finally evaluating the orthogonalized direction cosines using equations 102, 103, and 104.

6.11 Output Processing Subroutines

By evaluating the direction cosines, the flight simulation program has completely defined the attitude of the simulated vehicle in space with respect to the earth. The remaining requirement is to exhibit the end results to the person in the cockpit. This is accomplished by means of instrument displays, flight control pressures, etc. To provide these indications the Etcetera and the Instruments Subroutines programs must be performed. Within the Etcetera Subroutine the new variables which resulted from the current cycle of computation are computed. These quantities will be used not only for the generation of the inputs to the instrument displays in the present cycle, but also will form a part of the past history for the next cycle of computation. The Instruments Subroutine takes

these values, evaluates the necessary equations defining the particular instrument, and prepares the results for analog output instrument display.

6.11.1 Etcetera Subroutine

The Etcetera Subroutine performs the computation of a number of variables using the values of the linear and angular rates computed in the current cycle. The variables computed are:

$$v_{t_n} = 1/2\left(v_{t_{(n-1)}} + \frac{u^2 + v^2 + w^2}{v_{t_{(n-1)}}}\right) \approx \left(u^2 + v^2 + w^2\right)^{1/2} \quad \text{(air)} \tag{105}$$

$$v_{t_n} = u \quad \text{(land)} \tag{106}$$

$$\sin\alpha = \frac{w}{v_{t_n}} \tag{107}$$

$$\cos\alpha = \frac{w}{v_{t_n}} \tag{108}$$

$$\tan\psi = \frac{-v}{u} \tag{109}$$

$$\text{Mach} = \frac{v_t}{a} \quad \text{(where a is the speed of sound)} \tag{110}$$

$$\dot{\alpha} = \frac{d\alpha}{dt} \frac{(\sin\alpha)_n - (\sin\alpha)_{n-1}}{T} \quad \text{(where T = 50 milliseconds)} \tag{111}$$

$$= \psi^\circ \approx 57.3 \tan\psi \tag{112}$$

$$= \alpha^\circ \approx 57.3 \sin\alpha \quad (\text{for } 0^\circ \leq |\alpha| \leq 20^\circ) \tag{113}$$

$$= \alpha^\circ \approx 58 \sin\alpha \quad (\text{for } 20^\circ \leq |\alpha| \leq 39.9^\circ) \tag{114}$$

where the following limits are applied to the variables.

$$|\alpha| \leq 39.9^\circ$$

$$|\psi^\circ| \leq 15.9^\circ$$

$$|\dot{\alpha}| \leq 1.96 \text{ rad/sec}$$

$$v_t \leq 1183 \text{ feet/sec}$$

6.11.2 Instruments Subroutine

The Instruments Subroutine solves the equations that process the data for the instrument displays. The resultant data is converted and stored for outputting to the instrument positioning servos during the execution of the Function Generator Subroutine. Explicitly, the Instruments Subroutine solves the equations that describe the accelerometer, indicated airspeed and Mach, ball angle, turn rate, rate of climb, ground speed, gyro horizon, and true heading. The descriptive equations consider the peculiarities, such as non-linearity, of the respective instruments.

The gyro horizon and the heading indicator employ the direction cosines directly as inputs. A problem is introduced, however, when sine of theta approaches ±1. When this occurs, the level of one of the signals to the gyro horizon and the heading indicator servos diminishes to insignificance. The program handles this problem by shifting the appropriate direction cosines prior to outputting as $\sin\theta$ approaches unity (figure 89 and section 5.4.2).

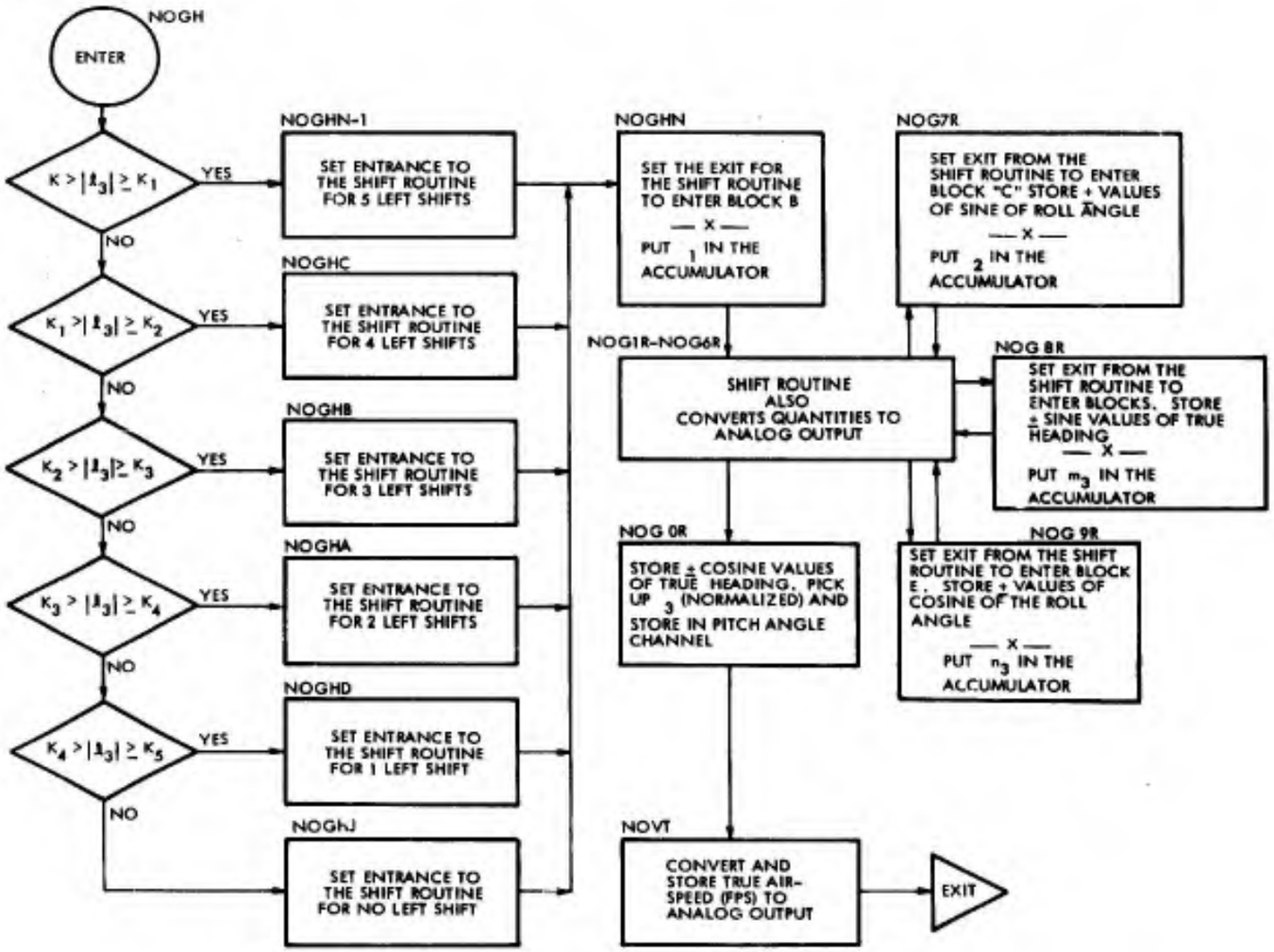


Figure 89. Gyro Horizontal Heading Flow Diagram

6.12 Decisions Subroutine

Once the program computation has been completed, the resultant solution must be checked to establish whether any of the structural limits of the aircraft have been violated, both in the air and on the ground. The following additional tests are performed: stall, stall warning, land/air, crash, and a determination as to whether or not the mode of computation has been modified.

The Decisions Subruoutine performs a series of tests and, based upon the results, controls the states of various discrete outputs or program flags.

a.) Land/Air (figure 90)

The aircraft is on the ground when

1. $H < 0$, or
2. $H = 0$ and the lift is less than the gross weight

The aircraft is in the air when

1. $H > 0$, or
2. $H = 0$ and the lift is greater than the gross weight.

b.) Crash (figure 91)

The aircraft is crashed if, when on the ground,

1. Landing gear is not down and locked.
2. Rate of descent is greater than 10 ft/sec.
3. $0° > \theta > 13°$
4. $-10° > \phi > 10°$

The aircraft is crashed if, when in the air,

1. dynamic pressure > 1663 lbs/ft^2
2. $-3 > g's > 7.33$
3. $v_1 > 600$ kts and $-3 > g's > 6$.

c.) Stall and Stall Warning (figure 92)

A stall warning will be indicated if

1. $C_L > 0.85$

A stall condition will occur if

1. $\alpha_{WR} > 14°$

This completes the description of the computation cycle for the longitudinal equations. At this point the program returns to the Governing Control program and starts a new cycle. (Actually, in the F-100A flight simulation program, the complete program considers flight systems, engine, and altitude before returning to the initial program, Governing Control.)

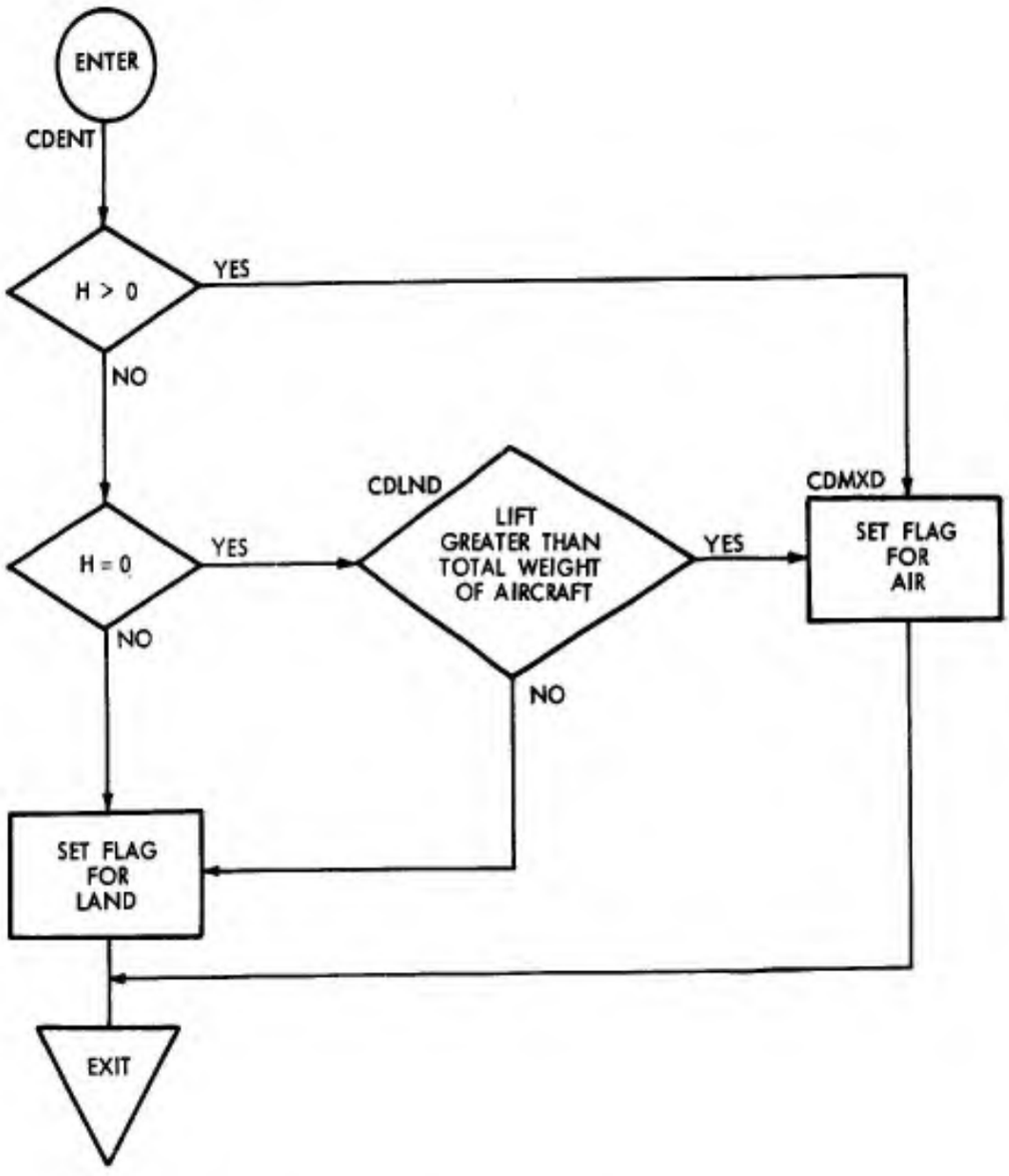


Figure 90. Land/Air Decisions Flow Diagram

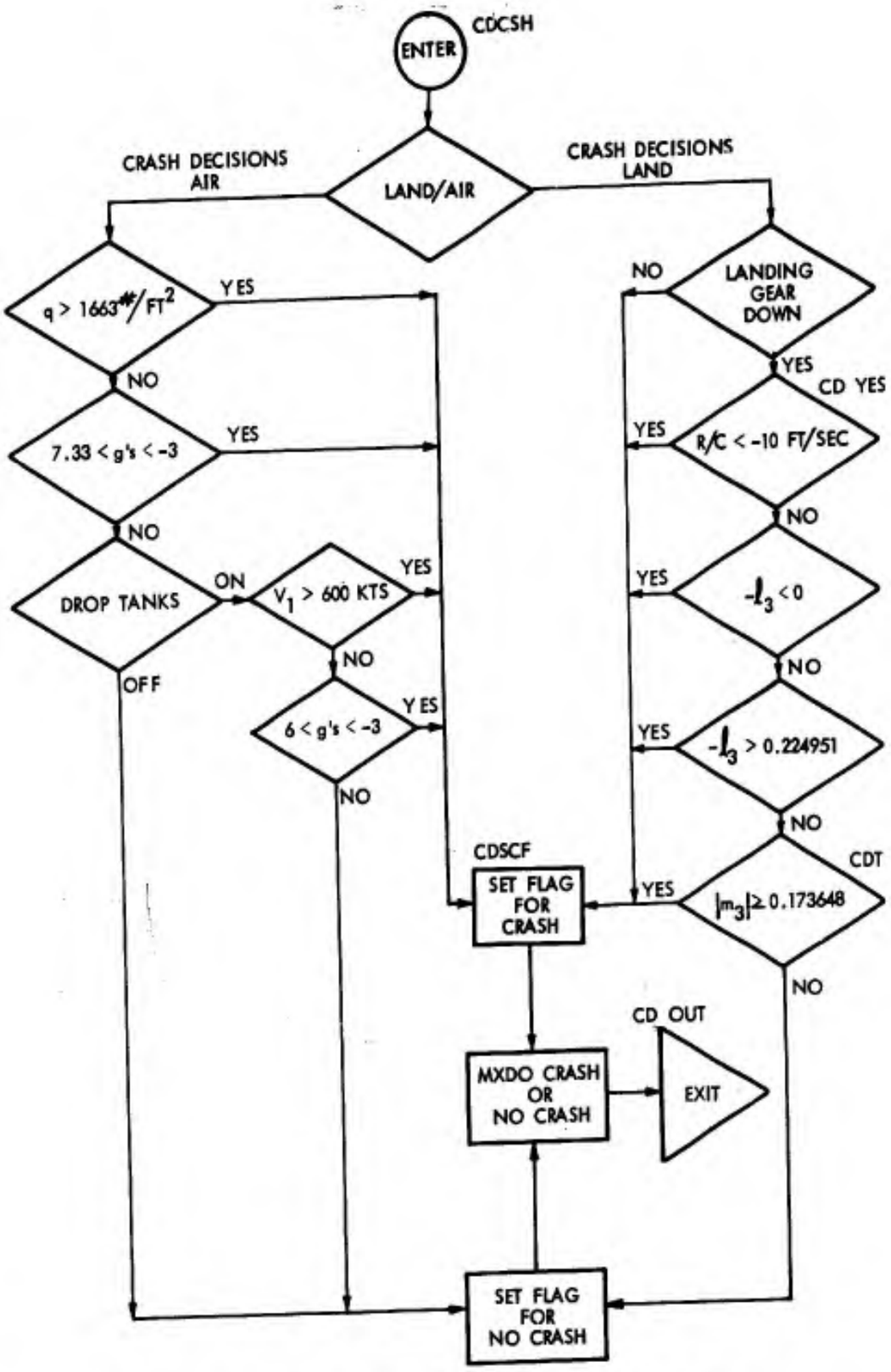


Figure 91. Land/Air Crash Decisions Flow Diagram

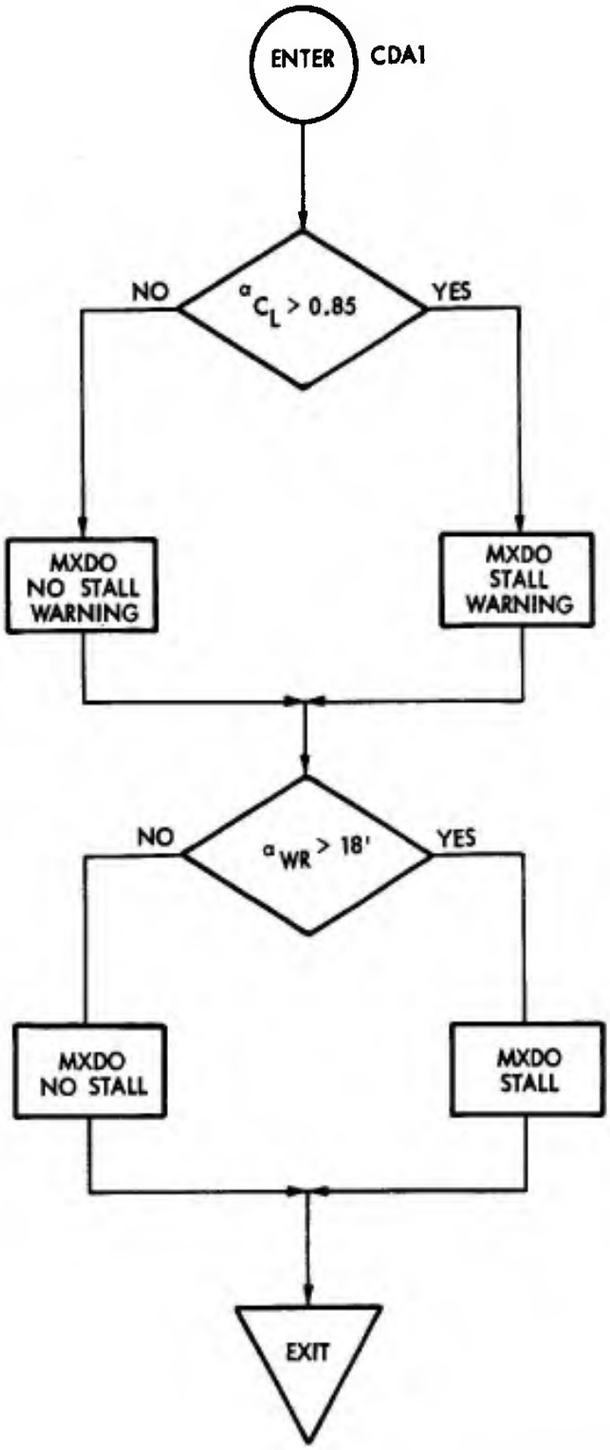


Figure 92. Stall and Stall Warning Decisions Flow Diagram

SECTION VII

TECHNIQUES FOR ESTABLISHING THE PERFORMANCE OF THE SIMULATED F–100A

Many problems, the more significant of which have been discussed in preceding sections, were encountered during the development of both the UDOFT computer and the aircraft simulation programs employed to determine the feasibility of digital flight simulation. Once the computer and the programs had been dealt with, the problems did not cease, for the success of the UDOFT project depended upon the ability of the integrated system to perform as a flight simulator. The final proof of the feasibility of this new approach to simulation therefore could not be established until the performance and the flying qualities of the simulated aircraft had been exhaustively tested.

Techniques used in the testing of analog flight simulators were not directly applicable to the digital counterpart; there were no servo shafts that could be locked or manually positioned. Special controls and computer programs had therefore to be established. The following discussion illustrates the specialized testing facilities that were incorporated into the UDOFT and the F-100A simulation program.

7.1 Special Test Controls

In order to force the simulation program into certain prescribed operating flight conditions, or to lock the program at certain current conditions, a number of specialized control functions were introduced into the aircraft simulation program. These control functions were designed primarily to facilitate testing and not essentially as aids to training. The following were established prior to, and were used during, the acceptance testing of the F-100A model:

Control Function	Discrete Input
Zero	02LWT
Freeze	30LWT
Altitude Increase	42LWT
Altitude Decrease	43LWT
Autopilot	44LWT
Altitude Lock	47LWT
Roll Angle Lock	50LWT
No Fuel Depletion	73LWT
True Airspeed Lock	76LWT
Center of Gravity Lock	77LWT

The following sections contain brief descriptions of these controls. Where inadequacies have been noticed during the time elapsed since acceptance testing, modifications have been made to the control functions and are noted.

7.1.1 Zero; 02LWT

Once a crash condition has occurred, a method of initializing the variables and the instruments to their ground values must be available. Also, during a simulated flight, the Zero Control will land the aircraft and force all variables to their ground values.

Initializing the variables is accomplished by forcing the equations of motion and position to their original on-the-ground values. Zeroing these equations is performed by introducing negative rates into the integration tables of the past values of the derivative terms for the calculations of the linear and the angular rates and the direction cosines. Typically, the angular rates are zeroed as a result of making

$$\dot{p}_n = -(p_{n-1}) \tag{115}$$

$$\dot{q}_{1_n} = -\left(q_{1_{n-1}}\right) \tag{116}$$

$$\dot{r}_n = -\left(r_{n-1}\right) \tag{117}$$

This substitution process occurs in the course of each successive integration cycle, maintained as long as the Zero Control is actuated. The result of the integrations performed with these parameters returns the equations of motion exponentially to the ZERO state at a rate determined by the time constant of the intergration formula. Zeroing the equations in this manner does not necessarily force all parameters identically to zero; this is due to both the fixed scaling of the quantities used in the integration formula and the fact that the integration formula can regress to ground level conditions for values of certain variables other than zero.

Since this can occur, an additional routine was inserted in the Zero program to make all pertinent parameters identically zero. The procedure involves examining Mach and altitude to determine if both are within prescribed minimum values; when they are, zeros are inserted into the registers storing the pertinent parameters.

Exceptions to the zeroing process are the altitude and power plant computations. Altitude is slewed to zero at a rate determined by the dynamic characteristics of the Analog Altitude Indicator System. For the F-100A altimeter, the maximum slew rate is 25 feet per iteration cycle (500 feet per second). The power plant equations are in no way affected when the simulation program is under the influence of Zero Control.

In addition to its normal use of returning the simulated aircraft to the ground condition, the Zero Control may be used to relinquish a crash condition without returning the aircraft to the ground. This is possible because the time required to achieve the zero state varies for the different linear and angular rates as a function of the maximum scaling of these quantities. Thus it is possible, if flight speed is safely above stall, to clear a crash condition by momentarily actuating the Zero Control.

7.1.2 Freeze; 30 LWT

The Freeze Control in the digital simulation program serves the same purpose as the Problem Freeze Control in the analog flight simulators. It serves to suspend the progress of the simulated flight and to maintain the currently computed conditions. Entry into a Freeze condition may be automatic or manual. Automatic Freeze occurs when a crash condition exists; Manual Freeze is exercised by the instructor.

In the case of testing the simulation program on the UDOFT computer, Manual Freeze is used extensively. Under this condition, the computation can be halted, allowing stored test data to be retrieved automatically from the computer memory by print-out, or manually by means of the computer console controls and indicators. Removal of the Freeze condition will cause the resumption of computation, commencing at the point where the Freeze condition had been injected.

The simplest method for freezing the computations would be to halt the computer, but this cannot be done in UDOFT without seriously affecting the cockpit instrument readings. This peculiar circumstance arises from the tendency of the analog output voltages to drift rapidly with time if not updated periodically. Therefore, it seems, only those instructions which serve to maintain the analog outputs would have to be executed. Initially, however, this approach to implementing the Freeze mode in the F-100A simulation program was not taken. The technique used was to by-pass the routines for Aerodynamic Coefficients, Total Moments and Forces (stability and airplane axes), Direction Cosines, Mach Number, Dynamic Pressure, Altitude, Mass of Fuel, and Center of Gravity, and to set the six angular and translational accelerations equal to zero. The velocity vectors, computed as usual, remained constant since the second derivatives had been set equal to zero. Later, this approach was deemed needlessly complex; it was replaced by a very simple modification to the Governing Control routine which causes only the Function Generator routine, which contains the analog output multiplexing instructions, to be executed.

One precaution must be observed, however, when preparing to extract test data from the computer. In order to maintain system status quo while the program is halted, the instrument positioning servos must be deactivated before halting the program.

7.1.3 Altitude Increase/Decrease; 42LWT/43LWT

The Altitude Increase and Decrease Controls permit manual modification of aircraft altitude while the simulation program is being executed. When either discrete input

is actuated, a maximum slew rate of 25 feet per cycle (500 feet per second) is introduced into the altitude program. The altitude will continue to change so long as the control is ON.

Aside from the use made of this control by an instructor, it serves as an aid to testing. Since most of the acceptance tests were performed from the cockpit, the altitude change controls provided a rapid means for altering test altitudes without the need of simulating flight to the new test altitude.

This facility has been used infrequently of late, due to the availability of a superior technique for attaining desired altitudes (see 7.1.5, Altitude Lock).

7.1.4 Autopilot; 44LWT

The term autopilot is a misnomer; its application to the F-100A simulation program belies its name. The F-100A autopilot is merely an aid to establishing flight equilibrium conditions in the longitudinal equations; it is not capable of automatically forcing the longitudinal equations to equilibrium.

The problem encountered in attaining longitudinal equilibrium revolves about the longitudinal stability of the aircraft. Any attempt to change readily from one altitude to another, or from one airspeed to another, is balked by the short-term oscillations that result from disrupting longitudinal equilibrium. The autopilot function overcomes this objectionable response by increasing longitudinal damping, due to pitch rate $\left(C_{m_{q_1}} \times q_1\right)$; to such an extent that the aircraft becomes extremely stable in terms of longitude.

The damping is increased by artifically stepping up the value of pitch rate, q_1, as a function of the difference between the instantaneous rate of climb and the desired rate of climb. Thus the succeeding computation of pitch moment, M_y, exhibits increased damping due to the large q_1 term. This aids in damping the short-term oscillation, but, because the artificial damping diminishes as the desired rate of climb is approached, piloting skill is still required to damp out the long-term oscillation (phugoid).

The autopilot function was used extensively to minimize the set time for achieving a steady-state straight and level flight or a specific climb condition. For straight and level flight, the desired rate of climb is set equal to zero; for specific climb conditions, the desired rate of climb is set according to the following table.

Rate of Climb Feet per Minute	Octal Representation Feet per Second, Scaled B12
-1000	-0020524
- 500	-0010252
0	0000000
+ 500	0010252
+1000	0020524
+2000	0041252
+5000	0123252
+7000	0164524
+10,000	0246524
+15,000	0371776

Once the octal representation of the desired rate of climb has been entered into the computer and the autopilot control function has been enabled, equilibrium is obtained by flying the simulator in the conventional manner and allowing the longitudinal equations to stabilize. One precaution must be observed, because if the difference between the actual and the desired rates of climb is too great when the autopilot input is actuated, the artificial damping term may assume the proportions of a forcing function, thereby causing excessive "G" loading which may result in a crash condition.

Since a true autopilot function is invaluable to the rapid creating of requisite conditions for nearly all flight testing, and because the original F-100A autopilot was lacking in so many respects, a more satisfactory automatic means for establishing steady-state straight and level flight equilibrium has been developed. (See Section 7.5.1, SSLFE Program.)

7.1.5 Altitude Lock; 47LWT

The purpose of the Altitude Lock Control is to maintain altitude constant at the value which existed when the control was actuated. Since altitude is obtained from simple expression

$$hp_n = hp_{n-1} + \Delta t(R/C_n) \qquad (118)$$

it is necessary only for the altitude routine to omit the incremental change in altitude due to the average rate of climb.

When this lock is used in conjunction with the Altitude Increase/Decrease Control, the altitude can be slewed to any desired quantity and locked. The term $\Delta t(R/C_n)$ is replaced by ±25 feet. Since the Altitude Lock function precedes the Altitude Change function in the program, the Lock must be disabled in order to change aircraft altitude.

Achieving a particular altitude accurately by following this procedure requires considerable manual dexterity. Also, since altitude is incremented by 25 feet per computation cycle, the actual altitude may be in error by as much as ±25 feet even if the Altitude Increase/Decrease switch is deactivated immediately upon reaching the desired altitude. In light of the improvements being made to the other special test control functions, it was decided to automate the achievement of a particular altitude. It then becomes necessary only to read into the computer the desired altitude, and to enable the Altitude Lock Control. The altitude will be incremented (or decremented) at the rate of 25 feet per cycle (500 feet per second) until the actual altitude is within 25 feet of the desired altitude, at which time the required adjustment will be made to bring the aircraft to the altitude desired. If the Altitude Lock Control is used strictly for locking altitude at its current value, it is necessary only to read 25 feet into the register to which desired altitude is assigned. When the Altitude Lock program examines desired altitude and finds it at 25 feet, it will lock altitude at the current value.

7.1.6 Roll Angle Lock; 50LWT

The Roll Angle Lock function forces the aircraft to assume a constant zero-degree roll angle. This Lock is used frequently to zero-out the lateral equations, thereby reducing the number of degrees of freedom and permitting the longitudinal equations to be stabilized more readily for a straight and level flight condition.

When the Roll Angle Lock Control is actuated, the direction cosine $-m_3$, i.e. $-\cos\theta \sin\phi$, is substituted into the integration table of past values for $\dot{p}$ and p for each succeeding cycle of computation. The result is that the roll angle, ϕ, is forced to zero degrees.

Because the Roll Angle Lock function has no effect on turn rate, rudder control from the cockpit is often required to zero-out yaw angle in the process of attaining the straight and level flight condition. As a result of this inadequacy, the Roll Angle Lock function has been augmented with a Yaw Angle Lock function which is effected by introducing the negative of the turn rates, r, into the integration table of past values of the turning acceleration term, $\dot{r}$, for each succeeding cycle of computation.

7.1.7 No Fuel Depletion; 73LWT

The purpose of the No Fuel Depletion function is to maintain the constant mass of the aircraft, and consequently its moments of inertia. Since the only significant quantities that make up the mass of the aircraft are the mass of the empty aircraft and the mass of fuel, obviously preventing any change in the mass of fuel will keep the gross mass of the aircraft constant. When the No Fuel Depletion function is activated, the fuel flow (W_f) calculations are performed as usual; however, the effect of fuel flow on fuel quantity (Q_f) is ignored.

By means of the instructor inputs, Main Tank Refuel, 35LWT, and Main Tank Dump, 36LWT, fuel may be added to or depleted from the aircraft, thereby providing a manual means for establishing aircraft gross weight. The following data is included to illustrate the extent to which fuel quantity may be used to adjust the weight of the aircraft.

	Mass (slugs)	Weight (pounds)
Aircraft, Empty	591	19, 030
Maximum Fuel, Main Tanks	153	4, 927
Drop Tanks, Empty	12.4	393
Maximum Fuel, Drop Tanks	111	3, 574

Once the appropriate weight has been established, actuation of the No Fuel Depletion Control will freeze the aircraft weight or mass at that value. After this technique had been used for some time, it was decided that a simpler approach would be to read directly into the computer the weight of fuel required to increase the aircraft weight to that required for the particular test.

7.1.8 True Airspeed Lock; 76LWT

The purpose of the True Airspeed Lock function is simply to prevent airspeed from changing; it is used whenever airspeed must be fixed at some constant value. The implementation of the function in the program resembles that of many other locking functions; zeros are stored in the integration table of the past values of $\dot{u}$ for each succeeding cycle of computation. An unfortunate consequence of this approach is that the acceleration terms are destroyed; thus it is possible that even though the airspeed is locked the aircraft may, in fact, be accelerating or decelerating (i.e. thrust $\neq$ drag). This phenomemon would cause misleading results when conducting static longitudinal stability tests. The implementing of this function has been revised so that the $\dot{u}$ terms are not destroyed. The revised function is described more completely in Section 7.5.1, SSLFE Program.

7.1.9 Center of Gravity Lock; 77LWT

The purpose of the Center of Gravity Lock function is to fix rigidly the distance between the center of gravity (c.g.) and the center of lift (MAC). When the control is actuated, the center of gravity is set at the position determined by the distribution of fuel-mass in the main fuel tanks and the drop tanks.

For each of the acceptance tests the center of gravity is specified; however, it is difficult to establish the required center of gravity by means of the related lock function. Manual control of the position of this center requires shifting fuel within and between the main fuel tanks and the drop tanks until the proper center of gravity is established. This is further complicated by the fact that the main fuel tank has an irregular shape, so that the shift in its center of gravity is not linearly related to the quantity of fuel within the tank.

To minimize set-up time, the required value of the position of the center of gravity should be inserted directly into the computer program (Center of Gravity Lock function actuated). The following tabulation indicates the quantities that have been used in the F-100A program for fixing this position.

Center of Gravity	Octal Representation in feet Scaled B0
29% MAC	5340000
30% MAC	4417300
32% MAC	2560100
34% MAC	0720300
35% MAC	0000000

Fuel quantity can be set independently of the above in order to achieve the proper aircraft gross weight.

7.2 Accumulation and Extraction of Test Data

Since performance testing of an aircraft flight simulator involves performing static tests, (the test results being indicated by parameter-values when the system has stabilized) the accumulation of test data in a digital flight simulator system is a trivial problem. The values of nearly all computed parameters are stored in memory immediately after their calculation. Therefore, if a significant parameter has been computed and stored, there is no data accumulation problem; the data is merely extracted and presented to an observer in readily understandable form. Extracting the data is performed by a very simple computer program; the form of output is hard copy from the output printer.

On the other hand, testing the flying qualities of the simulated aircraft requires data to be accumulated as a function of time, in many cases over a considerable period of time. Since the computer is constantly updating all variable parameters at a rate of twenty times a second, capacious data storage would be required to accept all data computed during the interval between initiation and completion of the particular dynamic test. The UDOFT computer is not endowed with unlimited data storage; therefore, another means for storing time-variant data is needed. This is readily taken care of by using the unused analog output channels. By inserting only two instructions into the program, the recorded behavior of a parameter can be extracted by means of an analog output. Ultimate storage of the data is accomplished by a strip recorder accepting the analog output.

7.2.1 Output Printer

This means of data extraction generates a permanent record of computed quantities in numerical form. The chief advantage of this method is that the data is presented in its most accurate form, i.e. that form in which it was used within the simulation program. The method has the disadvantage that the data cannot be extracted in real-time; therefore, due to the limited computer storage available for test results, only a limited amount of data can be extracted.

A simple printout program extracts data from the computer memories. In reality the program does not effect printout, but merely calls-out data from the number memory so that it can be acquired by the output printer control mechanism. The program consists of a sequence of Clear and Add (CLA) instructions which address the number registers storing the desired quantities.

An unlimited variety of such programs can be established; however, a most useful program calls out the following parameters:

Quantity	Symbol	Units	Scaling
Instantaneous Mass	MI	slugs	B17
True Airspeed	VTK	knots	B12
Thrust	T	pounds	B18
Percent Thrust	FN	percent	B9
Pressure Altitude	HP	feet	B18
Fuel Flow	WF	lbs./hr.	B15
Fuel Flow Afterburner	WFP	lbs./hr.	B18
Engine Speed	RPM	percent	B9
Tailpipe Temperature	TPT	degrees C	B12
Dynamic Pressure	Q	$lbs./ft^2$	B12
Mach Number	MACH	-	B3
Force Along X-Stability Axis	XS	pounds	B21
Rate of Climb	RC	feet/min	B12
Distance to Center of Gravity	D	feet	B0
Angle of Attack	ALP	degrees	B6
Stabilizer Angle of Attack	AHR	degrees	B6
Sin θ	L3	-	B1

The data extraction program consists of two parts. The first part organizes the data into a table, performing appropriate shifts to modify the scaling of the parameters. The change to scaled factors, which are multiples of three, facilitates the manual conversion of the quantities from octal notation to decimal form when using the IBM octal-to-decimal conversion tables. The second part consists of a sequence of CLA instructions which call the parameter values from the number memory, making them available for print-out.

The print-out procedure is as follows:

1. Halt the computer.
2. Set the Sequence Counter to 7532 (first instruction of call-out program).
3. Set Mode Control switches for Continuous Print Mode.
4. Depress Start Switch.

The output printer will print out (using the long form) the data as typified by the following:

(IMAD)	(OTNMAD)	(NMAD)	(SANUMBER)	(ACCUMULATOR)
7532	346335	6335	1273600	1273600
7533	346336	6336	0776006	0776006
7534	346337	6337	0154570	0154570
7535	346340	6349	1440000	1440000
7536	346341	6341	0005632	0005632
7537	346342	6342	1720660	1720660
7540	346343	6343	0000000	0000000
7541	346344	6344	1445242	1445242
7542	346345	6345	1012426	1012426
7543	346346	6346	1543610	1543610
7544	346347	6347	0612664	0612664
7545	343263	2363	-0011000	-0011000
7546	344714	4714	-0001004	-0001004
7547	340043	0043	-2234572	-2234572
7550	341640	1640	0173600	0173600
7551	341661	1661	-0067000	-0067000
7552	340062	0062	-0062240	-0062240

The data presented in the example was taken directly from a print-out made during F-100A performance testing to establish the curves of thrust available and thrust required as a function of true airspeed, for various power settings (Military) and altitudes above sea level. Following the extraction of data in this manner, it is necessary to convert the data to decimal form for the purpose of data plotting. Even though consideration had been given to facilitating the manual conversion process by "octalizing" the scale factors of the parameters, it would have been a tedious task to convert so much octal data. For this reason an octal-to-decimal conversion program was prepared for an in-house general purpose digital computer (Burroughs 220). The result of the conversion was a very neat listing of the UDOFT octal quantity, the UDOFT scale factor, the decimal equivalent, and the symbolic representation of the parameter.

The following demonstrates such a conversion using the same data as that presented in the preceding example:

Octal	Scale Factor	Decimal Equivalent	Parameter
1273600	17	22392.00000000	MI
776006	12	510.01171875	VTK
154570	18	6959.00000000	T
1440000	9	100.00000000	FN
5632	18	371.25000000	HP
1720660	15	7814.75000000	WF
	18	0.00000000	WFP
1445242	9	100.66455078	RPM
1012426	12	522.54296875	TPT
1543610	12	867.76562500	Q
612664	3	0.77119445	MA
-11000	21	-4608.00000000	XS
-1004	12	-1.00781250	RC
-2234572		-0.28826618	D
126300	6	1.47460937	ALP
-67000	6	-0.85937500	AHR
62240	1	0.02456664	L3

7.2.2 Analog Outputs

The analog output method of data extraction provides a permanent record of time-varying quantities in graphical form. This method is used primarily for time-history recordings, and indicates excellently the behavior of various parameters with respect to each other as a function of real-time. It has the advantage that data can be extracted while the simulation program is being performed. The computer need not be halted; needless to say, were the computer halted, no time-varying quantities would be available. The disadvantage, however, is that the recorded data is presented with decreased accuracy, as compared with the accuracy of the data obtained via the output printer. This is especially true for those cases where the parameters of interest have a large dynamic range and it is important to distinguish small changes.

To activate a particular analog output channel, a simple three instruction program is needed. The first instruction is a CLA instruction, to extract the current value of the parameter from number memory and introduce it into the Accumulator. The second instruction gates it into the Transfer Register. The third instruction, an MLXO instruction, moves the quantity in the Transfer Register to the analog output channel specified in the number memory address field of the MLXO instruction. If scale changes are desired, the appropriate shift instruction is inserted between the CLA and the following instruction. Due to the limitations of the analog output system, another MLXO instruction must not follow within 133 microseconds and the analog output channel must be updated at least once during every 50-millisecond interval.

7.3 Supplementary Test Programs

Special programs were generated to facilitate setting-up various flight tests and for correlating data. They were not significant in themselves; however, considering the variety and the quantity of programs that had been prepared, there is very little that cannot be done with the digital simulator to expedite the testing phase. Typical programs are:

a.) A program to actuate console discrete output indicators when certain pre-established flight conditions have been satisfied

b.) A program for converting data in UDOFT octal notation to BCD (binary coded decimal)

c.) A program to examine the dynamic behavior of selected parameters and to store their minimum and maximum values for printout after test; used as backup to the time-history recordings in an attempt to improve the accuracy to which the recordings may be read

The pressures of UDOFT acceptance testing did not allow further pursuit of this promising area of automatic aids to testing. Many supplementary programs were prepared after the completion of acceptance testing. Experimentation with the F-100A program, after formal completion of the project, has provided the impetus for consolidating the many minor test programs into complete testing packages applicable to the performance of a broad spectrum of acceptance tests. The steady-state straight and level flight equilibrium program (SSLFE) is a result of such consolidation.

7.4 Procedure for Performance Testing

In order to demonstrate the use of the discrete inputs in establishing test conditions, a typical test procedure will be reviewed. The test selected for this example covers Thrust Available and Required Vs. True Airspeed.

This test, one of the basic performance tests, provides a graphic indication of the thrust available at specific power settings and also the minimum thrust required to maintain a steady, straight and level flight for various true airspeeds. The test is conducted for several altitudes and configurations.

There are a number of reasons why this test is performed. For performance, it indicates the thrust available for climb and acceleration purposes, and the minimum thrust required to sustain level flight. To the simulator developer it indicates the degree of success with which the basic drag and lift of the aircraft have been simulated and, of equal importance, the accuracy with which engine performance for specific power settings has been simulated. Good results from this test aid in diagnosing problems that may arise in achieving acceptable climb and time-to-accelerate performance.

The nature of this test allows the accumulation of additional test data pertinent to other longitudinal tests. Tests such as Static Longitudinal Stability and Speed-Power may be performed concurrently with the Thrust Required section of the test. In general, a number of engine parameters such as fuel flow, tailpipe temperature (exhaust gas temperature), and pressure ratio can be obtained concurrently from the Thrust Available section.

7.4.1 Test Requirements

The following relates the conditions for which this test was performed on the F-100A model.

7.4.1.1 Thrust Available Vs. True Airspeed

Tests were conducted to determine the thrust available as a function of true airspeed for the power settings and altitudes given on the following page:

Power Setting

Afterburner (augmented thrust)

Military

Normal

Idle

Altitude

Sea Level

15,000 feet

25,000 feet

35,000 feet

45,000 feet

55,000 feet

7.4.1.2 Thrust Required Vs. True Airspeed

Under steady, straight and level flight conditions, tests were conducted to determine the minimum thrust required to maintain these conditions as a function of true airspeed for the gross weights, center of gravity positions, configurations, and altitudes listed below:

Clean Configuration No. 1 (drop tanks off)

Gross weight: 24,000 lbs

Altitude : Sea level; 15,000, 25,000, 35,000, 45,000 and 55,000 feet

Clean Configuration No. 2 (drop tanks on)

Gross weight: 24,000 lbs

Altitude : Sea level; 15,000, 25,000, 35,000, 45,000 and 55,000 feet

Clean Configuration No. 3 (drop tanks off)

Gross weight: 20,000 lbs

Altitude : Sea level and 35,000 feet

Clean Configuration No. 4 (drop tanks on)

Gross weight: 28,000 lbs

Altitude : Sea level and 35,000 feet

Landing Configuration

Gross weight : 28,000 lbs

Altitude : Sea level

Configurations: Gear down, speed brakes open, drag chute deployed

Gear down, speed brakes open

Gear down

Speed brakes open

Clean

7.4.2 Procedure for Conducting Thrust Available Tests

After loading the F-100A program into the computer and initiating computer operation, the procedure for conducting the tests was as follows:

a.) Actuate Roll Angle Lock control.

b.) Perform a normal engine start, and accelerate the engine to Military power plus afterburner.

c.) Accelerate to takeoff speed.

d.) When the simulator is airborne, actuate Altitude Lock control.

e.) Retract landing gear.

f.) Actuate Freeze control, halt computer, and read in value of airspeed for first data point; program causes discrete output to indicate that simulator has achieved desired airspeed.

g.) Clear the Freeze condition and Fly Simulator to maximum speed; may require placing simulator in a dive attitude.

h.) When Desired Airspeed discrete output indicator is on, place simulator in the Freeze mode, halt the computer and print out pertinent data, as described in section 7.2.1, Output Printer.

i.) Read in value of next desired airspeed.

j.) Clear the Freeze condition and decrease airspeed to the next desired value by placing simulator in a slight climb attitude.

k.) Repeat steps h, i, and j for all airspeeds.

l.) When tests for this power setting are complete, reduce power setting to Military power and repeat steps f through k.

m.) When tests for this power setting are complete, reduce power setting to Normal and repeat steps f through k.

n.) When tests for this power setting are complete, reduce power setting to Idle power and repeat steps f through k.

o.) When tests for this power setting are complete, those for this altitude are also complete.

p.) Release Altitude Lock control and Fly Simulator to next specified altitude. When this height has been reached, actuate Altitude Lock control. Increase power setting to Military power plus afterburner.

q.) Repeat steps f through p for all altitudes.

At the completion of the tests, the accumulated data is converted to decimal form and plotted on the appropriate acceptance test data sheets.

7.4.3 Test Procedure for Conducting Thrust Required Tests

The performance of the Thrust Required tests is not as straightforward as the performance of the Thrust Available tests, due to the added requirement that the initial conditions of the simulated aircraft be strictly achieved and maintained. Further, the tests are very time-consuming, due to the time required for stabilizing the longitudinal equations; equilibrium is attained only when $\dot{q}_1 - q_1 = \dot{w} - \dot{u} = 0$, and rate of climb and longitudinal pitching moment M_{y_s} are both equal to zero. To establish these exact equilibrim conditions from the simulated cockpit, even with the aid of the special test controls, requires an extraordinary amount of skill and patience; approximate equilibrium was therefore established as a satisfactory initial condition. Approximate equilibrium is defined as the condition existing if the following parameters can be forced to zero within the tolerances specified:

$$\dot{u} = 0 \pm 0.0625 \text{ ft/sec}^2$$

$$R/C = 0 \pm 1 \text{ ft/sec}$$

$$q_1 = 0 \pm 0.000732 \text{ rad/sec}$$

$$p = v = r = 0 \pm 0$$

The Thrust Required test procedure is essentially the same as that for the Thrust Available tests, with the notable exception that meaningful data may not be extracted from the simulator until straight and level flight equilibrium has been achieved. The accumulation and extraction of test data from the computer is performed by the same print-out program used for the Thrust Available tests. After the octal-to-decimal conversion, test data are plotted on the same Thrust Available and Required data sheets.

An indication of the accuracy of the UDOFT simulation of the static characteristics of Thrust Available and Thrust Required for the F-100A aircraft is illustrated by the acceptance data sheets (figures 93 and 94) for altitudes of 15,000 and 25,000 feet, respectively.

7.5 Dynamic Testing of the UDOFT F-100A Simulation Model

Static or performance testing of a simulated aircraft proves to be far less troublesome than dynamic testing. Dynamic testing involves much more of the mathematical model of the aircraft, thereby extending the test effort to more possible problem areas. Many problems can be traced directly to the possibly faulty modeling of the aircraft even with perfect modeling; however, erroneous results may accrue from improper execution of the tests. This latter is quite likely to occur, as dynamic testing is carried out predominantly from the simulator cockpit; thus human error enters into what would otherwise be an array of tests as mechanical as the performance tests.

The ideal way to perfect dynamic testing is indeed to make it mechanical; that is, remove the human element from the simulator cockpit. A number of UDOFT computer programs were accordingly prepared, which in essence replace the pilot. The sections that follow discuss these cockpit programs and the results obtained from a particular class of dynamic tests; namely, the short-term longitudinal stability tests.

7.5.1 Steady-State Straight and Level Flight Equilibrium (SSLFE) Program

To minimize the time and effort required for establishing steady-state straight and level flight equilibrium, a program was prepared to force the longitudinal equations to a SSLFE condition. Such a program appeared necessary, not only for eliminating the tedious process of establishing required initial conditions, but also for rapidly and accurately reestablishing the same conditions for the purpose of displaying the ability to repeat test results. Further, this program greatly lightens the burden of establishing a large number of SSLFE conditions, and is required when demonstrating static longitudinal performance. A graphic indication of the large number of data points that can be obtained readily is presented in figure 95.

Bringing the simulated aircraft to the SSLFE condition is a two-step operation. The lateral directional equations must first be zeroed; then equilibrium can be established in the longitudinal plane at the proper airspeed.

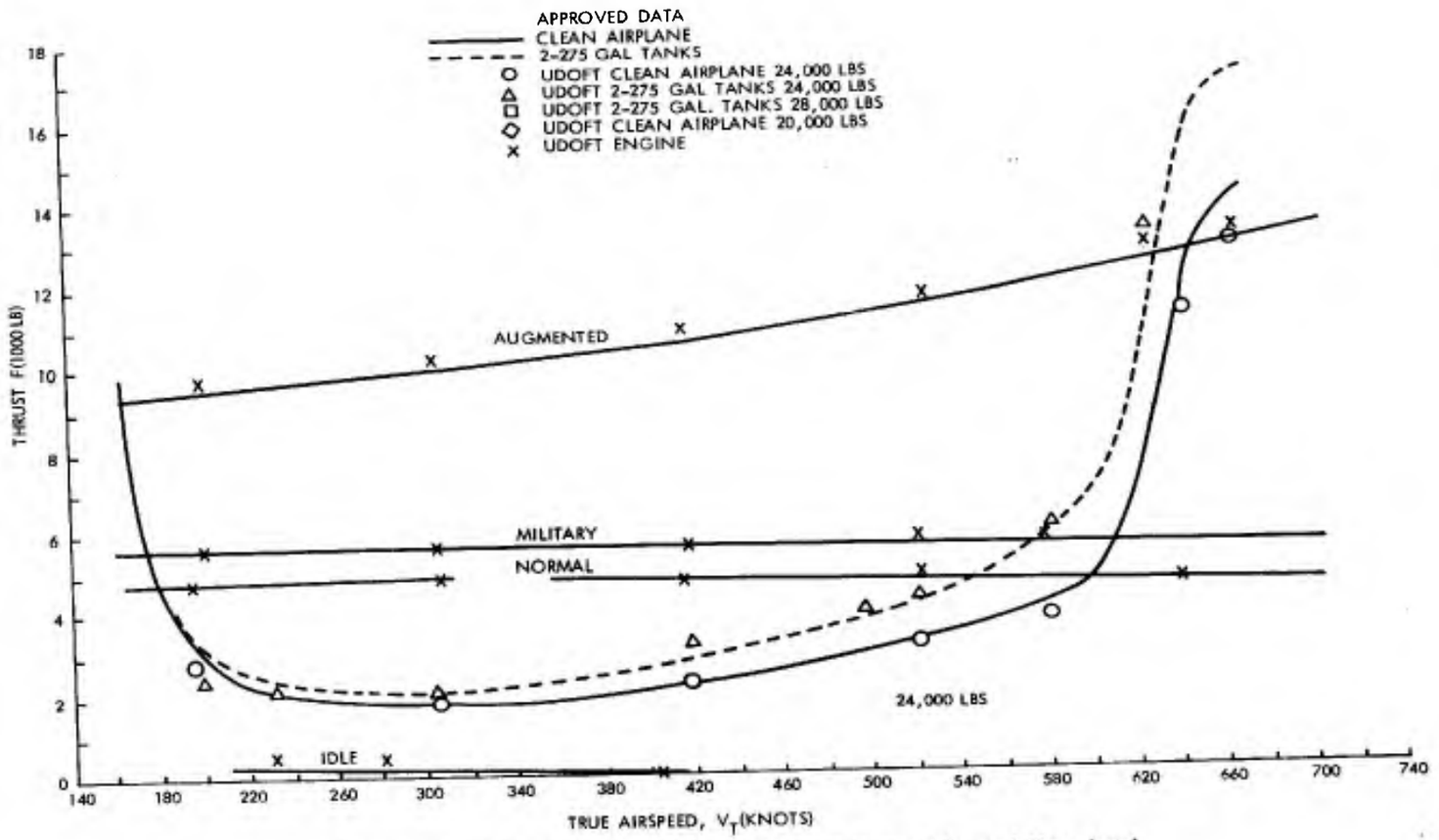


Figure 93. Performance Curves—Thrust Available and Required 15,000

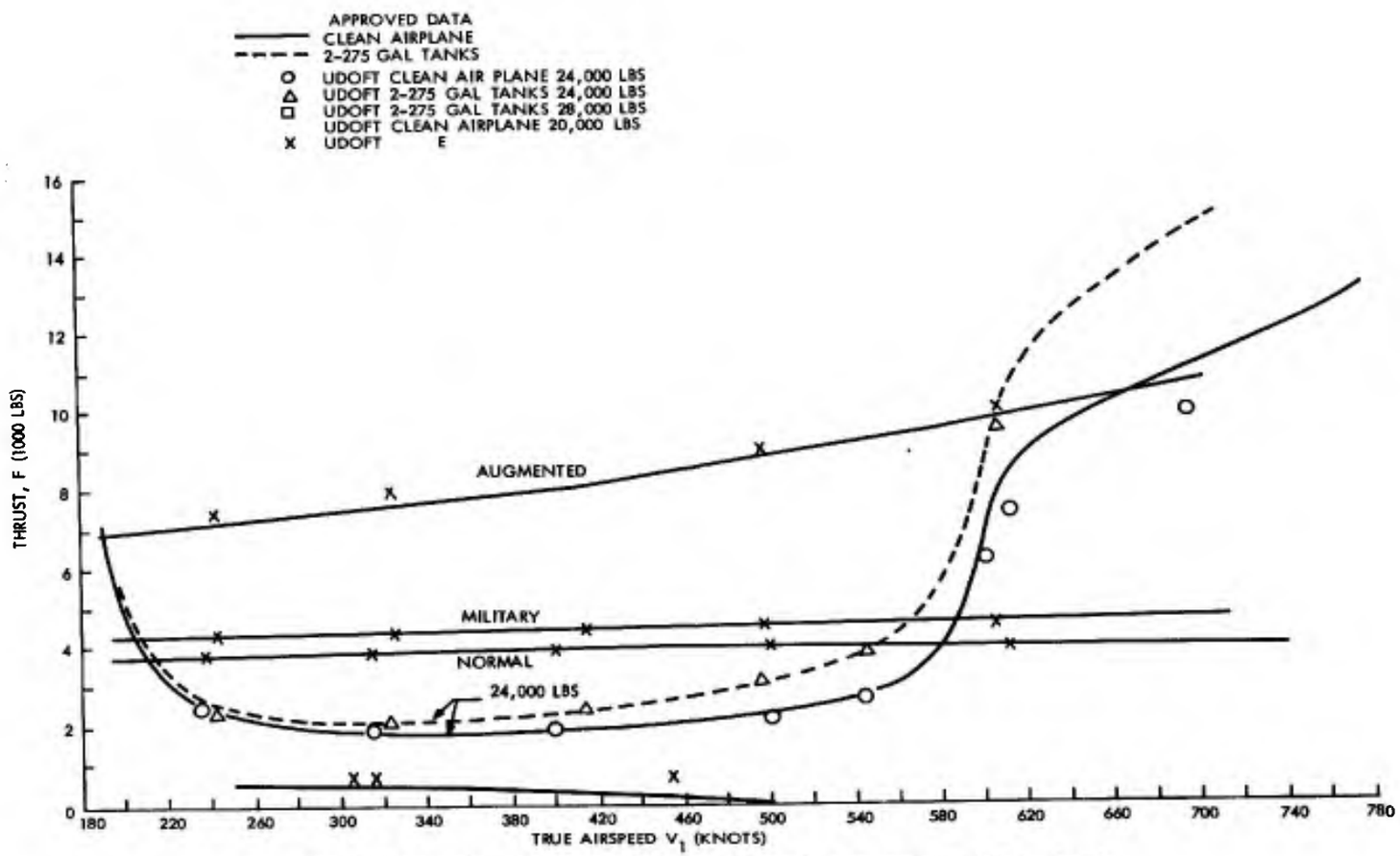


Figure 94. Performance Curves—Thrust Available and Required 25,000

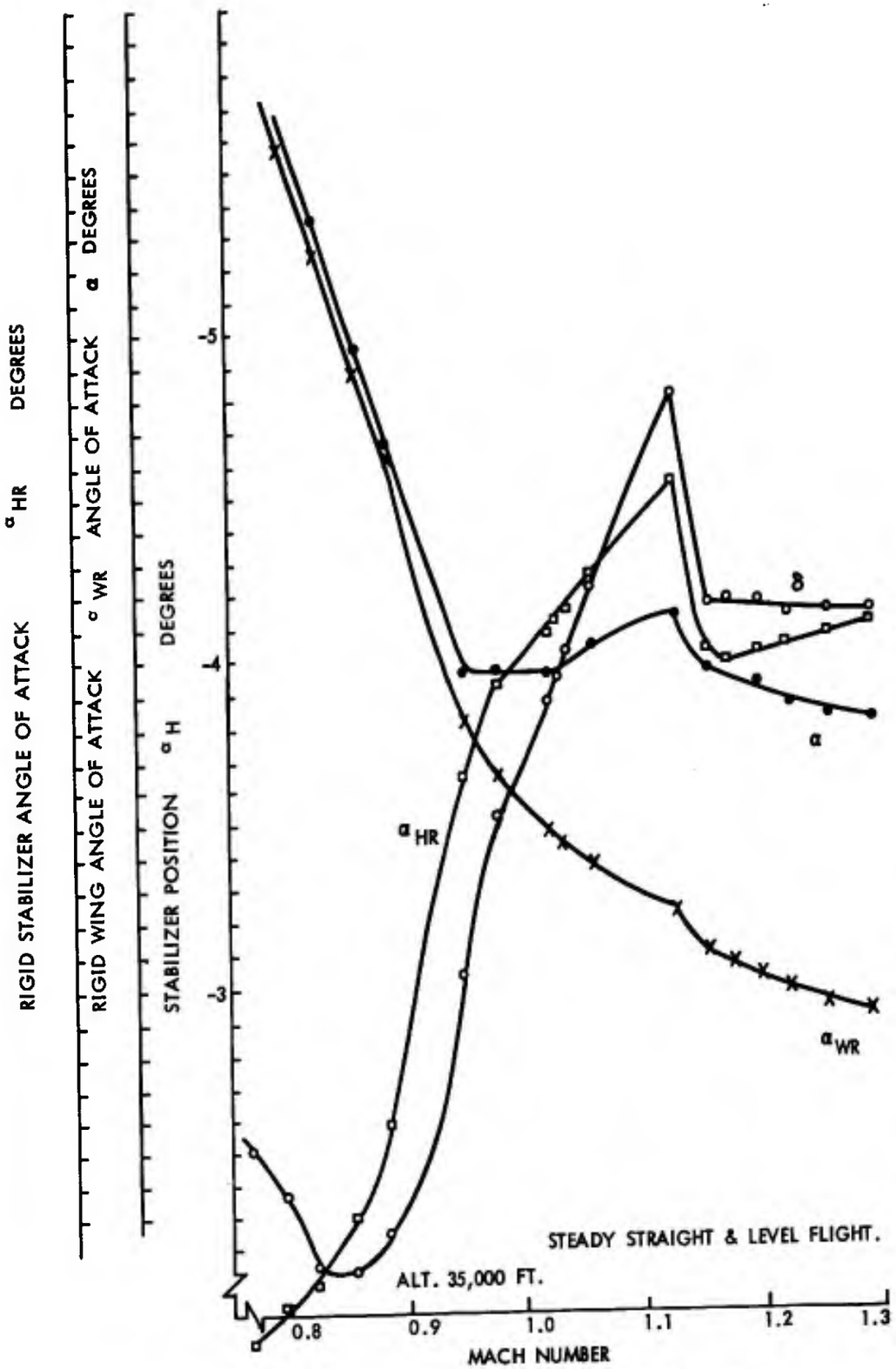


Figure 95. Performance Curves–Static Longitudinal Stability

Since the aircraft is symmetrical about the vertical plane passing through the longitudinal axis, i.e., the plane defined by the X and the Z body axes, it is necessary only to zero the accelerations p, v, and r in order to establish lateral-directional equilibrium. However, it is possible that the aircraft will stabilize in some nonzero orientation; that is, roll angle and yaw angle will not necessarily be zero. Therefore, it is necessary to zero the flight control positions δ_a and δ_r.

Static equilibrium in the lateral-directional planes is achieved by ignoring the calculated accelerations $\dot{p}$, $\dot{v}$, and $\dot{r}$, and by substituting for them, in the integration table of past values, the negative of the associated rates:

$$\dot{p}_n = -(p_{n-1}) \tag{119}$$

$$\dot{v}_n = -(v_{n-1}) \tag{120}$$

$$\dot{r}_n = -(r_{n-1}). \tag{121}$$

The system of lateral-directional equations will therefore return exponentially to zero, at the rate determined by the inherent time constant of the numerical integration formula. As mentioned previously in section 7.1, Special Test Controls, the rates may not be forced identically to zero. Since it is desirable to have them identically zero, the program includes a brief routine that makes the rates and the accelerations identically zero when they have approached zero to within some predetermined tolerance. When equilibrium is achieved for the lateral-directional equations, it is maintained by setting both δ_a and δ_r equal to zero. If a lateral-directional test is to be initiated from the SSLFE condition, δ_a and δ_r must be reactivated because aileron and rudder flight controls in the simulator cockpit have been made ineffective.

The next step is the establishment of equilibrium in the longitudinal plane, which requires that the desired rate of climb be specified to the SSLFE program. In addition, to allow the program to achieve an approximate rather than an exact equilibrium condition, tolerance must be specified for q_1, $\dot{R}/C$, and $\dot{u}$ (see section 7.4, Procedure for Performance Testing).

7.5.1.1 Description of the SSLFE Program

Control of the simulated aircraft by the SSLFE program is effected by three Discrete Input switches; one for zeroing the lateral-directional equations, one for zeroing the longitudinal equations, and one for establishing the required airspeed. The SSLFE program is not entered during the normal simulation program computation cycle unless the Discrete Input switch which controls lateral-directional equilibrium is actuated. Exceptions to this are the integration tables of past values for stabilizer and throttle position which are maintained, though not used, during the normal program cycle. The reason for this will become apparent later in this discussion. Upon actuation of the lateral-directional equilibrium control, the associated equations are zeroed. When the lateral-directional equations are identically zero, the longitudinal equilibrium control may be actuated. The longitudinal equations are forced to zero by controlling the stabilizer as follows:

$$\delta_H = \int_{O_{33}} \dot{\delta}_H \tag{122}$$

where

$$\dot{\delta}_H = K_1(\dot{R}/C_n - \dot{R}/C_D) - K_3 \int_{O_{33}} K_4(\Delta\dot{\delta}_H)$$

and

$$(\Delta\dot{\delta}_H) = \delta_{H_n} - \delta_{H_{n-1}}$$

The only restriction imposed upon the use of this approach is that the simulator must be in a condition of longitudinal equilibrium; i. e., the rate of climb must be constant. This restriction is necessary in order that the rate damping term,

$$-K_3 \int_{O_{33}} K_4(\Delta\dot{\delta}_H),$$

be initially synchronized with the forcing function, the rate of climb error, $K_1(R/C_n - R/C_D)$. This restriction imposes no great burden since a constant rate of climb is relatively easy to achieve. The constants in the equations have been assigned values of 1; however, the significance of this single bit is altered by means of shift instructions. This permits the selection of any time constant for achieving steady-state straight and level flight equilibrium.

When the longitudinal equilibrium control is in the deactivated state, the past values for $\dot{\delta}_H$ and $(\Delta\dot{\delta}_H)$ appearing in the tables for the integration

$$\int_{O_{33}} \dot{\delta}_H \text{ and } \int_{O_{33}} K_4(\Delta\dot{\delta}_H)$$

are made identically zero. However, the current values of stabilizer position, δ_{H_n}, are stored in the table of past values for the integration

$$\int_{O_{33}} \dot{\delta}_H$$

so that there will be no discontinuity in stabilizer position when the longitudinal equilibrium control is actuated. When the control is actuated, the stabilizer commands from the cockpit are ignored and stabilizer control is effected completely by these equations.

Once longitudinal equilibrium has been achieved, the required airspeed control may be actuated. Acceleration to the proper airspeed is accomplished in a manner similar to that used for achieving level flight except that thrust, T, is integrated:

$$T = \int_{O_{33}} \dot{T} \tag{123}$$

where

$$\dot{T} = K(Ma_n - Ma_D) - K_3 \int_{O_{33}} K_4(\Delta\dot{T})$$

and

$$(\Delta\dot{T}) = T_n - T_{n-1}$$

As in the case of the stabilizer control routine, the results of the normal thrust computation are prevented from directly affecting the flight conditions; however, the current values of computed thrust are retained in the table of past values for the integration

$$\int_{O_{33}} T$$

prior to actuation of the required airspeed control, so that there will be no discontinuity in thrust. The rate at which the desired airspeed is achieved is a function not only of the time constant of the thrust integration but also of the phugoid characteristics of the aircraft.

The remainder of the SSLFE program provides a means for monitoring, from the computer console, the progress of the SSLFE program. Two console Discrete Output indicator lights are associated with each of the four parameters, q_1, R/C, Ma and $\dot{M}a$. One light indicates that the value of the particular parameter is greater than the required value; the other light indicates that it is less than that value. When the value of a particular parameter reaches the required value, within the specified tolerance, both indicator lights are turned on.

The monitoring function of these indicators is important to the attainment of initial conditions for a number of acceptance tests. Since the SSLFE program is as yet not completely automatic, the Discrete Output lights indicate that a required condition has been achieved and that the next step of the process may commence. Because the lateral-directional equations are zeroed rapidly, no indication is required; therefore the longitudinal equilibrium control may be actuated immediately after the lateral-directional equilibrium control. The required airspeed control may not be actuated until satisfactory longitudinal equilibrium is signaled by the indicators for q_1 and R/C. Achievement of the required airspeed is displayed by the indicator for Ma; the $\dot{M}a$ indicator shows that the airspeed is constant. Thus, if both the Ma and $\dot{M}a$ pairs of indicators are turned on, then airspeed has stabilized at the required value. At this point the initial conditions have been established and the test may now be commenced.

If a particular altitude is also to be established, entry of the SSLFE program is deferred until the simulated aircraft has been taken to the desired altitude by means of the modified Altitude Lock Control. (See section 7.1.5, Altitude Lock.)

7.5.2 Dynamic Response Testing

To demonstrate the dynamic response of a flight simulator to a sudden change in the flight controls requires that the related parameters be available for recording as a function of time. In the UDOFT scheme, the recording of the transient responses of these parameters is accomplished by means of Analog Output channels which provide voltage signals for recording (refer to section 7.2.2, Analog Outputs). To aid in the recording of dynamic test data, the Main Test Pattern (MTP) program was devised.

The SSLFE program, together with the MTP program, enables all dynamic tests to be conducted semi-automatically from the computer console. The SSLFE program establishes the required initial conditions and the MTP program introduces the forcing function into the simulation program and causes the response to be recorded. Automation of the process of dynamic testing could have been extended to include analysis of the results. Automatic determination of the period and the time to damp to half-amplitude of the resultant oscillation would require a program that performs the calculation indicated in Section VI, Direct Determination of Damping and Natural Frequency from Time Responses, of Technical Report NAVTRADEVCEN 318-1, Dynamic Test Program for Weapon System Trainers. However appealing this may be, it has not been and probably will not be done.

7.5.2.1 Main Test Pattern (MTP) Program

The purpose of the MTP program is to aid in the calibration of the recording equipment, to initiate the dynamic tests, and to output the transient behavior of selected parameters to the recording equipment.

Calibration of the recording equipment is aided by the program generation of a square wave pattern of diminishing amplitude immediately preceding the initiation of the dynamic test and again immediately following the completion of the test (figure 96). The recorded pattern also gives scale significance to the transient response data recorded subsequently. The grid is formed by drawing lines which connect like cycles of the patterns that are recorded before and after the test is conducted; this implements reading the transient response data directly from the recording. The grid lines also provide some indication of the short-term drift of the Analog Output channels and the recorder amplifiers, and the linearity of the recording mechanism.

The MTP program (see flow chart, figure 97, and diagram, figure 98) is fairly simple and straightforward. Entry into the program is made at the beginning of each 50 millisecond iteration cycle. If both the pattern flag (program controlled) and the test Discrete Input are positive, the program is short circuited and entry into the main simulation program is made immediately. If, however, the test Discrete Input is negative (test Discrete Input has been activated), the program commences immediately to computer the amplitudes of the pattern for each channel.

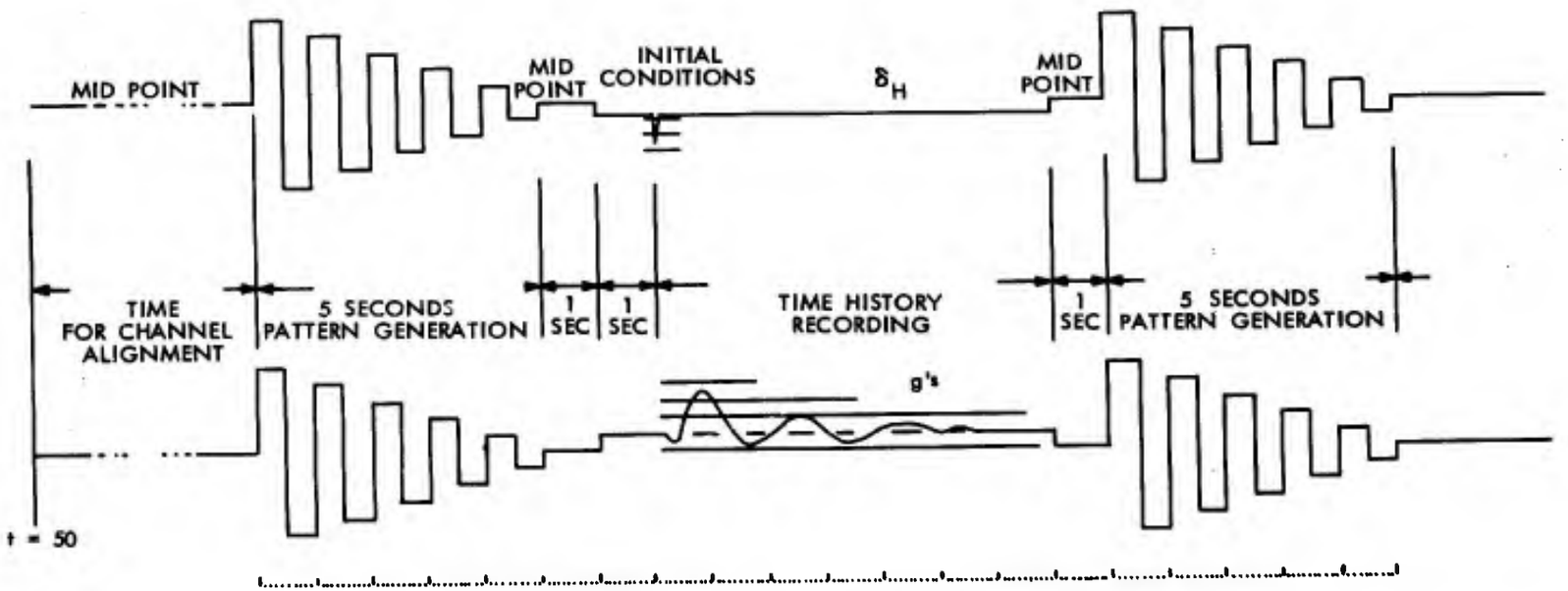


Figure 96. Main Test Pattern Recording

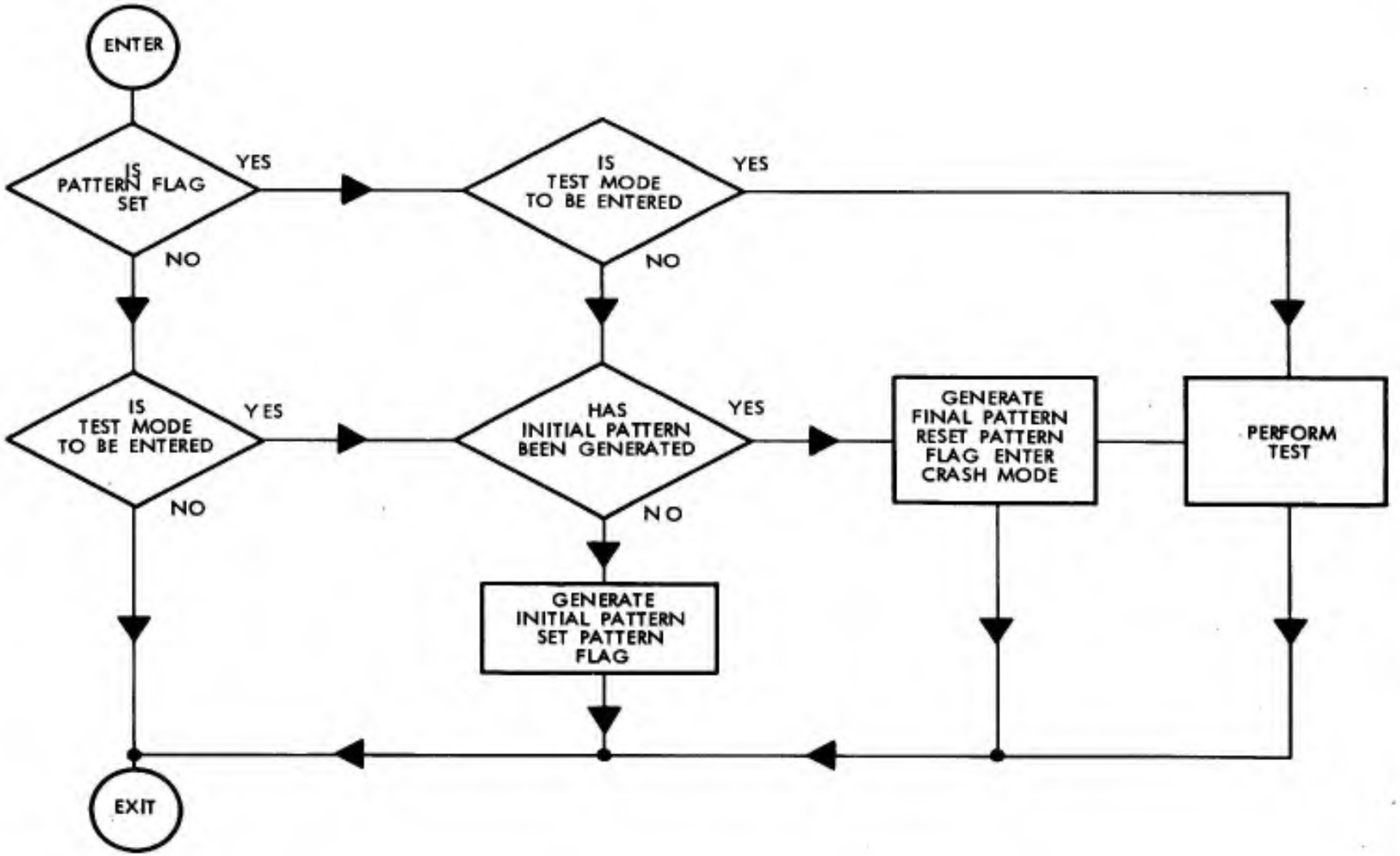


Figure 97. Main Test Pattern Program Flow Diagram

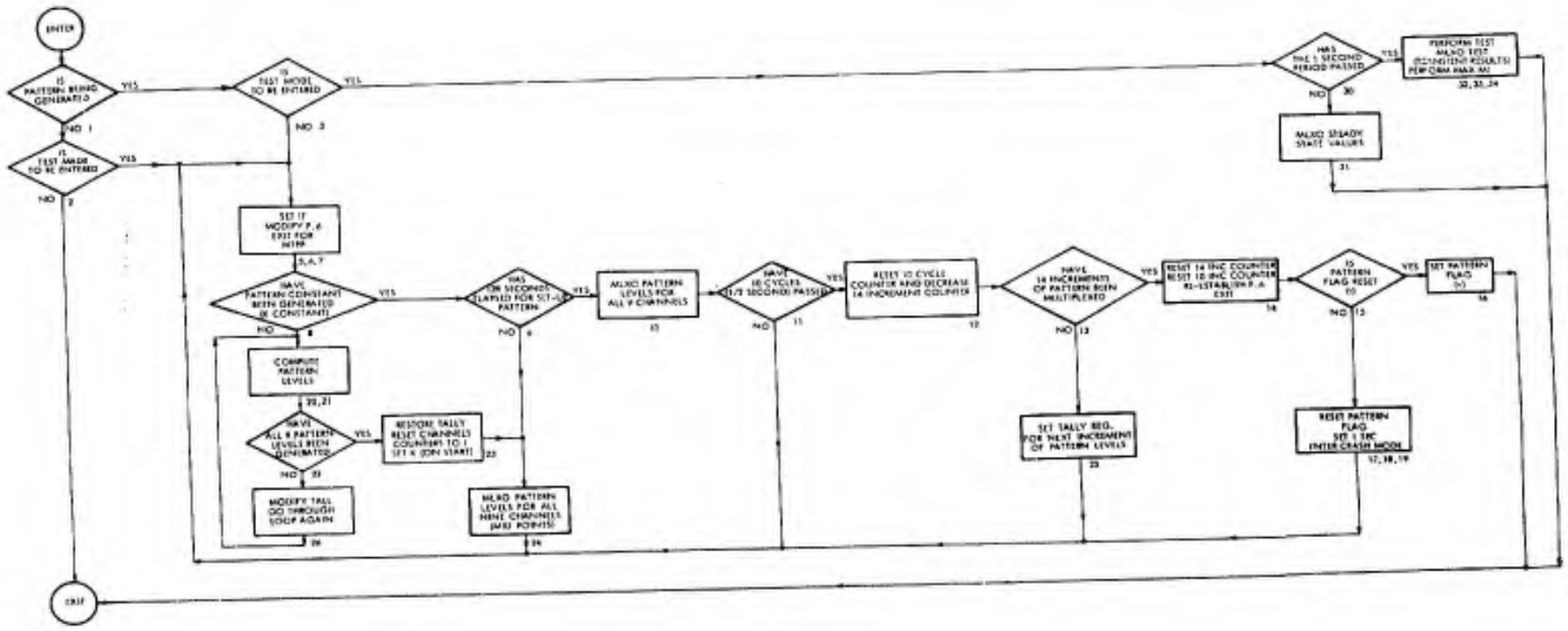


Figure 98. Main Test Pattern Program Flow Diagram

The pattern constants (amplitudes) are derived from the knowledge of the anticipated steady-state and maximum values of the parameters to be recorded. Numerical data describing the mid-range value and the maximum value of each parameter to be recorded is inserted into the computer prior to the execution of the test. The value of the parameter is essentially its steady-state value. In many cases, particularly those cases involving lateral-directional tests, the steady-state values of the pertinent parameters are zero; thus the mid-range value of the parameter, for pattern generation purposes, is specified as zero. For those cases where the steady-state values deviate only slightly from zero, the mid-range value is still specified to be zero. The program compares the specified mid-range and maximum values for each parameter to obtain the anticipated dynamic range of the parameter, which is designated Δ. From this, the program computes and stores sequentially for each output channel the following fourteen pattern data points: mid-range, mid-range, mid-range + Δ, mid-range - Δ, mid-range + 0.8Δ, mid-range - 0.8Δ, mid-range + 0.6Δ, mid-range - 0.6Δ, mid-range + 0.4Δ, mid-range - 0.4Δ, mid-range + 0.2Δ, mid-range - 0.2Δ, mid-range, and mid-range. Each of these points is passed to the recorder for a period of one-half second. Upon completion of the first pattern, program control is returned via the main program to the beginning of the MTP program. For a period of one second, the steady-state values of the selected parameters are passed to the recorder, followed immediately by the actual test. During the test, the behavior of the selected parameters is passed to the recorder, and also examined by the maximum routine, which extracts and stores the maximum and minimum values of each selected parameter. This numerical data and the values of the parameters at the end of the test are subsequently extracted and printed out by the output printer. This additional data provides another quantitative means of calibrating or verifying the scaling of the recorded parameters.

The test is terminated manually by means of the test Discrete Input switch. Immediately thereafter the second test pattern is recorded, at the completion of which the simulation problem is forced into a crash condition in order to freeze the program.

Once the data for the MTP program has been generated, the execution of the test problem is relatively simple. The procedure followed for a short-term longitudinal stability test is briefly:

a. Initialize simulator to required altitude, airspeed, and rate-of-climb using Altitude Lock control and Steady-State Straight and Level Flight Equilibrium control.

b. Freeze the simulator.

c. Read in Main Test Pattern program along with parameter selection and scaling information, pattern constant data storage locations for steady-state and test values of the selected parameters and the test program.

d. Printout initial steady-state values.

e. Return simulator to normal mode.

f. Start recorder and align channels.

g. When recorder is aligned and gain controls are set, activate test Discrete Input.

h. When recording indicates that the steady-state conditions have been regained (test complete) disable the test Discrete Input.

i. Wait for recording of final test pattern; disable recorder paper feed mechanism.

j. Printout maximum, minimum, and final steady-state values.

k. In order to conduct another test at a different altitude or airspeed return to Step a; Step c may be omitted if some scaling information is adequate.

The only problems encountered with the use of the MTP program are the derivation of the scaling data and the gain settings of the recorder. With regard to the scaling of the parameters to be recorded, it must be remembered that only the sign and the eleven most significant bits of number words in the UDOFT computer are converted to analog form. Thus, if a quantity to be recorded is described predominantly by the nine low order bits, it must be shifted to the left in order to attain any degree of significance. The extent of

shifting is determined, usually, by the anticipated maximum and minimum values of the parameter. The parameter is shifted so that its minimum value would be the least significant bits of the quantity to be converted to analog form. Care must be taken to prevent the maximum values of the parameter from causing an overflow. The quantity actually converted represents approximately the transient aspect of the parameter. If, however, the initial value of the parameter is zero, and the parameter varies about zero, the maximum absolute value of the parameter determines the extent of scaling.

Since the scaling is effected in powers of two, as a result of using shift instructions rather than multiply instructions, the MTP program provides only coarse control of the amplitude of the plot to be made by the recorder. Fine control of the recording is effected by the recorder. The gain settings are set as high as possible without causing the subsequent recordings to exceed the plus-or-minus one-half-inch range which provides the greatest recording linearity.

7.5.2.2 Recording Equipment

Test equipment developed by Cornell Aeronautical Laboratory Inc.*, was used extensively for collecting the real-time test data. This equipment was developed originally to aid in evaluating the dynamic response of d-c analog flight trainers. The equipment has many functions; however, only its recording function was used in conducting simulator dynamic test programs on the UDOFT system. The recording function is implemented with a ten-channel CEC 5-119P4 oscillograph, a CEC 5-036C Datarite unit, and ten oscillograph amplifiers designed by the same Laboratory.

The oscillograph mechanism is basically a number of optical galvanometers, with each galvanometer using the same incandescent bulb as the point-light source. The oscillograph used contains ten such galvanometer movements, spaced at one-inch intervals. There are no amplifiers in the oscillograph unit; gain and bias control are established by means of the specially built d-c amplifiers in the main test unit. The Datarite unit contains the recording paper which will be exposed in the oscillograph, the mechanism for advancing the paper, and the means for developing it, once exposed. Since the paper feed mechanism can be operated at a number of speeds, time calibration marks are provided on the recording paper. In the present arrangement, the fifth channel is used constantly to sweep the width of the paper every 100 milliseconds.

A viewing window of ground-glass plate allows observation of the position of the reflected light from all galvanometers. This display is most useful when aligning the various channels and establishing the maximum signal recording.

As with any other piece of electronic equipment, certain precautions must be taken in the use of the recording oscillograph:

a.) The test equipment d-c amplifiers and power supplies, the oscillograph galvanometer heater blocks, and the Datarite unit paper dryer must be on for at least thirty minutes prior to alignment of the amplifiers and the oscillograph.

b.) The quantity of developer solution should be checked frequently; it has a tendency to evaporate since its feed mechanism and reserve supply are in close proximity to the paper dryer.

c.) The developer solution feed mechanism must not be in contact with the paper when the recorder is not operating as the recording paper will adhere to the feed mechanism and ultimately tear when the recorder is started.

The procedure for using the recording equipment is relatively straightforward:

a.) Apply power to the test equipment and the recording oscillograph at least one-half hour prior to time of intended use.

---

*"Application of Dynamic Test Techniques to Weapon System Trainers", Contract N61339-318, U. S. Naval Training Device Center, Port Washington, N. Y.

b.) Check to see that Simulator Signal ON light, on test equipment, is lit. This light signifies that the signals displayed on the ground-glass viewing window are the signals to be recorded. When the Zero Signal light is lit, the galvanometers project the ground signal associated with the test equipment. This signal can differ considerably from the simulator signal, even to the extent of driving one or more of the galvanometers off-scale. The Simulator Signal ON light is controlled by the Signal Control pushbutton switch.

c.) Adjust gain for each channel to be used. Some previous knowledge of the behavior of the parameter to be recorded is required, since recordings of significant amplitude must be obtained without exceeding the plus-or-minus one-half inch maximum linearity constraints of the individual channels.

d.) Adjust the bias controls for each channel to be used. These controls are varied until the recordings of the parameter mid-range outputs from the UDOFT computer, as generated by the MTP program, are centered on the inch marks in the ground-glass viewing window. Channels 1 through 4 should be aligned to the left of the time-mark channel; channels 6 through 10, to the right.

e.) Start the recorder, (recording paper feed mechanism.)

f.) Adjust the developer solution feed mechanism to make contact with the recording paper. Allow sufficient paper to be fed through the Datarite unit to insure proper developing.

g.) Test may be initiated at this time.

h.) Upon completion of the test run, disengage the developer solution feed mechanism and allow additional paper to be fed through the Datarite unit before stopping the recorder.

7.5.3 Short Period Longitudinal Response

As part of the original F-100A acceptance testing program conducted on the UDOFT system, the response of the longitudinal equations to an abruptly changing input was demonstrated in a series of dynamic longitudinal stability tests. The initial conditions for which these tests were conducted were established manually; i.e., the simulator was hypothetically flown to the required altitude and airspeed, and trimmed-out from the cockpit. The longitudinal equations were forced to oscillate by establishing the conditions as prescribed by the initial values presented in the approved data sheets (see figure 99); i.e., the longitudinal flight control was displaced aft until the required acceleration was achieved and then returned rapidly to the original trim condition. The resultant oscillation was recorded and transferred to the approved data sheets. It is readily apparent from the test results (see the first and third columns of table XII) that the dynamic longitudinal performance of the simulated F-100A vehicle was unacceptable. Further study in this area was therefore undertaken.

The first problem undertaken was the elimination of inconsistencies in results obtained from different runs of the same test. This brought about the development of the Steady-State Straight and Level Flight Equilibrium program, by means of which it became possible to reestablish the same initial steady-state conditions from one test run to the next. In addition, a program means for automatically inserting a standardized forcing function (in this case a horizontal stabilizer deflection pulse) was developed. Although test results were far more consistent, there was no improvement in their acceptability. If anything, the test results deviated even more from the approved data. (cf. columns 1, 3, and 4 of table X.)

Since the data used to construct the F-100A digital simulation program had been reduced for the purpose of developing a special-purpose analog computer, it appears fruitless to attempt to improve the dynamic longitudinal performance below Mach 1.0 without reworking the original data. Neither time nor funding allowed such an undertaking, so the discrepancies in performance below Mach 1.0 were attributed to the data and not to the shape and nature of the program. However, such an obvious discrepancy as that which appeared at Mach 1.25 and 35,000 feet could not be passed over so lightly.

In an attempt to isolate the cause of these erroneous results, the aid of North American Aviation Inc., was solicited. N.A.A. was requested to undertake a brief

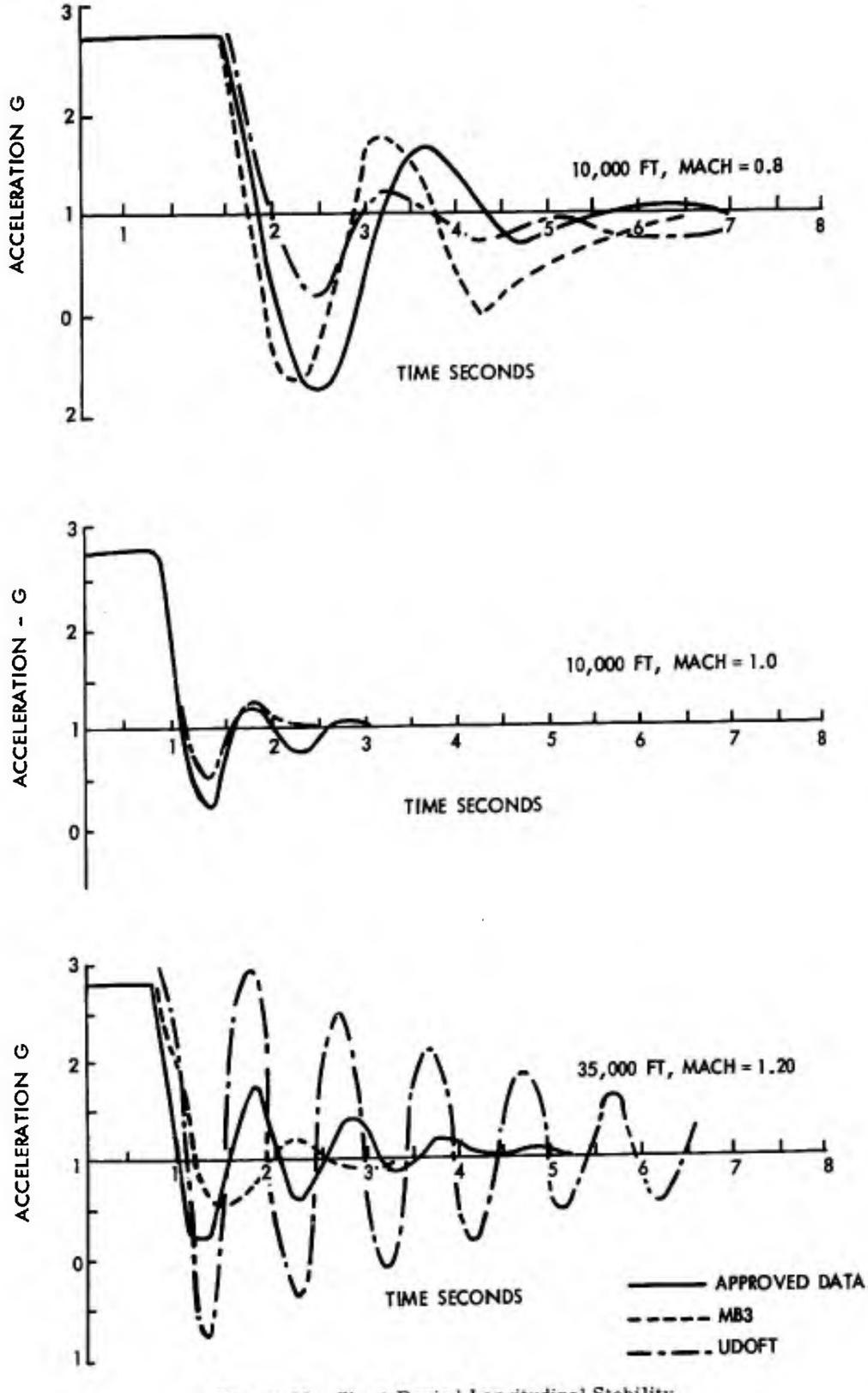


Figure 99. Short Period Longitudinal Stability

TABLE XII

COMPARISON OF RESULTS OF DYNAMIC LONGITUDINAL STABILITY TESTS FOR THREE FLIGHT CONDITIONS

CONDITION I	Mach = 0.8; Altitude = 10,000 ft.; Weight = 24,150 lbs.					
TEST	(1) NAA Approved Data	(2) Melpar Accept. Test Results	(3) UDOFT Accept. Test Results (Cockpit Control)	(4) UDOFT Accept. Test Results (Program Control)	(5) NAA Verification Test Results	(6) UDOFT Accept. Test Results (Improved)
PERIOD (SEC.)	2.4	2.0	1.7	1.35	.7261	1.4
TIME TO DAMP (SEC.)	0.8	1.4	1.0	0.4	.4326	0.6
CONDITION II	Mach = 1.0; Altitude = 10,000 ft.; Weight = 24,500 lbs.					
PERIOD (SEC.)	1.0	--	1.0	0.65	.4652	0.65
TIME TO DAMP (SEC.)	0.5	--	0.35	0.35	.3519	0.45
CONDITION III	Mach = 1.25; Altitude = 35,000 ft.; Weight = 24,150 lbs.					
PERIOD (SEC.)	1.0	1.4	1.0	0.95	.9173	0.92
TIME TO DAMP (SEC.)	1.1	0.8	2.5	7.2	1.385	1.1

program to verify the flight dynamics of the F-100A as simulated on the UDOFT system. That verification program was conducted on the basis of the data used to develop the digital simulation program and the digital computer programs* developed by N.A.A. for verifying the real-time response of analog simulators. Nine tests were conducted for this experimental verification, involving three flight conditions and three modes for each condition. The flight conditions were: low speed/low altitude, transonic, and maximum cruise; the modes were: short period longitudinal mode uncoupled, directional mode uncoupled, and five-degrees-of-freedom transient mode.

The comparative results of the nine tests varied considerably. Agreement in the roll and the yaw responses was fair; the differences can probably be accounted for by a small difference in aileron effectiveness. Agreement in the longitudinal mode was poor (cf. columns 4 and 5 of table X). No consistent pattern was evident which would indicate a specific problem area. For the first two conditions, the damping is acceptable; however, the periods differ by from 4:3 to 2:1. For the third condition, the period is acceptable but the damping times differ by a factor of 5:1.

It appears, on the surface, that the contribution of the horizontal stabilizer to the lift and moment equations was not being simulated properly. This aspect was, therefore, investigated further. Consideration was given to the equations for α_{HR} and α_{WR}; it had been observed that the Melpar equations resulted in an implicit solution for α_{HR}. The UDOFT program however, considered the equation as though it explicitly defined α_{HR} at time n by using the value of α_{HR} at time n-1 within the equation. Under steady-state conditions, the effect of using the past value within the equation is insignificant; however, under high dynamic conditions, such as are exhibited in the third test condition, the use of the past value can contribute appreciable phase shift to the closed-loop calculations. The first step then was to convert the implicit solutions of α_{HR} and α_{WR} to explicit solutions.

Secondly, during the conduct of tests by N.A.A., it was found that the UDOFT-derived trim conditions could not be duplicated. The cause of this discrepancy was again found to lie in the α_{HR} equation; more specifically, the coefficient of lift due to the horizontal stabilizer, C_{L_H}, was improperly multiplied by the wing area rather than by the stabilizer area. This error, obvious only in retrospect, would have been detected far sooner had static longitudinal stability tests been conducted; however, this had never been done for the F-100A aircraft.

Thirdly, incorrect stabilizer flexibility data, the stabilizer being assumed excessively flexible, was used in the simulation. The correct data was ultimately obtained and incorporated in the simulation program.

The net effect of these three changes on the longitudinal performance of the simulator is indicated in the sixth column of table XI. The most noticeable change occurred in the maximum cruise condition. If any improvement in longitudinal performance were to be expected, this is the condition most likely to benefit. Unfortunately, however, the final results in some cases are still not acceptable, and it is quite unlikely that any further improvement will be attempted because of the obstacle imposed by the inappropriate data used to develop the digital simulation program.

7.6 Conclusions

From the preceding discussion of the UDOFT system some may think that the quality of aircraft simulation derived from a digital system is inferior to that derived from an equivalent analog system. Because of the high-performance turbojet aircraft used in this program, the quality of simulation achieved by the UDOFT system is superior to that achieved by the analog counterpart. This evaluation of the digital system is supported by comparative data for three flight simulators; the F-100A, the F9F-2 and the F8U-1. In every case, the digitally simulated model was tested more extensively, and the test results were examined more critically. In some instances where the analog simulator was unable to perform, the same model simulated on the UDOFT system, displayed the desirable qualities.

*"An IBM Check-Out Program for Analog Mechanization of Up to Five Degrees of Aerodynamic Freedom," K.J. Dyda, Report No. NA59-854, North American Aviation Inc., Los Angeles, California, June 1, 1959.

Opponents of the digital approach criticize the superior quality of simulation achieved with a digital system and the ability of the digital system to represent the complex dynamics of an aircraft. These criticisms are derived from insufficient knowledge of the capability of the digital computer and partiality for using the time-honored analog approaches to the simulation of complex, real-time, man-machine systems. Just as adverse criticism was cast at the all electronic analog system when it was introduced as an improvement upon the electro-mechanical analog system, so too is the digital system being received cautiously. The same care exercised for developing an analog simulator must be exercised when preparing the simulation program that will be executed by the digital computer. Analog system design considerations of phase shift, drift, scaling, torques, inertias, and calibration have their counterparts in the digital environment. However, experience gained to date from the variety of flight simulation programs that have been developed for the UDOFT system indicates that developing a digital computer program from a complex mathematical model of an aircraft is accomplished more readily and more economically than the development of a comparable special purpose analog system.

The only apparent limitation to the use of the digital system is the simulation of system parameters that exhibit high dynamic characteristics. Operating at a twenty cycle per second solution rate the UDOFT computer, and in all probability any other digital computer with comparable computing speed, can reproduce dynamic responses up to two cycles per second. In order to reproduce responses of higher frequencies, a higher solution rate would be required. However, the solution rate cannot be increased greatly before practical limitations of computer speed and word length are encountered. Thus, there are limitations on the practicality of using the digital system. These limitations, however, are encountered only for a relatively small percentage of the simulation problems that are undertaken at the present.

It is the earnest opinion of those who have been associated with the UDOFT project that digital simulation of a high performance aircraft is feasible and practical. Further, the problems of this beginning art are no different from those encountered when developing any new technique in the scientific field and will be overcome as more research and development are performed, as is presently with the UDOFT system.

SECTION VIII

UDOFT SYSTEM UTILIZATION AND RELIABILITY

Since its installation in April 1960, the UDOFT system has been used for a variety of projects, ranging from the simulation of a submarine and a surface ship to that of an orbiting re-entry vehicle. The success of these projects has been somewhat dependent upon the reliability of the UDOFT computer. For those periods when computer operation has been flawless, significant results have been achieved from the research programs that have utilized the computer. Conversely, for those periods when computer operation has been sporadic, much valuable time has been lost to repairing the system and to re-running the problems until valid results have been achieved.

In order to indicate the degree of success attained by the UDOFT system in fulfilling its intended role of real-time simulation research system, the operation and the maintenance logs that are maintained by the system operating personnel have been summarized and the results are presented in the following sections. Two extended periods of operation are reviewed: namely, the first twenty months of operation (May 1960 through December 1961) and the twelve months of 1962. Before drawing any decisive conclusions from the material that follows, it must be remembered that the UDOFT computer is a unique device employing outmoded techniques. No radical or extensive system modifications have been made since the initiation of the UDOFT system development in 1956, whereby it would have been possible to improve the system operation due to the rapid advances in digital computer technology.

8.1 System Reliability - May 1960 through December 1961

During this twenty month period, the UDOFT system was actively manned for 3335.5 hours. Figure 100 indicates system usage wherein each major category of system utilization is plotted as a function of percent of total time. The terms presented in the graph are self-explanatory. However, because only the first category is defined as "available" time, one should not be led to believe that the computer system was inoperative for the remaining 46% of the manned time. The system was operable and could have been used for the activities comprising "scheduled downtime."

The graph is interesting from a cost viewpoint because it dramatically indicates those activities where costly maintenance effort has been applied. It also indicates those areas of computer operation which must be improved in order to reduce the effective cost of system operation by increasing the level of available time (available time is defined as that time during which the system is operated for the purpose for which it was intended). Ideally, this level should be 100%; however, this is a practical impossibility since such activities as system checkout and periodic preventive maintenance must be performed. In addition, time must be allowed to incorporate minor system improvements and other necessities. The time consumed by these categories of activity, as depicted on the graph, are reasonable and do not differ significantly from the levels anticipated at the time the system was installed.

The two disproportionate time consumers are unscheduled downtime due to corrective maintenance, and unscheduled downtime due to air conditioning system failure. Downtime must be anticipated on a large electronic system, however, the amount of unscheduled downtime required to perform UDOFT system repair is excessive. The total time lost by air conditioner was 8.5%, however, many more hours of unusable time may be indirectly attributed to the sporadic operation of the air conditioner. In many instances consistent high ambient temperature has been the known cause of computer system component failures.

The graph of figure 101 depicts on a weekly basis, the percentage of total manned system time that was available to a system user. When integrated over the period of twenty months, the data appearing in this graph, forms the bases for the Available Time entry in the graph of figure 100. The graph of figure 100 can be misleading from a reliability viewpoint. For this reason the term Operating Ratio (defined as the ratio of good time to attempted running time expressed in percent) referred to in figure 102 is used. Usable time which is utilized for preventive maintenance, system checkout, and system modifications, and unscheduled downtime which is attributable to the air conditioning

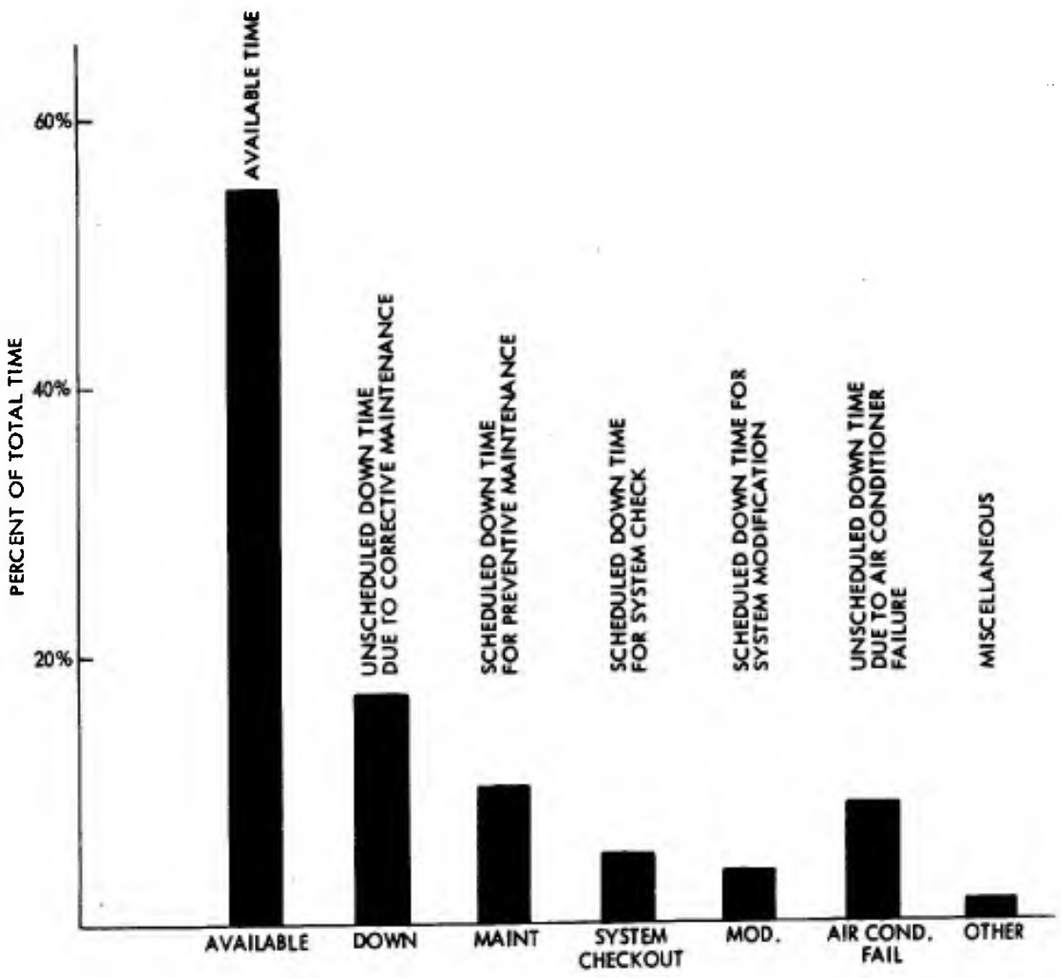


Figure 100. Total Hours 3335.5 During Period 15 May 1960 to 31 December 1961

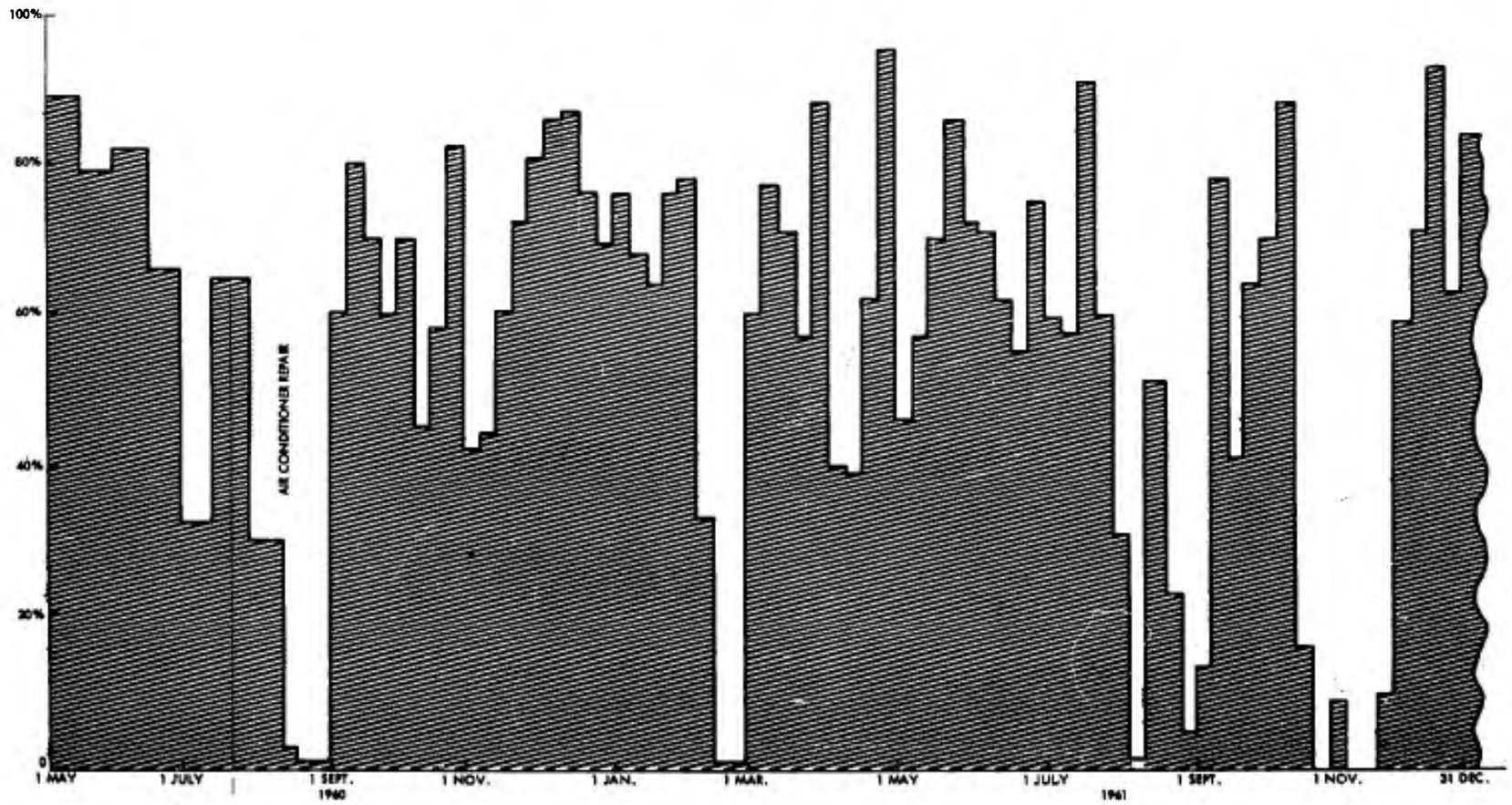


Figure 101. Available Time Versus Total Time of System Manning

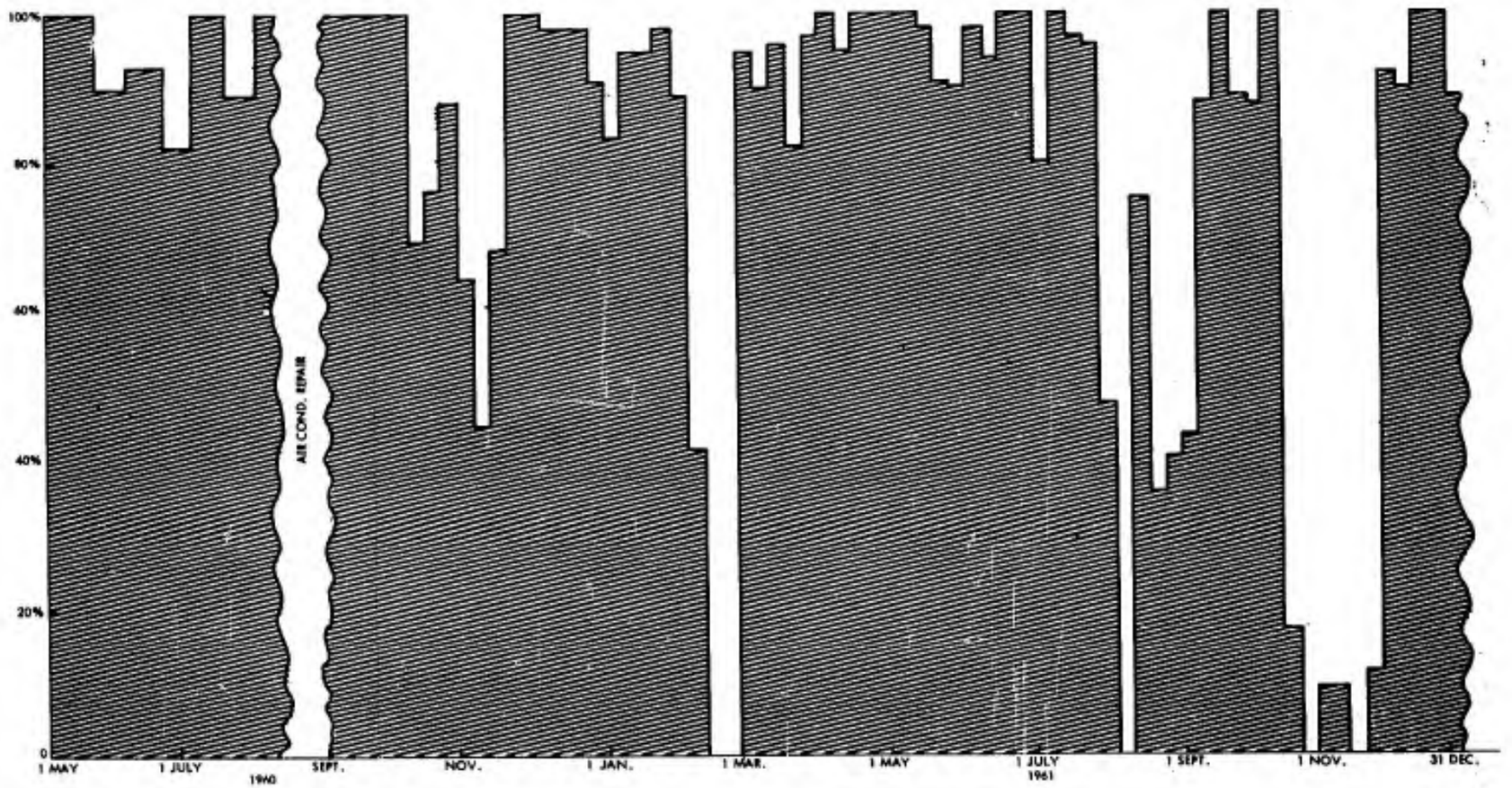


Figure 102. Operating Ratio

system are not considered as a part of attempted running time. Taking this into consideration, Operating Ratio can be defined also as the ratio of available time to the sum of available time and unscheduled downtime, expressed in percent. Using this form for a figure of merit for computer system operation, the average Operating Ratio for the entire period was 76%.

The graph of figure 102 depicts the weekly Operating Ratio for the same period. With the exception of the four periods of excessive downtime, the Operating Ratio was between 80% and 90%. The periods of excessive downtime were caused by several factors which are explained as follows:

1. October-November, 1960 - unreliable operation attributed to malfunctioning of computer discrete inputs.

2. February, 1961 - unreliable operation followed by inoperability attributed to misalignment of the five phases of the clock pulse.

3. August-September, 1961 - unreliable operation and inoperability attributed to improper memory drive currents and again misalignment of the five phases of the clock pulse and to insufficient preventive maintenance.

From the preceding it can be seen that excessive downtime was attributable qualitatively to the clock pulse system, memory drive currents, and computer discrete inputs. In order to indicate the quantitative contributions made by the major unreliable components of the computer system, the graph of figure 103 has been prepared. The graph very clearly indicates that the primary cause of system downtime was the clock pulse generation and distribution system.

A list of the major problem areas encountered, in order of importance, is as follows:

8.1.1 Clock Pulse Generation and Distribution

Misalignment of the five phases of the clock pulse was due to the aging of clock pulse system components. Field changes allowed improvement in operation of the system but only for a short time. Design of a new clock pulse system was initiated in late 1961 for installation in 1962.

8.1.2 Memory Drive Current

Improper writing into and reading out-of-memory due to inconsistant memory drive currents. This problem was eliminated by using higher quality diodes in the memory drive constant current source.

8.1.3 Discrete Inputs

The problem was not so much with improper operation of the discrete inputs as it was the difficulty of locating the faulty components. This was overcome by adding a maintenance control whereby all discrete inputs can be checked automatically under program control.

8.1.4 Memory Address Flip-Flops

Undesirable resetting of one or more address flip-flops during a memory cycle resulting in improper rewriting of information into memory. The resetting of the flip-flops was due to excessive loading of the aging triode vacuum tube used as address register flip-flop output cathode follower. The short-term remedy was the use of selected high-transconductance triodes in these places; further investigation in order to find a satisfactory solution was required.

8.1.5 Indicator Transistors

Inoperability of computer console indicators due to failure of transistor drivers. Due to the high mortality rate of the 2N35 transistor, it has been replaced by the 2N1304 transistor which has performed commendably.

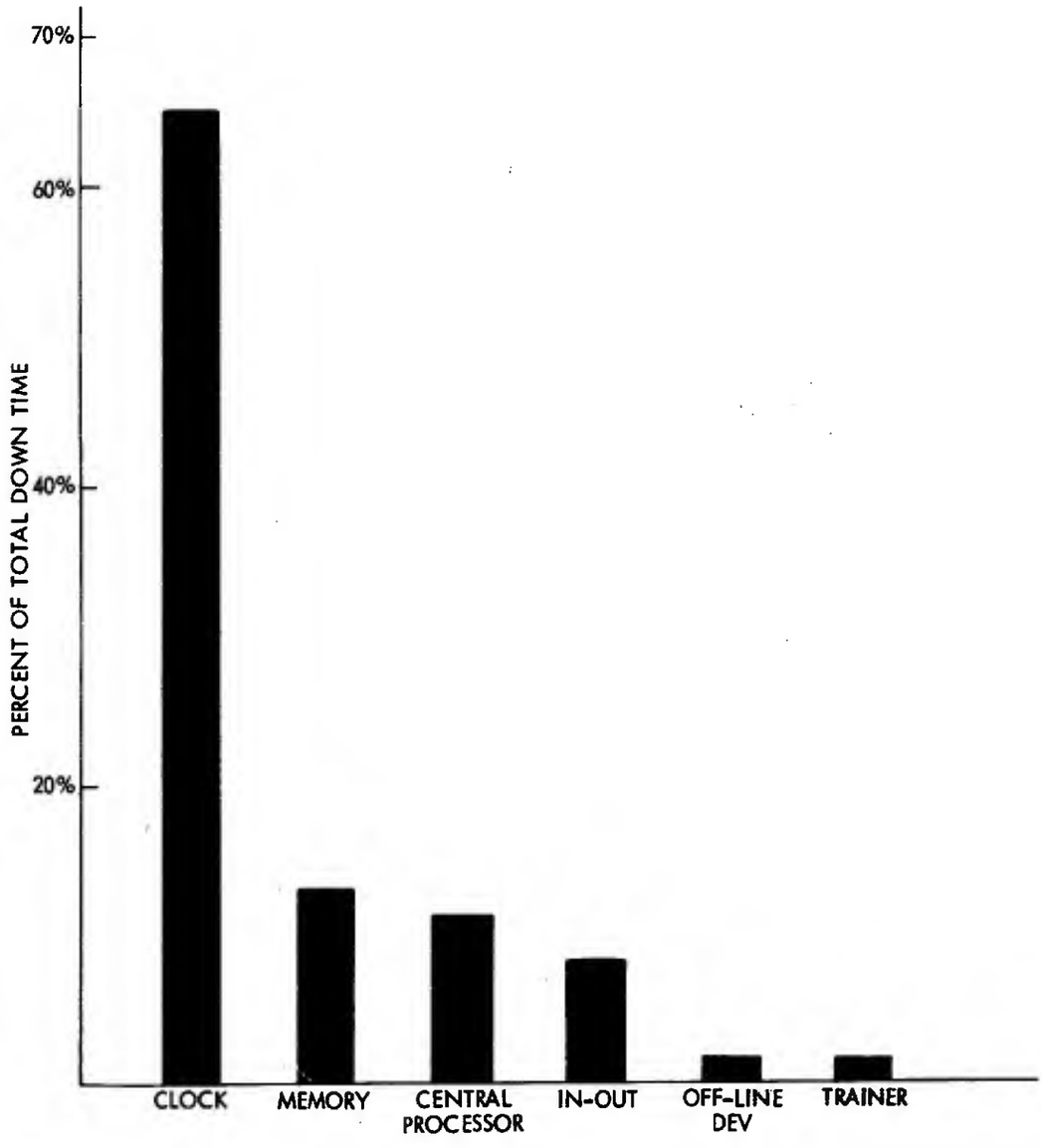


Figure 103. Causes of System Downtime

8.1.6 Console Switches

Failure of push-pull console switches. This problem has been overcome by replacing the switches, which had phosphor bronze spring actuators, with similar switches using a superior actuating mechanism.

8.1.7 Output Writer

Sporadic operation due to inadequate maintenance of the electric typewriter and the use of inferior relays in the output writer mechanism. This problem has been overcome only partially by improved maintenance of the typewriter and replacement of the relays.

8.1.8 Input Card Reader

Sporadic operation due to inadequate maintenance of the card reader. This problem has been overcome by improved maintenance of the card reader.

8.2 System Reliability - January 1962 through December 1962

In 1962 the UDOFT system performed well with the exception of one extended period early in the year. The graph of figure 104 indicates that the system was available for use 73% of the time that the system was manned, a marked improvement over the 54% level attained during the preceding twenty-month period. The graph of figure 105 depicts, on a weekly basis, the percentage of total manned system time that was available to the user. Compared with the similar graph of available time depicted in figure 101 for the preceding twenty-month period, it can be seen that available time has increased noticeably and the periods of totally excessive downtime have decreased. Except for the two periods of excessive downtime, the system was available for approximately 80% of the time. The graph of figure 106 depicts, on a weekly basis, the Operating Ratio of the computer's system. With the exception of the period early in the year, system Operating Ratio averaged over 90%.

The extended problem period was attributed to a combination of factors including trouble with the modified card read-in system, clock phasing, and inexperience of new maintenance personnel. In addition there were three rather persistent problem areas: marginal operation of the transistorized print register buffer packages, excessive drift in the analog output circuits, and intermittent problems with the output writer. The graph of figure 107 depicts the quantitative contribution of the major problem areas.

The following are explanations of the problem areas and corrections to them:

8.2.1 Clock Pulse Generation and Distribution

During this period, the clock system was modified to improve reliability and simplify adjustment. A transistorized central clock was installed which uses well isolated fixed delays to establish phase differences, single shot multivibrations to establish pulse widths, and cable drivers to drive equal length coaxial cables which distribute the clock phases to each cabinet. The only adjustment now required is the width of the pulses measured at the output of the clock repeaters in the Input-Output Unit.

8.2.2 Card Read-In

In July 1961 the card read-in system was modified to provide twelve words per card read-in. A critical parameter of this modification was a correctly timed gating pulse. To obtain this pulse the summary punch emitter contacts were used and the correct timing was obtained by displacing the rotor of the summary punch emitter on the shaft. This method proved erratic and was discarded due to the bounce of the contacts and the timing of the gating pulse. A one shot multivibrator and additional logic were added to replace the summary punch emitter in generating the gating pulse for card read-in. This method proved to be reliable.

8.2.3 Analog Outputs

The drift of the analog outputs, which is normally compensated for by the reference supplies, was found to be excessive for use with the Electric Boat Company and

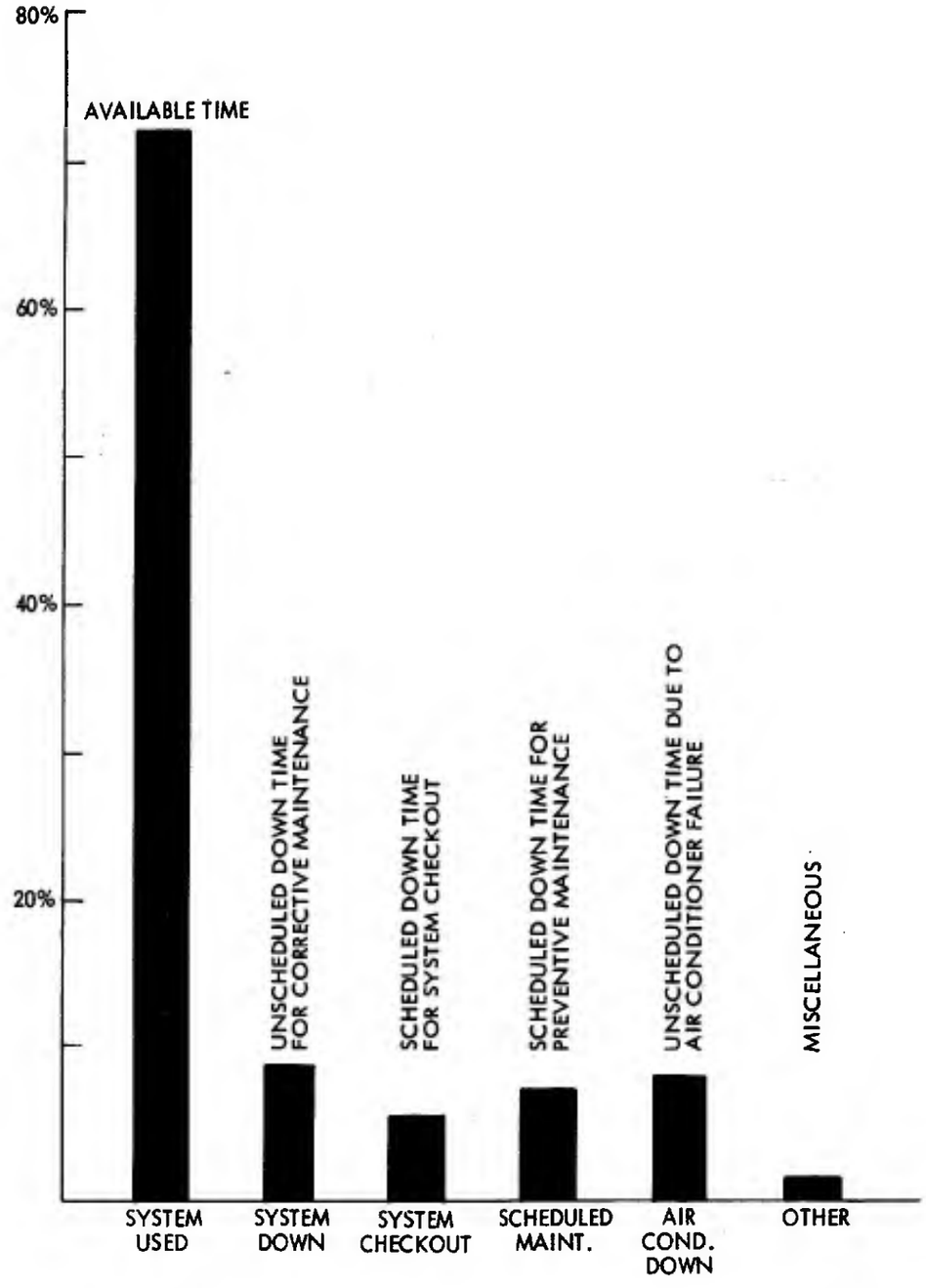


Figure 104. Total Hours 3615

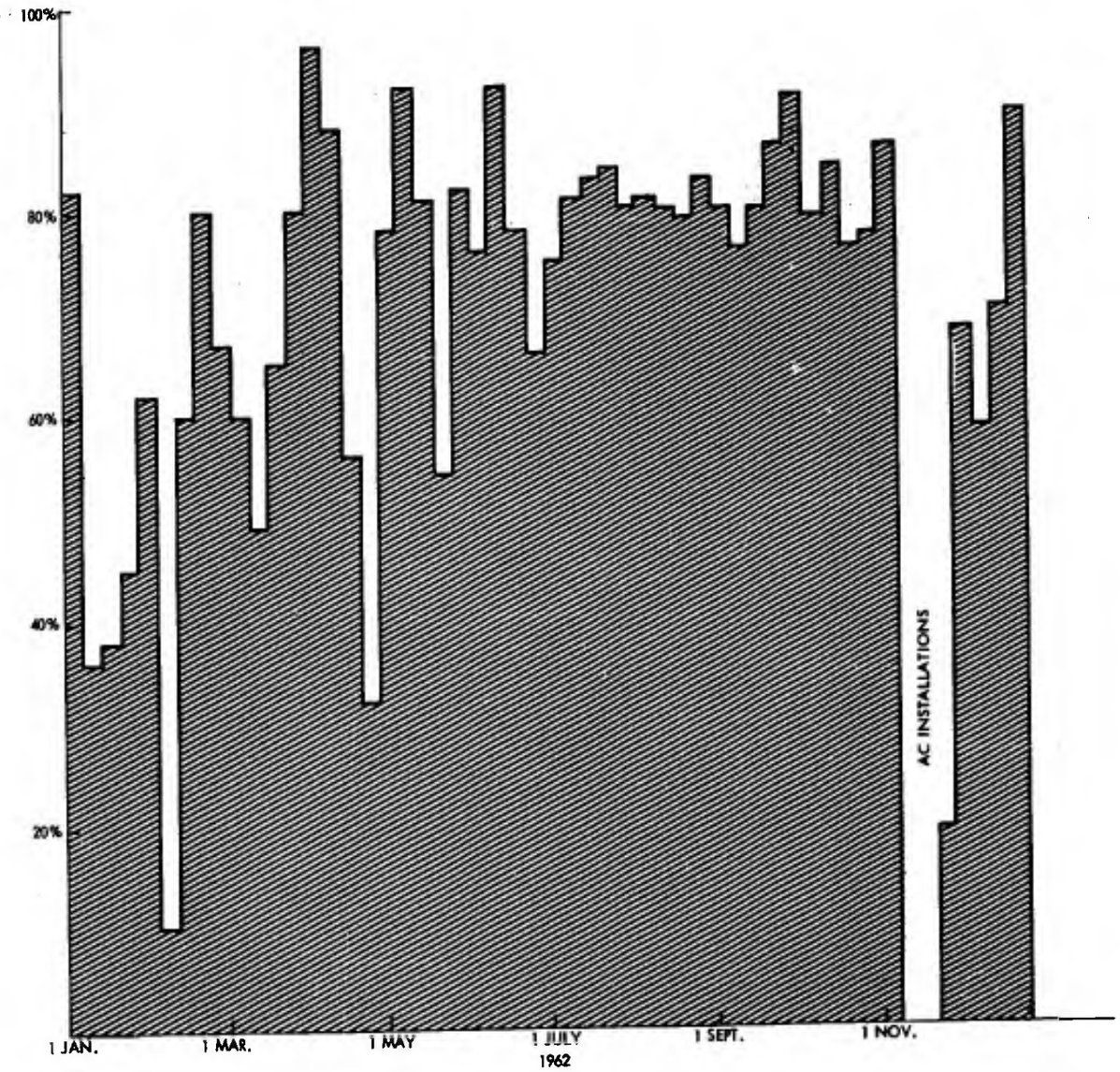


Figure 105. Available Time Versus Total Time of System Manning

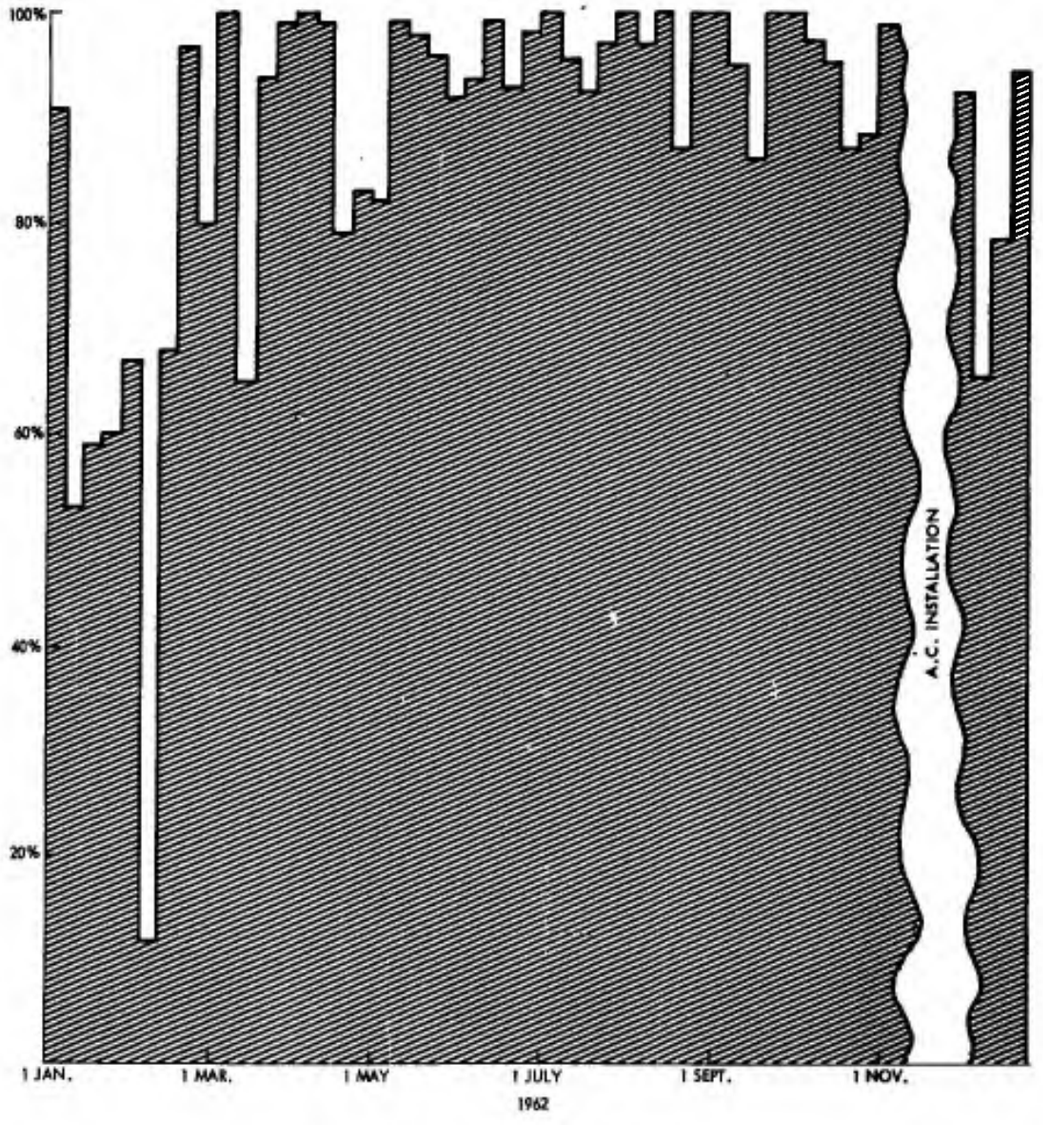


Figure 106. Operating Ratio

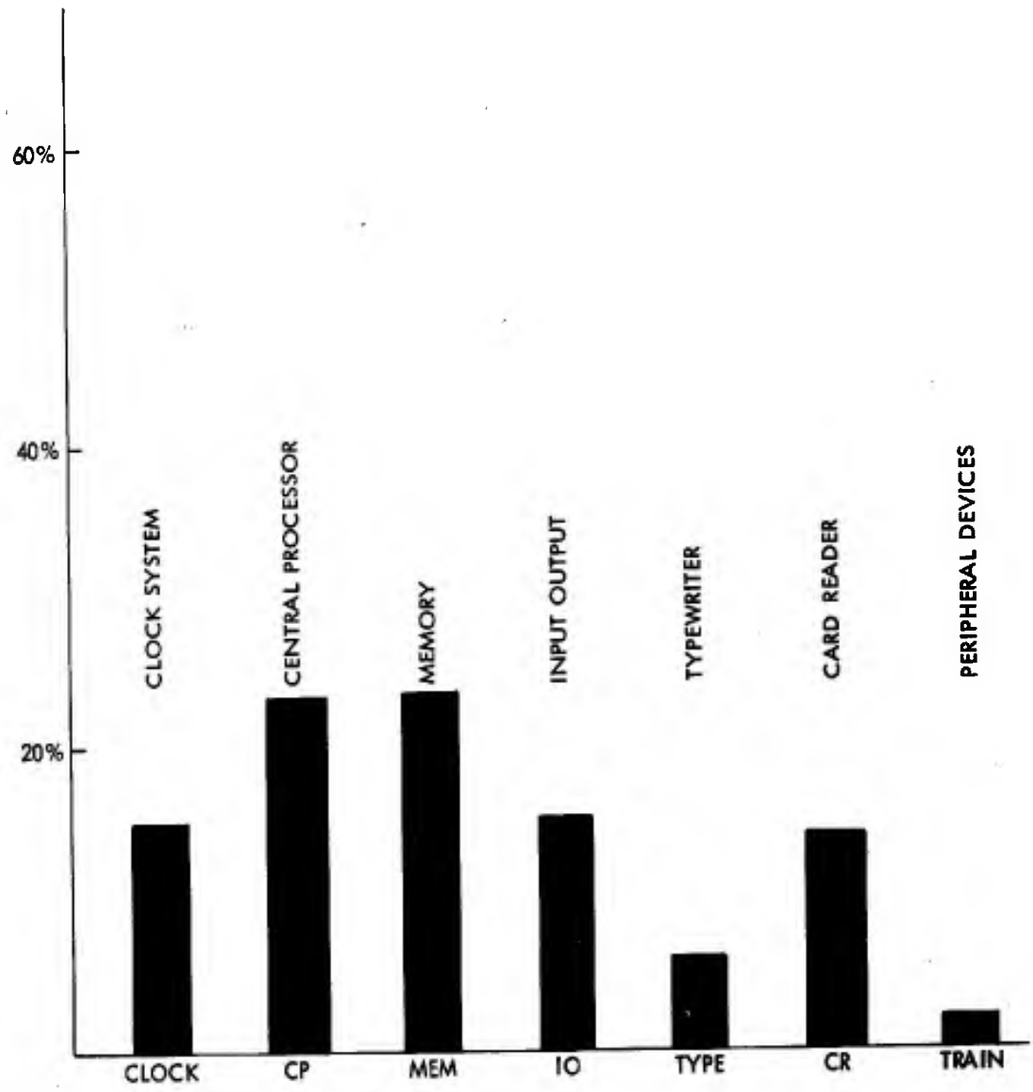


Figure 107 Cause of System Downtime

Sperry Rand Corporation equipment. The chief source of drift was found to be the output triodes of the multiplexer packages. Furthermore, it was found that all exhibited erratic drifting could often be associated with the time of day. From this fact it was concluded that small changes in filament voltages were responsible for the drift. To correct this, the filament voltage was increased by reducing the voltage drops in the filament leads by using buses; this helped considerably. Also tests were conducted on 12 of the multiplexer stages (six package locations) using a constant voltage transformer. The tests showed that these stages, independent of the package used, exhibited less drift. (The constant voltage transformers also produced a higher output voltage, 6.8V as compared to 6.3V.) In the near future all the filaments of the analog output packages will be supplied by constant voltage transformers. By using these transformers, the drift is reduced to a point where it does not present a problem to any of the users - less than 25 millivolts over an extended period.

8.2.4 Transistor Print Register Packages

Difficulty was encountered in the print register packages by the users, Electric Boat and Sperry, quite often shorting the outputs to ground and thus destroying the transistors of the output stage while checking out their equipment. In addition, the transistor print register packages displayed a certain amount of unreliability which was traced to poor grounding on the transistor packages. This was corrected by adding a jumper wire between the two sections of the ground bus on the package. Since this change, these packages have proved to be very reliable.

8.2.5 Address Flip-Flops

The memory address flip-flops performed reasonably well during this period. However, they contributed an unreasonably high percentage of downtime considering there are only twenty-four such memory address flip-flops in the computer. Because of the success of improving the analog outputs by increasing filament voltage, the filaments of the address flip-flops will be increased by using constant voltage transformers. This change should eliminate the need to choose "hot" tubes for the output cathode followers of the flip-flop. In addition, the output load may be reduced by increasing the size of the cathode resistor of the output stages.

8.2.6 Output Writer

The output writer was a problem source during the year. The typewriter itself is badly in need of overhaul. In addition, the relays of the control circuits are in poor shape due to excessive use. The control circuitry is to be replaced with a redesigned system using new components and it is hoped that time will permit an overhaul of the typewriter.

8.2.7 Conclusion

The remaining problem areas at the end of 1962 are the memory address flip-flops and the slow print system. Measures have been planned to correct both of these in the near future - as soon as the schedule permits.

Although the system has operated over ten thousand hours no trends have been noticed in component failure or in types of troubles. Therefore, it appears safe to assume that no large scale preventitive maintenance effort is needed now or in the near future. However, to insure a continued high Operating Ratio, maintenance effort must be kept up especially in the areas of maintaining good spare packages, weeding out marginal tubes in the computer, and grading tubes for the multiplexer output stage.

8.3 Conclusion

The evident conclusion drawn from the UDOFT program is that a digital computer is a highly flexible machine which when effectively programmed, can perform the task of real-time flight simulation. With another avenue of application open to the digital computer, economies in the development of simulator-trainer systems should grow. A single digital computer may be applicable to many simulator-trainer problems, thereby obviating the need for costly development of special-purpose control computers. The availability of such a machine, as an off-the-shelf item, can effect more rapid development of the whole simulator-trainer system.

These factors are important because delivery schedules for current simulator-trainer systems are very short and time is not available for sophisticated development programs. Flexibility is also important when it is realized that the development of a simulator-trainer system often is interrupted because of changes in the simulated system and training systems utilizing special purpose devices are not readily altered.

A digital simulator-trainer system has other attributes.

1. It provides a more accurate solution to the problem, particularly for small changes in the independent variables.

2. It is more reliable, due to the dependence of the computer upon the qualitative rather than the quantitative content of internal information signals.

3. It is a system which is more easily maintained because of its inherent characteristics of go no-go operation.

Flexibility of application has been demonstrated dramatically when one considers the various projects currently using the UDOFT system. Reliability of operation is demonstrated by the fact that approximately 10,000 hours of operating time have been logged on the computer without a failure of major proportions. As a result the Operating Ratio of the computer is approaching 95%, based upon average utilization of seventy-five (75) hours per week. Compared to the Operating Ratios for the better-known general purpose digital computers, 95% does not indicate outstanding performance by the computer. However, it must be remembered that the UDOFT computer is a unique device, whose operation has not been improved to any significant degree since the day it was installed.

Unfortunately, the present UDOFT system is burdened with some severe physical and logical limitations which restrict its potential application to real-time simulation of vehicular systems. These limitations include:

1. Unwieldy size and large power dissipation due to the use of vacuum tubes.

2. Lack of adequate program input-output facilities, rendering difficult the obtaining of computer-originated data required in system testing, acceptance testing, trouble-shooting, and system analysis.

3. Insufficient control instructions to utilize effectively the high-speed capabilities of the arithmetic unit.

4. An instruction repertoire plague with programming restrictions.

5. A fixed-point number word which is limited to twenty-one bits.

6. Lack of true index registers.

7. Inability to modify instruction words.

The UDOFT computer, although a product of the past and infested with restrictions that hamper its use, has been instrumental in proving the feasibility of real-time digital simulation and in the development of techniques to be applied to the design of future digital simulator-trainer systems.

APPENDIX I

GLOSSARY

1. Physical Parameters

a	Speed of sound in feet per second
b	Characteristic wing span, 36.6 feet
c	Mean aerodynamic chord
d	Distance from reference point (35%/MAC) to center of gravity in feet. Center of gravity is positive aft of reference point
g	Acceleration of gravity, 32.3 feet/sec^2 at sea level
H	Altitude above field in feet
hp	True pressure altitude in feet
hp_i	Indicated pressure altitude in feet
Δhp_i	Indicated altitude position error correction in feet
I_x	Moment of inertia about airplane X axis in slug-feet^2
I_y	Moment of inertia about airplane Y axis in slug-feet^2
I_z	Moment of inertia about airplane Z axis in slug-feet^2
ℓ_H	Distance from center of gravity to center of pressure of horizontal stabilizer in feet
Ma	Mach number
M_t	Mass of drop tanks in slugs
M_{ef}	Mass of external fuel in slugs
M_f	Mass of internal fuel in slugs
M_i	Instantaneous total mass of airplane in slugs
M_o	Empty weight of airplane in slugs
n	Normal acceleration in G's
q	Incompressible dynamic pressure, $1/2\rho V_t^2$
S	Characteristic wing area, 376 feet^2
S_H	Characteristic stabilizer area, 99 feet^2
V_t	Velocity of mass center in feet per second
W	Weight of airplane in pounds
ρ	Mass density of air in slug/feet^3

2. Angles and Angular Rates and Moments

α	Angle of attack in degrees; the angle between the X-Y wind plane and the airplane X-body axis measured in the plane of symmetry. α is positive for a nose up angle.
$\dot{\alpha}$	Rate of change of angle of attack in degrees/second

α_{WR} Rigid wing compressed angle of attack in degrees.

α_{HR} Rigid stabilizer angle of attack in degrees.

ψ Angle of yaw in degrees; the angle between wind X-Z plane and the airplane X-body axis measured in the wind X-Y plane. ψ is positive for a nose right angle.

Θ Euler angle of pitch in degrees; the angle between the horizontal plane and the airplane X-body axis measured in the wind X-Z plane. Θ is positive for a nose up angle.

q_1 Pitching rate; angular velocity about the airplane Y-body axis in radians/second.

$\dot{q}_1$ Pitching acceleration; angular acceleration about the airplane Y-body axis in radians/second^2.

Φ Euler angle of bank in degrees; the angle between the horizontal plane and the airplane Y-body axis measured in the Y-Z plane. Φ is positive for a right wing down condition.

p Rolling rate; angular velocity about the airplane X-body axis radians/second.

$\dot{p}$ Rolling acceleration; angular acceleration about the airplane X-body axis in radians/second^2.

ψ Euler angle of heading in degrees; measured in the ground X-Y plane. $\psi = 0$ for a plane flying North and $\psi = 90°$ for a plane flying East.

r Turning rate; angular velocity about the airplane Z-body axis in radians/second.

$\dot{r}$ Turning acceleration; angular acceleration about the airplane Z-body axis in radians/second^2.

δ_{SH} Control stick deflection in inches forward and aft of neutral at a radius of 22.75 inches; δ_{SH} is positive when the stick is forward.

δ_{sa} Control stick deflection in inches left and right of neutral at a radius of 22.75 inches; δ_{sa} is positive when the stick is right.

δ_{pr} Rudder pedal deflection from neutral; δ_{pr} is positive when the left pedal is forward.

δ_H Stabilizer deflection from neutral, in degrees; δ_H is positive when the leading edge is up.

δ_{at} Total aileron deflection from neutral in degrees; δ_{at} is positive when the left aileron is down.

δ_r Rudder deflection from neutral in degrees; δ_r is positive when the rudder is left.

δ_D Speed brake deflection in degrees.

δ_t Throttle position in degrees.

3. Linear Velocities, Accelerations and Forces

u Longitudinal velocity along X-body axis in feet per second; u is positive when it is in a forward direction.

$\dot{u}$	Longitudinal acceleration along X-body axis in feet per second^2.
v	Lateral velocity along Y-body axis in feet per second; v is positive when it is to the right.
$\dot{v}$	Lateral acceleration along Y-body axis in feet per second^2.
w	Normal velocity along Z-body axis in feet per second; w is positive when it is down.
$\dot{w}$	Normal acceleration along Z-body axis in feet per second^2.
V_T	Velocity along line of flight path feet per second.
X_a	Total force along X-body axis in pounds; X_a is positive when it is in a forward direction.
X_s	Total force along X-stability axis in pounds; X_s is positive when it is in a forward direction.
Y_a	Total force along Y-body axis in pounds; Y_a is positive when it is to the right.
Y_s	Total force along Y-stability axis in pounds; Y_s is positive when it is to the right.
Z_a	Total force along the Z-body axis in pounds; Z_a is positive when it is down.
Z_s	Total force along the Z-stability axis in pounds; Z_s is positive when it is down.
T	Thrust in pounds

4. Aerodynamic Coefficients and Derivatives (Stability axis)

C'_D	Basic drag coefficient
$C_{D_{dt}}$	Coefficient of drag due to drop tanks
$C_{D_{\delta t}}$	Rate of change of drag coefficient with speed brake deflection
$C_{D_{dc}}$	Coefficient of drag due to drag chute
$C_{D_{L.G}}$	Coefficient of drag due to landing gear
C_{y_ψ}	Rate of change of sideforce coefficient with yaw angle, $\frac{\partial C_y}{\partial \psi}$
$C_{y_{\delta r}}$	Rate of change of sideforce coefficient with rudder deflection, $\frac{\partial C_y}{\partial \delta r}$
$C_{y_{dt}}$	Rate of change of sideforce coefficient with yaw angle when drop tanks are attached
C_{y_p}	Rate of change of sideforce coefficient with roll velocity parameter, $\partial C_y / \partial \left[\frac{b(p + r \sin \alpha)}{2V}\right]$
C_{L_W}	Coefficient of lift due to wings
C_{L_H}	Coefficient of lift due to horizontal stabilizer
$C_{L_{\delta J}}$	Rate of change of lift coefficient with speed brake deflection, $\frac{\partial C_L}{\partial \delta J}$

$C_{L_{dt}}$ Coefficient of lift due to drop tanks

$C_{\ell_{\delta a}}$ Rate of change of rolling moment coefficient with aileron deflection, $\frac{\partial C_\ell}{\partial_{\delta a}}$

$C_{\ell_{\delta r}}$ Rate of change of rolling moment coefficient with rudder deflection, $\frac{\partial C_\ell}{\partial_{\delta r}}$

C_{ℓ_ψ} Rate of change of rolling moment coefficient with yaw angle, $\frac{\partial C_\ell}{\partial \psi}$

C_{ℓ_p} Rate of change of rolling moment coefficient with roll velocity parameter, $\partial C_\ell / \partial \left[\frac{b(p + r \sin \alpha)}{2V}\right]$

C_{ℓ_r} Rate of change of rolling moment coefficient with turning velocity parameter, $\partial C_\ell / \partial \left[\frac{b(r - p \sin \alpha)}{2V}\right]$

$C_{m_{\alpha_F}}$ Flexible pitching movement coefficient

$C_{m_{q_1}}$ Rate of change of pitching moment coefficient with pitching velocity parameter, $\partial C_m / \partial \left[\frac{C_{q_1}}{2V}\right]$

$C_{m_{\dot{\alpha}}}$ Rate of change of pitching moment coefficient with a rate of change of angle of attack, $\partial C_m / \partial \left[\frac{c \dot{\alpha}}{2V}\right]$

$C_{m_{Wa}}$ Rate of change of pitching moment coefficient with throttle position, $\partial C_m / \partial$ ✱

$C_{m_{dt}}$ Pitching moment coefficient due to drop tanks

$C_{m_{\delta_J}}$ Rate of change of pitching moment coefficient with speed brake deflection, $\partial C_m / \partial \delta_J$

$C_{m_{dc}}$ Pitching moment coefficient due to drag chute

$C_{m_{L.G}}$ Pitching moment coefficient due to landing gear

$C_{m_{G.C}}$ Rate of change of pitching moment coefficient with altitude H close to ground (ground effects)

C_{n_ψ} Rate of change of turning moment coefficient with yaw angle, $\partial C_n / \partial \psi$

$C_{n_{\delta r}}$ Rate of change of turning moment coefficient with rudder deflection, $\partial C_n / \partial_{\delta r}$

$C_{n_{\delta a}}$ Rate of change of turning moment coefficient with aileron deflection, $\partial C_n / \partial_{\delta a}$

$C_{n_{Wa}}$ Rate of change of turning moment coefficient with throttle deflection, $\partial C_n / \partial$ ✱

C_{np} Rate of change of turning moment coefficient with rolling velocity parameter, $\partial C_n / \partial \left[\frac{b(p + r \sin \alpha)}{2V}\right]$

C_{n_r} Rate of change of turning moment coefficient with turning velocity parameter, $\partial C_n / \partial \left[\frac{b(r - p \sin \alpha)}{2V}\right]$

APPENDIX II

EQUATIONS FOR VELOCITIES, FORCES AND MOMENTS RELATIVE TO AIRCRAFT AXES

This appendix supplies various data and formulaé used in the UDOFT Program. Illustrations included display the various axes, moments, vectors and other forces necessary to the mathematics involved.

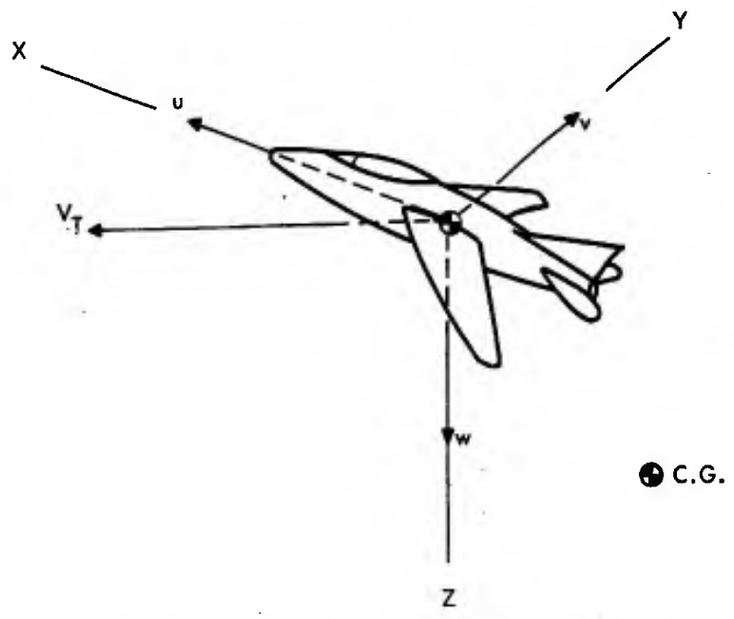


Figure II-1. Linear Velocities Along Airplane Axes

Longitudinal Velocity

$$u = V_T \cos \alpha \tag{124}$$

where

$$V_T = \int \dot{u}\, dt$$

Logitudinal Acceleration

$$\dot{u} = \frac{X_a}{M_i} - wq_1 + vr \qquad \text{(ground)} \tag{125}$$

$$\dot{u} = \frac{X_a}{M_i} - g \sin \Theta - wq_1 + vr \qquad \text{(air)} \tag{126}$$

Lateral Velocity

$$v = 0 \qquad \text{(ground)} \tag{127}$$

$$v = \int \dot{v}\, dt \qquad \text{(air)} \tag{128}$$

Lateral Acceleration

$$\dot{v} = \frac{Y_a}{M_i} + g \cos \Theta \sin \Phi - ur + wp \tag{129}$$

Normal Velocity

$$w = V_T \sin \alpha \qquad \text{(ground)} \tag{130}$$

$$w = \int \dot{w}\, dt \qquad \text{(air)} \tag{131}$$

Normal Acceleration

$$\dot{w} = -\frac{Z_a}{M_i} + g \cos \Theta \cos \Phi - vp + uq_1 \tag{132}$$

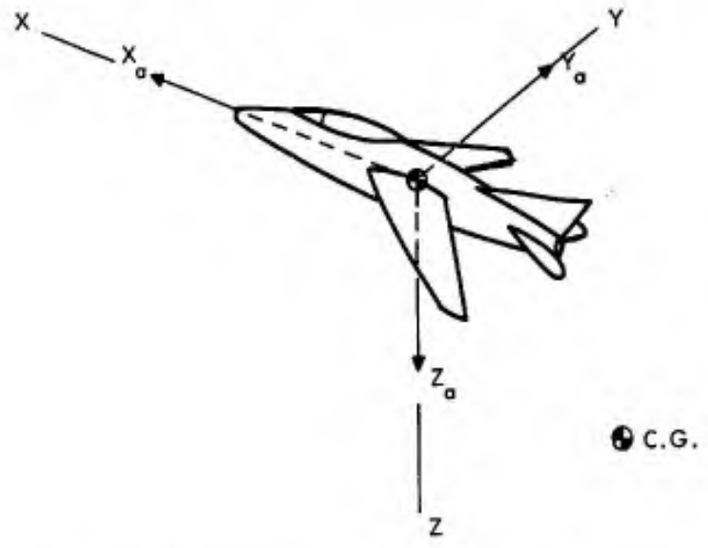


Figure II-2. Summation of Forces Along Airplane Axes

X_a = Total Force Along Airplane X Axis (+ Forward)

$$X_a = X_s \cos \alpha + T - D_{WM} - (F_{BR} - F_{BL}) - 600 \qquad \text{(ground)} \tag{133}$$

$$= X_s \cos \alpha - Z_s \sin \alpha + T - D_{WM} \qquad \text{(air)} \tag{134}$$

where

D_{WM} = Engine windmilling drag along airplane X axis

$$= 10,000\, f_{48}(Ma) f_5(hp) \tag{135}$$

Wheel Friction = 600#

Y_a = Total Force Along Airplane Y Axis (+ Right)

$$Y_a = Y_s \tag{136}$$

Z_a = Total Force Along Airplane Z Axis (+ Down)

$$Z_a = Z_s \cos \alpha + X_s \sin \alpha - 0.053T \tag{137}$$

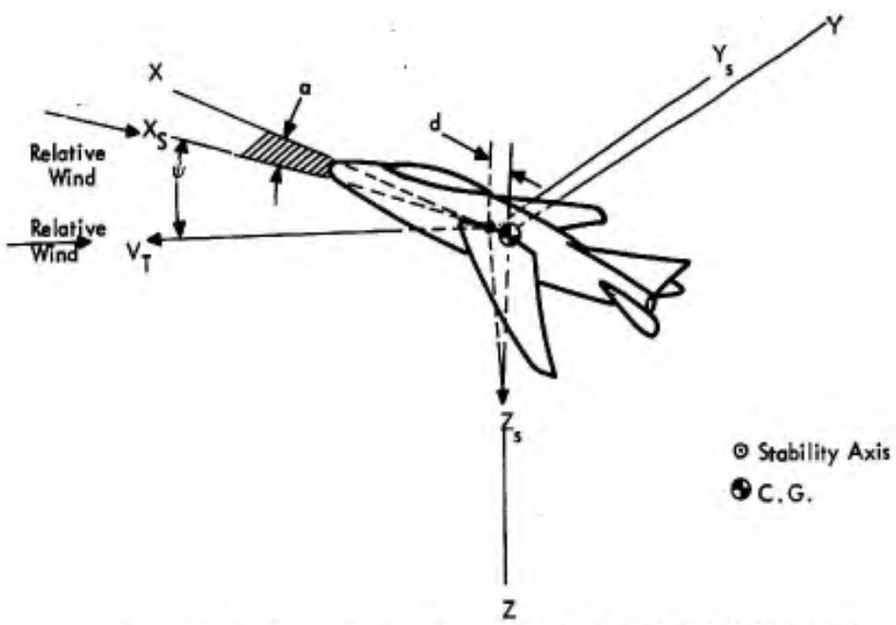


Figure II-3. Summation of Forces Along Airplane Stability Axes

X_s = Total Drag Force Along the Airplane X Stability Axis in Pounds

$$X_s = -(D_B + D_{dt} + D_{\delta_J} + D_{dc} + D_{LG}) \tag{138}$$

where

D_B = Basic drag in pounds

$D_B = 376\ qC'_D = qSC'_D$

C'_D = Basic drag coefficient

$= f_9(Ma) + C'_L f_{10}(Ma) + C'_L C'_D f_{11}(Ma) + C'^2_L f_{12}(Ma)$

where

C'_L = Clean lift coefficient formula

$= C_{L_W} + 0.263\ C_{L_H}$

(Note: see page 267 for C_{L_W} and C_{L_H})

D_{dt} = Drag due to drop tanks in pounds
$= 376\ qC_{D_{dt}} = SqC_{D_{dt}}$

$C_{D_{dt}}$ = Drop tanks drag coefficient
$= C_{D_{dt}}(Ma)$

$D_{dt} = 0$ (Drop tanks off)

D_{δ_J} = Drag due to speed brake deflection in pounds
$= 376\ qC_{D_{\delta_J}}\delta_J = SqC_{D_{\delta_J}}\delta_J$

$C_{D_{\delta_J}}$ = Speed brake drag coefficient
$= C_{D_{\delta_J}}(Ma)$

δ_J = Speed brake deflection degrees

D_{dc} = Drag due to drag chute in pounds
$= 376\ qC_{D_{dc}} = SqC_{D_{dc}}$

$C_{D_{dc}}$ = Drag force coefficient due to drag chute
$= 0.30585$

$D_{dc} = 0$ (Drag chute deflated or jettisoned)

$D_{L.G}$ = Drag due to landing gear in pounds
$= 10.2\ q = SqC_{D_{L.G}}$

$C_{D_{L.G}}$ = Drag coefficient due to landing gear
$= 0.0278$

$D_{L.G} = 0$ (Landing gear up)

Y_s = Total Side Force Along the Airplane Y Stability Axis in Pounds

$$= Y_\psi + Y_{\delta_R} + Y_{dt} + Y_p \qquad (139)$$

where

Y_ψ = Side force due to yaw angle in pounds
$= 376\ qC_{y_\psi}\Psi = SqC_{y_\psi}\Psi$ Ψ in deg.

C_{y_ψ} = Side force coefficient due to yaw angle
$= C_{y_\psi}(Ma)$

Y_{δ_R} = Side force due to rudder deflection in pounds
$= 376\ qCy_{\delta_R}\delta_R = SqCy_{\delta_R}\delta_R$ δ_R in deg.

$C_{y_{\delta_R}}$ = Side force coefficient due to rudder deflection

$= 1.238\left[0.002 - f_3(hp)\right]\left[f_{29}(Ma) - 1.00\right] + f_3(hp)$ (Mach $\leq$ 0.8)

$= f_3(hp) f_{29}(Ma)$ (Mach $\geq$ 0.8)

Y_{dt} = Side force due to drop tanks in pounds

$= 376\ qC_{y_{dt}}\Psi = SqC_{y_{dt}}\Psi$

$C_{y_{dt}}$ = Side force coefficient due to drop tanks

$= 0.002$

Y_p = Side force due to rolling rate in pounds

$= 6880\ qC_{y_p}\left(\frac{p + r \sin \alpha}{V_T}\right) = \frac{Sbq}{2V_T} C_{y_p}(p + r \sin \alpha)$

C_{y_p} = Side force coefficient due to rolling rate

$= C_{y_p}(\alpha)$

Z_s = Total Lift Force Along the Airplane Z Stability Axis in Pounds

$= L_W + L_H + L_{\delta_J} + L_{dt}$ (140)

$= \frac{L_W + L_H + L_{\delta_J} + L_{dt}}{0.0042H + 0.895}$

when

$H < 25$ feet

where

L_W = Lift due to wings in pounds

$= 376\ qC_{L_W} = SqC_{L_W}$

C_{L_W} = Coefficient of lift due to wings

$= \left[C_{L_{W_{1.1}}} - f_1(\alpha'_{WR}) f_5(Ma)\right] f_4(Ma) + C_{L_{W_0}}$

where

α'_{WR} = Rigid wing compressed angle of attack in degrees

$= \left\{\alpha - \left[C_{L_{W_{1.1}}} - f_1(\alpha'_{WR}) f_5(Ma)\right] f_4(Ma) f_1(q) f_3(Ma)\right\} f_2(Ma)$

L_H = Lift due to horizontal stabilizer in pounds

$= 376\ qC_{L_H} = SqC_{L_H}$

C_{L_H} = Coefficient of lift due to the stabilizer

$= 0.263\ f_1(\alpha_{HR}) f_8(Ma)$

where

α_{HR} = Rigid stabilizer angle of attack

$= \alpha + \delta_H - f_2(\alpha'_{WR})f_6(Ma) - [0.00233q + f_2(q)f_7(Ma)]\, C_{L_H}$

L_{δ_J} = Lift due to speed brake deflection in pounds

$= 376\ qC_{L_{\delta_J}}\delta_J = SqC_{L_{\delta_J}}\delta_J$

$C_{L_{\delta_J}}$ = Coefficient of lift due to speed brake deflection

$= C_{L_{\delta_J}}(Ma)$

L_{dt} = Lift due to drop tanks in pounds

$= 376\ qC_{L_{dt}} = SqC_{L_{dt}}$

$C_{L_{dt}}$ = Coefficient of lift due to drop tanks

$= f_1(\alpha)f_1(Ma)$

L_{dt} = 0 (Drop tanks off)

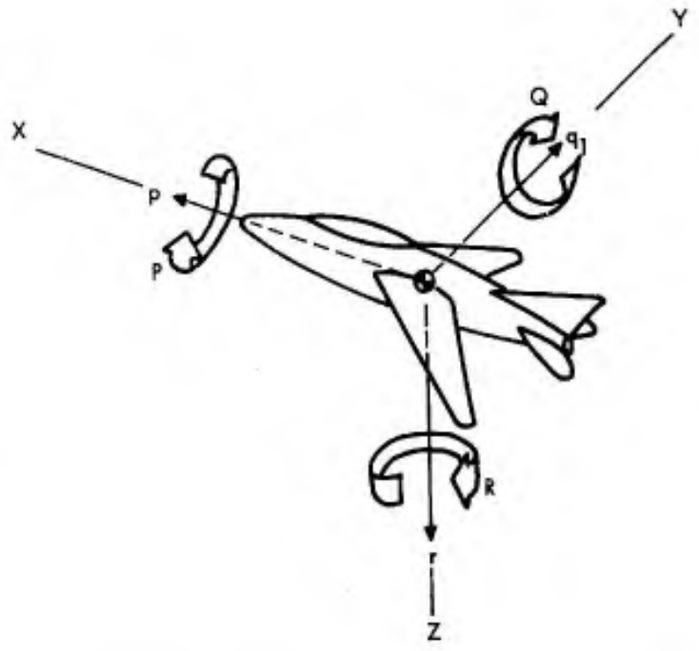


Figure II-4. Angular Velocities Along Airplane Axes

p = Rolling Rate About the Airplane X Axis in Rad/Sec

$$= \int \dot{p}\, dt \tag{141}$$

$\dot{p}$ = Rolling Acceleration About the Airplane X Axis in Rad/Sec^2

$$= 0 \qquad \text{(ground)} \tag{142}$$

$$= \frac{L_a + (I_y - I_z) q_1 r}{I_x} \qquad \text{(air)} \tag{143}$$

q_1 = Pitching Rate About the Airplane Y Axis in Rad/Sec

$$= \int \dot{q}_1\, dt \tag{144}$$

$\dot{q}_1$ = Pitching Acceleration About the Airplane Y Axis in Rad/Sec^2

$$= \frac{Ma + 54,200\, rp - 1250(g - Z_a/M_i) + \Delta M}{I_y} \qquad \text{(ground)} \tag{145}$$

$$= \frac{Ma + 54,200\, rp}{I_y} \qquad \text{(air)} \tag{146}$$

where

ΔM = Pitching moment due to nose wheel contact with the ground

$= 25,200(5 - \theta) - K\dot{\theta}$ when $\theta < 5°$

$\Delta M = 0$ when $\theta \geq 5°$

where

$K\dot{\theta}$ nose wheel damping factor

θ Pitch angle in degrees

K Experimentally determined quantity

r = Turning Rate About the Airplane Z Axis in Rad/Sec

$$= \int \dot{r}\, dt \tag{147}$$

r = Turning Rate About the Airplane Z Axis Due to Nose Wheel Steering in Rad/Sec

$$= - V_T f_{11}(\delta_{pr}) \qquad \text{when } \theta < 4° \tag{148}$$

$$r = 0 \qquad \text{when } \theta \geq 4° \tag{149}$$

$\dot{r}$ = Turning Acceleration About the Airplane Z Axis in Rad/Sec^2

$$= \frac{N_a - 41,300\, pq_1 + 6.21(F_{BR} - F_{BL}) - 1250\, ur + 420\, \dot{v}}{I_z} \qquad \text{(ground)} \tag{150}$$

$$= \frac{N_a - 41,300\, pq_1 + 420\, \dot{v}}{I_z} \qquad \text{(air)} \tag{151}$$

where

$6.21(F_{BR} - F_{BL})$ = Turning acceleration due to foot brakes

and

$420\,\dot{v}$ = Turning rate acceleration damping

= 0 (when yaw damper is off)

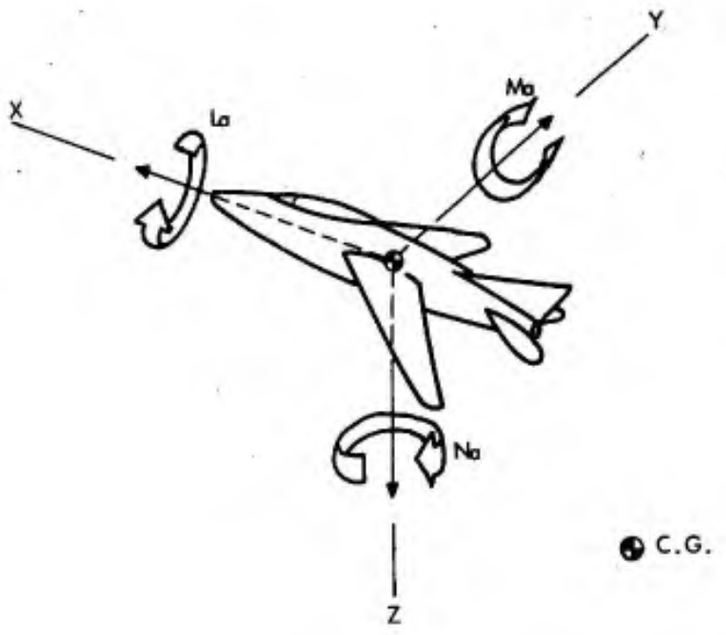


Figure II-5. Summation of Moments About Airplane Axes

L_a = Total Rolling Moment About the Airplane Z Axis in pound-feet

$$= L_s \cos\alpha - N_s \sin\alpha \tag{152}$$

Ma = Total Pitching Moment About the Airplane Y Axis in pound-feet

$$= M_s + Z_s d - 1.33\,T \tag{153}$$

N_a = Total Turning Moment About Airplane Z Axis in pound-feet

$$= N_s \cos\alpha + L_s \sin\alpha + Y_s d \tag{154}$$

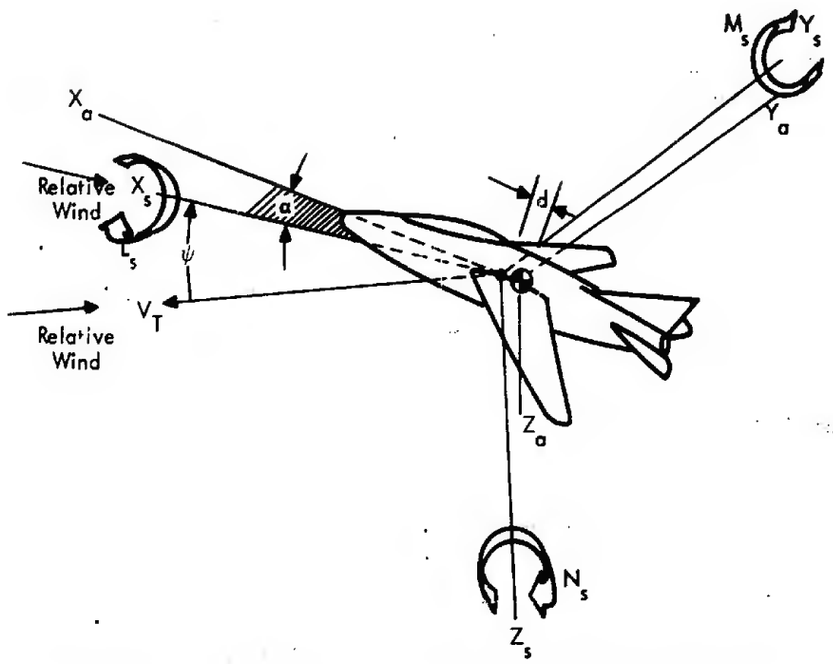


Figure II-6. Summation of Moments About Airplane Stability Axes

L_s = Total Rolling Moment About the Airplane Stability X Axis in pound-feet

$$= L_{\delta_a} + L_{\delta_r} + L_\psi + L_p + L_r \tag{155}$$

where

L_{δ_a} = Rolling moment due to aileron deflection in pound-feet

$$= 13750\, qC_{l_{\delta_a}} = SbqC_{l_{\delta_a}}$$

$C_{l_{\delta_a}} = C_{l_{\delta_a}}(Ma, \alpha, q, \delta_a)$ rolling moment coefficient as a function of Ma, α, q and δ_a

$$= \left\{ C'_{l_{\delta_a}} \left[1 - f_4(q) f_{41}(Ma) \right] - f_{46}(Ma) + f_{47}(Ma) f_6(q) \right\} f_2(\delta_a)$$

where

$$C'_{l_{\delta_a}} = f_7(\alpha) f_{45}(Ma) + f_8(\alpha) f_{40}(Ma) + 0.796 \qquad 0 \leq Ma \leq 1.1$$

$$= f_{40}(Ma) + 0.796 \qquad 1.1 \leq Ma \leq 1.73$$

L_{δ_r} = Rolling moment due to rudder deflection in pound-feet

$$= 13,750\, qC_{l_{\delta_r}} \delta r f_6(\alpha) = SbqC_{l_{\delta_r}} \delta r f_6(\alpha)$$

$C_{\ell_{\delta r}}$ = Coefficient of rolling moment due to rudder deflection

$= C_{n_{\delta r}}$

L_ψ = Rolling moment due to yaw in pound-feet

$= 13750\ qC_{\ell_\psi} = SbqC_{\ell_\psi}\Psi$

C_{ℓ_ψ} = Coefficient of rolling moment due to yaw

$= f_{35}(Ma) + \frac{hp - 62,000}{64,000} f_{36}(Ma) + f_4(\alpha)f_{37}(Ma)$

L_p = Rolling moment due to rolling rate in pound-feet

$= 252,000\ qC_{\ell_p}(\frac{p + r \sin\alpha}{V_T}) = \frac{Sb^2}{2V_T} qC_{\ell_p}(p + r \sin\alpha)$

C_{ℓ_p} = Rolling moment coefficient due to rolling rate

$= f_5(\alpha)f_{38}(Ma)\left[1 - f_3(q)f_{39}(Ma)\right]$

L_r = Rolling moment due to yawing rate in pound-feet

$= 252,000\ qC_{\ell_r}(\frac{r - p \sin\alpha}{V_T}) = \frac{Sb^2}{2V_T} qC_{\ell_r}(r - p \sin\alpha)$

C_{ℓ_r} = Rolling moment coefficient due to yawing rate

$= C_{\ell_r}(\alpha)$

M_s = Total Pitching Moment About the Airplane Stability Y Axis in pound-feet (156)

$= M_\alpha + M_{q1} + M_H + M_{\dot\alpha} + M_{W_a} + M_{dt} + M_{\delta_J} + M_{dc} + M_{L.G} + M_{G.E}$

where

M_α = Pitching moment due to angle of attack in pound-feet

$= 4260\ qC_{m_{\alpha_F}} = ScqC_{m_{\alpha_F}}$

$C_{m_{\alpha_F}}$ = Flexible pitching moment coefficient due to angle of attack

$= C_{m_{X_R}} - f_{19}(Ma)f_1(hp)f_3(\alpha'_{WR})$

where

$C_{m_{\alpha_R}}$ = Rigid pitching moment coefficient due to angle of attack

$= f_{13}(Ma)$ $\qquad 24° \geq \alpha_{WR} \geq 18°$

$= \left[f_{13}(Ma) - f_{14}(Ma)\right](\frac{\alpha'_{WR} - 12}{6}) + f_{14}(Ma)$ $\qquad 18° \geq \alpha'_{WR} \geq 12°$

$= \left[f_{14}(Ma) - f_{15}(Ma)\right](\frac{\alpha'_{WR} - 8}{4}) + f_{15}(Ma)$ $\qquad 12° \geq \alpha'_{WR} \geq 8°$

$$= \left[f_{15}(Ma) - f_{16}(Ma)\right]\left(\frac{\alpha'_{WR} - 5}{3}\right) + f_{16}(Ma) \qquad 8° \geq \alpha'_{WR} \geq 5°$$

$$= \left[f_{16}(Ma) - f_{17}(Ma)\right]\left(\frac{\alpha'_{WR} + 6}{11}\right) + f_{17}(Ma) \qquad 5° \geq \alpha'_{WR} \geq -6°$$

$$= \left[f_{17}(Ma) - f_{18}(Ma)\right]\left(\frac{\alpha'_{WR} + 18}{12}\right) + f_{18}(Ma) \qquad -6° \geq \alpha'_{WR} \geq -18°$$

$$= f_{18}(Ma) \qquad -18° \geq \alpha'_{WR} \geq -24°$$

M_{q_1} = Pitching moment due to pitching rate in pound-feet

$$= 24,133\, q \frac{C_{M_{q_1}}}{V_T} q_1 = \frac{Sc^2}{2V_T} qC_{M_{q_1}} q_1$$

$C_{M_{q_1}}$ = Pitching moment coefficient due to pitching rate

$$= f_{25}(Ma) f_2(hp) - f_{26}(Ma)$$

M_H = Pitching moment due to horizontal stabilizer in pound-feet

$$= -99\, qC_{L_H} \ell_H = -S_H qC_{L_H} \ell_H$$

C_{L_H} = Coefficient of left due to stabilizer (see page 267).

ℓ_H = Distance from c. g. to c. p. of horizontal stabilizer in feet

$$= \ell_H(Ma)$$

$M_{\dot{\alpha}}$ = Pitching moment due to rate of change of angle of attack in pound-feet

$$= 24,133\, q\left(\frac{C_{M_{\dot{\alpha}}} \dot{\alpha}}{V_T}\right) = \frac{Sc^2}{2V_T} qC_{M_{\dot{\alpha}}} \dot{\alpha}$$

$C_{M_{\dot{\alpha}}}$ = Pitching moment coefficient due to rate of change of angle of attack

$$= f_{27}(Ma) - \left(\frac{62,000 - hp}{64,000}\right) f_{28}(Ma)$$

M_{Wa} = Pitching moment due to air entering the duct in pound-feet

$$= 4260\, qC_{M_{Wa}} f_1(\delta)\alpha = ScqC_{M_{Wa}} f_1(\delta)\alpha$$

$C_{M_{Wa}}$ = Pitching moment coefficient due to air entering the inlet engine duct

$$= C_{M_{Wa}}(Ma)$$

δ = Thrust selector in degrees

M_{dt} = Pitching moment due to drop tanks in pound-feet

$= 4260\ qC_{M_{dt}} = ScqC_{M_{dt}}$ (Drop tanks ON)
(Drop tanks OFF)

$M_{dt} = O$

$C_{M_{dt}}$ = Pitching moment coefficient due to drop tanks

$= f_{24}(Ma) + f_{22}(Ma)$ $\quad 40° \geq \alpha \geq 20°$

$= \left[f_{24}(Ma) - f_{23}(Ma)\right]\left(\frac{\alpha - 10}{10}\right) + f_{23}(Ma) + f_{22}(Ma)$ $\quad 20° \geq \alpha \geq 10°$

$= f_{23}(Ma)\frac{\alpha}{10} + f_{22}(Ma)$ $\quad 10° \geq \alpha \geq 0°$

$= f_{22}(Ma)$ $\quad 0° \geq \alpha \geq -40°$

M_{δ_J} = Pitching moment due to speed brake deflection in pounds-feet

$= 4260\ qC_{M_{\delta_J}}\delta_J = ScqC_{M_{\delta_J}}\delta_J$

$C_{M_{\delta_J}}$ = Pitching moment coefficient due to speed brake deflection

$= f_{20}(Ma) + f_{21}(Ma)f_2(\alpha)$

M_{dc} = Pitching moment due to drag chute in pounds-feet

$= 115\ q\ell_p$

$M_{dc} = O$ (Drag chute jettisoned or not inflated)

ℓ_p = Drag chute moment arm in feet

$\ell_p = \ell_p(\alpha)$

$M_{L.G}$ = Pitching moment due to landing gear in pound-feet

$= 4260\ qC_{M_{L.G}} = ScqC_{M_{L.G}}$ (Landing gear down)
(Landing gear up)

$M_{L.G} = O$

C_M = Pitching moment coefficient due to landing gear

$= C_{M_{LG}}(\alpha'_{WR})$

M_{GE} = Pitching moment due to ground effects in pound-feet

$= (13.1H - 328)q$ $\quad H \leq 25$ ft.

$M_{GE} = O$ $\quad H \geq 25$ ft.

N_s = Total Turning Moment About the Airplane Z Stability Axis in pound-feet

$$= N_\psi + N_{\delta_r} + N_{\delta a} + N_{Wa} + N_p + N_r \qquad (157)$$

where

N_ψ = Turning moment due to yaw angle in pound-feet

$= 13750\, qC_{n_\psi} \Psi = sbqC_{n_\psi} \Psi$

C_{n_ψ} = Turning moment coefficient due to yaw angle

$= \alpha f_{30}(Ma) - f_{31}(Ma) - \frac{h_p + 2000}{64,000} f_{32}(Ma)$

N_{δ_r} = Turning moment due to rudder deflection in pound-feet

$= 13750\, qC_{n_{\delta_r}} \delta_r = SbqC_{n_{\delta_r}} \delta_r$

$C_{n_{\delta_r}}$ = Turning moment coefficient due to rudder deflection

$= - f_{33}(Ma) - \frac{h_p + 2000}{64,000} f_{34}(Ma)$

$N_{\delta a}$ = Turning moment due to aileron deflection in pound-feet

$= 13,750\, qC_{n_{\delta a}} = SbqC_{n_{\delta a}}$

$C_{n_{\delta a}} = C_{n_{\delta a}}(\delta a, \alpha)$ Turning moment coefficient as a function of aileron deflection and angle of attack

$= f_1(\delta a) f_3(\alpha)$

N_{Wa} = Turning moment due to inlet air momentum is pound-feet

$= 13750\, C_{n_{Wa}} f_1(\phi)\, \psi = sbqC_{n_{Wa}}$

$C_{n_{Wa}}$ = Turning moment coefficient due to inlet air momentum

$= C_{n_{Wa}}(Ma)$

N_p = Turning moment due to rolling rate is pound-feet

$= 252,000\, qC_{n_p} \left(\frac{p + r \sin\alpha}{V_T}\right) = \frac{Sb^2}{2V_T} C_{n_p} (p + r \sin\alpha)$

C_{n_p} = Turning moment coefficient due to rolling rate

$= C_{n_p}(\alpha)$

N_r = Turning moment due to turning rate in pound-feet

$= - 23,900\, q\left(\frac{r - p \sin\alpha}{V_T}\right) = - \frac{Sb^2}{2V_T} C_{n_r} (r - p \sin\alpha)$

C_{n_r} = Turning moment coefficient due to turning rate

$= 0.0948$

APPENDIX III

MASS MOMENT OF INERTIA AND LOCATION OF CENTER OF GRAVITY EQUATIONS

M_i = Total Instantaneous Mass in Slugs

$$= M_o + M_f + M_{ef} + M_{dt} \tag{158}$$

where

M_o = Weight of airplane empty in slugs

= 591

M_f = Weight of internal fuel in slugs

= 153

M_{ef} = Weight of external fuel in slugs

= 111

M_{dt} = Weight of drop tanks in slugs

= 12.4

I_x = Moment of Inertia About Airplane X Axis in Slug-Feet Squared

$= 10,200 + 100\, M_{ef}$

I_y = Moment of Inertial About Airplane Y Axis in Slug-Feet Squared

= 57,000

I_z = Moment of Inertia About Airplane Z Axis in Slug-Feet Squared

$= 60,000 + 75.0\,(M_f + M_{ef})$

d = Distance from Reference Point (35% MAC) to Center of Gravity, in Feet. (Positive if c.g. is aft of Ref. Pt.)

$$= \frac{-46.7 + 0.453\, M_{ef} - f_1(M_f) + K_1}{M_i}$$

where

K_1 = 28.4 in slug-feet, drop tanks ON

= 0 in slug-feet, drop tanks OFF

$f_1(M_f)$ = Mass of internal fuel

APPENDIX IV

DISPLACEMENT ABOUT AXES EQUATIONS

Θ = Pitch Angle with Respect to X Axis in Degrees

$$= 57.3 \int \dot{\Theta}\, dt \tag{159}$$

$\dot{\Theta}$ = Rate of Change of Pitch Angle in Radians per second

$$= q_1 \cos \Phi - r \sin \Phi$$

Φ = Bank Angle with Respect to Y Axis in Degrees

$$= 57.3 \int \dot{\Phi}\, dt$$

$\dot{\Phi}$ = Rate of Change of Bank Angle in Radians per second

$$= p + \Psi \cos \Theta$$

Ψ = Heading Angle with Respect to Z Axis in Degrees

$$= 57.3 \int \dot{\Psi}\, dt$$

$\dot{\Psi}$ = Rate of Change of Heading Angle in Radians per second

$$= \text{Sec } \Theta \; r[\cos \Phi + q_1 \sin \Phi]$$

APPENDIX V

AUXILIARY AERO EQUATIONS

Ma = Mach Number (160)

$= V_T/a$

where a = speed of sound in speed per second.

q = Dynamic Pressure in pound per foot^2

$= 1/2\ \rho V_T^2$

where ρ = Air density in slugs per foot^2.

α = Angle of Attack in degrees

$= 57.3\ w/V_T$

ψ = Yaw Angle in degrees

$= -57.3\ v/V_T = -\beta$

where β = Side slip angle in degrees.

R/C = Rate of Climb in feet /min

$= 60.0[u \sin \Theta - \cos \Theta\ (v \sin \Phi + w \cos \Phi)] = \dot{hp} = \dot{H}$ (161)

where $\dot{hp}$ = Rate of change of pressure altitude in feet/min

$\dot{H}$ = Rate of change of altitude above use field in feet/min.

n = Normal Acceleration in G's

$= -\frac{Z_a}{yM_i}$

Landing and Take off indications

Landed

$H = 0$ and $\frac{Z_a}{M_i} < 1\ G$

B = Ball Angle in degrees

$= 57.3\ V_a/Z_a$ (air)

$= -57.3\ rV_T/G$ (ground)

$f_1(V_i)$ = Indicated Airspeed (shaft position in degrees)

$= \log \frac{\Delta p}{p} + \log p - \Delta \log \frac{\Delta p}{p} + 1.93 \times 10^{-5}\ \Delta hp$ (162)

hp_i = Indicated Pressure Altitude in feet

$= hp - \Delta hp + 855\ \Delta p_o$

where

$\Delta hp = f_{42}(hp) f_4(hp)$

Δp_o = Departure from standard barometric pressure (29.92 inches of H_g)

Ma_i = Rotation of Mach Dial Counter Clockwise from the Mach = 1.0 opposite "V_i = 661 knots" Position in degrees

$= f_8(hp) - 0.00278\ \Delta hp$

where

Δh_p = Indicated altitude position error correction in feet

Ψ' = Magnetic Heading in Degrees – Position Eastward

$= \Psi + \Delta\Psi$ (163)

where

Ψ = True heading in degrees – position eastward

$\Delta\Psi$ = Magnetic variation in degrees – West variation positive, East variation negative

APPENDIX VI

AERODYNAMIC HINGE MOMENT EQUATIONS

HM_H = Stabilizer Hinge Moment in Inch-Pounds

$$= 4050\, qC_{n_H} \tag{164}$$

C_{n_H} = Stabilizer hinge moment coefficient

$$= -\left[0.0111\,\alpha f_{53}(Ma) + 0.025\,\delta_H f_{54}(Ma)\right] 935 f_8(q)$$

HM_a = Aileron Hinge Moment in Inch-Pounds

$$= 500\, qC_{ha} \tag{165}$$

C_{ha} = Aileron hinge moment coefficient

$$= 0.0186\,\alpha f_{52}(Ma) + 0.018\,|\delta a|\, f_{51}(Ma)$$

HM_r = Rudder Hinge Moment in Inch-Pounds

$$= 128\, qC_{h_r} \tag{166}$$

C_{h_r} = Rudder hinge moment coefficient

$$= \left[f_1(\psi) - 0.0225\,\delta_r\left(1 - \frac{|\phi|}{16}\right)\right]\left[f_{43}(Ma) f_7(q)\right]$$

CONTROL FORCES AND TRIM TERMS AND EQUATIONS

δ_{SH} = Stick Deflection Forward and Aft Neutral (positive when forward)

δ_{Sa} = Sticks Deflection Left and Right of Neutral (positive when right)

δ_{H} = Stabilizer Deflection from Neutral (positive with leading edge up)

δ_{aT} = Total Aileron Deflection from Neutral (positive with left aileron down)

δ_{PR} = Rudder Pedal Deflection from Neutral (positive with left pedal forward)

δ_{R} = Rudder Deflection from Neutral (positive with rudder left)

F_{H_T} = Total Stabilizer Stick Force Measured at a radius of 22.75 inches in pounds

$$= F_H + 2.85\,(n - 1) \tag{167}$$

where

n = load factor in G's

1. Stick limits from rig 10° forward to 13° aft
2. Trim limits from rig 85° forward to 29° aft
3. Trim speed 1.5° /sec of stick
4. Moment of inertia around the stabilizer stick pivot point 0.422 slug - ft^2

F_{P_T} = Total Pedal Force at a Radius of 15 inches in pounds

$$= F_P + \frac{2}{15}\,\frac{|HM_R|}{HM_R}\left[\,|HM_R| - 4330\right] \quad \text{when } HM_R \gtrless 4330 \tag{168}$$

$$= F_P \quad \text{when } HM_R \lessgtr 4330$$

$$= F_P + \frac{2}{15}\,HM_R \quad \text{(utility hydraulic system failed)}$$

1. Pedal limits from rig 12.5° forward to 12.5° aft
2. Trim limits from rig 3.8° forward to 3.8° aft
3. Trim speed 0.86° /sec of pedal
4. Moment of inertia around the rudder pedal pivot point, 0.365 slug-ft^2

δ_J = Speed Brake Deflection in Degrees

$= f_{44}(Ma) - f_5(q)$

where

δ_J maximum = 50°

APPENDIX VIII

PLOTTING BOARD TERMS AND EQUATIONS

E_I = Eastward Position of Interceptor for Plotting Board Reference Point in feet

$$= \int \dot{E}_I \, dt + E_{I_o} \tag{169}$$

where $\dot{E}_I$ = Rate of change of interceptor eastward position in feet/second

$= (u \cos \theta + w \sin \theta) \sin \Psi$

E_{I_o} = Initial position in feet

N_I = Northward Position of Interceptor for Plotting Board Reference Pound in feet

$$= \int \dot{N}_I \, dt + N_{I_o} \tag{170}$$

where $\dot{N}_I$ = Rate of change of interceptor Northward position in feet per second

$= (u \cos \theta + w \sin \theta) \cos \Psi$

N_{I_o} = Initial position in feet

UNCLASSIFIED