An American Federation of Information Processing Societies Publication

Volume 20

COMPUTERS - KEY TO TOTAL SYSTEMS CONTROL

Proceedings of the Eastern Joint Computer Conference
Washington, D.C., December 12-14, 1961

The Macmillan Company
New York

© Copyright 1961 by
American Federation of Information
Processing Societies

PRIOR CONFERENCE PROCEEDINGS

Number	Conference	Location	Date
1	Eastern	Philadelphia	Dec. 10-12, 1951
2	Eastern	New York City	Dec. 10-12, 1952
3	Western	Los Angeles	Dec. 4-6, 1953
4	Eastern	Washington	Dec. 8-10, 1953
5	Western	Los Angeles	Feb. 11-12, 1954
6	Eastern	Philadelphia	Dec. 8-10, 1954
7	Western	Los Angeles	Mar. 1-3, 1955
8	Eastern	Boston	Nov. 7-9, 1955
9	Western	San Francisco	Feb. 7-9, 1956
10	Eastern	New York City	Dec. 10-12, 1956
11	Western	Los Angeles	Feb. 26-28, 1957
12	Eastern	Washington	Dec. 9-13, 1957
13	Western	Los Angeles	May 6-8, 1958
14	Eastern	Philadelphia	Dec. 3-5, 1958
15	Western	San Francisco	Mar. 3-5, 1959
16	Eastern	Boston	Dec. 1-3, 1959
17	Western	San Francisco	May 3-5, 1960
18	Eastern	New York	Dec. 13-15, 1960
19	Western	Los Angeles	May 9-11, 1961

Manufactured in the United States of America by
McGregor & Werner, Inc., Washington 12, D. C.

PREFACE

On behalf of the Board of Governors of the American Federation of Information Processing Societies, it is my pleasure to welcome you to this conference, the first to be sponsored by the Federation rather than the National Joint Computer Committee. In May of this year, the AFIPS was created by the American Institute of Electrical Engineers, the Association for Computing Machinery, and the Institute of Radio Engineers, to be the unified national voice for the information processing and computer profession in the United States. Since then, there has been an orderly transfer of business from the NJCC to the AFIPS. As a society of societies, the AFIPS differs from the NJCC in that it can accept into membership other professional societies which are interested in information processing, and it is expected that it will grow significantly.

As stated in our constitution, the goals of AFIPS "shall be the advancement and diffusion of knowledge of the information processing sciences. . .for literary and scientific purposes. . .To this end, it is part of the purposes of the Federation. . .to serve the public by making available to journals, newspapers, and other channels of public information reliable communications as to information processing and its progress; to cooperate with local, national, and international organizations or agencies on matters pertaining to information processing; to serve as representative of the United States of America in international organizations with like interests; to promote unity and effectiveness of effort among all those who are devoting themselves to information processing by research, by application of its principles, by teaching or by study; and to foster the relations of the sciences of information processing to other sciences and to the arts and industries."

Some of these items we are well started on—others we will inaugurate soon. We represent the United States to the International Federation of Information Processing Societies and contribute financially to IFIPS in behalf of this country. We have assumed sponsorship and financial responsibility of the Joint Computer Conferences. We have accepted applications for membership from other societies. We have made our existence known to other professional societies. We are an active and growing organization acting to promote the interchange of information among information processing specialists through sponsorship of greater cooperative efforts between their professional societies. The American Federation of Information Processing Societies promises to be an instrument of tremendous utility to American technology in the exciting and dynamic years ahead.

Willis H. Ware
Chairman, Board of Governors
American Federation of Information
Processing Societies

FOREWORD

Early in the infancy of what is still a very young discipline, it was recognized that segmentation was taking place among those working in the computer sciences. Many new computing societies or specialized extensions of existing organizations sprang up, accentuating the divergence by specialization to the detriment of the industry.

It was with this thought in mind that the leaders of the Association for Computing Machinery, the Professional Group on Electronic Computers of the Institute of Radio Engineers, and the Committee on Computing Devices of the American Institute of Electrical Engineers established the Joint Computer Committee. The sole function of this Committee was to sponsor the Joint Computer Conferences, designed to provide a forum by which members of the three sponsoring societies, as well as all those interested in the computing field, could assemble and explore through the medium of technical papers, personal conversations, and technical exhibits the latest developments of interest to all.

The phenomenal growth of the field resulted in the contribution of an increasing number of excellent contributed papers. When confronted with the need to complete the conferences in a reasonable period of time, such as the traditional three days, organizers tended to resort, more and more as the years went by, to parallel technical sessions. The result has frequently been sessions whose subject matter and audience to a large extent paralleled the interests and membership of the various sponsoring societies. From this point of view the Joint Computer Conferences have not served their desired purposes, but have become a general extension of the individual meetings of the various societies.

Although tempted to expand the size of the conference by the many excellent papers contributed, either through greater duration or the establishment of parallel sessions, the Committee for the 1961 Eastern Joint Computer Conference has attempted to return to first principles, eliminate parallel sessions, and maintain the generally accepted three day length of the conference. The Committee has assembled a program which is believed to be of interest to most people in the computer field, regardless of whether their orientation is in programming, engineering, management, or other areas. The theme of this conference, "Computers—Key to Total Systems Control," lends itself particularly to this aim by its generality and its importance.

Eliminating parallel sessions and adhering to a unified technical program is particularly appropriate, too, by virtue of the fact that this 1961 Eastern Joint Computer Conference is the first conference under the sponsorship of the American Federation of Information Processing Societies. Whereas the Joint Computer Committee by its very organization was limited to the original three sponsoring societies, the charter of AFIPS provides specifically for the enlargement of its member organizations to include such other organizations interested in affiliation, whether they have a major interest in the computing field or simply peripheral interests in these activities.

Although this may be looked upon as somewhat of a noble experiment, the Committee believes that as more and more organizations become affiliated with AFIPS, the growth of multiple sessions that would result would lessen cross-fertilization—and, incidentally, tax the supply of public rooms in most hotels.

Another innovation pioneered at this meeting is the publication of these *Proceedings* in a permanent, hardbound form. It is the belief of the Committee that this form enhances the lasting value of the *Proceedings* to the registrants. It was an added attraction to the solicitation of worthwhile contributions to the technical sessions. Through arrangements with the publisher, the hardbound trade edition will provide wider distribution of these *Proceedings* far beyond the capabilities of the individual sponsoring societies. Acknowledgment is made to Mr. Robert Teitler of The Macmillan Company for his suggestions, encouragement, and solicitude in the preparation and publication of these *Proceedings*. His activities have been of considerable assistance and material benefit to the Committee.

Members of the Committee who have primary responsibility in different areas are listed elsewhere in these *Proceedings*. Space does not permit naming the many individuals who have assisted the various committee chairmen in their functions. Their creativity, diligence and attention to detail are responsible for the many arrangements necessary in preparing for a conference of this magnitude.

Jack Moshman
General Chairman

AMERICAN FEDERATION OF INFORMATION PROCESSING SOCIETIES (AFIPS)

AFIPS
P. O. Box 1196
Santa Monica, Calif.

General Chairman

Dr. Willis H. Ware
The RAND Corporation
1700 Main Street
Santa Monica, Calif.

Secretary

Miss Margaret R. Fox
National Bureau of Standards
Data Processing Systems Division
Washington 25, D. C.

Executive Committee

Mr. R. A. Imm (AIEE)
Dr. H. D. Huskey (ACM)
Dr. A. A. Cohen (IRE)
Dr. W. H. Ware, Chairman

IRE Directors

Dr. Werner Buchholz
IBM Corporation
South Road Laboratory
Poughkeepsie, New York

Dr. Arnold A. Cohen
Remington Rand Univac
Univac Park
St. Paul 16, Minnesota

Mr. Frank E. Heart
Lincoln Laboratory, Rm. B-283
P. O. Box 73
Lexington 73, Mass.

Mr. Harry T. Larson
Aeronutronic Div. of Ford Motor Co.
P. O. Box 486
Newport Beach, Calif.

AIEE Directors

Mr. R. A. Imm
IBM Corporation
Dept. 550, Bldg. 604
Rochester, Minnesota

Mr. F. S. Gardner
American Inst. of Elec. Engrs.
33 West 39th Street
New York 18, N. Y.

Mr. Claude A. R. Kagan
Western Electric Co.
Engineering Research Center
P. O. Box 900
Princeton, New Jersey

Dr. Morris Rubinoff
517 Anthwyn Road
Merion Station, Pennsylvania

ACM Directors

Mr. Walter Carlson
Engineering Dept.
E. I. duPont de Nemours & Co.
Louviers Bldg.
Wilmington 98, Delaware

Dr. Bruce Gilchrist
IBM Corporation
590 Madison Ave.
New York 22, N. Y.

Dr. Harry D. Huskey
Dept. of Mathematics
441 Corey Hall
University of California
Berkeley 4, California

Mr. J. D. Madden
System Development Corp.
2500 Colorado Ave.
Santa Monica, Calif.

1961 EASTERN JOINT COMPUTER CONFERENCE COMMITTEE

General Chairman	Dr. Jack Moshman, C-E-I-R, INC.
Vice Chairman	William L. Witzel, Computer Concepts, Inc.
Secretary	Herbert R. Koller, U. S. Patent Office
Finance	Solomon Rosenthal, Chairman Headquarters, U. S. Air Force
Public Relations	Isaac Seligsohn, Chairman IBM Federal Systems Division
Proceedings	Paul W. Howerton, Chairman Central Intelligence Agency
	Terence G. Jackson, Jr., Stanford Research Institute
Program	Bruce G. Oldfield, Chairman IBM Federal Systems Division
	George G. Heller, Assistant to the Chairman IBM Federal Systems Division
	Samuel N. Alexander, National Bureau of Standards
	Herbert S. Bright, Philco Corporation, Computer Division
	Saul I. Gass, IBM Federal Systems Division
	Herbert R. Koller, U. S. Patent Office
	Charles A. Phillips, Department of Defense
	Solomon Rosenthal, Headquarters, U. S. Air Force
	Howard E. Tompkins, National Institutes of Health

Hotel Arrangements	Henry S. Forrest, Chairman Control Data Corporation
	John W. Lacey, Control Data Corporation
	Clifford J. Leahy, Thompson Ramo-Wooldridge
Women's Activities	Ethel C. Marden, Chairman National Bureau of Standards
	Eleanor Alexander Jeanne Beiman Iby Heller Sarah Newman
Printing and Mailing	Mike Healy, Chairman System Development Corporation
Registration	John T. Harris, Chairman Remington Rand Univac
	W. B. Larson, Aeronutronic Div. of Ford Motor Co.
	Jack A. Neal, C-E-I-R, INC.
	E. R. Quady, Remington Rand Univac
Exhibits	Charles A. Phillips, Chairman Department of Defense
	W. Howard Gammon, Department of Defense
	L. David Whitelock, Department of the Navy
Exhibits Manager	John L. Whitlock Associates
Public Relations Consultants	Stavisky & Associates

LIST OF REVIEWERS

The Program Committee would like to express its deep appreciation to those listed below for their conscientious, thoughtful reviewing of the abstracts and summaries of the 242 papers submitted. Their unfailing efforts contributed significantly toward the selection of this year's EJCC program.

Mr. R. J. Arms, National Bureau of Standards
Mrs. Dorothy P. Armstrong, Bureau of the Census
Mr. James V. Batley, IBM Corp.
Mr. Wayne D. Bartlett, General Electric Co.
Mr. Noel D. Belnap, Jr., System Development Corp.
Mr. Martin A. Belsky, IBM Corp.
Mr. William Blodgett, Electronic Associates, Inc.
Mr. Robert Bosak, System Development Corp.
Mr. L. E. Brown, Aeronutronic
Dr. Edward A. Brown, IBM Corp.
Mr. James H. Burrows, Mitre Corp.
Mr. Robert Courtney, IBM Corp.
Mr. Robert P. Crago, IBM Corp.
Mr. Charles F. Crichton, C-E-I-R, INC.
Mr. J. A. Cunningham, National Bureau of Standards
Dr. Ruth M. Davis, David Taylor Model Basin
Mr. Arthur A. Ernst, National Bureau of Standards
Mr. James M. Farrar, Jr., IBM Corp.
Mr. Romeo R. Favreau, Electronic Associates, Inc.
Mr. Howard R. Fletcher, Bureau of Census
Miss Margaret R. Fox, National Bureau of Standards
Mr. R. F. Garrard, General Electric Company
Mr. Lewey O. Gilstrap, Jr., Adaptronics, Inc.
Mr. Seymour Ginsburg, System Development Corp.
Mr. Ezra Glazer, National Bureau of Standards
Mr. Geoffrey Gordon, IBM Corp.
Dr. Saul Gorn, University of Pennsylvania
Mr. Sidney Greenwald, Rabinow Engineering Company
Dr. Jerome J. Hahn, NIH
Mr. George M. Heller, Bureau of the Census
Mr. Thomas N. Hibbard, System Development Corp.
Mr. James Hill, Rabinow Engineering Company
Mr. E. W. Hogue, National Bureau of Standards
Mrs. Francis E. Holberton, David Taylor Model Basin
Dr. Grace M. Hopper, Sperry Rand Corp.
Mr. Richard A. Hornseth, Bureau of Census
Mr. Paul W. Howerton, CIA
Dr. Morton A. Hyman, IBM Corp.
Mr. Graham Jones, IBM Corp.
Mr. Horace Joseph, National Bureau of Standards
Mr. R. A. Kirsch, National Bureau of Standards
Mr. F. H. Kranz, IBM Corp.
Mr. M. R. Lackner, System Development Corp.
Mr. Chuck H. Lee, Bureau of the Census
Mr. Richard Lee, National Science Foundation
Dr. Herbert W. Leibowitz, IBM Corp.
Mr. Harry Loberman, National Bureau of Standards
Mr. J. D. Madden, System Development Corp.
Mrs. Ethel Marden, National Bureau of Standards
Dr. H. L. Mason, National Bureau of Standards
Mr. Phil W. Metzger, IBM Corp.
Mr. Robert J. Miles, IBM Corp.
Dr. A. H. Mitchell, IBM Corp.
Mrs. Betty S. Mitchell, Bureau of the Census
Miss Elsa Moser, IBM Corp.
Mr. Ralph Mullendore, Bureau of the Census
Mr. Simon Newman, Consultant

Mr. James P. Nigro, National Bureau of Standards

Mr. G. W. Petrie, IBM Corp.

Mr. J. L. Pike, National Bureau of Standards

Dr. W. T. Putney, IBM Corp.

Mr. Jack Rabinow, Rabinow Engineering Company

Mr. George M. Reitwiesner, National Bureau of Standards

Mr. Stanley B. Rosen, General Electric Company

Mr. David Rosenblatt, National Bureau of Standards

Mr. Arthur I. Rubin, Electronic Associates, Inc.

Mr. Bruce Rupp, IBM Corp.

Dr. Lindy Saline, General Electric Company

Mr. Isaac Seligsohn, IBM Corp.

Dr. Norman Shapiro, NIH

Mr. Jack E. Sherman, Lockheed

Mr. William Shooman, System Development Corp.

Mr. Robert A. Sibley, Jr., IBM Corp.

Miss Mary Elizabeth Stevens, National Bureau of Standards

Mr. Robert F. Stevens, IBM Corp.

Mr. Frank Stockmal, System Development Corp.

Mr. Jack A. Strong, C-E-I-R, INC.

Dr. J. H. Turnock, IBM Corp.

Mr. Walter D. Urban, National Bureau of Standards

Mr. Richard Van Horn, Rand Corp.

Mr. Kenneth Webb, IBM Corp.

Mr. Joseph H. Wegstein, National Bureau of Standards

Mr. Tom J. Welch, IBM Corp.

Mr. W. W. Youden, National Bureau of Standards

LIST OF EXHIBITORS

Charles W. Adams Associates, Incorporated, Bedford, Massachusetts

Aeronutronic - Division of Ford Motor Company, Newport Beach, California

American Data Machines, Incorporated, Hicksville, Long Island, New York

American Systems, Incorporated, Hawthorne, California

American Telephone & Telegraph Company - Long Lines Department, New York, N. Y.

AMP, Incorporated - Magnetics Division, Harrisburg, Pennsylvania

Ampex Computer Products Company, Culver City, California

ANelex Corporation, Boston, Massachusetts

Applied Dynamics, Incorporated, Ann Arbor, Michigan

Audio Devices, Incorporated, New York, N.Y.

Auerbach Corporation, Philadelphia, Pennsylvania

Autonetics Industrial Products - Division of North American Aviation, Incorporated, Los Angeles, California

Bendix Computer Division - The Bendix Corporation, Los Angeles, California

Boonshaft and Fuchs, Incorporated, Hatboro, Pennsylvania

Bryant Computer Products - Division of Ex-Cell-O Corporation, Walled Lake, Michigan

The Bureau of National Affairs, Incorporated, Washington, D. C.

Burroughs Corporation, Detroit, Michigan

Business Automation, Elmhurst, Illinois

California Computer Products, Incorporated, Downey, California

C-E-I-R, INC., Arlington, Virginia

C. P. Clare & Company, Chicago, Illinois

Clary Corporation, San Gabriel, California

Comcor, Incorporated, Denver, Colorado

Computer Control Company, Incorporated, Framingham, Massachusetts

Computer Systems, Incorporated, Monmouth Junction, New Jersey

Computron, Incorporated, Waltham, Massachusetts

Consolidated Electrodynamics Corporation, Pasadena, California

Control Data Corporation, Minneapolis, Minnesota

Dashew Business Machines, Incorporated, Los Angeles, California

Data Display, Incorporated, St. Paul, Minnesota

Datamation Division - F. D. Thompson Publications, Incorporated, New York, N. Y.

Datapulse, Incorporated, Inglewood, California

DI/AN Controls, Incorporated, Boston, Massachusetts

Digital Equipment Corporation, Maynard, Massachusetts

Digitronics Corporation, New York, N. Y.

Elco Corporation, Philadelphia, Pennsylvania

Electronic Associates, Incorporated, Long Branch, New Jersey

Electronic Memories, Incorporated, Los Angeles, California

Engineered Electronics Company, Santa Ana, California

Fabri-Tek, Incorporated, Amery, Wisconsin

Fairchild Semiconductor Corporation, Mountain View, California

Ferranti Electric, Incorporated, Plainview, Long Island, New York

GPS Instrument Company, Incorporated, Newton, Massachusetts

General Dynamics/Electronics - Information Technology Division, San Diego, California

General Electric Company - Defense Systems Department, Washington, D. C.

General Kinetics, Incorporated, Arlington, Virginia

The Gerber Scientific Instrument Company Hartford, Connecticut

Harvey-Wells Electronics, Incorporated, Natick, Massachusetts

Idaho Maryland Industries, Incorporated, Studio City, California

Indiana General Corporation, Valparaiso, Indiana

Industry Reports, Incorporated, Washington, D. C.

IBM Corporation, New York, New York

Invac Corporation, Natick, Massachusetts

Laboratory For Electronics, Incorporated, Computer Products Division, Boston, Massachusetts

Litton Systems, Incorporated, Beverly Hills, California

Micro Switch - Division of Minneapolis-Honeywell Regulator Company, Freeport, Illinois

Midwestern Instruments, Incorporated, Tulsa, Oklahoma

Minneapolis-Honeywell Regulator Company, EDP Division, Wellesley Hills, Massachusetts

Mnemotron Corporation, Pearl River, New York

Monroe Calculating Machine Company, Incorporated, Orange, New Jersey

The National Cash Register Company, Dayton, Ohio

Omnitronics, Incorporated, Philadelphia, Pennsylvania

Packard Bell Computer Corporation, Los Angeles, California

Philco Corporation - G & I Group, Philadelphia, Pennsylvania

Photocircuits Corporation, Glen Cove, New York

Potter Instrument Company, Incorporated, Plainview, Long Island, New York

Radio Corporation of America - EDP Division, Camden, New Jersey

Radio Corporation of America - Semiconductor & Materials Division, Somerville, New Jersey

Reeves Soundcraft Corporation, Danbury, Connecticut

Remington Rand Univac Division - Sperry Rand Corporation, New York, New York

Rese Engineering, Incorporated, Philadelphia, Pennsylvania

Rotron Manufacturing Company, Incorporated, Woodstock, New York

Royal McBee Corporation, New York, New York

Soroban Engineering, Incorporated, Melbourne, Florida

Sprague Electric Company, North Adams, Massachusetts

Sylvania Electronic Systems, Waltham, Massachusetts

Tally Register Corporation, Seattle, Washington

Tech Serv, Incorporated, College Park, Maryland

Telltype Corporation, Skokie, Illinois

Telex, Incorporated - Data Systems Division, Saint Paul, Minnesota

Texas Instruments, Incorporated, Dallas, Texas

Underwood Corporation, New York, New York

Uptime Corporation, Broomfield, Colorado

Wang Laboratories, Incorporated, Natick, Massachusetts

Washington Aluminum Company, Incorporated, Baltimore, Maryland

John Wiley & Sons, Incorporated, New York, New York

The above list was compiled as of the time this book went to press.

TABLE OF CONTENTS

MULTI-LEVEL PROGRAMMING FOR A REAL-TIME SYSTEM

A. B. Shafritz
Auerbach Corporation
Philadelphia, Pa.

A. E. Miller
Auerbach Corporation
Philadelphia, Pa.

K. Rose
Auerbach Corporation
Philadelphia, Pa.

Introduction

Recent computer literature has given considerable attention to the problem of matching high-speed data processing equipment with low-speed input-output equipment. The solution most often presented is the use of multiple input-output processing equipment, allowing for an increased computation load that keeps the central processing unit active. While input-output equipment is operating on one part of the program, the processor is not waiting, but working on some other part. If the benefits of such a configuration are to be realized most fully, an efficient multi-program system must be designed.

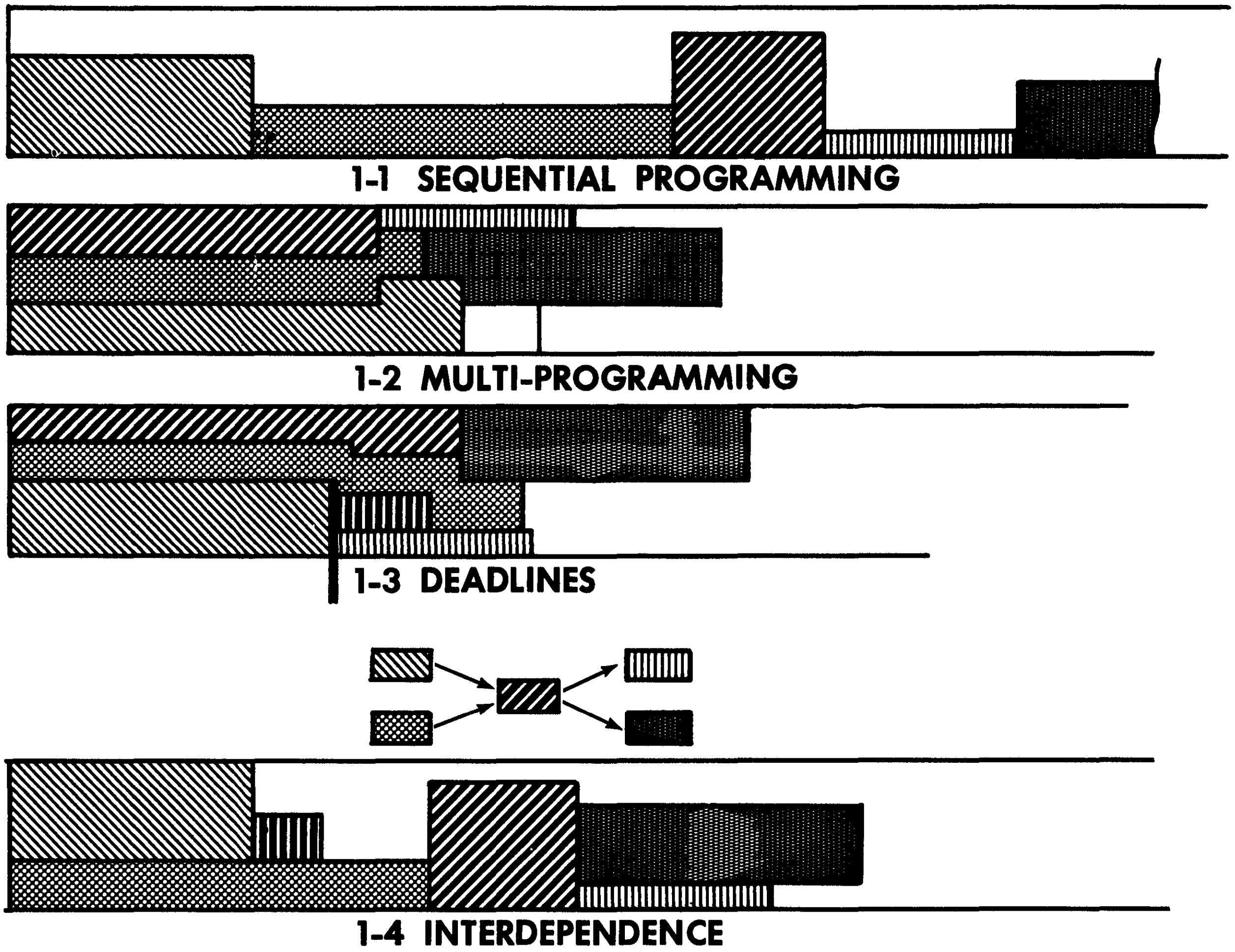
The efficiency that can be attained by multi-programming is suggested in Figure 1. The individual programs occupy only a fraction of a computer's capability, as indicated by the heights of the rectangles in the diagram. The assumption is that the central processing unit remains idle much of the time waiting for input data or access to output equipment in use. If the programs are carried out one after another, the total time (represented by lengths) is excessive (Figure 1-1, Sequential Programming).

One way to shorten this time is to give the computer some number of tasks to work on at once, and let the computer divide its time equally among them (Figure 1-2, Multi-Programming). Such a mixing technique might provide for the possibility of one task's not requiring its full fraction, and for use of this additional time by one of the other programs. When any program is finished, another is begun, so that the computer is kept busy as long as work is available. Under certain circumstances, this lack of idle time may be an adequate criterion of efficiency.

Real-time problems introduce a feature that is easy to recognize, but may be quite difficult to mechanize. If one or more programs have deadlines to meet, they might be given a greater share of the computer's time without reducing the overall efficiency (Figure 1-3, Deadlines). A collection of deadlines for many programs might easily prove incompatible, and call for compromises that take into account the severity of the penalty for transgressing each.

Another complication in multi-programming is that of interdependence of programs (Figure 1-4), Interdependence). In many important applications, the units to be scheduled are not independent programs, but parts of one major program. Any one of them may require the completion of others before it can begin. This program, in turn, may be a prerequisite for still others, in which case the efficient approach might be to carry it out promptly rather than have it share the computer's attention with other tasks.

When a multi-programming task involves both deadlines and interdependence, it becomes a challenge to the programmer. The modern computer features permitting the



1-1 SEQUENTIAL PROGRAMMING

1-2 MULTI-PROGRAMMING

1-3 DEADLINES

1-4 INTERDEPENDENCE

Figure 1. A Problem in Programming.

use of efficient multi-programming techniques do not, by themselves, solve all the problems. It is our concern in this paper to consider a programming technique which makes use of different levels of processing to take full advantage of the multi-programming capabilities of these computers. This technique has been used to varying extents on several projects by our programming staff. It is best described by tracing the development of a multi-level programming task through a particular problem. However, we will try as we go along to abstract the general features of the problem that make such an approach desirable.

Elements of a Store-And-Forward Communication System

The task at hand is to program a high-speed data processor to operate a Store-And-Forward Communication System. As Figure 2 shows, the Communication System is made up of a message switching center connected by two-way channels to a number of subscriber terminals. Messages from these subscribers are sent to the center, at will, to be transmitted to addresses at the earliest opportunity. One message may require transmission to several destinations. The memory at the center stores a message as it is received. The entire message is then retained in memory until the last transmission has been completed. Each transmission of a given message can be at a different time, and at any of a number of specified speeds. In the course of transmitting copies of a message, the computer may be called upon to translate the message from one code and format to another. The system calls for first-in first-out service, with complications that necessitate extensive processing.

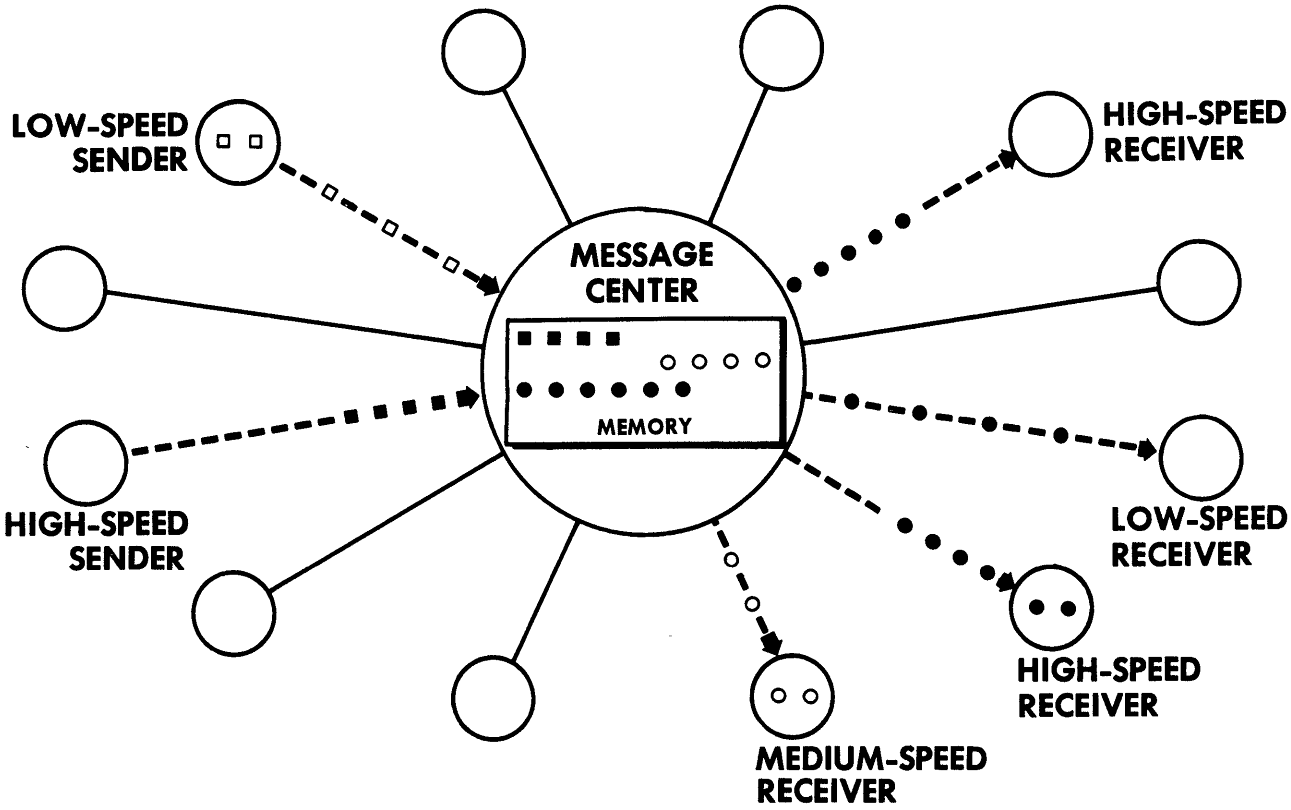


Figure 2. Store-and-Forward Communication System.

Equipment Configuration

Figure 3 shows a possible arrangement of the main equipment in the message switching center. Input buffer storage, in the form of a coincident-current memory (or a high-speed drum), is provided to receive the successive characters of each incoming message and collect them into message sections. Logically, a separate buffer is associated with each input line, the size of the buffer depending on the speed of the line. In this system it is assumed that each buffer is capable of holding at least a few seconds of traffic. The communications processor is responsible for servicing each buffer often enough so that the buffer never becomes full. If this responsibility is met, the total transmission rate is limited only by the line capacities, and the processor service is effectively continuous. If the processor under certain peak traffic conditions cannot get around to a buffer fast enough, the penalty paid is a momentary forced traffic slowdown that holds up the subscriber.

The output is similarly buffered, with the computer unloading message sections to be transmitted at the receiver's speed. The computer's responsibility is to keep output buffers from "running dry" as long as there are messages for the associated lines. The buffer status indicators, controlled by the two sets of buffers, indicate to the processor the contents of each portion of the buffers.

The processor has its own high-speed, random-access memory, and for the purposes of this paper the entire operational program and associated bookkeeping tables are assumed to be stored therein. The processor also operates several other devices that provide bulk storage with a more limited access. A set of drums constitutes the main message store. (This function might be accomplished with discs or other media, but it will be convenient to use "drum" as a short name for bulk storage in the remainder of this paper.) Magnetic tapes back up the drum with auxiliary storage space to handle heavy message backlogs which might occur at peak periods, or when one or more receiving

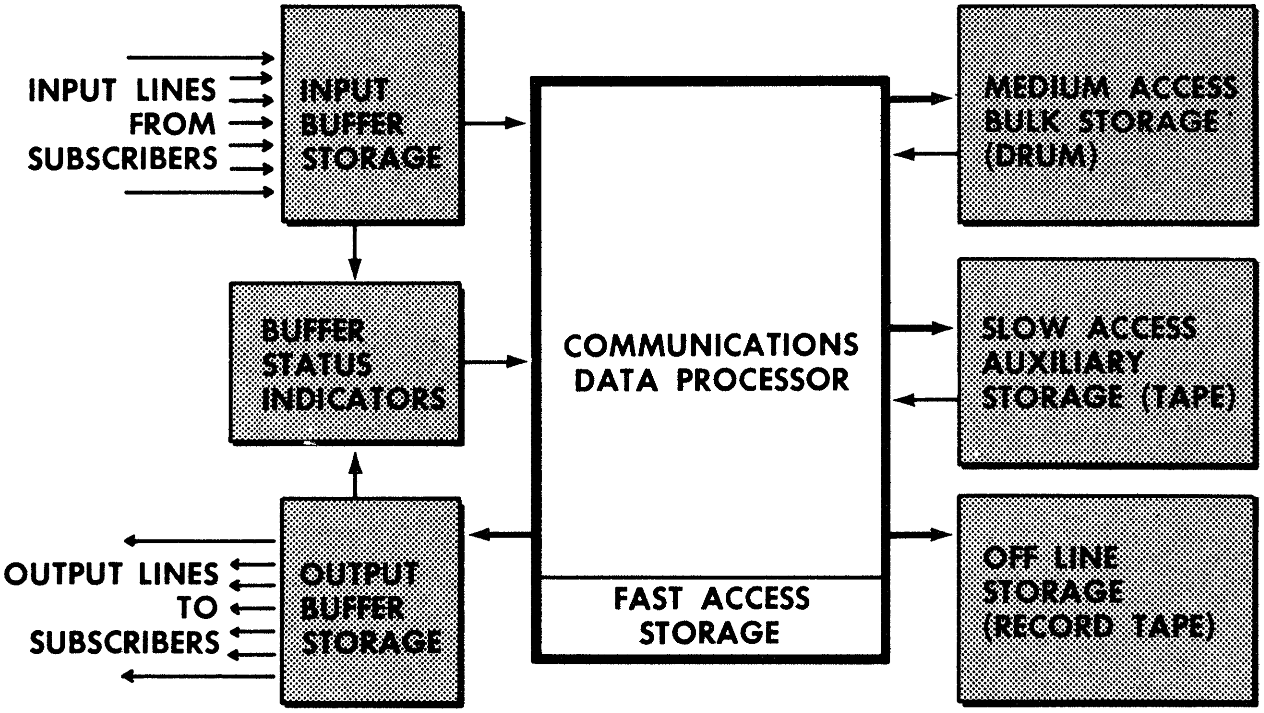

INPUT LINES FROM SUBSCRIBERS
INPUT BUFFER STORAGE
BUFFER STATUS INDICATORS
OUTPUT LINES TO SUBSCRIBERS
OUTPUT BUFFER STORAGE
COMMUNICATIONS DATA PROCESSOR
FAST ACCESS STORAGE
MEDIUM ACCESS BULK STORAGE (DRUM)
SLOW ACCESS AUXILIARY STORAGE (TAPE)
OFF LINE STORAGE (RECORD TAPE)

Figure 3. Message Switching Center.

stations are closed down. One tape unit is used for "off-line storage" that is not retrieved in the normal on-line operation. This is the record tape, on which a copy of every message section received is written, with enough other data to give a complete record of the traffic through the system.

The drums, tapes, buffers, and buffer status indicators are all treated as asynchronous peripheral devices by the processor. The processor gives them instructions to transfer information into or out of its high-speed memory, and then goes about its own business while the peripheral devices are operating. Several peripheral devices may operate simultaneously, along with the processor's central processing unit. It is the coordination of these asynchronous operations, together with the interweaving of the processing, that constitutes the problem under consideration. It is assumed that the processor has the ability to interrupt the main program upon the termination of an input-output operation. Such a capability is available, to varying degrees of sophistication, in most of the latest large-scale computers.

Interrupt Feature

Figure 4 illustrates the use of the interrupt feature and also suggests the concept of different programming levels. In the example presented, the main program performs data processing or logical operations in preparation for the operation of the first peripheral device, and then transfers control to the program level labeled "peripheral device control" to initiate the operation of the device. After the initiation, the main program continues with other tasks. During the operation of the peripheral device, brief periods of main frame time may be lost when words are transferred between the computer memory and the peripheral device. At the termination of the peripheral device operation, an interrupt transfers the program back to the peripheral device control level to perform any procedure associated with the termination of the operation (such as error checks or the storing of the contents of certain registers). Following the termination process, it is determined that a second peripheral device should be operated, and a similar sequence of actions is carried out for the second peripheral device operation. The concept of processing levels is developed more fully in the latter part of this paper.

Computation Cycle

In this, as in most real-time systems, there are a number of basic functions that

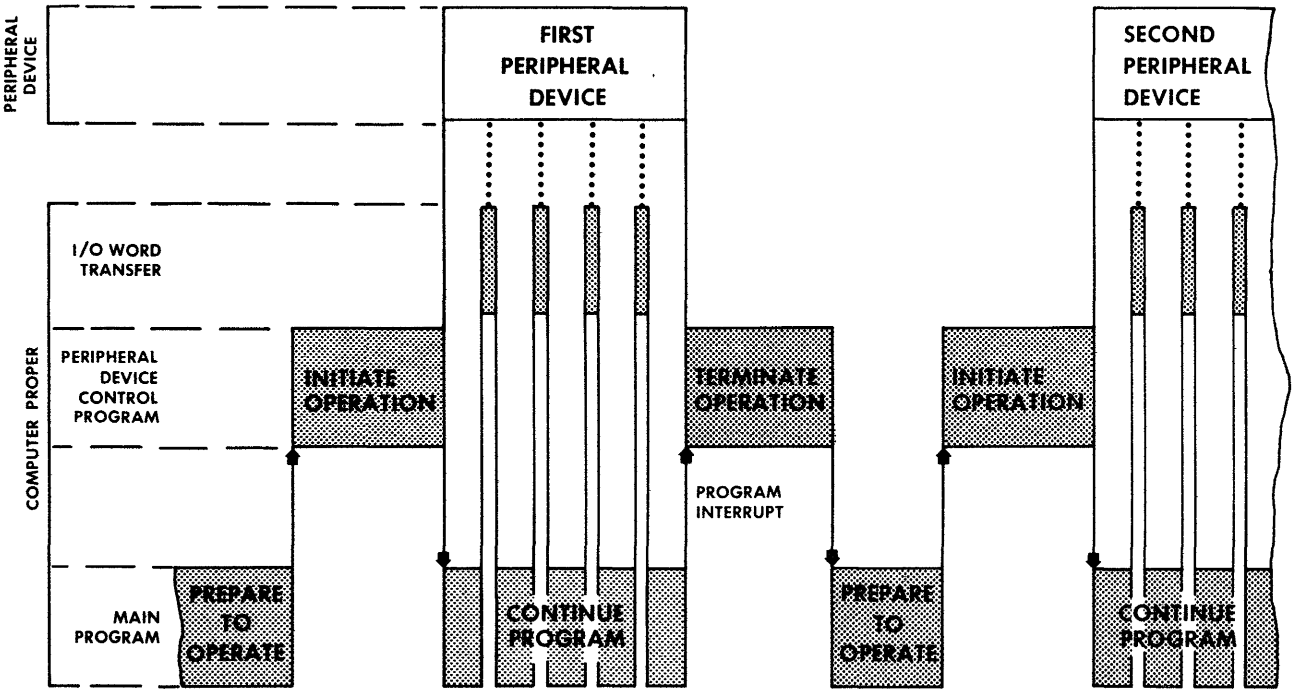


Figure 4. Interrupt System.

must be executed periodically. Ideally, each function should be carried out at its own best frequency. But to assign the lengths of periods independently would greatly complicate program control as well as entail the risk of irregular computer loading. At one time many functions might be demanding computer time simultaneously, while at another time the computer might be forced to remain idle for some period when all operations were out of phase. This could occur even under heavy traffic loads, and reduce the effective throughput of the system.

The compromise that is usually made is to establish a computation cycle, with a frequency that is close to that of as many of the basic functions as possible. Those that require more frequent execution may be done more than once in a cycle, and those whose periods should be longer may be programmed to occur in one cycle and then skip one or several cycles. If there are enough of these unusual periods, a complex of interrelated cycles may be employed.

Our cycle is concerned with two types of processing; one associated with input and the other with output. Briefly, the input processing consists of transferring message sections from the input buffer storage (and perhaps from tape) to the computer high-speed memory, storing the message sections on the drums and record tape, and performing associated bookkeeping operations. Conversely, the output processing consists of retrieving message sections from the drums, processing these, and writing them into output buffer storage and perhaps onto tapes.

Batch Size

It is important to estimate the size of the character batch, both in and out, to be handled in a cycle. The capacity of the input-output buffers does not in itself fix the cycle. It is true that every buffer should be serviced within the number of seconds for which it can hold traffic, but to serve all at once would call for an excessively large capacity in high-speed memory. At the other extreme, a small internal memory would tax the peripheral devices and the computer proper. For the drums, increasing the batch size helps the latency problem; for tapes, larger records may be written, cutting down on the start-stop wastage; and for the computer, fewer tests, setups, and general bookkeeping operations have to be performed.

In seeking the optimum batch size, the programmer must examine the system requirements, and in so doing he becomes confronted with the complication of "worst-case" design. The concept of an "average" traffic flow for the system would be meaningless. The message switching center must operate efficiently at "peak" loading, and must therefore be over-designed for average traffic conditions. The design should be such that correct operation can be maintained under the worst conditions that are within the realm of possibility. The processor must not break down even if all inputs lines attempt to send messages at their maximum rates simultaneously. It is not economical, however, to base the whole design on such remote possibilities. Statistics should be gleaned on the overall picture of traffic flow, to provide a description of a "reasonable worst-case." This should be chosen as a level that will be exceeded so seldom or so briefly that some inconvenience can be tolerated when this occurs.

From the "reasonable worst-case" conditions, it should be estimated how much processing time is required for each character going through the system. This is divided into two categories, peripheral-device time and computer time, and both are functions of batch size. Figure 5 shows these functions very much oversimplified. They are not necessarily monotonic nor even continuous; they depend on the equipment complex, techniques used, and many other factors. A batch size somewhere near where the two curves cross will produce an efficient and economical cycle. It is assumed that the equipment complex is well-suited to do the job at hand, and the batch size so chosen will afford an adequate excess capacity.

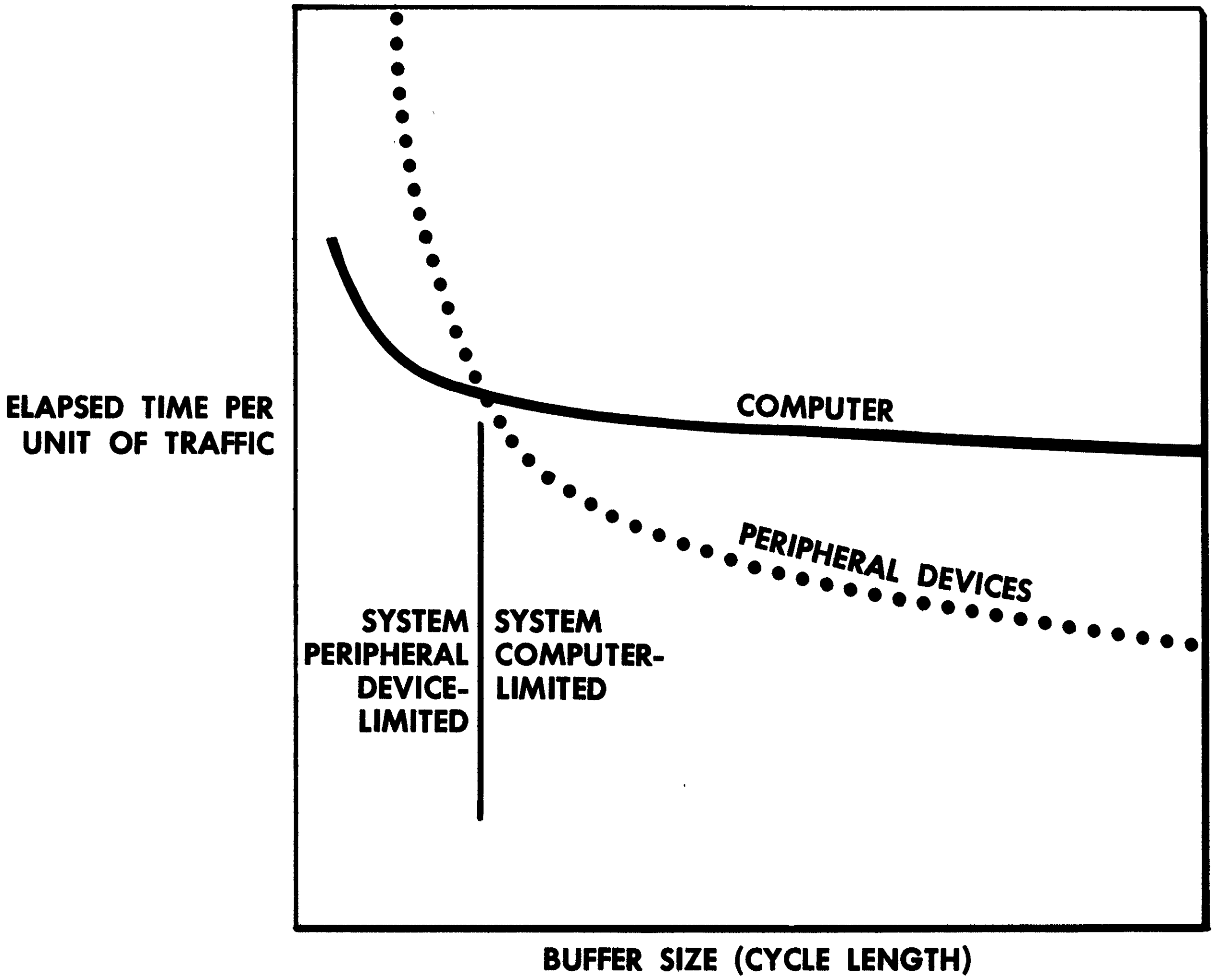


Figure 5. Cycle Length Determination.

Tasks to be Performed

With a size chosen for the batch, the actual work of programming begins. A list is made of all the tasks that are normally performed in an input-output cycle. It is helpful to have estimates of the time they will require, and this involves three parts: the computer time to prepare the data, the time for the peripheral device to operate, and the computer time to terminate the task. These time estimates need not be well defined; the programmer will want to know general orders of magnitude, or in some cases just comparisons, such as that a tape operation will not take as long as a drum operation.

The tasks for this program, as shown in Figure 6, are 11 per cycle (not necessarily in order) as follows:

1. Read Buffer Status Indicators (B.S.I.) to determine the contents of each input and output buffer.
2. Read Input Buffers. The computer portion of this task includes examination of the results of Read B.S.I. to select the input buffers to be serviced.
3. Input Processing. This operates on the message sections transferred by (2) and involves the updating of tables in the system that indicate when messages arrived, what their destinations are, and the like. It is set apart from the processing required to prepare to write drum.
4. Write Drum. This task is kept to a minimum of what must be done between the time the data arrives in high-speed memory and the time the drum write order begins. While the writing occurs, the information remains in high-speed memory and can still be accessed, although it must not be changed while the asynchronous writing is under way. Any processing that can meet the requirements is made a part of input processing rather than a part of the write drum operation.
5. Read Drum. This task calls for decisions of what to read, based on buffer status information associated with the output buffers. And again, any of this processing that can be, is relegated to output processing.
6. Output Processing. This involves selection of new messages for output lines, minor message format changes, and bookkeeping procedures. It includes any operation on output data that is not directly associated with the operation of a peripheral device.
7. Write Output Buffers. This does not itself involve much computer processing; the other tasks preceding it complete most of the decisions on what to write.
8. Write Tapes.
9. Read Tapes. These two tasks are normal and may occur in any cycle, but will probably not go on so regularly as (1) through (7). Part of the terminal processing required after reading tapes is the decision what to do with the data read. As tape is used in this system, messages from tape may go (via the computer memory) to the drum or onto another tape.
10. Write Record Tape. This is the longest peripheral operation in our system: the recording of every piece of an incoming message with additional data to identify sources. (From cycle to cycle, the pieces of a message may become separated on record tape.) While writing on record tape is being done, other tasks using the same data can be performed; thus this task may go on throughout the cycle.
11. Miscellaneous Tasks. This list of 11 tasks is greatly oversimplified, omitting even many regular tasks actually required in a message system. Tasks (1) through (10) can be thought of as including all those operations that must be totally carried out each cycle. "Miscellaneous tasks" can be thought of as those functions taken out of the main stream and made to extend over a number of cycles to regulate the cycle length. An example of such a function is message translation on a character-by-character basis. This can be a time-consuming process, but it will be required for relatively few messages. When it is necessary, it may delay the individual message involved, but others in the batch should be processed promptly. For this reason, message translation is not made a part of output processing. Miscellaneous tasks would also include non-periodic operations performed by the computer on request, or tasks such as recording of statistics,

which might be hourly or daily rather than once per cycle. Miscellaneous tasks may involve operating peripheral devices, possible even some that are not included in normal operations.

Interrelationship Between Tasks

In a fixed inflexible program, these tasks might be arranged in arbitrary order to constitute the program. In multi-level programming, the order of their execution can vary from one cycle to the next. But general limits must be established. For this purpose, it is required to determine which tasks are prerequisite to each other. Each task must wait until the completion of others, either in the same cycle or in the previous one, as shown in Figure 7. The eight input-output functions are the present concern; the three processing functions are treated later.

The reading of buffer status indicators (B.S.I.) must take place after the write output buffer operation in order that the indications be up to date. In general, prerequisites of a write order are the tasks that provide data to be written. A read order must wait until the completion of the write orders that clear out what was previously read. It should be noted that this program assigns a fixed buffer space to each input device; data are rearranged (if required) within an input buffer area and transferred from there to the output divices by means of the write instructions.

The establishment of prerequisites leads to a re-ordering of the list of tasks into a cycle. The sequencing may involve some "cut-and-try" work, but can be guided by a few well-defined principles:

1. Two operations with the same prerequisites (e.g., read drums, read tapes) should be in juxtaposition in the program. The one that serves as a prerequisite for the greater number of subsequent operations should be first.
2. When the prerequisites for one operation are a subset of those for another, the former should precede the latter (as write output buffers, write tapes).

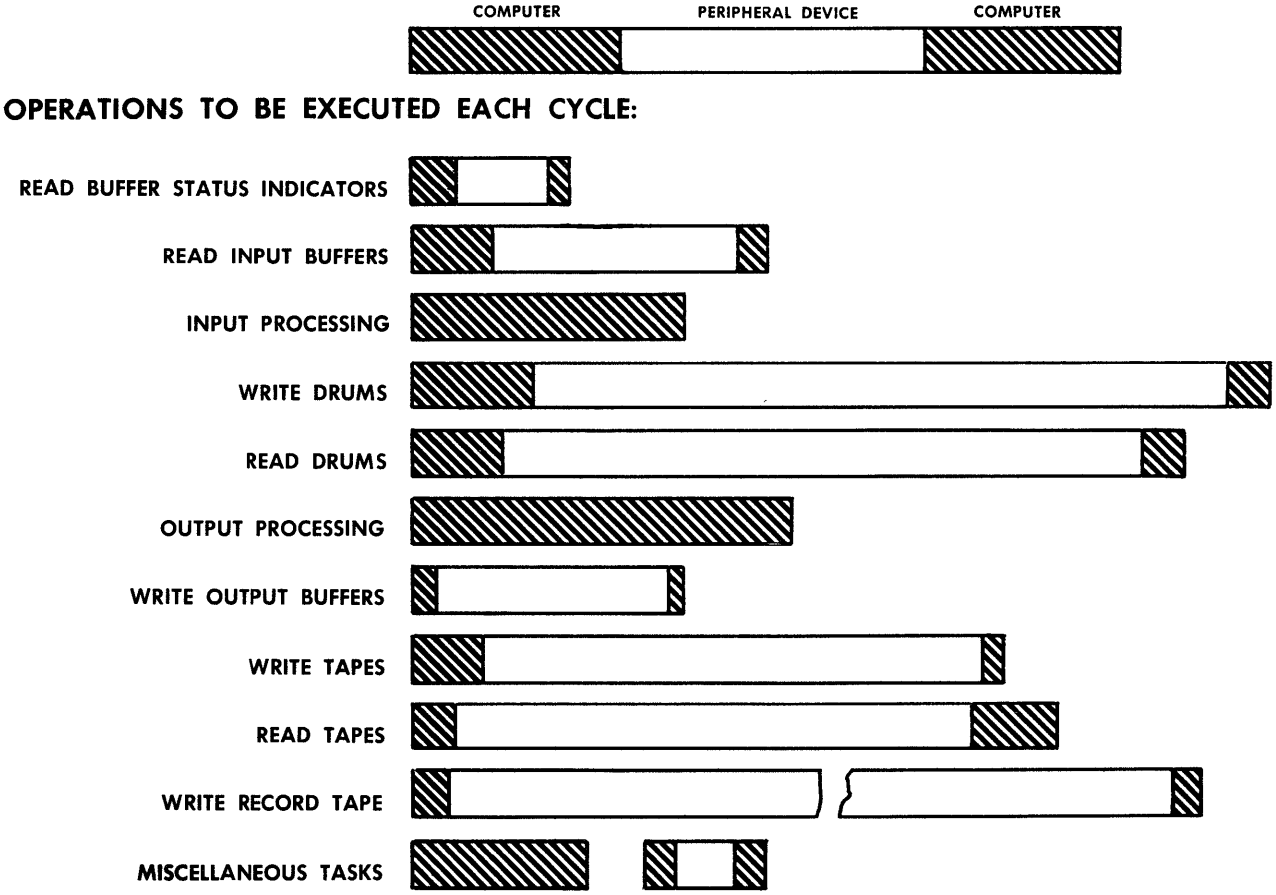


Figure 6. List of Tasks.

TASK	PREREQUISITES
READ BUFFER STATUS INDICATORS (B.S.I.)	WRITE OUTPUT BUFFERS
READ DRUMS	WRITE TAPES, WRITE DRUMS, READ B.S.I.
READ TAPES	WRITE TAPES, WRITE DRUMS, READ B.S.I.
READ INPUT BUFFERS	WRITE TAPES, WRITE DRUMS, READ B.S.I. AND WRITE RECORD TAPE
WRITE RECORD TAPE	READ INPUT BUFFERS
WRITE OUTPUT BUFFERS	READ DRUMS
WRITE TAPES	READ DRUMS, READ TAPES
WRITE DRUMS	READ DRUMS, READ TAPES, READ INPUT BUFFERS

Figure 7. Ordering by Prerequisites.

3. The circularity of the cycle should be kept in mind, and its "beginning" left flexible. In this list, Read B.S.I. is an obvious beginning, being prerequisite (by transitivity) to all other operations. But Read B.S.I. does not necessarily have to wait for the end of the previous cycle. In other words, one cycle can begin before another has ended.

In the problem under consideration, the cycle determined according to these principles divides itself into a read portion and a write portion. The central processor "inhales" information from all input sources, and after proper processing, it "exhales" into all output destinations. This gives the opportunity for information to be transferred from any peripheral device to any other during one cycle. The "read" and "write" portions of the cycle are not to be confused with system input and output. From the viewpoint of the central processor, all other parts of the system are external. Reading an input message from its buffer is the same as reading a message from the drum when that message is on its way out of the system. The input and output sub-cycles are fairly simple when considered separately.

Output Sub-Cycle

As Figure 8 shows, the output operation has three parts that are mutually prerequisite and thus determine a cycle. Each of the three parts is done by a peripheral device. When each operation is finished, a program interrupt occurs, and the computer prepares for the next step. For simplicity in this and following diagrams, some steps of the program have been removed. The computer operations required for termination of any peripheral device operation are assumed to take place as soon as an interrupt occurs. Then the preparation for the next step begins. This is shown in one block on the inner ring, which represents operation of the computer proper. The cycle is expedited by keeping these intra-peripheral operations to a minimum.

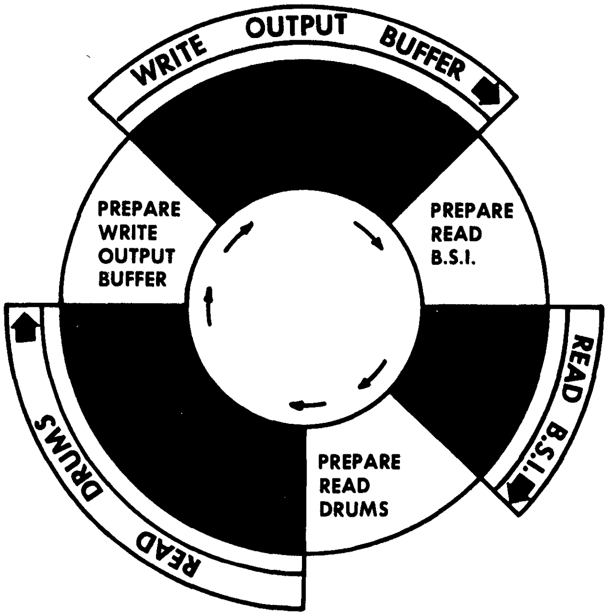


Figure 8. Output Sub-Cycle.

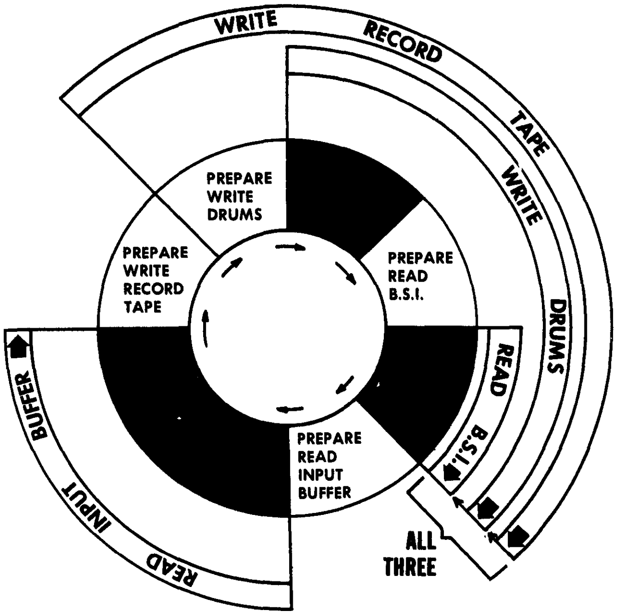


Figure 9. Input Sub-Cycle

It should be noted that the cycle (clockwise direction) is not fixed as to time. It proceeds as fast as the peripheral devices will allow, and may vary greatly in length. This is based on the simplified assumption that there is ample time to carry out all the required processing tasks during the peripheral device operations.

The black areas on Figure 8 represents time that the computer can spend on all its other functions. Some of these operations are scheduled into a larger cycle, as shown on subsequent figures. Others occur at variable times in the cycle, according to the time the processor has to work on them.

Input Sub-Cycle

Figure 9 shows the somewhat more complex input sub-cycle. Again, each interrupt is followed by termination operations and then the preparations indicated on the diagram. The prerequisities for reading the input buffer include the completion of three operations; write record tape, write drums, and read B.S.I. Generally these operations will not finish at the same time. The last of the three termination interrupts to occur initiates reading of the input buffers. When the previous operations are completed (whichever two they may be) the computer may perform the required terminal operations individually, but it will not go on to prepare read input buffer until all three are performed.

Input-Output Cycle

In Figure 10 the input and output functions are shown merged into one cycle. The computer steps required for one sub-cycle occupy some of the empty intervals in the other. There are still spaces left for input processing, output processing, and any other operations that use the computer but not peripheral devices. These are not scheduled at fixed times in the cycle, but executed as time permits, as will be explained later.

The time required by each peripheral device is so variable that these empty spaces may change size or even be eliminated. If a peripheral device operation is short or nonexistent, the interrupt will be early and the computer will not have to wait for it. There may even be a line-up of interrupts waiting for attention. The machine and program are designed to handle such situations.

Figure 10 has features added beyond the combination of the previous figures. The tape operations and preparations are shown, and an end of cycle checkpoint is included just prior to the read drums operation. The latter is a necessary safety feature in our variable cycle. It affords an interlock to insure that all the input processing, output processing, and scheduled miscellaneous

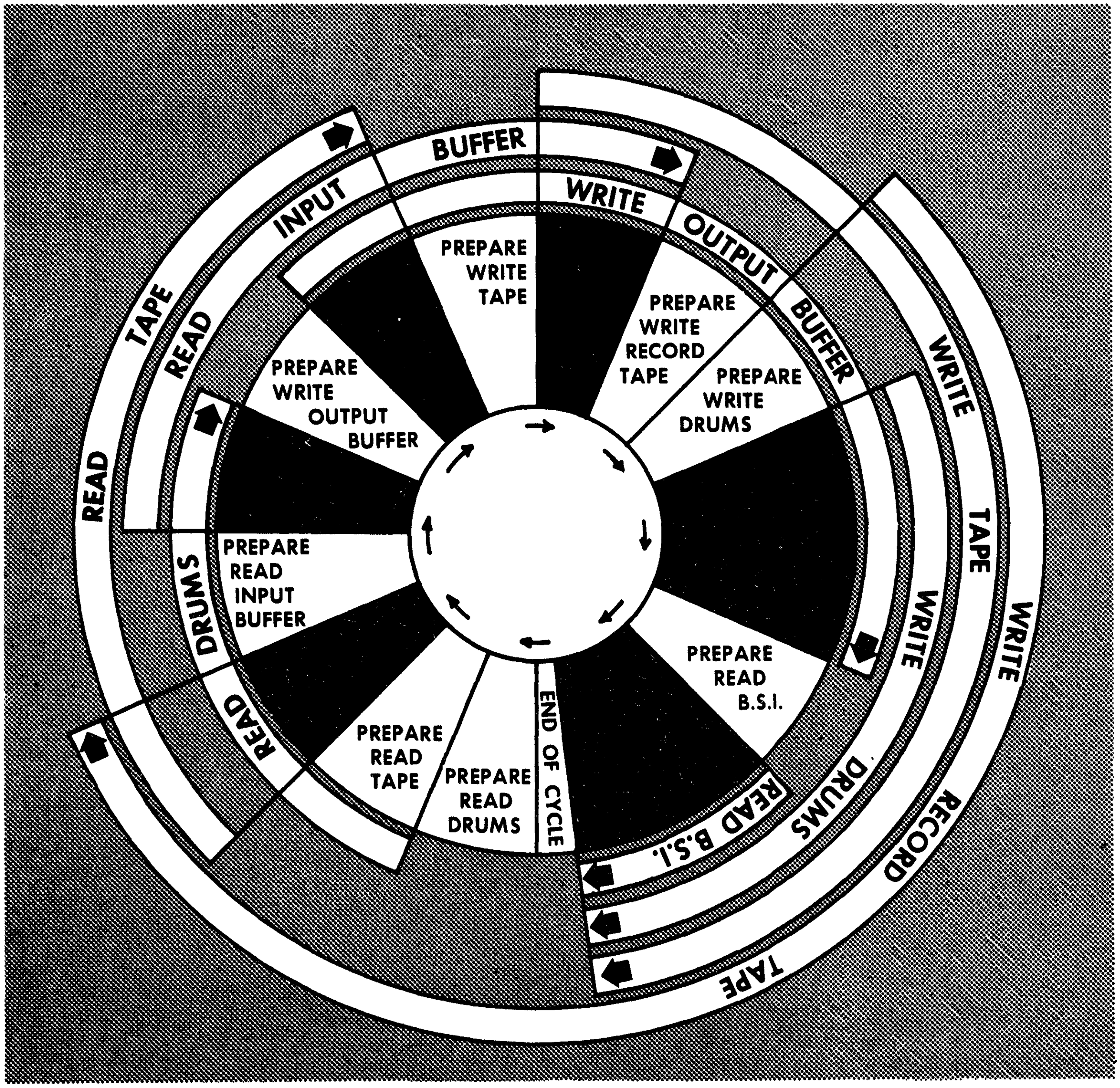


Figure 10. Input-Output Cycle.

processing have been carried out during the cycle. If the end of cycle routine finds that further processing remains to be carried out, the cycle is accordingly extended.

The end of cycle routine serves another purpose. If it finds that the cycle has been completed in an extra short time, it extends the cycle, permitting the program to do further miscellaneous tasks in the cycle rather then have the computer "spin its wheels" sampling buffers too frequently.

Program Implementation

It is only fair to point out that Figure 10 is not one of the steps in setting up the multi-level program, but rather an advance view of the result. The same steps are shown in Figure 11 as a flow chart that describes the cycle less graphically but more simply.

A program interrupt at the completion of a peripheral device operation generally leads directly to the initiation of some other

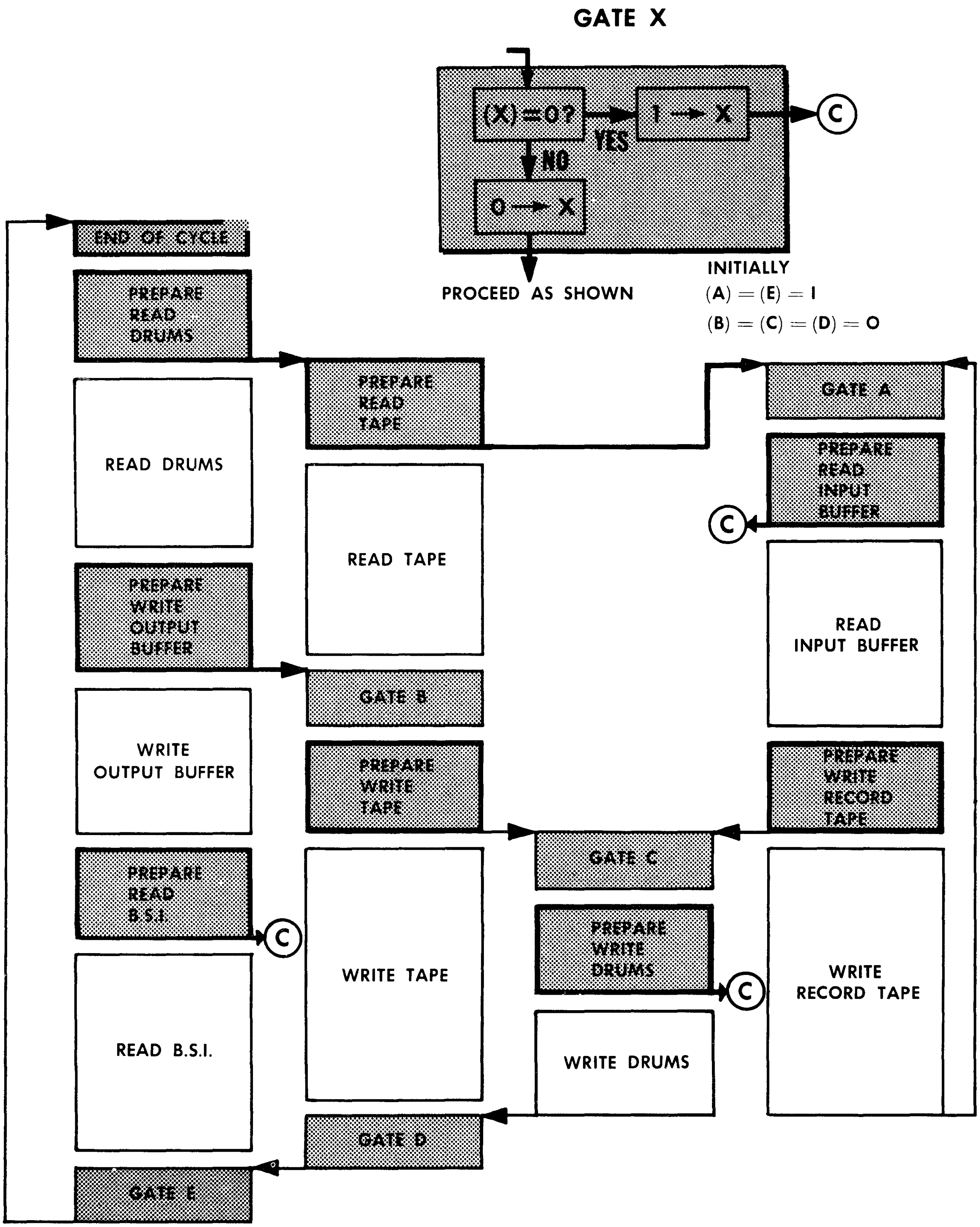


Figure 11. Input-Output Cycle Flow Chart.

peripheral device operation. The one completed is generally the last prerequisite for the other. Other prerequisites are implied by the sequencing of the parts.

One of the problems in this type of programming is to find ways to represent on diagrams the peripheral and computer operations. They are interconnected by initiations and interrupts, but operate asynchronously and do not lend themselves to representation with time as a fixed coordinate. Figure 11 uses some conventions to meet this problem. When the operation of a peripheral device begins, the computer goes on with some other task (shaded blocks). At such branch points, therefore, both branches must be taken. The computer and the device operate at the same time. When the peripheral device has finished its job, it initiates a new part of the program by the interrupt mechanism. This interrupt may sometimes not occur immediately, but it is most convenient to indicate it as the next step after the peripheral operation.

Figure 11 has five places where two branches meet, called gates. Each of these is a program step associated with a program control flip-flop. When either of the entrances indicated brings the computer to that spot, the flip-flop is examined to see if the other step has already occurred. If not, the state of the flip-flop is changed, and the Control Program determines the next function to be carried out. When the second of the two entrances occurs, the program proceeds as shown. Initially, the gates are set up in the same pattern that they reach at the end of each cycle when all jobs have been done.

When a cycle begins, the three read operations are promptly begun, and the computer goes to the Control Program, as indicated by the circled C. While the read operations are going on, the computer works on its other tasks, as will be described more fully later. As soon as the read input buffer operation is finished, the computer initiates the write record tape sequence, whether other reads are completed or not. Similarly, when the read drums operation is finished, the write output buffer operation is begun. When both drums and tapes have been read, the write tape sequence begins, and when all three read operations are over, the write drums operation is started.

Of the write operations, the write output buffer operation should be finished first. As soon as it is over, the indicators are read in preparation for the next cycle, which begins when all write operations are finished. The write record tape sequence may extend into the next cycle. However, it will not delay any of the read operations except the one for which it is prerequisite.

The length of this cycle, and even the sequence of events in it, depends on how long the peripheral operation takes. If the write tape sequence is long, other operations that do not depend on it are dispatched as soon as possible. Any of seven sets of operations may determine how long the cycle actually lasts. These can be seen by tracing through the diagram. The cycle length is the longest of the following seven combination (along with associated computer processing):

1. Read Drums, Write Output Buffer, Read B.S.I.
2. Read Drums, Write Tape
3. Read Drums, Write Drums
4. Read Tape, Write Tape
5. Read Tape, Write Drums
6. Read Input Buffer, Write Drums
7. Write Record Tape (portion remaining from previous cycle), Read Input Buffer, Write Drums.

Checking against the list of prerequisites will show that the cycle could not possibly be shorter than any of these sequences. Thus, the program does succeed in optimizing cycle length.

Non-Peripheral Operations

Both the cycle diagram and the flow chart (Figures 10 and 11) are concerned primarily with peripheral device operations, as the scheduling of these operations is the most critical part of the program under consideration. It is also important to incorporate efficiently the other operations, such as input and output processing, that must be performed during every cycle. On the cycle diagram, blank spaces in the computer ring indicate when these operations are carried out. On the flow chart, entrances to these portions of the program are indicated by a circled C. This represents transfer to a specific part of the control program, which takes place at three times in the program, and also any time one of the gates is found "closed."

Upon transfer to this location, the processor examines a list of tasks, arranged according to their prerequisites like the peripheral device operations already considered.

The first item on the list is advance preparation of peripheral device orders. This preparation is done in advance whenever possible, in order to expedite later parts of the program. When all that is possible has been done (or immediately if no such operations were ready to be done), the program continues to the next time. It goes on through input and output processing, and any other tasks that need to be performed once per cycle, doing as much of each one as is possible at the time.

All of these operations are subject to interruption when a peripheral device completes one of its tasks. After an interruption, the steps to terminate the peripheral operation will be dispatched. Then the task that was interrupted will be finished, and finally the processor will go back to the beginning of the list again. If any advance preparations, or other operations previously passed by, have become possible as a result of the operation just completed, they will be done next. The list is arranged in preferential order, with the tasks that are prerequisites for other operations considered first.

If the program reaches the end of cycle indication with some tasks on this list not yet done, the next cycle will be held up until they have been completed. This situation will not generally occur unless requirements for processor time exceed those for peripheral devices. Normally, the program will catch up on these operations frequently during the cycle, and have time to go into other operations of lower priority.

Base Level

We have seen the relationship between peripheral device control operation and processor operation. The former constitutes one of the levels of the multi-level program. Processor operations are divided into two levels, one of which has already been considered. The operations that must be done in each cycle, including the preparations for peripheral operations, constitute the base level of the program. It is related to the peripheral device control level as shown in Figure 12. Base routines are carried out successively under base control. They usually lead to peripheral device operations, and after initiating a peripheral device operation the computer returns to base control to determine what routine should be worked on next.

Interim Level

A third level, the interim level consists of miscellaneous tasks that need not be synchronous with the input-output cycle. The program reverts to the interim level whenever base control finds no routines ready to be performed. Interim control examines a list like that of base control, with tasks arranged in preferential order. The program goes through the list, performing any tasks for which the required data are available. Performance of possible tasks continues through the list until a peripheral device interrupt leads the program back to base level. If the peripheral device operation opens up new possibilities at the base level, the processing will not necessarily return to interim level for some time. When it does so, the task that was interrupted is finished, and the list is examined again from the beginning.

While a cycle may be held up if all base routines have not been performed, interim routines are allowed to extend over several cycles. When the computer is heavily loaded, this could result in long waiting for tasks at the end of the list examined by interim control. Even though they are not critical, these tasks must not be delayed indefinitely. To insure against this, the interim program differs from the base in one respect: each task is assigned a quota (either predetermined or program controlled) representing the amount that should be done in one cycle. The control program moves on to each task when the quotas are completed for the previous ones, rather than when the entire tasks have been completed.

If the end of the cycle finds tasks that have not received their quota of processor time, the cycle will be held up as for base operations. If, on the other hand, the quotas are all completed before the cycle is over, the interim control program may start through the list again, giving "second helpings," possibly smaller than the original quotas.

The last item on the list of interim tasks is error-checking routines. These can go on indefinitely, so that the processor will never be at a loss for something to do, even if it remains in the interim level for long periods of time. This of course, will not occur unless all the higher level tasks have been taken care of.

The base and interim levels are interruptible by peripheral devices. The control

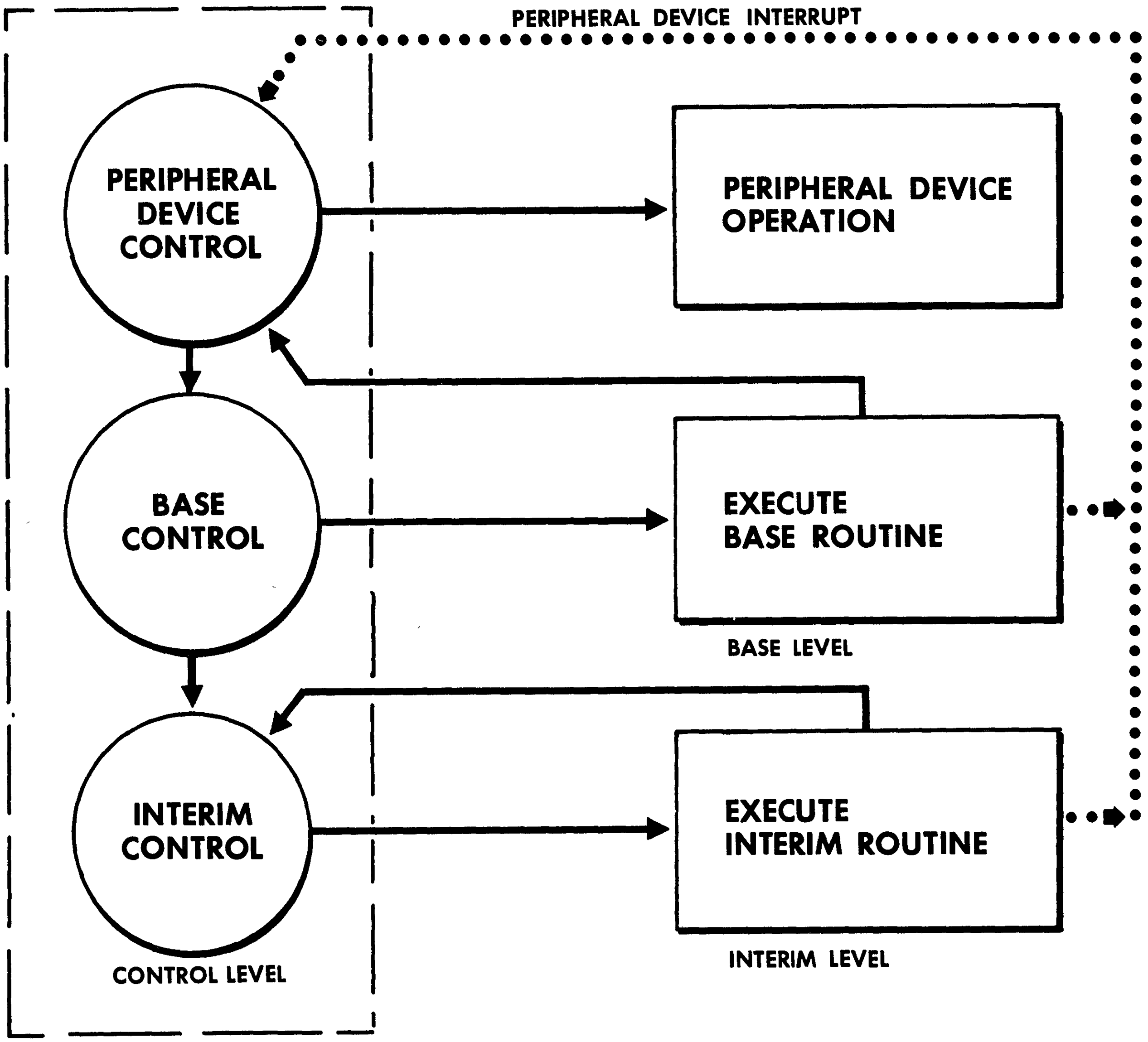


Figure 12. Program Sequencing.

program, which includes not only peripheral device initiation and termination but control routines for the other two levels, is not interruptible.

Multi-Level Sequencing

Figure 13 reviews the realtionship among the levels by showing a typical segment of the program in time sequence. A base routine prepares two instructions for a peripheral device, and then transfers to the control level. This level includes not only the initiation of the first peripheral operation, but the base control decision as to what routine will be worked on next. The transfer to the actual execution of this routine is shown as a shift to the base level.

The second base routine starts to prepare an operation for another device (these could be read drum and read tape in this program). When the first device finishes its first order, it needs the computer's attention very briefly to start its second order, and accordingly interrupts the base routine. When the brief control routine to start the second order is over, the base control makes the decision to return to the interrupted routine, and it is taken up from the point of interruption.

After the second peripheral device has been given its job, the control program examines the base list, and, finding nothing to

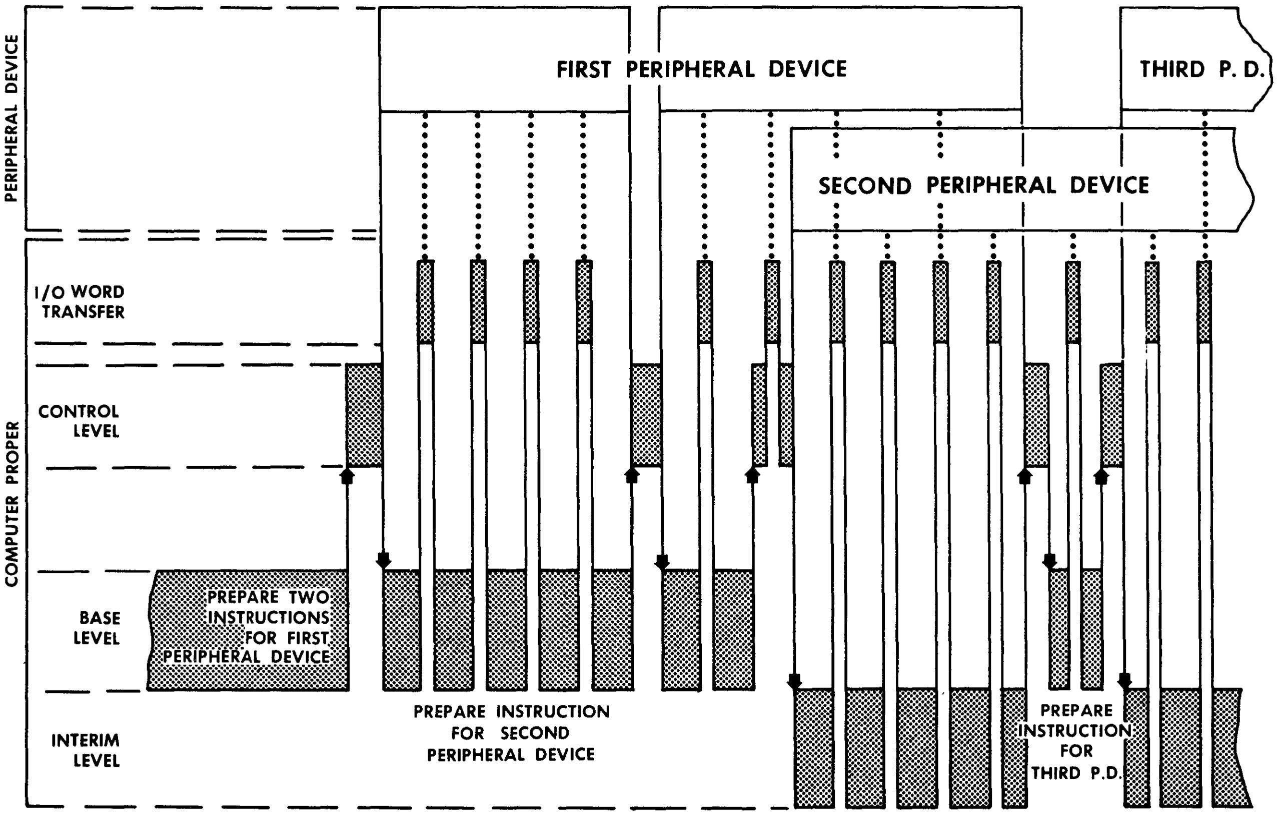


Figure 13. The Multi-Level Program.

be done, goes to interim control. From there, the program reverts to interim processing, to continue a task previously interrupted, or take up a new one from the list. The completion of the second order by the first peripheral device interrupts the interim program, but the processor does not return to it immediately after the control routine. This operation was the last prerequisite for a third peripheral device (possible the write output buffer). A new base routine that could not be done before is possible now. After it and the follow-up peripheral device order are done, the program goes back to the interim level. Here too there may be a new task possible that could not be done before.

This system of control, interim, and base programming, insures that there will not be delays unless traffic necessitates them. If the computer is ever idle (continuously performing error detection routines), it is because it has finished all its other tasks. Furthermore, peripheral devices are operating to bring it more work to do as soon as possible. Similarly, if peripheral devices are ever delayed, it is because the computer has been constantly busy and yet has not completed the tasks required of it.

Conclusions

Multi-level programming adjusts itself in each cycle to the traffic conditions. When the system is computer-limited, the computer works full time to meet the situation, holding up peripheral devices as necessary. When peripheral devices require more time than the central processor, their operations are optimally sequenced. Meanwhile, the processor keeps its own operations caught up, and whenever it is forced to wait for peripheral devices, it busies itself with low priority tasks or error checking.

This type of programming succeeds in keeping the processor and associated devices busy whenever possible, which is one criterion for judging the efficiency of multiprogramming. This system goes further, and makes a highly flexible choice of what tasks various devices will perform first. It anticipates deadlines by giving first precedence to operations on which others depend, and thereby avoids alternating periods when equipment is very busy and idle under the same conditions. Thus, while it represents a considerable programming effort initially, it pays off in truly efficient real-time operation.

DODDAC - AN INTEGRATED SYSTEM FOR DATA PROCESSING, INTERROGATION, AND DISPLAY

Walter F. Bauer and Werner L. Frank
Thompson Ramo Wooldridge Inc.
Canoga Park, California

ABSTRACT

With the DASA Department of Defense Damage Assessment Center a significant advance has been achieved in the establishment of a large-scale military data handling system which operates under real-time constraints. Although characteristics and requirements of this system are similar to those of Command/Control systems now in development, the integration of man to this configuration is of paramount concern. The DODDAC demands swift man and system interaction to elicit the information upon which decisions are based.

The system consists of a Control Data Corporation 1604 Satellite System and communication and display devices manufactured by Thompson Ramo Wooldridge Inc. The latter includes a Computer Communication Console and an on-line group display.

Introduction

The need for a damage assessment center that could rapidly handle and process extremely complex military data led to the expansion of the Defense Atomic Support Agency to include the Department of Defense Damage Assessment Center (DODDAC). Under the Deputy Chief of Staff, Damage Assessment Systems of DASA, DODDAC now provides damage assessment support to elements of the Department of Defense and the Joint Chiefs of Staff. It provides comprehensive information affecting peacetime and wartime decisions with rapidity after particular queries are received. Because of the great amount of data handled, the complexity of the processing, the comprehensiveness and the facility of output, and the fast system responses required for both man and machine, the design and implementation of the data system are especially challenging. In delivering this performance, DODDAC represents a significant advance in large scale data handling systems.

The characteristics and requirements of this system have many features common to the general class of military Command/Control systems now in development. They require, for example: large capacity random data storage devices, parallel processing of data, continual operation with near absolute reliability, real-time response, and servicing of queues formed by continual data entry and consumer requests for output.

Of paramount concern was the integration of man to the configuration, i.e., providing him with adequate tools by which he interacts with the data and the processing. In this respect especially, the design represents one of the more advanced real-time systems incorporating on-line interrogation and display.

The DODDAC system design and project management was provided by the DASA. The design was implemented by a Control Data Corporation Satellite System, a Ramo-Wooldridge Interrogation and Display System, and by programming and modelling accomplished by the System Development Corporation. This paper presents the technical aspects of the data processing and display systems, following a discussion of the mission, functions, and requirements of DODDAC.

Mission

DASA is a joint services organization with broad military responsibilities in the atomic energy field. DASA, formerly the Armed Forces Special Weapons Project, succeeded the Manhattan Project of World War II. As part of its responsibilities, DASA has for some years been performing computational studies of weapons effects, hazards, and vulnerabilities related to atomic warfare, exercised earlier by the DASA Deputy Chief of Staff for Weapons Effects and Tests and now, insofar as applications to real target systems, by the Deputy Chief of Staff for Damage Assessment Systems. Under the latter, DODDAC was established by a Department of Defense directive. On March 4, 1960, the Chief, Defense Atomic Support Agency, was designated as Executive Director of the DODDAC.

Should war occur, DASA (DODDAC) will support the Joint Chiefs of Staff and other designated military and government groups in assessing nuclear damage sustained by the armed forces and resources of the United States, its allies, and the enemy, supplementing its peacetime responsibility for appraisals of attack hazards and vulnerabilities.

Figure 1 depicts schematically the mission of DASA DODDAC by presenting the overall data flow through the system. Data comes into the DODDAC from the indicated agencies in three categories: target data, system parameters, and attack information. Target data is a comprehensive description of forces and resources, system parameters include certain data on weapon effects, and attack information is that volatile data related to a specific and unpredictable pattern of events during post attack phases.

The sources and users of data shown in Figure 1 is not an exhaustive list but a representative one. The principal user, as indicated, is the Joint Chiefs of Staff.

Functions and Requirements

A general statement of the functions of the DODDAC is as follows: perform hazard and vulnerability studies on a continuing basis, keep an up-to-date file of military forces and resources information, and in the event of hostilities, accept information quickly, process it rapidly, and provide display products suitable for top level command use.

The DASA-DODDAC has also been given the responsibility to support the war gaming activity of the Joint Chiefs of Staff. In this capacity the DODDAC works with the Joint War Games Control Group from which guidance is given leading to the development of gaming models, computer programs implementing the models, output designs and presentation methods.

More specifically, the imposing list of requirements is as follows:

1. Provide a data base of information necessary for damage assessment processing which can in the near future grow to 100 million characters.
2. Provide a means to up-date the data base on a day-to-day basis.
3. Design and implement a data handling system for handling these large amounts of data accurately and expeditiously.
4. Allow for the data transmission to DODDAC over hardened, secure, and reliable communication facilities, and devise and arrange for sources of information appropriate to the damage assessment process.
5. Accept attack data as real-time inputs and allow for their direct input to the computer system.
6. Develop damage assessment models which represent the delicate balance between comprehensiveness or realism, and speed.
7. Provide a technique for man/machine communications, allowing quick access to nearly all data in the files.
8. Provide a system for the automatic generation and presentation of high quality output products of fast response which are suitable for group viewing by top command personnel.

There is a special challenge in the last two requirements for there are few, if any, data processing systems in existence today which provide all flexible and all-purpose interrogation and display system required.

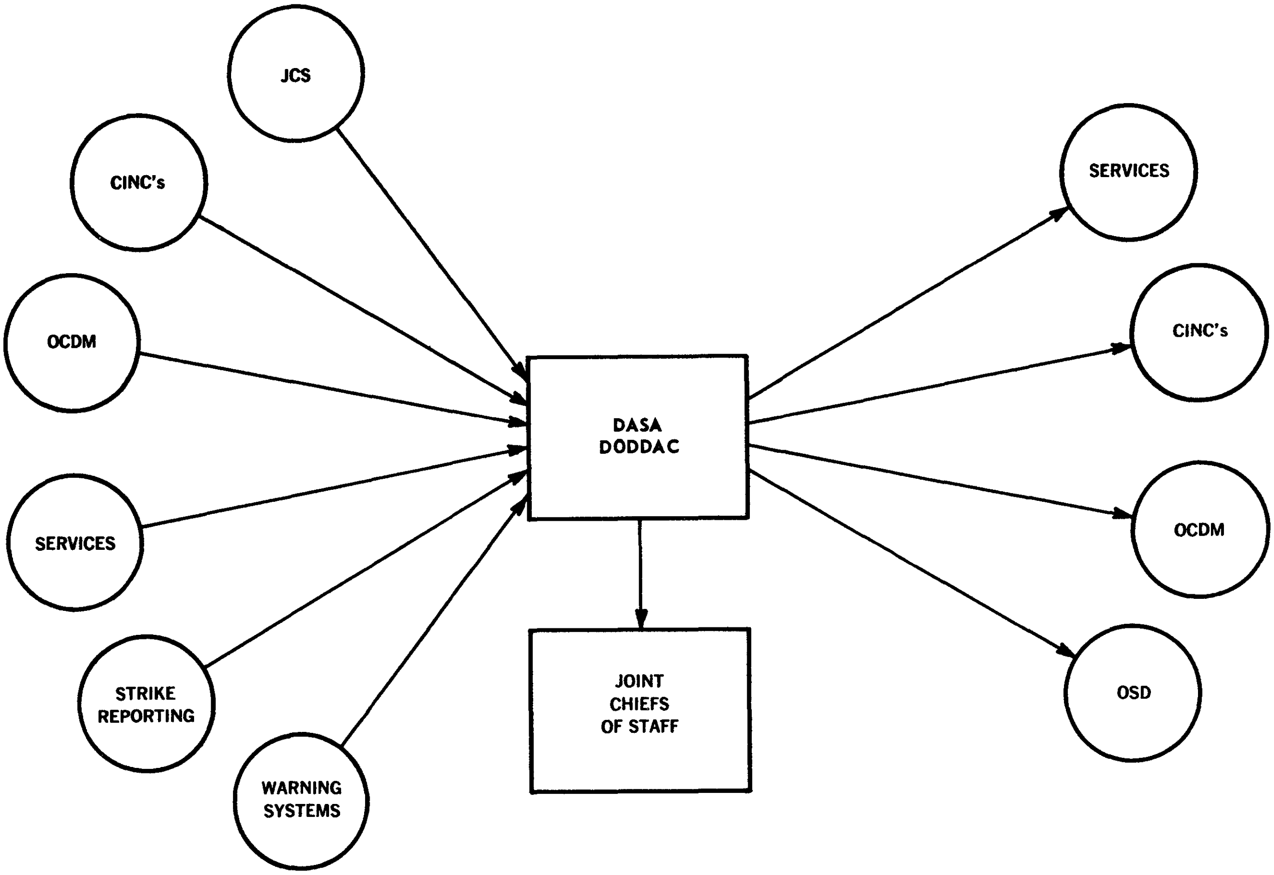


Figure 1. DASA--DODDAC Mission - Overall Data Flow

Figure 2 shows the functions to be performed by the data system. The input data, as shown, can come in through many sources: voice communications, teletype data links, specialized communication systems, and standard digital data links. The system must allow for automatic entry of data as well as manual type entry. The computations must allow for various kinds of damage assessment processes. For the various models, there must be a number of modes of operation which allow various trade-offs between comprehensiveness of the model and the speed with which results must be obtained. Standard output in terms of hard copy must be available. In addition, individual or console type displays are necessary for the individual analysis of file data.

A last requirement for output is a display for group viewing. Man/machine facilities for the input of data and for calling for requests, complete the required capabilities of the system.

Of special concern is the system for group display. In order to have the quality and comprehensibility of the group display required for command use, it was deemed necessary to have a full color display system. It was desired that this display system allow for the placing of symbols on maps and charts in full color, with no dilution of the color of the superimposed symbols. Finally, because of large amounts of information to be placed on the slides for group viewing and the fast response times necessary for their preparation, it was necessary to have completely automatic operation. This automatic operation must allow for the input of information directly from the computer, the automatic preparation of the slides, and the transportation of the slides to a projector for viewing as required.

Implementation Philosophy

In the fall of 1960 the first steps were taken toward the eventual operational DODDAC. In order to accelerate the establishment of the

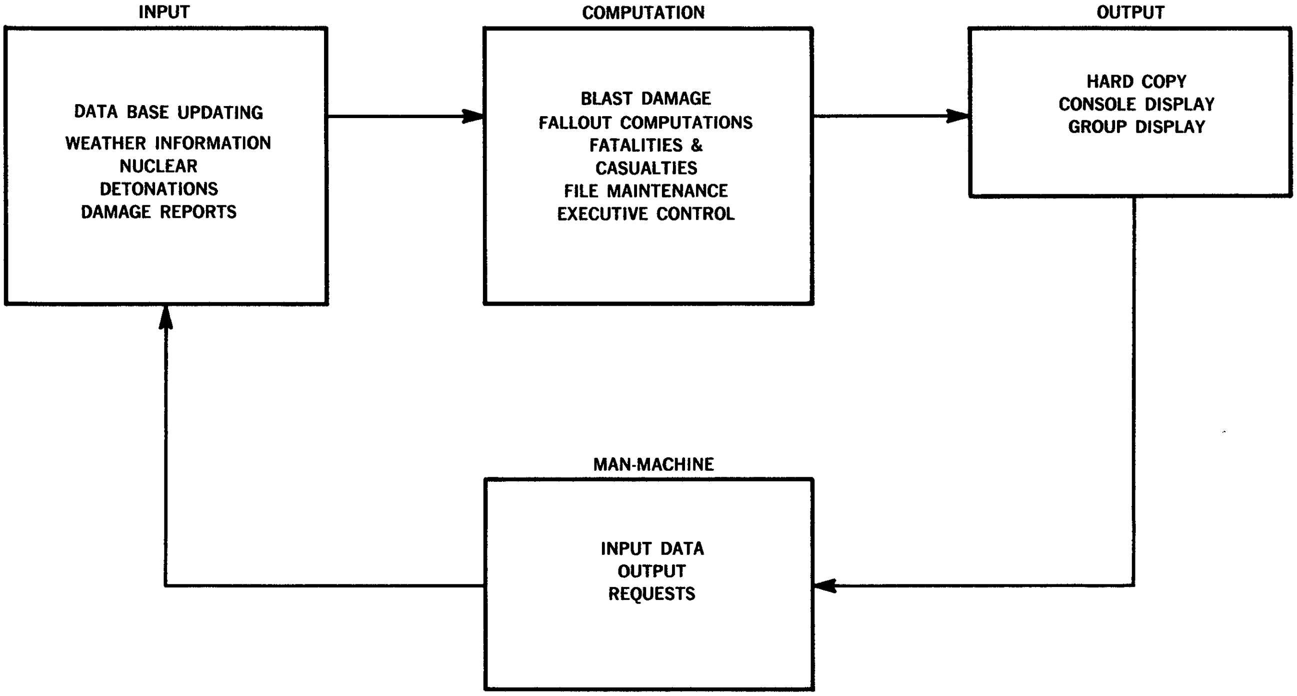


Figure 2. Data System Functions

necessary capability, an implementation philosophy was adopted which saw the following three activities in parallel development:

1. A semi-automatic manual system providing an immediate and basic capability.
2. A Developmental Center containing a simplex of equipments with which to study and test the operational DODDAC environment and requirements. This functional unit also represents a level of sophistication that fulfills many of the requirements set forth by the Department of Defense.
3. Develop a design of future systems, based on a better understanding of the problems and experience gained in operating the first two systems.

This plan allowed an early basic capability while recognizing that significant technology must evolve gradually in areas of computers, programming, and displays, with the passage of many months. It was recognized that starting immediately toward a full blown operational system would have been costly and inefficient; the plan allowed the maximum contribution to damage assessment required by national defense consistent with reasonable expenditures.

In this paper we consider aspects of the Developmental Center which is now equipped with data processing and display facilities. This system was integrated and established through the combined participation of military and industrial groups. As shown in Figure 3, the Project Management and system design was supplied by DASA which contracted the computer subsystem to Control Data Corporation, the programming data processing tasks to System Development Corporation, and the man/machine communication and display subsystem to Thompson Ramo Wooldridge Inc.

In the fall, 1961, the Developmental Center was equipped with the data handling equipments. The Developmental Center satisfies most of the requirements listed above. It is implemented with a large-scale multicomputer system and a modern man/machine communication and display system which will provide the basis for future developments of operational systems.

Data System Concept

The DASA-DODDAC's function is to provide command elements with appropriate

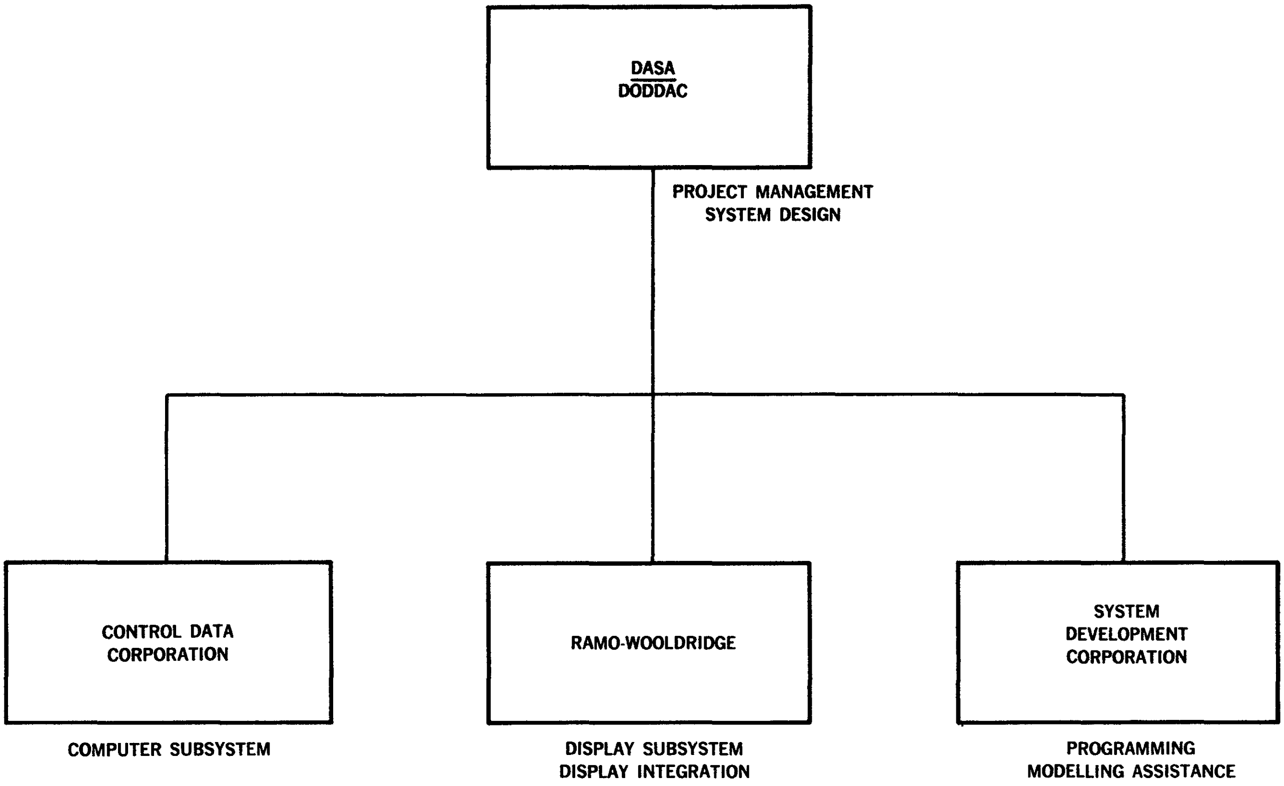


Figure 3. DASA--DODDAC - Contractor Team

information in a military environment with little delay. These requirements were translated into a data system having the following capabilities and attributes:

1. Parallel processing
2. Ease of communication interface
3. Flexibility of organization
4. Growth potential
5. High speed
6. Large, random access storage

The operational tasks were analyzed and are shown in Figure 4. Each of these areas is a functionally independent block requiring periodic intercommunication.

Computer Subsystem

The Control Data Corporation 1604 Satellite System was selected by DASA as the computer subsystem for the Developmental Center. This multi-computer system possesses attributes which meet the data processing requirements stated above. The requirement for a large, random access storage device was answered by the addition of a Bryant-320 Disc File to the CDC equipment. This unit is able to store over 30 million characters, and has a character transfer rate which ranges from 20 KC to 62.5 KC.

The complete system includes:

1	1604 Computer
2	160 Computers
3	1607 Tape/160 Computer controls
12	Magnetic tapes (Ampex FR-307, character transfer rate 30 KC)
1	Bryant 320 Disc File
1	1610 card reader, punch adapter
1	Card Reader (IBM 088)
1	Card Punch (IBM 523)
1	1606 Printer Adapter
1	High Speed Printer (Analex, 1000 lines per minute)

Each of the three computers has an associated Ferranti paper tape reader and teletype tape and punch. Also, a Soraban-modified IBM electric typewriter is included with the 1604 and one of the 160 computers.

The organization and interrelationships of the system are depicted in Figure 5. In this configuration one of the CDC 160 computers is designated to the input/output area, providing the necessary buffer and processing capabilities for data entry and hard copy output. This computer also has the function

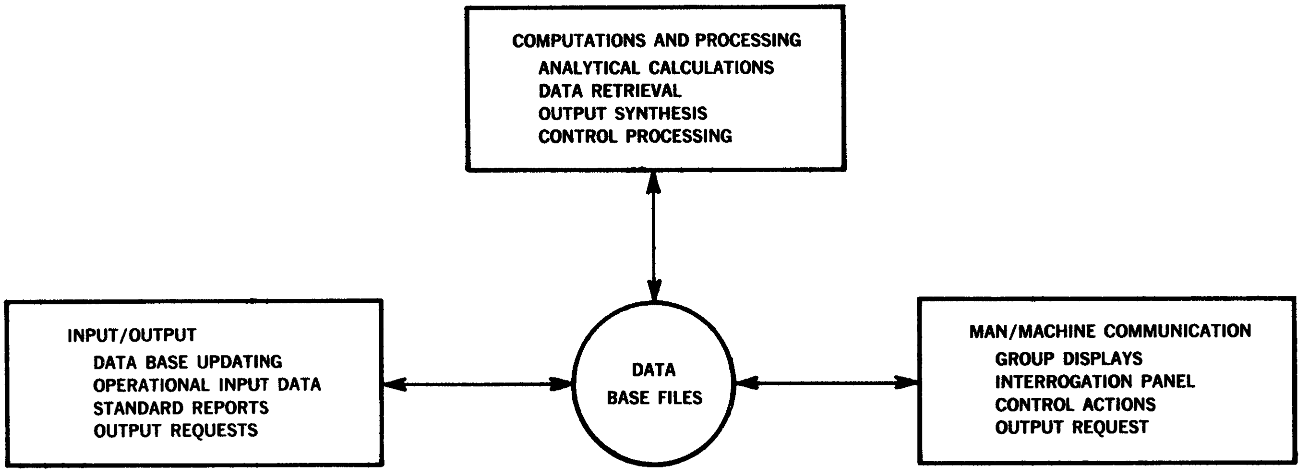


Figure 4. DASA--DODDAC Data System Functions

usually associated with off-line tape to card, card to tape and tape to printer operations.

The second CDC 160 is devoted to a time-shared operation between the three units electrically tied to itself. This includes the 1604, a communication console, and the large screen display. The latter items will be further discussed in the next section.

Finally, the 1604 computer performs the analytical and retrieval tasks. This includes the computation of the damage assessment process and the output processing. The scheduling and direction of data flow is under the executive control of a master program operating in the 1604.

It is significant that this system is completely integrated as an on-line functional unit. In this sense it is unique in adapting the satellite computers for both "off-line" type operations and as computer partners in support of the real-time processing operation.

An interesting design problem was faced in the assignment of data transfer channels to the various devices connected to the 1604. Since the computer is equipped with 3 input and 3 output buffer channels and a high speed transfer channel, a large degree of flexibility is afforded.

In the DODDAC, the 1604 has direct connection to the following:

a. Console typewriter
b. Console punch tape punch
c. Console Punch tape reader
d. Three 1607 tape systems
e. Printer control unit
f. Card reader/punch control unit
g. Disc file

Figure 6 indicates the channel assignments as established for the 1604. The rationale behind this selection was guided by the following principles:

1. Peak performance satisfaction during real-time operation
2. Accommodate individual data transfer rates
3. Avoid time sharing of high duty cycle devices
4. Provide alternate paths for reliability and back up.

Of the nine devices connected to the 1604, the priority equipments are the disc file, the 1607 associated with the display subsystem and the 1607 servicing the input/output. Accordingly, each of these units is assigned to separate buffer channel pairs.

Secondary assignments are now made for the remaining devices. The typewriter, tape reader and card punch are all associated with the input/output channel. This has two advantages:

1. Provides high priority I/O capability for short bursts which circumvent normal queuing procedures.
2. Gives alternate paths in the event of a 1607 or a 160 failure.

The remaining element requiring assignment is the third 1607. This is tied to the channel used by the disc since these two elements were usually not to be called upon at the same time. The associated tape units also provide

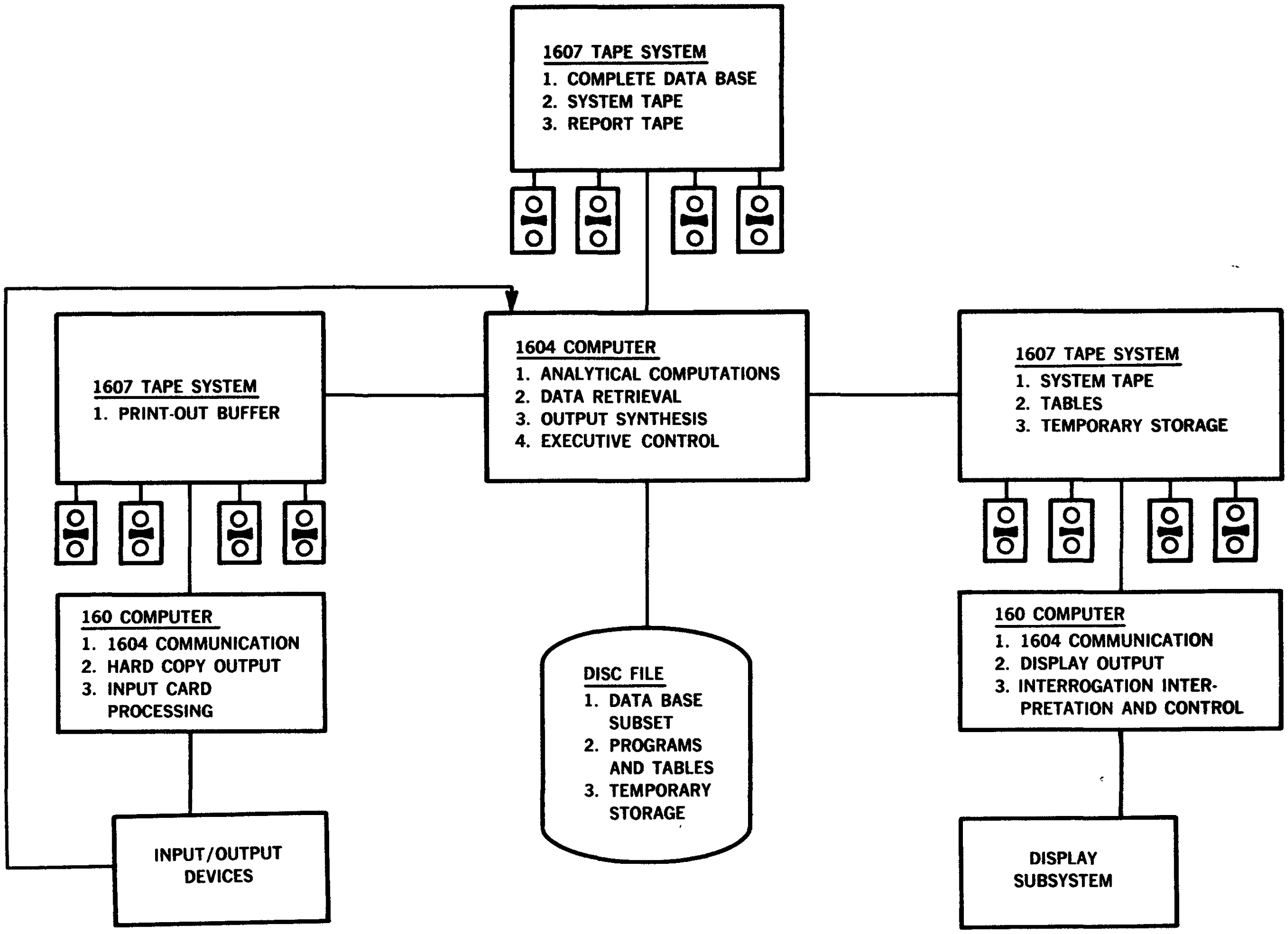


Figure 5. DSAS--DODDAC Computer Subsystem Organization

a potential backing to the disc since both devices contain the data base.

The assignments reflected above provide a clear channel for the interrogations and outputs serviced by the 160 computer attached to the display subsystem.

A design decision was made in attaching the disc file to the buffer channel instead of the high speed transfer channel. The disc information transfer rate falls between the transfer rates of the two types of input/output channels. Connection to the high speed channel would serve to slow down the effective computing capacity of the central processing; connection to the buffer channel would limit the servicing capability to the competing auxiliary devices whenever computer and data transfer is at a saturation point. The latter alternative was chosen and the executive program controls the additional channel activations whenever the disc is accessed.

Man/Machine Communication and Display Subsystem

The real-time aspect of the DODDAC requires man to dynamically interact with the system to obtain timely information concerning the status of internal information flow and data file content as they reflect the "outside world." This quest for facts is motivated by the need to make decisions which are time dependent.

In order to facilitate decision making on the part of DODDAC consumers it is necessary to provide adequate tools for summarizing and presenting data and methods by which interrogation can be readily processed.

These requirements are met by the Computer Communication Console (CCC) and an

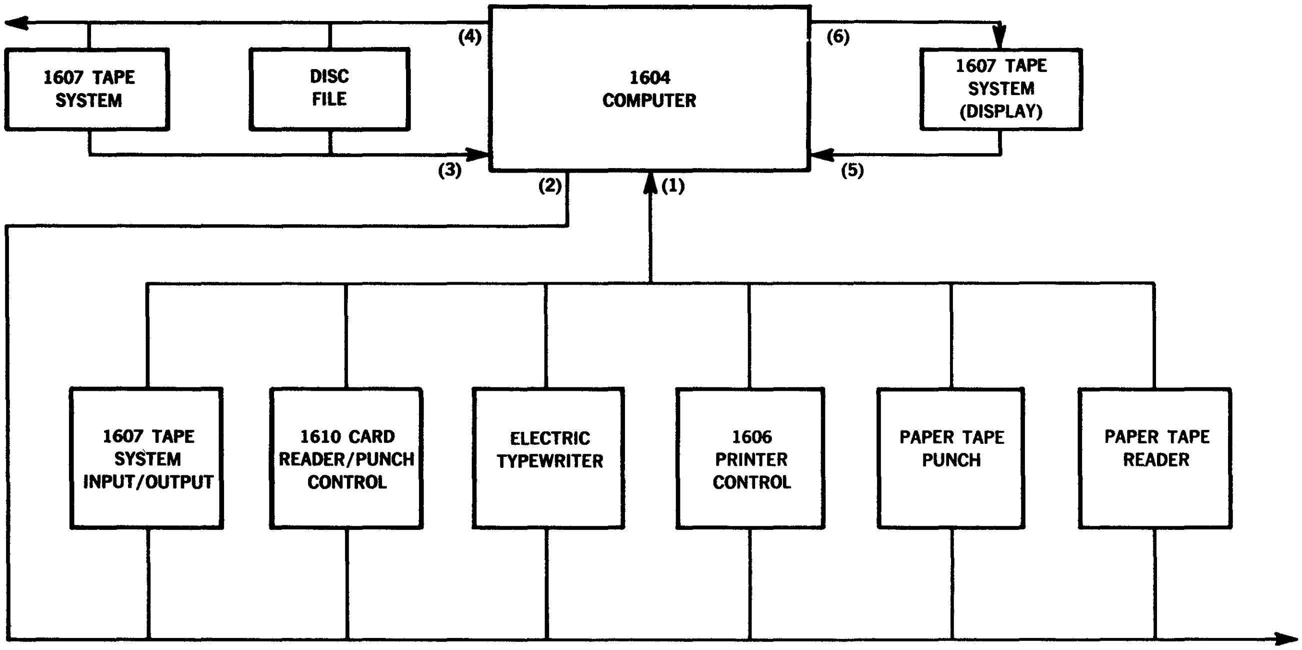


Figure 6. Buffer Channel Assignment in the DODDAC

on-line group display supplied by Ramo-Wooldridge. These items form the basis of the DODDAC Presentation Room. The CCC is a single operator device (see Figure 7) providing the ability to initiate interrogation and system control via a keyboard, or to display system status by use of a CRT read out. The group display generates film chips which are automatically delivered to a projector for large screen, full color viewing.

The CCC is a general purpose console facilitating communication between a human analyst/operator and a computer system in the performance of:

Data entry
Data analysis
Information retrieval
Display
System control.

These features are summarized as follows:

1. Control buttons on a console keyboard
2. Status lights for indicating system action
3. Alpha/numeric keys for data entry
4. CRT displays for alpha/numeric displays
5. Removable keyboard overlay for changing the problem orientation of the console.
6. Complete program control of all console keys, lights and displays.

Of particular significance are the CRT and the overlay. The electronic display permits the presentation of up to 720 symbols out of a font of 62 alpha/numeric and graphic symbols. Options are available for either outputting a full display of 720 characters or any number of symbols at selected grid points of the 20 row by 36 column display matrix. The refresh rate is under program control and is usually performed at 45 times per second in order to avoid flicker.

The overlay (see Figure 8) is the feature which gives the console application flexibility and user oriented characteristics. It consists of a plastic form which fits over a bank of 30 buttons designated as the Process Step Keys. When in place, the overlay supplies an associated labor for each button Thus, by properly engraving the overlay, the Process Step Keys become functionally tied to a particular job or operation.

There are 63 overlays possible, each one being identified to the computer, when in keyboard position, by a code generated by one to six prongs present on the underside of the plastic. Each overlay typically will have associated computer programs which are activated when buttons are pressed. Removal of an overlay and insertion of another effectively changes the orientation of both the console and the computer.

Each label and button of the Process Step Keys have associated illuminators whose state is under program control. These lights are utilized to indicate to the console operator current operating position by lighting the proper button, and next allowable steps by lighting selected labels. Sequential illumination of these lights, as a function of buttons pressed, serve to guide the operator through a particular process. Using these keys together with the data entry and CRT response capability, the console can outline and lead the operator in carrying out specific operations.

The on-line group display system produces photographic transparencies for immediate full color projection. The slide is a single, composite 70 mm film chip containing three images formed by a color separation process. These images are projected through the lenses of a special projector and recombined to give the color presentation. The slide content represents a combination of previously prepared backgrounds (such as maps, charts and photographs) with timely, computer-generated annotations.

The attributes of this display system are:

1. SPEED: The time lapse between the generating of annotation data and the

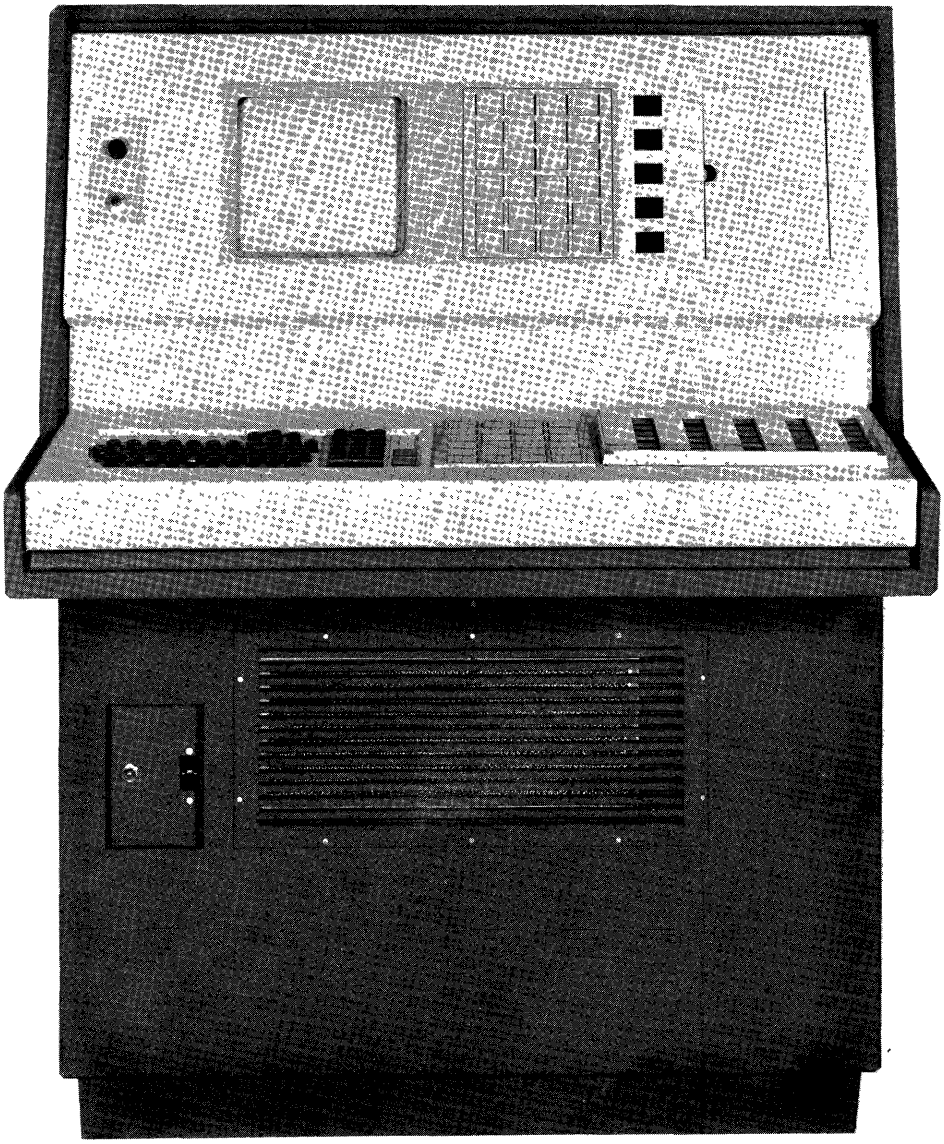


Figure 7. Computer Communications Console

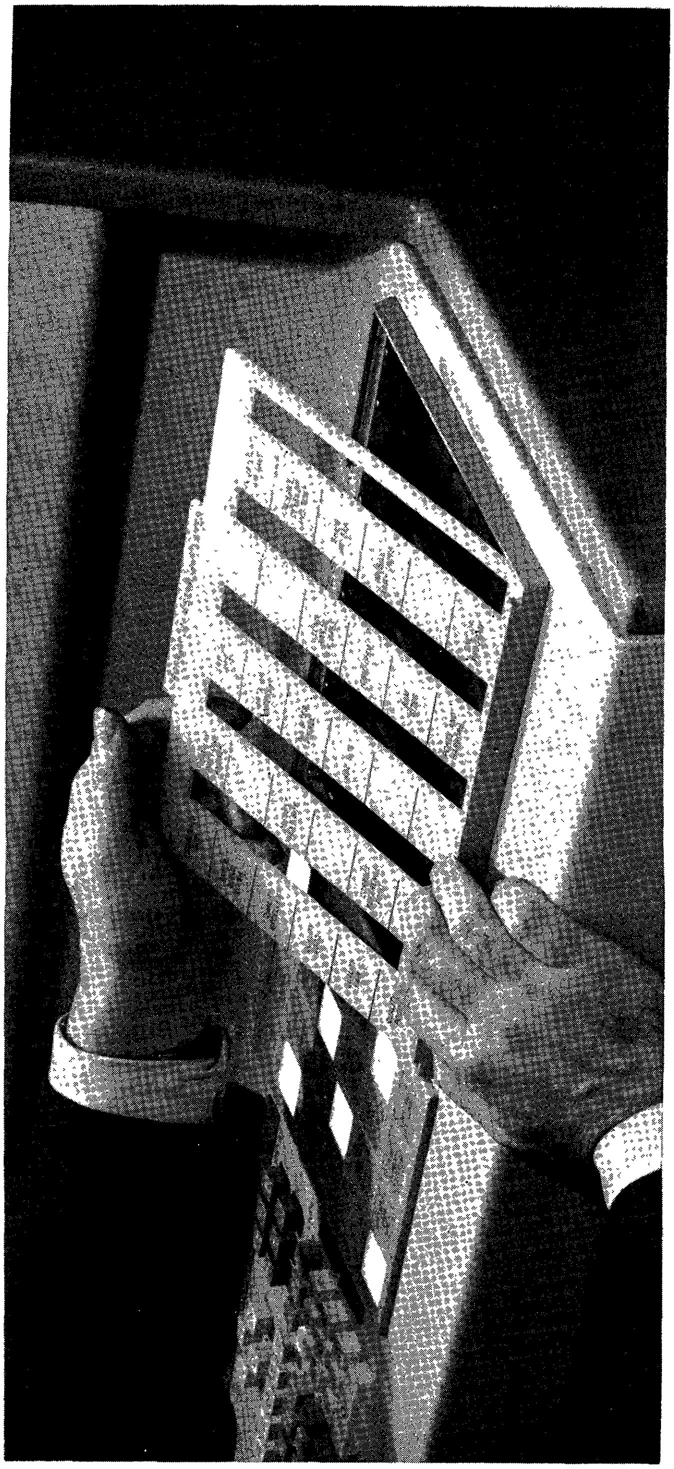


Figure 8. Console Overlay

projecting of the typical display chip is 30 to 60 seconds.

2. INFORMATION CLARITY AND DENSITY: The annotations appear as highly legible alphanumeric characters, lines, curves, or special symbols in any of eight fully saturated colors (including black and white); the background colors (full range) are equally well saturated. Background and annotations can thus be extensively color-coded.
3. INFORMATION PERMANENCE AND ACCESSIBILITY: Display chips constitute a permanent record easy to store and retrieve; individual chips are automatically selected from random access, 200-chip file magazines.
4. AUTOMATIC OPERATION: The film chip generation, processing and delivery is under complete program control.

These advantages are made possible by the use of various special techniques. The high overall response time is due largely to the high speed of photographic developing (1-1/2 seconds) resulting from a combination of two relatively new photographic techniques. One is that of color separation to create a color image from black and white film. The second contribution came from the use of Kalvatone film, which is a special type of black and white film that can be developed by heat instead of chemicals.

The legibility and color saturation of the displays derives from a "masking" technique that enables the annotations to be inserted into rather than superimposed onto the background so as to eliminate color mixing and weakening.

Convenient storage and retrieval derives from the unit record film chip. This slide can be projected on large screens or individual consoles. Slides can be duplicated and used for multiple projections or processed to standard color transparencies for dissemination or hard copy printing.

The equipment used to create the full color displays consists of four main units:

1. The Control Programmer, which receives display data and instructions from the computer and stores them temporarily for subsequent use by the Display Generator.
2. The Display Generator, which produces the actual display chip by obtaining the necessary instructions from the Control Programmer. According to these instructions, a display chip is sequentially exposed with particular sets of annotations, in given colors, and inserted in the desired space of the background specified. (This background can be any of 200 stored in the Display Generator's random access file.) It then develops the chip and, if requested, makes duplicates before passing it on.
3. The Monitor/Analysis Console, which is an operational viewing station used to perform quality control check on the production chips and maintenance control on the entire system.
4. The Display Projector (or projectors) receives finished chips from the Display Generator and projects them for individual or group viewing.

In the DODDAC these communication and display devices are electrically tied to the CDC Satellite System via one of the 160 computers. The relevant parts of this subsystem are shown in Figure 9. The 160 computer provides a satisfactory interface and is time shared between the following competing tasks:

1. refreshing of CCC's CRT display
2. control of CCC illuminators
3. monitoring of CCC output register
4. monitoring of 1604 communication
5. data output to the control programmer
6. message interpretation and composition
7. display subsystem control.

The display subsystem, however, entails more than the connection and operation of these equipments. The organization of data within the computer, the programming concerned with the retrieval of this information and the techniques employed in the requesting of data, are all integral parts of the man/machine communication. Because of the versatility of this display subsystem, the user is not restricted to a few standard displays from which he must deduce all the information required in the numerous situations he will encounter. Rather, he may request information for presentation in such a fashion that each display is tailored to the specific problem which exists at the time. Extraneous information is deleted from any display, leaving only that data which is pertinent to the situation at hand.

Accordingly, a great deal of flexibility is desired in making output requests. The CCC

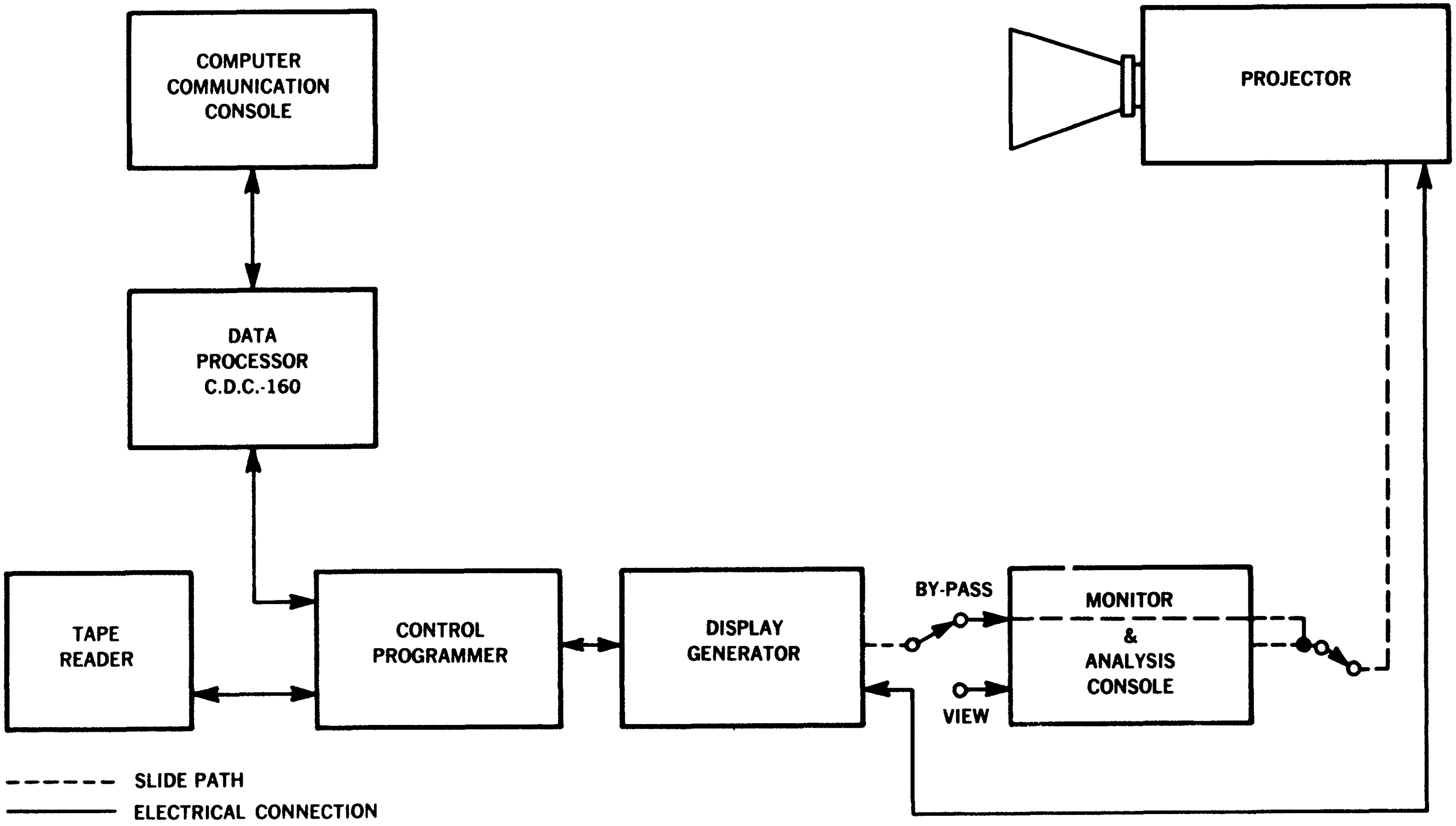


Figure 9. Full Color System Diagram

and the group display serve as the input and output elements for this process. In fact, the system user completes an information flow loop when he wishes to obtain data from the system as a result of viewing an output. This information flow is shown in Figure 10.

The output request overlay is instrumental in performing the interrogation. A suggestive implementation of such an overlay is given in Figure 11. After pressing the start button and registering the overlay to the computer, label lights are sequenced as a function of pressing each button. This sequencing is demonstrated in Figure 12.

Pressing any one of the keys will in addition to setting up this sequencing, also cause a CRT display to appear. These displays will generally be of two types. One allows the operator to make multiple choices of category items or alternatives. A second type will require data parameter inputs as specified in a form presented to the operator. In all cases, the parameters entered serve to set limits for the data retrieval and output processing programs.

DASA DODDAC Status and Developments

The integration of equipment, computer programs and man was initiated in the fall of 1961. In addition to having operational stature, the installation will be used to test the adequacy of damage assessment models, output displays, personnel requirements and general system design. Of particular concern is the establishment of communication interfaces and data input techniques.

Whereas the Developmental Center is located at the Pentagon, plans have been formulated for eventual DODDAC operation in other appropriate military environments. One of these is already installed and operating: the semi-automatic system at the Alternate Joint Communication Center the hub of which is an augmented IBM 1401 system.

As stated previously, we believe the Developmental Center System to be among the most modern large-scale data systems for real-time. The experience in using the system in the Center will add greatly to the design of future DODDAC systems which are responsive to the considerable requirements of the military mission.

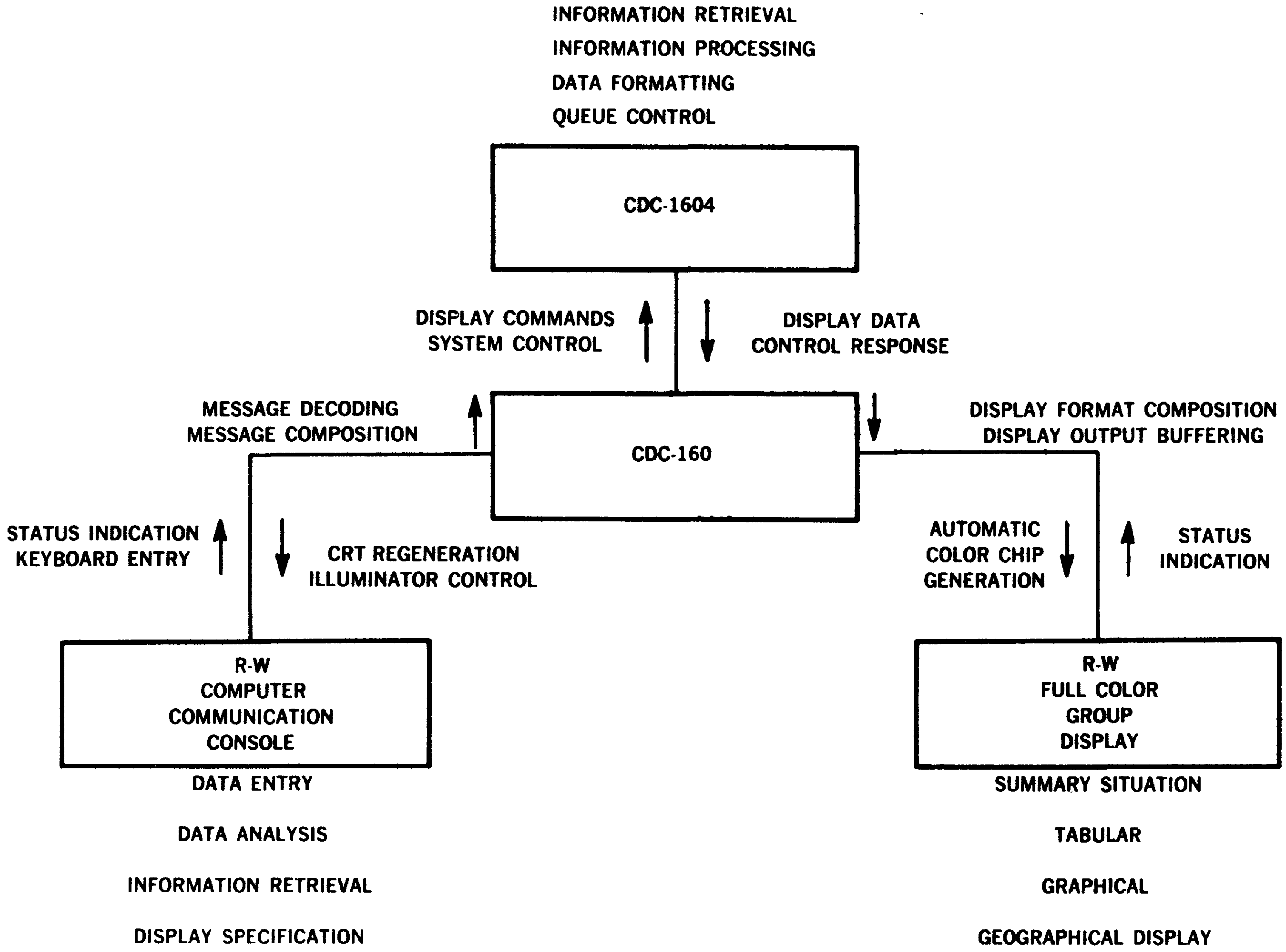
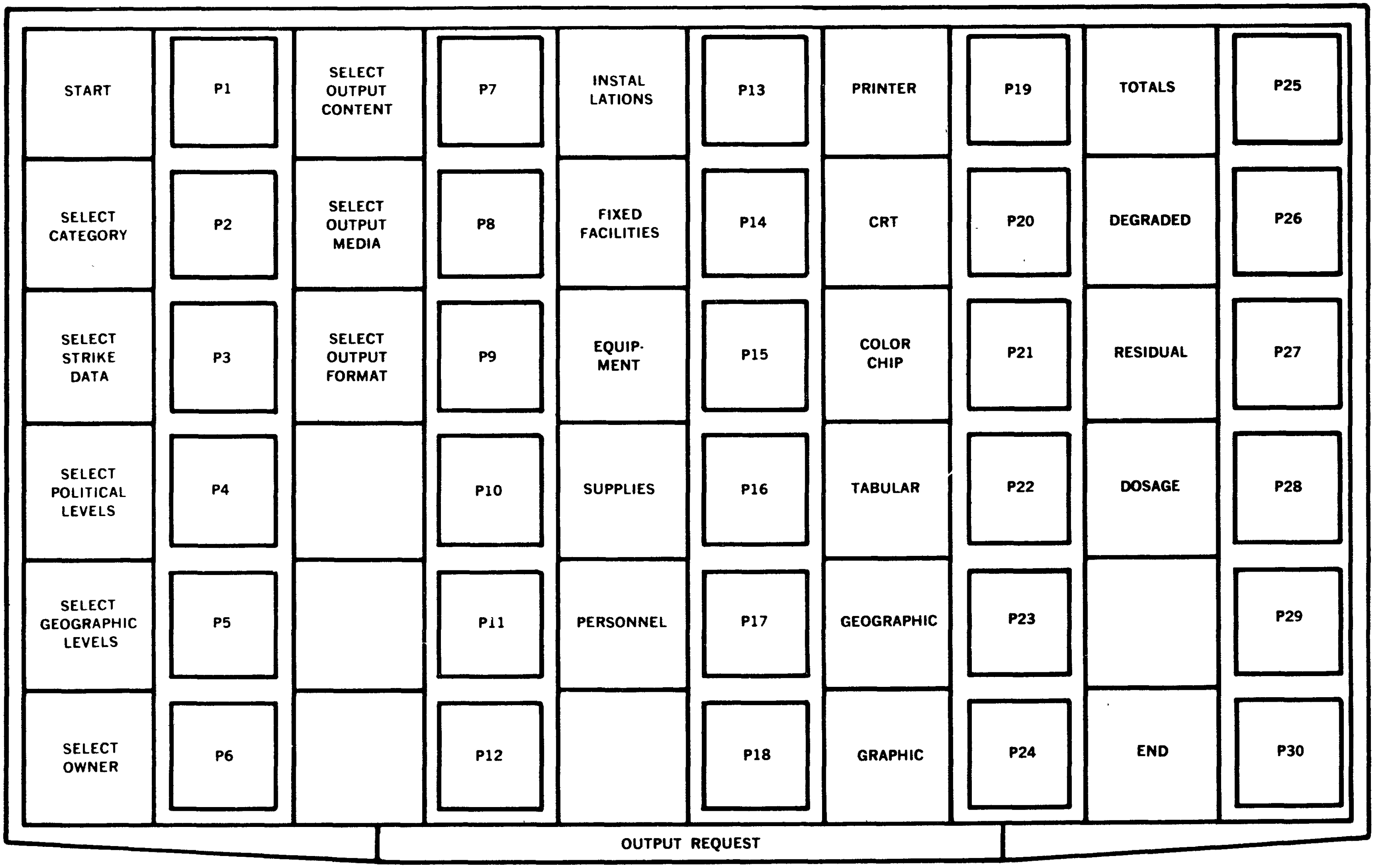


Figure 10. DODDAC Display Data Flow



START	P1	SELECT OUTPUT CONTENT	P7	INSTAL LATIONS	P13	PRINTER	P19	TOTALS	P25
SELECT CATEGORY	P2	SELECT OUTPUT MEDIA	P8	FIXED FACILITIES	P14	CRT	P20	DEGRADED	P26
SELECT STRIKE DATA	P3	SELECT OUTPUT FORMAT	P9	EQUIP-MENT	P15	COLOR CHIP	P21	RESIDUAL	P27
SELECT POLITICAL LEVELS	P4		P10	SUPPLIES	P16	TABULAR	P22	DOSAGE	P28
SELECT GEOGRAPHIC LEVELS	P5		P11	PERSONNEL	P17	GEOGRAPHIC	P23		P29
SELECT OWNER	P6		P12		P18	GRAPHIC	P24	END	P30

Figure 11. Process Step Key Overlay for Output Requests

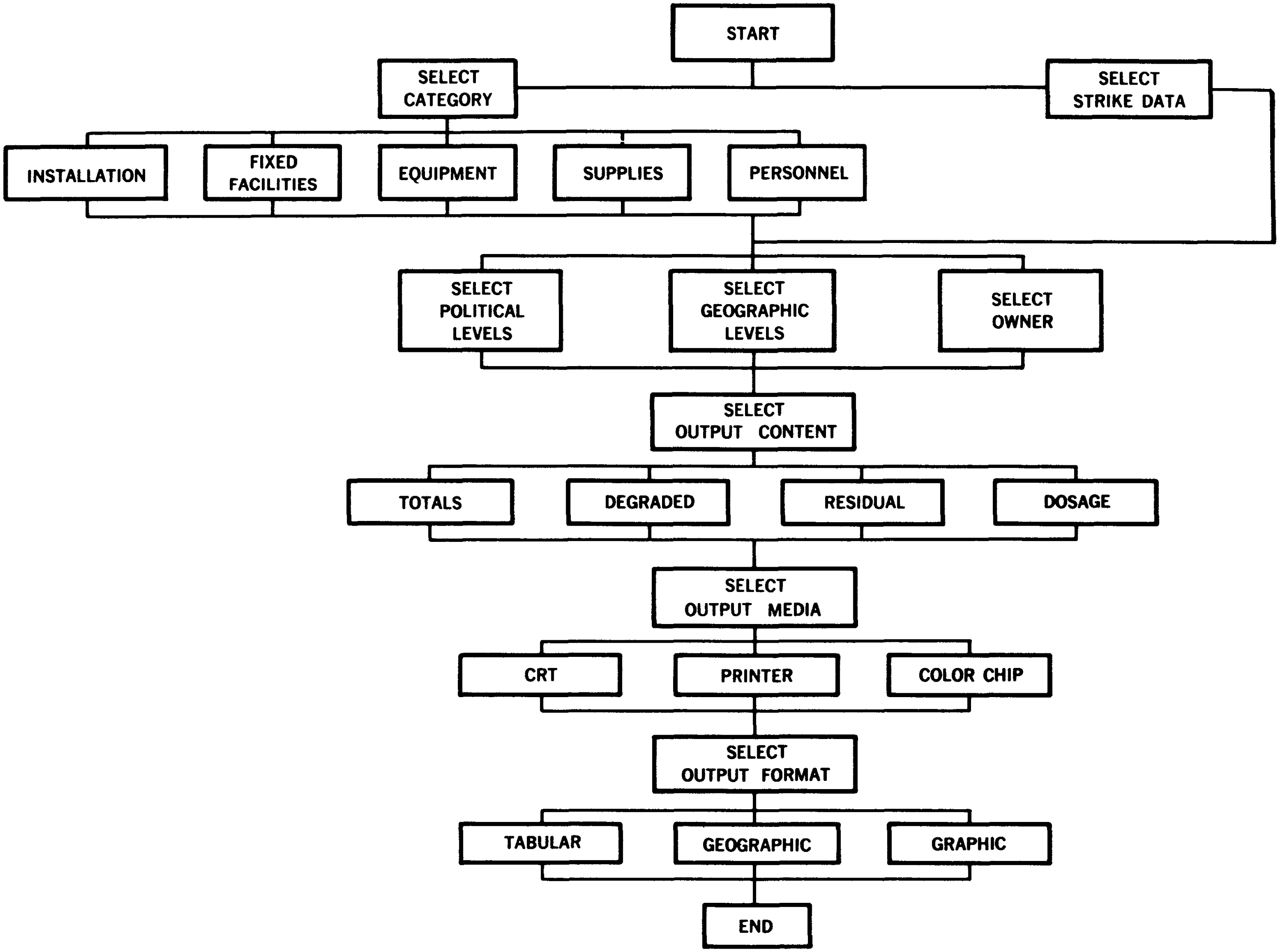


Figure 12. Sequence in Making Request for Data

PROJECT MERCURY REAL-TIME COMPUTATIONAL AND DATA-FLOW SYSTEM

A. The Role of Digital Computers in Project Mercury[1]

Saul I. Gass
International Business Machines Corporation
Federal Systems Division
Washington, D. C.

Introduction

Project Mercury (the man-in-space program of the National Aeronautics and Space Administration) has as its objectives: (1) the placing of a manned spacecraft in orbital flight around the earth; (2) the investigation of man's performance capabilities and ability to survive in a true space environment; and (3) the recovery of the spacecraft and man safely. Advance communications, guidance, computer, display and other subsystems have been integrated into the completed Project Mercury system. This paper will discuss and review the use of digital computers as an integral part of the real-time decision-making complex.

Computers are used throughout the Project Mercury mission to process observations made in the launch, orbit and re-entry phases and to supply a continuous, up-to-date record of the non-environmental status of the spacecraft. A series of processing programs which are united by a program monitor enable duplexed IBM 7090 computers at the Goddard Space Flight Center, Greenbelt, Maryland, to receive inputs from radars and computers at Cape Canaveral (during launch) and radars from the world-wide tracking system (during orbit and re-entry). The outputs of the system are transmitted to the Mercury Control Center at Cape Canaveral. This information enables the NASA flight controllers to make a GO or NO-GO decision during launch and indicates when the retro-rockets must be fired to safely bring the spacecraft down. In addition, an array of other orbital, location and time elements are displayed. An IBM 709 simplex computing system, which is located at the Bermuda Control Center, is used in a similar fashion to aid the Bermuda flight controllers to back-up the Cape Canaveral flight controllers.

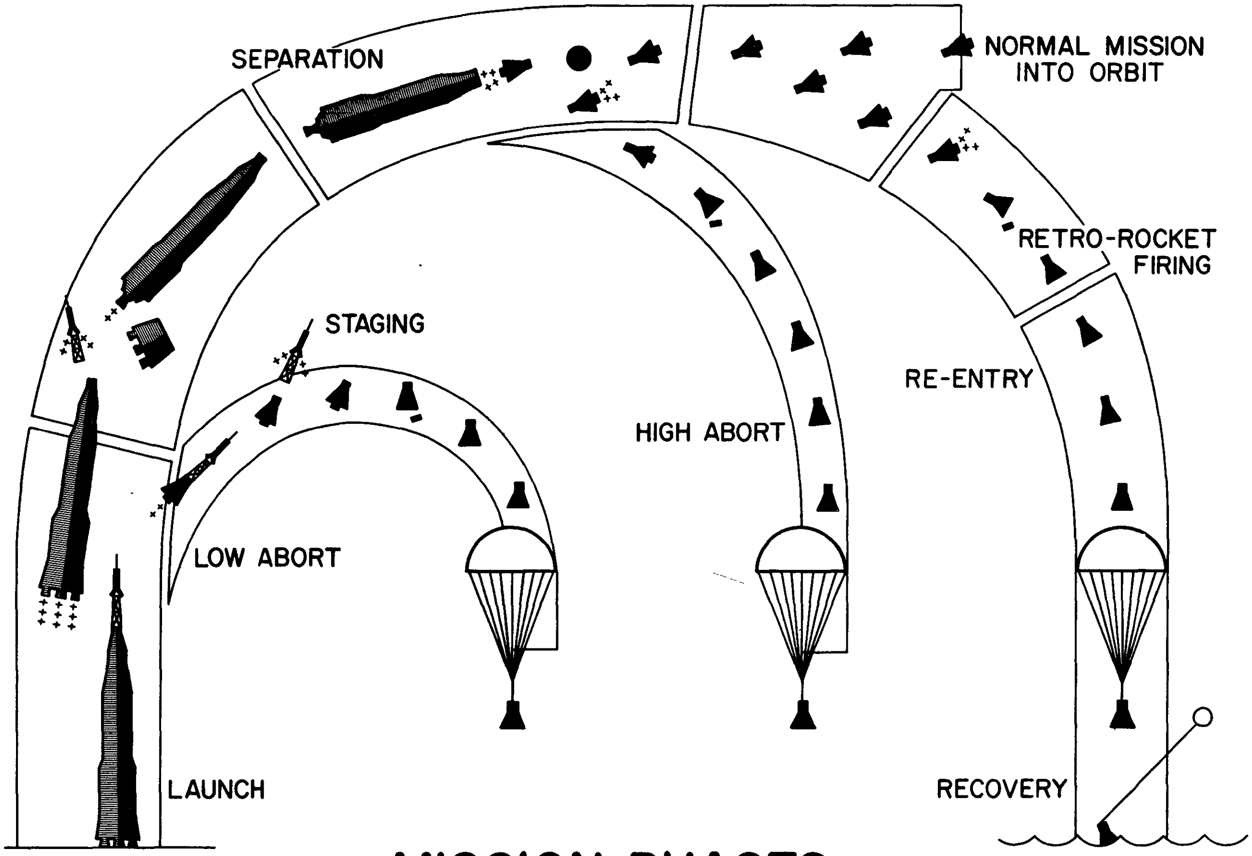
The computers at both locations are used to generate simulation techniques for both non-real-time and real-time simulation. These simulation procedures enable us to efficiently checkout the computer programming system and are also used in the training of the personnel responsible for making certain critical decisions.

As the Project Mercury range is truly a world-wide tracking system, it becomes imperative to have quick and accurate means of determining during a countdown what elements of the system are able to support a mission. A computer oriented procedure—CADFISS—(Computer and Data Flow Integrated Subsystem) has been developed which transmits requests for specified information to the sites, receives and analyzes replies from the sites and reports on the status of each element of the data flow subsystem. CADFISS is now a part of the Project Mercury countdown procedure.

[1] This paper is a summary of the work accomplished by the IBM Project Mercury staff. It represents the efforts of a highly skilled team of programmers, mathematicians, engineers and technical writers.

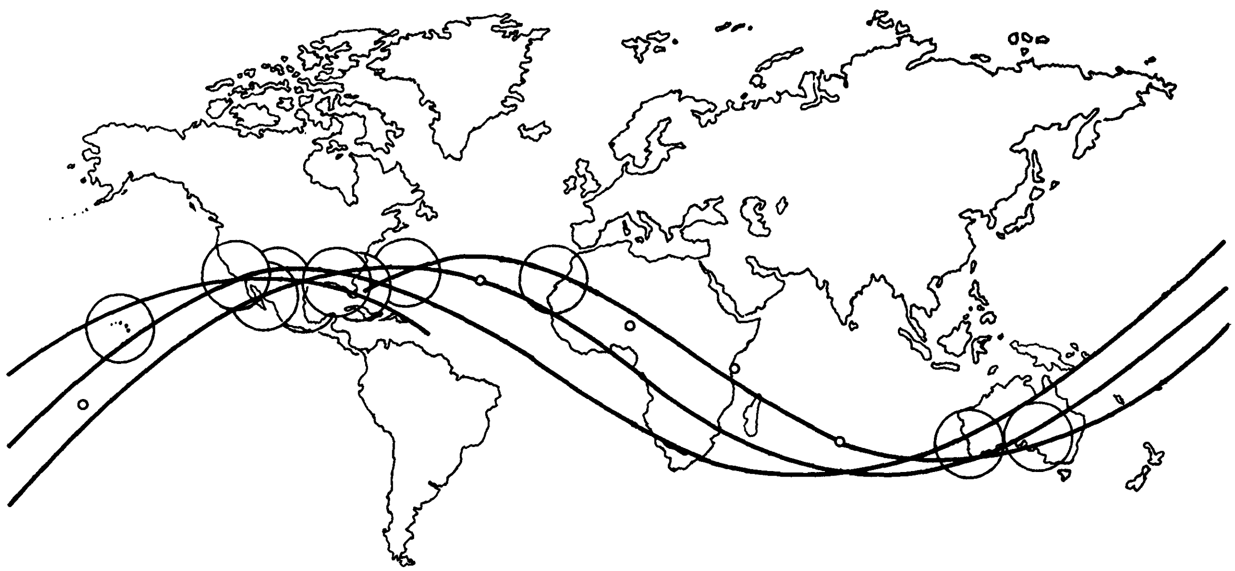
In order to emphasize the reliance of the Project Mercury mission on digital computers we shall first review the main aspects of a Mercury orbital mission (Figure 1). A typical launch phase is characterized by a set of discrete telemetry signals which are associated with corresponding physical functions of the booster and the spacecraft. For the purposes of this discussion, a successful launch phase is initiated by a liftoff signal, progresses through the booster engine cutoff and its detachment (staging), the separation of the escape tower and its rockets, the shutting down of the sustainer engine (SECO), the separation of the spacecraft, and the firing of three posi-grade rockets. This phase lasts approximately five minutes. Once in the orbital phase, the spacecraft will be allowed to circle the earth three times, after which three retro-rockets will be fired. This will cause the spacecraft to enter the re-entry phase and impact in a designated recovery area. Since emergency conditions may arise after the liftoff of the rocket, a number of escape procedures have been incorporated. The abort before staging is accomplished by the firing of the escape rockets (which lifts the spacecraft off the rocket so that within one second it is 250 feet away); while the abort after staging is done by the separation of the capsule and the normal sequence of firing the retro-rockets.

Once in orbit, the spacecraft will pass over the world-wide tracking and ground instrumentation system. The locations of the major radar sites and the path of a nominal three orbit mission is shown in Figure 2. It is a function of the computers at the Goddard Space Flight Center and Bermuda to recognize all of these phases and associated telemetry signals and to compute in real-time a variety of trajectory, present position, acquisition messages, impact and other decision-making elements throughout the length of a mission.



MISSION PHASES

Figure 1



TRACKING AND GROUND INSTRUMENTATION SYSTEM

Figure 2

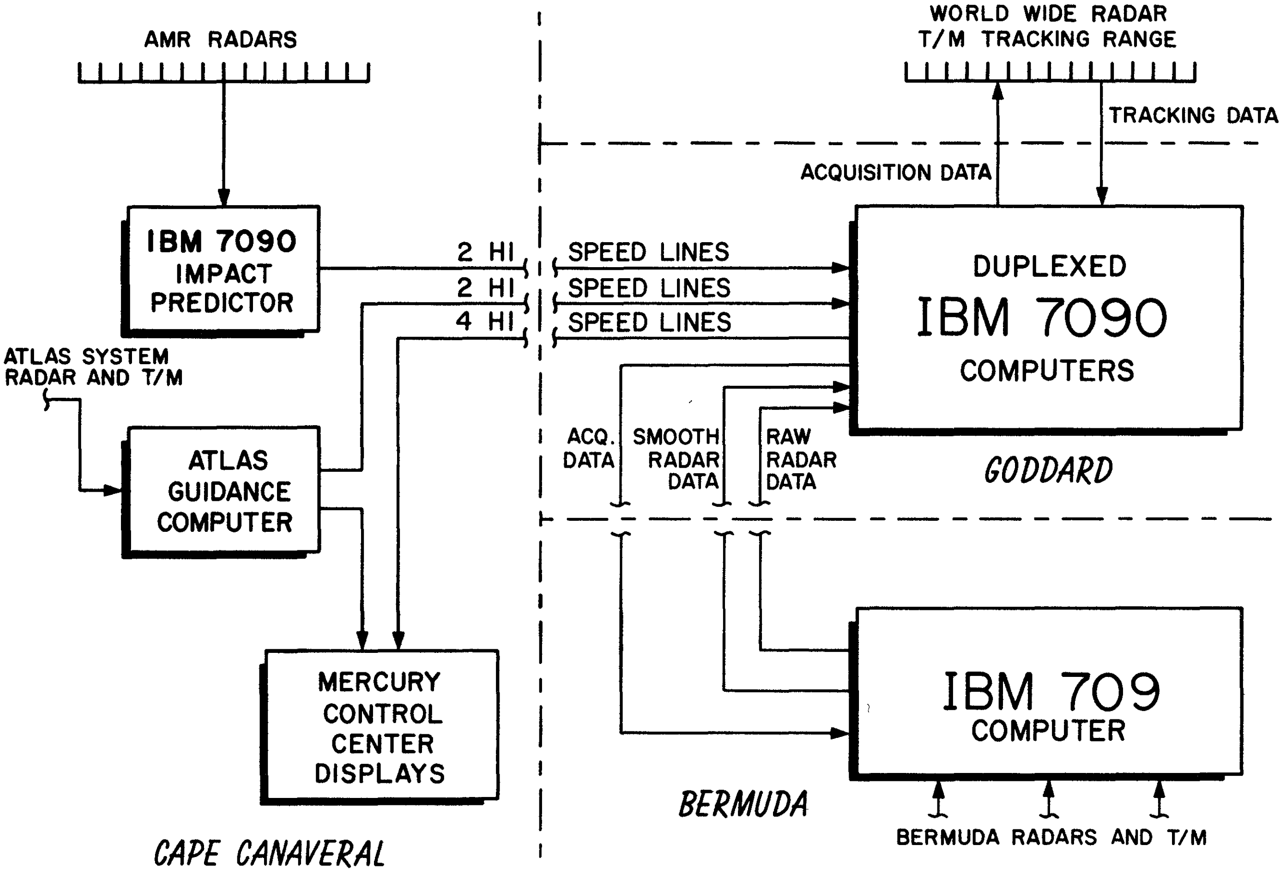
The Data Flow Subsystem[2]

The flow of data from the extremities of the world-wide tracking and ground instrumentation system to the computers and the Mercury Control Center is shown schematically in Figure 3. The link between the computers and this external data flow is accomplished by means of a Data Communications Channel (DCC), which is described in the paper by R. Hoffman and M. Scott, "The Mercury Programming System." During the launch phase of a mission the data sources feeding the Goddard 7090's are as follows:

Atlas-Guidance Computer

This special purpose computer, operating in conjunction with the radar and telemetry data received from the launch vehicle, tracks a beacon in the vehicle, checks its flight trajectory and generates the commands necessary to achieve optimum launch and insertion performance. As part of the launch phase guidance system, it plays an extremely critical role in Mercury mission launch sequence monitoring and control. The system is equipped to track the launch vehicle, not the capsule, therefore, it determines mission flight and speed profiles only until a few seconds after spacecraft separation occurs. During launch the computer converts raw tracking values into computed position and velocity vectors (geocentric inertial coordinate system redefined in time at each processing cycle) describing the launch vehicle's trajectory. These quantities, and time-of-transmission notices, are sent continually in real-time (one observation each 1/2 second) over two high-speed 1000 bits per second data circuits to the Goddard Space Flight Center computers, where they are processed to obtain pertinent information for display and mission control purposes. Certain values derived from capsule and vehicle telemetry signals, and from other sources, are added to the tracking message to Goddard by time-multiplexing them onto the high-speed lines. These quantities, called discretes, are contained in a message word which indicates the status of mission subphases (liftoff, booster

[2] The equipment features are discussed in detail in the paper by Hamlin and Peavey, "Mercury Launch Monitor Subsystem."



LAUNCH SUB-SYSTEM LOGIC

Figure 3

engine cutoff, sustainer engine cutoff) and which includes data flags as indicators of the quality and status of the calculations. A checksum indication is transmitted to validate message accuracy and data bit totals.

Impact Predictor 7090

This IBM 7090 computer, located in the IP 7090 building at Cape Canaveral, utilizes Azusa beacon and Cape Canaveral, Grand Bahama Island or San Salvador Island raw radar inputs of range, azimuth, elevation, and time of observation. The IP 7090 furnishes processed data from either the Azusa or AN/FPS-16 tracking systems, depending upon which data is chosen during the launch phase. From this real-time processing of raw radar quantities the IP computer produces position and velocity vectors describing the trajectory of the launch vehicle. Based on geocentric inertial coordinates, the position and velocity rates are redefined at each computing interval, time tagged with range time and identified by site and radar. IP 7090 messages received every 0.4 seconds by the Goddard computers contain calculated position and velocity vectors, telemetry information from both the launch vehicle and the spacecraft, and checksum parity notations to validate transmission accuracy. These frames are also transmitted at a rate of 1000 bits per second to Goddard over duplex communication lines.

Radars

Radar facilities of the pre-existing Cape Canaveral downrange complex serve as prime sources of data during the early stages of a Mercury mission. During launch, each site takes range, azimuth and elevation measurements on the launch vehicle/spacecraft, referenced to that particular radar installation. These values are sent by a high-speed real-time data transmission system to the IP 7090 (Impact Predictor 7090) building at Cape Canaveral where they are accepted as inputs for processing by the IP 7090 computer. As noted above, the computer-smoothed position/ time/velocity coordinates are interleaved with telemetry information, a time tag and a site identification note onto the IP 7090 high-speed message to the Goddard Center. In addition to being routed to the IP 7090 for processing and subsequent retransmission to Goddard, raw radar data from the supporting downrange stations can also be manually selected and transmitted at a ten-messages-per-second rate directly to the Goddard 7090's. In this case, raw radar quantities rather than smoothed coordinates are identified by site and combined with telemetry data. The procedure is exactly the same as for impact predictor-processed data; the IP 7090 is merely bypassed in the latter case. If AN/FPS-16 raw radar is selected instead of IP 7090 information, it is accepted, edited and smoothed by the Goddard 7090's to maintain the overall computing cycle.

Telemetry

Telemetry (TM) signals from the launch pad, the launch vehicle and the spacecraft indicate the status of mission subphases and certain critical events which happen or fail to happen, during a Mercury flight. These signals, picked up by telemetry receivers and converted to forms useful for display interpretation and as inputs to the Goddard IBM 7090 computers are extremely important indicatros of events and conditions regarding the spacecraft and the vehicle. Real-time telemetry quantities continuously transmitted directly to the Goddard IBM 7090 computers include spacecraft clock elapsed time and retrofire mechanism-setting readings, liftoff, staging, escape tower released from sustainer, sustainer engine cutoff and one, two or three posigrade rockets fired. These quantities combine to a total of 72 bits of telemetry data and they are packed in each frame of data (Atlas-Guidance, IP 7090, or raw radar). The 72 bits include a parity bit and the transmission in triplicate of critical signals (e.g., liftoff signal). During launch each observation (i.e., a frame of data from any source) made at Cape Canaveral is transmitted to Goddard, processed by the Goddard computers, and the results displayed back at the Cape Canaveral control center in approximately one second.

Once the spacecraft is inserted into its orbit the data to the Goddard computers is developed by the radars of the world-wide tracking network. During each pass of the capsule over a radar site, range, azimuth, elevation and time of observation readings are taken for as long as the spacecraft remains within range. The information from the radar is processed through conversion equipment at the site and transformed into a teletype format for transmission to Goddard at a rate of six (6) characters per second into the computer. A similar data flow is maintained during the re-entry phase of the flight.

A means of manually inserted critical data signals is also provided. By paper tape such information as exact time of liftoff (which is determined at Cape Canaveral) and the exact time of retro-rocket firing (which is determined by telemetry signals originating in the spacecraft) can be fed into the Goddard programs. The Goddard duplex 7090 computing system is shown in Figure 4.

The data flow into the Bermuda 709 computer is shown in Figure 5. The major task of the Bermuda programs is the calculating of position and velocity values during the flight's launch, insertion and initial orbital phases. This is accomplished by the analysis of data generated by both AN/FPS-16 and Verlort radars which are fed directly into core memory by a DCC.

Each radar maintains a flow of ten observations per second and the observations are combined with telemetry signals which originate in the spacecraft. Manual insertion of important data into the Bermuda computer is accomplished by means of a paper tape reader. The Bermuda program also smooths the radar observations of the spacecraft during launch and transmits these readings directly to the Goddard computers via teletype. Hence, the Bermuda computer also acts like a "refined" radar site as well as being a tool for supplying a GO or NO-GO answer to the question of orbital achievement of the spacecraft.

The Goddard Computer Programs

In all phases of the Mercury mission it is extremely vital that the large amounts and many forms of necessary calculations be performed with exact precision, and the data made available almost instantaneously. The

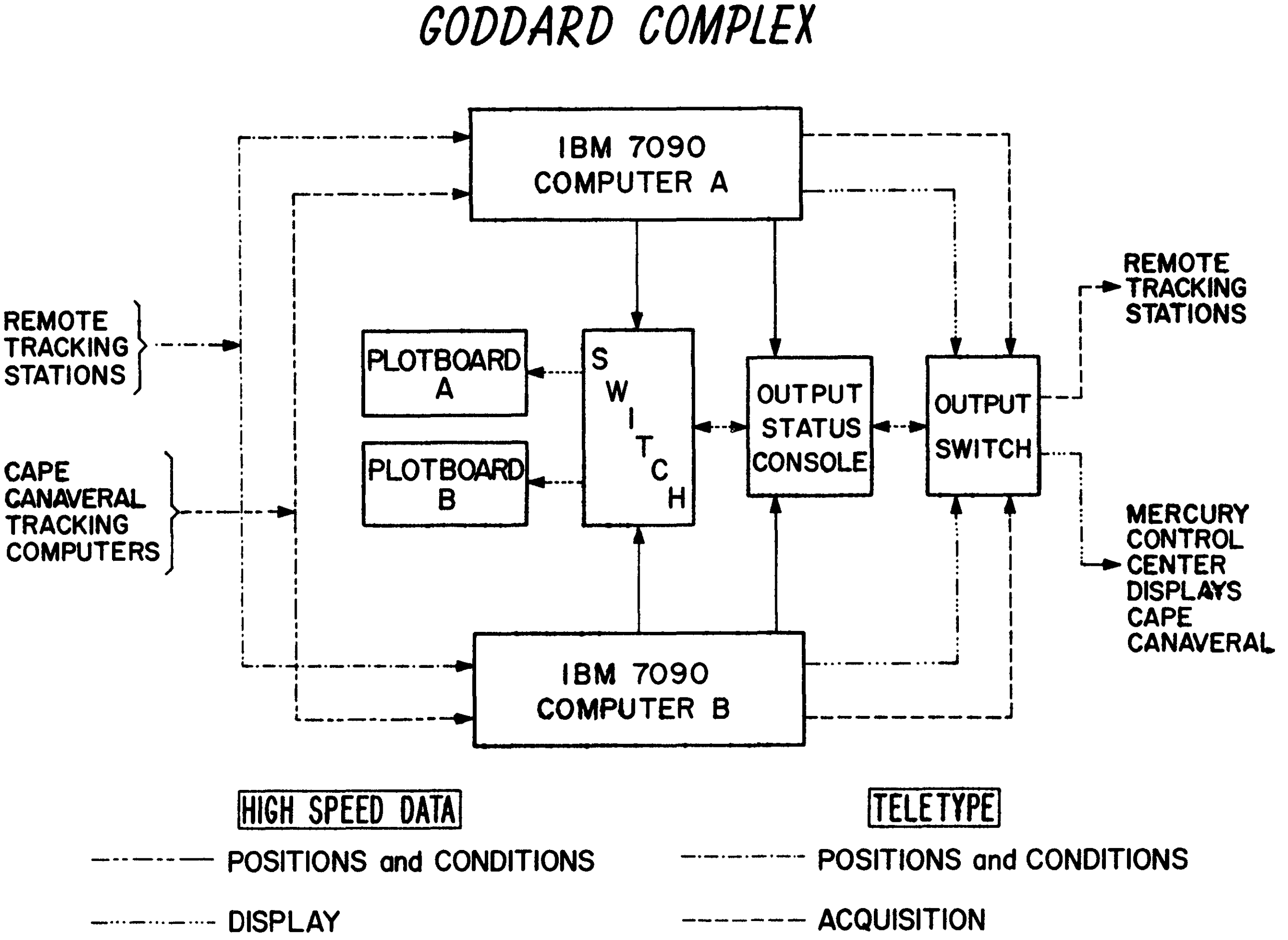


Figure 4

specific computational approach taken is of critical importance to the safety of the astronaut and the success of the project. The real-time computer requirements, the stringent accuracy and reliability necessary make the project one of the most demanding computer problems ever undertaken.

The placing of a manned spacecraft into orbit and the successful re-entry and safe recovery of the vehicle and its human occupant poses—under normal or emergency conditions—many severe computational problems. For example, in an extremely short period after the capsule is separated from the missile the computers must furnish data for evaluating whether or not the mission is to be allowed to continue. Since the time required for the spacecraft to perform a post-insertion 180-degree attitude rotation is necessarily limited, and because of the obvious desirability of a water landing following high level abort, the initial orbital parameters must be calculated extremely rapidly, and as much data as possible used to increase the reliability of computation.

As described above, Project Mercury is equipped with an array of interconnecting computational complexes, data collecting equipment, special supporting equipment and display devices to accomplish mission computing functions in the most complete, efficient manner possible. The core of the computing system is the duplexed Goddard IBM 7090 computer facility, complemented by the Bermuda station's IBM 709 computer.

The Goddard computers and special input/output equipment are duplexed to promote maximum reliability, validity of information and mission safety. Both computers accept the same input data and perform the same computations. Output status displays at Goddard enable the performance of each 7090 to be monitored. On the basis of this comparative information the output of one of the

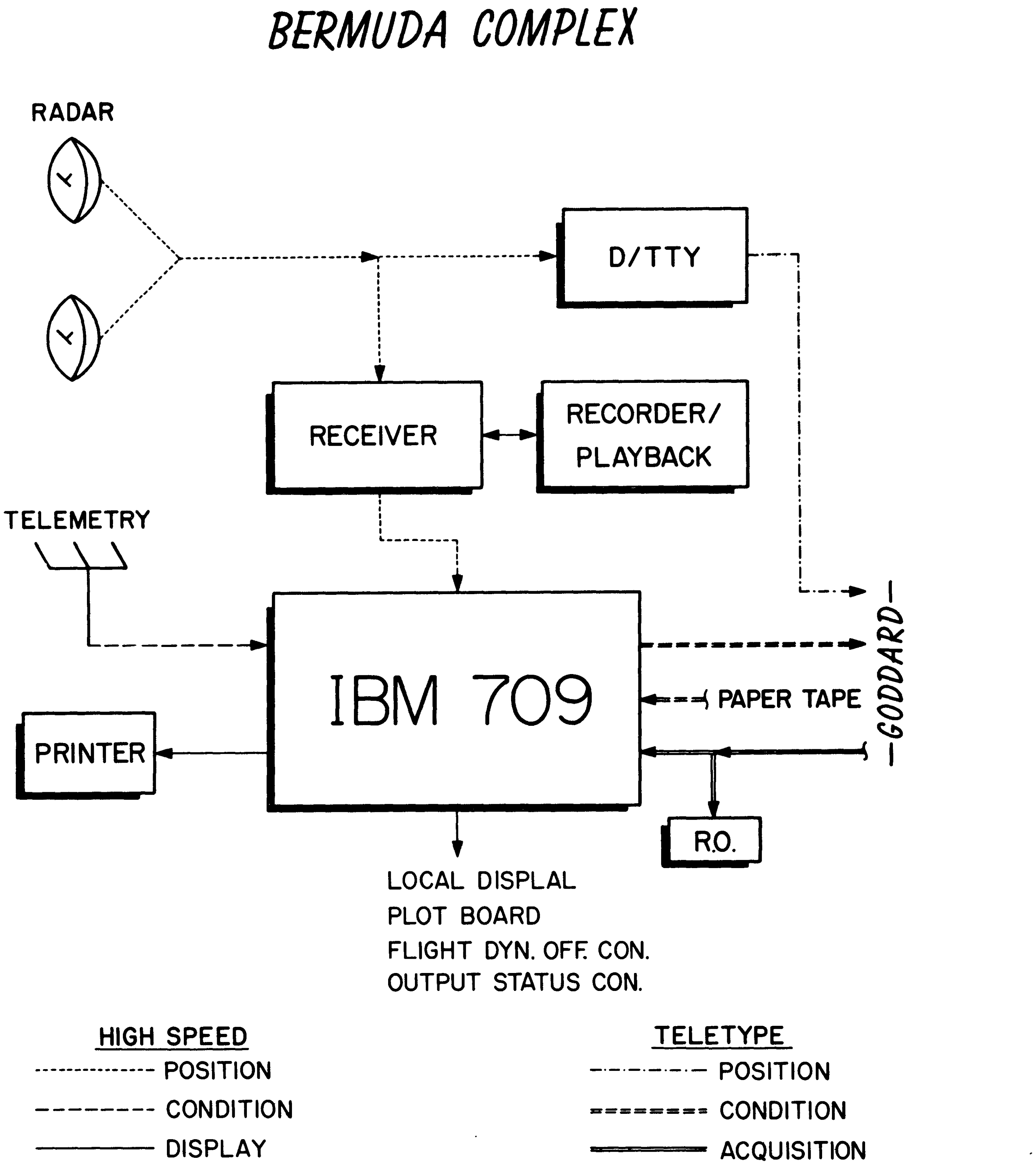

BERMUDA COMPLEX
RADAR
D/TTY
RECEIVER
RECORDER/ PLAYBACK
TELEMETRY
IBM 709
PRINTER
PAPER TAPE
GODDARD
R.O.
LOCAL DISPLAL
PLOT BOARD
FLIGHT DYN. OFF. CON.
OUTPUT STATUS CON.
HIGH SPEED
POSITION
CONDITION
DISPLAY
TELETYPE
POSITION
CONDITION
ACQUISITION

Figure 5

two computers is selected for transmission to the Mercury Control Center and radar sites.

In brief, Goddard computers' basic Mercury mission responsibilities are to:

perform launch calculations during capsule insertion and advise the Mercury Control Center of capsule dynamics;

compute acquisition data, orbit parameters and capsule present position, future position and life-time;

compute and transmit to the control center retrofire clock information and compute and transmit to the control center and recovery teams the probable impact point and impact time;

monitor, quantitatively, radar and computer performance and advise the control center of deviations from acceptable performance.

There are three generally distinct computational phases: launch (powered flight), orbit, and re-entry (descent). The programming system for all phases has been integrated into one automatically sequenced package. Upon receipt of liftoff signal over the high-speed communications the programs are activated (prior to this they are in the passive, but vigilant mode) and the launch computation begins. If the computer recommends GO and if the Cape Canaveral flight controller has no external reasons (e.g., aeromedical data) to abort the mission, the launch switch is thrown and a discrete signal is sent to the computers. The orbit program is then brought into the computer and this program is given the insertion conditions calculated by the launch program. In a similar fashion, upon receipt of manual (hence, positive) insertion of actual retrofire time, the program automatically shifts to the re-entry mode. As indicated below, different outputs are required for each phase. The abort during launch or a NO-GO decision by the flight controller after power-cutoff is also signaled by appropriate telemetry discretes. The phase of the computing program is then automatically altered. The data processing functions undertaken at Goddard during each of the phases are as follows:

Launch

The primary function of the Goddard computers during this first flight phase is to determine the GO NO-GO recommendation to continue the mission. The IBM 7090's decide whether or not a successful orbit has been attained at the time of insertion into orbit by computing the projected orbit lifetime; pertinent trajectory parameters are presented so launch status may be monitored for indications of a possible impending abort.

In the event of a launch abort, the computers determine the times retrofire must occur for the capsule to land in one of the several designated recovery areas. If the retrorockets are fired, the computers calculate spacecraft trajectory and the resulting impact point. Refined impact point data for several abort procedures is computed.

During the launch phase most data inputs to the Goddard IBM 7090's originate at the Cape Canaveral launch site. Launch data is, essentially, position and velocity information in forms not useful for evaluation at the control center. The Goddard computers operate on launch data in real-time, convert it into forms more suitable for decision-making and transmit this information back to the control center.

Orbit

Calculating the exact time to fire the retrorockets to land the spacecraft at the desired impact point is the major contribution which the IBM 7090 computers make in the orbit phase. The times at which the retrofire sequence must be initiated for a landing in a prescribed normal-mission recovery area, for re-entry at the end of any given orbit, and for landing at the specified emergency recovery points along the spacecraft's path are recomputed by the 7090's after each new set of orbit parameters is established.

In the orbital phase, data is collected and dispatched from the world-wide network of radar sites as the spacecraft passes over each station. Position/time quantities gathered from these radars are sent almost instantaneously to the computing and communications center at Goddard. In addition, to the primary purpose of determining retrofiring time, this position data is used at Goddard to refine orbit parameters from the previous set of orbit parameters (and, at initial entry into orbit, from those parameters estimated during launch), thus enabling spacecraft position to be accurately predicted for further radar acquisition. Acquisition messages are sent to each site from Goddard prior to the spacecraft's next pass over that site.

The mathematics of orbital computer processing involves solving by numerical methods the formulas derived from the basic Newtonian equations of motion. The numerical methods employed include Cowell's numerical integration for extrapolation and correction of orbital parameters, and the differential correction, conversion and partial coefficient calculation programs. To minimize the effects of radar values from stations whose input data is considered (on the basis of past results) less reliable, each site is weighted accordingly.

Re-entry

The main purpose of the Goddard IBM 7090's during re-entry is the all-important computation of impact point. The computers pinpoint the point of spacecraft impact from position/time data received from range stations whose radars follow the spacecraft during this period. Refined impact point information is provided to guide the Mercury recovery effort whenever new radar observations are transmitted to the computers.

Re-entry trajectory acquisition data is provided to remote sites from the Goddard center during this phase in the same manner as acquisition information is provided during orbit.

The real-time outputs of the Goddard computers are sent directly to a variety of digital and graphical displays at the Cape Canaveral Mercury Control Center. The following summary of the Goddard computer outputs will give the reader a feeling for the volume of data presented to the flight controllers and the type of information required to maintain a ground-based control center:

Plotboard 1

During launch, spacecraft flight path angle versus the ratio of inertial velocity to required velocity is shown. Three different scales marked with appropriate limits are employed as board overlays during launch. The difference between spacecraft radius from the earth versus spacecraft inertial velocity is presented during abort, orbit and re-entry.

Plotboard 2

Two plots are presented during the launch period. Spacecraft altitude versus downrange distance is charted on the bottom of the board (this quantity is also displayed in the event of a launch abort); crossrange deviation versus downrange distance is displayed simultaneously on a separate scale on the upper portion of the overlay. Two plots appear during orbital flight. Spacecraft altitude above an oblate earth, as a function of elapsed time, is depicted on the lower part of the board; on the upper portion of the board is displayed the difference between the semi-major axis of the orbit and the average radius of the earth, as a function of elapsed time. Displayed on board 2 during a normal re-entry is spacecraft altitude, as a function of time.

Plotboard 3

A variety of parameters is displayed during launch and abort. However, quantities furnished to the plotboard during this phase come directly from the Atlas-Guidance computer. Board 3 displays two plots during orbit. On the lower portion of the board appears longitude of perigee as a function of elapsed time and on the upper portion, eccentricity as a function of elapsed time. This plotboard is not used during re-entry.

Plotboard 4

Displays impact point computations made at Goddard during a normal launch and in the event of an aborted launch. The board's overlay is a map of the Atlantic Ocean area showing land and water masses and recovery areas. Plotted during launch are the latitudes and longitudes of impact point assuming that (1) the retrorockets will fire in 30 seconds (minimum delay) and (2) at 450,000 feet (maximum delay). Board 4 displays two items during orbit: the latitude and longitude of present spacecraft position over the Western hemisphere and the computed impact point for retrofire in 30 seconds. During abort and re-entry, spacecraft present position and the predicted impact point are shown.

Wall Map

Located on the observers' area of the Mercury Control Center is a large (50 feet long) wall map of the world. It illustrates the locations of all Mercury range stations and the ground track of the capsule. As the Mercury mission progresses, a miniature lighted capsule moves along the track, indicating the actual present position of the spacecraft. A small, moving light ahead of the spacecraft indicates, during abort and re-entry, the latitude and longitude of the refined impact point, or during orbit, impact point if the retrorockets are fired in 30 seconds.

Strip Chart

A high-speed multichannel (six recording pens) strip chart (also called the Data Quality Monitor) is used at the control center only during the launch phase. The device enables an operator to determine which sets of information are most valid for presentation to Mercury display equipment. The six pens are driven by data from three sources (Goddard processed Atlas-Guidance or IP 7090 data, or Atlas-Guidance direct), each charting two items of information. The displayed quantities are:

the difference between flight path angle and nominal flight path angle;

the difference between velocity ratio and nominal velocity ratio.

Digital Displays

Computed information from the Goddard IBM 7090's is routed at high speed to control center digital displays located variously on the Flight Dynamics Officer's Console, the Retrofire Controller's Console, the Recovery Status Monitor Console and a wall digital display.

Digital values are sent to the display register twice per second during the launch phase. To avoid flicker, however, the quantities are updated by the computer only once each second. All digital display messages are transmitted from Goddard at a frequency of 10 per minute during orbit. During re-entry, data quantities are routed from Goddard to the Cape at a 20-per-minute rate.

Flight Dynamics Officer's Console

Six digital displays are located on this important control console.

Display 1 indicates during launch and abort the GO NO-GO recommendation of the Goddard computers to continue or abort the mission;

Display 2 indicates spacecraft altitude in nautical miles and tenths of nautical miles during the entire mission, from launch to impact;

Display 3 indicates flight path angle in degrees, tenths and hundredths of degrees during launch and re-entry, and denotes apogee height in hundredths and tenths of nautical miles during orbit;

Display 4 indicates spacecraft inclination angle in degrees and tenths of degrees during launch and orbit;

Display 5 indicates orbit lifetime remaining from the time of the last pass over Cape Canaveral;

Display 6 indicates the ratio of spacecraft inertial velocity to required velocity during launch. After insertion into orbit, for the remainder of the flight, display 6 indicates actual spacecraft velocity.

Retrofire Controller's Console

Located on this console are nine digital displays. All time displays are represented in hours, minutes and seconds.

Display 1 indicates during abort and orbit the Greenwich Mean Time (GMT) to retrofire to land in an emergency abort area;

Display 2 presents during launch from 20 seconds after staging, the computed GMT to retrofire in the next recovery area, and during abort and orbit, the elapsed capsule time for retrofire to land in the next recovery area;

Display 3 indicates during orbit the GMT of retrofire computed for the end of the present orbit;

Display 4 indicates during orbit the elapsed spacecraft time for retrofire to land at the end of the present orbit;

Display 5 displays during orbit the GMT to retrofire to land in the normal three-orbit-mission impact area;

Display 6 indicates during orbit the elapsed spacecraft time to retrofire computed for the normal impact area following a three-orbit flight;

Display 7 presents during orbit the GMT of retrofire based on the present spacecraft setting, and displays during re-entry the elapsed ground time since retrofire;

Display 8 indicates during abort and orbit the incremental spacecraft time for retrofire computed for the next emergency recovery area;

Display 9 indicates for the entire mission the number of the orbit, and the presently designated recovery area.

Recovery Status Monitor Console

The digital displays located on this console indicate: the computed GMT of impact in hours and minutes during abort, orbit and re-entry; the computed longitude and latitude in degrees and minutes for the abort landing point (during abort), the normal end-of-mission impact point (during orbit) and the refined impact point (during re-entry).

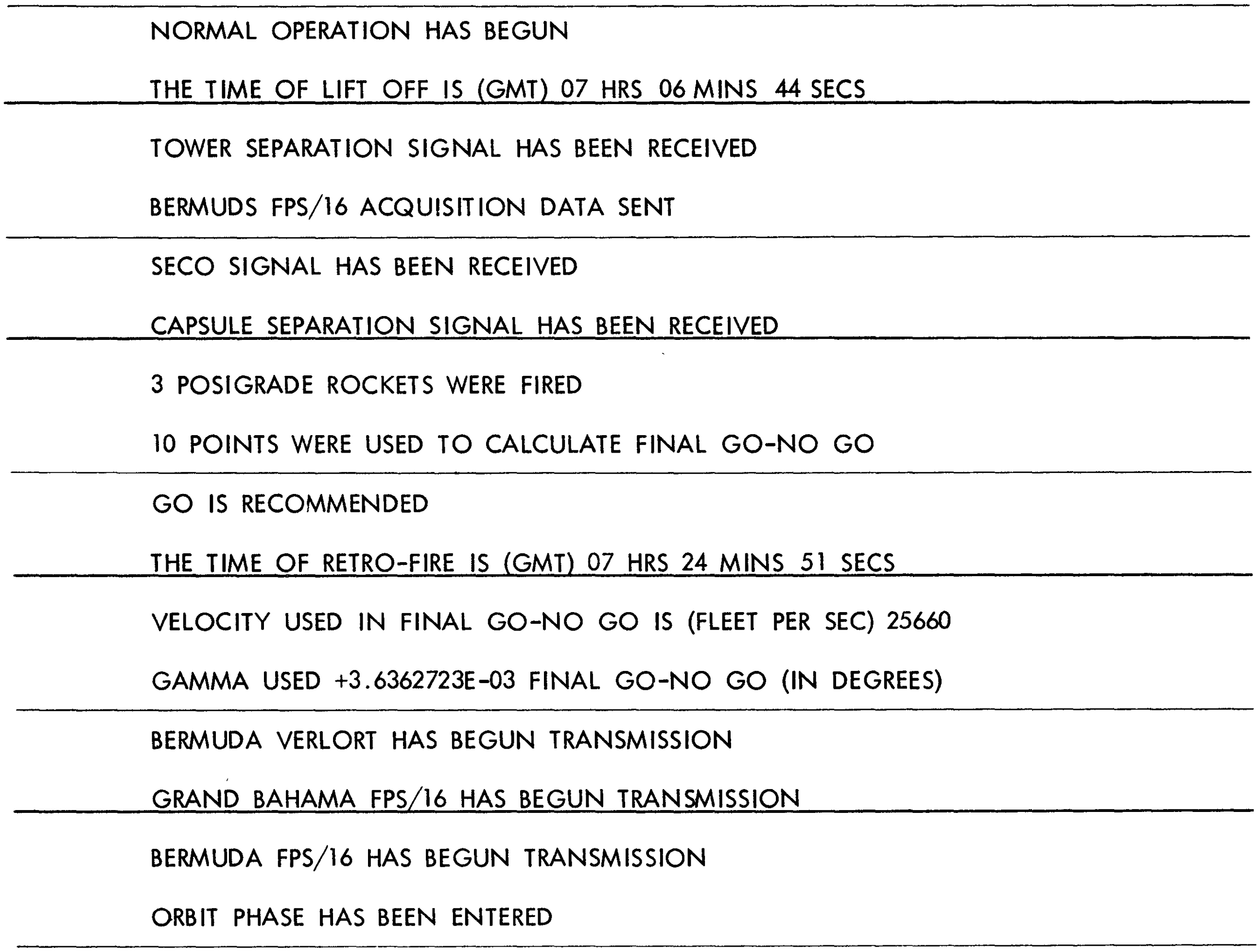
Wall Digital Display

Displayed in GMT hours, minutes and seconds during the launch and abort phases in the ground time remaining until retrofire. During re-entry this display indicates the ground time remaining until impact. Also presented on the wall display during orbit is the current orbit number.

In addition to the displays at the Mercury Control Center, the Goddard computers also drive certain plotboards at Goddard. Each computer is connected to a single plotboard and reproduces one of the plots sent to the Mercury Control Center. Much of this data is, however, printed on-line. A typical print-out (edited for space purposes) is shown in Figure 6.

The Bermuda Computer Program

A secondary computing station to supplement the Goddard/Cape Canaveral complexes is the IBM 709-equipped Bermuda site. The role of Bermuda in the overall computing picture is essentially twofold: Bermuda serves as an extension and backup for the control center, and as an ordinary range station. As a backup to Cape Canaveral, the Bermuda 709 computer is used to determine whether or not the spacecraft has entered an acceptable orbit at insertion; command an early re-entry, if necessary, and verify



```
NORMAL OPERATION HAS BEGUN

THE TIME OF LIFT OFF IS (GMT) 07 HRS 06 MINS 44 SECS

TOWER SEPARATION SIGNAL HAS BEEN RECEIVED

BERMUDS FPS/16 ACQUISITION DATA SENT

SECO SIGNAL HAS BEEN RECEIVED

CAPSULE SEPARATION SIGNAL HAS BEEN RECEIVED

3 POSIGRADE ROCKETS WERE FIRED

10 POINTS WERE USED TO CALCULATE FINAL GO-NO GO

GO IS RECOMMENDED

THE TIME OF RETRO-FIRE IS (GMT) 07 HRS 24 MINS 51 SECS

VELOCITY USED IN FINAL GO-NO GO IS (FLEET PER SEC) 25660

GAMMA USED +3.6362723E-03 FINAL GO-NO GO (IN DEGREES)

BERMUDA VERLORT HAS BEGUN TRANSMISSION

GRAND BAHAMA FPS/16 HAS BEGUN TRANSMISSION

BERMUDA FPS/16 HAS BEGUN TRANSMISSION

ORBIT PHASE HAS BEEN ENTERED
```

Figure 6

receipt; and notify the control center and recovery teams of such action.. When instructed to do so by the control center, or if communications with the control center fail or become poor, Bermuda can command an abort, an early re-entry or assume full control of the insertion period.

As an ordinary range station Bermuda accepts high-speed radar data into its computer and converts this information into orbital and trajectory parameters. At each orbital passage over Bermuda the IBM 709 computer processes the high-speed radar information and sends smoothed data to the Goddard center.

Trajectory parameters acquired by the local radars are used by the Bermuda computer to fulfill the following responsibilities immediately after insertion.

Determine if the spacecraft's orbit is acceptable. This is accomplished by determining from the radar data received after insertion whether or not orbit lifetime will exceed one, two or three revolutions; in the event of an abort, determine the times at which the retrorockets must be fired to land the spacecraft in one of the designated recovery areas; determine refined impact points for several abort procedures; determine orbit characteristics; provide quantities required to drive plotboards and displays at Bermuda; send post-insertion conditions to Goddard.

The Bermuda outputs are not as great in volume and types as Goddard. However, they are sufficient enough to enable the flight dynamics officer to recommend a GO or NO-GO. The displays are summarized below.

Flight Dynamics Officer's Console

A GO- NO-GO recommendation utilized during launch and abort to indicate the present computed status of spacecraft insertion; the Greenwich Mean Time (GMT) to retrofire, in hours, minutes and seconds if an abort is called for; the actual time of retrofire is presented during re-entry; the elapsed capsule time to retrofire, in hours, minutes and seconds during abort and the elapsed spacecraft time since retrofire during re-entry; spacecraft altitude in nautical miles and tenths of nautical miles; flight path angle in degrees, tenths and hundredths of degrees; velocity ratio (inertial velocity/required velocity) to four decimal places; after retrofire, inertial velocity is presented; during abort the incremental change in time of retrofire; the orbit number and next recovery area during abort and re-entry.

Plotboard

An overlay on the board depicts the Atlantic Ocean and peripheral land areas from Florida to Africa. Real-time information plotted on the overlay includes:

Prior to retrofire: latitude and longitude of impact point for immediate retrofire; flight path angle versus velocity ratio.

After retrofire: latitude and longitude of spacecraft present position; latitude and longitude of impact point.

Monitor Program

During a Mercury mission, many functions compete with one another for computer time. At any moment a given computation must be satisfied without sacrificing any others which may be critical at that time. In other words, the computer is required to deliver several competing quantities continuously on rigorous schedules. An output which fails to meet its schedule becomes worthless for real-time applications.

Producing a virtually real-time output according to strict schedules is a difficult task—even if incoming data arrives smoothly, adhering to plan. The nature of Mercury computation in real-time at both Goddard and Bermuda makes it mandatory that computer processing be automatically controlled, with perhaps a limited amount of manual intervention possible, if desired.

To solve Mercury data processing problems most reasonable, and to effect the real-time calculations which are vital to a mission, the control and coordination of all other programs in Mercury computing is assigned to a single control program—Monitor.

Mercury system operational programs must accomplish three tasks which are all fairly synchronous in time: they must accept input data which arrives on an asynchronous schedule; they must perform certain computations on the input information; and they must provide output quantities at specified time intervals. To meet these three requirements and ensure a smooth, overall Mercury data processing effort, the Monitor control program "supervises" the processors, coordinating computation according to the arrival of input data, calculations to be performed and output quantities needed. Monitor establishes the constantly-changing processing priorities on the basis of input, computational and output conditions.

A description of Monitor operation is given in the paper by Hoffman and Scott.

Simulation Techniques

The Project Mercury simulation efforts cover the broad requirements of non-real-time and real-time simulation procedures. These procedures have enabled us to checkout the operational programming system with a set of controlled experiments which reflect a wide variety of possible real-life situations and data degradations. They also have provided a means of training the flight controllers to recognize and cope with a number of emergency conditions as well as the nominal. Real-time simulations are accomplished by the playing of a launch trajectory tape containing pre-determined data from the Atlas-Guidance and IP 7090 computers at Cape Canaveral, (this tape is prepared at Goddard), and the transmission of associated telemetry discretes to the Goddard computers. The Goddard complex and programs react as if a rocket was launched and provide a full set of displays. Once the orbit phase is entered, the world-wide tracking sites transmit radar observations to the Goddard computers at designated times. The observations are on paper tape and were prepared at Goddard to match the launch conditions. A number of three orbit simulations have been run. Real-time simulations have become an integral part of the range countdown and checkout procedures. A full description of the Project Mercury simulation procedures is given in the paper by Green and Peckar, "Real-Time Simulation in Project Mercury."

CADFISS - (Computer and Data Flow Integrated Subsystem)

As Project Mercury was the first to utilize a world-wide tracking and communications network, it became quite imperative to have a means of determining which elements of the network (which includes the flow between the Goddard computers and Cape Canaveral and the Goddard and Bermuda computers) are "green" and are able to support an actual mission or a simulation. It became apparent that the most efficient and automatic way of accomplishing this task was to make the Goddard computers the driving force behind any such checkout procedure. The basic features of CADFISS are as follows:

One-at-a-time Test

In this test each site, e.g., the radar at Canary Islands communicating with the 7090's, is independently and thoroughly checked. The Goddard computers are programmed to send a request message (cue) to the site, directing site personnel to transmit specified data to Goddard. The computer analyzes the data transmitted by the site and produces a report for supervisory personnel indicating the quality of the data. The computer then sends the next cue to the site. Each cue specifies a test of a particular part of the system. All sites in the Mercury network can be tested simultaneously since cues are addressed to the specified site. A typical type of test is the Boresight Acquisition test. These tests demonstrate the ability of each radar to acquire a target within its raster search pattern, to originate correct data for that target, and, having so attained the target, to act as a source of angle tracking data for other antenna pedestals. Each radar will acquire its boresight tower and then be selected as the source of tracking data for all other antenna pedestals. After receipt of a cue from the computer, the radar's output will be transmitted to Goddard. The computer-generated cue for transmission of selected pedestal (indicator) readings will be delayed until after later tests to allow time for preparation of teletype tapes. Following receipt of this cue, each message will be transmitted twice to permit recognition of transmission errors. The radar data and pedestal readings transmitted to the Goddard computers will be automatically compared against the surveyed values for azimuth and elevation, and a summary report generated.

Roll-Call Test

This is an abbreviated one-at-a-time test designed to check each Goddard-computer-related portion of the Mercury system in rapid sequence to determine the readiness status of these elements.

The basic program of this system sends cues simultaneously to the Mercury sites and evaluates the responses received. (The site is expected to reply to a cue within a fixed time limit after the receipt of the cue or it fails the test.) The teletypewriter data evaluated consists of message patterns, radar boresight and range target data, and radar data from pointings in several critical directions. The data received by the computer is compared against the predetermined message or survey positions. In addition, the program sends high-speed patterns to Cape Canaveral and receives and evaluates similar data originating at the Atlas-Guidance and IP 7090 computers. Finally, it evaluates high-speed boresight data from the Cape radars. The evaluations consist of comparing expected data to that received and the periodic on-line printouts of up-to-date summaries of the tests. In addition, error data is written on magnetic tape for future analysis.

The Roll-Call test enables us to determine which elements required to support a simulation exercise or mission are ready. We have in CADFISS a means of evaluating a real-time ground support system in real-time.

B. The Mercury Programming System

Marilyn B. Scott and Robert Hoffman
International Business Machines Corporation
Federal Systems Division
Washington, D. C.

The Mercury Programming System was designed to meet the specific objective of tracking the Mercury spacecraft during all possible phases of light—launch, abort, orbit and re-entry, and to predict continually the spacecraft impact point so that a safe and speedy recovery of the Mercury astronaut could be effected.

To perform this task, two IBM 7090's, operating in parallel at NASA's Goddard Space Flight Center at Greenbelt, Maryland, receive data by teletype from digital data transmitters associated with radars around the world and by high-speed data links from the Cape Canaveral Complex. In turn, the 7090's feed high-speed data to displays at Goddard and the Cape and send acquisition messages by teletype to remote sites.

Each 7090 has attached to it 12 magnetic tape drives, an on-line printer, punch and card-reader, and an IBM 7281 Data Communications Channel (DDC), which allows data to be transmitted directly to and from the 32-thousand-word core memory of the 7090. Each Data Communications Channel can handle data on each of 32 subchannels. In the Mercury system we currently receive data through the DCC from 17 TTY inputs, two high-speed inputs, a WWV minute signal, and a half-second signal. We send data through the DCC to two high-speed outputs, three TTY outputs, and a sense output console.

The two 7090's operate in parallel to provide backup. Although both receive all inputs, only one is transmitting its computed output to remote sites at any given instant. Both machines, however, feed local displays and send output to the sense console which indicates the status of each and provides the means for switching outputs from one computer or the other.

All of the input/output devices used in the Mercury Programming System produce traps to signal entry or exit of data to and from core memory. A trap is the automatic transfer of control to a preset location in core when one or more conditions are met. In reading a magnetic tape, for example, if a certain instruction is used to request a record, an indicator will be set in the computer when that record has been brought into core memory. This condition, i.e., the "on" status of the indicator, will cause a transfer of control to lower memory. This transfer of control, or trap, will occur only if the machine is "enabled." The computer is said to be "enabled" when traps are permitted to occur and "disabled" when they are not. The machine may be enabled for one or more conditions and disabled simultaneously for others. The machine enters the enabled or disabled mode through execution of a programmed instruction.

There is one other mode of operation upon which the programming system relies. This is the "inhibited" mode. This mode is entered automatically when a trap occurs. While the machine is inhibited, no further traps may occur until the machine is re-enabled, thereby guaranteeing that necessary work may be accomplished without interruption.

These, then, are the three modes of operations: (1) enabled—a trap may occur; (2) disabled—a trap may not occur; and (3) inhibited—a trap has occurred and no more traps will occur until the machine is reenabled.

The Mercury Programming System is composed of three types of routines: those which are hardware-oriented; those which are task-oriented; and those which are independent of both task and equipment and serve to control the interaction and lines of flow among all the other routines in the system.

The first category of routines—those that are hardware-oriented—is concerned with receipt and transmission of input and output. These routines are entered through trapping and are called "trap processors." Since traps may occur asynchronously, even simultaneously, trap processors must respond fast enough to receive all the information from any given source without loss of information from any other source. This is accomplished by (1) allowing all trap processors to operate completely in the inhibited mode in which they are entered by trapping, and (2) requiring that every individual trap processor be written to act fast enough so that the worst combination of simultaneous input/output stimuli will not result in an undesirable loss of information from any source. A typical trap processor might complete its operation and restore trapping in less than one millisecond. At present there are 23 trap processors in the Mercury system to handle at least as many reasons for trapping.

The teletype input trap processor handles a six-character byte of teletype data from any one of 17 teletype transmitters, and moves it to an area for use by the teletype input program. The teletype output trap processor receives control each time six characters of teletype information have left the machine, and refills the output area until a complete message is sent. The high-speed input and output trap processors accomplish the same functions for data traveling at the rate of 1000 bits per second. The printer trap processor tells the system that the on-line printer is available for further output.

Two trap processors handle the timing for the system. One takes control every half-second on receipt of a half-second pulse and calculates the need for computing output and feeding displays. It also updates the current Greenwich mean time within the machine upon which all time tags are based. The other timing trap processor recognizes receipt of a WWV minute signal and checks that the timing of the system is correct.

Other trap processors signal the transmission of data to the sense console and between core memory and magnetic tapes. Tapes are used for bringing in programs when they are needed, for bringing in such data as location and atmospheric conditions of remote sites, for writing out a log of all data which enters or leaves the system, and for generating or reading restart parameters for use in the event of machine failure. Each of these operations, however, goes on in real-time and other work can be done while the tapes are moving because a trap processor will inform the system as soon as the operation is completed.

The second category of routines are those which are task-oriented. The objective of tracking and predicting the landing point of the Mercury capsule is accomplished by diverse procedures during each of the four possible phases of flight. During launch and low aborts, while the capsule is still within range of Cape Canaveral and its radars, the programs are receiving telemetry and computed positional data from the IP 7090 and the GE-Burroughs Guidance Computer. They also accept high-speed raw radar data. The inputs are arriving at the 1000 bits per second rate and displays are being fed every half-second in launch and every second in abort. The only station which receives acquisition messages during these two phases is Bermuda.

In high aborts, those occurring near insertion, both high-speed computed data and low-speed raw radar may be received. Many of the programs of both launch and orbit are required. For instance, the high abort phase requires the use of both the launch program to interpret telemetry data from high-speed sources and the orbital integration program to compute impact point. In orbit the output display rate changes to once every six seconds and no high-speed data is received. Acquisition is sent to more than 20 telemetry and radar sites around the world. The acquisition messages are sent at the six-character per second TTY rate and are sent three times per orbit for each station with lead times of approximately 45, 25, and 5 minutes before the spacecraft crosses the local horizon. The orbital programs generate a prediction table which cover three orbits, receive raw radar data, edit it, and use it to differentially correct the prediction table. From this table, data concerning present and predicted position are calculated. From it the time to fire the retro-rockets to bring the spacecraft back to earth is computed and displayed. During the re-entry phase the display rate changes to once every three seconds, twice the rate of the orbit displays. Different quantities are calculated and acquisition data is sent with less lead time.

The routines which combine to accomplish the tasks of the various phases are called "ordinary processors." They operate in the enabled mode, i.e., they may be interrupted at any point. These ordinary processors are programs in themselves, require no subroutine linkage and can be unit-tested before they are incorporated into the operating system.

At present there are more than 40 ordinary processors in the Mercury Programming System. They include edit program, teletype code conversion routines, coordinate conversion routines, output generators, a retro-fire calculation routine, a numerical integration program, a differential correction program, and a host of others. Because these processors accomplish specific tasks and are completely independent programs, they may be considered the building blocks of the Mercury system. By using them in different combinations and at different frequencies, we can use the same routines to perform different tasks. The use of such building blocks minimizes the amount of programming required to meet the objectives of the system, and provides good flexibility in meeting additional specifications. Furthermore, this modular concept spares programmers of ordinary processors the job of understanding the complexities of real-time control and equipment idiosyncracies, whereas the remaining programmers need not concern themselves with the mathematics of the ordinary processors.

But how do such independent processors fit into an over-all system? How is their interaction controlled?

The processors are fit into the system by means of short sequences of instructions called "prefixes" and "suffixes." There is at least one prefix and one suffix for each ordinary processor. A prefix precedes program and performs such functions as placing inputs required by the program in a prescribed location. The first instruction of the prefix is the entry point to the ordinary processor. The prefix may be said to set the scene for entering the processor. The suffix does just the opposite. It ties up loose ends after the processor is completed, such as making the computed results of the program available for use in the system.

In addition the prefixes and suffixes provide the liaison between the ordinary processors and the third category of routines, those that control the interaction and lines of flow between the various elements of the system. The programs in the third group are called the controllers or monitor routines. The controllers maintain the proper sequence of operations between the various processors, and, in accordance with the modular concept, are completely independent of these processors, i.e., they would accomplish the same functions no matter what processors they monitor.

Implicit in the nature of the ordinary processors and trap processors is a relative priority among them. Since the ordinary processors operate in the enabled mode, control may pass to one of the trap processors at any moment. Therefore, as a group, the trap processors enjoy a higher priority in the system than the ordinary processors. Priority among the trap processors is a function of data rate and built-in priority in the hardware. Among the ordinary processors, however, priority is dictated by urgency of introducing their output into the system and the length of time it takes to execute the routine. The routine to calculate a display every three seconds, for example, must enjoy a priority higher than the routine which must be accomplished every minute. A routine which requires more than three seconds for execution must be given a priority lower than the three second display program, or the system will fail to output on time. It may well enjoy a priority higher than the minutes processor, however. The relative priorities of the ordinary processors are established in a table called the priority table.

The heart of the monitor system is a controller routine called PRIO. Every ordinary processor and trap processor returns control to PRIO upon completion of their tasks. Both ordinary and trap processors can determine the need for entering other routines in the system. They convey this requirement to PRIO by setting indicators in the priority table. PRIO must examine these indicators and give control to the proper routine in accordance with their relative priorities.

Since ordinary processors can be trapped at any time or any place, this priority scheme affords multiple levels of interruption. For example, the orbit prediction updating program may be interrupted by the half-second clock trap processor, whereupon this trap processor determines that it is time for the higher priority, acquisition data generator to send predicted orbit parameters to sites ahead of the spacecraft. When the half-second trap processor returns control to PRIO, control must now be passed to the beginning of the acquisition data generator. But the fact that the orbit prediction update was originally interrupted must not be forgotten.

Once PRIO has given control to the acquisition data generator, it too can be interrupted, say, by the teletype input trap processor transmitting radar data from a tracking site. And now this trap processor indicates to PRIO the need for the teletype conversion processor, and this has a higher priority than the acquisition data generator, etc., until there may be as many interrupts as there are levels of priority. To control such situations, a system of remembering a stack of interrupts is provided by two controllers, SAVE and RTRN.

The very first duty of any trap processor is to enter SAVE to preserve the condition of the machine's program-accessible registers and the return address. Each time SAVE is entered, these items are preserved in a save block assigned to the interrupted routine. Whenever PRIO sees that all higher priority routines have been serviced, control is passed to RTRN which restores the machine's conditions according to the save block and returns control to the interrupted routine at the interrupt point. This save-and-restore procedure holds for all traps except those which occur within PRIO and RTRN. By permitting SAVE to ignore traps from within PRIO and RTRN, control functions are made more responsive to the expediencies of the real-time environment than to the lower priority requirements of the ordinary processors. Thus trapping takes priority over anything PRIO or RTRN are doing.

How is the priority system applied to the ordinary processors?

Associated with each ordinary processor are three kinds of indicators which convey control sequence requirements of the system to the controller PRIO:

1. The "A" indicator is turned on by a processor if it is in process.
2. The "B" indicator is turned on for a given processor if a need for its operation has been determined.
3. C, D, E . . . . indicators for a given processor are turned on if a need has been determined to suppress its operation.

The entry point to each ordinary processor appears in the priority table together with the indicators associated with that routine. (The indicators are merely bits in a word.) The routine whose indicators are examined first by PRIO has the highest priority.

Suppress indicators for a given ordinary processor take precedence over its A and B indicators in that PRIO examines them first. PRIO will not transfer control to any routine whose suppress indicators are on. Each ordinary processor has a specific suppress indicator for each individual condition which requires its suppression. If no suppress bit is on, PRIO will examine the A and B indicators for that routine.

If PRIO sees both the A and B indicators on for routine X, it knows (1) that X had been trapped while in process, and (2) that at least one additional request for X's operation has been made. When both A and B are on, therefore, PRIO will return control through RTRN to the interrupt point in X, leaving the B indicator on for future control. If only the B indicator is on for ordinary processor X, PRIO transfers control to the beginning of X. The first act of every ordinary processor thus entered at the starting location is to turn its own A indicator on, and its last act before transferring control back to PRIO is to turn off its A indicator. Thus, in the event it is interrupted before completion, each ordinary processor provides an indication to PRIO that it is in process.

Indicator B can indicate that more than one request has been made for the operation of its associated ordinary processor. This is accomplished by queueing, a procedure in which one or more requests for a given routine are indicated by placing information in a table, called a queue table. An entry in the queue table can simply indicate a request for operation, or it can also include information telling where input is to be found or where output is to be placed, etc. For instance, an ordinary processor which determines the need for sending acquisition data to radar sites places entries, by queueing, in the queue tables for the teletype output processors, telling them which sites are to receive data. Entries are placed in the queue table in the order in which they occur. After turning on its A indicator, the output processor turns off its B indicator if it has no queue or if it is not desired to have PRIO cycle control to it repeated for some reason. If the routine has a queue, the routine itself must enter the disabled mode and test, while disabled, whether it is not processing the last request in its queue. If the last request is being processed, B is turned off. If not, B stays on, so that the other requests may be answered in turn until the queue is emptied. In either case, the test must be followed by an enabling instruction since all ordinary processors must run enabled. The B indicator for a given routine may be turned on by any ordinary processor, trap processor or controller which determines the need for that routine to be executed.

The optimum assignment of priorities for ordinary processors can only be done through careful study of the system in a real or simulated environment. However, the assignment of priorities is not as rigid as it might appear, for both suppression and trapping in reality constitute dynamic modification of the priority system in response to real-time events. A reasonable first approximation of the optimum priority assignment might be constructed from consideration of the levels of input/output timing requirements.

This, then, is the basic Mercury Monitor real-time control system which combine groups of processors to handle the demands of any particular phase of the Mercury flight: controllers, which maintain sequence control between ordinary processors in an environment of asynchronous interrupts, handled and interpreted by trap processors.

The use of this modular concept has proved to be of enormous advantage in coping with the seemingly ever-changing system specifications.

One of the benefits derived from this approach is perhaps significant for all large, complex real-time control systems undergoing a continuous growth process: efficient use of large scale computer systems by trading off time to increase effective core storage capacity by buffering low priority routines from magnetic tape in real-time. One logical extension of the modular concept, which has been adhered to throughout the Mercury programming system, is the adoption of the simple programming standard that all program communication among ordinary processors themselves and between ordinary processors and the Monitor must take place through system prefixes, suffixes, communication cells, system tables and constants external to the ordinary processors. This is the final isolating step in the complete modularization of the ordinary processors. PRIO can therefore control all real-time buffering with no possibility of a breakdown in communications within the system. Since no communication and no transfer can take place between ordinary processors except through Monitor, PRIO, knowing (by indicator examination) whether or not a requested buffered routine is in memory, can control the system accordingly. If a requested buffered processor is not in memory, PRIO merely queues a monitor processor to load if from tape, remembers that the routine has been requested but not yet given control, and proceeds to handle other system control tasks while the routine is loaded.

Under the simultaneous input/output, computing abilities of the 7090, extremely little time is lost to the system by this technique. Any of the other system functions can be given control while processors are loaded from tape. PRIO simply ignores the routine being loaded until the trap which signals completion of reading is processed and PRIO is informed by unsuppression of indicators that the routine is now in memory. One buffering area in memory can be reserved for the use of low priority routines which can operate sequentially with no loss of system performance. For example, the orbit prediction generator waits for its input from the differential correction program, which, in turn, waits for its input from the editing processor.

Other processors with higher priorities can also be buffered with each other. In fact the only processors which cannot be buffered from tape are those whose timing or control requirements would be violated by the comparatively slow tape operations. Here it should be noted that the delay, due to buffering in execution of the largest buffered routine in the Mercury system (about 3,000 instructions), is about two seconds. The Monitor real-time loader utilized the trapping feature in such a manner that use of buffering

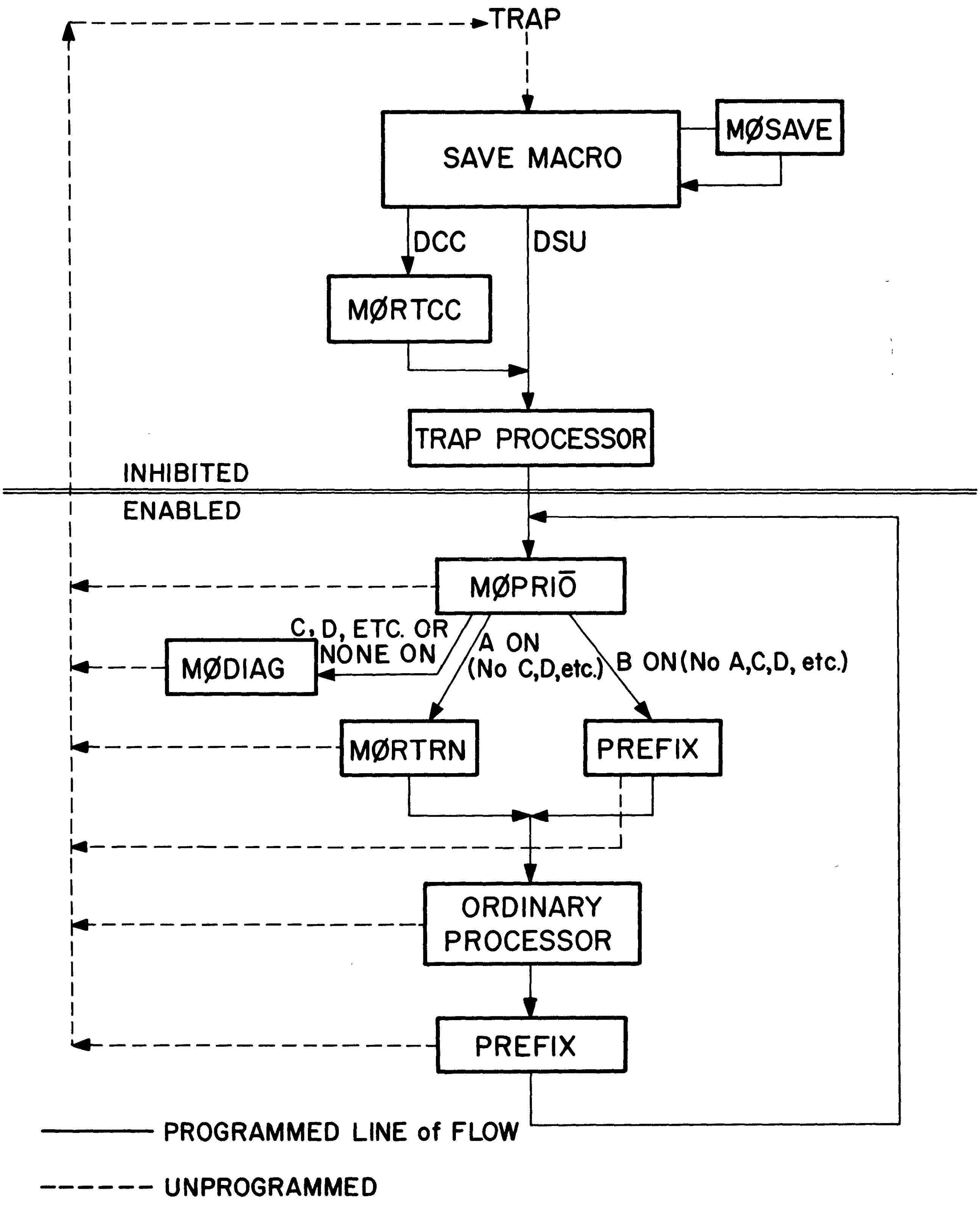

TRAP
SAVE MACRO
MØSAVE
DCC
DSU
MØRTCC
TRAP PROCESSOR
INHIBITED
ENABLED
MØPRIŌ
C, D, ETC. OR NONE ON
A ON (No C,D,etc.)
B ON(No A,C,D, etc.)
MØDIAG
MØRTRN
PREFIX
ORDINARY PROCESSOR
PREFIX
PROGRAMMED LINE of FLOW
UNPROGRAMMED

Figure 1

in the system cannot create a time delay of more than 30 milliseconds in the execution of higher priority, non-buffered ordinary processors. By appropriate use of the 7090's capability for simultaneous tape operations, real-time multiple buffering can be used to increase effective core storage capacity by several times without making the system tape bound. Reliability of tape operations is increased several orders of magnitude by a technique of programmed Hamming-coded error correction, bringing tape reliability into balance with that of other parts of an IBM 7090 Data Processing System.

Thus it is seen that through adherence to the building block or modular principle, the Mercury system performs all its tasks even though some large part of the system is always absent from memory. This is achieved only through Monitor's controlled scheduling of tasks to obtain maximum performance.

Real-time buffering of ordinary processors is but one example of a major modification to the Mercury Programming System's original design. The flexibility afforded by the system's basic structure has allowed us to incorporate such drastic innovations with surprising ease, and has convinced us we are far from exhausting the system's enormous growth potential.

References

1. "Programmed Error Correction in Project Mercury," B. Dimsdale and G. M. Weinberg, Communications of ACM, December, 1960.
2. "Real-Time Multiprogramming in Project Mercury," M. J. Buist and G. M. Weinberg, Ballistic Missile and Space Technology, Volume 1, 1960, Academic Press Incorporated, New York, N. Y.

C. Real-Time Simulation in Project Mercury

W. K. Green and Arnold Peckar
International Business Machines Corporation
Federal Systems Division
Washington, D. C.

Introduction

The testing of the Project Mercury computing system offers a unique problem. The Mercury program runs in real-time, with inputs arriving at specific time intervals and outputs being transmitted at varying time intervals. The computational programs and their associated controls are complexly interrelated and are all time interdependent. Therefore, in order to test these programs the data must be presented in such a manner as to allow the time function to show its effect.

To introduce the time function two simulation methods have been developed. The first method allows the operational programs to run in a quasi-real-time environment under the control of a simulation program. This control program called SIC (Simulated Input/Output Control) keeps track, by means of a real-time clock, of the amount of time which has been allocated to the Mercury Program. SIC also simulates the real-time input/output device called the DCC (Data Communications Channel) and at proper time intervals supplies simulated input data, and processes Mercury output data. Using SIC, "time" may be stopped whenever desired, intermediate results examined, and "time" restarted without the loss of the timing sequence. This then becomes a very powerful debugging tool since now a detailed analysis of the interaction of input, output, and time may be made.

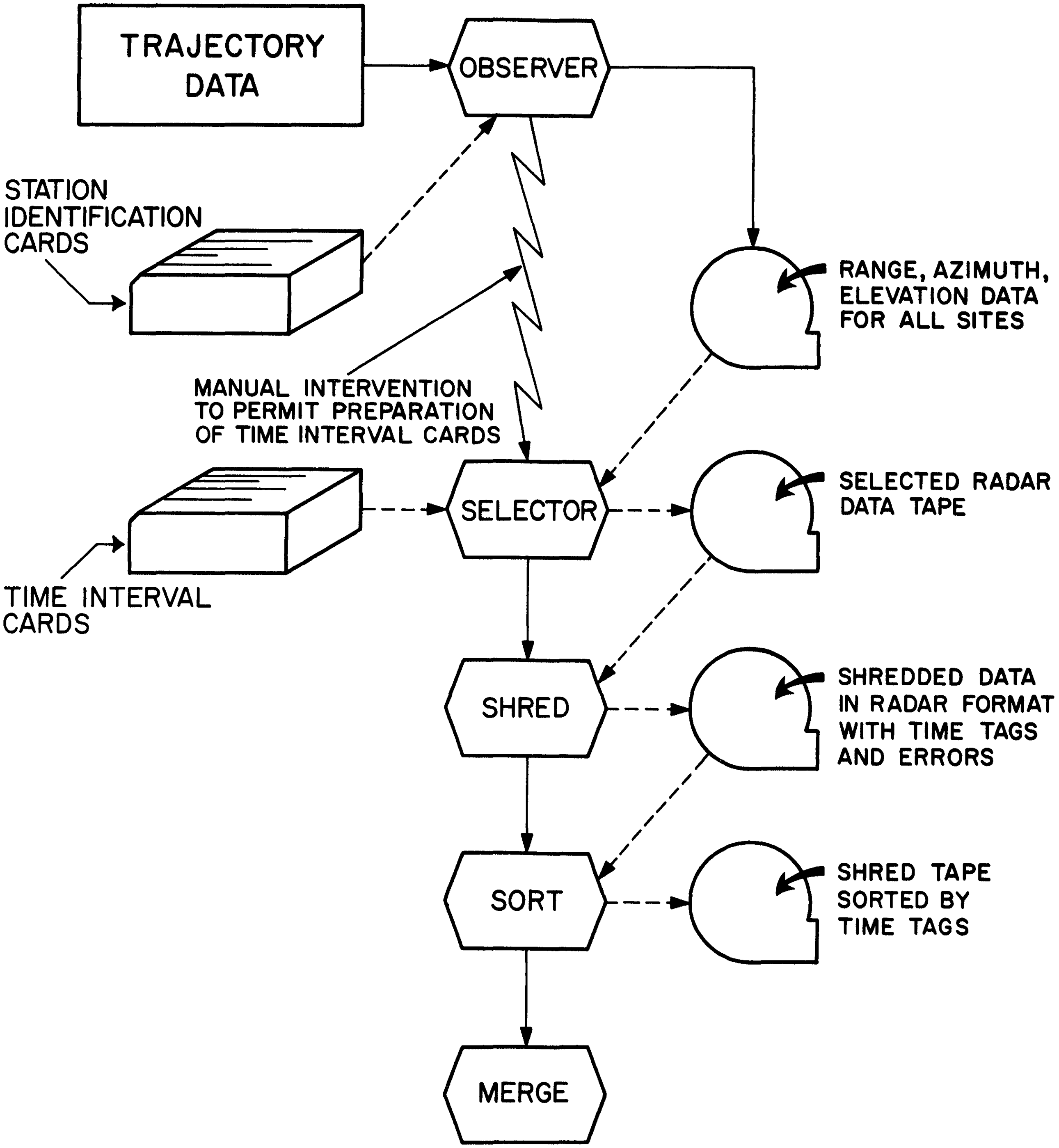
The second simulation method uses specially prepared simulation data which is transmitted to the Mercury computers from the remote sites in real-time. This data is presented to the system in such a manner that an actual flight appears to be in progress, and in fact the system cannot differentiate between this data and that of a real flight.

The simulation data takes two basic forms. The launch data, which is transmitted from Cape Canaveral using a special tape drive known as the "B Simulator," and the orbital and re-entry data, which is transmitted from the remote radar sites using teletype paper tape.

In a system of this nature the heart of the simulation lies in the generation of the test data. To generate this data a series of programs known as Observer, Selector, Shred, Sort, and Merge have been written. These programs have the capacity to generate basic flight profiles giving launch, orbit, abort, and re-entry data in the form which would normally be received from the sites. This data may also be perturbed by application of many types of errors to simulate the malfunctioning of the data system. In this way the limitations of the Mercury programs with regards to varying degrees of data degradation may be determined.

Test Data Generation

The test data for the different simulation methods is generated by a series of programs known as Observer, Selector, Shred, Sort, and Merge. Each of these programs acts as a step in the generation of a given set of data (see Figure 1). They are separate, however, so that between each step options may be taken with regard to modification of the given


TRAJECTORY DATA
OBSERVER
STATION IDENTIFICATION CARDS
RANGE, AZIMUTH, ELEVATION DATA FOR ALL SITES
MANUAL INTERVENTION TO PERMIT PREPARATION OF TIME INTERVAL CARDS
SELECTOR
SELECTED RADAR DATA TAPE
TIME INTERVAL CARDS
SHRED
SHREDDED DATA IN RADAR FORMAT WITH TIME TAGS AND ERRORS
SORT
SHRED TAPE SORTED BY TIME TAGS
MERGE

DATA GENERATION PROCEDURE
FOR MERCURY SIMULATION SYSTEM

Figure 1

flight profile and the entering of varying error conditions.

The Observer program is used to generate basic flight profile. If the profile is to contain launch data a series of position and velocity vectors for the powered flight portion of the trajectory are obtained from the missile guidance equations. From this data is obtained the insertion conditions at burnout. If launch data is not to be used then the insertion conditions must be given as input data.

Using the insertion conditions Observer now computes the orbit by numerical integration giving position and velocity vectors for as many orbits as specified by the input data. The data also specifies re-entry conditions. This is either in the form of retro-rocket firing time or desired impact area. If the retrofiring time is given the change in velocity is computed and the re-entry trajectory computed. If the impact area is given, Observer iterates on the retrofire time until a time is found which gives the requested impact.

The Observer now has the position and velocity vectors for the complete flight. It now reads in the positions of all the tracking stations and computers all the Range, Azimuth and Elevation readings which are in range of each station. These R, A, and E readings are all now written on Observers final output tape.

The Selector program uses as its input the R, A, and E tape produced by Observer. Its purpose is to take only that data required for a given test and place it along with certain error codes on a tape to be used by Shred. It is at this point that the many varied perturbations of the data are called for. Stations may be left out or made to start transmission late or early. Different types of errors for application to the data, such as different noise levels, bias errors, transmission errors and pathological errors are now called for. It is here that all the major troubles that can occur during a real flight may be specified so that the full capabilities and the limitations of the Mercury program may be tested.

The Shred program processes the Selector outputs to produce simulated inputs. Since the function of this routine is primarily one of format changing, unit conversion, and coordinate transformation it is not of general interest in itself. There are, however, a few points that may be of interest.

Shred has two sources of inputs. The primary source is the Selector input. The second source of inputs is provided by Space Technology Laboratories. These consist of launch inputs representing GE-Burroughs Atlas Guidance Computer outputs and position, velocity vectors which are used to calculate IP 709 outputs. From these two sets of inputs Shred produces images that reflect what would be in the Goddard or Bermuda computers at the time of an input interrupt. The different images are: GE-Burroughs, IP 709, and High-Speed raw radar for Goddard each with its 72 bit telemetry message; low-speed TTY Verlort or FPS-16 radar inputs for Goddard; high-speed Verlort and FPS-16 radar for Bermuda; and Bermuda's 64 bit telemetry message.

Shred has three methods for introducing variations into the true input values. Each routine is independent of the other and while none individually are in any sense sophisticated in combination they have proven to be quite satisfactory. One routine is used to introduce random errors which simulate the general radar noise. The second routine forces certain bits in the message to always be one's. We call these errors pathological errors and ostensible represent frozen encoders. The third type of error routine is for transmission error simulation.

Random errors are added to measurements according to an input standard deviation and a normal distribution. Sixteen different standard deviations are allowed for each run. A code associated with the input vector determines which sigma is to be used in the particular case.

So called pathological errors are bit patterns which are used to turn on the matching one bits in the true value. Again, 16 patterns are provided with the desired pattern specified by a code associated with the input vector. Pathological errors put a varying positive bias into the readings.

Transmission errors are simulated by either altering, dropping, or adding a computer word in the input message. A probability of error is associated with each word of output and if a calculated random number (from a rectangular distribution) is less than the probability of error, an error is applied. The form of the error is determined by a similar technique using three conditional probabilities. Each routine can have a different transmission error table and within one table three different probabilities of error are provided. The probability to be used is determined by a code associated with the input vector.

A constant error, that is, a bias error, is simulated by adding the desired amount of bias during the calibrating.

When writing Shred a primary assumption was made that, no matter how vehemently and dogmatically stated, all message formats and parameter units would change. Thus, the program was written to be modified and great use was made of tables, input parameters and switches. This turned out very well for sure enough there is not one output produced by Shred that has not changed at least once and some several times. Units have changed, sampling rates have changed, ranges of values have changed, telemetry bits have changed, message lengths have changed, subchannels to be used have changed. However, since this was expected, measures were taken to live with the situation.

A typical example of programmed flexibility is the unit conversion routine for radars. There are only two types of radars presently used and Shred's input vectors represent the radar readings although in different units, thus all that is required to convert a reading "r" to "R" is:

$$\text{if A, then } R = r \times X_A$$

$$\text{if B, then } R = r \times X_B.$$

However, since it is always possible a different radar type might be suddenly installed or biased conversions required, the following scheme was used. A table containing the X and a constant for each radar parameter was set up for each type of radar. A second table of addresses referencing one of the conversion factors table was set up using station number (a unique radar site identification) as an argument to define which of the conversion factors table to be used. Thus add a new radar type—make a new conversion table and add its location in the control table; change units of input—change the conversion table; if "biased" readings are required—make the constants non-zero. The tables have been modified several times in the past year and will be modified again but the basic calculating routine will not be touched.

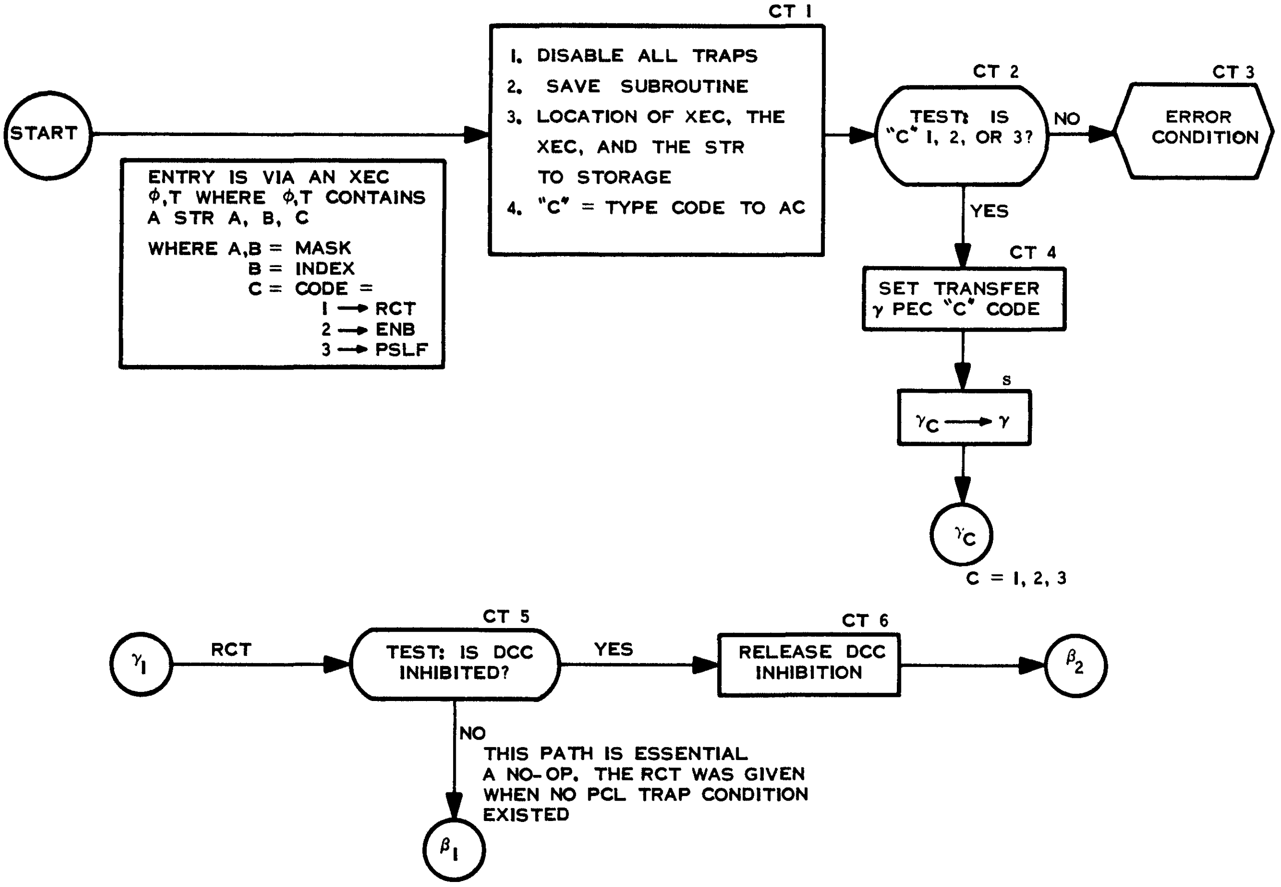
Shred will of course produce quite varied runs. It is of interest to note that most errors are applied at random thus a specific error for one observation is not a normal type of operation (it can be done with bias or with pathological errors). This was done primarily to discourage people from concentrating too much looking for certain oddball cases where the effect is essentially known anyway, and to encourage testing with general error levels. The feeling here is that the unexpected and potentially disastrous type or combination of errors could best be discovered by using a monte carlo technique as opposed to a preconceived set of errors.

On the whole this approach appears to have been successful. Many errors have been found and we apparently now have a solid working system. Of course, no program is ever fully debugged, however, our confidence in the system is very high. Even though we had good success using perturbed inputs, some of the most interesting bugs, however, showed up using error free data. One worthy note occurred in editing. Extensive use is made of statistical editing throughout the system, however, the original coding failed to protect against zero or very small standard deviations thus perfect inputs turned out to be unacceptable. The coding has now been changed so that no matter how good the inputs the system will run but the lesson that both upper and lower limits of tests need to be covered was once again delivered.

In all, there are roughly 25 functioning tables, forty some subroutines, one main and three submain chains in Shred. Detail flow diagrams are maintained for the whole program. The net effect is a very flexible program, readily modified, but also one that requires a good deal of esorteric skill to operate.

The output from Shred can take several forms. For the powered flight portion of the trajectory an unsorted high speed data tape is produced. This represents the data which is received from Cape Canaveral. For the orbit and re-entry positions of the trajectory an unsorted low speed data tape is produced. This represents the teletype data normally received from the remote sites.

For the majority of simulated runs the high speed and low speed data must be sorted and merged. For this purpose the Sort and Merge programs were written. The final output from these programs is the input to SIC which will be described next.



SIC FLOW CHART

SIMULATION OF ENABLE,PSLE (ACTIVATE DCC SUBCHANNELS) AND RCT (RESTORE CHANNELS FOR TRAPS)

Figure 2

Simulated Input/Output Control Program (SIC)

This section of the paper describes the simulation techniques used to debug and to analyze the real-time Project Mercury Goddard IBM 7090 computer and the Bermuda IBM 709 computer programs. This simulation package is used only with local computing equipment and is not used to test the communication lines or the training of flight control personnel. It has the obvious advantage of being self-contained thus requiring no time consuming coordination with other sites. In addition, it has the features of being able to maintain proper timing constraints while also permitting the suspension of time to allow debug macros such as "core" to be performed.

SIC shares the computer with the operational programs. (Figure 3) Its function is to feed inputs at the proper times to the operational program and to simulate all of the controls associated with real-time inputs and outputs. To do this SIC must have a means of keeping time—at present a one millisecond pulse is produced by the Data Communications Channel (DCC) causing a trap to SIC. Each time SIC gets control via its clock it adds a finite increment to its present estimate of time. Then based upon the new time SIC does the following: searches its input for messages that should now go to the operational program input region, tests for traps that require simulation, logs output or actually sends out output if desired, and adjusts itself to allow the operational program another cycle. These SIC functions plus the following are described in detail below:

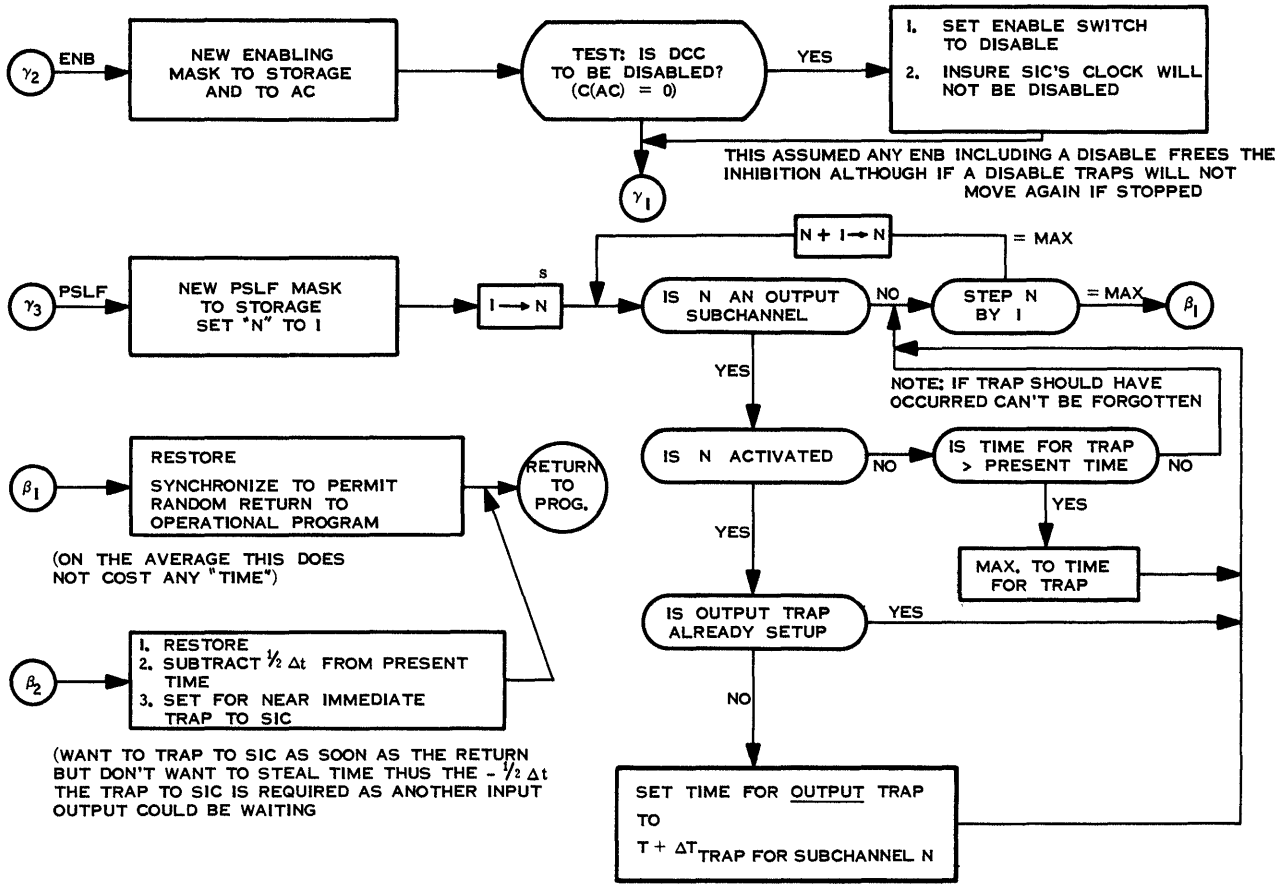

γ2
ENB
NEW ENABLING MASK TO STORAGE AND TO AC
TEST: IS DCC TO BE DISABLED? (C(AC) = 0)
YES
1. SET ENABLE SWITCH TO DISABLE
2. INSURE SIC'S CLOCK WILL NOT BE DISABLED
THIS ASSUMED ANY ENB INCLUDING A DISABLE FREES THE INHIBITION ALTHOUGH IF A DISABLE TRAPS WILL NOT MOVE AGAIN IF STOPPED
γ1
N + 1 → N
= MAX
γ3
PSLF
NEW PSLF MASK TO STORAGE SET "N" TO 1
s
1 → N
IS N AN OUTPUT SUBCHANNEL
NO
STEP N BY 1
= MAX
β1
YES
NOTE: IF TRAP SHOULD HAVE OCCURRED CAN'T BE FORGOTTEN
β1
RESTORE
SYNCHRONIZE TO PERMIT RANDOM RETURN TO OPERATIONAL PROGRAM
RETURN TO PROG.
IS N ACTIVATED
NO
IS TIME FOR TRAP > PRESENT TIME
NO
YES
(ON THE AVERAGE THIS DOES NOT COST ANY "TIME")
YES
MAX. TO TIME FOR TRAP
IS OUTPUT TRAP ALREADY SETUP
YES
β2
1. RESTORE
2. SUBTRACT ½ Δt FROM PRESENT TIME
3. SET FOR NEAR IMMEDIATE TRAP TO SIC
NO
(WANT TO TRAP TO SIC AS SOON AS THE RETURN BUT DON'T WANT TO STEAL TIME THUS THE - ½ Δt THE TRAP TO SIC IS REQUIRED AS ANOTHER INPUT OUTPUT COULD BE WAITING
SET TIME FOR OUTPUT TRAP TO T + ΔT TRAP FOR SUBCHANNEL N

Figure 2a

a) SIC Input
b) SIC's Clock
c) Trap Simulation
d) DCC Control Simulation
e) Debug Macros

The input to SIC is on a magnetic tape and is sequenced by time of arrival. Each logical record of input represents information that would be found in an input region after an input trap. Two times are associated with each input; first is the time the record should start to enter the input area (called TA, for "Time of Arrival"), and the second is the time the interrupt should be given (TT for "Time of Trap"). In addition, each record also has a control word which gives the number of words and the DCC subchannel of the ostensible transmission. All times are referenced to launch, thus a zero TA is for the launch vehicle still on the pad.

SIC's clock maintains a descrete estimate of continuous real-time by updating itself by a constant t after each trap.

To initiate the traps the DCC's we use have a simulation switch which when "on" transmits a pulse every millisecond which under program control can be used to cause a trap. Prior to getting a DCC a special device was attached to a DSU of the 709 which could cause traps at variable intervals the smallest being 5 milliseconds.

In using the clock there are two variables; the actual number of machine cycles between traps and the value of t assigned to represent the interval. Assigning t a value greater than the actual time between pulses in effect simulates a computer that has less computing speed than the one being used as the total number of machine cycles in a simulated time period would be less than the number in a real period and visa versa. This is a very useful feature. It permits simulation of Bermuda's 709 and Goddard's 7090 and Goddard's 7090 on the Space Center's 709. It also has been used to test the capacity of the system by increasing t until failures occurred.

SIC FLOW CHART
TIME ESTIMATE-INPUT- I/Ō TRAPS

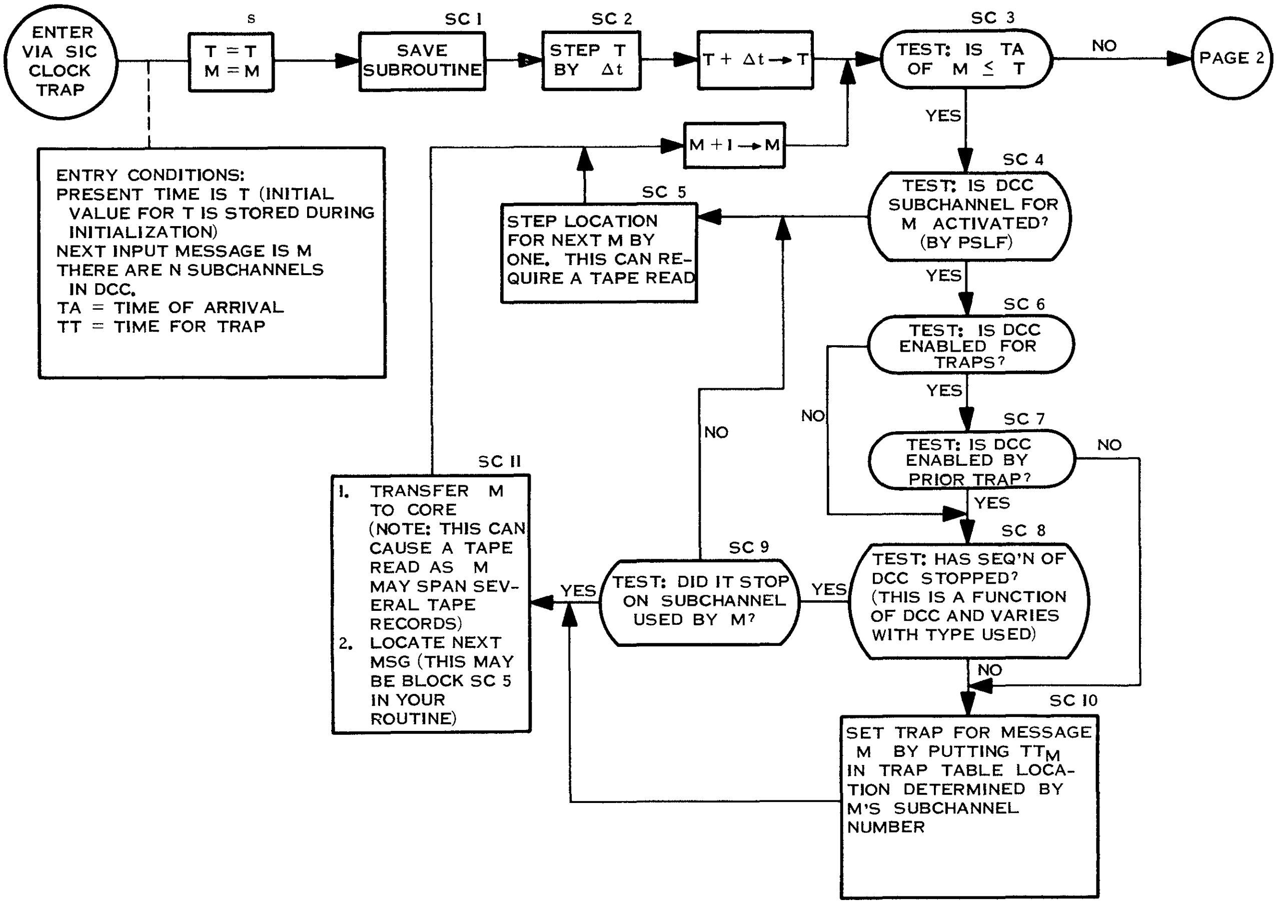

ENTER VIA SIC CLOCK TRAP
S
T = T
M = M
SC I
SAVE SUBROUTINE
SC 2
STEP T BY Δt
T + Δt → T
SC 3
TEST: IS TA OF M ≤ T
NO
PAGE 2
YES
M + I → M
SC 4
TEST: IS DCC SUBCHANNEL FOR M ACTIVATED? (BY PSLF)
ENTRY CONDITIONS:
PRESENT TIME IS T (INITIAL VALUE FOR T IS STORED DURING INITIALIZATION)
NEXT INPUT MESSAGE IS M
THERE ARE N SUBCHANNELS IN DCC.
TA = TIME OF ARRIVAL
TT = TIME FOR TRAP
SC 5
STEP LOCATION FOR NEXT M BY ONE. THIS CAN REQUIRE A TAPE READ
YES
SC 6
TEST: IS DCC ENABLED FOR TRAPS?
YES
NO
NO
SC 7
TEST: IS DCC ENABLED BY PRIOR TRAP?
NO
SC II
1. TRANSFER M TO CORE (NOTE: THIS CAN CAUSE A TAPE READ AS M MAY SPAN SEVERAL TAPE RECORDS)
2. LOCATE NEXT MSG (THIS MAY BE BLOCK SC 5 IN YOUR ROUTINE)
YES
SC 8
SC 9
YES
TEST: DID IT STOP ON SUBCHANNEL USED BY M?
YES
TEST: HAS SEQ'N OF DCC STOPPED? (THIS IS A FUNCTION OF DCC AND VARIES WITH TYPE USED)
NO
SC 10
SET TRAP FOR MESSAGE M BY PUTTING TT_M IN TRAP TABLE LOCATION DETERMINED BY M'S SUBCHANNEL NUMBER

Figure 2b

It should be noted by anyone planning to use a SIC type program that the smaller the interval between traps the more accurate the simulation of real-time, however, since smaller increments require more passes through SIC the longer the simulation run will take.

In our version of SIC it was desirable to start runs not necessarily at the beginning of the input tape and also to vary the value assigned to t. To do this, the time to start the run and the value assigned to t are put in the storage entry keys at the start of a run. The option to change the t during a run has been added to some versions of SIC.

There are two returns from SIC when it is entered by a clock trap. If no input/output traps require simulation, the return is to the point in the operational program from whence the clock trap occurred, however, if a trap is required, it is simulated as having occurred at the same point in the operational program as the clock trap to SIC.

SIC has an interrupt table which determines whether a trap is required. This table has an entry for each subchannel of DCC (only DCC traps are simulated in this version of SIC) and the value of the entry is the time for the next trap. If no traps are set up the value is set to the maximum. Input traps are set up in the table from the time of trap associated with each input record. Output traps are set up when the instruction to produce output is simulated (this is discussed later). The interrupt table is interrogated with every trap to SIC and if a time is found that is less than or equal to the present estimate of time (and the DCC is enabled and not inhibited and the subchannel activated) a trap is simulated and the time to trap for that subchannel set to the maximum. Only one trap can be simulated at a time (same as in the real case) thus

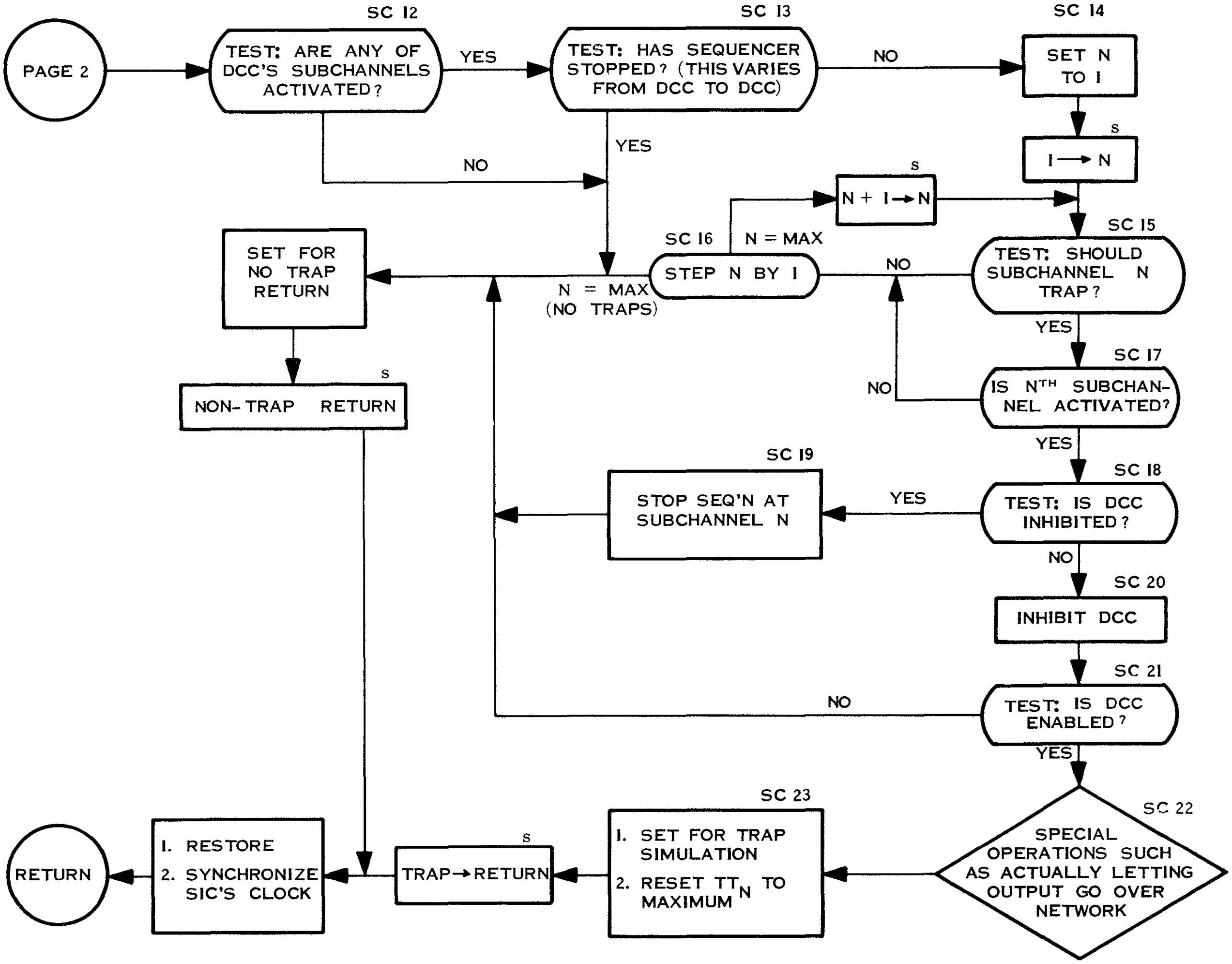

PAGE 2
SC 12
TEST: ARE ANY OF DCC'S SUBCHANNELS ACTIVATED?
YES
NO
SC 13
TEST: HAS SEQUENCER STOPPED? (THIS VARIES FROM DCC TO DCC)
NO
YES
SC 14
SET N TO I
s
I ⟶ N
s
N + I ⟶ N
SC 15
TEST: SHOULD SUBCHANNEL N TRAP?
NO
YES
SC 16
N = MAX
STEP N BY I
N = MAX (NO TRAPS)
SET FOR NO TRAP RETURN
s
NON-TRAP RETURN
SC 17
IS N^TH SUBCHANNEL ACTIVATED?
NO
YES
SC 18
TEST: IS DCC INHIBITED?
YES
NO
SC 19
STOP SEQ'N AT SUBCHANNEL N
SC 20
INHIBIT DCC
SC 21
TEST: IS DCC ENABLED?
NO
YES
SC 22
SPECIAL OPERATIONS SUCH AS ACTUALLY LETTING OUTPUT GO OVER NETWORK
SC 23
1. SET FOR TRAP SIMULATION
2. RESET TT_N TO MAXIMUM
s
TRAP ⟶ RETURN
1. RESTORE
2. SYNCHRONIZE SIC'S CLOCK
RETURN

Figure 2c

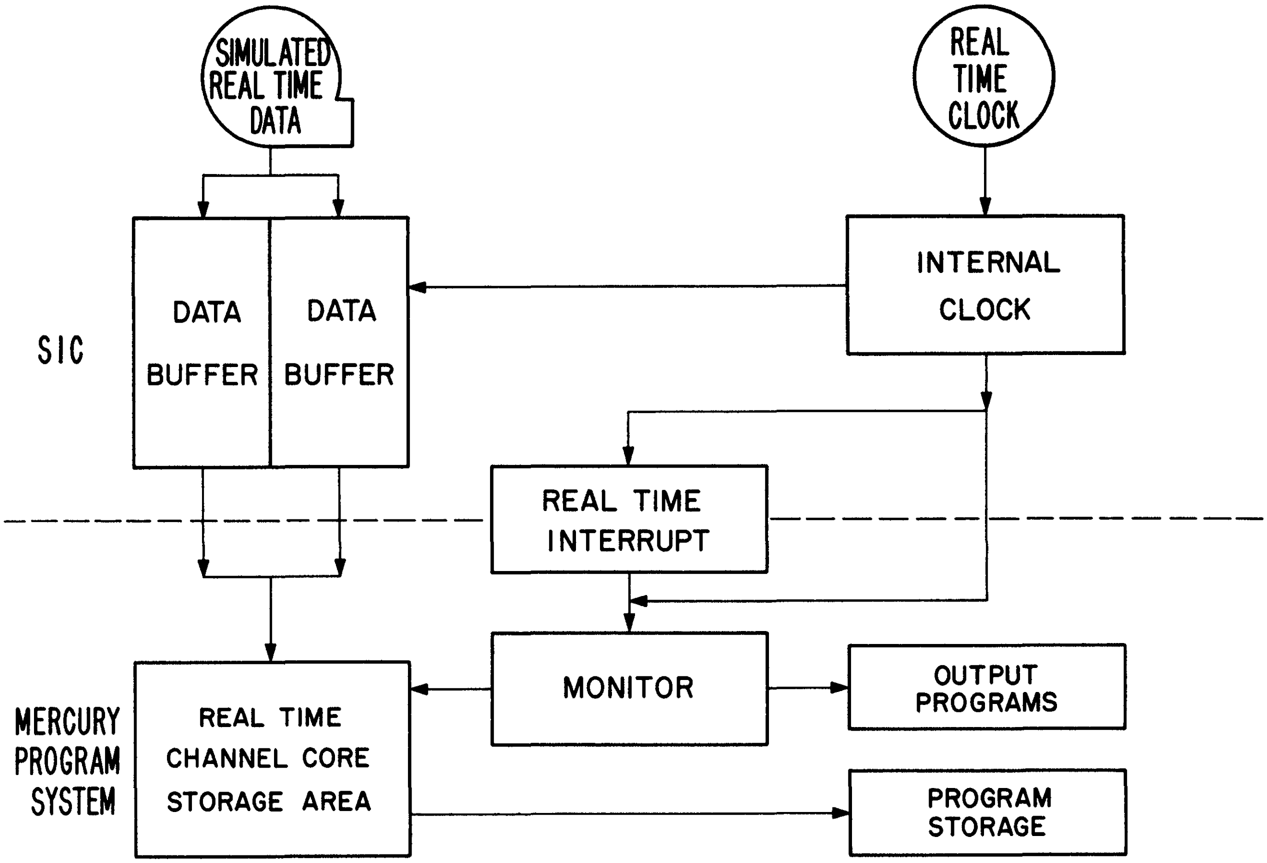
if one trap is found, the rest of the table is not interrogated. If more traps are waiting they will occur only after the operational program has processed the first one and enabled the computer. If SIC does not do this in time, the next message will be in the input region. With present versions of SIC this fact is not noted (as indeed, it would not be noted in the real-time case), however, in the generalized version shown in the flow diagrams a path is provided for special operations if this happens.

There are three instructions that control the trap features of the DCC device. The instructions and their functions are:

1. ENB A, T = Enable per contents of A, T. This allows traps to occur over the I/O channels of the 709(0) specified by the mask in A, T. The DCC is considered one of the I/O channels of the 709(0) computer.

2. PSLF B, T = Allows inputs or outputs (depending upon the DCC subchannel) to occur and permits the DCC to trap if it is enabled. The mask in B, T contains a bit for each of the 32 subchannels. If the bit in position i is 1, subchannel i is activated, if zero, deactivated. No traps can occur unless the DCC itself is enabled and the subchannel activated.

3. RCT = Restore channel trap—this restores the trap conditions to what they were prior to the last trap. The instruction, by itself, does not refer to a new enabling mask.



SIC AND THE MERCURY PROGRAM SYSTEM

Figure 3

When simulating, these instructions are not obeyed directly. To avoid confusion in going from the simulated to the unsimulated case, the operational program has in place of the actual trap control instruction an execute XEC A, T—where A, T refers to a table. If simulating, the table contains a store location and trap, STR B, T, C while in the real case the table contains the ENB, PSLF, or RCT. The STR gives control to the section of SIC used for simulating the trap controlling instructions, the B, T locates the ENB, or PSLF mask and the "C" is a code used to define which of the three instructions is wanted.

Since entry to this section of the program is arbitrary with respect to SIC's clock the return is the same way. Thus, for timing purposes the maximum gain or loss of computing capacity is less than one time interval with the sum of the gain and loss "expected" to be zero.

If reenabling (ENB or RCT) after a trap it is possible a second trap could occur immediately. Thus, if the DCC was inhibited at the time the instruction is given, a search of the interrupt table is needed. In the flow charts for the generalized version this is accomplished by letting the next SIC trap occur almost if not immediately after the return from this section of SIC. In older versions, a trnasfer directly to the clock updating, trap searching part of SIC was made. In the older versions the net effect is to steal a part of a cycle from the operational program, however, in the general case, one-half cycle is added so that again the expected gain or loss is zero.

Since SIC is maintaining the estimate of time it is possible to "suspend" normal calculations by stopping SIC's clock and during this period perform whatever debugging operations such as "cores" or "dumps" as desired then restart the program at the point of suspension. This is accomplished by two SIC routines which stop and restart the clock. Two macros were made for the calling sequences and these are used to sandwich all debug sequences in the operational program. This feature is one of the most powerful and well used tools for debugging. Routines are presently being written which will make SIC input from log tapes of actual runs so that it will be possible to replay actual shots and make use of this feature.

Real-Time Simulation

The simulation up to this point has been in a quasi-real-time environment which has only involved the Project Mercury computers. The next logical step is to now use a simulation system which does in fact operate in real-time and includes varying amounts of equipment in the data link. The output displays being part of this data link may then be driven by the computers to test the real-time output capabilities and to serve as a training exercise for the flight controller personnel at Cape Canaveral.

For the launch portion of the real-time simulation a special tape drive known as the B-Simulator has been developed (Figure 4). This drive which is located at Cape Canaveral uses magnetic tapes generated at Goddard as the data source. The data from the B-Simulator is played over the transmission lines to the data receivers at Goddard and then into the DCC's attached to the computers.

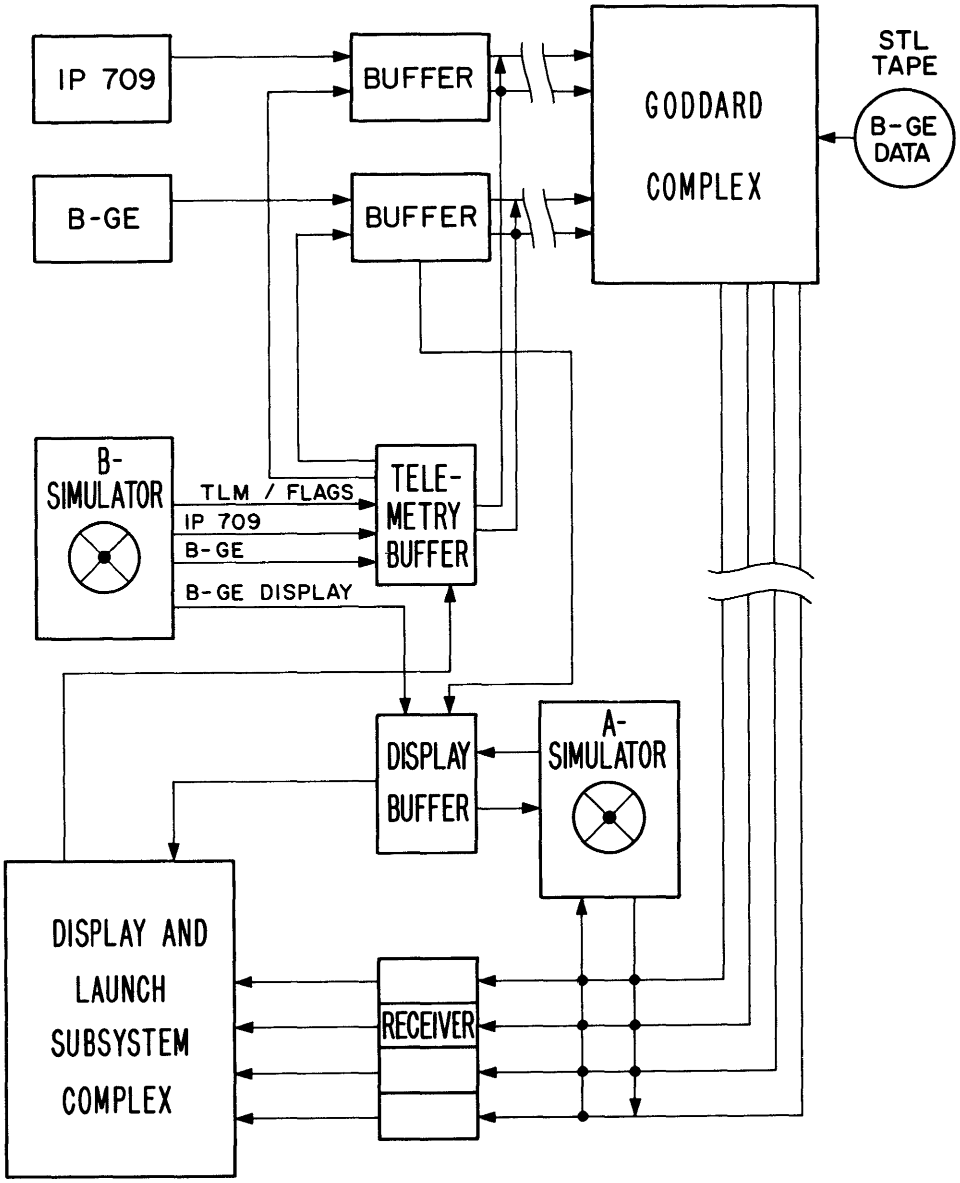
The B-Simulator tapes are specially prepared by a program which uses as inputs a sorted, high-speed data tape produced by Shred and Sort, the Atlas-Guidance Computer cards and a set of flight flag cards. This data is read into the IBM 7090 computer and written on tape at a density of 200 bits per inch. Each data source is written on a separate track, serially down the tape. A data bit is represented on the tape by two bits. These bits are six bit positions apart, so that when the tape is read on the B-Simulator at 60 inches per second these bits will cause a pulse duration of one-half millisecond. This pulse is the data form which is transmitted to Goddard as the simulation data.

An interesting problem presented by the generation of B-Simulator tapes was that since they run in real-time they may not have any data gaps. To prevent data gaps the full length is written as one continuous record. This is done by starting the tape and then having the computer prepare the data faster then it can be written. Four data blocks are used and while one is being written another is being filled. The block being prepared is always two ahead of the block being written, and these blocks are rotated as each successive block is completed.

The data used in a B-Simulator run can be identical with that used in a SIC run. If it is the same data then the results should be the same. This is not always the case, however, since now we have added to the system the errors caused by the transmission lines and the data receivers. The number and magnitude of these errors are of great interest since we can get a true feel for what to expect on a live launch. With this information, modification can be made to the operational programs so that errors of this nature will not prevent the proper functioning of the computing system.

A system similar to and using the B-Simulator was also developed for real-time testing of the Bermuda computer system (Figure 5). In this system a Bermuda SIC input tape is read into the IBM 7090 and a tape similar to the Goddard B-Simulator tape is produced. This tape, however, only has data representing the two Bermuda radars. This tape is then transported to Cape Canaveral and played on the B-Simulator. However, instead of the data being transmitted to Goddard it is rerecorded on another tape drive known as the A-Simulator (Figure 5). The A-Simulator tape is recorded in the same format as data received by the Bermuda radar data receivers. The A-Simulator tape is then sent to Bermuda where it is played at the data receivers so that real data appears to be on the lines. This data may then be presented to the Bermuda Computer for real-time simulation runs.

The objective of a Mercury flight is to orbit the spacecraft and return it to earth. The real-time simulation must therefore consist of orbital and re-entry data as well as launch data. The orbit and re-entry data normally arrives at the computer via teletype lines from the remote sites around the world. To simulate the teletype data a special program was written for the CDC 160 computer which reads an unsorted, low-speed Shred output tape and punches teletype paper tapes. Each paper tape represents the data that is received from a given site for a given pass over the site. The paper tapes are then

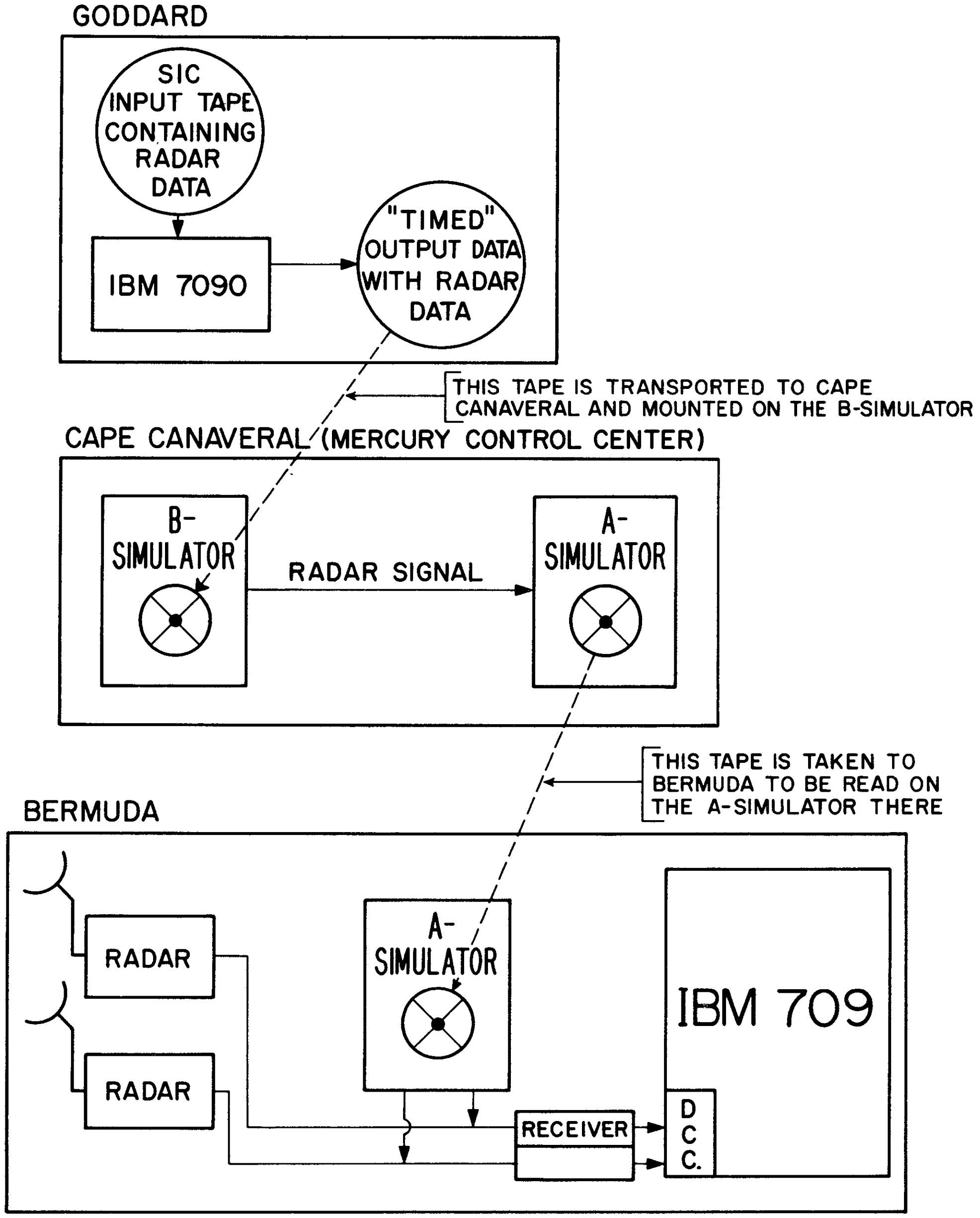


LAUNCH SUBSYSTEM WITH A AND B-SIMULATORS

Figure 4

transported to the remote sites and on proper time cue are transmitted in sequence over the teletype lines to the computers. The data then appears in the computer as actual teletype transmissions with the effect of equipment failures applied. Since this orbital and re-entry data was generated from a Shred output tape, it also may be identical to that used in a SIC run. And the different results between the two runs may be analyzed to determine the effect of the equipment upon the final computer outputs.

A very important feature of this real-time simulation system is its use as a training tool. There are many people who, during an actual mission, will have to make quick decisions based upon the computed output data. These decisions affect the safety of the astronaut and must, therefore, be based upon long experience with the system. To supply this experience many simulated runs are made with many different sets of data. These runs present to the flight controllers all the situations which they might see during a mission.


GODDARD
SIC INPUT TAPE CONTAINING RADAR DATA
IBM 7090
"TIMED" OUTPUT DATA WITH RADAR DATA
THIS TAPE IS TRANSPORTED TO CAPE CANAVERAL AND MOUNTED ON THE B-SIMULATOR
CAPE CANAVERAL (MERCURY CONTROL CENTER)
B-SIMULATOR
RADAR SIGNAL
A-SIMULATOR
THIS TAPE IS TAKEN TO BERMUDA TO BE READ ON THE A-SIMULATOR THERE
BERMUDA
RADAR
RADAR
A-SIMULATOR
IBM 709
RECEIVER
D C C.

BERMUDA CLOSED LOOP SIMULATION DATA FLOW

Figure 5

In this way they know what to expect and learn what actions should be taken. These runs can be made over and over again without the use of missiles so that expensive errors can be avoided and erroneous decisions corrected.

Simulation runs consisting of a complete three-orbit mission have been made. The high-speed launch data was supplied by the B-Simulator, and the low-speed orbit and re-entry data was supplied by paper tapes from sites around the world. The complete mission takes almost 5 hours. During that time, all the remote sites were using their equipment and all the displays at Cape Canaveral, which are driven by the computer, were in operation. The flight controllers performed their tasks as did all the communications personnel involved. The only difference between this test and an actual Mercury flight was that a launch vehicle/spacecraft never left the launch pad.

D. Project Mercury Launch Monitor Subsystem (LMSS)

R. D. Peavey and J. E. Hamlin
International Business Machines Corporation
Federal Systems Division
Washington, D. C.

I. Introduction to LMSS

The LMSS is an integral part of the World-Wide Tracking and Ground Instrumentation Network. Logically, the world-wide network consists of:

1. An array of radar and telemetry stations, strategically located geographically, to track and report successively on the Mercury spacecraft position during Orbit and Re-entry.
2. A smaller, more localized, complex of radar and telemetry stations to monitor and report the launch trajectory achieved by the spacecraft missile assembly.
3. Several computational subsystems to process the reports obtained via 1 and 2 above.
4. A world-wide communications network to tie 1, 2, and 3 together, both on a teletypewriter data flow and narrative voice basis.

An idea of the magnitude of this effort may be obtained by reference to Figure 1. From this figure, and from the rest to follow, it becomes obvious that the NASA-Goddard Space Flight Center is the hub of this communications networks. However, it must be pointed out that the decision-making functions of the Mercury flights are located at the Mercury Control Center.

The portion of this overall complex that represents the Launch Monitor Sybsystem is shown in Figures 2 and 3.

Figure 2 emphasizes the Systems aspect of the LMSS while Figure 3 broadly exemplifies the functional aspects at each location included therein from an equipment standpoint.

The purpose of this paper is to briefly present the functional and data flow aspects with as little operational programming detail as possible to enable a coherent understanding of the LMSS.

II. Launch to Insertion

During this portion of the flight, several hi-speed data flow paths are established from various sources in the Atlantic Missile Range (AMR) to and from the Goddard Space Flight Center (GSFC), including those for the purpose of local displays at GSFC. The return path of hi-speed data flow from GSFC to the Mercury Control Center (MCC) contains computed quantities to drive the displays and plotboards at MCC as required for decision-making purposes.

A. Broadly, these simultaneous data flow paths are: (Figure 2 and 3)

1. a. The output of the Atlas Guidance Computing System, interleaved with telemetry and discrete event information to the GSFC for processing and subsequent output back to the MCC plotboards, digital displays, wall map, and Data Quality Monitoring equipment.
   b. Selected outputs from the Atlas Guidance Computing System to be sent directly to certain of the displays and plotboards at the MCC.

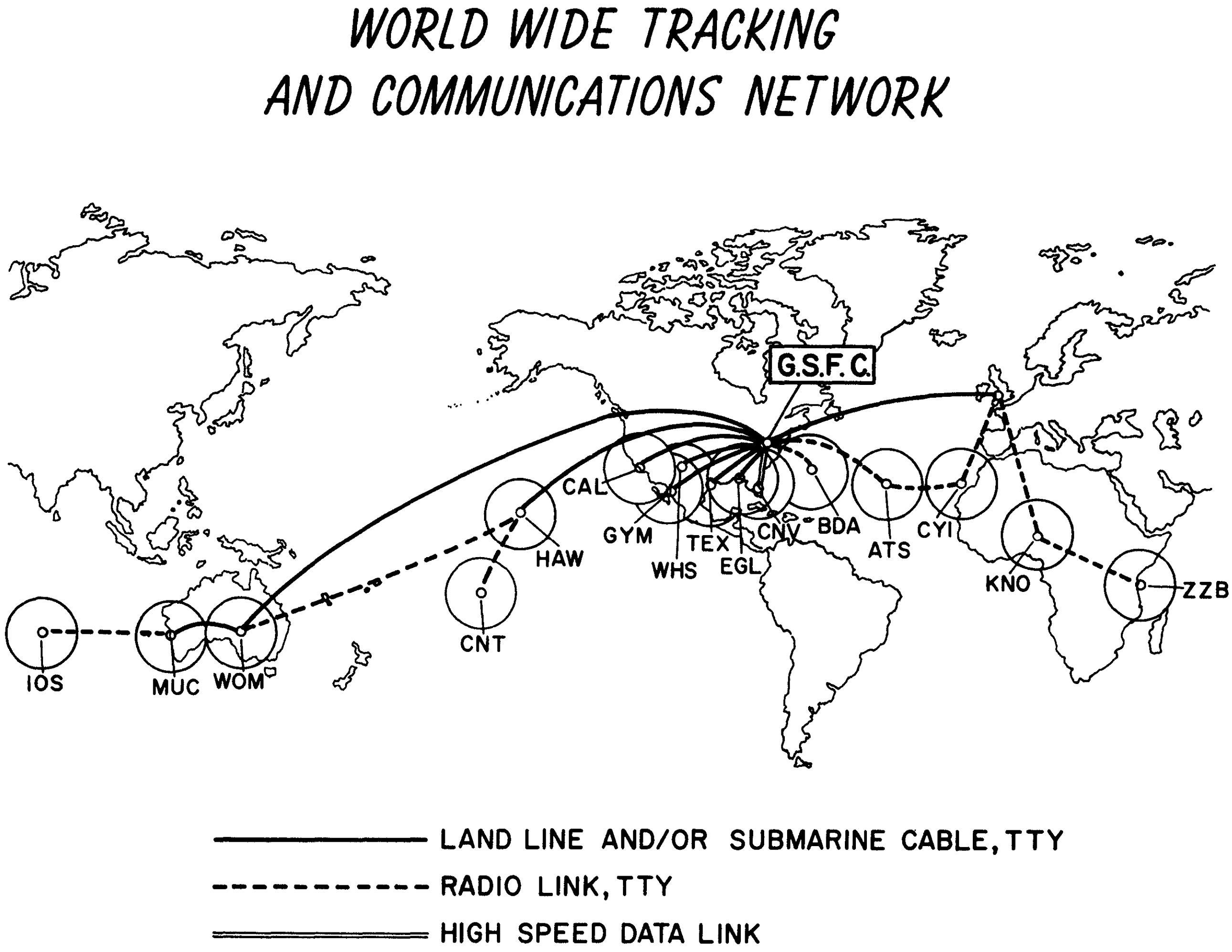


Figure 1

2. a. The output of the Impact Predictor (IP) Computing System interleaved with telemetry and discrete event information to the GSFC for processing and subsequent output back to the MCC plotboards, digital displays, wall map, and Data Quality Monitoring equipment.
   b. The raw radar tracking information from the AMR sites may be processed in lieu of the IP output over the same hi-speed circuits to GSFC and thence as before, computed for display quantities for transmission back to the Mercury Control Center.

To accomplish the above, during the Launch phase, unique and special configurations of equipment are required—operating at relatively high speed and in real-time. Special consideration was given to the reliability of the LMSS which resulted in a duplexed system configuration and redundant equipment as well as back up data sources.

B. Detailed Data Flow and Functional Aspects of Equipment (Figure 4)

As indicated in A.1.a and A.2.a, source information from each of two separate computing systems is combined with common telemetry (T/E) and discrete event information and sent to GSFC for processing.

The T/E buffer processes Binary Coded Decimal clock information (Spacecraft Elapsed Time and Retrofire mechanism setting) plus discrete event information (Liftoff through Retro rockets fired) cyclically each 74 milliseconds in a serial mode to the Atlas Guidance and IP Buffers respectively. In addition, the parallel 24 bit output of the Atlas

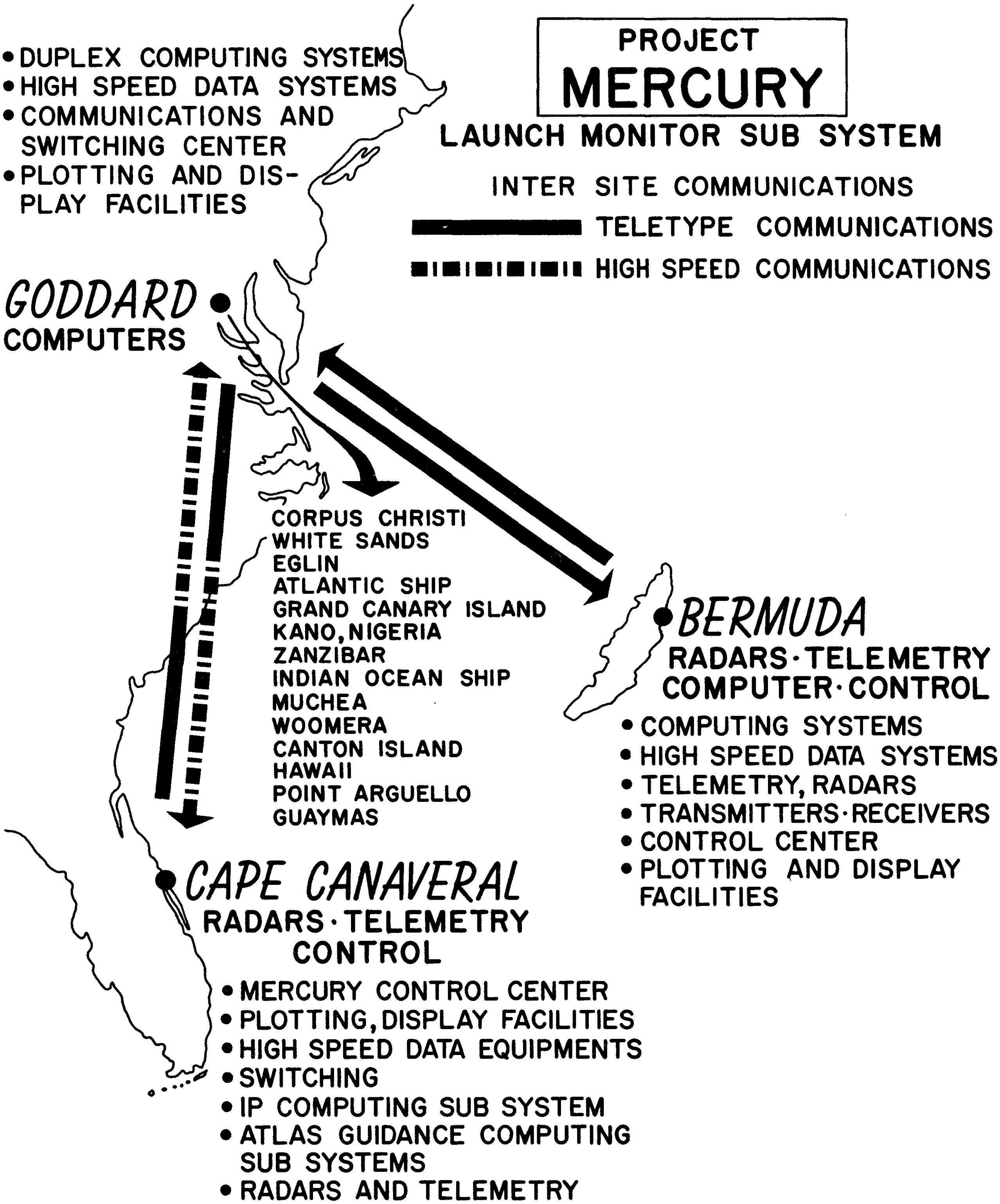

PROJECT
MERCURY
LAUNCH MONITOR SUB SYSTEM
INTER SITE COMMUNICATIONS
TELETYPE COMMUNICATIONS
HIGH SPEED COMMUNICATIONS
• DUPLEX COMPUTING SYSTEMS
• HIGH SPEED DATA SYSTEMS
• COMMUNICATIONS AND SWITCHING CENTER
• PLOTTING AND DISPLAY FACILITIES
GODDARD
COMPUTERS
CORPUS CHRISTI
WHITE SANDS
EGLIN
ATLANTIC SHIP
GRAND CANARY ISLAND
KANO, NIGERIA
ZANZIBAR
INDIAN OCEAN SHIP
MUCHEA
WOOMERA
CANTON ISLAND
HAWAII
POINT ARGUELLO
GUAYMAS
BERMUDA
RADARS·TELEMETRY
COMPUTER·CONTROL
• COMPUTING SYSTEMS
• HIGH SPEED DATA SYSTEMS
• TELEMETRY, RADARS
• TRANSMITTERS·RECEIVERS
• CONTROL CENTER
• PLOTTING AND DISPLAY FACILITIES
CAPE CANAVERAL
RADARS·TELEMETRY
CONTROL
• MERCURY CONTROL CENTER
• PLOTTING, DISPLAY FACILITIES
• HIGH SPEED DATA EQUIPMENTS
• SWITCHING
• IP COMPUTING SUB SYSTEM
• ATLAS GUIDANCE COMPUTING SUB SYSTEMS
• RADARS AND TELEMETRY

Figure 2

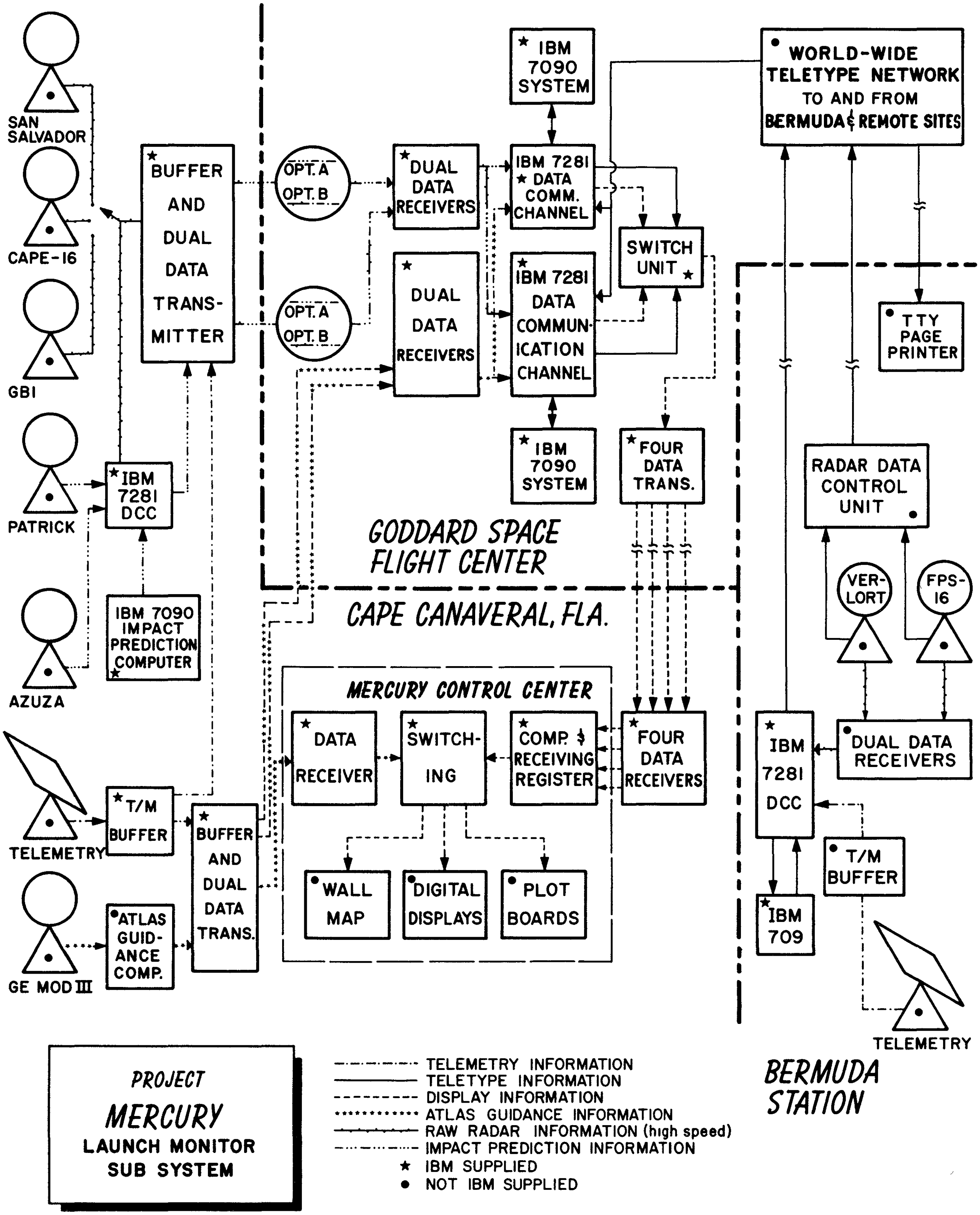
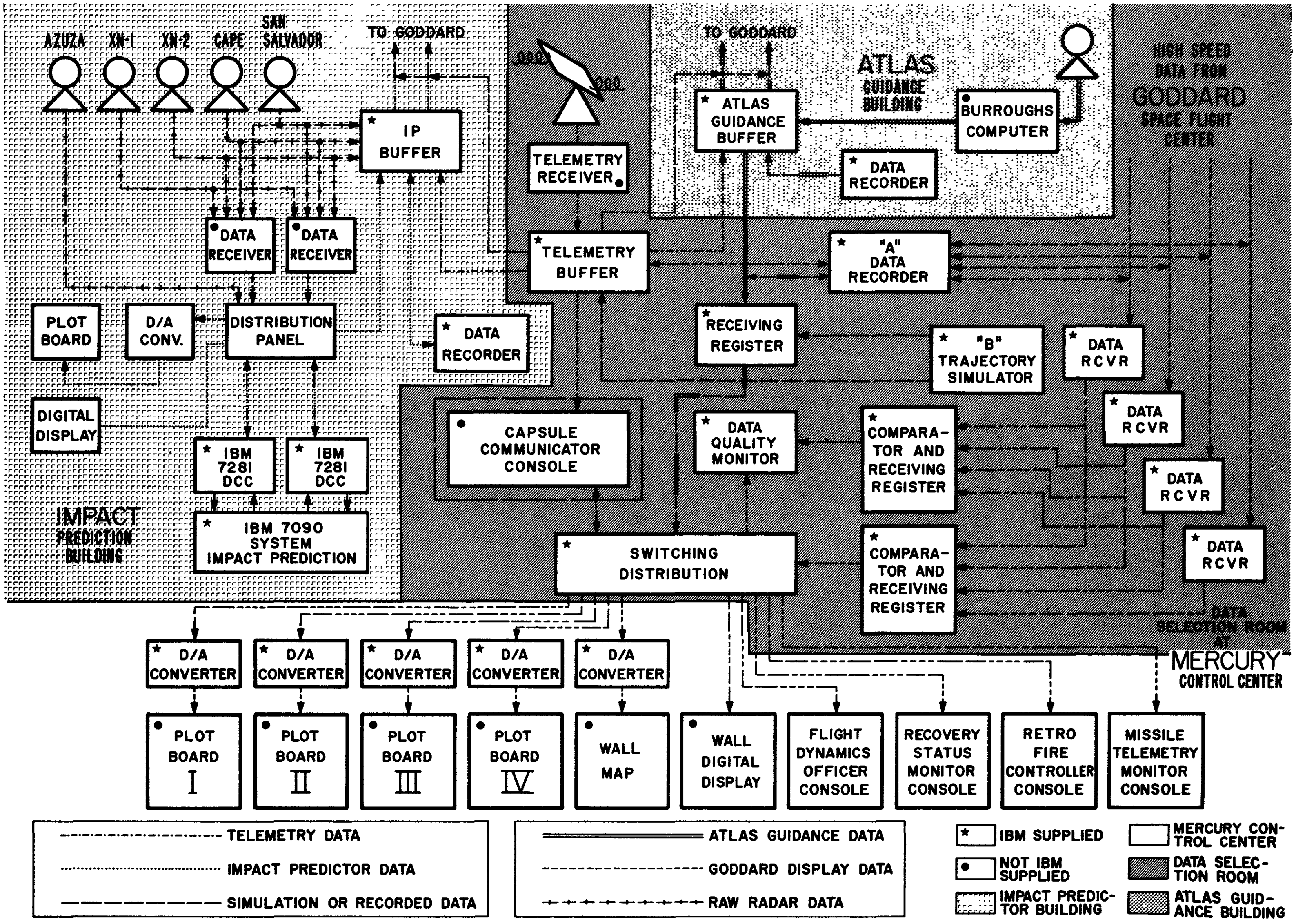

SAN SALVADOR
CAPE-16
GBI
PATRICK
AZUZA
TELEMETRY
GE MOD III
★ BUFFER AND DUAL DATA TRANS-MITTER
★ IBM 7281 DCC
IBM 7090 IMPACT PREDICTION COMPUTER ★
★ T/M BUFFER
● ATLAS GUID-ANCE COMP.
★ BUFFER AND DUAL DATA TRANS.
OPT. A
OPT. B
OPT. A
OPT. B
★ DUAL DATA RECEIVERS
★ DUAL DATA RECEIVERS
★ IBM 7090 SYSTEM
IBM 7281 ★ DATA COMM. CHANNEL
★ IBM 7281 DATA COMMUN-ICATION CHANNEL
★ IBM 7090 SYSTEM
SWITCH UNIT ★
★ FOUR DATA TRANS.
● WORLD-WIDE TELETYPE NETWORK TO AND FROM BERMUDA & REMOTE SITES
● TTY PAGE PRINTER
GODDARD SPACE FLIGHT CENTER
CAPE CANAVERAL, FLA.
MERCURY CONTROL CENTER
★ DATA RECEIVER
★ SWITCH-ING
★ COMP. & RECEIVING REGISTER
★ FOUR DATA RECEIVERS
● WALL MAP
● DIGITAL DISPLAYS
● PLOT BOARDS
RADAR DATA CONTROL UNIT ●
VER-LORT
FPS-16
● DUAL DATA RECEIVERS
★ IBM 7281 DCC
● T/M BUFFER
★ IBM 709
TELEMETRY
BERMUDA STATION
PROJECT MERCURY LAUNCH MONITOR SUB SYSTEM
TELEMETRY INFORMATION
TELETYPE INFORMATION
DISPLAY INFORMATION
ATLAS GUIDANCE INFORMATION
RAW RADAR INFORMATION (high speed)
IMPACT PREDICTION INFORMATION
★ IBM SUPPLIED
● NOT IBM SUPPLIED

Figure 3



MERCURY CONTROL CENTER

Figure 4

Guidance Computer and the parallel 36 bit output of the IP Computer are transferred in microseconds to their respective buffers. At each of these two buffers, serial message units are arranged as outputs. These consist of successive Position and Velocity Vectors describing the trajectory, plus discrete T/E information indicating the phase of the mission and whether certain other events such as BECO, SECO, etc. have occurred. The T/E portion also contains the CET and Retrofire mechanism settings. These buffers are independent of each other and operate in a shuttle mode. That is, while the T/E portion of a message is being transmitted towards GSFC, the high speed transfer of data from the respective Computing Complex is being effected. Subsequently, while the Trajectory portion of the data is being transmitted, the T/E data is being stored for retransmission. Duplexed output transmission is accomplished at the rate of 1000 bits per second from each buffer by transmitters over 3KC (nominal) voice circuits to GSFC.

For the Atlas Guidance Buffer a complete message consists of 384 bits composed of two 192 bit subframes. Because the subframes contain different information and consequently must be processed differently at GSFC, each subframe is identified. The nominal rate for repeating the complete message frame is once each 500 milliseconds. In addition, the Atlas Guidance Buffer processes information serially at a 1000 bits per second rate to a receiver and associated 128 bit receiving register located in the MCC. This process is repeated each 500 milliseconds. The inputs to this unit are converted into parallel DC levels and presented by functional grouping to the Switch Unit for further distribution to the Digital-to-Analog Converters (D/A's) associated with Plotboards I, II, III, and the Flight Dynamic Officer's Console (FDO). As will be discussed in more detail later, the Switch Unit routes sources of data being fed to the various display facilities at the MCC under remote switch control of the Data Quality Monitor Console. Several factors have been utilized to insure reliability of data transmission. These are: the use of check sums associated with the Atlas Guidance and IP Computer data, the use of parity bits in the T/E data format, plus redundancy of bit positions for certain critical discrete events. In addition, the duplexing and separate geographical routing of hi-speed transmitting lines between Cape Canaveral and GSFC in Maryland was accomplished. Also, comparable messages are time offset, one from the other, on each of the duplexed lines to insure that loss of data due to transmission difficulties is minimized. The transmission link from Cape Canaveral to GSFC is about 1000 miles, which results in an approximate 10 millisecond delay in transmission.

The IP Buffer processes information to GSFC in exactly the same manner as the Atlas Guidance Buffer, operating on a different time cycle of about 400 milliseconds, using the same safeguards as above with respect to reliability. In addition, however, there is the possibility that due to malfunction or loss of output from the IP Complex, it may be desirable to utilize raw radar returns directly. In this case, during the transient switching time from IP Computer output to "raw radar messages" composed of range, azimuth, and elevation, the Buffer output would become 0's interspersed with 1's at specific locations in the serial data chain to indicate to the GSFC computers that faulty data was being received. The raw radar format and transmission method is similar to the preceding, with the added distinction of identifying the radar source in addition to the data subframe.

All high speed information arriving at GSFC (refer to Figure 5) enters a two distinct and complete duplexed IBM 7090 computing systems. The specific entry is accomplished via individual hi-speed receivers operating from each of the 4 hi-speed lines into the duplexed real-time input-output units called Data Communications Channels (IBM 7281-DCC's). The function of the hi-speed receiver is to convert the serial data transmission into parallel digital impulses, at the same time resynchronizing the data to eliminate the time shift and converting the data into appropriate parallel 16 bit words for presentation to the two hi-speed input subchannels within the DCC. These 16 bit words are composed of two 8 bit groups representing a duplexed output from each of the two buffers located at Cape Canaveral. In the event that one 8 bit group does not arrive, within certain tolerances, concurrent with the other 8 bit group, logic circuitry enables the leading receiver to transfer its 8 bit group into the computer and inhibit the other.

The DCC enables its respective IBM 7090 Computing System to accept asynchronous

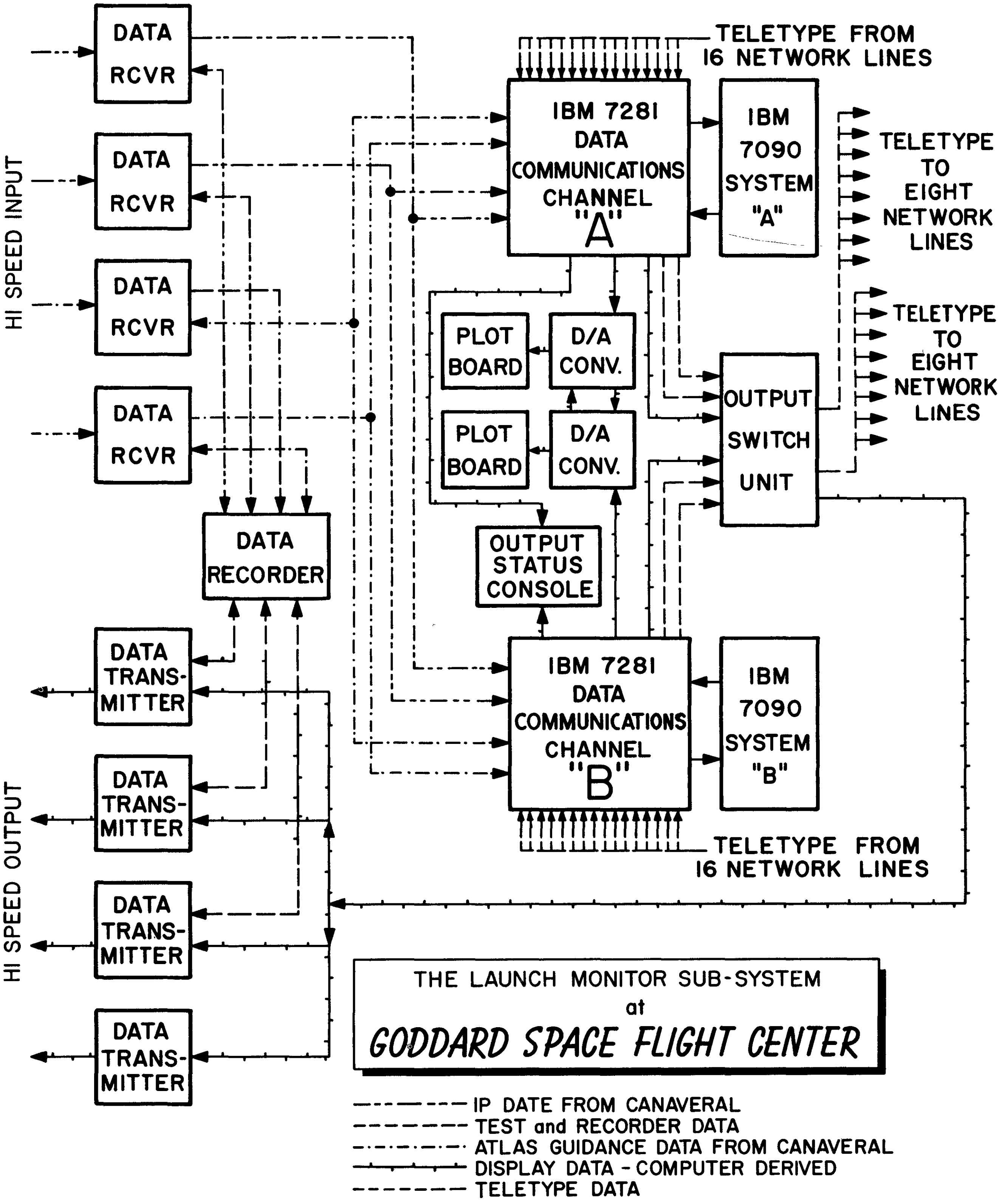

HI SPEED INPUT
DATA RCVR
DATA RCVR
DATA RCVR
DATA RCVR
DATA RECORDER
HI SPEED OUTPUT
DATA TRANSMITTER
DATA TRANSMITTER
DATA TRANSMITTER
DATA TRANSMITTER
TELETYPE FROM 16 NETWORK LINES
IBM 7281 DATA COMMUNICATIONS CHANNEL "A"
IBM 7090 SYSTEM "A"
TELETYPE TO EIGHT NETWORK LINES
PLOT BOARD
D/A CONV.
PLOT BOARD
D/A CONV.
OUTPUT SWITCH UNIT
TELETYPE TO EIGHT NETWORK LINES
OUTPUT STATUS CONSOLE
IBM 7281 DATA COMMUNICATIONS CHANNEL "B"
IBM 7090 SYSTEM "B"
TELETYPE FROM 16 NETWORK LINES
THE LAUNCH MONITOR SUB-SYSTEM at GODDARD SPACE FLIGHT CENTER
IP DATE FROM CANAVERAL
TEST and RECORDER DATA
ATLAS GUIDANCE DATA FROM CANAVERAL
DISPLAY DATA - COMPUTER DERIVED
TELETYPE DATA

Figure 5

data in real-time via independent subchannels from a diverse number of sources. These subchannels are designed to accommodate arbitrary message formats and store real-time received data in assigned sections of the Computer's memory with minimal interference to the operating program. Associated with each of the GSFC DCC's are the following inputs: 2 hi-speed 16 bit parallel input subchannels; 16 subchannels, of a 5 bit parallel low speed input subchannel for operation with the remote sites via teletype (TTY) link; and one 5 bit parallel low speed subchannel for local TTY paper tape entry.

The clock subchannels make available: a one minute interrupt, a half second program interrupt, a binary counter continuously incremented at 8.33 milliseconds intervals, and an interval timer binary counter operating at the same increments. In addition, the DCC has certain output capabilities as follows: three 5 bit parallel output subchannels having the ability to address individually each of 8 terminal TTY output devices for processing slow speed acquisition messages via the world-wide communications network to the remote radar and telemetry tracking stations; 2 serial output hi-speed subchannels to present, at 1000 bits per second, display data to the MCC; one sense output subchannel under program control, locally and to operate indicator lights which denote certain status information.

The Mercury operational program, as discussed in detail elsewhere in this volume, processes the hi-speed launch data utilizing either the Atlas Guidance, IP Computer Complex or Raw Radar as its prime source. In any case, local as well as MCC data is computed and made available. The outputs of the duplexed computer complexes are continuously monitored by means of the Output Status Console, local Plotboards, and on-line Printouts. Inasmuch as it is not the purpose of this paper to discuss the programming aspects of the Mercury system, we shall confine ourselves to generalities as to what is displayed and where.

Output Status Console

This console provides a set of indicators permitting the Console operator to judge the progress of each computer's output. The console also contains the necessary controls to permit selection of the computer output desired via the output Switch Unit. In addition, the Output Status Console contains relay switching to effect functional interchange of the local Plotboards and their associated D/A's.

Plotboards

There are two X-Y plotters at GSFC, each driven by a D/A converter. Information plotted during launch consists of the Flight Path Angle versus Velocity Ratio and during orbit, longitude versus latitude of present position. A monitoring person stationed at each observes the plotted values to form a judgment of computer output quality. Each monitor reports his observations to the Output Status Console Operator.

Each duplexed 7090 operates upon all input data so that in the event of malfunction of one portion of a computer complex, the other can be switched under control of the Output Status Console Monitor to provide all output data.

Function plots of Flight Path Angle versus Velocity Ratio leave either computer via one of the hi-speed output subchannels of the DCC (in serial fashion) and are converted to analog signals by the respective D/A converter for presentation to its associated Plotboard. The other hi-speed output subchannel is processing digital information serially at high speed to the common input of the 4 Data Transmitters. A complete message consisting of Odd and Even frames of data (408 bits per frame) is transmitted at the rate of 1000 bits per second, each second during Launch and each two seconds during Abort, five times a minute during Orbit and ten times a minute during Re-entry. This information is transmitted over four special equalized, separately routed, telephone circuits to Cape Canaveral. The data on each of the 4 lines is identical although deliberately offset in time. It should be noted also that the format for all messages sent from GSFC is constant for all phases of the mission, the only difference being the repetition rate and the contents, whose variation can be explained by the fact that certain information is useful in one flight Phase, but not another.

All data from GSFC is received in the data selection room, MCC at Cape Canaveral (refer to Figure 4). Each of the four Data Receivers at the MCC is connected to one of these lines. The receivers remove the time differentials and convert the transmitted data into d-c levels suitable for the operation of the digital equipment and displays. The output of three of the receivers is connected to duplexed comparator and receiving register units. The comparator unit dynamically compares the bits of the three receivers (bit by bit) and accepts as valid any two that agree. The validated bits are then converted from serial to parallel form, suitable for distribution by the switching system to the proper displays. A secondary function of the comparator is to make a comparison of the validated bit with each receiver output (bit by bit) pulsing a meter associated with each receiver when a discrepancy is noted and, in addition, actuating an alarm whenever the error rates exceed a specified level. (It should be noted that any 3 of the 4 data receivers are readily selected for operational use.)

The Receiving Register contains 408 bits for conversion purposes and has a storage register capacity of 579 bits to store the output for alternate frames of data.

The Switch Unit is the device, under control of the Data Quality Monitor (DQM) and Console that routes information to the proper displays and plotboards during different phases of the flight. It should be noted, however, that certain displays and plotboards are unaffected by switch action. Specifically this unit:

a. Accepts the duplexed outputs of the Comparator-Receiving Registers and under switch control from the DQM console, selects one set of outputs to feed all digital displays, the D/A converters for the wall map display, Plotboard IV, and the DQM, as well as the relays which select the Atlas Guidance Computer source via GSFC or the IP Computer source via GSFC for Plotboards I, II and III.

b. Selects under switch control from the DQM, Atlas Guidance or GSFC Computer source data to feed the GO-NO GO recommendation indicator on the FDO console and the D/A converters for Plotboards I and II.

c. Selects under switch control from the DQM console, Atlas Guidance or GSFC Computer data to feed the D/A converter for Plotboard III.

Inasmuch as the DQM and Console plays an important part in the selection of data sources during different phases of the flight, it behooves us to take a quick look at its functions and capabilities. As we have seen, certain of the data displayed at MCC during the launch phase may be produced by several alternate means: The Atlas Guidance Computer; the GSFC Computer with Atlas Guidance data, IP Computer data, or raw radar data as source. The Data Selection Supervisor determines which of these means will be used to present the data displayed. The DQM assists him by enabling rapid visual comparison of two variable quantities as produced from each of the sources. These variable quantities are the deviation from the velocity ratio and the deviation of the flight path angle from nominal values. These quantities are displayed on the strip chart associated with the DQM and updated each 1/2 second. The DQM console is an adjunct to the DQM and is used for monitoring certain display quality indications as well as providing the specific switch action for source data selection by the Data Selection Supervisor. Briefly, 4 data quality signals are displayed indicating reliability of data generated by the Atlas Guidance Computer. In addition, four status signals indicate sources of data currently being transmitted to other displays throughout the MCC. Also, of course, the necessary control signals must be developed to accomplish the switching function previously outlined.

Specifically, the Atlas Guidance Computer output is functioning as follows when indicators are lighted:

1st indicator - Computer integrating rates of change to obtain range, azimuth and elevation.

2nd indicator - Computer differentiating range, azimuth and elevation to obtain rates of change.

3rd indicator - Computer differentiating track data to obtain lateral rates only.

4th indicator - Computer is not receiving sufficient data to generate guidance commands.

The switch control on the DQM console, in addition to determining data sources as previously discussed, also initiates signals to the GSFC computers as follows:

A signal is sent back via the T/E transmitting buffer indicating whether the GSFC computers are to use Atlas Guidance or IP Computer data as the prime source for deriving display quantities.

A signal is used to indicate to the GSFC computers via the T/E transmitting buffer what flight phase the mission has achieved.

Assuming that the proper switching has taken place, the following ground rules as outlined in Figure 7 are observed with respect to data sources and the various digital displays and plotboards.

All Digital Displays at the MCC as follows, although not considered specifically a part of the LMSS, are driven by the GSFC Computing Complex:

Flight Dynamics Officers Console (FDO), Retrofire Controllers Console, Recovery Status Monitor Console, Wall Digital Display, Wall Map, and Capsule Communications Console.

The Launch Vehicle Telemetry Monitor displays BECO (Booster Engine Cutoff) and SECO (Sustainer Engine Cutoff) as received directly from Booster Telemetry.

III. Insertion to Orbit and Re-Entry

During the period following insertion, radar data will be received at GSFC via TTY transmissions from Bermuda, Grand Canary, Muchea, Woomera, Hawaii, Point Arguello, White Sands, Eglin, Guaymas, and Corpus Christi, and telemetry data will be received from all these sites and from Kano, Zanzibar, Canton Island, Atlantic Ship and Indian Ocean Ship. Although all the above stations and information to GSFC Computing Complex, only the 709 Computer Complex at Bermuda and associated data flow are considered part of the LMSS. In the following paragraphs this complex and its communication with GSFC will be briefly discussed:

A. Bermuda Computing and Data Flow Complex (Refer to Figure 6)

The Bermuda Mercury station is a combination Radar, Telemetry, Control Center and Computing Site. The Bermuda station is strategically located relative to the orbital insertion point during a mission and should have excellent radar and command control. Consequently, this site is able and has responsibility to make a final GO-NO GO decision as an operational backup for the function of the MCC at Cape Canaveral.

At Bermuda, Radar data is acquired via two precision radars, an FPS-16 (C Band) and a Verlort (S band). An anolog to digital converter, associated with each radar, produces Range, Azimuth, and Elevation information in parallel binary form from the Verlort, and serial binary form from the FPS-16. Digital to TTY converters and transmitting equipment place the data in the appropriate format for transmission to GSFC. A choice of prime sources is made (FPS-16 or Verlort) by the Ground Communication Coordinator at Bermuda. Normally, the Verlort would acquire the spacecraft first and be so selected. However, due to its greater accuracy, as soon as the FPS-16 receives valid data, it becomes the prime source and will remain so until it no longer receives valid data, at which point reversion will be made back to the Verlort. In any case, the radar source not selected has its data stored on paper tape for later transmission after each pass. Data selection (radar source) is handled logically in the same manner with respect to hi-speed computer input at Bermuda.

The binary digital data from each of these radars is presented to a transmitter whose output is a 1000 bits/second and thence are received in the computer complex by hi-speed receivers which reconvert the serial information back to parallel binary digital data and present it, 8 bits at a time (from each radar), into a 16 bit hi-speed input subchannel of the DCC at Bermuda. Synchronization of the bit-by-bit transfer from the individual hi-speed transmitters is accomplished by pulsing from an on-site time standard synchronized to WWV. The rate of transfer from the radars to the DCC is 10 frames per second.

The DCC at Bermuda is similar to those at GSFC. The major difference consists of a T/E Buffer input subchannel which accepts data in 8 bit parallel words at the rate of one word per sequence scan under program control. As at GSFC, there is one paper tape input and three subchannels for real-time reference. Three subchannels have been allocated to handle output data, including a Sense Output Subchannel furnishing data to an Output Status Console, a TTY output subchannel which sends data to a TTY distributor, and a hi-speed output subchannel for sending data to a receiving register, which in turn, operates into a D/A converter and thence to a plotboard. Additionally this receiving register provides for a parallel readout to the Control Center Digital Displays.

The operational programs for the Bermuda 709 Computer are under control of a Monitor

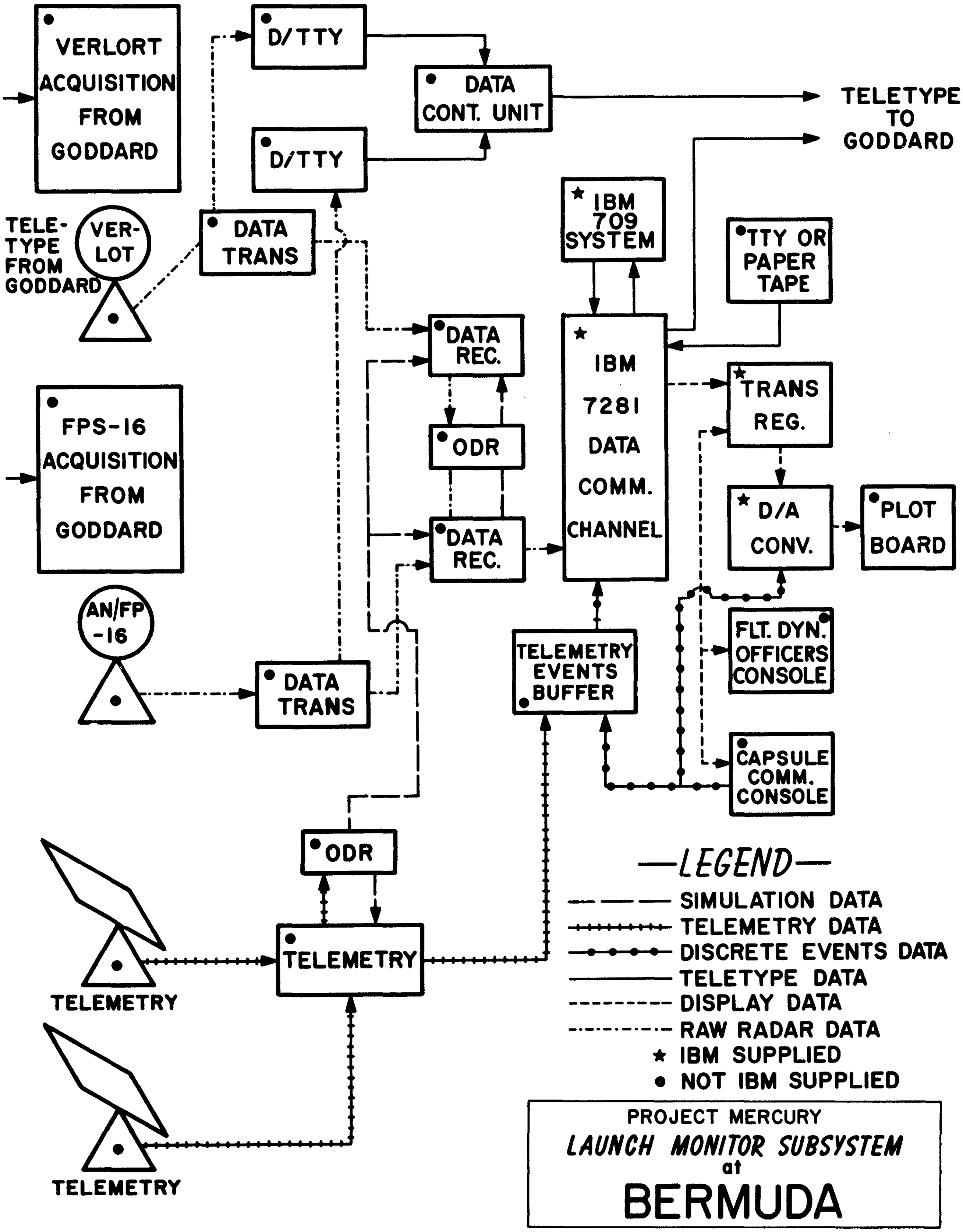

VERLORT ACQUISITION FROM GODDARD
D/TTY
D/TTY
DATA CONT. UNIT
TELETYPE TO GODDARD
TELE-TYPE FROM GODDARD
VER-LOT
DATA TRANS
IBM 709 SYSTEM
TTY OR PAPER TAPE
DATA REC.
IBM 7281 DATA COMM. CHANNEL
TRANS REG.
FPS-16 ACQUISITION FROM GODDARD
ODR
D/A CONV.
PLOT BOARD
DATA REC.
AN/FPS-16
FLT. DYN. OFFICERS CONSOLE
TELEMETRY EVENTS BUFFER
DATA TRANS
CAPSULE COMM. CONSOLE
ODR
TELEMETRY
TELEMETRY
TELEMETRY
—LEGEND—
SIMULATION DATA
TELEMETRY DATA
DISCRETE EVENTS DATA
TELETYPE DATA
DISPLAY DATA
RAW RADAR DATA
★ IBM SUPPLIED
● NOT IBM SUPPLIED
PROJECT MERCURY
LAUNCH MONITOR SUBSYSTEM
at
BERMUDA

Figure 6

SOURCE DATA

PLOTBOARD #1	Launch	Abort, Orbit & Re-entry
Launch-Velocity ratio versus Flight Path Angle Abort-Velocity versus height above spherical earth Orbit & Re-entry-same as abort	Atlas Guidance Complex or GSFC Complex a. Atlas Guidance Source b. Impact Predictor Source c. Raw Radar	GSFC Complex
PLOTBOARD #2		
Launch-Horiz. distance versus height above oblate earth Abort-Horiz. distance versus Flight Path Angle deviation Re-entry-Elapsed time versus height Orbit-Elapsed time versus height (spherical earth)	Atlas Guidance Complex or GSFC Complex a. Atlas Guidance Source b. Impact Predictor Source c. Raw Radar	GSFC Complex
PLOTBOARD #3		
Launch-Elapsed time versus velocity, Time to go versus velocity deviation Elapsed time versus acceleration (T/M) Time to go versus height of insertion Orbit-Elapsed time versus perigee of longitude Elapsed time versus orbit eccentricity	Atlas Guidance Complex	GSFC Complex
PLOTBOARD #4		
Launch-Longitude versus Latitude (impact point) Longitude versus Latitude for IP of retrofire in 30 seconds Longitude versus Latitude for IP of retrofire using maximum delay Abort-Longitude versus Latitude (impact point) Longitude versus Latitude (present position) Orbit-Longitude versus Latitude for IP of retrofire in 30 sec. Longitude versus Latitude (present position) Re-entry-Longitude versus Latitude (impact point) Longitude versus Latitude (present position)	Atlas Guidance Complex or GSFC Complex a. Atlas Guidance Source b. Impact Predictor Source c. Raw Radar	GSFC Complex

IP = impact point

Program which was written specifically for Bermuda's role in the Mercury mission. By sampling the information provided via the Telemetry input subchannel, the Monitor Program follows the status of the mission, controls Input and Output data processing and by flight progress status temporarily transfers control to the proper subroutine for necessary computations. Prior to SECO + 10 seconds, the computer uses a modified least squares method for curve fitting for output to local displays. After this time, a short arc determination is utilized until the spacecraft is out of range. By means of the hi-speed output subchannel, the following display quantities are provided locally:

Bermuda Plotboard (Control Center)

Launch	Velocity ratio versus Flight Path Angle Longitude versus Latitude (impact point-immediate retrofire)
Abort (after retrofire)	Longitude versus Latitude (impact point) Longitude versus Latitude (present position)

Digital Displays on the Bermuda Flight Dynamics Officers Console are: altitude above oblate earth, flight path angle, time to retrofire, Velocity ratio, ICTRC-incremental change to achieve proper retrofire (Computer recommended), GMTRC-GMT for retrofire computed, ECTRC-Elapsed capsule time for retrofire-computed, Recovery area, and the Computed GO-NO GO recommendation.

In addition to the local displays, the Bermuda Computer sends via its TTY subchannel, smoothed composite values of Range, Azimuth and Elevation to the GSFC Computer Complex.

All double radar sites operate in the manner described before in ultimately sending data to GSFC. Single radar sites begin transmission of data as soon as valid "on track" contact is made with the capsule and continue until such contact is lost. Generally speaking with regard to TTY communication facilities to GSFC, there are at least two separate geographic routings for transmission reliability. In addition to the raw radar information, telemetry summary information is also transmitted to GSFC where an extract of particular data relative to the time parameters of CET and Retrofire Mechanism setting is entered to the Computers via the Paper Tape input subchannel of the DCC.

During orbit, radar acquisition messages are processed and repeatedly updated prior to spacecraft arrival into the radar zone of each succeeding site. These messages, containing radar antenna pointing information are received at the site by a TTY receiver and presented to the Radar Acquisition Aid Operator.

A SIMULATION MODEL FOR DATA SYSTEM ANALYSIS

Leon Gainen
The Rand Corporation
Dayton, Ohio

SUMMARY

The author asserts that designing a data system to support management objectives is, at present, no more than a highly specialized art. This paper then develops the thesis that data system designers can bring their profession closer to a predictive science. Only by adapting for data system analysis purpose analytical tools which make possible prediction and quantification of data system behavior within the management system structure can this be done.

This paper discusses one such tool, a generalized data system model, and describes a technique of simulating dynamic system operation with such a model in order to provide the data system designer insights on the behavior to expect from the data system as it would operate. Some of the benefits possible through such simulation are explored. The paper concludes that the major use for the present of this analytical technique is to test the feasibility of a data system design before acquisition of actual hardware.

I. Introduction

The main object of this paper is to promote the use of simulation techniques for the analysis of data systems that have been designed for management information processing. I will describe our research goals at The RAND Corporation for the development of an all-machine data system analysis model designed for this purpose. Before doing so, I shall present some views on the necessity for employing more objective analytical tools to improve the quality of data system design, and indeed, to bring our profession closer to a science.

A data system designer tries to optimize the use of data processing techniques and equipment as tools of a management system. Accordingly, one must recognize that optimizing the methods and means of information flow is not the objective of the management system. Nevertheless, this subject area is worthy of serious study because of the great potential inherent in present and promised data processing hardware to shape the management system objectives and, therefore, to enhance the concepts of management. It is also a nontrivial fact of life that data processing elements, e.g., data generators, communications devices, computers, are the most costly segments of the modern, complex management systems. Thus, it behooves a data system designer to consider the tradeoffs between contribution and cost for each data processing element in any proposed management information system.

The technique of system simulation has proven invaluable for research and system analyses in many technical fields but has largely been ignored by data system designers. Yet, every data system design proposed to management is in the nature of an hypothesis, since we cannot claim to have studied data system design theory sufficiently to have established incontravertable laws in this field. Even when data system design is expertly done, present techniques cannot in any quantitative way assure the "best" system has been achieved, or even that a system will operate as reasonably as can be expected under the time and dollar constraints imposed by management. How can data system designers prove the merit of their assertions if they are often based on visionary concepts, hypothetical equipment, or revolutionary departures from standard procedures? Indeed, in the light of present disenchantment by some managements with their data processing systems, one may ask if data system designers can prove even the feasibility of their handiwork.

I contend that data systems can be modeled to sufficient detail to allow investigation of significant data system design parameters, and that dynamic system operation can be simulated. Data system performance studied through simulation can provide insight on system behavior long before actual hardware acquisition. In this way, modification in system design can be made before it becomes too late to make a change. If data systems are to be adequately analyzed before being cast in hardware, then it would appear that simulation of system operation is required in order to weigh properly how well the uncertainties and the variabilities that usually underlie system design parameters have been accounted for. Furthermore, as long as proven analytical tools are available, data system designers cannot continue to rely on the *ipse dixit* approach for solving their theoretical problems. Hopefully, our research will determine how well simulation can improve data system design.

II. The Data System

Let me first describe the data system we are considering for representation in our model. The data system is the set of hardware and human resources plus the procedures defined that either generate, record, communicate, process, store or report data and information to prescribed management system entities in a prescribed form and format. Figure 1 is a schematic of a generalized data system, with each of the above functions symbolized by a particular geometric shape. It is not necessary for each of these functions explicitly to occur in the operation of a data system as, for example, when data are generated, recorded, and communicated simultaneously to the processing and storage function. For convenience, the first process is the one represented in the schematic of Figure 1.

III. The Problem of Data System Design

In general the problem of data system design is to create a system which optimally uses the resources selected to do the assigned job. This is not a problem, as stated, except that constraints exist to limit the freedom of choice and utilization of candidate resources. Generally speaking, the major constraints are limitation in the number of dollars and the level of skills which can be applied to effecting the desired solution. Data system restrictions also may be imposed by management system objectives. Finally, even if given the freedom to create the best data system possible, there are the underlying uncertainties in the design parameters with which we usually deal. (See Figure 2.)

A balanced set of data system elements would provide an ideal solution to the data system designer's problem. This means that each system function is provided just the correct types and proper mixes of resources to satisfy each function's requirement for data manipulation. But isn't it true that the stochastic nature of data generation, random equipment failures, and differing data system responsiveness required by various management system sub-functions all mitigate against achieving the desired balance in the data system? Thus, to arrive at a data system design which approaches a balanced system requires study of the system giving due consideration to the effect of these factors on the operation of the system.

IV. Structure and Operation of the Data System Model

The structure of the simulation model proposed in this paper for data system analysis is simply illustrated in Figure 3. From this figure it is seen that all the functions of a data processing system are considered. Each of these must be implicitly or explicitly considered in the data system design, and, therefore, is represented in order that one can evaluate its effect on total system performance and its contribution to system costs. These functions are:

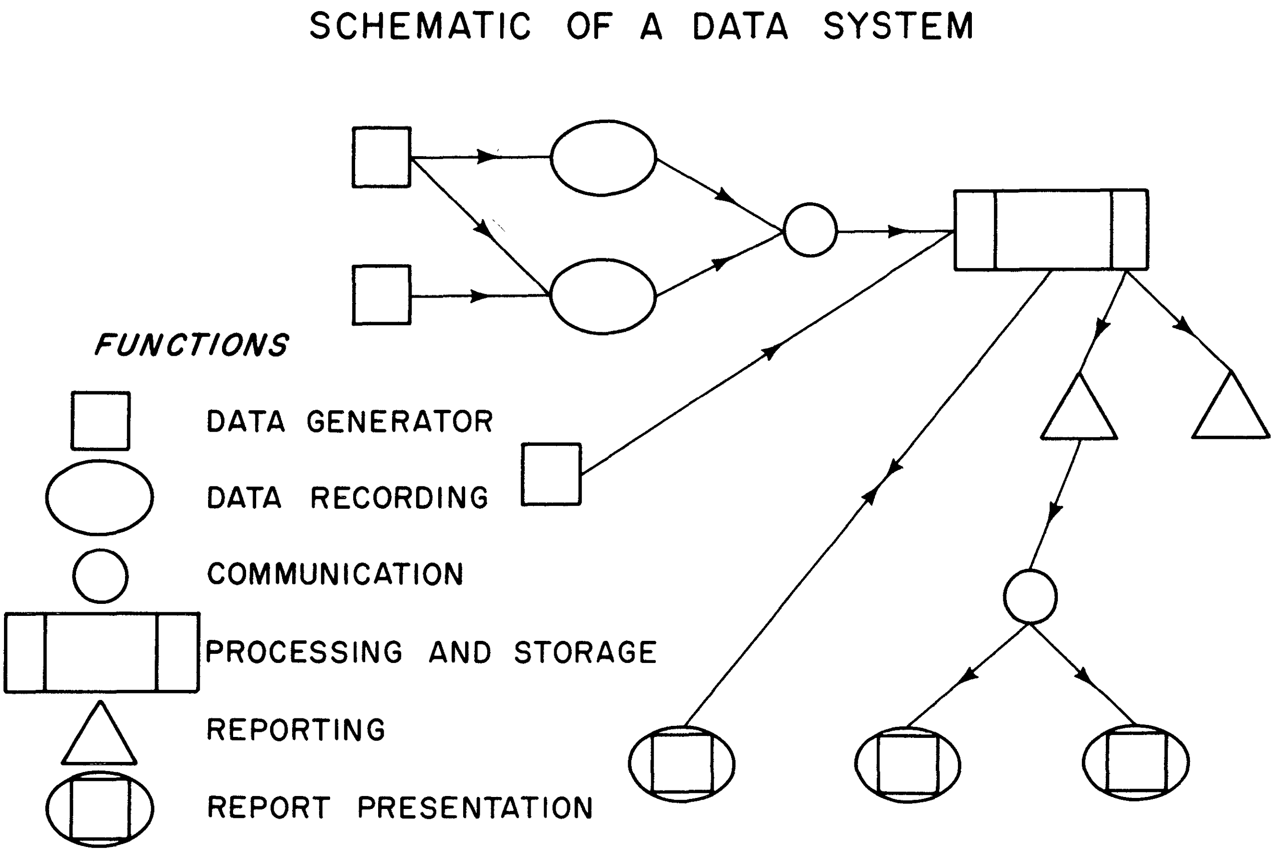
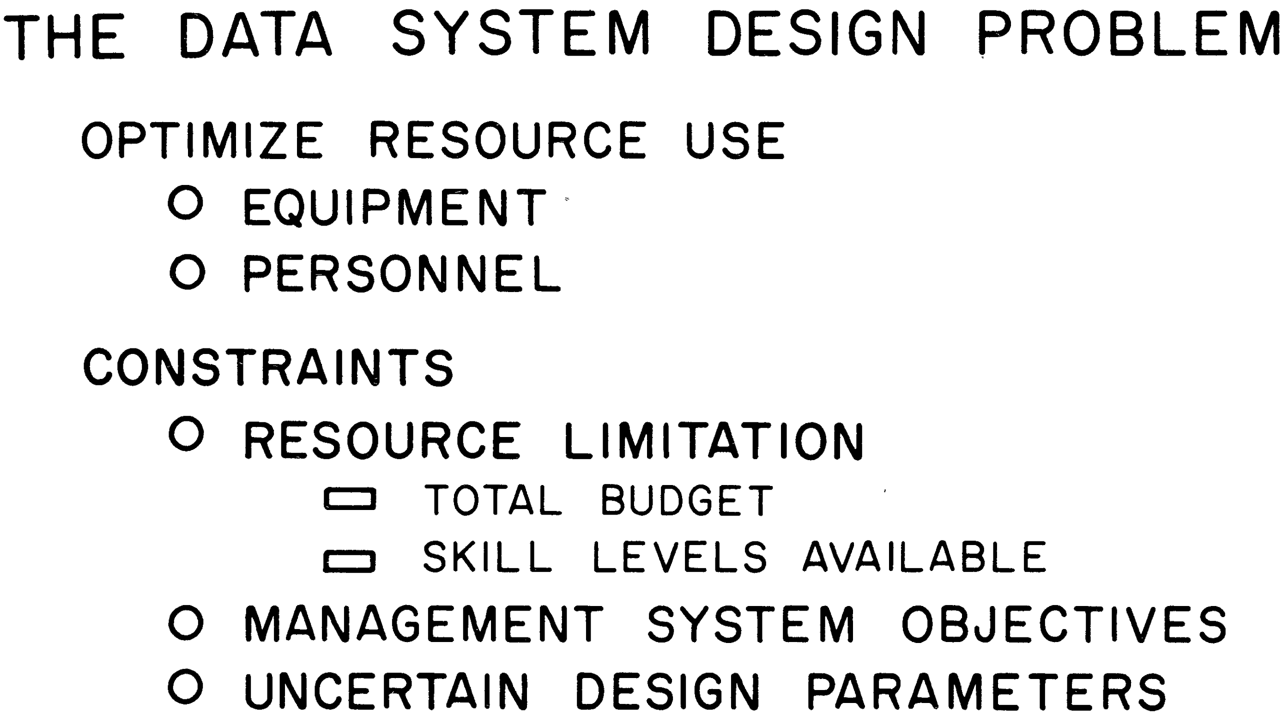


Figure 1. Schematic of a Data System



THE DATA SYSTEM DESIGN PROBLEM

OPTIMIZE RESOURCE USE

- ○ EQUIPMENT
- ○ PERSONNEL

CONSTRAINTS

- ○ RESOURCE LIMITATION
  - ▭ TOTAL BUDGET
  - ▭ SKILL LEVELS AVAILABLE
- ○ MANAGEMENT SYSTEM OBJECTIVES
- ○ UNCERTAIN DESIGN PARAMETERS

Figure 2. The Data System Design Problem

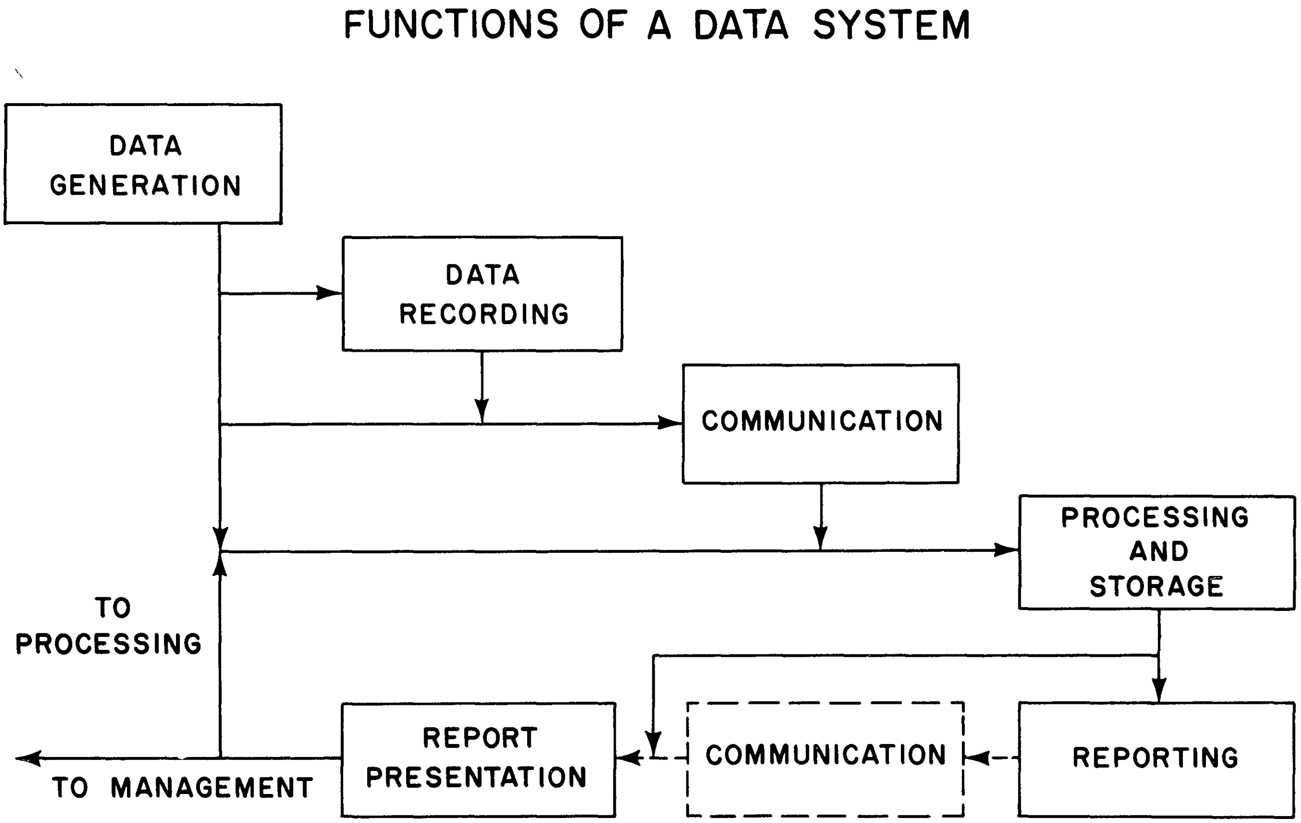


Figure 3. Functions of a Data System

(1) Data generation
(2) Data recording
(3) Communication
(4) Processing and Storage
(5) Reporting
(6) Report Presentation.

Our model is intended to be general enough for one easily to compare various data system proposals for a given management system. Before analysis, however, comes design. Use of our model requires that a management system be studied in sufficient depth for an information set to be described and a resource allocation made in terms of people and/or equipment required for each data system function. Using the same known (or anticipated) rates of data generation, computation, information growth that led to the design of the data system, and the specifications of the human or hardware resources, each data processing function is simulated. When the complete data system cycle is run through the simulation, or when the number of cycles required for analysis is completed, one can assess the effectiveness of each data system segment as part of the whole data processing system, provided that a severe data system bottleneck* has not been exposed during the simulated operation. Costs of a system can be weighed against management system effectiveness for each alternative configuration of data system design. This is done by repeating the analysis using alternative data system design statements. If these alternatives are only in hardware specifications, these new values are the only change required in model input. If an alternative system structure is specified as a change in data flow, the reanalysis is a little more complex.

*When such bottlenecks appear to the analyst to be merely in the number or the quantity of hardware, then (cost constraints allowing) a solution to the design problem is obvious, i.e., get more or better hardware. This solution assumes that the policies of system operation either are not responsible for the bottlenecking, or they cannot be relaxed.

Reanalysis of the system operation would be required with any new hardware assignment in order to insure that the bunching in the data flow had not been passed on to the data system segments which follow.

Simulation Control

The basic control of all events occurring in the simulation of system activity is geared to a master clock which counts the passing intervals of time during the simulation. A management system may include more than a single function. Since the significance of time may vary from one management system function to another, e.g., 15 minutes for inventory control, one-half day for finance, the least significant interval (LSI) of all the functions included in the simulation determines the intervals which the master clock uses in stepping through time.

The model examines all data system events and all equipment simulated, at least implicitly, every least significant time interval of each simulated day's operation. These events include the types and amounts of data input to each data system function, either originating from the function or as a necessary step to another data system function. Demands for equipment are converted from the data input statement, given in terms of message form, frequency and size, to the basic unit of measurement, the LSI, using the given equipment datatransfer rates. For example, if a tape-to-card device is not backlogged and a data workload of less than one LSI of tape-to-card time is demanded at the beginning of an interval, a scheduled job will be considered completed and no backlog for this equipment is carried forward to the next time interval.

Simulated Functions

Data Generation. The locations at which system data are generated often represent an important consideration in the design of an information flow system. Thus, the management functions for which data are gathered and the system locations at which they originate are variables in the data system model. Since in many management systems data generate at one location for many functions, such situations shall be anticipated within the data system model.

The characteristics of generated data that must be described to the model by the system designed for each function-location pairing are those which the data system designer must know in order to prescribe a data system structure. These are listed in Figure 4.

DATA CHARACTERISTICS

○ MESSAGE SIZE

▭ CHARACTERS

▭ UNITS

○ MESSAGE FREQUENCY

○ MESSAGE FORM

○ REPORTS REQUIRING MESSAGE DATA

○ DATA SYSTEM ROUTES

Figure 4. Data Characteristics

RESOURCE CHARACTERISTICS

○ DATA TRANSFER RATES

○ CAPACITY LIMITATION

○ ERROR FUNCTION

○ UNUSUAL FEATURES

Figure 5. Resource Characteristics

A table of resources, characterized as in Figure 5, for each data generation station, e.g.,personnel, equipment,prescribed by the designer includes their data transfer rates and makes possible within the simulation a conversion of data characteristics to simulated workload units, expressed as a number of least significant intervals. In addition, we allow an error function associated with each station in order to simulate resource malfunction, e.g., equipment downtime.

Data processing starts with data generation. For any management system function for which the present interval is a significant time interval we determine the amount and type of activity data generated at each location. This activity can be either predetermined for each internal or generated stochastically, depending on how the "message frequency" statement is presented to the model. This yields completely described message groups for each significant function-location pair at every significant time interval of every function. The conversion to LSI units is the next model action. Then, a delay is computed, dependent on the resources assigned and available at each location, describing both the time to generate the message groups and the time to transfer these data from each location to the next data system operation, as prescribed by the "data system routes" statement.

Data Recording. Normally, only those generated data are recorded which are to become part of the requirement for information in the management system being simulated. Recording is defined as arranging data in a form and format required for communication to the processing and storage function of the data system by whatever means prescribed by the data system designer.

The number of recording devices, their locations with respect to the next processing step, and the types of recording devices are variables of data system design. Data handling rates for each device, and the error function assumed for each method of recording are inputs to this model segment*. Certain ground rules, i.e., policies, will be required to permit allocation of resources to workload when queues form. Previously generated data routed to the recording function are assumed to have arrived in a single batch to recording stations. Each LSI,the requirement for recording is matched against the availability of the recording device-type that is specified for each message group within

*In a very sophisticated model it would be necessary to assign a failure function for each type of recording device. This error generating function can indicate errors that are either discovered or not during recording process.

the "data system routes for messages" statement.

By computing an appropriate delay, we simulate the number of intervals required to record messages. Using specified policies, allocation to each available device is made. Devices are committed at the beginning of each least significant interval, so that backlogs are either present or not at each device at the next interval. Again, as in the Data Generation function, the number of intervals required to present recorded data to the next data system step is computed for each message group. This delay is simulated before these message groups are allowed to arrive at the data system function which is planned to be their next destination.

Communication. The similarity between the Communication and the Data Recording portions of the model is obvious. Messages arriving and equipment available are treated identically. All functions are included as part of the model in order to allow for analyses of alternative data systems designs which postulate various mixes of resources within these functions.

Processing and Storage. Activity at a processing center is usually well defined in any management information system which controls some important commercial or military function. Data systems designers concentrate almost entirely on this aspect of the data system. Equipment for each processing run is usually scheduled; as a minimum, a sequence of runs is stipulated, and an equipment assignment for each run can be determined fairly accurately.

The data system model would require as input a list of all devices used in the processing center, both on-line and peripherally, plus the schedule for each device either predetermined or implied by a required sequence of operations. Certain important processing hardware parameters are required model input, such as numbers, data transfer rates, and data densities of input and output devices, and the amount, type and operating rates of directly addressable storage. Finally, we require a description of the input-output and computing overlap capability of the processor, or any unusual feature that affects processing and storage.

Programming parameters, hopefully, can be considered also. These can be stated as estimates of data and instruction storage requirements for programs, and are related to the decision level of the processing required.

The problem as presented to this section of the model is as follows: data are arriving from the generation, recording, and/or communication functions of the data system for storage and processing. These data must be:

1. Stored for later use, e.g., three runs hence, the next day, or weekly cycle.
2. Stored for recall during the next run, i.e., with no setup time required.
3. Immediately processed.
4. Combinations of 1, 2, and 3.

This much about the data and the processing cycle are usually known after preliminary system design: the equipment needed to process the data, the desired product, the required steps of processing, the capacity limitations to storage, and the sets of data needed to effect the end result of processing the data.

Simulation of the processing and storage activity of the data system will include the assignment of message groups to pre-processing periphery equipment, to the main processing steps, and to the reporting function in accordance with the scheduling rules that are system design facts input to the model. The amount of time each machine run requires each piece of equipment in the processing list is computed. This is determined from the size, number, and types of message groups arriving for processing, the equipments' rates of processing, and the error function associated with each piece of hardware in the system.

Equipment bottlenecks in the processing and storage function will be recorded for post-modeling system analysis. Length of slack periods and useful operation will be measured for each equipment. The cost for each equipment can be computed for use in analyzing the effectiveness of the data processing and storage function of the management information system.

Reporting; Report Presentation. For those reports required by system managers which are not direct products of the processing step, the report form and format produced and the operation of equipment proposed can be simulated in much the same manner as in the data recording function. In this way the contribution of specialized data reporting equipment as part of the overall data system equipment mix can be measured and costed.

V. Use of the Model

For what purpose might a data system analysis model be used? Feasibility testing of a proposed management information system design under anticipated operational and environmental conditions is, for the present, the most likely candidate. The fact that total management system optimization may not be possible ought not limit the model's application. Used early enough in the design stage of a management information system, a feasibility test made possible by use of a simulation model can indicate the way to avoid serious data processing system imbalances before the final commitment for hardware has been made. On the other hand, if the die is cast and a management system is already implemented, simulating data system growth factors can anticipate future data system requirements. Sounder decisions on the type of additional resources to be required, and when these must be on hand can be made on confident predictions of how and where the system will be deficient. We know too little now about complex information system behavior to reject the present objective of a data system analysis model, feasibility testing, limited as it may seem, as an insufficient payoff for our research. On the contrary, we feel that we will achieve the means of better understanding data system interactions through this research.

VI. Measures of Effectiveness

Repeated feasibility testing, with various data processing system configurations assumed as model inputs, provides a means of selecting a particular data processing system design as the best of a set of possible alternatives, provided a measure of effectiveness for data processing systems can be agreed upon. Any means of comparison, even though not guaranteeing optimization, provides a basis for system designers, early in system design, to quantify appraisals of a management system's effectiveness, which at present can be no more than pure judgment. Confidence to restructure a data system design can be provided by such quantification of analytical processes through the simulation procedure outlined in this paper. We could hope for optimal systems designs were we able to agree on what we mean by data system effectiveness.

It ought not be difficult for data system analysts to agree on the measures of effectiveness that are most important in systems design. It ought not, but it is. We can readily agree that the ability of a system to meet the specified program is extremely important; but whether excess capacity versus system cost, or system cost for work accomplished, or additional system cost to perform the next required increment of system mission is the significant measure of effectiveness of a particular data system configuration is certainly not an accepted standard. One might avoid this problem by saying that management makes the ultimate decision and this is a policy matter beyond the realm of data processing expertness. Be that as it may, with the use of the data system analysis model we can assess the feasibility of a system to perform the mission for which it is designed, and concurrently (1) list the unused capacity within each data system function, (2) cost out the operation under standard contractual procedures (e.g., purchases are in integral units of equipment, rental is on complete or partial shift basis), and (3) with the above data assess, preferably with further simulation, the information system's capability to take on additional functions. Since the measures indicated in (1) and (2) above will be outputs of the simulation for all the data processing resources in the system, requirement for additional resources necessary to perform any increased workload can be pinpointed to those functions of the information system which appear most likely to need bolstering.

VII. Conclusion

In conclusion, I submit that the research goals outlined in this paper appear ambitious. Yet, one can hope that the process of study itself will enrich our comprehension of the data system design process.

I have no doubt that in time analytical tools will provide data system designers capability to measure, to quantify, and to predict the behavior of important characteristics of the data system. I have no doubt that one day simulation will routinely be a part of the data system design process. We believe that our research may contribute to the early recognition of the value of data system simulation as one tool to enhance the profession of data system design.

A GENERAL PURPOSE SYSTEMS SIMULATION PROGRAM

Geoffrey Gordon
International Business Machines
Advanced Systems Development Division
White Plains, N. Y.

1. INTRODUCTION

Recent years have seen a very rapid growth in the use of digital computers for simulation work, particularly in the field of system studies. The need for such simulation has been generated by the ever-increasing complexity of systems that are being designed, while the speed and capacity of modern digital computers have provided the means by which to expand simulation efforts. The amount of simulation and the complexity of the studies that are being conducted have presented several difficulties in organizing the use of simulation.

An immediate difficulty that arises at the beginning of a simulation project is to provide an adequate description of the system to be studied. The description can be no more detailed than the current information available to the engineers or analysts studying the system, but, at the same time, it must be sufficiently complete to form the basis of a computer program. Further difficulties are created by the fact that there are often many design variations to be studied, and the information on which the simulation is based usually changes rapidly. It becomes very important to be able to reduce the amount of effort required to produce a working simulation program and to minimize the amount of time spent introducing changes in the simulation to follow systems alterations. Simulation programs for particular studies are, of course, written with as much flexibility as possible, but it is extremely difficult to foresee all eventualities, and, even if it were possible, it would present a formidable task to incorporate all the possible changes within one program.

The process of initiating a simulation study involves two major tasks. First, a model of the system to be studied must be constructed and then a program that embodies the logic and action of the model must be produced. If some formal method of describing models through a well-defined language can be established, then it is possible that the process of producing a simulation can be made more automatic. A single program could then be written to accept any logical statement made within the language and the problem of initiating a simulation is reduced to finding a description of the system to be simulated within the language. Such a program would be general purpose in the sense that it would simulate any system that can be described within the language.

The use of block diagrams as a method of describing systems is well established, and it forms a natural choice on which to base the input language for such a general purpose program. An earlier paper [1], of which the present writer was a part author, described a program, based on the use of sequence diagrams, that was initially designed for studying problems in the design of telephone switching systems. The work carried out then indicated that programs based on block diagram language could be applied more widely. The main purposes of this paper are to describe such a program that has been developed from this early work and applied successfully to the study of a large number of different systems and also to summarize some of the experience gained with the program.

The program that is to be described is written for the IBM 704, 709 and 7090 Data Processing Systems in versions suitable for either 8,000 or 32,000 word memory computers. To make use of the program the system to be simulated must be described in terms of a block diagram drawn in the manner that is detailed in this paper. No knowledge of the computer operation is assumed. The user need only know the rules by which models are to be constructed.

The simulation allows the user to study the logical structure of the system, to follow the flow of traffic through the system and to observe the effects of blocking caused either by the need to time-share parts of the system or by limiting the capacity of parts of the system. Outputs of the program give information on:

- The amount of traffic that flows through the complete system or parts of the system.
- The average time and the distribution of time for traffic to pass through the complete system or between selected points of the system.
- The extent to which elements of the system are loaded.
- The maximum and the average queue lengths occurring at various parts of the system.

Statistical variations can be introduced in the simulation, and arrangements are made to sample the state of the system at various points of time. The effect of assigning levels of priority to units of traffic can be studied. It is also possible to simulate the effects of peak loads by varying the load on the system with time or by varying speeds of operation with load.

The following Section 2 describes the principles upon which the program is based. Section 3 describes the basic block types used by the program. Section 4 describes the operation of the program and its outputs. Some worked examples are given in Section 5 and a final Section 6 discusses some of the experience gained with the program.

2. PRINCIPLES OF THE PROGRAM

2.1 System Block Diagrams

Block diagrams are a widely used method of describing the structure of systems. They consist of a series of Blocks each describing some step in the action of the system. Lines joining the blocks indicate the flow of traffic through the system or describe the sequence of events to be carried out. Alternative courses of action that arise in the system are represented by having more than one line leaving a block and one block may have several lines entering it to represent the fact that this block is a common step in two or more sequences of events. The choice of paths where an alternative is offered, may be a statistical event or it may be a logical choice depending upon the state of the system at the time of choice.

The units of traffic that the system operates upon depend upon the system. They might be messages in a communication system, electrical pulses in a digital circuit, work items in a production line or any number of other factors. For convenience, the units upon which the system operates in this program will be described as Transactions.

Although a block diagram is a commonly used means of describing systems, the manner in which a system is described in normal block diagrams depends upon the system and the person describing the system. For the purpose of this program, certain conventions have been established for constructing block diagrams and a number of block types have been defined, each having properties that correspond to some basic action that can be expected to occur in a system. The program requires that the system to be simulated be described in terms of these block types in a manner that is consistent with the conventions. A total of 25 block types is allowed by the program. Each is distinguished by a name which is descriptive of its action and most of them have been given a distinctive symbol for representation in a system block diagrams.

Every block introduced into the simulation must be given a number called a Block Number to identify it in the program. There is an upper limit on the total number of blocks that may be used in the simulation. In the standard deck assembled for 32,000 word memory the limit is 2047 and in the case of the 8,000 word version the limit is 511.

Information about each block of the diagram describing a system is punched in a single card. The deck of all block cards together with certain control and data cards form a problem deck which constitutes the input to the simulation program.

2.2 Block Times

The program operates by creating transactions and effectively moving them from block to block of the simulation model in a manner similar to the way in which the units of traffic they represent progress in the real system. Each such movement is an event that is due to occur at some point in time.

The program maintains a record of Clock Time and the Block Departure Times at which these events are due to occur and the simulation proceeds by carrying out the events in their correct time sequence. Where transactions are blocked and cannot move at the time they should, the program moves them as soon as the blocking conditions change.

The program does not simulate the system at each successive interval of time, instead, it updates the clock time to the next time at which an event is to occur. The controlling factor in the amount of computing time used by the program is, therefore, the number of events to be simulated and not the length of real system time over which the simulation is being made.

All times in the simulation are given as integral numbers. The unit of time represented by a unit change of clock time is to be determined by the program user.

When a transaction enters a block, the program assigns to it a Block Time that determines the period for which the transaction will stay at that block before attempting to leave. The block time represents the time that is required in the system for the action the block simulates. This action time is not always well-defined. The block time is therefore arranged so that it can vary in a random fashion. For each block there are specified a Mean and a Spread. When a transaction enters a block, a block time is computed by choosing at random, an integral number between the limits of the mean plus and minus the spread. The spread may be 0, making the block time a constant; the mean may also be 0, making the block time always 0.

2.3 Block Successors

With the exception of blocks that remove transactions from the system, all blocks must have at least one successor, referred to as Next Block 1. For most block types there can be an alternative successor, Next Block 2. The number of times that any one block may appear as the successor to other blocks, either as next block 1 or 2 is unrestricted.

Except in the case of the Branch block, which is described in Section 3, no block has more than two exits provided as alternatives. It is always possible, however, to place blocks with zero block times in cascade, forming networks that provide any number of alternatives.

Whenever a choice between alternative paths is to be made at a block, a number, called the Selection Factor, is given at that block to indicate the manner in which the choice is to be made. The selection factor is denoted by the letter S. Where only one exit is specified at a block the value of S is taken to be 0.

When S has a value greater than 0 but less than 1, the probability of a transaction going to next block 1 is 1 - S; the probability of going to next block 2 is S.

When S has the value 1, the transaction will always attempt to go to next block 1. If, however, this path is blocked, the transaction will attempt to go to next block 2. In this way, the program is able to divert the transactions to an alternative course of events when it finds that one section of the system is busy.

2.4 Equipment

The system to be simulated will usually include certain components that operate upon the transactions or are engaged by the transactions during the course of their progress. Allowance is made in the program for such components by introducing the concept of Items of Equipment. Certain block types are concerned with describing the interactions between the transactions and the items of equipment.

Many items of equipment can only handle one transaction at a time and a distinction is made between this type of equipment and the type of equipment that can have multiple occupancy. Equipment of the first type which can be used by only one transaction at a time is usually referred to as time-shared equipment. Equipment of the second kind is referred to as space-shared equipment. Allowance is made in the program for both these types of equipment. Time-shared items of equipment are called Facilities and space-shared items of equipment are called Stores.

The relationships between the items of equipment of the simulation and the components of the system vary from one study to another and, indeed, may vary within one study. For example, in simulating a system that includes a computer as one of its components, one study might regard the computer as a single item of equipment. Another study may require more detail and the major components of the computer—such as the memory, the various registers, the input-output devices, etc.—might need to be defined as separate items of equipment each with a characteristic capacity. In another example there might be a communication link with several channels connecting two points. One study might treat the link as a single store with capacity equal to the number of lines. Another study, in which the performance of the individual channels is of importance, might treat each channel as a separate facility.

The ability of one transaction to block another by controlling a facility can be used to introduce control into a model. Frequently, facilities that have no particular correspondence to a component of the system are introduced into the simulation model to act as logical control elements. If, for example, there are two or more parts of a system which are such that only one part may be in use at a time, entrance to any one part can be made contingent upon seizing a facility that will then block off the other parts.

Similarly, there are many occasions on which a store in the simulation does not correspond to an actual component of the system being simulated. Instead, the store is introduced as a control element to limit the number of transactions that can enter a section of the block diagram.

In particular a store can, if desired, be arranged to have a unit capacity, in which case it restricts access to some part of the diagram to a single transaction. There is an important distinction between control exercised in this way by a unit capacity store and a facility. The program does not keep record of which particular transaction enters a store. It is possible therefore, to fill the store with one transaction and to empty it with another transaction. This provides a form of control different from that exercised by a facility where the same transaction that seized a facility must release it.

2.5 Parameters and Functions

It is possible to assign to any transaction a number referred to as a Parameter, which can be used for several purposes. A major use of parameters is in connection with the functions that are described later. These functions can be used to allow a parameter to control the time that the transaction to which it is assigned spends at a block. At other points of the simulation, the parameter can be used as a weighing factor in controlling equipment and in gathering statistics.

The program allows for the inclusion of a number of Functions each of which is expressed as a table of numbers. These functions can be used for two purposes; to modify block times and to act as a source for parameters. The manner in which functions are expressed is illustrated in Figure 1. If a function is to be used to modify a block time, the fact is indicated by referring to the number of the function on the card describing the block. The same function may be referred to by any number of blocks. A function does not give block times directly. Instead, the program computes a block time from the mean and spread at the block and scales the result with the function value.

Block times can be modified in four different ways, or modes, by using a different quantity as independent variable to the function.

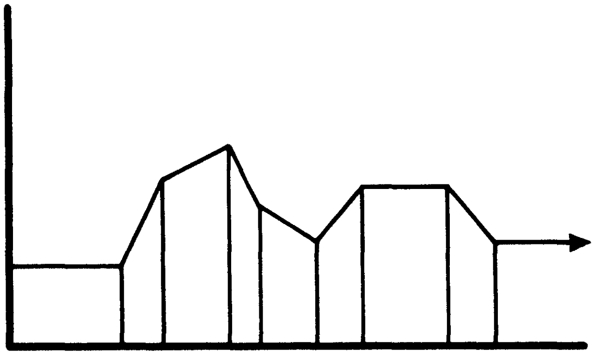


Figure 1. Arrangement of Functions.

(a) By using parameters as input variables a transaction can control its own block time. This mode allows the program to make processing depend upon such factors as message length, number of items in an order, etc., by attaching these meanings to the parameters assigned to transactions.

(b) The function can provide a random distribution of block times that is more specific than the rectangular distribution derived from a mean and spread. The distribution may be the summary of some known data relating to the system or it may be an approximation to some theoretical relationship.

(c) By using clock time as an independent variable, block times can be made a function of clock time. One use of this mode is to be able to vary the load on the system with respect to time. This is done by applying the function to those blocks creating transactions; the block times of these blocks control the rate at which transactions are introduced into the system.

(d) By using the number of transactions currently in a particular store as independent variable, block times can be made to depend upon the number of transactions in a system or some part of the system, and so reflect the effect of system loading on operation times.

The use of a function for assigning parameters to a transaction is called for at the Assign block type that is described in the next section. The value produced from the function becomes the parameter directly. The function being used for assigning parameters can, if desired, be used in any of the four modes that have been described. This includes the parameter mode so that, if desired, the existing parameter on a transaction can be used to reassign the parameter on the same transaction.

2.6 Priorities

Frequently two or more transactions may be directed to take over an item of equipment at the same instant. They may have arrived at that point simultaneously or they may have been waiting for the use of some item of equipment that has just become available.

Under these conditions it is possible to assign priorities to blocks that will determine the order in which the transactions will advance. A total of four levels of priority can be assigned, including the normal level in which no specific mention is made of priority.

Every transaction assumes the priority level of the block which it is currently occupying. When two or more transactions are competing for an item of equipment, the program will give precedence to the transactions coming from the block of highest priority.

3. BLOCK TYPES

3.1 Entering and Removing Transactions

Six of the block types are concerned with entering and removing transactions from the simulation. These are illustrated in Figure 2. Originate and Generate blocks are both used to create new transactions and enter them into the simulation. At the beginning of the program run an initial transaction is created at every Originate and Generate block. New transactions are created and entered independently as the simulation proceeds. As each transaction is created, a block departure time for the transaction is computed. The means at these blocks therefore control the rate at which these blocks create transactions.

In the case of Originate blocks a new transaction is created when clock time reaches the block departure time of the preceding transaction irrespective of whether

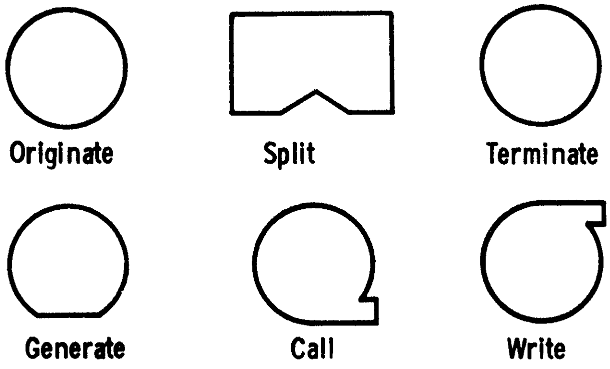


Figure 2. Entering and Removing Transactions.

the preceding transaction leaves the block. A Generate block is identical with an Originate block except that when a transaction attempting to leave a Generate block is blocked, further creation of transactions is held up. An Originate block is used, therefore, where the flow of transactions is not under direct control of the system as might happen, for example, with automobile traffic in a road system. A Generate block is used where the system can stop and start the flow of transactions; for example, a computer system calling for records from a tape.

Split blocks are concerned with creating copies of transactions that are already in the system. When a transaction enters a Split block, an exact duplicate of the transaction is created and sent to next block number 2 while the original leaves via next block 1. All details of the original transaction are copied in the duplicate. If desired, both can go to the same block by making next blocks 1 and 2 the same. There is no restriction on the number of times a transaction can be split so that, by cascading Split blocks, any number of copies can be created.

Call blocks are used to enter transactions from a prepared tape. If desired, this tape may have been produced in a previous simulation run by using the Write block, described below.

Terminate blocks are used to remove transactions from the simulation. In addition Write blocks may remove transactions. The primary purpose of the Write block, however, is to make a record on tape of each transaction entering the block. These records may be used for data analysis or they may be entered into a later simulation at a Call block. Having written a record, there is a choice of whether the transaction is destroyed or passed on to another block.

3.2 Advancing and Marking Transactions

Six of the block types are concerned solely with advancing transactions or marking transactions in a specific manner. These block types are illustrated in Figure 3.

Advance blocks represent the simplest type of block action in the program. Their principal use is to represent some step in the system that requires a period of time but does not involve an item of equipment. Since Advance blocks accept all transactions that want to enter, they cannot cause blocking. They are frequently employed, therefore, as buffers at the output of other block types that use equipment.

A Branch type block is the same as an Advance type block except that the successor may be selected from n blocks, where n is any number between 2 and 127 inclusive. The blocks between which the selection is to be made must be numbered consecutively. The lowest numbered block is referred to as though it were next block 1 and the highest number block is referred to as though it were next block 2.

The selection factor at a Branch block can only be 0 or 1. When the selection factor is 0 the choice of exit from the block is made on a statistical basis. The probability of choosing any exit is assumed to be the same as choosing any other. When the selection factor is 1, the program will attempt to send a transaction that is due to leave by way of the lowest numbered exit. If this is blocked, the program will try the next highest and so on.

There are many situations where it is desirable to have a section of the block diagram made a common segment shared by transactions from different parts of the system. To achieve this, it must be possible for transactions to enter into the common segment from a number of points and, at the end of the common segment, to return to different points according to the original point of entry.

A Tag block has associated with it a number referred to as the tag. When a transaction enters a Tag block, a record of the tag is entered into the transaction. The block is otherwise the same as an Advance block. The

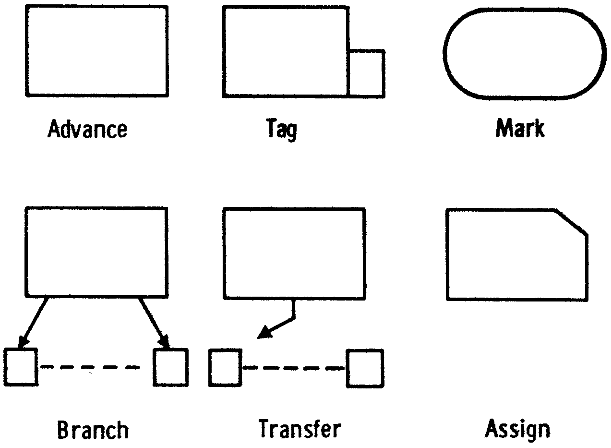


Figure 3. Advancing and Marking Transactions.

tag is used to note the point to which a transaction should be returned before sending the transaction into a common segment of the system.

The tag is entered into the transaction by adding into a field set aside for the tag that is initially set to zero. Consequently, if a transaction is sent through more than one Tag block before reaching a Transfer block, the sum of all the tags will appear in the field. In this way it is possible to carry out the function of multiple tagging.

Transfer blocks are used to send a transaction to a block whose number has been recorded in the transaction itself by Tag blocks. Since the destination of the transaction is predetermined, Transfer blocks do not make reference to a next block 1 or 2; nor do they have a selection factor.

A Mark type block records in the transaction the clock time at which it entered the Mark block. This is used for the purposes of gathering statistics.

An Assign block is used to attach a parameter to a transaction. The program selects the parameter from a function referred to by the block. The function can be employed in any of the four modes described in Section 2.5.

3.3 Engaging Facilities

Six block types, illustrated in Figure 4, are concerned with allowing transactions to control the use of facilities. Since facilities are defined as time-shared items of equipment, only one transaction can have control of a facility at a time. There can be many blocks at which control of the facilities is taken over indicating different points at which the facility is required. The particular facility associated with a block is indicated on the card describing the block and is shown on the block diagram by entering its number in the flag attached to the symbol for the block.

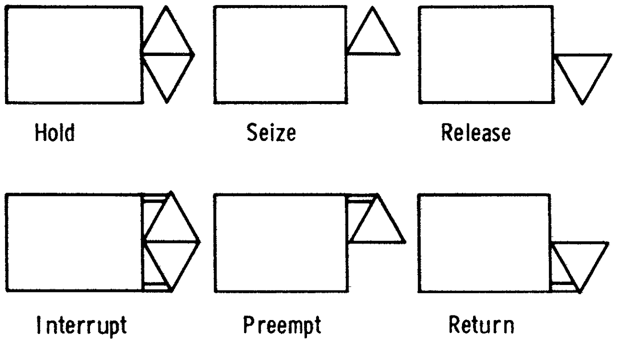


Figure 4. Engaging Facilities.

A Hold block allows a transaction to take over control of a facility as the transaction enters the block. The transaction gives up control of the facility when it leaves the Hold block.

A Sieze block allows a transaction to take over control of a facility as the transaction enters the block. The transaction does not give up control when it leaves the Seize block.

A Release block causes a transaction that has gained control of a facility at a Seize block to give up control as it enters the block. Between the points of seizing and releasing the facility, blocks can be inserted to represent the sequence of events the transaction will go through while it has control. A transaction that has gained control of a facility will deny use of the facility to all other transactions attempting to enter a Hold or Seize block until it relinquishes control. Two other block types are specifically designed to allow a transaction to interrupt a transaction that has gained control of a facility at a Hold or Seize block.

An Interrupt block allows a transaction to take over control of a facility as the transaction enters the block, even if the facility was controlled by a transaction at a Hold block or had been taken over at a Seize block. The interrupting transaction gives up control of the facility when it leaves the Interrupt block.

A Preempt block is the same as an Interrupt block except that control of the facility is not given up at the time the transaction leaves the block.

A Return block causes a transaction that has gained control of a facility at a Preempt block to give up control as it enters the block.

The Interrupt and the Preempt blocks allow a transaction to take over control of a facility whether or not it means interrupting another transaction. When an interruption is necessary, control is always passed back to the transaction that was interrupted. A transaction that has been interrupted remains frozen at whatever block it was situated at the time of interruption and, when it gets back control of the facility, it continues to remain in the block for a time equal to the residue of the time it was due to stay at the time of interruption.

It is not possible to interrupt a transaction that has gained control of a facility by entering an Interrupt or Preempt block even though that transaction may have gained control without actually effecting an interruption. A transaction attempting such an interruption must wait until the first interruption is completed and it will then take over before control is passed back to any transaction that may have been interrupted.

3.4 Occupying Stores

Three block types, illustrated in Figure 5, are concerned with transactions occupying stores. Each such block is associated with a store, which is indicated in the flag attached to the block diagram signal. The capacity of the store is defined on a separate card at the time of starting the simulation. The same store may be referred to by more than one block allowing the store to be employed at different parts of the system.

A Store block allows a transaction to take up storage space when the transaction enters the block. The space is given up when the transaction leaves the block.

An Enter block allows a transaction to take up storage space when the transaction enters the block but the space is not returned when the transaction leaves the block.

Whenever there is insufficient storage capacity, transactions will be stopped from entering the Store or Enter blocks until capacity does become available.

Each block type associated with a store can operate in one of three modes, which control the amount of storage space that is taken up or returned. In the normal mode the amount of storage space involved is a single unit. In a parameter mode the amount of storage space is equal to the size of the parameter (including the possibility of a zero). In a total mode all available storage is taken up or returned. The mode of operation can be chosen separately at each block concerned with a particular store.

The principal purpose of the total mode is to enable stores to be used as counters controlling loops. The blocks can set or empty a counter and the number of loops can be counted by repeatedly entering or leaving the store. Gate blocks, which will be described later can be used to check when a store is completely filled or emptied.

3.5 Gathering Statistics

Two block types, illustrated in Figure 6, are specifically designed for the purpose of gathering certain statistics from the simulation.

Transactions will on occasion become blocked because equipment is busy or the transactions may require that the equipment be in a particular state before they advance. In such cases, queues can build up at blocks not involving the use of equipment. There is no limit on the size of the queues other than an overall limit that applies to the total number of transactions in the entire system.

A Queue block causes the program to compute the average number and the maximum number of transactions that are kept waiting in the Queue block. No statistics are kept on queues occurring at any other block type.

There is one queue associated with each Queue block. It is indicated on the card describing the block and it is numbered in the flag attached to the Queue block symbol. The same queue may be associated with more than one Queue block.

Tabulate blocks are used to derive two types of statistics on transactions moving

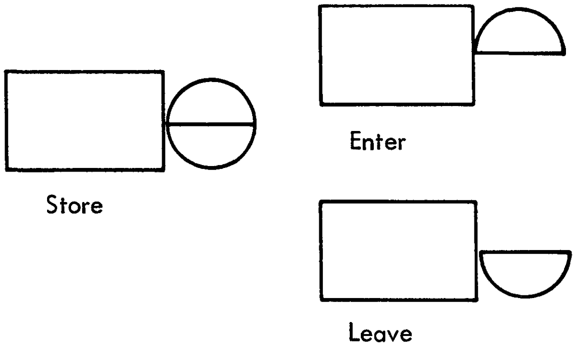


Figure 5. Occupying Stores.

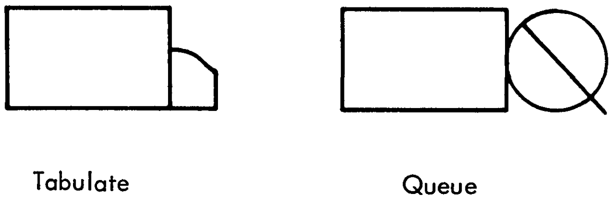


Figure 6. Gathering Statistics.

through the system and compile tables of the statistics. In a Transit Time mode the program compares the time at which a transaction enters the Tabulate block with the mark time entered in the transaction. The difference is entered in the table so that the table maintains a distribution of the time taken by transactions to progress from the Mark to the Tabulate point. In an Interarrival Time mode the time at which a transaction enters the Tabulate block is compared with the time at which the previous transaction entered the Tabulate block. The difference in these times is entered in the table.

Associated with each Tabulate block there is a table which is identified by number and is indicated in the flag attached to the block diagram symbol for a Tabulate block. More than one Tabulate block can refer to the same table, in which case a common total is maintained.

Tables can also be used to derive statistics without being associated with a Tabulate block. At regular intervals of time, which can be selected, tables can be arranged to collect statistics on the rate of arrival of transactions at any block by tabulating how many transactions arrived in each interval or they can give the distribution of occupancy in a store or queue by tabulating the number in the store or queue at the end of each interval.

3.6 Control of Transaction Flow

The ability of a transaction to advance can be made to depend upon whether it is able to engage an item of equipment and, as has been pointed out previously, items of equipment are often introduced for the sole purpose of providing control of the movement of transactions. Some decisions in the control of the system may depend upon whether an item of equipment is available without requiring that the transaction engage the equipment. Other decisions may require that an item of equipment is not available before a transaction can advance or may require some correlation between the movement of transactions. Decisions of these types can be introduced by the block types illustrated in Figure 7.

A Gate block has associated with it an item of equipment that may be either a facility or a store. When a transaction tries to enter a Gate block, the program will decide whether to allow the transaction to advance into the Gate block according to the status of the equipment. Several conditions may be set for the program to test. In the case of a facility, the test is to see whether the facility is busy or free. In the case of a store the test is to see whether the store is full, not full, empty, or not empty. Whichever condition is tested, the Gate block can be arranged to either allow or not allow an advance into the Gate block when the condition being tested is met.

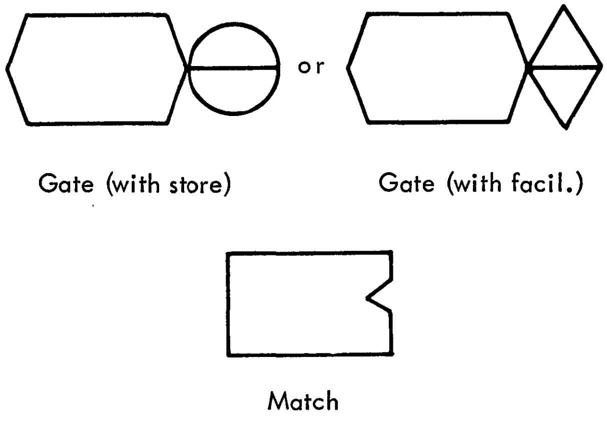
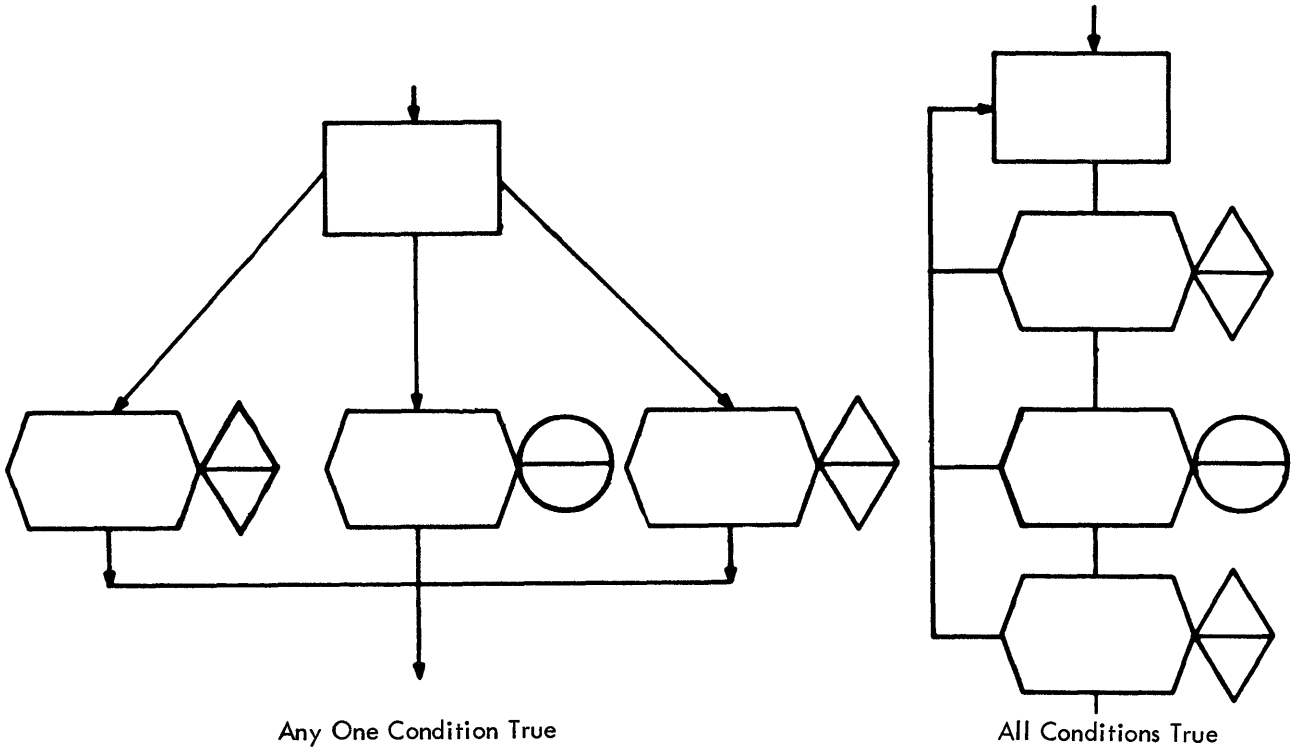


Figure 7. Controlling Transaction Flow.

Ways of testing multiple conditions are illustrated in Figure 8. By associating a Branch block with a group of Gate blocks a transaction is allowed to advance if any one of several conditions is satisfied. An arrangement of stringing Gate blocks in series allows a transaction to advance if, and only if, all conditions are met simultaneously.

Another type of decision that frequently arises occurs when two or more concurrent sequences of events are required to be completed before proceeding to another phase of the system operation. A typical example of such a situation would be a manufacturing process where several parts are made concurrently but all parts must be completed before assembly can proceed. The Split block that has been described allows for the creation of extra transactions at some point of a system, simulating the initiation of such concurrent sequences of events.

A Match block arranges that two transactions resulting from the same original transaction through splitting are synchronized at a later point. One Match block is paired with a second Match block, referred to as the conjugate block, by entering the number



Any One Condition True

All Conditions True

Figure 8. Use of Gate Blocks to Test Multiple Conditions.

of the second block as its next block 2. If desired, a Match block may be its own conjugate.

When a transaction enters a Match block, a search is made to see if another transaction with the same transaction number is waiting at the conjugate Match block. If not, the transaction is indefinitely delayed at the Match block. When a match is found at the conjugate, both transactions continue to advance in a normal manner. There is no restriction on the number of times any given transaction may be matched if the appropriate number of copies have been created by splitting.

4. OPERATION OF THE PROGRAM

4.1 Program Input

To use the program a deck of punched cards must be prepared to describe the problem and give instructions on controlling the simulation run. This problem deck is the input data to the simulation program. The principal part of the deck is made up of one Block card for each block in the simulation. In addition there is one card for each store, one card for each table associated with the simulation and two cards for every function that is used.

A Storage card identifies a store and specifies its capacity. A Table card identifies a table and gives the tabulation intervals associated with the tabulation. There are 16 equally spaced intervals and the table card gives the lower limit and the increment between intervals. The two Function cards identify the function by number and give the mode in which the function is to operate. One card gives eight values of the independent value and the other gives the value of the function at these eight values of the independent variable.

Arrangements can be made to print certain outputs during a run. These outputs, referred to as Snaps, are described later. Snaps are made when the number of terminations reach certain values indicated by a Snap card.

When the program is started, the program will create an initial transaction at every Originate and Generate block. If desired, the simulation can be initialized with transaction at other blocks by using an Initialize card at these blocks. The card identifies a block and indicates the number of initial transactions that are to be entered at that point.

The program begins by loading input cards until it reaches a Start card at which point it commences the simulation. There must be one start card, therefore, for each program run and it must be the last card of a problem deck; the order of loading the deck is otherwise irrelevant. The start card must also carry information about the length of the simulation run. It does this either by specifying the number of terminations to be accumulated or by specifying the amount of time before the program ends.

The start card indicates one of three modes in which to begin simulation. The first mode is for initiating a new simulation run, at the end of which an output is produced. By following the first start card with another start card, set to a restart mode, the simulation can be continued from where it was stopped. The restart can simply continue the simulation run, or it can be arranged that the clock is set back to zero and a fresh start is made in gathering statistics. Before restarting it is possible, if desired, to modify blocks or add blocks by inserting block cards between the start cards.

In addition to restarting a problem, it is possible to run two or more separate problems with only one loading of the program. This is achieved by placing one problem deck behind the other so that at the end of one problem, the program reads the next problem into the computer. When this is done a Job card, which carries a title for the problem, must be placed at the beginning of the second problem deck to clear out the preceding problem.

All cards may be listed as part of the program output and Remark cards may be inserted at any point to annotate the deck.

4.2 Program Outputs

During the running of the program two types of snaps may be written on tape for printing. One gives information about individual transactions as they terminate, the other gives information about all transactions in the system at a particular point. These snaps may also be printed on-line to show the progress of the simulation run.

At the end of the simulation run a series of outputs are written on an output tape. If they are not required, most of these outputs can be suppressed. The clock time at the end of the run is given together with a count of how many times each block has been entered by transactions. For every facility there is information about the fraction of time it was in use and for every store there is information about the average utilization of the store. Figures showing the maximum and the average queue lengths at the Queue blocks are then shown.

A series of tables giving the statistics that were requested either by entering Tabulate blocks or Table cards will then be printed. Depending upon the selection made on the cards, the tables give the distribution of transit times for transactions to cover sections of the system, the distribution of interarrival times at specified points, the rate of arrival at specified points or the distribution of the occupancy of any stores or queues.

5. PROGRAMMING EXAMPLES

5.1 A Disc File Problem

As an example of how the program is used, consider the following problem. A computer is generating messages which, for their processing, require that a record be read from a disc file. Access to the disc file is by way of a half-duplex communication channel (i.e., a channel that is used for input and output). At the file, there are four actuators each servicing one disc. An actuator can only search for one record at a time. The computer forms separate queues of messages requiring access to each of the actuators. It is assumed that the distribution of requests over the four actuators is random with an even distribution.

When a message reaches the head of its queue, the record address is examined and the actuator is positioned over the correct track. This takes an average time of 120 milliseconds with a spread of 80 milliseconds. The discs rotate once every 50 milliseconds so that after positioning the actuator, there will be a delay of from 0 to 50 milliseconds before the record appears under the reading heads. If the communication channel is available at that instant, the record will be read into the computer. If the channel is not available then the actuator must wait for one full revolution of the discs and try again. It keeps trying until the record is read. The channel is used for 5 milliseconds to transmit the record into the computer and, upon arrival, some processing which varies in time between 5 and 15 milliseconds, is carried out. Processing for 10 percent of the messages is completed at this point and they are removed from the computer. The bulk of the messages, however, require that a new record be written back in the disc file before the message is completed.

During all this time in which a record is being located, read into the computer, processed and possibly written back, the actuator is being held. Only when the message finally leaves the system can another message make use of the same actuator. Each actuator, however, is operating independently of the others except for the fact that they must all share the same communication channel.

A block diagram for this problem is shown in Figures 9 and 10. Transactions are created at an Originate block to represent the messages. They enter store 1 which represents the computer memory. Two factors that are to be derived from the study are the lengths of queue involved in waiting for actuators and the distribution of time from the origination of messages to the time an actuator is set up for the message record. Transactions are, therefore, marked and, after being split into four streams at a Branch block, they are entered into queues waiting for access to an arm.

Upon leaving a Queue block a transaction seizes an actuator and the transaction will not release the actuator until its processing is completed. In this example, a unit of clock time will represent 1 millisecond. The mean used at the Seize blocks is 120 and the Spread is 80. The block time will therefore represent the lengths of time taken to position the actuator.

The sequence of events from this point on is the same for all transactions, except that, upon completion of their processing, they will be releasing different numbered actuators. Each of the four streams is, therefore, tagged and then entered into a common stream. First a tabulation is made to give statistics on the distribution of time between creation of a transaction and the time at which the system has positioned an actuator for the corresponding record.

The mean at the Tabulate block is 25 and the spread is also 25. The time spent at this block is therefore between 0 and 50 and this represents the waiting period for the record to come under the reading heads after the actuator has been positioned. If the communication channel, which is represented here by facility number 5, is available at the time the transaction leaves the Tabulate block, the transaction will occupy the channel. If not, the transaction must wait 50 milliseconds, and try again. To simulate this, it is arranged that upon leaving the Tabulate block the transactions enter an Advance block that has a selection factor of 1.0. Next block 1 of this block is a Hold block with facility 5. Next block 2 is another Advance block with a Mean of 50 and a Spread of 0. When the transaction leaves the Tabulate block it will move forward into the Hold block if the channel is available at that instant. Otherwise, it will enter the alternative Advance block, wait for 50 milliseconds and try again. It will keep retrying in this manner until it succeeds in gaining access to the channel.

A transaction occupies the channel for 5 milliseconds and then passes to another Hold block using facility 6, which represents the computer. An Advance block is inserted between these two Hold blocks so that the channel is freed at the end of transmission even if the computer is busy processing at that time.

The Hold block, that gives a computing time of 10 ± 5 milliseconds and which has a selection factor of 0.1, splits the processed transactions into two streams. One stream represents the 10 percent of transactions that complete their processing at this point. The bulk of the transactions, however, leave by way of exit 1 to move into a string of blocks which write a record back into the file. The procedure used for getting access to the channel is the same as that used for reading out the record. Since the communication channel in this example is to be used for input and output, the same facility appears at the read and write points. A transaction, whichever stream is followed, arrives at a Transfer block which directs it to a block for releasing an actuator. Transactions then leave the computer memory and are removed from the system.

Figures 11 and 12 show the input and outputs obtained on one run of the program. In this run messages were generated at the rate of one every 100 milliseconds with a

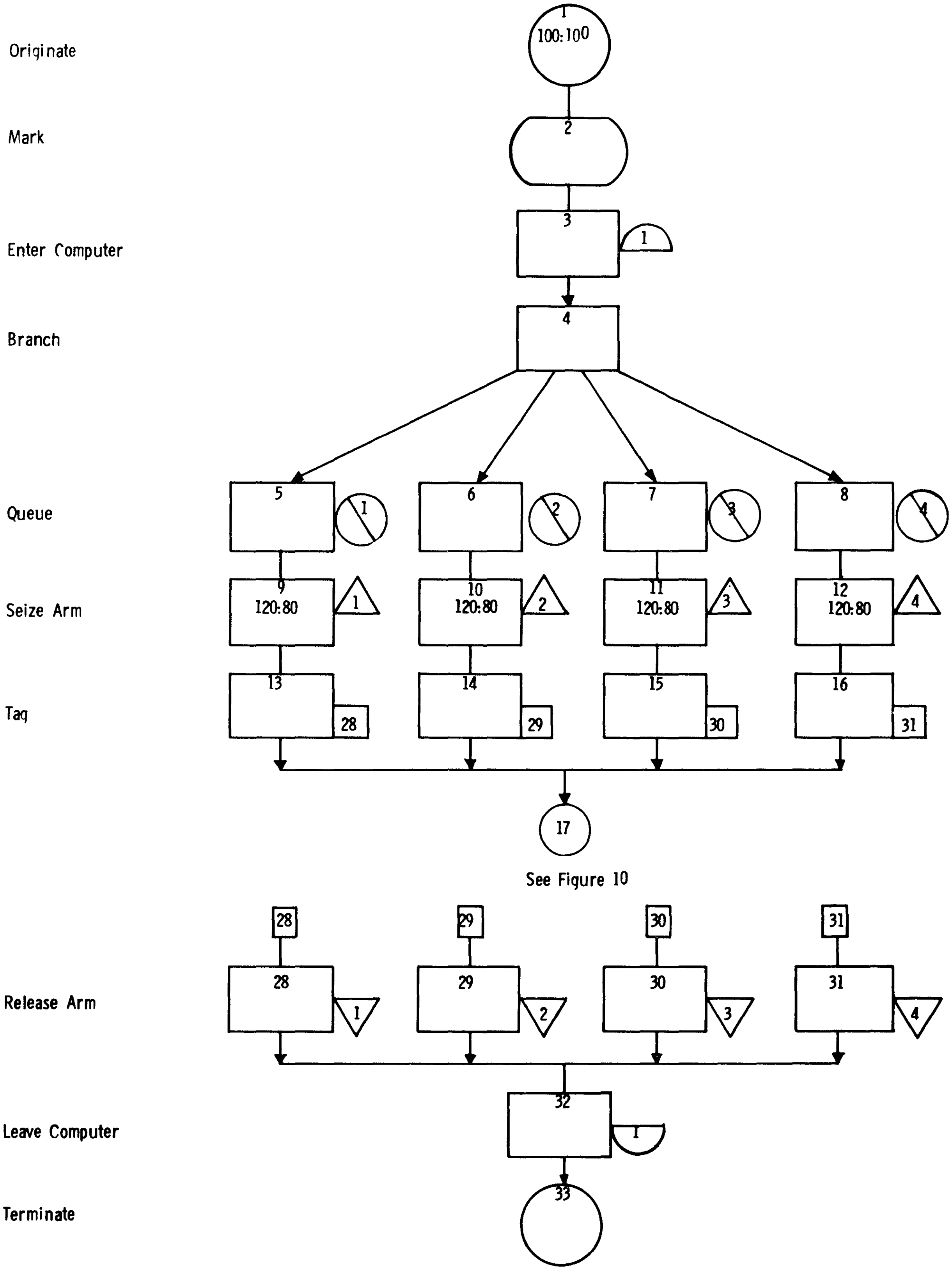


Figure 9. Disc File Problem Part 1

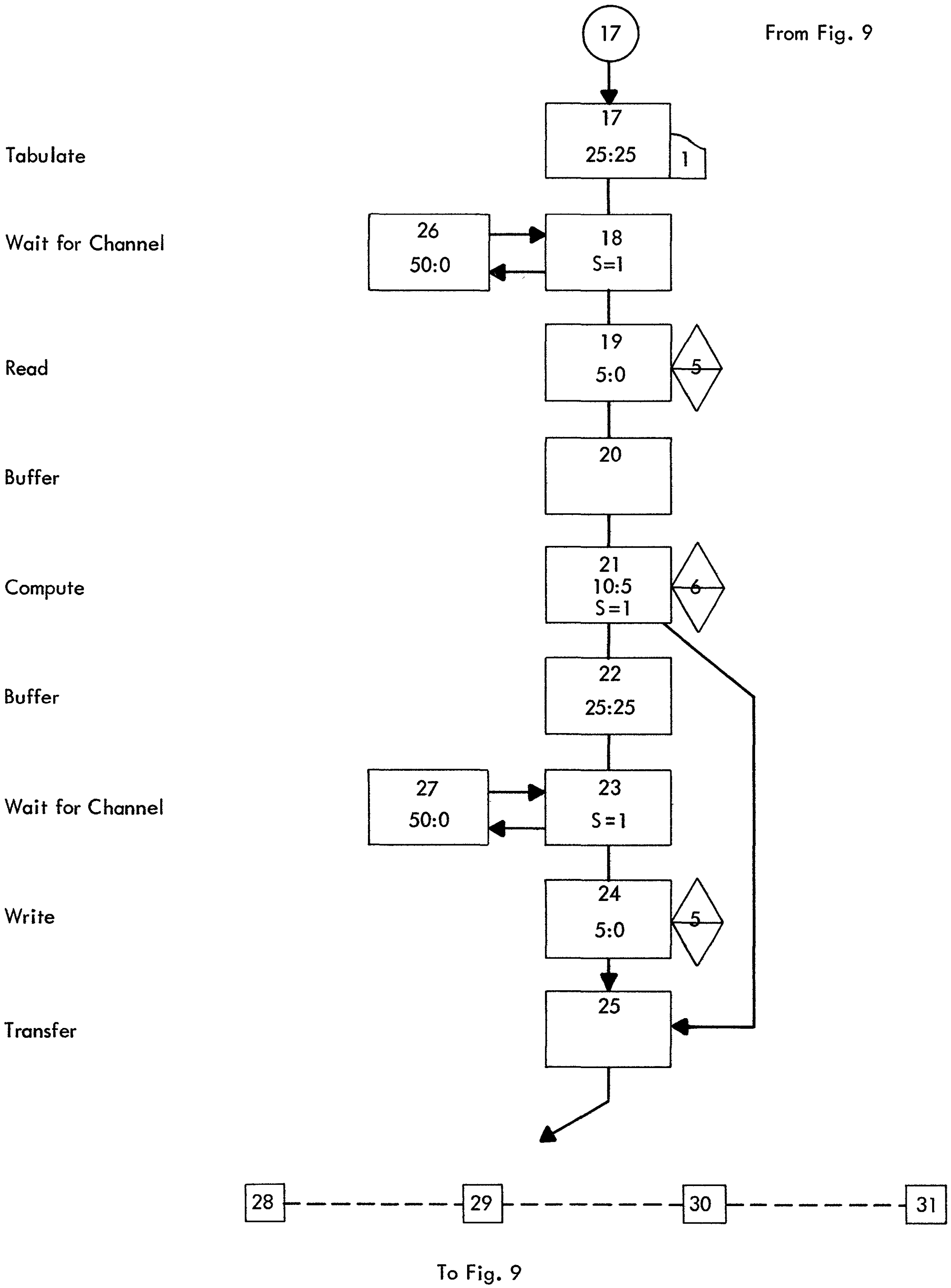
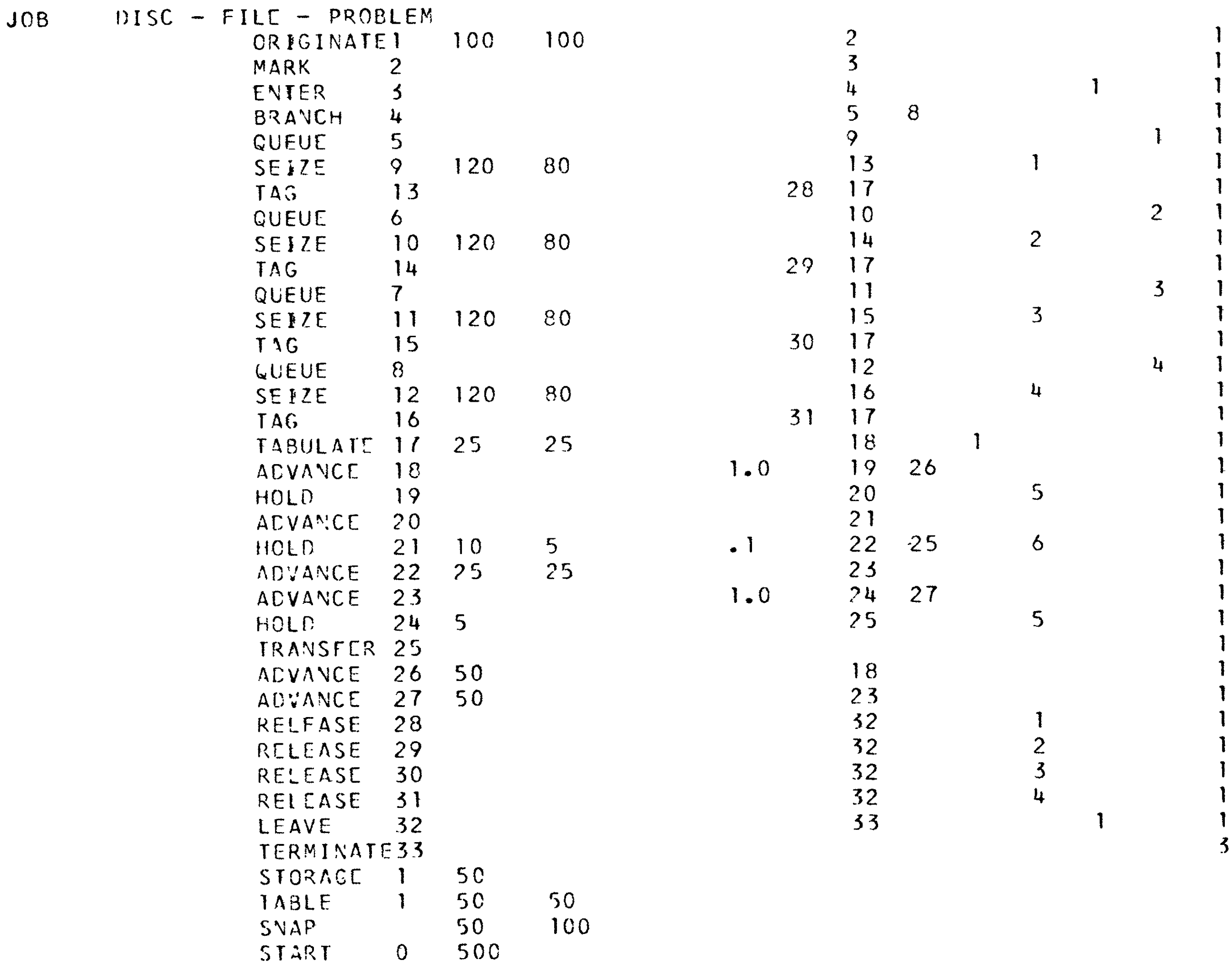


Figure 10. Disc File Problem Part 2.



```
JOB      DISC - FILE - PROBLEM
                     ORIGINATE1     100      100                      2                             1
                     MARK     2                                       3                             1
                     ENTER    3                                       4                   1         1
                     BRANCH   4                                       5    8                        1
                     QUEUE    5                                       9                        1    1
                     SEIZE    9     120      80                       13             1              1
                     TAG      13                                 28   17                            1
                     QUEUE    6                                       10                       2    1
                     SEIZE    10    120      80                       14             2              1
                     TAG      14                                 29   17                            1
                     QUEUE    7                                       11                       3    1
                     SEIZE    11    120      80                       15             3              1
                     TAG      15                                 30   17                            1
                     QUEUE    8                                       12                       4    1
                     SEIZE    12    120      80                       16             4              1
                     TAG      16                                 31   17                            1
                     TABULATE 17    25       25                       18        1                   1
                     ADVANCE  18                            1.0       19   26                       1
                     HOLD     19                                      20             5              1
                     ADVANCE  20                                      21                            1
                     HOLD     21    10       5              .1        22   -25       6              1
                     ADVANCE  22    25       25                       23                            1
                     ADVANCE  23                            1.0       24   27                       1
                     HOLD     24    5                                 25             5              1
                     TRANSFER 25                                                                    1
                     ADVANCE  26    50                                18                            1
                     ADVANCE  27    50                                23                            1
                     RELEASE  28                                      32             1              1
                     RELEASE  29                                      32             2              1
                     RELEASE  30                                      32             3              1
                     RELEASE  31                                      32             4              1
                     LEAVE    32                                      33                  1         1
                     TERMINATE33                                                                    3
                     STORAGE  1     50
                     TABLE    1     50       50
                     SNAP           50       100
                     START    0     500
```

Figure 11. Disc File Problem Input

spread of 100 and the program ran until 500 transactions were processed.

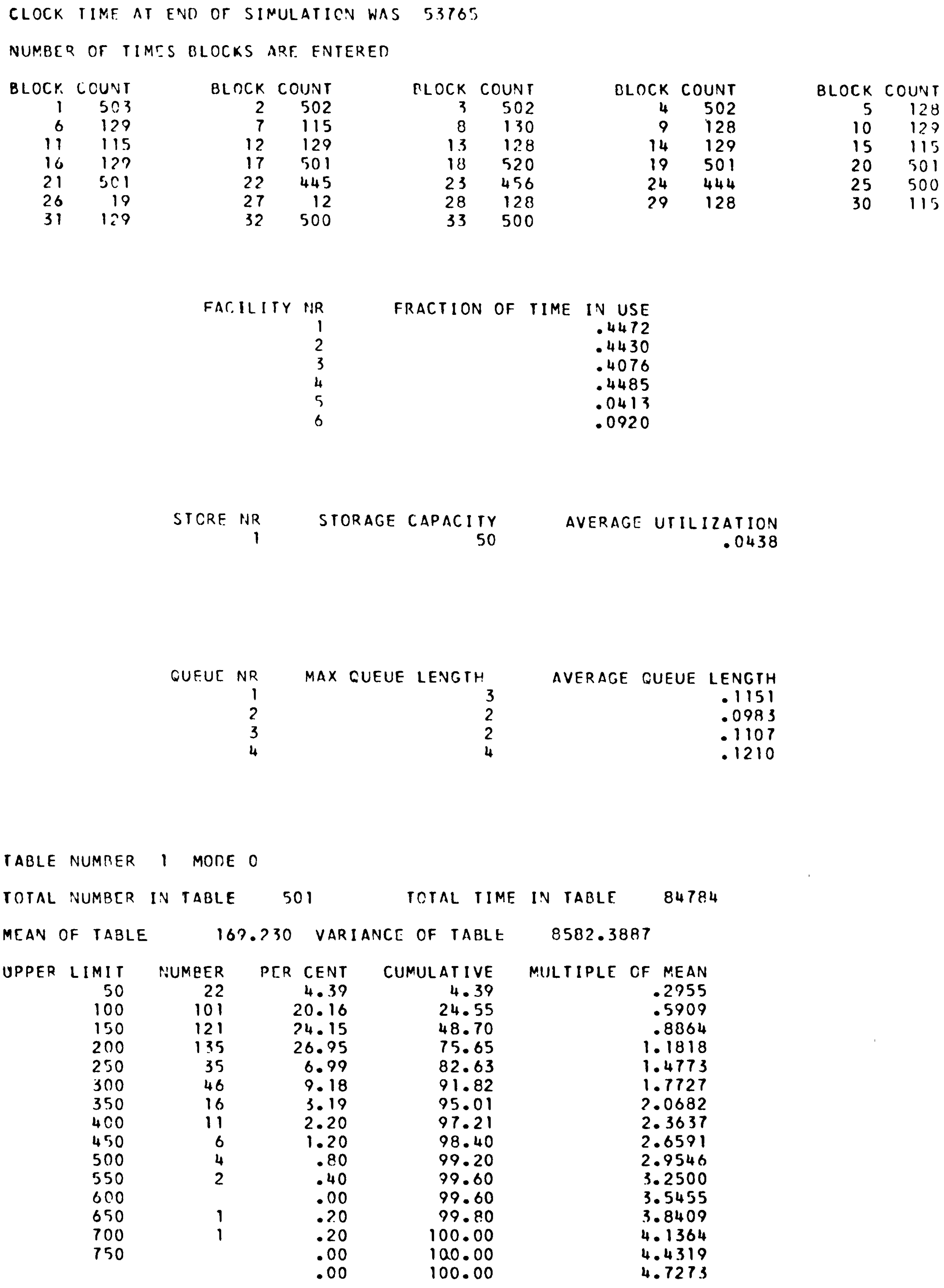
This example is only given for purposes of illustration. The system arrangement and the figures used are not necessarily representative of any particular system or items of equipment. The example, however, does indicate how systems can be simulated and it should be apparent how the problem can be modified to simulate the effect of design changes.

For example, the effect of having separate communication channels for input and output to the file can be arranged by using different facilities at blocks 19 and 24. The effect of having more than one channel available can be arranged by using a store instead of a facility to represent the communication channels. Increasing the number of actuators that service the file can be allowed for by inserting extra streams at the branch block. By adding extra Seize and Release blocks it would be easy to simulate a system in which the actuator is not tied up while one record is processed but is released after reading and is seized again for writing, and so on.

5.2 A Manufacturing Problem

A second example is shown in Figure 13 to illustrate some of the methods by which control over the flow of transactions is exercised. The example represents a simple manufacturing process in which one job at a time is processed. The job requires that three parts be made and these are made concurrently. When all parts are ready, they are assembled and the next job is started.

Referring to Figure 13, the Generate block creates a transaction which immediately



```
CLOCK TIME AT END OF SIMULATION WAS  53765

NUMBER OF TIMES BLOCKS ARE ENTERED

BLOCK COUNT          BLOCK COUNT          BLOCK COUNT          BLOCK COUNT          BLOCK COUNT
    1   503              2   502              3   502              4   502              5   128
    6   129              7   115              8   130              9   128             10   129
   11   115             12   129             13   128             14   129             15   115
   16   129             17   501             18   520             19   501             20   501
   21   501             22   445             23   456             24   444             25   500
   26    19             27    12             28   128             29   128             30   115
   31   129             32   500             33   500


                    FACILITY NR      FRACTION OF TIME IN USE
                              1                        .4472
                              2                        .4430
                              3                        .4076
                              4                        .4485
                              5                        .0413
                              6                        .0920


                 STORE NR        STORAGE CAPACITY         AVERAGE UTILIZATION
                        1                      50                       .0438


                 QUEUE NR       MAX QUEUE LENGTH         AVERAGE QUEUE LENGTH
                        1                      3                        .1151
                        2                      2                        .0983
                        3                      2                        .1107
                        4                      4                        .1210


TABLE NUMBER  1  MODE 0

TOTAL NUMBER IN TABLE     501          TOTAL TIME IN TABLE    84784

MEAN OF TABLE        169.230  VARIANCE OF TABLE     8582.3887

UPPER LIMIT    NUMBER    PER CENT    CUMULATIVE    MULTIPLE OF MEAN
         50        22        4.39          4.39               .2955
        100       101       20.16         24.55               .5909
        150       121       24.15         48.70               .8864
        200       135       26.95         75.65              1.1818
        250        35        6.99         82.63              1.4773
        300        46        9.18         91.82              1.7727
        350        16        3.19         95.01              2.0682
        400        11        2.20         97.21              2.3637
        450         6        1.20         98.40              2.6591
        500         4         .80         99.20              2.9546
        550         2         .40         99.60              3.2500
        600                   .00         99.60              3.5455
        650         1         .20         99.80              3.8409
        700         1         .20        100.00              4.1364
        750                   .00        100.00              4.4319
                              .00        100.00              4.7273
```

Figure 12. Disc File Problem Output

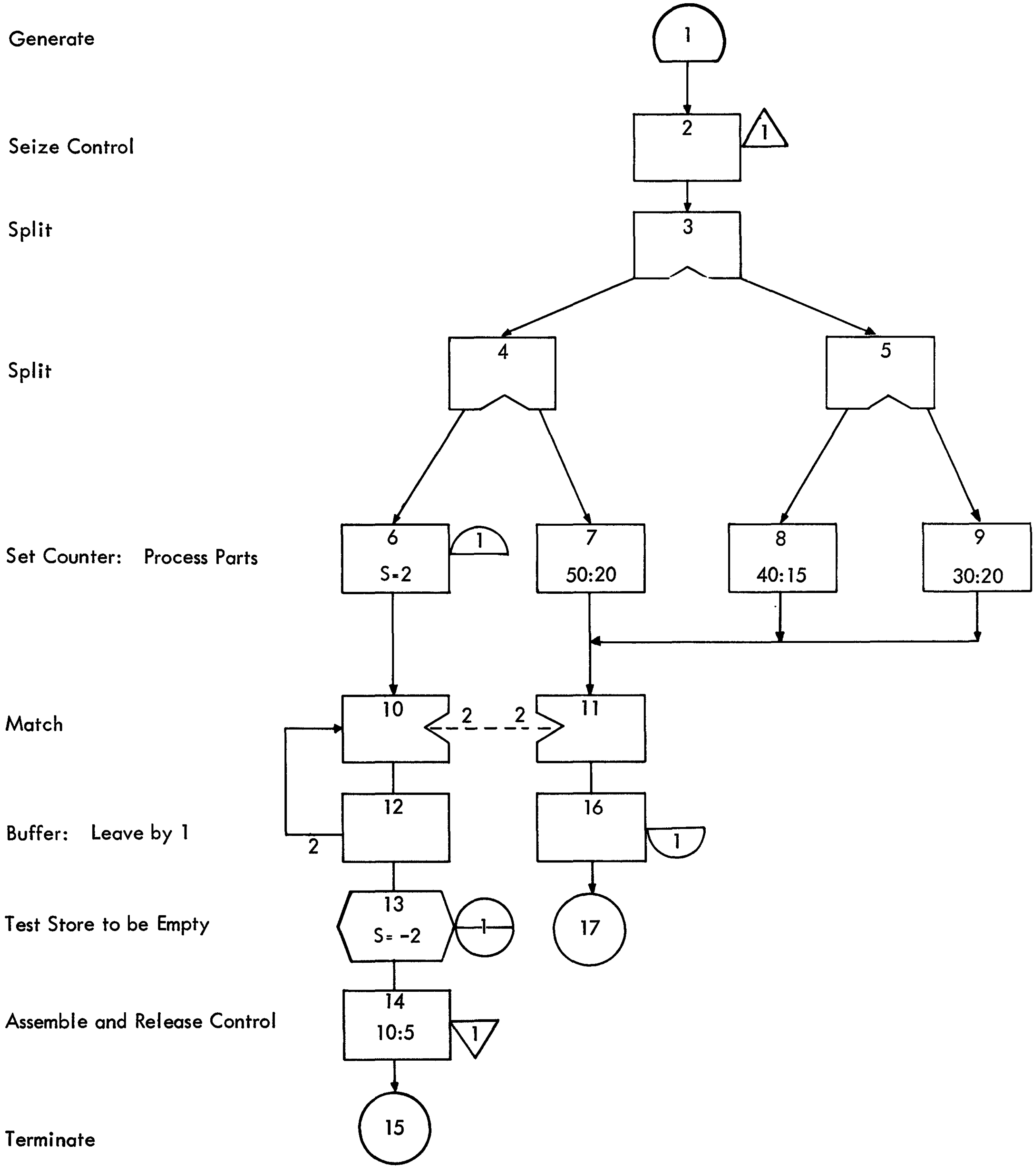


Figure 13. Manufacturing Problem

seizes facility 1. The facility is not released until the job is ready for assembly, thus preventing more than one transaction from entering at a time. A network of Split blocks creates four copies of the transaction, three of these represent the parts to be manufactured and they go to Advance blocks simulating the manufacturing processes. The fourth transaction sets a counter by entering a store of capacity three in a total mode. It then waits at a Match block for the processed parts to arrive.

As each part arrives, a Match is effected. The transaction representing the part removes the count of 1 from the store and is then destroyed. The control transaction uses a Gate to check whether the store is empty. If not, it returns to wait for another Match until finally all parts arrive and the transaction representing the completed parts passes through the Gate to be assembled and release the system for the next job.

If the system to be simulated allowed more than one job to be in the system at a time, facility number 1 would be removed and the checking of the parts would need to be carried out by a string of three Match blocks rather than by a loop around a single Match block.

6. APPLICATIONS AND EXPERIENCE

The program has been extensively used for some time in a variety of applications. It has proved to be particularly useful in problems involved in the analysis and design of data processing, telecommunications and switching systems and the combinations of such systems that arise in the real-time use of computers. It has also been employed for construction models of various manufacturing and business systems while the logical ability of the program has been sufficient to handle problems in the design and operation of control circuits and to programming problems.

The program has been very successful in meeting its main objective of providing a means by which systems analysts and engineers can get simulations going quickly and easily. With very little formal training they are able to understand the basic actions of the problem and can interpret the actions of the system they are studying in terms of the program language.

Inevitably, through being general purpose, the program involves compromises and it seldom meets exactly the requirements of any one user. The particular block types that have been selected represent the accumulated experience of many users. They do not cover all possible actions but an extremely wide range of system concepts have been simulated by combinations of blocks. Experience has shown that the rather limited language provided by the program has not been a significant restriction on the users; instead it has tended to clarify the description of system concepts and has improved the communication of ideas among the people engaged in a system study.

A feature of the program that has been particularly appreciated is the ease with which changes in the simulation model can be made. This has ensured that help from simulation has been available at all stages of a system design. In particular it becomes a simple matter to expand the simulation detail as more information becomes available or, conversely, to reduce sections of the model that have already been examined in detail to a few simple blocks or functions.

The ability to run a succession of jobs with one loading of the program makes it easy to maintain simulation service for a group of people. The running time of the program depends greatly upon the model and the amount of congestion in the system. The principal factor controlling computing time is the time required to transfer a transaction from one block to another. On an IBM 7090 such transfers take an average of about 0.8 microseconds each.

ACKNOWLEDGEMENTS

I would like to acknowledge the contributions of many people to the work described in this paper; in particular, the contributions of Dr. J. P. Runyon of Bell Telephone Laboratories, Professor D. L. Dietmeyer of Wisconsin University and Messrs. R. Barbieri, R. Efron and R. Merikallio among my coworkers at IBM.

REFERENCE

An Interpretive Simulation Program Estimating Occupancy and Delay in Traffic-Handling Systems Which are Incompletely Detailed: D. L. Dietmeyer, G. Gordon, J. P. Runyon, B. A. Tague, AIEE Pacific General Meeting, August 1960, Conference Paper CP60-1090.

USE OF A COMBINED ANALOG-DIGITAL SYSTEM FOR RE-ENTRY VEHICLE FLIGHT SIMULATION

Dr. Allan N. Wilson
General Dynamics/Astronautics
A Division of General Dynamics Corporation
San Diego, California

SUMMARY

Simulation of the re-entry phase of space flight for a vehicle on a satellite or lunar mission is being done at General Dynamics/Astronautics on its combined analog-digital system. Vehicle dynamics are simulated in real time on a large general-purpose analog computer, while an on-board digital guidance computer is simulated by digital program on a high-speed digital computer. An unusual feature of the closed loop operation is the facility for breaking the loop and inserting a pilot with manual over-ride capabilities. Problems of computer system control, check-out procedures, synchronization, sources of error, and results of the simulation are presented.

Introduction

Real-time flight simulation of space vehicles presents special challenges to computing technology, in either man-piloted or automatically controlled space flights. At General Dynamics/Astronautics, the problem is being solved through the joint use of an Electronic Associates general purpose analog computer and an IBM 7090 digital computer. These are linked together by a device called an Addaverter, made by Epsco, Inc. This combination analog-to-digital and digital-to-analog converter is operated under digital computer program control. The complete closed loop operation, on the other hand, is under the control of the analog computer operator.

The present project is part of a continuing program of combined simulation development at General Dynamics/Astronautics [1-4]. The first use of the Addaverter in 1956 entailed a hookup with another digital computer, namely a Remington Rand 1103. At that time a closed loop operation was used to simulate the flight of a radio-guided Atlas missile. The purpose was to determine if the design could proceed with the relatively high frequency, low accuracy dynamics of the missile treated as de-coupled from the relatively low frequency, high accuracy guidance computations. The simulation verified this fact. Shortly thereafter the computing facilities were moved to a new location, and preparations were made to connect the Addaverter with an IBM 704. New terminal equipment had to be developed, and various delays resulted. Meanwhile successful flight tests on the Atlas missile reduced the urgency for such an elaborate simulation, and interest in the closed loop was no longer so intense. A number of interesting open loop applications were developed with the Addaverter conversion equipment, and were conducted first with the 704, and then with its successor, the 7090.

Recently the need for a good combined simulation has become apparent in connection with various space programs, and this paper is intended to report on the efforts that have been made in this direction. It covers the general outline of a re-entry simulation, and some of the problems associated with it.

Description of the Problem

One of the most critical phases of manned space flight is the re-entry into the earth's atmosphere. In the span of a few hazardous moments, the vehicle must be safely slowed to near sonic velocity. It must be held within a narrow return "corridor," avoiding burn-up in too steep an approach, and rebound in too shallow an approach. Although there are many other interesting phases of a manned space journey, our attention here will be confined to the re-entry.

A primary purpose in our combined simulation is to evaluate a proposed on-board digital guidance computer. This is simulated by a digital program on the 7090 digital computer. The dynamic equations of motion are solved in real-time on the analog computer. Values of problem variables are transmitted periodically to the 7090. The guidance program makes up-dated trajectory predictions, and then correction signals are transmitted back to the analog-simulated auto-pilot. The magnitude of the entire calculation is such as to preclude an all-digital solution in real-time. In addition to this limitation, introduction of a man into the loop leads very naturally to the use of analog computer simulation, since cockpit displays and controls generally entail analog signals. On the other hand, a digital guidance computer which is not available as hardware can best be simulated with a digital program. Therefore a combined analog-digital simulation is the only feasible method of attack.

The simplified model for the flight dynamics entails 4 degrees of freedom for the re-entry vehicle. There are two coordinates for the motion of the center of mass in the orbital plane, and two angular coordinates about the center of mass. One of these angles is simply the flight path angle, or angle by which the line of motion dips below the local horizon. The other angle is the roll angle, which in the simplified case takes on quantized values of zero or 180°. Since the simplified model assumes a fixed angle of attack for the aerodynamic shape, the two possible roll values merely indicate whether the lift force on the vehicle is upward or downward. The simulation takes into account the effects of aerodynamic forces for altitudes ranging from 60 or 80 miles down to about 10 miles. The Newtonian forces acting on the vehicle consist therefore of gravity, lift, and drag.

Later phases of the study will extend to the full 6 degrees of freedom. Cross-range forces are considered, and a third coordinate assigned to center of mass motion. A sidewise yaw angle comes into play, and the roll angle is now treated as continuous. This leads to an interesting maneuver known as roll modulation, wherein the vehicle may be made to spiral about a desired ideal trajectory. All possible motions of the vehicle are taken into account in this general case.

Description of the Combined Analog-Digital System

A block diagram of the complete system is shown in Figure 1. Four consoles of Electronic Associates analog computing equipment are used. There are around 100 operational amplifiers, used either as integrators or summers. Then there are several dozen pieces of non-linear equipment, such as multipliers, resolvers, function generators, etc. The analog computer simulates the flight through an atmosphere of varying density, with provision for mid-course corrections by use of attitude control rockets. Automatic control equipment associated with these rockets is included in the simulation. Multi-channel recording equipment is used to present the problem variables as continuous functions of time.

The Addaverter has the capability of transmitting 15 channels of analog-to-digital data, and 10 channels of digital-to-analog data. Each data word carries 18 bits, which is half of a 7090 word. There are 17 numerical bits and a sign bit. The most significant bit corresponds to 50 volts of analog voltage, while the least significant is around 800 microvolts. Actually the last 5 bits at least are continuously submerged in noise, and 12 bits of information are the most to be expected under the best conditions. The assigned coding of voltage permits the operation of

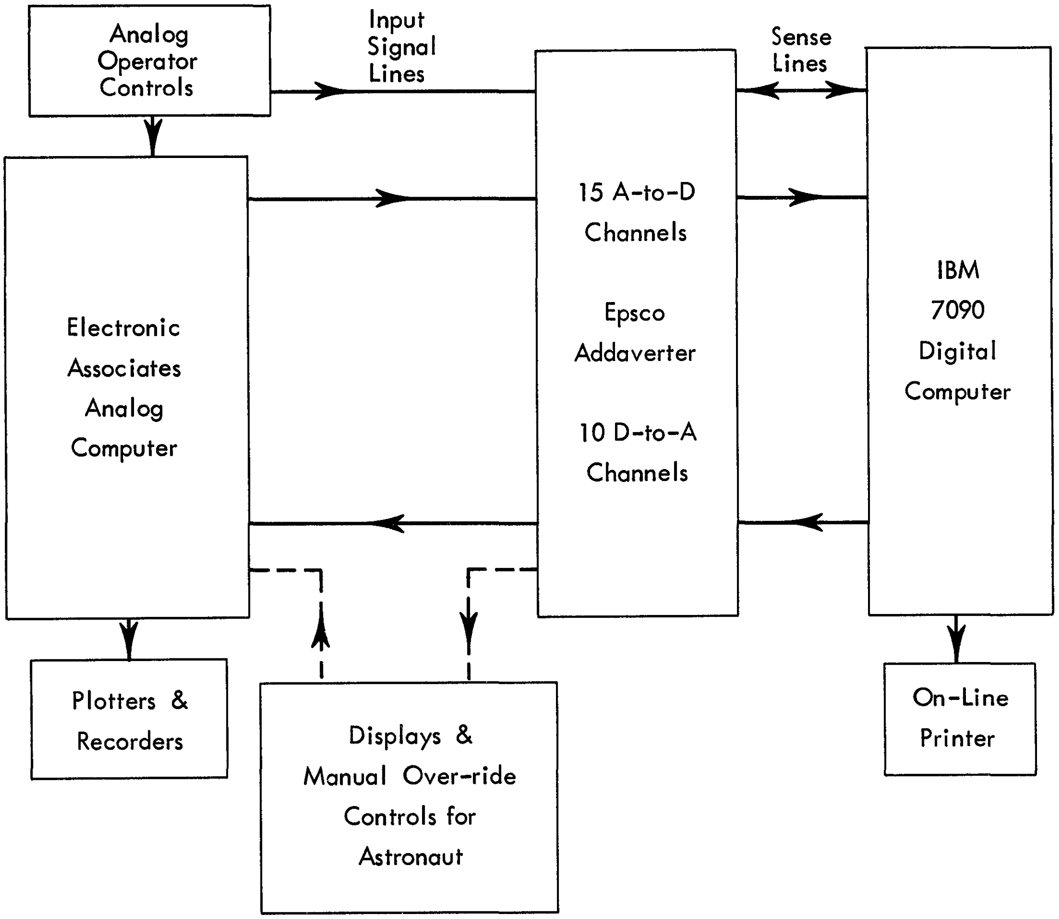


Figure 1. Block Diagram for Combined Simulation System.

the analog computer over the usual +100 to -100 volt range.

There is a basic cycle of four operations in the Addaverter: sample, read, write, and present. (1) Sample causes the analog voltages to be converted to digital values, and stored to await reading into the digital computer. The conversion is effected in less than 140 microseconds, and for usual analog rates of change, may be considered instantaneous. (2) Read causes data to be transmitted through an input-output channel and stored in specified location in the 7090. (3) Write causes data to be transmitted from the 7090 through an input-output channel, and stored in a buffer in the Addaverter, to await presenting to the analog computer. (4) Present causes digital values to be converted to analog voltages, and made immediately available to the analog computer.

This cycle of four commands requires less than 400 microseconds. Usually digital computations are performed in the 7090 between steps (2) and (3). In the event of negligible computation time, it is evident that the system could operate at a 2.5 kilocycle rate. The four commands are sent to the Addaverter by appropriate binary-coded signals on some additional sense lines which are connected from the 7090 to the Addaverter. These 10 sense lines also permit other coded signals for control purposes to the analog computer and other external devices. There are 10 more sense lines running to the 7090, which may be used for control in that direction. Usually the Addaverter is under program control in the 7090, via the sense output lines from the 7090. However, the Addaverter may also be controlled by other external sources, as will be discussed later.

In Figure 2 is given a typical set of instructions for control of the Addaverter in a subroutine in a 7090 program. This may be of interest to anyone who is conversant with 7090 coding and the role of the "Direct Data

```
ATØD    PSLB    =Ø000140000000     Sample Addaverter code.

        AXT     31,1               Delay while

        TIX     *,1,1                  sampling.

        PSLB    =Ø000101000000     Read/Write Addaverter code.

        RADB                       Read.

        RCHB    X

        TCØB    *

        CLA     DATA1

        TSX     FXFLØ,4            Coded voltage to

        . . . . . . . . . . .          floating point.

        CALL    CØMPU              Compute subroutine.

        . . . . . . . . . . .

DTØA    PSLB    =Ø000101000000     Read/Write Addaverter code.

        WADB                       Write.

        RCHB    Y

        TCØB    *

        PSLB    =Ø000120000000     Present Addaverter code.

TEST    ENK                        Digital console control

        XCA                            instead of external

        TZE     ATØD                       oscillator control.

        . . . . . . . . . . .

X       IØCP    DATA1,,1

        IØCP    DATA2,,1

        IØCD    DATA3,,1

Y       IØCD    ANGLE,,1
```

Figure 2. Typical 7090 Program for Addaverter Control.

Device." No attempt will be made here to explain this in detail.

The total role of the 7090 in the combined analog-digital system will be covered more extensively in the next section entitled "control logic." At this point it is desired only to make a few remarks about the guidance calculation program. The guidance prediction scheme requires as input about 10 quantities, such as velocity, altitude, time, flight path angle, etc. It takes these values as initial conditions, and projects ahead in time through the "survival" phase. If continuation on the current trajectory would lead to intolerable "g" stresses on the astronaut, then a correction in roll angle is assigned, and transmitted to the Addaverter for digital-to-analog conversion. For the simplified planar orbit, this is the only signal which is developed. For the more general 3-dimensional solution, several corrective commands are sent, and correspondingly several digital-to-analog channels are utilized.

The evaluation of the guidance scheme calls for examining (a) various numerical integration formulas (b) various time intervals in the use of these formulas, and (c) varying degrees of complexity in the dynamic model for the trajectory prediction. The principal criterion for success is that the time lag between receipt of the analog values and the return of the corrective signals be satisfactorily short. A later phase of the test may also include whether the pilot can respond adequately to the corrective signals.

Control Logic

A flow chart showing control features for the entire operation is given in Figure 3. Once the digital program has been loaded, the operation is under the control of the analog operator. This is indicated schematically by the circles on the right side of the figure. The analog computer actually has three modes, (1) initial condition, (2) hold, and (3) operate. For purposes of our discussion we need consider only hold and operate.

When the analog computer is switched from hold to operate, a pulse generator is triggered on. This generator has a repetition rate corresponding to the desired rate of executing the guidance computation. If computations can be performed faster than needed, there results some dead time after each computation, during which the digital computer is inoperative. Since it is desirable to know the actual time consumed in execution, a clock is read at the beginning and end of each computation.

The controlling command from the pulse generator is a voltage change which energizes the 7090 start relay. Thereupon the clock is read, and a signal sent to the analog console turning off the ready light. Then the compute bit is tested. This bit will have been turned on when the analog computer was switched from hold to operate. Since the test on the compute bit is positive, the digital computer enters the main subroutine. Analog voltages are sampled and read into the 7090, the guidance computations are performed, and the guidance correction values are written back into the Addaverter, and presented to the analog computer. This completes the main subroutine. Then a signal is sent to the analog console, turning on the ready light. Finally the clock is read, and the 7090 stops.

As long as the analog computer is in operate, the pulse generator is still running, and the compute bit is on. Hence at the next pulse from the generator, the 7090 is started again and the process is repeated. When the analog is switched back to hold, then the computations, both analog and digital, are suspended. At this point several courses of action are available to the operator: (1) Obtain a print-out or dump from the 7090. (This may be desired whether the run has been a success or not.) (2) Set up a new case. (3) End the run and get off the 7090. (4) Make some special program revision at the digital console. Each of these alternatives is assigned a particular code. As shown in the figure, the first three are under control of switches on the analog console; the fourth may be set in at the digital console.

A test on any of these alternatives may be initiated by a manually operated command signal at the analog console. This by-passes the pulse generator, and provides a single start signal to the 7090. From there, the sequence of tests and associated subroutines are as shown in the figure. Since the analog computer is at this point in the hold mode, or possibly the initial condition mode, the compute bit is not on. Therefore the test on the compute bit is negative, and the usual computing subroutine is not entered.

The choice of placing the entire operation under the control of the analog operator rather than under complete digital program control is a very practical one. The analog operator, sometimes in the role of the pilot, watches numerous continuous recordings during the run. He is in the best position to render a judgment on what the next step should be. The ready light informs him when the 7090 is no longer in execution, but is standing by for his next command. This, and other features, have been designed to facilitate the operation in his hands, and to assure positive control of the computing system at all times. In addition, he is in telephone contact with the 7090 programmer-operator, in the event of malfunction or need for special digital console procedures. In a

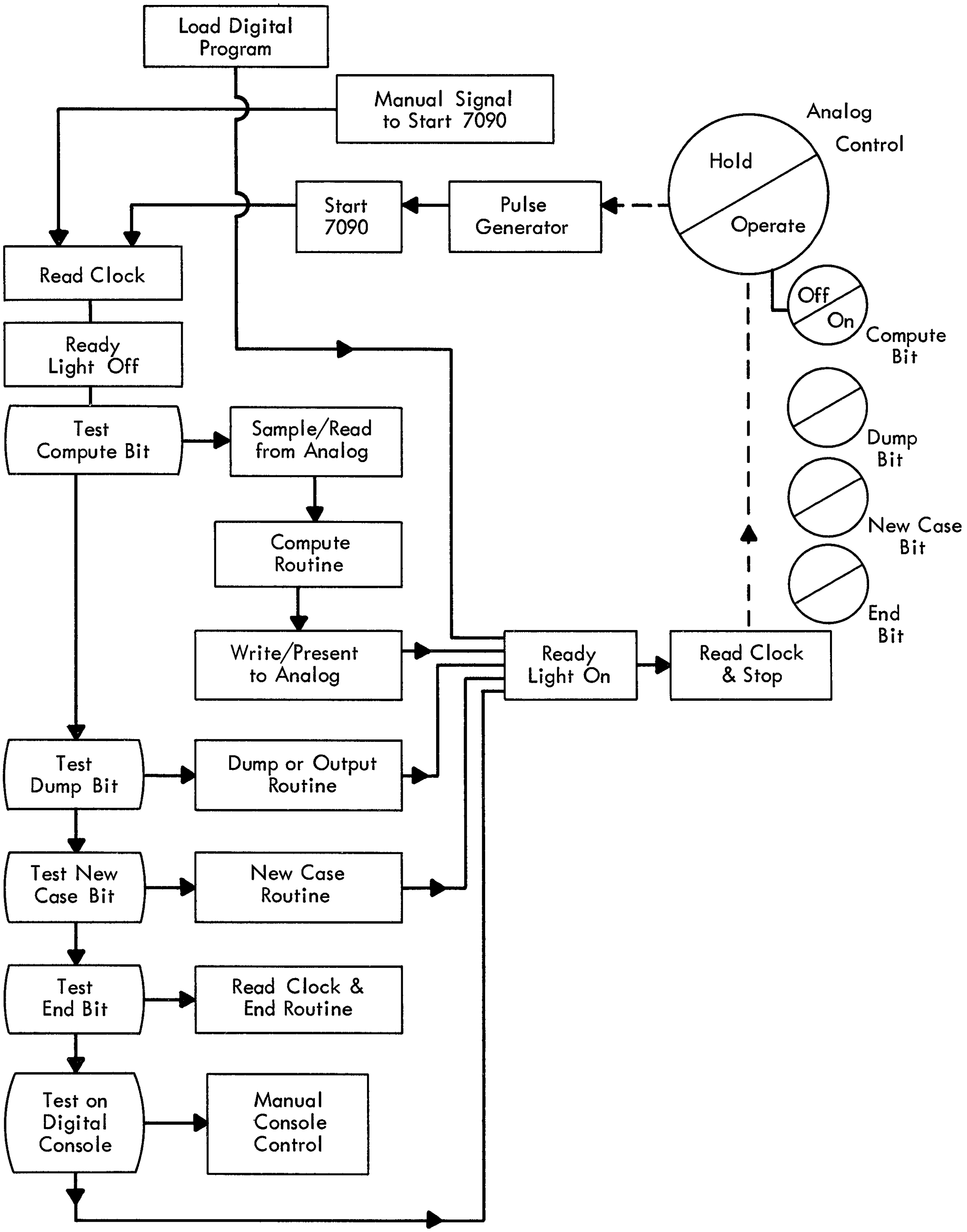


Figure 3. Control Logic for Complete Operation.

well-rehearsed run, the decision-making time is only a small fraction of the entire running time.

Check-Out Procedures

Economic considerations dictate that highest priority be given to efficient use of the 7090 time. Therefore it is necessary to check out the system as completely as possible prior to connection to the 7090. For this purpose, a device called a 7090 "simulator," for Addaverter control, has been built. It has the capability of duplicating the sample, read, write, and present signals which are usually sent from the 7090 under program control. This cycle can be processed at the fastest possible rate, or it can be paced at a slower rate under control of an external oscillator. Sampled data can be read out of the A-to-D path and back in to the D-to-A path, as if in the customary write mode. But of course this is a straight data transfer, and no arithmetic is involved.

The simulator is used in the normal daily check-out of the Addaverter. A symmetrical triangular wave form is sampled and read around through the system. The output is displayed on a recorder, and also the difference between input and output. These two displays permit examination for bit drop-out in the A-to-D conversion, and for extent of the noise level, respectively.

When the combined analog-digital system is to be checked out, the analog computer is given a check with static voltages. The Addaverter, with simulator, is then placed on line, and the analog switched to operate. Analog data is sampled and returned back to the analog, thereby affording verification of continuity and of proper functioning of the Addaverter. At the earliest opportunity after this, assuming the 7090 is "up," the full combined simulation run is started.

Synchronization Considerations

In general it is impossible to accomplish exact synchronization in the starting of an analog computer and a digital computer. This is because of the variance in the drop-out times in the hold relays of the analog computer, which may amount to several milliseconds. In some problems, particularly where the sampling cycle is very fast, synchronization may be critical. In such cases, it may be necessary to build an appropriate delay on one side or the other, in order to optimize the synchronization. Fortunately, in the re-entry simulation, it is not necessary at all to attempt this. This is because there is no clearcut point during the re-entry into the atmosphere at which the first guidance correction must be made. Therefore the analog computation may be commenced at a sufficiently high altitude so that the vehicle is in free fall without aerodynamic forces. As the atmospheric forces come into play, the digital guidance may be switched on at some arbitrary point. It is apparent that synchronization is unimportant for the assumed model.

Sources of Error in the System

The combined simulation system is a form of sampled-data system. Such systems are subject to errors inherent in the basic sampling process, as well as to noise and bias in the conversion system. It is well known that the sample and hold process exerts a destabilizing influence on a computational procedure. In the case of simple linear systems, analysis may be conducted by the use of z-transform theory and by difference equations. Work of this type has been done with an auxiliary sampled-data simulator and the Addaverter at Space Technology Laboratories by Shumate [5]. Investigations of a similar nature have been conducted at our own laboratory using analog computer integrators, the Addaverter, and either of two methods of closing the loop. The first utilized the 7090 control signal simulator mentioned previously, and the second used the 7090 itself. In the latter case, experimentation was done with predictive interpolation formulas to compensate for transport delay. A simple non-linear system has also been studied. The case considered is the pair of second order non-linear differential equations which describe the polar coordinate motion of a satellite in an eccentric orbit. The results of these studies will be published in a subsequent paper.

The present re-entry simulation system is a very special form of a sampled-data system. Since the guidance prediction computing time occupies several seconds, it is far longer than the time consumed by the sample-read-write-present cycle. Since the analog voltages are scaled to provide a signal

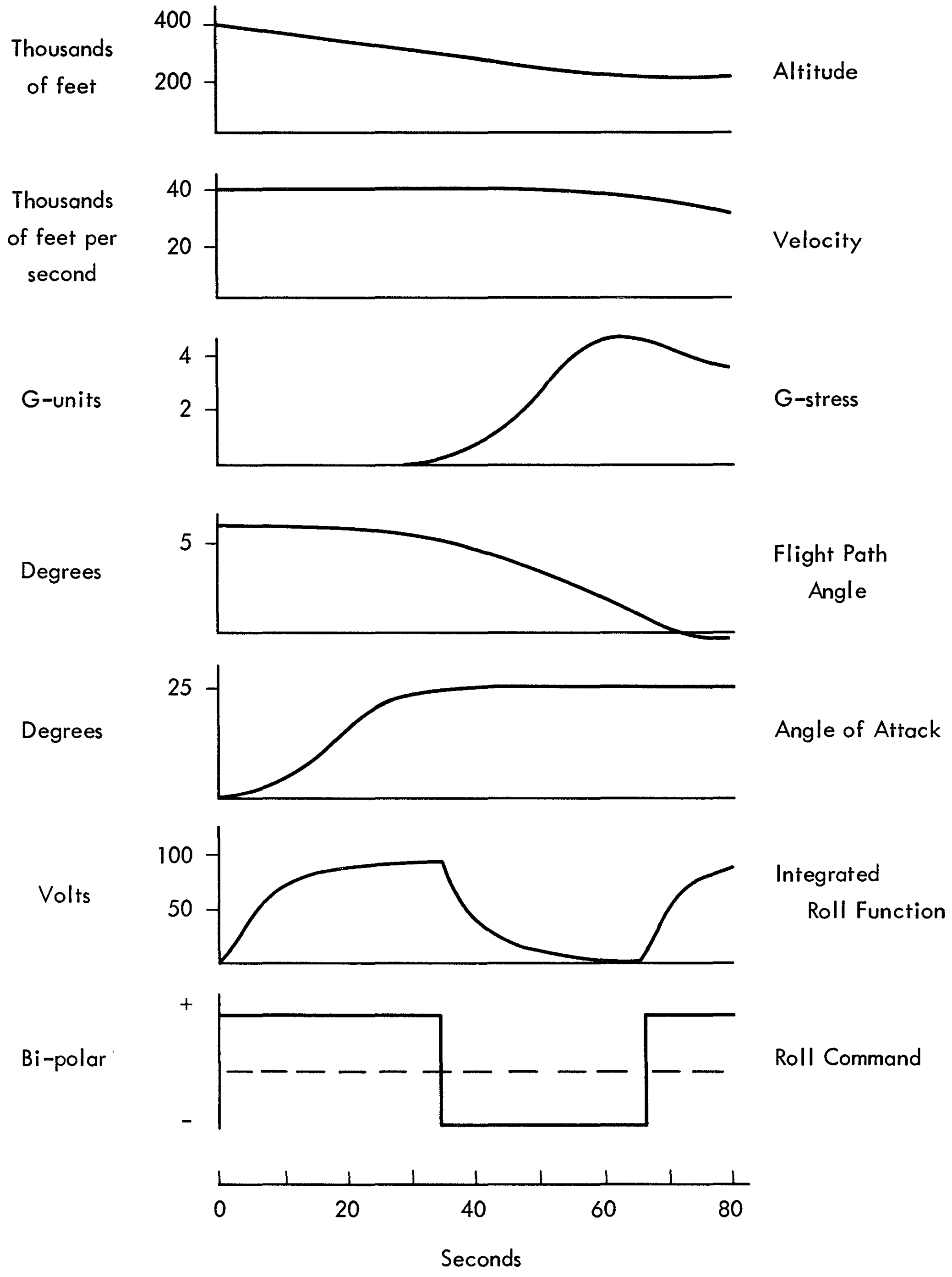


Figure 4. Typical Set of Flight Traces.

to noise ratio of 100 or 1000 to 1, the error contribution of the sampling and data conversion processes is clearly several orders of magnitude smaller than that inherent in the delayed prediction scheme. Accordingly the simulation is useful primarily for checking out the stability of such a guidance method. Provided the process of "hunting" or oscillation about an ideal trajectory does not entail deviations greater than several percent, the feasibility of the technique during the high deceleration survival phase may be considered proven. Final landing maneuvers may of course be expected to be carried out with greater accuracy.

Results

In Figure 4 is shown a typical set of traces for a flight, ranging in altitude from about 80 miles down to 40 miles. This particular run was terminated shortly after the g-stress passed its maximum, for by this point it had been shown that the guidance system was capable of controlling g-stress within allowed limits. A very substantial reduction in velocity is seen during the peak g period. This high deceleration is primarily in the tangential direction, as verified by the record of the flight path angle. Initially this angle dips about 5° below the local horizon, and flattens out to about zero near the end of this run. The bi-polar roll command is seen in the bottom trace.

Since the details of the vehicle motion would be of more interest to the rocket technologist, it is our purpose here to indicate only enough of the results to make them appear plausible. The simulation is regarded as very successful. Later phases of it will incorporate integrated displays and control devices which will test certain aspects of a space pilot's capability to over-ride the automatic equipment, and guide the mission safely on his own. In conclusion, it is felt that combined simulation represents not merely an excellent tool for studies of this type, but actually a very necessary tool.

Acknowledgments

Since my own role was principally that of coordinator on this combined simulation project, I would like to give credit to those who did the actual detailed hard work. Members of this highly cooperative team, with whom it was a great pleasure to work, included (a) David Arris, of the analog computer laboratory, who was the Addaverter specialist (b) Alan Nelson and Michael Hurley, of the dynamics group, who "flew" the vehicle (c) Aaron Cohen, of the guidance analysis group, who worked out the guidance prediction scheme, and (d) Hoyt Bonner and Eugene Etheridge, of the digital programming staff, who wrote the programs and assisted as operators at the 7090 console. Important contributions to the success of this mission were made by all of these General Dynamics/ Astronautics engineers.

REFERENCES

1. R. M. Leger, "Specifications for Analog-Digital Converting Equipment for Simulation Use," AIEE Paper No. 56-860, June 1956.
2. R. M. Leger and J. L. Greenstein, "Simulate Digitally, or by Combined Analog and Digital Computing Facilities," Control Engineering, September 1956.
3. R. D. Horwitz, "Testing Systems by Combined Analog and Digital Simulation," Control Engineering, September 1959.
4. A. N. Wilson, "Recent Experiments in Missile Flight Dynamics Simulation with the Convair Addaverter System," Combined Analog Digital Computer Systems Symposium, December 1960.
5. M. S. Shumate, "Simulation of Sampled-Data Systems Using Analog-to-Digital Converters," Western Joint Computer Conference Proceedings, March 1959.

COMBINED ANALOG-DIGITAL SIMULATION

Arthur J. Burns and Richard E. Kopp
Grumman Aircraft Engineering Corporation
Bethpage, New York

Introduction

The inherent limitations of the analog computer and the digital computer when each is used exclusively give rise to the need for combined analog-digital techniques. The analog computer, which is ideally suited for dynamic calculations, is at a distinct disadvantage where logical decisions are required, or where demands on accuracy and resolution are stringent. In contrast, digital calculations maintain the required accuracy but can be very time consuming. This is a particular burden when many runs are needed for a statistical analysis of a system.

The calculation of missile trajectories, including missile dynamics, is an area which has been particularly troublesome. The need for high resolution when relatively small, accurate miss distances are sought, prohibits an all-analog simulation. On the other hand the cost of an all-digital simulation can be excessive because of the amount of computer time required to calculate the missile dynamics. The need for a new computing technique is therefore clearly indicated.

A typical missile intercept problem has been simulated using all-analog, all-digital, and combined analog-digital techniques. The results of the hybrid method illustrate the advantages of this type of simulation. This paper will discuss the analog-digital computer system which was employed, the hybrid technique involved, and a comparison of this method with the all-digital and all-analog simulation.

Data-Link

In Figure 1 is shown a block diagram of the Data-Link system used to interconnect an IBM 704 digital computer with a Reeves analog computer (REAC). The system consists of five analog-to-digital (A-to-D) converters, five digital-to-analog (D-to-A) converters, and the necessary control and terminal equipment. It is readily expandable to fifteen channels in either direction. Transistorized circuitry is used throughout with the exception of the cathode followers in the terminal equipment specified by the IBM input-output requirements. The A-to-D converters generate a 12 bit (11 bits plus sign) binary word representing an input voltage in the range of ±100 volts at a maximum rate of 5 thousand conversions per second. Analog voltages are sampled simultaneously and then read into the IBM 704 sequentially by means of the A-to-D buffer register. In the digital-to-analog direction, the digital words are written sequentially into D-to-A registers, converted to analog voltages, and applied in parallel to the REAC.

The control equipment consists of the logical circuits required to operate the system. Included in this equipment are the channel select counter and a clock pulse generator. The counter selects the channel through which the IBM 704 can supply or receive information. It automatically steps to consecutively numbered channels with each IBM 704 Copy instruction, or may be "jammed" to a specific channel upon appropriate command from the IBM 704. The latter may only be

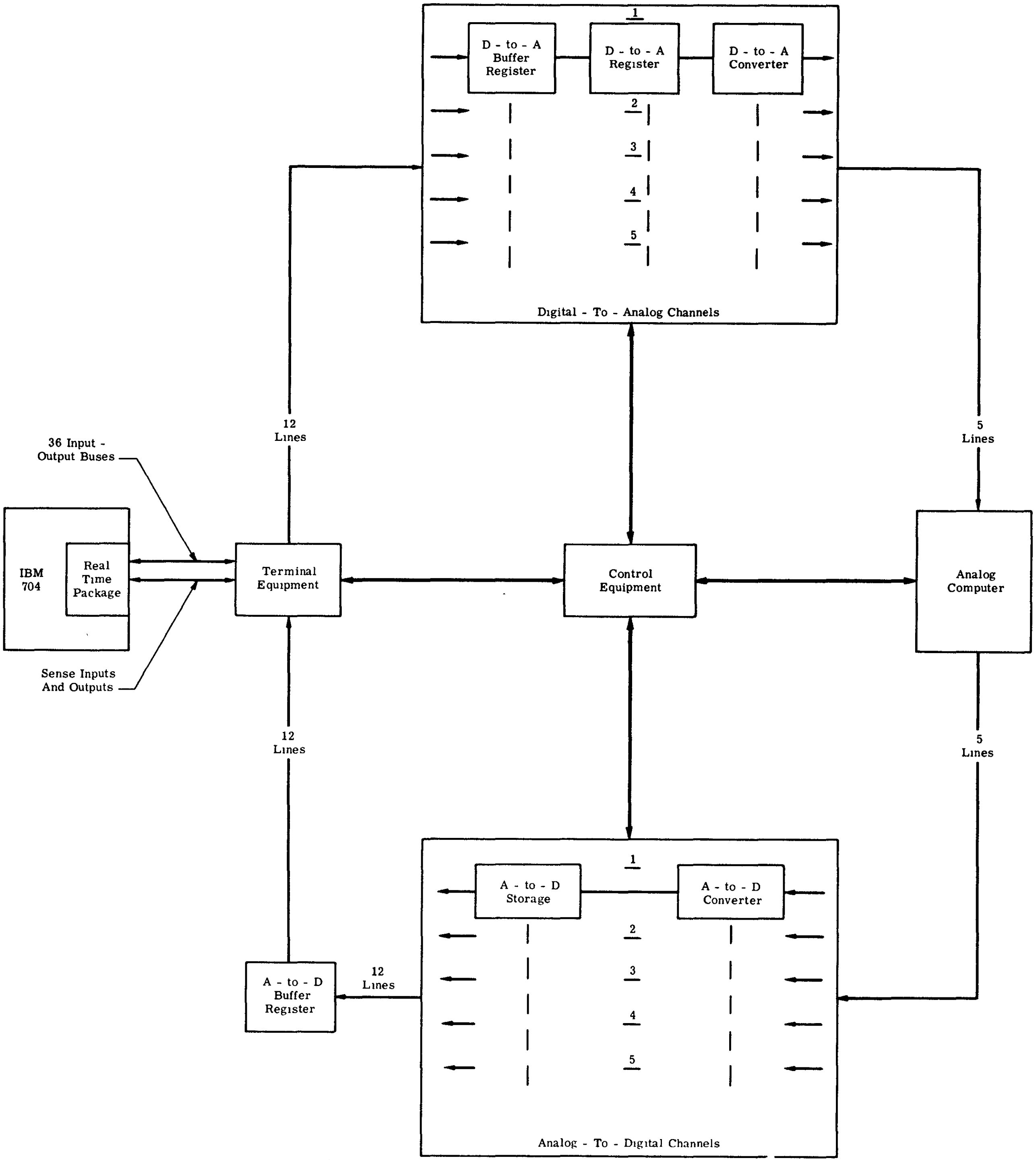


Figure 1. Data-Link System.

accomplished if the IBM 704 has Write Selected the Data-Link. The clock pulse generator, whose rate may be varied from 0.1 cps to 1 KC, is used to initiate a transfer of information between the analog and the digital computer. Analog to digital conversions are triggered by this clock or, in the case of asynchronous operation, by the IBM 704.

In addition to the control equipment terminal equipment is required to make the outputs and inputs of the computer link compatible with the IBM 704. In particular, it translates the IBM 704 logic levels into the computer link logic levels as well as the reverse procedure. Also included in the system is what is known as IBM terminology as a "Real Time Package." This is actually an accessory piece of IBM hardware which provides the necessary connections to the 36 input-output buses and sense inputs and outputs of the IBM 704. Two sense inputs are utilized in the computer link. One indicates whether the REAC is in the "Operate" or "Reset" mode. The second is an overload signal generated when one of the A-to-D converters has an input exceeding ±100 volts. One of the sense outputs is used to complement the state of the REAC; that is send it into "Operate" or "Reset." Another is used to cause analog-to-digital conversion and, if the REAC is in operate, simultaneously cause digital-to-analog conversion. A third causes digital-to-analog conversion exclusively.

Ordinarily 17 of the 36 input-output buses of the IBM 704 are utilized by the link. As shown in Figure 2 these correspond to the 11 information bits (bit positions 25-35), sign (bit position 5), channel address (bit position 1-4), and what is called an "inhibit" bit (bit position 5). Since the channel counter may be "jammed" to a specific channel only when the IBM 704 executes a "writing" sequence with the appropriate channel addressed, the need for this "inhibit" bit arises. In order to select a specific channel for "reading" it is necessary, therefore, to first "write" into the next lower channel so that the counter will be in the correct position at the end of the Copy instruction. Transfer of erroneous information (which would be all zeros) to D-A storage is prevented in this "pseudo-writing" sequence by the inhibit bit. This slightly inefficient method of channel selection for "reading" can be eliminated when an IBM 7090 is used. In this case the additional sense outputs can be used for channel selection prior to either "reading" or "writing."

The same 5 digital-to-analog channels used to transfer information in the "Operate" mode may be used to put initial conditions on the analog integrators while the REAC is in "Reset." Figure 3 shows a simplified block diagram of a digital-to-analog channel and a typical REAC integrator. The initial condition (I.C.) input is only effective while the computer is in "Reset." When it is in

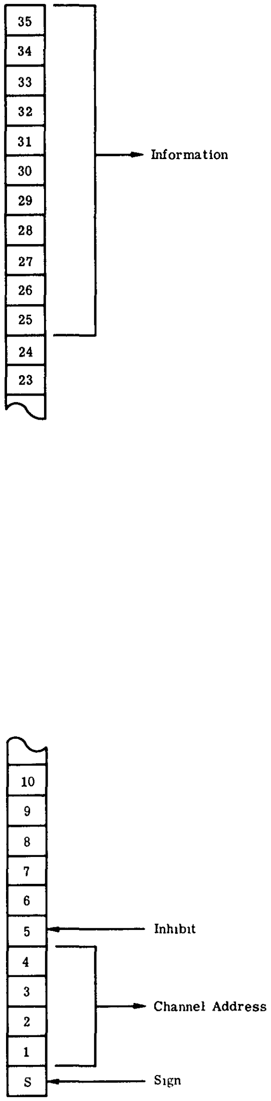
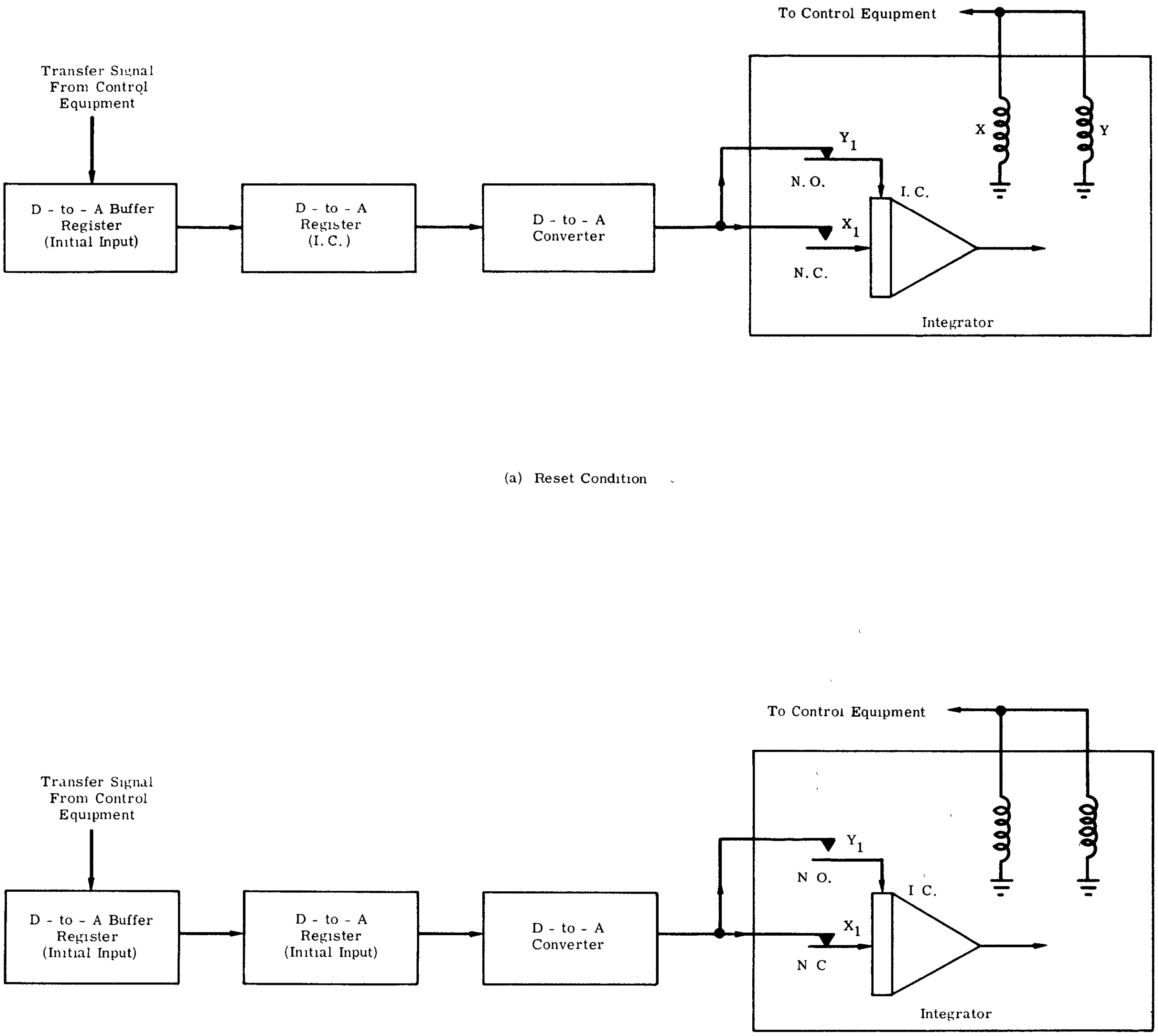


Figure 2. Input-Output Bus Locations Utilized by Data-Link

"Operate," the integrator integrates the voltage at the input terminal. Designating the input to the integrator at time t = 0 the "initial input," it is possible for the initial conditions to be present on the outputs of the digital-to-analog converters, and for the "initial inputs" to be simultaneously present as a digital number in the buffer register. When a signal from the IBM 704 commands the REAC to go into "Operate" the X and Y relays start dropping out. Relay X is slower in falling out than relay Y. There is a finite length of time, therefore, where the input to the I.C. capacitor and input to the amplifier


To Control Equipment
Transfer Signal From Control Equipment
D - to - A Buffer Register (Initial Input)
D - to - A Register (I. C.)
D - to - A Converter
Y1
X
Y
N. O.
I. C.
X1
N. C.
Integrator
To Control Equipment
Transfer Signal From Control Equipment
D - to - A Buffer Register (Initial Input)
D - to - A Register (Initial Input)
D - to - A Converter
Y1
N O.
I C.
X1
N C
Integrator

(a) Reset Condition

(b) Operate Condition

Figure 3. Switching from Reset to Operate Mode.

are both disconnected. It is during this period that the "initial inputs" are automatically transferred from the buffer registers to the digital to analog converters where they are converted to voltages. When relay contact Y_1 has closed the computer is in "Operate," and the "initial inputs" will start being integrated from the initial condition value. Obviously it is not necessary for the output of the D-to-A converters to go to the I.C. and input jacks of the same integrator but may be routed to the inputs of any other amplifier as well.

Missile Intercept Simulation

Figure 4 illustrates the geometry for the terminal phase of a missile intercept problem in two dimensions. The velocity of the missile and target are designated by V_M and V_T respectively, and the range is designated by R. All the angles are defined by the

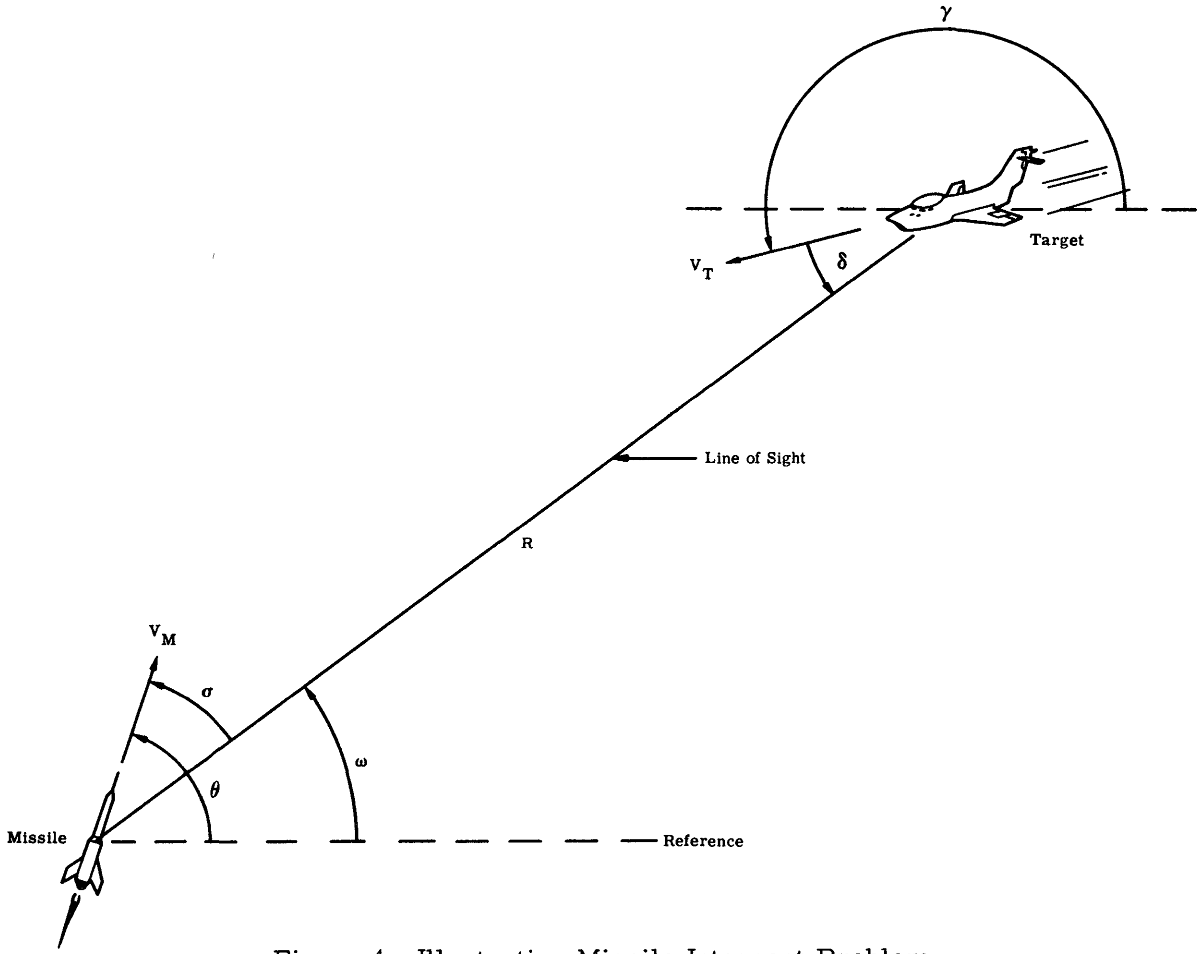


Figure 4. Illustrative Missile Intercept Problem.

geometry. By equating the velocity components normal to the line of sight we obtain,

$$R\dot{\omega} = -V_M \sin\sigma + V_T \sin\delta, \qquad (1)$$

and obtain similarly for the velocities transverse to the line of sight,

$$-\dot{R} = V_M \cos\sigma + V_T \cos\delta. \qquad (2)$$

From the geometry considerations we also have the angular equations,

$$\delta = \omega - \gamma + \pi, \qquad (3)$$

$$\sigma = \theta - \omega. \qquad (4)$$

The angle θ_c not shown in Figure 4 is defined as the control surface angle of the missile. For proportional guidance and a second order lag in the missile control system

$$\Theta_c(s) = \frac{N\dot{\Omega}(s)}{(1+\tau s)^2}, \qquad (5)$$

where the capital Greek letters are the Laplace transforms of variables designated by the respective lower case letters, and N is a parameter of the guidance system. For simplicity, the missile transfer function is assumed to be of second order and of the form,

$$\frac{\dot{\Theta}(s)}{\Theta_c(s)} = \frac{K}{\frac{s^2}{\omega_n^2} + \frac{2\xi s}{\omega_n} + 1} \qquad (6)$$

where K is a gain constant, ξ the relative damping ratio of the missile, and ω_n the natural frequency of the missile.

To include the effects of "g" limiting, the command signal to the missile is limited to $\dot{\omega}_M$. With this constraint Eq. (5) is rewritten in the time domain giving,

$$\tau^2\ddot{\theta}_c + 2\tau\dot{\theta}_c + \theta_c = N\dot{\omega}^*, \qquad (7)$$

where

$$\dot{\omega}^* = \begin{cases} \dot{\omega}_M \text{ when } \dot{\omega} > \omega_M \\ \dot{\omega} \text{ when } -\dot{\omega} < \dot{\omega} < \dot{\omega}_M \\ -\dot{\omega}_M \text{ when } \dot{\omega} < -\dot{\omega}_M. \end{cases} \quad (8)$$

and Eq. (6) is rewritten.

$$\frac{\dddot{\theta}}{\omega_n^2} + \frac{2\xi\ddot{\theta}}{\omega_n} + \dot{\theta} = K\theta_c \quad (9)$$

The equation of target motion is the final equation needed to completely define the problem with the exception of initial conditions. It is assumed in this problem that $\dot{\gamma}$ is constant.

A basic difficulty with an all-analog solution to this intercept problem is the scaling. Range R is chosen to be initially 50,000 feet. Resolution on the order of ±2 feet is desired. To account for the entire variation of R requires solving for R/500 which corresponds to 100 volts for R equal to 50,000 feet. When R is equal to 2 feet, R/500 is 4 millivolts which is well below the probable noise level and practical operation of the analog computer. A second variable which presents a scaling problem is $\dot{\omega}$, the rate of change of the line of sight angle. Initially, when R is large, $\dot{\omega}$ is small. However, when R approaches zero, $\dot{\omega}$ approaches a large value. This does not offer any particular problem in the guidance equation loop since this signal is limited due to the "g" limiting of the missile. On the other hand the true $\dot{\omega}$ is required to compute the kinematics of the problem. Until the missile travels to within a few hundred feet of the target, an all-analog solution is possible. The solution could be obtained to that point in real time. However, the last part of the simulation would have to be run as a separate stage with new scaling to obtain sufficient resolution on R and $\dot{\omega}$.

There is no basic computing difficulty to an all-digital simulation. The disadvantage of this method lies in the relatively longer amount of computing time required for each solution. The total elapsed time is a function of the integration interval used which in turn depends on the accuracy requirements. A 0.025 second time interval, which is sufficient for the dynamics of the problem can be used until the missile is within about 100 feet of the target. A time interval of 0.001 second is necessary from then on due to the ±2 feet resolution requirement. The digital solution, when obtained in this manner consumes about 6-1/2 minutes of IBM 704 time. This is roughly 8 times real time. If a parametric study or statistical analysis of the system is required, the total digital computer time becomes excessive.

The difficulties encountered in the all-analog solution are overcome in the combined analog-digital method by allowing the digital computer, with its eight significant figures to handle the kinematic and guidance equations and, in particular, to solve for R and $\dot{\omega}$. The analog computer is then programmed to compute the higher frequency, lower accuracy portion of the problem represented by the missile dynamics. Thus the equations solved on the REAC are,

$$\ddot{\theta}_c = \frac{N\dot{\omega}^*}{\tau^2} - \frac{2\dot{\theta}_c}{\tau} - \frac{\theta_c}{\tau^2}, \quad (10)$$

and

$$\dddot{\theta} = K\omega_n^2\theta_c - 2\xi\,\omega_n\ddot{\theta} - \omega_n^2\dot{\theta}, \quad (11)$$

while the equations solved in the IBM 704 are,

$$\gamma = \gamma_0 + \dot{\gamma}t, \quad (12)$$

$$\dot{\omega} = -\frac{(V_M \sin\sigma)}{R} + \frac{(V_T \sin\delta)}{R} \quad (13)$$

$$-\dot{R} = V_M \cos\sigma + V_T \cos\delta, \quad (14)$$

$$\delta = \omega - \gamma + \pi \quad (15)$$

$$\sigma = \theta - \omega \quad (16)$$

$$\dot{\omega}^* = \begin{cases} \dot{\omega}_M \text{ when } \dot{\omega} > \dot{\omega}_M \\ \dot{\omega} \text{ when } -\dot{\omega}_M < \dot{\omega} < \dot{\omega}_M \\ -\dot{\omega}_M \text{ when } \dot{\omega} < -\dot{\omega}_M. \end{cases} \quad (17)$$

A block diagram of the combined analog-digital solution is shown in Figure 5. The digital and analog computations are carried out as shown until the range decreases to 350 feet. At that time, the digital computer is used exclusively for the remainder of the run.

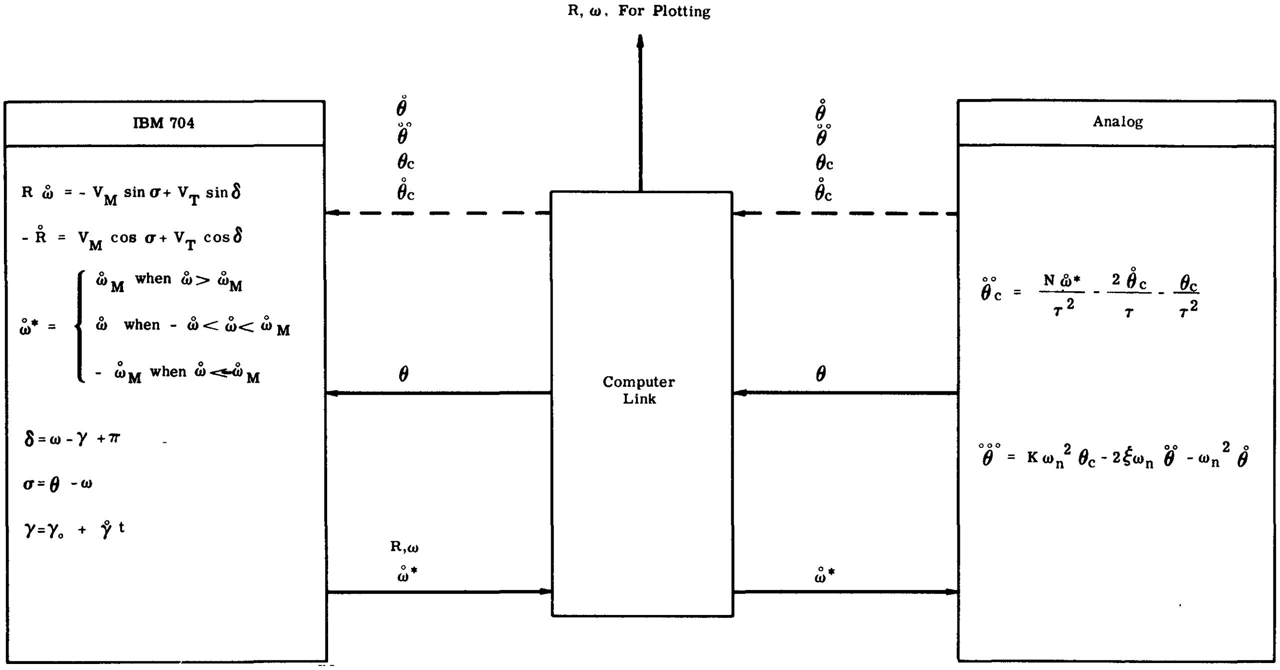

R, ω, For Plotting
IBM 704
$R \mathring{\omega} = -V_M \sin\sigma + V_T \sin\delta$
$-\mathring{R} = V_M \cos\sigma + V_T \cos\delta$
$\mathring{\omega}^* = \begin{cases} \mathring{\omega}_M \text{ when } \mathring{\omega} > \mathring{\omega}_M \\ \mathring{\omega} \text{ when } -\mathring{\omega} < \mathring{\omega} < \mathring{\omega}_M \\ -\mathring{\omega}_M \text{ when } \mathring{\omega} < -\mathring{\omega}_M \end{cases}$
$\delta = \omega - \gamma + \pi$
$\sigma = \theta - \omega$
$\gamma = \gamma_o + \mathring{\gamma} t$
$\mathring{\theta}$
$\overset{\circ\circ}{\theta}$
θ_c
$\mathring{\theta}_c$
θ
R,ω
$\mathring{\omega}^*$
Computer Link
Analog
$\overset{\circ\circ}{\theta}_c = \frac{N\mathring{\omega}^*}{\tau^2} - \frac{2\mathring{\theta}_c}{\tau} - \frac{\theta_c}{\tau^2}$
$\overset{\circ\circ\circ}{\theta} = K\omega_n^2 \theta_c - 2\xi\omega_n \overset{\circ\circ}{\theta} - \omega_n^2 \mathring{\theta}$

Figure 5. Block Diagram for Combined Analog-Digital Solution.

The flow chart for the digital program is shown in Figure 6. The entire solution is under the control of the IBM 704. After writing the initial condition θ_0 and initial input $\dot{\theta}_0$, it transfers the REAC to "Operate." A clock controlled sampling rate was previously chosen based on dynamic considerations. The IBM 704 is put in the Read Select mode and the variable θ, which upon arrival of the first clock pulse has previously been sampled from the REAC and digitized, is then read into storage. This will be a fixed point number but is converted to floating point to facilitate all floating point arithmetic in the IBM 704. If no overloads have occurred, Eqs. (12) through (17) are then calculated. This is followed by a test on $\dot{R}$ to determine whether it is positive. An affirmative answer indicates a miss and the run would then be terminated. If $\dot{R}$ is negative a test is made on R to determine if the missile is within 350 feet of the target. If it is not, R, $\dot{\omega}^*$, and ω are converted to fixed point numbers and the IBM 704 then goes into the Write Select mode. The command signal in the form of the limited $\dot{\omega}^*$ is sent to the REAC along with R, and ω for plotting purposes. The IBM 704 then awaits the next clock pulse before going to the Read Select mode, and the cycle repeats until the missile comes within 350 feet of the target. When this occurs the IBM 704, in addition to reading θ, reads $\dot{\theta}$, $\ddot{\theta}$, θ_c and $\dot{\theta}_c$. The REAC is sent into "Reset," and the IBM 704 solves the remainder of the run using the previously sampled five variables as initial conditions.

Analysis of Combined Analog-Digital Results

The combined analog-digital solution is obtained in approximately real time with a resolution of ±2 feet. Real time for the problem is 50 seconds while the combined simulation requires 60 seconds. The ten second difference arises due to the all-digital calculation of the last few hundred feet.

A sampling interval of 30 milliseconds corresponded to about 200 samples per cycle for the highest frequency present in the simulation (the natural frequency of the missile). The resulting curves for this run as shown in Figure 7a indicate a negligible deviation from the all-digital and all-analog solutions. The digital print-out of the last portion of the combined simulation revealed that the missile came within 21 feet of the target as compared to 17 feet in the all-digital solution.

At sampling rates of 20 per cycle, the deviation from the all-digital run becomes noticeable as shown in Figure 7b. In this

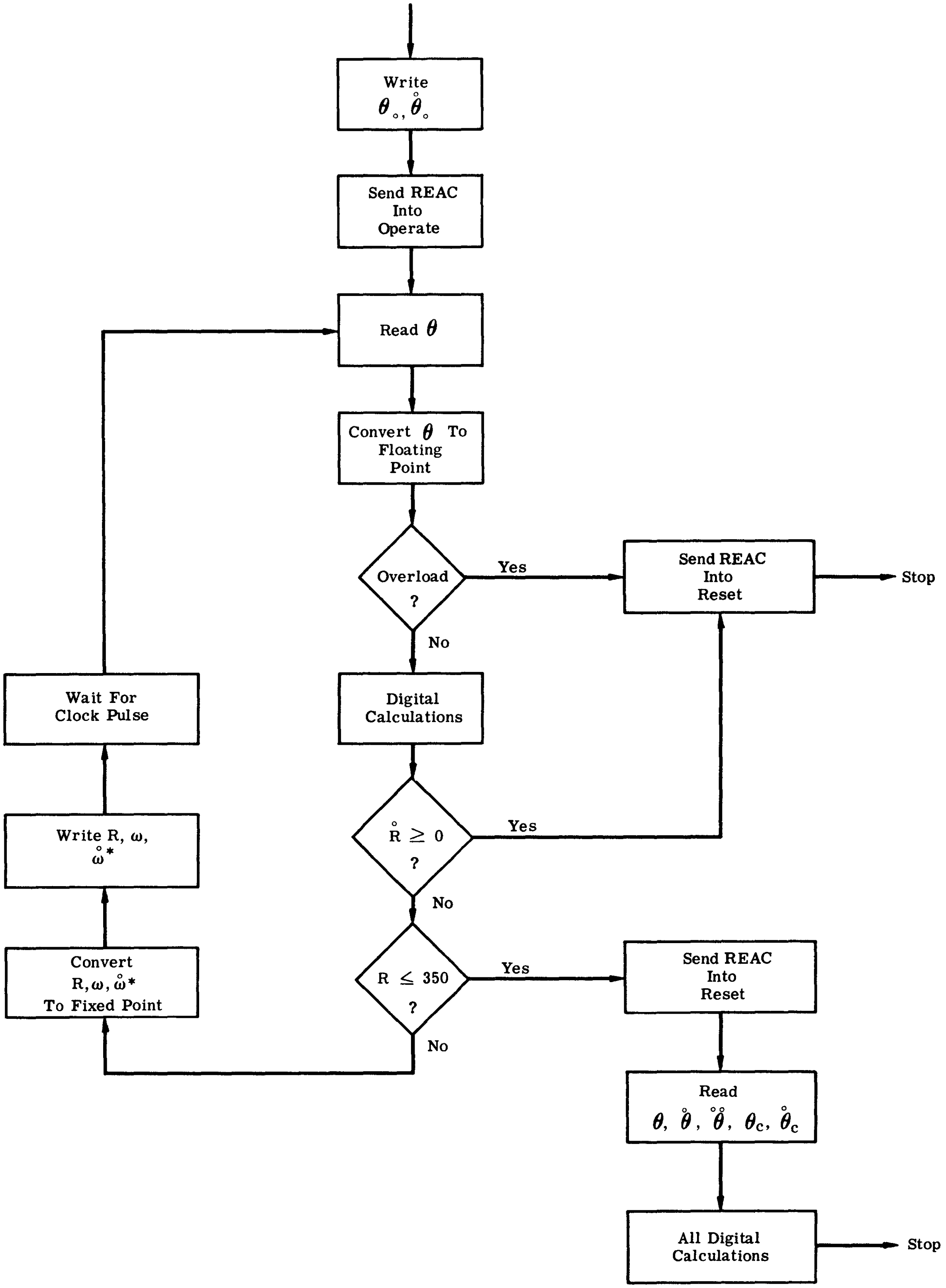
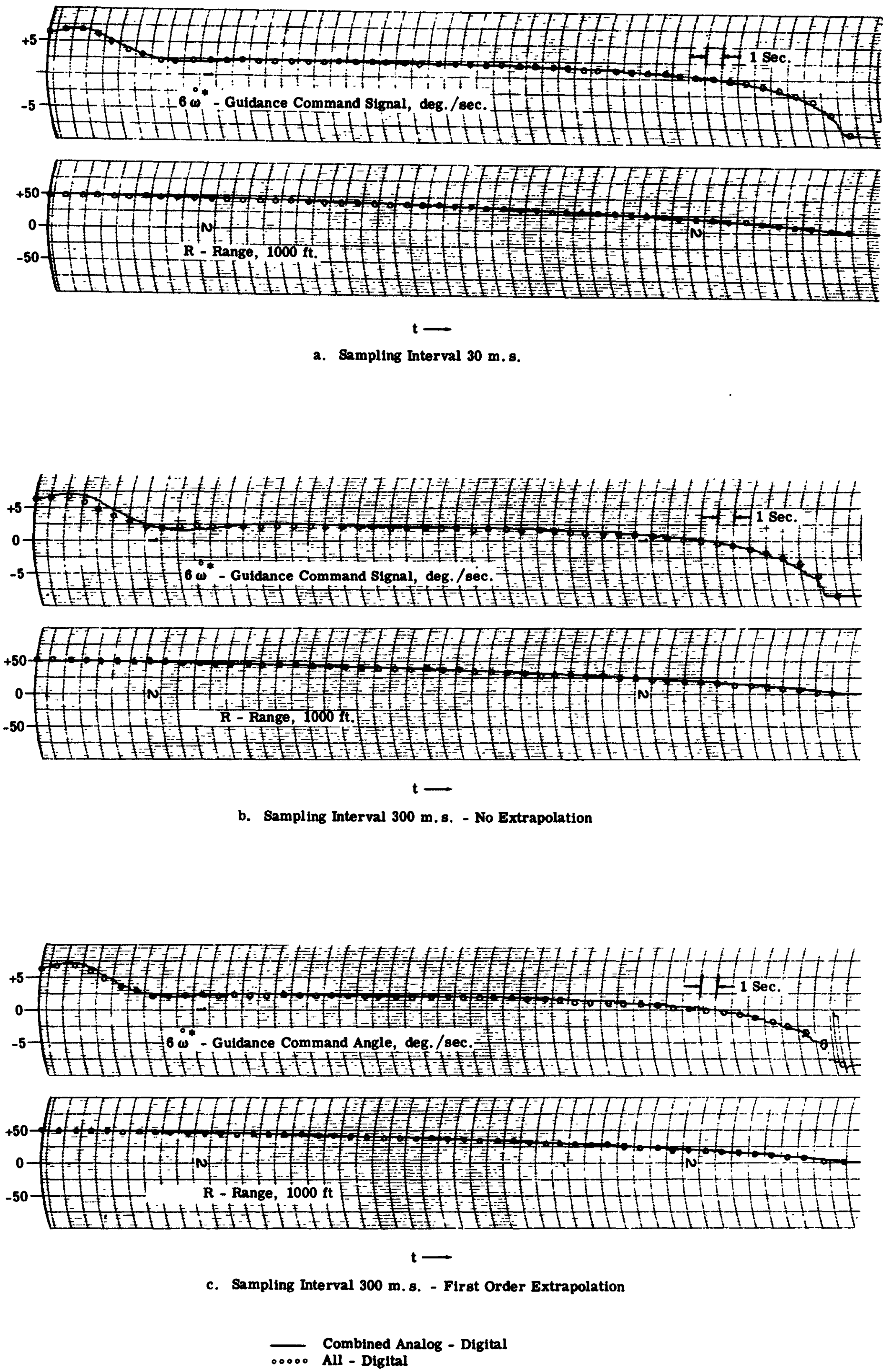

Write
$\theta_o, \overset{o}{\theta}_o$
Send REAC Into Operate
Read θ
Convert θ To Floating Point
Overload ?
Yes
Send REAC Into Reset
Stop
No
Digital Calculations
$\overset{o}{R} \geq 0$?
Yes
No
$R \leq 350$?
Yes
Send REAC Into Reset
No
Read
$\theta, \overset{o}{\theta}, \overset{oo}{\theta}, \theta_c, \overset{o}{\theta}_c$
All Digital Calculations
Stop
Wait For Clock Pulse
Write R, ω, $\overset{o}{\omega}$*
Convert R,ω, $\overset{o}{\omega}$* To Fixed Point

Figure 6. Flow Chart for Combined Analog-Digital Solution.



a. Sampling Interval 30 m.s.

b. Sampling Interval 300 m.s. - No Extrapolation

c. Sampling Interval 300 m.s. - First Order Extrapolation

Figure 7. Missile Intercept Solutions.

case the time delay due to sampling, for which no attempt at compensation has been made, is taking effect. In particular, the guidance command signal $\dot{\omega}^*$ is delayed .3 second. When a simple extrapolation technique is applied on $\dot{\omega}^*$ the variation from the all-digital solution is reduced considerably as shown in Figure 7b. The switch-over to the all-digital portion of the combined analog-digital simulation occurs when the range reaches approximately 180 feet. The final range in this case is 19 feet.

The dynamic equations being solved by the analog computer require no nonlinear elements and the equations being solved by the digital computer have parameters which do not vary rapidly. These facts coupled with the knowledge that a sampling range of 20 per cycle with extrapolation is sufficient, indicate the feasibility of running the problem in one-tenth of real time. The combined analog-digital solution then results in a time improvement of about 65 to 1 over the all-digital solution.

Conclusions

The missile intercept simulation with its associated scaling and expensive computer time problems exemplifies only one area where a combined analog-digital technique is useful. Computer controlled systems that are presently envisioned for future aircraft, missiles, and other space vehicles will require the handling of discrete and continuous information. Hybrid computation is the logical choice for the simulation of these inherently hybrid systems. Since an analog-digital computer link is by nature a sampled-data device it is a valuable tool for research in sampling theory. Stability studies and error analyses are planned using the hardware to simulate the sampled data-system under investigation.

Acknowledgment

The authors wish to acknowledge the valuable assistance of Mr. Hugh Winton who, in connection with these combined analog-digital investigations, was responsible for programming and operating the digital computer.

CONTRANS
(Conceptual Thought, Random-Net Simulation)

David Malin
*Walter Johnson High School**
Rockville, Maryland

ABSTRACT

CONTRANS is a computer simulation of a physiologically-oriented reasoning and problem-solving model. The model employs several layers of semi-random nets to recognize and associate sensory patterns, a screen-like memory and representation medium, and several motor functions enabling the nets to control scanning of the memory. Meaningful assemblies of data are created and manipulated by the model which then utilizes them to modify its own ability to handle future data.

Data and heuristic principles accumulate as stored results of past experience. Two kinds of logical operations are employed; one to explore the data rapidly to evaluate tentative strategies and solutions, the other to proceed by demonstrable intermediate steps, creating progressive intermediate goals.

Introduction

CONTRANS is designed to contribute to the implementation of a general plan for gaining insight into human intelligence. This plan is related to work by Hebb [1], Rosenblatt [2], Uttley [3], and others on the design of randomly connective network models for learned perception and on the non-physiological modeling of thought processes in the heuristic programming work of Minsky [4], McCarthy [5], and Newell, Simon, and Shaw [6]. The strategy involves the application of physiological findings to the construction of random-net and other brain models, and the adaptation of logical systems and heuristic methods to psychological findings. The objective which is particularly sought in CONTRANS is the integration of logical systems and heuristic methods with the function and organization of physiologically-oriented neural models.

The work reported here is a computer simulation of a reasoning and problem-solving CONTRANS model modified by the substitution of computer matching procedures for the cognitive functions of semi-random nets. It is planned next to simulate the neural networks themselves. Later work will involve the simulation of highly inductive

*The author was a student in "Introduction to High-Speed Digital Computation," an ACM sponsored high-school course conducted by G. G. Heller of IBM to whom he is most grateful for encouragement and guidance. He is also greatly in debt to J. M. Farrar, Jr., also of IBM, for his kind and patient assistance and review. Appreciation must also be expressed for the constructive editorial criticism of H. E. Tompkins of NIH.

methods of hypothesis formation and testing as applied to the model, will attempt to remove constraints on the form of input data, and may deal with special applications.

The model for CONTRANS includes, along with recognition networks based on the perceptual models of Hebb and Rosenblatt, a medium for representing intact assemblies and statements of data, which, when focused upon, are available as input to the sensory units of the entire net. This provision was suggested by the discoveries of "experiential memory" by the clinical neurophysiologists, Penfield and Roberts [7]. Also, the random nets of Hebb and Rosenblatt are modified substantially to permit the recognition of conceptualized expressions (instead of statistically describable stimuli) and to form "cell assemblies" for association of different expressions on a temporary basis and after one exposure to the stimuli.

The operational aspects of the system, as simulated in the present work, may be described in input-output terms. Input data statements consist of two expressions, each within a set of parentheses, one of which expressions must include a relationship, such as "implies," "equals," "includes," "hits," "is hit by," etc. An expression within a set of parentheses may be a simple substantive or may itself consist of two expressions of a form similar to that of the entire data statement but capable of using such additional relationships as "divided by," "or," "near," etc. There is no limitation on the number of possible relationships within relationships.

Statements defining the nature of the desired conclusion may be formed in the same manner as data statements, except that their imperative character must be noted. Such goal statements, along with heuristic statements and general principles, may be stated with one or more semantically undefined terms (which the program will define by position or relationship).

Output consists of a series of statements which obey the same rules, are deducible from the input statements, and are relevant to the general problem-solving strategy. These culminate in a statement meeting the requirements for the conclusion.

Two different methods are used in conjunction to obtain this result. One method operates primarily by making substitutions between various components of different input statements, thus creating intermediate statements which may be acted on in the same manner; new sub-goals are continually generated in this mode of operation. In order to give direction to this process, a companion method is employed which (due to chains of associations within the simulated net) can evaluate tentative strategies and tentative solutions without resorting to demonstrable intermediate steps.

CONTRANS is capable of benefitting from any heuristic or logical system whose principles can be stated according to the rules for input statements. For example, the "transformational" method of linguistic synthesis is particularly applicable. However, the system by itself is incapable of solving problems unless provided with at least the same data and principles an uninformed human would need (e.g., for mathematical problems, and transposition laws of algebra). On the other hand, past data and conclusions of all sorts can remain in memory; thus each new problem might be seen in the light of past experience and its solution might provide help in organizing future experience.

Related Work

CONTRANS is related to a number of other programs for the mechanization of nervous function and thought processes. Some of these models represent the learning process involved in neural discrimination and perception. Among the dynamic models whose learned perception possibilities have been tested in this manner—with varying results—are D. O. Hebb's cell assembly theory [1] as simulated by N. Rochester, et al [8], F. Rosenblatt's perceptrons [2], and A. M. Uttley's conditional probability machines [3].

All three of these models of the learning of discrimination and perception rely on the variation of the response patterns of neurons (partial models of nerve cells) in accordance with statistical probabilities rather than upon a specified predetermined structure of fixed connections such as would be involved in a system entirely grounded in symbolic logic.

Indeed, in models of this kind, connections between individual components (e.g., neurons) are usually assumed to be randomly distributed, hence the generic designation of this type of model as "random nets." Learning is visualized in these models as involving the attainment of differentiated powers of transmission (or "amplification," "weight," etc.) by pathways, as a function of past experience. Thus, in Uttley's system, the amplification factor of a pathway between two nodes is directly proportional to the probability that when one of the nodes is excited the other will become excited too; and this probability is induced from the past history of simultaneous and separate firings of the two nodes.

The random-net, biologically-oriented models have several advantages over conventional digital methods of pattern recognition. Due to the simultaneous use of many parallel pathways, recognition is accomplished instantaneously and with a low logical depth as contrasted with the many sequential steps which must be taken in pattern analysis by digital techniques. Also, successful performance of a given perception is not greatly dependent upon any one component. In addition, some random-net models, such as the class C' perceptron of Rosenblatt, are capable of spontaneous self-organization without any interference from the experimenter. These advantages of random-net models generally have been used to discriminate among and to classify statistically describable objects, notably geometrical figures. It should be noted, however, that some of these advantages are possessed by the models but not by their computer simulations. Because of the parallel nature of the models, simulation on a sequentially oriented computer is rather awkward and time-consuming.

Another, quite different, approach to the mechanization of nervous processes, has been developed in the field of "artificial intelligence" or "heuristic programming." Here the aim is simulation of human thought processes rather than the simulation of a model for neural action. Particular emphasis is given to the formation and evaluation of problem-solving strategies. Examples of research along this line are found in the work of Minsky [4]; of McCarthy [5]; of Newell, Simon and Shaw [6]; and of Hovland and Hunt [9].

Thus, we have in one approach a number of biologically-oriented neural models which usually have been applied only to learned perception problems, and in another approach a number of heuristic models which attempt more general mathematical and linguistic problem-solving and reasoning tasks without being based on specific neural models.

The author's objective has been to find some medium to integrate these two types of approaches so that the simulation of thought processes (particularly, reasoning, problem solving, and linguistic synthesis) could proceed directly from the simulation of biologically-oriented models of neural networks. Interestingly, several workers in these fields have made statements which might be considered to indicate the potential value of an integrated approach. M. Minsky [10] has remarked that lack of parallel processing is a hindrance in heuristic work. Likewise, Y. Bar-Hillel [11] has written that the work of the Yngve-Chomsky group in mechanical translation probably will have to delve into learning automata before its theoretical basis can advance much further. On the other side, it was F. Rosenblatt, who, in a discussion with the author on the possible extension of perceptron techniques to new problem areas, first directed him to the work of the "heuristic programmers."

The key to one possible method of integration was provided by neurological experiment. The neurologists W. Penfield and L. Roberts electrically stimulated certain cortical areas of conscious patients undergoing brain surgery [7]. In some cases, the electrode stimulation brought forth a sequence of memories. In fact, the patients claimed that their visions were so vivid and complete that they were *reliving* rather than merely remembering the sequence of experiences, and that they again perceived everything of which they had taken notice in the original experiences.

Whether or not the last point is completely valid, certain empirical conclusions seem to be indicated by these results. There must be some form of memory (of currently unknown embodiment) which is capable of being controlled by the electrical activity of the networks, and whose entries are in such a form as to become available, on being called up, to the entire "consciousness," i.e., available as input to the receiving elements of an entire network. For example, a revolving tape containing a series of images and moving in front of a matrix of sensory units would be a memory system capable of *re-presenting* experiences in such an intact manner. The learned weighting of neural pathways, while making possible the *re-cognition* of a stimulus, would usually not make possible its *re-creation*, as seemingly required by the experimental evidence. Penfield and Roberts call the phenomenon they discovered "experiential memory." As mentioned before, nothing definite is known about its biological embodiment. It might be neural or even embodied in extra-neural electric fields. Because a screen, upon which images are projected, revolving before a set of sensory elements represents an analogy to this memory, we shall hereafter refer to this class of phenomena as a "screen-memory."

The suitable combination of a "screen-memory" and appropriate network recognition models allows us not only to recognize structures of data, as with the firing of a single neuron, but also to make records of such data in its original integral form, and through this to create and manipulate new meaningful assemblies of data.

Concepts Embodied in CONTRANS

It is necessary to distinguish between the conceptual model which serves as a basis for the simulations and the simulations themselves. The model is a conception of certain components indirectly abstracted from neurological evidence, together with some hypothesized relationships among these components. The simulations are computer representations of the behavior of this model.

The conceptual model which is simulated in CONTRANS is composed of random-net associational networks, a "screen-memory" as defined in the preceding section, and certain motor functions which allow events in the net to manipulate the memory and entries upon it. The "screen-memory," in all early simulations, will serve both input and storage functions, i.e., a computer storage address assigned as part of this memory may store either newly introduced data or some entry associated with a previous problem.

In the present work—which is Phase I of a series of simulation projects—the cognitive and associative capabilities of the networks are replaced by computer matching and list-making procedures. The networks themselves will be simulated in Phase II, work projected for the future.

It is not known how the "experiential memory" is interrelated with the action of human nerve networks. The prescribed "motor functions" in CONTRANS represent some of the possible forms this relationship might take. One motor function stops scanning of the memory when certain neurons are stimulated. Another causes substitutions of entries from one location to another when, under certain conditions, the same neurons are excited by the stimuli at both locations.

The above elements work together to perform the basic, recursively used logical algorithm of the system. This process is primarily a mechanization of syllogistic reasoning. The process may be illustrated by a highly simplified example. If one of the statements in memory is to the effect that A implies B, and we want to deduce further implications about A from this, we focus on B and find a statement, such as B implies C. The networks learn to associate A with B and B with C so that when C is focused on, the neuron representing B is excited, in turn exciting the neuron representing A. This causes C to be transferred to a position opposite A, replacing B, forming the expression (A)(C).

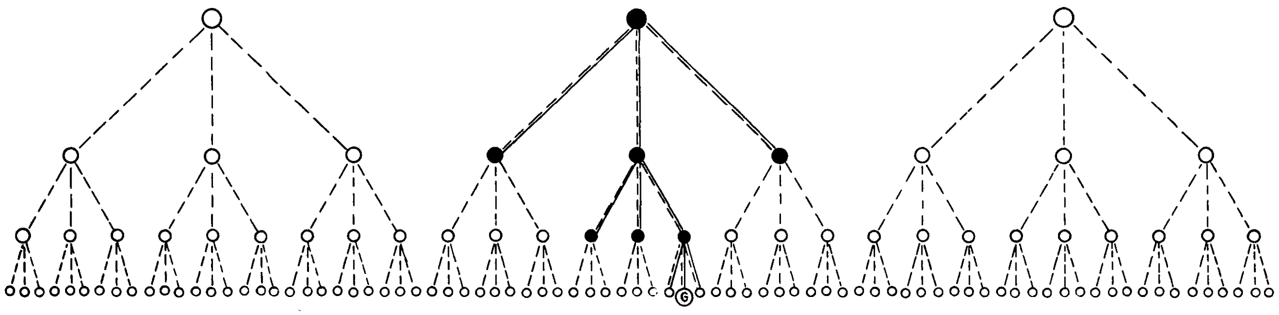
Similar processes are used on a somewhat more sophisticated level to substitute small components into long expressions and to modify entire statements without changing their meaning.

If this substitution process were the only problem-solving technique available, the CONTRANS reasoning process would simply be an exhaustive search for possible substitutions as represented by the dotted lines on Figure 1.

In order to form a short-cut through such an inelegant and uneconomical procedure, several built-in heuristics are included to organize the reasoning search. The whole process is motivated by a statement, written in much the same manner as other CONTRANS statements, as to what sort of a conclusion is desired.

In the problem-solving process we start with a data statement, one of whose components is identical to one of the components of the goal statement. The associated part of the data sentence then becomes a sub-goal. The system then concerns itself only with other statements that contain this same sub-goal.

As substitutions are performed, new intermediate goals are formed in succession to replace the original sub-goal and, hopefully, to lead closer to the required relationship with the other sub-goals. As each syllogism is performed, the new intermediate statement formed is printed as output. The



SYLLOGISTIC CHAIN PROCESSES

Each node is a statement.

Each line is a syllogism by which the lower node is derived from the upper node.

Ⓖ is the final desired conclusion.

A dashed line represents a syllogism permitted by exhaustive search and substitution procedures.

A solid line represents a step permitted by CONTRANS procedures.

Figure 1

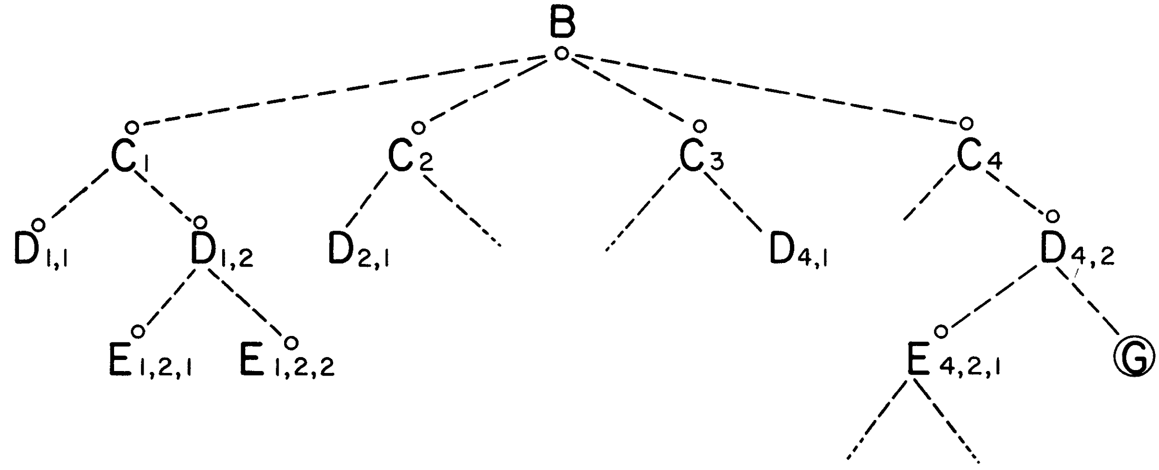
sequence of intermediate statements is analogous to the sequence of intermediate deductions in a geometric proof.

While this method provides some direction to the process by eliminating consideration of obviously unrelated statements, alone it would allow the search to select, and continue in, directions which take it away from rather than toward the answer. For example, let B be a sub-goal. Let us assume that B can be replaced by any of C_1, C_2, C_3, ..., C_n. Each C expression can in turn be replaced by another set of expressions. Perhaps, only C_4 leads along the chain to the final goal. However, without some additional heuristic, the process would have to start at C, and explore all the possibilities branching from C_1, and all the possibilities branching from these possibilities, and so on until dead ends are reached. Then the process would have to start over again at C_2.

We could have avoided this gigantic search if the system had known at an early stage that the choice of C_4 would take it towards the answer. In more general terms, it would seem that in order to go with any directness toward a solution, the system would have to know enough beforehand about the solution to avoid taking wrong turns that would lose it amid irrelevant alternatives. See Figure 2.

CONTRANS meets this problem with a second built-in heuristic—a method for directly or "intuitively" (i.e., without demonstrable intermediate steps) assessing whether a given substitution will take the process down the right path or down a blind alley. This process operates by scanning over the field of data and forming associations entirely within the neural network, rather than making substitutions on the screen memory.

Some of the simpler logic involved in this method is demonstrated by the example below. Let us assume statements to the effect that (1) A is B, (2) B is C, (3) B is D, (4) C is E, and (5) D is G. Let us assume that the statement A is G satisfies the goal statement (the



A SAMPLE SYLLOGISTIC CHAIN PROBLEM

A line such as / indicates further. branching not depicted.

Otherwise, notation is as indicated in Fig. I and in the text.

Figure 2

general form of the answer). Obviously the relation, A is G, is implied by the statements as they now stand. Random nets scanning over this series could learn to associate A with B, B with D, etc., until neurons representing A is G are stimulated, signalling the sufficiency of the data. If, however, we combine statements (1) and (2) and substitute C for B in (1) (a step which takes us in a wrong direction) we get (1) A is C, (2) B is C, (3) B is D, (4) C is E, and (5) D is G.

In this case A is G is not implied, the correct neurons will not fire, and the substitution must be undone. If we now go on to substitute D for B in (1) the network scan will give the go-ahead and we will proceed down the correct path.

The method allows us to make mistaken moves such as substituting C for B. However, it corrects all such moves before they can lead to further mistakes, e.g., it does not allow us to go on and substitute E for C. Thus the "intuitive" aspect of CONTRANS allows us to make tentative false starts but not to wander down blind alleys. The contrast between the extent of the black and dotted lines on Figure 1 demonstrates the extent to which the built-in heuristics of CONTRANS limit a typical search.

This aspect of CONTRANS might be considered akin to feedback concepts and Ashby's homeostat [12], [13] in particular. The latter operates by the weakly constrained generation of information and then by the more highly specific selection of that information. In CONTRANS, information generation is initially constrained by the requirement of performing correct syllogisms only and by the system of intermediate goals. Information thus generated is then selectively evaluated by the "intuitive" check to keep the system as close as possible to the direct path toward the deisred state.

Another feature of CONTRANS is that the intermediate and final conclusions formed in one reasoning experience can accumulate on the "screen-memory" to aid in organizing future experience. In this way, the effectiveness of the system would tend to grow.

Input used with a CONTRANS system takes the form of slightly modified English sentences as described later on. Goal statements and output are written in the same form. The model is capable of benefitting from any heuristic or grammatical principle that can be written according to these minimal rules. Particularly applicable are transformational statements, rules defining a transformation of an expression of a certain class, such as would leave unchanged the informational content of that expression.

Of course, it is possible that no solution may be implied by any of the data or past experience available. The "intuitive check" process is used at the outset to determine whether or not this is the case.

The experimenter, when dealing with an inexperienced CONTRANS simulation, must provide as much data for problem solving as a logically competent but unknowledgeable human being would need. When dealing with a somewhat experienced simulation, i.e., one that has accumulated useful data and conclusions from previous attempts to solve related problems, the experimenter must gauge the extent of that experience before deciding how much data will be necessary, if he wants to be absolutely sure of getting an answer. This is analogous to the process of a teacher deciding how much information he must provide along with a test question in order to make it a fair one for his pupils.

While the CONTRANS model itself requires hardly any particular rules of syntax, in order that consistent results be obtained, consistent syntax must be used. For example, it makes no difference to CONTRANS whether statements with transitive relationships are written with their meaning proceeding from left to right or vice versa. However, incorrect implications will be drawn unless all transitive statements are written one way or another. Transformational statements may be included with other input to aid the handling of syntax. For example, a statement might be included to the effect that a substantive S followed by a transitive verb is equivalent to S alone. This would aid in the formation of transitive substitution chains.

Because of the flexibility of CONTRANS, the behavioral complexity of the system is primarily a function of the complexity of its experience rather than the complexity of its initial design. For this reason, after a certain amount of experience a CONTRANS simulation would take on the seeming unpredictability of human behavior, unless close watch were kept historically on all the input statements and their possible interactions within the system.

To summarize, the model is a combination of "screen-memory" for storage of intact assemblies of data, "neurodynamic" nets to recognize and associate entries, and motor functions which, stimulated by excitation of certain neurons in the nets, control scanning of the "screen-memory" and cause substitutions to be made between various entries of that memory.

In the model, these components work together to perform syllogisms—the nets forming associations between various parts of selected statements found on the screen memory, and the motor functions, triggered by the nets, causing substitutions to be made between appropriate parts of these statements. This basic process is controlled by a succession of intermediate goals which are in turn selected through a "rapid-scan" use of the model. Except for a few rules, input requirements are flexible and externally introduced heuristics may be used. The model, motivated by a goal statement, analyzes the data statements and outputs a series of intermediate conclusions, followed by a final conclusion, the answer.

Some Troublesome Designations

When dealing with the mechanization of intelligent processes, it is all too easy, unfortunately, to give false impressions of accomplishment. The most obvious and classic example was the widespread public use of the term "electronic brain" in connection with high-speed computers. Similar—but more subtle—misconceptions can occur readily in regard to the use of computers to simulate various neuro-behavioral models. In particular, there is always the question of how much the success of a particular mechanization really implies about the true functional organization of the nervous system.

At present, the question arises: Is CONTRANS, with its component cell assembly nets, really a "brain-model"? The author believes that CONTRANS is not a close model of the nervous system since, because of the complexity involved, it has not been possible to fit the characteristics of the components to many of the experimentally discovered parameters of neurophysiology; since, because of lack of evidence, several straightforward but unverified assumptions had to be made about the relationship of the networks to the "screen-memory"; and since, because of almost total lack of evidence, no biological embodiment has been hypothesized for the "screen-memory."

CONTRANS would seem, however, to be more than another variety of artificial intelligence; if not an actual brain model, it is, nevertheless, physiologically oriented since its component neuro-dynamic nets represent a type of function which, in view of current evidence, seems typical of the nervous system and since the "screen-memory" feature is inferred directly from experimental findings. The first simulations explore a method of organization for brain models. In this sense, CONTRANS is a model for brain models. The logic of the general system is such that, as more accurate perceptual models are derived, they could be fitted into the larger system, and CONTRANS could approach brain-model status.

The Phase I Simulation

The present work, Phase I, simulates the CONTRANS model with computer matching and list-making procedures replacing the cognitive and associative functions of the neural nets. The general procedure of the simulation is outlined below and represented in flow diagram form in Figure 3.

Input to the system is in the form of English sentences modified according to several simple rules. Parentheses are included to indicate the extent of relationships. For sheer convenience, a limit is imposed on the depth of parentheses within parentheses, but a method is indicated for adding additional depth, by the use of satellite statements resembling subordinate clauses.

Also for programming convenience, single letters are agreed to stand for words, and for phrases and clauses when it appears to be unessential for those phrases and clauses to be broken down any further.

A goal statement is presented along with data and heuristic statements; it is written in the same fashion as other CONTRANS sentences. As explained previously, the use of CONTRANS does not imply any particular grammatical syntax, but the syntax used should be consistent.

Output statements have the same characteristics as input statements. Output consists of a series of intermediate statements culminating in a final answer. In keeping with the cumulative nature of the "screen-memory," output statements are punched onto cards to be entered with future input (as well as printed out).

Once data is entered into core memory, the CONTRANS program analyzes each statement into individual parentheses. The program then finds a place to start its reasoning process by searching for a data statement that contains an element of the goal statement. The associated part of the data statement then becomes a sub-goal. For instance, if side 2 of statement 5 is found to be equivalent, by matching, to a side of the goal statement, then side 1 of statement 5 becomes a sub-goal. (A side is one of the two prime divisions of a statement, somewhat analogous to a subject or predicate in conventional grammer; in a CONTRANS statement it is the contents of one of the two principal sets of parentheses.) The program then searches for elements of the sub-goal in other statements. If one of these elements is found, a tentative substitution is made and control is transferred to a program which simulates the "intuitive check" method.

In simplified terms, the method operates by going over the data to form lists of associated expressions. If, at the end of this process, both sides of the goal statement are found in the same list, then the tentative substitution is made permanent, the modified statement is punched and printed out as an intermediate conclusion, and the process begins again with a new sub-goal.

If, at the end of the process, the condition is not met, the substitution is undone and the search for elements of the sub-goal begins again. This type of operation is done recursively until the final answer is derived. In this manner the program, motivated by goal statements, accepts data and heuristic statements in modified but comprehensible English, and draws from them a series of implications relevant to, and culminating in, a final answer.

As of this writing the program for Phase I is undergoing debugging. The program is written in FORTRAN for the IBM 709.

Future Phases

In Phase II, scheduled for the Fall and Winter of 1961, a class of cell-assembly neural net models will be simulated and their learned recognition and association properties will be substituted for the computer

GREATLY SIMPLIFIED FLOW DIAGRAM

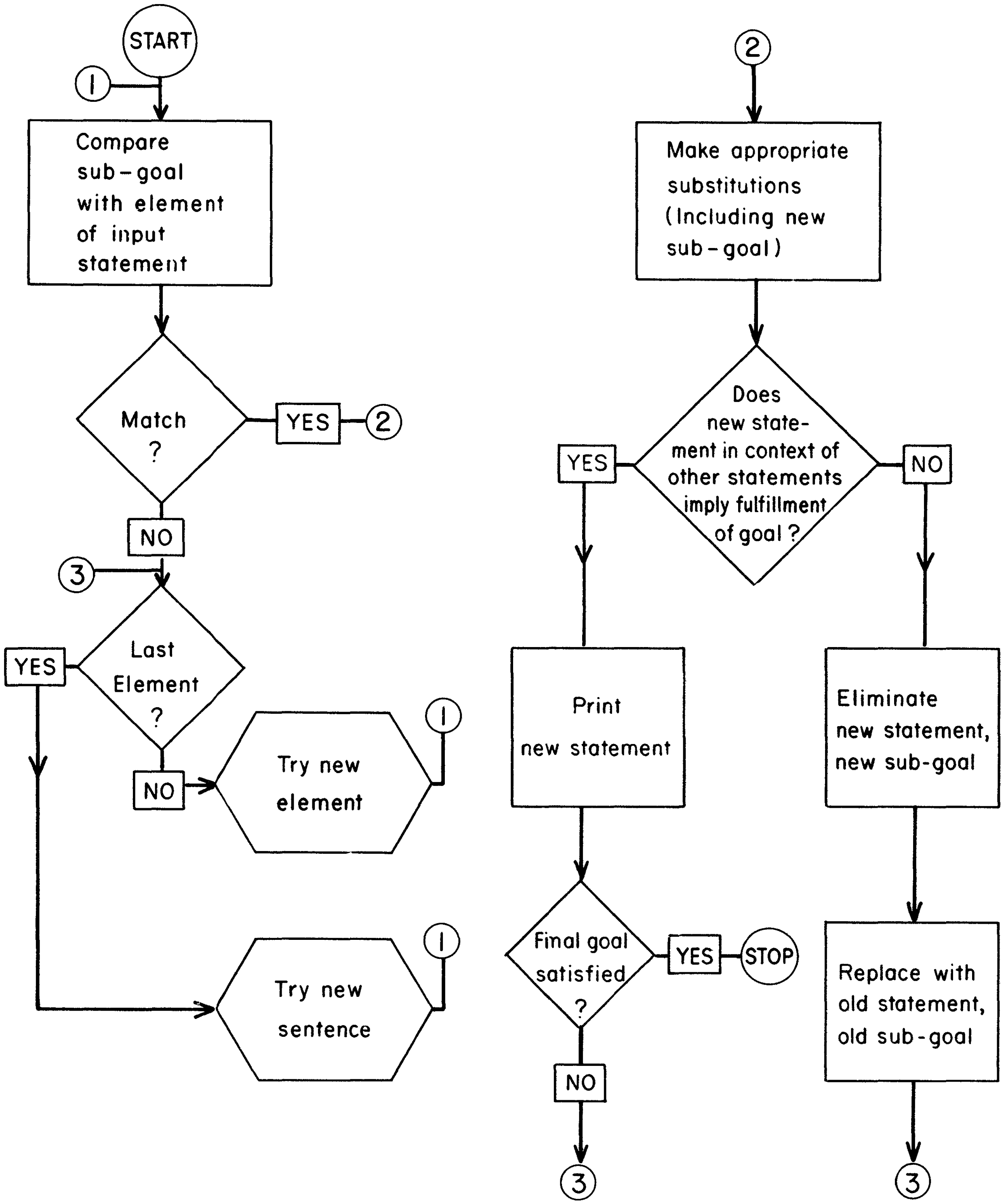

START
1
Compare sub-goal with element of input statement
Match ?
YES
2
NO
3
Last Element ?
YES
NO
Try new element
1
Try new sentence
1
2
Make appropriate substitutions (Including new sub-goal)
Does new statement in context of other statements imply fulfillment of goal ?
YES
NO
Print new statement
Eliminate new statement, new sub-goal
Final goal satisfied ?
YES
STOP
NO
3
Replace with old statement, old sub-goal
3

Figure 3

matching and list-making processes used in Phase I.

It should be noted that the neurodynamic nets, with their many parallel processes, are capable of recognizing stimuli without the many consecutive logical steps taken in sequential digital recognition or matching techniques. Thus, a physical embodiment of the full CONTRANS model would be a step toward combining the perceptive speed of a perceptron-type network with the flexibility and logical power of heuristic programming. However, since simulating a parallel operating network on a serially operating computer is a rather awkward process, the advantages of connective nets will not be reflected in actual running time.

The design of neurodynamic networks for CONTRANS presents several special problems. These problems arise from the fact that, while most neural net simulations are concerned with the discrimination of statistically describable stimuli such as geometrical figures, CONTRANS is concerned with the recognition and manipulation of symbol chains, expressions which are already conceptualized.

It will be assumed that geometrically perceiving networks, such as the perceptrons of Rosenblatt, are capable of recognizing letters and of representing different characters by the firing of different neurons. It will be assumed that the output of these neurons will be the input to the CONTRANS nets for the recognition and association of words, phrases, and clauses. It is primarily the latter networks that Phase II will be concerned with.

These networks, unlike most similar models, will be required to associate expressions after only one learning trial, and to recognize expressions on the basis of the order of their constituent symbols in time.

The first requirement may be met by the selection of an appropriate expression for the growth of the amplification value of a particular interneural pathway as a function of excitation and reinforcement.

For example:

Let V be an amplification scale.

Let $0 < V_o < 1$

Let N_i stand for one neuron and N_j for another.

Let an output P_i from N_i imply an input V_{ij} (P_i) to N_j

Let T be a time at which N fires.

Then

$$V_{ij_{t+1}} = V_{ij_t} + \frac{P_{j_t}}{P_{i_t}} (1 - V_o) - V_{ij_{t-1}} (1 - V_o)$$

This is a simple but satisfactory equation for the weight of the pathway leading from N_i to N_j. As simulated by FORTRAN programs for the IBM 1620 (with V_o equalling 0.2 and 0.4), V_{ij} rises sharply toward 1 with the first simulatneous firing of N_i and N_j and then tends asymptotically towards 1 with additional simultaneous excitations. With a firing of N_i without N_j, V_{ij} decreases sharply toward zero and tends asymptotically toward zero with further such firings. Thus this growth function allows sharp—almost "yes-no"—learning in one trial. Such a sharply reacting growth function would play havoc with the statistical sophistication of a geometrical perceptron. However, this equation would be quite suitable for the one trial association of two abstractions or identified symbols.

The second requirement, for recognition of order in time, will be met through the provision of reverberatory networks and neurons representing intermediate parts of words, such as syllables, and intermediate parts of sentences, such as phrases.

In the future, it should be possible to make more of a serious attempt at designing CONTRANS random nets that better fit the characteristics of the nervous system without changing the logic of the system as a whole.

In Phase III, CONTRANS will be used in the formation and testing of inductive hypotheses, and will thus attempt to make generalizations from separate occurrences.

It is hoped that, in Phase IV, constraints on the form of input sentences, particularly in regard to the use of parentheses, will be removed. An attempt will be made to divide "screen-memory" into temporary or permanent cumulative segments to reduce unnecessary scanning time.

In addition, there are possibilities for special applications and special embodiments. In final form, and in special-purpose embodiment, the system would accept ordinary sentences as instructions and data, would process them (perhaps in a not easily predictable manner) with the benefit of accumulated data and conclusions and, due to its parallel form, would in general, operate with much greater speed then its computer simulations.

References

1. Hebb, D., The Organization of Behavior, John Wiley and Sons, New York, 1949.
2. Rosenblatt, F., The Perceptron - A Theory of Statistical Separability in Cognitive Systems, Cornell Aeronautical Laboratory, 1958.
3. Uttley, A., "Conditional Probability Machines and Conditional Reflexes," in Automata Studies, ed. by C. E. Shannon and J. McCarthy; Princeton University Press, 1956.
4. Minsky, M., "Some Methods of Artificial Intelligence and Heuristic Programming," in Mechanisation of Thought Processes, Her Majesty's Stationery Office, London, 1959.
5. McCarthy, J., "Programs With Common Sense," in Mechanisation of Thought Processes, Her Majesty's Stationery Office, London, 1959.
6. Simon, H., "Modeling Human Thought Process," Proceedings of the Western Joint Computer Conference, National Joint Computer Committee, 1961.
7. Penfield, W. and Roberts, L., Speech and Brain-Mechanisms, Princeton University Press, Princeton, 1956.
8. Rochester, Duda, Haibt, and Holland, "Tests on a Cell Assembly Theory of the Action of the Brain, Using a Large Computer," in Transactions of Information Theory, IRE, 1956.
9. Hunt, E. and Hovland, C., "Programming a Model of Human Concept Formation," in Proceedings of the Western Joint Computer Conference, National Joint Computer Committee, 1961.
10. Minsky, M., 'Descriptive Languages and Problem Solving," in Proceedings of the Western Joint Computer Conference, National Joint Computer Committee, 1961.
11. Bar-Hillel, Y., "The Present State of Mechanical Translation," in Advances in Computers (edited by F. Alt), Academic Press, New York, 1960.
12. Ashby, W. R., "Design For an Intelligence Amplifier" in Automata Studies (ed. by C. Shannon and J. McCarthy), Princeton University Press, Princeton, 1956.
13. Ashby, W. R., Design for a Brain, John Wiley and Sons, New York, 1960.

DIGITAL-TO-VOICE CONVERSION

Evan L. Ragland
Motorola, Inc.
Military Electronics Division
Chicago, Illinois

INTRODUCTION

For the past two years Motorola has maintained a program to study and develop digital-to-voice conversion techniques. This program has examined the needs for digital-to-voice convèrsion and has, through feasibility experiments, studied some of the promising techniques. This paper reports on this work.

This report considers the entire subject of digital-to-voice from the basic need to the ultimate application. Particular emphasis is placed on possible technological solutions and on experimentation to determine the feasibility of these approaches.

The problem statement presents the basic needs and technological problems associated with digital-to-voice conversion. Several possible solutions are examined in the consideration of this problem. From these solutions a system operation is described and schematically outlined. A series of feasibility experiments, performed in the laboratory directed toward establishing the necessary background for development effort, are reported. The results of these experiments are covered in detail. Finally, some future applications of ultimate development equipment are discussed.

PROBLEM STATMENT

General

The increasing digital environment in which man must coexist creates a basic need for improved communications from machine to man. As the use of data processing and data communication systems expand there are numerous instances where voice communications to the man operator or monitor from systems of automatic assistance or control are imperative.

Economic

Generally, these needs are accompanied by a requirement for an inexpensive method for conversion of digital data to voice since and in most instances, the need arises in a system of control or centralized information storage where outputs to many human communicators are required. Therefore, it is important that the approach for conversion of the digital information to voice be inherently inexpensive.

Application

The nature of information to be transmitted in such systems varies widely. Therefore, it is important that the system for conversion not constrain the machine to some limited group of messages. Unless several constraints are placed upon the variation of communciation a general purpose capability must be considered. Flexibility must be such that the machine can basically communicate any given instruction or information sequence.

Human

The human communicators must not be inconvenienced or disturbed by the form of the messages. It must be as closed to human speech and human conversational delivery as is possible. Therefore, it is important to have imperceptible delays between the vocalization of words and the words vocalized in a non-synthetic manner.

Physical

The conversion equipment, in many cases, must meet the requirements of light weight and compact size. Therefore, it is important that the techniques employed for conversion require little space, weight or power.

Technical

The black box digital-to-voice converter must operate with any digital data communication or processing system. It must not constrain, because of its own limitations, either the coding or format of these systems. It must not require that the system store messages forms or structures. Essentially, the converter should be able to accept any code input and translate this to vocal information form. It should be capable of buffering lengthy messages and translating these at the vocalization rate for the human communicator. This can be accomplished in several different ways.

POSSIBLE SOLUTIONS

There are two principal avenues of approach to the construction of audible, intelligent English word forms from digital information. These are:

1-synthetic reproduction from tones and
2-vocabulary look up.

The latter of these methods has been the principal study and subject of experimentation in the work reported. Vocabulary look up was selected since it seemed to offer the general purpose capability for message transmission within the bounds of the operational, physical, human, economic and technical requirements of application.

It can be shown statistically that within the bounds of the English language the general purpose message capability can be accommodated by a limited vocabulary. For example, consider the chart of Figure 1. The entire English language consists of some 600,000 words. The frequency of usage of these words varies dramatically. The average human vocabulary is about 15,000 words. Of these, 50 comprise approximately 50% of normal conversation and correspondence. This usage remains remarkably stable, even when highly specialized technical papers are the subject of account. One thousand words will accommodate over 80% of normal usage. With a 1000-word vocabulary it is possible to express most ideas or informational sequences. It therefore seems reasonable to assume a vocabulary of several thousand words could provide machines a conversational capability.

It has been concluded that a 1 to 5000 word vocabulary provides a communications capability from which substantial advantages in machine to man communications can be

AVERAGE CHARACTER-FORM WORD.......	256,000,000,000,000 WORDS
ENGLISH LANGUAGE..................	600,000 WORDS
AVERAGE WORD-FORM.................	64,000 WORDS
AVERAGE VOCABULARY................	15,000 WORDS
80% USAGE.........................	1,000 WORDS
50% USAGE.........................	50 WORDS

Figure 1. Vocabulary Statistics

derived. The objective of this study has been to examine methods of storing and retrieving vocabularies of from 1 to 5000 audible English language words. The storage retrieval system should be a self-contained unit and should meet the general operational requirements stated previously.

One encounters a number of physical and technical problems in considering the storage of audio information. For example, the average time required for a spoken word is less than three quarters of a second, but many frequently used words require up to one second for reproduction. Therefore, it is necessary to accommodate variable reproduction time for vocabulary words to meet this random requirement. Also, it is necessary to find a means for high-density packing of information. Consider, for example, that the average word requires 0.6 second for vocalization and that it may contain up to 3000 pieces of information per second. If 5000 words are to be stored, the capacity for storing must be 5000 x .6 x 3000 bits of information—or 9,000,000 bits—a substantial memory requirement. There is available, however, another important characteristic in vocabulary storage. It can be fixed storage. It need not be changeable or erasable. Therefore, photographic storage is open to consideration.

In storing large vocabularies it is necessary to develop appropriate access time which permits both the imperceptible access of words and the scanning of words at audible rates.

This problem of scanning is illustrated by Figure 2. The essential requirement present is for some method for translation of information time base from the high-speed scan requirements of the digital system to the low-speed vocalization rate. The time for scan is determined by the permissible time for look up and size of vocabulary. Time for read out is determined by the vocalization time required for a word and varies from a fraction of a second to as long as one second.

Considering the basic problem in Figure 2 and the operational requirements for the converter, it is possible to suggest three attractive solutions. First, it would be possible to mask a cathode-ray tube with the audio information and scan a digital field, directly associated with this audio information, to look up words in the memory store. Such a system is illustrated in Figure 3A. By changing the rate of scan of the electron beam between the digital and voice fields, the time base for the information read out can be effectively translated.

Another method of look up would be to have a moving photographic storage consisting of two fields. Look up of the digital field would be accomplished photographically and information would be strobed from the moving

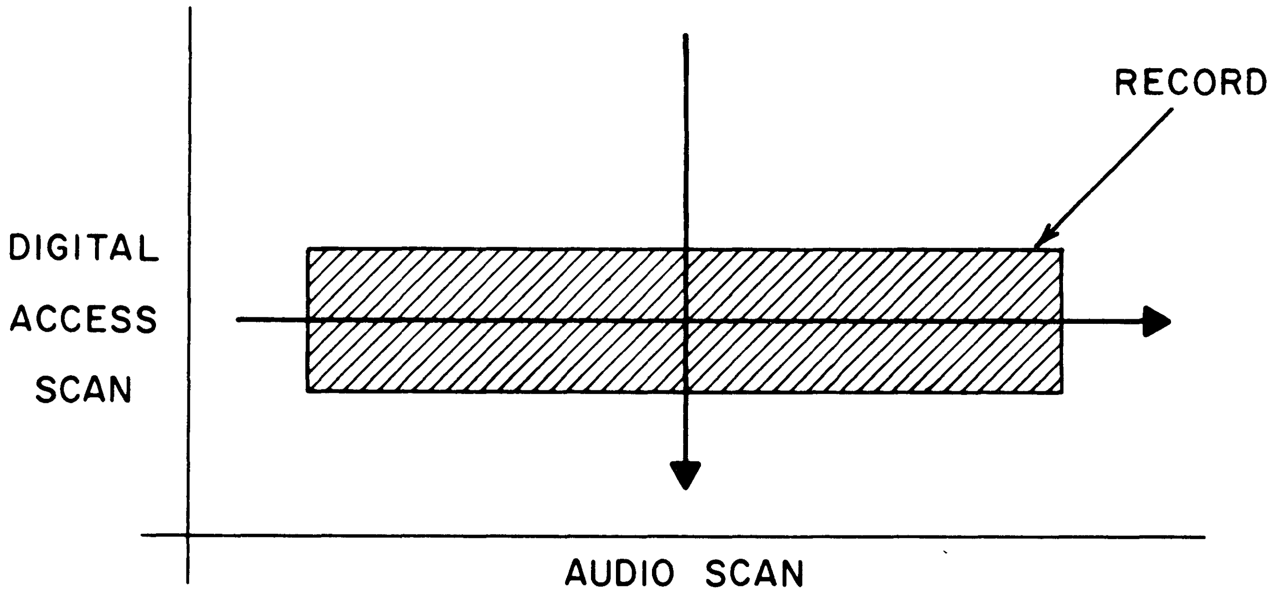


Figure 2. Scan Schematic

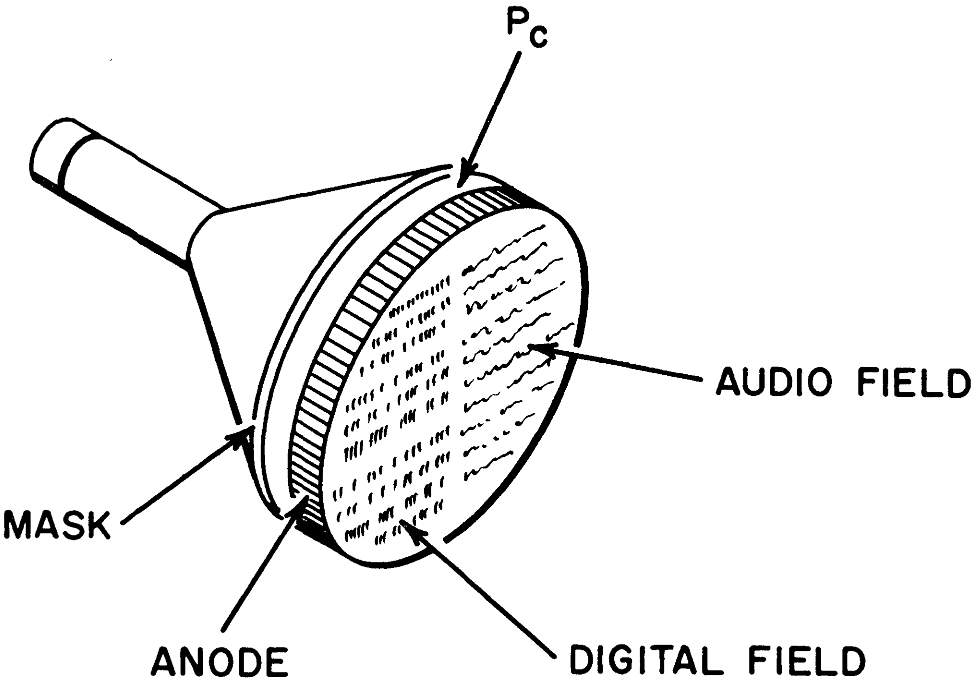


Figure 3A. Cathode Ray Tube Converter

audio record field by a short duration light strobe. There are two possible applications for this method as far as read out is concerned. Read out could be accomplished by a moving electron beam across the electrostatic image on a vidicon screen. Read out could also be accomplished by detecting the discharge pattern created by the strobed records on the surface of a memory photoconductor. These two methods are illustrated in Figures 3B and 3C. Of the three approaches illustrated, the approach using a moving photographic storage and a moving electrostatic image has been selected for the principal study to date. This method was selected because it could be readily implemented with simple laboratory apparatus and because it would not have the inherent requirement for high power supply and component stability required by the all electronic cathode-ray scanning approach.

System Explanation

The application of the selected solution is illustrated in Figure 4. This schematic organizes a buffer memory for storing digitally-encoded messages, an optical system for scanning a digital field, a coincident system for selecting from an audio record field one of the audio records and an electrostatic system for reproducing the selected audio record from an electrostatic image.

To describe the operation of the device, consider that a digital message is received and stored on the magnetic band of the rotating drum. The first word of this digital message, which consists in the example of 11 bits, is shifted into the shift register by standard digital circuit techniques. When this word is in place in the register, the system compares the permutations of its stored value and the continuously changing permutation of the digital field. This is accomplished by a standard comparator circuit and a parallel 11-bit photoconductive detector. With parity, the comparator provides an output which triggers a short-duration strobe light. This short-duration light pulse exists only long enough to project the image of one word in the audio record field onto the moving photoconductive surface. The digital and audio fields are spatially related so the look up of the digital word in effect selects the audio word. The moving photoconductive surface has been charged to a uniform potential by a charge station. The incidence of light on the surface causes discharge in the pattern of the image. This discharge pattern is detected by an electrostatic pick-up, amplified and vocalized.

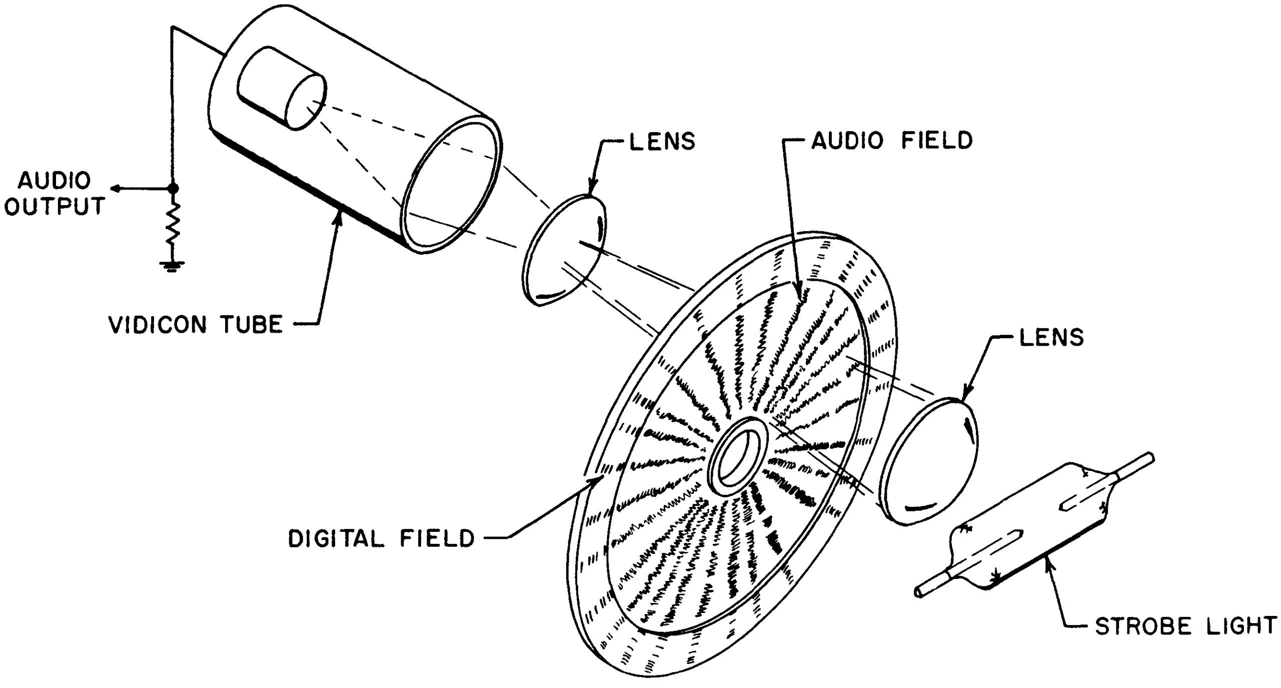


Figure 3B. Vidicon Converter

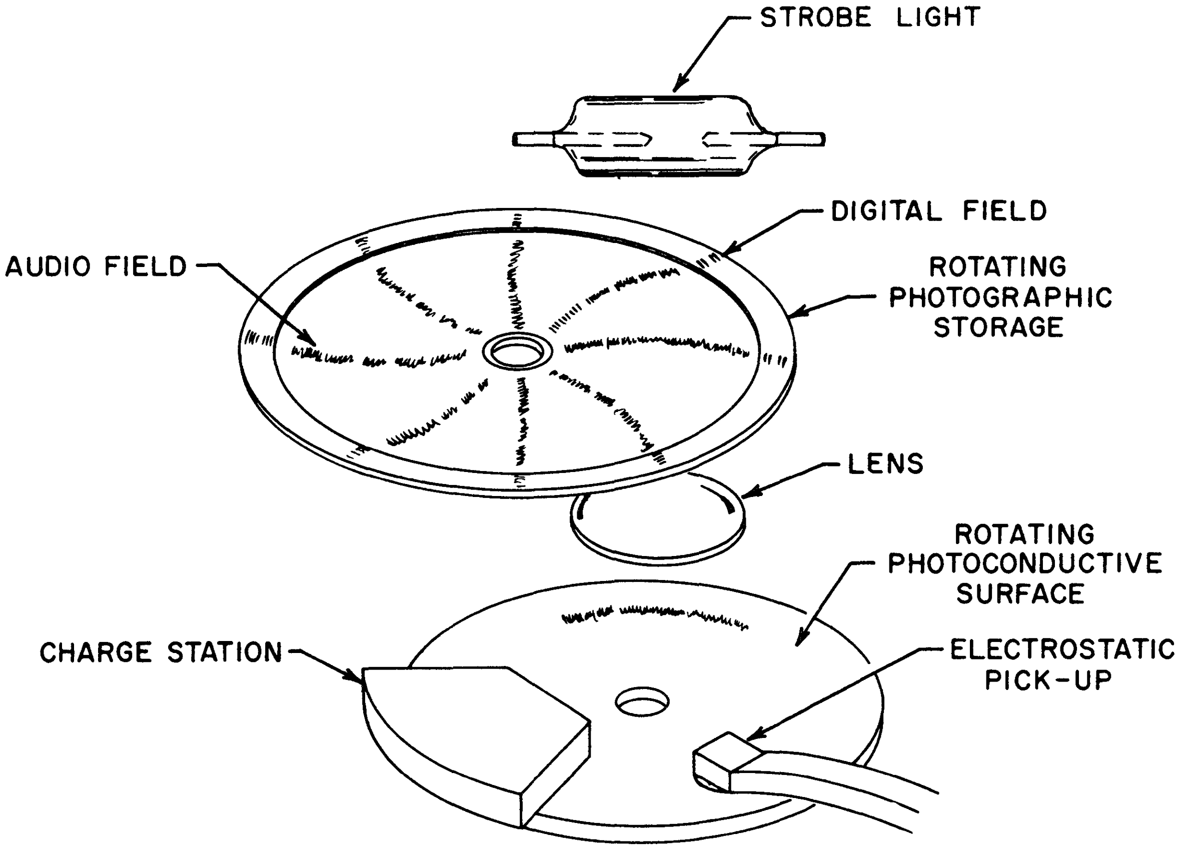


Figure 3C. Photoconductor Converter

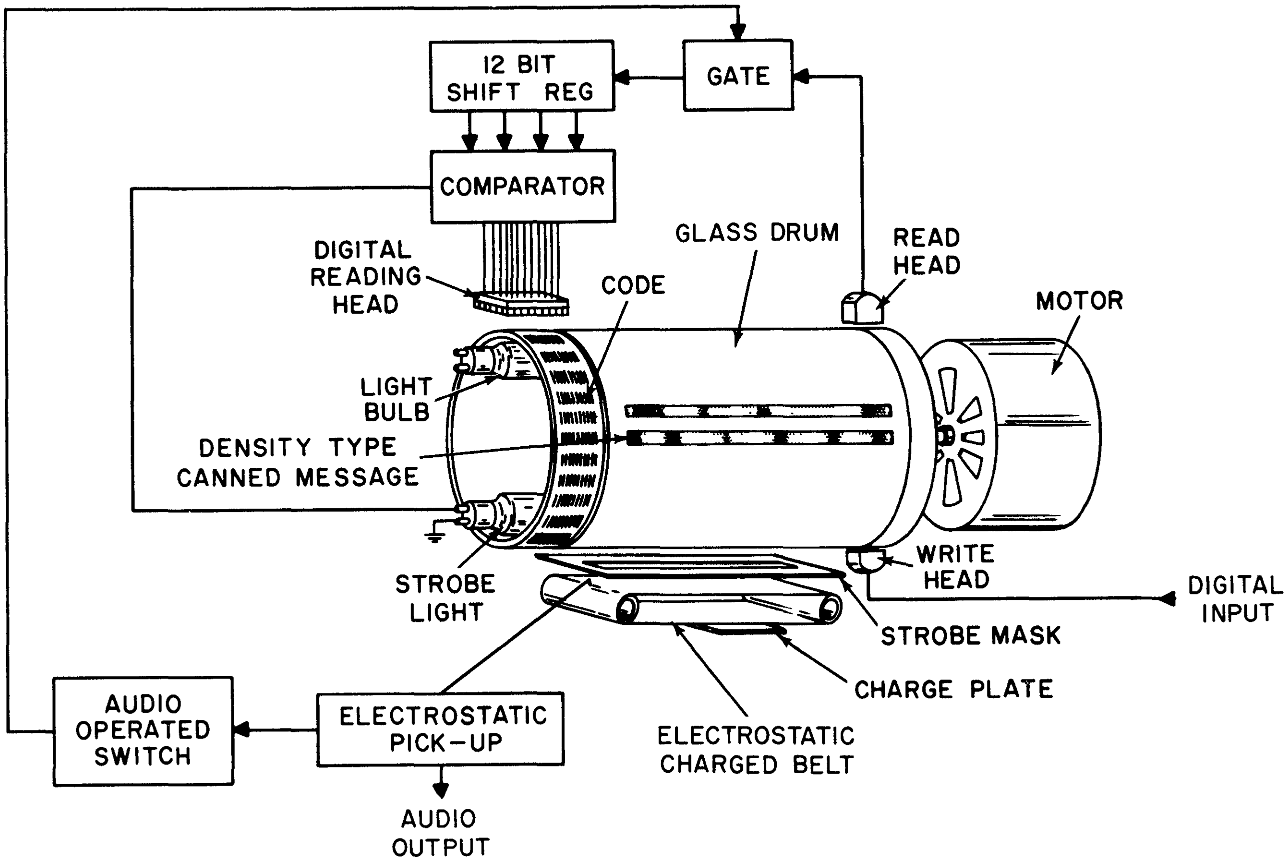


Figure 4. Digital-to-Voice Converter Functional Diagram

At the conclusion of the read out, it is possible to shift the subsequent digital word from buffer storage and to project it in electrostatic image form for immediate reproduction. Since it is possible to strobe information from photographic storage in times as short as two microseconds, the look-up time for even a large vocabulary of 1000 to 5000 words is imperceptible to the listener. Of course, it would be possible for the look-up time to be extended by taking advantage of the time for vocalization of the previous word. In some applications this is practical while in others it would require the communicating digital system to remain on line during vocalization. Therefore, it is important to have available the capability for high-speed look-up that is imperceptible to a listener between words.

Feasibility Experiments

The approach described incorporates a number of techniques which are well known and standard. These are techniques such as buffer memory, digital switching, etc. There are techniques required in the converter approach that are not as well known and deserved laboratory experimentation and consideration particularly in light of their conjunctive application. These were:

1. high-speed strobing
2. photoconductive imaging
3. electrostatic detection
4. system optics
5. methods for recording

It was decided that a test apparatus would be developed which would typically represent these techniques. This apparatus would provide a source for data and feasibility experimentation.

The schematic for this apparatus is illustrated in Figure 5. To shorten the length required for the optical system, a system was constructed in which the path was reflected through three mirrors. A strobe light was mounted in this optical system so that its high intensity flash, through a mask placed in the light path, could be focussed just before the lens. This system projected the image on a

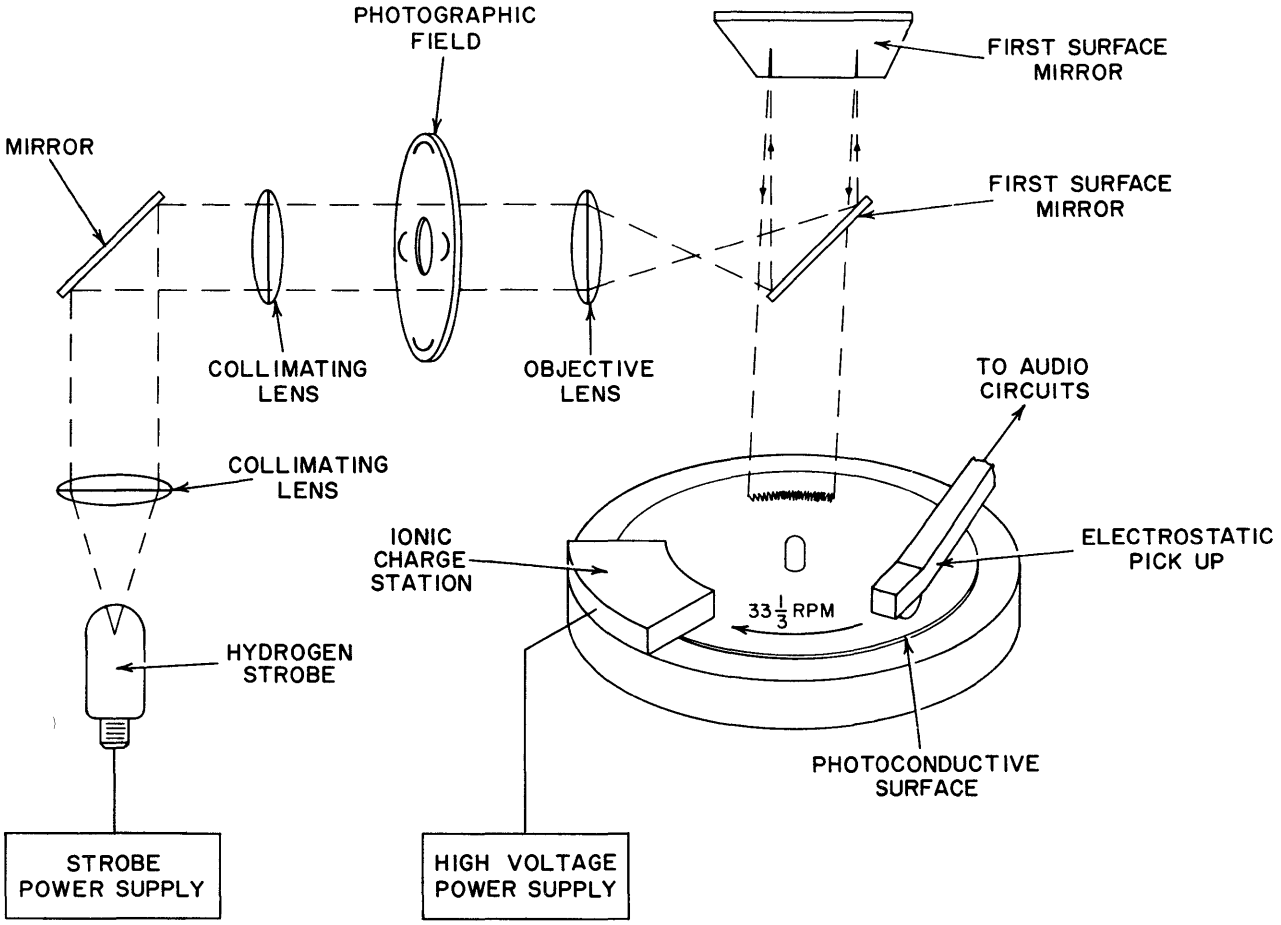


Figure 5. Feasibility Apparatus Schematic

moving turntable. The turntable, rather than a drum, was used because such a table was readily available and possessed good stable dimensional and dynamic characteristics. An ionic charge station was constructed and a mount was developed that permitted a manipulation of various types of experimental pick-up devices. Figures 6A and 6B show the laboratory apparatus developed for this purpose.

The problem of strobing is one of developing sufficient light intensity to create the necessary photoconductivity in the material in a short time span. The immediate objective of the early experiments was to determine whether or not millisecond light flashes would be reasonable. Later experiments moved into the microsecond region for the duration of light flashes. A General Electric type FT-230 hydrogen strobe light has been used in these latter experiments. This lamp requires 2000 volts and generates light pulses as short as 2 microseconds duration.

Photoconductive imaging on the surface of the moving turntable has been a subject of some study. From the work by Dutton, at the University of Rochester, and others, it was known that the grain of deposited amorphous selenium coatings, such as used in standard xerographic processes, had been measured to micron dimensions so this was not a principal point for investigation. The image problems studied were the response of the material to the short duration light flashes and the degradation of the image due to field dispersion and surface conduction. These experiments have been conducted by two methods. These are: electrostatic probing of the charge patterns and charged particle dusting of the selenium plates.

Electrostatic detection presents some serious problems. This is an area in which there has been only a limited amount of work and although H. W. Katz and others report in their text the fundamental types of probes which have been technically developed, the

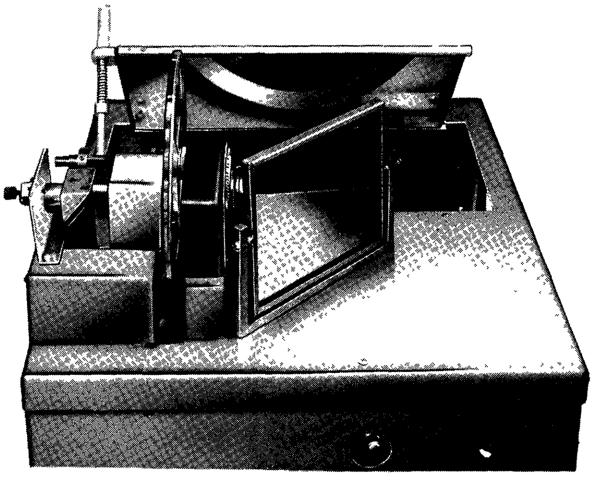


Figure 6A. Feasibility Apparatus, Optical Section

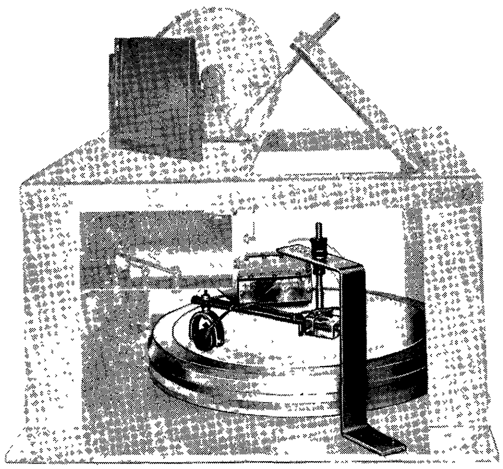


Figure 6B. Feasibility Apparatus, Pick-Up Section

development of a high resolution electrostatic probe still has not been accomplished. Very little has been done to improve upon the probes discussed. The probe used in the laboratory device is illustrated in Figure 7. The resolving capability has been observed to be as expected, from 3 to 6 times the spacing of the probe to the plate.

System optics might not be considered critical; however, the ultimate requirement for the converter to be compact demands that the optical storage be small. This will require high magnification ratios between storage and detection. Of course, this is related to the resolution of the detector, and with better detectors this problem becomes less critical. It was anticipated, however, in beginning the experiments that audible tracks as narrow as 10 mills would be required in the system and that the frequency recording capabilities of these records and the transmission of the optical system should be at least 1 kc/inch of length. Various track diameters and widths have been tested by the program. Optical records of the dimensions described have been successfully made. Severe problems of registration have been encountered with multiple record vocabulary composition. Current effort is being directed toward new techniques to solve these registry problems.

Two methods are available for photographic recording of audio. These are: (1) density and (2) variable area. In density recording, the density of the photographic image is used to indicate the level of audio. In variable area recording, the integrated area of the sound track for a given resolving segment is used to indicate the audio. For proper evaluation of the photosensitive process, it was necessary to test both methods. For this purpose, and because the sound tracks due to the nature of the test apparatus were circular, it was necessary to build a photographic recording apparatus. This is illustrated in Figures 8 and 9. Both density and variable area records were made and tested.

Results

The experiments have definitely demonstrated the feasibility of the approach selected for digital-to-voice conversion. Good, intelligent audio words have been reproduced by the apparatus. The normal characteristics, which are measured in audio, indicate that the performance of this system from the audible standpoint is acceptable. For example, the frequency response is 20 to 1500 cps at 15 inches per second. The signal-to-voice ratio is 15 to 20 db.

Experiments show that adequate light can be obtained from the high-speed strobe to create useful images when the pulse duration of the strobe is as short as 2 microseconds. The particular strobe light used was not the highest intensity light source available. Higher intensity strobes having different electrode materials can be constructed.

The photoconductive imaging was found to be more than adequate for the purposes of

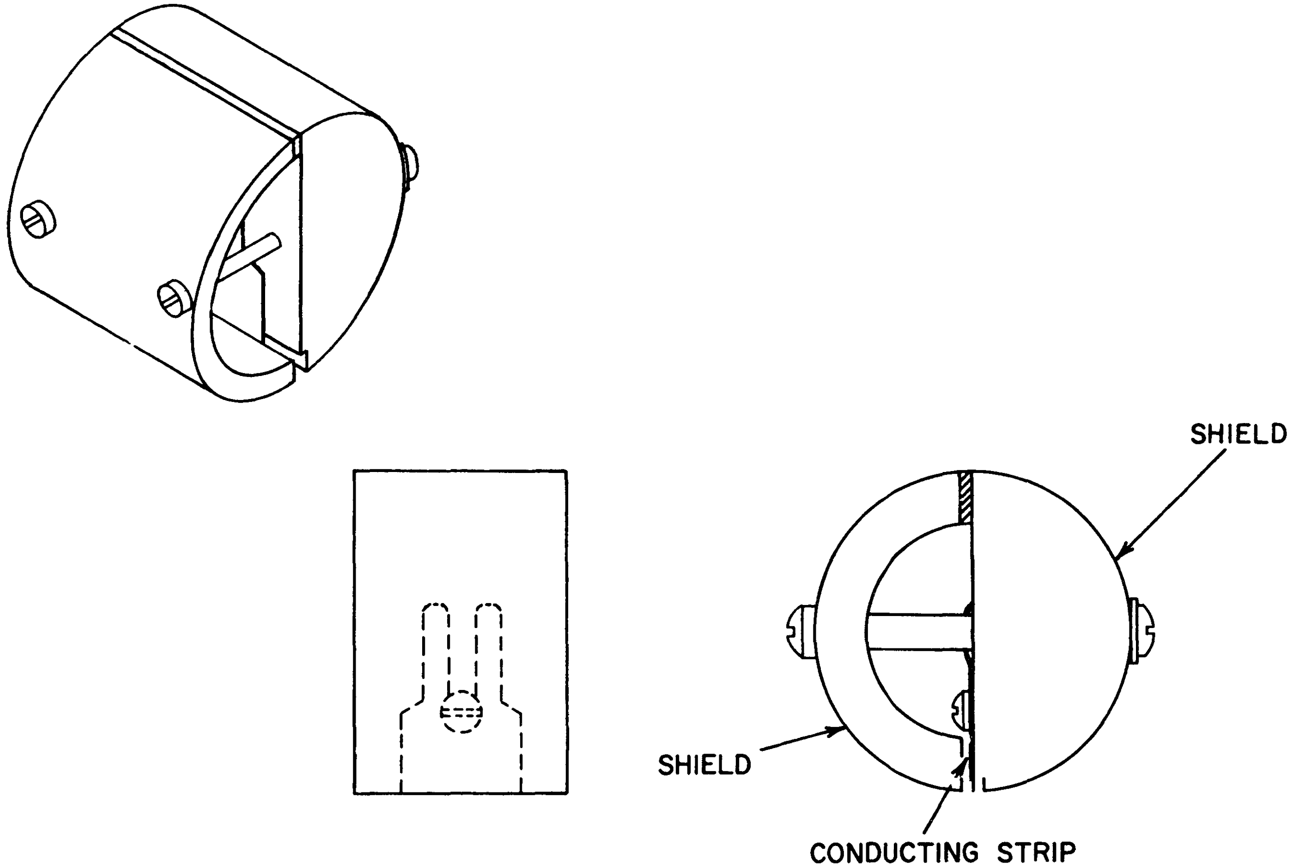


Figure 7. Electrostatic Probe

audio reproduction. The image patterns varied in intensity from field strengths at the surface of the material near 600 volts to strengths of less than 100 volts. The methods of measuring introduced some error in these figures, but no difficulty was encountered due to poor image definition caused by the photoconductive material. Considerable difficulty was encountered from loss of image due to surface conductivity on occasions of high atmospheric humidity.

The electrostatic detection has presented a considerable problem. The basic limitation of the electrostatic detection probes plus the problems created by the micro-dimensions necessary for separating and shielding have yielded less than satisfactory results. It is anticipated that improvements in method of detection and specifically the incorporation of a push-pull or positive-negative system of reproduction will overcome some of these problems. It is important that the signal-to-noise ratio of the system be increased. It is also highly desirable to increase the resolving capability of the detector to permit the use of smaller photoconductive plates. This performance has led to the serious consideration of the use of a vidicon tube rather than a moving plate for storage.

The optics of the system, while relatively crude, have proven entirely adequate for the purposes of the experiment. It is not anticipated that any difficulties would be encountered in developing an optical system for a prototype converter.

Audio Records

Both the density and variable area records have been reproduced successfully. Because of the exposure characteristic of the photoconductive material, variable area recording has been selected. This characteristic is represented in Figure 10. With variable density, it would be necessary to operate on the linear region of this curve. With variable

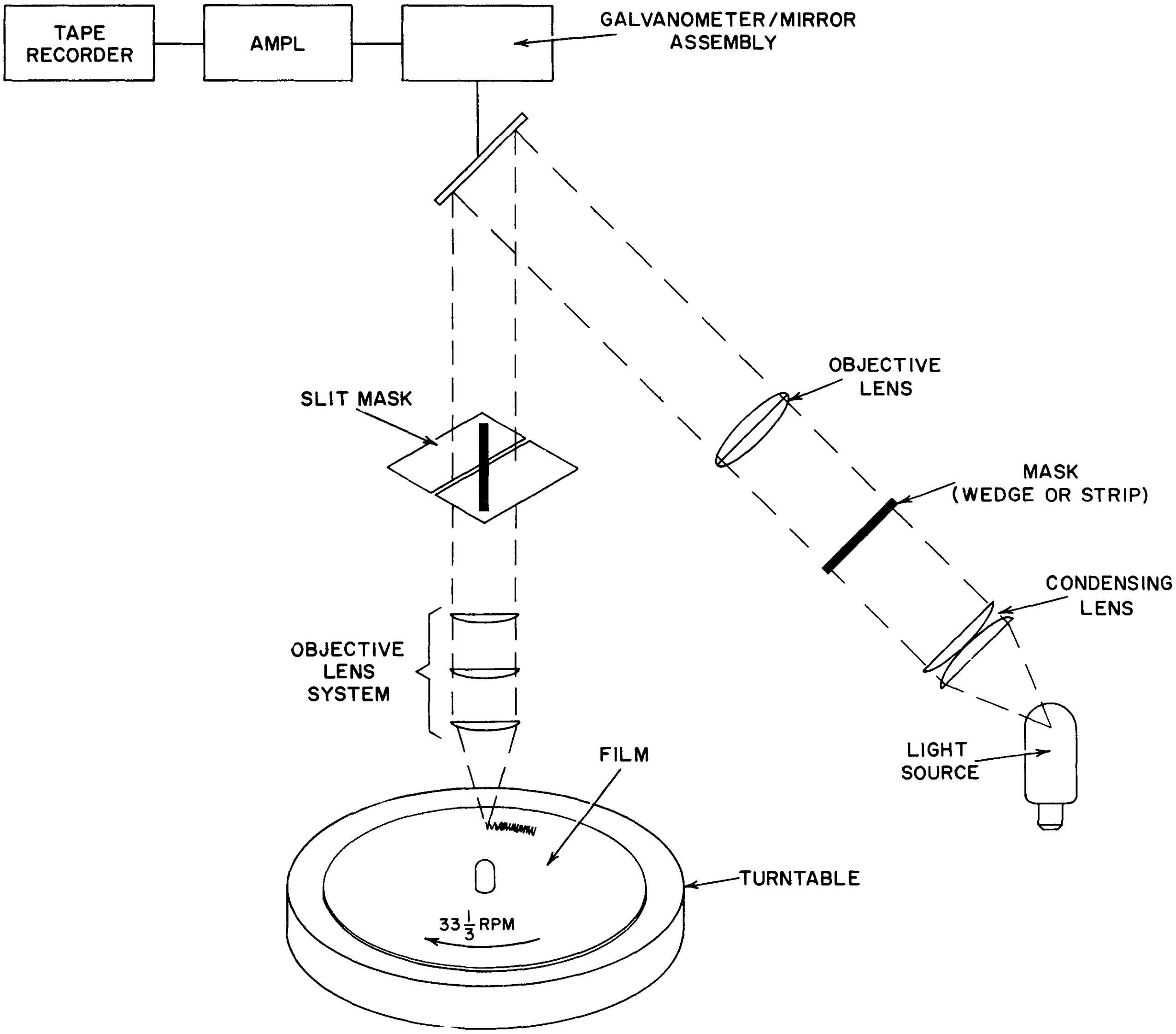


Figure 8. Recording Apparatus Schematic

area, it is possible to operate in saturation. This broader operating region has led to the better performance and stability of variable area records and is the reason for their selection.

Figure 11 illustrates the variable area recording of the word, "Motorola." A single variable area track (outer) and push-pull dual variable area track (inner) are shown. At this writing, no conclusive data has been gathered to show advantages of dual vs single tracks.

This effort continues with further laboratory tests and experiments. Currently, signal-to-noise ratio is a problem of concern with the original apparatus. In an effort to improve this, a positive-negative sound track is being used as a source for differential audio signals. It is expected that the ability of the differential system to reject common mode noise and the increase in 6 db of signal strength should improve the signal-to-noise ratio by a factor of 20 db.

APPLICATIONS UNDER CONSIDERATION

The work to date has produced results sufficiently satisfactory to permit the consideration of various applications of digital-to-voice conversion. Several such applications have been considered in detail. These are principally associated with systems where

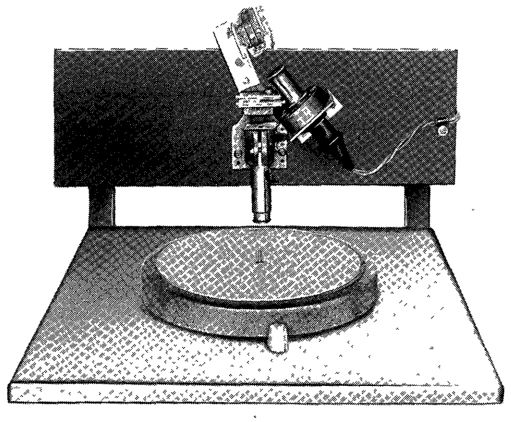
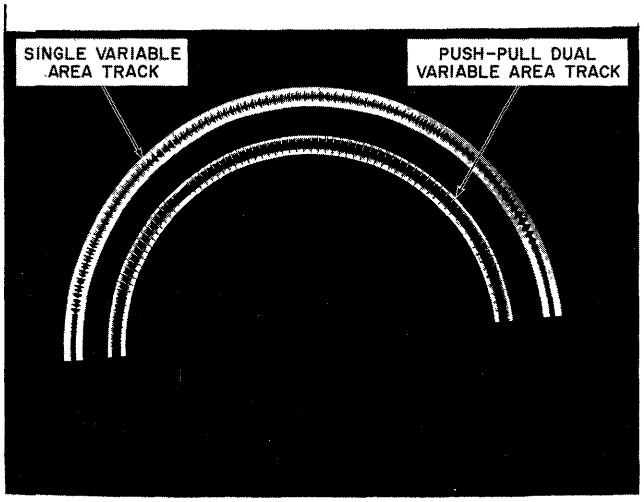


Figure 9. Recording Apparatus



SINGLE VARIABLE AREA TRACK
PUSH-PULL DUAL VARIABLE AREA TRACK

Figure 11. Sound Tracks

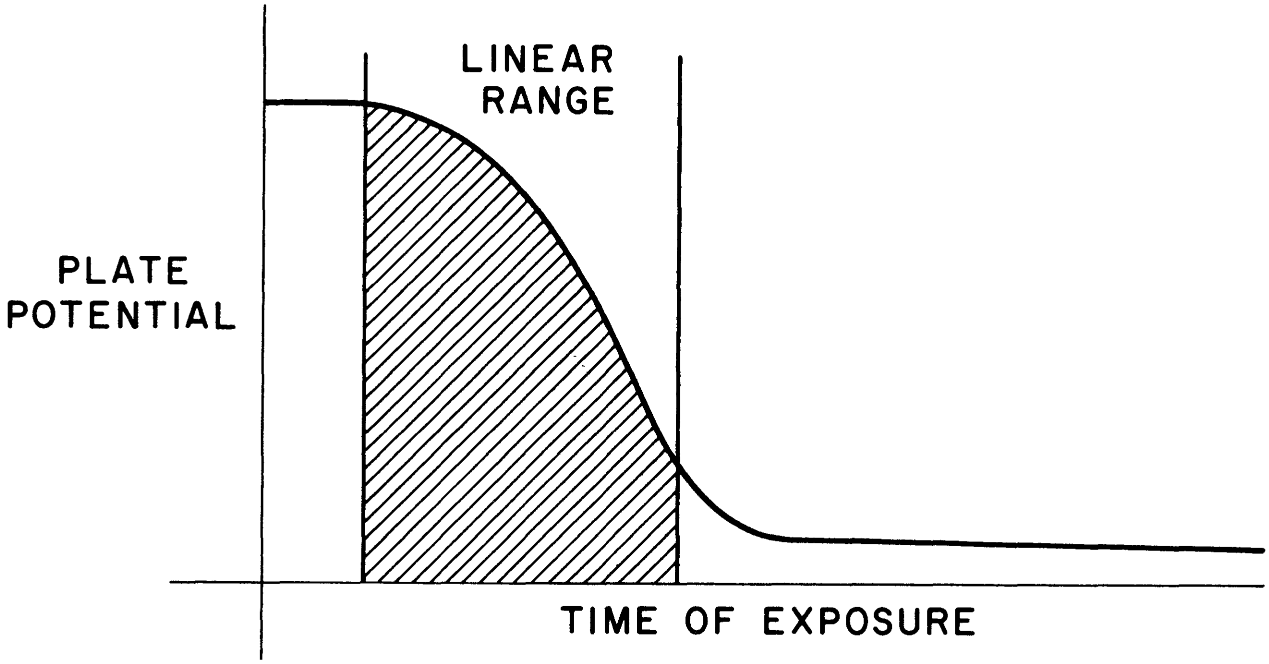


Figure 10. Photoconductive Material Characteristics

large data processors are used in the control or in the information organization for manual systems of operation. For example, in the air traffic control system there are three clear instances where voice conversion from digital information is desirable. These are: (1) flight plan production, (2) clearance information messages and (3) weather data transmissions from flight service stations. In the now developing weather system for the Air Force and FAA there is another good opportunity for application of digital-to-voice conversion. In this system, it is anticipated that approximately 90% of information dissemination will be over the telephone to callers who have dialed in to request up-to-date weather information. The machine will have this information available in its stores and can, using the voice converter, communicate directly to the caller over the telephone circuit. The reduction in requirements for operator handling of calls as a result of the application of the digital-to-voice converter is dramatic in this instance.

These applications, as it happens, are all concerned with government or military systems. This does not mean, however, that there are not in the future numerous commercial applications for voice converters. For example, banks, traffic control, automatic inventory control and other systems of data processing are currently expanding in commercial application. All offer the basic characteristics necessary to create need for digital-to-voice.

CONCLUSION

The digital-to-voice converter is a new and powerful tool to add to our work shop of input and output devices. It not only provides for some particularly unique operations where other methods of machine output are not well suited, but can be expected to provide an important back up to existing output methods. For example, printed information could be sent along for purposes of record while the vocal information was transmitted at the same time for its transient and immediately useful importance.

Conversion by the techniques discussed in this paper are definitely feasible. There is no reason to assume that voice messages cannot be rapidly composed from photographically stored data. All of the tests indicate at least a satisfactory level of performance.

The techniques tested promise to ultimately result in compact and inexpensive equipment. This is a characteristic requirement of any widely used digital output device.

Further development is needed before a compact, inexpensive converter is available. This development should investigate and explore some of the alternate methods of conversion mentioned, but not yet studied. In addition, this development should be devoted to the creation of a vocabulary for communication and to the further improvement of the techniques that have been already established as feasible.

This continuing program is under way in the Motorola laboratories. It is anticipated that the solutions for the major problems will be developed within the next few years. For this reason attention is now being directed toward the application of digital-to-voice converters. Perhaps within the next few years the application of these converters will be commonplace.

BIBLIOGRAPHY

R. G. Breckenridge, B. R. Russell, E. E. Halm; Photoconductivity Conference; John Wiley and Sons, New York, 1954.

H. W. Katz, Solid-State Magnetic and Dielectric Devices; John Wiley and Sons, New York, 1959.

R. M. Schalfert and C. D. Oughton, Xerography—A New Principle of Photography and Graphic Reproduction; Journal of the Optical Society of America, Vol. 38, No. 12, Dec. 1948, p. 991-998.

Gregg, Leslie, Zoubek; Gregg Shorthand Manual; Gregg Publishing Manual, 1949; Ref: Sounds, Syllables, Symbology.

George A. Miller; Language and Communication; Ref: phonetics, statistical approaches, words, sets, and thoughts.

Julia E. Johnson; Basic English; Ref: Examples, Pro and Con, of the basic English vocabulary.

Anthony G. Oettinger; Automatic Language Translation; Ref: Processes of initial selection, vocabulary, outline of the fundamental, lexical, and technical problems of translation.

Godfrey Dewey; Relative Frequency of English Speech Sounds; Ref: Statistics on relative frequency of words, syllables, and sound.

Leonard Bloomfield; Language; Ref: Phonetic structures, morphology.

Charles C. Fries, American English Grammar; Ref: Grammar phenomena, inflection and substantives.

CARD RANDOM ACCESS MEMORY (CRAM): FUNCTIONS & USE

Leon Bloom and Isador Pardo
National Cash Register Company
Electronics Division
Hawthorne, California

William Keating and Earl Mayne
National Cash Register Company
Data Processing Systems & Sales
Dayton, Ohio

The National Cash Register Company's Card Random Access Memory (CRAM) is a unique device which provides for the first time a single practical unit for both random and sequential processing. This flexibility, when applied to the actual requirements of data processing installations, permits the use of powerful techniques resulting in highly efficient operations. In order to understand how this is achieved it will be necessary to have a knowledge of the basic CRAM mechanism and of the relationship of the CRAM to the NCR C-315 Data Processing System (Section A). Use of the CRAM in general application areas will then be discussed (Section B), and finally a brief description will be given of the programming packages developed for CRAM (Section C).

A. CRAM AND THE 315

1. The Cram Mechanism

The CRAM is illustrated in Figure 1 and an interior schematic view is shown in Figure 2. The basic data storage medium of the CRAM is a 0.005-inch, oxide-coated, Mylar Card, 14" long by 3-1/4" wide (Figure 3). Each card has a set of binary coded notches at one end, which permits the automatic selection, at random, of any one of 256 cards from the CRAM magazine. When loaded into a CRAM unit the cards hang from 8 rods which may be turned in such a way as to cause one, and only one, of the cards to be released. Two gating rods at the side of the cards insure proper selection and release.

When the card is released it is allowed to fall freely until it reaches a rotating drum to which it is pulled by means of a vacuum, and the card is rapidly accelerated to the surface speed of the drum, 400 inches per second. Shortly after attaining this speed, the leading edge of the card reaches the read-write heads.

While passing the heads, a portion of the card is pulled away from the drum by a combination of removing the vacuum from the drum and creating a vacuum at the head, insuring good head contact. Once each portion of the card has passed the heads it is brought back into contact with the drum. The write head is encountered first, with the read head 1/2" behind. The read head, besides its normal function of reading, is also used to perform an immediate check after writing.

After reading or writing the card may remain on the drum, to be recirculated past the heads on the next revolution, or it may be released and returned to the magazine. The release mechanism consists of a simple gate which in one position does not affect the card but allows it to remain on the drum; in the other position it peels the card from the drum and directs it back to the magazine.

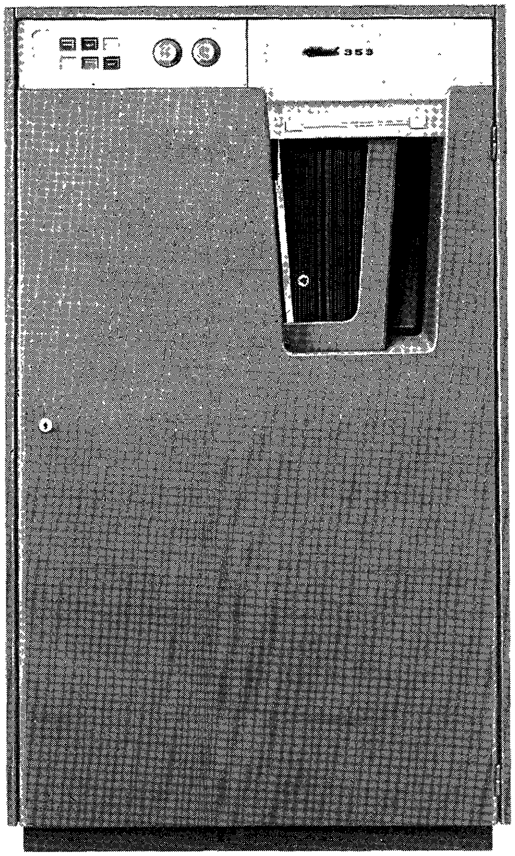


Figure 1

The return to the magazine is accomplished by allowing centrifugal force to provide the momentum which sends the card through the raceway and into the magazine. Then the loader plate pushes the card back onto the rods where it is once again available for selection. The sequence of the cards on the rods is, of course, inconsequential, since the selection of a card is independent of its position in the magazine.

2. Data, Format, and Timing

There are 56 channels across each card divided into 7 data tracks of 8 bit channels each. A single track, which may be read or written each time a card passes the heads, consists of 6 information bits, a clock bit and an odd parity bit. Information is recorded at a density of 250 bits per linear inch which, when coupled with the card rate of 400" per second, gives a transfer rate of 100,000 six-bit characters per second. Blocks, in lengths of up to 3100 characters, may be written in each of the seven tracks, with the block length in one channel having no bearing on the block lengths of other channels. Each block has a longitudinal parity character recorded following the last data characters. Reading or recording of blocks must start at the beginning of the track. Electronic switching allows random selection of any of the seven tracks.

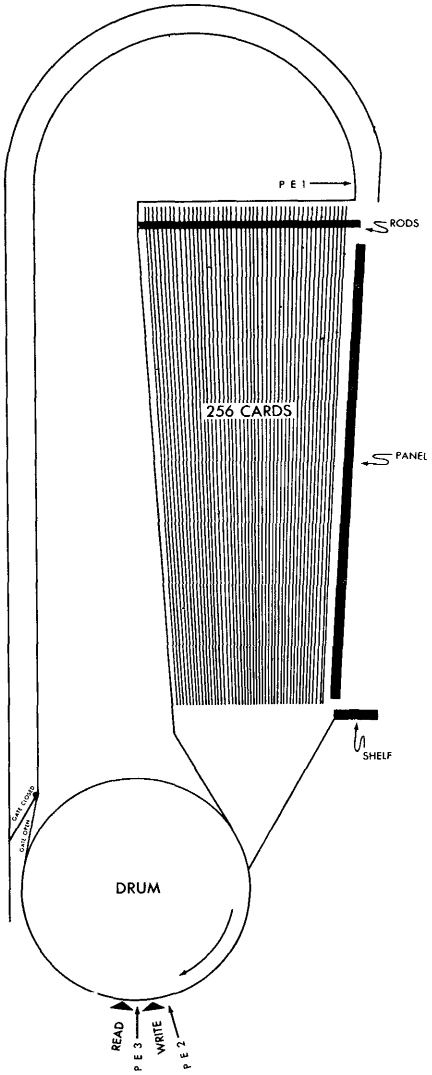
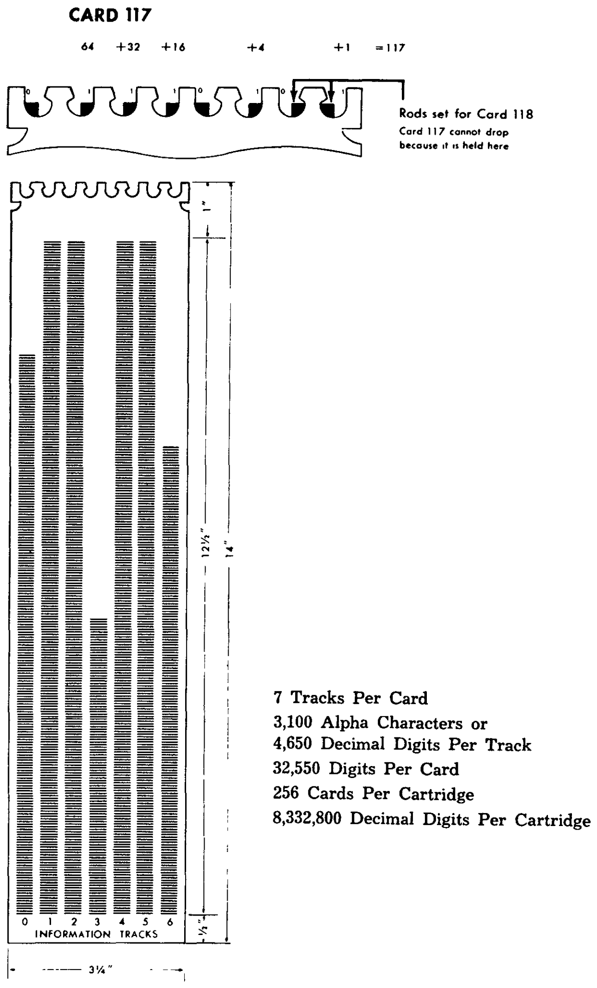


Figure 2

From the time a card is selected until it reaches the heads is approximately 200 ms. The rotation time of the drum is 46 ms. Since



7 Tracks Per Card
3,100 Alpha Characters or
4,650 Decimal Digits Per Track
32,550 Digits Per Card
256 Cards Per Cartridge
8,332,800 Decimal Digits Per Cartridge

Figure 3

each card contains 7 times 3100 or 21,700 characters, it follows that once a card is on the drum, this much information is available to the processor with an average access time of 23 ms. As there are 21,700 characters per card and 256 cards in a magazine the total capacity of each CRAM unit is 5,555,200 alphanumeric characters or 8,332,500 numeric digits or any proportional combination. Extending this number by the 16 CRAM units which may be tied to a C-315 at one time, each system can have available to it more than 88 million alphanumeric characters of information with a maximum access time to any of it of 200 ms.

In addition to this capacity for on-line information there is a fundamental difference between CRAM and contempory random access devices. That is, the ability to change magazines in no more time than it takes to change a reel of tape.

While a CRAM is in use, the cards are kept separated from each other, and spread evenly along the rods, by air jets. When the hinged faceplate and loader plate are swung aside, the jets are automatically turned off and the deck of cards hanging on the rods may be compressed by hand.

A cartridge, consisting of three sides and a bottom, is then hung on a horizontal guide bar, fastened to the front of the CRAM cabinet. The cartridge is swung onto the magazine, so that the cards hang inside it. One side of the cartridge is movable and spring-loaded; when released, it further compresses the deck, with sufficient force so that the cards are held securely at any attitude. The cartridge is then slid to the right, along the guide bar, until the cards are off the rods, and the cartridge is remov from the guide bar. For loading the cards into the CRAM magazine, the procedure is the exact inverse.

A dust-proof cover is provided, for shelf storage of the cartridges.

With this extremely fast and simple process, it is practical to maintain very large files with a minimum of CRAM units.

3. Control

Three photocells provide the prime source of control of the CRAM mechanism. The first photocell, PE 1, is located at the entrance to the loader mechanism to detect the leading and trailing edge of the card. Upon detection of the leading edge, the loader plate opens to receive the card. Upon detection of the trailing edge, the loader plate, with suitable delay, is actuated to return the card to the magazine.

PE 2 precedes the write head by 0.8 inches, and PE 3 is halfway between the write and read heads. The leading edge of the card, detected at PE 2, indicates completion of the "dropping" phase of card selection, and also transmits the demand-interrupt signal to the processor. When the card reaches PE 3, this terminates the "Ready" state of the CRAM, as it is now too late to initiate a read or a write at the beginning of a track. Sequence interlocks at PE 2 and 3 also control operation of the release gate, to prevent any ambiguity as to which card is being released.

4. Relation to the C-315

The photocells in the CRAM unit are combined with the interrupt and branching features of the C-315 to provide maximum utilization of both processor and CRAM time. First, the instruction which causes one of up to 16 CRAM units to be selected and one of the 256 cards on that unit to be released, is separated from the instructions which initiate reading and writing on one of the seven channels of the selected card. This separation allows the processor to cause a card to be dropped and then, during the 200 ms. that it takes for the card to reach the area of the read-write heads, continue with other program steps among which may be instructions to select cards on other CRAM units.

When the leading edge of a selected card gets within about 2 ms. of the write head it passes photocell #2 and may cause the processor program to be interrupted by Demand signal from the CRAM. Whether or not the interruption actually takes place is controlled by the unit Demand flag provided in each CRAM unit and by the Demand permit flag located in the processor. Both these flags, which are program controlled, must have been set in order for the program to be interrupted. When the program branches due to the interruption, the address of the next command in the program which would have been executed had the interruption not occurred, is stored in a link register. Here it is easily accessible for use in returning to the main routine after the interruption routine has been executed.

It is of course possible that the processor program, as organized by the programmer, will not respond to the demand in time to read or write at the beginning of a card track. In such cases, PE 3 will cause an indicator to be set so that, if a read or write instruction should be issued, the data transfer will not take place until the card is again in position to read or write.

By using these interrupt features several CRAMs may be used simultaneously without programmer confusion, and with maximum time sharing.

Additional control in the CRAM unit allows the next card to be selected while the present card is still on the drum. This instruction causes a card to start dropping, but allows one more read or write on the present card, before it is automatically returned to the magazine. Safeguards are provided to insure that the read or write command will actually be performed on the present card.

An alternative mode of the selection command, on the other hand, locks out all reading and writing until the next card has reached the drum.

Selection of another card is inhibited while a previous card is still dropping to the drum. In such case the CRAM unit is considered to be "busy" and the processor will take a "busy" branch.

One interesting feature results from the fact that the card occupies only about 2/3 of the drum circumference. This allows 15 ms. for computation, between reading a track and recording the updated version of that track. In 15 ms., the processor can execute 250-300 instructions. Thus each read, update, and write often requires only two drum-revolutions, including computation time, and still leaves 15 ms. for other processing before accessing this card again.

This description of the relationship between the CRAM and the C-315 would not be complete if it did not include several of the other instructions, safeguards, and checks available to the system. For example, a card which is on the drum may be released by instructions, or if this CRAM unit is not accessed for 750 ms. the card will automatically be returned to the magazine. A read head follows the write head so that odd parity on each character and longitudinal parity on each track may be checked while reading, or immediately after writing. An error discovered during either of these operations will cause the processor to branch so that proper action may be taken by the CRAM executive program.

All control areas which require hardware control have been adequately covered in the 315-CRAM system. These will work with the PAckaged CRAM Executive (see Section C) to insure correct and efficient utilization of CRAM.

B. THE APPLICATION OF CRAM

Specifications alone cannot adequately portray the facts about any computer configuration. This is especially true if such a configuration involves new concepts. This section will explain how the 315-CRAM System copes with widely dissimilar applications. Instead of singling out any particular application (although specific instances will be cited) the capabilities of CRAM will be discussed in two generalized areas: 1) file updating and 2) sorting. For clarity, analogies will be drawn with tape systems and current random access systems so as to underscore certain efficiencies, economies, and flexibilities of the 315-CRAM System. All performance figures will be compared with a 315-tape system.

The prime factor in determining final performance of data processing systems is the organization and basic efficiency of the secondary storage devices. Usually the type of device chosen for secondary storage (tapes, disc, drum, or cards) determines the applications areas for which it is best suited. Information storage requirements may change completely from one application (or one part of an application) to another. The information storage flexibility of the CRAM and its attendant efficiency are its greatest assets. A single CRAM unit combines the abilities of contemporary random access devices and magnetic tape handlers. The flexibility of CRAM makes it possible to have a single type of secondary storage peripheral which economically satisfies all data processing requirements. To exemplify CRAM's flexibility, its use for file updating will be outlined.

1. File Updating

The general file updating process considered here will be that which is standard for typical business data processing. A Master Historical File must be posted and updated for each transaction which has been input into the system. Three basic techniques of updating may be used efficiently in 315 installations, vis., Random, Serial Selective and Serial Copy. Most data processing problems, however, do not require clear cut versions of any of these. The CRAM permits the use of techniques which skillfully blend these methods into an optimally efficient system.

a. Random Access (Figure 4)

The CRAM provides the 315 with an exceptional random access memory. Maximum access time of 200 milliseconds, a reaccess time of 15 ms, the ability to store 5,555,200 characters on each CRAM unit, and the 315's potential of controlling 16 such units each with independent access, all attest to this statement. The ability to change CRAM memory cartridges in approximately 30 seconds make CRAM unique for a device of its capabilities. This makes it possible to segment storage allocations in such a manner that information pertinent to separate random access jobs can be loaded as required.

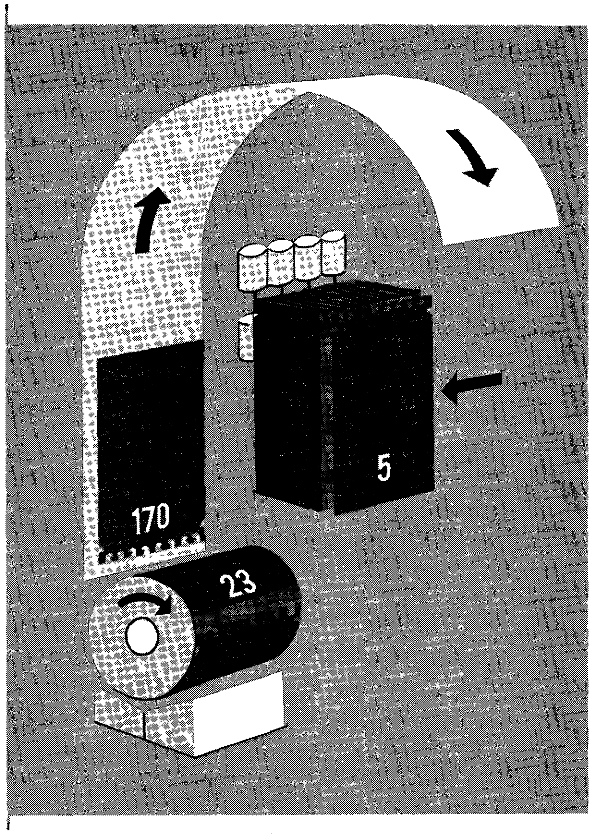


Figure 4

In order for other known random access systems to accomplish the same function, information must be unloaded to another storage medium, e.g., tapes or cards. Since this technique unnecessarily burdens the system the alternative approach is usually taken of providing enough storage capacity for all jobs which will use the memory. The prevailing concept of "immobile information" therefore turns out to be an expensive restriction which is obviated by the CRAM file replaceability concept.

It should be noted that this concept not only provides for efficient utilization of secondary storage capacity, but also provides for back-up should one of the units become inoperative. The cartridge on that unit can easily be removed and mounted on a spare unit, thus making the data once again available to the system with a minimum of down time.

For random file processing, commonly used addressing methods are employed to select a CRAM unit, drop a card and then read a track. These methods include the use of directories, direct addressing, or calculated addresses. When the appropriate track is read it is scanned by the program for the proper logical record. Storage overflow methods are incorporated if the record being sought or written could not be stored on the selected track. Methods of address selection and storage overflow control are conventional and will not be described here.

The appropriate track of the selected card is available for reading 200 ms. after the card has been called. This track is read in 31 ms. As was explained earlier, the card remains rotating on the drum and 15 ms. elapse before the card can again be read or written upon. During this 15 ms. the record in the track can be found by program and in most cases updated (the 315 is a fast computer with an average command execution time of 50 u sec). The card can then be written on in 31 ms. during its next revolution. If for any reason the updating takes over 15 ms., the recording of the updated record can be performed on subsequent revolutions since the card remains on the drum effectively until released or until another card is dropped. This procedure applies as well to the case where a second track of the same card must be read as in the case of storage overflow. During the writing of the updated track, a second card may be accessed and dropped on the same unit. When cards are being dropped simultaneously on separate units, the program is advised by interrupt when a card is ready to be read or written.

The net time to randomly update a file stored on one CRAM is approximately 4 transactions per second. When the file is stored on multiple CRAMs and cards dropped simultaneously, this speed approaches 13 transactions every second.

b. Serial Selective Updating (Figure 5)

The serial selective file updating technique is actually a modification of the random updating technique. As in random processing, track directories, as well as storage overflow methods, are employed to facilitate record addressing. In this method, the entire Master File is sequenced and stored over one or several CRAM cartridges. Transactions are sorted into Master File sequence and posting to active cards and active tracks takes place selectively, i.e., those cards and tracks which are not affected by transactions are left undisturbed; those cards and tracks which are affected are read, updated in memory, and recorded over the old information.

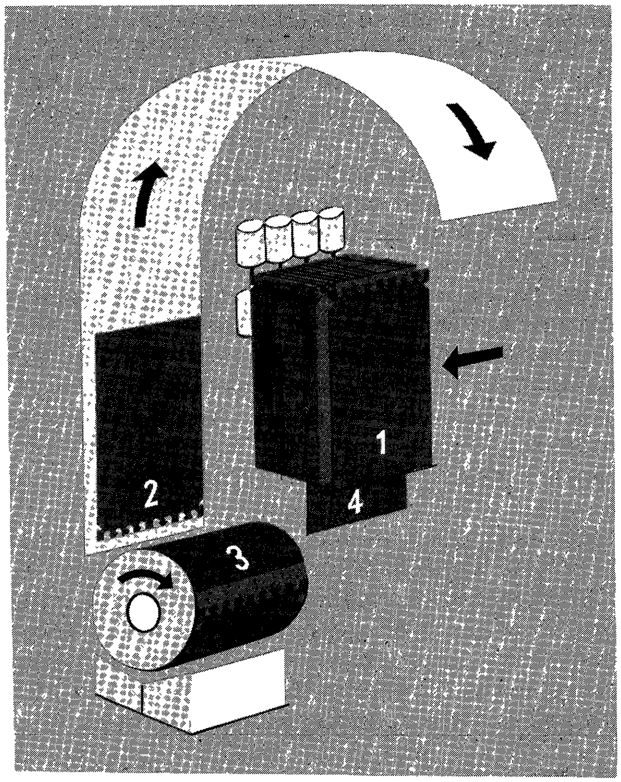


Figure 5

This method, while requiring batch processing, has two definite advantages over random processes. The number of cards which must be dropped and the number of tracks which must be read and written is never more then the number of cards and tracks which must be updated. Secondly, the entire Master File need not be available to the computer at one time. As the file is being processed serially, a single CRAM unit (or two if alternation is desired) is sufficient regardless of the magnitude of the file. The first cartridge of the file is mounted, updated, and replaced by a second and so on through the file.

For low activity files, this method of updating provides the most efficient utilization of CRAM. For instance, if the average number of active records is one per card, the 315 to match this performance would require tapes which operate effectively at 236,000 characters per second with no start and stop time. Even for a moderately high activity averaging one transaction per track (typically a track contains 5 to 15 logical records), the effective 315 tape rate would have to be 77,000 characters per second. With current stop start times and assuming blocking similar to CRAM, 315 tapes (which are generally similar to other tape systems) would need to operate at over 100 kc to achieve such an effective rate. The net updating time per transaction (one to a track) using this serial selective CRAM method of updating and employing a single CRAM is 12 transactions per second.

In practice it is often found that the very low activity files are by their nature quite extensive. This leads to the problem of on one hand requiring a random access devised to achieve fast processing and on the other hand requiring mass storage devices such as tapes to inexpensively store the files. However, random access devices are uneconomical in storing data and tapes are inefficient in processing low activity files.

The serial selecting updating method in conjunction with the CRAM's fast access and interchangeable file concept provides an efficient solution to the problem.

c. Serial Copy Updating

The CRAM unit can also be used exactly like an ordinary magnetic tape handler. Each cartridge can be considered a tape reel storing 1792 blocks (tracks of cards) each of which stores up to 3100 alphanumeric characters or 4650 numeric digits. All CRAM tracks are then considered sequentially in this method. Master Files can be organized for conventional Father-Son (Copy) updating. In this method the file is sequenced, as are all transactions which must be posted against the file. The old (Yesterday's) Master File is read, all active records updated and a completely current New Master File written. In such a method of file updating, all tracks of all cards on the Old Master File must be read while all tracks (as updated) of all cards are written on the New Master File. Usually this method employs two CRAMs (an old and new) or four if cartridge change time is to be shared. With such a method, those files which lend themselves best to Magnetic Tape File Processing can be handled at least as efficiently with CRAMs. All the features of retaining yesterday's files for emergencies, the handling of high activity files where records are being added, deleted, and changed in size are retained with CRAM, identically as with tapes.

The efficiency of such a system is outstanding. Using but one read in and one write-out memory area, the time to copy a single CRAM card is approximately 1 second. An effective transfer rate of approximately 42,000 characters per second is achieved. This would require a 50 kc tape unit to duplicate. Using a two buffer input and a two buffer output area, most drop access and drum access times can be eliminated. While processing information in buffer 2, the interrupt is set to advise the read and write macros when the appropriate card is in position to read or write buffer 1. No time is lost in accessing the card and an effective rate of up to 100,000 characters per second can be achieved.

One particular use of this method is in the Demand Deposit Accounting application for banks. The characteristics of a checking account file are high activity, and constantly changing record sizes. These requirements are satisfied by such a Father-Son updating technique with CRAM.

Added Considerations of CRAM file Processing

i. Control

Although the random or Serial Selective method may be used for a given application, the need for periodic sequential processing is ever present. Such activities as insertion of new records, deletion of old, creation of rescue files or file dumps all require sequential file access. These activities on conventional random access devices are expensive, time consuming, and often require that additional non-homogeneous equipment such as tape units be included in the system. The CRAM, on the other hand, cannot only execute these periodic operations efficiently, but can actually create sequential expanding files, such as transaction files, while doing random processing. This ability plus the discreet nature of the CRAM file permitting direct access to the affected portion of the file greatly facilitates reconstruction and the maintenance of audit trails.

Such blending of several techniques but always using the same peripheral CRAM provides high efficiency with safety without increasing the peripheral requirements of an installation.

ii. Absence of Rewind

Whenever the CRAM is being used in a magnetic tape fashion there is one major feature of CRAM which cannot be overlooked. Rewind is completely eliminated. Time between jobs is appreciably reduced. On completion of a run, CRAM can be changed immediately whereas tapes are required to be rewound. Many applications require multitape reel or multi-cartridge files. In tape systems the lack of an alternate handler involves the loss of both reel rewind as well as reel changing times. Since only cartridge change time is involved in CRAM systems, the requirement for alternate CRAM units is less severe.

iii. Multi File Cartridge

Not only do multiple CRAM cartridges make up a single file but each cartridge can also be used to store multiple small files.

For instance, cards 2-70 can store the Program File, cards 71-130 a Transaction File, 131-200 an Output File, and 201-255 a temporary working file (Figure 6). Searching and in most cases recording restrictions on magnetic tape make such multi file tape reels very difficult and cumbersome to work with at best. The importance of being able to organize file data in this manner is readily apparent. Applications using such small files usually require more time between runs to accomplish tape set ups than they do for processing. Much of this set up time can be eliminated by properly pooling multi files on one or two CRAM cartridges.

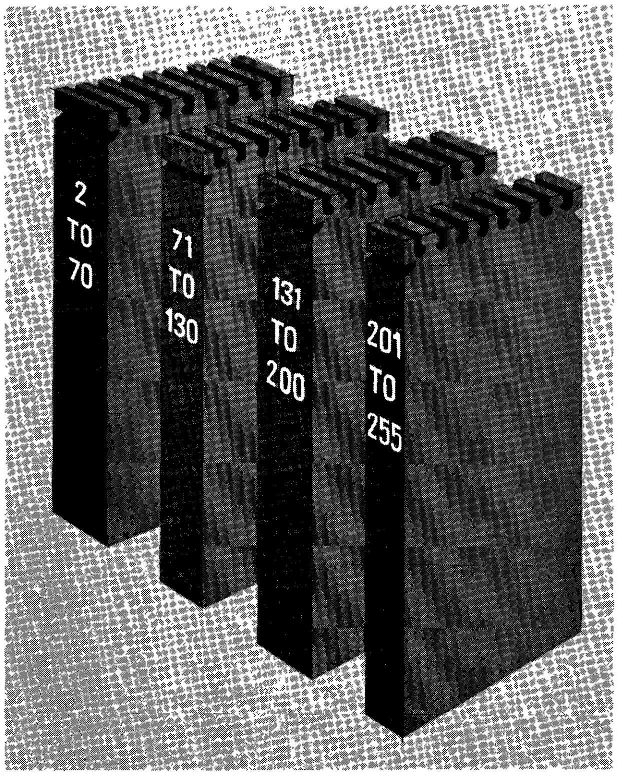


Figure 6

In many applications several tape processing runs may be combined into a single CRAM run because of the availability of all necessary files at one time. With CRAM such consolidation can become natural with proper information organization.

Fewer CRAM units are required to do a particular job. A typical example is NCR's own COBOL translator which requires a minimum of six tape handlers. Using the exact same system organization but allowing each CRAM to be segmented into multi files, COBOL will require but two CRAM's for translation (with a speed increase over our 315 tape system). With some loss of efficiency, COBOL will later be coded to run on one CRAM.

d. Summary

For file updating the 315 CRAM System offers a versatility of file organization. Any one of three normally acceptable file methods can be employed with high efficiency and all on the same unit. Numerous advantages are possible with CRAM over older peripherals, all of which were designed with but a single method of file maintenance in mind.

2. Sorting

Besides file updating another major data processing operation is sorting. Sorting capabilities are, and should be a major test of equipment. Even on a randomly posted file, the sorting of output and of various other auxiliary files is of prime importance. The sorting capabilities of the CRAM are exceptional. Methods outlined here are straightforward. The basic concept is to divide the CRAM or CRAMs into four logical files. Sorting is then accomplished exactly as if a four tape sort were being performed. Much of the logic of the 315-tape sort generator is carried over directly to a 315-CRAM sort generator which will produce programs using 1, 2, or 4 CRAMs. The following table indicates the effectiveness of CRAMs versus tapes.

Time required to sort 30,000 records on a 20,000 character 315 with various peripherals:

	40 Character Records	80 Character Records
4 Tape Units (40 kc)	17.0 min.	29.1 min.
1 CRAM	22.7 min.	40.3 min.
2 CRAM	12.2 min.	19.5 min.
4 CRAM	10.1 min.	14.2 min.

Note that while a single CRAM is somewhat slower than a tape sort, 2 CRAMs are more effective and a 4 CRAM sort is even more effective. In the method employed, internal sort times are identical for tape or CRAM. Tape times, however, are decreased by 43% and 66% respectively for 2 or 4 CRAM sorts.

Such savings are achieved by the CRAM's capabilities of sharing card drop and transferring information at very high rates. In addition, the CRAM is not encumbered with start-stop and rewind times as are tapes.

As well as employing straightforward techniques in the CRAM sort generator, more advanced methods can be devised to make full use of the CRAM's capabilities. One of these techniques takes advantage of the CRAM's ability to handle each track as a separate logical tape. By using a combination of block sorting techniques and then conventional merging techniques, the number of passes through the data to be sorted is reduced. This, of course, reduces the total sorting time from those figures mentioned above.

For the basic needs of data processing applications, the unique features of the CRAM proved exceptional in all studies. Not only is basic efficiency, with flexibility of approach, inherent in CRAM but the wide range of small CRAM-315 to large CRAM-315 systems allows National to enter a wide scope of application areas with one computer. Systems employing as few as one CRAM and as many as sixteen are valid and efficient (depending on the job). Increased capabilities are readily achieved as CRAM units and memory modules are added to a smaller CRAM-315 System.

The hardware capabilities of the CRAM have been outlined as have many of its general uses. However, the CRAM picture would not be complete without also describing the software programs provided with the system.

C. PROGRAMMING WITH CRAM

All NCR tape systems (304 tape, 315 tape) have from early conception been designed to work with supervisory routines. These routines, termed STEP (Standard Tape Executive Program) have handled all tape contingencies, end of tape procedures, labeling, and job to job executive functions. These programs have been used very successfully in the field. A similar CRAM control program termed PACE (PAckaged Cram Executive) has been coded and will be used by all programs which intend to manipulate CRAM. This is a supervisory routine which will be stored in a fixed memory location at all times. All CRAM reads, writes, drops or other more advanced input-output functions will be accomplished by macros, which are coded to work with and facilitate PACE control. PACE is designed to simplify all CRAM programming, to provide all necessary system checks and control, and to retain intact the entire flexibility of CRAM. This executive supervisory program has three functions: 1) CRAM processing control, 2) File label checking and semi-automatic file storage allocation, and 3) Program to program executive functions.

1. CRAM Processing Control

CRAM uses a magnetic reading and recording technique very similar to magnetic tape and therefore, like tape, requires certain error controls. These include making multiple attempts to read or write when temporary dropouts occur or bypassing the flaw should the dropouts prove permanent.

Unlike tapes, the CRAM is subject to other operator and programmer contingencies which require close supervision and control. Among these are the possibility that an operator may erroneously change a CRAM cartridge or that a program might attempt to read or write on a card when no card or the wrong card was on the CRAM drum.

Throughout PACE the philosophy has been to handle as much error control as is reasonably possible with a high degree of automaticity. The programmer merely codes his appropriate Drop, Read, Write or other macro instructions. During assembly time automatic linking to and from PACE is set up to enable PACE to completely supervise and control these instructions during object program running time. This insures that almost all possible CRAM difficulties that could be caused by a careless operator or programmer are handled properly.

To implement PACE every new CRAM cartridge introduced is first set up completely by a standard service routine. Set up occurs but once in the life of a cartridge. The first two cards first eight characters of every track of every card are reserved for PACE control. The first two cards are used to store file labels, PACE overlays, and to logically replace flawed tracks. The first eight characters of each track store card number and a cartridge serial number.

Control is exeucted in the following manner. Any read or write errors cause re-execution. A repeated write failure (probably a flawed area) causes this track to be saved on one of the first two CRAM cards reserved for the PACE system. Reading of such a track is controlled by PACE so that the programmer never need be aware of such occurrence.

All card drops are checked for proper card number. This insures proper selection both by CRAM and by the programmer. Any attempt to read or write on the wrong card or a non-dropped card causes the PACE system to reselect the proper card without disrupting the program. Any excessive difficulties are reported to the operator on the 315 console. All card drops are also checked for proper cartridge serial number, thus insuring against file mounting errors.

2. File Label Checking and Semi-Automatic File Storage Allocation

As with tape reels, each cartridge has a recorded file label for every file stored on the cartridge. These labels contain file name, cartridge sequence number, data recorded, obsolete date, and information as to what cards are used to store the file.

Before writing a new output file, the programmer can designate two modes of storage allocation: fixed or automatic. In the fixed mode, the actual cards upon which the file is to be written are named. All file labels on the cartridge are checked to verify that only obsolete cards will be recorded upon. A new file label is created for this file and processing started. In automatic mode, all file labels are scanned to determine a contiguous group of obsoleted cards. These cards are then allocated for that particular file. A label is created and processing commenced. On filling any particular area, PACE will scan for more available space on the same cartridge before executing an End of Cartridge procedure.

Before reading an input file, all file labels are checked for appropriate Name and Date. When found, the card numbers bounding this file (first and last) are stored for macro and programmer access. Processing continues thru the file to completion or till this cartridge is exhausted for this file. In the latter case, an End of Cartridge procedure is initiated.

The End of Cartridge procedure either alternates CRAM units or demands a change of cartridge, depending on the number of units allocated to this file. Cartridge sequence number is augmented, a memory dump executed if desired, and the file initiated, as outlined above, for the new cartridge.

3. Program to Program Executive Functions

Programs are set up and maintained on CRAM by the CRAM Librarian. Programs may be stored on a separate CRAM Program Cartridge or as a program-file on a CRAM Cartridge which is also being used for various data file functions. When a program is completed, PACE locates the next program to be executed (provided that it is stored on a cartridge currently mounted on a CRAM), loads the program, accomplishes all necessary dating functions, and initiates execution of this program. Overlay Macros will locate and call in any desired program overlays during the running of this program. Because of the random capabilities of CRAM, its use for overlay storage is efficient and economic.

4. Basic Features of the Packaged CRAM Executive

This executive program is a fixed program which is automatically included with all CRAM programs by the 315 Assembler or by COBOL. The basic control portions of the program are in memory in the same location at all times. Familiarity with PACE is achieved by programmers, operators, and servicemen. The basic program utilizes only 1800 characters. File Opening, label checking, end of cartridge logic and Run to Run Supervisor are all stored as an overlay on CRAM. No 315 memory is required for permanent storage of these programs. A portion of memory is saved on a reserved CRAM track, the overlay is read into this memory area, the required PACE functions are performed, and memory is restored. Throughout all PACE operations, required operator messages are typed out. These are kept brief but clear. Likewise, the ability to override certain functions is available though excessive use of these overrides is naturally discouraged.

5. CRAM Macros

Presently there are approximately ten CRAM macros. These include the basic read, write, and drop card macros. Macros are always used for these functions rather than machine instructions to insure proper PACE control. More sophisticated macros are also available. The NEXT IN and NEXT OUT macros perform logical record read and write. These make extensive use of the many 315 index registers for record advancing. They are designed to handle straight tape-like files (fixed or variable records) with single or multiple input and output areas. The multiple buffer, as mentioned earlier, results in much more efficient input and output. Another set of macros SEEK, ADVANCE, and RELEASE will be used for random and for serial selective methods of updating and are being expanded to handle several standard random access addressing schemes automatically as part of the macro. COBOL extensions have been made for the 315 NEAT COBAL to handle such methods of processing.

Besides COBOL, Neat Compiler, PACE, CRAM Sort Generators, and CRAM librarian, a set of CRAM utility programs for CRAM prints, copying, and manipulating of stored information will also be available.

The availability of a software package will simplify CRAM programming and enable one to realize all the flexibility and efficiency inherent in this new and unique piece of hardware.

D. CONCLUSION

The CRAM is a powerful and unique data processing tool. It makes an ideal bulk memory device for small, medium, or large data processing problems. Its versatility of information storage allows it to be used in a manner best suited for a wide scope of different applications. It has definite advantages for both minimal or maximum jobs.

It can be used as both a random and a serial access memory. Its power is coupled with the ability to change cartridges in no more time than is required to change a tape reel; no rewind of a CRAM cartridge is necessary and the price of a CRAM unit is comparable to that of a high performance tape unit. CRAM is here to stay and represents a definite advance in the critical area of peripherals. CRAM offers a general purpose file device for a general purpose computer.

THE LOGIC DESIGN OF THE FC-4100 DATA-PROCESSING SYSTEM

W. A. Helbig, A. Schwartz, C. S. Warren,
W. E. Woods, and H. S. Zieper
Radio Corporation of America
Data Systems Division
Van Nuys, California

SUMMARY

The FC-4100 is a parallel binary computer with a thirty-bit word length. It operates at a 1.0 mc clock rate and performs approximately 50,000 operations per second. The internal memory capacity is 4,096, 8,192, or 16,384 words of random-access core storage with an access time of 1.3 microseconds. The entire data processor, with core memory, contains less than 3,000 transistors. A special feature of the computer is a program interrupt system: up to sixteen independent programs can be executed on a pre-assigned priority basis. Each program is provided with its own set of six index registers and its own program counter. Other feastures include: a special repeat mode for controlling small, iterative loops without the usual branch instructions; relative addressing; half-word multiply and divide instructions for faster operation at reduced precision; and a normalize instruction to facilitate floating-point operations.

INTRODUCTION

Historically, efforts to reduce the cost of computing systems have resulted in a more-than-proportional reduction in performance. Furthermore, these efforts have led to the imposition of many programming constraints which reduce ability and increase operating costs.

The purpose of this paper is to describe a militarized data-processing system, the FC-4100, in which high-speed components, coupled with an advance in machine organization efficiency, have yielded high performance at low cost without sacrificing flexibility, versatility, or simplicity of programming.

DESIGN OBJECTIVES

The primary design objective of the FC-4100 was to build a "small" computer. The package size selected as target permitted no more than 3,000 transistors in the central computer. The secondary design objective was to retain, within the size constraint, the versatility and flexibility of large-scale data-processing systems. Important considerations in meeting these objectives were:

- Random access memory for ease of programming.
- Minimization of programming constraints and "housekeeping" problems.
- Unspecialized instruction repertoire.

- Efficient input-output system.
- Simplicity of maintenance and operation.
- Efficient utilization of logic elements thru time sharing.
- 30-bit word length.

MACHINE ORGANIZATION

Inspection of the block diagram of the FC-4100 (Figure 1) will help to illustrate the degree to which the design objectives have been achieved. A combination of time-shared registers and bus organization was employed to minimize the number of registers and the number of transfer paths among them.

The entire machine centers around a 30-bit data transfer bus. Associated with the bus are three full-word registers, a 30-bit parallel adder, and three half-word registers. The "R", "G" and "A" full-word registers supply inputs to, and receive outputs from, the adder ("S") network. The "A" register also retains instruction results and provides shifting and complementing capability. The "R" register doubles as the core-memory regeneration register. The "S" network not only generates algebraic sums, but also supplies an exclusive—or function for logical operations.

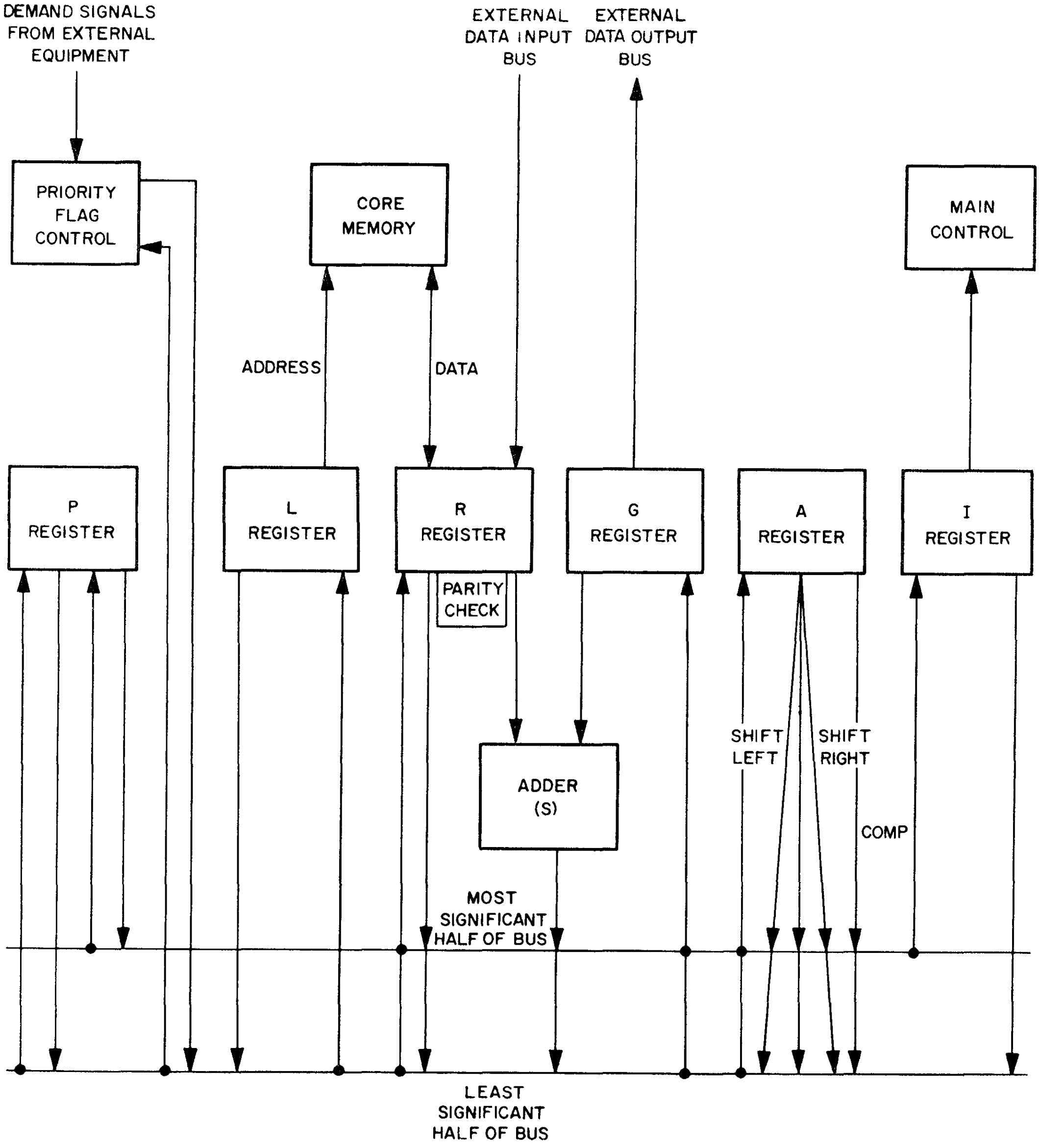


Figure 1. FC-4100 Block Diagram

Storage for the operation control bits of the instruction word is provided by the half-word "I" register. Core-memory addressing is supplied from the half-word "L" register. An additional half-word register, "P", was added to the design when detailed investigation showed that improved performance would be achieved with a net decrease in logic hardware. The "P" register receives the adder output during program-counter advance, index-register decrement, and address modification operations. "P" and "L" also are used together as a full-word multiplier-quotient register.

Thus, in the FC-4100 system, no unique area of logic is assigned for the artihmetic unit, index registers, program counter, etc. For example, investigation has shown that the most efficient utilization of hardware is achieved by having the index registers and program counter stored in the core memory. Thus, it was economically feasible to have a large number of index registers, enhancing programming versatility.

PRIORITY PROGRAM INTERRUPT SYSTEM

The program interrupt feature of the FC-4100 data-processing system represents a novel approach to the problem of coupling peripheral equipment (such as tape stations, line printers, etc.) as well as other computers to a central data processing system. The earliest technique used to implement input-output functions in computers was a completely programmed approach. The programmer was responsible for scheduling data transfers between the system and external devices, and for checking to determine the status of the transfers. This approach is quite attractive when used with low-speed peripheral devices because it requires very little hardware.

A major increase in system speed is obtained when a "multiplexed-exchange" system is added to the data processor. Such a system contains additional data registers, address registers, and control logic. Once a given channel is initialized, blocks of data are handled automatically without disturbing the system program. Higher operating speeds are obtained at a significant increase in hardware. Unfortunately, the system program is still required to make periodic checks in order to determine whether data has entered the system.

The primary purpose of the priority program interrupt system is to eliminate the need for program-controlled scheduling and status-testing of input-output transfers. The priority interrupt feature can be used as an adjunct to programmed input-output and/or multiplexed exchange systems. However, it should be noted that the program simplification which the interrupt feature effects permits the use of the programmed input-output technique with peripheral equipment whose speed might otherwise require the more complex multiplexed-exchange hardware.

The FC-4100 incorporates a priority program interrupt feature which permits response to 14 randomly arriving channel "demand" signals. The memory may be considered as divided into two functional sections: order space and executive space. Order space is available to the programmer in the normal manner for storage of instructions and/or data. Up to sixteen independent or interdependent programs may be stored in this area. Associated with each program in the order space are eight sequential locations in the executive space consisting of one location for program counter storage, six locations for index register storage and one work space.

Each program is assigned a priority identification number and an associated indicator flag. Each flag may be set either under program control or on command of a peripheral device. However, each flag may be reset only by an instruction in its own program. During the execution of each instruction, the complete set of indicator flags is examined to determine which flag set has the highest priority. If this flag is that of the currently active program, the program proceeds without interruption. Otherwise, the system enters the program interrupt mode of operation.

Since the program counter index registers and overflow indicator are stored in memory, only two actions must be taken to implement this mode. First, the program counter associated with the interrupting program is retrieved from its assigned executive space location. This is an automatic system procedure. Second, if the interrupting program contains an instruction which changes the content of the accumulator (A) register, it must first preserve, in its own work space, the accumulator content left there by the interrupted program. Such a program must later retrieve the contents of this work space and place it in the accumulator register.

As an illustration of the operation of the program interruption system, consider that a program having a priority identification number of 8 is currently active. Assume that one of the instructions in this program activates the higher priority Flag 3. For purposes of discussion, this instruction is designated I8. During the final timing steps of I8, the program counter is modified, as required, to the value N8 and is replaced in the core storage. The interrupt control logic is interrogated; the interrupt is sensed. The program counter of Program 3 is retrieved from its executive space location. The next instruction to be executed by the system is located at an address specified by the new program counter. In accordance with established programming constraints, and assuming that this program modifies the accumulator, an early instruction in this program stores the content of the accumulator register, A_8, in the work space of Program 3. At the conclusion of each instruction in Program 3, the interrupt control logic is interrogated to determine whether an interrupt is to occur. If a higher active priority flag is detected, the system will enter the interrupt mode again. Assuming that no higher priority flag is set, Program 3 continues to operate, checking the interrupt control logic at the conclusion of each instruction. During the execution of Program 3, it is quite possible for a number of lower priority flags to be set, either by external devices or by instructions within this program. These lower priority flags do not interrupt Program 3. Before the conclusion of Program 3, the A_8 word is retrieved from the work space of Program 3 and is placed in the accumulator register. The final instruction of this program resets the priority 3 flag.

Interrogation of the interrupt control logic now indicates that a change in programs is required and the system enters the interrupt mode. Suppose that priority 6 is now the highest flag set and that this program will also modify the accumulator. As indicated before, after retrieval of the program counter content associated with Program 6, an early instruction stores the content of the accumulator register (A_8) in the Program 6 work space. Program 6 now operates until the final instruction of Program 6 resets its priority flag. Near the end of Program 6, A_8 is again placed in the accumulator.

This process of transferring A_8 from the accumulator to a given program work space and back to the accumulator continues until the system priority level returns to number 8. At this point, A_8 is in the accumulator, and the program counter for Program 8 is retrieved. The next instruction executed is N8, and Program 8 continues as if nothing had happened.

The top priority program is permanently preassigned to the function of analyzing overflow alarm conditions; its flag is set only by arithmetic overflows not anticipated by the program in which they occur. Next in the priority sequence are executive control routines, followed by the necessary input-output routines and the main programs. The lowest priority program is ordinarily a self-check routine, and is automatically activated whenever the main program is awaiting the arrival of new data.

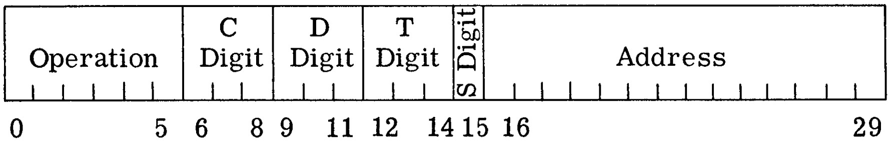
Assignment of relative priority between the various data interchange programs should be based on the allowable waiting time between data transfers. A program for handling the transfer of data between the computer and peripheral equipment is generally short—often a single instruction. For direct transfer to or from memory, the input-output routines normally do not disturb the accumulator. Once data has been transferred to the memory, the lower priority programs are called on to process the new information. Because the various input-output and data-handling programs have different priority levels, the programmer need not perform periodic tests for the arrival of data. The program interrupt system automatically activates the proper program to process new data which has been transferred into the memory by other programs.

INSTRUCTION REPERTOIRE

The complete repertoire of the FC-4100 is shown in the appendix.

The instruction word (see Figure 2) is divided into an operation half-word and an address half-word.

The operation, tag, and address bits are typical of a standard single address instruction format, and specify the major action to be taken by the system. Significant variations of the major action are controlled by the C,



Bits	Definition
0-5	The operation to be performed (such as ADD, MULTIPLY, etc.)
6-8	The conditional branch digit (C), indicating the branch condition test
9-11	The relative destination digit (D), associated with C, determining the extent of the automatic loop
12-14	The tag digit (T) specifying one of six index registers or the program counter to be used for modifying the base address in the instruction
15	The suicide bit (S) controlling the resetting of the "active" program's priority flag
16-29	The memory base address for most instructions

Figure 2. Instruction Word Format

D and S bits which form a secondary instruction used for automatic loop control and priority interrupt flag control. The S bit controls the reset of the priority flag for a given program. When this flag is reset, the program becomes dormant until its services are required by another program or by an external device.

The C and D bits normally are used to control the branching of the program. Either the accumulator, overflow indicator, or an index register may be tested (see tabulation following).

For example an "add" instruction not only accomplishes an arithmetic operation, but also may replace the normal branch instruction required in conventional machines to control small loops.

If the test conditions specified by the C bits are not met, the program advances to the next sequential instruction. When the test conditions are met, the program jumps back the number of instructions specified by the D bits. Using the TST, SKP, and JMP instructions, the memory or accumulator may be tested individually or compared with each other to control branching.

TIMING

Each instruction in the FC-4100 is executed by a suitable sequence of "micro-operations," each of which is capable of resetting a register or transferring data to or from the bus, etc. Each occurrence of each micro-operation is generated by the intersection of the operation control bits with the master time grid described below.

The execution time of each instruction is divided into a number (two to nineteen) of cycles. Each cycle is subdivided into 4, 6, or 8 one-microsecond "slots." The five types of cycles which may exist in the machine are:

(1) A 6-slot cycle to retrieve the instruction word from memory.

(2) A 6-slot cycle, if necessary, to retrieve an index register and compute the effective address of the instruction.

(3) A 6-slot cycle, if necessary, to retrieve the operand and, in most cases, execute the required operation.

(4) Cycles as required to complete the operation.

C Digit	Interpretation
0	DO NOT jump back D instructions; reset overflow flag if D = 1, 3, 5, or 7
1	Jump back D instructions if the specified index register not equal to zero and decrement said index register by 1
2	DO NOT jump back D instructions; decrement specified index register by 1
3	Jump back D instructions if accumulator register greater than or equal to zero
4	Jump back D instructions if accumulator register greater than zero
5	Jump back D instructions if accumulator register non-zero
6	Jump back D instructions if accumulator register equal to zero
7	Jump back D instructions if accumulator register less than zero

(5) A 6-slot cycle to update the program counter and check the program priority control. (Should this interrogation result in an interrupt, one additional cycle is required to retrieve the new program counter.)

In those cycles which use the memory, the address is placed in the L register during the first time slot and a read command is issued. The second time slot is executed and the system waits for a "read complete" signal before proceeding to the third and fourth slots. During the fourth slot, data is stabilized in the R register and a "write" command is issued. The write process is completed during the fifth and sixth slots. Because of the limited number of registers available, implementation of each instruction requires sophisticated manipulations of partial results during each cycle.

A typical illustration is the action required to modify the content of the program counter. (During this cycle, the A register must remain inviolate in order to preserve the result of the previous instruction. The L register is used to address the memory and cannot be disturbed during the read/write cycle. Therefore, only the P, R, and G registers and the adder logic are available to perform the required modification, which is made during the last execution status level of the instruction. Furthermore, the R register content may be changed only during the third and fourth slots).

Slot 1:

The address of the memory location which contains the program counter contents is transferred to the L register and a memory read cycle is started. Simultaneously, the branch control is set if the conditions specified by the operation and C bits are met.

Slot 2:

The content of the program counter associated with the current instruction is retrieved from memory, and transferred to the R register. The G register is set with the proper increment or decrement as determined by the C and D digits of the operation bits.

Slot 3:

The internal overflow indicator is tested and may be reset. The modified value of the program counter is transferred from the adder to the P register.

Slot 4:

The new value of the program counter along with the state of the inferred overflow indicator is transferred to the R register.

Slot 5:

The priority interrupt flags are examined to determine the highest set priority. The internal overflow indicator and the I register are reset.

Slot 6:

If no higher priority flag is set, the next cycle retrieves an instruction from the location specified by the modified program counter. The internal overflow indicator is set to contain the same data as the most significant bit of the R register.

If a higher set priority flag is detected in Slots 4 and 5, this cycle is repeated with the following modifications:

Slot 1:

The memory location addressed is that of the program counter associated with the interrupting program. No branch action is taken.

Slot 2:

The content of program counter associated with the interrupting program is retrieved from memory.

Slot 3:

No change.

Slot 4:

The R register content is not disturbed.

Slot 5:

No change.

Slot 6:

No change. If a higher priority is set, the modified set of actions is repeated as many times as required.

Because of the general nature of the control timing grid and the way the various micro-control signals are mechanized, significant variations of the normally implemented instructions may be incorporated in the system at negligible cost. Tailoring of the system to a specific need is thus simple and feasible.

LOGIC IMPLEMENTATION

The building block of the FC-4100 is a NOR gate—a diode-coupled, diode-biased, grounded-emitter amplifier, coupling the advantages of high fan-in, high fan-out, and restandardization of signal levels. Use of this type of gate results in noise-free operation and simplicity of maintenance.

High fan-out is achieved, with moderate power drain, through the use of silicon diodes as a non-linear base resistor. The high storage of these "stabistor" diodes and the polarity of the logic diodes provide compensation for transistor charge storage effects. The gate output is clamped to speed the output signal-level transitions and to provide a uniform signal level throughout the system. The signal delay through an AND/OR pair of these NOR gates is less than 100 nanoseconds.

A typical register stage (see Figure 3) uses only 4 gates: two for the flip-flop, and two for transmitting to, and receiving from, the central bus. The 30-bit parallel adder, which generates both full sum and modulo-two sum in less than 0.4 microsecond, use less than 8 gates per bit. (Detailed discussion of the adder design is beyond the scope of this paper.)

CONCLUSION

The equipment required to implement the FC-4100 system has been minimized through

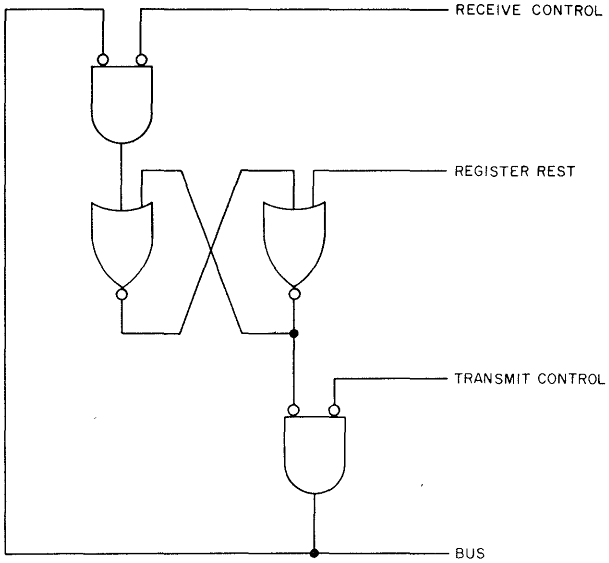


Figure 3. Typical Register Stage

the use of memory locations for index registers and program counters, time-sharing of registers for several functions, and the use of a common bus to interconnect the registers. A program interrupt system was also provided to minimize the amount of equipment required to implement the input/output function.

Program bookkeeping has been minimized through the use of a repeat mode for automatic control of small iterative program loops.

The FC-4100, therefore, represents an integrated solution of the system design goals.

APPENDIX: INSTRUCTION REPERTOIRE OF THE FC-4100

Operation	OP Code	Description	Execution Time (Microseconds)	Indexable
PMA	61	Put Memory in Accumulator	19.2	Yes
PAM	41	Put Accumulator in Memory	19.2	Yes
PMB*	67	Put Memory in Index	25.6	No
PBM*	47	Put Index in Memory	25.6	No
PMO	63	Put Memory to Output	25.2	Yes
PIM	43	Put Input in Memory	19.2	Yes
PSW	57	Swap Memory and Accumulator	21.2	Yes
PZM	45	Put Zero in Memory	19.2	Yes
PZA	73	Put Zero in Accumulator	12.8	No
LAN	34	Logic "And" to Accumulator	19.2	Yes
LAR	35	Logic "And," Replace	19.2	Yes
LIO	32	Logic "Inclusive or" to Accumulator	19.2	Yes
LIR	33	Logic "Inclusive or," Replace	19.2	Yes
LEO	36	Logic "Exclusive or" to Accumulator	19.2	Yes
LER	37	Logic "Exclusive or," Replace	19.2	Yes
LIM	53	Logic "Inclusive or" to Memory	19.2	Yes
ADD	10	Add	19.2	Yes
ADR	11	Add, Replace	19.2	Yes
ADM	12	Add Magnitude	19.2	Yes
AMR	13	Add Magnitude, Replace	19.2	Yes
SBT	14	Subtract	19.2	Yes
SBR	15	Subtract, Replace	19.2	Yes
SBM	16	Subtract Magnitude	19.2	Yes
SMR	17	Subtract Magnitude, Replace	19.2	Yes
MPY	22	Multiply	137.2	Yes
MPH	23	Half Word Multiply	77.2	Yes
DVD	24	Divide	133.2	Yes
DVH	25	Half Word Divide	77.2	Yes
JMP	72	Jump	12.8	Yes
TST	77	Test	19.2	Yes
SKP	76	Skip	19.2	Yes
LSH	01	Logic Shift	**	Yes
LCI	03	Logic Circulate	**	Yes
SFT	05	Shift (Algebraic)	**	Yes
FLS	71	Flag Set	12.8	No
DBN	65	Decrement Index	19.2	No
HLT	00	Halt	--	No

*Tag specifies Index Register affected

**Execution time of shift instructions = 14.8 + 2n (n odd)
12.8 + 2n (n even; shift left)
16.8 + 2n (n even; shift right)

A VERSATILE MAN-MACHINE COMMUNICATION CONSOLE

P. B. Lazovick, J. C. Trost, A. W. Reickord
Radio Corporation of America
Astro-Electronics Division
Princeton, New Jersey

R. S. Green
Auerbach Corporation
Philadelphia, Pa.

SUMMARY

This paper describes a unique man-computer communication and buffering device which meets the need for English-language formulations of business, industrial and scientific problems. The console allows individuals not trained in machine language to use a computer directly. Translation and machine-language editing are completely controlled by the console. The time differential between a man's actions and a computer's responses are automatically buffered. The console can be used in a wide variety of information-retrieval and data-processing applications. It was developed by RCA for the Office of Assistant Chief of Staff for Intelligence (OACSI), headquarters, Department of the Army, under Contract No. DA49-083 OSA-2338.

INTRODUCTION

It became obvious during the development of the ACSI-MATIC system that no adequate communication and buffering console existed which would meet the requirements of untrained, non-computer oriented personnel who would have to deal with a complex data-processing system. Accordingly RCA undertook to develop an experimental prototype of such a console for the purpose of testing and evaluating different communication and buffering schemes. The prototype was to have great versatility so that many types of console operation could easily be simulated. The resulting console, called PAC for Prototype Analyst Console, has itself proven to be a powerful and interesting solution to the problem of providing man-machine communication and time buffering. RCA is now in the process of evaluating this prototype for the purpose of making more specialized consoles for the ACSI-MATIC system. In addition, many other programmed applications have been successfully tested on PAC, which demonstrate the generality of the original concept.

Figure 1 shows the operating position of PAC, and Figure 2 the overall console. The console proper contains a CRT display circuit, a core memory, the operator panels, various control circuits, and power supplies. The soft-copy (CRT) display uses a 21" tonotron (storage/display) tube. Careful design has provided an extremely legible display although the number and bulk of the circuits is minimized.

The actual controls and indicators on PAC are functional devices only in terms of the

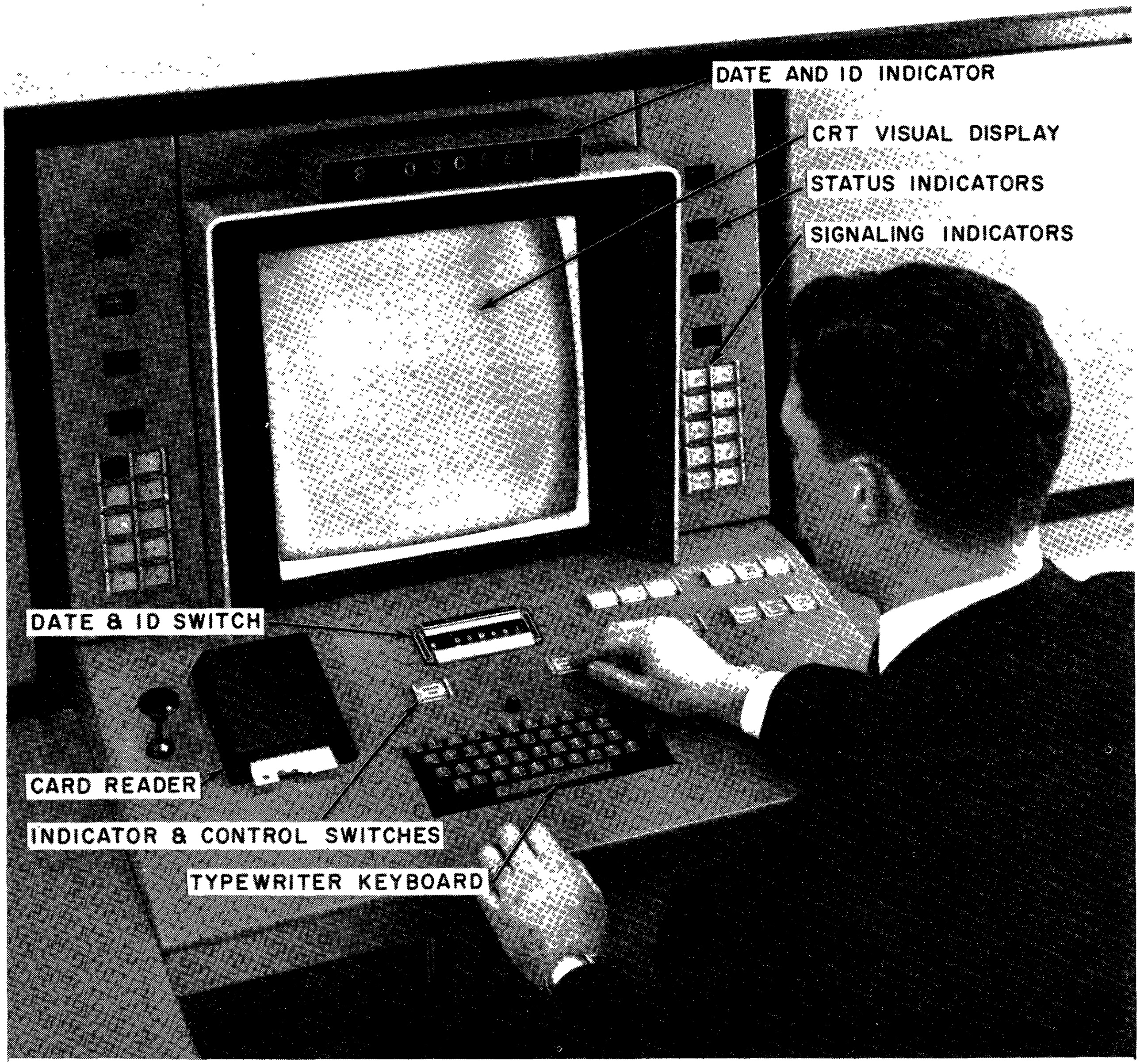


Figure 1. Operating Position of Prototype Analyst Console (PAC)

operator performance for any particular application. The actual function of each control depends upon the computer programming and the restrictions the user wishes to place on the operator. It therefore involves no change whatever in the PAC circuits to completely change the types of console modes and operator identification requirements. The INTERROGATE switch, for example, could just as simply be labelled INTEGRATE, or the SECURITY CLASSIFICATION indication could be changed to BUSINESS AFFILITATION. The only change required would be in the labelling of the indicators and in the type of program read onto a magnetic tape associated with the console.

This tape is called the format tape, and it contains the form of all allowable inputs to the computer. These forms are written on the format tape as soon as computer programs are established to process them. Each format contains both the machine designations required to indicate the program to the computer and a statement, in written English, which indicates the format to the console operator. When the operator selects a particular format,

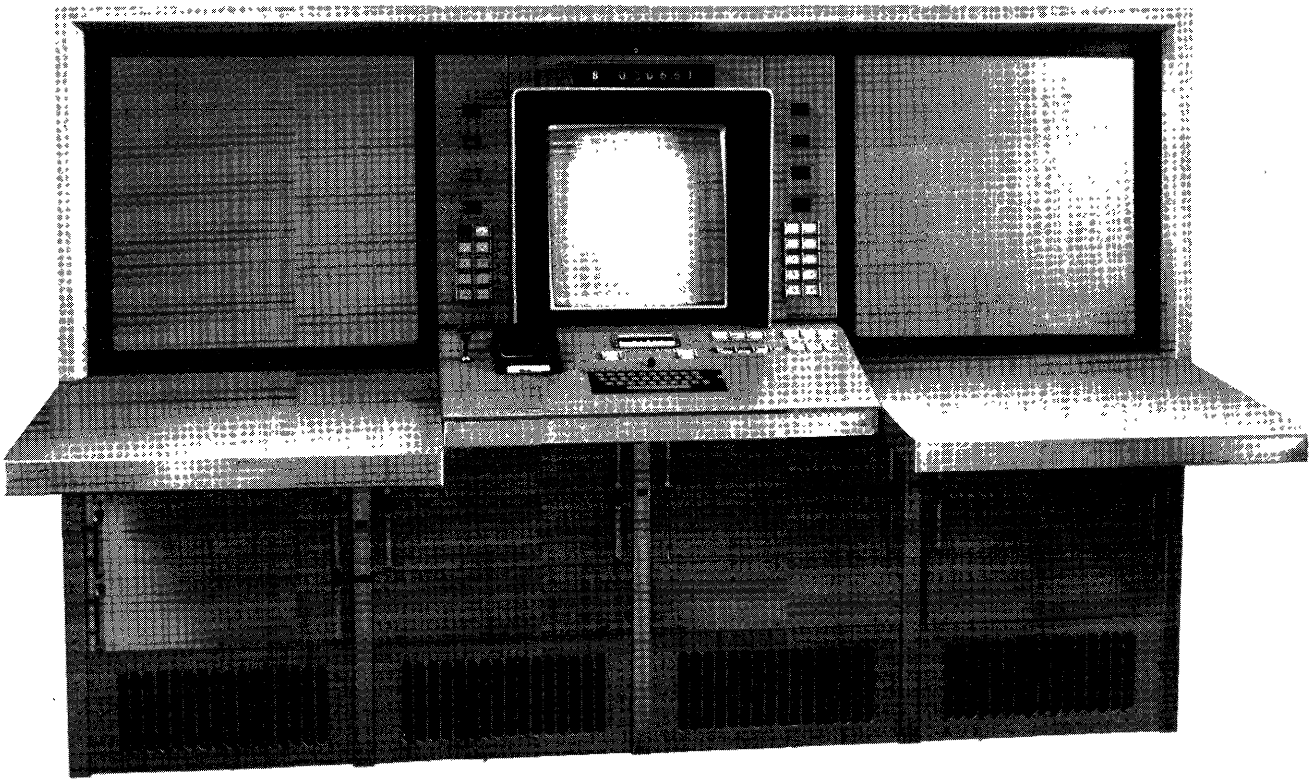


Figure 2. Overall View of Prototype Analyst Console (PAC)

the PAC reads the entire format into its internal core memory and then displays the written statement on the CRT. It then indicates the type of parameter which the operator must insert to compose an allowable message.

The operator inserts his parameters in the format by operating a standard typewriter keyboard. He types the parameter in English and it is displayed, beside the format statement, on the CRT. Each parameter is also read into a composing core memory, where it is grouped with the appropriate machine language items from the format.

When the operator verifies the last required item in the message, the PAC reads the data out of the composing memory and sends it to the paper-tape punch. The punch control then edits the message, removing all data except the header indicating the program, an operator ID, the characters used to separate items, and the operator's parameters.

When the message has been punched on tape, the PAC signals the computer that a message is available. The computer may then read the message at its leisure, placing an answer on the magnetic operator tape. The answer may be retrieved by the operator from this operator tape just as simply as he selected the format.

Messages on the operator tape are preceded by a three-character header which indicates the operator and the question answered. When the PAC locates the desired item on the tape, it reads it into its core memory and then displays the first page on the CRT, roughly the same as a standard typewritten page. The operator then has his choice of printing that page and displaying the next, erasing that page and displaying the next, or printing the entire message. He has identical options for each page displayed.

Security requirements for the PAC are met by providing each operator with an identification card upon which is encoded an ID number. When taking his position at the console the operator must set the control panel ID switches to his number and then insert his card. If the encoded number and the number selected do not agree, a security alarm sounds and the PAC is inoperative until attended by a security officer.

The machine inputs to and outputs from the PAC are standard computer words comprising seven digits in parallel. Since the PAC is only a communications device, it may be used for any system using similar word construction. The only basic change involved would be to place the new formats on the format tape. In addition, relatively straightforward circuit changes can be made in PAC to enable operation with any number of digit computer words.

THEORY OF OPERATION

Figure 3 shows a simplified block diagram of the circuits and components of PAC. The Master Sequencing Control and Nine Operating Routines blocks contain the most important functions from the standpoint of understanding PAC's theory of operation. A brief description is given here of the remainder of

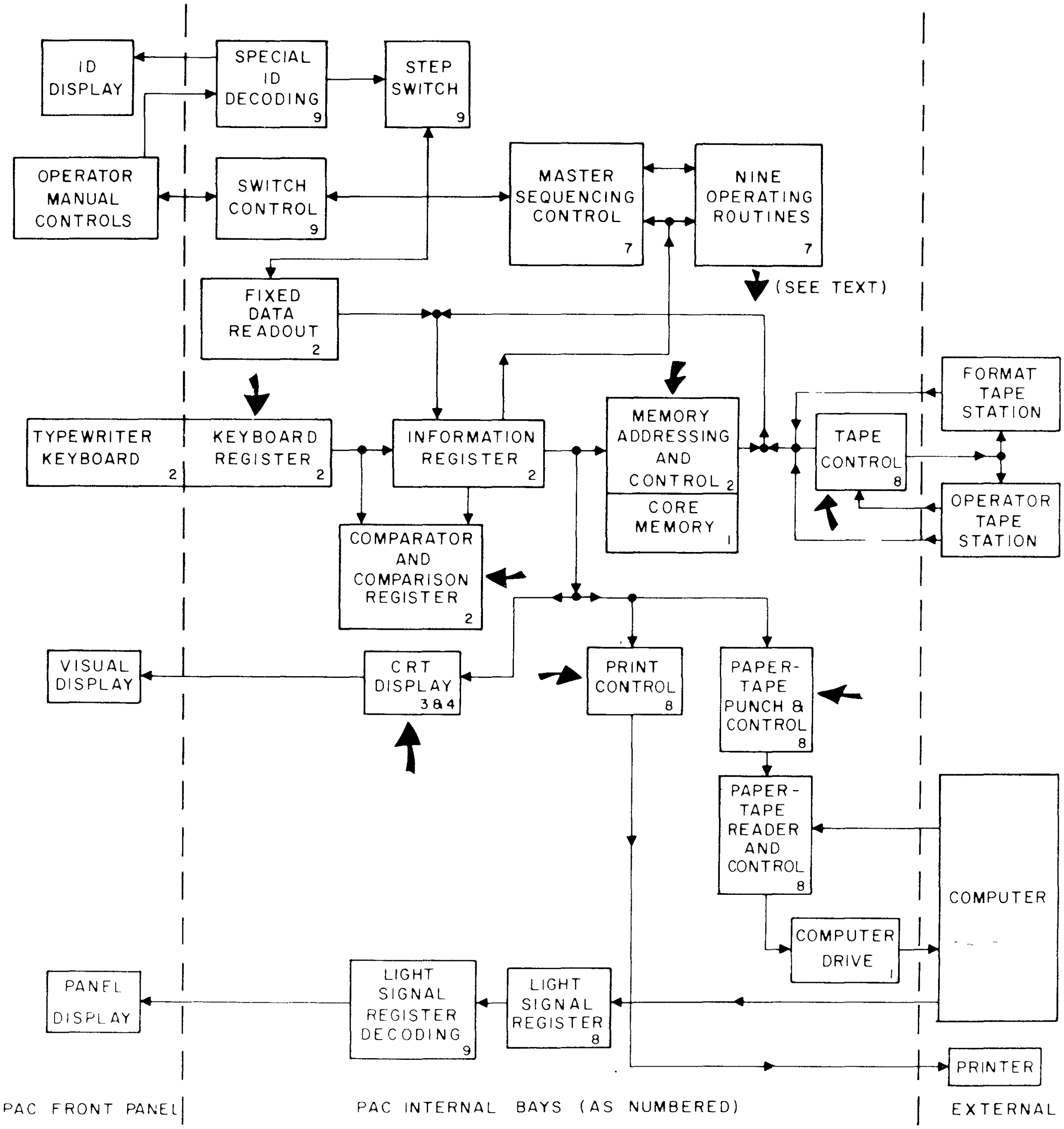


Figure 3. Simplified Block Diagram of Prototype Analyst Console (PAC)

the blocks shown in Figure 3, and the next section describes PAC's operating controls through the Master Sequencing Control block.

The Operator Manual Controls, located on the front of PAC (Figure 1), include the card reader, seven ten-digit switches, and a series of indicator switches. The card reader provides an introduction between the operator and the computer, including an identification (ID) number, the operator's security clearance, his particular area of interest or specialization, etc. The seven ten-digit switches indicate the MONTH, DAY and YEAR, the operator ID, and any other routine information desired. The indicator switches are both indicators and switches which determine the various modes of operation of the console. A typewriter keyboard is provided on the console similar to a conventional typewriter, including a space bar and a carriage return. A verify character is added to this keyboard. When the operator is required to supply information, the indicator lamp above the keyboard is lighted; after typing the requested information which is displayed on the visual CRT display character by character, the operator verifies that he has not made a mistake in typing in his information by pressing the verify character.

The numbers in the lower right hand corners of the remainder of the blocks in Figure 3 indicate the physical bay in which they are located within PAC.

BAY 1

The Core Memory is used to facilitate message composition and enable printing one page of data while displaying another. It is therefore divided into two separate storage units, a composing and a format section each with 2,048 characters, the maximum number of characters per page. The Computer Drive accepts the data output of the Paper-Tape Reader and Control, and ensures that the data is exactly matched to the requirements of the particular computer with which PAC is being used.

BAY 2

The Keyboard Register translates the Typewriter Keyboard data into digital character codes required for the Information Register and the Comparator and Comparator Register. The Information Register accepts and routes all data. A combination of gates at the input of the Information Register reads the incoming 7-bit codes into a storage register of 7 flip-flops, which control gates that apply data pulses to the Core Memory, the Comparator and Comparator Register, the Paper-Tape Punch and Control, the CRT Display, the Print Control, or combinations of these circuits, depending upon the operation required. An arrangement of gates also checks for odd parity, which, if not present, stop operation and light the RESTART indicator. The Comparator and Comparison Register stores the three-character codes used by PAC to search for parameters on the format, prior to comparison with the characters read from tape. If they do not compare, a signal is sent to the Memory Addressing and Control to clear the memory and prevent further read-in. The Memory Addressing and Control controls the read-in and read-out of Core Memory data, adding or extracting the data from the required positions in the core matrix. Fixed Data Readout upon command pulses the Stepping Switch to read the operator identification data through the Information Register.

BAYS 3 AND 4

The CRT Display circuits, which consists of the CRT and the circuits which actually write the data on the screen, accepts the 7-bit character codes from the Information Register. The inner face of the CRT is divided into independent areas by a fine mesh, and PAC divides the usable portion of the mesh into 4000 separate scanning areas, each a 5 by 7 matrix of 35 "dots." The 7-bit characters applied to the CRT Display circuits are decoded by an array of AND gates, each causing a specific series of OR gates to be activated, with each OR gate corresponding to a dot on the matrix, according to the particular character being written. As the CRT beam scans a particular area, the activated OR gates cause their associated portion of the screen to brighten, which allow a variety of characters to be presented on the face of the screen. The number of characters written per line is counted, and when the number reaches 80, the trace automatically moves back to the beginning of the next line. After an individual matrix has been scanned, the trace moves to the next matrix position. When the space signal is received, no gates are activated and nothing is written. When a carriage return is decoded, the appropriate gate drives the trace to begin the next line.

BAY 7

Bay 7 contains the Master Sequencing Control which is the control logic for the overall operation of PAC, described below. The logic for the Nine Operating Routines is also contained in this bay.

BAY 8

The Tape Control located in Bay 8 is the interface between the computer and the Format and Operator Tape Stations. Both of these tape stations can be located with the computer system. The Tape Control upon command switches the output of the appropriate tape to the Information Register, driving the tape forward in search of the information or of the end-tape symbol or until a stop command is given. At end-tape symbol the Tape Control automatically rewinds the tape and searches again. If the end-tape symbol is read a second time, the Tape Control stops the tape and lights the REQUEST NOT ON TAPE indicator. Status signals are provided to indicate whether or not the computer is using the tape. The Print Control controls the hard-copy printing device used with PAC. Upon command from the Master Sequencing Control (when one of the print switches is operated), the Print Control activates the printer and provides the pulses which operate the keys. Upon command from the Master Sequencing Control, the Paper-Tape Punch and Control edits and translates the output of the Information Register into voltage levels capable of driving the Paper-Tape Punch. The Paper-Tape Reader contains the circuits which read the punched-paper tape to develop data signals for transmission to the computer. A counter associated with this reader counts the number of messages punched. When the computer senses a signal indicating that there is a message on the paper tape, it applies an enabling signal to the Paper-Tape Reader which starts the transport mechanism and reads the entire message from the tape into the Computer Drive circuit. As each message is read, a count is subtracted from the message counter. The Light Signal Register accepts the status and alarm indications from the computer and translates them into signals for lighting the appropriate indicators.

BAY 9

The Special ID Decoding circuit is a set of relays which are operated or released in accordance with the settings of the control panel digital switches and the information read by the card reader. This provides the fixed data through the Stepping Switch for transmission with the input messages. This circuit also checks the operator identification number against the number set on the ID switch, and if it agrees, it closes a path which enables the power to be applied to the control panel. The Stepping Switch is a rotary switch which samples the output of the relay circuits in the Special ID Decoding circuit. It is driven by the Fixed Data Readout through a series of 11 positions for the purpose of reading out the information. The Switch Control translates the operation of switches into electronic signals for driving the Master Sequencing Control. The Light Signal Register Decoding controls the lighting of the indicators on the Panel Display.

OPERATING CONTROL

When the operator depresses any of the input mode switches (in the case of PAC utilized with the ACSI-MATIC system these modes are INTerrogation, ORDER to change files, HYPothesis, and NEW MESSAGE input, marked with the capital letters on switches on the front of PAC), it closes a relay in the Switch Control which pulses the Master Sequencing Control. This places the machine in the particular mode designated. It should be understood that the modes are arbitrarily defined by the programming of the computer for any particular application of PAC. The different formats available for any particular mode are given 3-character ITEM Codes, and if the analyst knows the format he is going to use, he depresses the ITEM switch on the front panel and types in the ITEM code on the keyboard. This immediately displays the format on the CRT for his use. If he does not know the ITEM code, the operator presses the INDEX switch, and a list of all of the formats available to him for any particular mode of operation is displayed on the CRT for his reference. He can note the ITEM code of the desired format and then proceed to press the ITEM switch as before for selection of any particular format.

Figure 4 indicates a typical format for an INTerrogation mode with the parameters which might be typed in by an operator for a personnel information retrieval application of PAC. The < symbol is the start message symbol, and the > symbol is the end message symbol. The ♦ symbol is the item separator. The first line indicates interrogation item number (AO6), and the identification of the operator (2386100635). Next is the actual format for the operator to use, with the unknown parameters in parentheses. The first parameter which the operator must insert is displayed on the next line, and he types and verifies the word MANAGERS, which represents that he desires information about MANAGERS. After he has typed in his first parameter and verified it, the second parameter appears on the CRT, namely PLANT DESIGNATION. He then types and verifies WEST COAST CENTRAL. The final line provides him with a means of identifying his interrogation, requesting an INDEX after he has typed and verified the PLANT DESIGNATION. The operator enters 15 as his index for the particular interrogation. The index will be used as a search parameter when the operator wishes to inspect the computer response. This allows batching of inputs at his discretion.

After the operator has composed his message(s), PAC forwards the data to the paper-tape station for punching and storage on tape until the computer can read the information. After the computer has read the information and processed it, it writes its responses onto the Operator Tape and lights the Operator Indicator on the display panel indicating that it has some information for the operator. The operator can select the INSPECT-OUTPUT mode to observe the computer response. This gives the operator an option of viewing the information on the CRT, or of printing the first page and viewing the rest, or of printing the entire response from the computer. The storage persistence of the CRT is such that each page will remain legible for a maximum of about 10 minutes.

The actual logic of PAC which performs the above operations through the Master Sequencing Control is by means of the following nine operating routines: 1) Set-Up, 2) Tape Search, 3) Load Memory From Tape, 4) Display Memory, 5) Read Fixed Control, 6) Rotate Format Memory, 7) Load From Keyboard, 8) Correct Mistake, and 9) Print or Punch. The block diagram Figure 3 indicates where these routines are applied throughout the circuits of PAC by broad arrowheads. Generally, each of these operating routines can be briefly characterized as follows:

1) The Set-Up routine is the sequence by which the operator informs PAC of the item requested, loading the Comparator with the search parameters, either automatically when the INDEX is requested, or manually when the operator types the ITEM.

2) The Tape Search routine controls the comparator and tape circuits to find the message identified by the 3-character header which is loaded in the Comparator.

3) The Load Memory From Tape routine controls the insertion of data into PAC Core Memory from either the format or Operator Tape.

4) The Display Memory routine enables displaying the computer responses or redisplaying the input information after a character or parameter has been erased if it is desired. This allows the operator to make a mistake in composition and to correct it.

5) The Read Fixed Control routine controls the reading of the operator identification and other fixed data from the Stepping Switch into the Core Memory.

<AO6 ♦ 2386100635 ♦

HOW MANY (PERSONNEL CATEGORY) AT (PLANT DESIGNATION)

(PERSONNEL CATEGORY) MANAGERS ♦

(PLANT DESIGNATION) WEST COAST CENTRAL ♦

(INDEX) 15 ♦>

Figure 4. Typical Format for Personnel Information Retrieval Application of PAC

6) The Rotate Format Memory (Into Assembly Memory) routine controls the reading of the format message into the composing half of the Core Memory from the format half.

7) The Load From Keyboard routine controls the operation of the keyboard and the storage and display of information inserted via the keyboard.

8) The Correct Mistake routine reads zeros into Core Memory to replace either the last character typed, or all characters following the last parameter category. It is controlled by the ERASE CHAR or ERASE PAR switches.

9) The Print or Punch routine controls the reading of information out of Core Memory for duplication on a hard-copy printer or on the paper-tape punch.

CONCLUSIONS

The console described in this paper has been in operation with an RCA 501 computing system and has proven to be a successful man-computer communication device which could readily be understood and used by individuals with no computer experience, in a very short time. PAC has sufficient check schemes built in to prevent mistakes from being entered into the computer and taking important time from processing. It presents a calm atmosphere for operation remote from the actual computer installation if desired, and it can operate simultaneous with other similar devices at the same or other locations. PAC is an invaluable tool both from the standpoint of developing the final operational console to be used in the ACSI-MATIC system, and of demonstrating the potentialities of a computer to specialists whose interest is not in computer problems and jargon, but in realistic results.

Acknowledgement is given here to Joseph E. Karroll for considerable technical contributions to PAC, and to Robert E. Mueller and Leon Rosenberg for their assistance in the preparation of this paper.

REFERENCES

1. Miller, L., Minker, J., Reed, W. G., and Shindle, W. E., "A Multi-Level File Structure for Information Processing," Proceedings of the Western Joint Computer Conference, May 1960.
2. Gurk, H. M. and Minker, J., "The Design and Simulation of and Information Processing System," Journal of the Association for Computing Machinery, April, 1961.

DATAVIEW, A GENERAL PURPOSE DATA DISPLAY SYSTEM

R. L. Kuehn
Ford Motor Company
Aeronutronic Division
Newport Beach, California

During the past decade significant progress has been made with automatic, high-volume processing of data. Management and command systems have undertaken to exploit this ability of machine computation, correlation, storage, and retrieval; as a result, the number of computer-centered complexes is increasing daily. Where it is feasible to design or program into the equipment all the decisions which attend a process, as for an on-stream oil refinery, only the simplest form of status indicators need be provided. In contrast, the evaluation of a missile development program or the determination of the order of battle, which produce complex patterns of alternatives, must ultimately depend upon human decisions. The computer excels in the province of deductive logic, but man is still clearly the master in the province of inductive reasoning. Man, however, must understand before he reasons. A data processing system may ingest and digest information at inconceivably high rates, but when the machine delivers the results of its omnivorous appetite to man, a communications problem immediately develops. The solution of the problem lies in the field of data displays. Even the earliest and simplest of computing machines were required to generate an output suitable for human assimilation. Printing devices are common to nearly every system; thus, alphanumeric or syntactical outputs may be obtained. These, however, are not adequate for many conditions today. We are encountering more and more requirements for automatic processing and display of situation analogs.

A typical information svstem is the military map. Positions, distances, and directions are closely analogous to their actual counterparts. Understandable symbolic and color coding methods further enhance the presentation. In a military as well as in many other environments, the situation may be time variant; it may be most satisfactorily perceived in analog form. The characteristics of a display configuration are often restrictive to accommodate a specific set of requirements. In general, however, a useful display system must:

1. operate with any data processor.
2. provide both large and individual sized viewing screens.
3. maintain its own storage off-line to the computer.
4. respond with reasonable rapidity.
5. provide color and symbolic coded outputs as well as alphanumeric characters.
6. possess a high order of visual resolution and geometric (positional) accuracy.
7. be of adequate brightness for comfortable viewing under ambient, light conditions.
8. be able to produce any symbol designation without redesign or rewiring of equipment.
9. be of reasonable physical size.
10. be independently operable in the event of breakdown of electronic elements.

The DATAVIEW system arose from the specific, yet broad demands of a concept for electronic processing and display of tactical information. All but the first of the characteristics listed above are prominent in the Army Tactical Operations Central (ARTOC) displays. This system, bearing the nomenclature AN/MSQ-19, has been developed under the cognizance of the U. S. Army Signal Research and Development Laboratory in Fort Monmouth, New Jersey. The complete system consists of data processing, display, and communications equipment and procedures. Inputs to the system are messages pertaining to tactical friendly and enemy situations. Operationally, the system evaluates, analyzes, correlates, and displays information for the commander, assisting him toward the rapid, accurate formulation of plans and decisions.

The basic concept is diagrammed in Figure 1. A symbol generator may be a part of the configuration, or the central processor could be programmed to perform symbolic construction. Graphical or syntactical information is received by the display generator which produces, off-line, one or more photographic slides. The latter are delivered pneumatically to the console and large screen displays within which the slides are stored for local retrieval and veiwing.

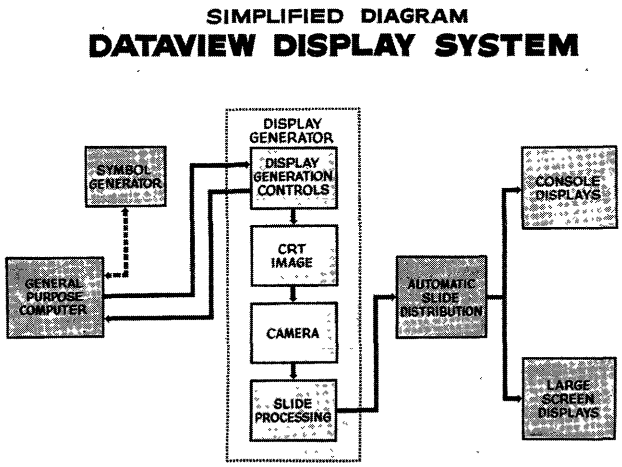


Figure 1. Diagram, Dataview Display System

BASIC CONCEPTS

Before examining the individual equipment elements, it may be of value to describe some of the basic concepts of DATAVIEW. The system is based, first, upon the philosophy of photo-optical quality and permanence, and, second, upon the doctrine of central image generation with localized random access storage. A number of distinct advantages accrue from the concepts. Since an individual display device contains a complete record in graphical format of the most current information, as well as any special or historic data which may be of interest, loss of important electronic elements does not seriously disable the operator. Photo-optical techniques are superior in the realm of additive super-position, color rendition, brightness, resolution, geometric precision and large-screen capability. Furthermore, there is no requirement for electronic storage and high speed iteration in order to provide refreshed displays or to avoid flicker.

To implement the preceding rationale, a slide of the type illustrated in Figure 2 is utilized. There are two parts in the assembly: the image-bearing film material and the steel blade to which the film is bonded. On one edge of the blade are 30 teeth which are selectively removed for coding. Clearance holes are provided in the steel so that registration may be accomplished by means of a pair of film sprocket holes. The film used is Kalfax, a photolytic, daylight-handling emulsion manufactured by the Kalvar Corporation. Kalfax positives, or bright-line images, are obtained by contact printing with a silver halide negative which produces any number of prints from a single data message.

A few words are in order with regard to the silver halide-Kalfax relationship in the system. Conventional photographic emulsions are highly sensitive since, at the time of exposure, the large silver halide granules present substantial capture cross-sectional areas to the incident photons. A high probability exists, therefore, of capturing the one or few photons that may be required, even though not many are present. This condition implies low light levels, and hence high sensitivity. Each developed silver granule now absorbs a large amount of light to produce a dense optical image. The Kalfax emulsion, on the other hand, consists of a photolytic compound which yields volatile products, and which is dispersed in a polymeric vehicle. Two photons, at least, are required for each photosensitive molecule which is considerably smaller than the corresponding silver granules. The latent image results from internal stresses caused by partial volatilization of the compound. Upon subsequent exposure to

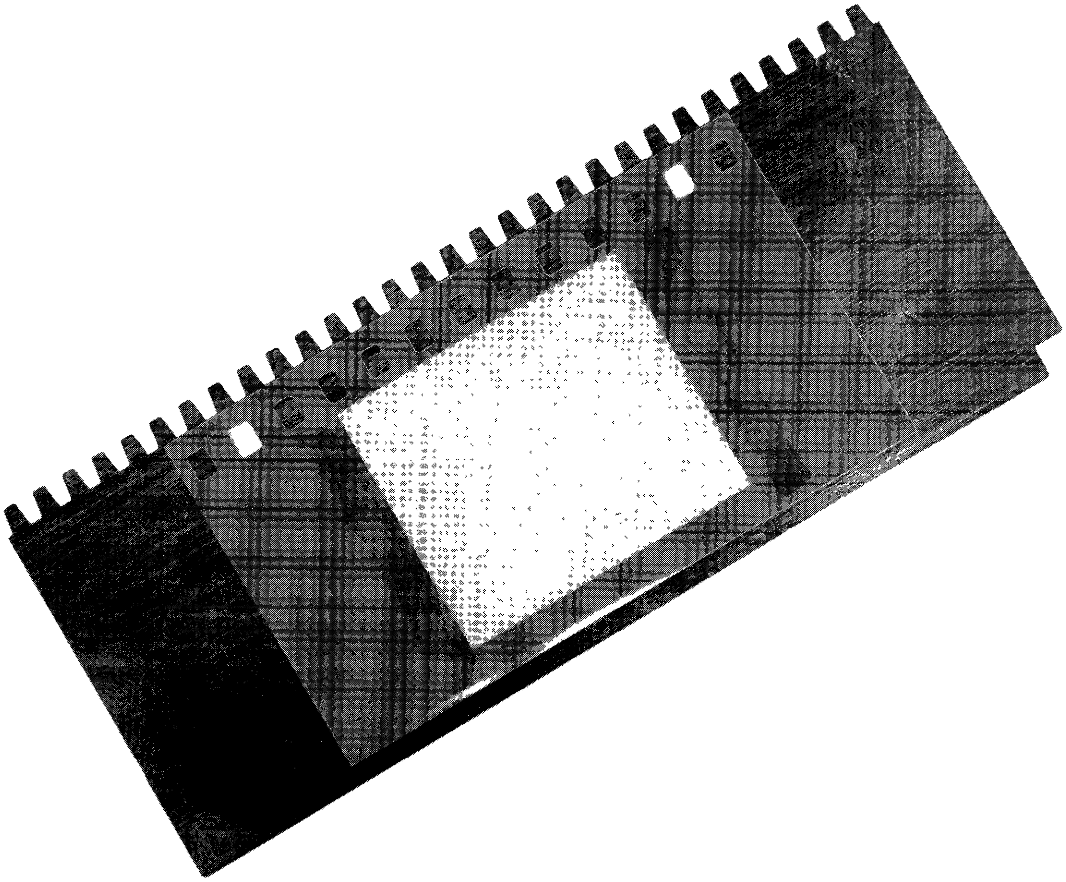


Figure 2. Dataview Slide Showing Teeth and Mounted Film

elevated temperatures, the polymer relaxes and an ordered reorientation and recrystallization occurs corresponding to the pattern of the original stresses. These latter portions of the polymer scatter light to produce the desired visual image. Kalfax emulsions have a spectral sensitivity curve lying in the range of 350 to 440 millimicrons with a peak at 380 millimicrons. Minimum exposure of 1.6 watt sec/cm^2 followed by thermal development of 0.5 cal/cm^2 results in a density of 0.5 above background. Conventional, or silver halide, film emulsions possess the requisite orders of sensitivities and spectral range as well as high resolution, which allows a high quality record to be made from CRT screens. Since more than one copy of a given message may be desired for several display devices, it is convenient to produce prints from the initial negative. Kalfax may be rapidly exposed and developed without the use of chemicals. Thus, it serves as an ideal copy medium.

The local slide storage inherent in the DATAVIEW system must provide a means to obtain information which is current and which may not be found in any given display device. Currency may be maintained by immediate or routine updating of those slide files affected. A number of input equipments (which are beyond the scope of this paper) can be used to direct any available data to a requesting console or large screen display.

The description of the primary elements in the system will serve to illustrate further how the basic concepts have been implemented.

DISPLAY GENERATOR

The Display Generator produces and distributes finished slides. A block diagram of the essential functions is shown in Figure 3. Signal input is serial binary, with 8 bits per word to conform with Army FIELDATA format. Four words define completely any one unique deflection position of the cathode ray tube spot, its luminance level, size, and other requisite data.

Since the electron beam in the CRT is positioned by quadrature fields, eleven bits locate a desired point in X and eleven bits in Y. A set of shift registers accumulate positional information to be converted to analog deflection currents. An absolute accuracy and stability of one half of one location coordinate is achieved by eliminating driving amplifiers for the CRT deflection coils. Digital to analog conversion is accomplished at power levels commensurate with deflection requirements.

A minimum redundancy form of scanning is utilized. Location coordinates are specified at intervals which vary in accordance with the number of points necessary to adequately define a given line or curve. These points are integrated in the deflection circuits to produce smooth, continuous lines. Every data point is traversed only once in any given message, being immediately recorded by the camera assembly as it occurs. Where discontinuities necessarily exist in the image lines, the electron beam is held cut-off, or blanked. The latter is the normal condition of the CRT; unblanking instructions serving to illuminate the screen when required.

Since the data rate is constant and the inter-coordinate distances differ to suit the fine line structure, a varying spot velocity results. Super-imposed on the latter is the non-linear velocity attributable to the integration circuits which must achieve better than 99.9% of final value during the inter-coordinate interval. Compensatory intensity

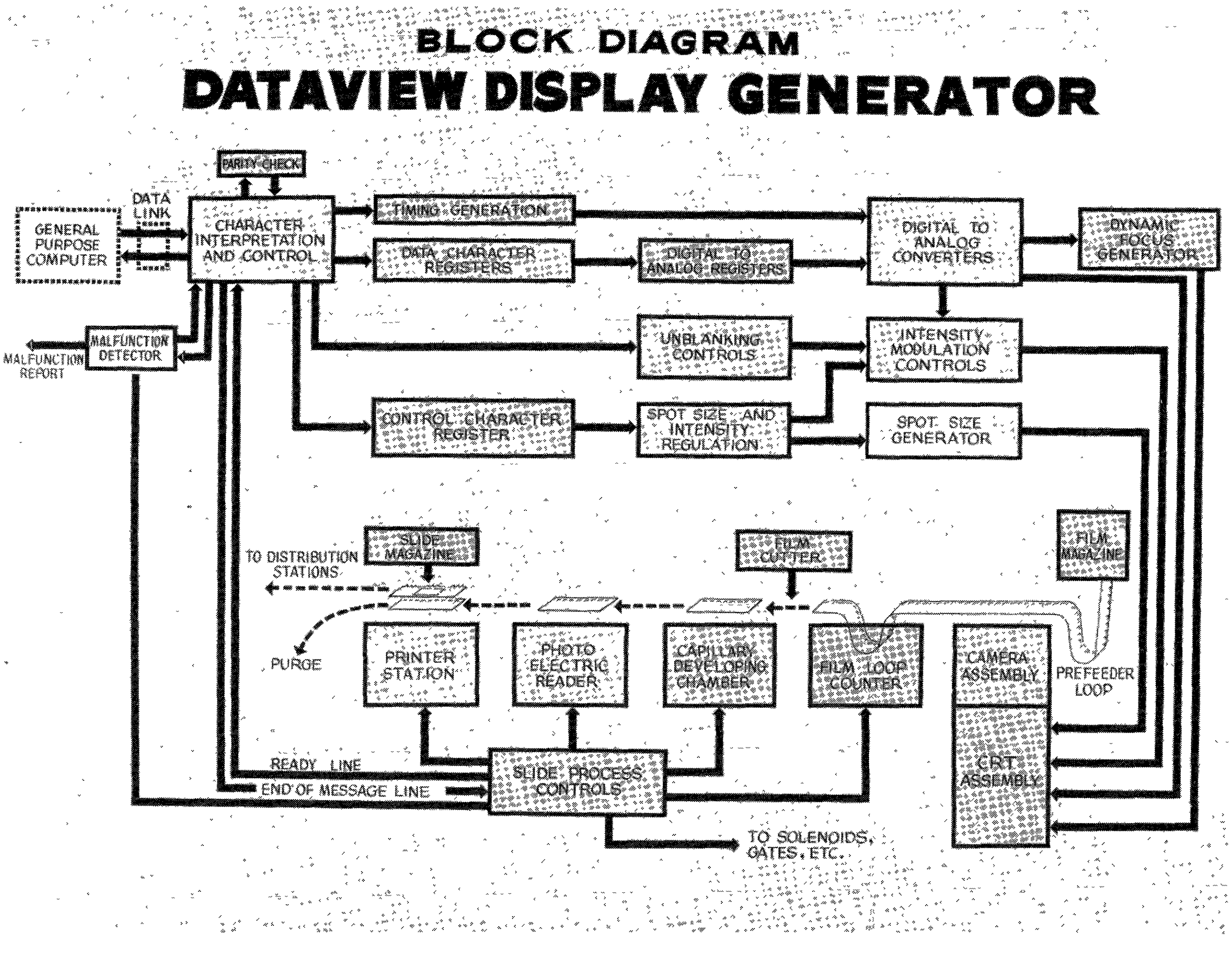


Figure 3. Diagram, Display Generator

modulation is provided to offset both of the latter effects.

A focused spot, in the center of the CRT screen employed, measures approximately 0.001 inch. For this dimension, it does not seem advisable to obtain measurements by the shrinking raster method. As a first estimate, direct microscopic examination with a calibrated reticle has been very successful and correlates well with more refined techniques. By recording a number of line segments on film, it is possible to obtain microdensitometric curves representing the cross-sectional energy distribution in the excited phosphor. Line width is then defined by the half amplitude points. Critical focus is a non-linear function of deflection angle. In order to preserve optimum definition, therefore, dynamic focus is provided.

Variable line widths may be generated as required by the data formats for emphasis or other reasons. Ideally, the energy in the electron beam cross-section would be rectangularly distributed. The actual distribution is approximately Gaussion. Defocusing to achieve wider lines would result in a loss of edge definition due to more gradual sloping of the distribution skirts. Instead, a high frequency circular motion is superimposed on the electron beam to effect a Lissajous pattern of predetermined amplitude. The resultant helical structure is so fine that it is not resolvable.

When all the information intended for one slide has been transmitted, an end-of-message character actuates the slide process controls. The exposed negative film frame is advanced to a two-frame buffer loop, and a ready-to-receive signal is returned to the computer. Film is advanced through the developing chamber as long as exposed frames remain in the loop. Upon entering the chamber the exposed frame is cut away from the loop.

Negative processing takes place by injecting, sequentially, relatively small amounts of developer, fixer, and wash. One fluid displaces another by capillary action. A total fluid volume of approximately 11 cc is required to achieve optimum image quality. The chemicals used for each frame are completely exhausted, and are drained off as waste. A jet of hot air removes the wash solution and dries the negative. The entire cycle requires between six and eight seconds, depending on the desired results. Full development and fixing is obtained in approximately two seconds.

The developed negative consists of three major dark line image areas. First is the data which will appear on the display screens. Lying just below this (and parallel to the long, or horizontal, axis) is an alphanumeric code for visual identification which has been also written by the cathode ray tube. Below the visual code is a series of 40 identifying light and dark areas used for machine reading. The latter are sensed photoelectrically in the final negative station which immediately follows the capillary developing chamber. Two conditions are established in accordance with the dot code: first is the number of positive slides to be made and their destination; and second is the setting of a notching mechanism. A pre-mounted unexposed Kalfax slide is fed from a magazine into contact, with the negative, emulsion to emulsion. Exposure is accomplished by a pulsed xenon arc; then a heated platen is brought into contact with the Kalfax film base to develop the latent image. Two slides per second are thus produced until the programmed quantity is reached. As the positives are made, teeth are notched or removed from the metal carrier to conform with the information contained in the photoelectric reader dots. Thus, each slide also carries a mechanical code denoting its identity. Finally, the negative is purged and the entire process is repeated for subsequent messages.

SLIDE DISTRIBUTION

Housed within the display generator is a slide distribution module shown in Figure 4. In reality, it serves two generators which occupy a single cabinet, and up to twelve display devices may be served by a pair of generators. One generator may be on or off-line as required. When a slide has been made, it is delivered to the distribution module and falls past a series of twelve deflecting gates. The latter are pre-set at the time the negative is examined photoelectrically. Each of the twelve gates is served by a pneumatic tube leading to an individual console or large screen display. Each gate also has a local storage bin. A given slide is deflected into a tube or a bin depending on the pre-programmed condition of that channel. Upon being deflected, the slide is sensed and the gate is re-set so that the next slide may

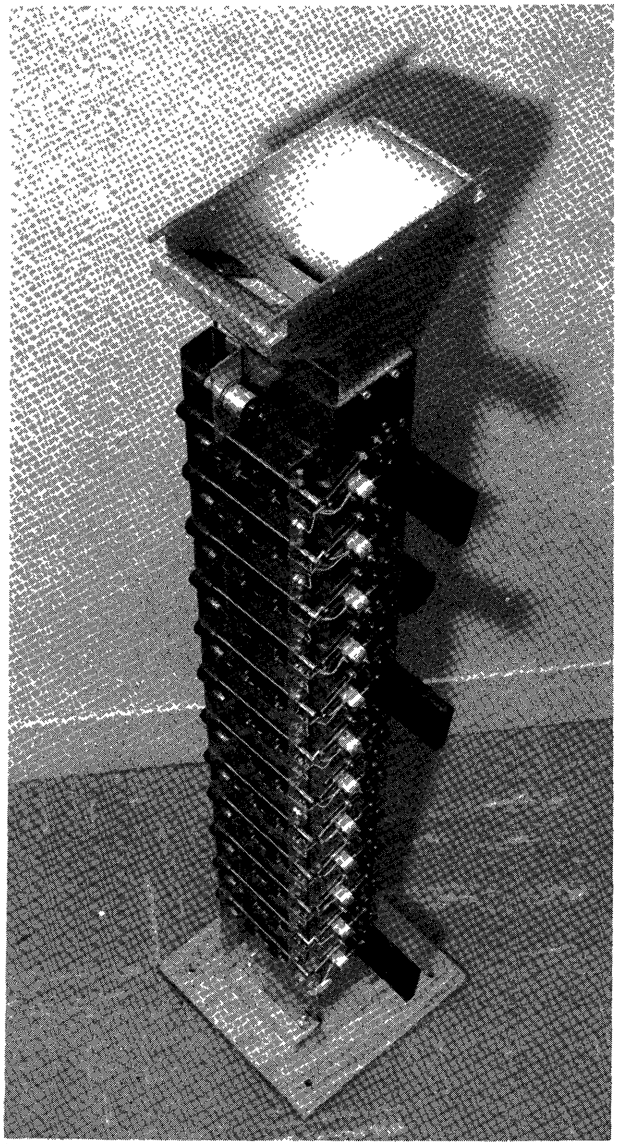


Figure 4. Slide Distribution Module With Dataview Slide in Chute

proceed to the next intended channel. After the last scheduled slide is in place, air pressure is applied to the pneumatic tubes by means of a manifold. Delivery may also be scheduled for one specific channel or for any specific group of channels.

When the Kalfax is bonded to the steel carrier blade, a "propelling" flap is allowed to extend beyond the image area. Air pressure lifts the flap, partially sealing the tube as shown in Figure 5. Thus, an effective piston is created which causes the slide to move with the air stream. Velocities of 50 feet per second have been achieved, and velocities of 100 feet per second are clearly feasible. The pneumatic tubing is a plastic extrusion which may be formed into twists, bends, and compound shapes, to accommodate every conceivable installation requirement. Convenient lengths of tubing are held together by clamps which serve both for mechanical support and for air seal.

DISPLAY UNITS

Functionally, the console display and the large screen display are identical; both devices can therefore be described simultaneously except for certain obvious differences in implementation. A block diagram of the display units appears as Figure 6.

A slide delivered to a console or a large screen display may contain tabulated information, syntactical information, or data referenced geographically to a map or other appropriate background. In any case, the recorded image consists of bright lines with a substantially black surround. Additive superposition of several overlays is thus feasible. Five optical apertures are provided. One of these projects the map or other background which may be in full color.

A single source of illumination is intercepted along two mutually perpendicular axes. The console lamp is a 450 watt xenon arc. The illuminant possesses a number of distinct advantages over incandescent sources. These are listed below.

1. Spectral energy distribution very closely approaches that of the sun, thus improving color rendition.
2. Lamp life is over 1000 hours compared to perhaps 25 to 50 hours.
3. Lamp failure appears as a gradual reduction in luminous output rather than an abrupt blackout.
4. Luminous efficiency is considerably higher.
5. Optical efficiency, due to the small arc size, is substantially greater.

Color. separation for additive superposition is obtained by dichroic mirrors rather than by absorption filters. Losses are thereby minimized. Four mirrors are arranged so that white and nominal red, yellow, and blue light beams are obtained. The band-pass characteristics of the mirrors are selected so that the xenon arc spectral distribution is divided into segments of equal photopic brightness. Each projected light beam may be changed in intensity at the control panel to suit individual

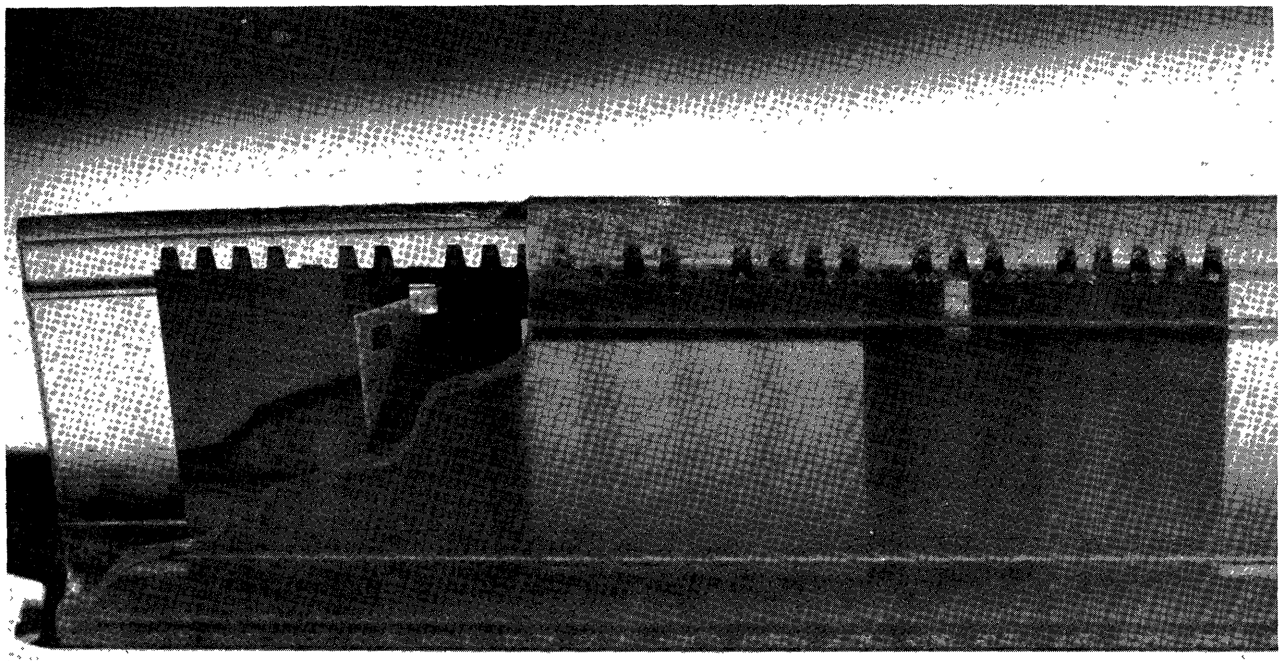


Figure 5. Slide in Pneumatic Tube Showing Erected Propelling Flap

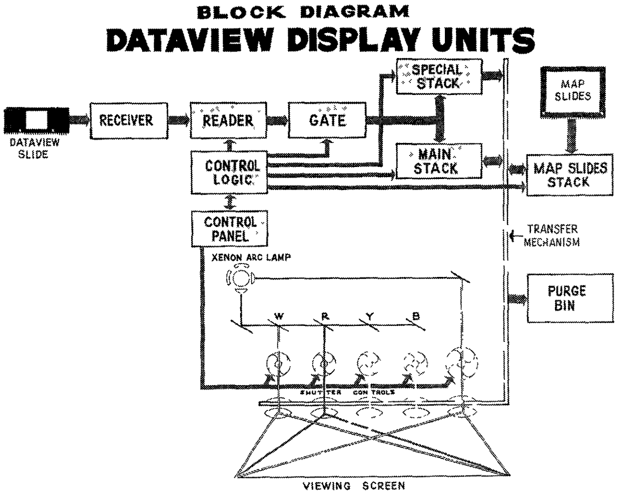


Figure 6. Diagram, Display Units

preference or the nature of the data. It is interesting to note that choice of color for any given format may be forced by the slide code, or may be determined by the operator's preference.

Those channels not in use may be dimmed to black. Three discrete colors, plus white, are directly available. However, by proper programming of the data messages, seven colors may be obtained through additive superposition.

The slides, which are delivered pneumatically to the display unit, enter a magnetic receiver. A manual insertion device is also a part of the receiver. Prior to being loaded into the main magazine or stack, the slide is interrogated by mechanical reading of the notched tooth code. One of several actions ensues:

1. If the arriving slide is a first issue of a given category, it is entered into the main stack.
2. If the slide is an updated version of a slide already in storage, the latter is purged and the new version is loaded into the main stack.
3. If the slide is a "special request" or is a type not normally stored by a given display unit, it is loaded into a small special storage bin; its presence is indicated on the control panel.

The slides are stacked between permanent magnets as shown in Figure 7. Self-alignment takes place in the magnetic field regardless of the number of slides present, up to the maximum capacity. No basic limitation in size exists, but to stay within over-all equipment limitations, the console display has been designed for approximately 1000 DATAVIEW slides and the large screen display for 2000 slides.

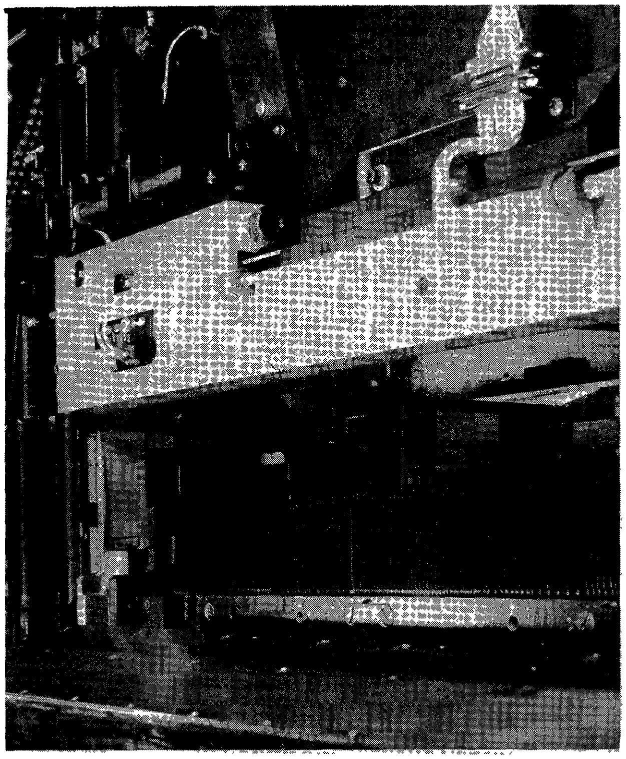


Figure 7. Partial View, Console Slide Storage and Transfer Mechanisms

Access for retrieval is essentially a random process, and is based on the keysort principle. Activating the desired decimal code at the control panel causes a series of corresponding keybars to be extended along the entire stack length. That slide which has been appropriately notched is free to move by one tooth pitch; all others are locked in place. The requested slide can then be withdrawn from the stack regardless of its position and without searching. Once extracted, the slide is picked up by a transfer mechanism which carries it to the intended projector aperture. The entire retrieval action takes less than three seconds. When it is necessary to remove a slide from the files, either automatically or on manual command, it is transferred by the same action into the purge bin. The bin is emptied periodically, and its contents disposed of as required. A complete class of slides may be purged simultaneously if necessary.

Maps or other reference backgrounds are stored and handled in a similar manner, but in separate stacks. The capacity of background slide stacks is approximately 100 slides in the case of the console, and 750 slides in the case of the large screen display. To achieve the high orders of total resolution encountered in many maps and aerial photographs, 70 mm film is used instead of the DATAVIEW 35 mm size. Overlays and associated maps are coded to prevent improper pairing.

System distortions and individual registration errors may contribute to an incorrect position being displayed in an overlay situation analog. The percent error may be defined as:

$$E = 100 \frac{\Delta d}{W}$$

Δ d is the straight line distance between any image point, as displayed, and the theoretical, reference, or intended position for the point; W is the display width. There are a number of individual sources which contribute to the error E. These may be assumed to be random, independent, and normally distributed. Under these conditions, an over-all three sigma value for E is 0.6% applied to a large number of DATAVIEW systems.

Due to the nature of the system, related adjacent image points which lie within a given character or symbol are mutually registered within 0.05% of the maximum symbol dimension. Figure 8 illustrates a typical map and multiple overlay situation displayed on a console screen.

SERVO-CONTROLLED POINTER

Provision has been made for a dial telephone module on all display units. When communicating with one another, operators of display devices may have occasion to discuss a situation analog of mutual interest. They will often find it necessary to point out the specific area in question on one or more remote displays. To accomplish this a servo-controlled reticle may be positioned anywhere on those display screens which are in telephonic contact.

Two frequency modulated carriers in the 300 to 400 cycle region convey a joystick position signal over existing telephone lines. The carriers are purposely made audible, but kept at levels which will not impair intelligibility of the telephone conversation. Raising the joystick from the retracted position actuates the reticle movement. A lockout feature prevents any other joystick from enabling the deflection servos. Rate of response is high; there is no lag associated with reasonable human manipulation of the joystick.

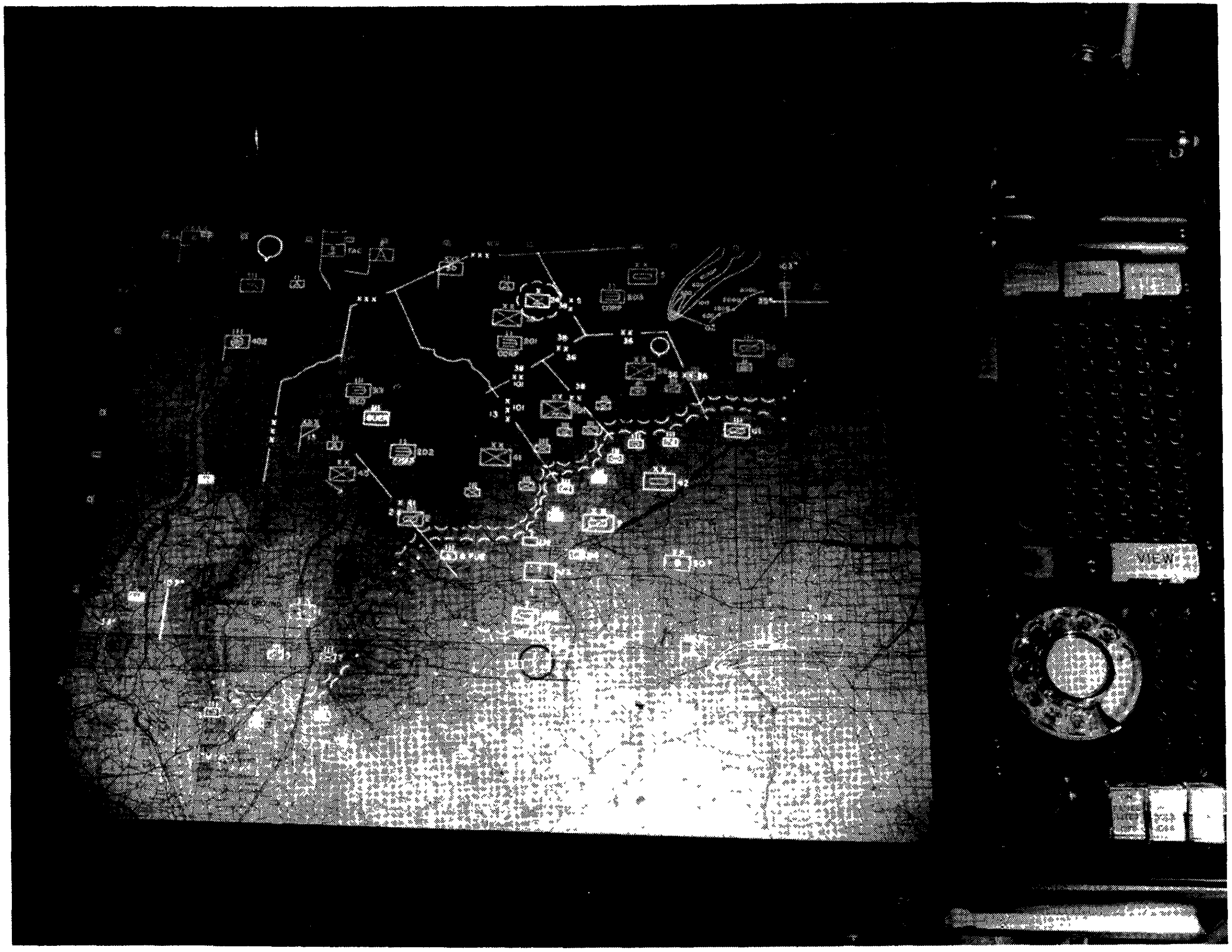


Figure 8. Full Situation Analog Display on Console Screen

The pointer system on a display device may also be used for local briefings when a connection with remote units is not required. This is particularly valuable when a group of people in a single location are examining a complex presentation on a large screen display.

LARGE SCREEN DISPLAY

In terms of the projection angle and design of the viewing screen, the large display is unique. Due to severe limitations imposed by the ARTOC environment, the projector must be very close to the screen, and the screen must be very large. The screen measures 7 x 9.3 feet. In addition, the physical conditions indicate the desirability of rear projection. Even a cursory inspection of the applicable laws of optics reveals that the screen illuminance is seriously reduced for off-normal rays.

Ordinarily, non-uniform brightness in a projected image is caused by the directive properties of the screen. In the case of wide angle projection, the non-uniformity problem is compounded by the so-called "$\cos^4$-law" having greater effect on the incident illumination. The screen illuminance at a given angle from the projection axis is proportional to the fourth power of the cosine of that angle. Since the projection angle on the diagonal is 90°, it would appear that the corner illuminance would be 25% of that in the center of the screen.

The $\cos^4$-law applies, however, to an extended light source possessing a Lambert radiation characteristic, and imaged by an Abbe Sine Condition condenser. An isotropic point source, to which the xenon arc is closely equivalent can be imaged by a specially designed condenser in such a way that the light distribution in the slide aperture compensates for the cos^4-law. Thus, the projector-screen design has been integrated to achieve a high degree of uniformity in over-all brightness.

Despite the 90° diagonal projection angle, focus and distortion are immeasurably different from the center to the corners of the screen. Of the available luminous flux emitted by the lamp, better than 30% reaches the screen compared to 5 or 6% ordinarily expected.

CONCLUSION

The DATAVIEW display system has been described in terms of generalized basic concepts and selected technological details. It has not been possible to explore methods of application to any extent; these methods are best revealed by a closer examination of the equipment itself. Nor has any mention been made of the computer programming which has been developed and implemented to take advantage of the system flexibilities. The necessarily brief overview presented here may, however, serve to acquaint those interested with the elements of at least one display solution to command and control problems.

ACKNOWLEDGEMENT

The significant contributions of many engineers, scientists, mathematicians, technicians, and supporting personnel to the development of DATAVIEW should be acknowledged here although their names cannot be listed. Also, the support and understanding of the U. S. Army Signal Research and Development Laboratory has played a large role in advancing progress of the entire ARTOC project of which the display system is a part.

A COMPUTER FOR DIRECT EXECUTION OF ALGORITHMIC LANGUAGES

James P. Anderson
Burroughs Corporation
Paoli, Pennsylvania

INTRODUCTION

The so-called general-purpose computer is, in a sense, an incomplete design for any given problem. The program is the device which renders the hardware of such a machine a special-purpose computer for a particular problem. In a loose sense, then, programming can be considered a form of machine logical design for specific problems, but at a higher level of abstraction than that generally considered in logical design. This loose equivalence between program and computer has long been recognized by some [1] but exploitation of the relationship has apparently received little impetus, except in a trivial way. Practical benefits of this equivalence have, thus far, been limited to such things as index registers for automatic order modification, built-in floating-point operations, and the like. The justification for implementing such features as additional hardware usually stems, at present, from an inordinately long run time for a particular function, or from excessive difficulty in performing the operations programmatically.

One area yet to be exploited by machine designers is, in fact, that of programming itself. Justification for doing so abounds, with high-priced, high-speed machines frequently employed up to two-thirds of total operating time in the nonproductive tasks of compilation and debugging [2].

The machine outlined in this paper represents another of a recent series of efforts to make computers more easy to use [3]. The major problem which has been addressed is that of programming. It is hoped that the machine described will become a forerunner of a class of computers for which problems are so readily programmed, that programmers can begin to view problems again, rather than details of machine organization and operation.

Although this paper stems from work in the general area indicated in the title, ALGOL '60 was chosen as the basis for the machine because of the well-defined syntax it offers as a guide in structuring similar machines for other algorithmic languages[4].

OUTLINE OF ALGOL '60 FEATURES

ALGOL '60 contains, as separate entities, statements of several forms, each statement representing an abstraction of a computational form. These statements include declarations usually associated with the definition of data in a program, as well as those associated with the definition of subroutines (procedures). The imperative statements of the language can be considered to be of two classes. One class includes the assignment statements and procedure calls which are the computational instructions of ALGOL. All other imperative statements constitute a second class which can be characterized as control statements (conditional statements, go-to statements, for statements, compound statements, dummy statements) used to direct or mark the path of computation, to define loops, and to determine the scope of other control statements.

ALGOL, like many other algorithmic languages, is based upon the sequential execution of a series of statements or statement groups. The flow of control is normally from one statement to another. This sequencing through the statements of an ALGOL program is interrupted when encountering: a) a conditional statement, where the evaluation of a Boolean expression causes a new sequence to be initiated, b) an unconditional transfer of control (go-to statement), and c) iterated statements, where the range of the iterated variable is defined in the structure of the program statements that follow. It is evident, then, that the ALGOL structure abstracts the control form of "general-purpose" computers. It differs in that the "instructions" of ALGOL (assignment statements) may be arbitrarily complex to the extent of being entire programs (procedures), and the name of a procedure or function for obtaining a data value may be used in any expression where that data is required.

CONTROL STRUCTURE

The reference language provides syntactic rules for joining different elements of the language to make meaningful statements. These same rules permit the recognition of syntactic elements of statements and the assignment of meaning (through execution of the appropriate operations) to the syntactic elements.

In addition to abstracting more conventional coding structures, ALGOL '60 is recursive. That is, any of the control statements may appear nested within others or themselves to an arbitrary depth. As an example,

```
If if (Boolean expression) then r else s then
   begin
      for i: = 1 step 1 until n do
                  begin
                  (assignment or procedure statement)
                  end
                  (assignment or procedure statement);
         end else (statement);
```

illustrates the recursive use of a conditional statement and a compound statement. The control for sequencing ALGOL statements must reflect the recursive structure of the language.

A pushdown list, or current control-state stack, can be used to keep track of the current control state, or statement type, effectively recording the structure of a program. A stack is a first-in, last-out device. When data is to be placed in a stack, all previous data is pushed down one position, and the new data entered into the top of the stack. Data is obtained destructively from a stack; data removed is erased, and all previous data is pushed up one position. The entries into this stack, which can be considered "states" of the program, determine the interpretation (in terms of control) of subsequent program symbols.

The sequence of delimiters in an ALGOL program establishes control states corresponding to the type of control or computational statement involved. In some cases (for example, for statements) subcontrol states are established. The handling of iteration and conditional statements can be accomplished by manipulating only the top of the current control-state stack and the next sequential control operator or separator from the program. In a similar manner, the control necessary for assignment and procedural statements can also be obtained.

Syntactic recognition of elements can be illustrated in matrix form, indicating, for a given current control state, the action to be taken by the machine upon encountering each of the symbols of the program. Figures 1 through 4 are matrices indicating the control states in the machine. The rows of the matrices correspond to the current control state, while the columns correspond to the various delimiters and identifiers that can occur in a program. In each of the figures, only those delimiters that are important to the states in question are shown. The occurrence of delimiters other than those shown in either undefined or an error condition. The control state names are roughly suggestive of their corresponding control functions.

The following notation is used for the control sequences in the recognition and control matrix and subsequent sections:

Enter
(control state name) — Push down the control-state stack, and record the control state named in the top of the stack; establish the control state named upon completion of the sequence.

Replace with
(control state name) — Record the control state named in the top of the stack; establish the control state named.

READ — Obtain the next program symbol.

REPEAT — Push up the control-state stack, and establish the new control state of the control state uncovered; execute the control operation specified by the new state and the old program symbol.

EXPRESSION CONTROL

In general, actual computation is carried out in the EXPN state. In this state, two additional stacks are used, in order to execute expressions with due regard for the precedence of the operators. This precedence is necessary to resolve apparent ambiguities in expressions where parentheses are omitted. If the operators are ordered as follows:

↑ × ÷ + - ≤ < = > ≥ ≠ ¬ ∧ ∨ ⊃ ≡

then there are essentially three control sequences to be handled. The sequences are based upon the contents of the top of an operator stack, and the next operator obtained from the program [5], [6], [7]. All identifiers, regardless of the state of the operator stack, are entered in the operand stack.

The control sequences based upon the relative precedence of the next program symbol (operator) and the operator in the top of the operator stack are as follows:

Next program symbol	<·	Top of operator stack	Enter in operand stack; READ; execute the instruction in top of operator stack using operand(s) in top of operand stack; REPEAT
Next program symbol	≐	Top of operator stack	Execute instruction in top of operator stack using operand(s) in top of operand stack; replace top of operator stack with new symbol; READ
Next program symbol	·>	Top of operator stack	Enter in operator stack; READ

The notations ·>, <·, and ≐ are read "greater precedence," "lower precedence," and "equal precedence." REPEAT, terminating the sequence for <·, is similar to the REPEAT used in the control sequences. REPEAT here means "raise the operator stack, and determine the next control sequence upon the basis of the new operator in the operator

NEXT SYMBOL / STATE	BEGIN	END	STEP	WHILE	UNTIL	DO	,	;	:=	(ALL OTHER DELIMITERS)
FORST									replace FS5 enter FS1 save symbol counter enter EXPN READ	
FS1			save symbol counter replace FS2 enter FS4 READ	save symbol counter replace FS6 enter FS4 READ			save symbol counter replace FS4 READ			
FS2		reset symbol counter replace FS6 enter FS4 READ			add step value enter FS3 enter EXPN READ			reset symbol counter enter EXPN READ	duplicate last identifier stored save symbol counter enter EXPN READ	
FS3						(step value limit reached) replace FS7 READ - - - - - - - (step value limit not reached) replace ST READ	(step value limit reached) replace FS7 READ - - - - - - - - (step value limit not reached) replace FS4 READ			
FS4	READ		READ	READ	READ	replace ST READ	READ	READ		READ
FS5		reset symbol counter enter EXPN READ						reset symbol counter enter EXPN READ	reset symbol counter enter FS1 enter EXPN READ	
FS6							(preceding EXPN true) replace FS4 READ - - - - - - - (preceding EXPN false) replace FS7 READ			
FS7	count skip counter up by 1 READ	count skip counter down by if zero, REPEAT else READ						if skip counter zero, REPEAT else READ		

Figure 1. Iteration Control

stack and the program symbol previously found." The EXPN state is left (via a REPEAT) whenever one of the delimiters in Figures 1 through 4 is encountered.

MACHINE LANGUAGE

The machine language is based upon the specification for ALGOL '60. Programs are strings of symbols drawn from a 2048-character alphabet making up declarations and statements imbedded in the ALGOL control structure. In general, the attempt has been to implement the major computational aspects of the language without unduly complicating the control. The following is a partial list of the deviations from the ALGOL reference language:

a) No own declarations
b) No arithmetic expressions as actual parameters
c) Omission of comment from programs
d) Omission of strings as data elements
e) Limited use of designational expressions.

COMPUTER ORGANIZATION AND ELEMENTS

The major elements of the system shown in Figure 5 are a program memory, a value memory, an arithmetic unit (with associated arithmetic registers), three stack memories used for interpretation and control, and an address table. The arithmetic unit and value memory are much like those in conventional

NEXT SYMBOL STATE	IDENTIFIER	ARRAY	,	;	:	:=	[	]
AS1	place in identifier stack READ		enter AS2 READ	REPEAT				
AS2			enter AS2 READ	REPEAT			enter AS3 enter EXPN READ	assign array storage from next available locations to identifier in top of identifier stack, record type appropriate READ
AS3			compute range for bound pair enter AS4 enter EXPN READ		obtain absolute value of previous expression enter EXPN READ			compute array storage REPEAT
AS4								compute another dimension of array REPEAT
RS	place in identifier stack READ	replace AS1 enter AS2 READ	assign next available location to identifier in top of identifier stack, record type appropriate READ	REPEAT				
SW1	place in identifier stack READ			REPEAT		assign location of switch table to identifier in top of identifier stack enter SW3 READ		
SW3	place in identifier stack READ		record identifier in identifier stack as control word READ	REPEAT				

Figure 2. Declaration Control

processors. The three stacks are used for control and temporary storage of the operand names (identifiers) and operators, for proper execution of expressions.

The program memory stores the individual symbols of a program, one to a word, or syllable. One bit is used to specify to the machine whether a syllable is an identifier or a delimiter. Associated with the program memory is a symbol counter (SC). The symbol counter, which corresponds to a program counter in a conventional machine, addresses successive symbols of the program in the program memory, and places these symbols in a staticizing register, or window register (W). There, the type of symbol is determined, and, depending upon the control state, is entered into the appropriate stack. The two auxiliary registers (IVR and RVR) are used in iteration.

One of three stack memories (C), with its associated stack pointer (K), records the current control state. The control delimiters of the program cause generation of the appropriate control state. This state, entered into stack memory C, then determines a new control regime. The other two stack memories (0 and A), with their associated stack pointers (S and H), record arithmetic operators and identifiers, respectively.

The address table is a mechanism for relating the identifiers (names) of data with the locations in the value memory where the values are stored. The entries in the address

NEXT SYMBOL STATE	END	THEN	IF	ELSE	;
CONS		(preceding EXPN true) replace ST3 enter ST READ - - - - - - - - - (preceding EXPN false) replace ST1 READ			
ST1			count skip counter up by 1 READ	(if skip counter is zero) replace ST2 READ - - - - - - - - - (if not) count skip counter down by 1 - - - - - - - - - (if skip counter is now zero) replace ST2 READ	
ST3	REPEAT			replace FS7 READ	REPEAT

Figure 3. Conditional Statement Control

table are of two parts: a level part and an address part. The level part is used to differentiate between identical identifiers occurring in different blocks (levels) in an ALGOL program, and to assist in re-establishment of the correct level when an unconditional transfer of control to a higher level block occurs. The address part is the location in the value memory of the data corresponding to a particualr identifer. The identifiers correspond one-to-one with the positions of the address table. Associated with the address table is the address table register (ATR). The transfer of data from the value memory to the arithmetic registers occurs by placing the address part in the memory address register (M).

The words of the value memory contain the values corresponding to identifiers in the program. Either a uniform representation of integer and real values similar to that found in the Burroughs B5000 system, or an integer floating-point tag on each data word is required, since the operators do not differentiate between integer and real values. Two counters (P and Q) are used with the value memory to record the extent of data and to file control words. Data storage starts at the low end of memory and runs toward the high end, as shown in Figure 5. In addition to arithmetic variables, the value memory is used to store control words that are, in the main, declared variables, the identifier of which has been used again in a lower level block of

NEXT SYMBOL / STATE	BEGIN	END	FOR	IF	PROCEDURE	IDENTIFIER	ARRAY	SWITCH	REAL INTEGER BOOLEAN	(	;	:=	:
INIT	enter CPS READ	stop, end of program											
CPS	enter CPS READ	REPEAT	enter FORST enter EXPN READ	enter CONS enter EXPN READ	count level counter up by 1 replace BLK	enter EXPN	count level counter up by 1 replace BLK	count level counter up by 1 replace BLK	count level counter up by 1 replace BLK			enter ASGN enter EXPN READ	place identifier in top of identifier stack as label READ
BLK		count level counter down by 1 REPEAT			enter PRO READ		enter AS1 enter AS2 READ	enter SW1 READ	set switch to fill appropriate type mark enter RS READ				
PRO		REPEAT				place in identifier stack READ				record position of procedure as control word enter FS7	REPEAT		
ST	replace CPS READ		replace FORST enter EXPN READ	replace CONS enter EXPN READ		place in identifier stack enter EXPN READ					REPEAT	enter ASGN enter EXPN READ	place identifier in top of identifier stack as label READ
ASGN		store result of expression in location specified by identifier in top of identifier stack REPEAT									store result of expression in location specified by identifier in top of identifier stack REPEAT		

Figure 4. Assignment and Block Structure Control

the program. Control words are recorded starting at the high end of memory, and are run toward the low end.

The arithmetic unit proper is conventional, except, perhaps, that the specification of arithmetic type is carried in the data word rather than in the instruction. The arithmetic registers (T and U) are the accumulator and extension register as well as the top two positions of an arithmetic stack.

OPERATION OF THE SYSTEM

Declarations

Declarations encountered in the program are executed to reserve space in the value memory corresponding to the identifiers used. The sequence upon encountering real, integer, or Boolean declarations causes the location in the value memory assigned to that identifier to be recorded in the address table in the position corresponding to the identifier named. An array declaration causes a block of storage corresponding to that identifier to be reserved by advancing the data storage counter (P) by the amount indicated by the bound pair list.

The old value of the address table is recorded as a control word in the value memory if any identifiers have been used at a higher level before the new storage position is recorded.

For example, an integer declaration of the form

integer a ; . . .

has the following control effect:

1. Upon encountering integer in the program, cause a fill switch to be set and the RS state to be entered.
2. Upon encountering the identifier a in the window register (W),
   a. (H) + 1 → H increase the A stack list pointer by one
   b. (W) → H* enter W into the A stack
3. Upon encountering the semicolon,
   a. (P) + 1 → P advance P counter
   b. (H*) → ATR A stack → address table register
   c. (H) - 1 → H reduce A stack pointer by one
   d. if (ATR*(L)) ≠ φ, if address table position has been used, do e and f; else go to g
   e. (Q) - 1 → Q advance Q counter
   f. (ATR*) → Q* record previous address table entries control word
   g. (P) ⊕ (L) → ATR* record P and L in address table
   h. type mark → P* set type mark (integer, real) in data word

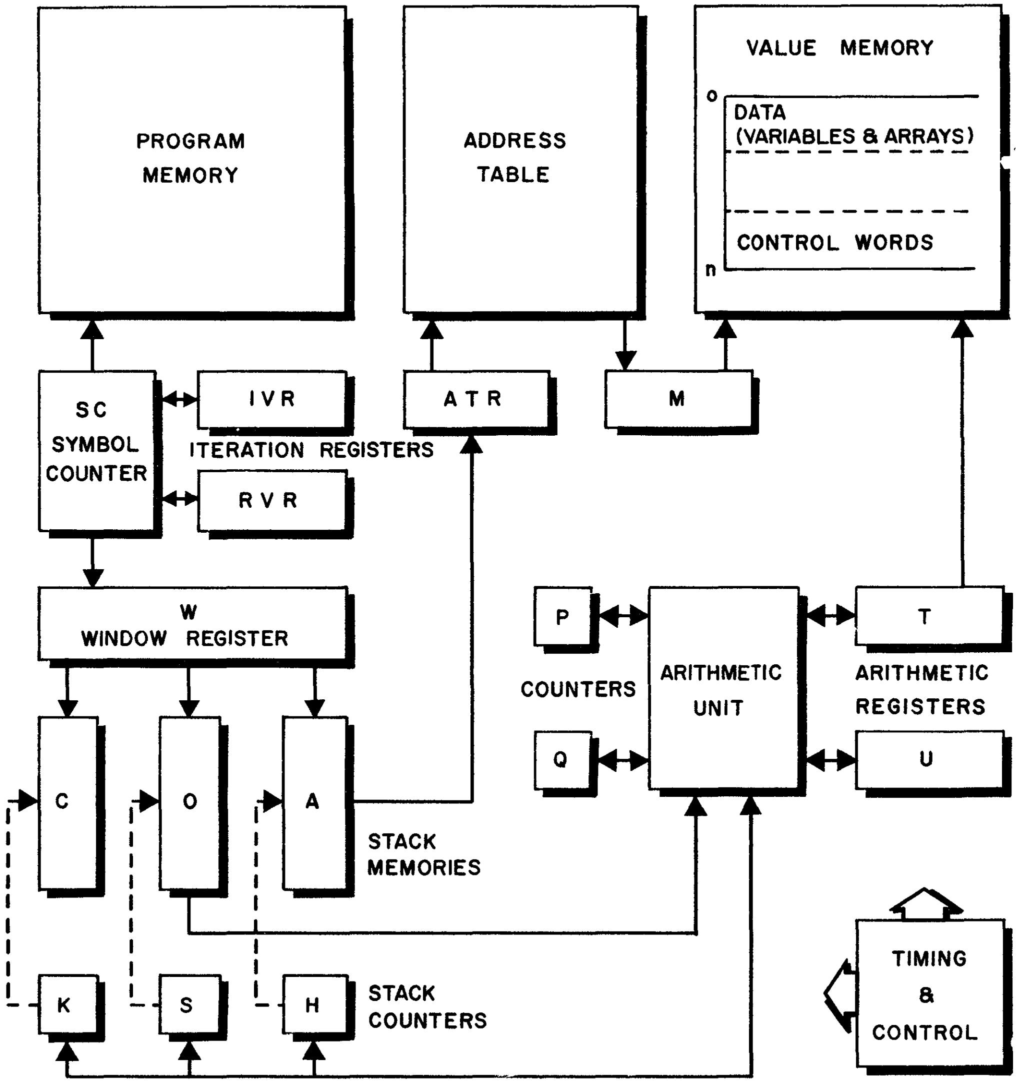


Figure 5. Computer System Diagram

In these expressions, the asterisk is read as "addressed by," such that H* means the location (in the A stack) addressed by the contents of H. The symbol ( ) means "the contents of," such that (A) is read as "the contents of A," and (A*) is read as "the contents of the location addressed by A." The symbol ⊕ means that the adjacent words are concatenated, or merged into a single word.

In a manner similar to the above, other declarations are executed to reserve storage and to retain the previous identifier in the address table.

Arithmetic and Boolean Expressions

Arithmetic and Boolean expressions, occurring in assignment statements, designational expressions, and the like, describe the computations to be performed. The computations represented by these syntactic forms are executed in the EXPN state. In this state, the identifiers enter the A stack, and the operators enter the 0 stack. The sequence of execution is determined by the precedence of the arithmetic and logical operators.

For example, an assignment statement of the form

C: = a + b;

is executed via the following sequences (where W, again, is the identifier encountered in the window register):

W	STATE	Control Sequence	Comment
C	EXPN	(H) + 1 → H, (W) → H*, READ	symbol → A stack
:=	EXPN	(K) - 1 → K	REPEAT
:=	ST	(K) + 1 → K,<ASG> → K*, (K) + 1 → K, <EXPN> → K*, READ	ENTR ASG, ENTER, EXPN, READ
a	EXPN	(H) + 1 → H, (W) → H*, READ	symbol → A stack
+	EXPN	(S) + 1 → S, (W) → S*, READ	operator → A stack
b	EXPN	(H) + 1 → H, (W) → H*, READ	symbol → A stack
;	EXPN	(H*) → ATR, (H) - 1 → H, (ATR*) → m, (m*)→ T, TFF = 1, (P) + 1 → P, (U) → P*, (T)→ U, (H*) → ATR, (H) - 1 → H, (ATR*) → m, (m*) → T, (EXECUTE T OP[S] U) → U, TFF =ϕ,(S) - 1 → S, REPEAT	EXECUTE a + b change state
;	ASSIGN	(H*) → ATR, (H) - 1 → H, (ATR*) → m, (T) → m*, (P*) → T, (P) - 1 → P, TFF = 1, (S) - 1 → S, READ	assign (A + B) to C

Iterated Statements

Iterated statements are controlled by statements of the form

for <variable> := <iteration list> do.

The scope of the running variable is determined by the structure of the subsequent statements. The two auxiliary registers (IVR and RVR) hold the appropriate place in the iteration list of the for statement. When the scope of the running variable is terminated, the symbol counter is set to the appropriate value (contained in IVR or RVR) to obtain the next element of the iteration list. The process continues until the end of the list is reached, at which time the statement to which the running variable applies is skipped over.

Conditional Statements

In a like manner, conditional statements are executed. The CONS state is recorded in the C stack, and the EXPN state is then entered to evaluate the Boolean expression. Upon termination of the evaluation, the truth or falsity of the expression is determined in the CONS state, and the appropriate state is entered to execute either the true statement(s) or the false statement(s). In either case, provision is made for skipping over the alternate statement.

Procedures

The procedural form of ALGOL '60 is an abstraction of subroutine structure, and differs substantially from ordinary machine subroutines in that it permits recursive calls on subroutines. This provision implies a mechanism for assigning storage at run time. Such a mechanism exists in this machine inasmuch as the execution of variable declarations reserves storage for each variable, when encountered. The PRO state merely records in the address table the position of the procedure in the program memory and skips over the body of the procedure. Execution of the procedure causes a change of level, a recording of the return point, and entry of the program memory location of the procedure in the symbol counter. The specifiers and variables for the procedure are executed, previously used labels being filed in the value memory as control words. Procedures declared within procedures are handled in a similar manner.

RELATION OF MACHINE STRUCTURE TO ALGORITHMIC LANGUAGES

Several features of this machine have generality beyond ALGOL '60. These features can be identified in relation to characteristics of the language. The control stack is a device for recording the static structure of a program as a series of states which determine the interpretation of subsequent symbols of the program. This device takes advantage of the sequentiality of computation processes. Such a structural device is inadequate, of itself, for representing control of parallel computational processes, a subject that has become of increasing interest recently. Any language form that provides indicators for parallel processes may be handled by multiple arithmetic and control units in a manner similar to that employed in the Burroughs B5000 computer system.

The address table feature of the machine is a direct analogue of the tag tables maintained in an algebraic translator. In this version of the machine, the address table is a limited form of associative memory. To provide the feature of variable-length identifiers, a link-list memory would suffice, but at some sacrifice in efficiency.

The A and 0 stacks for identifiers and operands are another analogue borrowed from the construction of compilers. These stacks are necessary because of the precedence attached to various operators. Without the A and 0 stacks, some kind of pre-execution translation would be necessary for the machine.

For similar languages, somewhat different recognition logic may result, but the structure of the machine would remain essentially the same.

CONCLUSION

The task of working toward an isomorphism between machine organization and the manner in which a user expresses a problem has occupied a major segment of the computer industry for many years. The machine outlined represents another step in this process. More than anything, the machine illustrates the enormous strides made in constructing algorithmic languages that properly abstract programming techniques. It further illustrates the functional requirements necessary to directly implement current forms of algorithmic languages. The emphasis has been on control rather than arithmetic expressions, because of the large body of work that already exists on the latter.

The organization outlined above is perhaps an oversimplification, but since good programming languages properly reflect the concepts and abstractions for a particular problem class, their efficient implementation can be a useful design goal when considering a new machine.

ACKNOWLEDGMENTS

The author is indebted to E. L. Glaser and K. Speierman of the System Research Department, Burroughs Laboratories, for providing the milieu and many valuable suggestions in connection with this work. A particular debt is owed R. Barton, a consultant for Burroughs Corporation, for the valuable insights in connection with this machine and others that placed many aspects of the programming abstractions represented by ALGOL in their proper perspective.

REFERENCES

1. Gorn, S., "On the Logical Design of Formal Mixed Languages" (July, 1959).
2. ________"Progress Report Number 8," Computation Center, Massachusetts Institute of Technology, Cambridge 39, Mass. (January, 1961).
3. ________"The Descriptor, a Definition of the B-5000 Information Processing System" (Bulletin 5000-20002-P), Sales Technical Services Equipment and Systems Marketing Division, Burroughs Corporation, Detroit, Michigan (February, 1961).
4. Naur, P. (ed.), "Report on the Algorithmic Language ALGOL 60," Communications of the ACM, vol. 3, pp. 229-314 (1960).
5. Dijkstra, W. E., "Recursive Programming," Numerische Mathematik, vol. 2, pp. 312-318 (1960).
6. Huskey, H. D., "Compiling Techniques for Algebraic Expressions," Computer Journal, vol. 4, no. 1, pp. 10-19 (April, 1961).
7. Samelson, K., and Bauer, F. L., "Sequential Formula Translation," Communications of the ACM, vol. 3, pp. 76-83 (1960).

EDDYCARD MEMORY—A SEMI-PERMANENT STORAGE

Takashi Ishidate, Seiichi Yoshizawa, Kyozo Nagamori
Nippon Electric Co., Ltd.
Kawasaki, Japan

SUMMARY

A high-speed semi-permanent memory using closed-loop conductors as coupling media is described. Sense wire detects eddy currents in the closed-loop, which are induced by the drive current. Closed-loops on the Eddycard are cut out by a punch to store the information ZERO. Complete loops represent the information ONE.

After the discussion of basic fundamental results, 1024 word 45 bit experimental Eddycard memory system is introduced. It works at a read cycle time of 100 millimicroseconds in wide environmental conditions.

INTRODUCTION

It is already well known that a memory, or storage, is one of the important parts in electronic digital computers.

The memory can store both the numbers to be processed and the instructions which run the computer automatically. But the faster the computing speed becomes, the greater the necessity to minimize the access time of memories. Using thin magnetic films or tunnel diodes as memory elements, this problem has been partly solved, but they are mostly expensive and memory capacities in bits are generally so limited that full swing use of high speed memory is still not realized.

Recently, several papers concerning the permanent memory, or read-only memory, were introduced [1-5]. Although it is impossible to change the stored information so quickly, the permanent memory is a powerful solution to shorten an access time of the memory.

This paper deals with a high-speed semi-permanent memory "Eddycard"—a card on which conductors, carrying eddy currents, are placed according to the information to be stored.

The Eddycard memory is similar to the Unifluxor in principle that both make use of eddy currents as coupling media of the drive wire and the sense wire. However, in stead of a conductor slug in the Unifluxor, a closed-loop is used in the Eddycard. This fact makes it possible to analyze the behavior of eddy currents easily and to store the information by only cutting out a small portion of the closed-loop.

The behavior of this closed-loop conductor is discussed both in principle and example. In this paper, the results of experiments on a large-sized conductor are described in detail, i.e., the effects of a conductor size in reference to the distance between paired wires, the output levels against the displacement of the conductor from the normal position and etc. are introduced.

In the remainder, this paper describes an experimental Eddycard memory with a capacity of 1,024 words each having 45 bits, from which the information can be read in each 100 mμsec.

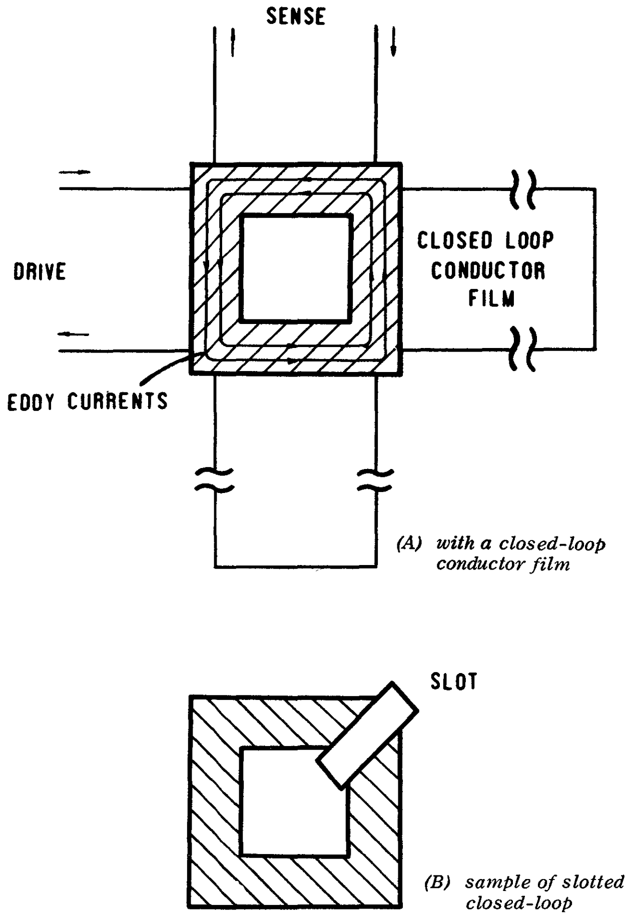
The operator of the card punches a small hole in the close-loops on the Eddycard according to the program, and lays the card upon the Eddycard panel. A cover is prepared to fix the Eddycard firmly at the right position so that each closed-loop conductor is placed just upon the intersection of the drive wire and the sense wire.

PRINCIPLE OF OPERATION

The principle of the Eddycard memory was accidentally discovered. At first, one of the authors was examining a semi-permanent memory using thin magnetic films electroplated upon each conductor. During the measurement of output signal waveforms against the thickness of a thin magnetic film, the contribution of a conducting device to the sense wire was discovered. After several experiments, the configuration shown in Figure 1(A) was considered to be reasonable and started analysis and designing an experimental Eddycard memory system of 1,024 words.

Although the Eddycard is similar to the Unifluxor in principle that both make use of eddy currents as coupling devices between drive wires and sense wires, the conductor configuration in this paper makes the theoretical calculations easy and makes it possible to write the information electromechanically.

The first experimental configuration is shown in Figure 2, and consists of a closed parallel drive wire crossed by an orthogonal sense wire and a conducting film. Without a conductor, as shown in Figure 3, there is no interlinkage of magnetic flux between two orthogonal wires. But if a conductor as shown in Figure 2(A), is placed at the intersection of the wires, eddy currents will occur in the conductor film and begin circulating as shown in Figure 2(A). These eddy currents will then induce an output voltage in the sense wire. Time relations of these waveforms are shown in Figure 4. Figure 5, shows a small scale model of the semi-permanent storage primarily experimented.



(A) with a closed-loop conductor film

(B) sample of slotted closed-loop

Figure 1. Configuration of Eddycard Memory

As the second step of the research, the distribution of eddy currents in a conductor film was surveyed. Since the mathematical analysis was so complicated and it seemed to be unnecessary for direct application of the phenomena to the eddycard memory, we prepared large-sized copper films for experiments, each having a square hole of different sizes in the center. Against the sizes of hole the output signal levels were observed, because it was thought that the output signal deviation caused by the hole would roughly indicate the contribution of the eddy currents flowing in the same area of the complete conductor. The relation of the sense wire outputs and the hole size is shown in Figure 6.

Figure 6, shows the fact that most of the eddy currents are concentrated to the peripheral parts of the conductor and the remainder contributes little or nothing toward carrying the eddy currents. Thus the closed-loop type was accepted for the final configuration of the Eddycard conductor.

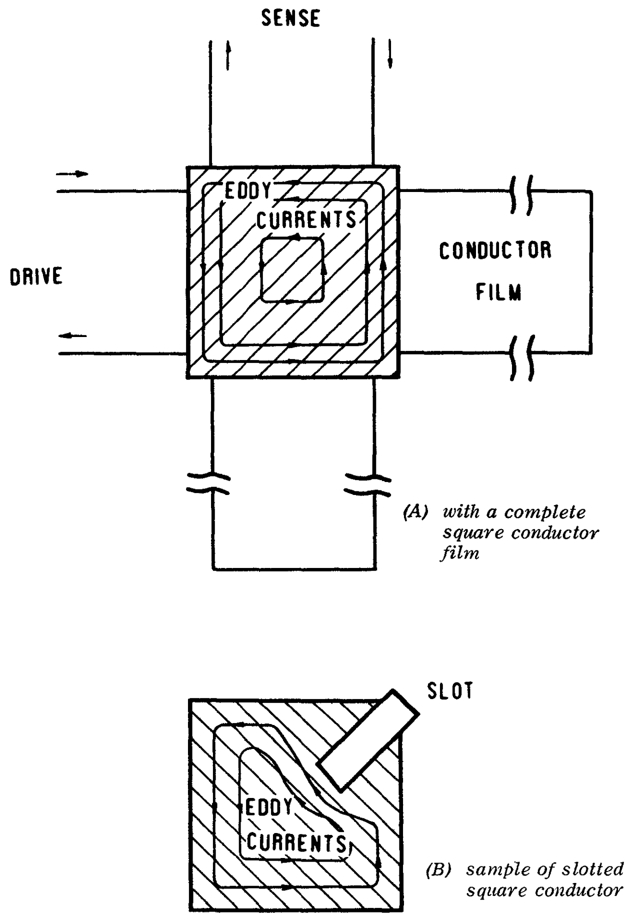

SENSE
DRIVE
EDDY
CURRENTS
CONDUCTOR
FILM
(A) with a complete square conductor film
SLOT
EDDY
CURRENTS
(B) sample of slotted square conductor

Figure 2. Primarily Experimented Configuration

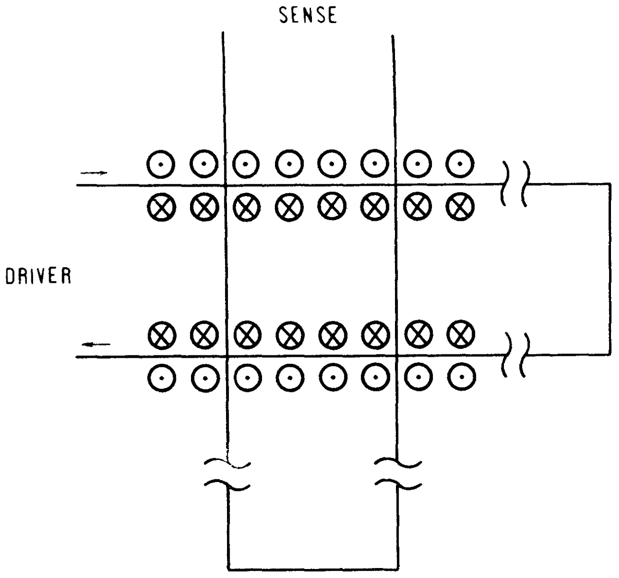


Figure 3. Magnetic Flux Without a Conductor

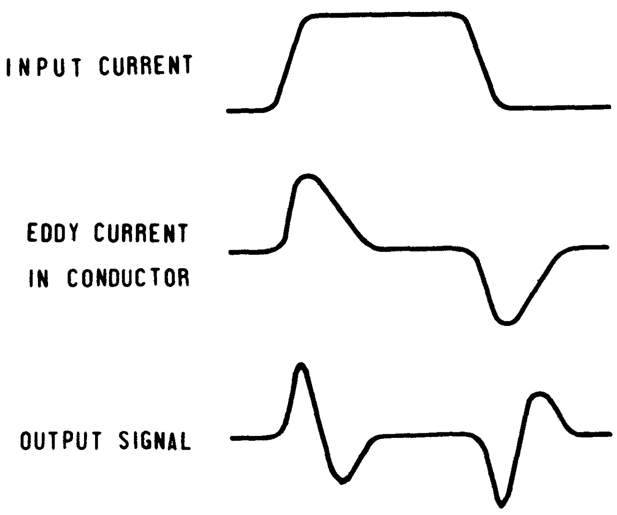


Figure 4. Time Relation of Waveforms in Figure 2.

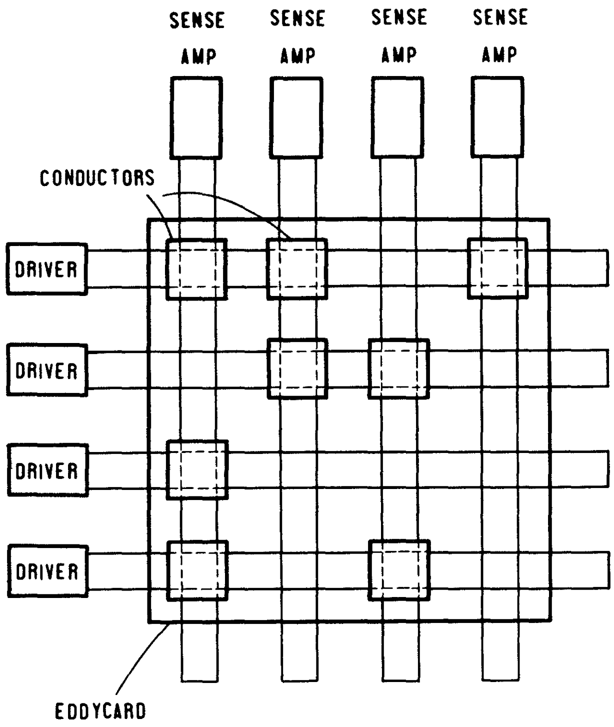


Figure 5. Simplified Model of Primarily Experimented Memory

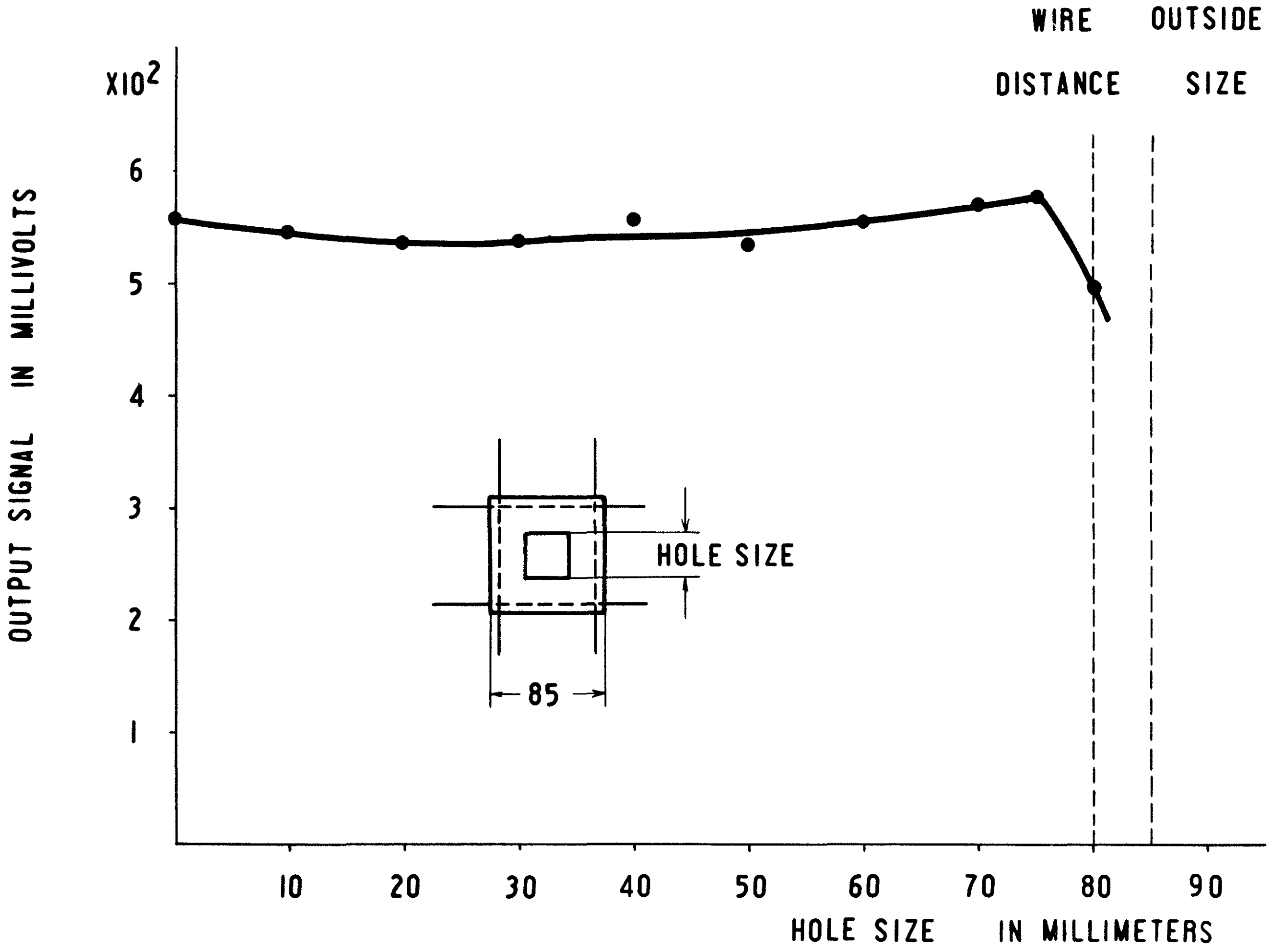


Figure 6. Effect of Hole Sizes to Output Signal

Write-in of the information was originally done by chemical method. Using an epoxy glass card on which square conductors were etched, an operator painted water-proof wax on the spots where the conductor film should be placed, and submerged the card in nitric acid solution for a moment and wasted out the unnecessary conductors. But this chemical processing took a long time and it was dangerous for an unexperienced person to process the card.

To punch out a conductor together with the base was considered as an alternate method of chemical processing. However, it was felt that the mechanical strength of the punched card would be very poor as the packing density increased, i.e., the space ratio of the conductors to the remainder of the card increased. If the information ZERO was to be stored all over the card, the strength would be extremely poor, especially for a thin card which shows excellent punch characteristics. Then the method was considered to write the information ZERO using as small a punched area as possible.

The closed-loop conductor, meets this problem. If the closed-loop is cut out by means of punch, there will be no circulating eddy currents, and consequently the information ZERO can be stored by a small punched slot at the limited portion of the loop.

Figure 1(B) and Figure 2(B) show samples of slotted conductor. In the conductor in Figure 2(B), eddy currents are still flowing. But no major eddy currents flow in the closed-loop conductor shown in Figure 1(B), though there may be minor local eddy currents which will not induce significant output in the sense wire.

Simple Equivalent Circuit

If a closed-loop conductor is used as coupling means, a simple equivalent circuit, shown in Figure 7, can be obtained. The circuit consists of a driver coil L_1, and Eddycard closed-loop circuit in which an inductance L_2 and a resistor R_2 exist, and a sense coil L_3. Between L_1 and L_2 there exists a mutual inductance M_{12}, and between L_2 and L_3 a mutual inductance M_{23} exists.

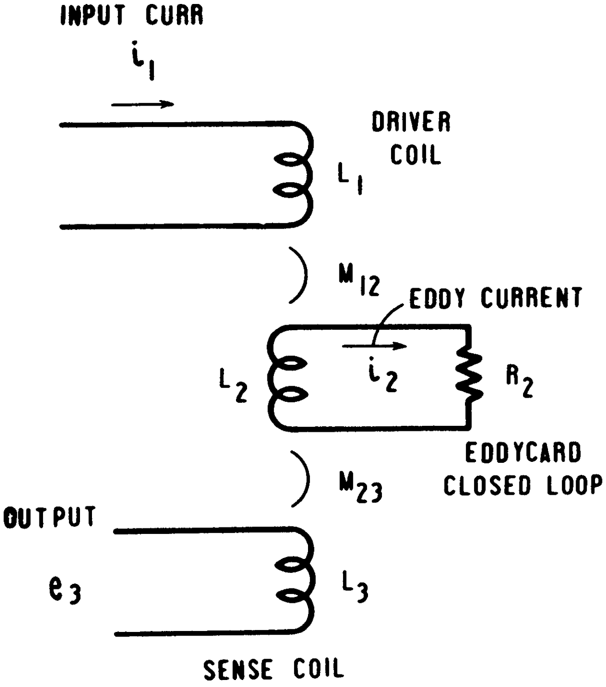


Figure 7. Simple Equivalent Circuit of Eddycard Memory

If an input current i_1 is fed to the driver coil L_1, the eddy-current i_2, expressed in Eq. (1), will flow in the Eddycard Loop.

$$L_2 \frac{di_2}{dt} + R_2 i_2 = M_{12} \frac{di_1}{dt} \quad \ldots\ldots (1)$$

And the sense coil output voltage e_3, induced across L_3, will be

$$e_3 = M_{23} \frac{di_2}{dt} \quad \ldots\ldots\ldots (2)$$

On the other hand, if we assume that an alternating current with an angular frequency of ω and an amplitude of i_1 is supplied to the drive coil, the output voltage e_3 is obtained as

$$e_3 = \frac{\omega M_{12}\, M_{23}}{R_2 + j\omega\, L_2} i_1 \quad \ldots\ldots (3)$$

Eq. (3) shows that for $R_2 \ll \omega L_2$ e_3 increases in accordance with ω and for $R_2 \gg \omega L_2$ e_3 is proportional to the square of ω. To examine this feature, the tendency of the output was calculated by Eq. (3) and compared with the observed results in Figure 8. A comparison between the output level determined experimentally and theoretically shows a good agreement in the lower frequency region. The peak in the experimental output is supposed to be caused by the increase of a loop resistance due to the skin effect.

The experimental results in Figure 8 were obtained with the 1024 word Eddycard memory system, applying an oscillator output. Output decrease in the high frequency region is not due to the response of sense amplifier.

Closed-Loop Conductor Including Circuit Elements

If a capacitor is included in the closed-loop, oscillatory currents will flow in the loop. This experiment was carried out with a large model in which a capacitor was connected in series. Though damping time varied by the time constant of loop circuits, 2-10 microseconds were easily obtained.

The results promise that signal-to-noise ratio will be improved if a capacitor included Eddycard is used, because the temporary storage of energy in the closed-loop will distinguish the coupling noise which ends soon after the termination of driver current.

One of other fundamental experiments was on a diode included loop. The diode inhibited backward eddy currents which occurred when the driver current ended. Using this type of closed-loop, considerably short cycle time was experimentally obtained.

EXPERIMENTAL RESULTS WITH LARGE SIZE MODELS

Experiments were carried out with large size conductors. Large conductors were prepared to avoid errors which might arise from gain fluctuation and limited frequency response of the high gain amplifier and from the unaccuracy in position by using a smaller one. Two amperes of 50 $m\mu$sec pulse was supplied from Rese 1200 Program Pulse Generator to a large size driver wire which

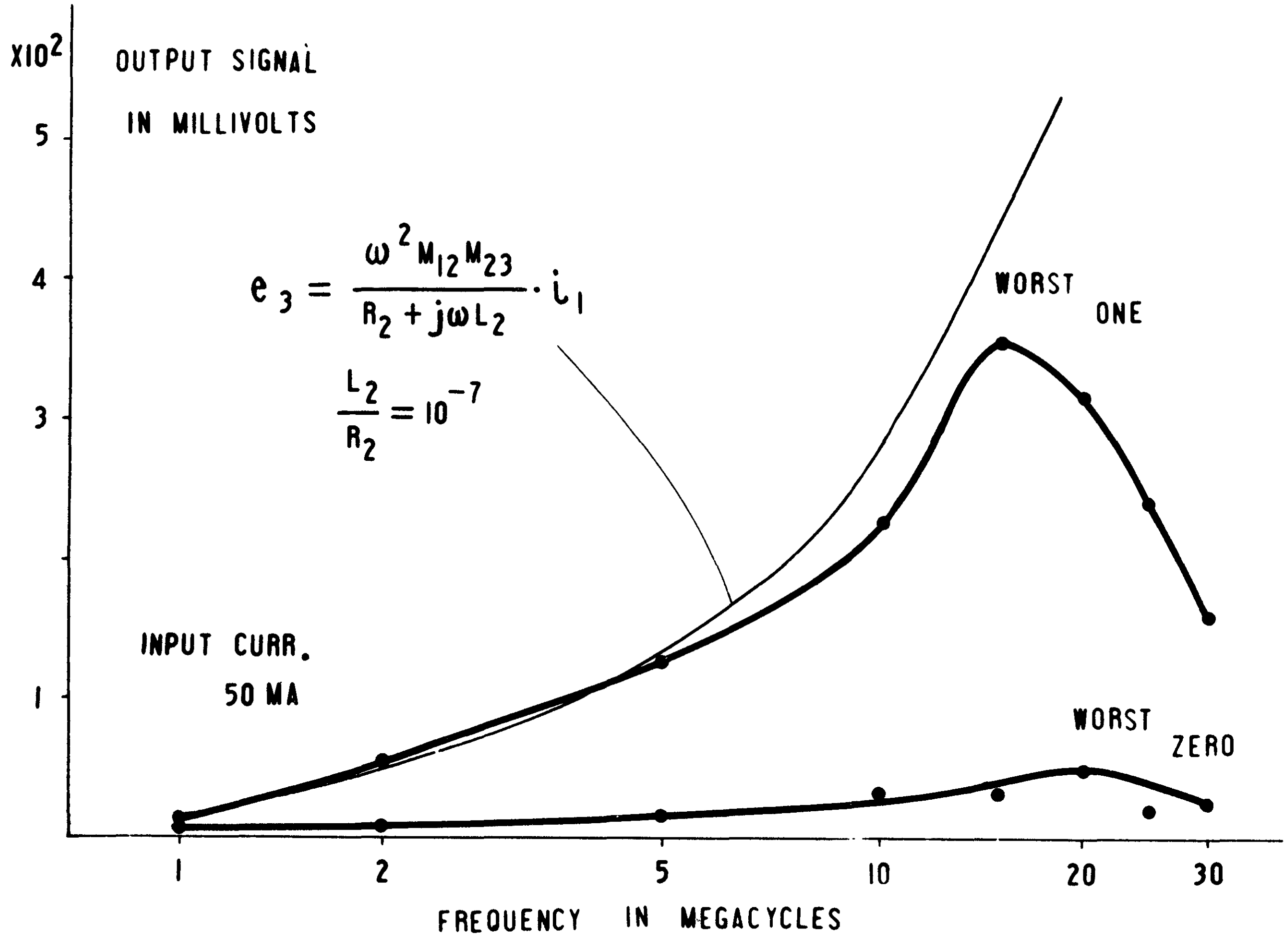


Figure 8. Frequency Characteristics

was made up of two parallel wires laid 8 cm apart. The sense wire, composed of two wires with a distance of 8 cm, was also laid across the driver wire. Output signals were measured by Tektronix 555 oscilloscope and a vacuum tube volt meter.

Optimum Conductor Size Against Wire Distance

In order to decide the optimum conductor size for a certain distance between the paired wires, output signal levels were observed by changing the size of a complete conductor square film. The results, plotted in Figure 9, show that the output signal reaches its maximum value when the conductor size becomes slightly larger than the intersection of the driver and sense wires. The output signal decreases quickly inside the wire and slowly outside the wire. Although the results of Figure 9 were obtained by pressuring a conducter upon the wires, the output signal maximum point would be far beyond the edge if the gap between conductor and wire plane surface increased. This will be discussed again in the following description.

Permissible Displacement

As described above, experiments revealed that a conductor which just covered the intersection of drive and sense wires generated the maximum output signal in the sense wire. The authors, however, expected that such a conductor would require a severe tolerance of the conductor position to the wires. Thence the relations of output signal and displacement from the optimum position on the wire surface were measured with different size conductors, and compared in Figure 10 and Figure 11. Conductors were displaced along the drive wire as well as the sense wire, but there were no significant differences between two cases.

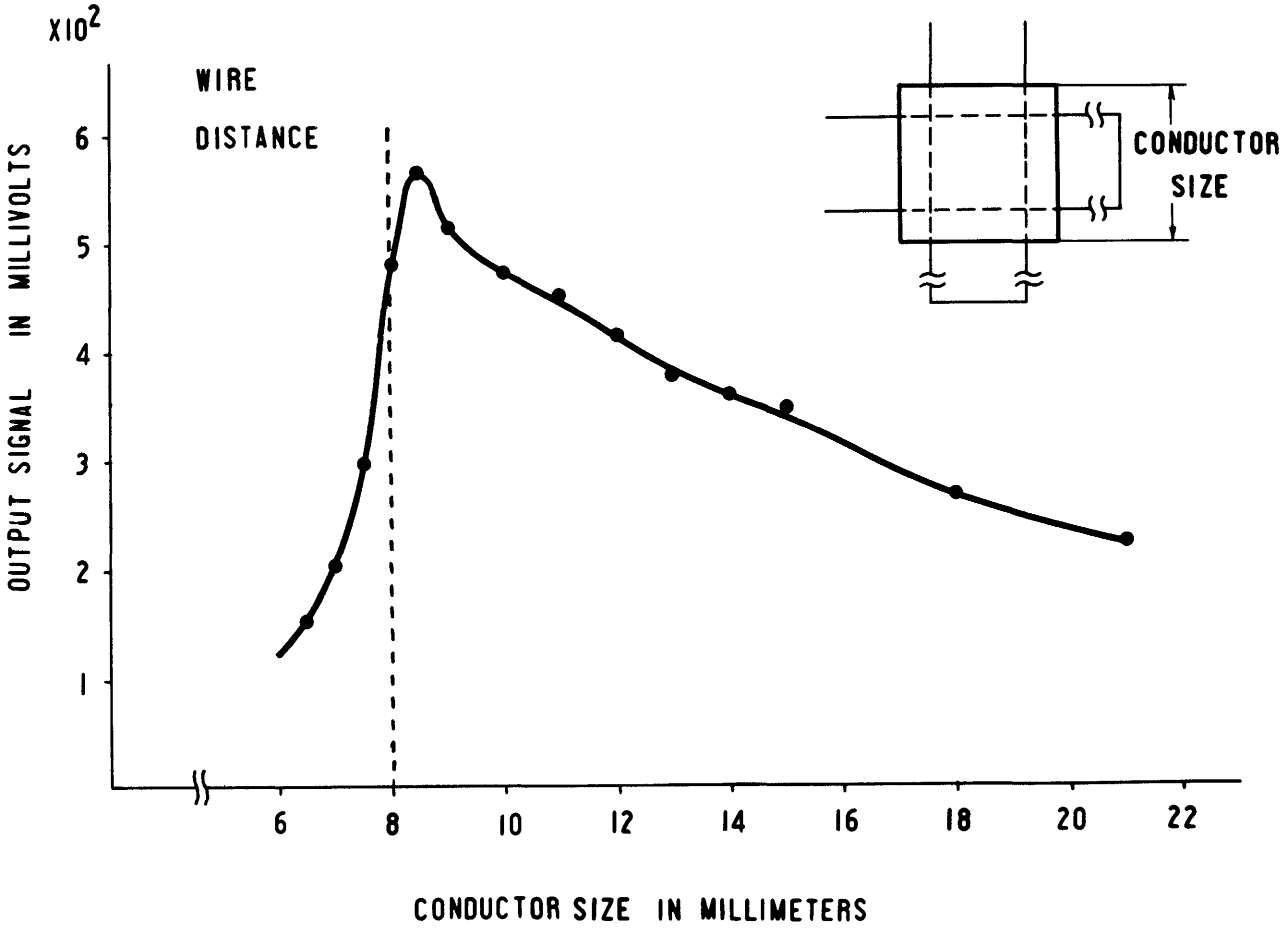


Figure 9. Conductor Size vs. Output Signal

In Figure 10, the solid line represents 8.5 x 8.5 cm complete square copper film, and the dashed line is for closed-loop type copper film, whose outside is 8.5 x 8.5 cm and inside is 7.5 x 7.5 cm. The figure shows that the closed-loop type induces more output at the right position, but the output decreases faster than that of complete conductor as the displacement increases.

Results of another experiment using larger conductors are shown in Figure 11. Measured samples were a 10.5 x 10.5 cm complete conductor and a 10.5 x 10.5 cm outside 9.5 x 9.5 cm inside closed-loop conductor. If a complete square conductor is used, Figure 11 shows that there exists a region within which the displacement causes little output signal decrease.

For a closed-loop type, the figure shows the output signal increases as far as a certain displacement where one side of the loop comes just upon the wire.

If the permissible displacement is defined as the distance at which the output signal decreases to 3 db below the maximum value, Table 1 will be obtained from the figures. Values in parentheses are ratios of permissible displacement to conductor size in percentage.

Table 1. Permissible Displacements

Size / Type	85 x 85 mm	105 x 105 mm
Complete Square	10 mm (12%)	18 mm (17%)
Closed-Loop	6 mm (7%)	18 mm (17%)

Output signal level also depends upon the gap between the conductor and wire surface. In Figure 12, the relations were plotted using

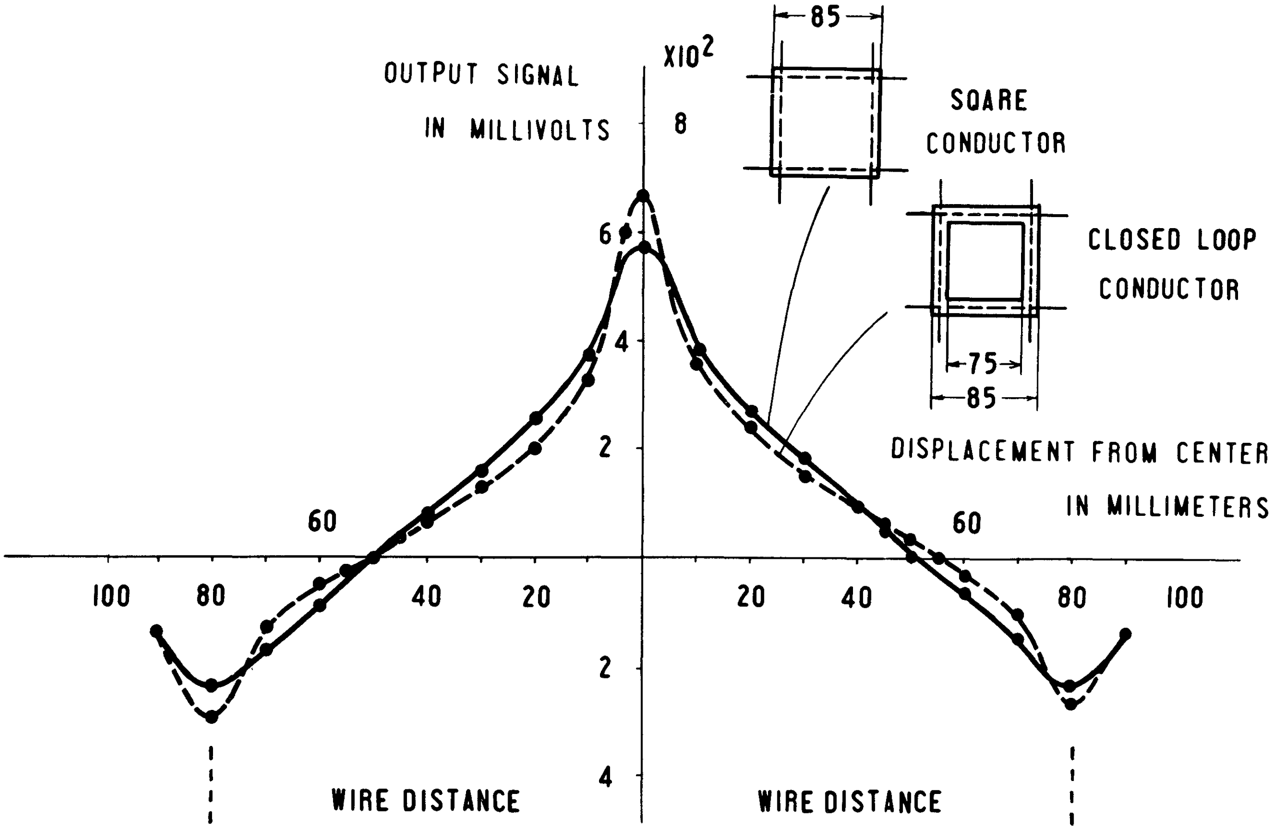


Figure 10. Displacement vs. Output Signal for 8.5 x 8.5 cm Conductors

three kinds of conductors. A large conductor showed lesser gap effects, though its output signal at the wire surface was lower than that of smaller conductors.

1024 WORD EXPERIMENTAL EDDYCARD MEMORY SYSTEM

General

Based upon the preliminary experiments, an 1,024 word 45 bit memory system has been constructed. The system employs linear selection read out with a cycle time of 100 millimicroseconds. The block diagram of the system is shown in Figure 13.

As shown in Figure 13, the system consists of four units. The selection core matrix supplies a drive current to selected one of 256 words of each unit simultaneously, and consequently outputs are obtained from four units at the same time.

The strobe pulse is provided to improve signal-to-noise ratio as well as to select the wanted channel from four outputs.

Memory Panel

A memory panel consists of 128 drive wires and 45 orthogonal sense wires. Sense wires are etched on a thin epoxy glass film, with a thickness of 0.1 mm, cemented upon the printed drive wire panel. The sense wires of two panels are connected in series to compose a 256 word unit. The length of sense wires per one amplifier is limited so that the propagation delay in a sense wire may be negligible compared with the access time.

Using eight panels, or four units, a memory stack is constructed. Its dimensions are 27.5 by 63.5 by 33 centimeters excluding the selection core matrix.

A sponge cover is prepared to pressure the Eddycard firmly upon the memory panel surface. Positioning is determined by two holes of the Eddycard.

Figure 14 shows a memory panel with a cover removed and the Eddycards placed upon it.

Eddycard

An experimental Eddycard was made of 27 cm long, 3.3 cm wide and 0.4 mm thick

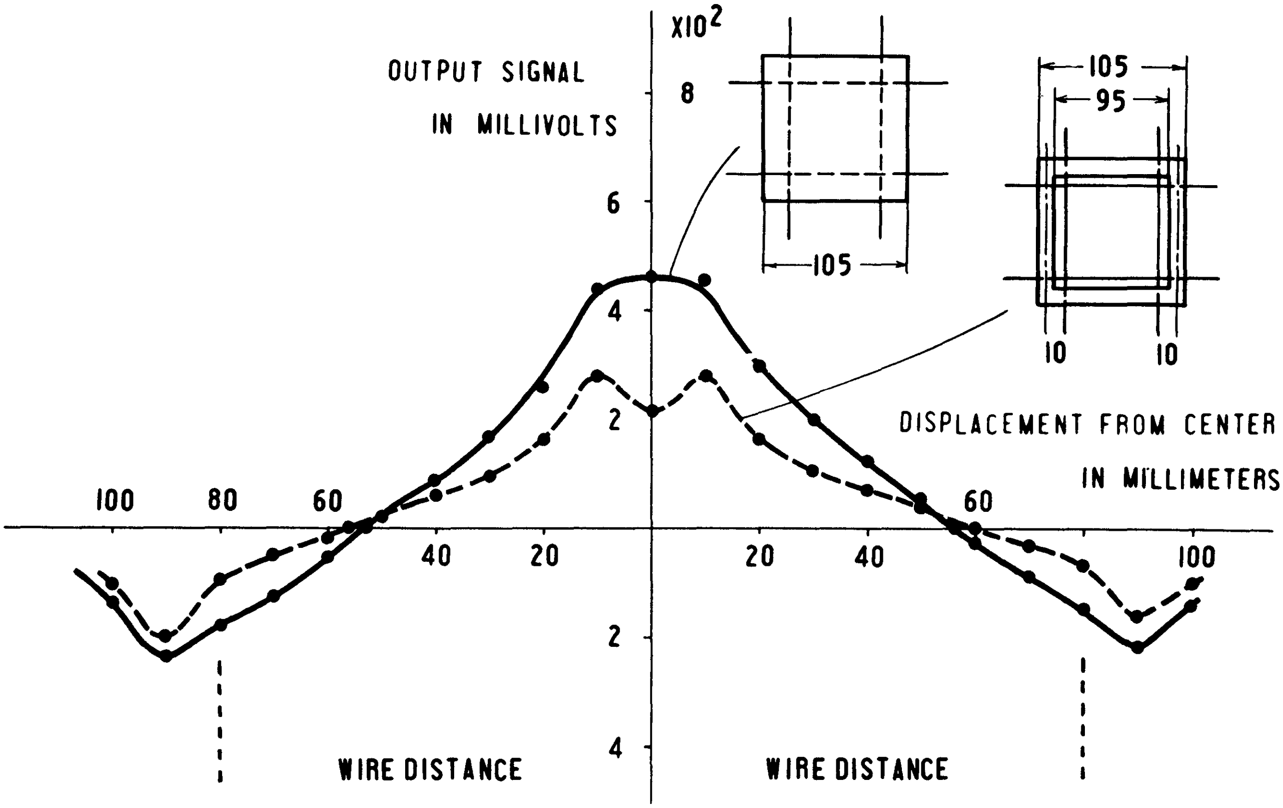


Figure 11. Displacement vs. Output Signal for 10.5 x 10.5 cm Conductors

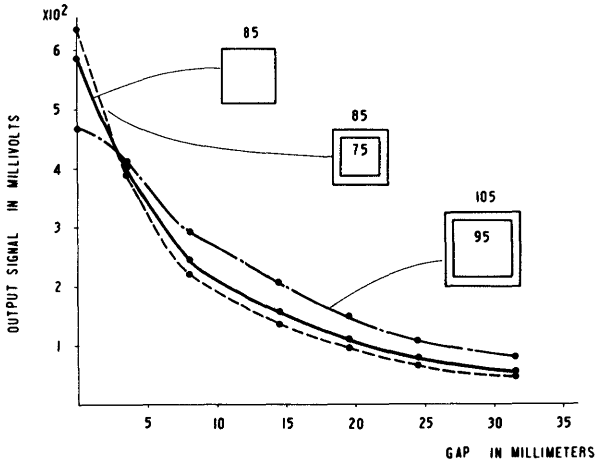


Figure 12. Gap vs. Output Signal

epoxy glass sheet upon which silver coated copper conductors were etched. The card contains 45 by 8 closed-loop conductors, 3 mm square outside and 1.2 mm square inside, on one side. Conductors are spaced 4 mm apart from each other measured between the centers, but the marking spaces are provided to separate the 45 bits of each word into nine 5-bit characters to assist the operator. Relative address numbers are also printed at the left side of the conductor group, and in the right side there are etched tabs on which the operator can describe the absolute address numbers and memos of the contents. On the other side of the card, etched tabs are provided for the card number and program identification.

Figure 15 shows the photograph of Eddycards, one is placed the conductor side up and the other is placed the conductor side down. Close-up view of the punched Eddycard is shown in Figure 16. In Figure 16, closed-loop conductors are cut by circular punched holes. This configuration was accepted by convenience.

Sense Amplifier

The sense amplifier for the Eddycard memory consists of four differential amplifiers and a gate circuit. Differential amplifiers are used to reduce the noise due to the capacitive coupling between drive and sense wires. They linearly amplify the sense wire outputs of their own from 1 millivolt level up to approximately 0.2 volts. Transistor

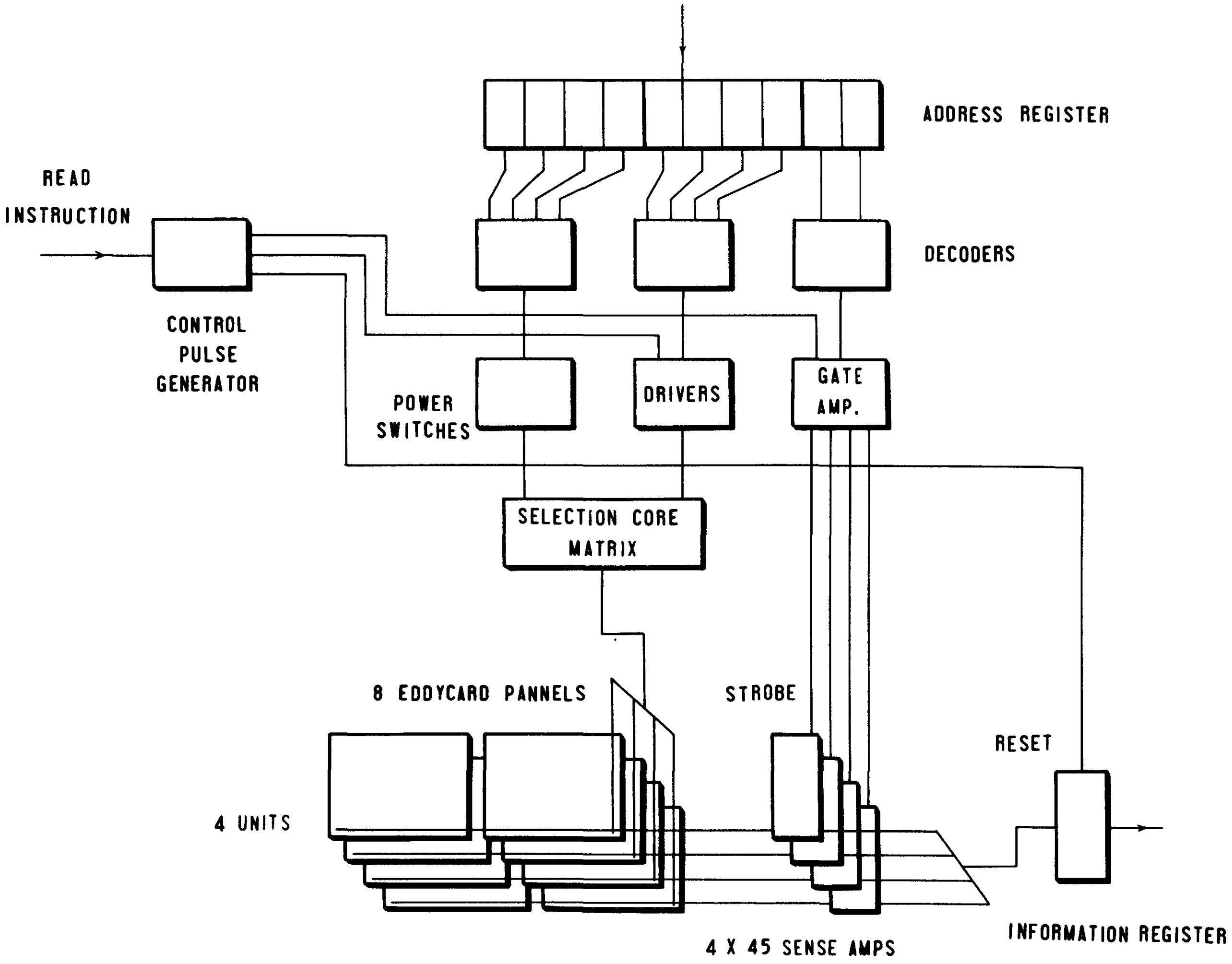


Figure 13. Blockdiagram of 1024 Word Eddycard Memory System

2SA244's comprise a negative feedback amplifier with a gain of 46 db at middle frequencies, and the gain versus frequency curve falls off by 3 db at 30 mc.

The outputs of the linear amplifiers are then applied to the second stage, a gating circuit where 30 mμsec strobe pulse selects one out of four amplifier outputs.

The strobing is also used to eliminate the noise from near-by Eddycard loops. The eddy currents in neighbouring Eddycard loops induce the output to the sense wire, but of opposite polarity. This effect reduces ONE output to some extent and increases ZERO output, or noise. Since ZERO output waveform, noise from near-by loops, is out of phase compared with ONE wave-form, the discrimination can be done by a time selection gate.

In Figure 17, (A), (B) and (C) show the differential amplifier outputs of worst ONE, worst ZERO and without a card, respectively. The Waveforms in Figure 17 were viewed on a Hewlett-Packard 185A sampling oscilloscope with a 187A preamplifier. The time bases were 10 millimicroseconds per division, and the voltage axes were 100 millivolts per division.

The gated signal will be then sent to the corresponding information register which can be switched its states in a few millimicroseconds.

Driver

A driver wire is driven by turning on a corresponding driver and switch combination. At the intersection of the selected driver line and switch line, the diode conducts and

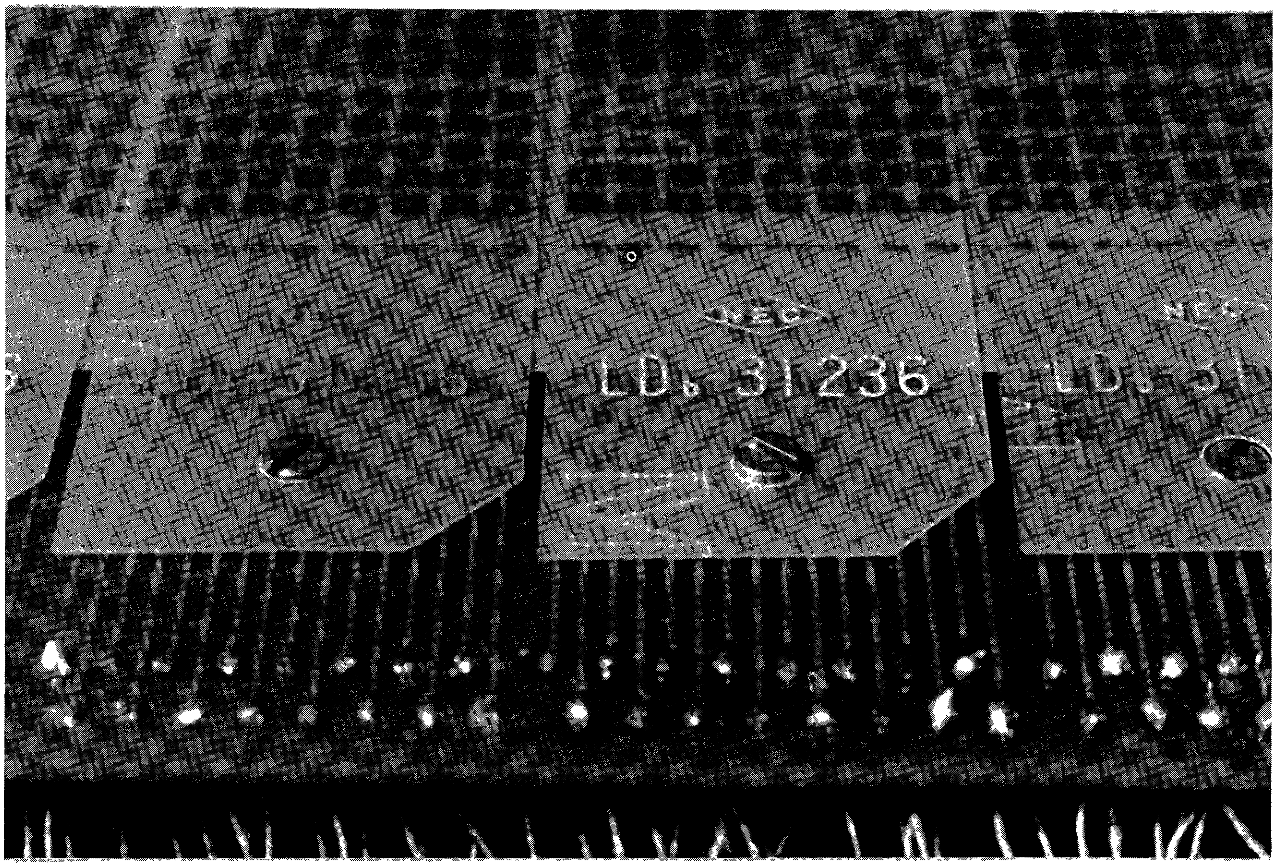


Figure 14. Photograph of Memory Panel with a Cover Removed

supplies a current pulse to the selected drive wire via a coupling transformer. A coupling transformer with a frequency response extending from 1 mc to nearly 50 mc, is used to reduce the capacitive coupling noise to the sense wire.

The current driver delivers a 300 ma pulse with a rise time of 20 mμsec and 50 mμsec wide. As shown in Figure 18, the driver is comprised of four 2SC30 Si mesa transistors connected in parallel at the last stage. They are used base-grounded so that the driver may be considered as a constant current supply.

Eight additional transistors are required in the circuit, four 2SC30's and two 2SA244's, Ge mesa type, for amplification and a 2SA244 for an input emitter follower.

Experimental Results

The worst signal-to-noise ratio of 3:1 was obtained, the noise being mostly caused by inevitable unbalance of differential amplifiers. Maximum propagation delay in the sense wire was measured as 12 mμsec, so strobe gates were actuated at the time 15 mμsec after the leading edge of the drive current.

Measurement of permissible displacement with actual Eddycard revealed that the sense amplifier output decreased to one half of its normal value, when the card was displaced 0.8 mm on the panel surface or gapped 0.5 mm with the panel.

In order to minimize punched area, slot has been recommended as the punch configuration. In this experiment, however, holes were used because the card was sufficiently strong.

FOR FASTER PERFORMANCE

For never-ending desire of faster cycle time, experiments with single cell have started to realize 50 mμsec read cycle time. Experiments revealed that driver and sense amplifier mainly limited the cycle time to about 100 mμsec. Coupling noise, time delay and ringing in wires, delays in decoding circuits and registers were also hard troubles

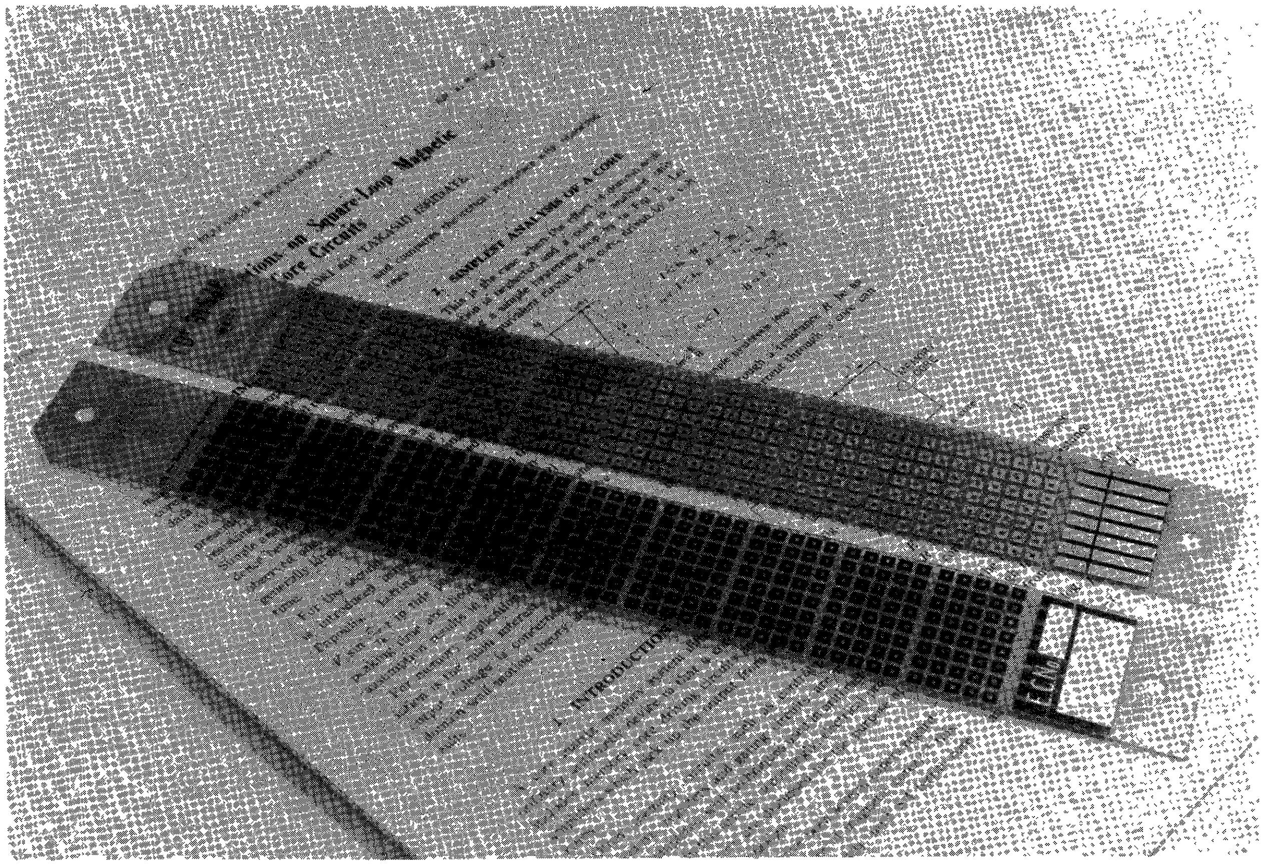


Figure 15. Photograph of Eddycards

in experimental circuits for faster performance.

However, drive and sense circuit together with decoders and registers will be improved with a remarkable progress in semiconductor devices, and time delay and ringing in wires will be reduced by micromodule techniques now being advanced. Then it can be concluded that the Eddycard memory with 50 mμsec cycle time will be constructed in years.

CONCLUSIONS

The Eddycard memory—a semipermanent storage has been described. This memory offers a number of advantages. The closed-loop type conductor permits the use of a simple equivalent circuit, thus giving considerable promise in various applications. Small punch area on the card makes it possible to write the information with a high-speed punch, which can be electrically connected with other information sources.

An experimental Eddycard memory system with a capacity of 1,024 words 45 bits each has been constructed after the fundamental researches. Results indicate that a read cycle time of 100 millimicroseconds is feasible.

Although problems are left to increase the packing density, because of its simplicity in basic principle, the Eddycard memory works at high repetition rates under wide environmental conditions.

ACKNOWLEDGMENTS

The authors wish to express their thanks to Tsuneji Koyama and Toshikatsu Tanaka for their assistance in obtaining the experimental data and designing the peripheral circuits.

REFERENCES

1. D. H. Looney, "A Twistor Matrix Memory for Semipermanent Information," Proc. W.J.C.C., pp. 36-41, 1959.

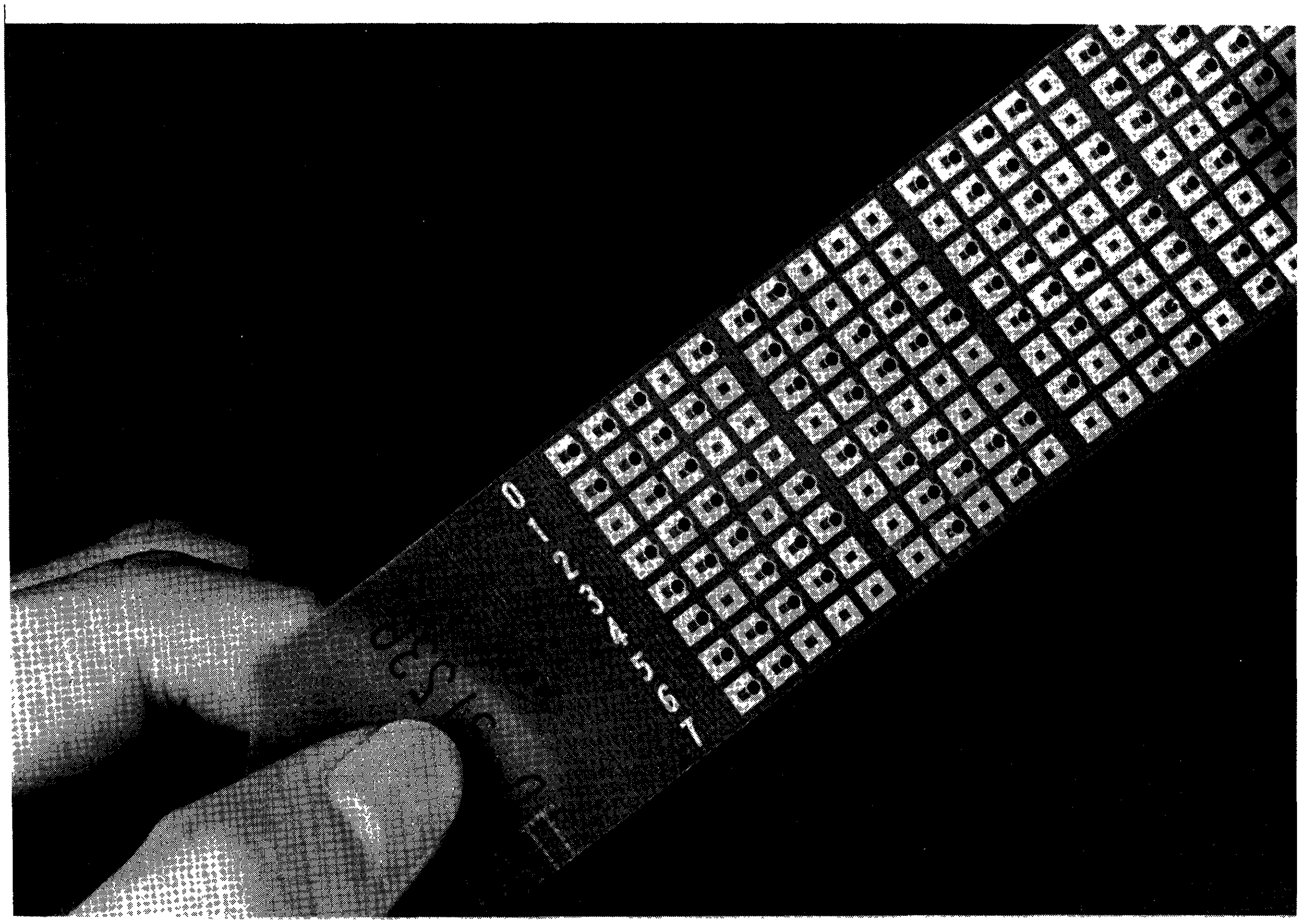
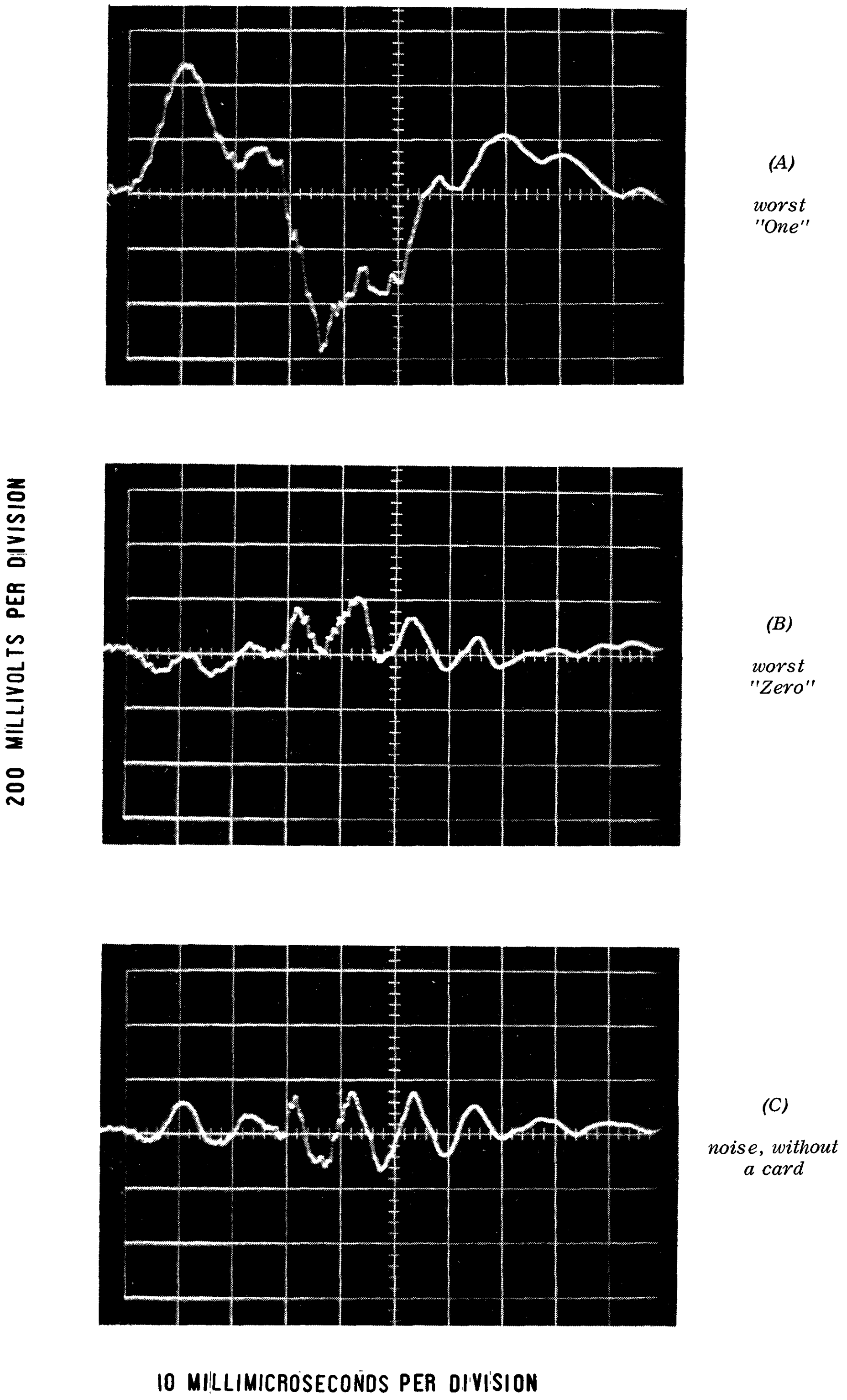


Figure 16. Close-up View of Punched Eddycard

2. J. J. DeBuske, J. Janik, Jr. and B. H. Simons, "A Card Changeable Nondestructive readout Twistor Store," Proc. W.J.C.C., pp. 41-46, 1959.
3. T. Kilburn and R. L. Grimsdale, "A Digital Store With Very Short Read Time," Proc. I.E.E., 107 Pt. B. 36 pp. 567-572, Nov. 1960.
4. A. M. Renard and W. J. Neumann, "Unifluxor: A Permanent Memory Element," Proc. W.J.C.C., pp. 91-96, 1960.
5. H. R. Foglia, W. L. McDermid and H. E. Peterson, "Card Capacitor—A Semi-permanent, Read Only Memory," IBM J. pp. 67-68, Jan. 1961.



(A)

worst "One"

(B)

worst "Zero"

(C)

noise, without a card

Figure 17. Waveform of Sense Amplifier Output

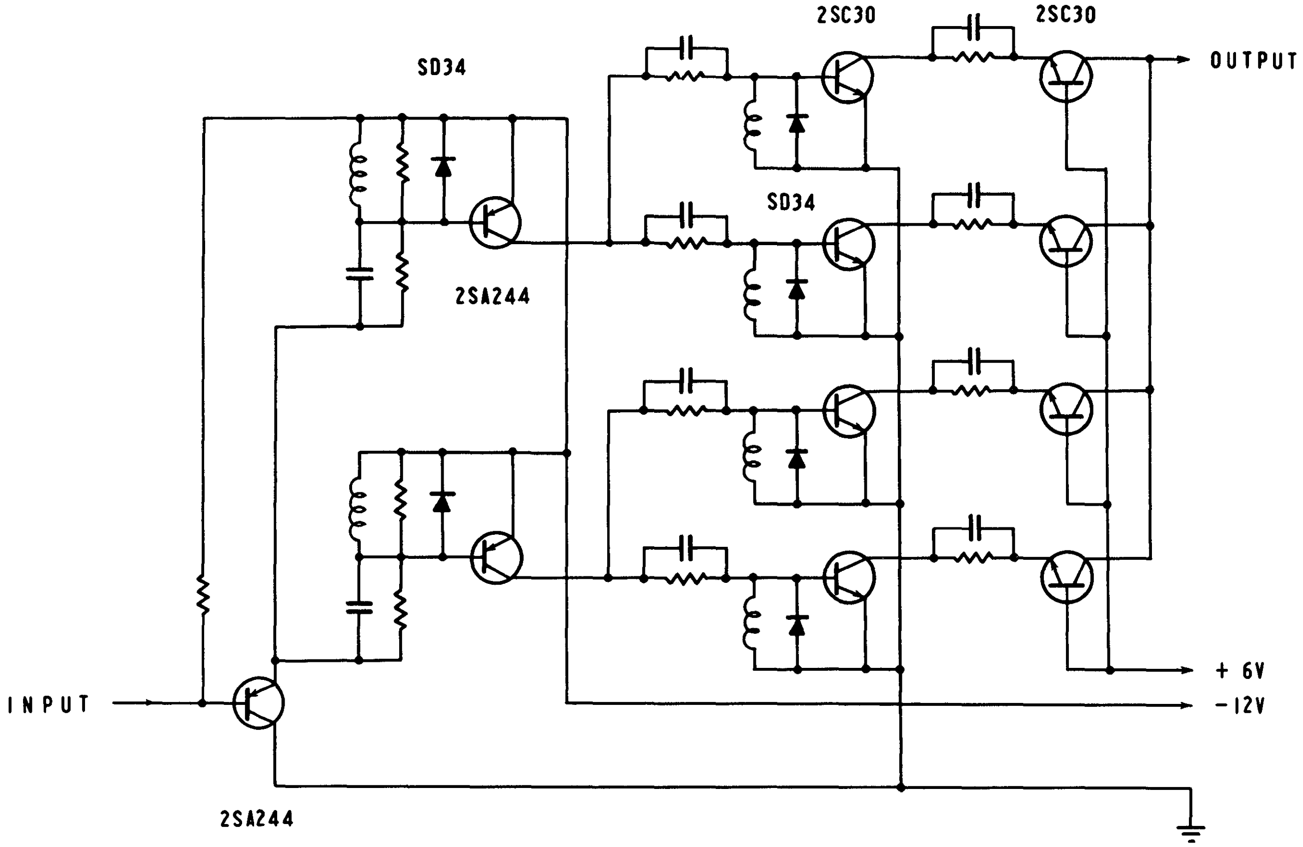


Figure 18. Schematic Circuit of Drivers

DIGITAL DATA TRANSMISSION: The User's View

Justin A. Perlman[*]
Hughes Aircraft Company
Culver City, California

The state of the art in digital transmission theory, technology, hardware, and use has made rapid strides in the last two years. Today the user can choose from among hardware available in several speed regimes provided by a growing number of manufacturers. A user with an immediate requirement for data transmission can procure hardware which will pass data at his desired rate, if he is willing to pay (in most cases) a premium rental and accept minor difficulties in use. In essence, brute force solutions and some semi-sophisticated answers are here. Users will have only themselves to blame if the current propitious beginnings are not greatly extended to provide transmission equipment more in consonance with their practical requirements for greater capability and flexibility, at sharply reduced cost.

Toward the end of making user requirements known to suppliers, members of eight major firms in the aerospace industry met in July 1960 to establish an informal Data Transmission Study Group. Objectives of the group are exchange of knowledge, standardization of terminology, discussion of total systems requirements, development of reasonably uniform requirements for equipment and service, forecasting of long range needs, and acting as a coordinating group for these users with the common carriers and equipment manufacturers. Both data collection and data transmission are covered by the Study Group.

Members of this group are faced with the wide range of problems inherent in providing scientific computing and business data processing services to multi-plant research, development, and manufacturing organizations. It has been enlightening to note the similarity of data transmission needs present in the group. These needs can be described in four categories:

a. Load-sharing among major computer centers (i.e., centers with capacity equivalent to one or more 7090s).

b. Data pick-up from remote test sites (or from airborne tests). In some cases real-time processing and re-transmission of results to the test site would be desirable.

c. Providing access for Plant A to a computer center at Location B. Plant A might have a medium-scale, small-scale, or no computer of its own.

d. Data pick-up from dispersed plants and offices for processing and incorporation in overall reports. The dispersed points might be in the same locality as the processing center, or possibly as much as several thousand miles away.

These needs have been listed in descending order of volume and speed requirements. In the aerospace industry, load sharing among a company's computer centers (category "a", above) can generate very large daily transmission volumes, in some cases on the order of 50 reels of IBM low-density magnetic tape, or more. There is usually a distinct economic

[*] Now General Manager, Modal Systems, Inc., La Jolla, California

and/or intangible benefit obtainable from rapid transmission.

The value of rapid transmission and processing of test data to allow decisions to be made while tests are in progress has already been proven in the development of jet engines and rocket motors. Depending on the types of tests being conducted, and the sophistication of the instrumentation, fairly large data loads can be generated for short periods of time. The need for real-time transmission and processing will probably increase in the future when it becomes feasible to continuously monitor flight tests with a remotely located multi-processing general purpose computer and have the computer control the test sequence according to test data received.

Category "c", providing access for Plant A to a computer center at Location B, can generate a medium or light load depending on the size of Plant A, computer equipment installed at A, sophistication of its information system, and the research and engineering content of work accomplished at A. These last two factors require some amplification. Sophistication of the plant's (and the company's) information system can be a major determinant of the amount of processing required both at the plant and at a separate computer center. Generally speaking, brute force information systems (which are frequently just computerized versions of manual or tab systems) generate greater transmission and processing loads than sophisticated systems which have been designed to take full advantage of a computer's capabilities. In part, this extra load is due to the need in a brute force system to manipulate the full detail of each transaction, rather than only the changed information. Several companies in the aerospace industry are now experimenting with pilot installations of large random access information systems. These sophisticated systems should eventually reduce both transmission and processing loads.

The research and engineering content of work accomplished at Plant A can influence transmission load even if the plant has its own medium-scale computer. Many technical areas now being ventured into will require computing and storage capability in excess of what could normally be justified at Plant A. Problems in magnetohydrodynamics are only one example of work which might be more suitably handled on a single large-scale computer within a multi-plant company. Although transmission loads for these problems would not be large, the ability to obtain solutions quickly to facilitate rapid advancement of the scientific investigation would be highly desirable.

Although the basic categories of transmission needs described above have been cast in terms familiar to the aerospace industry, similar needs exist or will soon exist in a variety of other industrial and commercial organizations. While these basic needs will be present, the specifics of each case will determine how much a high-speed, medium-speed, or low-speed solution is worth, and whether equipment exists to do the job.

This is the point at which user groups such as the Data Transmission Study Group in the aerospace industry can play a significant part in helping focus the development of data transmission techniques and equipment toward more flexible and more economic solutions than those which are available today. To do this, however, users must look ahead with greater care to their future requirements, try to establish a reasonable degree of similarity in these requirements to those of other users (where possible), and then insist that suppliers either meet these requirements or provide the desired flexibility and economy by alternate means.

This type action by users actually provides several benefits to the suppliers of data transmission equipment. Introduction of any new product requires extensive (and expensive) market research, product planning, etc. Typically, even after this is accomplished, the manufacturer is faced with conflicting statements of requirements by several of his initial customers. Whether the manufacturer decides to slant the product toward a single customer or attempts to compromise the differences, the usual result is an initial product of limited utility which cannot be priced attractively due to its uncertain marketability. Coordination and statement of reasonably uniform requirements by a group of potential users can help reduce the manufacturer's development expense and his uncertainty as to product design and marketability; this should yield a more suitably designed device, available to users sooner and at a more desirable price. Moreover, the user will normally receive another benefit: by reducing the uncertainties and some of the risks in entering the data transmission market, a number of capable small companies may find it feasible to develop transmission devices, thus providing users increased freedom of selection.

Now, how does a user go about defining his requirements and what are some of the questions he must resolve?

First, he must be thoroughly familiar with his total information system: as it exists today, and as he plans it will evolve one year, three years, and five years hence. While the modes of processing and the general types of processing equipment planned for use are important, of equal importance are the types of demands which will be placed on the system and the timeliness with which they must be met.

Second, he should define his current costs of moving data from one location to another. It would be well to state here that one of the most effective and economic means of data movement is still physical transfer by messenger, mail, or air freight. Large volumes of data can be moved quickly, and there is no problem as to the accuracy of the received data. If higher speeds are required in a local area, movement by helicopter or light plane should be investigated along with transmission by electrical means.

Third, actual or estimated volumes of data currently requiring movement should be recorded, together with projections of future loads. It is important at this point to list the medium, format, and coding of the data before and after movement, and the time urgency of each type.

A difficult fourth step is to try to assess the tangible and intangible "opportunity" costs of not moving the data, and of moving it at say 1000, 3000, 20,000, and 500,000 bits per second. It is fairly simple to record the cost of unutilized prime shift computer time at one location vs. second shift time used elsewhere; it is quite difficult, though, to attribute a cost to forcing a research scientist to wait an extra day to receive problem solutions from the computer. It is wise to attempt to ascribe values to these factors, however.

Next, the actual equipment and operating costs of various devices for physically moving and electrically transmitting data should be compared. Here the potential user will find himself faced with a variety of alternatives, each with different cost implications. For example:

a. Physical movement vs. wire transmission vs. grouped channel (Telpak) vs. private microwave transmission. Each of the electrical means has different costs associated with different speeds of data transfer.

b. Present data medium (punched cards, paper tape, magnetic tape, computer storage). Depending on the medium, format, and coding of the data to be transmitted, is it desirable to convert to another medium, format, or coding for transmission?

c. Serial vs. parallel transmission. This can have a bearing on future expansibility of the transmission system to handle different data coding and different transmission speeds. Signal-to-noise ratio and consequent error rates in transmission may also be affected by this choice.

d. Buffered vs. unbuffered transmission. This choice can have a major effect on the actual rate of data transfer vs. the nominal rate of transmission.

e. Accuracy required of received data. The instinctive requirement for 100% accuracy is usually costly, and frequently unnecessary. A realistic appraisal should be made of the reliability required in the output. If very high reliability is needed, is it sufficient to be able to recognize and mark received errors rather than correcting them during the initial transmission? How can these errors be corrected subsequently at low cost?

f. If high accuracy is required at the time of transmission, what type of error detecting - error correcting code - automatic retransmission scheme is necessary? Higher orders of error correcting means become increasingly expensive both in hardware, and in transmission and checking of the non-information bits.

g. Are synchronization or other non-information bits required in one system to a greater extent than in competitive systems?

h. Automatic monitoring and switching of inoperative equipment sub-systems may be expensive in terms of hardware but economic in terms of system utilization.

i. Off-line vs. on-line operation of the transmission system, and ease of change. Programming difficulties and tie-up of expensive computer installations can result from a poor choice here.

j. In the case of grouped channel or private microwave operation, availability of bandwidth for other services: telephone, teletype, computer inquiry, low and high-speed facsimile, closed-circuit TV.

This is merely a sampling of the factors to be weighed in selecting the most suitable data transmission system for a particular application. Some general statements can be made, however, as to user-desired features of an ideal data transmission system.

The system should be capable of accepting data via the medium, format, and coding in which the data is currently recorded. Normally, the transmission equipment should not require the user to make modifications to his computer or peripheral hardware or to develop "marriage boxes" between this hardware and the transmission system. Minor modifications to the user's existing hardware may be acceptable if made by the hardware or the transmission system supplier. The transmission system should come equipped with a power supply or a plug-in means of obtaining power from existing hardware.

Data transmission systems should be readily adaptable to work with peripheral hardware of several manufacturers rather than merely that produced by the transmission system supplier. Excessive duplication of equipment should not be necessary here.

Considering the data terminal and transmission link portions of the system as separate entities, both should be modifiable over time to suit a user's needs precisely. For example: a company's first experience in data transmission may involve sending data from one location which has only paper tape or card capability to another location for processing. Presumably, only low-speed operation over a single wire pair would be necessary. As the remote location grows, volume would increase, making higher speed transmission necessary. A modularly-designed flexible data terminal might allow additional modules to be added to provide transmission over several circuits.

As the remote plant grew further, at some point magnetic tape capability might be added. This would require changing some of the logic and other cards (printed-circuit boards) in our ideal data terminal. As volume increased, a grouped-channel transmission link might be required, possibly with a different error correcting scheme, necessitating further changes in the modular data terminal. If volume subsequently grew still further, a common carrier or private microwave link might turn out to be most suitable and would require adjustments in the terminal. Changes in computer equipment, magnetic tape formats, etc. at the computer centers should be just as easily accommodated.

Conversely, if due to improvements in the information system (or a decline in business levels) the transmission load decreased, this ideal transmission system would be susceptible to orderly decreases in capability and cost.

Today, this fully flexible terminal and transmission link do not exist. In the foreseeable future, flexibility over the speed range of a thousand bits to several megabits per second will probably remain an unrealized ideal. Several suppliers are working on this type of flexibility in more limited speed ranges, however, and such equipment should become available within the next three years. The Data Transmission Study Group in the aerospace industry would like to encourage formation of other interested user groups to help focus this development work toward practical user needs. This can be an effective, inexpensive way to insure that your needs are met in an advantageous and timely manner.

TELE-PROCESSING* SYSTEMS

J. D. Shaver
International Business Machines Corporation
Data Systems Division
White Plains, New York

Data transmission's impact on information processing has started a new evolution, perhaps, a revolution, and the next few years will clearly show how it will enhance the scientific operation of a business. Data transmission is not new; in fact, it is quite old by today's measurements. In May of 1844, the first data was transmitted over public telegraph by Samuel Morse. In June of 1876, at the Philadelphia Centennial, the first words were transmitted by public telephone. Both mediums, telegraph and telephone, transmit and receive information - data - and are commonly accepted today as a way of life. Yet the full implications of communications in information processing are just beginning to be realized.

Better ways of record keeping and information handling are now foreseen. They will be brought about by applying communications to data and information handling tasks. This, in turn, will provide new systems concepts for operational control of an enterprise. Systems is an important word in this concept of information handling. While hardware, equipment, computers and even special black boxes are important, the balanced system, using those components, must be designed and assembled to fulfill the needs of the user.

Both government and industry are placing new demands on business machine suppliers. As advances are made in the management sciences, management is asking for, and will receive, improved techniques to effectively control their operations. No longer are managers satisfied with making decisions based upon historical records alone. They face a dynamic competitive market place that points to expanding output, reducing costs and improving deliveries, as well as service. Management demands that future information handling systems include:

1. The entry of data into the system when it first is generated,
2. The bridging of the time and distance gap,
3. The direction and control of several operations or the entire enterprise.

These requirements point to the need for communication links that will make possible on-line, in-line, operational control systems. The common carriers have recognized this need and are working with business machine suppliers to this end. They offer a variety of facilities and devices that may be used for data transmission.

Data Transmission Equipment

Point-to-point data transmission equipment has been in use since World War II. The first such equipment was off-line and was used to convert punched card data into five channel paper tape for data transmission. At the receiving end, tape-to-card converting equipment provided machine language for input to the tabulating system. Next came the transmission of cards on-line from point-to-point over land lines. This equipment has been in use for the past seven years. It can be used on a dial-up basis, as well as over private telegraph or telephone networks.

Three more recent developments are magnetic tape-to-magnetic tape transmission

*Trademark

over telephone lines, either private or dial-up, and high-speed telegraph lines, memory-to-memory type communication between computers, and remote tape control systems which provide for computer control of data transmission over eight telephone lines to and/or from punched card equipment (including printers) and magnetic tape units.

The "TELE-PROCESSING" System

The above are the forerunners of "TELE-PROCESSING" Systems which can best be described as a combination of products, components, and communication links permitting the processing of data at a location remote from its point of origin. Generally, such systems are designed to function as an in-line on-line operation, processing data as it originates and up-dating all records affected. They are not, however, limited to in-line operation since batch or scheduled type data may also be handled. They are considered to be operational control systems that provide direction to an enterprise or control over several major applications within an enterprise, by furnishing information where it is needed, when it is needed, upon computer control or operator intervention, over communication links. An airlines reservation system is a good example of an operational control system.

In such a system the data to be processed may be:

1. Batched at a data originating terminal for transmission to a data processing center at a pre-scheduled time.
2. Transmitted upon inquiry (on demand) from a receiving terminal or the data processing center.
3. Transmitted immediately as the related event occurs.

These aspects and the many other related factors, including the configuration of the system to be specified for a given application, are items that must be identified and defined in the case of each user's requirements. In other words, these systems must be customized to user applications.

A "TELE-PROCESSING" System can be generalized into four functional areas as follows:

1. Processors
2. File Storage
3. Network Logic and Data Transmission Control Devices
4. Terminals

The number and complexity of individual components varies for each "TELE-PROCESSING" System. However, the functions of these components are required for all such systems.

Processors - The "Processor" is the heart of a "TELE-PROCESSING" System. It is a general purpose computer able to perform complex logical and arithmetic tasks and monitor its own operations.

These basic functions must be extended to include multiprocessing, multiprogramming, priority assignment and control, queueing protection and privacy of records, and multiple I/O channels.

File Storage - Because of the centralization of all records, "TELE-PROCESSING" Systems characteristically demand a high proportion of storage to processing.

For the same reasons, file operations occur more frequently, and must be handled without unnecessary intervention by the processor. This requires rapid addressing, rapid search, and priority control.

Consequently, "TELE-PROCESSING" file storage must possess a degree of sophistication and capability for largely autonomous operation that has not normally been associated with files.

It has become apparent that file storage will become of at least equal importance to the processor in future "TELE-PROCESSING" Systems.

Network Logic and Data Transmission Control Devices - These include both the equipment used to electrically interconnect remote terminals with the transmission facilities and the interface equipment between the communication links and the processing center and file storage. These devices often contain limited logical capability. For example, they may engage in decision-making such as line or terminal addressing. They may provide for code conversion with error detection and control. They may have buffers for a character, word, or record and may include a multiplexor with a speed exchange allowing for several low speed incoming channels to time share a higher speed communication channel.

Modulating-demodulating units generally furnished by the common carrier provide the interconnecting point between these devices and the communication links.

Terminals - They are the means for entering data into and receiving data from the

"TELE-PROCESSING" System. Terminal design and functional capability is a critical consideration in any "TELE-PROCESSING" System, as the quantity required is reflected as a major cost item in the system. The characteristics of terminals depend upon operating requirements, traffic patterns, response requirements and compatibility with transmission facilities. These devices may be "job oriented"; that is, uniquely designed to be in accord with their operating environment. They may be operated by people, or may be process-operated. They may provide for data entry by the means of keys, or their own sensors. They may provide for display of data in record form, audible signals, and machine languages, according to system needs. Generally, these terminals are paced to the rate of data entry, or the rate of data acceptance.

The Total System - Figure 1 shows a representative "TELE-PROCESSING" System, encompassing all of the above devices. This representation is a simple one; "TELE-PROCESSING" Systems will vary as to size and complexity. The number of terminals maybe a few or several thousand. The communication links will in general be provided by the common carriers. Private microwave networks may be required to meet specific needs. A "TELE-PROCESSING" System is system oriented; not hardware oriented. It is a balanced system handling the defined problem areas of a business in the best possible way.

The Balanced "TELE-PROCESSING" System

The basic functions of the elements required in a "TELE-PROCESSING" System are known and the business machine suppliers have developed or are developing equipment to meet these requirements. The data processing systems now being produced are designed for easy adaptation to communication links. Random access devices needed for "TELE-PROCESSING" Systems are available and are being continually improved.

Terminal and transmission control products are developing rapidly. The availability of job-oriented terminals (i.e., banking and air reservation), general purpose terminals, and transmission control devices, combined with the new offerings and new devices supplied by the common carriers provide a response to the needs of industry and government.

The major problem now lies in the design of a balanced system that will satisfy the individual user. To provide a balanced "TELE-PROCESSING" System, terminals, communication links, and central processors must be considered as a whole. The actual hardware must be used to augment the total system requirements. Every element of the system must be considered and measured. Such things as type of data, traffic flow, computer programs, memory speed requirements, possible extensions, growth requirements, value of input/output speeds, human operator consideration, and a host of other factors must be studied. Can the terminals time share the communication links? Can the central processor handle the queueing functions and housekeeping functions? Should any logic be placed at the terminal or the multiplexor? Should small computers be used as intermediate points communicating with a central processor? These are the types of questions that must be answered to attain a balanced system, and some of the procedures required before they can be answered will now be discussed.

The Design Problem

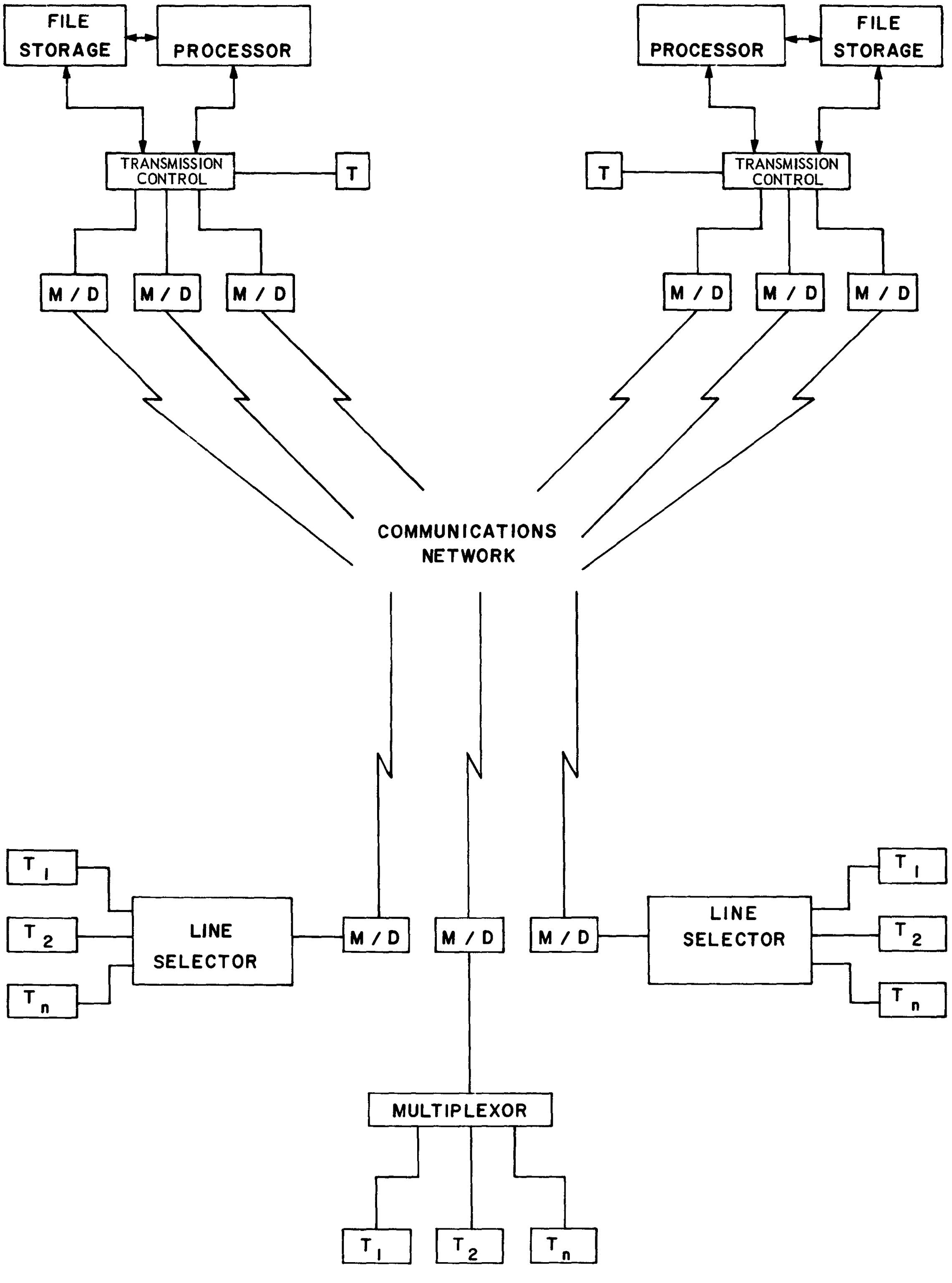
A "TELE-PROCESSING" System provides three basic functions:

1. Communications
2. Processing
3. Control

These three functions are provided by the hardware of the system acting under logical control dictated by the system interconnections and programs.

The prime problems of designing a "TELE-PROCESSING" System lie in effecting a proper balance of hardware and logical control to meet the user's requirements. The system must deploy its capabilities so that its overall capacity is properly shared by all concerned. The goal of the design is to see that each element of the organization gets service consistent with its needs and that a minimum of the total system is in a "waiting" role while a given element is being served.

The user's needs can be summed up as function at a required availability level within a profitable cost picture. Since function and availability are inherently inconsistent with



A TELE-PROCESSING SYSTEM

Figure 1. The "TELE-PROCESSING" System

cost, a proper balance must be sought on these criteria.

The first step in design is to determine the system requirements. This consists of the study of operations to determine application areas, the determination of the relative importance of the application areas, the gathering of the statistics of information flow in each application area, the study of the methods and procedures presently in use, and the development of new methods and procedures consistent with user acceptance and desired systems performance.

The results of these efforts are then used to produce the functional requirements for the system. These functional requirements include the quantitative and qualitative measurements which guide systems design and establish performance criteria. It is important to note that these requirements and criteria undergo frequent re-evaluation and revision during systems design to assure that a proper balance between function, availability and cost is achieved. In view of this constant need for reassessment, it is very important that the original data gathering and reduction be in a form that permits re-evaluation of the source data.

Systems solutions satisfying these functional requirements must be evaluated to determine the systems capability. At present, the design itself is still a human process. However, the magnitude and complexities of the systems dealt with dictate the development and utilization of a constantly expanding array of mathematical tools and techniques for system evaluation. The build-and-discard method of system design is simply inadmissible when dealing in systems of this size. A basically sound design must be assured before commitment to hardware. The mathematical tools discussed below make this assurance possible.

Queueing Analysis - or waiting line analysis - is the mathematical method for solving problems of organization and planning in the face of a fluctuating demand for service. This fluctuation may be due either to the lack of control over demand, as in people arriving at a subway turnstile, or the inability to maintain a schedule as in the arrivals of aircraft at an airport.

In either case the occurrence of long gaps of time between arrivals of elements of demand (e.g., people or planes) will result in periods of idleness of the service facilities (e.g., turnstiles or airports). This time is lost forever and subsequent periods of short intervals between arrivals result in the formation of queues. Thus, even though the service facility has excess service potential, intermittent congestion will occur.

An example of the application of queueing theory is in the design of storage sub-systems. It is possible to determine the required number of disk files (and associated channels) or drums (and associated channels) under varying traffic conditions. It is also possible to substitute the characteristics of various types of storage equipment into the formulae. In this way, the quantities of different types of equipment required to do the job over the life of the system as defined in the performance criteria established could be rapidly and economically established. This, in turn, aids in the price-performance evaluation of alternative sub-systems. The power of such formulae is that they yield directly the amount of equipment capability required to do the job as a function of known parameters. The impact of proposed operational changes, changes in equipment specifications and the availability of new types of equipments necessitate the constant utilization of these techniques on a continuing basis for each system under development.

Simulation - provides a means for the evaluation of a system by the cut and try examination of a mathematical model on a large-scale computer.

Before construction of the simulator, it is necessary that a functional model of the system be formulated in the form of a group of separate operating expressions.

The functional model must then be expressed in machine language. Care must be shown in the application of the simulation program in order to keep machine running time at a minimum while still building the most powerful test model. The ease of model modification is also a prime factor in this choice.

Once the model has been created, testing of system performance proceeds. Service times are measured. Waiting times are measured. Queue lengths are determined. These and other system design parameters are varied until the proper mix is established. This also provides the means for assuring that the proper utilization is made of the equipment capacity of the system. However, simulation techniques do not accomplish design. Their prime value lies in the ability to simulate the total performance of a complex system, and to measure the specific points of performance that the designer seeks to evaluate.

Communications - In designing a "TELE-PROCESSING" System, as much consideration must be given to the communication system as is given to the data processing elements of the system. They must work together. Therefore, they must be properly balanced in function, availability, and cost throughout the design.

Communications facilities used, or projected for use, in data communications must be characterized and their suitability for inclusion in "TELE-PROCESSING" Systems determined. These facilities include transmission channels and systems of all kinds, switching systems and equipment, modulation and demodulation units, and numerous minor equipments presently existing in the common carriers' plant.

Much work remains to be done, for example, in determining how best to use existing facilities for data communications. The problems involved include developing efficient methods of error control, techniques of synchronizing sending and receiving terminals, and modulation techniques which permit efficient utilization of channel bandwidth.

The communications network design is dependent upon:

The traffic to be carried by the network, specified as to origin and destination.

The response time of the network, the time between the initiation of an action and its completion, which can be as short as a few seconds for airlines systems or can be as long as an hour.

The required reliability and accuracy of the system, which can range from the absolute accuracy required by a banking system to the less rigid requirements of sending names and addresses for an airline reservation system.

The cost of the over-all data transmission network.

In planning a "TELE-PROCESSING" System, there can be no set technique. There are many different and frequently unique factors to be considered; hence various network layouts must be constructed and costed before a solution is found. Once again simulation procedures are used as a tool for obtaining a balanced (economics vs. efficiency) network layout.

"TELE-PROCESSING" System Programming - Programming of a "TELE-PROCESSING" System is an integral part of the over-all system design. Since it is possible to trade off performance of a given function between equipment and programming, it is necessary to do a thorough analysis of the relative cost and performance alternatives available. The system availability can be either increased or reduced, depending upon the nature of the programmed function and the way it is applied. These complex interrelationships require close coordination between the programming system design and all the other elements of system design.

One of the features of a "TELE-PROCESSING" System is its ability to accept demands for the execution of any of hundreds of stored programs, schedule the execution of these programs such that the components of the system are used optimally, and return the results to the originator of the demand. Since the criteria for arranging an optimum schedule varies from one application to another, from one configuration of system components to another, and, in a real time system, from one instant to the next, these functions can generally be handled more flexibly by a program than by hardware. (This control program has the responsibility of keeping the "TELE-PROCESSING" System operating at top efficiency. To fulfill this responsibility, the program controls the operation of the communications system, allocates core storage, supervises the transfer of data to and from all auxiliary storage media, handles error checking, communicates with the operator, etc.

Only The Beginning

The above considerations indicate the complexity of design of a "TELE-PROCESSING" System, yet this is a mere introduction to the subject. Each area in the design process must be fully explored for each system and as these explorations expand into various industry application areas, new techniques using new technologies will evolve. Much has been accomplished in this new field of "TELE-PROCESSING" Systems, but the full impact of data transmission on information processing is yet to come. This is only the beginning.

COMMUNICATIONS FOR COMPUTER APPLICATIONS

A. A. Alexander
American Telephone and Telegraph Company
New York, New York

On the East Coast a computer service bureau is able to show a net savings of over $9,000 per month by using high volume data communications to link computer locations separated by 70 miles.

A major airline has recently put into operation the first phases of what will be a computer controlled country-wide reservation system. This is inventory control at its best.

In New Jersey a banking institution is using a central computer to record and control the daily financial transactions and operations of its main and branch offices. If necessary the list could go on but the point has been made.

We are seeing the beginnings of an era in which centralized computer capabilities are extended to wherever they are needed and at the time they are needed. This extension is made possible by electrical communications.

Computer applications like this have been successful and are of value to the user, because Systems Integration in its broadest sense has been achieved. That is, the internal operations of the business, the computer functions and the communication system have been welded into an effective and economical operating unit. Systems integration like this does not come easily; it is at best an arduous task. But the rewards are well worth the effort.

I would like to discuss some of the things that we in the communications business feel are the major, even critical, factors involved in integrating communications with computers or data processing systems. My remarks will be limited to those factors associated primarily with communications requirements.

Time Value

Certainly one of the first considerations is whether or not communications are needed. For purposes of this discussion let us consider only the field of electrical communications rather than the more general one of information interchange by whatever the means.

Electrical communications in a data processing system are useful and economical whenever there is sufficient time value associated with the rapid transfer of the information. There is no question that more information per unit of cost can be sent by the U. S. mail than any other way, and if time is not of the essence, this is the way to go. For on the basis of cost per unit volume alone, electrical communications can not compete successfully with the mail service. However, where special messenger service is involved electrical communications are more economical over all but the shortest distances.

The importance attached to information having considerable time value is demonstrated by the continuing activity being directed toward reducing the cost of random access and stored program facilities in data processing systems. Progress in this direction is opening up entirely new markets for data processing machines and communication. These are the markets having high volume, short message requirements such as stock exchanges, credit bureaus, airline reservation systems, etc. with which many of you are familiar, I am sure.

Each one of the applications mentioned at the beginning, for example, are proving in communications because the essentially instantaneous transfer of information has produced savings or additional revenues which were not realizable without them.

Cost Factors

If we are going to investigate time value, we must know something of the costs of communications for our data processing system. Those of you who have had experinece in this area know that comparing the costs of alternate ways of doing a job does not come easily and it takes a lot of detailed effort. However, perhaps we can offer some guide posts to follow.

Initially, the most important factor to determine is the volume of information that must be transferred in terms of say characters or bits per day, or some other convenient interval. It is easy to be led down the wrong path on this. The fact that a computer will read, write or process information at say 15,000 characters per second does not mean, per se, that communications must operate at that same speed. Very little significant work has been done in determining the volumes necessary to provide economical and effective transfer of information in communication-computer complexes. Much more needs to be done in this area and we recommend it for your consideration.

A knowledge of the traffic pattern, the length and time distributions of messages, and the points of interconnection involved is also necessary if we are to specify our communications properly. It is not just possible to put too much emphasis on the value associated with such traffic pattern information. The entire economic success of the computer-communications complex hinges on a knowledge of such factual information. Perhaps the best way to underline its importance is to say that without it a satisfactory systems integration is quite unlikely to occur.

What communications are available, or are likely to be available to meet the volume and traffic needs? The Table shown as Fig. 1 lists communications services and facilities that are readily available. Shown in tabular form are three general categories, DATA-PHONE* Service (using the regular telephone network), DATA-PHONE Service with a WATS† (Wide Area Telephone Service) access line, and leased or Private Line facilities. Under each category are shown characteristics that our experience indicates will generally make this facility most effective and economical. The more characteristics that are met under the one listing, the greater is the likelihood that the requirement can be met by that communications facility.

Figure 2 describes the approximate costs associated with using the telephone network via the DATA-PHONE Service for data transmission. The circuit costs are based upon regular long distance telephone charges. The cost for the data set at the terminal depends upon the particular application and speed selected and varies between about $5/mo. and $100/mo.

Figure 3 describes the costs associated with using the WATS (Wide Area Telephone Service) line for data communications. This service is essentially long distance service on a flat rate basis. Costs are shown for unlimited calling as well as on a total time of usage basis.

The tabulation shown is for a medially located WATS line. In some areas the costs will be higher and some lower.

Leased or private line approximate costs are shown on Fig. 4. While at best an over simplified comparison, it does show that, on a cost per message basis, the higher the information transfer rate the lower the communications cost; if the higher transfer rate can be maintained.

This brings us to the question of utilization of leased or private line facilities. A systems study of data communications must not neglect to consider the other requirements for communications that exist between points in a network. For an analysis of the traffic pattern will probably show that the increased load of data communications can be satisfied on fewer additional circuits than if considered separately. Also by providing arrangements

*DATA-PHONE is a trademark of the American Telephone and Telegraph Company identifying Bell System equipment used in this Bell System service.

†WATS ia a trademark of the American Telephone and Telegraph Company.

DATA COMMUNICATION SERVICES TODAY
CONSIDERATIONS IN SELECTION

REGULAR DIAL TELEPHONE NETWORK	WIDE AREA TELEPHONE SERVICE	(PRIVATE) LEASED LINE
A. LOW VOLUME OF DATA MESSAGES	A. HIGH VOLUME OF DATA MESSAGES	A. HIGH VOLUME OF DATA MESSAGES
B. LARGE VARIETY OF POINTS TO CONTACT	B. LARGE VARIETY OF POINTS TO CONTACT	B. SMALL NUMBER OF LOCATIONS TO CONTACT
C. POINTS OF CONTACT CONTINUALLY CHANGING	C. POINTS OF CONTACT CONTINUALLY CHANGING	C. POINTS OF CONTACT REASONABLY FIXED
D. STAND-BY OR EMERGENCY SERVICE TO BACK UP PRIVATE (LEASED) LINE	D. DATA SERVICE IS INCIDENTAL TO LARGE VOLUME VOICE REQUIREMENT	D. CONTINUOUS INTERCHANGE OF INFORMATION AMONG MANY POINTS
E. AUXILIARY SERVICE TO HIGH VOLUME (PRIVATE) LEASED LINE	E. SPEED REQUIREMENTS CAN BE MET WITH 1200 BITS PER SEC. (2000 BITS/ SEC.)*	E. REQUIREMENTS CAN BE MET WITH SPEED UP TO 1200 BITS PER SEC. (2400 BITS/SEC.)* OR
F. REQUIREMENTS CAN BE MET WITH SPEED UP TO 1200 BITS PER SEC. (2000 BITS/SEC.)*		F. HIGH SPEEDS REQUIRED UP TO 500,000 BITS/SEC.

* AVAILABLE 1961-1962 PERIOD.

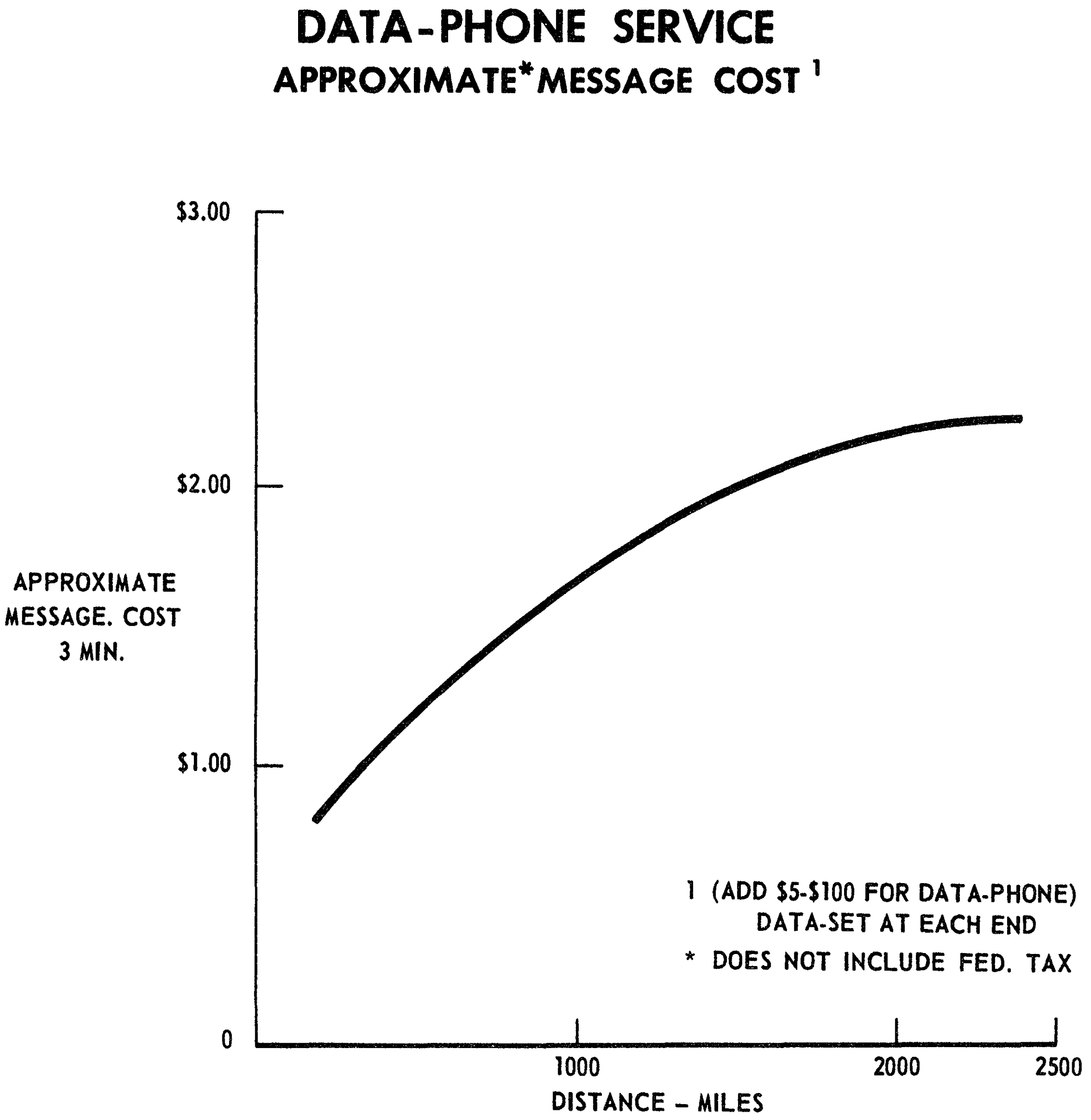
Figure 1

to make it possible to use existing communications facilities on an alternate basis with data, utilization can be increased and the cost per message minimized for all the services. Figure 5 shows, conceptually, the effect of poor utilization on costs. It also shows that where data communication requirements are small compared to others, they can frequently be met on an incremental basis with existing communication facilities at very low cost.

Reliability

The more short time value information that is transmitted by a data communications system the more important it is to have the facilities available when they are needed. (Error performance is discussed separately below.) Experience to date has shown that in general the down time of well designed and conceived communications systems has been less than the down time resulting from the maintenance requirements of computer apparatus. In most cases there is little reason to require standby communication facilities unless operations are so critical that standby computer equipment is also required.

The problem of determining whether or not standby facilities are necessary is the very simple one of assigning a value to



DATA-PHONE SERVICE
APPROXIMATE*MESSAGE COST[1]

1 (ADD $5-$100 FOR DATA-PHONE)
DATA-SET AT EACH END
* DOES NOT INCLUDE FED. TAX

Figure 2

continuity of service. To put it in more familiar terms, it is necessary to determine the penalty that is paid for service interruption, and how much should be spent to minimize this possibility. In these terms it is a problem of insurance against risk. As we are well aware, insurance does not guarantee against the occurrence of an event, it merely minimizes the loss that can occur.

Realizing the need for a minimum of lost time due to service interruptions, this is by the way, not limited only to data services, the Bell System has built in automatic apparatus in many of the long distance, high circuit density communication routes. This equipment detects the onset of trouble and establishes an alarm condition. In many cases the alarm condition causes standby facilities to be automatically switched into the connection in place of the section experiencing trouble. The speed of this automatic switch is so fast that it usually passes unnoticed even when it occurs on a system carrying network television program material. On common user communications systems this can be done with a minimum of "insurance premium" because one standby system can be used to protect several in constant use.

DATA TRANSMISSION
WIDE AREA TELEPHONE SERVICE
NORTHERN ILLINOIS
(Approximate Costs *)
(Add $5-100/MO for Data Set)

CIRCUIT TIME	AREAS					
	I(a)	II	III	IV	V	VI(b)
UNLIMITED	$700	$900	$1200	$1400	$1500	$1950
30 HRS./MO.	$495	$607.50	$715	$775	$805	$962.50
20 HRS./MO.	$345	$422.50	$495	$535	$555	$667.50
15 HRS./MO.	$270	$330	$385	$415	$430	$520

(a) CONTIGUOUS STATES.
(b) CONTINENTAL UNITED STATES.

* DOES NOT INCLUDE FED. TAX.

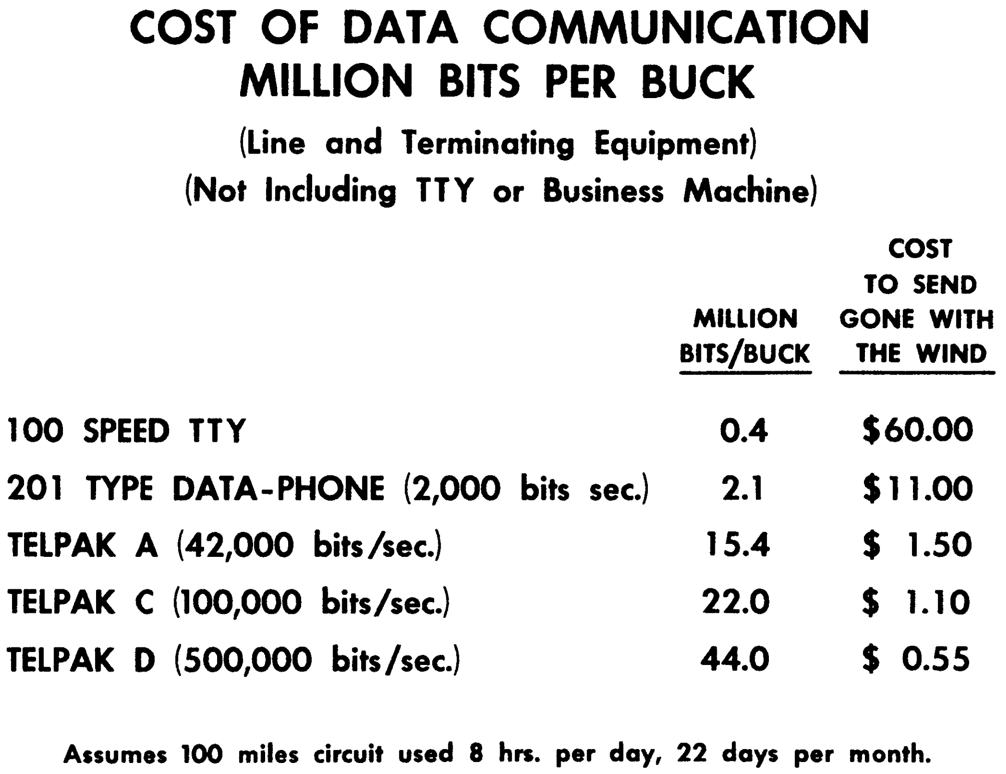
Figure 3

Error Performance

Communication facilities do cause errors to occur in data transmission systems. The same can be said to a more or less extent of any component of a data handling or processing system. A considerable amount of work has been done to get a handle on the best way to minimize the occurrence of error and its effect. It has been fairly well established, for example, that forward acting error correction, while quite interesting in concept, is not justifiable on the basis of efficiency or economy. The technique of block transmission, error detection and block retransmission has, however, on the other hand been very successful. It is quite obvious that where error control is required on communication facilities the detection and retransmission path will be followed. This technique has made the systems man conscious of circuit efficiencies in terms of per cent of retransmissions. At least one such system transmitting in 100 character blocks has experienced per cent retransmissions well under one per cent.

It has been the practice for data processing machine manufacturers to provide error control for each processing, reading or writing function. When communications are introduced, the question is often asked whether or not the communications component should have its own error control capabilities. This certainly has been considered and in most all cases the economics have been against it. The simple reason is that the error control capabilities already built in the data processing equipment at the terminals are entirely capable of doing the job. It just does not make sense to put duplicate capabilities in tandem.

Where communications are not used "on line" with say the computer or other data processing or storage device separate error control facilities may be necessary. In the final analysis the decision should be made on



COST OF DATA COMMUNICATION MILLION BITS PER BUCK

(Line and Terminating Equipment)
(Not Including TTY or Business Machine)

	MILLION BITS/BUCK	COST TO SEND GONE WITH THE WIND
100 SPEED TTY	0.4	$60.00
201 TYPE DATA-PHONE (2,000 bits sec.)	2.1	$11.00
TELPAK A (42,000 bits/sec.)	15.4	$ 1.50
TELPAK C (100,000 bits/sec.)	22.0	$ 1.10
TELPAK D (500,000 bits/sec.)	44.0	$ 0.55

Assumes 100 miles circuit used 8 hrs. per day, 22 days per month.

Figure 4

the basis of the penalty or loss that occurs as a result of the error and the cost that is justifiable to minimize its occurrence.

Today the occurrence of error due to human failings is in the order of 10 to 100 or even more times more frequent than errors due to communication facilities. Before error control is specified for the communications link it is wise to consider the source of the information and its eventual user.

The Bell System as well as the rest of the telephone and communications industry is well aware of the problems caused by errors in data transmission. Considerable effort is and has been under way to minimize disturbances which cause errors to occur. Where such disturbances are of the man-made variety and controllable you should expect to see continuing efforts to reduce these to insignificant numbers. For those that are essentially uncontrollable we will just have to live with or work around them.

Interfacing with Communications

Satisfactory interface is achieved when there are known, agreed upon, controllable and therefore usually fixed conditions at the point of interconnection. Figure 6 shows the interfacing arrangements which are used for the variety of connections to communications systems. One does not have to be exposed very long to communications engineering problems to appreciate the problems associated with standardizing an interface. These come about very simply because there is just no single type of communications system available that can be economically justified to fit every possible need. Sometimes the communication requirement can be met simply by providing pairs of wires in a shielded cable and some line amplifiers to maintain good signal levels. Where a requirement may exist for spanning a continent chances are a microwave radio system is the answer. Between these two lies an almost infinite variety of possibilities.

Historically, communication systems have been designed to accommodate the human voice. Voice communication can be satisfactorily transmitted and recognized in bandwidths or channels about 3 kc or less in bandwidth. Frequency multiplexing cable and radio carrier systems are used which permit

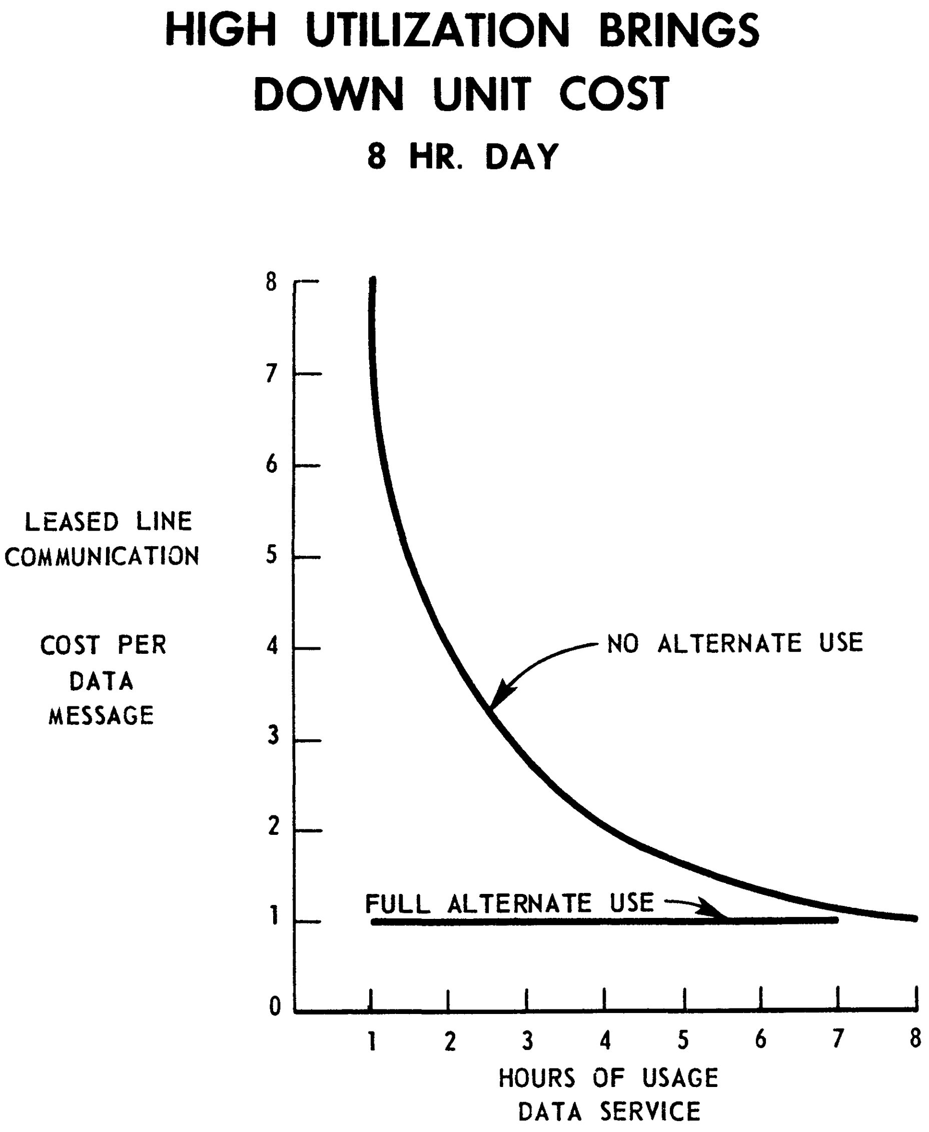


Figure 5

the simultaneous transmission of many such voice channels over a single communication system. In the frequency multiplexing technique it is convenient to handle the voice bandwidth channels in discrete groupings, and to translate them up and down in the frequency spectrum together.

The frequency band in many multiplex systems equivalent to that taken by 12 voice channels is calleda group. Five such groups are also treated as a unit in some systems and translated up and down in the frequency spectrum together. These are called super-groups. Figure 7 shows diagrammatically this arrangement. Communications made up in this manner have a whole variety of bandwidths and, therefore, data volume capabilities. For example, a voice circuit can be used to transmit up to say 2,000 bits per second. It is quite feasible to make use of the bandwidth normally taken up by say 12 voice channels, the group bandwidth, by eliminating the channelizing for the 12 voice circuits. We are working on terminal apparatus which will permit sending 40,000 bits per second on such a channel. Similarly, the entire frequency space of the super-group canbe made availableby eliminating both the voiceandgroupchannelizingfacilities. There is no reason why such a channel could not be

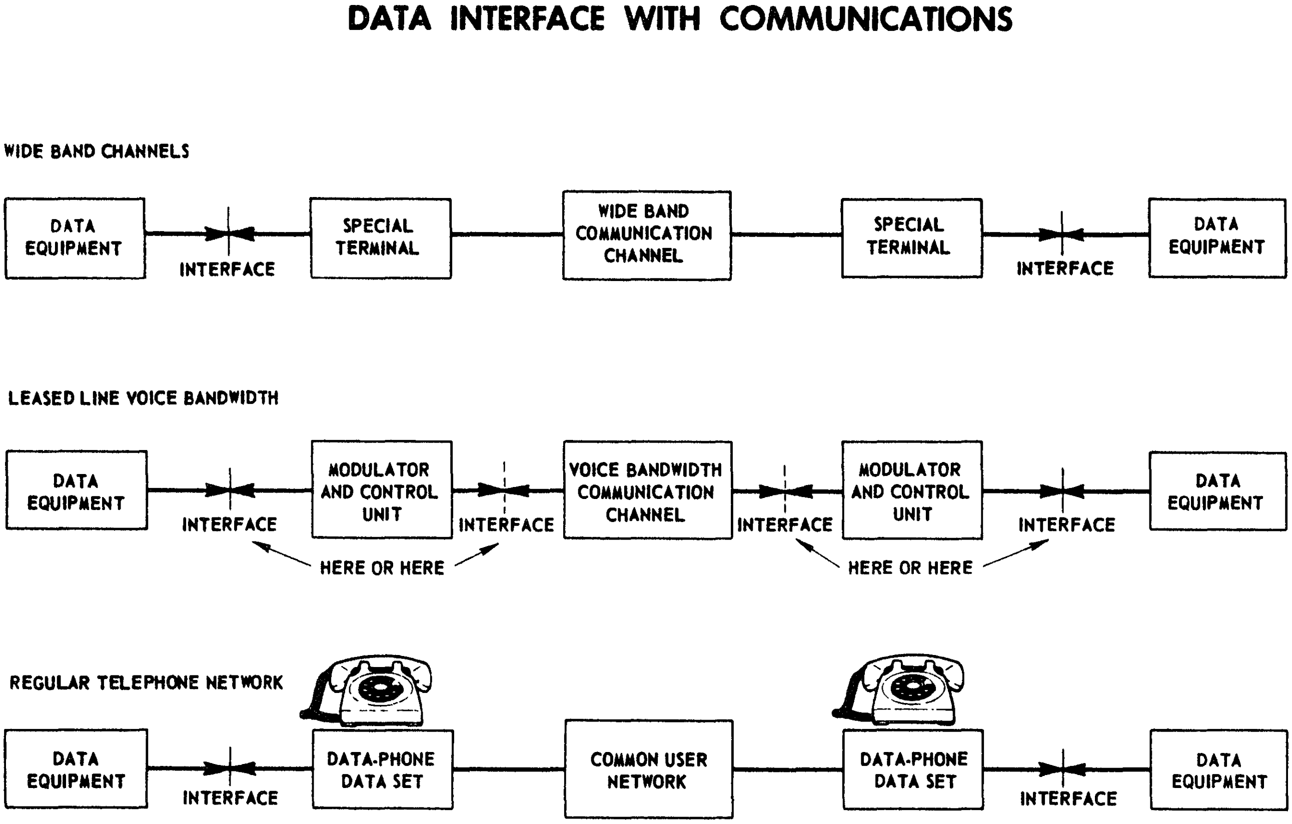


Figure 6

used for as much as 100-200,000 bits per second. As you might expect, this is what is done to provide the wide band data terminal facilities in the TELPAK* service. Other intermediate or even larger grouping are also possible.

The interface requirement in using these wider than voice bandwidth channels is to provide the necessary devices, couplers or terminals, to connect the data processor to the communications. In providing such arrangements, there are at least 12 different multiplex systems in the Bell System that may be used. The frequency assignments are different for all 5 groups and each of 10 super-groups. This means that the same computer or data processor on different circuits may require different communication terminals. In fact, the same circuit may require different terminals at each end. There are also requirements to provide pilot signals that operate automatic regulators, and there are specific requirements on channel filter characteristics, signal levels and frequencing stability. Add this to the fact that the entire carrier multiplex terminal, of which the wide band facility is only a small part, may be more suitably located away from the data processor it would seem most desirable that the coupling device be an integral part of the communications system. The interface to the data processing machine should then be designed to handle the data and control signals in the natural mode of the machine as far as possible.

The interface problem with the wide band, high speed communication is one in which the manufacturers of data processing machines would be well advised to have their system planning and circuit development people take an active interest. In the early stages of planning and circuit design it is frequently quite a simple matter to adjust signal wave shape, energy content and bandwidth requirements to make optimum use of communications facilities. After the system design has

*TELPAK is a trademark of the American Telephone and Telegraph Company.

TYPICAL FREQUENCY MULTIPLEXING ARRANGEMENT

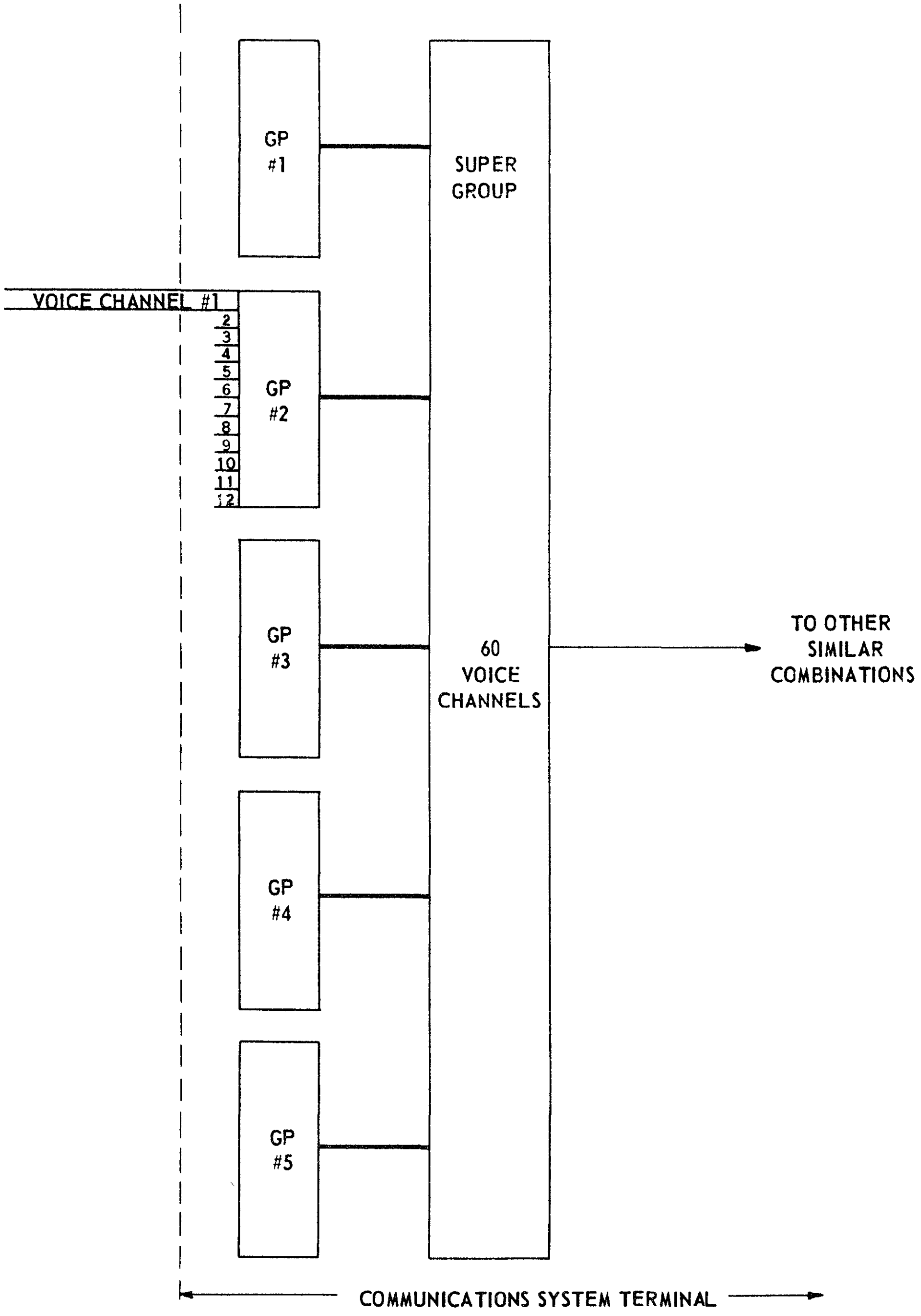


Figure 7

been frozen and equipment is coming off the assembly line it is quite a shock to realize that the unique shape of the data signal requires a bandwidth twice what it might have, had it been designed with communications requirements in mind.

We, as communications suppliers, are making available a line of simple but flexible terminals that are designed to operate with the types of channels we have. The interface has been determined almost exclusively by the requirements of the few data processing systems requiring such communication facilities.

We expect that we will have to do some custom engineering of this type for awhile. The rapidly changing and uncertain nature of this part of the business may make this necessary. However, on a long term basis we would like to work with manufacturers of business equipment and data processing machinery and other interested organizations in a program directed toward standardization of interface characteristics. Until this is done the market for such high volume systems will not be adequately exploited.

Where there is a need to interface at the very lowest bandwidth capability, say voice, the requirements for leased or private line facilities are somewhat easier to meet. Most of the problem relates to converting the natural mode signal, or baseband as it is frequently called, to signals suitable for transmission over the voice type facilities. Devices which perform this conversion as well as provide various specialized control features are available both as an integral part of the communication system or as adjuncts to the data processing equipment.

Where it is desired to use the regular telephone network to dial up connections and use them for data transmission the signal conversion requirement is still extant but of equal if not greater importance is the requirement to maintain proper control.

The telephone network is a large (at the moment certainly the largest) decentralized, special purpose computer. The telephone instrument and its associated dialing mechanism initiate action which instructs the computer to set up a connection. The computer does this in most cases for long distance calls by using stored program techniques. The telephone instruments at each end control the connection until one end or the other terminates the call, that is, "hangs up." Then the computer obliges by sequentially disconnecting all the links and making them available for other connections.

The ways that this control are manifested vary considerably and depend upon the type of switching system used and the type of transmission system available to transfer the control information. These control or signaling and supervision systems, as they are sometimes called, are almost continuously being expanded, changed and improved. When new features are introduced it is necessary that they be compatible with existing arrangements, otherwise it is necessary to replace the incompatible equipment on a scheduled basis. In the DATA-PHONE Service signal conversion and all control operations are included in the interface equipment provided.

Thus, it is possible to agree upon interface conditions which can be made independent of changes in the communications network. The DATA-PHONE data set permits continual improvement in both the data processing and communications arts without the need for continual re-evaluation of the effect changes in one will have on the other. In essence it permits unheeded progress technically and economically.

Capabilities of "Voice Grade" Circuits for Data Transmission

Considerable information has been published about the electrical characteristics of "voice grade" telephone connections. It has been quite well demonstrated that connections made for the satisfactory transmission of voice signals do not have bandwidths of 3 kc. As a matter of fact, the usable bandwidths are considerably less in most cases. Just a brief exposure to communication theory shows that with reasonable signal-to-noise ratios voice channels have considerably more information handling capacity than the 2,000 bits/sec., which at this time is the maximum speed soon to be available on DATA-PHONE Service.

The hard practical facts of life, however, demonstrate that nonlinearities in electronic apparatus and the inefficiencies in the simpler modulation and demodulation techniques make today's practical economic limit in transmitting binary information somewhere about one bit per cycle of bandwidth. In order to get greater information carrying capacity with existing techniques the cost of terminal

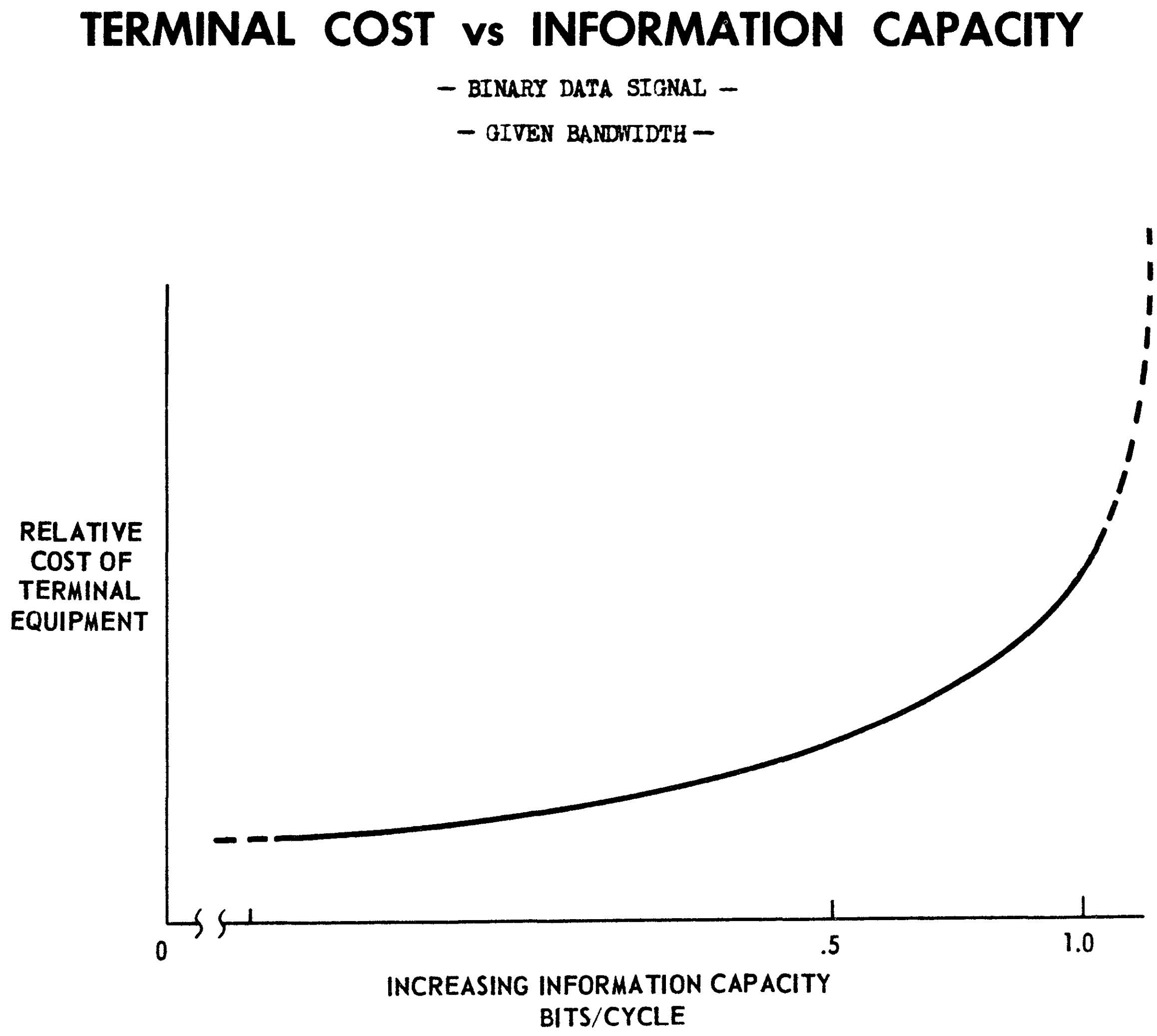


Figure 8

apparatus becomes increasingly prohibitive. Figure 8 shows graphically how the relative cost of terminal apparatus, modern devices, increases with higher information transfer rates at a fixed bandwidth for a binary data signal.

Selection of Proper Communications Facility for the Job

It is Systems responsibility to make certain that the most satisfactory communications facility available is selected to meet the requirements of the job at hand. Most satisfactory in terms of cost and performance. It has been mistakenly accepted in some areas that wire cable and carrier multiplex systems are capable and useful only to handle voice frequency bandwidths and that for anything having wide band requirements microwave radio is the only answer. We in the communications business must assume the responsibility for this impression. We just haven't done the type of public education and information job to get the facts where they will do the most good. We will be and are doing more to remedy this situation.

The facts are that wire cable properly used is a very excellent transmission medium for even the highest bandwidths over limited distances. There are a number of computer to computer transmission systems currently in operation using wire cable in all or part of circuits which operate in the range of 100-400 thousand bits/sec. In most major cities network television distribution of the 4.0 mc

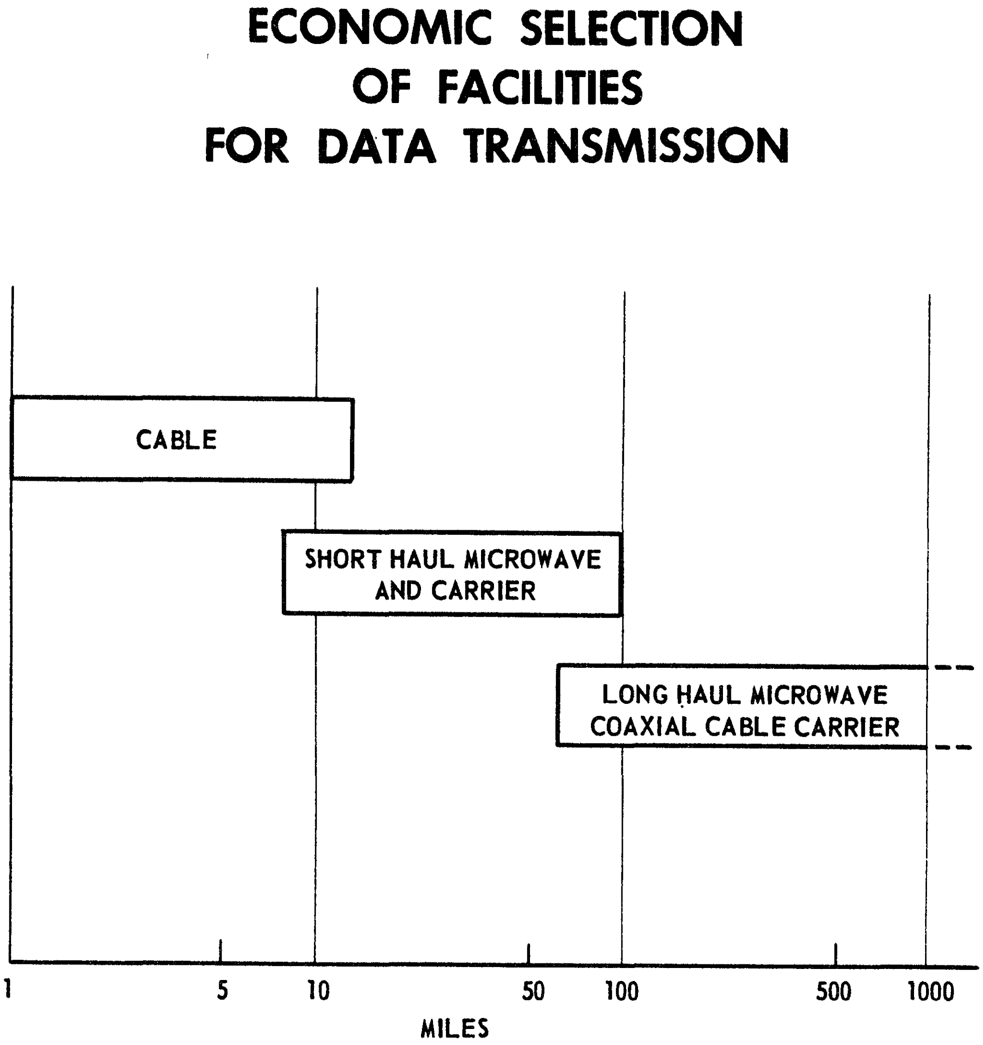


Figure 9

video signal is carried on shielded paired cable.

Cable carrier systems have a variety of bandwidths available, as has been described previously. Microwave radio is a carrier system having exceptional bandwidth capabilities and is very economical in proper applications. We in the Bell System have been using microwave for close to 20 years. Most of our longest circuits are carried on microwave radio systems. It is a mistake, however, to assume that any one type of facility is best for the whole range of requirements. Not the least of the considerations is cost.

Figure 9 shows in graphic form what our experience indicates to be the areas in which the various forms are most economical. Any one application, of course, has to be evaluated on its own particular requirements so Fig. 8 should only be used as a general guide rather than a fixed rule.

Communications Capabilities to Meet Future Requirements

On the whole the technical capabilities of communications have kept pace with developments in other fields requiring their use. Many of our current data systems problems relate to the best way to use what is available rather than the need for new capabilities. However, we see many requirements ahead which will challenge our ingenuity and resourcefulness. These requirements are challenging because they must be met by a variety of capabilities. We must have greater capacity at reduced cost. For simple point-to-point applications we see no technical limitation on speed. Transmission systems can be made to work as fast as a computer. There is a transmission system in final stages of development which can handle 1.5 million bits/sec. if required to do so. In parallel operation, say for a 7 channel tape, this system can handle 10 million bits/sec. There is also a wave guide transmission system in development which can handle over 200 million bits/sec. There is no question that high transfer rates will continue to be provided as required.

We think that it will be necessary to provide "dialed up" type of connections so that the dialing period as well as the connection set up time takes only a few seconds. This will make real time computer operations for short message traffic more attractive.

Varieties of speed capabilities must be provided not only for digital information but analog signals as well. One of the real problems in the use of high speed and thus wide band communication systems is utilization. For example, 400,000 bits/sec. equates to about 6 or 7 full length novels per minute. No matter how you look at it that is a tremendous capacity to have 24 hours a day. It takes considerable effort to keep a circuit like that busy.

It should be possible to provide means for making available wider than voice band facilities on an as needed basis. Call it a wide band switching capacity if you will. This too is most satisfactorily accomplished on a common user basis. Today there is no technical reason why this can not be done. So far, however, there hasn't been sufficient interest expressed by enough probable users to make it economically sound.

Time division multiplex systems with storage capabilities make excellent volume concentration devices. They may well be used in combination with high speed signaling and switching to enhance the existing common user network.

Electronic switching with its inherent flexibility is reaching a reliability and cost level where it will begin to be a serious competitor for the larger system applications. Arrangements such as these are particularly attractive where the requirements for other means of communication can be added to the data concentration and switching requirements. The economies of shared or alternate use can not be ignored.

The extent to which these new arrangements and techniques will be applied and how soon is quite uncertain. It will depend in large measure upon the progress made in fully understanding the customers needs from the systems standpoint.

In the data business today one frequently runs into the comment that "if the cost were only less what a tremendous market could be serviced." And this is applied to computers as well as communications.

As true as this may be it is still a fact that it is not always possible to meet all possible requirements at a satsifactory price level. That good five-cent cigar, for example, has managed to elude us despite the historic need. And today it looks as though we are farther away than ever from it.

However, we as communicators expect and intend to see to it that we are ready to serve data communication customers just as we serve other customers in other communication fields. This includes doing all in our power to make the required facilities available when and where they are required at a reasonable price.

THE SATURN AUTOMATIC CHECKOUT SYSTEM

Joseph Heskin
Packard Bell Computer Corporation
Los Angeles, California

ABSTRACT

Space vehicles and missiles have become so complec it is almost mandatory that they be tested by automatic methods under computer control for the following reasons:

(1) Increased reliability by eliminating human error and subjective evaluations.

(2) It is almost impossible to test a complex system manually within a practical period of time. Thus, manual methods for the checkout of complex systems produces excessive wear of critical mechanical components.

(3) The quantity of data and the quantity of reliability analyses required to check out a complex system are impractical to obtain by any means other than the use of a general purpose computer.

(4) The use of automatic equipment reduces the skills required of most of the people involved in testing, and reduces the number of man-hours required to perform a given test.

This article deals with the programming problems encountered in automating the checkout procedures for the booster stage of the Saturn vehicle. A brief technical description of the Saturn Automatic Checkout System is given and a detailed solution of the programming problems is discussed. In addition, pertinent conclusions are made regarding the future of real time process control.

INTRODUCTION

Generally, the computer utilized in a real time process control system has been a special purpose stored program type with the major emphasis on reliability since down time in these systems is extremely expensive. The control programs, with few exceptions, have been hand-coded special purpose routines written under stringent reliability and timing restrictions. Much time and effort were justified for these ventures since the programs, once debugged, were rarely modified. A more detailed analysis of this problem is discussed in the reference.[1]

Recently however, control problems are arising that are beyond the capabilities of these rather rigid systems. For example, modern space vehicles that are designed for manned space navigation present intricate

[1] "Programming for Process Control," E. R. Borgers, presented at the AIEE Joint Automatic Control Conference, Cambridge, Massachusetts, September G-9, 1960.

and involved checkout problems. In particular, automating the checkout procedures for the booster stage of the Saturn vehicle, presented control problems that could not be solved by existing techniques. It is the purpose of this paper to describe the system and techniques designed to solve the Saturn checkout problem.

A. THE SATURN BOOSTER CHECKOUT PROBLEM

1. Booster Stage of The Saturn Vehicle

The booster stage of the Saturn vehicle now in production for the National Aeronautics and Space Administration is a large complex of electrical and mechanical gear. The booster stage measuring approximately 21 feet in diameter and 80 feet in length, is the first and largest stage of the initial two-stage Saturn vehicle. This booster is roughly equivalent to a grand total of 24,000 cubic feet of some highly intricate equipment. Furthermore, the reliability requirements demand an extremely thorough checkout from the basic components through the subassembly stages, to the final overall assembly.

Booster stage testing is divided into seven areas of responsibility with an individual group in charge of each area: (1) R F Systems, (2) Instrumentation and Telemetry, (3) Guidance and Control, (4) Mechanical Components and Systems, (5) Ground Support Equipment, (6) Electrical Components and Systems, and (7) Networks Performance.

(1) R F Systems: this test area checks out the transponders, command receivers, and radar beacons within the vehicle.

(2) Instrumentation and Telemetry: at this test area measurement is made of transducers, signal conditioning equipment, signal distributors, measurement power supplies, as well as FM/FM, PM/FM/FM, and SS/FDM/FM type modulation schemes.

(3) Guidance and Control: the control computer, stabilized platform, and flight sequencer are functionally tested in this area.

(4) Mechanical Components and Systems: hydraulic and pneumatic systems are tested in this area. The tests include leakage under pressurization, functional tests, and also, weight and moment of inertia determinations for the booster.

(5) Electrical Components and Systems: cable networks, small subsystems, and components are tested in this area.

(6) Ground Support Equipment: cable networks, control panels, relay modules, patchboards, and cables are tested in this area.

(7) Networks Performance: this test area covers the overall system.

2. Checkout Considerations

The checkout procedures for the booster stage of the Saturn vehicle are being automated through the use of a computer-controlled, real time closed loop system. The automation is being done by Packard Bell Computer for the Quality Division at the Marshall Space Flight Center, NASA. This is a multicomputer system which utilizes a highly advanced approach towards problems of real time control.

The tests are so numerous and complex that manual checkout of the Saturn booster requires full time effort of more than 100 men and takes approximately 8 months to complete by conservative estimate. By overlapping schedules and increasing personnel teams, the time period can be reduced somewhat, although, an extensive time period still remains. It should not be assumed that the automated system is being installed to reduce the time required for checkout. In fact, the automated systems will most likely result in a 6 month checkout period; the reduction in manpower will be minimal since human handling time for test set ups will be about the same.

The lengthy equipment running time involved in manual test procedures causes intolerable wear and deterioration of critical components. The failure of a $56 valve which was worn out or damaged during a manual test can result in the failure of a $20,000,000 vehicle. A valve failure of this type is not likely to be induced by a computer-controlled system because the test is made at millisecond speed. Since automatic checkout produces minimum wear and deterioration of system components it is not only justified, but imperative, that such checkout be automated.

Other advantages obtained from a computer-controlled system are: (a) test reliability, (b) flexibility through programming, (c) automatic data acquisition and retrieval, and (d) automatic mathematical analyses and data manipulation.

(a) Test Reliability: Tests are repeated in precisely the same manner; marginal, inconsistent or erroneous test settings cannot be made by the computer whereas they are frequently made by the human operator.

(b) Program Flexibility. New concepts in system test, reflecting technological advances in instrumentation, more complex systems, or more efficient methods for evaluation of acquired data are easily accommodated in a computer-controlled system. Alteration of the internally stored program, as required, will accomplish this task with little or no change in hardware design.

(c) Automatic Data Acquisition and Retrieval. Inherent in the system is the automatic acquisition and retrieval of data in a standard prescribed format. The desired quantity of data is so great that it is literally impossible to handle it without the aid of a computer.

(d) Automatic Analyses and Data Manipulation. The large quantities of incoming raw data can be converted and stored during the test period. Immediate corrective action can be taken if any test is out-of-limits. All stored data may later be analyzed and reduced.

Major assembly testing requires the use of the vehicle, on which only one group (or at most, two groups) can work at one time. However, the automatic checkout system must be capable of testing the seven test areas simultaneously or independently. This capability is provided by a multicomputer system. Once the feasibility of simultaneous testing was established, an additional possibility became apparent: the simultaneous testing of two Saturn vehicle boosters with the same multicomputer system.

Finally, the checkout system must cope with the fact that no two vehicles will be identical in all respects. Each will be an improvement on its predecessor. This means that test procedures are not constant and will vary from vehicle to vehicle.

B. THE SATURN AUTOMATIC CHECKOUT SYSTEM

1. System Hardware Description

The overall system is under computer control. Three general purpose computers, expandable to 10, can communicate with one another by sharing common memory elements, under a master-slave structure (Figure 1). The master computer (MAC) can communicate with each of the slave computers (SLAC), and vice versa; however, SLACs cannot communicate with each other. The flow of information proceeds under program supervision and no special hardware is used to monitor MAC-SLAC communication. Each computer can communicate with any one of 11 test stations, expandable to 32. Each station contains the hardware necessary to check out a particular test area of the vehicle. The hardware forms a closed loop system that provides stimuli generation, switching, and response retrieval all under cumputer control. Only one station is accessible to a computer at any one time.

a. Stations

Each station contains, in addition to the checkout equipment for that station, an operator console, Flexowriter, and various display lights. Up to eight small portable remote indicators are also available where necessary. An operator may carry an indicator as he makes manual adjustments on board the vehicle, thus providing closed-loop communication with the system. It displays, under computer control, the piece of equipment that requires adjustment and also whether the adjustment is high, low, or within limits. If the operator finds that he cannot make the required adjustment, he may press a termination button which indicates to the computer that he wishes to go on to the next piece of equipment.

A further advantage of multiple stations, is that any station with an accompanying computer may be van-transported to new sites for use. This is particularly desirable if the vehicle is fabricated at more than one site.

b. The Central Station

One station is designated as the central station and contains the computers,

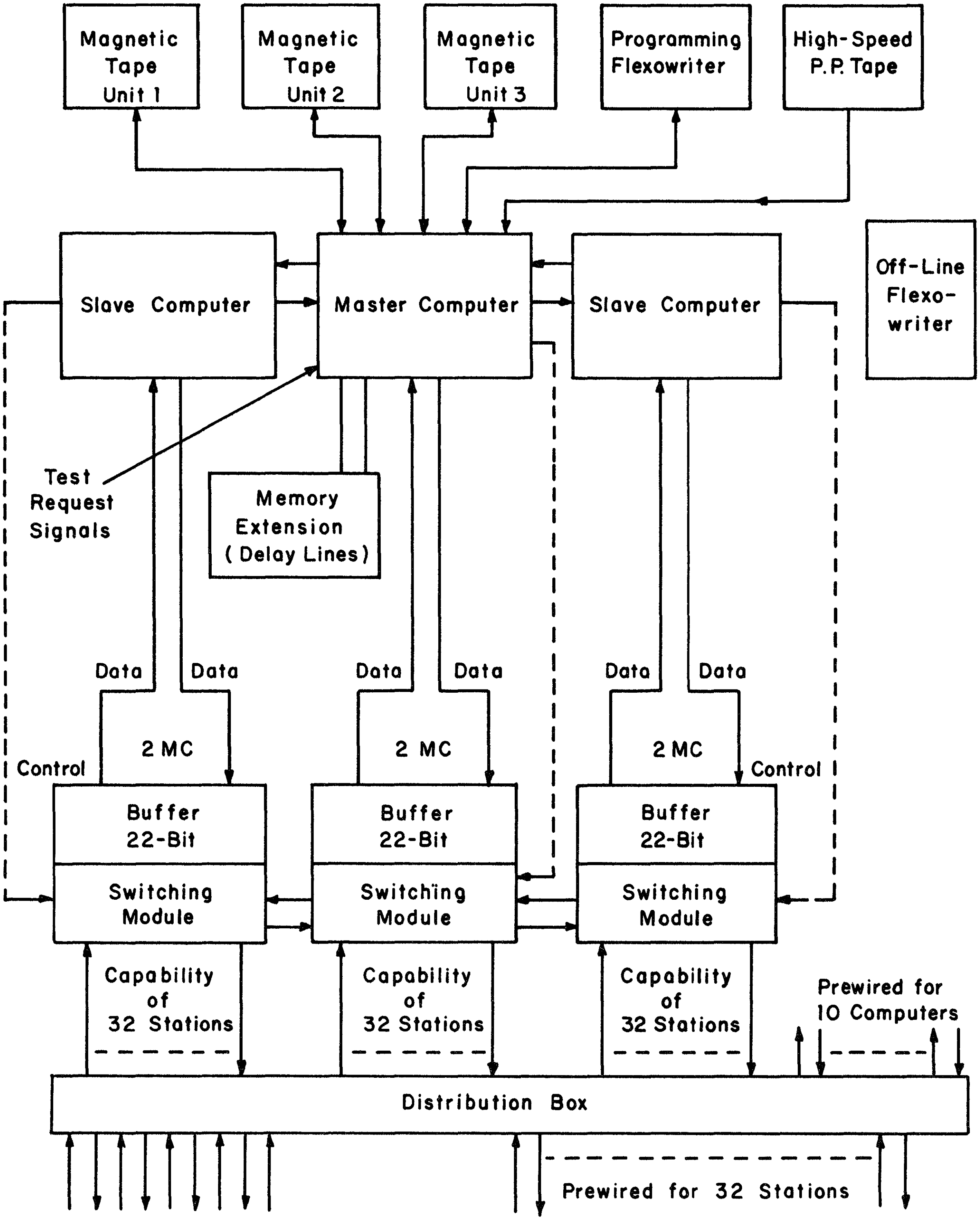


Figure 1. Saturn Automation System Central Computer Station

magnetic tape units, high-speed paper tape reader, high-speed paper tape punch, and two Flexowriters, one off line and one on line.

Three magnetic tape units, one for programs and two for data, are connected to the master computer. The number of units is expandable to six, if desired. All magnetic tape input/output is sent through the master computer, thus relieving the SLACs of this chore. Large amounts of data are generated as a result of the extremely thorough checkout which the vehicle undergoes. The aim is to be able to trace the performance of any component or subassembly through the tests with dates, failures, manufacturer, etc. Eventually, thorough statistical analysis may be applied to improve system design. These large amounts of data are captured on magnetic tape and sent to an outside data processing system for evaluation at a later date. Blocks of output may be stacked up within MAC, which connects to a memory extension chassis that doubles the amount of memory available in the master computer. If the output storage area is full, the programs in the slave computers must wait until the stacked output is processed.

c. Failsafe Provisions

If the master computer becomes inoperative, a slave may be substituted by manually switching the peripheral gear. Program input and data output can be accomplished with the high-speed reader and punch in the event that the tape handlers become inoperative.

A slave computer may be taken out of the system by manual switching and used to run miscellaneous programs. In this manner, the computer can be used as a general-purpose machine for scientific computation if desired. The off-line Flexowriter is available for input/output in this instance. Thus, an off-line computer doing low priority but useful work can be considered as a spare MAC or SLAC available in a few minutes.

2. Programming Requirements

When test procedures change from vehicle to vehicle, it is imperative that new machine language programs be generated immediately to accommodate the new changes. If these programs were coded by hand, the process would require many months. As a result, this problem was solved by the use of a compiler which permits the rapid generation of new or revised machine language programs. In addition, the compiler permits test engineers, instead of professional programmers, to generate these programs.

a. The Compiler

Automating varying test procedures creates the requirement for a language in which these procedures are easily written and a compiler that translates the resulting statements into machine language. The advantages of a compiler are numerous and well-known. In effect, the compiler automates programming since lengthy hand-coding techniques are eliminated, coding errors are greatly minimized, the compiler language is not restricted to any particular machine, the test engineer can understand the compiler language quite easily, and compiler language is mathematically precise and unambigous.

At this point, a brief description of the physical operations involved in control processing is necessary for an understanding of the requirements that a compiler of this type must satisfy (Figure 2).

Basically, a closed loop in real time control systems functions as follows:

(1) Generalized digital-to-analog converters serve as stimuli generators. The d-a is activated by digital information from the computer. Upon receipt of the digital information, the d-a converter outputs a corresponding analog signal (a-c or d-c voltage, pneumatic or hydraulic pressure, etc.) to a switching matrix.

(2) An input switching matrix accepts the incoming analog signal and directs the signal to a designated point on the unit under test (UUT). The switching matrix in conjunction with other hardware of the closed loop system is under computer control.

(3) An output switching matrix accepts the analog response from the UUT and transfers this information to an analog-to-digital converter which serves as a response conditioner.

(4) The a-d converter converts the analog response to a digital equivalent and returns the

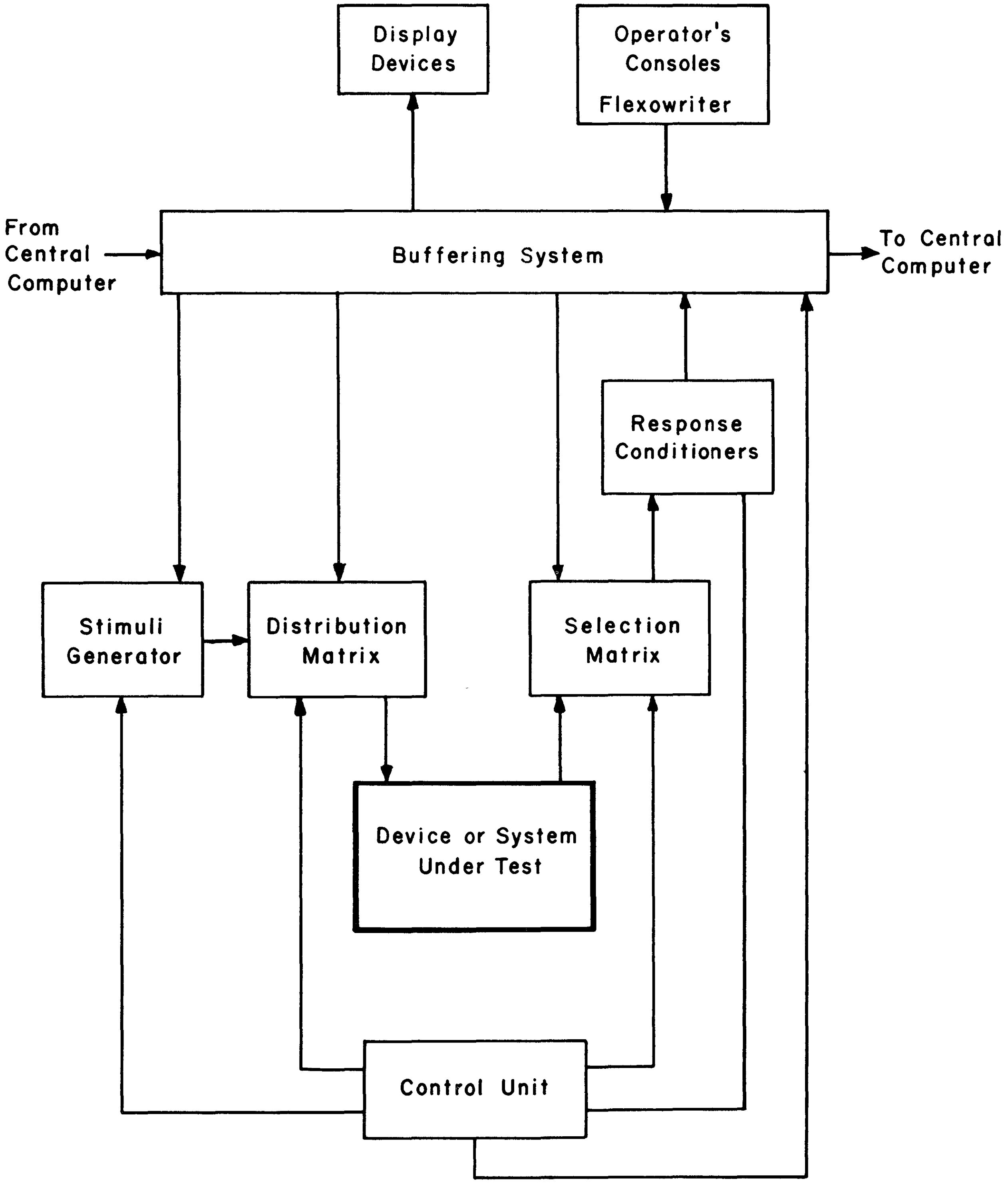


Figure 2. Remote Test Station General Block Diagram

information to the computer for processing.

In addition to these basic problems presented by a closed loop real time control system, the Saturn Automatic Checkout System presented the usual communication problems between the hardware and the computer. However, in the Saturn Automatic Checkout System this communication problem is magnified because there are numerous test stations performing different tasks which must be coordinated by the computer. Therefore, it was essential to develop a generalized communication system applicable to all present and future stations.

Communication between the computer and a test station which contains checkout equipment is accomplished by a buffering arrangement with one buffer at the computer and one at the station. Under computer control, the station buffer is connected to the computer buffer. The station signals the computer when it is free to accept information. At this time it is possible for the computer to fill its buffer with 22 bits of information (which constitute a computer word) that is automatically transferred to the station buffer. The computer signals the station that it has initiated a transfer of information which is then decoded by a control unit at the station according to the following format:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
ADDRESS							COMMAND			VALUE											

(1) ADDRESS utilizes 7 bits to specify a device such as a stimuli generator, a response conditioner, a switching matrix, etc.

(2) COMMAND utilizes 3 bits to specify a primitive function which the specified device is to perform. Three basic commands are provided:

(a) STORE. Under this command, the VALUE portion of the buffer word is sent to the device specified by the ADDRESS.

(b) LOAD. This command causes the addressed device to send its information to the buffer.

(c) EXECUTE. In this command, a device is activated or deactivated.

(3) VALUE utilizes 12 bits to indicate levels or ranges of stimuli, paths through matrices, and on or off states for activating or deactivating various devices.

b. SOL, the Compiler Language

SOL (Systems Oriented Language) was devised for the Saturn Automatic Checkout System. SOL provides for data input/output, arithmetic and Boolean computation, the inclusion of machine language subroutines, process control, and the necessary on-line monitoring that is characteristic of this type of operation.

With SOL, data can be displayed, typed out, punched onto paper tape, or written on magnetic tape. Data can be input by paper tape, Flexowriter keyboard, or magnetic tape. Sense switches at each station can be used to direct the course of the checkout program.

A detailed description of the language would be too lengthy to be given here. However, there are three SOL language statements that deal directly with the station hardware.

(1) SET (X, V);
X, symbolically, designates a physical device which may be a stimuli generator, a response conditioner, a switching matrix, a timer, etc. V may be a number or the words "on, off." This statement is used to set devices to desired levels or ranges, and then to activate or deactivate them.

(2) PATH (X, Y);
This causes a path to be closed through a matrix which connects the points symbolically designated by X and Y. The matrix need not be named explicitly since the compiler always selects the right path.

(3) MEASURE (U, Z);
This causes the device U to send the desired response to the computer. The response is the symbolic variable Z upon which arithmetic computation or limit testing can be done.

c. Station Tables

There is a table for each station available to the compiler describing the devices within that station. The tables contain symbolic identifiers referring to specific devices, matrix connections, formats of incoming or outgoing data, and, generally, the information necessary to the compiler for processing the hardware statements.

As stations change, or as new stations are added to the system, the tables are updated. Under this scheme, SOL can be used in any situation for which the relevant station tables can be filled out.

d. Monitoring Program Requests

A request for a program run from a test station is initiated by pressing a request button located at each station console. If a SLAC is available, the operator may type in the number of the program he desires which is then read from magnetic tape into the slave. The slave connects its buffer to the station buffer and the program is run.

If no slave is available, requests are stacked by MAC for later processing under a priority system. Communication between stations are carried on by Flexowriter. In addition, the executive program in MAC logs requests, program start and stop times, etc., at the central station and the test station, when applicable.

Six sense switches at each console can be tested by the running program for ON, OFF conditions. This allows the station operator to monitor the programs. He can also enter data on line via Flexowriter. In addition, pressing the request button halts the program and allows the operator to transfer control to any one of 62 different sections of his program, or to terminate the program in the event of station hardware failure.

The section locations must have been declared in the source program. Whenever possible, programs are segmented to allow short sections to be run. In this way, more stations have access to the computers and long waiting times are reduced. Segmentation points must be declared by the test engineer.

e. Program Checkout

Since the compiled programs will be used to drive expensive gear, it is imperative that they be debugged prior to their actual use. Debugging must proceed at a higher level than machine language which is not understood by the program originator.

The procedure adopted is to compile in a special mode such that at run time the object program types out the information that would normally be sent to the station buffer. The information is typed at the station Flexowriter in octal format. Responses are simulated by typing in octal data. The remainder of the program is compiled in the normal manner.

In this way the station operator can simulate any hardware condition he wishes. When he is satisfied that the program is performing as he intended, or at least that no damage to the hardware will ensue, the program is recompiled and placed on the program tape with a special tape editor program.

f. Checkout of Station Hardware

Controls at the station console provide for disconnecting the test station Flexowriter from the central station. In case of computer station failure, the Flexowriter can then be used to manually drive the station hardware. Buffer words are entered by the keyboard or punched tape one at a time in octal form; the first character entered determines the first bit, and the remaining 7 characters determine the last 21 bits. Hardware responses are displayed in lights at the console. This feature is useful for maintaining and debugging the system hardware rather than checking out the vehicle, although it could be used for the latter purpose.

CONCLUSIONS

It is apparent that many of the problems posed by the Saturn Automatic Checkout System were solved by the powerful combination of a multiple computer array and SOL, the compiler language.

However, the question arises: how far have we gone towards the solution of all real time process control problems?

Undoubtedly, the effort made today by Saturn programmers will appear crude by future standards, but it is believed that we have made a step in the right direction. The reasons for this belief are twofold: (1) an expandable multicomputer approach provides the highest degree of flexibility since it is a completely general approach to all large real time process control systems which require parallel processing. In addition, parallel processes under this multicomputer scheme can be monitored independently, and (2) the language SOL, is a language which can be expanded and used for any control system situation that requires a computer powerful enough to accommodate the compiler. However, in some cases, the compiler may not produce coding efficient enough to meet stringent timing requirements. This problem can be alleviated somewhat by resorting to a hardware solution; that is, the use of special timers and a good interrupt system.

In pioneering this Systems Oriented Language, it was discovered that there is a great lack of uniformity in closed loop process control hardware from various manufacturers. If future process control compilers are to handle complex problems of even greater magnitude, the need for hardware standardization becomes more imperative. Otherwise it will be impossible to write station tables for all future hardware designs. Any programmer who has attempted to coordinate bcd, octal, straight binary, bi-quinary and other bastard codes into one system, is well aware of this need for standardization.

In conclusion, it is strongly recommended that programmers work closely with the design engineers during the early design phases of a large automatic checkout system to insure hardware standardization. In this manner, a great many of the problems arising from nonstandard units will be eliminated.

"INFORMATION HANDLING IN THE DEFENSE COMMUNICATIONS CONTROL COMPLEX"

T. J. Heckelman
R. H. Lazinski
Philco Corporation
Communications Systems Division
Fort Washington, Pa.

INTRODUCTION

This paper describes methods of handling information within the Defense Communications Control Complex (DCCC) and its control center, the Defense National Communications Control Center (DNCCC).

DEFENSE COMMUNICATIONS SYSTEM

In May, 1960 the Department of Defense established the Defense Communications Agency (DCA) to provide a centralized, single management for the long-haul portions of the DOD Communications Systems. These communications systems, including all Army, Navy, and Air Force world-wide, long-haul communications networks, were unified to form the Defense Communications System (DCS). The services continue to operate these communications facilities under operational control of DCA. This system includes all world-wide, long-haul, Government owned and leased point-to-point circuits, terminals, control facilities and tributaries required by the DOD to provide communications for command and control, operations, intelligence, weather, logistics and administration.

DEFENSE COMMUNICATIONS CONTROL COMPLEX

In order to establish this operational control over the Defense Communications System, DCA established the Defense Communications Control Complex. At the heart of this Complex is the Defense National Communications Control Center (DNCCC). Figure 1 shows the relationship between the Department of Defense and the Defense Communications Agency. By maintaining and displaying data on all DCS communications facilities, including current status information, the system supervisor at the DNCCC has, at all times, a complete picture of the state-of-readiness of the DCS and is able to most effectively exert operational control over the system to provide maximum utilization of the available facilities.

Figure 2 shows the operation of the DCCC, and illustrates how status information on the DCS is made available to the system supervisor at the DNCCC. At each DCS station, information is collected on the status of communications facilities and traffic at that station. This includes outages of trunks, channels, circuits, and facilities, reasons for outage, users affected, temporary measures to restore service, etc. Traffic information includes the quantity, type, and precedence of traffic to each destination. This status information is composed into preformatted teletype messages and sent to the DNCCC through interim data collection centers (IDCC'S) maintained by each military service.

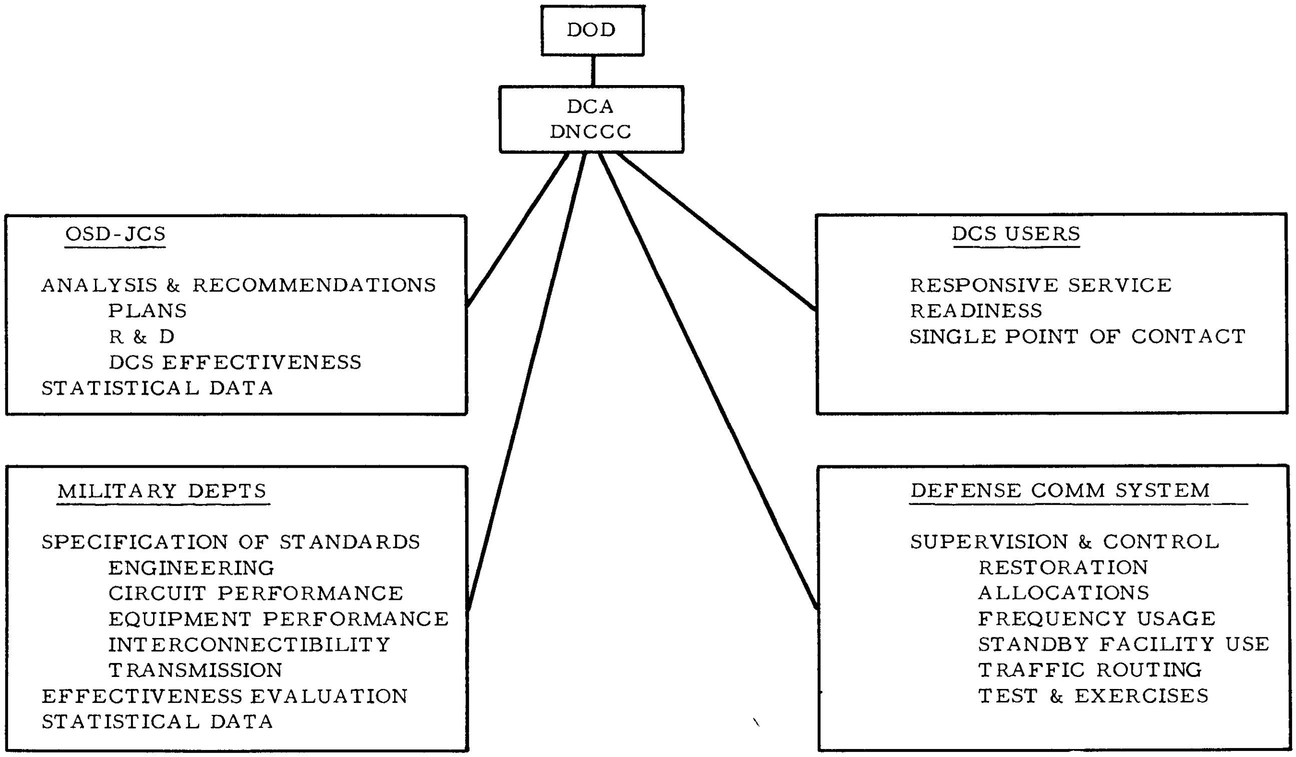


Figure 1. Relationship of DCA to DOD (Block Diagram)

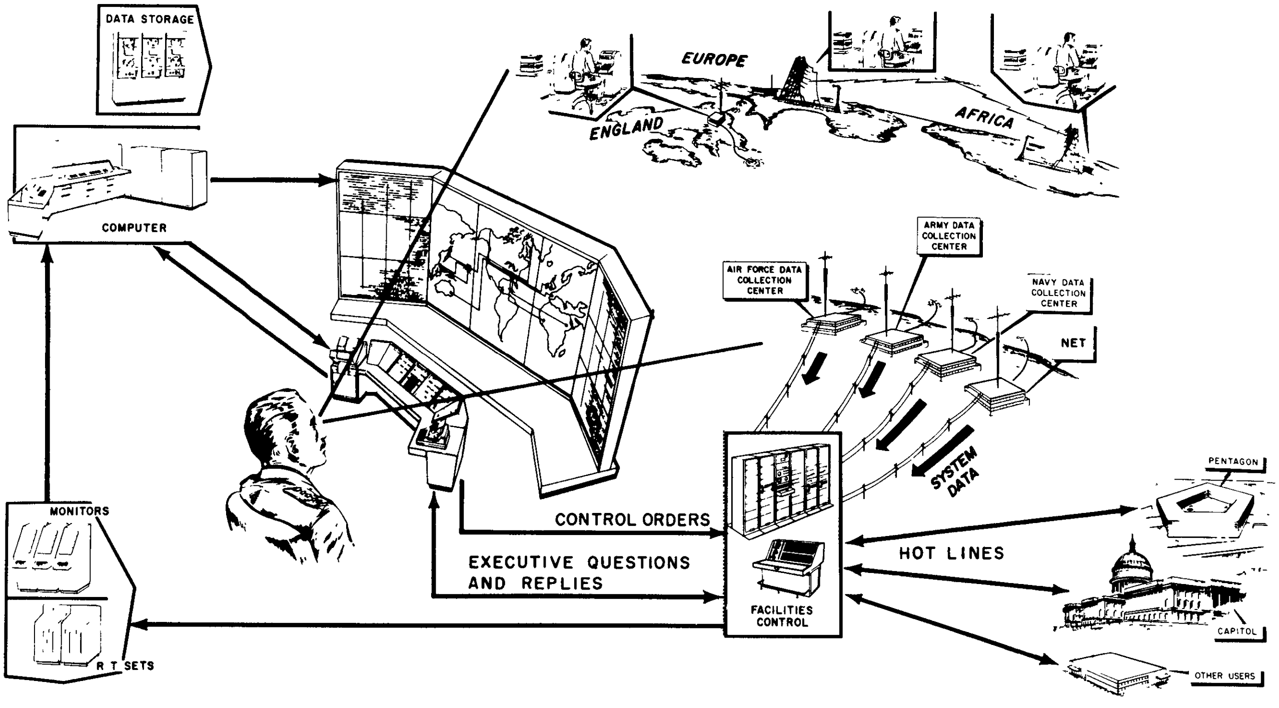


Figure 2. Communications Data Flow (Illustrated Block Diagram)

These messages are received at the DNCCC, monitored, buffered in tape repeater sets, and then read directly on-line into the Philco 2000 Computer, where it updates the data base. Part of this updated information is then automatically displayed to the system supervisor, with additional details available when queries are asked using the supervisor's console. This gives the supervisor a complete view of the actual status of the DCS, and provides required information for control decisions. The control commands are transmitted to the DCS stations by voice or teletype lines. Information on the status of the DCS is also made available to DCS users through direct telephone lines.

In addition to the system supervisor, there is also a network supervisor who is provided with additional displays and information regarding networks of special interest within the DCS.

A block diagram showing the interconnection of equipment at the DNCCC is shown in Figure 3. All on-line connections to the Philco 2000 Computer are handled through the Computer Access Device (CAD). Input to the computer can be either the system status data previously discussed or a query from a supervisor's console. Outputs may go directly to displays on the main display boards and consoles, or to high speed printers associated with each console. To achieve a clear understanding of the data flow in the system, consideration will be given in turn to each of the major data handling functions in the following order:

(a) Status data input
(b) Display data output
(c) High Speed Printer Activation
(d) Query data input
(e) Computational Sub System

Thus, this paper follows the communications data received from the services and the network data gathering centers through the buffer storage to the CAD and thence to the computer. The course of the digital

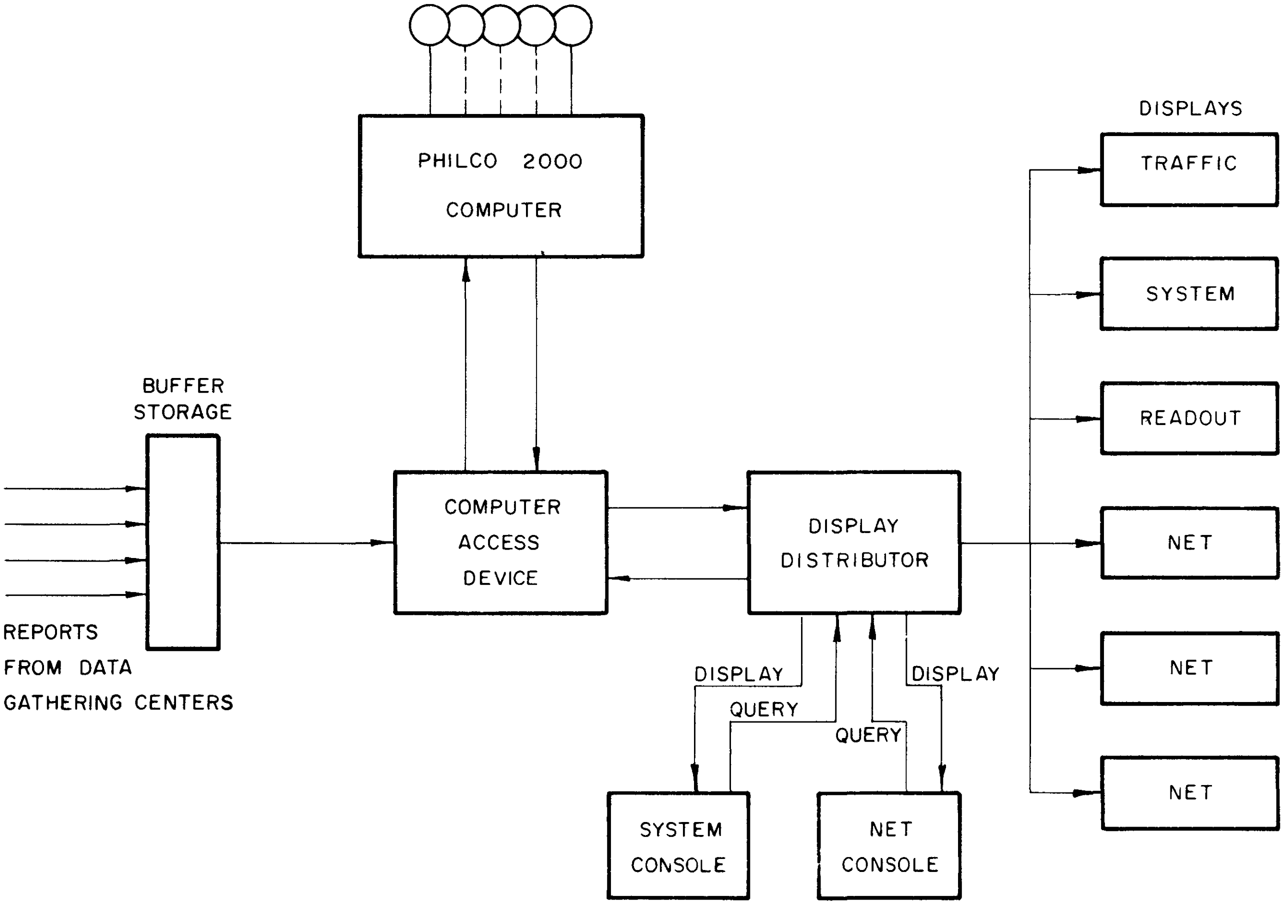


Figure 3. DNCCC Simplified Block Diagram

information is then followed outward from the computer to CAD and to the displays; at which time a portion of the man-machine interface will be discussed to illustrate how the digital information derived from the computer is converted into display-activation signals which, through the displays, convey the status of the DCS to the system supervisors. The discussion will include an explanation of how gross and detailed information is presented by these displays so that rapid comprehension of the system status is immediately available for the making of decisions.

The flow of a query initiated at a console will then be traced through the digital system to the CAD. Following this will be a discussion of data interchange across the input-output interface of the Philco 2000 Computer; its internal configuration will be described. Finally, the functional configuration of the data processing subsystem at the DNCCC will be described.

STATUS DATA INPUT

All DCS status information is transmitted via teletype to one of the four data gathering centers. From them, it is relayed to the DNCCC through a facilities control.

These teletype formatted messages are precoded at the point of origin as a message which is both readable by man and suitable for entrance directly into the Philco 2000 Computer. Each of the data gathering centers relays the teletype messages in a bit-serial teletype code to an associated tape repeater RT set at the DNCCC. At the DNCCC, the start and stop pulses are stripped and a 5-level edge-printed chadless tape is produced in each RT set. This tape is fed into a tape storage bin from whence it goes to a 200 words-per-minute tape reader. All four of these RT-set tape readers can read the last character punched. All four tape readers are read in unison, each producing a parallel 6-bit character (5-bit code plus a strobe). The information from 4 simultaneously read tapes is combined to form a series of 24-bit parallel words. Any tape having no data on it causes the associated portion of the word sent to the CAD to be represented by zeros. Each 24 bit word is put into the CAD for transfer into the computer. A block diagram showing the path of this communications input for DCS status data is shown in Figure 4.

DISPLAY DATA OUTPUT

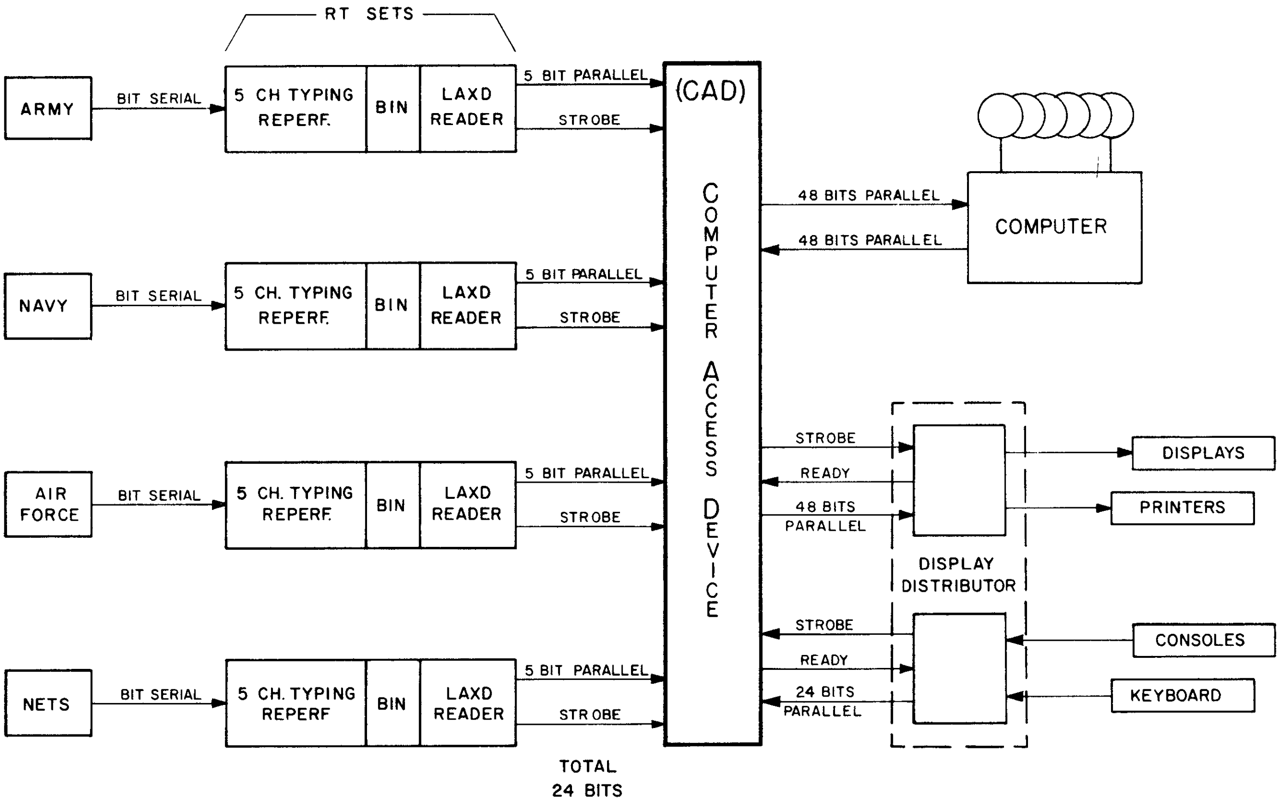
The display data presented to the system supervisors is in two forms:

1. Status display, as illustrated in Figure 5 by the System Status Board in the center, the Traffic Status Board on the left, and the Supervisor's Console in the foreground.
2. Printed clear text or tabular information, available at the high speed printer on the left of the console, or projected white with a black background on the Readout Board.

All of the Status Board and Console Display Modules have computer controlled multicolor capability. Alarm conditions are easily recognized by their blinking red indication. The system continues to display a newly deteriorated condition by causing the associated indicators to continue in the blinking state until the supervisor acknowledges their presence.

The System Display Board, 8 by 15 feet, is a geographical presentation of the world upon which are super-imposed the locations of all the DCS relay stations. Between these locations, their inter-connections are shown by lines representing trunk groups. Each half of these lines has a different color capability, thus showing the condition of the communications link in both directions. The appearance of a white halo around any location rectangle indicates that there is a station at that location which has a serious traffic backlog. Reference for detailed information must then be made to the Traffic Status Board, shown to the left in Figure 5. The representation on this board conveys to the system supervisor information as to the communications capabilities of the various stations comprising the DCS and the effectiveness of the communications between these locations.

The Traffic Status Board is arranged to correspond to the relative geographical location for all DCS reporting stations and shows their traffic backlog condition for both High and Low precedence messages. This display board has the capability of, upon interrogation at the console, displaying the traffic status condition from any station to any of the stations directly connected with it. Each of these modules has computer controlled multicolored capability to indicate status condition.



DNCCC COMMUNICATIONS INPUT DATA FLOW

Figure 4. Communications to CAD Data Flow

The Console shown is used both as a display and as system-status interrogation point. It is a display in that it shows status of the individual stations, their geographical locations and their connectivity to other stations in the DCS. The console also includes a hard copy printer which produces alphanumeric answers to queries. It is an interrogation point in that each of the display devices which shows station status is also a button by which that station designation is entered into the system as part of a query.

On the console there are two keyboards; one for introducing fixed queries and controlling the displays and one for entering variable format queries.

Other displays at the DNCCC which depict status conditions of special networks are similar to the system and traffic status boards described. The console for the Network supervisor has the same design as does the system supervisor's console.

The Readout Board is used as a variable format alphanumeric presentation which can display even under bright ambient light conditions, characters easily seen at 25 feet. This display produces either tabular or clear text answers to system supervisors' queries entered at the consoles. The Readout image is produced by a modified Anelex High Speed Printer, the operation of which is described below in the section entitled, "High Speed Printers."

The way in which the displays and printers are computer activated and the method for entering queries into the computer are also described below.

The functions of each of these displays, its method of portraying information and its relationship in establishing the interrogation-decisions man-system interface has been described in greater detail in two other papers (1,2).

CAD-TO-DISPLAY DIGITAL DATA FLOW

The 48-bit parallel word generated by the computer for updating displays or for printing answers to queries can take one of two

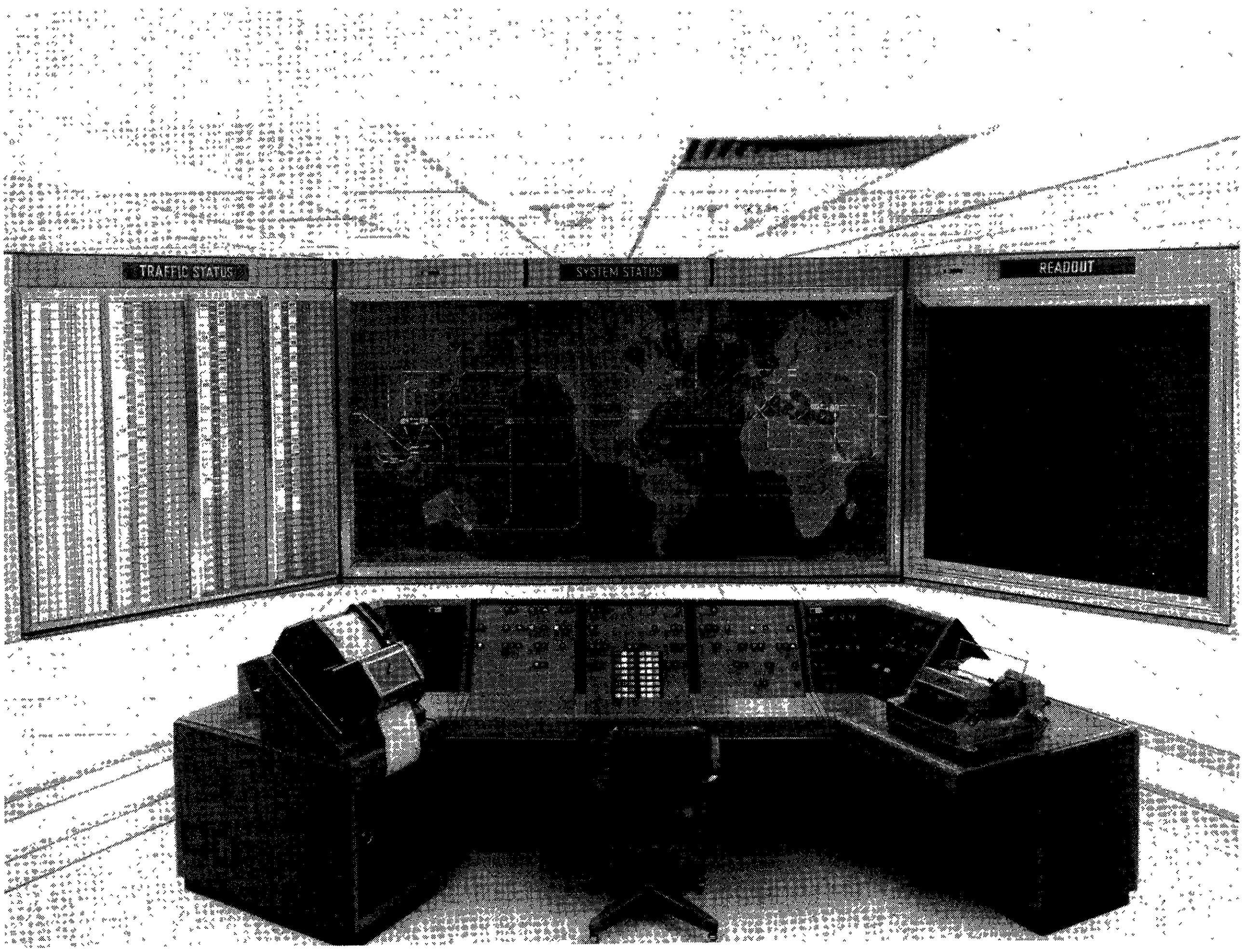


Figure 5. Photo of System Supervisor's Position

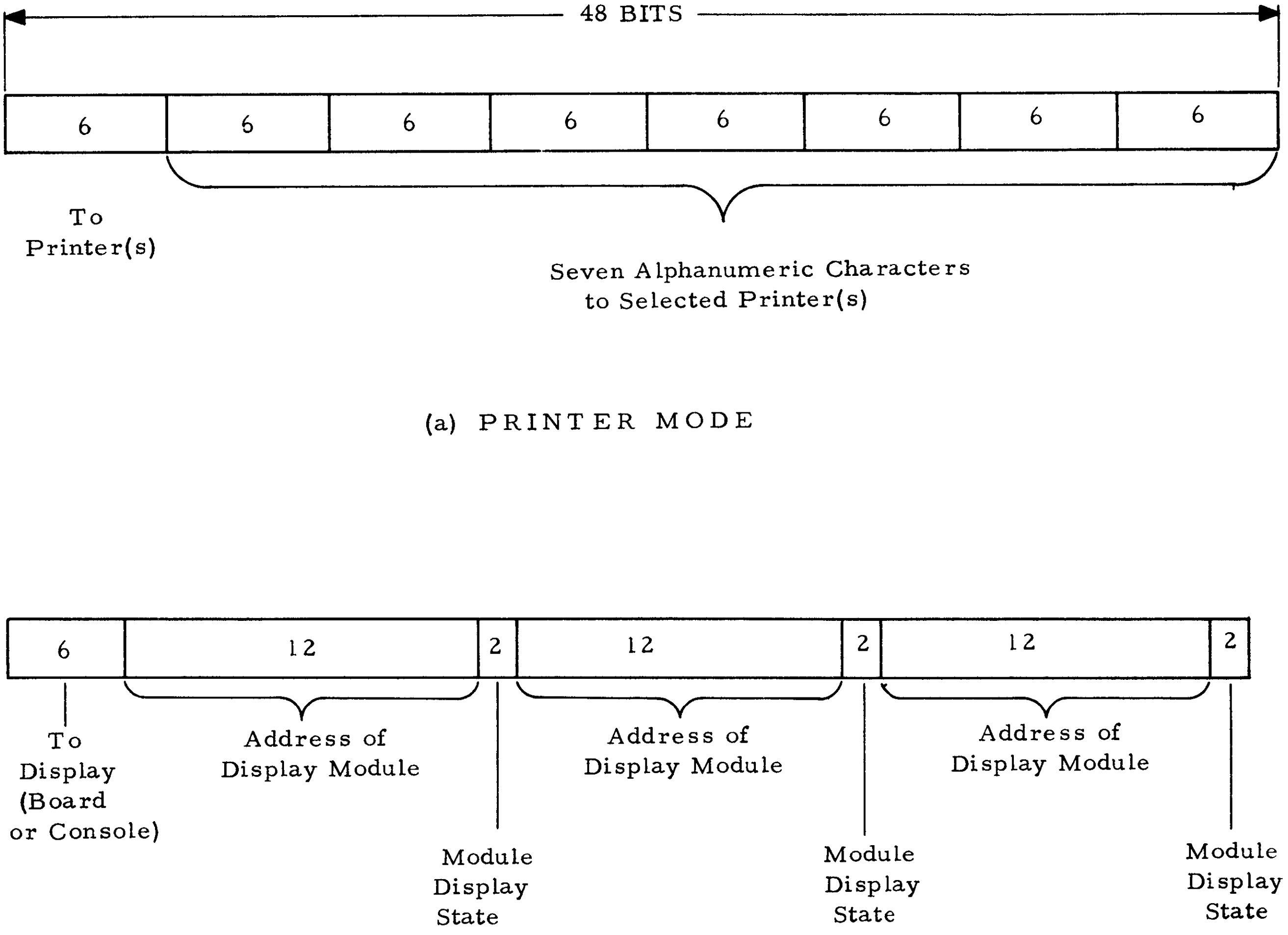
configurations: One configuration results in the generation of printed information on one or more of the high speed printers. The other results in the updating of the display of status information on the displays or consoles. The composition of these words is illustrated in Figure 6. A word which activates one or more of the printers is composed of an address code, 6 bits which indicate which high speed printer(s) will print out the alphanumeric information contained in the rest of the 48-bit word, and the remaining 42 bits of the word, composed of 7 6-bit alphanumeric code designations. See Figure 6a.

The type of word generated for the updating of one of the display boards has the composition shown in Figure 6b. A 6-bit code indicates to which device the display data is directed, i.e., which of the display boards or which of the console displays. One of the 12-bit codes addresses one of the modules of that display. Two bits then designate the color presentation of that module - green, yellow, red, or blinking red. There are three module addresses and three state indications in each 48-bit display word. A block diagram of the digital system for routing and designation to module or printer is shown in Figure 7, "DNCCC Presentation Subsystem CAD-To-Display Block Diagram."

DISPLAY MODULE BINARY ACTIVATION

As indicated above, each of the displays is composed of modules. The modules are in the form of:

1. Location indicators or sections of trunk group lines on the System Status Board.


48 BITS
6
6
6
6
6
6
6
6
To Printer(s)
Seven Alphanumeric Characters to Selected Printer(s)
6
12
2
12
2
12
2
To Display (Board or Console)
Address of Display Module
Address of Display Module
Address of Display Module
Module Display State
Module Display State
Module Display State

(a) PRINTER MODE

(b) DISPLAY MODE

Figure 6. CAD to Display Words

2. High or low precedence backlog indicators on the Traffic Status Board.
3. Stations or communication links on the Special Network Boards or,
4. Station indicators on the consoles.

Each of these modules has four states of operation. The module indicates status by color, either green, yellow, red or blinking red. An operational check is automatically performed, in that, absence of color indicates a malfunction. Each of these modules has a continuous operating life of over 100 years, and also has its own four-state memory. This module memory will retain stored information even in the event of power failure. Thus, once a module is addressed and has stored two-digital bits which determine the module's display state, it need not be addressed again until that module is to change its display state. The addressing of a display module which has its own storage capability relieves the computer and routing circuitry of the burden of repetitious excitation which some types of displays need to maintain a continuous non-flickering presentation.

HIGH SPEED PRINTER ACTIVATION

The DNCCC incorporates 4 high-speed Anelex printers. One 120-character per line, 900 lines per minute printer is driven directly by the computer. Three 72-character per line printers are used in the Presentation Subsystem to furnish information to the system supervisors. One is mounted on each of the two consoles, and one, a modified printer,

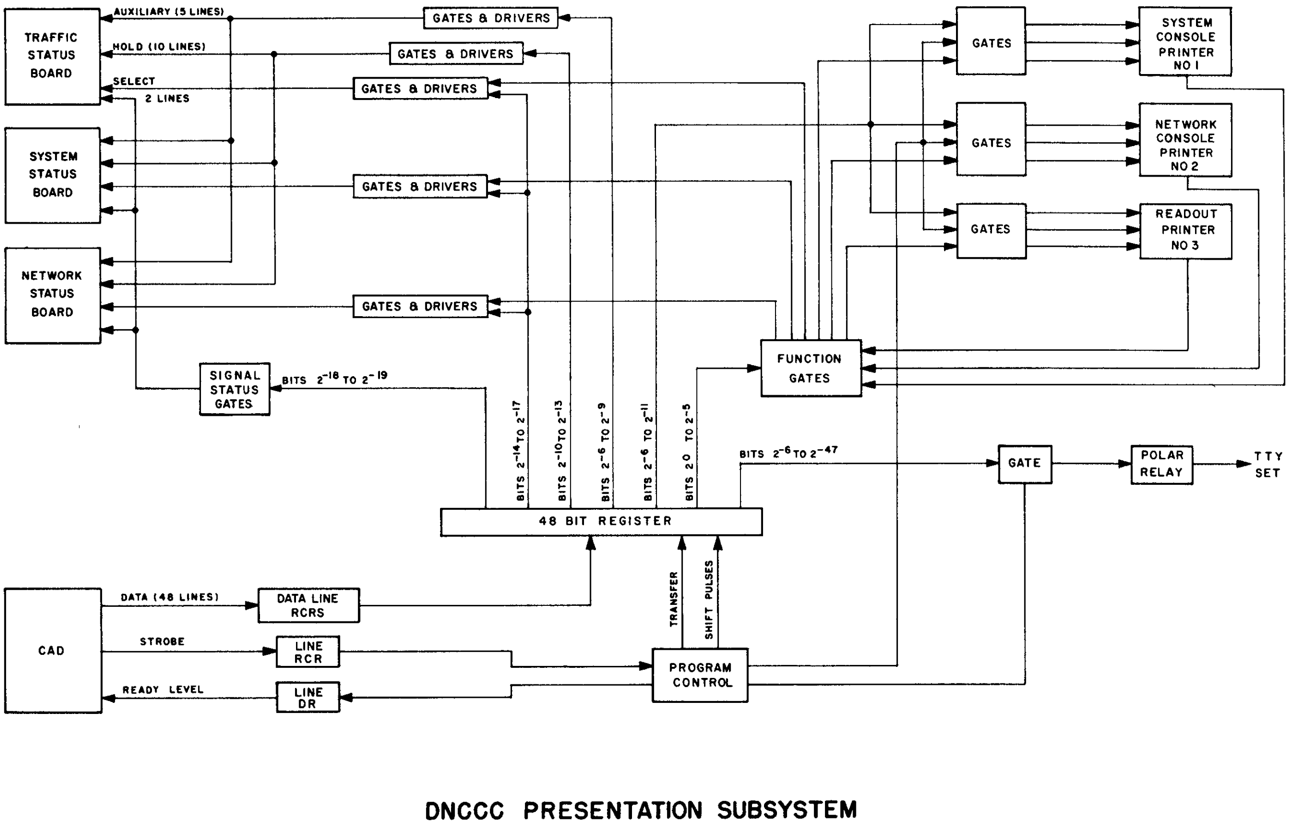


Figure 7. DNCCC Presentation Subsystem CAD to Display Block Diagram

is part of the Readout Projection system. Each of these printers has a rotating drum which has rows of metal type parallel with the axis of rotation. Each row contains 72-type representations of one character. Rows of the different alphanumeric characters are distributed around the periphery of the drum. This drum is so mounted that each row of type passes adjacent to a length of inked ribbon, on the other side of which is a length of paper. Situated behind the paper is a row of 72 magnetically-activated hammers. These are so positioned that when activated, they cause the paper to push the ribbon against a type face in one of the columns of the continuously rotating drum. The striking of a hammer is so rapid that a clear image of the letter appears on the paper. To get a line of printed information, it is thus necessary that each of the 72 hammers strike as the proper letter passes in front of it.

Rotating with the drum is a digitally encoded wheel. On this wheel there is a binary representation of each character on the drum. Associated with each hammer is a 6-bit storage register and a comparison circuit. When the digital representation of a character in a printer register is identical with that on the encoded wheel, the hammer is activated, causing the character to be printed on the paper in the position associated with that register.

Printer control-functions also appear as 6-bit codes. For economy and because of human limitation, the printers of the presentation subsystem are not operated at 900 lines per minute. The character recognition circuitry is time-shared, decreasing its speed and thereby reducing the cost and complexity.

To print a line on one of the printers, words from the CAD, directed by the display distributor, are read to the printers as a serial stream of 6-bit parallel characters. The printers commutate the characters and put them in a proper comparison registers so that the appropriate character is printed in any given line position.

The Readout Board incorporates a projection system which rapidly produces, by non-chemical means, a transparency of clear text or tabular alphanumeric information. This transparency is produced by a modified Anelex printer at printer operating speeds, similar to the ones described above. This transparency is used in projecting an easily readable image on a 7 x 7 ft. screen. The image may be read at 25 feet in a room having an ambient lighting of 25-foot lamberts, standard office illumination.

Details of this display system are described in greater detail in another paper (2).

QUERY DATA INPUT

Queries can be asked by the system or net supervisor by activation of buttons on the appropriate console. Answers to queries appear either on the printers, or, in some cases, on the displays, as described below.

CONSOLE TO CAD DIGITAL DATA FLOW

Each of the station display indicators on the System Supervisor console is a pushbutton, which, when depressed, is connected to a decimal-to-binary converter as is shown in Figure 8. Depressing a pushbutton results in a unique 8-bit word being placed in 8 of the positions of the 48-bit register associated with that console when the console is operating in the console mode. As shown in Figure 9a, 4 pushbuttons can be depressed, each contributing 8-bits to the 48-bit word which will be sent to the CAD. The following 6-bits of the word indicate the type of query being asked, the next 6-bits dictate on which printer(s) the answer to the query will be printed. The last 4-bits of the 48-bit word specify the originating console and in which mode it is operating, either console or flexowriter. If the system supervisor wishes to put in a variable format query, he must put the console in the Flexo mode. The depression of the flexowriter keys results in the storage, in the 48-bit register of the console, of 7 alphanumeric characters as shown in Figure 9b.

The symbols are simultaneously printed on the flexowriter so the supervisor has a

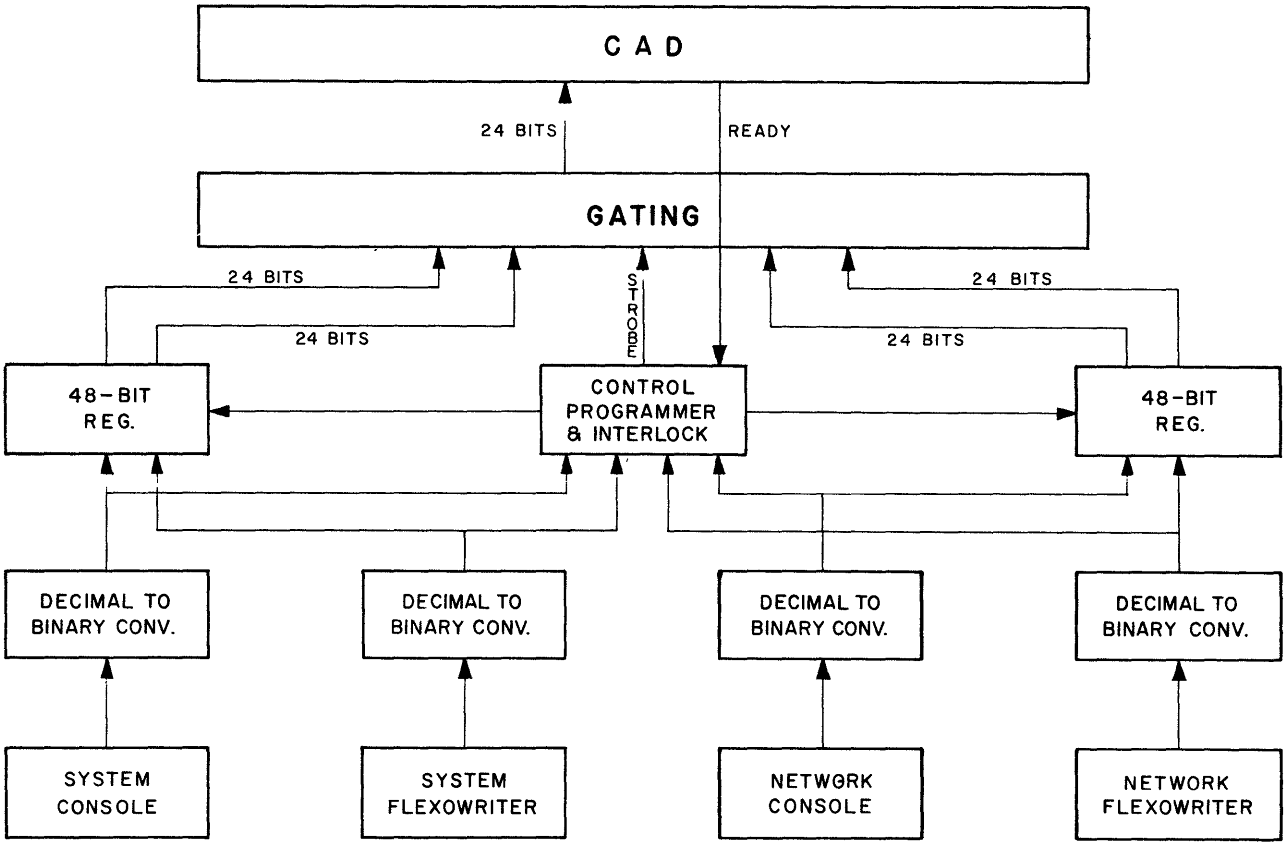
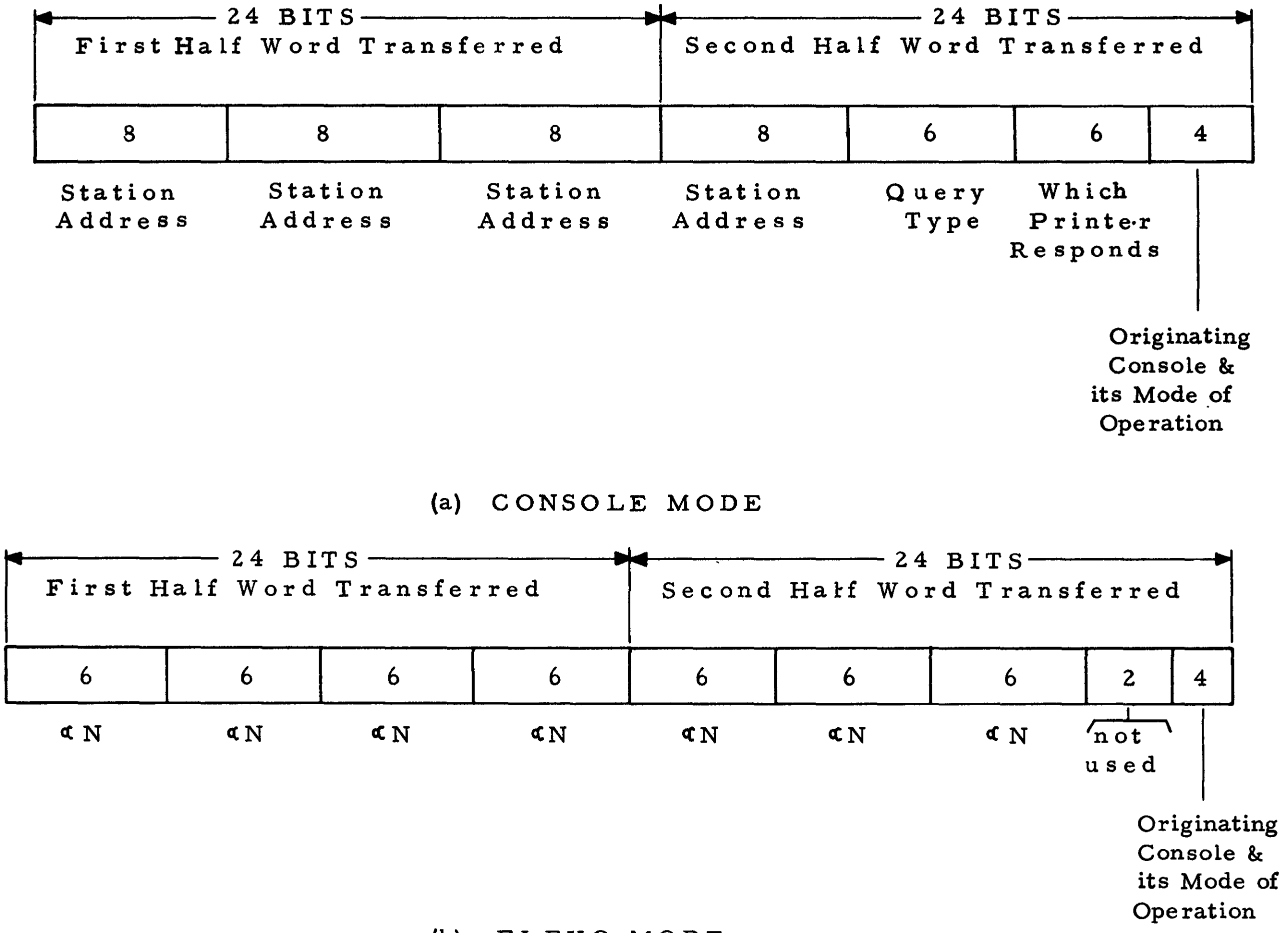


Figure 8. DNCCC Presentation Subsystem - Console to CAD Block Diagram


24 BITS
First Half Word Transferred
24 BITS
Second Half Word Transferred
8
8
8
8
6
6
4
Station Address
Station Address
Station Address
Station Address
Query Type
Which Printer Responds
Originating Console & its Mode of Operation
24 BITS
First Half Word Transferred
24 BITS
Second Half Word Transferred
6
6
6
6
6
6
6
2
4
αN
αN
αN
αN
αN
αN
αN
not used
Originating Console & its Mode of Operation

(a) CONSOLE MODE

(b) FLEXO MODE

Figure 9. Console to CAD Words

record of his variable query input. Each alphanumeric character has a unique 6-bit designation. Two bits of the 48-bit word are not used, the last four designate the originating console and the mode in which it is operating.

The 48-bit word accumulated in a console register is sent 24-parallel bits at a time to the CAD. In the event a query is initiated from the netwrok and System consoles simultaneously, the query is not lost. One is given priority to the CAD, causing the other to be delayed only a few milliseconds.

DNCCC COMPUTATIONAL SUBSYSTEM

Up to this point, the discussion of the digital and information flow in the Defense Communications Control Complex has dealt with the relationship of DCA to the associated agencies, the services to be performed by DCA, the monitoring of the DCS by the Defense Communications Control Complex, the means by which messages reach the DNCCC, the conversion of communications information and supervisor's queries to digital inputs for the computer through the CAD, and finally the method by which the computer operates the displays and printers via the CAD. It is now appropriate to discuss the computational subsystem which has been called the computer in the preceding discussion, and also to discuss its off-line capabilities. There will be no attempt in this paper to discuss the programming of this subsystem.

A block diagram of the DNCCC Computational Subsystem may be seen in Figure 10. It is composed of:

(1) Computer Access Device (CAD)
(2) Magnetic Core Storage
(3) Arithmetic Section
(4) Program Section

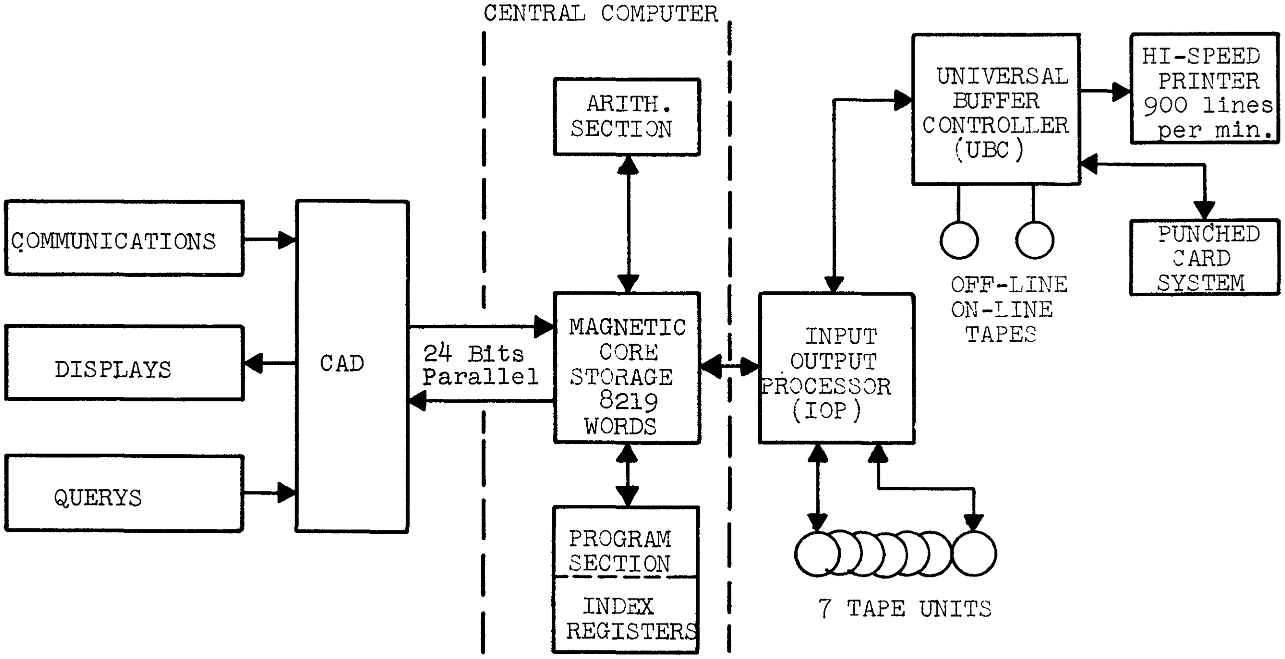


Figure 10. DNCCC Computational Subsystem

(5) Input-Output Processor
(6) Universal Buffer Controller
(7) Punch Card System
(8) High Speed Printer
(9) Nine magnetic tape units

Items (2), (3) and (4) as listed make up the Central Computer.

1. CAD

The CAD arranges all input and output data in 48-bit parallel words, this word length being one which can conveniently be handled by the Processing Subsystem.

2. Magnetic Core Storage

The Magnetic Core Storage is capable of storing 8,192 words, each one 48 bits in length. This unit contains in part a data register and an address register. The magnetic core storage is shared by central computer, the input-output processor and the CAD. This sharing interweaves the memory-access periods of the various units so that none must wait more than a few microseconds to use the memory. Therefore, with memory sharing, the program runs almost continuously, without interruption by input-output operations.

3. Arithmetic Section

Data is processed in the arithmetic section by arithmetic comparison, shifting, and extracting operations. These arithmetical operations are performed in the adder network. This network is extremely fast since it contains no storage registers and operates in parallel on all 48 bits of a word.

4. Program Section

The central computer processes the stored data in accordance with the program instructions. The program section interprets and executes the instructions.

5. Input-Output Processor

The input-output processor is the interconnecting and control link between the central computer and 16 input-output channels. Each of the channels couples either a magnetic tape unit or a Universal Buffer-Controller to the central Computer. Standard data-transfer rate over a channel is 90,000 alphanumeric characters per second. By multiplexing, the input-output processor can automatically connect any two of the channels to the central computer for input or output.

6. The Universal Buffer-Controller

The Universal Buffer-Controller acts as a buffering device between any two input-output units. It permits off-line conversion between any two media.

7. The Punchcard System

The Punchcard System can read 80-column punched cards at a rate of 2000 cards per minute. Cards can be punched at a rate of 100 cards per minute. The card image mode facilitates handling of binary information and packs 12 bits, or two characters in each column. Plug-board editing provides format control.

8. High Speed Printer

The High Speed Printer and its printer controller operate in conjunction with the Universal Buffer-Controller. The printer prints 900 lines of one hundred and twenty characters each line per minute. By skip feeding, non-printed areas are passed at ten times the normal printing rate.

9. Tape Units

The Magnetic tape units record data in fixed block form to provide efficient operation, checking, and tape reading in forward and backward directions. This bi-directional tape reading is especially valuable for saving time in sorting operations. The read-write speed is 90,000 alphanumeric characters per second. The size of a standard block is 128 words, recorded in 512 frames. Each frame contains 12 data bits, two parity bits and two timing bits. As an additional control, a 513th frame adds to each of the fourteen horizontal channels an even parity bit for longitudinal parity checking of the block.

The above listed components are interconnected as dictated by the Programming, to perform the functions needed by the DNCCC Computational Subsystem. A few examples of the internal computational equipment connections required for performing some of these functions are illustrated in Figure 11 and given below.

Mode 1. Initial load - taking programs which have been manually punched into cards and entering these programs into the appropriate computer facility storage areas.

Mode 2. Load to correct - used to enter changes or corrections into already utilized storage areas.

Mode 3. In-line path I - normal operating condition for the system.

Mode 4. In-line path II - operating condition in the event of a detected error in the input data.

Mode 5. Query path I - a console request for stored information to be presented on display subsystem equipment.

Mode 6. Query path II - a console request for stored information to be printed using the computer facility high speed printer.

Mode 7. Tape Print - the transfer of information from magnetic tape storage to hard copy via the computer high speed printer. Used primarily to print out Error Tape data and Record and Report Tape data.

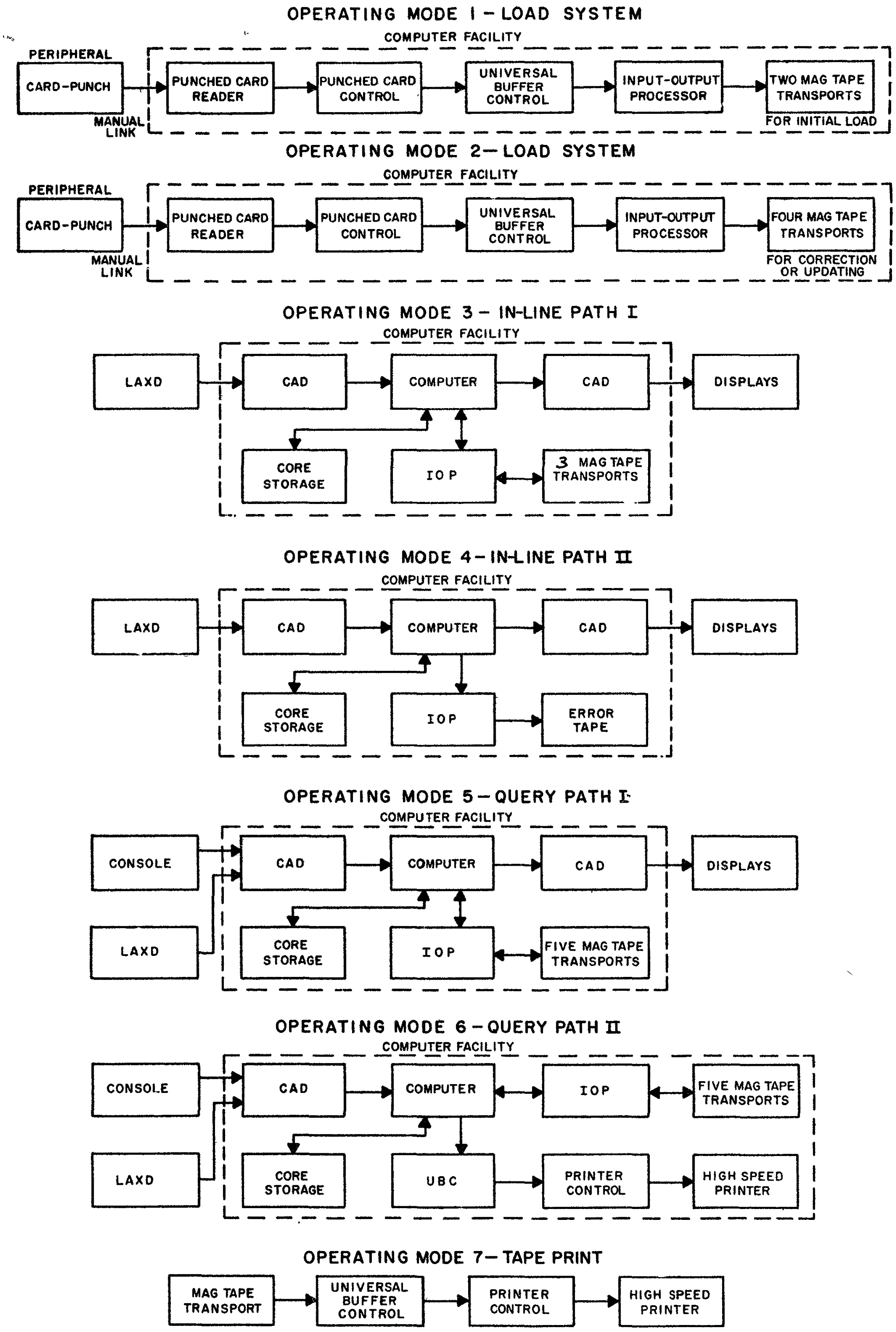
The asynchronous nature of this Philco 2000 Computational Subsystem enables very rapid up-dating of the data base, manipulation and direction of data flow, retrieval of stored information and presentation of answers to queries. A well rounded understanding of the general operation of the DNCCC should include a knowledge of the manipulation of the data base as well as the control center method of deriving and disseminating information.

DATA BASE

One element critical to effective DNCCC operation is the Data Base. This is a representation of the DCS which is stored within the computational subsystem. The data base includes fixed structural data describing DCS, and the more dynamic allocation of facilities to users. It also includes information needed for current operations, predicted future operations and planned operations.

CIRCUIT DIRECTORY

The circuit directory consists of that data which remains comparatively stable (e.g., trunk, channel, and circuit numbers, type of facility and service, owner, etc.). This directory is used to facilitate the control of the DCS; to analyze trunk group, trunk, channel, and circuit usage; to provide detailed information to the managers of the DCS that will assist them in making efficient use of all



OPERATING MODE 1 – LOAD SYSTEM

OPERATING MODE 2 – LOAD SYSTEM

OPERATING MODE 3 – IN-LINE PATH I

OPERATING MODE 4 – IN-LINE PATH II

OPERATING MODE 5 – QUERY PATH I

OPERATING MODE 6 – QUERY PATH II

OPERATING MODE 7 – TAPE PRINT

Figure 11. Functional Arrangement of Computational Subsystem Components

of the communication facilities of the DCS, particularly during emergencies; and to furnish data on trunk groups, trunks, channels and circuits that may be used in the future development of the DCS.

Circuit directory information is filed on punched cards which are modified, added, or removed as changes occur. A card similar to the one shown in Figure 12 is prepared for each communication link within the DCS.

STATUS REPORTS

Reports sent to the DNCCC by the data gathering centers of the DCS are called Status Reports. They are prepared in a format which enables the DNCCC Computational Subsystem to process them automatically as they are received. These contain information on outages, backlogs, routing, users, and traffic.

ERROR CORRECTION AND PURGING

Each information line of a status report is automatically checked for range and legality by the Computational Subsystem. If a status report does not pass all of the tests stipulated by the error detection program the erroneous portions of the report are forwarded to an operator at the Model 28ASR console through program selection of a magnetic tape output which is printed out on the high speed printer off-line through the Universal Buffer Controller. The error correction operator will make corrections and re-enter the data through the DNCCC communications input via his keyboard.

DNCCC COMPUTATIONAL SUBSYSTEM TAPES

The physical counterpart of the Data Base and the programming instructions in the DNCCC Computational Subsystem is found on its magnetic tapes. A better understanding of the manipulation and storage of the Data Base information within the DNCCC may be derived by reviewing the types and functions of the computational tapes used in this system.

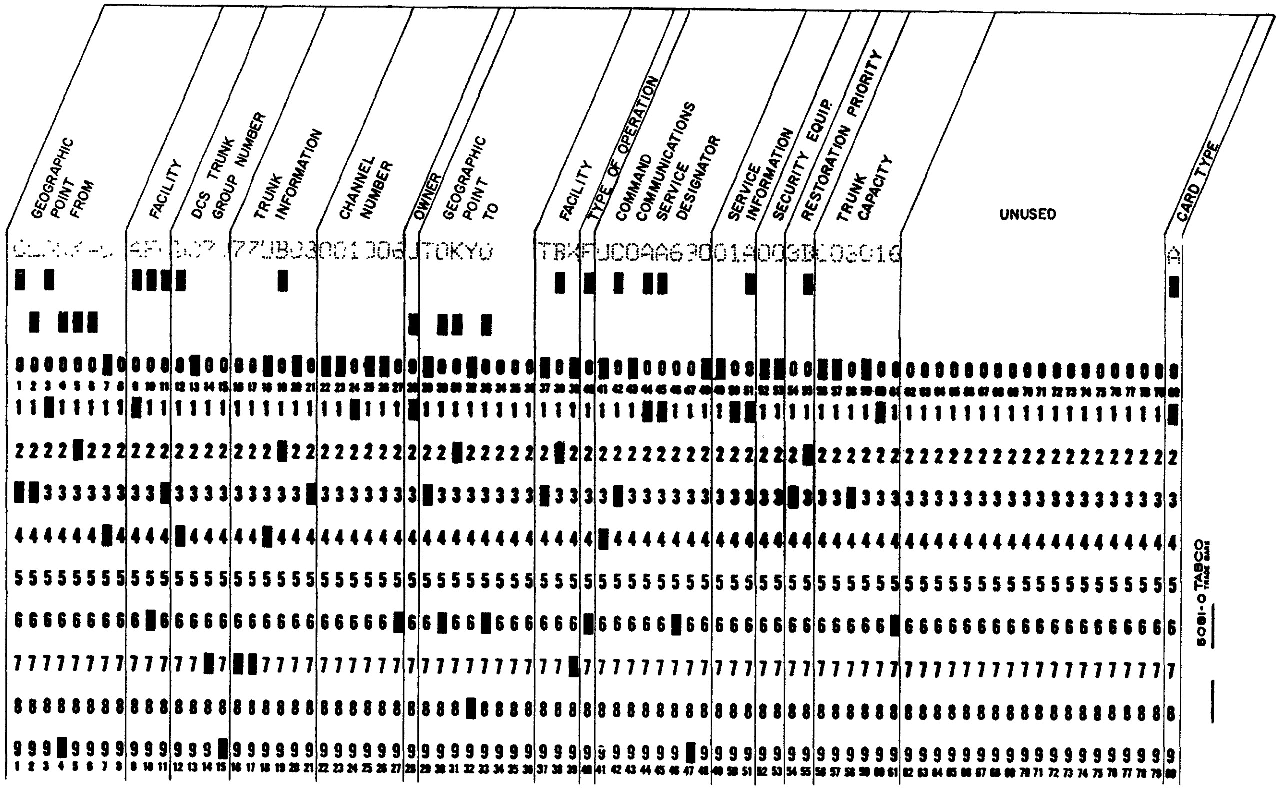


Figure 12. Circuit Directory Card

MASTER TAPE

This tape contains all the DNCCC programs and subroutines, initial system parameters, and some utility-type programs.

STATION DIRECTORY TAPE

The station directory tape includes all data on stations, trunk groups, trunks, channels and allocated circuits within the DCS.

JOURNAL STATUS TAPE

This tape contains, in chronological tabular form, all the latest information on the status of each component of the DCS.

JOURNAL STATUS PRIME TAPE

This tape contains a rearranged set of Journal Status data which has been sorted according to station by the sort and merge program.

RELIABILITY LIBRARY TAPE

This tape contains, for each trunk and channel, a reliability figure on a monthly running basis. It also contains all reasons for outages by percentage for that outage.

ERROR/OUTPUT TAPE

This tape is used to store the output of detected error contained in the incoming status reports. It is read out onto the computational subsystem high speed printer on demand.

SCRATCH TAPE

The Executive and Start-over routines use this tape for such purposes as saving core, Journal Status backup, and intermediate working storage. Other routines which need storage in excess of that available to them in core also use this tape.

SIMULATION TAPE

For introducing controlled inputs during training periods, a simulation facility can construct tape images similar to the inputs normally read from the CAD.

OFF-LINE CAPABILITY OF THE DNCCC COMPUTATIONAL SUBSYSTEM

This paper has emphasized the on-line capability of the DNCCC. It should not be inferred that the computational subsystem is used more than a small portion of the time to carry out this on-line function. The design of the Defense National Communications Control Center has given it inherent processing and display storage. For all the displays have their own internal storage either in display modules or in the form of hard copy, thus eliminating the need for repetitive use of the computer to maintain displayed information. Tapes and core are the computational subsystems storage media. This storage capacity allows the computational subsystem to devote most of its processing time to the performance of computation required for off-line management of the Defense Communications System.

SUMMARY

The purpose of this paper is to give the reader an understanding of the method of handling information "on-line" in the Defense Communications Agency's Defense Communications Control Complex.

The control system described here is only the initial phase of the implementation of the Defense Communications Agency Control Center Complex (DCA CCC) a series of communications control centers throughout the free world.

CONCLUSION

On 29 September 1960, the Defense Communications Agency selected Philco Corporation as prime contractor to set up the Defense National Communications Control Center discussed above. Normally such a program would take more than a year to complete. In less tahn 22 weeks, however, Philco designed, developed, manufactured and installed the complex equipments which make up this Center.

REFERENCES

1. "Defense Communications Control Complex," T. J. Heckelman, presented at 7th National Communications Symposium, Utica, N. Y.

2. "Coupling Man, the Decision Maker, to the Defense Communications Control Complex," R. H. Lazinski, presented at 7th National Communications Symposium, Utica, N. Y.

AN AUTOMATIC DIGITAL DATA ASSEMBLY SYSTEM FOR SPACE SURVEILLANCE

Marvin S. Maxwell
U. S. Naval Weapons Laboratory
Dahlgren, Virginia

ABSTRACT

The Navy Space Surveillance System (SPASUR) was developed to detect, track, and determine the orbital elements of unannounced non-cooperative satellites. To improve its response time and data handling capability, the Spasur System is being made automatic through the application of digital techniques. Digital data from many remote stations will be sent continuously at 2,500 bits per second over telephone data lines to a central assembly system. Selected data is assembled into a form for entry via a tape channel into an IBM 7090 computer for on-line processing to produce orbital elements within seconds of satellite passage through the detection beam.

INTRODUCTION

The Navy Space Surveillance System, known as SPASUR, was developed by the U. S. Naval Research Laboratory (Washington, D. C.) to detect, track and predict the orbits of unannounced non-cooperative satellites passing over the United States. This system (1) has been in operation 24 hours a day since February 1959. In February 1960, the U. S. Naval Space Surveillance System (NAVSPASUR) was established under a Commanding Officer to operate the system to meet Navy and national requirements. NAVSPASUR is under the operational control of the North American Air Defense Command.

The Spasur System now requires a considerable amount of manual data handling. To improve its capability an automatic system for processing the data is now under development. The Naval Research Laboratory has overall system development responsibility. The Naval Weapons Laboratory, Dahlgren, Virginia and the Naval Research Laboratory have joint responsibility for the digital equipment at the remote stations and the telephone data lines. The Naval Weapons Laboratory is also developing all of the equipment associated with the automatic system in the Operations Center at Dahlgren, Virginia and the computer programs for final reduction of data.

A brief description of the detection system and general operating procedures will provide useful background information. The detection network consists of 4 receivers and 3 transmitters located on a great circle of the earth at about 33°N latitude. The transmitters each send out a continuous wave of radio energy in a fan shaped pattern which is very narrow in the north-south direction and very wide in the east-west direction. The receiving systems have similar antenna patterns and all are coplanar so that when a satellite enters the transmitter antenna pattern, it reflects back a signal to one or more receiving stations. The angle of arrival of the reflected signal is measured in both north-south and east-west direction by radio interferometry, that is, the relative phase of the detected signal between each of many antenna pairs is measured. Precision in angular measurement is achieved by use of very long base line separation in some of the antenna pairs; a series of progressively-shorter base line antenna pairs is used to eliminate ambiguity in the angles measured with the longest base lines.

The data taken from the receiving station provides information which makes it possible to determine the time an object passes through the beam, the sight angle to the object, a measure of the rate of change of angle in the north-south and east-west directions, a measure of the doppler shift and of the instantaneous signal strength of the reflected signal.

The only information sent to the computer via the Automatic Digital Data Assembly System (ADDAS) is that data identified by equipment in the remote station as probably containing information about a satellite. Data to be sent to the computer is never delayed more than one second between the time the measurements are made at the receiving station and the time the data is in the memory of the computer.

The basic operation of the automatic space surveillance system is as follows: An IBM 7090 computer using the orbital elements of all known satellites, generates predictions about each satellite as to when it should cross the transmitter beam, and estimates of the observation angles from the ground stations, plus other parameters. Data from the Remote Stations indicating a possible satellite is processed through the data assembly system and sent to the computer where additional processing occurs and further checks are made to see if the data is valid. All valid data is reduced to generate an observation specifying the time and angles from the various ground stations to the object as it went through the transmitter beam. If this observation is sufficiently close to a predicted set of values for a known satellite it is assigned as a new observation of that satellite. After a number of new observations are assigned to a given satellite, they are used to generate new, more accurate, orbital elements which are used to generate new, more precise, predictions to be compared against future received data. In this manner the system keeps track of all known satellites. If a valid signal is detected that does not correspond to a prediction, this observation is classified as an unknown, and a preliminary orbit is estimated from the data; an immediate message is sent from the computer via a direct communication channel to selected users of this information. The system design is such that the computer should be capable of sending out these preliminary orbital elements within seconds after the unannounced satellite passes through the transmitter beam. After further passes of this new satellite through the transmitter beam, a more precise orbit can be established and this satellite will be entered into the list of known satellites. The computer has available to it many programs which generate the many types of tabular and graphical output required and which also monitor the performance of the detection and data assembly system.

This paper will give a description of the equipment used in assembling the data from the remote sources into a suitable form for processing in the computer. Some of the design details of the ADDAS have resulted from the original plan to employ the Naval Ordnance Research Computer (NORC) for processing ADDAS output. An example is the inclusion of a binary to BCD conversion in the message assembler, since the NORC is a binary coded decimal computer. While the use of an IBM-7090 is now intended, compatibility of the data format with NORC has been retained.

Automatic Digital Data Assembly System - ADDAS

The ADDAS system is designed to facilitate fully automatic operation from the initial reception of signals at the remote stations to the introduction of data into the computer at NWL.

The system consists of the digital equipment at the remote stations, the digital equipment at the Space Surveillance Operations Center, and the communication links between them. Important practical considerations in the selection of the data transmission equipment and in the design of the ADDAS system were as follows:

a. 24-hour operation for continuous surveillance.

b. Short delay from the time data is detected at the remote station to the time it is presented for processing in the computer.

c. Multiple data sources - initial system to include 4 sources with capability for expansion to as many as 15 sources.

d. Remote data sources - from 500 to 3000 or more miles distant from the Space Surveillance Operations Center.

e. Random data times - any number of stations, from none to 15, reporting significant (alerted) data at any instant of time.

f. High data transmission rate - 2500 bits per second, because of the large quantity of data needed during the time a satellite is being observed.

g. A record on magnetic tape at the Space Surveillance Operating Center of all data received from every source, for use in special searches or to recover data lost by equipment malfunction.

h. A magnetic tape record of all data sent to the computer, to recover from computer malfunction.

i. Analogue display of selected channels of incoming data, for monitoring system performance.

The system shown in Figure 1 meets all of these objectives. The system currently under construction is being equipped for the 4 present SPASUR stations; however, the design will accommodate up to 15 remote stations without modification.

The information flow in the ADDAS starts at the Remote Station where all of the needed data is digitized and sampled at a nominal rate of 19.5 observations per second per channel. This data, along with guard bits for

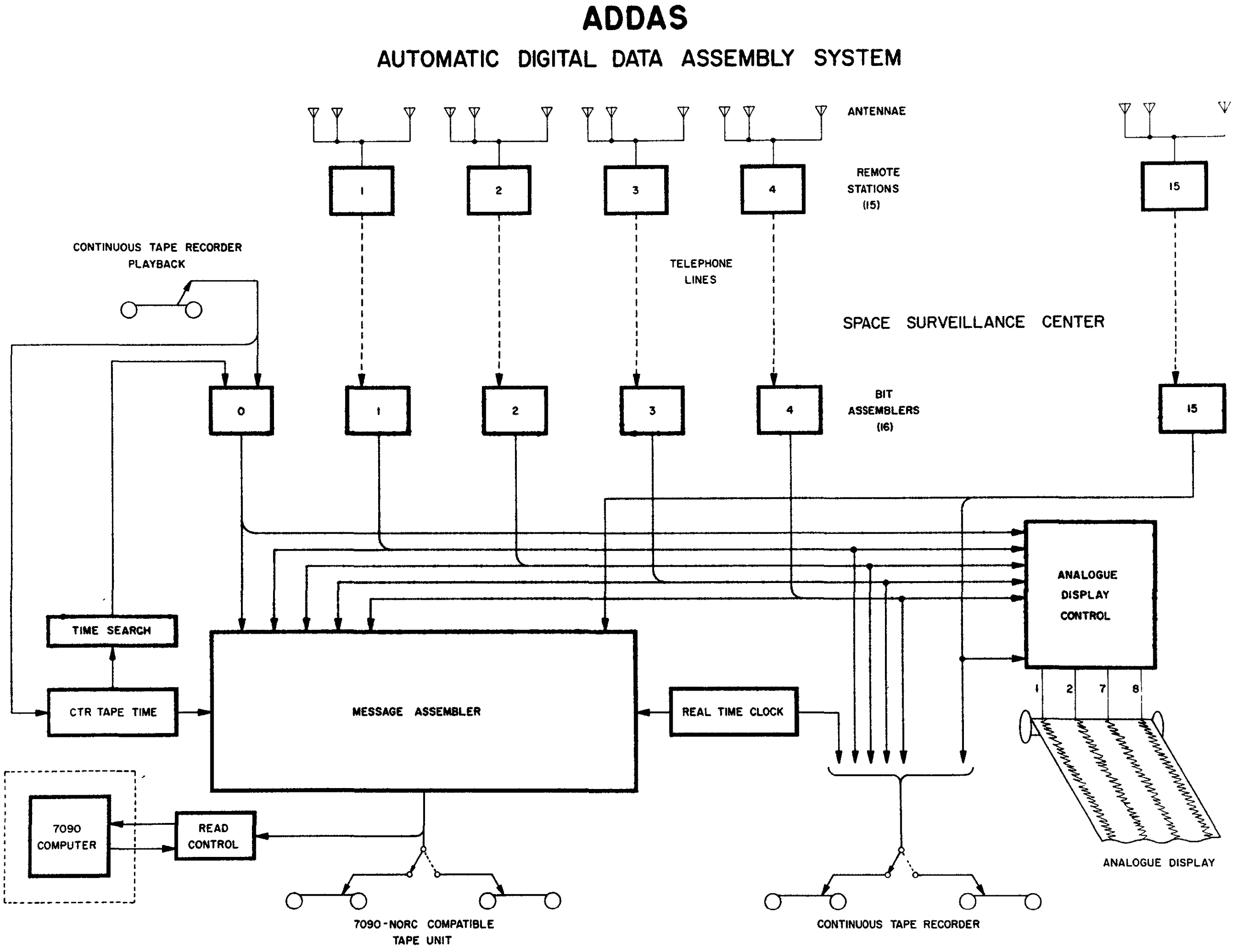


Figure 1

a frame mark, alert bits, and three parity check bits is assembled into a 128 bit frame. The parity check is a modulo 8 count, minus 1, of the binary 1's transmitted; a transmission failure resulting in a frame containing either all 1's or all 0's will be detected as an error. In each frame 117 bits are available for data; a patchboard, with replaceable inserts, allows flexibility in partitioning these bits among the various channels so as to send the necessary data with the maximum precision. A channel generally measures a single quantity, representing its value in pure binary, and may use 4, 5, 6, or 7 bits, except that a 7 bit channel is restricted to a maximum value of 99. The partitioning rules are such that the maximum number of channels is 24 and the minimum is 17. The data is transmitted serially, bit by bit, at a rate of 400 microseconds per bit by Rixon Sebit 24 high speed data modems over telephone lines to Dahlgren, Va.

In the current stations approximately 21 channels of data are digitized with a precision between 1 part in 32 and 1 part in 100 and assembled into a frame. A valid signal from a satellite may have a duration as brief as 0.2 seconds, a nominal duration of 0.75 to 1.5 seconds and a maximum duration of many seconds depending upon the orbit and size of the satellite. Equipment in the remote station continuously monitors the received signals and, when the signals have certain characteristics which are present in most valid satellite signals, an alert is generated and a bit pattern is set in the frame which is then called an alerted frame or alerted data. To reduce the false alert rate to manageable proportions, it is also required that the alerting conditions must be present for a minimum time interval, nominally 0.2 seconds, before the alert is marked in the frame. This means that there are nominally 2 to 4 frames of valid data preceding the first alerted frame.

The digitized data from each station is received continuously at the operations center at 2,500 bits per second and converted into standard digital form by the data modem. All of the data received from one remote station enters the Bit Assembler assigned to that station. In the Bit Assembler the data is synchronized, checked for message format and parity and is grouped into characters of 4 to 7 bits, one character per channel, and transferred to the Message Assembler. The Bit Assembler inserts a special character as the last character in each frame which indicates if any unusual conditions were detected during receipt of this frame, such as frame synchronization troubles, maintenance switch in use, or detected transmission failures. The Bit Assembler prepares the address specification of the location in the memory of the Message Assembler where the data is temporarily stored. The alert status of the received frames is detected in the Bit Assembler and it gives control information to the Message Assembler so that the appropriate data is assembled and sent to the computer. The Bit Assembler also sends data to the Analogue Display Control and within itself it converts any preselected 4 channels of data from digital form to 4 analogue voltages and makes the signals available to a remote recorder.

In order to store the data for possible exceptional handling each Bit Assembler immediately sends out all of the received data to be recorded in real-time along with the time of day on one of two slow speed, 7 1/2 inches per second, Continuous Tape Recorders. A single recorder can store the data from 9 different stations plus the time code, and two recorders are operated in alternation to provide full time coverage of 24 hours per day. The data from any single station along with the proper time code can later enter the system via the Continuous Tape Recorder Playback and Bit Assembler #0 at a normal rate or at 8 times normal rates for search for special data or for recovery from malfunctions. The CTR Tape Time unit presents the recovered CTR time to the Message Assembler, the Time Search unit and to a visual display unit. The Time Search unit allows the operator to specify a start time and stop time to gating circuitry which conditionally enables the transfer of data into the Message Assembler.

The Message Assembler receives data from the Bit Assemblers via a multipole 16 position electronic switch which scans all of the Bit Assemblers sequentially (except for #0) until it locates one with a character to be stored in the assembly region of its memory. The memory consists of 2048 words of 18 bits each with random access time of 10 microseconds. Each character with its parity bit (which is generated in the Message Assembler) occupies one-half of a word and the address is a combination of the bit assembler number to which the electronic switch is set and a frame count and character count which are sent from the Bit Assembler. Each Bit Assembler has its own assembly region in memory which holds 6 frames of data. Bit Assembler #0 is scanned by the switch eight times as often as any of the others since it can present data from the Continuous Tape Recorder Playback at 8 times the normal rate. When the Message Assembler observes by inspection of the address specification of the character being sent from the Bit Assembler that it is the first character of a frame, the Message Assembler goes into a special cycle storing in a part of the memory region for that frame the Bit Assembler number (station number) sending the data and the current time of day from the Real Time Clock. The time is specified in Julian days mod 100, subdivided into hours, minutes, seconds, and hundredths of seconds.

The Message Assembler has a second, independent, 16 position electronic switch which scans to see if any Bit Assembler has detected an alerted frame. When this condition is detected, the oldest complete frame from this Bit Assembler, which was received 5 frames before the present frame, is transferred from the assembly region of memory to the next available space in a buffer region of memory. The scanner then proceeds to the next bit assembler to see if it has an alerted frame. This procedure assures that the five frames preceding the first alerted frame are sent to the buffer region, and from there to the computer, but it also would cause the system to lose the last five alerted frames. To prevent this, the Bit Assembler contains a circuit which extends the alert condition to the Message Assembler for 7 frames following the last alerted frame. This extending of the alert causes the Message Assembler to then transfer to the buffer region the remaining 5 alerted frames plus the following 2 unalerted frames.

There are two buffer regions in the memory of the Message Assembler, each holding 16 frames of data. When one of the regions is filled, a cycle is initiated to transfer this data directly into the computer and onto magnetic tape. While one buffer region is being emptied, data may be loaded into the other buffer region, and when this one is full, its data sent out. A double length record is generated whenever the second buffer is completely filled before the first buffer is completely empied.

The data transferred out of the buffer region for entry into the computer is converted from binary to binary coded decimal by a modification of a shift and convert procedure (2) and assembled into a NORC and IBM high density compatible tape format. The binary data in each channel is converted into a two decimal digit number and a complete frame consists of 32 digit pairs of information or 64 information decimal digits plus 8 special characters required for NORC magnetic tape compatibility. The 32 information digit pairs are assigned in the following way. The identification of the Bit Assembler or Remote Station number which is sending in the data uses one digit pair. The time code uses 5 digit pairs, one each for days, hours, minutes, seconds, and hundredths of seconds. There is one digit pair for the alert indication, 24 digit pairs for the data (one for each of the possible data channels) and one digit pair for the summary data generated by the Bit Assembler. Each of the digit pairs could span the values from 00 to 99 but the digit pair for a 5 bit channel for example will have a maximum valid value of 31. If the station configuration is such that all of the possible 24 information channels are not used, the unused channels are filled with zeros.

This data is sent out at twenty microseconds per character for recording on magnetic tape at one hundred inches per second on one of three units and to level conversion circuitry for direct entry into an IBM 7090 computer. ADDAS data enters the IBM 7090 via a tape channel used exclusively for this purpose. There is special level conversion circuitry with outputs that are equivalent to those generated by an IBM 729-IV magnetic tape transport. Thus, the computer acts as if it is connected to a standard tape unit, while it is actually connected to the ADDAS. When operating, the computer continuously attempts to read data from this "tape unit" and the channel "stalls" (but the computer keeps running) until the ADDAS transfers data to the level conversion circuitry which sends it via the tape channel into the memory of the computer, generating a program interrupt or trap at the completion of the data transfer. Each frame of data occupies twelve computer words and in the tape image there is an additional three word beginning of record and a three word end of record identifier. Thus a standard sixteen frame record occupies 198 words in the computer.

The rate of data transfer from the ADDAS to the computer varies from zero, when there are no satellite type signals being received at the Remote Stations, to very high rates as indicated in the following table. The Continuous Tape Recorder (CTR) data is played back at 8 times normal rate so that its flow rate is equal to that of 8 stations.

Sixteen frame records of 198 computer words are generated at a rate of 1.2 records per station per second if the station is sending alerted data. This leads to the following rates if all of the named stations are sending alerted data:

1 station	1.2 records/sec.
4 stations	4.8 records/sec.
10 stations	12.0 records/sec.
15 stations	18.0 records/sec.
CTR	9.6 records/sec.
CTR+4 stations	14.5 records/sec.
CTR+10 stations	21.6 records/sec.

At CTR+10 stations and greater, occasional records of 32 and 48 frames will be generated due to the lack of synchronism among the data sources, and the fact that there is insufficient time to generate the interblock gap with this high average data flow rate.

The Real Time Clock presents to the Message Assembler the time of day in days, hours, minutes, seconds, and hundredths of seconds which are established by counting down from a precision 100 kc source. The day counter presents Julian days taken modulo 100. There are the usual provision for setting, and advancing or retarding the clock as it is compared against WWV transmissions.

The 100 kc signal from the clock is distributed through the system as a pulse one microsecond wide occurring every ten microseconds for use as a master timing signal for synchronization of data transfers through the system. The stepping of the stages of the clock that are used to develop the time of day in the Message Assembler are conditioned by timing signals from the Message Assembler so that all carries have been propagated from stage to stage before the clock is read.

With the Analogue Display Control and its associated eight channel thermal pen recorder, the operator can easily select and view any eight channels of data from any remote station. Old data can be reviewed by playing the Continuous Tape Recorder data into Bit Assembler #0 and selecting it as the input for the Analogue Display Control.

The ADDAS provides the input data for an on-line real-time space surveillance operation which allows no scheduled maintenance time for this equipment. The organization and design of the system is such as to achieve a maximum of good time without duplication of the entire system. The operational system will eventually have two computers with rapid switching of tape and disc files between them and the ADDAS data will be available to both computers. Other input and output of the computers will be switched between the devices and the direct communication channel on each computer. Within the ADDAS the Continuous Tape Recorders provide for data recovery from malfunctions in the Message Assembler or in most of the Bit Assembler. The IBM compatible output tapes provide for data recovery if both computers should be down at the same time. For both of these tape systems there are three identical units, so that there is a spare unit available for maintenance which can be put into service with no loss of data. There is a spare telephone line data modem located in a Line Terminal rack where these lines can be monitored and checked. This spare data modem can be switched onto any data line and its output switched into any desired Bit Assembler if its data modem fails. If a Bit Assembler fails, Bit Assembler #0 can be rapidly substituted for any of the other Bit Assemblers. The design of the Bit Assembler is such that it could easily send its data to two Message Assemblers, however it is not currently planned to duplicate the Message Assembler. There is a spare Real Time Clock which can rapidly be substituted for the standard unit.

To speed up the process of locating troubles and repairing them there are a variety of built in checks and maintenance aids. In the Remote Station there is a "Station Exerciser" which generates test wave forms which can be passed through all of the data channels. Detection of an alert condition by other equipment at the station automatically returns the station back to normal condition. There is also a provision to transmit random patterns over the data modem to aid in equalization of the telephone line.

The "ADDAS Exerciser" is located in the operations center and is an important maintenance aid. This unit can generate various patterns which are inserted into the Bit Assemblers automatically at periodic intervals and replaces the data from the Remote Station for approximately 1/2 second. These patterns are chosen to check as many of the features of the system as possible. The Exerciser cycles through all of the Bit Assemblers, one after another, inserting appropriate patterns for each one. This data passes through the Bit Assembler, the Message Assembler and into the computer where a check is made that each pattern is precisely as it should be, and if an error is detected, the maintenance staff is alerted. The unit then serves as a maintenance aid by generating controlled patterns while the defective element is being repaired. The exerciser system has another independent section which is used to check and maintain both types of magnetic tape units in the system.

There are also a variety of built-in checks in the system. There is a check in the Remote Station that its counters used in digitizing the phase data remain locked in step to a reference pulse. The Bit Assembler checks that the received data has the proper structure and parity, and failures are logged on an event monitor; if the failure rate becomes too high, an audible alarm is generated to alert the maintenance staff and a signal is automatically sent back to the remote station to have it start running local recorders. The Bit Assembler also checks that each character and alert condition sent to the Message Assembler is received. Within the Message Assembler there is a parity check on the memory, and checks on the binary to binary coded decimal converter if it generates an output digit greater than 9 or receives an input value greater than 99. There are also automatic read after write checks on the tape units in the system.

The size of the system can be estimated from the following information. Each of the following units occupies a space approximately equal to that of a 19 inch relay rack; the Real Time Clock, each Bit Assembler with its data modem, the Analogue Display Control, the Analogue Display Recorder, each Continuous Tape Recorder, each IBM compatible tape unit, the Level Conversion circuitry to the IBM 7090, the CTR Tape Time and Time Search unit, and the Exerciser. The Message Assembler with its memory occupies four racks. There are a variety of other units in the system, such as Master Control Consoles, Telephone Line Terminal rack and power distribution and monitoring controls. The present four station system easily fits into a space of 1,000 square feet, and a full 15 station system would fully occupy this space. This does not allow for the space required for the IBM 7090 computer which must be located nearby.

The status of the system as of the time this paper is being prepared in August 1961 is as follows: Data is being received experimentally from two stations, one near Savannah, Georgia and one near San Diego, California. The assembled data is being recorded on NORC-7090 compatible tapes and is being processed through the NORC for evaluation. By December 1961 all four scheduled stations will be operating and the direct link to the IBM 7090 will be checked out. The computer programs to operate on this data are currently being formulated and the automatic space surveillance system is scheduled to become operational in the summer of 1962.

REFERENCES

1. The Navy Space Surveillance System by R. L. Easton and J. J. Fleming, Proceedings of the IRE, Vol. 48, No. 4, April 1960.
2. BIDEC - A Binary-to-Decimal or Decimal-to-Binary Converter by John F. Couleur, IRE Transactions on Electronic Computers, Vol. EC-7, No. 4, December 1958.

FOUR ADVANCED COMPUTERS—KEY TO AIR FORCE DIGITAL DATA COMMUNICATION SYSTEM

R. J. Segal and H. P. Guerber
Radio Corporation of America
Electronic Data Processing Division
Camden, New Jersey

SUMMARY

Two types of computers of advanced design play a central role in increasing the performance, data handling capability, speed and reliability of the Air Force Data Communications System (Phase I--ComLogNet). They perform decision-making, control, checking and auxiliary data processing functions not previously attainable in data communication systems. Store-and-forward message switching and the circuit switching functions are accomplished in a number of Automatic Electronic Switching Centers. A typical Switching Center incorporates two Accumulation and Distribution Units and two Communication Data Processors. An additional off-line computer used to process magnetic tape from the system is also described.

A brief description of the ComLogNet from a data handling point of view is presented. The input-output devices for the system are described. Functions performed by ComLogNet and the equipment used to mechanize these functions are discussed. The on-line computers (Accumulation and Distribution Units and Communication Data Processors) are described in some detail. The off-line computer (Tape Search Unit) is similarly described.

1. THE COMBAT LOGISTICS NETWORK

The application of modern computer technology and automatic techniques to logistic data operations provides the Air Force with more effective control of its personnel and materiel. The Combat Logistics Network is a high-speed data transmission and switching network involving high capacity telegraph and microwave communications which link remote transmission stations (Tributary Stations) to automatic switching facilities (Automatic Electronic Switching Centers). The Switching Centers automatically carry out military communications among members of the network by acting as "clearing houses" for messages on a priority basis. Subscribers can communicate with each other despite differences in their native codes, formats, speeds, control and operational requirements. In addition, each Switching Center can be supplied with Tape Station Converters to permit high-speed data exchange with local computers by means of magnetic tape.

Initially, ComLogNet channels operate at 75, 150, 1200 or 2400 bands. Both user-to-user circuit switching (CSU) and store-and-forward message switching (MSU) capabilities are provided. An MSU can provide either a 50 or 100 full duplex channel capability. A

CSU can service up to 200 subscribers. CSU subscribers can avail themselves of MSU facility and vice versa. The initial computer complex is equipped to handle approximately 7 million punched cards or equivalent daily.

The Western Union Telegraph Company has been appointed Systems Manager of the Combat Logistics Network by the Air Force. The major digital data handling equipment, including the Circuit Switching Unit and the Automatic Electronic Switching Centers, are being designed, produced and installed for Western Union by the Radio Corporation of America. The ComLogNet system is shown in Figure 1.

2. INPUT/OUTPUT DEVICES

Information is brought into or obtained from the ComLogNet message switching center via Subscriber Terminal Stations. Each station is characterized by the type of media and devices used as shown in Figure 2 and discussed as follows. Three types of media—teletype, punched cards or magnetic tape—can be used interchangeably in accordance with the logistics of military communication requirements.

a. The Magnetic Tape Terminal permits transmission at the rate of 1200, 2400, or 4800 bits per second. Magnetic Tape Terminals receive their information from and transmit to Magnetic Tape Stations included as an integral part of the terminals. The Magnetic Tape Terminals are being designed and produced for ComLogNet by the Radio Corporation of America.

b. The Compound Terminal permits transmission rates of 75, 150, 300, or 600 bits per second. The Compound Terminal receives its information from punched card or teletype equipment. Compound Terminals are being designed and produced for ComLogNet by another supplier.

c. Teletype Stations transmit data at the rate of 60, 75, or 100 words per minute in a five-element, start-stop code. Teletype stations receive or transmit their information from Model 28 Teletype apparatus or the equivalent.

In addition Electronic Data Processing equipment which is co-located with the

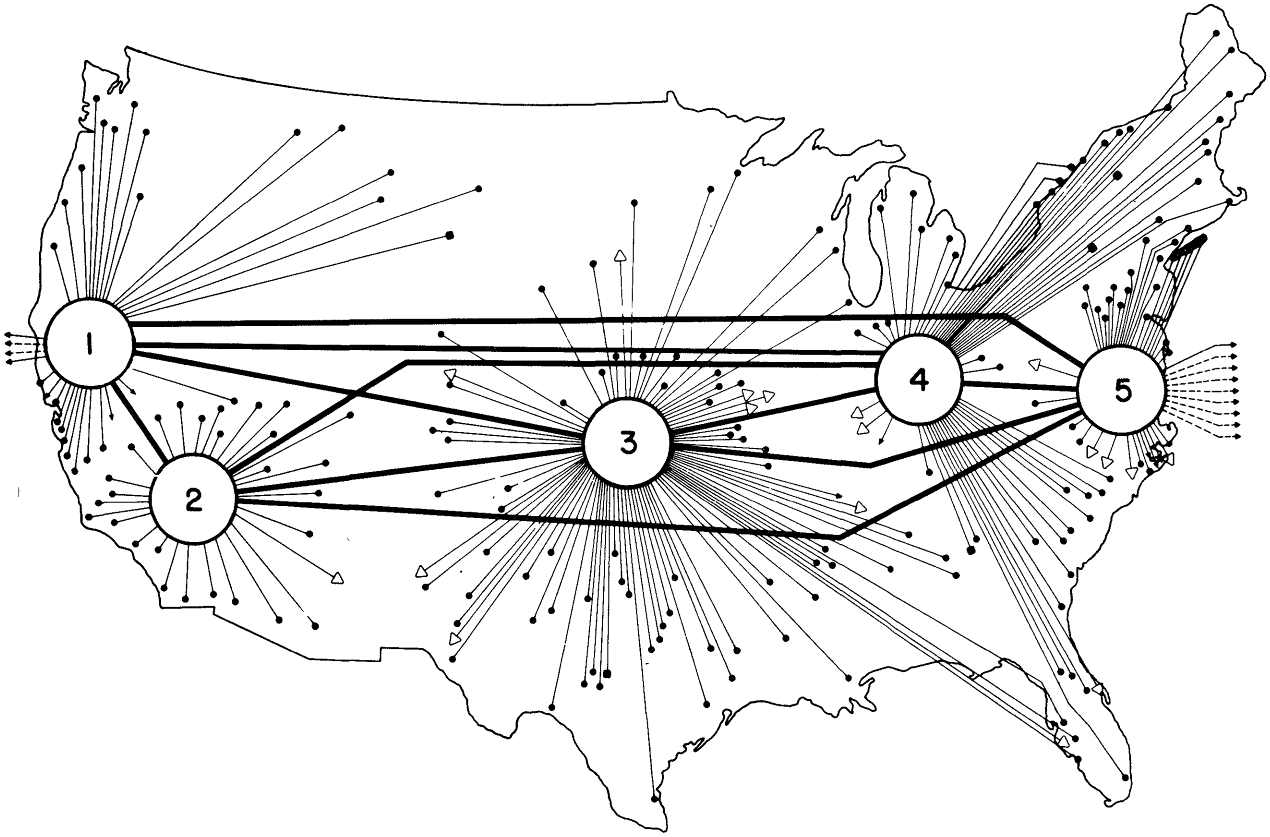


Figure 1. Combat Logistics Network - ComLogNet

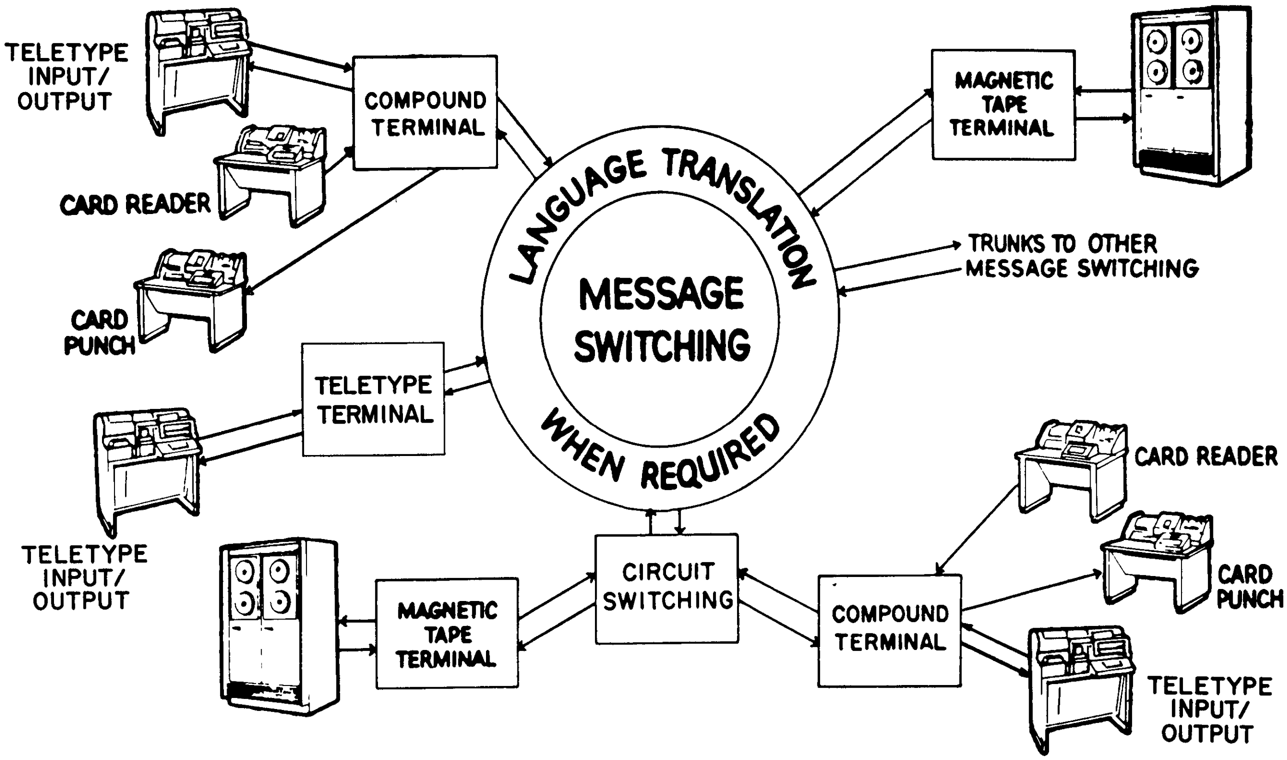


Figure 2. ComLogNet Input/Output Devices

Switching Center may send or receive messages at high speed via magnetic tape by means of Tape Station Converters. Compound Terminals and Magnetic Tape Terminals may communicate directly with the Message Switching Centers or may communicate by way of the Circuit Switching Unit.

3. FUNCTIONS OF THE AUTOMATIC ELECTRONIC SWITCHING CENTER

Functions performed by the message switching equipment are shown in simplified form in Figure 3. The basic functions may be described as the Input Function, the Output Function and the Message Processing Functions.

a. The Input Function coordinates the transfer of inbound messages. It acts as a temporary storage device between the incoming channels and message processing function. It scans each channel cyclically. It provides temporary storage for characters and performs code conversion, if necessary. This function will also automatically detect errors and assist in their correction.

b. The Output Function coordinates the transfer of outbound traffic. A limited amount of storage is provided for each outgoing channel to permit continuous transmission and electronic commutation, and to provide for automatic error detection and retransmission techniques.

c. The Message Processing Function is shown in greater detail in Figure 4. This function interprets the heading of each message and directs it to the appropriate outgoing channel or channels. As the blocks of the message are received they are transferred to the Intermediate Store; only when the message is complete does it become a candidate for transmission. As each outgoing channel becomes available, the Message Processing function transfers the oldest message in the highest precedence category for that output channel. A Communication Data Processor is used for this function. Messages to and from local electronic data processing equipment and other high-speed input/output devices have access to the Switching Center by way of the Message Processing function.

The Intermediate Store provides a reservoir for messages; thus permitting any subscriber to send a message without waiting for a through connection. Intermediate storage allows all output lines to be used in an

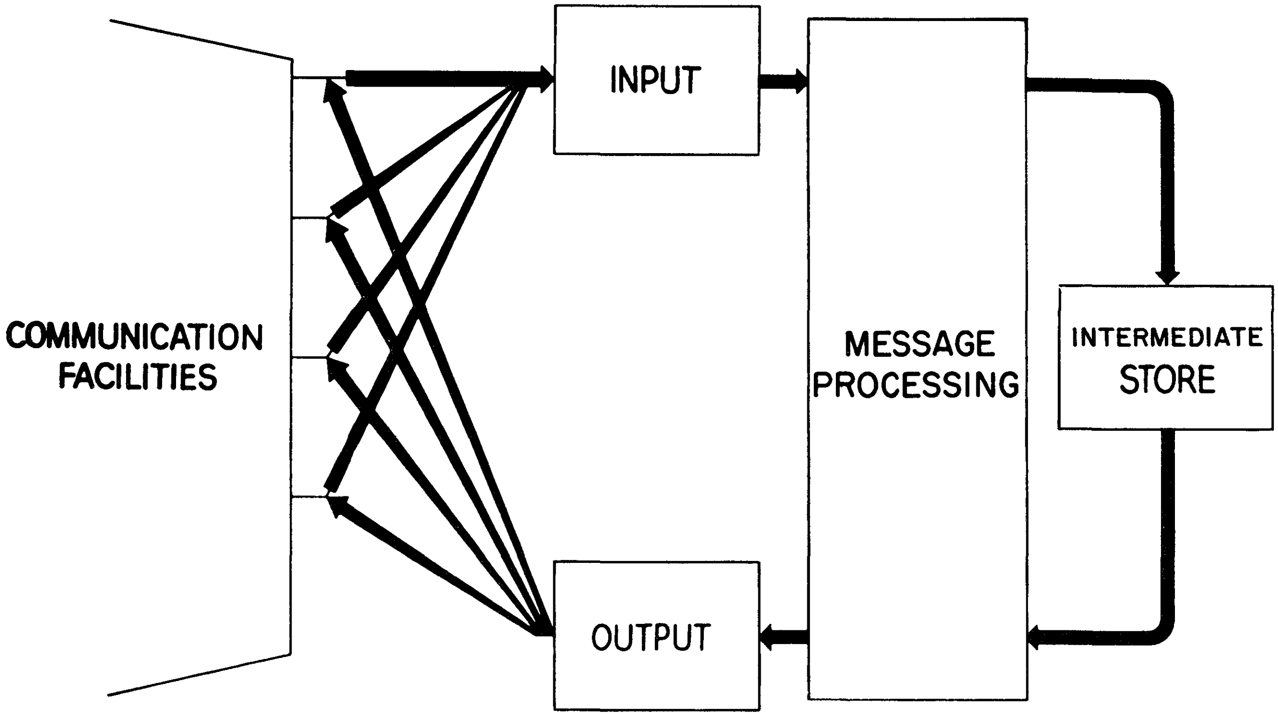


Figure 3. Simplified Block Diagram, Automatic Electronic Switching Center

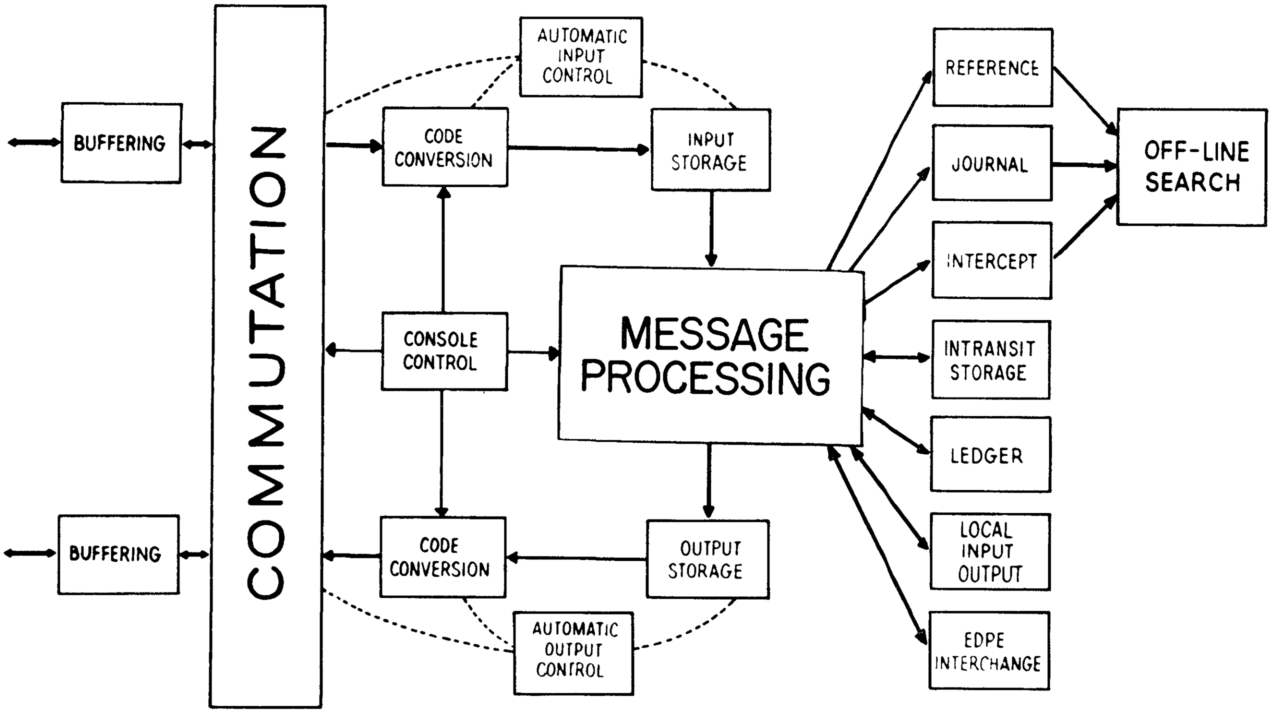


Figure 4. Functional Block Diagram, Automatic Electronic Switching Center

optimum fashion for message traffic on a first-in, first-out basis by precedence. Further, it allows top priority traffic to automatically interrupt lower priority messages and subsequently to repeat the interrupted messages.

The magnetic drums provide rapid access bulk storage for messages that are queued awaiting the availability of outgoing channels. The magnetic tapes of the Message Processing function provide larger storage for any overflow from the drums.

In certain cases the Intercept Tapes will provide additional storage capacity. When one or more tributary or trunk channels are known to be out of service, the traffic may be routed to the intercept magnetic tape station for automatic storage until the channels are again open for service. Intercept tape may also be used to store messages for channels having a heavy backlog of messages.

The following additional features of the Automatic Electronic Switching Center augment the data handling and decision-making capabilities of the ComLogNet system:

(1) The Precedence and Priority-Interrupt feature establishes the relative order in which messages are handled. Messages are transmitted in the order of their precedence. In ComLogNet six priority levels have been assigned. Certain of these levels may interrupt the transmission of lower precedence messages. Messages of precedence levels 1 and 2 are released upon collection of a complete message, even if this requires interruption of messages of precedence 3 or lower.

(2) The Alternate Routing feature permits the traffic from a particular Switching Center to be distributed over one of a number of available paths to another Switching Center according to the established trunk routing plan. The first Switching Center establishes the path to be followed. The established trunk routing plan may be changed by manual insertion of the desired program change.

(3) The Virtual Cut-Through feature provides a virtual user-to-user connection for message switch subscribers. Messages granted this service are transferred directly from the input function to the output function without using intermediate storage.

(4) The Categorization feature permits each record of a message addressed to this service to be "broken out" into one of ten separate categories, and to be sent to the relevant station which is to receive this category of information. Upon completion of the break-out process, the Switching Center automatically creates new messages which are transmitted to the appropriate terminals.

(5) The Reference feature provides a permanent copy of all messages handled by the Switching Center. Each message received is recorded on the Reference Magnetic Tape.

(6) The Journal feature provides a general traffic record on magnetic tape of the system's activity. This information includes the times of arrival and dispatching of messages.

(7) The Ledger Balance feature provides an up-to-date record of the current activity within the Switching Center, showing at any given time the status of all input and output channels and tapes. These records which are stored on drums for the sake of accessibility show which messages are coming into and going out of the system and are modified continually as the system operates.

(8) The Statistical Data feature allows a Switching Center to collect and print out useful housekeeping data including the number of messages awaiting transmission, channel utilization and errors detected in transmission, and other statistical information regarding the performance of the system.

4. MECHANIZATION OF THE AUTOMATIC ELECTRONIC SWITCHING CENTER

ComLogNet Message Switching hardware is based upon RCA AutoData equipment. A typical Automatic Electronic Switching Center consists of 120 racks of hardware which contain 155,000 transistors, 620,000 resistors, 155,000 capacitors and 780,000 diodes. The hardware configuration for a typical 50-line Switching Center is shown in Figure 5. The equipments may be readily identified by function in the block diagram shown in Figure 6.

Three types of Buffers (teletype, low speed and high speed) are packaged in modular form and may be inserted into the Buffer racks in accordance with the requirements of the particular Switching Center. A standard Buffer Rack accommodates ten Buffers. Data storage equipment includes Drum Storage Units and Magnetic Tape Stations as well as the High-Speed Memory associated with each Communication Data Processor. A Tape Station Converter permits high speed

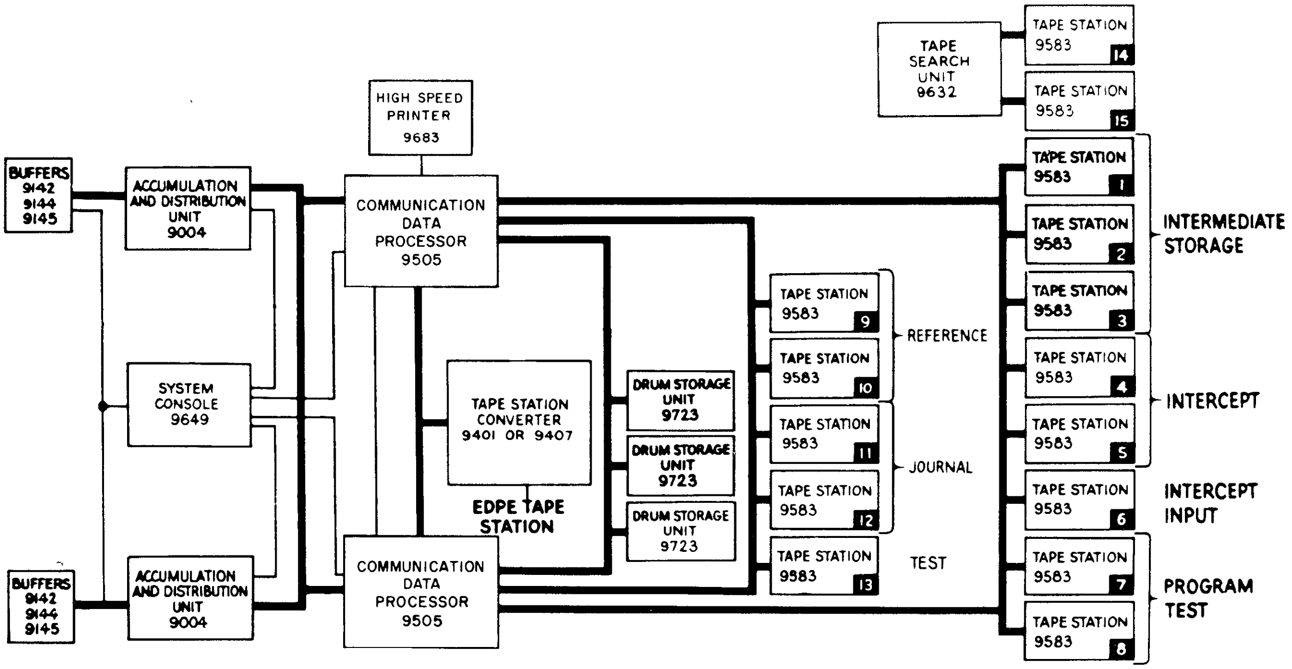


Figure 5. Equipment Block Diagram, Automatic Electronic Switching Center

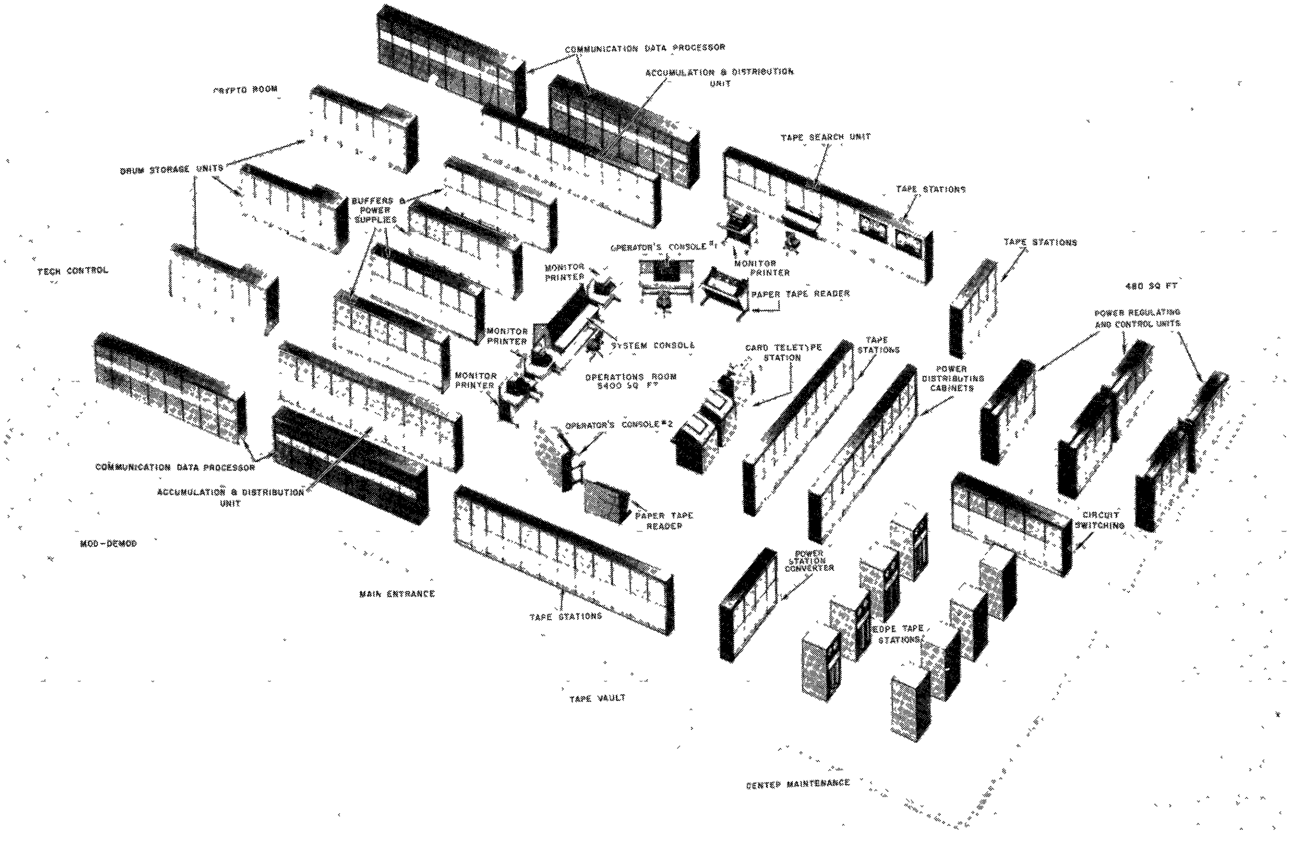


Figure 6. Typical Floor Plan, Automatic Electronic Switching Center

communication by means of magnetic tape with a local electronic data processing center. A System Console permits the operator to monitor and control operations in the Switching Center. A typical system contains two Accumulation and Distribution Units and two Communication Data Processors. The addition of Buffers and Accumulation and Distribution Units will permit a 50-channel system to be expanded to 100 channels. The four computers which are the heart of a 50-channel system are two Accumulation and Distribution Units and two Communication Data Processors. These computers are described in more detail in sections 5 and 6.

The off-line Tape Search Unit assists in the Intercept, Journal, and Reference functions and is described in section 7.

5. THE ACCUMULATION AND DISTRIBUTION UNIT

The Accumulation and Distribution Unit (ADU) is a special-purpose, digital computer utilizing both wired and partially stored program techniques to coordinate Channel operation in conjunction with the Communication Data Processor.

The ADU will accommodate up to twenty-five full-duplex (simultaneous receiving and transmitting) channels, or fifty one-way channels, or any mix of these services within the maximum stated.

a. It provides channel coordination with the tributary stations.

b. It provides storage of data on each input and each output line.

c. It provides code conversion between the transmission code and the common language code of the Communication Data Processor, and vice versa.

d. It provides for high-speed communication with the Communication Data Processor.

Two ADU's will be supplied for a 50 full-duplex channel ComLogNet Message Switching Center, and four ADU's for a 100 full-duplex channel ComLogNet Message Switching Center. Each channel may be either a high-speed (601 to 4,800 bands) channel or a low-speed (up to 601 bands) channel. Each ADU is initially equipped for transmitting and receiving on eight High-Speed Buffers and seventeen Low-Speed Buffers. The eight high-speed channels can be operated as low-speed channels by providing Low-Speed Buffers in place of the High-Speed Buffers, and by effecting change-over routines within the ADU and Technical Control. Teletype channels may be substituted in place of the high and low speed channels by means of a simple patchboard arrangement and use of Teletype Buffers.

The ADU can be expanded from an initial capability of servicing eight full-duplex, high-speed channels and 17 full-duplex, low-speed channels to a maximum of 25 full-duplex, high-speed channels by the addition of two plug-in magnetic core stacks and associated drivers. Since the ADU Data Store Memory is of modular construction, it can be expanded from two to four stacks by the addition of stacks and drivers in the space provided in the rack. No additional power supply or wiring is required for this change.

Figure 7 is a Block Diagram of the Accumulation and Distribution Unit. The ADU consists of five major sub-units as follows:

Commutation determines which channels are to be serviced. As the receiving portion of a buffer is filled, the information is transferred to an appropriate zone in the ADU Data Store Memory. When the transmitting portion of a buffer is found to be empty, a character is brought into the buffer from the ADU. The Commutator function is capable of sequentially servicing the transmit and receive section of each Buffer every 1.5 milliseconds.

The Procedure Memory (PM) is a 24,576 bit high-speed memory, which operates on a 5-microsecond regenerative read/write cycle. The PM consists of a 16 x 16 x 96 ferrite-core memory stack with associated drivers, sense amplifiers, and control logic. The PM has a capacity of 256 words of 96 bits each, and is used to store a code look-up table (code and procedure field) and the bookkeeping information required for data transfer (input-output tally field). Data is written into, or read from, the Data Memory one word at a time. The code conversion feature of the Procedure Memory is illustrated in Figure 8.

The Data Store Memory is an 8192-word, 24-bit per word, magnetic-core memory with a read-write cycle of 15 microseconds. The Data Store Memory accumulates and stores blocks of data received on the incoming channels until the CDP is ready to receive the data. Conversely, it stores blocks of data received from the CDP until the information can be transmitted on the outgoing channels. Messages are stored in the Data Store Memory in 80-character blocks as shown in Figure 9.

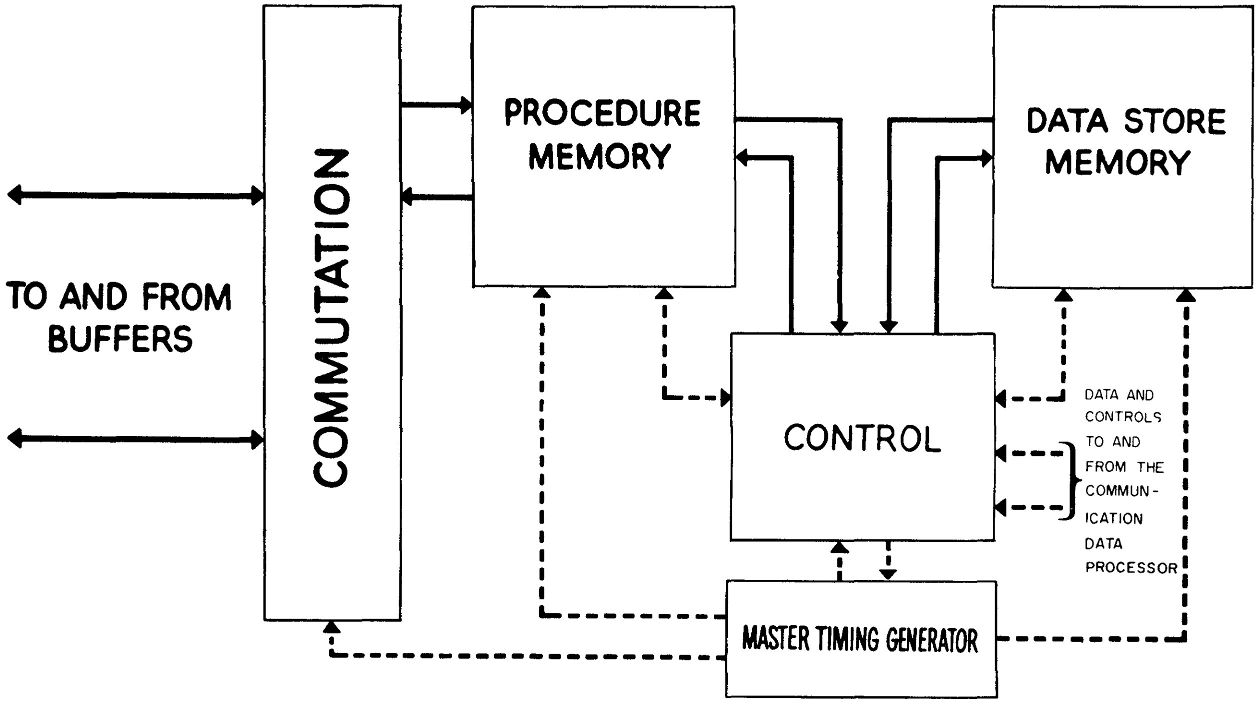


Figure 7. Block Diagram, Accumulation and Distribution Unit

MODE 1	MODE 2	CHARACTER ADDRESS								
COMMON LANGUAGE CHARACTER	TELETYPE CHARACTER	Y-ADDRESS				X-ADDRESS				
		C_2	C_1	I_2	I_1	4_1	3	2	1	P
A		0	1	0	0	0	1	1	0	0
	A	0	0	0	0	0	0	1	1	1
N	2	0	1	0	1	0	0	1	1	1
	N	0	0	0	0	1	1	0	0	1
2		0	1	1	1	0	0	0	1	1

TELETYPE CODE

COMMON LANGUAGE CODE

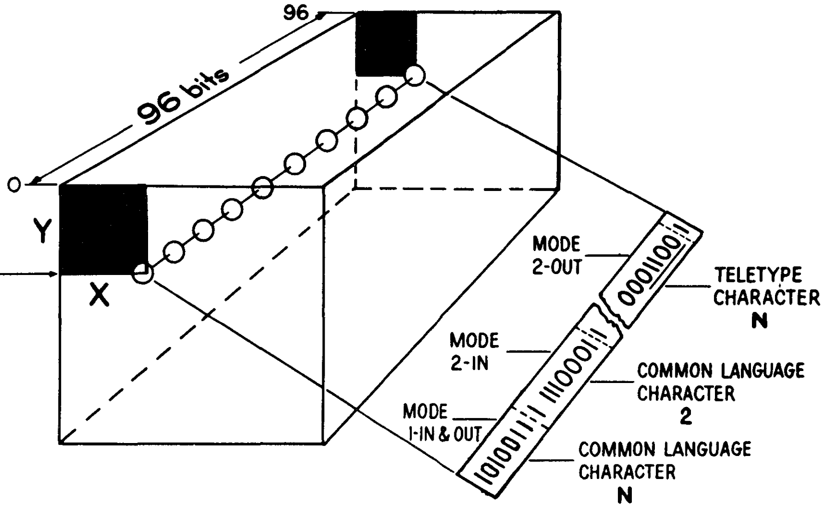


Figure 8. Code Conversion in Procedure Memory, Accumulation and Distribution Unit

The Control portion of the ADU is the decision-making unit which directs the flow of data and/or instructions to their predetermined destinations. It controls all of the tally updating and transfer of data by means of appropriate timing pulses from the Master Timing Generator.

The Master Timing Generator determines the time at which information will be transferred to and from the ADU. In conjunction with the control logic, it sets the duration and sequence of the various internal functions of the ADU. A photograph of the Accumulation and Distribution Unit is shown in Figure 10.

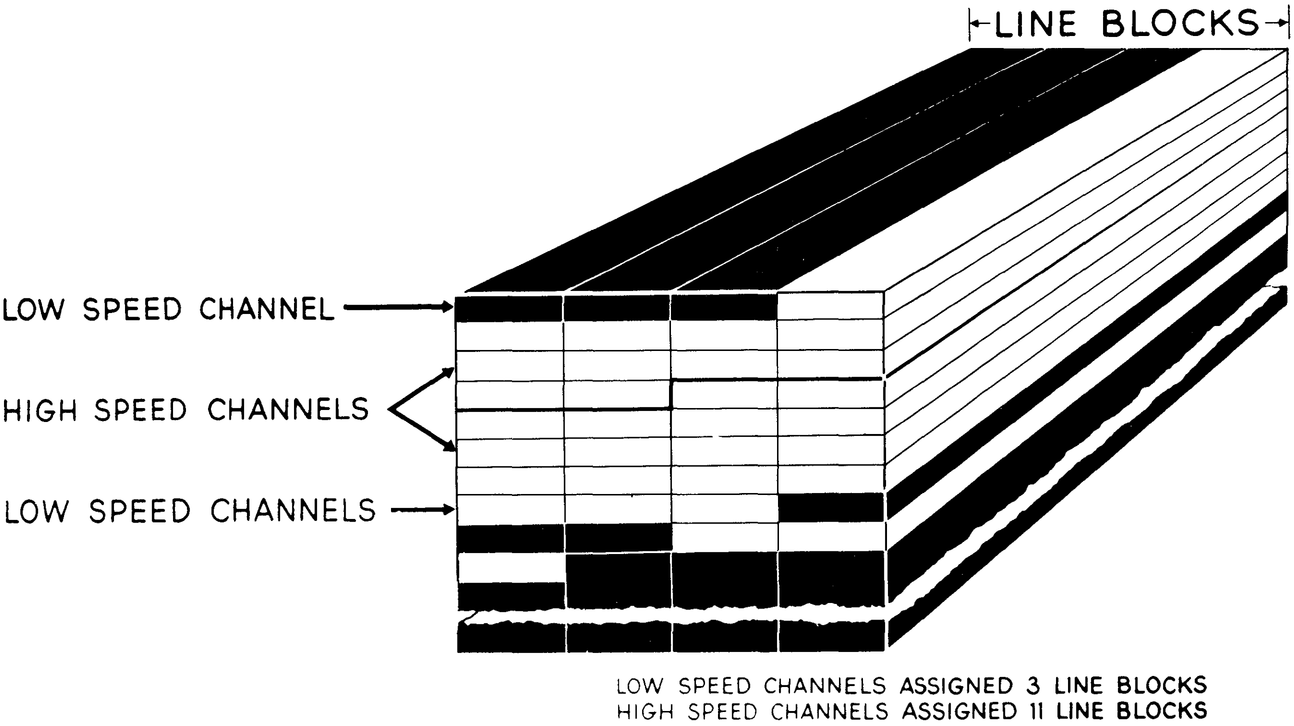

LINE BLOCKS
LOW SPEED CHANNEL
HIGH SPEED CHANNELS
LOW SPEED CHANNELS
LOW SPEED CHANNELS ASSIGNED 3 LINE BLOCKS
HIGH SPEED CHANNELS ASSIGNED 11 LINE BLOCKS

Figure 9. Data Memory, Accumulation and Distribution Unit

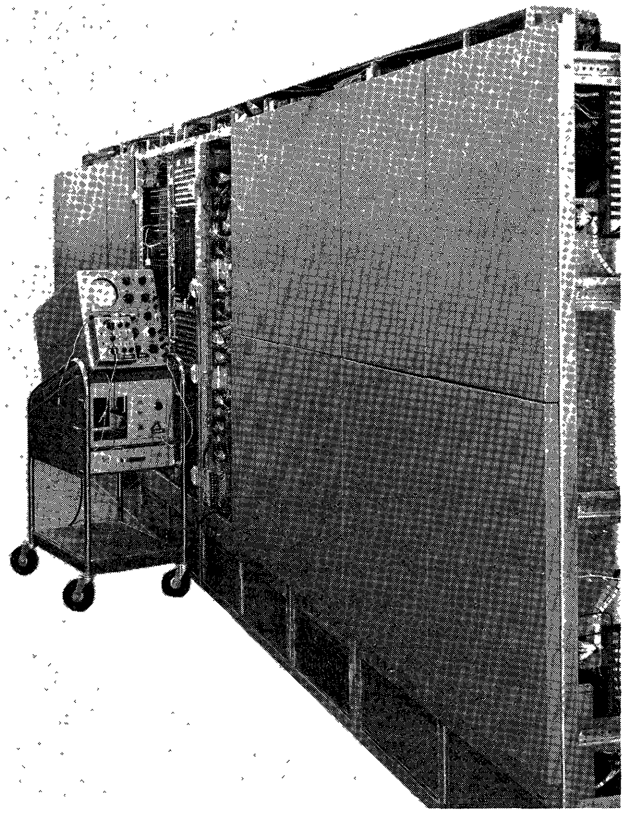


Figure 10. Photograph--Accumulation and Distribution Unit

Figure 11 shows hardware details of the ADU.

6. THE COMMUNICATION DATA PROCESSOR

The Communication Data Processor (CDP) is an all solid-state high-speed advanced design digital computer. There are two on-line Communication Data Processors in each Switching Center. One CDP handles traffic, while the other CDP is performing test and auxiliary house-keeping functions. In the event of trouble in the CDP handling traffic, control of traffic is automatically transferred to the alternate CDP without loss of data.

A block diagram of the Communication Data Processor is given in Figure 12. The CDP consists of the following:

Basic Processing Unit
High-Speed Memory (HSM)
Consoles
Input/Output Transfer Channels
Input/Output Switches

Basic Processing Unit

The Basic Processing Unit is the center for data handling and control; it performs the following functions:

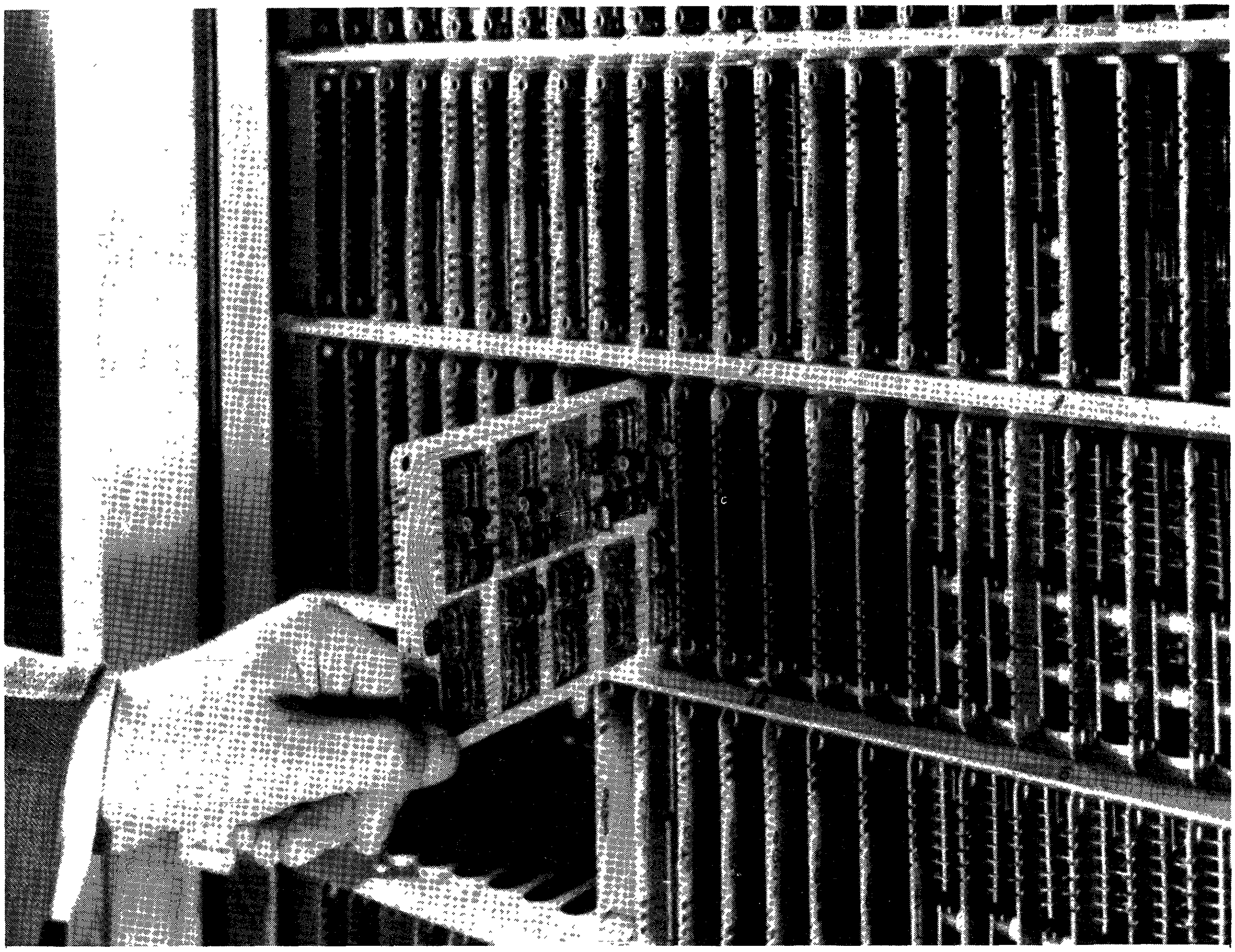


Figure 11. Hardware Details - Accumulation and Distribution Unit

Transfers data to and from the High Speed Memory.

Executes stored instructions and elementary operations.

Performs arithmetic operations.

Controls Input/Output operations.

Automatically transfers control to the alternate CDP as required.

High-Speed Memory

The High-Speed Memory (HSM) is a magnetic-core, random-access memory which is packaged in modules of 8,192 words, each word containing 56-bits. Up to four modules may be incorporated in each CDP, expanding the storage capacity to 32,768 full-words or 65,536 half-words.

The basic memory cycle can be executed in its entirety in 1.5 microseconds. Data in the form of 56 or 28 parallel bits may be read from or written in a storage location specified by a high-speed memory address. This address consists of twenty bits allocated as follows: 13 bits specify one out of the 8,192 unique word locations in a single storage unit; 2 bits specify one out of the four memory modules; 1 bit is used to specify the half word; 3 bits indicate the character location; and 1 bit is used for accuracy control functions. The High-Speed Memory operates in three modes. These are the Read-regenerate cycle, the Write cycle and the Split cycle.

Consoles

The CDP is provided with three consoles. The Operator's Console provides controls and visual displays to permit the operator to communicate directly with the Basic

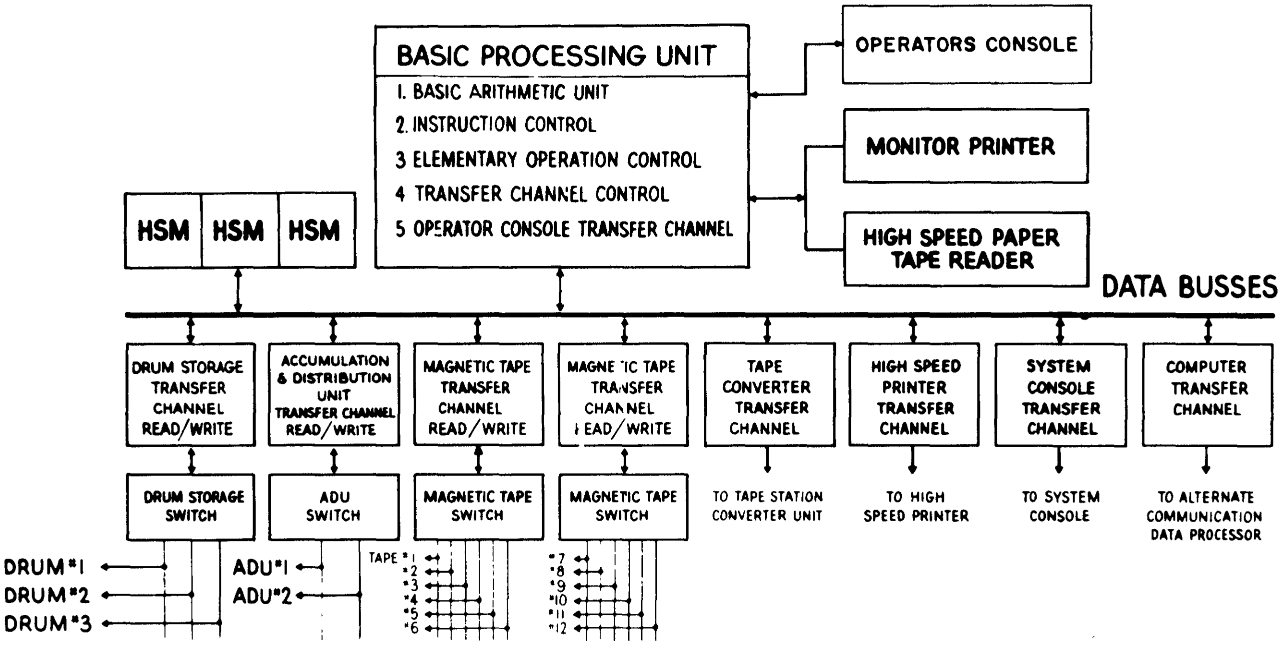

BASIC PROCESSING UNIT
1. BASIC ARITHMETIC UNIT
2. INSTRUCTION CONTROL
3 ELEMENTARY OPERATION CONTROL
4 TRANSFER CHANNEL CONTROL
5 OPERATOR CONSOLE TRANSFER CHANNEL
OPERATORS CONSOLE
MONITOR PRINTER
HIGH SPEED PAPER TAPE READER
HSM
HSM
HSM
DATA BUSSES
DRUM STORAGE TRANSFER CHANNEL READ/WRITE
ACCUMULATION & DISTRIBUTION UNIT TRANSFER CHANNEL READ/WRITE
MAGNETIC TAPE TRANSFER CHANNEL READ/WRITE
MAGNETIC TAPE TRANSFER CHANNEL READ/WRITE
TAPE CONVERTER TRANSFER CHANNEL
HIGH SPEED PRINTER TRANSFER CHANNEL
SYSTEM CONSOLE TRANSFER CHANNEL
COMPUTER TRANSFER CHANNEL
DRUM STORAGE SWITCH
ADU SWITCH
MAGNETIC TAPE SWITCH
MAGNETIC TAPE SWITCH
TO TAPE STATION CONVERTER UNIT
TO HIGH SPEED PRINTER
TO SYSTEM CONSOLE
TO ALTERNATE COMMUNICATION DATA PROCESSOR
DRUM #1
DRUM #2
DRUM #3
ADU #1
ADU #2
TAPE #1
#2
#3
#4
#5
#6
#7
#8
#9
#10
#11
#12

Figure 12. Block Diagram - Communication Data Processor

Processing Unit. The Operator's Console works with two peripheral devices: the Monitor Printer Console and the Paper Tape Reader Console. The Monitor Printer provides on-line printing of data from the CDP. The High Speed Paper Tape Reader provides on-line entry of data via punched paper tape into the system at 1000 characters per second.

Input/Output Transfer Channels

Eight types of Transfer Channels provide the means for sequencing communications between the Basic Processing Unit and peripheral devices. The data being sent to or received from a peripheral device passes through a Transfer Channel buffer or simultaneous data register. Whenever a read instruction from memory to a Transfer Channel is to be executed, the first memory location in the High-Speed Memory is specified and data words are read out sequentially. Similarly, if a write instruction is to be executed, the first memory location of the data to be stored is specified.

Transfer Channels are available for matching the speed and control requirements of the following types of input/output devices:

Drum Storage Units
Accumulation and Distribution Units
Magnetic Tape Units
Tape Station Converters
High Speed Printers
System Console
Alternate Computers

The Transfer Channels may be interconnected with more than one input/output device by means of the Transfer Channel Switches which are transistorized switch matrices. For example, a total of 48 Magnetic Tape Stations can be accommodated by two Magnetic Tape Transfer Channels and eight Switches.

A typical input/output complement for CDP includes three Drum Storage Units, two Accumulation and Distribution Units, 13 Magnetic Tape Units, and an on-line High-Speed Printer. In addition, Tape Station Converters are available which will permit communication between the ComLogNet Switching Center and local electronic data processing installations.

The functions of the CDP are illustrated in Figure 13. The CDP serves as the master in the store-and-forward service performed by ComLogNet. Its high speed permits the servicing of up to four ADU's. Categorization, accuracy checking, message protection, and maintaining records of all messages handled are additional functions of the CDP. Information brought into the CDP Memory is stored there only long enough to have pertinent data extracted from its heading; it is then forwarded to Intermediate Drum Store until it

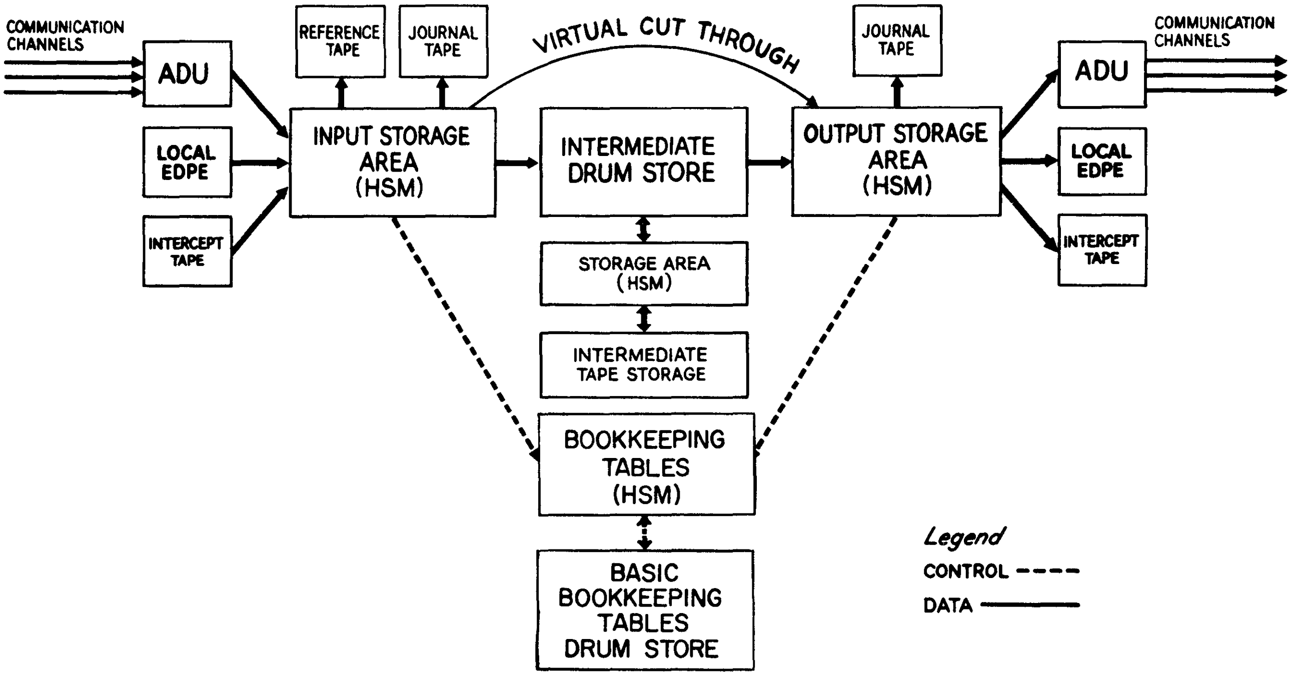


Figure 13. Functional Block Diagram - Communication Data Processor

is ready to be transmitted to its destination, or destinations. The time in intermediate storage is determined by traffic conditions and message priority. Under normal traffic conditions, all data is recorded on Storage Drums. If traffic increases and the drum nears capacity, low priority messages are transferred to Intermediate Tape Storage.

All messages entering the system must have a heading specifying priority and destination. This information with the time of arrival is queued into the Bookkeeping Tables of the HSM, and is also written on the Drum Store and on the Ledger tape. The latter two records are used for accuracy control to insure that no message is lost or remains in storage longer than its priority permits. A print out is made automatically when any message is kept in intermediate storage too long. A permanent record of all traffic handled is kept on the Journal Tape.

The CDP provides for multiple programming and multi-level simultaneity. The facility to run several programs simultaneously takes maximum advantage of the high internal speeds of the computer. A large variety of input/output functions may proceed simultaneously with each other and with computer processing. The degree of simultaneity is determined by the user through his choice of equipments and their configuration. Unlimited indirect addressing levels make it possible to transfer addresses of data rather than the data itself, thus effecting savings in processing time. A number of self-incrementing, automatic address modifiers can be applied to addresses, making interactive coding techniques rapid and efficient. The CDP handles fixed-word, fixed-field, and variable-data formats with equal ease. Data formats may thus be choosen to insure efficient utilization of memory and tapes.

Figure 14 shows a portion of the Communication Data Processor under test. Two racks at the right are High Speed Memory, four racks at the left are CDP logic. Figure 15 shows the Operator's Console.

7. The Tape Search Unit

The Tape Search Unit (TSU) is an all solid state off-line, high-speed computer. Its equipment complement could include three Magnetic Tape Stations, a Monitor Printer and a Tape Search Console. (See Figure 16) The TSU provides the ComLogNet Switching Center with a facility for automatically searching a magnetic tape for messages or selected portions of messages. The criteria for search may be specified by the operator

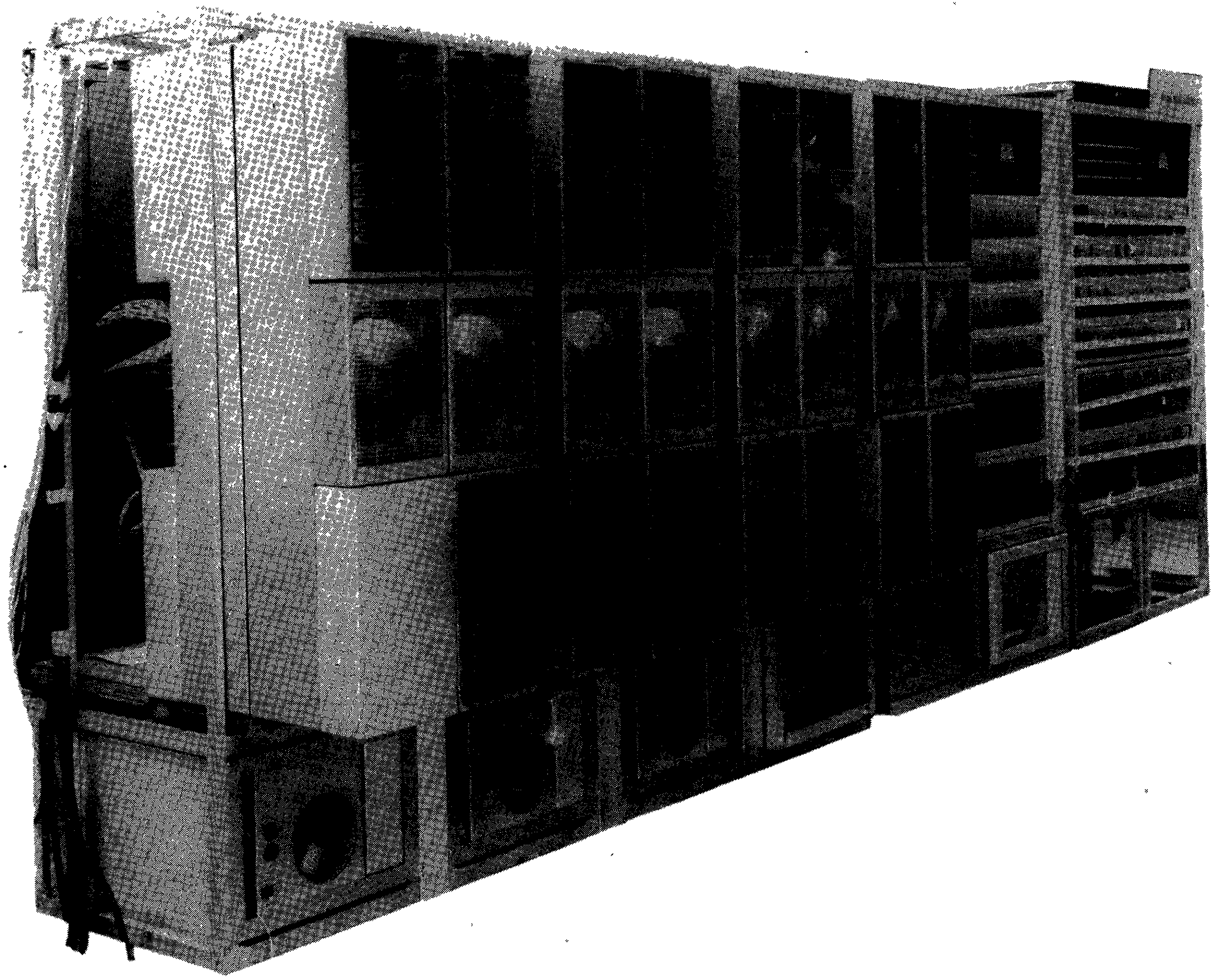


Figure 14. Photograph - Communication Data Processor, Covers Removed

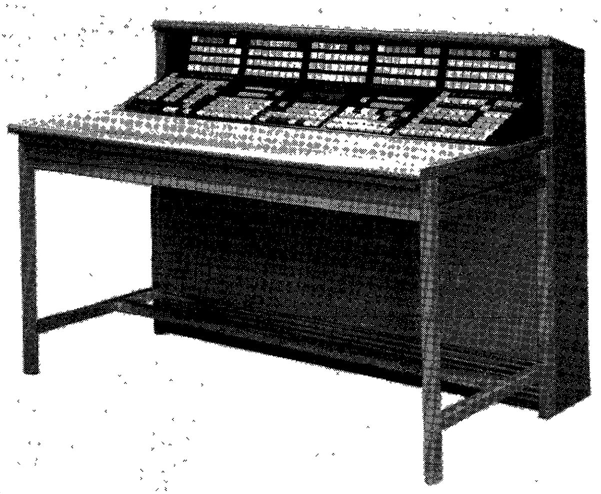


Figure 15. Photograph - CDP Console

in accordance with the mode of operation selected. The TSU extracts the desired information from the tape being examined. The output data may be recorded on another Magnetic Tape or punched paper tape. A hard copy of the output may be obtained by way of the Monitor Printer.

Information in serial pseudo-character and bit-parallel form is transmitted to, and received from, the Magnetic Tape Stations by the TSU. The TSU also receives and transmits information serially by character in bit-parallel form when communicating with the Monitor Printer. The Tape Search Unit processes data in pairs of common-language characters. It is programmed using an eighteen-bit, single-address instruction word.

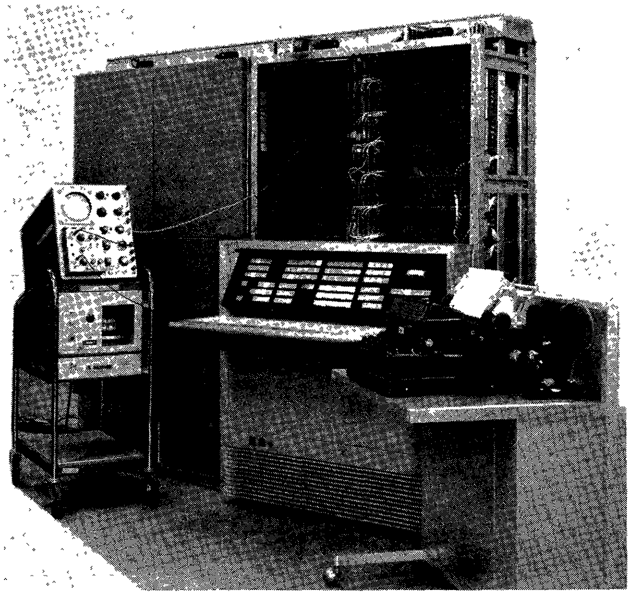


Figure 16. Photograph - Tape Search Unit

Messages are recorded onto magnetic tape in blocks, each block containing up to 1,600 seven-bit pseudo-characters. This corresponds to 1,200 common-language characters. The psuedo-characters themselves are used as an equipment convenience for data handling purposes. As data is read from magnetic tape, the pseudo-characters are automatically converted into common-language characters. Then they are stored within the TSU's memory prior to performing the required search.

Extraction of specific translated characters will take place when a sensing comparison reaches agreement with a given criterion. Comparison sensing includes the following logical operations:

Equal to
Equal to or Greater than
Equal to or Less than
Greater than
Less than

The TSU conducts a search using any one, or a combination, of the comparison techniques listed below:

Character by Character
Field of Known Address
Field of Unknown Address
Parity Error Count

The Tape Search Unit consists of six subunits as shown in Figure 17, the block diagram. These are the Input-Output logic, Memory Unit, Memory-Addressing logic, Instruction Control logic, Data Control, and the Data Transfer Bus.

CONCLUSION

The theme of this Eastern Joint Computer Conference is "Computers-Key to Total Systems Control." The ComLogNet Switching Center uses four on-line computers and one off-line computer to control store-and-forward message traffic. All are monitored from the System Console (shown in Figure 18) which displays the status of the computers and related equipment. These advanced computers incorporated in the Automatic Electronic Switching Centers are indeed the key to the ComLogNet system.

ACKNOWLEDGMENT

The able assistance of T. T. Patterson, J. A. Kalz, and others in the preparation and editing of this paper is gratefully acknowledged.

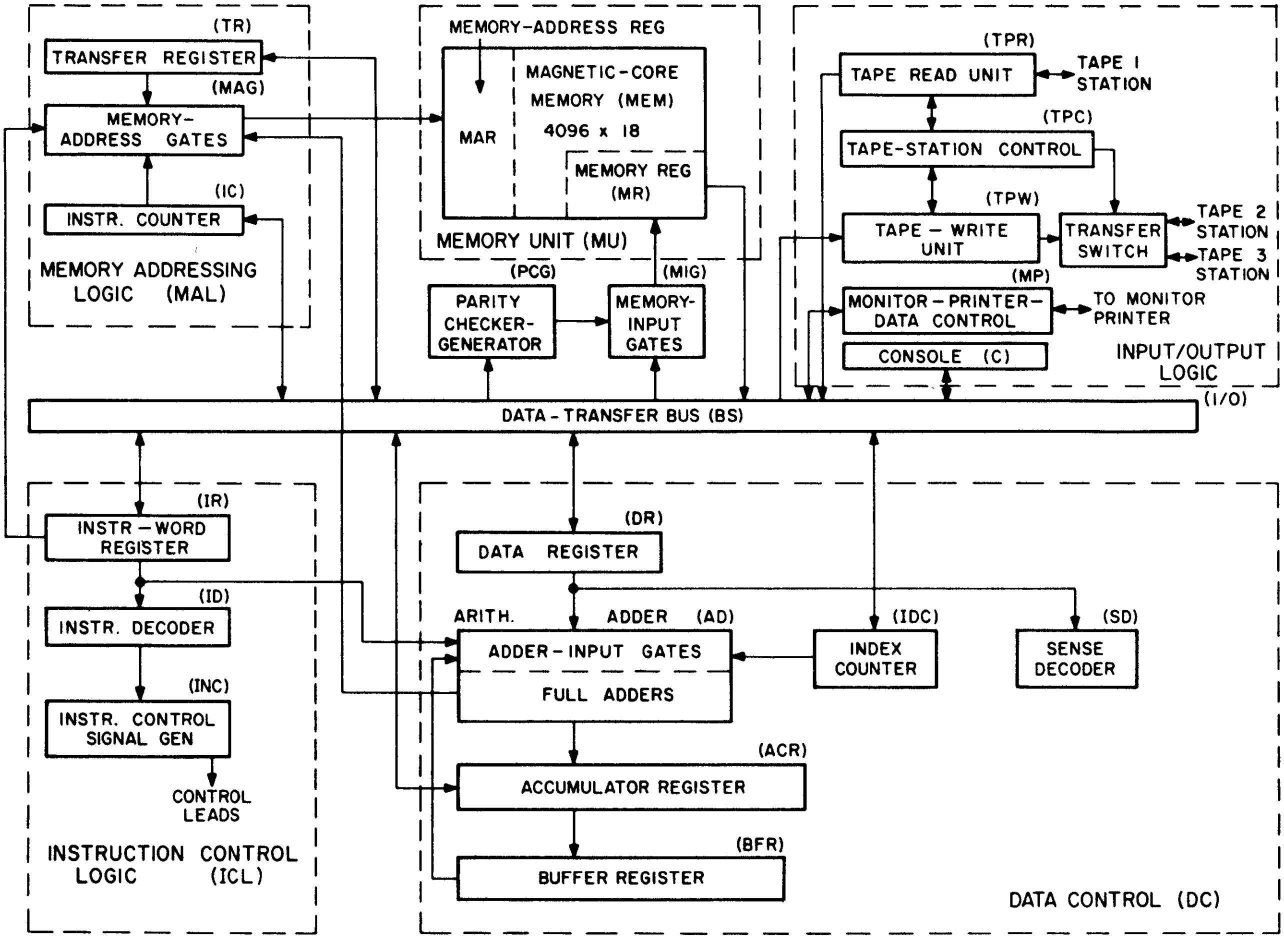


Figure 17. Tape Search Unit - Block Diagram

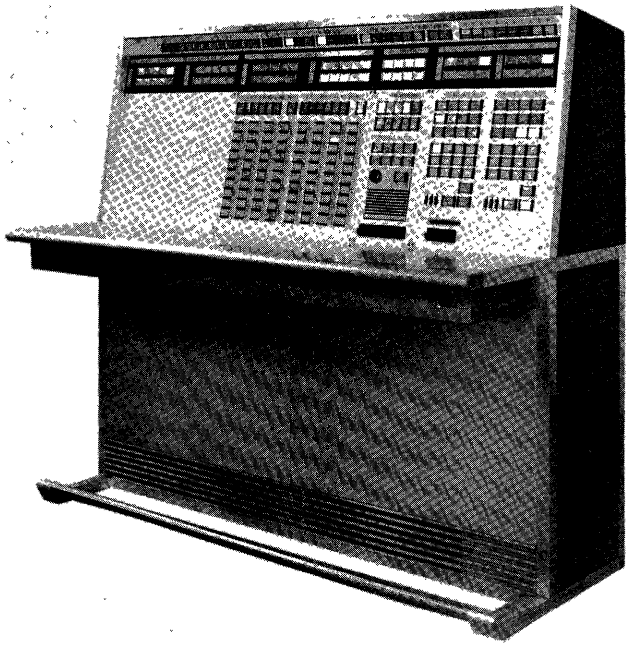


Figure 18. ComLogNet 50-Channel System Console

THE ATLAS SUPERVISOR

T. Kilburn and R. B. Payne
The University of Manchester, Manchester, England

D. J. Howarth
Ferranti Limited, London, England

1. INTRODUCTION

This paper gives a brief description of work originating in the Computer Group at Manchester University. Atlas* is the name given to a large computing system which can include a variety of peripheral equipments, and an extensive store. All the activities of the system are controlled by a program called the supervisor. Several types of store are used, and the addressing system enables a virtually unlimited amount of each to be included. The primary store consists of magnetic cores with a cycle time of under two microseconds, which is effectively reduced by multiple selection mechanisms. The core store is divided into 512 word "pages"; this is also the size of the fixed blocks on drums and magnetic tapes. The core store and drum store are addressed identically, and drum transfers are performed automatically as described in Section 3. There is a fixed store which consists of a wire mesh into which ferrite slugs are inserted; it has a fast read-out time, and is used to hold common routines including routines of the supervisor. A subsidiary core store is used as working space for the supervisor. The V-store is a collective name given to various flip-flops throughout the computer, which can be read, set, and re-set by reading from or writing to particular store addresses.

The accumulator performs floating point arithmetic on 48-bit numbers, of which 8 bits are the exponent. There are 128 index registers, or B-lines, each 24 bits long; in the instruction code, which is of the one address type, each instruction refers to two B-lines which may modify the address and the average instruction time is between one and two microseconds. There are three control registers, referred to as "main control", "extracode control", and "interrupt control", which are also B-lines 127, 126 and 125. Main control is used by object programs. When main control is active, access to the subsidiary store and V-store is prevented by hardware, and this makes it possible to ensure that object programs cannot interfere with the supervisor. The fixed store contains about 250 subroutines which can be called in from an object program by single instructions called extracodes. When these routines are being obeyed, extracode control is used: extracode control is also used by the supervisor, which requires access to the "private" stores. Interrupt control is used in short routines within the supervisor which deal with peripheral equipment. These routines are entered at times dictated by the peripheral equipments; the program using main or extracode control is interrupted, and continues when the peripheral equipment routine is completed.

The first Atlas installation at Manchester University will include:

*A paper has been written on Atlas and it is hoped will be published by the Institute of Electrical Engineers.

16,384 words of core store
8,192 words of fixed store
1,024 words of subsidiary store
98,304 words on drums
8 magnetic tape mechanisms
4 paper tape readers
4 paper tape punches
2 teleprinters
1 line printer
1 card reader
1 card punch

Other Atlas installations will include different amounts of store and peripheral equipments, but the supervisor program herein described is of sufficient generality to handle any configuration through the minor adjustment of parameters.

2. THE CO-ORDINATION OF ROUTINES

The Structure of the Supervisor

The supervisor program controls all those functions of the system that are not obtained merely by allowing the central computer to proceed with obeying an object program, or by allowing peripheral equipments to carry out their built-in operations. The supervisor therefore becomes active on frequent occasions and for a variety of reasons--in fact, whenever any part of the system requires attention from it. It becomes activated in several different ways. Firstly, it can be entered as a direct result of obeying an object program. Thus, a problem being executed calls for the supervisor whenever it requests an action that is subject to control by the supervisor, such as a request for transfer to or from peripheral equipments or the initiation of transfers between core store and magnetic drums; the supervisor is also activated when an object program requires monitoring for any reason such as exponent or division overflow, or exceeding store or time allocation. Secondly, the supervisor may be activated by various items of hardware which have completed their assigned tasks and require further attention. Thus, for example, drums and magnetic tapes call the supervisor into action whenever the transfer of a 512 word block to or from core store is completed; other peripheral equipments require attention whenever the one character or row buffer has been filled or emptied by the equipment. Lastly, certain failures of the central computer store, and peripheral equipments call the supervisor into action.

The central computer thus shares its time between these supervisor activities and the execution of object programs, and the design of Atlas and of the supervisor programs is such that there is mutual protection between object programs and all parts of the supervisor. The supervisor program consists of many branches which are normally dormant but which can be activated whenever required. The sequence in which the branches are activated is essentially random, being dictated by the course of an object program and the functioning of the peripheral equipments.

Interrupt Routines

The most frequent and rapidly activated parts of the supervisor are the interrupt routines. When a peripheral equipment requires attention, for example, an interrupt flip-flop is set which is available to the central computer as a digit in the V-store; a separate interrupt flip-flop is provided for each reason for interruption. If an interrupt flip-flop is set and interruptions are not inhibited, then before the next instruction is started, the address 2048 of the fixed store is written to the interrupt control register, B125, and control is switched to interrupt control. Further interruptions are inhibited until control reverts to main or extracode control. Under interrupt control, the fixed store program which is held at address 2048 onwards detects which interrupt flip-flop has been set and enters an appropriate interrupt routine in the fixed store. If more than one flip-flop is set, that of highest priority is dealt with first, the priority being built-in corresponding to the urgency of action required. By the use of special hardware attached to one of the B register, B123, the source of any interruption may be determined as a result of obeying between two and six instructions.

The interrupt routines so entered deal with the immediate cause of the particular interrupt. For example, when the one-character buffer associated with a paper tape reader has been filled, the appropriate interrupt flip-flop is set and the "Paper tape reader interrupt routine" is entered. This transfers the character to the required location in store after checking parity where appropriate. The paper tape reader meanwhile proceeds to read the next character to the buffer. Separate interrupt routines in the fixed store control each type of peripheral equipment, magnetic tapes and drums. The interrupt technique is also employed to deal with certain exceptional situations which occur when the central computer cannot itself deal adequately with a problem under execution, for example, when there is an overflow or when a required block is not currently available in the core store. There are therefore interrupt flip-flops and interrupt routines to deal with such cases. Further routines are provided to deal with interruptions due to detected computer faults.

During the course of an interrupt routine further interruptions are inhibited, and the interrupt flip-flops remain set in the V-store. On resumption of main or extracode control, interruptions are again permitted. If one or more interrupt flip-flops have been set in the meantime, the relevant interrupt routines are obeyed in the sequence determined by their relative priority. In order to avoid interference with object programs or supervisory programs, interrupt routines use only restricted parts of the central computer, namely, the interrupt control register, B-lines 123 and 111 to 118 inclusive, private registers in subsidiary store and the V-store and locked out pages in core store (see Section 3). With the exception of the B-lines, no object program is permitted to use these registers. No lock out is imposed on the B-lines, but interrupt routines make no assumptions concerning the original contents of the B-lines and hence, at worst, erroneous use of interrupt B-lines by an object program can only result in erroneous functioning of that particular program. Switching of control to and from an interrupt routine is rapid, since no preservation of resetting of working registers is required.

The interrupt routines are designed to handle calls for action with the minimum delay and in the shortest time; the character-by-character transfers to and from peripheral equipments, for example, occur at high frequency and it is essential that the transfers be carried out with the minimum possible use of the central computer and within the time limit allowed by the peripheral equipment for filling or emptying the buffer. Since several interrupt flip-flops can become set simultaneously, but cannot be acted upon while another interrupt routine is still in progress, it is essential that a short time limit be observed by each interrupt routine. The majority of calls for interrupt routines involve only a few instructions, such as the transfer of a character, stepping of counts, etc., and on conclusion the interrupt routine returns to the former control, either main or extracode. On some occasions, however, longer sequences are required; for example, on completion of the input of a paper tape or deck of cards, routines must be entered to deal with the characters collected in the store, writing them to magnetic tape where appropriate, decoding and listing titles and so on. In such cases, the interrupt routine initiates a routine to be obeyed under extracode control, known as a supervisor extracode routine.

Supervisor Extracode Routines

Supervisor extracode routines (S.E.R.'s) form the principal "branches" of the supervisor program. They are activated either by interrupt routines or by extracode instructions occurring in an object program. They are protected from interference by object programs by using subsidiary store as working space, together with areas of core and drum store which are locked out in the usual way whilst an object program is being executed (see Section 3). They operate under extracode control, the extracode control register of any current object program being preserved and subsequently restored. Like the interrupt routines, they use private B-lines, in this case B-lines 100 to 110 inclusive; if any other working registers are required, the supervisory routines themselves preserve and subsequently restore the contents of such registers. The S.E.R.'s thus apply mutual protection between themselves and an object program.

These branches of the supervisor program may be activated at random intervals. They can moreover be interrupted by interrupt routines, which may in turn initiate other S.E.R.'s. It is thus possible for several S.E.R.'s to be activated at the same time, in the same way as it is possible for several interrupt flip-flops to be set at the same time. Although several S.E.R.'s may be activated, obviously not more than one can be obeyed at any one moment; the rest are either halted or held awaiting execution. This matter is organized by a part of the supervisor called the "co-ordinator routine" which is held in fixed store. Activation of an S.E.R. always occurs via the co-ordinator routine, which arranges that any S.E.R. in progress is not interrupted by other S.E.R.'s. As these are activated, they are recorded in subsidary store in lists and an entry is extracted from one of these lists whenever an S.E.R. ends or halts itself. Once started, an S.E.R. is always allowed to continue if it can; a high priority S.E.R. does not "interrupt" a low priority S.E.R. but is entered only on conclusion or halting of the current S.E.R. The co-ordinator has the role of the program equivalent of the "inhibit interrupt flip-flop", the lists of activated S.E.R.'s being the equivalent of the setting of several interrupt flip-flops. The two major differences are that no time limit is placed on an S.E.R., and that an S.E.R. may halt itself for various reasons; this is in contrast to interrupt routines, which observe a time limit and are never halted.

In order that the activity of each branch of the computing system be maintained at the highest possible level, the S.E.R.'s awaiting execution are recorded in four distinct lists. Within each list, the routines are obeyed in the order in which they were activated, but the lists are assigned priorities, so that the top priority list is emptied before entries are extracted from the next list. The top priority list holds routines initiated by completion of drum transfers, and also routines entered as a result of computer failures such as core store parity. The second list holds routines arising from magnetic tape interruptions and the third holds routines arising from peripheral interruptions. The lowest priority list contains one entry for each object program currently under execution, and entry to an S.E.R. through an extracode instruction in an object program is recorded in this list. On completion of an S.E.R., the co-ordinator routine selects for execution the first activated S.E.R. in the highest priority list.

The central computer is not necessarily fully occupied during the course of an S.E.R. The routine may, for example, require the transfer of a block of information from the drum to the core store, in which case it is halted until the drum transfer is completed. Furthermore, the queue of requests for drum transfers (see section 3), is maintained in the subsidiary store, may be full, in which case the S.E.R. making the request must be halted. When an S.E.R. is halted for this or similar reasons, it is returned to the relevant list as halted, and the next activated S.E.R. is entered by the co-ordinator routine. Before an S.E.R. is halted, a restart point is specified. A halted routine is made free to proceed when the cause of the halt has been removed—for example, by the S.E.R. which controls drum transfers and the extraction of entries from the crum queue. The S.E.R. lists can therefore hold at any one time routines awaiting execution and halted routines; interrupt routines are written in such a way that the number of such S.E.R.'s activated at any one time is limited to one per object program, and one or two per interrupt flip-flop, depending upon the particular features of each interrupt routine. When an S.E.R. is finally concluded, as distinct from halted, it is removed from the S.E.R. lists and becomes dormant again.

Although S.E.R.'s originate in many cases as routines to control peripheral equipment, magnetic tapes and drums, it should not be supposed that this is the sole function of these routines. Entrances to S.E.R.'s from interrupt routines or from extracode instructions in an object program initiate routines which control the entire operation of the computing system, including the transfer of information between store and peripherals, communication with the operators and engineers, the initiation, termination and, where necessary, monitoring of object programs, the monitoring of central computer and peripheral failures, the execution of test programs and the accumulation of logging information. Each branch of supervisory activity is composed of a series of S.E.R.'s, each one activated by an object program or an interrupt routine and terminated usually by initiating a peripheral or magnetic tape transfer or by changing the status of an S.E.R. list or object program list. The most frequently used routines are held in the fixed store; routines required less frequently are held on the magnetic drum and are transferred to core store when required. Supervisor routines in core and drum store are protected from interference by object programs by use of hardware lock-out and the basic store organization routines in the fixed store.

Object Programs

The function of all supervisor activity is, of course, to organize the progress of problems through the computer with the minimum possible delay. Object programs are initiated by S.E.R.'s, which insert them into the object program list; they are subsequently entered by the co-ordinator routine effectively as branches of lower priority than any S.E.R. Although object programs are logically subprograms of the supervisor, they may function for long periods using the computer facilities to the full without reference to the supervisor. For this reason, the supervisor program may be regarded as normally dormant, activated and using the central computer for only a small proportion of the available time.

In order to allow object programs to function with the minimum of program supervision, they are not permitted to use extracode control or interrupt control directly, enabling protection of main programs and supervisor programs to be enforced by hardware. Object programs use the main control register, B127, and are therefore forbidden access to the V-store and subsidiary store. Reference to either of these stores causes the setting of an interrupt flip-flop and hence entrance to the supervisor program.

Access to private stores is only obtained indirectly by use of extracode functions, which switch the program to extracode control and enter one of a possible maximum of 512 routines in the fixed store. These extracode routines form simple extensions of the basic order code, and also provide specific entry to supervisor routines to control the transfer of information to and from the core store and to carry out necessary organization. Such specific entrances to the supervisor program maintain complete protection of the object programs. Protection of magnetic tapes and peripheral input and output data is obtained by the use, in extracode functions, of logical tape and data numbers which the supervisor identifies within each program with the titles of the tapes or information. Blocks of core and drum store are protected by hardware and by the supervisor routines in fixed store as described in Section 3.

An object program is halted (by S.E.R.'s) whenever access is required to a block of information not immediately available in the core store. The block may be on the drums, in which case a drum transfer routine is entered, or it may be involved in a magnetic tape transfer. In both cases the program is halted until the block becomes available in core store. In the case of information involved in peripheral transfers, such as input data or output results, the supervisor buffers the information in core and drum store, and "direct" control of a peripheral equipment by an object program is not allowed. In this way, immobilization of large sections of store whilst a program awaits a peripheral transfer can be avoided. A program may however call directly for transfers involving drums or magnetic tapes by use of extracode functions, which cause entrance to the relevant supervisor routines. Queues of instructions are held in subsidiary store by these routines, in order to allow the object program to continue and to achieve the fullest possible overlap between tape and drum transfers and the execution of an object program.

While one program is halted, awaiting completion of a magnetic tape transfer for instance, the co-ordinator routine switches control to the next program in the object program list which is free to proceed. In order to maintain full protection, it is necessary to preserve and recover the contents of working registers common to all programs such as the B-lines, accumulator, and control registers, and to protect blocks in use in core store. The S.E.R. to perform this switching from one object program to another occupies the central computer for around 750 + 12p μsecs, where p is the number of pages, or 512 word blocks in core store. On the Manchester University Atlas, which has 32 pages of core store, the computing time for the round trip to switch from one program to another and to return subsequently is around 2.5 m.secs. This is in contrast to the time of around 60 μsecs. to enter and return from an S.E.R. and even less to switch to and from an interrupt routine. It is therefore obvious that the most efficient method of obtaining the maximum overlap between input and output, magnetic tape transfers, and computing is to reduce to a minimum the number of changes between object programs and to utilize to the full the rapid switching to and from interrupt and supervisor routines. The method of achieving this in practice is described in Section 6.

Compilation of programs is treated by the supervisor as a special case of the execution of an object program, the compiler comprising an object program which treats the source language program as input data. Special facilities are allowed to compilers in order that their allocation of storage space may be increased as need arises, and to allow exit to the supervisor before the execution of a problem or the recording of a compiled object program.

Error Conditions

In addition to programmed entrances to the supervisor, entrance may also be made in the event of certain detectable errors arising during the course of execution of a problem. A variety of program faults may occur and be detected by hardware, by programmed checks in extracodes, and in the supervisor. Hardware causes entry to the supervisor by the setting of interrupt flip-flops in the event of overflow of the accumulator, use of an unassigned instruction, and reference to the subsidiary store or V-store. Extracode routines detect errors in the range of the argument in square root, logarithm, and arcsin instructions. In the extracodes referring to peripheral equipment or magnetic tapes, a check is included that the logical number of the equipment has been previously defined. In extracodes for data translation, errors in the data may be detected. The supervisor detects errors in connection with the use of the store. All problems must supply information to the supervisor on the amount of store required, the amount of output, and the expected duration of execution. This information is supplied before the program is compiled, or may be deduced after compilation. The supervisor maintains a record of store blocks used, and can prevent the program exceeding the preset limit. In addition, an interrupt flip-flop is set by a clock at intervals of 0.1 secs, and another flip-flop is set whenever 1024 instructions have been obeyed using main or extracode control. These cause entrances to the supervisor which enable a program to be "monitored" to ensure that the preset time limit has not expired, and which are also instrumental in initiating routines to carry out regular timed operations such as logging of computer performance and initiation of routine test programs.

The action taken by the supervisor when a program "error" is detected depends upon the conditions previously set up by the program. Certain errors may be individually trapped, causing return of control to a preset address; a private monitor sequence may be entered if required enabling a program or a compiler to obtain diagnostic printing; failing specification of these actions, some information is printed by the supervisor and the program is suspended, and usually dumped to magnetic tape to allow storage space for another program.

The following sections describe in detail the action of certain supervisor routines, namely, those controlling drums, magnetic tapes, and peripheral equipment and those controlling the flow of information in the computer.

3. STORE ORGANIZATION

Indirect addressing and the One-Level Store

The core store of Atlas is provided with a form of indirect addressing which enables the supervisor to re-allocate areas of store and to alter their physical addresses, and which is also used to implement automatic drum transfers. With each page, or 512 word block, of core store there is associated a "page address register" which contains the most significant address bits of the block of information contained in the page. Every time access is required to a word of information in the core store, the page containing the word is located by hardware. This tests for equivalence between the requested "block address", or most significant address bits, and the contents of each of the page address registers in parallel. Failure to find equivalence results in a "non-equivalence" interruption. The page address registers are themselves addressable in the V-store and can thus be set appropriately by the supervisor whenever information is transferred to or from core store.

One of the most important consequences of this arrangement is that it enables the supervisor to implement automatic drum transfers. The address in an instruction refers to the combined core and drum store of the computer, and the supervisor records in subsidiary store the location of each block of information; only one copy of each block is kept, and the location is either a page of core store or a sector of the drum store. At any moment, only some of the blocks comprising a particular program may be in the core store and if only these blocks are required, the program can run at full speed. When a block is called for which is not in the core store, a non-equivalence interruption occurs, which enters the supervisor to transfer the new block from a sector of the drum to a page of the core store. During this operation the program that was interrupted is halted by the supervisor.

The block directory in subsidiary store contains one entry for each block in the combined core and drum store. It is divided into areas for each object program which is in the store; a separate program directory defines the area of the block directory occupied by each program. The size of this area, or the number of blocks used by a program, is specified before the program is obeyed in the job description (see Section 6). The entry for block n contains the block number n together with the number of the page or sector occupied by the block, and, if possible, is made in the n^th^ position in the area; otherwise the area is filled working backwards from the end. In this way, blocks used by different object program are always kept distinct, regardless of the addresses that are used in each program. A program addresses the combined "one-level store* and the supervisor transfers blocks of information between the core and drum store as required; the physical location of each block of information is not specified by the program, but is controlled by the supervisor.

There are occasions when an object program must be prevented from obtaining access to a page of the core store, such as one involved in a drum or tape transfer. To ensure complete protection of such pages, an additional bit, known as a lock out bit, is provided with each page address register. This prevents access to that page by the central computer, except when on interrupt control, and any reference to the page causes a non-equivalence interruption. By setting and resetting the lock out bits, the supervisor has complete control over the use of core store; it can allow independent object programs to share the core store, it can reserve pages for peripheral transfers and can itself use parts of the core store occasionally for routines or working space, without any risk of interference. This is done by arranging that, whenever control is returned to an object program, pages that are not available to it are locked out.

A block of information forming part of an object program may also be locked out from use by that program because an operation on that information, controlled by the supervisor, is not complete. A drum, magnetic tape, or peripheral equipment transfer involving this block may have been requested. The reason for the lock out of such a block is recorded in the block directory, and if the block is in the core store, the lock out digit is also set. If reference is made to such a block by the object program, a non-equivalence interruption occurs and a supervisor extracode routine halts the program. This S.E.R. is restarted by the co-ordinator routine when the block becomes "unlocked", and the object program is re-entered when the block is available in core store.

The Drum Transfer Routine

The drum transfer routine is a group of S.E.R.s which are concerned with organizing drum transfers, and updating page address registers and the block directory. Once initiated, the transfer of a complete block to or from the drum proceeds under hardware control; the drum transfer routine initiates the transfer and identifies the required drum sector by setting appropriate bits in the V-store. It also identifies the core store page involved by setting a particular "dummy" block address, recognized by the drum control hardware, in the page address register; at the same time, this page is locked out to prevent interference from object programs while the transfer is in progress.

On completion of a transfer, an interruption occurs which enters the drum transfer routine. The routine can also be entered from the non-equivalence interrupt routine, which detects the number of the block requested but not found in the page address registers. Finally, the drum transfer routine can be activated by other parts of the supervisor which require drum transfers, and by extracode instructions which provide a means whereby object programs can if they wish exert some control over the movement of blocks to and from the drum store. A queue of requests for drum transfers, which can hold up to 64 requests, is stored in the subsidiary store; when the drum transfer routine is entered on completion of a transfer, the next transfer in the queue is initiated.

*A paper on the one-level store has been written, and will, it is hoped, be published by the Institute of Radio Engineers.

Whenever the supervisor wishes to enter another request for a drum transfer, three possible situations arise. Firstly, the queue is empty and the drum transfer can be started immediately. Secondly, the queue is already partly filled and the request is entered in the next position in the queue. Thirdly, the queue is full. In this case the routine making the request is halted by the co-ordinator routine, and is resumed when the queue can receive another entry. In the first two cases the supervisor routine is concluded when the request reaches the queue.

A non-equivalence interruption, which implies a drum transfer is required, is dealt with as follows. The core store is arranged to always hold an empty page with no useful information in it, and when required, a transfer of a block of information from the drum to this empty page is initiated. While this drum transfer is proceeding, preparation is made to write up the contents of another page of core store to the drum to maintain an empty page. The choice of this page is the task of the "learning program" which keeps details of the use made of blocks of information. This learning program will be described in detail elsewhere; it predicts the page which will not be required for the largest time, and is arranged with a feed-back so that if it writes up a block which is almost immediately required again, it only does this once. The number of the chosen page

the drum queue entry is converted to a request to write this page to the drum. This supervisor routine is now concluded and returns control to the co-ordinator routine.

When the drum transfer is completed, the drum transfer routine is again entered. This updates the block directory and page address register, makes the object program free to proceed and initiates the next drum request, which is to write the chosen page to the drum. This routine is now concluded and the co-ordinator is re-entered. The supervisor is finally entered when the write to drum transfer is complete. The block directory is updated, a note is made of the empty page, and the next drum request is initiated.

The Use of Main Store by the Supervisor

Some routines of the supervisor are obeyed in the main store, and these and others use working space in the main store. Since the supervisor is entered without a complete program change, special care must be taken to keep these blocks of store distinct and protected from interference. The active supervisor blocks of main store are recorded in the area for program 0 in the block directory. There are also some blocks of the supervisor program which are stored permanently on the drum; when one of these permanent blocks is required, it is duplicated to form an active block of the supervisor or, as in the case of a compiler, to become part of an object program.

Of the possible 2048 block numbers, 256 are "reserved" block numbers which are used exclusively by the supervisor and are not available to object program; object program are restricted to using the remaining "non-reserved" block numbers. Blocks with reserved block numbers may be used in the core store at any time by the supervisor, and the co-ordinator routine locks out these pages of core store before returning control to an object program. The supervisor also uses some blocks having non-reserved block numbers to keep a record of sequence of blocks of information such as input and output streams. When a non-reserved supervisor block is called to the core store, the page address register is not set, since there may be a block of an object program which has the same block number already in the core store. Instead, the page address register is set to a fixed reserved block number while it is in use, and is cleared and locked out before control passes to another routine.

Not all the reserved block numbers are available to the supervisor for general use, since certain block numbers are temporarily used when drum, tape, and peripheral transfers are proceeding. These block numbers do not appear in the block directory. For example, when a magnetic tape transfer is taking place, the page of core store is temporarily given a block number which is recognized by the hardware associated with that tape channel. When the transfer is complete, the appropriate block number is restored. During a peripheral transfer, and also on other occasions, it is necessary that a block should be retained in the core store and should not be transferred to the drum. The relevant page of core store is "locked down" by setting a digit in the subsidiary store; the learning program never selects for transfer to the drum a page for which this lockdown digit is set.

4. MAGNETIC TAPE SUPERVISOR ROUTINES

The Magnetic Tape Facilities

The tape mechanism used on Atlas is the Ampex TM2 (improved FR 300) using one inch wide magnetic tape. There are sixteen tracks across the tape - twelve information tracks, two clock tracks, and two tracks used for reference purposes. The tapes are used in a fixed-block, pre-addressed mode. Information is stored on tape in blocks of 512 forty-eight bit words, together with a twenty-four bit checksum with end around carry. Each block is preceded by a block address and block market and terminated by a block marker; the leading block address is sequential along the tape, and what is effectively the trailing block address is always zero. Tapes are tested and pre-addressed by special routines before being put into use, and the fixed position of the addresses permits selective overwriting and simple omission of faulty patches on the tape. Blocks can be read when the tape is moving either in the forward or reverse direction, but writing is only possible when the tape is moving forward. The double read and write head is used to check read when writing on the tape. When not operating the tape stops with the read head midway between blocks.

Atlas may control a maximum of 32 magnetic tape mechanisms. Each mechanism is connected to the central computer via one of eight channels, all of which can operate simultaneously, each controlling one read, write or positioning operation. It is possible for each tape mechanism to be attached to either one of a pair of channels, the switching being under the control of supervisory program through digits in the V-store. Fast wind and rewind operations are autonomous and only need the channel to initiate and, if required, terminate them. Transfer of a 512-word block of information between core store and tape is effected via a one-word buffer, the central computer hesitating for about 1/2 μsec, on average, each time a word is transferred to or from the core store. During a transfer the page of core store is given a particular reserved block number and the contents of the page address register are restored at the end of the transfer.

Supervisory programs are only entered when the block addresses are read before and after each block, and when the tape stops. As each block address is read, it is recorded in the V-store and an interrupt flip-flop is set, causing entrance to the block address interrupt routine.

The Block Address Interrupt Routine

This routine is responsible for initiating and checking the transfer of a single block between tape and core store, and searching along the tape for a specified block address. Digits are available in the V-store to control the speed and direction of motion of the tape and the starting and termination of read or write transfers. The block addresses are checked throughout and, in particular, a write transfer is not started until the leading block address of the tape block involved has been read and checked. Hardware checking is provided on all transfers, and is acted upon by supervisor routines. A 24-bit check sum is formed and checked as each block is transferred to or from a tape, and a digit is set in the V-store if any failure is detected. Similarly a digit is set in the event of failure to transfer a full block of 512 words. These digits are tested by the block address interrupt routine on the conclusion of each transfer. Parity failure either on reading from core store or on formation of the parity during a transfer to core store causes the setting of interrupt flip-flops. If a tape fails to stop, this is detected by the block address interrupt routine as a particular case of block address failure. Failure to enter the block address routine (for example, through failure to read block markers) is detected by the timed interrupt routine at intervals of 100 milliseconds. Finally, failures of the tape mechanism, such as vacuum failure, set a separate interrupt flip-flop. The detection of any of these errors causes entry to tape monitor routines, whose action will be described later.

Organization of Tape Operations

Magnetic tape operations are initiated by entrance to the tape supervisor routines in the fixed store from extracode instructions in an object program or, if the supervisor requires the tape operation for its own purposes, from supervisor extracode routines. From a table in subsidiary store, the logical tape number used in a program is converted to the actual mechanism number, and the tape "order" is entered to a queue of such orders, in subsidiary store, awaiting execution. A tape order may consist of the transfer of several blocks and any store blocks involved are "locked out" to prevent subsequent use before completion of the transfers; if any block is already involved in a transfer, the program initiating the request is halted. Similarly, the program is halted if the queue of tape instructions is already full. If the channels to which the deck can be connected are already occupied in a transfer or positioning, the tape supervisor returns control to the object program, which is then free to proceed. A program may thus request a number of tape transfers without being halted, allowing virtually the maximum possible overlap between the central computer and the tape mechanisms during execution of a program. Should a channel be available at the time a tape order is entered to the queue, the order is initiated at once by writing appropriate digits to the V-store, and by writing reserved tape transfer block numbers to the appropriate page address registers if the order involves a read or write transfer. The tape supervisor then returns control to the object program or supervisor routine.

One composite queue of tape orders is used for orders relating to all tape mechanisms and orders are extracted from the queue by S.E.R.'s entered from the block address interrupt routine. On reading the penultimate block address involved in an operation (for example, the last leading block address in a forward transfer) the next operation for the channel is located, and if it involves the same mechanism as the current order, and tape motion in the same direction, the operation is "prepared" by calling any store block involved to core store. On reading the final block address and successfully concluding checks, the block address interrupt routine initiates the next operation immediately if one has been prepared, thus avoiding stopping the tape if possible. If no operation has been prepared, the interrupt routine stops the tape by setting a digit in the V-store, and a further "block address interruption" occurs when the tape is stopped and the channel can accept further orders. This interruption enters an S.E.R. which extracts the next order for the channel from the tape queue, and the cycle of events is repeated until no further order for this channel remains. As each transfer is concluded, any object program halted through reference to the store block is made free to proceed.

An exception to the above process is when a long movement (over 200 blocks) or a rewind is required. In this case, the movement is carried out at fast speed, with block address interruptions inhibited, and the channel may meanwhile be used to control another tape mechanism. The long movement is terminated by checking the elapsed time and at the appropriate moment, entering the tape supervisor from the timed interrupt routine. The mechanism is then brought back "on channel" and the speed is returned to normal. When reading of block addresses is correctly resumed, the search is continued in the normal manner.

The Title Block

The first block on each magnetic tape is reserved for use by the supervisor, and access to information in this block by an object program is through special instructions only. This block contains the title of the tape, or an indication that the tape is free. When magnetic tapes are required by the supervisor or by an object program, the supervisor prints instructions to the operator to load the named tape and to engage the mechanism on which it is loaded. The engage button of each mechanism (see section 5) is attached to a digit in the V-store, and these digits are scanned by the supervisor every one second. When a change to "engaged" status has been detected, the tape supervisor is entered to read the first block from the tape. The title is then checked against the expected title. In this way, the presence of the correct tape is verified, and furthermore the tape bearing the title becomes associated with a particular mechanism. Since the programmer assigns a logical tape number to the tape bearing a given title, this logical tape number used in extracode instructions can be converted by the supervisor to the actual mechanism number. Other supervisory information is included in the first block on each tape, including a system tape number and the number of blocks on the tape. Special supervisory routines allow Atlas to read tapes produced on the Ferranti Orion computer, which used the same tape mechanisms but can write blocks of varying lengths on the tape. These tapes are distinguished on Atlas by a marker written in the title block.

Magnetic Tape Failures

All failures detected by the interrupt routines cause the block address interrupt routine to stop the tape at the end of the current block when possible, and then to enter tape monitor supervisory routines; if the tape cannot be stopped, it is disengaged and the tape monitor routines entered. These routines are S.E.R.'s designed to minimize the immediate effect on the central computer of isolated errors in the tape system, to inform maintenance engineers of any faults, and to diagnose as far as possible the source of a failure. As an example of the actions taken by monitor routines, suppose a check sum failure has been detected while reading a block from tape to core store. The tape monitor routines make up to two further attempts to read the block; if either succeeds, the normal tape supervisor is re-entered after informing the engineers. Repeated failure may be caused by the tape or the tape mechanism; to distinguish these, the tape is rewound and an attempt is made to read the first block. If this is successful, a tape error is indicated, and an attempt is made to read the suspect block with reduced bias level. Failure causes the mechanism to be disengaged and the program using the tape to be suspended. If the "recover read" is successful, the tape is copied to a free tape and the operators instructed to re-address the faulty tape, omitting the particular block which failed. If on rewinding the tape, the first block cannot be read successfully, failure in the tape mechanism is suspected and the operator is instructed to remount the tape on another mechanism. Other faults are monitored in a similar manner, and throughout, the operator and engineers are informed of any detected faults. Provision is made for the program using the tape to "trap" persistent tape errors and thereby to take action suitable to the particular problem, which may be more straight-forward and efficient than the standard supervisory action.

Addressing of new tapes and re-addressing of faulty tapes are carried out on the computer by supervisory routines called in by the operator. A tape mechanism is switched to "addressing mode", which prohibits transfers to and from the core store, permits writing from the computer to the reference tracks and to the block addresses on tape, and activates a timing mechanism to space the block addresses. When a new tape is addressed, addresses are written sequentially along the tape and the area between leading and trailing block addresses is checked by writing ones to all digit positions and detecting failures on reading back. Any block causing failure is erased and the tape spaced suitably. On completion, a special block address is written to indicate "end of tape" and the entire tape is then checked by reading backwards. Any failures cause entry to the re-addressing routine. Finally, the tape mechanism is returned to "normal" mode, a little block is written containing the number of blocks on tape, a tape number, and the title "Free", and the tape is made available for use. A tape containing faulty blocks is re-addressed, omitting such blocks, by entry to the re-addressing routine with a list of faulty blocks; the faulty blocks are erased and the remaining blocks are re-labelled sequentially, the tape being checked as when addressing a new tape.

5. PERIPHERAL EQUIPMENT

Peripheral Interruptions

As mentioned in the Introduction, a large number and variety of peripheral equipments may be attached to Atlas. However, the amount of electronics associated with each equipment is kept to a minimum, and use is made of the high computing speed and interruption facilities of Atlas to provide control of these equipments and large scale buffering.

Thus the paper tape readers, which operate at 300 characters per second, set an interrupt flip-flop whenever a new character appears. (Characters may be either 5 or 7 bits depending on which of two alternative widths of tape is being read.) Similarly the paper tape punches, and the teleprinters which print information for the computer operators, cause an interruption whenever they are ready to receive a new character; these equipments operate at 110 and 10 characters per second respectively.

The card readers read 600 cards per minute, column by column, and interrupt the computer for every column. The card punches at 100 cards per minute, punch by rows and interrupt for each row.

The printers, which have 120 print wheels bearing 50 different characters, cause an interruption as each character approaches the printing position so that the computer may prime the hammers for those wheels where this character is to be printed. There are therefore 50 interruptions per revolution, or one every 1-1/2 millisecs.

All the information received from, or sent to, these peripheral equipments does so via particular digit positions in the V-store. For example, there are 7 such bits for each tape reader, and 120 for each printer, together with a few more bits for control signals.

The majority of interruptions can be dealt with simply by the interrupt routine for the particular type of equipment. Thus the paper tape reader interrupt routine normally has merely to refer to a table of characters to apply code conversion and parity checks, and to detect terminating characters and if all is well to store the new character in the next position in the store.

The interrupt routines for the printers and card punches are not expected to do the conversion from character coding to row binary; this is done by an S.E.R. before the card or line of print is commenced. The card routines are however complicated by the check reading stations; punching is checked one card cycle afterwards, and reading is checked 3 columns later. The interrupt routines apply these checks, and in the event of failure a monitor S.E.R. is entered.

The printer interrupt routine counts its interruptions to identify the character currently being printed; a check is provided once each revolution when a datum mark on the printer shaft causes a signal bit to appear in the V-store. This counting is maintained during the paper feed between lines, which occupies several milliseconds.

Attention by Operators

Whenever equipment needs attention it is "disengaged" from the computer. In this state, which is indicated by a light on the equipment and a corresponding bit in the V-store, it automatically stops and cannot be started by the computer.

The operator may engage or disengage an equipment by means of two buttons so labelled. The equipment may also be disengaged by the computer by writing to the appropriate V-store bit, but the computer cannot engage it.

The "engage" and "disengage" buttons do not themselves cause interruptions of the central computer. Instead, the "engaged" bits in the V-store are examined every second (this routine is activated by the clock interruption) and any change activates the appropriate S.E.R. Disengaging a device does not inhibit its interruptions, so that if the operator disengages a card machine in mid-cycle to replenish the magazine or to empty the stacker, the cycle is completed correctly.

There are also other special controls for particular equipments, e.g., a run-out key on card machines, and a 5/7-hole tape width selector switch on punched tape readers.

Most devices have detectors that indicate when cards or paper are exhausted or running low. These correspond to bits in the V-store that are read by the appropriate S.E.R. The paper tape readers however have no such detector, and the unlikely event of a punched tape passing completely through a reader (due to the absence of terminating characters) appears to the computer merely as a failure to encounter a further character within the normal time interval. This condition is detected by the one-second routine.

Store Organization of Input and Output Information

In general, input information is converted to a standard 6-bit internal character code by the interrupt routine concerned, and placed in the store 8 characters to a word. (An exception to this occurs in the case of card readers when they are reading cards not punched in a standard code, in which case the 12 bits from one column are simply copied into the store and occupy two character positions. A similar case is 7-hole punched tape, when this is used to convey 7 information bits without a parity check. Such information is distinguished by warning characters, both on the input medium and in the store.)

A certain amount of supervisor working space in the core store is set aside to receive this information from the interrupt routines, and is subdivided between the various input peripherals. The amount of this space depends on the number and type of peripherals attached; the first two Atlases will normally use one block (512 words). This block will be locked down in a page of the core store whenever any input peripheral is operating (i.e., most of the time).

As each input equipment fills its share of this block, the information is copied by an S.E.R. into another block devoted exclusively to that equipment. These copying operations are sufficiently rare that the latter block need not remain in core store in the meantime; in fact it is subject to the same treatment as object programs by the drum transfer routine, and may well be put onto a drum and brought back again for the next copying operation. Thus only one page of core store is used full time during input operations, but nevertheless each input stream finds its way into a separate set of blocks in the store.

The page that is shared between input peripherals is subdivided in such a way as to minimize the number of occasions on which information must be copied to other blocks; it turns out that the space for each equipment needs to be roughly proportional to the square root of its information rate.

Similarly, information intended for output is placed in a common output page, subdivided for the various output devices, and is taken from there by the interrupt routines as required. The interrupt routines for teleprinters and tape punches do the necessary conversion from the internal character code used by the device. As soon as the information for a particular device is exhausted, an S.E.R. is activated to copy fresh information into the common output page. Again, the page is subdivided roughly in proportion to the square roots of the information rates.

For card punches and printers, whose interrupt routines require their information arranged in rows of bits, a further stage of translation is necessary. In these cases, on completing a card or line, internal 6-bit characters are converted by an S.E.R. into a card or line image also in the output block. In fact, the desirable amount of working space for output buffering somewhat exceeds one block, and the spare capacity of the subsidiary store will be utilized to augment it.

6. THE OPERATING SYSTEM

The following is a synopsis of work explained in detail in a paper the Computer Journal.

Input

The fast computing speed of Atlas and the use of multiple input and output peripheral equipments enable the computer to handle a large quantity and variety of problems. These will range from small jobs for which there is no data outside the program itself, to large jobs requiring several batches of data, possibly arriving on different media. Other input items may consist of amendments to programs, or requests to execute programs already supplied. Several such items may be submitted together on one deck of cards or length of punched tape. All must be properly identified for the computer.

To systematize this identification task, the concept of a "document" has been introduced. A document is a self-contained section of input information, presented to the computer consecutively through one input channel. Each document carries suitable identifying information (see below) and the supervisor keeps in the main store a list of the documents as they are accepted into the store by the input routines, and a list of jobs for which further documents are awaited.

A job may require several documents, and only when all these have been supplied can execution begin. The supervisor therefore checks the appearance of documents for each job; when they are complete the job scheduling routine is notified (see below)

Normally, the main core and drum store of the computer is unlikely to suffice to hold all the documents that are waiting to be used. The blocks of input information are therefore copied, as they are received, onto a magnetic tape belonging to the supervisor, called the "system input tape." Hence, if it becomes necessary for the supervisor to erase them from the main store, they can be recovered from the system input tape when the job is ready for execution.

The system input tape thus acts as a large scale buffer, and indeed it plays a similar part to that of the system input tape in more conventional systems. The differences here are that the tape is prepared by the computer itself instead of by off-line equipment, and that there is no tape-handling of manual supervision required after the input of the original documents - an important point in a system designed to handle many miscellaneous jobs.

This complete bufferage system for input documents is called the "input well." Documents awaiting further documents before they can be used are said to be in "input well A"; complete sets of documents for jobs from "input well B." Usually documents being accepted into input well B must be read from the system input tape back into the main store so that they are ready for execution; often however they will already be in input well A in the main store, so that only an adjustment of the block directory is required.

One result of this arrangement is that the same tape is being used both to write input blocks, in a consecutive sequence, and to read back previously written blocks to recover particular documents as they are required. The tape will therefore make frequent scans over a few feet of tape, although it will gradually progress forwards. The lengths of these scans are related to the main store space occupied by input well A. For example, so long as the scans do not exceed about 80 feet (130 blocks) the waiting time for writing fresh blocks will remain less than the time for input of three blocks from a card reader, so that comparatively little main store space need be occupied by input well A. To ensure that scans are kept down to a reasonable limit, any documents left on the system input tape for so long that they are approaching the limit of the scannable area are copied to the system dump tape (see below). If the number of these becomes large, the computer operators are warned to reduce the supply of documents through the input peripherals.

Output

The central computer can produce output at a much greater rate than the peripheral equipments can receive it, and an "output well" is used in a manner analogous to the input well. This well uses a "system output tape" to provide bulk buffering.

Output for all output peripherals is put onto the same tape, arranged in sections that are subdivided so that the contents of a section will occupy all currently operating peripherals for the same length of time. Thus if, for example, a burst of output is generated for a particular peripheral, it is spaced out on the system output tape, leaving spare blocks to be filled in later with output for other peripherals (this is possible because Atlas uses pre-addressed tape). In this way, the recovery of information from the tape into "output well B" as required by the various peripherals merely involves reading complete sections from the tape.

Again, there is a limit to the amount of information that can usefully be buffered on the output tape, due to the time required to scan back and forth between writing and reading regions, and this limit depends on the space available in the main store for output well B. An S.E.R. keeps a check on the amount of information remaining in output well B for each equipment, and relates this to the present scan distance to decide when to start to move the tape back for the next reading operation. If the amount of output being generated by object programs becomes too great some of it is put instead on the dump tape (see blow) or a program is suspended.

The System Dump Tape

The system input and output tapes operate essentially as extensions of the main store of the computer. Broadly speaking, documents are fed into the computer, programs are executed, and output is produced. The fact that the input and output usually spends some time on magnetic tape is, in a sense, incidental. This input and output buffering is, however, a continuous and specialized requirement, so that a particular way of using these tapes has been developed and special S.E.R.'s have been written to control them.

When demands on storage exceed the capacity of the main store and input and output tapes, a separate magnetic tape, the system dump tape, is used to hold information not required immediately. This tape may be called into use for a variety of reasons. Execution of a problem may be suspended and the problem recorded temporarily on the dump tape if other problems are required to fill the output well, or alternatively if its own output cannot be accommodated in the output well. Also, as already described, the input and output wells can "overflow" to the system dump tape. This tape is not used in a systematic manner, but is used to deal with emergencies. However, the system is such that, if necessary, the system input and output tapes can be dispensed with, thereby reducing the input and output wells and increasing the load on the system dump tape. In an extreme case, the system dump tape itself can be dispensed with, implying a further reduction in the efficiency of the system.

Headings and Titles

Every input document is preceded by its identifying information, mentioned above.

This consists of two lines of printing, forming the heading and the title respectively.

The heading indicates which type of document follows. The most common headings are

COMPILER followed by the name of a program language, which means that the document is a program in the stated language;

DATA which means that the document is data required by an object program; and

JOB which means that the document is a request for the computer to execute a job, and gives some relevant facts about it.

The last type of document is called a "job description." It gives, for example, a list of all other documents required for the job, a list of output streams produced, any magnetic tapes required and upper limits to the storage space and computing time required. Many of these details are optional; for example if storage space and computing time are not quoted a standard allowance will be made.

For example, if a program operates on two data documents which it refers to as data 1 and data 2, the job description would contain:

```
INPUT

    1 followed by the title of data 1
    2 followed by the title of data 2
```

The program would appear in this list as data 0. Alternatively, a job description may be combined with a program, forming one composite document, and this will usually happen with small jobs.

Each output stream may be assigned to a particular peripheral or type of peripheral, or may be allowed to appear on any output equipment. The amount of output in each stream may also be specified. For example, a job description may include:

```
OUTPUT

    1 LINE PRINTER 20 BLOCKS
    2 CARDS
    3 ANY
```

Each magnetic tape used by a program is identified by a number within the program, and the job description contains a list of these numbers with the title that appears in block 0 of each tape to identify it; for example:

```
TAPE

    1 POTENTIAL FIELD CYLIND/204-
       TPU5.
```

If a new tape is required, a free tape must be loaded, which the program may then adopt and give a new title. This is indicated thus:

```
TAPE FREE

    2 MONTE CARLO RESULTS K49-
       REAC-OR4.
```

The loading of tapes by operators is requested by the supervisor acting on the information in job descriptions.

Finally, the end of a document is indicated by

***

and if this is also the end of the punched tape or deck of cards it is followed by the letter Z. On reading this the computer disengages the equipment.

Logging and Charging for Machine Time

As problems are completed, various items of information on the performance of the computing system are accumulated by the supervisor. Items such as the number of program changes and the number of drum transfers are accumulated and also, for each job, the number of instructions obeyed, the time spent on input and output, and the use made of magnetic tapes. These items are printed in batches to provide the operators with a record of computer performance, and they are also needed for assessing machine charges.

The method of calculating charges may well vary between different installations, but one desirable feature of any method is that the charge for running a program should not vary significantly from one run to another. One difficulty is that the number of drum transfers required in a program may vary considerably with the amount of core store which is being used at the same time for magnetic tape and peripheral transfers. One method of calculating the charge so as not to reflect this variation is to make no charge for drum transfers, but to base the charge for computing time on the number of instructions obeyed in a program. This, however, gives no incentive to a programmer to arrange a program so as to reduce its drum transfers, and more elaborate schemes may eventually be devised. The charge for using peripherals for input and output can be calculated from the amount of input and output. For magnetic tapes, the charge can be based on the length of time for which the tape mechanism is engaged, allowance being made for the time when the program is free to proceed but is held up by a program of higher priority. All this information is made available to the S.E.R. responsible for the costing of jobs.

Methods of Using the Operating System

The normal method of operating the computer is for documents to be loaded on any peripheral equipment in any order, although usually related documents will be loaded around the same time. The titles and job descriptions enable the supervisor program to assemble and execute complete programs, and the output is distributed on all the available peripherals. Usually programs are compiled and executed in the same order as the input is completed, but the supervisor may vary this depending on the load on different parts of the system. For example, a problem requiring magnetic tape mechanisms which are already in use may be by-passed in favour of a problem using an idle output peripheral; a problem which computes for a long time may be temporarily suspended in order to increase the load on the output peripherals. By these and similar methods, the S.E.R. responsible for scheduling attempts to maintain the fullest possible activity of the output peripherals, the magnetic tape mechanisms and the central computer.

Documents may also be supplied to the computer from magnetic tapes; these tapes may be either previous system input tapes or library tapes or tapes on which "standard", frequently used, programs are stored. Such documents are regarded as forming part of Input Well B and are read into main store when required. An alternative method of operating may be to use the computer to copy documents to a "private" magnetic tape, rather than to use the system input tape, and at a later time to supply the computer with a succession of jobs from this tape. Similarly, output may be accumulated on a private magnetic tape and later passed through the computer to one or more peripheral equipments. Routines forming part of the supervisor are available to carry out such standard "copying operations.

Provision is also made for the chief operator to modify the system in various ways; for example, priority may be given to a particular job, or a peripheral equipment may be removed from general use and allocated a particular task. An "isolated" operating station may, for example, be established by reserving a particular output equipment for use by problems loaded on a particular input equipment.

7. Conclusion

The Atlas supervisor program is perhaps the most advanced example so far encountered of a program involving many parallel activities, all closely interconnected. Although a great deal of it has already been coded at the time of writing, there are still a few details to be thrashed out, and no doubt many changes will have to be made to suit conditions existing at various installations. The overall structure of the program is therefore of prime importance: only if this structure is adequate, sound and systematic will it be possible to complete the coding satisfactorily and to make the necessary changes as they arise.

The structure described in this paper, involving interrupt routines and "supervisory extracode routines" controlled by a co-ordinating routine, has proved eminently satisfactory as a basis for every supervisory task that has so far been envisaged, and it is expected that all variations that might be called for will fit into this structure.

There is no doubt that its success as a supervisory scheme for a very fast computer is due largely to certain features of the Atlas hardware, in particular the provisions for protecting programs from interference, and the page address registers. The way in which the latter permit information to be moved around at run time without the need to recompile programs gives the supervisor an entirely new degree of freedom. The possible uses of this facility have turned out to be so extensive that the task of utilizing such a large computer effectively without it seems by comparison to be almost impossible.

Acknowledgments

The work described in this paper forms part of the Atlas project by a joint team of Manchester University and Ferranti Ltd., whose permission to publish is acknowledged. The authors wish to express their appreciation of the assistance given by the other members of the Atlas team.

A SYNTAX DIRECTED GENERATOR*

Stephen Warshall
Computer Associates, Inc.
Woburn, Massachusetts

I. Introduction

The recent proliferation of algebraic translators or "compilers"—programs which translate from an algebraic language (like ALGOL, IT, or L_0) to the hardware language of a digital computer—has stimulated a good deal of work on techniques of reducing the construction cost of such programs. There have been several essentially different approaches to this problem, notably:

1. The development of a common intermediate language (UNCOL for Universal-Computer-Oriented Language [1]); for each algebraic language there would be written a translator from that language to UNCOL and for each new machine there would be written a translator from UNCOL to the language of that machine.
2. The development of general-purpose translators which accept descriptions of the particular languages between which translation is to be effected. Such programs have been called "syntax-directed" compilers, because the general algorithm is driven by what are in essence tables of syntax.

It is our feeling that the second approach is a superior one and the sequel describes work performed from that point of view.

Many investigators in the field have devoted considerable effort to identifying and separating out the essential functions of algebraic translation from the idiosyncratic special functions suggested by the particular source language and target machine in question. These researches have met with great success with respect to the first phase of translation: the syntactic analysis of input strings. However, the second phase—the synthesis of target-language statements—has proved more difficult to generalize. This is, of course, not at all surprising. The algebraic languages so far considered are all nice, well-behaved, formal languages for which "destruction rules" can readily be deduced from their customary description (which is usually given in production rules, "Backus normal form" [2], or some equally straight-forward formalism). Thus, one may easily construct a simple language-independent algorithm which uses syntax-descriptive tables for a particular language to effect analysis of strings in that language. The sorts of information to be found in the tables (priority of operators, scopes, and the like, or rules for syntactic type formation) are readily derivable from the language description, and the processing algorithm itself may be rather small. Extremely elegant programs of this kind have already been constructed [3]. On the other hand, digital computers seem to have been planned with positive malevolence from the viewpoint of translator design. The formal implications of the order vocabulary of a machine are highly particular and almost defy uniform treatment. Each machine has a plethora of special-purpose registers and special-purpose orders

*This work was begun by the author at Technical Operations, Inc. under Air Force Contract AF 33(600)-41896 and completed at Computer Associates, Inc. through a subcontract from Technical Operations, Inc. under the same prime contract.

designed to increase efficiency; since a good "generator" (by this word we designate the second, synthesis phase of the translator) must produce a program which is not only correct, but efficient, a way of generalizing and tabulating these individual quirks and peculiarities is a necessity for a satisfactory solution. This generalization has proved very elusive, and some clever schemes for sidestepping the issue by implicitly putting the machine characteristics into the source language syntax tables have, therefore, been attempted with some success [4]. These methods suffer, however, from two inherent defects. First, the sorts of tables needed for those syntax descriptions essential for analysis are extremely inappropriate and clumsy forms in which to describe the large structures used in generating globally good code. Second, every syntax table in the compiler is perturbed by a change of target machine. In our efforts to devise techniques for a more completely explicit and separate tabularization of the machine description, we came to the conclusion that the feasibility of this approach dependedupon certain features of compiler organization. We were then led to the development of a novel organization which appears to be of considerable interest. In order to explain our approach, it is necessary to make a brief digression into the organization of translators.

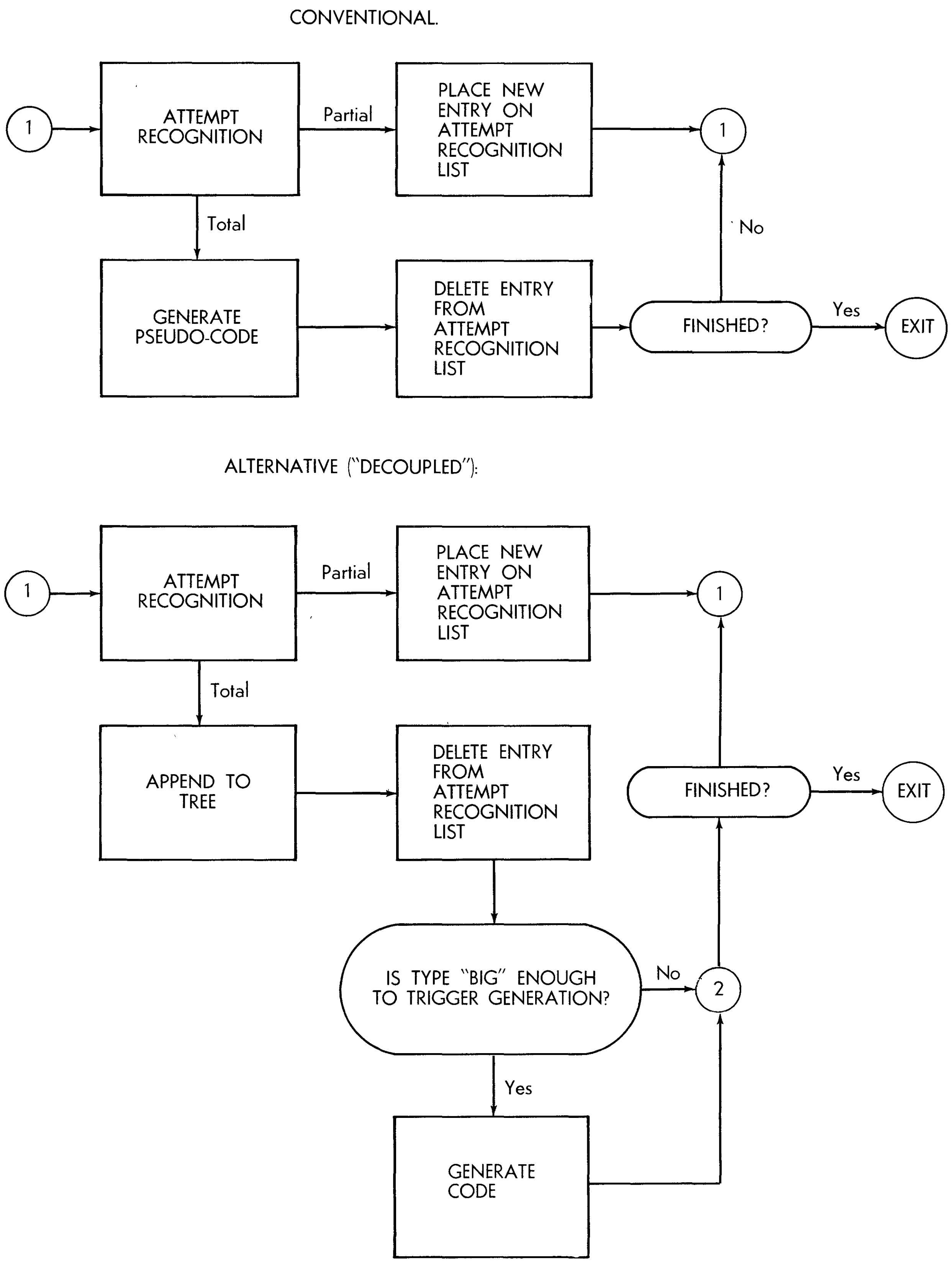
II. Recognition and Generation: Conventional Organization

Characteristically, algebraic translators are organized as follows: The first phase (or "analyzer") courses through the input string endeavoring to "recognize" (completely determine the syntactic type of) some thing "big" enough syntactically to require the generation of code. Generally, some forms are partially recognized—and presumably placed on a list—before a form of interest has been totally recognized. Once this recognition has occurred, the generator is called in; the latter is given the name of the type recognized and the names of the variables being combined to form it. The generator selects a "paragraph" (a sequence of lines of code), inserts the names, places the result at the end of the output, and returns control to the analyzer. (Frequently, the paragraph selected is not of code proper, but rather of some "machine-independent" one- to three-address pseudo-code; this difference does not affect the sequel.)

This conventional organization follows directly from the observation that, in the familiar algebraic languages, the order in which syntactic types are completely recognized corresponds very closely with the order in which independent acts are to be performed by the computer to accomplish the computation specified by the input; in short, the order of generation is close to the order of recognition. Generated code is usually carried as a sequence of independent atomic commands whose origins and relationships to each other are lost, since this information has no further syntactic interest and uses up storage.

This conventional organization has considerable natural appeal; the translator does as much as it can at any time, thus minimizing the requirements for temporary storage and reducing redundant passes. However, there are certain important drawbacks.

Increase in the efficiency of generated code is gained in two ways: first, by judicious use of the properties of the special (e.g., arithmetic or index) registers of the machine, a matter we will take up later; and second, by permitting dependence of the criteria for paragraph selection upon arbitrarily large amounts of contextual information. This latter procedure is made inherently clumsy, if not downright impossible, by the standard organization of translators. Even though we may have completely recognized a syntactic form and may, therefore, be able to select a correct paragraph of code, it may still be the case that selection of a best paragraph should wait upon the determination of context which has not yet been recognized. When contextual search is to be performed, it is precisely the lost elements of the string's global structure which are the data of the search. Thus, we are not interested—except in the trivial sense of eliminating very local redundancies—in the fact that some addition follows a multiplication in recognition sequence; we are concerned, rather, with the fact that some addition is syntactically in the scope of an assignment which is in turn within the scope of a relation. The recognition of such complex contextual structures is essential to any powerful generator, and if the information needed to perform this kind of recognition is either lost or well-buried, the cost of post-editors to improve output code efficiency becomes extremely high.



CONVENTIONAL.

1
ATTEMPT RECOGNITION
Partial
PLACE NEW ENTRY ON ATTEMPT RECOGNITION LIST
1
Total
No
GENERATE PSEUDO-CODE
DELETE ENTRY FROM ATTEMPT RECOGNITION LIST
FINISHED?
Yes
EXIT

ALTERNATIVE ("DECOUPLED"):

1
ATTEMPT RECOGNITION
Partial
PLACE NEW ENTRY ON ATTEMPT RECOGNITION LIST
1
Total
APPEND TO TREE
DELETE ENTRY FROM ATTEMPT RECOGNITION LIST
FINISHED?
Yes
EXIT
IS TYPE "BIG" ENOUGH TO TRIGGER GENERATION?
No
2
Yes
GENERATE CODE

Figure 1. Compiler Organization

III. Recognition and Generation: Alternative Organization

We therefore decided to effect a complete decoupling of the two phases of translation, permitting a complete analysis of arbitrarily large chunks of the input string to antedate any generation. In fact, it is by no means necessary to analyze a whole program before generation; there is always in practice some syntactic type past the boundaries of which flows no information of interest to the generator. One simply analyzes up to this type, generates over the result of this analysis, then returns to the analyzer as before. The point is that this type will, in general, be substantially larger than the smallest type completely recognized, which is what drives the conventional generator.

A number of advantages accrue from this sort of organization. First of all, the decoupling of recognition from generation makes convenient the use of quite complex and oddly sequenced contextual search to determine paragraph selection. Second, this same decoupling makes the translator usable over a wider class of languages, including those in which recognition order and generation order have very little to do with each other. This last feature is of more than theoretical interest, for not only do natural language translations appear to have this property, but also certain postulated data-processing languages require such freedom from sequencing constraint; we will return to this topic briefly in a later section. A third benefit which derives from this arrangement is a substantial reduction of the workload of the analyzer, obtained indirectly by making contextual inspection easy for the generator. In a language in which the local semantics (and thus paragraph selection) are highly context-dependent, the analyzer commonly spends a good deal of its time considering the input string, deciding which version of some operator is meant this time. Since an efficient generator must be able to do this sort of thing with facility anyway, it seems sensible to let the generator perform as much of the task as possible and restrict the role of the analyzer to that of "parsing" the input string.

Thus, the analyzer in our scheme is an extremely short program which utilizes information about the scope and priority of operators—the "destruction rules" of the language—to produce a tree-representation of the input string in which terminal nodes are names (of numbers, functions, etc.) and nonterminal nodes are operators or combiners. The only context problems that it need handle are those inherent in the destruction rules (inversion of operator priorities in the scope of some other operator, and the like). A tree was chosen as the vehicle for communication between the analyzer and the generator since it explicitly displays the formation history of the input string and thus is a very natural domain over which to define contextual search algorithms and contextual naming schemes of various kinds.

IV. The Generator: Gross Description

The input to the generator is a collection of tables. First, there is the SOURCE table, which is output by the analyzer and is essentially a tree-representation of the input string. (In our prototype, we use a binary tree to simplify naming; this is, of course, no restriction.) There is an algorithm, whose description we will defer for the moment, called DETSYN (for Determine Syntactic Type), which, given a place on the tree, responds with a type number. This number is referred to as the "syntactic type" of the "tree node" named by the input variable and is determined on the basis of an arbitrarily complex, table-driven search of context. There is a table called SYNTAB (Syntax Table), each line of which is associated with exactly one of these syntactic types. A line of SYNTAB includes a collection of indicators which dictate the actions to be performed when a node of that type is encountered and the decision is made to process it. One particular item on each line of SYNTAB is the item called SEQ, for Sequencing Type. When a given syntactic type is encountered it may be decided to process (for example) its two "sons" in the tree before processing the node itself; it is for selecting among sets of such sequencing rules as this that SEQ is provided. One item of the table SOURCE is a Counter (CTR) which counts the number of times the generator has coursed through each node.

Sequencing, then, operates as follows: the sequencing variable (NODE) is initialized to the "trunk" of the tree. At each stage DETSYN(NODE) gives a line number in SYNTAB. The ordered pair given by the SEQ of this line and by the value of the counter

CTR(NODE) together select an element of a small sequencing matrix (SEQMAT) whose entries can assume four values, indicating the decision to either process this node now, or replace NODE by its father, first son, or second son in the tree, respectively.

Once the decision has been reached to process a node, an algorithm called PRCNOD is invoked. This routine essentially runs through the tasks indicated by the items in the line of SYNTAB associated with the syntactic type of the node being processed. There are about a half dozen such tasks possible, some of them employing rather ornate trickery. The most interesting task to be performed is quite straightforward, however, and that is the selection and preparation of paragraphs for output.

Sample paragraphs of code are effectively stored (really, pointers to them are stored) in a table called PARA, in which are indicated the number of holes to be filled, the "relative tree names" of the variables which are to fill them, and a variety of other information associated largely with local redundancy elimination. In the general case, there may be several paragraphs associated with a given line of SYNTAB. When a particular node is being processed and there exist entries in PARA for its syntactic type, PRCNOD picks up all the possible paragraphs—these are semantically interchangeable—converts "relative tree names" to "absolute tree names," and throws the result onto a tentative output list called LNKTAB. In practice the paragraphs themselves are not copied; we need merely record the location in PARA where the paragraph may be found and copy the tree-absolute version of the names.

In time some node is processed whose syntactic type directs output (in our notation, a line item of SYNTAB, OUTPUT (DETSYN (NODE)) = 1). When this occurs, a routine called OUTSEL (for Output Selector) is called in; OUTSEL courses through the tentative output indicated by LNKTAB and selects the best of the alternative paragraphs at each stage on some simple criterion (in our experimental version, shortest code is the desideratum), suppressing redundant lines as necessary. Then the resulting code is thrown out of the machine by a small routine called TOFILE.

This very brief description of the gross organization of the generator should serve as background to the ensuring action, which will briefly trace the treatment of certain special matters by the various parts of the generator.

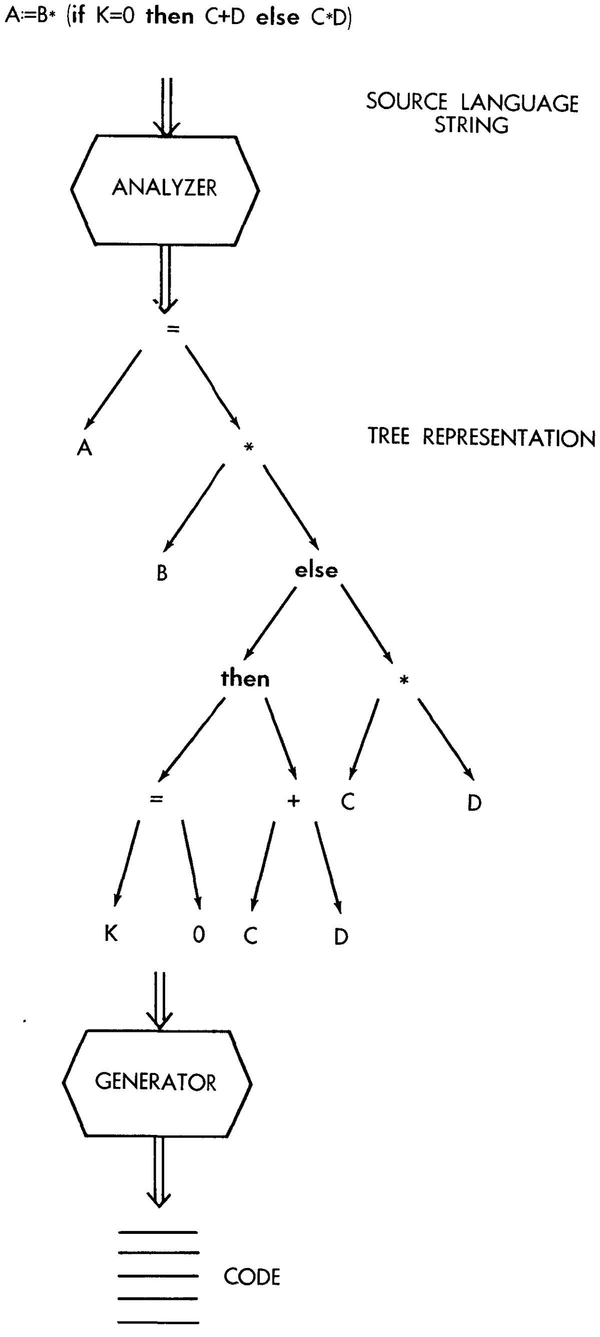


Figure 2. Phases of the Complier

V. The Generator: Some Matters of Detail

1. Names: We have used the terms "relative tree name" and "absolute tree name"

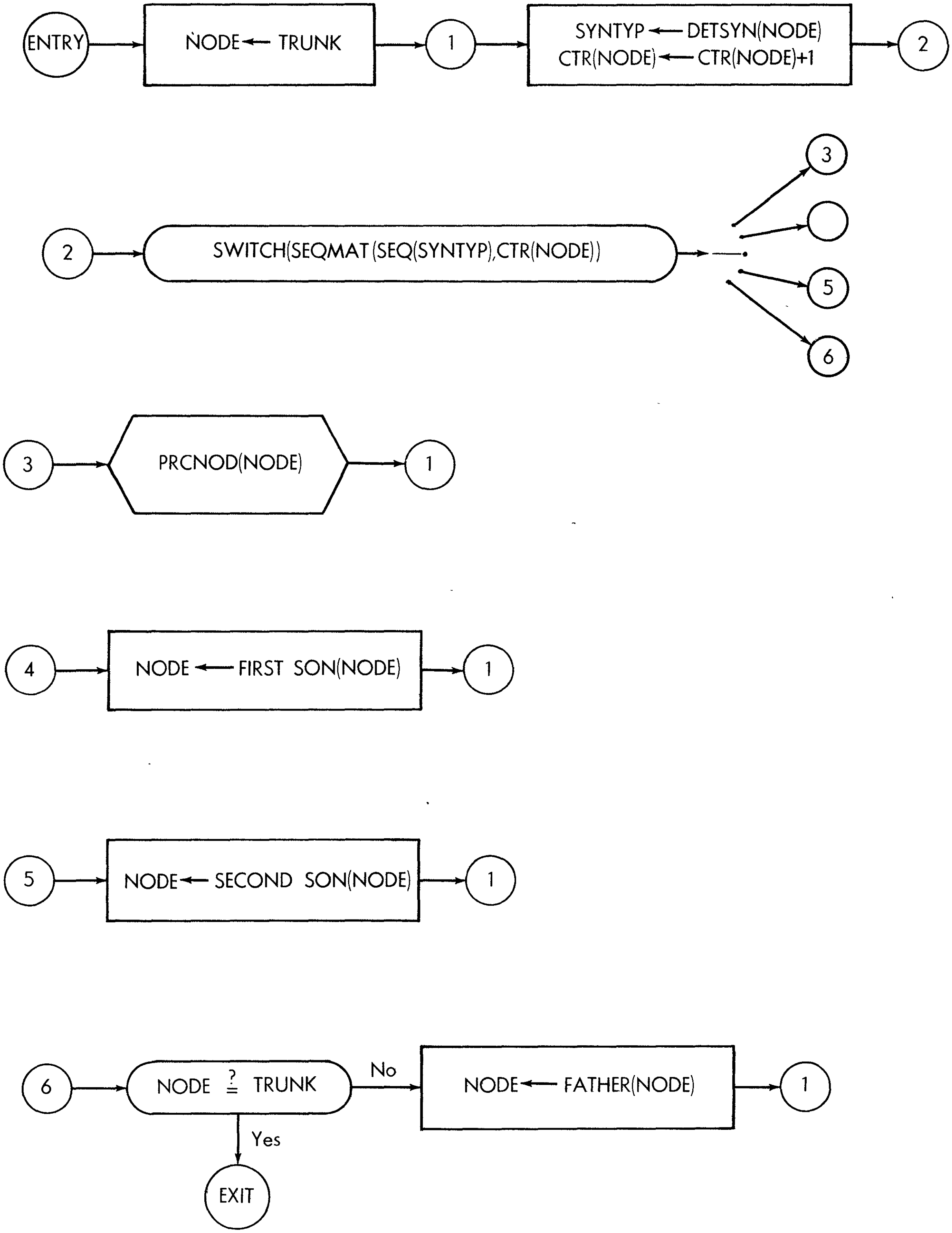


Figure 3. Generator Sequencing Control

without definition, in order to avoid adding to the inevitable clutter of an English description of a flow chart. At this point, it seems appropriate to state that by an "absolute tree name" we mean a line number in SOURCE; by a "relative tree name" we mean, formally, a function whose domain is the set of absolute tree names and whose range is the union of its domain and a special "failure" marker.

The intent of a relative tree name is to signify something like "the second son of this node" where "this" is replaced by the input variable while an absolute tree name is something like "node number (SOURCE line number) 63." In PARA, a typical paragraph might include a hole which is to be filled by the name of the second son of the node which causes its selection. If PRCNOD selects this paragraph, this relative name will be picked up and replaced by its absolute equivalent, i.e., by the name of the second son of the node currently being processed. This will either be a node number (a line number in SOURCE) or, in the case of a terminal node (hence, denoting a simple operand), the name of an entry in a symbol table. If this absolute tree name is a node number (and the line is in fact selected for output by OUTSEL), it will be converted either to the name of a temporary storage location (if it denotes the result of an operation) or to a relative address (if it denotes a jump destination); this last conversion is handled by TOFILE.

2. Special Registers: Each special register (accumulator, index register, or whatever) may be assigned a name (which is, naturally, a number). Also, a set of special registers may be assigned a name. In PARA, it may be indicated that a hole is to be filled by a set of special registers; this is interpreted to mean that one element is to be selected from that set, and that all other references to that set are to be interpreted as references to that element for the scope of the paragraph. A typical use of this facility is to indicate the set of index registers. For transferring information between the special registers, there is a matrix (n x n, where n is the number of named special registers (not sets)) whose (i, j)th entry points to code in PARA for moving a number from the ith special register to the jth. Associated with lines of PARA is such information as the effect (usually destruction, or nothing) of each line upon the special registers and a marker to indicate that a line is a "suppressible fetch" (a line whose sole function is to get a number into a particular special register, and which is completely eliminable if the number is already there).

3. The Output Selector (OUTSEL): The output selector would ideally consider all the possible alternative codes indicated in LNKTAB and globally optimize. Such an OUTSEL is trivial to construct, but seemed unnecessary for our purposes, since the cost of searching so many alternatives was adjudged excessive in comparison with the expected increase in efficiency. Therefore, our prototype OUTSEL operates as follows: it considers in turn the alternative paragraphs at each stage. For each of the possible choices it makes a cost estimate, checking the state of the special registers to see if lines may be suppressed, fixing the local values of sets of special registers, keeping track of anything that is being destroyed, inserting lines to transmit data from one special register to another if this will cheapen the cost, using the running time (in microseconds) carried in PARA for each line as the criterion of cost. Then it selects the cheapest paragraph, destroys the others, updates the condition of the special registers as of this end of this paragraph, and begins relative costing of the next set of alternatives.

4. Temporaries: Some paragraphs leave a "result" in a special register; these are marked accordingly. When OUTSEL selects such a paragraph, the special register records are marked and future allocation of quantities to special registers is constrained to avoid destruction of the contents of this special register. When a "suppressible fetch" of the quantity (node) in question is encountered, OUTSEL assumes that the quantity is being used and destroys the marker (called SAVE) which protects it. If the special register must be used before a protected quantity has been utilized, lines are inserted by OUTSEL which put the quantity away in some temporary; the temporary is named by the number of the node whence came the paragraph of which it is the result, and that name is recorded on a list. Future references to that node are converted to references to temporary number x, where x is the line number (on the list of temporary names) of that node number. The total number of spaces for temporaries allotted at the end of the translation is set equal to the largest value

the index of the temporary list has ever assumed during generation (TOFILE reinitializes this index).

5. Handling of location names: Some lines in SYNTAB direct the marking of certain nodes as places which are to be jumped to (or "breakpoints"). Thus, OUTSEL can assume nothing about the state of the special registers at the time the first code generated for the subtree named by such a node (i.e., the subtree of which the node is the trunk) is entered. PRCNOD, whenever it selects paragraphs, checks whether the node being processed either is itself, or has an ancestor that is, marked as a breakpoint. If this is the case, the currently generated LNKTAB entry is so marked (for OUTSEL to see later) and the node marker is erased. In a similar manner, absolute tree names which name locations rather than values are carried down to the first code-producing node in the subtree named by the originally named node.

6. A special bit of trickery: The contents of a line of SYNTAB may direct PRCNOD to set or release a special kind of lock (called LOCSAV) on the contents of a special register. The special register may either be named directly or named by a name which may be interpreted as "the special register selected from among set S by that node over there." This rather odd kind of special register name has its analogue in a kind of special register name, used to describe holes in PARA, which means something like "the special register locked by that node over there." The purpose of all this is to provide a facility for passing around special register set names which are to be constant over a scope larger than a single paragraph. This is necessary in the production of optimal coding over large constructs, as in the generation of loop coding, for example. If the lock cannot be honored by OUTSEL, the contents of the special register are saved in the appropriate named storage cell.

7. Structural looseness in the generator: There are several ways to implant special algorithms in the generator without altering its basic structure. First, there is an item in SYNTAB called ALGO which may specify the name of an algorithm to be performed when nodes of this syntactic type are processed; this algorithm may be wholly arbitrary. Second, it should be observed that DETSYN is unconstrained in content, and may in particular reference (and alter) the values of the counter CTR. Thus, the very same node may be treated as a completely different syntactic type on different times through; since there is no restriction on the size of the sequencing matrix, it is possible (albeit clumsy) to set arbitrarily long sequences of algorithms at a node by just having it identified as a different syntactic type each time through, having ALGO active at each of the corresponding lines of SYNTAB, and having a long stretch of "process this node now" indicators in SEQMAT (or even having DETSYN or ALGO decrement CTR).

VI. An Evaluation of the Generator: The Current Status

From the foregoing description, it should be manifest that the current generator has fallen short, in several respects, of the original goal: complete tabularization of machine description. The tables of special register description and communication do indeed seem a decent first cut at a part of the problem. The characterization of paragraphs in terms of their effect upon special registers and of fetch instructions in terms of their intent also appears a step in the right direction. The handling of complex contextual search, however, is a considerable departure from our initial intent. Ideally, one should develop a means of tabulating the machine's programming manual, and the general compiler should deduce the strategy of the desired contextual search from such tables. This beautiful solution was rapidly abandoned, not so much because of the difficulties inherent in devising tables—although these difficulties are considerable—as because the tabular structures, once defined, would not be the most appropriate vehicles for automatic evaluation of coding strategy.

The procedure of searching through the order vocabulary of a digital computer and devising useful larger constructs—handy paragraphs of code—is an activity at which people are very good and machines, very bad. Without such constructs already to hand, the process of optimal code generation is a combinatorial problem of astronomic proportions. It seemed natural, therefore, to alter our goal; we would continue to permit people to intervene rather extensively between the information in the manual and the compiler. We would view this intervention as a prepass, designed to shrink the information contained in the manual into a set of reasonable size. Then we would bend all effort to the development of a reasonable scheme for tabularizing this second, smaller inventory of information. Our compiler organization, by creating an intermediate form (the tree) in terms of which it is easy to describe contextual constraint, has provided the necessary framework for such a scheme.

Any evaluation of the current generator as an element of a "universal compiler" is in large measure an estimation of, on the one hand, its completeness (i.e., its suitability for all machines of at least the current generation) and, on the others, its facility of use by the programmer who wishes to insert tables for a particular machine. The only data we have so far on these matters are, of course, those derived from our own experience.

A prototype compiler composed of an analyzer and a generator, both syntax driven, has been constructed and run on the IBM 7090 computer. Tables have been constructed to control translations from L_0 (the algebraic language of the CL-I Programming System [5]) to 7090 code and from ALGOL-60 to CDC 1604 code. This compiler was experimental in nature and designed to test the suitability of our technique. No effort was made to minimize either the size or the running time of the compiler itself other than the utilization of a competent programming staff.

We found that the primitive analyzer required by our scheme need not exceed a few hundred words in length; the generator seemed to demand slightly in excess of 1000 words of code. The time required to construct control tables was impressively short: a complete set of tables for translation from one of the source languages to one of the machines could be constructed by an expert programmer in less than one week. The control tables themselves for a given translation occupied about 1500 36-bit words. The running time of the compiler was at least as fast as that of comparable conventional compilers.

Our conclusion is that a useful syntax-driven generator, and therefore a more or less adequate "universal compiler," is feasible; in fact, we have built one. It is our claim that, given such a device, a fairly high quality program for translation from any algebraic language of the usual kind of the language of any of the present generation of machine can be produced in a matter of man-weeks.

It is by no means our feeling that the prototype generator and its associated analyzer described in this document are in any sense ultimate. Research is currently being carried on along several lines, among them:

1. To determine the relative merits of source language description by formation rules versus by scopes and priorities.

2. To determine the payoff in going to n-ary trees as an intermediate form which appears to make algebraic macro definition and the sensing of common sub-expressions much cheaper, but renders tree naming more expensive.

3. To devise better external languages for defining the control tables, so that they may be built more readily by personnel unfamiliar with the details of the compiler.

4. To consider the implications of doing partial determination of three syntactic types during the analysis phase.

In addition, the criterion of short compiling time is being applied, now that we are convinced of the feasibility of the technique. A second version of the compiler, incorporating the results of this additional work, should be completed by the time this paper appears in print.

VII. Additional Payoffs of the Generator

Although the primary motivation for the design of the generator was to reduce the high cost of compilers, several other advantages appear to accrue from the design to which we were led, and several new paths of investigation are strongly indicated. In this last section, we will assemble a few rather disconnected remarks about these additional matters.

1. Remark on Natural Language Translation

Although the analyzer used in our system or indeed any analyzer built to handle simple formal languages is obviously useless for any messy source language, it is our feeling that the generator—or at least its organization—may be of some use in this area. Given a parse—in the schoolmaster's sense—of a sentence in the form of a tree, it seems to us that a rather natural way to describe the synthesis of target language statements, even in a nonformal language, is in terms of a sequencing discipline through the tree and a decoupled set of local phrase formation rules, stated in terms of tree names and subject to arbitrarily complex contextual control (also stated in tree terms) of local interpretation. Although we claim no especial competence in this area, the naturalness of the technique does seem rather convincing in the few examples we have tried, and some restricted natural languages (constrained English for querying data files, for example [6]) with which we are familiar seem to fit into some such framework.

2. The System Environment

This translator was designed to operate in the environment of a large and powerful programming system with a large number of calls on the system fighting for time, space, and priority. The advantage, therefore, in a fixed algorithm for translation which can handle different languages (source and target) simply by switching tables is especially great. Moreover, we would like to call attention to the OUTPUT signal in SYNTAB. This signal is meant to be confined to syntactic types "large" enough so that temporary storage preservation, implicit naming records, and similar locally interesting information need not be preserved, by and large, after TOFILE has been invoked. This means that it is possible at these output barriers (if they are well selected) to interrupt processing with very little temporary information having to be saved. Aside from the obvious convenience of such stopping points for partial processing, there is a particular trick which should be mentioned. We envision several alternative sets of tables even for a given language-pair. Perhaps one set might have as few syntactic types as there are semantically distinguishable operators and no more than one paragraph per type. Another might have an enormous collection of extra syntactic types corresponding to a greater use of contextual aid in selecting code; it might also have several different paragraphs optional for many of its types. Presumably, the compiler would operate more slowly with the second set of tables, but produce superior code. The system, as a function of its current work load, the user's priority, and his desires (high quality code, fast compilation, or whatever) could decide which set of tables to use and revise its decision at any output barrier according to changes in workload or other similar criteria.

3. Language Embedding

The facility of language switching by table switching introduces some interesting possibilities for embedding of statements in one language inside statements in another. If one introduces a node type whose first son is the name of a language and whose second son is the trunk of a tree representation of a statement in that language, it is easy to see how the processing of such nodes might be trivially included in the processing of the charts in which they appear. One must, of course, exercise care in erecting the output barrier so-that no interesting syntactic types cross it.

4. Further Compiler Cost Reductions

4.1 Evolutionary redundancy elimination: Instead of trying to anticipate all redundancies, the compiler-builder can simply start with tables sufficient for reasonably good code, release the compiler for limited use, then extend the tables (or even let the user extend the tables) to eliminate the redundancies and inefficiencies which are found to be common.

4.2 Compiler construction by inexperienced personnel: It seems possible to refine the tabular descriptions required by the analyzer and generator to the point that--with suitable cheap translators, really rearrangers, editing to the refined form—very elementary descriptions of the algebraic language and of the intended machine code may automatically be transformed into a working compiler (albeit not a very good one, since paragraph selection, etc., will tend to be local and context-independent). Although this is not something which we expect to be able to do immediately, it does seem like a feasible goal with a table-driven compiler of this kind available. The implications of such an arrangement—in which users may approach the machine (plus programming system), describe the language they wish to use, and then use it—are, to say the least, rather appealing.

References

1. Strong, J., et al, "The Problem of Programming Communication with Changing Machines: A Proposed Solution," Communications of the ACM, August and September, 1958.
2. Notation used in Backus, J. W., et al, "Report on the Algorithmic Language ALGOL-60," Communications of the ACM, May, 1960.
3. Irons, E. T., "A Syntax-Directed Compiler for ALGOL-60," Communications of the ACM, January, 1961.
4. Glennie, A. E., "On the Syntax Machine and the Construction of a Universal Compiler," Technical Report #2, Carnegie Institute of Technology, Computation Center, July 10, 1960.
5. Leonard, G. F., "The CL-I Programming System User's Manual," Report #TO-B-61-3, Technical Operations, Inc., January, 1961.
6. Cheatham, T. E., Jr. and Leonard, G. F., "Exercise and Evaluation of Command and Control Systems," CA-61-2, Computer Associates, Inc., May, 1961.

AN AUTOMATED TECHNIQUE FOR CONDUCTING A TOTAL SYSTEM STUDY

A. O. Ridgway
International Business Machines Corporation
Federal Systems Division
Bethesda, Maryland

SUMMARY

In conjunction with a comprehensive analysis of the data processing at an Air Force Base, a system study technique was developed which utilized EAM equipment for recording and analyzing all primary data processing applications. The study effort also included the initial postulation of a total data processing system.

The automated study technique facilitated systems analysis by: relating significant data, providing thorough analyses of existing systems, simplifying documentation correction, and automatically preparing flow charts.

The study technique is completely independent of the data processing complex being analyzed, the organization involved, the mission it supports, and hardware considerations.

INTRODUCTION

Early Efforts at Systems Analysis and Design

In mechanizing data processing by means of punched-card equipment, different applications or procedures generally are considered on an individual basis. This situation is often caused by the fact that different business functions have been considered at different points in time and/or by different people. A strong contributing factor is the fact that, in general, punched-card equipment is very limited in its ability to perform multi-functions and, consequently, to automatically interrelate applications.

Unfortunately, although Electronic Data Processing machines have essentially eliminated the equipment restrictions inherent in punched-card machines, history has to a large extent repeated itself. As in the case of punched-card equipment, most application developments have taken place independent of one another. This independent development also has been caused, to some extent, by the initial emphasis being centered on automating existing procedures. It was only natural to want to calculate, sort, tabulate, etc., faster.

Motivation for Total Systems Approach

The independent growth of data processing applications has led to a high degree of redundancy in the storing and processing of information. There are few parts of a business' data processing which are isolated from all other data processing.

Efforts directed toward improving procedural efficiency frequently attack one phase of a data processing complex in order to solve an immediate problem. When one analyzes the file data from separate organizations within a business, one often finds a tremendous amount of duplication. Likewise, even a cursory examination of reports will reveal the same information repeated many times. By properly mechanizing the interrelation of data processing functions, these redundancies can be minimized. This same interrelation, when properly planned, can effect a greater system responsiveness because of the simultaneity with which data may be handled.

As attention was focused on EDP hardware, opportunities for procedural consolidations could be seen. Large memories made possible the storing of tables, so that, for example, labor hours could be accumulated for jobs performed simultaneously with the processing of payroll information. The use of magnetic tape made possible the design of records which in many cases encompassed information long needed, and encouraged consolidation of procedures, e.g., preparing insurance premium notices while, at the same time, processing inquiries regarding dividends, policy loan status, accumulated reserves, etc. With the advent of large random access memories, further consolidation was encouraged by having the ability to store a number of information files in rapidly accessible memory. This development brought forward the in-line processing concept typified in manufacturing by the simultaneous updating of such information files as inventory, sales, demand history, etc., based on a single input, such as an order. As one looks at advances in hardware design contrasted with that of systems design, it is clear that the former has far outstripped the latter. Today's hardware has the inherent capability to operate a total system for a good-sized business; the problem is designing the system.

Management Considerations

Interrelating data processing within an organization may be done by moving horizontally or vertically across organization and functional boundaries.

The building blocks, by which one may proceed to interrelate data processing functions, are the documentation of individual applications. The path along which the integration should take place must be decided. In most cases this decision will be based on economics. In the commercial world this means turning dollar savings into greater profits. In the military sphere this dollar savings may be used to develop a system more responsive to the mission of the establishment.

In a large system each organizational level will interpret policies and regulations differently. Hence, in an integrated system a strong management is essential. Management must have sufficient insight into local differences to intelligently compromise them into a single system and still allow for those "unreconcilables" that will always exist. Serious consideration must be given to how to handle regulatory changes, especially when these emanate from two or more independent organizational elements whose data processing functions have been interrelated.

A basic question of whether to aim at "total" integration or concentrate on an interim solution must be considered. There are advantages to both approaches. The somewhat paradoxical answer is that both approaches are needed if an effective job is to be done. The long-range concept furnishes a logical goal to work toward and an over-all framework within which to move in the interim. The incremental approach makes it possible to achieve early benefit, to gain immediate experience, and to pay and pave the way toward the ultimate goal. If one aims only at the long-range, "total" integration concept, his efforts are likely to be somewhat theoretical. If, on the other hand, one concentrates solely on the short-range interim improvements, he probably will not come close to the level of integration represented by the long-range approach.

Need for Study Technique

The complexity of developing a total system requires a logical study technique. One approach has been to determine the "mainline" of a business, chart the data processing directly associated with this central theme, and then attach functions less directly associated with the theme. For example, in the case of a manufacturing business, data processing centers around order processing; in the utility business, billing is the big data processing job. However, in large corporations and in many military facilities, regardless of size, data processing functions are so intermixed that one must be able to deal with all aspects of the problem with equal facility. One approach is to draw flow charts of existing separate data processing applications and by connecting the inputs and outputs to advance toward a total system. Tabular techniques offer an interesting alternative to flow charting.

A documentation technique aimed at the systems level with just enough detail to enable the systems analyst to understand the current system and to redesign its component parts to form a new systems approach was needed. The complexity of many systems is such that it precludes meaningful, manual manipulation of study data. Therefore, it was felt that a technique that would allow mechanized listing and analysis of data pertaining to today's system should be developed.

General Approach

Conduct of a total data processing system study has been divided into six phases:

1. Data Collection and Recording

Organization charts, policies, and statements of responsibility are studied. Areas of data processing to be studied are selected. Informal flow charts of present procedures are drawn. Basic information pertaining to the flow and storage of data is formally recorded on specially designed forms.

2. Analysis of Existing Systems

With information regarding the current system on punched cards, analyses may be made of data flow channels, similarities in contents of files, processing procedures, influence of time factors, redundancies in static data, etc. Current system requirements, in effect, are defined.

3. Total System Postulation

All data processing activities are grouped by common endeavor. A broad concept of a total system encompassing all data processing is developed. This system configuration is broadly defined in terms of central processing units (henceforth referred to as processors), memories, and input-output communication devices.

4. Concept Analysis

Data processing elements identified in the postulation of a total system are grouped by similar end product. The resultant grouping facilitates detailed systems design.

5. Detailed Systems Design

Process steps are broken down to a level of detail from which programming may be done or detailed procedures may be written. Input-output and file records are designed. Total system processes and records, with their respective frequencies and volumes, are used to determine the communication network of the system.

6. Determination of Functional Specifications

Volumes, sizes, and frequencies of the processing, input-outputs, and files are summarized in statements of: processing requirements, storage requirements, and communication requirements.

These phases are described more fully in succeeding sections.

Scope

The technique described in this paper was developed in conjunction with a comprehensive study of the data processing at an Air Force base. The original study included the data collection, recording and analysis of the existing systems. It later was extended to include the postulation of a new system. The study technique thus has been successfully applied to Phases 1, 2, and 3, which are described in detail. An outline of an approach to the remaining study phases is given.

Application of the study technique has been illustrated in terms of the air base analyzed. However, the technique is independent of the data processing complex being analyzed, the organization involved, the mission it supports, and hardware considerations.

DATA COLLECTION AND RECORDING (PHASE 1)

Background

The purpose of Phase 1 was to collect and record data in an organized, manageable form that would completely describe the existing system. This documentation, combined with an appreciation of management needs and long-range planning, show total system needs, its true purpose and the validity of various files and lines of communication.

Using maps, organization charts and mission statements, the broad view of the air base's functions was obtained. The study team gained a clear, qualitative understanding of the over-all systems structure and major relationships of the individual parts. Data collection was done by functional area. Areas studied were: supply, personnel, operations, finance, transportation, civil engineering, and maintenance.

Discussions with commanders revealed the mission of the various organizational units. Directives and manuals were studied. Data regarding the number of people employed, their activities and grade level, were collected. Major files and their uses were enumerated. Data processing procedures were recognized. Major data flow patterns were traced. Known problem areas were explored.

This groundwork in the functional area was required as: (1) a foundation of knowledge on which more detailed study could be based; (2) a unique identification of the applications to be considered for further study; and (3) a means of lending direction to the over-all study task, i.e., time and effort required for each sub-task now could be estimated.

A data processing application was defined as "a sequence of procedural steps which may logically be grouped as an entity." Examples from the Civil Engineering functional area include: Work Order Processing, Maintenance Costing, and Real Property Records. Examples from the Transportation functional area are: Operation of the Motor Pool and Ground Vehicle Maintenance. Some applications, of course, will lose their identity in the total system to be designed; however, this definition provides a convenient frame of reference.

To determine the scope of work, applications to be studied must be selected. To place a reasonable limitation on the number of procedures to be documented and analyzed in depth, an application was defined as "primary" if it met one or more of the following criteria:

- Costly, in terms of dollars, manpower, and/or time.
- Mechanized at any base or command within the Air Force.
- Readily adaptable to mechanization.
- Of a priority nature.

"Secondary" applications were those which could not meet the above criteria. Seventy-five unique primary applications were defined. Sixty primary applications which included the majority of data processing in the seven major functional areas, were selected for detailed documentation. Although secondary applications were not documented in detail, they were identified and considered in the conceptual design of the new system.

Initially, data was collected in depth for one sample application in each functional area. Upon completion of this assignment, the study team easily determined additional work prerequisite to completing the data collection and recording phase. Since a punched-card technique was planned, these pilot applications provided information required to define the size of field lengths on the cards. Other minor changes were made at this time. Attention was again paid to system vocabulary. Even the common word "file" had to be defined. Its usual meaning was expanded to include memory of many types, i.e., wall charts and even informal notes. Codings required for eventual system data management were established. For example, blocks of numbers were associated with each functional area and a unique three-digit code was associated with each identifiable organizational unit on the base. The origin and destination of all data recorded and communicated was thus uniquely identified. The pilot application determined the anticipated time required to carry out the over-all study. A master schedule was established against which progress could be measured.

Informal On-Site Recording

The systems analyst studying each application interviewed individuals actually performing data processing tasks and commanders concerned with evaluating and using results generated. The analyst carried out the following steps:

1. Prepared a summary of the nature and scope of the application in narrative form.
2. Determined and recorded the tasks involved. This included forms used, regulations involved, reports required, input and output instruments, manpower used, types of files, etc.
3. Determined the data flow. This included data origins and destinations,

input and output volumes, types of transactions, processing, inquiry requirements, coordination required with other activities, etc.

4. Prepared a functional organizational chart to orient the application within the functional area and base.

In addition to the notes taken, the analyst obtained copies of each document coming into application area, of the work sheets and files used, and of those documents being sent out of the area. For system study purposes, a file was defined as "information stored for possible future use." This definition encompassed data that might be eventually needed in a computer memory.

Copies of all documents were filled out with information to illustrate dynamic data actually required and for a later recording and interpretation. When samples of files were not obtainable (e.g., wall charts), a copy was made for later interpretation.

An informal flow chart of the application was drawn. This chart was not made at the programming level, but rather was aimed at establishing an understanding of the operation. The chart provided a convenient communication media between the analyst and those performing the data processing, and, with the sample documents collected, provided the basic source information for the formal documentation of the present system.

The problem of completely comprehensible communication among team members and between the team and outside personnel arose at the start of the study and continued, but in diminishing degrees, until its completion. This problem was met by evolving an informal dictionary of terms pertinent to the study effort.

Another facet of the problem of communication was encountered in expressing in a definitive manner the names of fields of data in current use within the base's data processing systems. One solution lies in the construction of a "system dictionary." This approach, however, required an effort disproportionate to the results obtained. The problem was successfully solved by assuring adequate intra-team cross-talk and periodic reviews.

It is advisable to obtain standardization among team members in: (1) system approach, (2) system analysis, and (3) written expression in system documentation. Such standardization, of course, must be restricted to the extent that the individual talents of team members are not unduly subjugated. The problem in system approach was typified in obtaining from team members a standard recognition of data processing applications. The placing of boundaries around groups of data processes having meaningful beginnings and endings, and of reasonably equal degrees of complexity and magnitude, was obtained only by discussion between the analyst and team management. The problem of standardization in system analysis became apparent when the rough flow charts were first examined. Essentially, it was a matter of arranging the same degree of analytical depth and amount of detail in all areas. This arrangement was generally obtained by administrative review and suggestion. The problem of obtaining reasonably standard written expression in system documentation became apparent during the formal documentation activity. Interpretation of format field names and use of columns varied with each team member. Written instructions with examples of completed forms and verbal explanations gave satisfactory results.

Formal Documentation Forms

The informal flow chart prepared for each application provided an understanding of the over-all intent of that portion of the system. It lacked the ability to present the detailed data. A method of documentation was developed which made it possible to prepare by machine both flow charts and the statistical compilations required for systems analysis. This section describes the forms used for recording the data collected.

The study technique will be illustrated by carrying one application through each stage of systems analysis. The application used for illustration is the "Operation of the Base Motor Pool." This application has been chosen because:

1. It can be examined without requiring any detailed knowledge of more specialized military operational type application areas.
2. The clerical functions performed typify manual data processing operations which take place thousands of times daily in many data processing complexes.

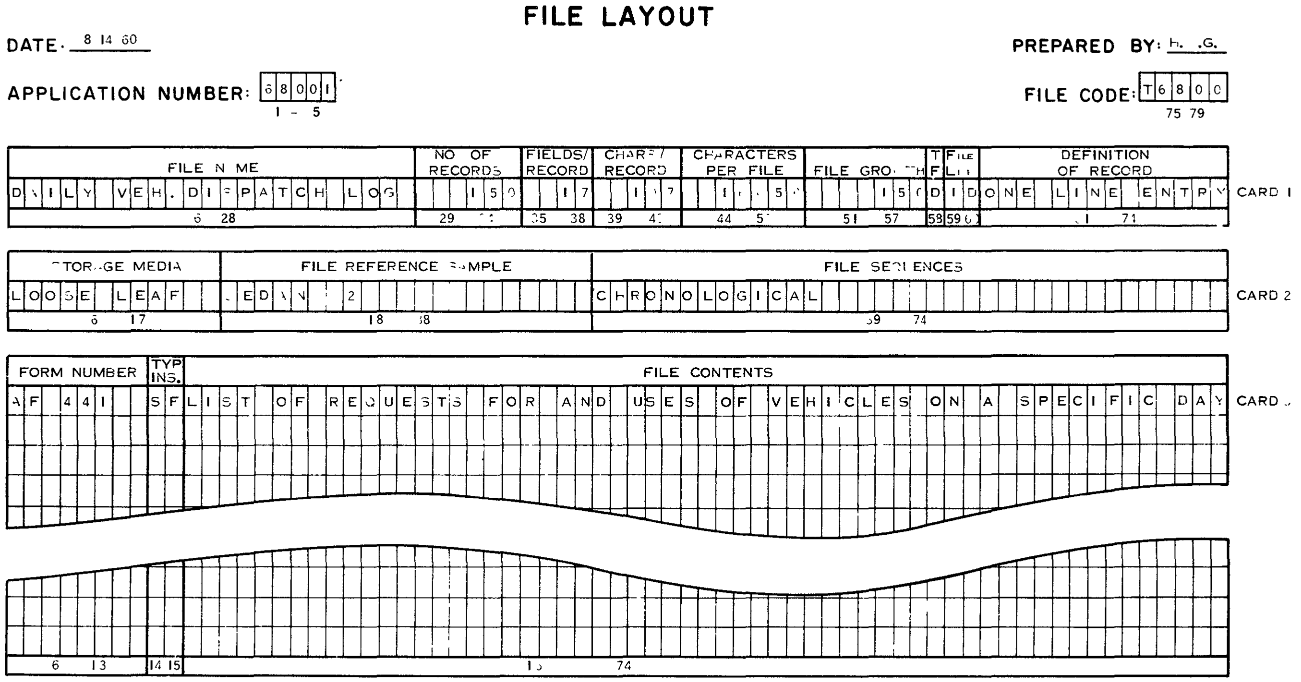
The motor pool has the responsibility of providing ground motor vehicle transportation support to the base and to the aircraft wings operating from the base. The transportation squadron receives requests for use and dispatches both general purpose personnel and cargo carriers and special purpose materials handling equipment. The base motor pool controls utilization and maintenance, and prepares use and cost records and other reports required by Air Force regulations. The documentation forms used will be described by considering basically the single administrative function of receiving a request for the use of a vehicle and dispatching that vehicle. In all illustrations in this paper fictitious data has been used.

File Layout (Figure 1)

The motor pool dispatcher maintains a daily vehicle dispatch log. This log serves as a record of vehicles currently in use. Since information on this log is changed as requests are made and fulfilled, the log is considered to be a file.

The application number is recorded in the upper left portion of the form. In this case the code 680 has been assigned to the base motor pool. (A unique three-digit code was associated with each organizational element on the base.) A two-digit suffix (01) indicates this is the first data processing application associated with organization 680. In the upper right portion of the form, a code of T 6800 has been assigned to this file. For convenience, the first two digits of the file code were chosen to correspond to the code identifying the application with which the file is associated. The last two digits were used to sequentially number files associated with the application. The files associated with the operation of the motor pool were numbered consecutively from 6800 through 6807. A one position prefix is added to the basic file code, an "A" or a "T". The letter "A" refers to an actual (in the mind of the user) file. A "T" refers to a "technical" file that is a form, a wall chart or some other media which, for the convenience of the analyst, has been considered to be a file. In all cards Col. 80 is used to identify card number.

Card 1 is used to record the basic quantative information associated with the file. In this case there are 150 records in the file, 17 fields per record, etc. When the number of characters, fields and/or records vary, average figures are used. Recording a specific number of records would appear to imply a preconceived notion regarding the design of the new system. This is not the case. Considering the dispatch log as a file, an arbitrary decision was made that each line entry should be considered a "record."



FILE LAYOUT

DATE: 8 14 60

PREPARED BY: H. G.

APPLICATION NUMBER: 68001 (1 - 5)

FILE CODE: T6800 (75 79)

FILE NAME	NO OF RECORDS	FIELDS/ RECORD	CHAR/ RECORD	CHARACTERS PER FILE	FILE GROWTH	TF	FILE LIFE	DEFINITION OF RECORD	
DAILY VEH. DISPATCH LOG	150	17	117	17,550	150	D	1D	ONE LINE ENTRY	CARD 1
6 28	29 34	35 38	39 43	44 50	51 57	58	59 60	61 71	

STORAGE MEDIA	FILE REFERENCE SAMPLE	FILE SEQUENCES	
LOOSE LEAF	.EDAN 2	CHRONOLOGICAL	CARD 2
6 17	18 38	39 74	

FORM NUMBER	TYP INS.	FILE CONTENTS	
AF 441	SF	LIST OF REQUESTS FOR AND USES OF VEHICLES ON A SPECIFIC DAY	CARD 3
6 13	14 15	16 74	

Figure 1. File Layout.

Since a line entry is manually posted to the log each time a request is made, this appears to be a reasonable definition. With all related items being recorded in a logically consistent manner, it is then possible (in the systems design phase) to consider a record as a smaller or larger increment of the dispatch log. File growth is 150 records per day (Col. 51-58). The life of the file is one day (Col. 59-60), since a new dispatch log is started every 24 hours.

Card 2 is used to record basic qualitative information relative to the file.

Card 3 is used to describe the contents of the file. The type of instrument (Col. 14-15) refers to the instrument conveying the data to the file and was indicated by a two-digit code as follows:

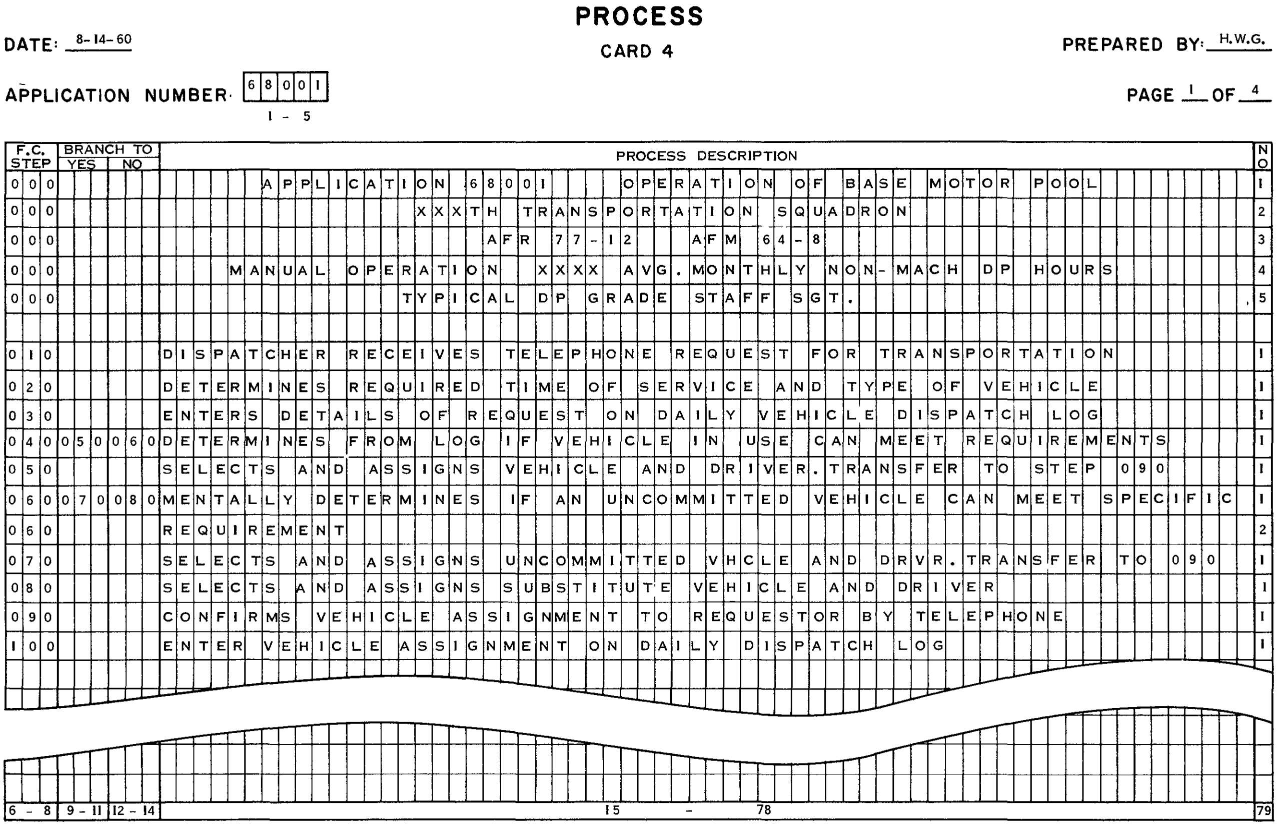
MF	Manual Form
PC	Punched Card
SF	Standard Form
TC	Telephone Call
TT	Teletype Message
RM	Radio Message
VE	Verbal
PG	Process Generated
CK	Common Knowledge

Process (Figure 2)

Card 4 is used to record a brief description of each logical step of data processing as it occurs in the application. Columns 6-8 are used to record the flow chart step number (taken from the original informal flow chart). The ten steps associated with the handling of a request for transportation by the dispatcher are shown in this illustration. Branching (Col. 9-11 and Col. 12-14) indicates appropriate flow chart steps (within the application) to reflect procedural decisions affecting the basic application flow. Col. 79 (No.) is used to number consecutively these cards within a given flow chart step. In addition to recording each process step, these cards also are used to identify the application, organizational element involved, appropriate regulations, non-machine data processing hours expended, and the typical grade level of persons doing the data processing.

Input/Output Transactions (Figure 3)

Card 5 is used to record inputs to and outputs from the application. In addition,



PROCESS

CARD 4

DATE: 8-14-60

PREPARED BY: H.W.G.

APPLICATION NUMBER 6 8 0 0 1
1 - 5

PAGE 1 OF 4

F.C. STEP	BRANCH TO YES	BRANCH TO NO	PROCESS DESCRIPTION	NO
000			APPLICATION 68001 OPERATION OF BASE MOTOR POOL	1
000			XXXTH TRANSPORTATION SQUADRON	2
000			AFR 77-12 AFM 64-8	3
000			MANUAL OPERATION XXXX AVG. MONTHLY NON-MACH DP HOURS	4
000			TYPICAL DP GRADE STAFF SGT.	5
010			DISPATCHER RECEIVES TELEPHONE REQUEST FOR TRANSPORTATION	1
020			DETERMINES REQUIRED TIME OF SERVICE AND TYPE OF VEHICLE	1
030			ENTERS DETAILS OF REQUEST ON DAILY VEHICLE DISPATCH LOG	1
040	050	060	DETERMINES FROM LOG IF VEHICLE IN USE CAN MEET REQUIREMENTS	1
050			SELECTS AND ASSIGNS VEHICLE AND DRIVER. TRANSFER TO STEP 090	1
060	070	080	MENTALLY DETERMINES IF AN UNCOMMITTED VEHICLE CAN MEET SPECIFIC	1
060			REQUIREMENT	2
070			SELECTS AND ASSIGNS UNCOMMITTED VHCLE AND DRVR. TRANSFER TO 090	1
080			SELECTS AND ASSIGNS SUBSTITUTE VEHICLE AND DRIVER	1
090			CONFIRMS VEHICLE ASSIGNMENT TO REQUESTOR BY TELEPHONE	1
100			ENTER VEHICLE ASSIGNMENT ON DAILY DISPATCH LOG	1
6 - 8	9 - 11	12 - 14	15 - 78	79

Figure 2. Process, Card 4.

internal inputs and outputs in the sense of data entering or leaving a file are recorded. In this illustration (in step 010) a request for a vehicle is shown being received. Such a request cannot be associated with a specific organizational element (in the sense that 680 was associated with the base motor pool), since it may come from anyone. For such cases the code 999 with suffix XX (Col. 50-54) was used. Step 090 illustrates an output in the form of the dispatcher verbally (Col. 42-43) notifying the requestor of the vehicle assignment. Steps 030, 040, 050, 070, 080, and 100 illustrate operations upon files. In step 030 a request for vehicle causes file T 6800 (described earlier) to be updated (Col. 72-74). In step 040, information regarding the availability of the specific type of vehicle required is extracted from the same file. Volumes associated with each of these operations are recorded and the number of fields associated with each input and output are divided into identifier and dynamic groupings. Identifier fields are those which identify the input or output record while dynamic fields make up the remainder of the record. The utility of this grouping will be shown later. The functions of inputs and outputs relative to files (Col. 72-74) were coded as follows:

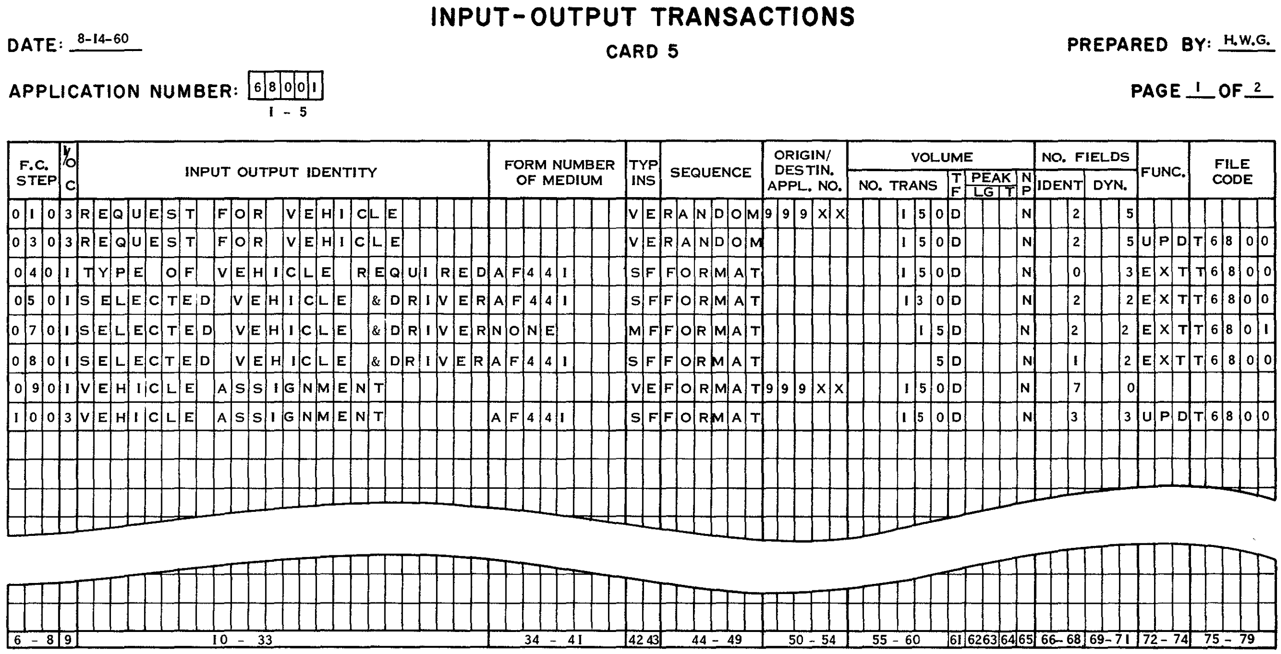
UPD - Update	INT - Initiate
EXT - Extract	CLO - Close
COM - Compare	

Column 9 (input-output code) was used to show whether the data refers to an input or output and whether an original or a copy: (1) output - original, (2) output - copy, (3) input - original, and (4) input - copy. A separate line entry was used on the form for each. A name was given to each input and output. In some cases this name was the document name if the entire document was used. In other cases only certain fields of data were used and this was stated. If the document had an official number it was so identified (Col. 34-41).

Input/Output Fields (Figure 4)

Card 6 was used to list each field name, the number of characters within a field, and the flow chart steps in which that field was used as an input or output. In addition, fields were segregated into two classes: identifier and dynamic (Col. 37). Also recorded was whether the field was numeric or alphanumeric in nature (Col. 38). Column 39 was used to indicate those cases in which the information was process generated. An example would be Total Hours (which had been accumulated manually). The data recorded, was the data actually used in each process step, as opposed to all data following past that step.

By the time formal documentation had been completed the study team was sufficiently



INPUT-OUTPUT TRANSACTIONS

CARD 5

DATE: 8-14-60

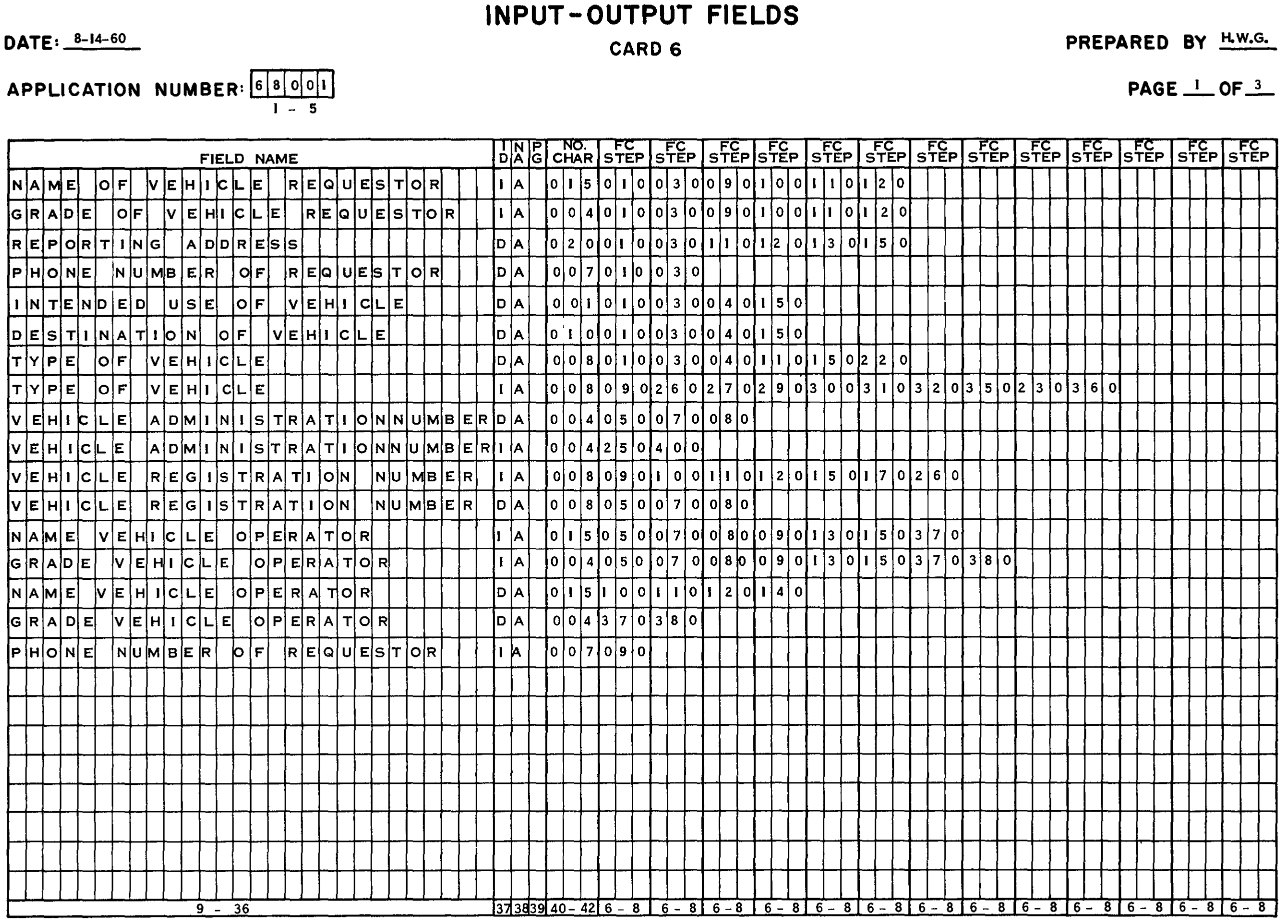
PREPARED BY: H.W.G.

APPLICATION NUMBER: 68001
1 - 5

PAGE 1 OF 2

F.C. STEP	I/O C	INPUT OUTPUT IDENTITY	FORM NUMBER OF MEDIUM	TYP INS	SEQUENCE	ORIGIN/ DESTIN. APPL. NO.	VOLUME					NO. FIELDS		FUNC.	FILE CODE
							NO. TRANS	T F	PEAK LG	PEAK T	N P	IDENT	DYN.		
010	3	REQUEST FOR VEHICLE		VE	RANDOM	999XX	150	D			N	2	5		
030	3	REQUEST FOR VEHICLE		VE	RANDOM		150	D			N	2	5	UPD	T6800
040	1	TYPE OF VEHICLE REQUIRED	AF441	SF	FORMAT		150	D			N	0	3	EXT	T6800
050	1	SELECTED VEHICLE &DRIVER	AF441	SF	FORMAT		130	D			N	2	2	EXT	T6800
070	1	SELECTED VEHICLE &DRIVER	NONE	MF	FORMAT		15	D			N	2	2	EXT	T6801
080	1	SELECTED VEHICLE &DRIVER	AF441	SF	FORMAT		5	D			N	1	2	EXT	T6800
090	1	VEHICLE ASSIGNMENT		VE	FORMAT	999XX	150	D			N	7	0		
100	3	VEHICLE ASSIGNMENT	AF441	SF	FORMAT		150	D			N	3	3	UPD	T6800
6 - 8	9	10 - 33	34 - 41	42 43	44 - 49	50 - 54	55 - 60	61	62 63	64	65	66-68	69-71	72-74	75 - 79

Figure 3. Input/Output Transactions, Card 5.



INPUT-OUTPUT FIELDS

CARD 6

DATE: 8-14-60

PREPARED BY H.W.G.

APPLICATION NUMBER: 6 8 0 0 1
1 - 5

PAGE 1 OF 3

FIELD NAME	I D	N A	P G	NO. CHAR	FC STEP	FC STEP	FC STEP	FC STEP	FC STEP	FC STEP	FC STEP	FC STEP	FC STEP	FC STEP	FC STEP	FC STEP	FC STEP
NAME OF VEHICLE REQUESTOR	I	A		015	010	030	090	100	110	120							
GRADE OF VEHICLE REQUESTOR	I	A		004	010	030	090	100	110	120							
REPORTING ADDRESS	D	A		020	010	030	110	120	130	150							
PHONE NUMBER OF REQUESTOR	D	A		007	010	030											
INTENDED USE OF VEHICLE	D	A		001	010	030	040	150									
DESTINATION OF VEHICLE	D	A		010	010	030	040	150									
TYPE OF VEHICLE	D	A		008	010	030	040	110	150	220							
TYPE OF VEHICLE	I	A		008	090	260	270	290	300	310	320	350	230	360			
VEHICLE ADMINISTRATIONNUMBER	D	A		004	050	070	080										
VEHICLE ADMINISTRATIONNUMBER	I	A		004	250	400											
VEHICLE REGISTRATION NUMBER	I	A		008	090	100	110	120	150	170	260						
VEHICLE REGISTRATION NUMBER	D	A		008	050	070	080										
NAME VEHICLE OPERATOR	I	A		015	050	070	080	090	130	150	370						
GRADE VEHICLE OPERATOR	I	A		004	050	070	080	090	130	150	370	380					
NAME VEHICLE OPERATOR	D	A		015	100	110	120	140									
GRADE VEHICLE OPERATOR	D	A		004	370	380											
PHONE NUMBER OF REQUESTOR	I	A		007	090												
9 - 36	37	38	39	40-42	6 - 8	6 - 8	6 - 8	6 - 8	6 - 8	6 - 8	6 - 8	6 - 8	6 - 8	6 - 8	6 - 8	6 - 8	6 - 8

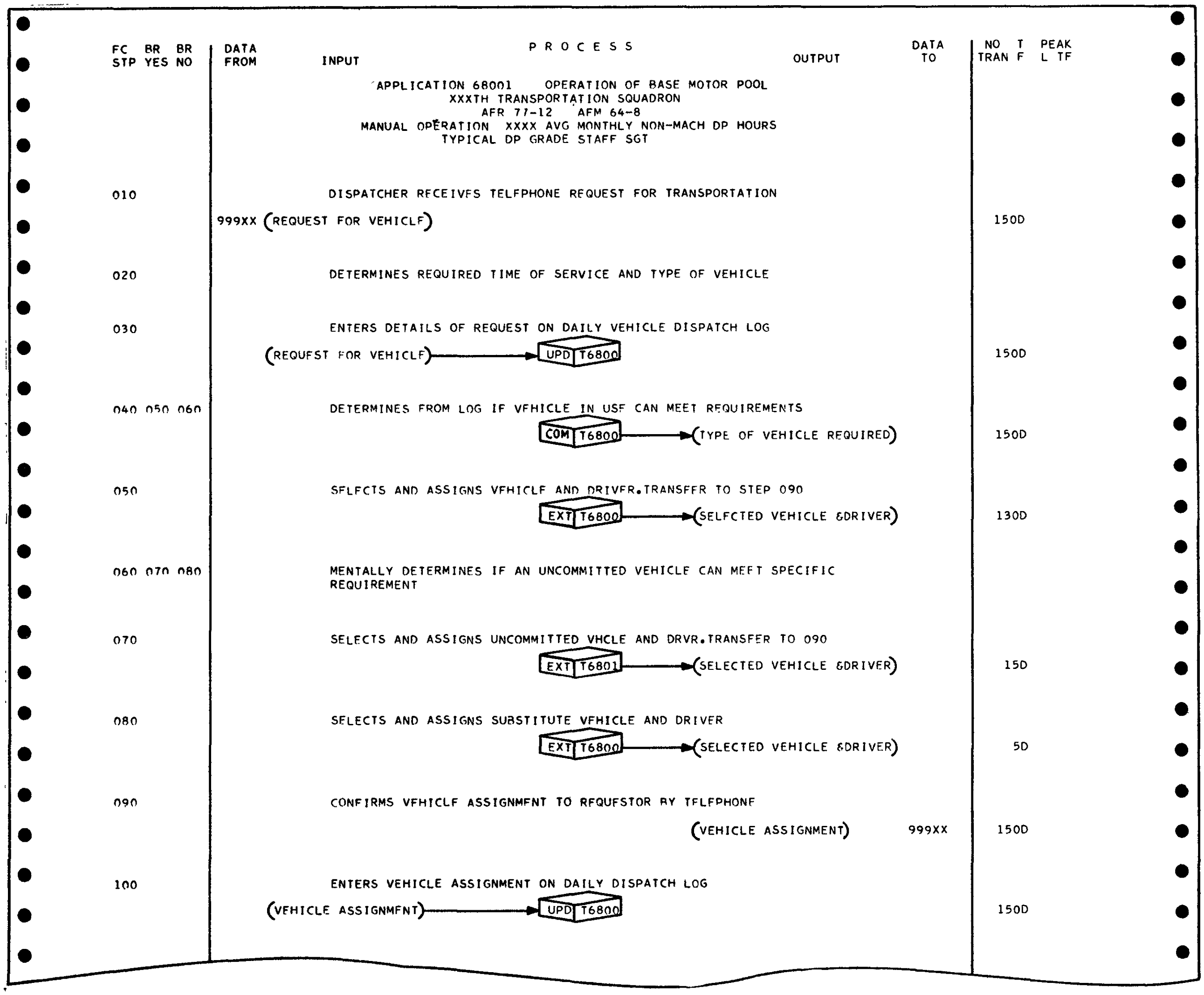
Figure 4. Input/Output Fields, Card 6.

knowledgeable regarding today's system that many short-range improvements became obvious, and appropriate recommendations were made.

ANALYSIS OF EXISTING SYSTEMS (PHASE 2)

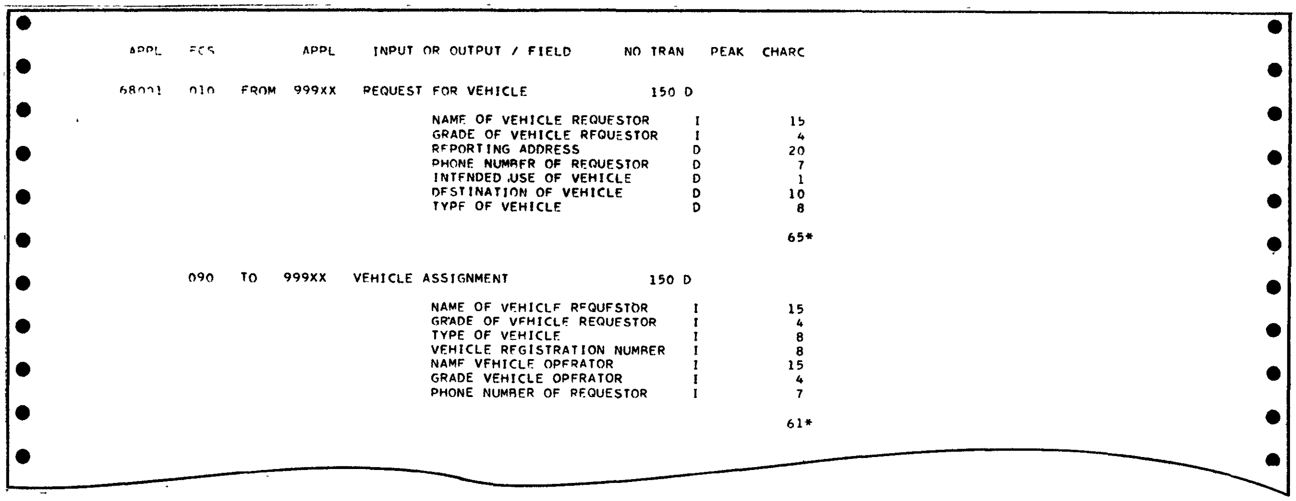
Having all information regarding the current system on punched cards, a wide selection of analyses was possible. The basic analysis is made from the sequential listing of process steps as in flow charts. By simply merging the process cards (No. 4) and input/output transaction cards (No. 5) on flow chart step number, a flow chart may be prepared by the IBM 407 Accounting Machine. The flow chart prepared for the first part of the motor pool operation is shown in Figure 5. The listing prepared by the 407 is completely satisfactory for use in a systems analysis. To make the listing look more like a conventional flow chart (to facilitate communication), a plastic template was used to draw file boxes and input-output arrows.

One example will be given to illustrate the type of analyses which it is possible to prepare from the punched cards. In planning an integrated system one of the most important requirements is to obtain a clear picture of the data traffic flow in and out of applications and between applications. Consider the input-output fields (No. 6) cards. For each distinct field a card was punched to correspond to each flow chart step in which that field was used. These cards were then sorted by flow chart step number and merged with the input-output transaction (No. 5) cards by flow chart step number. A listing of the resulting card deck gives a clear picture of the traffic flow related to the application being considered. A listing of the flow associated with steps 010 and 090 in the motor pool application is shown in Figure 6. To save space the flow associated with traffic internal to the application has been omitted from this



```
FC  BR  BR  | DATA                         P R O C E S S                        DATA  | NO   T  PEAK
STP YES NO  | FROM      INPUT                                          OUTPUT     TO   | TRAN F  L TF
            |          APPLICATION 68001    OPERATION OF BASE MOTOR POOL                |
            |                  XXXTH TRANSPORTATION SQUADRON                            |
            |                       AFR 77-12   AFM 64-8                                |
            |        MANUAL OPERATION  XXXX AVG MONTHLY NON-MACH DP HOURS               |
            |                   TYPICAL DP GRADE STAFF SGT                              |
            |                                                                           |
010         |           DISPATCHER RFCEIVFS TELFPHONE REQUEST FOR TRANSPORTATION        |
            | 999XX (REQUEST FOR VEHICLF)                                               |  150D
            |                                                                           |
020         |           DETERMINES REQUIRED TIME OF SERVICE AND TYPE OF VEHICLE         |
            |                                                                           |
030         |           ENTERS DETAILS OF REQUEST ON DAILY VEHICLE DISPATCH LOG         |
            |    (REQUFST FOR VEHICLF)----------->[UPD|T6800]                           |  150D
            |                                                                           |
040 050 060 |           DETERMINES FROM LOG IF VFHICLE IN USF CAN MEET REQUIREMENTS     |
            |                                [COM|T6800]------->(TYPE OF VEHICLE REQUIRED) |  150D
            |                                                                           |
050         |           SFLFCTS AND ASSIGNS VFHICLF AND DRIVFR.TRANSFFR TO STEP 090     |
            |                                [EXT|T6800]------->(SELFCTED VEHICLE &DRIVER) |  130D
            |                                                                           |
060 070 080 |           MENTALLY DETERMINES IF AN UNCOMMITTED VEHICLE CAN MEFT SPECIFIC |
            |           REQUIREMENT                                                     |
            |                                                                           |
070         |           SELECTS AND ASSIGNS UNCOMMITTED VHCLE AND DRVR.TRANSFER TO 090  |
            |                                [EXT|T6801]------->(SELECTED VEHICLE &DRIVER) |  15D
            |                                                                           |
080         |           SELECTS AND ASSIGNS SUBSTITUTE VFHICLE AND DRIVER               |
            |                                [EXT|T6800]------->(SELECTED VEHICLE &DRIVER) |  5D
            |                                                                           |
090         |           CONFIRMS VFHICLF ASSIGNMFNT TO RFQUFSTOR RY TFLFPHONF           |
            |                                          (VEHICLE ASSIGNMENT)     999XX   |  150D
            |                                                                           |
100         |           ENTERS VEHICLE ASSIGNMENT ON DAILY DISPATCH LOG                 |
            |    (VFHICLE ASSIGNMFNT)----------->[UPD|T6800]                            |  150D
```

Figure 5. Base Motor Pool Flow Chart.



```
   APPL   FCS          APPL    INPUT OR OUTPUT / FIELD       NO TRAN   PEAK  CHARC

  68001   010   FROM   999XX   PEQUEST FOR VEHICLE                150 D

                                   NAME OF VEHICLE REQUFSTOR       I          15
                                   GRADE OF VEHICLE RFQUESTOR      I           4
                                   RFPORTING ADDRESS               D          20
                                   PHONE NUMRFR OF REQUESTOR       D           7
                                   INTFNDED USE OF VEHICLE         D           1
                                   DFSTINATION OF VEHICLE          D          10
                                   TYPF OF VEHICLE                 D           8

                                                                              65*

          090   TO    999XX   VEHICLE ASSIGNMENT                  150 D

                                   NAME OF VFHICLF RFQUFSTOR       I          15
                                   GRADE OF VFHICLF REQUESTOR      I           4
                                   TYPE OF VEHICLE                 I           8
                                   VEHICLE RFGISTRATION NUMRER     I           8
                                   NAMF VFHICLE OPFRATOR           I          15
                                   GRADE VEHICLE OPFRATOR          I           4
                                   PHONE NUMRER OF RFQUESTOR       I           7

                                                                              61*
```

Figure 6. Data Processing Flow.

illustration. Referring to Figure 6, it is seen that seven fields of information (two of them of an identifier nature, 5 of a dynamic nature) are associated with a request for vehicle. The total data recorded is 65 characters in length. This input occurs 150 times a day. Corresponding information is shown relative to step 090.

Other analyses that could be made include the following:

1. List all input-output fields in the system and each input-output name with which they are associated. This would show multiple use of data elements to help in minimizing redundancies.
2. List input-output fields and applications within which they occur. This shows multiple use of data elements in different applications to assist in minimizing redundancies.
3. Summarize volume data relating to information flow between functional areas, by time frame. This gives a gross picture of the data traffic flow.

TOTAL SYSTEM POSTULATION (PHASE 3)

Using the data developed in the first two phases a broad concept of a total system encompassing all data processing at an air base was developed. The system concept for data processing was designed without regard to existing Air Force base organizational structure. Although the system was confined to a base the nature of inputs originating off-base and the nature of outputs forwarded from the base were taken into consideration. The system concept designed in this phase will serve as a guide to subsequent phases of the study.

It should be emphasized that the concept design phase was undertaken with no preconceived ideas as to system or hardware configuration. Even such fundamental matters as the tentative number of processing units were not determined until the analyses were accomplished.

The efforts expended in this phase can be categorized as follows:

1. Determination of the goals of each functional area and of the base as a whole.
2. Selection of the applications, base wide, that are prerequisite to attainment of these goals ("slicing").
3. Initial design and description of the data processing systems responsive to the requirements of each goal (slice).
4. Adaptation of the data processing of each slice to the base system configuration.
5. Documentation of the total system in broad terms.

The determination of goals was a critical step. Goals were chosen which related to data processing having definite effects on the response of the base to its mission. Mission statements for the base and each functional area were studied. The relative importance of each application to the functional area and to the base was established. Lists were prepared showing the objectives of each application, what is required of each application by the base, what is required of the functional area by the base, and the base-wide location of similar files. Problem areas associated with each application were examined. The goal of the motor pool is to provide ground motor vehicular transport to the base and to the aircraft wings operating from the base.

All data processing activities on the base were now grouped by common endeavor. This categorization of data processing activities inferred a restructuring of the data processing systems from their vertical alignment within functional area into a horizontal alignment (slicing). A slice consists of a set of logically related data processing activities. Slices were chosen to cover areas encompassing major segments of the operation of the base. The purpose of this step is to organize the data processing systems so as to most effectively meet the goals of the base.

In a broad sense any business may be categorized into that which involves people, material, dollars, operations or facilities. As a first approximation to slicing, each goal was associated with one of these categories. Major data flows, problem areas, and governing regulations were studied. The location of and reason for similar files were examined. The mission of the base and the functional areas were re-examined. Applications now apparently pertinent to the slice and the goal were collected. Finally, it must be established that the tentative slice fully responds to the goal.

All data processing applications documented in Phase 1 and all data processing endeavors occurring at base level, but not documented, were examined and assigned for concept postulation to the pertinent slice. Data processing activities, which were not conducted at base level, but which were deemed advisable to encompass if data processing capability were existent, were defined and assigned to the pertinent slices.

The result of this analysis was the categorization of all data processing on the base into five functional groups or slices. The five categories were:

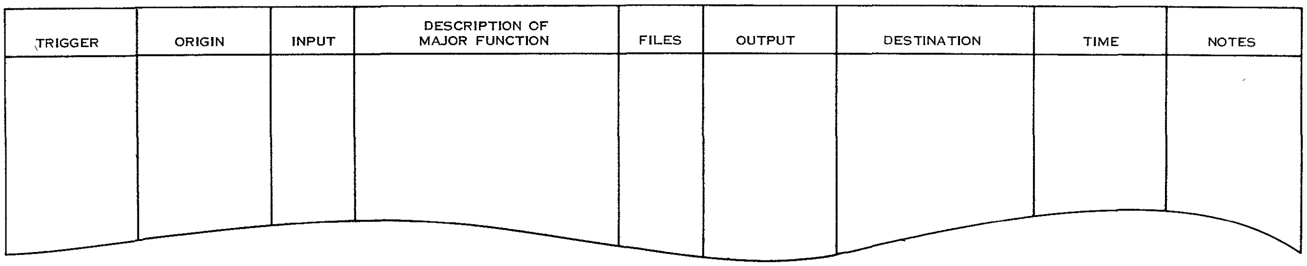
1. Aircraft Management - the data processing necessary to provide information needed to efficiently manage the supply support, maintenance, and scheduling of aircraft to achieve the response required by mission demands.
2. Facilities Management - the data processing necessary to provide information needed to efficiently maintain base facilities and to operate base-wide systems.
3. Personnel Management - the data processing necessary to effect initial and daily job assignments and training of personnel to achieve full utilization of total group capability.
4. Financial Management - the data processing necessary to plan the use, receipt, safeguarding, disbursement, and accounting of public funds.
5. Personal Needs - the data processing necessary to provide the information needed to initiate and sustain services responsive to personal needs of assigned personnel.

The following will illustrate the relation between the old and new data processing structure. The Facilities Management slice encompasses 23 of today's applications. These applications include, for example, Real Property Costing and Real Property Maintenance Records from the present Civil Engineering functional area. The slice also includes Operation of the Base Motor Pool and Ground Vehicular Maintenance from the present Transportation functional area.

After the goals of the data processing system for the base were established and the data processing applications prerequisite to the attainment of each goal were determined, it was necessary to establish and informally document the broad blocks of data processes of each slice. ("Broad block" is a term used to describe an unquantitized number of process steps expressed as an entity.) The development of the broad blocks required that the major data processing functions be sequenced within the slice in such a way as to permit the different analysts, as they worked with the slices, to be able to understand the content of the slice and its relationship to the other slices. It was necessary to identify the inputs, outputs and files of the broad blocks in general terms. The time frame was then added to the broad blocks to identify when information within the broad block would be available or needed.

To insure uniformity in documentation among study team members, a standard format for the informal documentation of each slice was established which could also be used in the documentation of the total system. The major data processing functions were recorded in the form shown in Figure 7.

The development effort of the previous activity resulted in the establishment of an initial data processing system concept responsive to the objectives of each slice, i.e., five subsystems. Inasmuch as the requirement was the establishment of a total base-wide system, the next task undertaken was the establishment of a base-wide systems configuration (with due consideration being given to the processing required by each slice to accomplish the over-all base mission). Here, for the first time in the system development plan, consideration was given to needed hardware capabilities. No effort



TRIGGER	ORIGIN	INPUT	DESCRIPTION OF MAJOR FUNCTION	FILES	OUTPUT	DESTINATION	TIME	NOTES

Figure 7. Data Processing

was, or should have been, made to relate system requirements to specific hardware. Rather, this activity was properly confined to such basic factors as: (1) determination of the number of processors, and (2) whether memory should be associated with each processor separately, or common to all.

The establishment of a tentative base-wide configuration of an automated system was considered a prerequisite to the alignment of the blocks of data processing actions within the total system. Phase 1 documentation provided helpful volume and frequency data. Examination of the initial system concepts documented for the five slices provided qualitative information.

By relating this information it was possible to determine major data flow channels, processing similarities and dissimilarities, similarities in input-outputs, processing time factors, similarities in filed data, and the nature of communication requirements. Study of this information led to the establishment of a tentative configuration of two processors sharing a common memory. The processors and memory would be electronically accessible to specified organizational elements by means of selected communication devices. It is entirely possible that the detailed system design phase would change this configuration. However, the use of the technique is illustrated for this tentative configuration.

These efforts guided the assignment of the processes in each slice to specific processors. The data processing required in the previously defined functional groupings pertaining to Aircraft Management and Facilities Management was assigned to Processor 1. The data processing required by Personnel Management, Financial Management and Personal Needs groups was assigned to Processor 2.

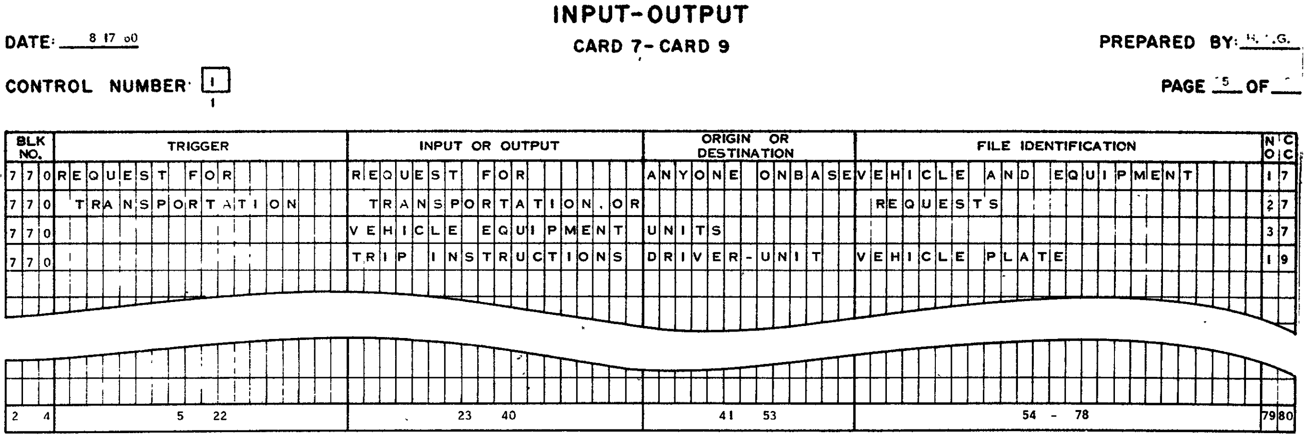
It was then necessary that each processing block contained in the concept be placed in logical order within the processing unit to which it had been assigned. Generators and recipients of data were connected and the concept of system memory, automated and manual, was defined by type, user, contents and mode of access. The concept for system communication of input and output, automated and manual, was defined by type of device (if automated), user, time originated, and mode of communication.

With the system concept developed to the broad block level, it was necessary to document the data in a manageable and standardized format. Again, punched cards were used.

The use of the forms will be demonstrated by illustrating the processing block in the concept solution which includes the motor pool operation associated with the handling of a vehicle request (used in describing Phase 1 documentation).

Input-Output (Figure 8)

Card 7 and Card 9 were used to record input and output information respectively. Control Number designates the processor through which the input or output data is processed. "Trigger" identifies the instrument, situation or condition causing the initiation of action described by the block. Origins of inputs and destinations of outputs (Col. 41-53) were entered as applicable.



INPUT-OUTPUT

CARD 7- CARD 9

DATE: 8 17 60

PREPARED BY: H. G.

CONTROL NUMBER: 1

PAGE 5 OF

BLK NO.	TRIGGER	INPUT OR OUTPUT	ORIGIN OR DESTINATION	FILE IDENTIFICATION	N O	C C
770	REQUEST FOR	REQUEST FOR	ANYONE ONBASE	VEHICLE AND EQUIPMENT	1	7
770	TRANSPORTATION	TRANSPORTATION, OR		REQUESTS	2	7
770		VEHICLE EQUIPMENT	UNITS		3	7
770		TRIP INSTRUCTIONS	DRIVER-UNIT	VEHICLE PLATE	1	9
2 4	5 22	23 40	41 53	54 - 78	79	80

Figure 8. Input/Output, Card 7/Card 9.

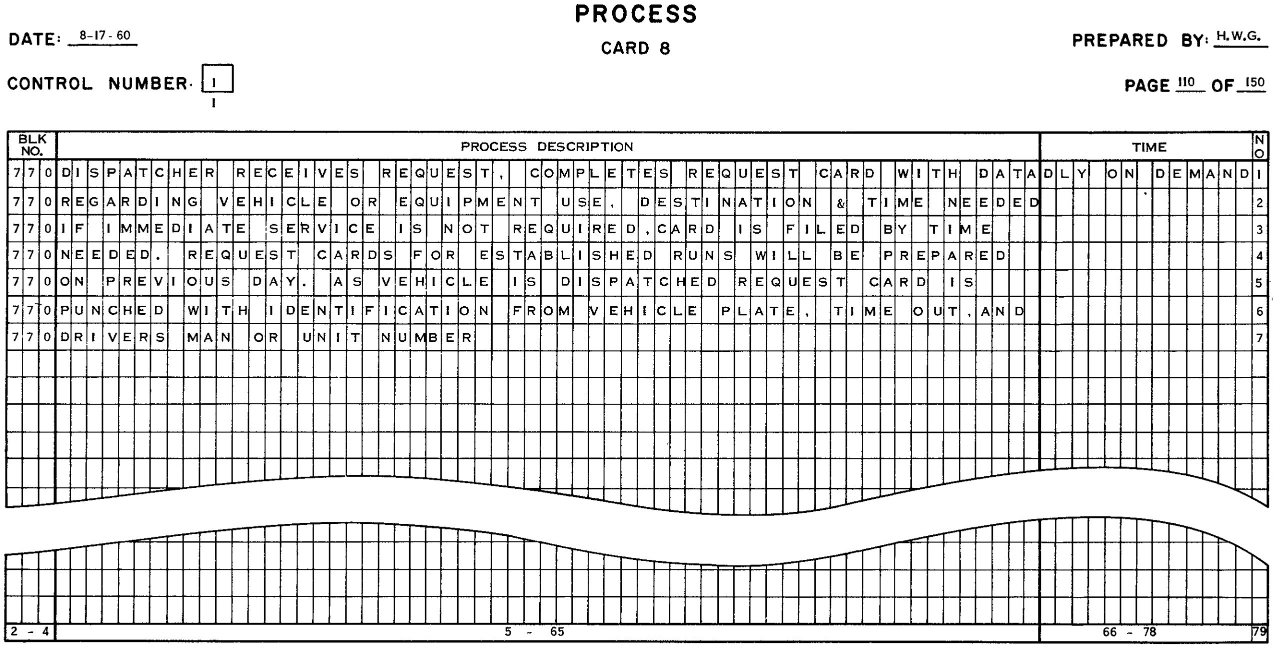
Origins or destinations of data outside of the processor were described by functional names. If data within the processor originated in the previous step or had the subsequent step as its destination, no designator was entered. Movement of data between processors was shown by using the processor designator (Control Number) and the pertinent block number. File identification was used to name all files used with the block. Col. 79 (No.) is used to number consecutively these cards within a given block.

Process (Figure 9)

Card 8 was designed for the entry of data describing the action which takes place in a block. The process description (Col. 5-65) was the major area in which the system concept was reflected. Time (Col. 66-78) refers to either the time within which the process (taking place within the block) must occur, or the interval at which a process was started. Col. 79 (No.) is used to number consecutively these cards within a given block.

The result of merging the above described cards on block number and listing them is shown in Figure 10. This "process listing" serves as the basic documentation of the total system concept.

The system concept was amplified by means of file analyses. These analyses included, for each file, such information as: definition of file contents and expanded statement of file use, identification of the mode of communication between file and user, and a statement of the reason for use of the file.



PROCESS

CARD 8

DATE: 8-17-60

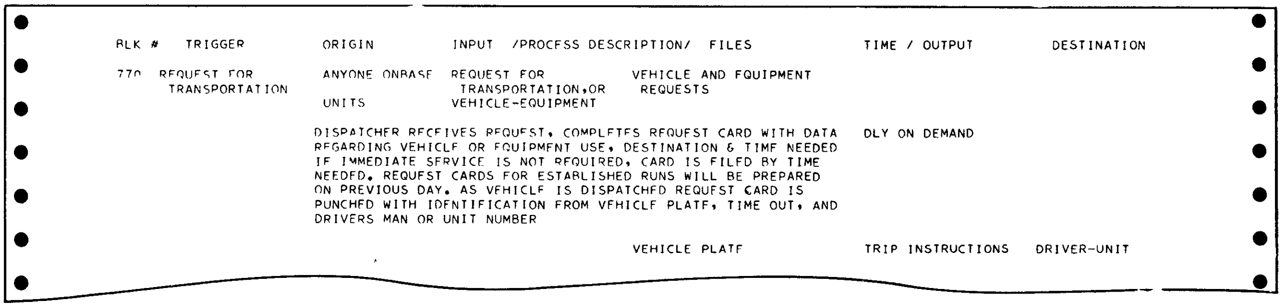
PREPARED BY: H.W.G.

CONTROL NUMBER· 1

PAGE 110 OF 150

BLK NO.	PROCESS DESCRIPTION	TIME	NO
770	DISPATCHER RECEIVES REQUEST, COMPLETES REQUEST CARD WITH DATA	DLY ON DEMAND	1
770	REGARDING VEHICLE OR EQUIPMENT USE, DESTINATION & TIME NEEDED		2
770	IF IMMEDIATE SERVICE IS NOT REQUIRED, CARD IS FILED BY TIME		3
770	NEEDED. REQUEST CARDS FOR ESTABLISHED RUNS WILL BE PREPARED		4
770	ON PREVIOUS DAY. AS VEHICLE IS DISPATCHED REQUEST CARD IS		5
770	PUNCHED WITH IDENTIFICATION FROM VEHICLE PLATE, TIME OUT, AND		6
770	DRIVERS MAN OR UNIT NUMBER		7
2 - 4	5 - 65	66 - 78	79

Figure 9. Process, Card 8.



```
BLK #   TRIGGER           ORIGIN         INPUT  /PROCFSS DESCRIPTION/  FILES            TIME / OUTPUT       DESTINATION

770  RFQUFST FOR          ANYONE ONBASF  REQUEST FOR          VEHICLE AND FQUIPMENT
       TRANSPORTATION                     TRANSPORTATION,OR    REQUESTS
                          UNITS          VEHICLE-EQUIPMENT

                         DISPATCHFR RFCFIVES RFQUFST, COMPLFTFS RFQUFST CARD WITH DATA  DLY ON DEMAND
                         RFGARDING VEHICLF OR FQUIPMFNT USE, DESTINATION & TIMF NEEDED
                         IF IMMEDIATE SFRVICE IS NOT RFQUIRED, CARD IS FILFD BY TIME
                         NEEDFD. REQUFST CARDS FOR ESTABLISHED RUNS WILL BE PREPARED
                         ON PREVIOUS DAY. AS VFHICLF IS DISPATCHFD REQUFST CARD IS
                         PUNCHFD WITH IDFNTIFICATION FROM VFHICLF PLATF, TIME OUT, AND
                         DRIVERS MAN OR UNIT NUMBER

                                               VEHICLE PLATF         TRIP INSTRUCTIONS  DRIVER-UNIT
```

Figure 10. Process Listing.

To complete the picture, each broad block of the total system was analyzed to determine if it contained inputs and/or outputs. If inputs were found, was there a problem in data recording that a special input device could solve? Did the time requirements make it mandatory that information be transmitted automatically or would manual transmission be satisfactory? Finally, were outputs being generated in one processor that needed to be transmitted to the other processor for use as an input? The same type of analysis was performed regarding outputs. In all blocks where the answer to the above was affirmative, the time requirements, use of the device, and estimated traffic flow were used to arrive at a device expressed in broad terms.

REMAINING PHASES

Introduction

While a detailed approach to conducting the remaining phases of a total data processing systems study has not been developed, indications are that the material developed in conducting the phases already described will be of considerable assistance in the remaining effort.

The basic requirement is that of exploding the system concept into a detailed design. This detailed design will include the entire system—that which is to be mechanized and that which should be done manually. Process blocks must be broken into individual steps so that programming can be done for that portion of the system which is to be mechanized, and procedures can be written for that portion which is to be done manually. Input-outputs must be described in terms of individual data elements, which in turn must be sequenced. Likewise, information files must be formatted. Of course, the functional capabilities required of the system must be determined. This in turn gives rise to hardware considerations. The following is an approach to the remaining phases of a total systems study.

Concept Analysis (Phase 4)

To make possible detailed systems design, blocks identified in the process listing prepared in Phase 3 can be grouped by similar end product. This will be referred to as "segmenting" into subsystems. This should not be thought of as regression from the "total" system approach, but rather a convenient grouping of the data processing to facilitate management and control of the design effort. A first approximation to segmentation might be the data processing groupings or slices that were identified in postulating the new system concept.

When segments have been chosen, blocks may be assigned to them. Individual blocks must now be examined. In some cases it will be necessary to divide into sub-blocks. It may be found that certain blocks or sub-blocks are missing, i.e., simply not included in the original concept. After the necessary regrouping, blocks should be aligned within each segment in order of relative influence. Thus, if we label blocks B_1, B_2, B_3, . . . a segment can be shown schematically as in Figure 11. The system at this point consists of a number of these segments. By working from the top down (using the top block as a "control"), changes may be minimized. In the case shown, a subsystem would be laid out for B_{12} first. As subsequent systems are planned for B_3 and B_{17}, they would be "fitted" to agree with the subsystem associated with B_{12}. The corresponding procedure would continue until all blocks in the segment are covered.

It is now possible to prepare Phase 3 type listings by segment. Descriptions of processes, files, and input-outputs can be put in order corresponding to the priority of the blocks within each segment, segregated by segment.

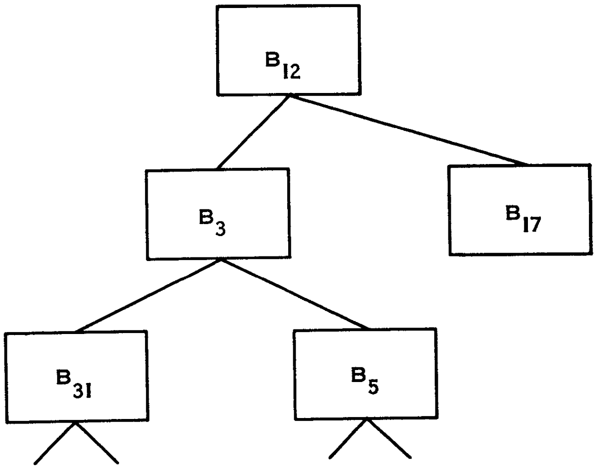


Figure 11. Data Processing Segment.

Detailed System Design (Phase 5)

Consider one block. From Phase 3 the process listing gives the over-all processing associated with that block; the file analyses furnish such basic information as "why," the file, the user, and the type of accessibility envisioned relative to the total system approach, while the input-output analysis gives basic system requirements in that area. Next, study the information collected and developed in Phases 1, 2 and 3. An examination of earlier documentation is necessitated by the fact that the design of a large system may require the efforts of many people, that considerable time may transpire between conduct of phases of the study, and that the analyst doing the detailed system design in most cases will not have done all of the earlier work pertinent to his present endeavor.

Following this initial study, a first pass solution can be developed. Here files, inputs, outputs, and process steps related to this block are determined. In effect, a subsystem for the block is developed. This is the creative part of the effort. No machine or system technique is going to completely perform this task today. However, the work done in Phases 1, 2 and 3 will be of considerable help. The narrative description contained in the process listing of Phase 3 gives a guide to breaking the processing down to the next level of detail. The Phase 3 file analyses give, in addition to general information mentioned above, file sequences which may be used as a first approximation. The input-output analyses, in addition, identify users and note appropriate time frames.

It is in developing a second pass solution that the detailed design phase comes into focus. Process steps now will be broken down to a level of detail from which programming may be done or detailed procedures may be written. Phase 1 listings of existing system input-output record layouts may be used as a starting point in designing records. Once data fields have been selected for inclusion, record formatting may be facilitated by separating cards by "Identifier" and "Dynamic" categories. Of course, in the case of files, file redundancies must be examined for those cases where a file is shared by more than one block.

At this point a subsystem exists for each block. These subsystems must be meshed. The Phase 3 process listing will serve as a guide to putting the subsystems in order. Independently, a determination of the number of processors and the blocks to be associated with each must take place. The process steps for each block to be mechanized are associated with a particular processor in a particular order. All input-outputs between blocks handled by the same processor may be regarded as internal transfers. Information transmitted between processors or between the system and the outside world will be treated as true input-outputs.

Determination of Functional Specifications (Phase 6)

Information from which hardware capability can be derived will have, to a large degree, evolved simultaneously with the detailed systems design. For example, early in design such basic decisions as which information files must be available on a random basis and which can be available serially will have been decided. However, the type and amount of memory remains to be determined. The input-output analyses of Phase 3 define functional requirements in that area. Input-output and processor speeds must be determined. A documentation of the detail system solution somewhat comparable to that used to document the present system (Phase 1) is foreseen. This would provide a vehicle for totaling file references by time frame so as to determine access time requirements, leading toward determination of memory hardware requirements. A similar analysis could be performed with respect to input-outputs. Programming a sample of the processing is probably the safest way to estimate processor requirements.

Other Phases

With the total system designed a logically phased implementation plan may be instituted. Programming and installation may take place along a route directed at the desired end result.

CONCLUSION

Motivation for the total systems approach to data processing is strong. The magnitude of the tasks and the amount of data involved in the design of a total system in many cases requires an automated study technique. The punched-card approach described is a step in the right direction. It has expedited systems analysis by mechanically relating significant data in the analysisphases. Clearly, more sophisticated design analyses can be carried out with a computer approach making it possible to automate a larger part of the total systems study effort. However, much research needs to be done.

ACKNOWLEDGEMENT

A number of persons contributed to the development of the technique described in this paper. In particular, Mr. R. N. Sweeney originated many of the ideas and was a principal contributer to all aspects of the work. Substantial contributions were made by Mr. H. W. Gidley and Mr. C. W. Gooch, III.

BIBLIOGRAPHY

1. Kavanagh, Thomas F., "TABSOL - A Fundamental Concept for Systems Oriented Languages," Proceedings of the 1960 Eastern Joint Computer Conference.
2. Evans, Orren Y., "Advanced Analysis Method for Integrated Electronic Data Processing," IBM General Information Manual No. F20-8047. White Plains, New York.
3. Grad, Burton, "Tabular Form in Decision Logic," Datamation, July 1961. Chicago, Illinois.

DISPLAY SYSTEM DESIGN CONSIDERATIONS

R. T. Loewe and P. Horowitz
Ford Motor Company
Aeronutronic Division
Newport Beach, California

INTRODUCTION

This paper presents several significant design considerations (non-equipment) for computer oriented display systems. It begins with a discussion of display objectives and criteria. Then coding, formats, display request and control, and amounts of information displayed are discussed. Factors related to retinal resolution, display size and detail, group and individual displays and viewing environment are presented. After these miscellaneous considerations, a checklist of primary display characteristics is presented.

DISPLAY OBJECTIVES AND CRITERIA

A thorough understanding of display objectives and criteria is essential to good display system design. Unfortunately, there are usually no quantitative measures of effectiveness for display systems. In fact, it is often difficult to state clear cut qualitative criteria.

A single all-inclusive measure of effectiveness is highly desirable. Such a measure is, "probability of success per dollar," which suits some types of systems. A complicating factor is that display systems are invariably subsystems of larger systems. Display criteria stated in terms of the primary system criteria seldom aid in selecting display alternatives, or in display subsystem evaluation.

Lack of a single all-inclusive objective forces consideration of many display objectives and criteria. Some of the more appropriate ones for displays are described below. They are still subjective. It seems that quantitative criteria cannot be stated except at the level of detailed specifications or characteristics such as: brightness, contrast, resolution, response time, etc.

The following statements attempt to present meaningful objectives of display systems. Not all of the statements apply to all systems and they are definitely redundant, frequently expressing similar thoughts in different manners.

Improve Decision Making

The most general objective of display systems is to improve human decision making. This is difficult to measure because it must be appraised subjectively. Even if decision quality could be measured adequately, it is difficult to separate the influence of the displays from the intellect of the decision maker.

Decision making is used in a general sense here. It ranges from a commander's strategic or tactical decisions to a radar operator's decisions distinguishing targets from noise.

Improve Understanding

The quality of decision-making is based on understanding and insight of the problem.

Thus, a primary display system objective is to improve understanding and insight of the entire problem or situation. The display system should allow the user to assimilate, perceive, comprehend, and apprehend the information available from the balance of the system. The display system should also take the initiative in notifying users of emergencies, exceptions, and urgent information.

Provide Effective Man-Machine Communications

This cliche expresses the major objective well. Displays are generally the primary communication link from the machine to the man. The display system should allow the users to extend their decision making capabilities by use of the memory and processing capabilities of the balance of the system. An ultimate objective might be that of providing a man-machine communication link which approaches telepathic capability.

Clarify Complex Relationships

Another important objective of display systems is to present complex situations in such a manner that significant relationships, conflicts, correlations, and extrapolations can be quickly, clearly, and correctly comprehended. It should be possible to display separate categories of information in a manner that optimizes the user's ability to detect these relationships. Displays should allow details to be understood in context with the whole situation and should allow the whole situation to be examined in detail.

Reduce Reaction Times

The ability to make well founded decisions quickly is an important objective. Availability of proper information in an effective form is necessary for timely decisions. Reduction of staff reaction time is important in: detecting trends, conflicts, and exceptions; responding to requests from top level decision makers; and in planning for various contingencies. Top level decision makers seldom need speed in direct presentation of input data. They generally need speed in the interpretation, evaluation and recommendations based on such input data in order to hasten their decisions.

Improve Coordination

Improved coordination within diverse operations is an objective of display systems. Instead of each working group or staff having different data, which may cause contradictions, conflicts and misunderstandings, each user should have current, identical data. Even within a given display, different data should have consistent "as of" times to reduce confusion.

Improve Availability of Information

Any display system involving data storage and retrieval has a classification and indexing problem. Users must be able to quickly and conveniently identify and request various data categories or combinations of information characteristics. Quick, accurate means which allow the user to obtain the exact information he needs is an important objective. It is often desirable to request displays by logical statements regarding relationships between classification parameters and data characteristics.

Provide Dependability

Dependability includes concepts such as reliability of data and equipment, maintainability, and degree of confidence in the operation and in the data presented. The display system should provide thorough, accurate, dependable data which will minimize human errors and improve the effectiveness of human judgement.

Improve Control

Some display systems provide information necessary for control of a process or system. Such display systems should provide displays that aid personnel in executing improved control and monitoring.

Improve Briefings

Display systems are often used in briefings to appraise key personnel of status, problems, and plans. Improvement in the quality, clarity and duration of briefings is an important objective. A display system should also reduce briefing preparation time and should facilitate better answers to questions during briefings.

Simplify Operations

Simplicity of operation is an important objective in display systems. If equipment is difficult or inconvenient to operate, it will not be used effectively. Training requirements should be minimized and as much interchangeability of operations and functions as possible should be provided.

Provide Compatability With Data Entry

Display systems used in conjunction with data processing systems are sometimes closely related to data entry functions such as; editing, formatting, verifying, correcting, intervention, and interrogation. The displays should provide supporting information and should aid data entry by displacing formats and data being entered.

Provide Flexibility

Flexibility and adaptability are perhaps the most important criterion of display systems. Display requirements often change drastically with time due to:

(1) Changes in commanders (or executives) and staff personnel
(2) Changes in organization and staff functions
(3) Changes in military threats (or commercial markets)
(4) Changes in defensive capabilities (or products)
(5) Changes in information available
(6) Experience from system operation and exercises
(7) Improvements and additions to computer programs
(8) Technological improvements
(9) Modifications in mission or goals

Thus, flexibility to adapt to such changes quickly and effectively is a primary objective of display systems.

Some of the aspects of display systems where flexibility is desired include:

(1) Formats
(2) Content
(3) Symbology and coding
(4) Display control and utilization
(5) Distribution of personnel
(6) Distribution of displays
(7) Growth potential in both quantity and performance

Satisfy Human Factors Requirements

Ultimately, the effectiveness of a display is dependent on the user's perception, which has definite psychological limits. Certain display design parameters must be adhered to in order to effectively utilize human perception. Some of the more salient points which must be considered on information displays are: retinal resolution, brightness, contrast, coding and symbology, critical flicker-fusion ratio, and the general viewing environment. Other considerations such as; human response times, display formats, individual display preferences, and other psychological differences also warrant consideration in the evaluation of display systems.

CODING AND SYMBOLOGY

There are many different ways of representing information for visual interpretation. This discussion is confined to visible markings on a two-dimensional display surface.

The earliest known attempts to represent information in visual form date back to drawings in caves of the Cro-Magnon period. Pictures are a familiar and effective means of representing information. They are visual simulations of the actual object or concept. Egyptian hieroglyphics and Chinese writing have evolved from crude pictures or drawings. Abstract concepts not amenable to pictorial representation were represented by remote pictorial associations.

The Phoenician alphabet and Arabic numerals were perhaps the first attempts to represent information in coded symbolic form where symbol shape has no pictorial value. Coded symbols (letters) are combined into more complex codes (words) which represent real things or abstract concepts.

Although written language is one of man's most indispensable tools of communication, it is not necessarily the simplest or most efficient means of representing thoughts. It is a compromise which limits the number of different symbols required. We still resort to special symbology to facilitate visual communication, particularly where the thought represented has a characteristic pictorial shape that is easy to learn and remember. It is more effective to represent an aircraft by its outline than the combinations of letters "Aircraft." Use of special symbols for abstract thoughts such as "validity," is limited by human learning and memory capabilities.

Evaluation of coding depends upon a number of factors, such as; the physiological limits of human discernment (e.g., the point at which the human sensor cannot reliably discriminate between two brightnesses or colors). Relative compatibility with other coding modalities, relative efficiency of the code (how long does the average operator take to make a positive identification) psychological effect of the code on the operator, (is the code irritating, fatiguing or perhaps hypnotic, as in the case of some flicker rates). These and other considerations must be examined prior to evaluation, selection or use of a particular coding modality.

Some means of coding information are:

(1) color
(2) size
(3) shape (includes symbols and font)
(4) position
(5) orientation
(6) line width (boldness)
(7) number (quantity)
(8) flicker or blink rate
(9) intensity (includes brightness and grey scale)
(10) line length
(11) line type (dotted, dashed, cross-hatched, etc.)

The American Standards Association has suggested the use of particular colors for coding; publication Z53-1-1-1953. This list is quite detailed and complete.

DISPLAY FORMATS

Figure 1 indicates some of the wide variety of formats used in displaying information. Combinations of formats and coding can improve presentations. There are also many ways of displaying three variables in only two dimensions.

In system design presentation formats must be determined after determining what information is to be displayed. These decisions are as important to display system effectiveness as many technical decisions. Coding and symbology capabilities must be considered in selecting formats.

Display systems are often used to compare different activities, coordinate different functions, and detect conflicts. Thus, there is need for determining the best method of simultaneously presenting different types of information in formats which most clearly show interactions, relationships and conflicts.

Certain types of data cannot be presented effectively in alpha-numeric form. Examples are: fallout contours, flight paths, coverage of surveillance devices or weapons, planned maneuvers, meteorological patterns, and areas of responsibility. Line drawings are an effective means of representing such information that is still compatible with computer outputs. Colored or shaded areas are good representations for such information, but are less compatible with computer outputs.

AMOUNT OF INFORMATION IN THE SYSTEM

The amount of information in a system at any one time is a very significant characteristic of display systems. Some systems can display all their information in a single presentation, e.g., radar systems. Other display systems are based upon large data storage and retrieval systems. Whenever there is more information required than can be presented in one display, the information in the system must be divided into categories for updating and display. The division of information into categories is a complex problem involving: file organization, classification and indexing, data entry and processing considerations, and information requirements of the users.

Selection of categories for data entry, file organization, and display is an extremely important part of system design. Organizing information into predetermined categories limits flexibility by hampering the ability to request data other than by the predetermined categories. Defining many categories complicates the system but improves flexibility. Ease in redefining categories is a good way of providing flexibility.

Some systems may have 100 or more different display categories. Manual preparation of different programs for each category would be expensive and inflexible because of the inevitable changes required in the display categories. An approach which develops basic programs for each type of display format has significant advantages. Such programs generate or compile new programs each time a new display is requested. The request message supplies the needed parameters and control information.

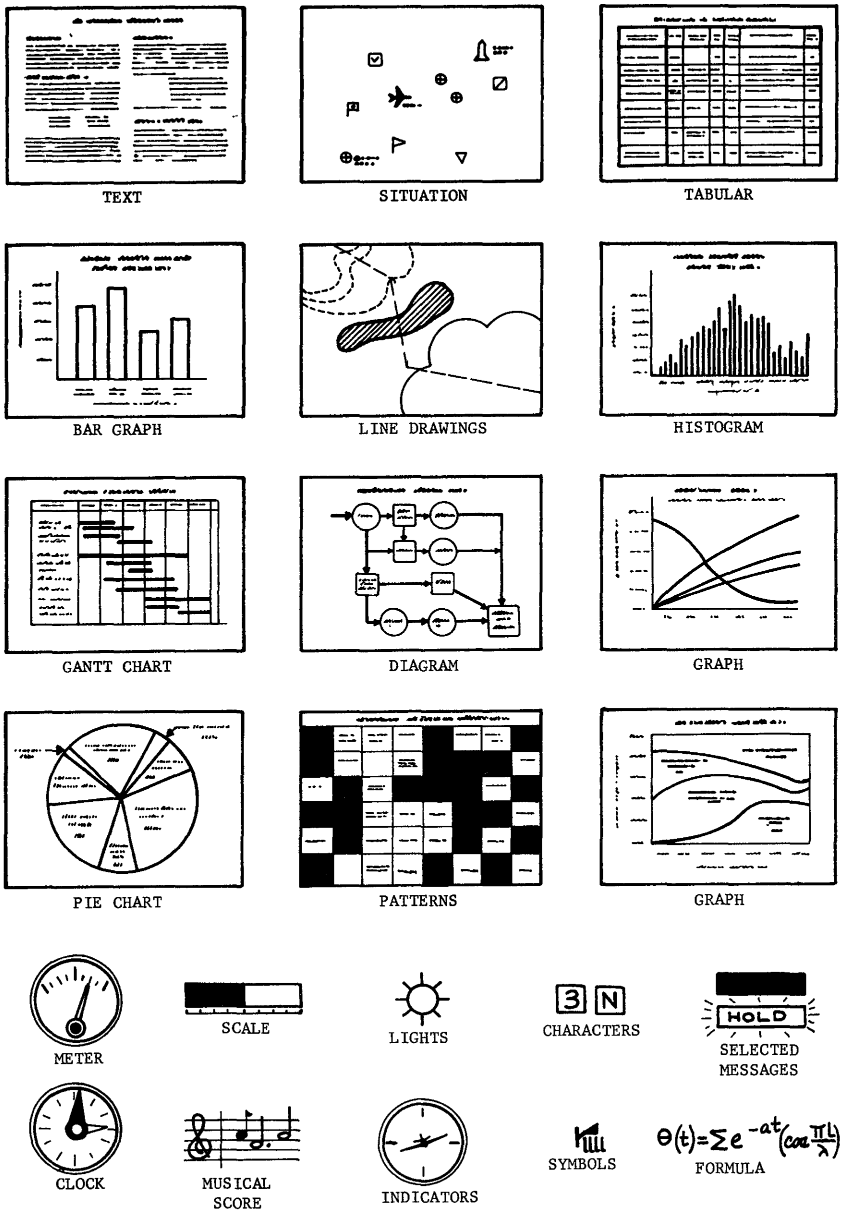


Figure 1. Typical Display Formats.

This approach provides tremendous versatility and adaptability. It may provide users the capability of defining their own display formats and contents without requiring assistance from programming specialists or analysts. This may reduce the amount of analysis of operational requirements preliminary to display programming.

DISPLAY REQUESTS

To change the information categories presented on a display screen, the user must indicate the information he wants displayed (unless the system itself determines what data is to be displayed and when on the basis of previous analysis). A number of different ways of selecting the desired display information are described below.

(1) A keyboard with a separate button for each category to be displayed or removed can be used. This approach is made more flexible by using a single keyboard and changing the meaning of the keys by overlays or other means.

(2) Where there are many categories, they may be selected by a code and keyboard. For example, when decimal numbers are used to indicate display categories, 100 different displays can be identified by two activations of a decimal keyboard.
(3) Displays can be requested by a limited syntax language or by logical statements. In this manner, a user can define the information combinations and formats he desires.
(4) Another method of defining information to be displayed, is by pointing at an item on an existing display. For example, a user requests details of a particular symbolized item by pointing at that symbol on a display.
(5) Searching is another method of obtaining the desired display. The actual display may be sequentially changed with the operator deciding when to stop the sequence of changes.

DISPLAY CONTROL

The display control function includes: routing display information to display units, updating display information, and rules for changing displays.

The computer may store information regarding the routing of display information. It may have a rule which states; "whenever certain data files or display information are updated, certain display units should be notified or given the displays." Security considerations also affect routing of display data.

Display updating rules may also be handled by a computer. Displays may be updated on a real-time basis so that all changes are immediately reflected in the appropriate displays. Displays may also be updated according to any of the following criteria: at given time intervals, after collection of several input messages, each time new information arrives, or when display requests are made.

Priority considerations may be incorporated in the updating rules also. Thus, one updating rule is used for one priority class while a different rule is used for another priority class. Forced displays, too, may be under computer control. If the computer recognizes occurrences which match predetermined forcing rules, selected displays can automatically be changed to show the detected critical information.

It is also possible for certain personnel to force displays on units operated by other personnel, although this is generally an extreme measure. This risks interruption of more important work and is annoying at times. A less extreme approach notifies the user that critical information is ready for viewing. Such notification may also indicate priority or type of information. The user may then accept the new display at his convenience.

AMOUNT OF INFORMATION DISPLAYED

When the information required by an individual to perform his functions exceeds the capacity of one display, there must be means of accessing other displays.

Two primary methods of accessing different displays are given below.

(1) A single display screen is provided, and the information displayed upon it is changed at the command of the user.
(2) All available display information is simultaneously shown in a number of different displays. The user accesses different information by visual searching or memory and then focuses his attention on the desired display; this can be accomplished rapidly for moderate amounts of information. (A variation of this is the use of some magnifying means to access detailed information in a display.)

An important characteristic of any display system is the display access time. This is the time lapse between a user's request and when the display is actually viewed. Shifting the eyes or head presents a convenient and rapid display access time. However, the amount of information available in this manner is limited.

There are several limitations to the amount of information that can be displayed at one time. Visual acuity or resolving power is the ability of the eye to perceive and discriminate precise sensory (visual) impressions. The human eye can, in normal light, discriminate lines that are about 1.5 minutes of arc apart, using parallel black lines separated by intervals equal to the line width. This gives a resolving power of 2,300 line pairs per radian.

The visual angle at the eye is 2 arctan L/2D; the angle subtended at the cornea by a viewed object in which L is the size of the object measured at right angles to line of sight and D is the distance between the eye and the object.

The solid angle about an individual and the resolution of the eye limit the total amount of information that can be displayed simultaneously to any one person. Of course, the complete solid angle around the person cannot be utilized for display purposes, but cockpits, for example, frequently use much of the total solid angle for displays and controls.

Since the amount of information that can be presented in a single display depends primarily on the solid angle subtended and the resolution of the eye; the distance of the display from the user is not significant. This contradicts the popular concept that one must have a big display to see the big picture.

Figure 2 shows the relationship between three interrelated variables: size of display screen, acceptable viewing distance, and the amount of detail or number of characters which can be displayed. This shows that for a given viewing distance, the display size must be increased if the amount of detail displayed is to be increased; the curve holds true both for individual displays and group displays.

Another limitation on the amount of information displayed is the response time required and rate at which information can be assimilated. If a person must respond to certain information very rapidly, he does not have time to comprehend a great amount of detail. Thus, highly summarized information should be presented when rapid decisions are necessary. With enough time, however, a complex and detailed display can be absorbed or understood after thorough study. In lengthy analysis, interpretation, and detail planning functions, considerable information can be presented for study.

Several different displays can be simultaneously presented on the same display unit. For example, three-fourths of a display surface may display a geographic situation with the remaining one-fourth used for tabular or graphical details. Some categories and formats contain little detail in comparison to the detail capability of a display unit. In such cases, several categories and formats can be presented simultaneously in different portions of a common display screen.

GROUP AND INDIVIDUAL DISPLAYS

The value of group displays versus individual displays is a controversial topic. A group display is a display intended for simultaneous viewing by more than one person. This definition is independent of display size; although common use tends to link "large screen" with "group display." This may be due to the fact that most displays intended for individuals can be viewed by more than one person. The value of a group display depends on how the people work together and on their information requirements. A list of advantages of group and individual displays follows.

Advantages of Group Displays

(1) Where information requirements of a number of people are similar or identical, a group display may provide a more economical method for providing a large number of people with the data they need.

(2) A small working group can communicate more efficiently, relevant to a group display; whereas, such personal interchange is less convenient and effective if personnel are situated at separate consoles.

(3) In many cases, the senior person of any group indirectly controls the content of the group display. This allows the support personnel engaged in detail work to know the concern and interests of the top-level decision maker.

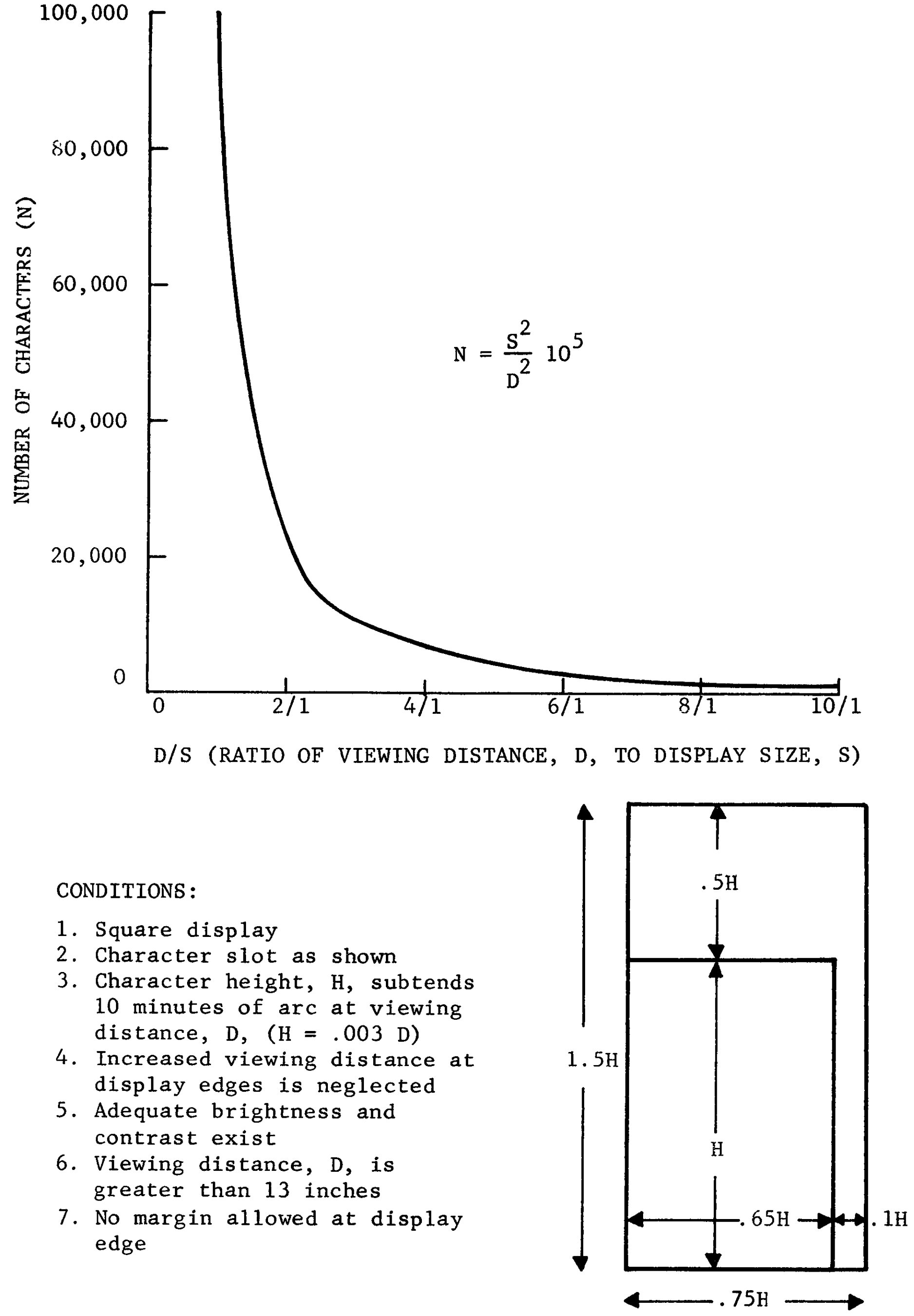
(4) One group display may be more economical than several individual displays.

(5) There may not be enough physical space for several individual displays.

(6) With a group display, coordination and integration may be improved since all concerned are seeing the same information.

(7) Manual backup procedures are more effective for group displays than for individual displays.

(8) Many users have become accustomed to operations based on manual group displays.



CONDITIONS:

1. Square display
2. Character slot as shown
3. Character height, H, subtends 10 minutes of arc at viewing distance, D, (H = .003 D)
4. Increased viewing distance at display edges is neglected
5. Adequate brightness and contrast exist
6. Viewing distance, D, is greater than 13 inches
7. No margin allowed at display edge

Figure 2. Number of Characters That May Be Displayed vs. Ratio of Viewing Distance to Display Size.

Advantages of Individual Displays

(1) Information may be displayed which exactly fits the particular task requirements of an individual, rather than a composite display planned to fit group needs and not optimized for any one specifically.

(2) Individual displays can be changed at will, thereby gaining access to greater amounts of information without interrupting the information displayed to others.

(3) More information can be presented on individual displays than on group displays because the individual can lean over or bend over portions of an individual display to get the detail.

(4) It is possible to display identical data on all individual displays. This approaches the capability of the group display for allowing many people to simultaneously refer to the identical display information.

(5) The deployment of personnel and equipment has more flexibility than group displays.

(6) More diverse capabilities are available with the current state of the art.

GENERAL VIEWING ENVIRONMENT

The environment (conditions and influences, other than display "output," which affect the perception and comprehension, of display information) of both the individual viewer and the total viewing audience is an important input to the overall system design. The more important environmental factors include: ambient lighting, seating, console design, work area configurations, temperature, humidity, and noise level.

Ambient illumination should not exceed the display brightness; although work surfaces should have an illumination of at least 25 ft. candles. The light bandwidth, or color, is dependent on the particular type of display used. The lighting should complement the display, and if possible, be pleasing to the viewing audience.

Seating is designed for the average individual using the equipment. The seats should offer the individual comfortable support for both viewing and other work tasks. The console is designed for optimum viewing at a particular configuration (seating or standing) depending on whether the viewer is to be active for extended or limited time periods.

Temperatures between 68 and 72°F and humidity from 30 to 50 percent allow extended periods of alertness and efficiency from the viewing group.

Noise control is necessary if direct verbal communication between the viewers is warranted. The ambient noise level in a direct communication system should not exceed 60 decibels.

CHECKLIST OF DISPLAY SUBSYSTEM CHARACTERISTICS

Some primary display systems characteristics are listed below. They may be used as a checklist for specifications, requirements, and evaluation. Not all of these topics have been discussed above, but they are all important characteristics which must be considered during the design of a display system. Each display system will also have other considerations specifically related to the particular application.

Amount of Display Information Available
Updating Response Time
Rates of Change of Display Data
Display Access Time
Display Request Rates
Number of Display Units
Display Size
Resolution
Audience Size
Coding
Symbology
Display Formats
Brightness
Ambient Lighting
Contrast
Accuracy
Distortion
Flicker or Blink Time

ABSTRACT SHAPE RECOGNITION BY MACHINE

Mary Elizabeth Stevens
National Bureau of Standards
Washington, D. C.

ABSTRACT

Graphic pattern recognition in the development of total systems for information selection and retrieval is considered. A particular hypothetical machine model for shape recognition is described. Examples are given of recognition of 12 to 20 categories of geometric shapes, by use of a contour-projection principle. Within certain limitations, the model identifies these graphic patterns, regardless of size, location, and certain rotational transformations. It provides means for detecting classes of patterns which would or would not be ambiguous with respect to any of the patterns that are recognizable. Problems in practical application, including possibilities for recognition of constrained handdrawn figures, are considered. Possible means of implementation are also discussed.

1. INTRODUCTION

The storage, search, selection and retrieval of information represent one of today's most challenging areas for research and development. The urgent need to improve the effectiveness with which our total scientific resources (men, machines, methods and findings) are utilized, requires increased ability to avoid replication of old work, to follow up new leads promptly, and in general to cope with the exponential growth of technical literature. Not only machine technology, but also our understanding of mechanizable processes, must be pushed to new limits to achieve information processing systems adequate for these demands.

The prospective integration of many information-handling functions in a single system emphasizes that mechanizable operations must be sought out and identified in all phases of processing. We must determine the full extent to which the machine can replace the man, or supplement his efforts, or serve him as an efficient tool. We should consider, for example which steps can be automated in the preparation of files to be searched by machine, in the content-analysis of incoming material for identification of appropriate selection criteria or "retrieval hooks," and in the framing and renegotiation of search-selection prescriptions.

Because "one picture is worth a thousand words," the graphic material interspersed with natural-language text in the system's input must also receive analysis and storage. Therefore experiments in mechanical recognition of graphical patterns go hand-in-hand with investigation of possible methods for machine processing of natural-language text, and for communication by man, in more or less his own language, with the machine.

A number of possible approaches to mechanical recognition of certain geometric shapes have been proposed. This paper contains a detailed description of a particular method, partly based on the contour-projection technique suggested by Deutsch.[8*] The method will be shown, within restrictions described later, to distinguish among octagons, squares, rectangles, some triangles, and certain other classes of polygons. The procedure can be used (by hand simulation, if desired) to generate large classes of patterns, not in the model's "vocabulary," which would be misidentified as some member of the family; it can also be employed to generate classes of patterns for which no such misidentifications are possible. Thus the procedure can in a sense be used to gain insight into its own limitations. It can be applied to both solid or outline figures, either black-on-white or white-on-black.

Shapes recognizable by the method can usually be identified regardless of variations in size or location in the field. Anomalies and ambiguities are encountered, however, for some shapes at critical sizes or proportions; some illustrations of this will be given. Certain rotated variants and mirror images can be distinguished for some, but not all, of the shapes. In such cases, rotated versions may be distinguished from their originals (e.g., a diamond in contradistinction to a square), or may if desired be given a qualified identification with the originals (e.g., a diamond recognized as a "tilted square"). With minor additions to the recognition or decoding logic, some shapes might be recognized both generically and also more specifically; e.g., a diamond might be identified as "parallelogram," "particular parallelogram-square," and "rotated square."

Examples of distinct shapes which are indistinguishable by this particular procedure will be given, and limitations regarding sloppily drawn figures will be noted. Problems likely to arise in practical application are considered, including questions of instrumentation, resolution and quantization, as well as the complications introduced by multiple figures simultaneously distributed in the input pattern field, or by shapes enclosed within the boundaries of other shapes. Practical implementation appears feasible for "closed vocabularies" of up to at least 70 distinguishable shapes, in the case of suitably constrained handdrawn figures such as chemcial structure and electrical circuit diagrams.

2. RELATED RESEARCH IN PATTERN RECOGNITION

The literature shows considerable work in several areas related to machine recognition of graphic patterns. First, there are alphanumeric character recognition devices, commonly requiring either a specially stylized font or a closely restricted alphabet of characters in one or a few type styles. Under certain restrictive conditions, these devices may prove practicable for symbol-transliteration or input operations on large bodies of text, for either an information selection system or a mechanical translation program. The restrictions typically include a requirement for consistently high quality of the input patterns, and a limitation of the recognizable alphabet to at most several hundred members with each allowable size and style variation of each character counting as a separate member.

Second, various techniques have been developed for machine detection and recognition of classes of input patterns that can be associated with the same desired output. Examples cover many areas of practical interest, from the processing of cytological smears and bubble chamber tracks to the analog representation of the enunciations of various sounds of speech. Studies of alphanumeric character recognition for large alphabets, including hand-printed and hand-drawn alphabets, are especially relevant to the design of an integrated information selection-and-retrieval system.

Third, a number of investigations have been directed toward the explanation, simulation or "black-box modelling" of processes which might underlie perception, recognition, learning, recall and generalization phenomena, especially in lower-level living organisms. Such studies have dealt with hypothetical mechanisms for perception and recognition, for image-improvement, and for translations of variations in size or position into differences in normalized times-of-arrival. They include work on the use of perception-recognition phenomena as demonstrations of "learning" capabilities in

[*] Contour projection techniques for pattern recognition have also been suggested, without detailed discussion, by Selfridge [33] and Minsky [26].

simulated neural networks which "self-organize" from initial random configurations. Examples in the literature of pattern recognition research range from classic hypotheses by Pitts and McCullough [28], Hebb [20], Schade [32], and others, through image improvement mechanisms such as those considered by Kovazny, Arman, and Joseph [25], to computer simulations and working machine models of "learning," "conditioned reflex," or "self-organizing" phenomena, by Uttley [41]; Clark, Dineen, Farley, and Selfridge, [7, 10, 13, 33]; Rosenblatt [31]; Harmon [19]; Roberts [29]; Uhr [39]; Barus [4]; and Baran and Estrin [3], among others.

For size-constained, and usually position-normalized, input patterns there have also been intriguing proposals for recognition by machine of various shapes (especially shapes common to constrained hand-drawn alphanumeric characters) by such investigators as Doyle [11]; Sherman [35]; Johnson [22]; Dimond [9]; Bomba [6]; Grimsdale, et al [18]; Kamentsky [23]; Unger [40]; Taylor [38]; as well as by others, some of whom have not as yet reported their results in the open literature.

Other investigations (e.g., that of Alt [1] on the use of moments for digitalized pattern recognition) have been based on properties of characters or shapes which are largely independent of variations in size and locations, and are relatively invariant under certain rotations, certain minor stylistic differences, and certain amounts of noise. Randomly-generated operators for the extraction of relatively invariant features of graphic patterns, especially alphanumeric character patterns, are considered in detail by such investigators as Bledsoe and Browning [5]; Uhr and Vossler [39]; and Novikoff [27]. There also have been interesting studies of possibilities for explaining shape-discrimination phenomena actually observed in living organisms in terms of potentially machinable processes, especially by Sutherland [37] and Deutsch [8].

There are other ways of classifying past research in pattern recognition which is relevant to the maximal mechanization of the information-processing function. For example, one might ask whether or not a proposed method provides for adaptive modification ("self-organization") with respect to perception-recognition behavior. This will be discussed later. Second, one might ask whether or not template-matching principles are used in reaching an identification decision.

On this last point, we note that template-matching schemes are generally considered to be inadequate for large alphabets whose characters are subject to appreciable differences in size or location in the field, or to substantial rotations. Templates (as the term is used here) may be photographic-negative images of the shapes or characters belonging to the alphabet to be recognized, or they may be of coordinate descriptions of those shade-quantized cells in a superimposed grid which would be black or would be white if the input pattern is a particular character. Templates may also be in the form of descriptions of topological features which can be obtained for each of the members of the alphabet-set, for example by curve-tracing techniques. For Perceptron devices [31], in which there are initially random connections between sensory-receptor cells and "association" cells and between association cells and response cells, the recognition templates eventually "learned" depend, in effect, upon coordinate descriptions in A-unit space which have been "reinforced" for "correct response."

Template-matching systems which are adaptive provide little or no capability for generalization from a shape of one size to other examples of the same shape in other sizes and orientations unless prototypes for these variations have previously been introduced to the system. Fain [12] has pointed up the difficulties of requiring a number of prototypes by estimating that the total number of possible coordinate descriptions for any one shape subjected to size, positional, and rotational transformations lies between $\pi k^2/6$ and πk^2 where k is the number of cells in the superimposed grid or retinal array. On the other hand, template-matching techniques for small, closed-end alphabets, where recognizable patterns have all, in effect, been "seen" before, are well-adapted to devices with relatively little programmed logic, low storage requirements, and rather simple means for making identification decisions, e.g., by "best-fit" integration for the response of a photocell array to light reflected through character masks.

In most of the "criterial" or "distinguishing features" approaches to the problem of pattern recognition, relatively invariant properties of recognizable patterns (such as combinations of horizontal, vertical, and diagonal strokes that make up a character, [34] number of corners left after stripping away most black cells [7, 10], "lakes" and "inlets" or cavities open to the left or right [11, 30], connectivities of selected topological features [15, 36], and the like) are used to provide tolerance for some transformations which may occur for the "same" shape. This advantage is typically achieved at the price of extensive processing in the extraction of the discriminating features, or of considerable storage capacity, or of comparatively sophisticated recognition logic.* Systems utilizing context-dependency principles, such as those discussed by Bledsoe and Browning [5], whether based on template-matching or criterial feature approaches, suffer the same respective disadvantages as those approaches involve. In addition, they require means for storage and recall of previous recognition decisions.

A third basis for categorizing prior results in the field of pattern recognition is the degree of orientation toward hardware. Here there is a spectrum ranging from the development of practical character recognition equipment for a given, usually small, alphabet, through investigations directed toward the determination of relatively invariant features of selected classes of patterns, to modelling of perception-recognition phenomena observed to obtain in living organisms.

In the latter area, investigations such as those of Harmon [19] and Singer [36] are concerned primarily with machine design implementations of procedures for recognizing selected geometric shapes with size, position, and rotational invariance. On the other hand, the studies of Sutherland [37] and Deutsch [8] are concerned with mechanisms to explain perception and recognition phenomena in lower-order living organisms, including anomalies of behavior in shape discrimination. For example, they discuss findings which indicate that the octopus can apparently distinguish mirror images in the case of certain shapes but not others. Sutherland has proposed a principle of measurement of the horizontal extent of a shape at each point of the vertical axis and of the vertical extent against the horizontal axis. In his discussion of subsequent tests of this principle, he claims that, to the best of his knowledge: "no other existing theory of shape discrimination would predict that animals should be able to discriminate some mirror images but not others." (Sutherland [37], 1959)

In 1955, Deutsch [8] suggested a contour-projection procedure which accounted for mirror-image ambiguity but not for mirror-image discriminability. He described this procedure as follows:

"Let us assume there is an array of units (or cells) arranged in two dimensions, each unit having many neighbors. This plane composed of cells has messages arriving on it from light receptors . . .

"1. Each unit on the two-dimensional array can be excited by a counter falling on the region of the retina to which it is joined.

"2. When such excitation . . . . arrives each unit will pass on a pulse down what will be called a final common cable. It also will excite its neighbor . . . . Therefore when a contour is projected on the two-dimensional array . . . . a message consisting of one pulse will be passed down the final common cable by each cell or unit on which that contour lies . . . .

"3 Second, the contour will excite all the cells which lie next to it on the two-dimensional array. These will pass the excitation on to their neighbors. . . The assumption is made that a cell will pass on its excitation at right angles to the contour of which it happens to be a component. . .

"4. As such lateral excitation from a point in a contour advances another pulse will be sent down the final common cable as soon as it coincides with another point in a contour imposed on the two-dimensional array.

"This message down the final common cable will at each moment give a measure of the number of points thus brought into coincidence. . . Thus the message for any rectangle will consist of three sharp volleys, the proportions of the last two being governed by the ratio of the longer to the shorter sides."

The present paper reports the results of preliminary investigations of a particular variation of this Deutsch model. This model

*For example, the requirement: "The height of the left leg of a V-shaped figure less than half the height of the right leg," in Unger's SPAC [40].

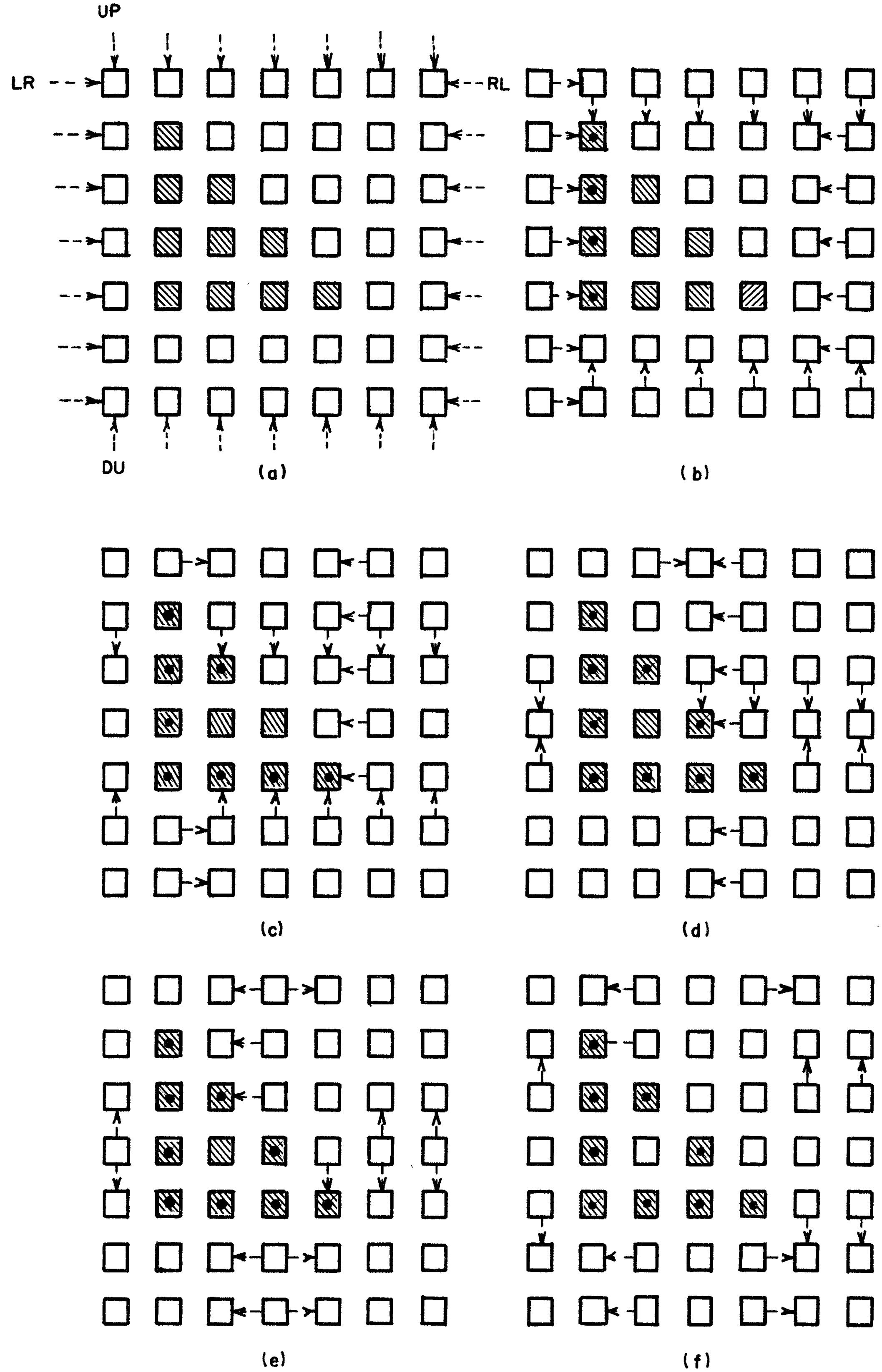


Figure 1. Scan-Mode Operations

was developed with the following objectives in mind:

(a) it will consist of mechanizable processes but avoid extensive computational, housekeeping, feature-extraction, and storage requirements.

(b) It will successfully recognize a certain number of categories of abstract shapes, together with large numbers of size, positional, and certain rotational variants for each such shape, and it may exhibit some of the anomalies with respect to rotational transformations observed by Sutherland.

(c) It will be useful for some practical applications, e.g., for closed vocabularies of selected shapes which are carefully drawn or deliberately constrained, as might be the case for stylized schematic drawings.

3. OPERATION OF THE MODEL

In this particular model for shape recognition, the principle used is that of fired-cell coincidence counting with respect to the projection of excitation from given contour-indicating cells against the cells which are also members of contours. A retinal mosaic or quantization grid in the form of a rectilinear array is assumed. Each cell has eight immediately adjacent neighbors and there are eight possible directions for a straight-edge contour line.

For convenience, we will make certain simplifying assumptions and we will defer for later discussion questions of noise, resolution, effects of quantization, and the matter of multiple shapes distributed in the input pattern field. However, shapes entirely within other shapes can be handled one at a time. We assume that correctly drawn shapes are imposed on a square array of n x n units of size; that the line width is approximately equal to the width of a cell, or resolution unit, and that input patterns are at least several resolution units in size. We require a space free of objects (i.e., a border) in order to determine the background color. Thus only shapes which can be inscribed in an n-2 x n-2 area will be processed consistently. Means for detection of contour-membership (such as those described by Babcock [2]) are assumed to be included in the scan-mode operation, described below.

In the scan-mode operation, horizontal and vertical scans are made with an individual scan-line for each row and each column of the array, in both directions, as shown in Figure 1.* The scan-mode activities shown in Figure 1 proceed along each scan-line until a first "black" cell** is encountered. This is a contour-indicating cell. Whenever a scan-line reaches such a cell, it stops. Otherwise the scan-mode activities proceed along that line until the border of the field is reached.

In example (a) of Figure 1, no cells have as yet been identified as contour-indicating cells. Example (b) indicates the status as of this initial step, plus one.† At this and in succeeding steps, precisely those cells which are the first blacks with respect to the scan-line along which they were reached will "fire." This firing is illustrated in the examples of Figure 3 by the dots superimposed on the appropriate shaded (input-pattern-affected) cells.

As we have noted, scan-lines which encounter a contour-indicating cell will terminate at this point. For this reason, in this particular model of shape recognition, it is clear that the resultant activity based on fired cells will be the same for either a solid or an outline shape, examples (e) and (f) of Figure 1, and for shapes entirely containing other shapes or noise. Some contour-indicating cells will be reached by more than one scan-line, but in this model the effect is the same for both single and multiple encounters.

Scanning and contour-detection operations are followed by contour-projection activities as illustrated in Figure 2. This contour-projection activity will be initiated in any fired (contour-indicating) cell which has an immediately adjacent contour-indicating cell, and it will proceed at right angles to the line joining such neighbors. A line of two or more

*For purposes of illustration, the cells of the array are shown as displaced from their immediate neighbors by one unit in all directions.

**That is, the cell at the edge of the object with enough black impinging on it to meet the quantization requirements, assuming the background is white. For a black background, this would be the first "white" cell.

†Questions of the timing and sequence of scanning are immaterial because cells fired by the scan are counted only after completion of scanning and propagation activities. Neither absolute nor relative timing considerations are required for recognition in the present model.

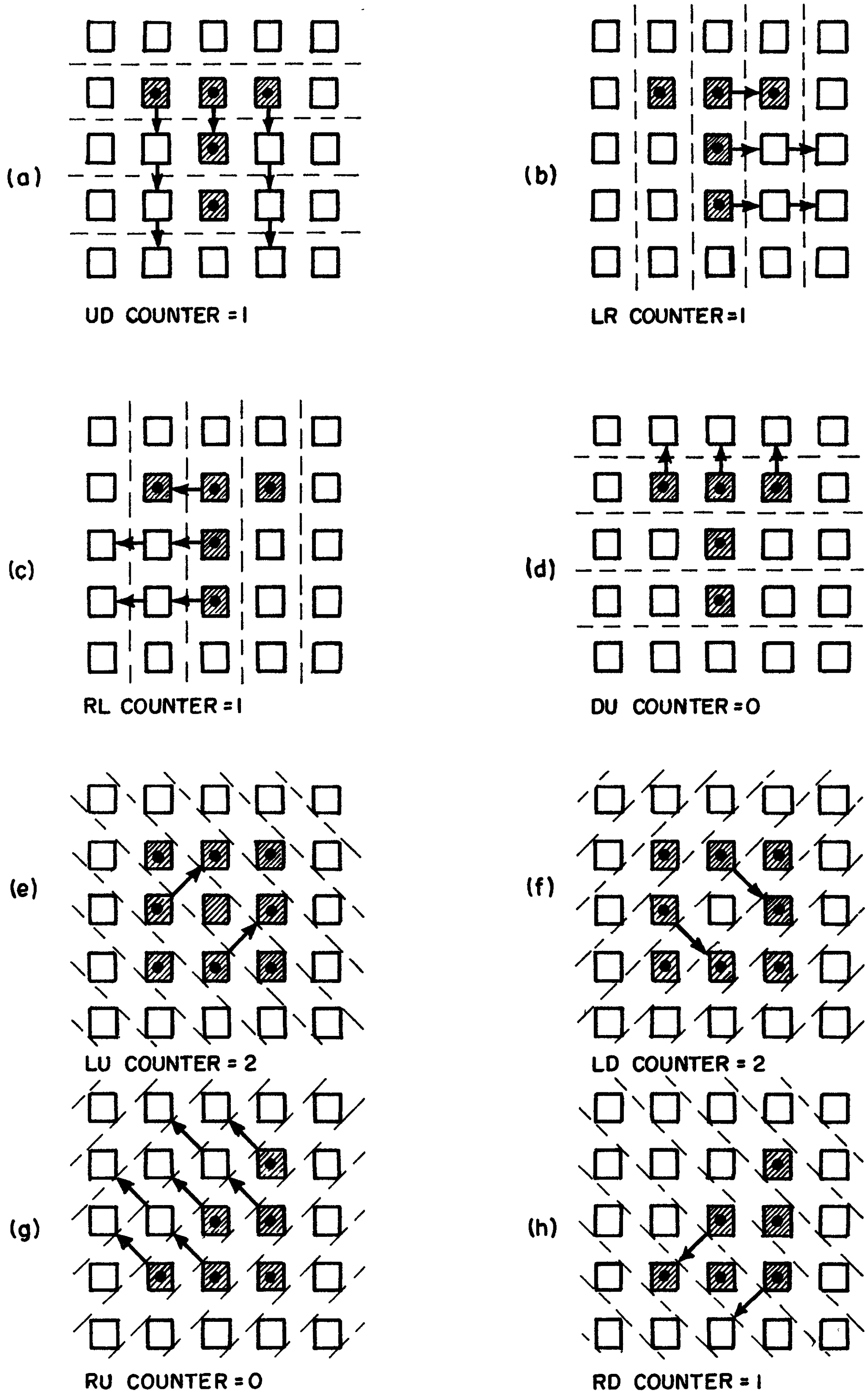


Figure 2. Contour-Projection Fronts in the Model

immediately adjacent fired neighbors thus defines projection activity for each in two directly opposed contour-projection "front" directions. These we categorize as "Left-right" ("LR"), and so forth for the eight possible front directions, as shown in Figure 2.

Each of the activated projection-lines proceeds until it reaches another contour-indicating cell, at which time a tally of this coincidence hit is made in the appropriate directional category. It is to be noted that the projection-line encounter possibilities are not limited to the case of diametrically opposite contour firings. No tally is made if the border is reached by the projection-propagation line without encounters occurring.

Further, only one coincidence hit per projection line is allowed in this model (that is, in effect, we start projections from the borders of the field and register only the first encounter, if any, along a given activated line). For instance, although four cells are projection-activated in the from-left-upward direction in example (e) of Figure 2 (these four meet the requirements both of scan-mode first encounter and of co-fired neighbors), only the two lowermost become effectively active in this "front" direction. Example (g) of Figure 2 illustrates a case in which two different fronts are established in the same direction (for this example, the from-right-upward), but in which no hits would occur in this direction. Figure 2 also gives the directional-counter hit-tallies which would occur for the appropriate ones of the projection-front-directions for the input patterns shown. Again, timing considerations are immaterial since fired cells remain active until, and use of counts takes place only after, the completion of all scanning and projection activities.

Once the projection operations have been completed and all coincidence-hits have been tallied in the appropriate directional counter, the recognition-identification operations begin. These are based upon equality-inequality comparisons for the number of coincidence hits tallied in the eight directional counters, taken two counters at a time. For instance, in the case of the square, examples (e), (f), of Figure 2, the results of such comparisons which are necessary and sufficient to distinguish these squares from other shapes distinguishable by the model (Figures 3, 4, and 5, for example) are as follows:

(1) LR = UD
(2) LR = RL
(3) LR = DU
(4) UD = RL
(5) UD = DU
(6) RL = DU
(7) LU = LD
(8) LU = RU
(9) LU = RD
(10) LD = RU
(11) LD = RD
(12) RU = RD
(13) LR ≠ LU
(14) LR ≠ LD
(15) LR ≠ RU
(16) LR ≠ RD
(17) UD ≠ LU
(18) UD ≠ LD
(19) UD ≠ RU
(20) UD ≠ RD
(21) RL ≠ LU
(22) RL ≠ LD
(23) RL ≠ RU
(24) RL ≠ RD
(25) DU ≠ LU
(26) DU ≠ LD
(27) DU ≠ RU
(28) DU ≠ RD

Precisely these relationships will obtain for larger squares which have equal numbers of quantized units per side, whether outline, solid, or enclosing other shapes.

These results for equality-inequality comparison between the contour-projection coincidence-hit counters serve to distinguish an input pattern that is either a square or a diamond (45° tilted square) from other shapes such as those indicated in Figures 3, 4, and 5. However, in order to distinguish the square from the diamond, if desired, it is necessary to provide a tie-breaking decision based upon the number of contour-projection directional counters which have non-zero tallies. That is, all eight counters will have tallies in the case of the square*, but in the case of the diamond only the four diagonal counters will have tallied hits**. Some of the other pairs of shapes shown in Figure 3 can be distinguished from each other only by the tie-breaking check of the number of counters having non-zero tallies for the given input pattern.

The hypothetical model, operating as described above, has been checked by manual simulation against a variety of geometric shapes constructed of straight-line segments crossing near the centers of affected cells, specifically including the shapes shown in Figure 3. Obviously, shapes that can be separately discriminated from each other may also be grouped into sub-sets in various ways. Such grouping possibilities may be used to treat as equivalent the members of

*With the minor exception of the square that has only two resolution units per side, to be discussed later.

**See Figure 6, examples (c) of Cases I and II, and examples (a) of Case I and (b) of Case II, respectively.

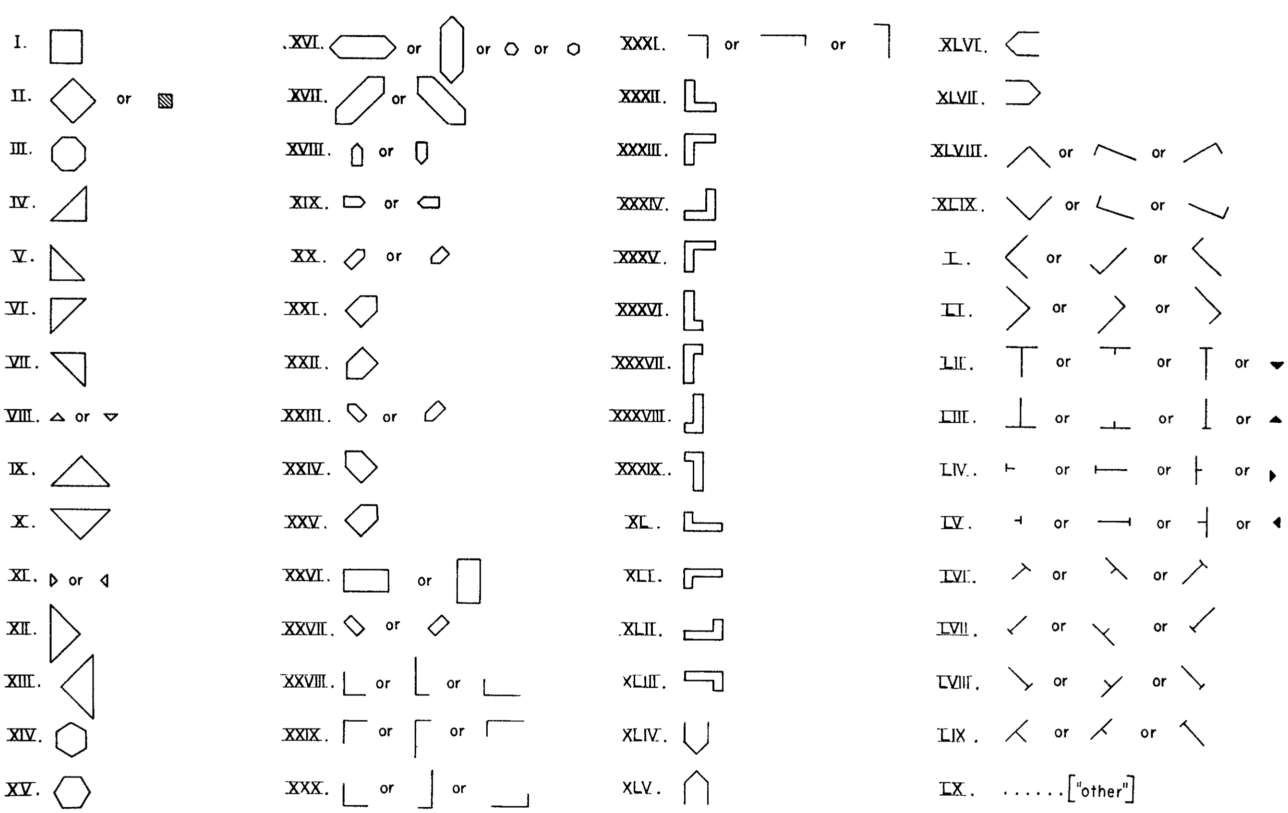


Figure 3. Examples of Shapes That Can be Separately Identified

particular sub-sets chosen as categories of shapes which may be of interest, such as those shown in Figure 4. Categories such as those of Figure 4 may be further regrouped in various other ways. For example, we might require different responses for the tilted right angles and the "up right" single-line right angles shown in Category IX, but assign the same output response requirement (i.e., "upright right angle") both to the latter and to the members of Category X.

Once decisions have been made as to grouping, the recognition logic may be based either on all 28 comparisons (made sequentially or in parallel) plus the 8 tie-breaking checks for number of counters with non-zero counts, or on shorter paths through a decision-tree network or equivalent decoding matrix in which the results of one comparison determine which of the other comparisons need also be made. Recognition-decision paths chosen to eliminate redundancy for a particular set of shapes may be selected on the basis of systematic procedures such as those suggested by Gill [16] or Glovasky [17], or on a trial-and-error basis specifically including the possibilities of reward-reinforcements for "learning" from experience with various "teaching" examples.

A typical set of trial-and-error decision paths is shown in Figure 5, in the form of a computer program flowchart*. This was determined from observation of results for various sizes, proportions, and 45° rotations of the 20 categories of shapes shown in the output boxes. These categories represent a further illustrative grouping of sub-sets of shapes, where a given shape may (double outline or solid "L") or may not (triangle) be given the same output response assignment as one or more of its rotational transforms.

An alternative output response, "other," is also shown in Figure 5, but in general it is presupposed that recognizable input patterns belong to some one of the indicated shape categories and that they are reasonably carefully drawn to meet the constraints implicit

*In Figure 5 and later figures, the notation (x) $\neq$ 0 means "exactly x of the eight counters have non-zero tallies."

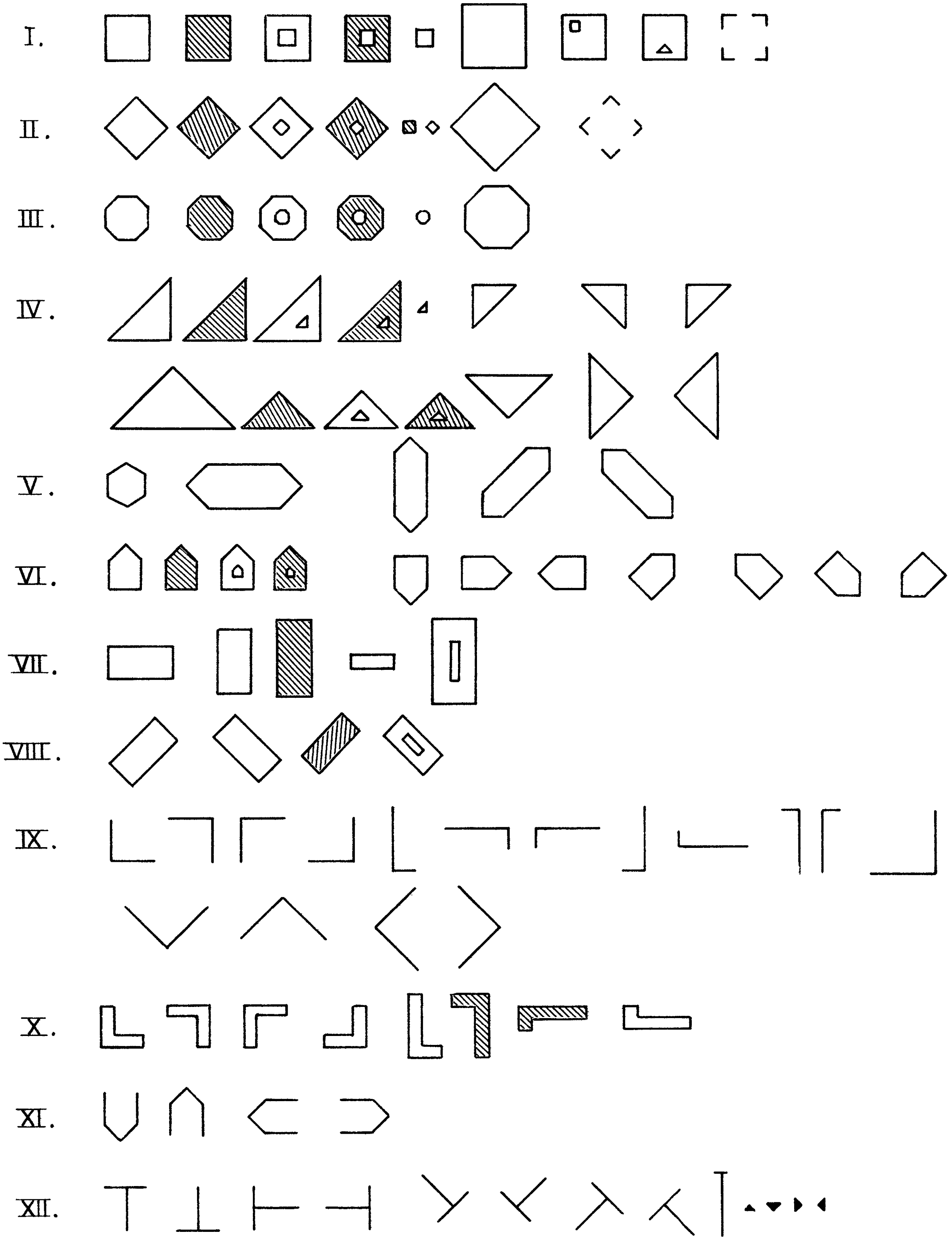


Figure 4. Examples of Shapes That Can Be Recognized as Equivalent

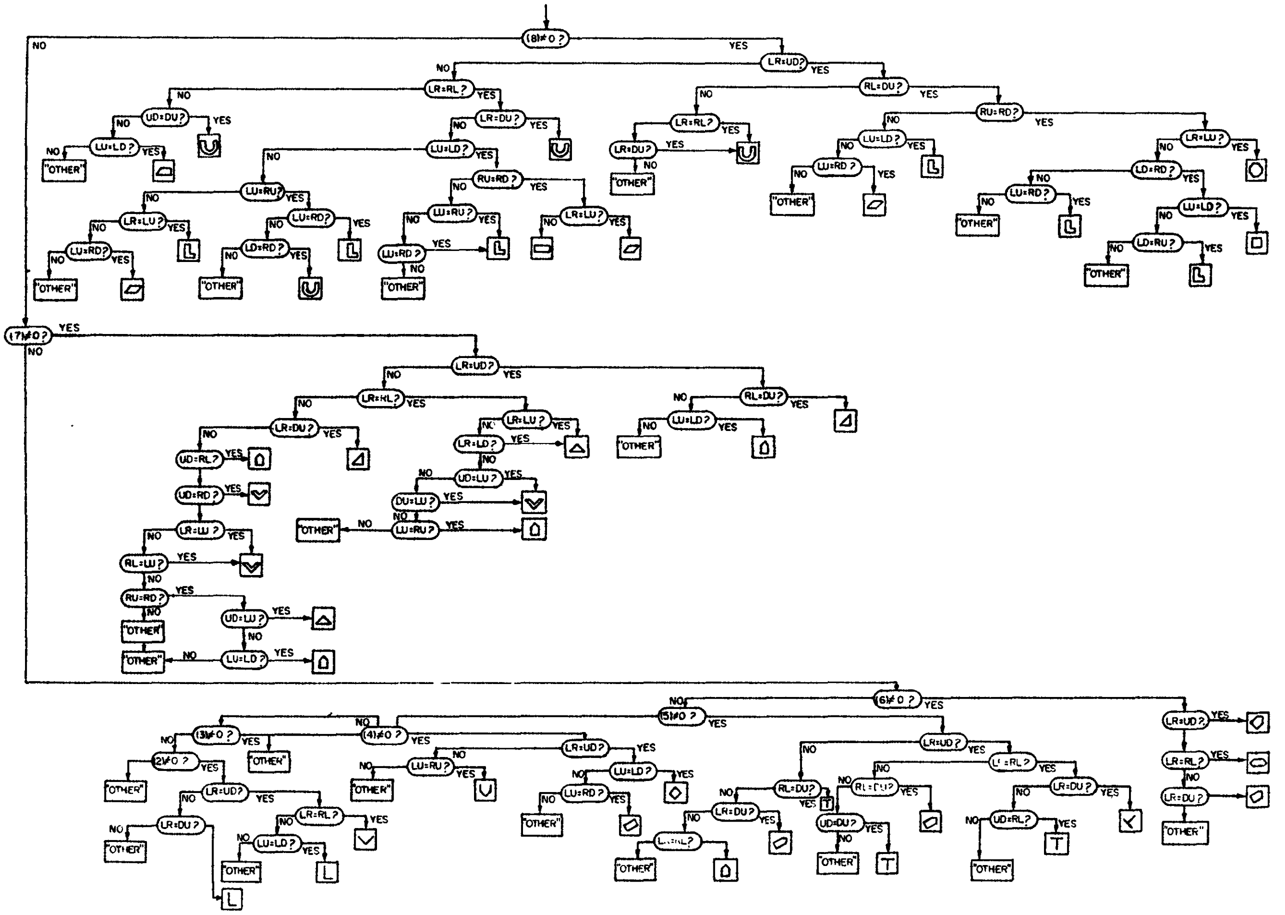


Figure 5. Recognition Logic For Selected Shapes

in the system. Within these limiting conditions, this program (or an equivalent coincidence-gating network) would make a very high percentage of correct recognition-decisions regardless of size, position, and certain rotational transformations. However, in some cases, especially for some shapes (e.g., double outline or solid "V" and "U") not included in the groups of recognizable categories in Figures 3 and 4, anomalies and ambiguities will occur for certain critical sizes or proportions.

4. AMBIGUITIES, ANOMALIES, AND PROBLEMS OF PRACTICAL APPLICATION

Some of the anomalies and ambiguities observed in the model are illustrated in Figure 6. In each of the cases shown in Figure 6, example (a) is ambiguous with respect to example (b), but not so with respect to example (c). Case I of Figure 6 involves the exception previously noted that a square of the minimal size that is identifiable in the model (i.e., two cells or resolution units per side) behaves as though it were a "tilted square," or diamond. We consider this case one of anomaly, since if we wish to treat the 45°, 90°, etc. rotational transforms of a given shape as being the "same" shape, there would be no problem. We reserve the term "ambiguity" for cases in which a particualr shape is confused by the model with a shape belonging to an "obviously" different category. Case II of Figure 6 illustrates a persistent ambiguity of the cross and the diamond.

Figure 6 next shows (Case III) the special case of the triangle which has 3x3x5 dimensions after quantization. This presents the anomaly that the mirror images cannot be distinguished from each other, whereas for larger triangles of the same type this distinction may be made if desired. It is therefore an example of anomalies occurring at a certain critical size. Other cases such as the ambiguity of the double outline or solid "V" with the pentagon, and the anomaly with

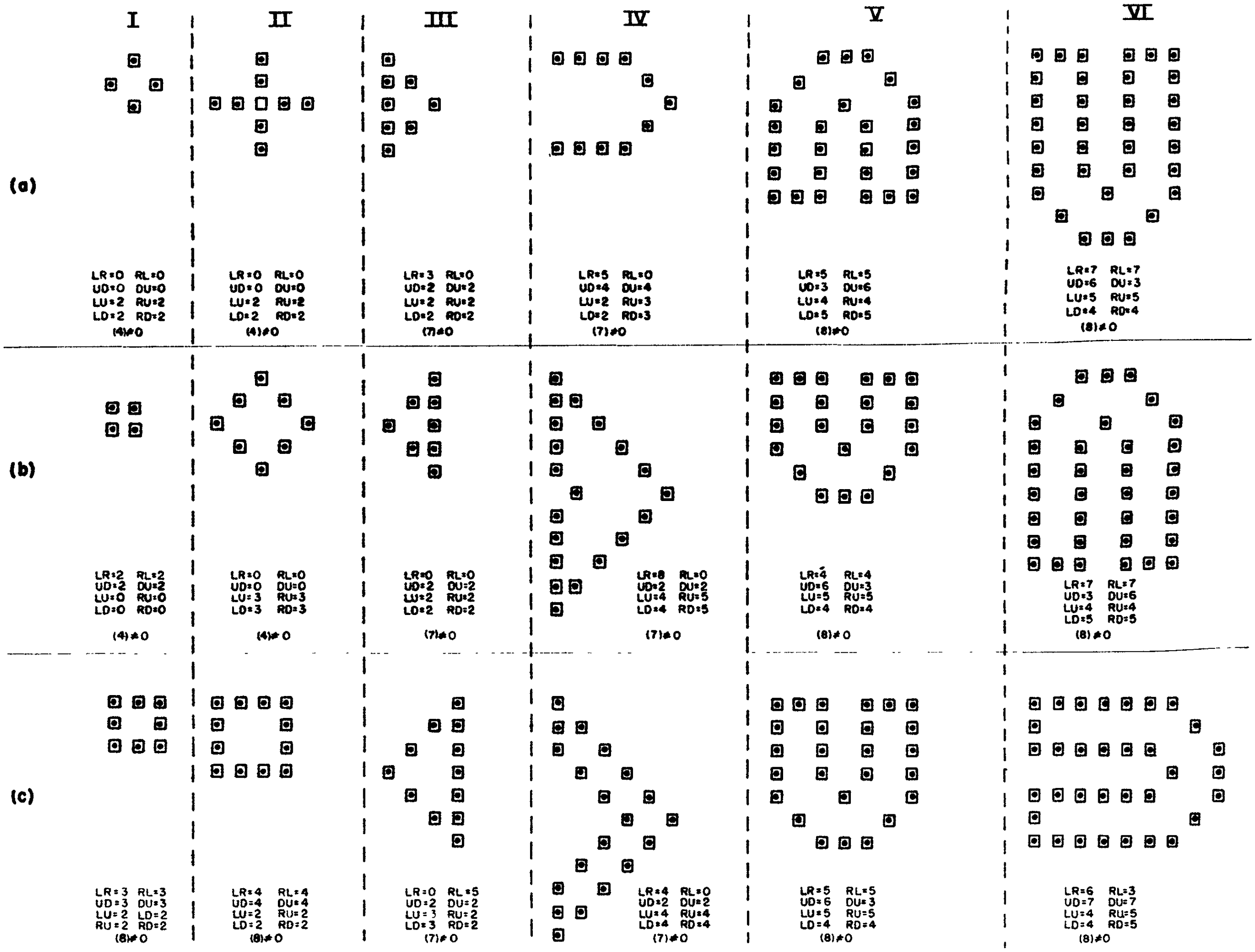

I
II
III
IV
V
VI
(a)
LR=0 RL=0 UD=0 DU=0 LU=2 RU=2 LD=2 RD=2 (4)≠0
LR=0 RL=0 UD=0 DU=0 LU=2 RU=2 LD=2 RD=2 (4)≠0
LR=3 RL=0 UD=2 DU=2 LU=2 RU=2 LD=2 RD=2 (7)≠0
LR=5 RL=0 UD=4 DU=4 LU=2 RU=3 LD=2 RD=3 (7)≠0
LR=5 RL=5 UD=3 DU=6 LU=4 RU=4 LD=5 RD=5 (8)≠0
LR=7 RL=7 UD=6 DU=3 LU=5 RU=5 LD=4 RD=4 (8)≠0
(b)
LR=2 RL=2 UD=2 DU=2 LU=0 RU=0 LD=0 RD=0 (4)≠0
LR=0 RL=0 UD=0 DU=0 LU=3 RU=3 LD=3 RD=3 (4)≠0
LR=0 RL=0 UD=2 DU=2 LU=2 RU=2 LD=2 RD=2 (7)≠0
LR=8 RL=0 UD=2 DU=2 LU=4 RU=5 LD=4 RD=5 (7)≠0
LR=4 RL=4 UD=6 DU=3 LU=5 RU=5 LD=4 RD=4 (8)≠0
LR=7 RL=7 UD=3 DU=6 LU=4 RU=4 LD=5 RD=5 (8)≠0
(c)
LR=3 RL=3 UD=3 DU=3 LU=2 LD=2 RU=2 RD=2 (8)≠0
LR=4 RL=4 UD=4 DU=4 LU=2 RU=2 LD=2 RD=2 (8)≠0
LR=0 RL=5 UD=2 DU=2 LU=3 RU=2 LD=3 RD=2 (7)≠0
LR=4 RL=0 UD=2 DU=2 LU=4 RU=4 LD=4 RD=4 (7)≠0
LR=5 RL=5 UD=6 DU=3 LU=5 RU=5 LD=4 RD=4 (8)≠0
LR=6 RL=3 UD=7 DU=7 LU=4 RU=5 LD=4 RD=5 (8)≠0

Figure 6. Examples of Ambiguity and Anomaly

respect to mirror-image discriminability for some, but not all, "U"-shaped figures, are related to critical proportions. Mirror-image indistinguishability for other shapes, such as the oblique rectangle, are persistent.

We have hitherto assumed that the cells (resolution units) are approximately equal to the width of a single line, but it is to be noted that for other levels of resolution, many of the same anomalies and ambiguities would appear, as shown in several examples of Figure 7. They are related to critical size or critical proportions in terms of the resolution units used.

Figure 7 illustrates some of the effects of quantization for various levels of resolution. The first example shown is that of the effect of quantization which explains the anomaly previously to be noted in Figures 3 and 4 for the case of the triangle having, after quantization, dimensions of 3x2x2 resolution units. This will be confused with any single line "T" whose cross-bar is oriented in the same direction as the hypotenuse of this triangle.

It will be noted from the examples of Figure 7 that, for resolution appreciably coarser than the width of the single line, additional ambiguities will appear unless micropositioning constraints are imposed. For example, in situations where the resolution level differs appreciably from the width of the single line, or, in any case to insure sufficient accuracy of free-hand construction of input patterns which would be consistently recognizable, we might insist that the lines or boundary edges must cross the centers of units corresponding to the resolution units used for quantization.

With respect to noise, it is to be noted that random noise entirely included in a shape has no adverse effect on the recognition of the shape in which it is included. Protrusions

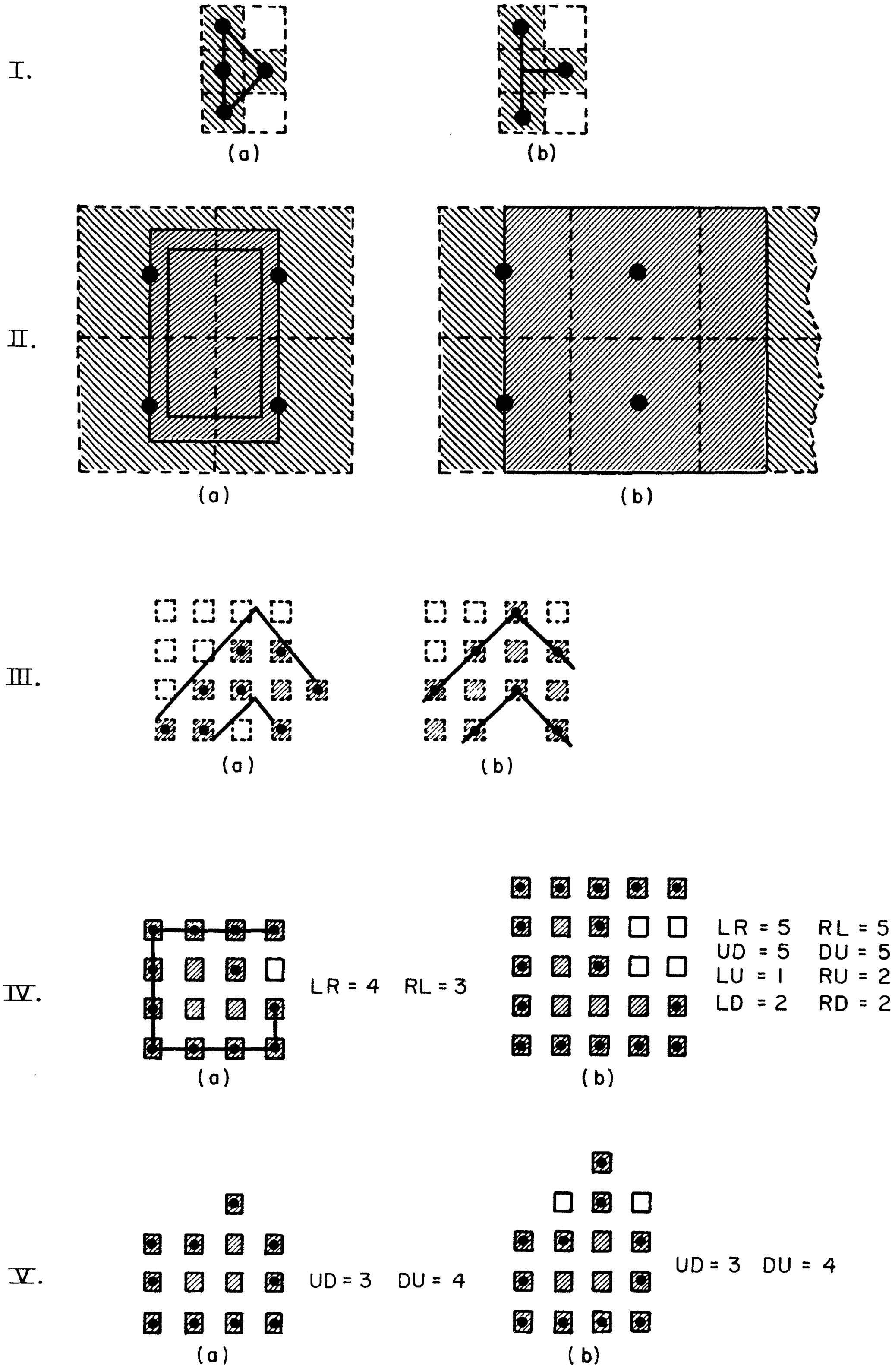


Figure 7. Examples of Effects of Quantization and Noise

sufficient to cause the quantization threshold to be exceeded in adjacent cells external to the proper contour will obviously result in incorrect recognitions in many cases. External random noise which does not occur on any scan-line that is involved for a given shape will not interfere with recognition of that shape, but will behave, if large enough to meet the quantization requirements for one or more cells, as a case of multiple patterns in the field.

It is obvious from the examples shown in Figures 6 and 7 that certain limiting conditions must be met if this particular recognition model is to give consistently accurate performance for the shapes it can discriminate with respect to each other. First is the restriction that the input patterns to be recognized must be so drawn or so processed that shapes recognizable as the same shape retain the same geometric properties (with respect to the centers of resolution units affected) after quantization as before. This may be achieved either by pre-centering (by crossing the centers of units proportional to resolution units when the original input pattern is constructed), or by subjecting the input pattern to image improvement operations including straight-edging and clean-up processes where the straight-edges applied must conform to an appropriate 45° slope through the centers of immediately adjacent resolution units or by slight jittering or shifting so that lines or edges do cross the centers of as many affected cells as possible. A second restriction is that input patterns to be consistently recognizable must be at least two, and preferably three, resolution units long per contour side.

The third restriction in the present model is that the members of the set of shapes to be separately recognized must be distinguishable from each other in terms of specific functions of the 28 equality-inequality comparisons and the 8 tie-breaking possibilities using the exact number of counters that have non-zero counts. For instance, the diamond and the cross (Figure 6) cannot both be included in such a set. As we have noted previously, however, the model provides a means for detecting whether or not large numbers of possible other shapes would be ambiguous with respect to any of the members of a particular set. This, in Figure 8 are shown two arbitrary shapes constructed by a Monte Carlo Technique. These shapes, for example, are not ambiguous with respect to any of the shapes shown in Figures 3, 4, or 5.* Moreover, by determining what other connected cells (cells marked "1", "2", "3", etc., in Figure 8) would <u>not</u> change the counter tallies if they were <u>also</u> black, large families of shapes, none of which would be ambiguous with respect to the members of the reference set, but all of which would be ambiguous with respect to each other, can be defined. Alternately, for any given shape investigation of connected cells which if black <u>would</u> change the counters** detects large families of shapes that are non-ambiguous with respect to that shape, although they may or may not be ambiguous with respect to one another.

The three limiting conditions, taken together, certainly restrict the area of possible practical application of this particular model for abstract shape recognition. The first two conditions, however, can be satisfied if the preparation of input patterns (manually or otherwise) is suitably constrained. The third restriction as to the closed vocabulary or alphabet of shapes that can be separately identified, moreover, does not mean that the closed set need be small. On the contrary, if we accept the first limiting condition, and insist further that acceptable patterns must be at least three resolution units long per contour line, that they must be approximately straight-line figures, and that they conform to the required angular resolution, the closed vocabulary may include many members of the set of non-contradictory combinations of the 28 counter comparison results. Each such member may also involve an appreciable fraction of the lower limits of Fain's [12] estimate for the number of possible size, position, and rotational variants.

If we consider the non-anomalous cases of Figures 3 and 5, the number of separately identifiable shapes will approach at least 60-70 different responses. This means that a closed alphabet set with at least as many members as a set comprised of upper and lower case alphanumeric characters can be used, for example, for the encoding of results of the subject content analysis of items to be

*That is, with respect to the 28 counter comparisons. Additional or different decision points may be required in a less redundant recognition logic such as that of Figure 5.

**E.g., some of the cells marked with an "x" in Figure 8.

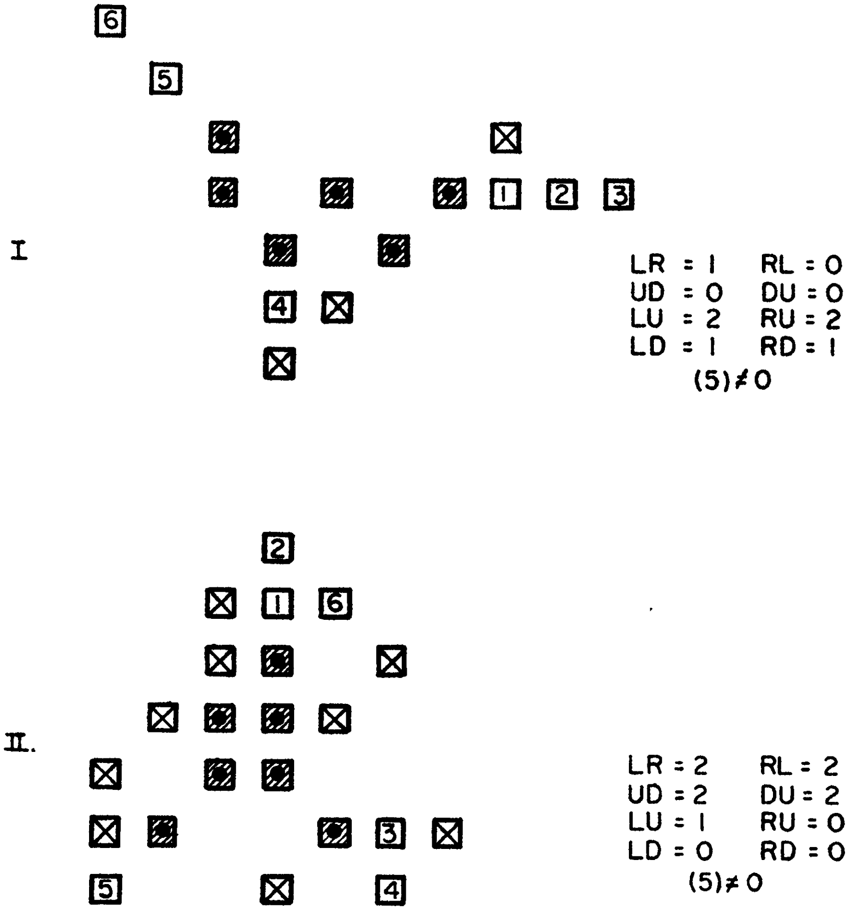


Figure 8. Determination of Additional Distinguishable or Indistinguisable Shapes

stored in a selection and retrieval system in a form directly readable by machine. Some at least of the members of this set (e.g., hexagons, pentagons, right-angled isosceles triangles) might well have greater associative value than alphanumerics, both upper and lower case, especially for such cases as chemical structure or circuit diagram representations. This set may be extended to include additional random shapes checked-out for non-ambiguity with respect to the other members, in accordance with the procedure previously indicated. Further, for a given recognition set of non-contradictory functions for the 28-coincidence counters and the 8 tie-breakers, additional specific functions may be found which lead to shapes that can be easily constructed within the constraints and which would still be readily distinguishable to the human eye as well as to the machine. These, by suitable provision for simultaneously - occurring multiple - pattern processing, may be linked together in various ways.

Such possibilities for application of the model therefore pose the problems of more than one shape simultaneously present in the field. If a shape is entirely included within another shape, it may be separately processed in an erase-repeat-scan operation. For example, after processing of the outer shape, a second scan-mode operation might be initiated in which cells fired in the first scan will not refire in the second scan but cells which are the next inmost neighbors will if they also present boundary discontinuities with respect to the background. Alternatively, the second scan-mode operations might, on re-encountering cells fired on the first scan, reverse the background-field-determination requirements (that is, treat previously fired cells as border-zone cells) and proceed as before.

In the case of multiple patterns not included one within another, we would assume that there are means to detect the presence of more than one object in the input field (e.g., blob-counting techniques such as those described by Kirsch, et al, [24] or the "numatete" mechanism discussed by Babcock [2]) and that, for example, each detected pattern can be isolated in an envelope to which scan and projection operations would be applied as though the envelope were the entire field. With added logic, determinations of relative position of one pattern with respect to another occurring simultaneously in the field might also be made. Decisions with reference to direction and distance between, or comparative size of, two or more distinct input patterns objects in the input pattern field would obviously involve more sophisticated logic.

5. IMPLEMENTATION CONSIDERATIONS

The first consideration with respect to possibilities for implementation or instrumentation of the model is that of preparation or construction of shapes to be used as input patterns. In possible applications such as the recording of results of subject content analysis, we might provide the analyst with forms having preprinted dots so arranged as to conform to resolution unit centers and ask him to "connect-these-dots" with reasonably straight lines. Alternatively, various mark-sensing techniques could be used in conjunction with the filling-in or crossing of preprinted boxes (e.g., the preprinted boxes of mark-sense punch cards).

Without such constraints, we must assume a resolution level approximately equal to the line width (this is not unreasonable and is found in many available scanning devices and in optical character reading equipment), or else that micro-registration uncertainties can be resolved. We would also assume that the input patterns are carefully drawn or are produced mechanically as by a printing device. Various image enhancement, noise reduction, edging and clean-up operations are available in existing equipment, but they require other devices than are implied by our present model.

This model requires as minimum components mechanisms for the following:

(a) Detection of a color-contrast pattern (e.g., black-on-white or white-on-black) quantized in a rectilinear array (e.g., an array of photocells).

(b) Scanning for contour cells in the pattern, (e.g., by exclusive-or processing for black cells encountered by scan-lines).

(c) Detection of a coincidence hit for a particular cell with some other cell which is along a line through neighbors, under conditions as previously discussed.

(d) Eight coincidence-counting devices of n-2 tallying capacity for the n x n array.

(e) Making of 28 equality-inequality decisions with respect to the tallies stored in the coincidence counters.

(f) Determination of the number of coincidence counters which have been activated (i.e., have non-zero counts);

(g) Connection of the results of the equality-inequality comparisons and the number of counters with non-zero counts to the desired output responses.

Figure 5 presents a redundancy-reducing technique for instrumentation of means (g). It also suggests an obvious method for implementation of a complete version of the model—that of simulation by computer. In such case we would assume either a direct input from a scanner attached to the computer, such as the SADIE device for SEAC [24], or an input such as a punch card with holes for cells that would be black for a given pattern, or a machine-generated input.

Obviously, however, we do not need the power of a programmed general-purpose computer, nor of any extensive computational procedures, to achieve the results indicated by the theoretical model. Instead, the elementary logic required can be achieved by relay networks, coincidence-gating networks, or matrix correlation techniques of various types. The latter might include the use of movable connections such as pluggable jumper wires, as proposed, for example, by Fitch [14], in a patent for alphanumeric character reading equipment. Such re-arrangeable connections might be used by an operator-observer to minimize decision-tree paths for a particular set of recognizable shapes, to increase or decrease the sensitivity with respect to rotational transformations for shapes in which these can be distinguished, and to check-out and add new members to the set of identifiable patterns.

A third possible method for implementation of the model would involve the use of various parallel-processing devices, such as the spatial and pattern-detecting special computers suggested by Unger [40] and Kamentsky [23], associative memory devices, or iterative circuit machines. Various simple operations are suggested by Holland [21] for his iterative circuit modules—path-building, path-erasing, module-special-status marking or changing either by input or as a result of internal operations, addition, equality-inequality comparisons. These are well-suited to the recognition procedures we have suggested, but much less so to more complicated computational and housekeeping requirements involved in more sophisticated schemes. In particular, the built-in possibilities in the proposed Holland machines for determining lines of neighbors and for locating operands at the terminals of various branches of a path are of interest.

6. CONCLUSIONS

The results of preliminary investigation of a particular version of contour-projection principles in abstract shape recognition have been reported. These results show that some geometric shapes, constructed in accordance with constraints which have been discussed, can be consistently discriminated. The number of such shapes that can be constructed and recognized would appear to provide a non-trivial vocabulary for possible special-purpose applications. The results to date also suggest various possibilities for further investigation.

Consideration might be given to the incorporation of the present model into a more complex system. Thus, if recognition-results with reference to a rectilinear array were coupled with results with respect to an array arranged hexagonally, the vocabulary could be extended to include shapes having lines or edges at 60° intervals. A system that incorporates use of the actual values of the tally counts, of the direction of inequalities, or of relative-time-of-encounter determinations, would obviously have considerably more versatility, both in resolving anomalies and in detecting comparative size, such as a "large square" and a "small rectangle."

Improvements to the present model and alternate versions would also be considered. For example, other versions in which scan-line contour-detection proceeds in eight rather than four directions, or in which the possible coincidence-bits for a given projection line in a particular direction are not limited to first encounters, would obviously give different results, for instance, for some solid shapes by comparison with outline representations of the same shape. Incorporation of determinations as to which particular counters are non-zero would resolve certain of the anomalies and ambiguities observed in the present model. The finding that both mirror-image ambiguity and mirror-image discriminability occur, in many cases with particular reference to critical size or critical proportions, may suggest further shape-discrimination experiments which might be tried with lower-level living organisms.

Finally, there is an implication that perception-models of this type might be considered in simulated learning or self-organizing system experiments. Stimulus-response learning models which involve recognition means directly related to the specific receptor cells affected by a given input pattern can be developed so as to provide some degree of overlap generalization, [13, 31] by exposure to a number of samples. If, instead, we assume that stimulus-response reward reinforcements are applied to initially random links between response cells and "association" cells which indicate or are directly related to equality-inequality comparison results for contour-coincidence, it is obvious that experience with only a few "squares" can be generalized to successful recognition of very large numbers of other squares. Moreover, such squares can be distinguished from large numbers of other shapes which meet the limiting conditions described. Like the octopus, however, this particular model would apparently never learn to distinguish between an oblique rectangle and its mirror image.

BIBLIOGRAPHY

1. Alt, F. L., "Digital Pattern Recognition by Moments," publication pending, Journal of the Association for Computing Machinery.
2. Babcock, M. L., Some Physiology of Automata," Proceedings of the Western Joint Computer Conference, Los Angeles, May 9-11, 1961, vol. 19, p. 291-298.
3. Baran, P. and G. Estrin, "An Adaptive Character Reader," 1960 IRE WESCON

Convention Record, vol. 4, part 4, p. 29-41.

4. Barus, C., "Machine Learning and Pattern Recognition, a Progress Report on 'A Study of Learning Machines'." Swarthmore, Pa., Dept. of Electrical Engineering. Swarthmore College, 22 December 1959, 25 p.

5. Bledsoe, W. W. and I. Browning, "Pattern Recognition and Reading by Machine," Proceedings of the Eastern Joint Computer Conference, Boston, December 1-3, 1959, vol. 16, p. 225-232.

6. Bomba, J. S., "Alpha-numeric Character Recognition Using Local Operations," Proceedings of the Eastern Joint Computer Conference, Boston, December 1-3, 1959, vol. 16, p. 218-224.

7. Clark, W. A. and B. G. Farley, "Generalization of Pattern Recognition in a Self-Organizing System," Proceedings of the Western Joint Computer Conference, Los Angeles, March 1-3, 1955, vol. 5, p. 86-91.

8. Deutsch, J. A., "The Plexiform Zone and Shape Recognition in the Octopus," Nature, 185:4711 (February 13, 1960), p. 443-446.

- - - - - - "A Theory of Shape Recognition," British Journal of Psychology, 46: Part 1 (February 1955), p. 30-37.

9. Dimond, T. L., "Devices for Reading Handwritten Characters," Proceedings of the Eastern Joint Computer Conference, Washington, December 9-13, 1957, vol. 12, p. 232-237.

10. Dinneen, G. P., "Programming Pattern Recognition," Proceedings of the Western Joint Computer Conference, Los Angeles, March 1-3, 1955, vol. 5, p. 94-100.

11. Doyle, W., "Recognition of Sloppy, Hand-Printed Characters," Proceedings of the Western Joint Computer Conference, San Francisco, May 3-5, 1960, vol. 17, p. 133-142.

12. Fain, V. S., "The Quantity of Coordinate Descriptions of Images in Systems for Recognition of Visible Patterns," In: Soviet Developments in Information Processing and Machine Translation. New York, U. S. Joint Publications Research Service, 28 July 1960, 14 p. (JPRS: 3570).

13. Farley, B. G., "Self-Organizing Models for Learned Perception." In: Self-organizing Systems; Proceedings of an Interdisciplinary Conference, 5 and 6 May 1959, edited by M. C. Yovitz and S. Cameron, New York, Pergamon Press, 1960, p. 7-30.

14. Fitch, C. J., "Character Sensing and Analyzing System," U. S. Patent 2,682,043; patented 22 June 1954.

15. Garmash, V. A., V. S. Pereverzev, and V. M. Tserlin, "Quasi-topological Method of Letter Recognition," In: Foreign Developments in Machine Translation and Information Processing, No. 19. Washington, U. S. Joint Publications Research Service, 23 January 1961, p. 1-6. (JPRS: 6633.)

16. Gill, A., "Minimum-Scan Pattern Recognition," IRE Transactions on Information Theory, IT-5:2 (June 1959), p. 52-58.

17. Glovazky, A., "Determination of Redundancies in a Set of Patterns," IRE Transactions on Information Theory, IT-2:4 (December 1956), p. 151-153.

18. Grimsdale, R. L., F. H. Sumner, C. J. Tunis, and T. Kilborn, "A System for the Automatic Recognition of Patterns," Proceedings of the Institution of Electrical Engineers, 106, Part B (March 1959), p. 210-221.

19. Harmon, L. D., "A Line-Drawing Pattern Recognizer," Proceedings of the Western Joint Computer Conference, San Francisco, May 3-5, 1960, vol. 17, p. 351-364.

20. Hebb, D. O., "The Organization of Behavior." New York, Wiley, 1959, 335 p.

21. Holland, John, "A Universal Computer Capable of Executing an Arbitrary Number of Sub-Programs Simultaneously," Proceedings of the Eastern Joint Computer Conference, Boston, December 1-3, 1959, No. 16, p. 108-113.

22. Johnson, R. B., "Indicia-Controlled Record Perforating Machine." U. S. Patent 2,741,312; patented 10 April 1956.

23. Kamentsky, L. A., "Pattern and Character Recognition Systems; Picture Processing by Nets of Neuron-Like Elements," Proceedings of the Western Joint Computer Conference, San Francisco, March 3-5, 1959, No. 15, p. 304-309.

24. Kirsch, R. A., L. Cahn, L. C. Ray, and G. H. Urban, "Experiments in Processing Pictorial Information with a Digital Computer," Proceedings of the Eastern Joint Computer Conference, Washington, December 9-13, 1957, No. 12, p. 221-229.

25. Kovasznay, L. S. G. and H. M. Joseph, "Image Processing," Proceedings of the IRE, 43:5 (May 1955) 560-570.

- - - - - - and A. Arman, "Optical Autocorrelation measurement of Two-Dimensional Random Patterns," Review of Scientific Instruments, 28:10 (October 1957) 793-797.

- - - - - - and H. M. Joseph, "Processing of Two-Dimensional Patterns by Scanning Techniques," Science, 118:3069 (23 October 1953) 475-477.

26. Minsky, M. L., "Steps Toward Artificial Intelligence," Proceedings of the IRE, 49:1 (January 1961) 8-30.

27. Novikoff, A. B. J., "Statistical Properties of Geometric Figures Invariant Under Translation and Rotation." In: Transactions of the Symposium on Principles of Self Organization, University of Illinois, June 1960, edited by H. Von Foerster. London, Pergamon (in press).

28. Pitts, W. and W. S. McCulloch, "How We Know Universals—The Perception of Auditory and Visual Forms," Bulletin of Mathematical Biophysics, 9:3 (September 1947) 127-147.

29. Roberts, L. G., "Pattern Recognition With an Adaptive Network." IRE International Convention Record, 8 part 2 (1960) 66-70.

30. Rochester, N., J. R. Johnson, G. M. Aondahl, and W. E. Mutter. "Recognition of Recorded Intelligence," U. S. Patent 2,889,535, patented 2 June 1959.

31. Rosenblatt, F., "The Perceptron: A Perceiving and Recognizing Automaton." Buffalo, N. Y., Cornell Aeronautical Laboratory, January 1957. Project PARA. Report No. 85-460-1

- - - - - - "Perceptron Simulation Experiments," Proceedings of the IRE, 48:3 (March 1960) 301-309.

- - - - - - "Perceptual Generalization Over Transformation Groups." In: Self-Organizing Systems; Proceedings of an Interdisciplinary Conference, 5 and 6 May 1959, edited by Marshall C. Yovits and Scott Cameron. New York, Symposium Publications Division, Pergamon Press, 1960, p. 63-100.

32. Schade, O. H., Sr., "Optical and Photoelectric Analog of the Eye," Journal of the Optical Society of America, 46:9 (September 1956) 721-739.

33. Selfridge, O. G., "Pattern Recognition and Modern Computers," Proceedings of the Western Joint Computer Conference, Los Angeles, March 1-3, 1955, No. 5, p. 91-93.

34. Shepard, D. H., P. F. Bargh, and C. C. Heasly, Jr., "A Reliable Character Sensing System for Documents Prepared on Conventional Business Devices," IRE WESCON Convention Record, 1, part 4 (1957) 111-120.

35. Sherman, H., "A Quasi-Topological Method for the Machine Recognition of Line Patterns," In: Information Processing; Proceedings of the International Conference on Information Processing, UNESCO, Paris, 15-20 June 1959. Paris, UNESCO, Munchen, R. Oldenbourg; London, Butterworths, 1960, p. 232-237; addendum, p. 237-238; discussion, p. 238.

36. Singer, J. R., "Model for a Size Invariant Pattern Recognition System." In: Bionics Symposium, 13-15 September 1960. Wright-Patterson Air Force Base, Ohio, Directorate of Advanced Systems Technology, Wright Air Development Division, Air Research and Development Command, December 1960, p. 239-245. (WADD technical report 60-600) ASTIA doc. No. AD-252,300.

37. Sutherland, N. S., "A Test of a Theory of Shape Discrimination in Octopus Vulgaris Lamarck," Journal of Comparative and Physiological Psychology. 52:2 (April 1959) p. 135-141.

38. Taylor, W. K., "Pattern Recognition by Means of Automatic Analogue Apparatus," Proceedings of the Institution of Electrical Engineers, 106, part B (March 1959) 198-209.

39. Uhr, L., "Intelligence in Computers the Psychology of Perception in People and in Machines," Behavioral Science, 5:2 (April 1960) 177-182.

- - - - - - and C. Vossler, "A Pattern Recognition Program that Generates, Evaluates, and Adjusts Its Own Operators," Proceedings of the Western Joint Computer Conference, Los Angeles, May 9-11, 1961, vol. 19, p. 555-569.

40. Unger, S. H., "A Computer Oriented Toward Spatial Problems," Proceedings of the IRE, 46:10 (October 1958) 1744-1750.

- - - - - - "A New Type of Computer Oriented Toward Spatial Problems," Proceedings of the Western Joint Computer Conference, Los Angeles, May 6-8, 1958, No. 13, p. 234-239.

41. Uttley, A. M., "Imitation of Pattern Recognition and Trial-and-Error Learning in a Conditional Probability Computer," Reviews of Modern Physics, 31:2 (April 1959) 546-548.

CHRYSLER OPTICAL PROCESSING SCANNER (COPS)

A Character Recognition System Which is Independent of Character, Translation, Size or Orientation

D. N. Buell
Chrysler Corporation
Centerline, Michigan

ABSTRACT

A character recognition system is described which incorporates a lens-and-retina input and a relatively simple computer. The system operates effectively with the image focused anywhere on the retina and may operate so as to be independent of image, size or orientation.

The characters which may be unambiguously, discriminated are those which have distinct transform functions T^*. A simple algorithm is given for obtaining T^* for both curvilinear and rectilinear figures. Examples are given and possible means of resolving the ambiguities discussed.

The Transform of a Pattern

This paper describes the conceptual mechanization of an optical scanner designed to accomplish a specific, well defined task. After the machine is described, there is given a brief discussion of processes, events and components which form a part of the psycho-physiological visual system. It appears that there are, at least superficially, a number of analogous concepts.

Computers designed to recognize patterns may take either of the forms shown in Figure 1. Some preprocessing on the inputs results in a transform (the round dot in the Figure). It is this transform which is identified or recognized by the computer. Usually, the transform is a binary number. The number may be identified as such, existing in some specific registers in the machine, or it may be physically or logically distributed, and represented by the binary 0 - 1 state of several elements of different circuits. But, in

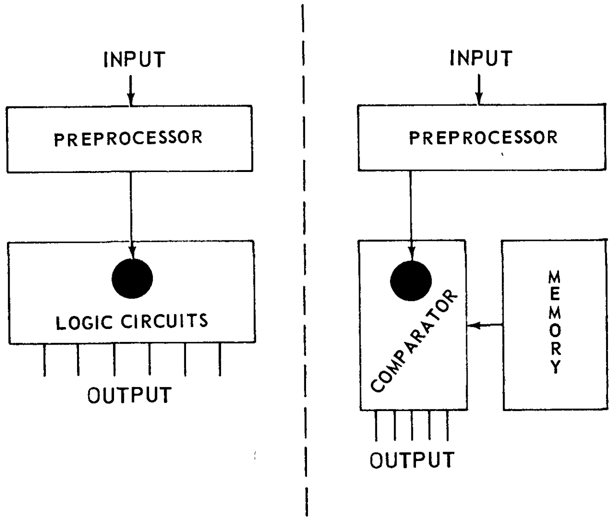


Figure 1. Diagrams of generalized scanner principles illustrating that identification is based upon a transform of the input, rather than the input itself.

either case, it is this number—this transform—on which the identification or recognition logic operates.

The two schemes shown here are, perhaps, only different diagrams of the same functional blocks. In any case, it is the intent of this paper to discuss a class of preprocessing operations and the transforms which they produce. It is taken for granted that if a binary number exists in the machine which contains sufficient information to discriminate the pattern of the input, then logical operations can be formulated to produce as output a signal identifying the pattern presented. But the form of the transform function is important because it has a strong influence on the capacity of the memory or the complexity of the logic required for recognition.

The preprocessing includes, typically, an optical-to-digital transducer system, a magnetic reader, or an acoustic-to-digital transducer system. It may, also, include certain logical operations on the digital data. And, of course, it may use an analog, rather than digital techniques.

The Chrysler Concept

Figure 2 shows the system which is to be discussed. We are concerned with producing a transform; that is, a group of digital bits which can be uniquely associated with the character presented.

For some applications, it would be nice to associate the characters in Figure 3 as being alike and to produce the same set of binary bits as the transform for each of them.

More formally, it would be nice to have the machine identify a square form, regardless of its size, its orientation, or the location of its image on the retina.

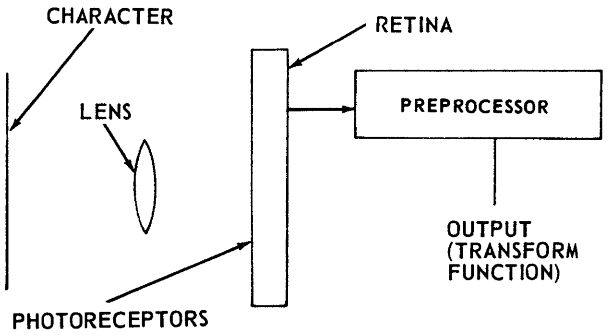


Figure 2. Schematic of generalized optical scanner.

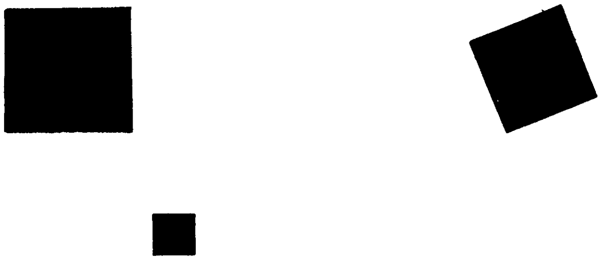
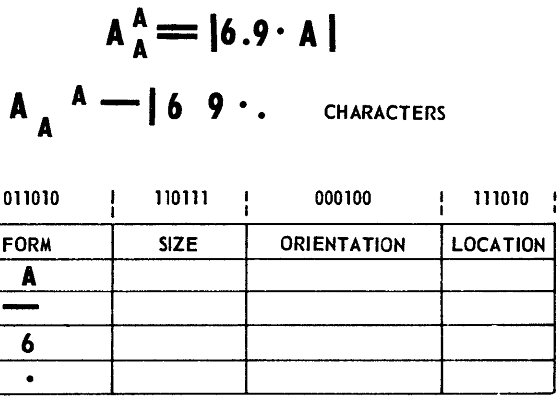


Figure 3. Typical characters which should be identified as similar by a form-perceiver which is not affected by size of orientation.

In other cases, however, these differences are as important as the form. If the machine scans the equation at the top of Figure 4, it will see 12 separate symbols, but these are made up of only 4 forms, (A, Bar, 6, and dot).



011010	110111	000100	111010
FORM	SIZE	ORIENTATION	LOCATION
A			
—			
6			
•			

Figure 4. A typical equation to be scanned, and a hypothetical transform in which disjoint sets of bits discriminate form, size, orientation and location of the image.

If we can generate a transform (shown conceptually as a binary number in Figure 4), in which certain digits are identified with shape or form, others with size, some with orientation, and still another set with location, the logical operations of recognition will be much simplified. The number of shapes which must be memorized by the machine will be substantially reduced. It is not the intent here to justify the desirability of having disjoint sets of bits, each set associated with some property of the character. There is enough literature on the subject already, both in the computer field and in the psychological analysis of human perception.

To a limited degree, the system to be described does just this. Of course, this is not the whole problem. A general purpose pattern recognizer may be confronted with Figure 5 and must recognize not only the similarities but also the differences between these sets of characters.

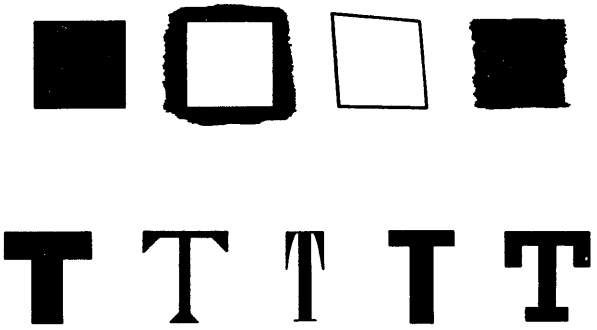


Figure 5. A typical problem for a general pattern recognizer which should discriminate both the similarities and the differences in the sets of figures.

But, even though it doesn't do everything that might be asked of it, there seems to be a measure of utility in the system shown in Figures 6 and 7. Here a part of the preprocessing is done optically. An optical wedge is rotated so that the image is caused to nutate on the retina. The wedge driver motor also drives a commutator which puts out timing signals. The circuitry is shown in Figure 7.

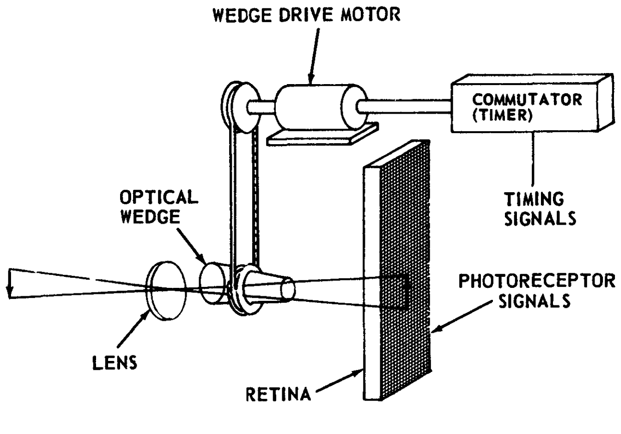


Figure 6. The conceptual arrangement of the Chrysler Optical Processing Scanner.

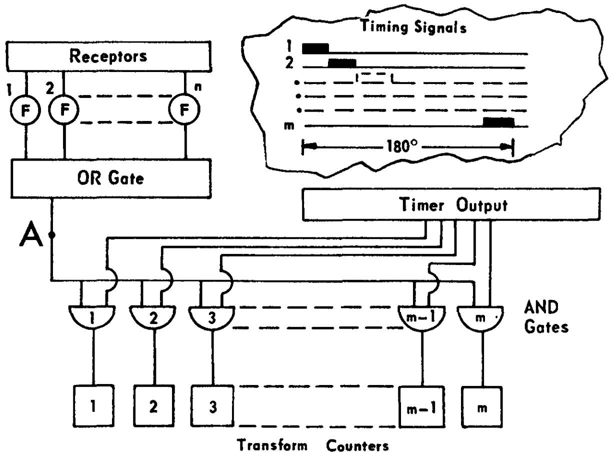
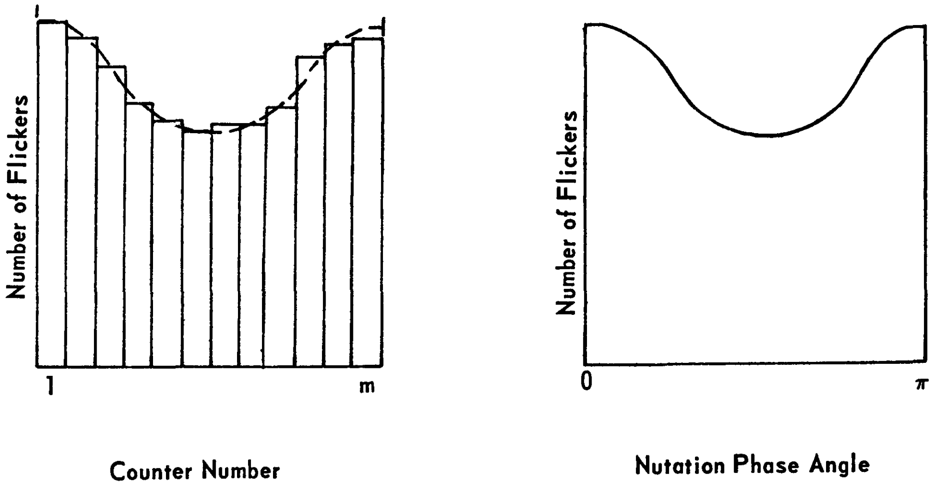


Figure 7. Schematic circuits of the Chyrsler Optical Processing Scanner.

Each of the n photo receptors in the retina is connected to a differencing circuit or "flicker filter," (Symbol F), which compares the signal now with what it was an instant ago and emits a pulse if they are different. These pulses are connected to one large bank of OR gates. The output at Point A, Figure 7, is one pulse every time any one of the receptors changes state. The number of receptors and duration of single pulses are such that the overlapping of signals from two receptors to form a single output pulse at "A" is statistically a rare event. The effect is ignored. The m timing signals direct traffic of these pulses so that, for the first increment of rotation of the wedge, the pulses are accumulated in Counter No. 1, the next increment of rotation reads pulses into Counter No. 2, and so on. If there are 180 counters, each will, after 180° of wedge rotation, have counted the flickers during 1 degree of travel. The counters are emptied every 180°.

The counter readings may be plotted as in Figure 8. The curve these points represent may properly be called a transform of the character which generated it. It is denoted as the T* transform.

Two kinds of differences exist between a real transform and an ideal one. In the left half of Figure 8A is shown the result of using an infinitely divisible retina and a finite number of counters. The curve is approximated by a step function, but no better approximation is possible without using smaller increments. In the right half of the same Figure, both a finite size and a number of retinal



A. The effect of a finite number of counters.

B. The effect shown in "A", together with the effect of a finite retinal matrix.

Figure 8. Real transforms (A) and idealized transform (B), generated by Chrysler Optical Processing Scanner.

elements and a finite number of counters are used. The approximation is somewhat more irregular.

In what follows the idealized transform in Figure 7B is discussed. It is assumed that there are infinitely many retinal elements and counters. The degree of approximation permitted in a real machine will depend on the job it has to do.

The T* transform in the counters may be cyclically shifted, as shown in Figure 9, until the smallest value lies in Counter No. 1. The result is denoted as the T_R* transform of the character. If after this operation.each counter is divided by the number in Counter No. 1, a $T*_N$ transform is produced.

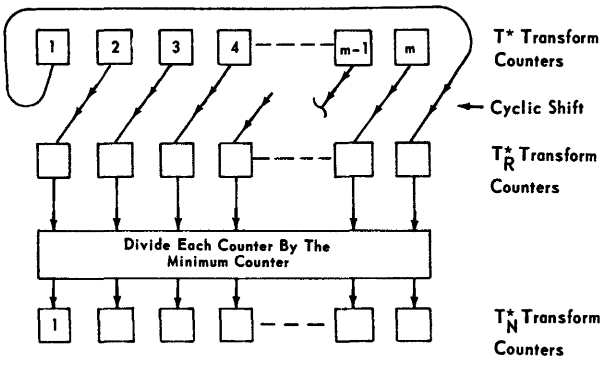

1
2
3
4
m-1
m
T* Transform Counters
Cyclic Shift
T^*_R Transform Counters
Divide Each Counter By The Minimum Counter
1
T^*_N Transform Counters

Figure 9. Shift-and-divide operations which generate the T_R* transform (invariant with respect to character rotation) and the T_N* transform (invariant with respect to size).

With this description of the system in mind, we can now begin to discuss its characteristics. The machine description above is entirely conceptual. It must be emphasized that such terms as wedges, counters, commutators, et al are only convenient terms for description; the hardware of a real scanner would be functionally the same, but physically far different.

Discriminability and Invariance

The transform of a circle is a horizontal line, (Figure 10). The rate at which the circle covers and uncovers the receptors as it nutates is constant.

The transform of a square is proportional to $/\sin \theta + \cos \theta /$, where θ is the nutation or phase angle; with a reference established by setting $\theta = 0$ at the time the pulses begin to accumulate in the first counter.

Since the transforms are different, at least these two characters can be discriminated.

Consider next the two characters in Figure 11. The transforms of these two characters are identical, except that one is out of phase with the other by an amount equal to the difference in angular orientation. But in the computer, the T_R* transforms will both have been shifted (by different amounts) until they have the same position in the counters.

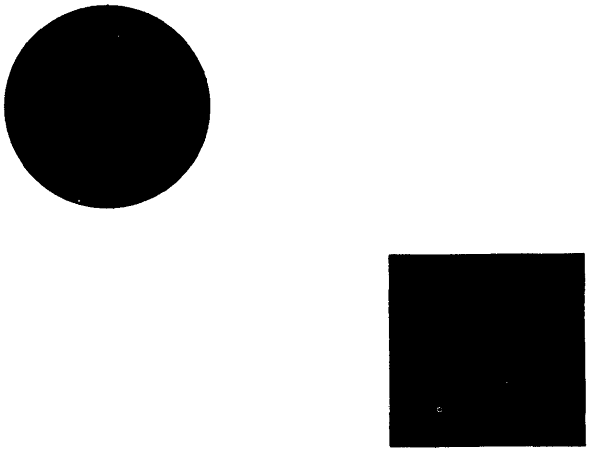


Figure 10. Two simple characters which are discriminated. The T* transform of a circle is a horizontal line; for a square, it is a (sin θ + cos θ).

Figure 11. Two characters having distinct T* transforms, but identical T^*_R transforms.

Therefore, the T_R* transforms of these two characters are identical. If we wish to compare the transforms with a number of prestored bits, one set of bits in the memory will match either character.

But if, in the shifting of the transforms in the counters, we count the number of shifts performed to bring the minimum counter reading into the first counter, the result is a number which gives a measure of the orientation of the character of the retina.

In Figure 12, one more variable is introduced—size. The larger figure will "sweep out"—i.e., cover and uncover—more receptors and produce more flickers than the smaller one. And the number of flickers generated will be linearly proportional to a linear dimension of the characters. But, in generating the normalized transform, T^*_N, every ordinate of the T_R* transform (or every counter reading) was divided by the minimum value. The result is that Counter No. 1, which holds the minimum value, reads "1", for both transforms—and the T_N* transforms of the two characters are identical.

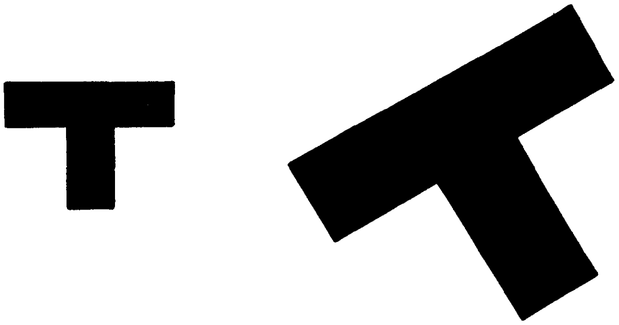


Figure 12. Two characters having distinct T^*_R transforms, but identical T^*_N transforms.

The information as to the size of the image, however, need not be lost. The reading in Counter No. 1 of the T^*_R transform gives a measure of the size of the character.

The foregoing results are stated more formally in the Appendix.

At this point, we may summarize the properties of the scanner, as in Figure 13. The form of a character generates a group of bits, T^*_N, which are identical for all sizes, orientations and locations on the retina. It is truly a measure of form or shape only, unaffected by the other three properties.

Form	Size	Orientation	Location
T^*_N	Counter #1 T^*_R	Shift Count T* to T^*_R	?

Figure 13. Summary of the specific transform bits associated with form, size, and orientation.

The orientation of the character is given by the count of the number of shifts to bring the minimum reading into Counter No. 1. This orientation has no meaning if we wish to compare the orientation of Character A with Character B. It is only meaningful in comparing the orientation of 2 A's, one of which can, of course, be a standard which is displayed erect and stored in the memory.

The size of a letter, measured by the entry in Counter No. 1 of the T^*_R transform likewise does not compare the size of an A with a B, except, perhaps, very approximately.

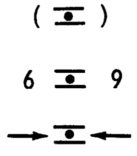
In this device, then, it appears that the perception of form is independent of the other parameters; the others are not, however, completely independent of form.

It is patently impossible for the systems described to provide a measure of the location of the image. The processing takes note of how many receptors flicker and at what time during the nutation cycle. Nowhere is there any information as to which receptors change state. This is not the only deficiency in the device.

Two characters, which are the same except for a 180° rotation, are not discriminable. This is evident from the fact that the transform is cyclic with a 180° period. For the same reason, we cannot distinguish a reorientation of the figure through 30° from one through 210°.

A second form of ambiguity arises because if two characters or parts of characters are scanned simultaneously, the transform is the sum of the transforms of the two when scanned separately. This source of ambiguity may also be ascribed to the inability of the machine to sense the location of an image, or part of an image, on the retina.

These two ambiguities are illustrated in the next two Figures. Figure 14 shows figures which are ambiguous due to 180° rotation—polar symmetry.



$$[T^*]_\phi \equiv [T^*]_\phi + 180$$

Symbol ≑ denotes two charactors which have equal transforms.

Figure 14. Examples of characters not discriminable (by the basic flicker-count circuits) because of polar symmetry.

Figure 15 shows figures which are ambiguous because they are each the sum of the same elements, the elements being located differently with respect to each other. An element is defined as a boundary between a black area and a white one.

A machine which cannot distinguish a 6 from a 9 is somewhat limited in its usefulness.

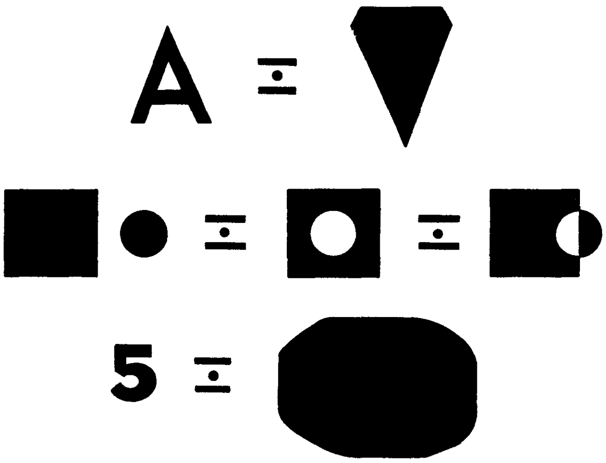


Figure 15. Examples of characters not discriminable (by the basic flicker-count circuits) because of failure of these circuits to provide image-location information.

The obvious question is how to add necessary circuits without losing the desirable characteristic that form, size and orientation are separately identified; and, also, without complicating the circuitry by trying to recognize which receptor is excited at a given time.

In a 360° cycle, the number of receptors turned on will always equal the number turned off. It is evident, with a little geometric reasoning, that each receptor which is affected at all must in 360° change state an even number of times. But, consider the characters in Figure 16. These two characters have identical transforms, but in one case there are two changes-of-state in each affected receptor and in the other there will be four changes-of-state in some receptors. A count of the number of receptors which change state four or more times becomes an additional

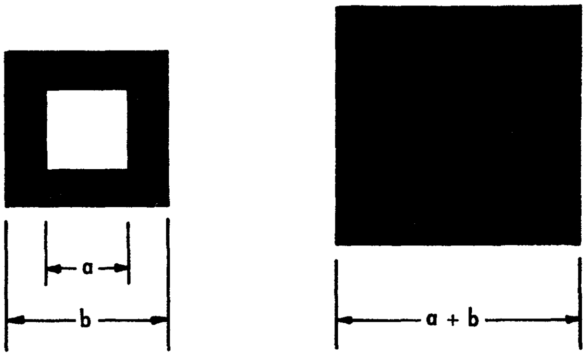


Figure 16. Examples of characters discriminable by counting receptors which have multiple changes-of-state.

transform and provides the information which will distinguish these two characters. This scheme is, of course, in addition to, rather than instead of, what has already been described.

There are other schemes—many of them—which will assist in the discrimination of characters. Some of these schemes are better at resolving one form of ambiguity, others are better for other characters. Two of these will be discussed.

Imagine a feed-back circuit associated with each receptor, such that when the receptor changes state its output is blocked for a short time interval. This has the effect of inhibiting the second of two flickers which occur too close together in a single receptor. Figure 17 illustrates the effect of this circuit on two otherwise ambiguous characters. The effect is to produce for the two characters transforms which are different. The flickers which would normally be produced by the segments shown dotted do not appear when the dotted segments follow too closely the adjacent line. It is somewhat as if fillets were added, but these fillets vary in size as the image nutates and only appear at all when the nutational phase angle is in the appropriate quadrants. It is, therefore, feasible in principle to utilize the transform as modified by this device instead of the ones previously discussed.

The next ambiguity-resolving scheme was derived from trying to make a 6 and a 9 fundamentally different by optical distortion. If the image formed on the retina results from reflection from a warped mirror, so as to produce distortion as in Figure 18, the polar symmetry disappears and distinguishable transforms results. However, the equivalent of optical distortion can also be produced as shown in Figure 19. The size and spacing of retinal elements is progressively diminished near the bottom of the retina. The effect is to increase the number of flickers produced by the lower half of the character exactly as if the image had been optically distorted as before. The transform can still be "normalized" as before to produce invariance with respect to size, location and orientation, although the computations to do so are more complex than the simple shift and divide operations described above.

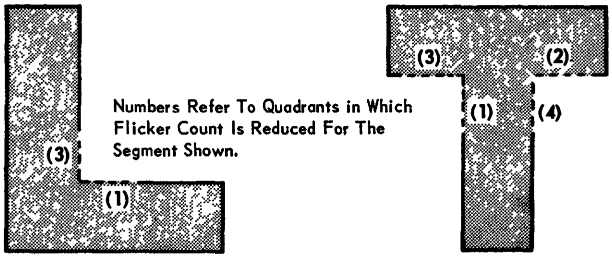

(3)
(1)
Numbers Refer To Quadrants in Which Flicker Count Is Reduced For The Segment Shown.
(3)
(2)
(1)
(4)

Figure 17. Examples of characters which are discriminable by circuits which inhibit the second two changes-of-state occurring too close together.

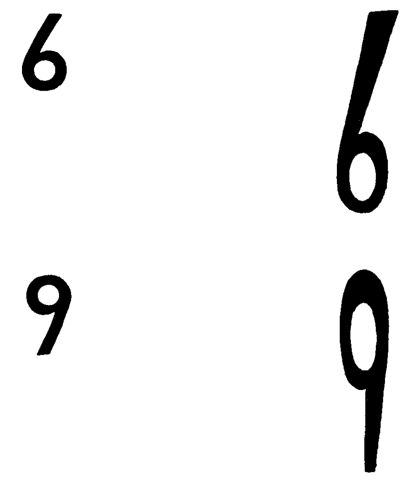


Figure 18. A (hypothetical) strategem for producing discriminable images from characters not discriminable by the basic flicker-count circuits by optical distortion.

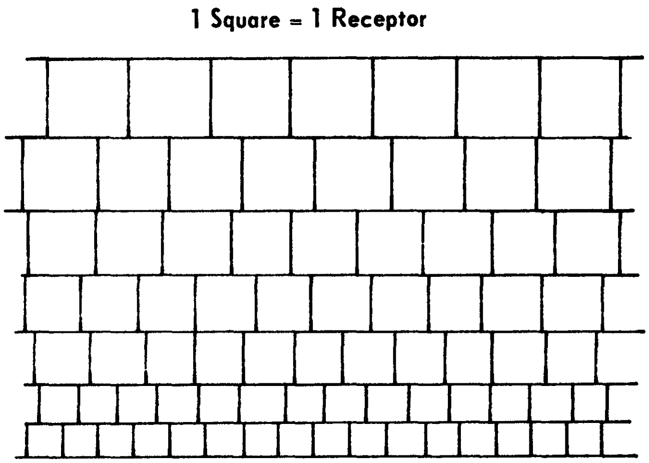


Figure 19. Schematic representation of a non-uniform retina which simulates electronically the effect of optical distortion.

This scheme suffers from the fact that the retina at the top must be fine enough to produce sufficiently smooth transforms, and finer still at the bottom. Consider the next modification in the sequence, (Figure 20).

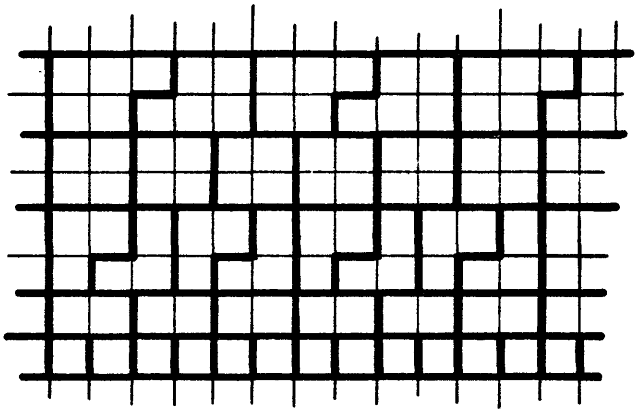


Figure 20. Schematic representation of a non-uniform grouping of retinal elements which produces the same result as the retina of Figure 19, but without variable size or spacing of retinal elements.

Here, a uniform retinal matrix is used and transforms are generated, as illustrated in Figures 6 and 7. In addition, taps from the receptor outputs are connected to additional circuits. Near the top, groups of receptors are grouped in these circuits so that they behave as single receptors. To do this, the circuits effectively pass the flicker pulse associated with the first of the group to change state and then inhibit the remainder of the group from emitting pulses. In these circuits, therefore, the group acts as a single receptor. The effect of grouping receptors in this way is the same as if the optically distorted 6 or 9 had been focused on the retina and the characters are, therefore discriminable.

The number of variations and additions to these concepts has not been exhausted in the descriptions above. For implementation in a generalized pattern-recognizing computer, several of these schemes may be combined.

Further discussion of the ambiguities resulting from the simple circuits and of means of resolving them is not fruitful without more rigorous mathematical formulation. It is possible, however, to list a few of the transform-generating schemes which may be useful for specific applications.

1. Count the total number of changes-of-state and read them into counters by the use of timing pulses. (This is the basic count-and-sort arrangement described above.)

2. Count and sort the changes-of-state while the image moves off the edge of the retina. Alternately, after 180° of motion, switch out or inactivate those receptors just ahead of the moving image so that, effectively, an "edge of the retina" is artificially produced.

3. Count and sort separately the number of receptors which change state two, four, six or more times.

4. Count the number of receptors illuminated (or dark) at some instant.

5. Number the rows of receptors, from I to U, and let each row generate a number of pulses r, such that r = f(U). Together, with normalizing circuits, this scheme provides one coordinate of the location of the image.

6. Number the column, as in 5, to obtain the other coordinate for the location of the image.

7. Count and sort only the first change-of-state for each receptor in each cycle.

Imperfect Patterns

Some comment on the uncertainty in the shape or form to be recognized is appropriate. The analysis has been based upon the assumption that characters to be recognized as identical were, in fact, congruent, pure black on a pure white background.

Figure 21 shows an enlargement of typical characters from 12 point type. The irregularities on a vertical outline will effectively increase the flicker count most during that portion of a cycle when the image is moving vertically. The result is to increase the flicker count—never to reduce it.

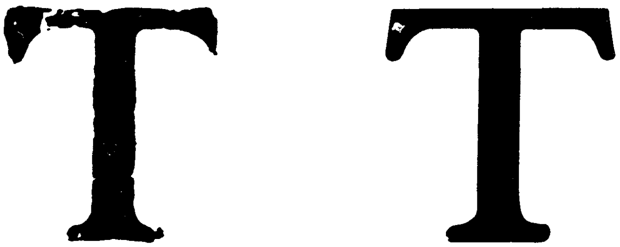


Figure 21. A typical enlarged character, originally set in 12 point type, and illustrating characteristic irregularity of outlines.

Several "hole filling" or contour smoothing techniques have been discussed in the literature. It appears possible that a relatively simple strategem can be effective in systems utilizing image nutation, as is described here. If the image is optically defocused, a gray outline is formed, like that of Figure 22. The lower figures represent the effect of equal blurring of the characters above. If these fuzzy characters were scanned, using autocorrelation techniques depending on which receptors were stimulated above a given threshold, the thresholds or sensitivities would have to be very carefully controlled. Otherwise, the low brightness gradient would magnify, rather than diminish, the irregularity of the border.

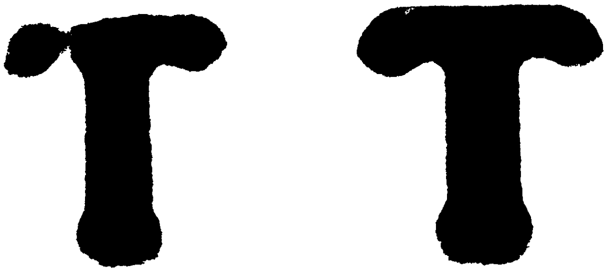


Figure 22. A pictorial description of the influence of defocusing or blurring on a typical and a perfect letter. The blurred images are more nearly alike.

However, consider an image moving normal to such a fuzzy edge. The receptors which are at the low end of the threshold tolerance would change state late, those at the high end early. Since we are simply counting the changes-of-state and don't care at all which receptors are involved, these effects tend statistically to cancel or "average out."

During that portion of the nutation cycle when the image moves parallel with a fuzzy edge, low and high threshold receptors are likewise less likely to undergo unwanted changes-of-state. This technique is, in fact, akin to certain hole filling schemes, except that the filling is accomplished by distributing light from a point to adjacent points, rather than by distributing calculated probabilities.

The blurring of the image may also be created by other means, as by superposing a high frequency vibration on the lens-object-image system, or by passing the image through a silk screen, or equivalent grid. This latter technique may be considered to be borrowed from the photographers who "soften" their portraits to remove harsh contrast.

Nutation and Other Image Motions

Mention should be made of the reasoning which led to the use of nutation as the image motion which is most effective. It is intuitively evident (and is simply derived mathematically) that line segments of characters produce little or no flickers and hence do not input a signature to the computer while their motion is parallel with the segment. To pick up lines of character in any orientation, there should be some interval during which the motion is normal to every line. Nutation is such a motion.

It may be more formally stated that no motion generates a transform having more information content than nutation. There are, however, many motions which do as well. Any of these may be resolved into a nutation and some superposed motion. Consider, for instance, the superposition of a lateral translation and nutation, which produces a net motion, such as is shown in Figure 23, for each point in the image.

The translation adds or subtracts from the lateral (horizontal) component of nutation velocity. So long as the superposed lateral velocity V is always less than the velocity of the nutation, $R\omega$, the image vector will, at some time during its cycle assume every direction from a $+\infty$ to a $-\infty$ slope. Such a motion produced no loss of information. So long as the superposed motion is known, the flickers it produces or inhibits do not subtract from or add to the information content of the transform. When such a superposed image movement exists, it may be found desirable to generate timing pulses of unequal duration.

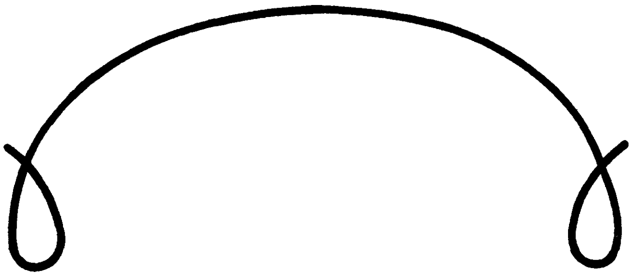


Figure 23. Sketch of the cycloidal path of an image-point over a retina which results from superimposing nutation and lateral translation.

It is, therefore, nearly as easy for the scanner to "read" a moving character as a fixed one. The capability for discrimination of images without bringing them to rest in front of the lens can be a useful asset for some applications.

The technique of moving the image may be thought of as a sort of time-sharing of circuitry in which the switching is accomplished optically. Instead of connecting "edge autocorrelation" circuits to various receptors in sequence, the image is "connected" sequentially with different receptors in sequence. Thought of in this light, the optical wedge replaces the switching circuitry.

Mirrors and Fiber Optics

There are a few further strategems which may be used to reduce still further the number of logic blocks and receptors required. Not all of these can necessarily be used simultaneously. The choice of which to use depends upon the application.

In Figure 24 is shown a character being nutated over a semi-circular retinal array of receptors. A system of mirrors is so arranged that as the image leaves the retina on top, its reflection enters below. The number of receptors and the circuits associated with each is halved.

In Figure 25 is shown a different arrangement. The matrix of retinal elemqnts is composed of the ends of optical fibers. The other end of the fibers carries light to the photo receptors proper. All of the fibers bearing the same number communicate to the same receptor. In the scheme shown, 459 fiber ends are connected to 91 photo multiplier tubes.

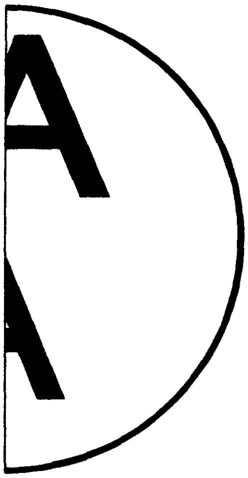


Figure 24. Sketch of the image formed when a mirror system is used to bring a reflection onto the bottom of a retina as it leaves the top so that fewer receptors are required.

If the largest character to be scanned can be completely enclosed within a circle inscribed in one of the large bold-face hexagons, no loss of information will result. It is evident that with the size restriction it will never occur that one part of a character moves to turn on or illuminate a tube which is already on, or vice versa.

It is probably true that the highly organized connectivity shown, where the optical fibers are accurately grouped, is not really necessary. Within limits, the fiber ends could be connected "at random" to the photo receptors. In this case, if the multiple connections are carried too far, large sample statistics govern, and the instantaneous "sample" of elements turned on and off will always be the same as that of the population. The "population" of flickering cells have an average of zero, since every receptor turned on will be turned off during a cycle. There will, therefore, be no net change of illumination and no flickers. But if the statistical scheme used to define "random connectivity" is properly selected, random connections may be used with only small degradation of information content in the transform.

Composite Images—Words vs Letters

Some interesting properties of this scheme arise from the theorem, stated earlier, that the transform of a character is the sum of the transforms of its parts, taken as if they were separately scanned. Suppose the image focused on the retina consists of two letters, A and B. The transform will then be the sum of their separate transforms. Assume, for the moment, that there are a sufficiency of the ambiguity-resolving circuits to discriminate each letter separately. If there are no two or more characters, the sum of whose transforms is identical with that of a third one, then the transform of AB may be uniquely associated with these two letters, but not their order.

This property opens up the possibility of reading words rather than letters. Obviously, if the output is an electric typewriter, no advantage accrues to the system. If, however,

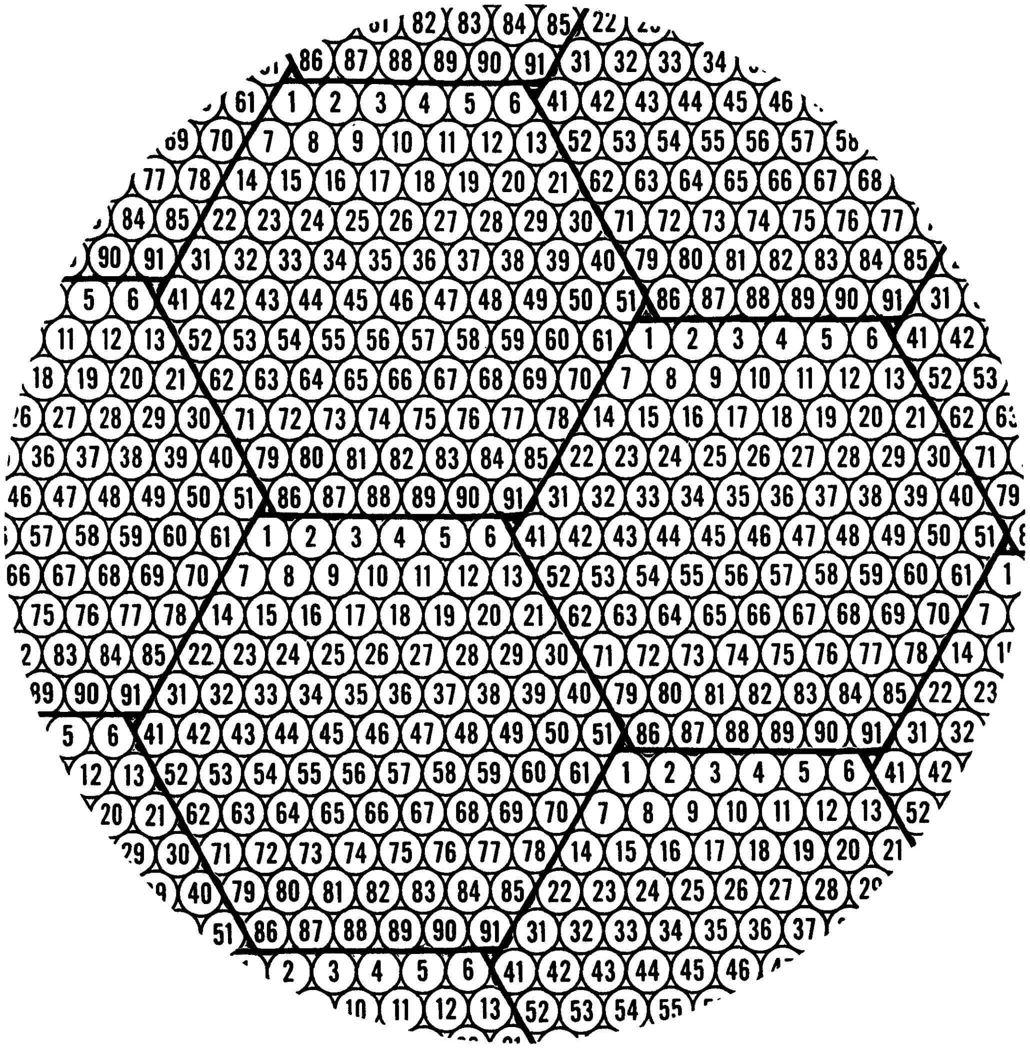


Figure 25. Another scheme for diminishing the number of photo receptors required by connecting the optical fibers (shown as circles) to photo receptors (numbered).

the output is to a language-translating computer, which must have a dictionary stored in its memory, anyway, the scanning of words rather than letters may be useful.

Without going into detail, this same theorem may be said to offer the possibility of analyzing a character into its separate parts. The analysis will, in general, not yield completely unambiguous results. However, if one leg of a character is missing, it is reasonable to consider predicting probabilistically the intended character and the rechecking by other techniques.

Scanner Characteristics Summary

The Chrysler Optical Processing System exhibits the following characteristic behavior:

1. There is generated within the logic a transform of the image in which character form, size, orientation, and location are represented by disjoint sets of bits.
2. Image movement may, if convenient, be superposed on nutation—characters need not be brought to rest to be read.
3. Character imperfections may be effectively smoothed out by blurring, without complex hole-filling circuitry or precise biasing of receptor thresholds.
4. Accurate servo-centering of images on the retina is unnecessary.
5. Words may be read instead of letters; characters may be analyzed into their elements.

6. Various means, both optical and electronic, may be used to reduce the number of retinal elements and logic blocks required to obtain the effect associated with a fine-grained retinal matrix.

Each of the six statements above should include the phrase, "—subject to limitations as discussed in the text."

The Psycho-Physiological Parallel

The description of the scanner and a survey of its capabilities, both immediate and potential, is complete. It would be amiss to omit mention of the striking, though possibly superficial, similarities which exist between this scanner and the processes, events and components of the human visual system. There are several of these psycho-physiological analogies.

The image formed on the human retina is not still, even when the observer attempts to fixate a particular point. The motion of the fixation point on the retina has been measured and looks something like that of Figure 26. The movements take the form of sharp, fast flicks, with a slower drift between flicks, rather than the smooth nutation which is convenient for the machine. Superposed on this is a high frequency tremor.

It has been reasoned that the nutation of the image is an optimum image-movement tactic. As a speculation, it would be of interest to examine how well these drifts and jerks approximate nutation. The obvious first choice for a straight-line approximation to a circular motion is a regular polygon.

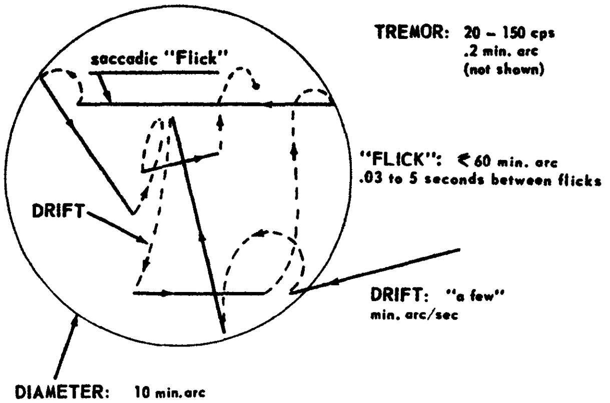


Figure 26. Typical movement of the fixation-point of the human eye, consisting of a high frequency tremor, a series of jerks, and a saccadic jump to compensate for the cumulative drift due to jerk.

In the inaminate scanner, we must accumulate the transform for whatever time is required to nutate 180° before the transform is complete. If after a transform is generated the machine memory could only be exposed to two small parts of it, what parts would yield the most information? Vertical motion yields the most information about horizontal lines and vice versa. It would be to our advantage to select the two parts of the transform so that they represent the count of changes-of-state of receptors at two instants when the image motions are perpendicular to each other. Or, conversely, after the image has moved vertically, and the changes-of-state recorded, the next motion should not be nearly-vertical since this adds less to the information already on hand than if the second movement is approximately horizontal. Suppose, therefore, that we retain the regular polygon as an approximation, but traverse the legs out of order. Furthermore, movement in either direction, A to B, or B to A, produces basically the same set of changes-of-state.

One further charge in the "regular polygon" idea is required. If the polygon is exactly regular, and of n sides, the directions of movement on the n + 1st movement will exactly coincide with the first. It would be better if we chose a figure whose sides progress angularly.

In summary—

1. Select a regular polygon of n sides (Figure 27A).
2. Adjust the angles slightly so that no two sides of any group will be exactly parallel (Figure 27B).
3. Rearrange the sides out of order so that the angle between adjacent sides is approximately 90° (Figure 27C).
4. At random, reverse the direction of the sides, taking BA instead of AB, for instance (Figure 27D).

With these considerations in mind, the movements of the image in the eye seem not too dissimilar to what we would build into our engineering artifact if mechanical convenience were not a factor. Straight-line motions which start and stop are just not as easy or as cheap to make out of metal as they are of muscle. It may be noted that the machine described fails to provide "location" information. It can, of course, be supplied by additional circuits which have not been described.

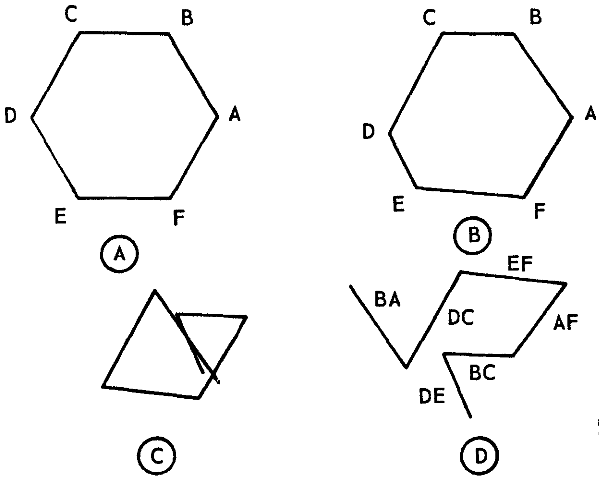


Figure 27. Various steps in the deduction of an optimum image-movement for a mechanical scanner constrained to utilize straight-line image movements.

If image signals are generated in the human retina during both the "flick" and the drift, it is not unreasonable to suppose that the analysis of the signals generated by two different motions are analyzed by two different schemes. More specifically, can one of these motions be considered to generate signals which are analyzed in the brain without attention to which receptor is active while the other motion is associated with the location of image parts?

It would be interesting to follow the speculation a little further and see what sort of eye movements are observed if the subject is exposed to a visual field composed of heavy vertical lines and nothing else.

Vertical eye movements in such a field will not produce any changes-of-state and hence produce no information. This is probably not strictly true, except in an optically perfect eye without aberration. If the principles discussed in this paper are truly analogous to human vision, and if there is feedback in the system, then a vertically-ruled field should induce more horizontal jumps in the eye scan.

The visual process of observing a moving object superposes another motion on the image motion due to scanning. In the inaminate device, information is lost if the superposed translational velocity exceeds the velocity due to nutation. In a speculative sense, it might be interesting to try to discover in a human eye what rate of movement of an object is associated with a loss of information as to its form, and to relate this to the scanning speed.

Some of the other analogies between machine and eye are certainly obvious to the reader. Both are sensitive to changes of illumination rather than illumination directly.

The machine use of a refractory period which inhibits the second of two pulses too close together has a direct analog in the refractory period of neurons. The grouping of local clusters of photo receptors to a single channel output was discussed as a means of producing the equivalent of optically distorted images. This is a reflection of the multiple connections of rods to a single neural path. Neither of these two phenomena has, to the author's knowledge, been associated previously with form perception, yet they appear, in the inaminate machine at least to be well-suited to the task.

The circuits which count the receptors which flicker more than twice requires what in the computer field is termed a flip-flop; and in the field of psychology a form of "temporal summation."

The machine requires a number of counter registers in which to store the flicker count. It is inappropriate to discuss such counters in the physiological system. In the machine, signals travel to the counters at the speed of light and are held in the counters as long as needed, as in Figure 28. In the brain, signals travel slower. We may surmise that at any instant the various counters have their counterpart in the various links in the neural

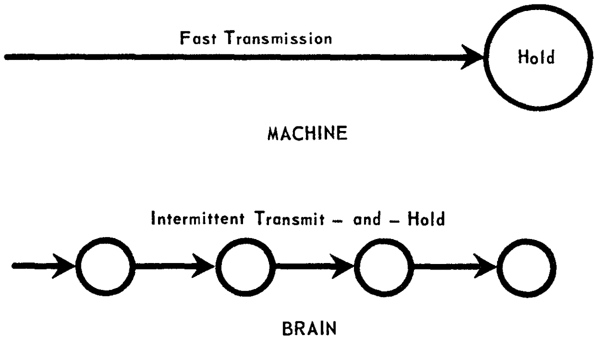


Figure 28. Comparison of the machine and brain storage of information showing the possibility that information in the brain may be "stored" so as to be available to have logical operations performed on it while in transit from neuron to neuron. This possibility is lacking or difficult to exploit in a computer because of the speed of transmission.

pathway. That is, the several counters are analogous to the several stations along an appropriate set of neural paths which carry the signal at a given instant.

These signals may reverberate back and forth, perhaps undergoing phase shifts and other normalizing operations in the process.

Figure 7 showed the mechanical scheme by which the count of changes-of-state are switched or directed into the various counters in sequence by the timing pulses from the commutator. Figure 29 shows the same schematic in which the inhibitory or excitatory effects of one set of synopses accomplish the timing function and direct traffic for a set of receptor outputs. There is little or no direct evidence that some of the neural pulses are image signals and others are timing signals related to the scanning motion of the eye movements, but there does not appear to be any direct evidence that such a concept is impossible. The possibility also exists that a given axon, synapse or receptor acts sometimes as a timer and sometimes as a signal detector.

Conclusion

As a peroration, certain rather abstract observations must be brought forward. The neuro-visual process may be rather vaguely defined as a "mapping" of certain changes-of-state of the retinal receptors into the brain. This mapping process is simply another way of stating that a "transform" is generated in the brain as a result of retinal activity. The map or transform may be either static or dynamic; that is, it may be defined by specifying either the state of a set of neurons, or a sequence of changes-of-state. In either case, there is a powerful appeal, both intuitively and logically, in a system where the transform of two images is the sum of their individual transforms. Such a scheme is, for different reasons, appealing to the computer engineer and the psychologist.

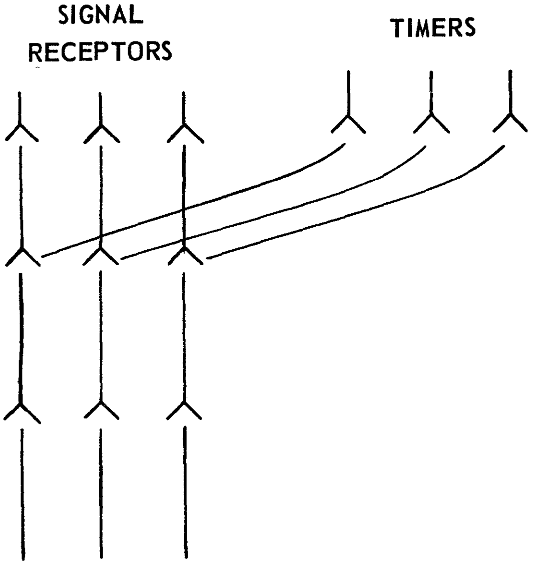


Figure 29. Conceptual sketch of neural circuits in which some signals are analogous to the commutator timing pulses of the Chrysler Optical Processing Scanner and others are purely image-signals.

The details of machine circuitry discussed are not likely to carry over entirely intact their usefulness into the psychological field. The machine principles may be described:

1. A moving image on a retina can provide much information about form, even with circuits which do not distinguish one receptor from another.
2. More information about form can be developed in the scanner if the direction of the scanning motion associated in time with change-of-state of a receptor is available in the form of timing signals.
3. Means can be conceived and circuits defined for form discrimination in which the transform identified is the sum of the transforms of its separate parts.

Stated in these terms, the principles on which the machine operates may be equally interesting to the psychologist.

This paper does not claim to present a new psycho-physiological theory of form perception. The description of the machine is a sequence of declarative sentences—the psychological discussion of a series of interrogative ones. The questions it asks may stir up a fruitful train of thought, but the answers are likely to go far afield from the present simple concept.

APPENDIX

Certain of the relationships between characters and the transforms they generate are repeated here in a form more suitable for mathematic analysis than oral presentation.

1. Let a character be defined as the set of points contained within a finite number

of closed boundary lines and distinguished from points not in the set. If points within the character are distinguished by being black and those outside are white, the character is positive; if the character is white on black, it is negative.

2. A positive character is solid if a closed figure can be drawn which contains all the black points and no white ones; otherwise, it is hollow.
3. A character is rectilinear if all of its boundary lines are straight. It is curvilinear if one or more of its bounds are curved.
4. A rectilinear character is convex if all of its internal angles are $\leq \pi$; otherwise, it is concave.
5. A curvilinear character is convex if all of the internal angles between its rectilinear elements are $\leq \pi$ and if no tangent can be drawn to its curbed bounds which intersects the character.
6. An element of a character is any segment of its boundary (and which, therefore, has white points on one side and black ones on the other).
7. The T* transform of an element of a character which is nutated with radius R and has an instantaneous position, as indicated by θ (Figure 30), is the rate at which the element sweeps out area, expressed as a function of R, θ, and the dimensions of the element. Specifically, by definition

$$T^*(E) = S\,R\,d\theta/dt \qquad (1)$$

where S = the projected length of E on a line parallel with the instantaneous radius R.

Letting θ = wt

gives

$$T^*(E) = R\,w\,S \qquad (2)$$

8. If E_1 and E_2 are congruent elements which can be superposed by pure translation without rotation, then

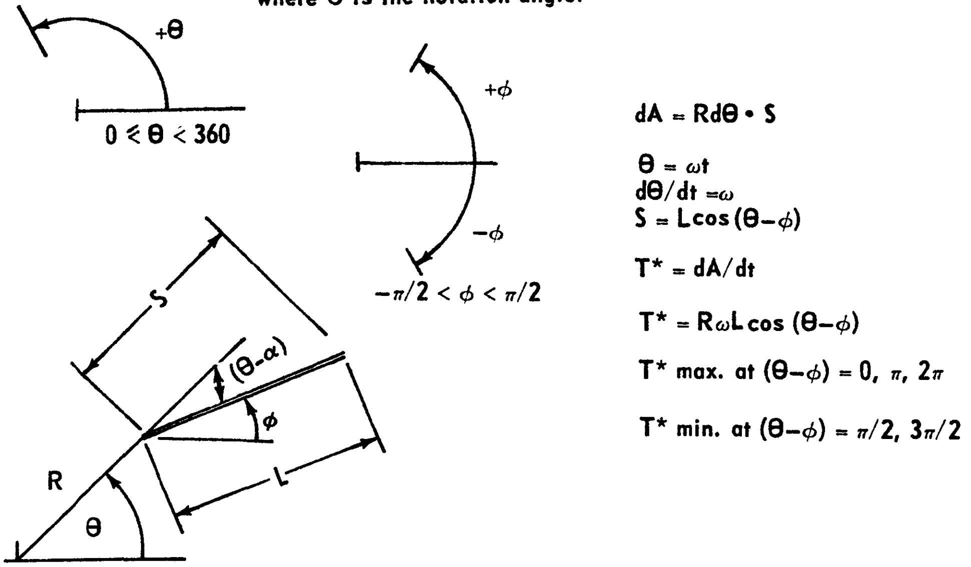
$$T^*(E_1) = T^*(E_2) \qquad (3)$$

9. The transform of a character C is the sum of the transforms of its elements.

$$T^*(C) = T^*(E_1) + T^*(E_2) + \ldots + T^*(E_n) \qquad (4)$$

10. The transform of the combination of two elements is the sum of their separate transforms.

$$T^*(E_1 + E_2) = T^*(E_1) + T^*(E_2) \qquad (5)$$


T* transform of a linear element of length L inclined at an angle ϕ to the line $\Theta = \Theta_0$ where Θ is the nutation angle.

+Θ
$0 \leqslant \Theta < 360$
+φ
−φ
$-\pi/2 < \phi < \pi/2$
S
(Θ-α)
φ
L
R
Θ
$dA = Rd\Theta \cdot S$
$\Theta = \omega t$
$d\Theta/dt = \omega$
$S = L\cos(\Theta - \phi)$
$T^* = dA/dt$
$T^* = R\omega L\cos(\Theta - \phi)$
T^* max. at $(\Theta - \phi) = 0, \pi, 2\pi$
T^* min. at $(\Theta - \phi) = \pi/2, 3\pi/2$

Figure 30. Notation and derivation of T* transform for a straight-line element.

The transform (2) may be written

$$T^* (E) = R\, w\, f\, (\theta, di, \phi) \qquad (6)$$

where θ is the nutation angle measured from a fixed arbitrary reference

di are the dimensions of the element

ϕ is the angle between the reference axis θ = o and any dimension of the character and which, therefore, specifies the angular orientation of the character.

11. If two characters are congruent, their T* transforms can be made equal by rotation of the reference axis for θ = o through an angle $|\phi_1 - \phi_2|$. Hence, if the reference axis for θ = o is taken at the same angular orientation with respect to ϕ, a new transform T_R^* is defined which is independent of character orientation.
12. If two characters C_1 and C_2 are congruent,

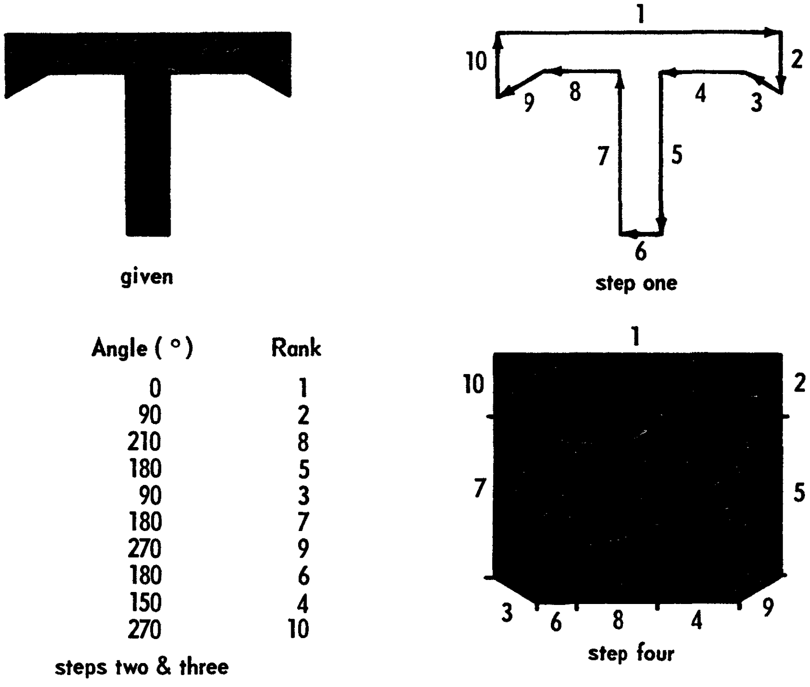
$$T_R^* (C_1) = T_R^* (C_2) \qquad (7)$$

13. There corresponds to every solid convex character one and only one transform T_R^*.
14. Every hollow character is concave.
15. For every concave character, there corresponds one and only one solid convex character which has the same transform T*.
16. The family of concave characters, all of which correspond to the same solid convex character, are called an ambiguous family. An ambiguous family are denoted by primes with the same base and the unique solid convex character by the unprimed notation. Hence,

$$T_R^* (C_1^{\,1}) = T^* (C_2^{\,1}) = T^* (C) \qquad (8)$$

C is called the unique character of the family $C_i^{\,1}$.

17. An algorithm for determining the unique member of a rectilinear family for which one concave member is given is as follows (Figure 31).
    1. Number each line of the boundary and add arrowheads in sequence around the boundary.
    2. Tabulate the angle each vector makes with the first one.
    3. Retabulate the bounding vectors, ranking them in increasing order of their corresponding angles.



given

step one

Line	Angle (°)	Rank
1	0	1
2	90	2
3	210	8
4	180	5
5	90	3
6	180	7
7	270	9
8	180	6
9	150	4
10	270	10

steps two & three

step four

Figure 31. Example of generating the convex solid (unique) member of an ambiguous family given one concave member.

4. Redraw the figure, using the same vectors, head to tail, but taking them in the order in which they were ranked in Step 3.

18. The algorithm for determining the unique member of a rectilinear family for which one hollow member is given is the same as in 17 above. Number and rank the vectors in both the exterior and the interior boundary, as in Figure 32.

19. If two characters or elements have polar symmetry with respect to each other, their T* transforms are identical.

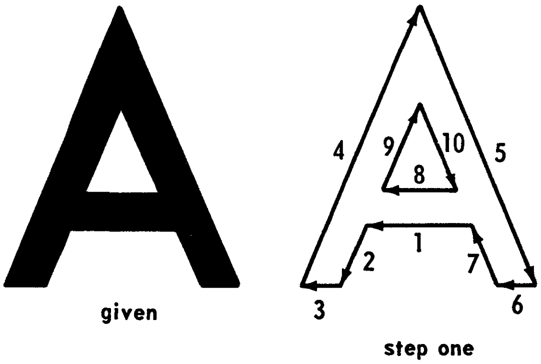
$$T^* (C)|_{\theta} = T^* (C)|_{\theta+\pi}$$

20. Two curvilinear elements of a character are parallel if their chords are parallel, taking the chords as vectors having the same sense as the vectors of the curvilinear elements.

21. Given a curvilinear element E_1 described by $y = f_1 (x)$ and having a derivation $y' = f'_1 (x)$ which can be solved for x to give $x = \phi_1 (y')$ and given, also, a second parallel element characterized by $y = f_2 (x)$; $y' = f'_2 (x)$ and $x = \phi_2 (y')$, then a third element E_3 defined by the equation $x = \phi_1 (y') + \phi_2 (y')$ has the property that

$$T^* (E_3) = T^* (E_1) + T^* (E_2)$$

22. An algorithm for finding the unique member of an ambiguous family, when one curvilinear concave member is given is as follows (Figure 33).
    1. Add arrowheads and number the elements as in 17 and 18. In dividing up the boundary into elements,
        (a) Each straight line segment is an element.
        (b) Each segment of a curved line is an element where the segment ends occur: (1) where a straight line meets a curve; (2) where two curved lines meet with a discontinuous first derivative; (3) at a point of tangency of a curved line with a tangent parallel to any rectilinear element having the same sense as the curve; (4) where a point is required to divide a curve into an element parallel with another and a non-parallel element.
    2. Where two or more curvilinear elements are parallel with each other, eliminate them and substitute a third element which has an identical transform, using 21.
    3. Rank the elements by their angles, as in 17, using the chord of an

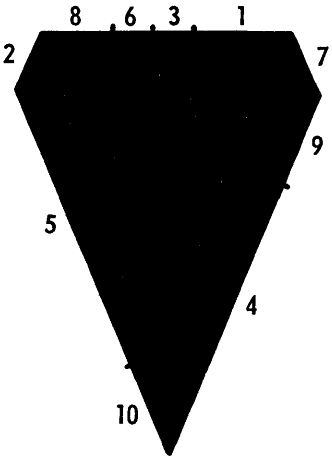


given

step one

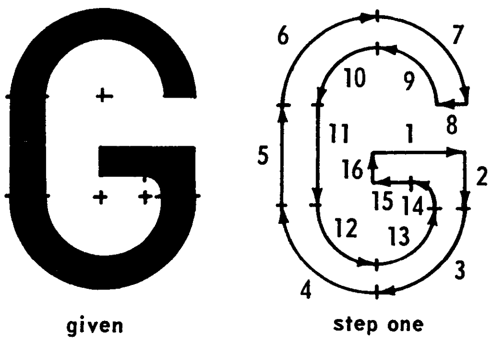
Line	Angle (°)	Rank	Line	Angle (°)	Rank
1	0	1	6	0	3
2	67 ½	5	7	292 ½	10
3	0	2	8	0	4
4	247 ½	8	9	247 ½	9
5	112 ½	6	10	112 ½	7

steps two & three



step four

Figure 32. Example of generating the unique member of an ambiguous family given one hollow member.



Line	Angle (°)	Rank	Line	Angle (°)	Rank
1	0	1	9	225	8b
2	90	3	10	135	5b
3	135	5a	11	90	4
4	225	8a	12	45	2b
5	270	9	13	315	11b
6	315	11a	14	225	8c
7	45	2a	15	180	7
8	180	6	16	270	10

steps two & three

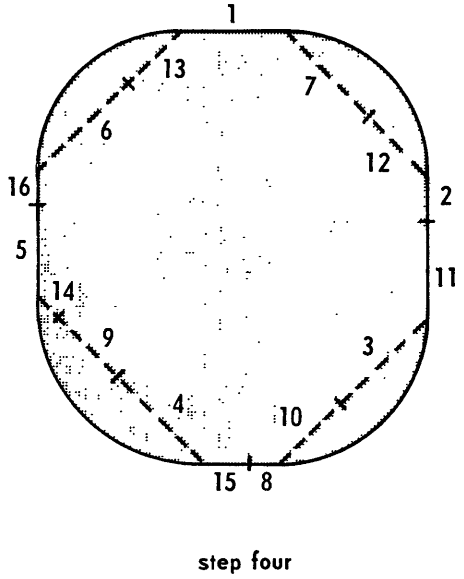


Figure 33. Example of generating the unique member of an ambiguous family given a concave curvilinear member.

element to determine its angular orientation.

4. Proceed, as in 17, to draw the figure which is the unique member of the family.

23. No motion of the image of a character over a retina can provide more information about the character than nutation; nutation defines completely the unique member of the family.

24. Any motion which translates the image without rotation so that a tangent can be drawn to the hodograph of a point having any slope from -∞ to +∞ will provide as much information as nutation and will define exactly the unique character of a family.

25. Specifically, a linear translation, such as would be produced by moving characters in a straight line past the lens, may be superposed upon the nutation and so long as the net nutation velocity $R\,\omega > V$, the linear velocity, it will identify the unique member of the family.

26. The unique member of a family is completely identified by 180° of nutation.

27. If two characters C_1 and C_2 are similar, and two corresponding sides have the ratio 1: a respectively, then

$$a\,T^*_R(C_1) = T^*_R(C_2) \qquad (9)$$

and the corresponding transforms are proportional.

28. Define a normalized transform T^*_N by

$$T_N^{1*} = T^*_R/[T^*_R] = \theta_o \qquad (10)$$

In which each value of the transform is divided by its value at an arbitrary nutation angle θ_0.

29. Then, if two characters C_1 and C_2 are similar,

$$T^*_N\,(C_1) = T^*_N\,(C_2) \qquad (11)$$

BIBLIOGRAPHY

1. Bledsoe, W. W., Bomba, J. S., Browning, I., Evey, R. J., Kirsch, R. A., Mattson, R. L., Minsky, M., Neisser, U., and Selfridge, O. G., Discussion of Problems in Pattern Recognition, 1959 Proceedings of the Eastern Joint Computer Conference, pp. 233-37.
2. Bledsoe, W. W., and Browning, I., Sandia Corporation, Albuquerque, New Mexico, Pattern Recognition and Reading by Machine, 1959 Proceedings of the Eastern Joint Computer Conference, pp. 225-232.
3. Bomba, J. S., Bell Telephone Laboratories, Inc., Alpha-Numeric Character Recognition Using Local Operations, 1959 Proceedings of the Eastern Joint Computer Conference, pp. 218-224.
4. Brown, Laurence R., Briggs Associates, Inc., Nonscanning Character Reader Uses Coded Wafer, Electronics Magazine. November 25, 1960, pp. 115-17.
5. Crescitelli, F. G., Physiology of Vision, Annual Review of Physiology. Volume 22, 1960.
6. Ditchburn, R. W., Eye-Movements in Relation to Perception of Colour, Journal, Physiology, 145. London, 1959.
7. Gill, Arthur, Minimum-Scan Pattern Recognition, IRE Transactions on Information Theory. June, 1959, pp. 52-58.
8. Greanias, E. C., and Hill, Y. M., International Business Machines Corporation, Endicott, New York, Considerations in the Design of Character Recognition Devices.
9. Harmon, Leon D., Bell Telephone Laboratories, Inc., Line-Drawing Pattern Recognizer, Electronics Magazine. September 2, 1960, pp. 39-43.
10. Horne, E. Porter, and Whitcomb, Milton A., Editors, Vision Research Reports. Washington, D. C.: National Academy of Sciences—National Research Council, Publication 835, 1960.
11. Kirsch, R. A., Cahn, L., and Urban, G. H., National Bureau of Standards, Washington, D. C., Experiments in Processing Pictorial Information with a Digital Computer, Proceedings of the Eastern Computer Conference, pp. 221-29.
12. Rosenblat, Frank, Cornell Aeronautical Laboratory, Inc., The Perceptron, A Theory of Statistical Separability in Cognitive Systems (Project PARA). Report No. VG-1196-G-1, January, 1958.
13. Scott, Bowman, and Curry, Peter A. M., Automatic Printed Character Reading, Journal of the SMPTE, Volume 68. April 1959, pp. 240-41.
14. Stevens, S. S., Editor, Handbook of Experimental Psychology. New York: John Wiley & Sons, Inc., 1951.
15. Wiener, Norbert, Cybernetics. New York: John Wiley & Sons, Inc., 1948.
16. Wulfeck, Joseph W., and Taylor, John H., Editors, Form Discrimination as Related to Military Problems. Washington, D. C.: National Academy of Sciences—National Research Council, Publication 561, 1957.
17. Yovits, Marshall C., Office of Naval Research, and Cameron, Scott, Armour Research Foundation, Self-Organizing Systems, Proceedings of an Interdisciplinary Conference, 5 and 6 May 1959. New York: Pergamon Press, Symposium Publications Division, 1960.

TECHNIQUES FOR THE USE OF THE DIGITAL COMPUTER AS AN AID IN THE DIAGNOSIS OF HEART DISEASE*

C. A. Steinberg
Department of Medical and Biological Physics
Airborne Instruments Laboratory
Deer Park, Long Island, New York

W. E. Tolles
Department of Medical and Biological Physics
Airborne Instruments Laboratory
Deer Park, Long Island, New York

A. H. Freiman, M.D.
Cornell University Medical College
Associate, Sloan-Kettering Institute
Clinical Assistant, Memorial Hospital
New York City, New York

Sidney Abraham
Instrumentation Unit, Heart Disease Control Program
Division of Chronic Diseases, Public Health Service
U. S. Department of Health, Education and Welfare
Washington 25, D. C.
and
C. A. Caceres, M.D.
Instrumentation Unit, Heart Disease Control Program
Division of Chronic Diseases, Public Health Service
U. S. Department of Health, Education and Welfare
Associate in Medicine, George Washington University Hospital
Washington, D. C.

The rapid computational capabilities and large storage capacity of the digital computer can provide the physician with a powerful tool for diagnostic procedures. Numerous techniques are available that can be used in attempts to use the digital computer as an aid in diagnosis [1]. For this reason, a study in the use of a general-purpose digital computer in analyzing physiological waveforms of the heart and their relationship to cardiovascular pathology has been undertaken, and a pattern recognition program for automatically recognizing clinically useful parameters in the electrocardiogram (ECG) has been developed. The techniques presented are components of a system that can be used as an automated aid for the physician in his diagnostic process [2].

I. COMPUTER CLASSIFICATION OF DATA

A program based on multi-dimensional probability density functions was used to

*Portions of this work were performed as part of the U.S.P.H.S. Contracts Number SAph-75508 and Number SAph-70926.

classify normal and pathological subjects [3, 4, 5]. The basis of this technique is similar to that used by the cardiologist in the diagnosis of heart disease. The cardiologist records physiological signatures related to the function of the heart. He studies the resulting waveform, recognizes and measures the important parameters in the waveform, and then compares the measured parameters with these he has memorized for normal and pathological subjects. By this comparison he makes a "diagnosis." The computer program is based on similar logic, but substitutes the probability density function for the cardiologist's experience. Measurements of the important parameters are programmed to characterize normal and pathological subjects and then to classify unknown subjects with respect to the previously defined characterization. An example of the use of this technique is given below.

A. Input Data

Simultaneous curves from four arbitrarily selected electrophysiological signals were recorded. These signals were the electrocardiogram (Lead V5), phonocardiogram (mitral area), ballistocardiogram (acceleration), and arterial pulse (radial). Clinically important parameters from these curves were measured manually.

The electrocardiogram (ECG) is a measurement of the electrical potential generated by the heart during the depolarization and repolarization of the heart muscle as measured from the surface of the body. There are 12 leads customarily used in clinical practice to measure the potentials. Each of these leads measures the heart potentials from different vantage points. In each lead the P wave represents atrial depolarization, the QRS complex represents ventricular depolarization, and the T wave represents ventricular repolarization.

The phonocardiogram (PCG) is the measurement of the sounds caused by the mechanical activity of the values in the heart and movement of blood within the heart. A microphone strapped to a specified area of the chest surface is the usual transducer for measuring the PCG. There are normally two major sounds recorded with the PCG. The first sound primarily reflects the closure of the valves between the atria and the ventricles; the second sound reflects mostly closure of the aortic and pulmonic valves after blood has been ejected from the heart into the great vessels.

The ballistocardiogram (BCG) measures the movement of the body as it recoils from the contraction of the heart and the movement of blood throughout the body. The type of sensor used in this study was a low-frequency table, free to move in one dimension, with a modest amount of built-in damping. There are three classical modes of measuring the ballistocardiogram: (1) displacement, (2) velocity, and (3) acceleration. We chose to record acceleration, which provides information about the force of heart contraction. Movements of the table were sensed by means of a velocity transducer and the BCG acceleration was obtained by differentiating the velocity signal. Acceleration measurements filter out the low frequency (0.1 - 0.5 cycle) respiratory components which affect interpretability in the routine displacement recording.

A surface arterial pulse (AP) is a measurement of a pressure wave transmitted along an artery, damped by the tissue between it and the transducer. The transducer used in this study to measure the surface radial arterial pulse was a variable capacitance microphone which was held in place over the radial artery at the wrist by an inflatable cuff.

Figure 1 is a sketch of the arrangement of a subject with transducers and resulting physiological signatures. The subjects were positioned along the length of the ballistocardiogram table. Forty-five subjects were used. Fifteen subjects were normal, 15 had hypertension, and 15 had aortic insufficiency. The pathological subjects had been diagnosed according to standard diagnostic procedures.

For each of the 45 subjects tested we measured the peak amplitudes, durations, and intervals of the waveforms and four characterizing data points, systolic blood pressure, pulse pressure, age and weight (Figure 2). The variables were tabulated on IBM punch cards. These data cards served as the input to an LGP30 computer.

B. Analysis and Results

In order to determine which clinical parameters were statistically significant in separating normal from pathological subjects, the means and variances of each of the

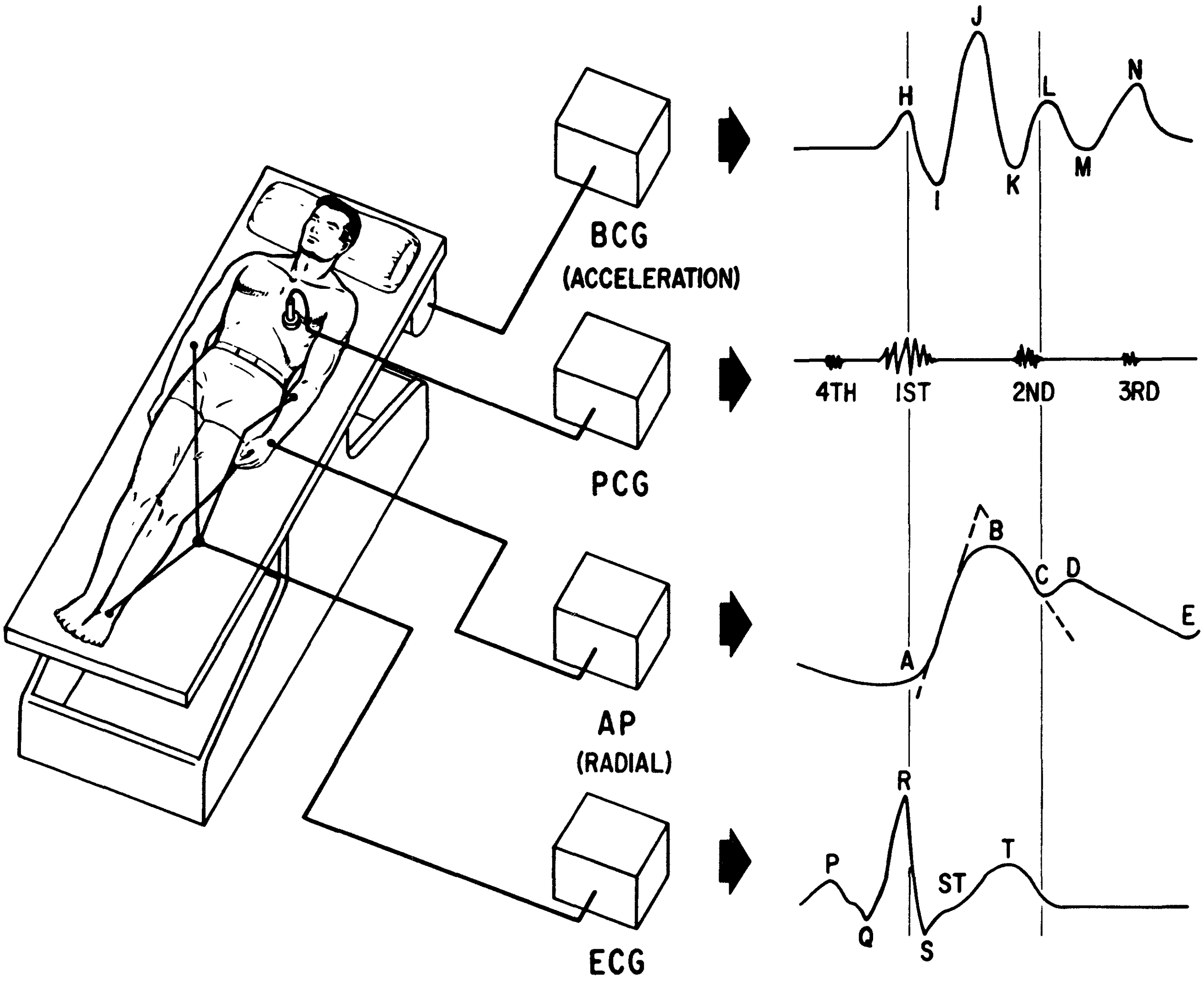


Figure 1

variables, and all of their correlation coefficients were calculated for each group. The significant tests used were the modified t-test, the F-test and z-test.

Figures 3 and 4 describe the significance of the variables in distinguishing the normals from the pathological subjects. At the left, from top to bottom, are the symbols of each of the 45 variables measured; the first 12 are the ECG variables, the next 6 are the PCG variables, the next are the 12 BCG variables, followed by the AP and characterizing variables.

The first columns (top and left) have opaque entries when the mean of each variable is significant at the 99.9 percent level or greater in distinguishing between normal and pathological subjects. The second columns have opaque entries where the variances are significant at the 99.9 percent level or greater.

The significant correlation coefficients are also shown in Figures 3 and 4. Each opaque entry signifies a significant correlation between the two corresponding parameters at the 99.9 percent level or greater. Only one-half of the correlation matrix is filled in since the matrix is symmetrical.

All parameters not yielding a mean or a variance of at least one correlation coefficient significant at the 99.9 percent level or greater were eliminated from further analysis. In addition the Gram-Schmitt routine [6] was used to eliminate all parameters having linear dependence since these parameters do not add any new diagnostic information.

The results of the significance and linear dependency test yielded a total of 14

SOURCE	PEAK AMPLITUDES	DURATIONS	INTERVALS
1. ECG 12 DATA POINTS	P_a Q_a R_a S_a T_a ST_a	P_d QRS_d QR_d T_d	PQ_i QT_i
2. PCG 6 DATA POINTS	1_a 2_a	1_d 2_d	$Q1_i$ $Q2_i$
3. BCG 12 DATA POINTS	H_a I_a J_a K_a	H_d I_d J_d K_d	QH_i QI_i QJ_i QK_i
4. AP 9 DATA POINTS	B_a C_a D_a	AB_d BC_d CD_d DE_d AE_d	QA_i
	DESCENDING SLOPE - DS ASCENDING SLOPE - AS		
5. STATIC 4 DATA POINTS	SYSTOLIC BLOOD PRESSURE - SP PULSE PRESSURE - PP AGE WEIGHT		

Figure 2

parameters that were significant for diagnosing these two pathological groups and one normal group. Of these 14 parameters, 9 were from the electrocardiogram, 4 were from the phonocardiogram, and one was from the ballistocardiogram. These 14 parameters were then used to form a 14-dimensional Gaussian probability density function (Appendix A). The analysis yields a score to characterize groups of subjects with known entities. Use of the same parameters in the same equations but derived from an unknown subject allows accurate classification of the unknown into a known group.

The ratio of the value of the hypertensive group's probability density function to that of the normal group's probability density, or the hypertensive likelihood ration, was also calculated for each subject. Similarly the aortic insufficiency likelihood ratio was calculated. These ratios in addition to probability density values can be used for subject classification. The distributions of both likelihood ratios are shown in Figure 5. One

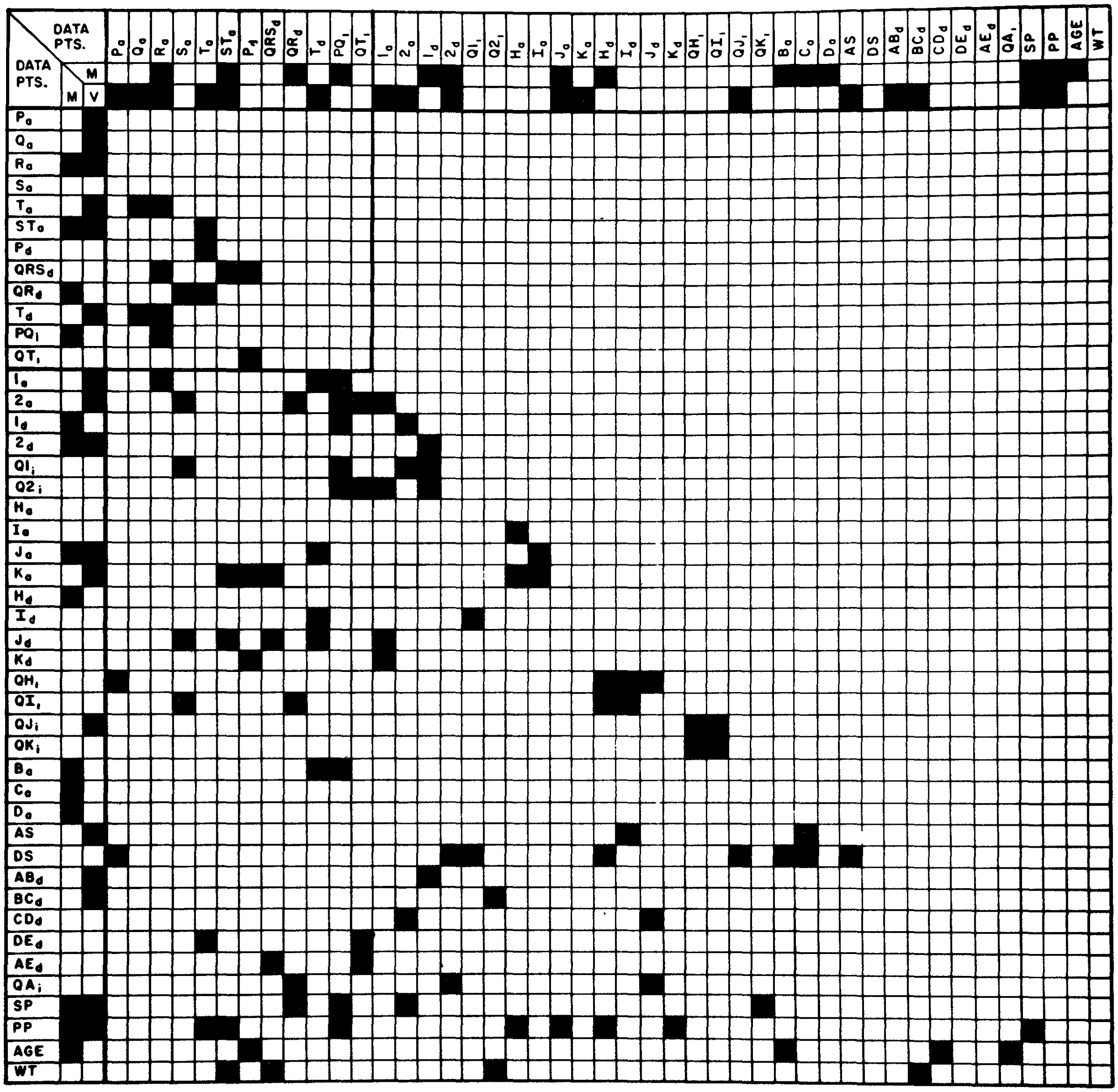


Figure 3. Significant variables, aortic insufficiency versus normal.

hypertensive subject has a likelihood ratio that is in the same range as those of the nonhypertensive subjects. Three aortic insufficiency subjects have likelihood ratios in the same range as those of the hypertensive group.

II. COMPUTER CHARACTERIZATION OF DATA

The results of the previously described computer program demonstrate that the technique described can be used to classify normal and pathological subjects. The classification was based upon manually derived measurements of parameters that characterize physiological waveforms related to the heart.

It is also possible with computer techniques to characterize data automatically. To do this pattern recognition programs are required. A pattern recognition program for the electrocardiogram has been developed [7, 8, 9]. The techniques and methodology used can serve as a basis for pattern recognition programs for other physiological waveforms.

A. Input Data

The basic objective was to develop a program that would automatically recognize and measure parameters from any electrocardiogram taken with any lead system from any

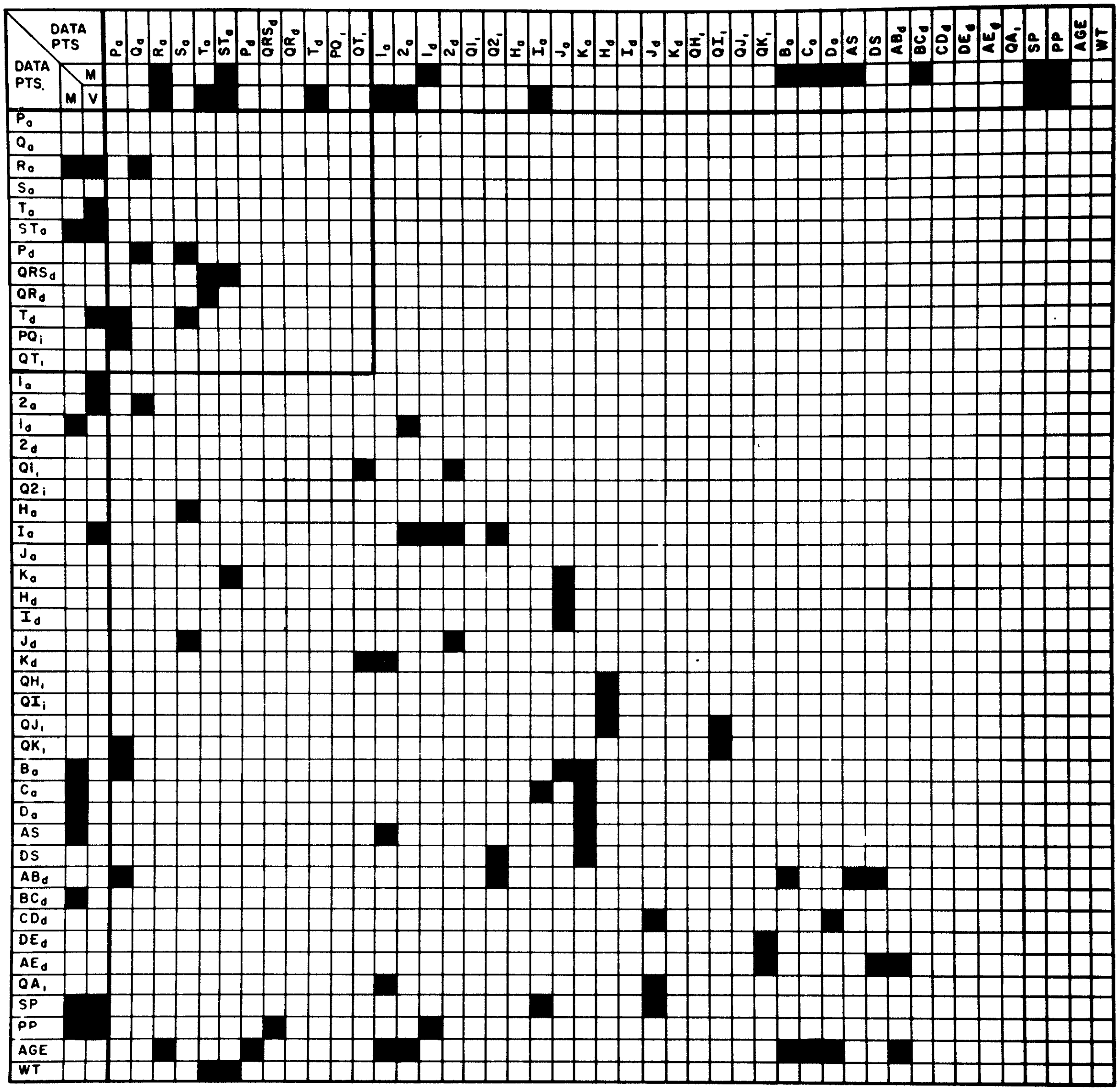


Figure 4. Significant variables, hypertensive versus normal.

subject. The clinical parameters desired were the P, Q, R, S, T and U waves, and the PQ, ST, QT, and RR intervals.

The ECG lead was converted from its analog form to digital values at a rate of 625 samples per second, and was recorded in digital form on magnetic tape. Time measurements were thus accurate to each 0.0016 second. The data was digitized to an accuracy of one part in one thousand. The digitization process was designed to be able to start at any arbitrary point during the electrocardiographic recording. Figure 6 is a block diagram of the data processing system.

Prior to pattern recognition it was necessary to eliminate noise by smoothing the ECG signal. Several techniques are available to do this. One technique is the use of a moving average. With a moving average, the more samples that are averaged together the greater the degree of smoothing. As the number of averaged samples is increased, a value will be reached where further smoothing will severely degrade the signal. In this case, along with noise elimination, there was serious attenuation of the amplitude and characteristic peaks in the ECG waveform. Another type of smoothing is based on

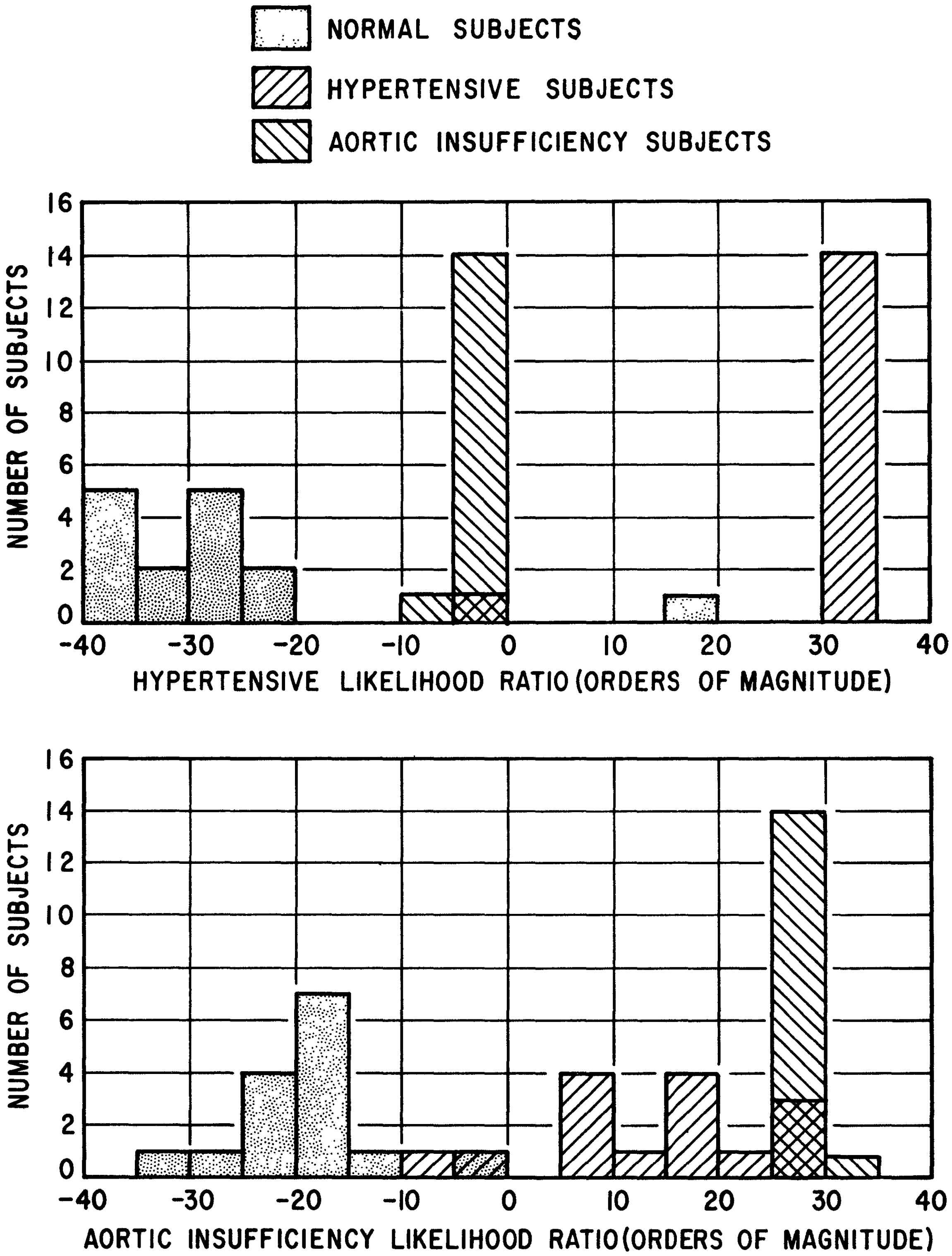

NORMAL SUBJECTS
HYPERTENSIVE SUBJECTS
AORTIC INSUFFICIENCY SUBJECTS
NUMBER OF SUBJECTS
16
14
12
10
8
6
4
2
0
-40
-30
-20
-10
0
10
20
30
40
HYPERTENSIVE LIKELIHOOD RATIO (ORDERS OF MAGNITUDE)
NUMBER OF SUBJECTS
16
14
12
10
8
6
4
2
0
-40
-30
-20
-10
0
10
20
30
40
AORTIC INSUFFICIENCY LIKELIHOOD RATIO (ORDERS OF MAGNITUDE)

Figure 5

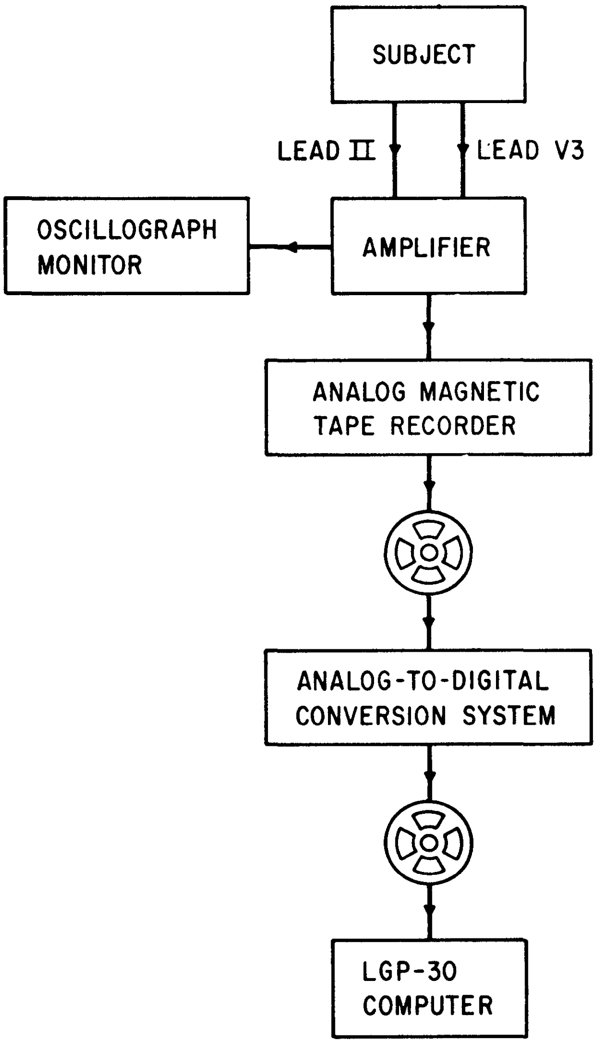


Figure 6

successively determining the parabola that best fits a series of ECG samples. Such a parabola is defined as that which minimizes the mean square difference between the parabola and the value of the ECG samples at corresponding points. The value of the parabola at the center sample point is the value of the smoothed data. To avoid interpolation, the number of sample points is always made odd. As with the moving average, the degree of smoothing increases with the number of ECG samples over which the best parabola is determined, and the degradation of the data increases with an increase in the number of points. For the same degree of smoothing, signal degradation was found to be much less with parabolic smoothing than with moving-average smoothing. The nine-point (n = 9) least square parabola yielded sufficiently smooth data and decreased the amplitudes of the signals to be measured by less than 0.005 mv. and was used.

B. Analysis and Results

Preliminary rules and definitions based on conventional ECG criteria were used to define wave onset, wave peak, wave termination, significant voltage fluctuations and time intervals in the ECG. These definitions were programmed, tested, and reprogrammed as necessary until final definitions suitable for the computer program were formulated. Definitions for the program were suitable when they encompassed all possible variations of waveforms in the electrocardiographic leads.

A basic requirement for any logic for pattern recognition is to locate a point of reference that is repeatable in any subject. This point is particularly needed if the digitization is to be started at any time in the signal. The point of greatest negative rate of change in the ECG signal was used as this constant point of reference. This point, which always occurs after the peak of the R wave or before the peak of the S wave, was obtained by locating the greatest negative value of the derivative in various portions of signal and then finding the absolute minimum of those values. With a reference point such as the one used and ECG complex can be isolated automatically.

The next step in the process is to locate the peaks of the waves. The presence of R waves is ascertained by determining if the value of the maximum positive derivative proceeding the maximum negative derivative exceeds an empiric value. The presence of S waves is ascertained by determining if the derivative is greater than a given value after the maximum negative derivative. Once ascertaining that R or S waves are present the peaks are found by locating the maximum or minimum within a given number of samples. Q wave peaks are found by searching for a minimum value in a given interval before the R wave peak. The P wave peak is searched for in a fixed interval before the location of the maximum negative derivative. The maximum and minimum values of the derivative are found in intervals during P wave duration. The order of maximum and minimum derivatives defines the type of P wave present (positive, negative, diphasic, bifid or trifid).

The T wave peak is located at the maximum absolute amplitude in the smoothed ECG data in a fixed interval after the S wave peak, or the R wave peak if no S wave is present.

The next order of procedure is to determine wave duration, which requires definition of wave onset and end. The point before the first P wave peak having a derivative of less than 1.875 mv/sec for 0.016 seconds was defined as the P wave onset in this study. This and all criteria were empirically determined through trial and error.

To follow clinical practice, a baseline was constructed as the straight line connecting the start of the P wave in the second heartbeat with the start of the P wave in the third heartbeat. This baseline was used as a guide for arbitrary but reproducible definition of wave onset and end. To find the end of the S wave, the first baseline crossing after the S wave peak in the smoothed ECG signal is located. The point after the S wave peak where the derivative is less than or equal to 1.875 mv/sec for 0.008 second is then found. If either a baseline crossing or the point where the derivative is less than 1.975 mv/sec is found, the end of the S wave is located at whichever is present. If both a baseline crossing and a derivative of less than 1.875 mv/sec are found, the end of the S wave is located at the baseline crossing. Similar methodology was followed to find all other waveform points of onset and termination. With these points it was possible to determine wave duration. Amplitudes of the ECG signal were measured from the baseline to the peak of each waveform.

This technique for pattern recognition provides quantitative characterization of data necessary and reasonable to use in a computer. The data can be obtained with the speed and in the volume required for economical use of these machines. The results obtained are reproducible and are comparable to those obtained by cardiologists using manual techniques.

III. SUMMARY

Parameters from physiological waveforms related to the electrical, mechanical, and acoustical properties of the heart can be used as a basis for classifying and characterizing normal and pathological subjects. The use of the multi-dimensional probability density function and the associated likelihood ratio is well suited for this purpose.

Parameters from the ECG, BCG, PCG and AP wave were combined into a multidimensional probability distribution. The different distributions for the normal and pathological groups of patients were formulated and stored in the computer. The compatibility of an unknown subject's parameters with those stored in the computer can be calculated for classification, into a known group.

A pattern recognition program has been developed to automatically recognize and measure the principal components of the electrocardiogram. The techniques described in this paper can be used as an automated system to aid the physician in the diagnosis of disease.

APPENDIX A

The probability density function was formulated using the following equation.

$$P_i = \frac{|A_o|^{1/2}}{(2\pi)^{12}} \epsilon^{-1/2} \sum_{q=1}^{12} \sum_{j=1}^{12} a_{jq} (X_{ji} - m_j) (X_{qi} - m_q)$$

where X_{ji} is parameter j from subject i m_j equals the mean of parameter j $|A_o|$ is the determinant of the inverted covariance matrix. Small a_{jq} is the element of the j th row and the q th column of the inverted covariance matrix.

REFERENCES

1. Caceres, C. A., Rikli, A. E., The Digital Computer as an Aid in the Diagnosis of Cardiovascular Disease. Trans. New York Academy of Sciences. 23:240, 1961.
2. Rikli, A. E., Caceres, C. A., Abraham, S., Tolles, W. E., Steinberg, C. A., An Electronic System for Electrocardiographic Analysis. (Abstract). Circulation. In Press.
3. Tolles, W. E., Carbery, W. J., Freiman, A. H., Caceres, C. A., Role of the Modern Computer in Analyzing Electrocardiographic Data. (Abstract). Circulation. 22:824, 1960.
4. Rikli, A. E., Tolles, W. E., Steinberg, C. A., Carbery, W. J., Freiman, A. H., Abraham, S., Caceres, C. A., Computer Analysis of Electrocardiographic Meassurements. Circulation. In Press.
5. Tolles, W. E., Steinberg, C. A., Carbery, W. J., Freiman, A. H., Experimental Techniques and Results of a Study Using a Computer as a Diagnostic Aid. Trans. New York Academy Sciences. 23:246, 1961.
6. Birkhoff, G., McLane, S., A Survey of Modern Algebra, MacMillan Company, New York City, 1953.
7. Caceres, C. A., Steinberg, C. A., Carbery, W. J., Abraham, S., Computer Extraction of Electrocardiographic Parameters. Clinical Research, 9:137, 1961.
8. Caceres, C. A., Steinberg, C. A., Abraham, S., Carbery, W. J., McBride, J. M., Tolles, W. E., Rikli, A. E., Computer Extraction of Electrocardiographic Parameters. Circulation. In Press.
9. Steinberg, C. A., Carbery, W. J., Caceres, C. A., Pattern Recognition in the Electrocardiogram. Presented at the 4th International Conference on Medical Electronics, New York City, New York, July 14 - 21, 1961.