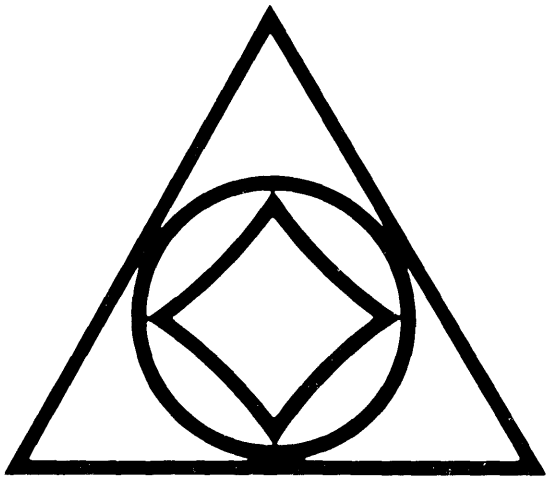


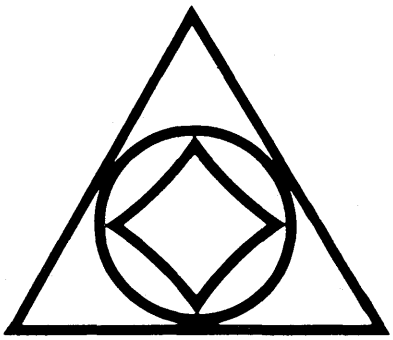
AFIPS

CONFERENCE PROCEEDINGS

VOLUME 22

1962

FALL JOINT COMPUTER CONFERENCE



AFIPS

CONFERENCE PROCEEDINGS

VOLUME 22

1962

FALL JOINT COMPUTER CONFERENCE

SPARTAN BOOKS

6411 CHILLUM PLACE, N. W. • WASHINGTON 12, D. C.

List of Joint Computer Conferences

1. 1951 Joint AIEE-IRE Computer Conference, Philadelphia, December 1951
2. 1952 Joint AIEE-IRE-ACM Computer Conference, New York, December 1952
3. 1953 Western Computer Conference, Los Angeles, February 1953
4. 1953 Eastern Joint Computer Conference, Washington, December 1953
5. 1954 Western Computer Conference, Los Angeles, February 1954
6. 1954 Eastern Joint Computer Conference, Philadelphia, December 1954
7. 1955 Western Joint Computer Conference, Los Angeles, March 1955
8. 1955 Eastern Joint Computer Conference, Boston, November 1955
9. 1956 Western Joint Computer Conference, San Francisco, February 1956
10. 1956 Eastern Joint Computer Conference, New York, December 1956
11. 1957 Western Joint Computer Conference, Los Angeles, February 1957
12. 1957 Eastern Joint Computer Conference, Washington, December 1957
13. 1958 Western Joint Computer Conference, Los Angeles, May 1958
14. 1958 Eastern Joint Computer Conference, Philadelphia, December 1958
15. 1959 Western Joint Computer Conference, San Francisco, March 1959
16. 1959 Eastern Joint Computer Conference, Boston, December 1959
17. 1960 Western Joint Computer Conference, San Francisco, May 1960
18. 1960 Eastern Joint Computer Conference, New York, December 1960
19. 1961 Western Joint Computer Conference, Los Angeles, May 1961
20. 1961 Eastern Joint Computer Conference, Washington, December 1961
21. 1962 Spring Joint Computer Conference, San Francisco, May 1962
22. 1962 Fall Joint Computer Conference, Philadelphia, December 1962

Conferences 1 to 19 were sponsored by the National Joint Computer Committee, predecessor of AFIPS. Back copies of the proceedings of these conferences may be obtained, if available, from:

- Association for Computing Machinery, 14 E. 69th St., New York 21, N. Y.
- American Institute of Electrical Engineers, 345 E. 47th St., New York 17, N. Y.
- Institute of Radio Engineers, 1 E. 79th St., New York 21, N. Y.

Conference 20 and up are sponsored by AFIPS. Copies of AFIPS Conference Proceedings may be ordered from the publishers as available at the prices indicated below. Members of societies affiliated with AFIPS may obtain copies at the special "Member Price" shown.

Volume	List Price	Member Price	Publisher
20	$12.00	$7.20	Macmillan Co., 60 Fifth Ave., New York 11, N. Y.
21	6.00	6.00	National Press, 850 Hansen Way, Palo Alto, Calif.
22	8.00	4.00	Spartan Books, 6411 Chillum Place, NW, Washington 12, D. C.

The ideas and opinions expressed herein are solely those of the authors and are not necessarily representative of or endorsed by the 1962 Fall Joint Computer Conference Committee or the American Federation of Information Processing Societies.

Library of Congress Catalog Card Number: 55-44701

Manufactured by McGregor & Werner, Inc.
Washington, D. C.

CONTENTS

PREFACE

The theme of the 1962 Fall Joint Computer Conference is Computers in the Space Age. Today there is a two-way street in which computing equipment has contributed vitally to the success of space age technology, but the space-age demands have had their major effects on the design of computers. Of these we can readily discern three outstanding results: (1) development of more efficient interfacing between man and machine, (2) radical reduction of the size of systems, and (3) the maturing of the theory and implementation of cooperative systems, including multi-point operating complexes.

Naturally these achievements are irrevocably to be reflected in the stationary equipment that benefits business and science. We already know that for the purposes of the Space Age, computing equipment is to provide facility for command-decision and for control of a new order of complexity. But we are just becoming aware of the products of this progress. The social implications of advances in the precise selection of information via recursive interplay between man and machine—though barely perceptible at the present time—are rapidly assuming major influence on the structure of the near future.

Altogether, the interaction of the space age and computer technologies has brought about a rich growth in new and potent national resources. Indeed, the record of the United States in the field of information and data processing systems is pre-eminent in the present world. It is helping therefore very directly to give us pre-eminence in space.

J. Wesley Leas
Chairman
1962 Fall Joint Computer Conference

PROCESSING SATELLITE WEATHER DATA - A STATUS REPORT - PART I

Charles L. Bristor
U. S. Weather Bureau
Washington, D. C.

SUMMARY

Less than 500 radiosonde observations are available for the current twice daily three dimensional weather analysis over the Northern Hemisphere—a coverage far less than is required for short term advices and for input to numerical prediction computations. Global observations from operational satellites as a complement to existing data networks show promise of filling this need. TIROS computer programs now being used for production of perspective geographic locator grids for cloud photos, and other programs being used to calibrate, edit, locate and map infrared radiation sensor measurements, have provided a background of experience and have indicated the potentialities of a more automated satellite data processing system. The tremendous volume of data expected from the Nimbus weather satellite indicates the need for automatic data processing. Each pass around the earth will produce ninety-nine high resolution cloud pictures covering about ten percent of the earth from pole to pole and infrared sensors will provide lower resolution information but on a similar global basis. Indications are that machine processing of the 280-odd million binary bits of data from each orbit can materially reduce the human work load in producing analyzed products for real time use. The main programming packages in support of the presently developing automatic data processing systems are explained under ingestive, digestive, and productive headings. Tasks under these headings are explained for both the photo and infrared data. The individual program modules and subroutines are discussed further in an appendix. Reference is made to the second part of this report which expands on the logical design of the digital and non-digital data handling system complex and extends the discussion into data rates, command and control concepts and the executive program which manages the overall process.

INTRODUCTION

The need for more meteorological data is an old refrain which is almost constantly being revived. Why do we always desire more data? Among the many very good answers to this question are some which are pertinent to the subject of meteorological satellites. A most generalized answer might be expressed in two parts:

1. because as the scope of human activities increases, new applications of weather information arise and new needs for meteorological advice are generated and
2. because potential economic gains provide a tremendous impetus for attempting to improve the quality and scope of our present weather services.

Within the category of the first answer one may cite the expansion of global air travel over routes that are practically devoid of weather observations of any kind and

the similar deployment of air and sea defense forces to remote areas. Even the man in space program is generating a need for global weather information. In the thirties and into World War II a marked expansion of weather observing networks took place—mainly through expansion of weather communications to communities where observations facilities could be installed. Because of communications and logistics costs, this type of expansion cannot take place indefinitely to fulfill the ever growing need for detailed observations on a global scale. However, within the scope of the first answer, such a global network would be extremely valuable merely as a means of providing current weather information and very short term warnings and advisories.

Beyond immediate operational advice is the need implied by the second answer—the problem of weather prediction. The American Meteorological Society (1962) has recently restated its estimate of current skills in weather forecasting.

> "...For periods extending to about 72 hours, weather forecasts of moderate skill and usefulness are possible. Within this interval, useful predictions of general trends and weather changes can be made...."

Few would deny the economic importance and increased application of more precise 3-day forecasts.

Since the mid-fifties numerical weather prediction has had a significant influence on the level of skill in weather forecasting generally. The method involves a mathematical description of the atmosphere in three dimensions utilizing the hydrodynamic equations of motion and the laws of thermodynamics. The partial differential equations of such a "model" are arranged in a prognostic mode such that only time dependent partials remain on the left side. The finite difference version of such an equation set is then integrated in short time steps to produce prognostic images of the various data fields which served to describe the initial state of the fluid. Phillips (1960) has summarized the current view which delimits the potential of numerical weather prediction—to the extent that lack of observations prevents adequate description of the atmosphere on a global basis. Figure 1 indicates the present network of observing stations which provide the current three dimensional description of the atmosphere together with a grid overlay indicating intersections at which information is required concerning the current state of the fluid in order that the finite difference equations may be integrated. Obviously a poorly distributed collection of less than 500 observations cannot adequately establish values for nearly two thousand grid points. Areas the size of the United States are indicated without any upper air soundings whatsoever. The situation in the Southern Hemisphere is much worse.

This brief discussion of the meteorological data problem points up the need for a detailed global observational network and offers the real challenge to meteorological satellites. Can indirect sensing via satellite fill the need for global weather data? D. S. Johnson (1962) has summarized the meteorological measurements carried out thus far by satellites and discussed others planned and suggested for the future.

Indications are that, whereas satellite observations will likely never supplant other data networks, they hold great promise in providing complementary data on a truly global basis. Limited experience with satellite weather data already obtained is very encouraging.

The following is a description of current efforts in processing the ever growing volume of this data. First, limited computer processing of TIROS data is discussed. The latter portion of this report and the second paper in this two part series describe in some detail the current status of computer programming in support of the truly automated real time data processing systems under construction for the Nimbus satellite system.

EXPERIENCES WITH TIROS

Since April, 1960 cloud photos from TIROS satellites have been made available to the meteorological community on an intermittent operational basis. Details of the satellites' construction including its slow scan cameras have been given elsewhere along with an account of certain difficulties in geographically locating the cloud photos because of meandering in the spin axis (NASA - USWB, 1960). A cloud photo sample is presented in Figure 2. Even without a meteorological background, one would likely concede, on the basis of intuition, that such cloud patterns could provide

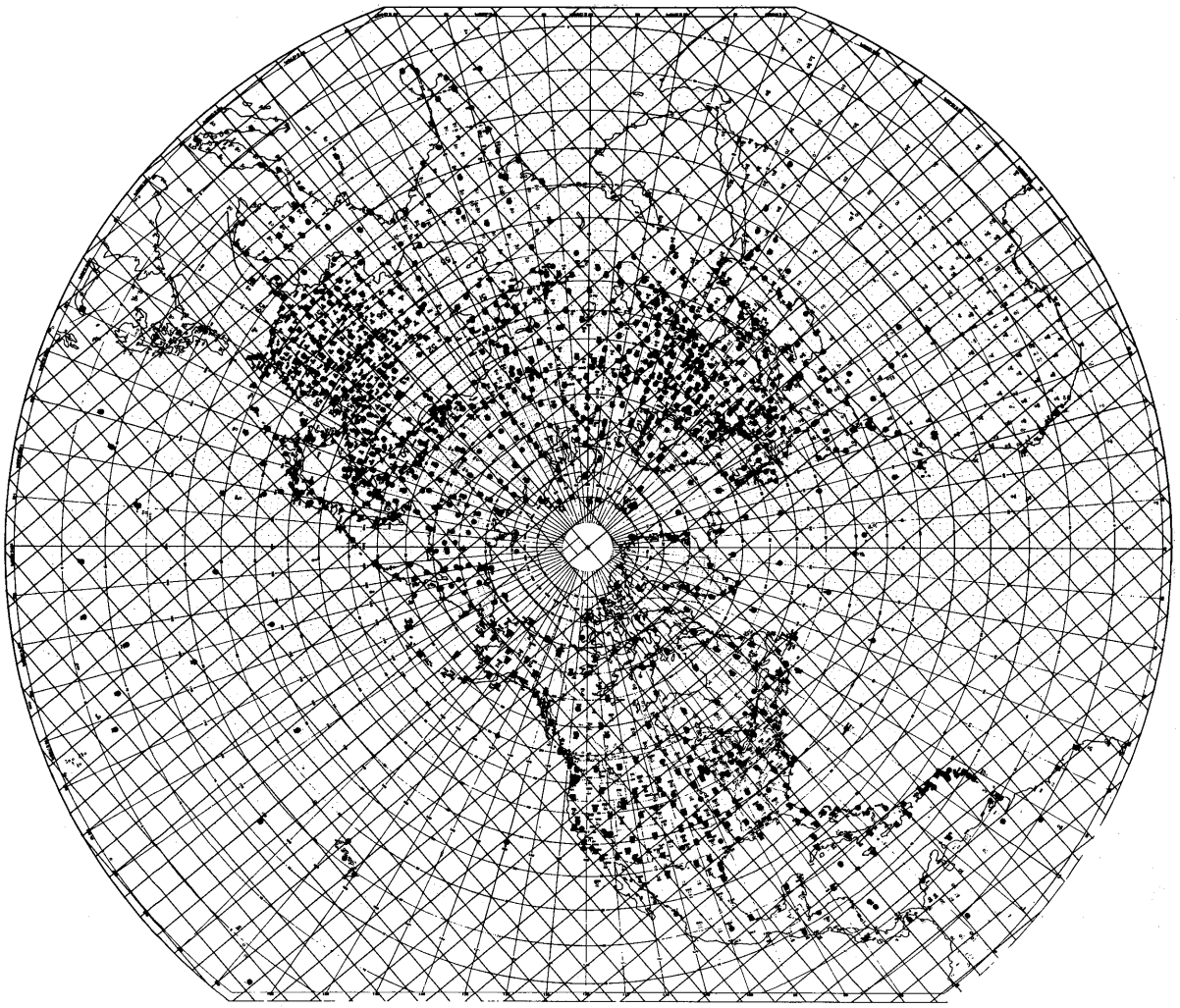


Figure 1. Northern Hemisphere map showing upper air reporting stations and computation grid used in objective weather map analysis and numerical prediction. The Weather Bureau's National Meteorological Center uses a somewhat denser grid of more than 2300 points. Less than 500 of these reports are routinely available for specification of quantities at the grid points.

valuable observational evidence concerning the state of the atmosphere. A considerable research effort is now going on in an effort to extract quantitative information from such images (NASA - USWB, 1961). For the present, computer processing has been confined largely to the production of geographic locator grids as an aid to further interpretation of the cloud patterns. The locator grid superimposed on the picture in Figure 2 and the sample grids shown in Figure 3 are produced at a rate of 10 seconds per grid on the IBM 7090 (Frankel & Bristor, 1962). Line drawn output is produced on an Electronic Associates Data Plotter or, alternately, by General Dynamics High Speed Microfilm Recorder.

Input for each grid includes latitude and longitude of the sub-satellite point, altitude of the satellite as well as azimuth and nadir and spin angles which describe the attitude and radial position of the camera with respect to the earth. An auxiliary program is required for the production of image to object ray distortion tables. These tables correct for symmetric and asymmetric distortions due to the lens and the electronics of the system and are produced from pre-launch calibration target photos taken through the entire camera system. An additional feature of the gridding program is the large dictionary of coastline locations from which transformations to the perspective of the image

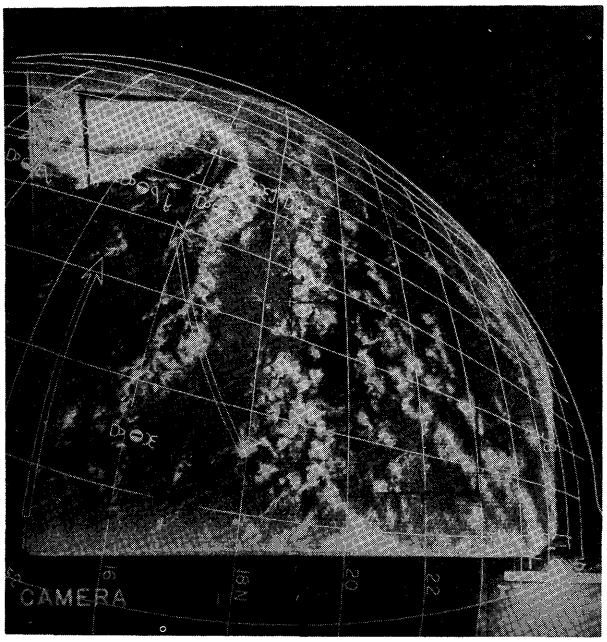


Figure 2. Sample cloud picture with perspective geographic locator grid. This photo, taken by TIROS III, shows hurricane Anna near 12°N, 64°W (lower left) on July 20, 1961 together with large streamers projecting toward another vortex pattern to the east (right).

are made as an aid in mating the cloud image and grid. Some 10,000 such grids have been produced thus far for selected cloud photos taken by TIROS I and TIROS III and are available in an archive, along with the pictures, for research applications. A somewhat less detailed but similar gridding procedure is being utilized on a smaller Bendix G-15 computer at the TIROS readout sites for the current real time hand processing of the picture data (Dean, 1961). A typical example of such a nephanalysis (cloud chart) composed from a group of photos is shown in Figure 4. Features from the several images are replotted in outline form or reduced to symbolic form on a standard map base for facsimile transmission to the weather analysts and forecasters.

Starting with TIROS II in November, 1960, infrared sensors have furnished experimental radiation measurements in five selected wavelength intervals (NASA - USWB, 1961 and Bandeen, 1962). Although these data have not been available in real time, an extensive 7090 program has been produced for their reduction to a usable form. The IR information has been utilized in a quantitative manner in several research studies. Fritz and Winston (1962) have demonstrated its usefulness in cloud top determinations and Winston and Rao (1962) have used it in connection with energy transformation investigations on the planetary scale.

The data reduction program accepts raw digitized sensor values read out from the satellite, rejects space viewed samples, converts the earth viewed responses to proper physical quantities through a calibration procedure and finally combines the data with orbit and attitude information to create a final meteorological radiation tape (FMRT). Data from one orbit is thus reduced to an archivable file on magnetic tape by the 7090 in less than twenty minutes. This tape becomes the data source for other programs which have been produced for the purpose of mapping selected samples of such data on standard maps for use with other meteorological charts. A sample is shown in Figure 5.

The above discussion indicates the nature of the data obtained thus far by meteorological satellites and the kinds of computer support provided. Experience gained in programming the earth location of sensor measurements obtained from satellites, the conversion to standard maps, the calibration and logical sorting of raw data and the experience gained with distortion and attitude programs have all provided background for programs now being produced for direct application in an automatic system. Meanwhile research with TIROS data is suggesting new uses which are likely to lead to a requirement for more kinds of products and interpretations. Experience from past efforts is thereby supporting present efforts in developing an automated, real time system for the processing of global data coverage which will be coming from the Nimbus satellite series.

THE NIMBUS DATA PROCESSING TASK

The Nimbus satellite represents a significant advancement over TIROS as an operational satellite. The spacecraft system (Stampfl and Press, 1962) provides more camera coverage of higher resolution, and earth stabilization assures maximum photo coverage. One downward and two oblique looking cameras will view a broad strip of the earth athwart the vehicle's path as shown in Figure 6. The three views overlap slightly.

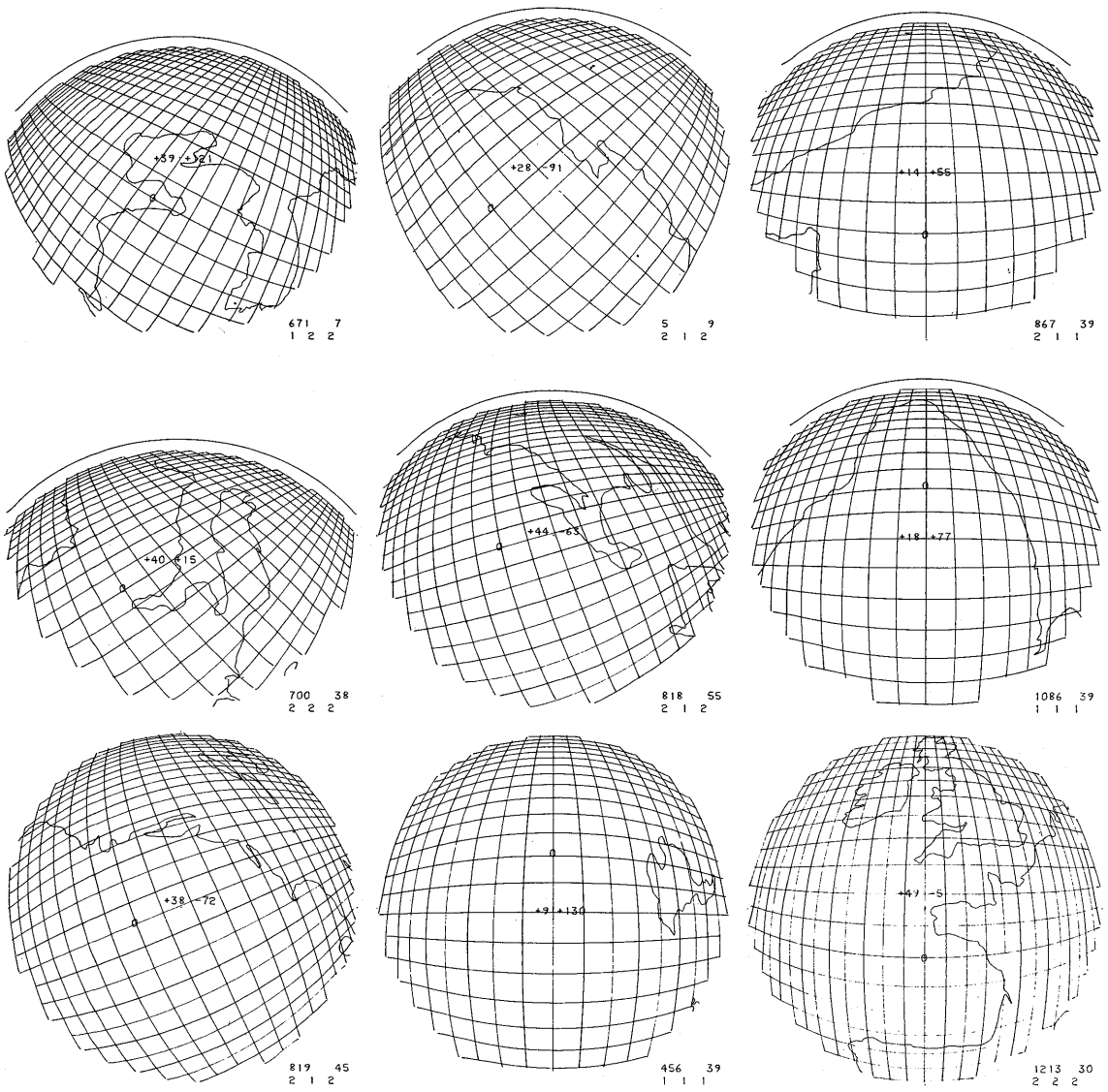


Figure 3. TIROS grids with familiar coastline features. The set of digits bracketing a central intersection indicate the latitude (left-hand number, plus for North) and longitude (right-hand, plus for East) of that point. A zero is plotted along the meridian at the next intersection to the South. Legend in the lower right indicates orbit and frame number for the matching photo (top line, from left) as well as readout station, mode (taped or direct), and camera (single digits, from left). Horizon arc is indicated beyond the truncated grid pattern at the top where appropriate.

The extremely foreshortened region near the horizon is not viewed. Thirty-three such photo clusters will be obtained from each pass around the earth. Considerable overlap in the wings is obtained from cluster to cluster as shown in Figure 7. The near polar orbit will assure global coverage daily. Overlap from orbit to orbit is minimal at the equator but is very great near the poles (Figure 8). During the polar summer one would expect to see a view such as is covered in Figure 9 on every orbit. The slight inclination of the orbit in a retro sense (injection into orbit with a westerly direction component) will provide controlled illumination for the pictures in that local sun time will remain unchanged from orbit to orbit. Each slow scan TV camera (1" diameter Vidicon tube) contains 833 lines of picture information giving a maximum image resolution of about 1/2 mile when looking directly downward from a nominal orbit of 500 nautical

miles. Such a picture will thus contain nearly 700,000 picture elements. If each of these scan spots is converted on a 16 segment gray scale into a 4 bit binary number, then the 99 pictures obtained from each 108 minute orbit will produce almost 275 million bits.

Scanning radiometers will provide IR information as does TIROS but again will obtain optimum scans from horizon to horizon athwart the vehicles's track. One narrow angle high resolution sensor (HRIR) will respond in a water vapor "window" portion of the infrared spectrum and will effectively provide cloud top temperatures or, in cloudless areas, surface temperatures. A mosaic of such scans on the dark portion of each pass will provide a night time cloud cover picture from pole to pole.

The first such HRIR sensor with a .5 degree viewing cone will provide maximum resolution of about 5 nautical miles. Since the earth will be viewed about one third of each scan revolution, 240 non-overlapping measurements can be obtained from each scan. Approximately 2800 non-overlapping scan swaths will be required to cover the dark half of the orbit. Since these sensors have a wider usable response range, each scan spot will occupy a 7 bit binary number. The HRIR response from each orbit will

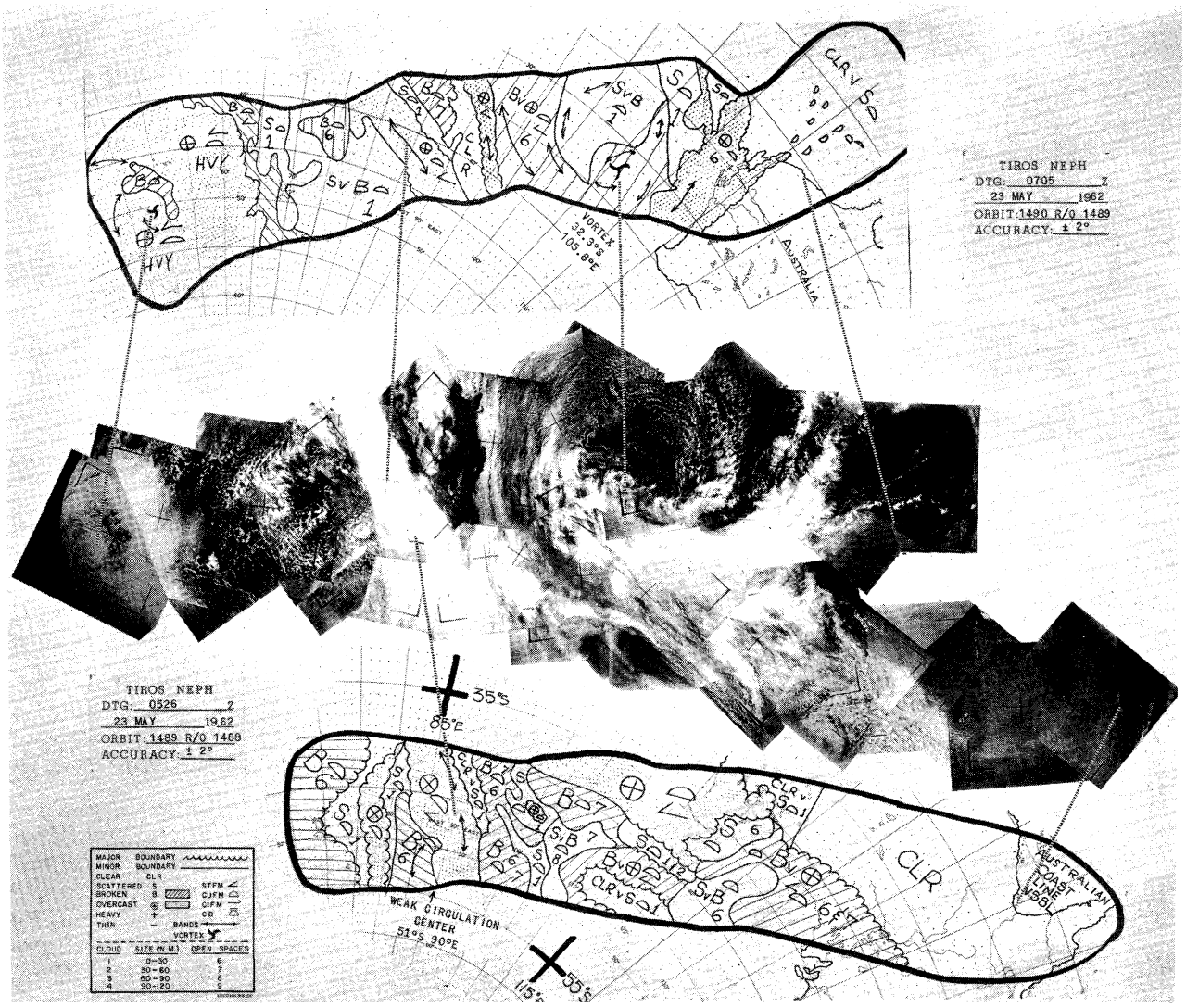


Figure 4. Nephanalyses (cloud charts) prepared by TIROS readout station meteorologists. Features of the cloud patterns from two successive orbits are extracted in outline form and placed on a standard polar stereographic map base for facsimile transmission to weather analysts and forecasters. Vortex centers are located along with other distinctive features.

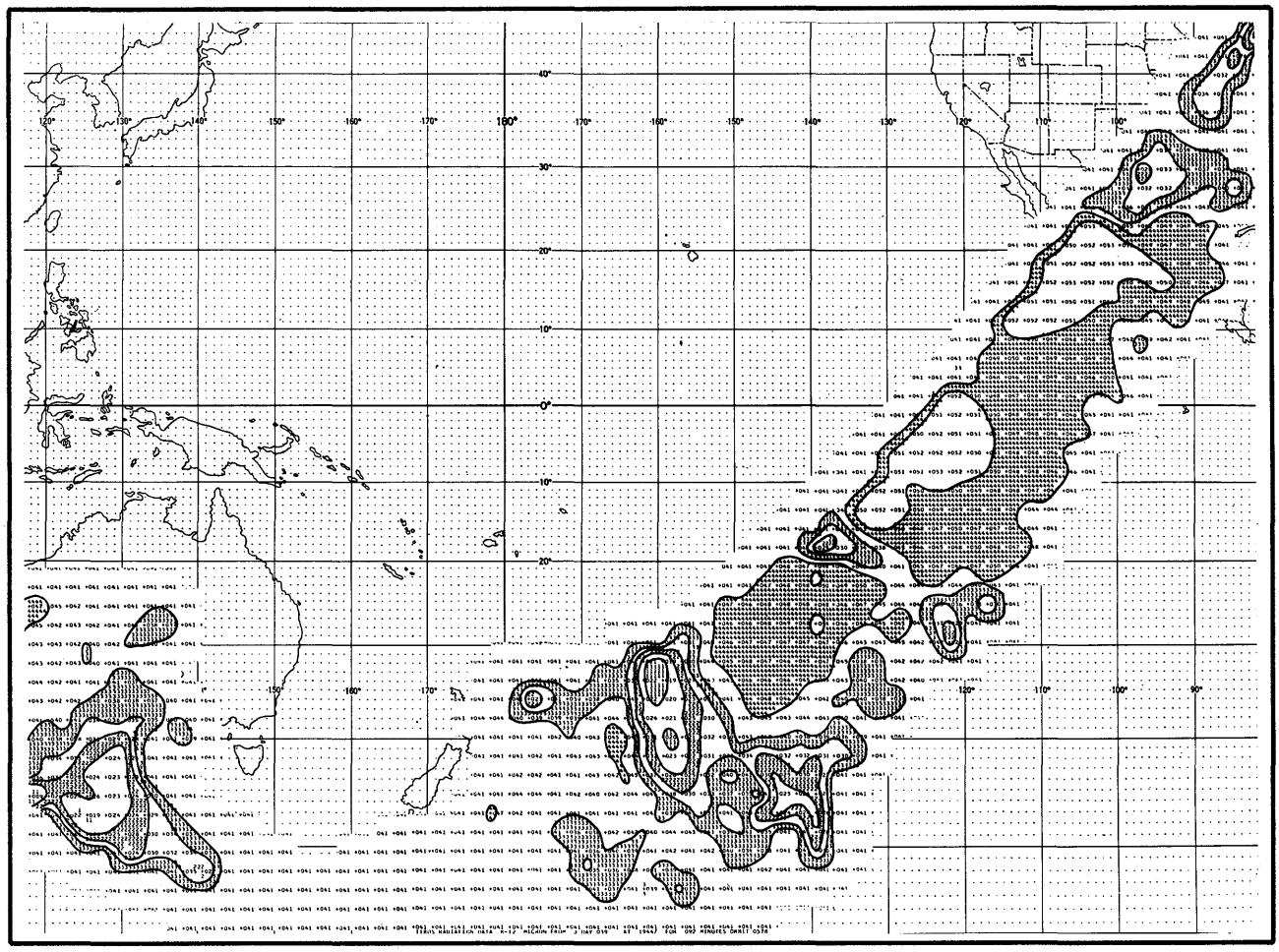


Figure 5. TIROS II infrared analysis. Part of the 8-12 micron water vapor "window" data read out on orbit 578 has been summarized in grid squares on a polar stereographic map base. Radiation coming essentially from cloud tops or from the surface is expressed in watts per square meter.

therefore contain more than 4.7 million bits.

Another 5 channel medium resolution infrared scanner (MRIR) will provide additional information throughout each orbit. The five degree view of the MRIR sensors will provide about 42 separate earth measurements per scan revolution from each channel. Approximately 700 non-overlapping scans are required for a full orbit so that (again using 7 bits per measurement) the MRIR response from each orbit will contain more than 1 million bits.

The volume of information expected from each pass is indeed impressive especially when one realizes that this information is to come night and day on a continuous basis for immediate real time utilization. A marked increase in the present number of TIROS data analysts and helpers is indicated for Nimbus data processing if present semi-hand procedures continue. With plans for higher resolution sensors of increasing variety, automatic processing of satellite weather data is becoming a necessity.

STATUS REPORT ON NIMBUS DATA PROCESSING PROGRAM

The automatic data processing system under construction will be located at the Weather Bureau's National Weather Satellite Center (NWSC), Washington, D. C. and will receive its input data from the command and data acquisition (CDA) facilities at Fairbanks, Alaska through multiple broad band microwave communication facilities. The system at NWSC contains a complex of

components in addition to the digital computers. A detailed explanation of the system is beyond the scope of this report although a brief description from the computer oriented viewpoint is given in the second part. Let it suffice at this point to say that the system is evolutionary in design in that computations will continue in support of semi-hand processing procedures. For this purpose the system's IBM 7094 with attached 1401 will be utilized to produce a picture gridding tape. Information in the form of override signals at specified Vidicon scan line and scan spot numbers, when melded with the analogue picture signals, will produce a kinescope recording of the original cloud photo with a super-imposed dotted line locator grid such as Figure 10. A small CDC 160A computer, interruptable by Vidicon synch pulses, will synchronously meld the digital information from the gridding tape onto spare tracks of an analogue picture tape. Other non-digital devices will then combine the synchronized information on this tape as it is fed into the kinescope recorder.

The 7094 program is being produced essentially as an extension of present TIROS programs. A simulated output of this program has provided check out facility for the 160A program which now awaits the unique non-digital hardware complex for final checkout. A complexity of supporting programs are involved in this effort as indicated in the appendix which briefly describes each program module. This effort will permit a TIROS type semi-hand processing of the photo data but with hand melding of grid and picture now automated.

The far greater task of the system involves duplication of the semi-hand processing by automatic means. In the beginning these efforts must be experimental in that application of the data is still exploratory. Methods of presentation, quantities to be extracted from the basic data, the scale of atmospheric phenomena to be described (resolution) are all in exploratory stages. A

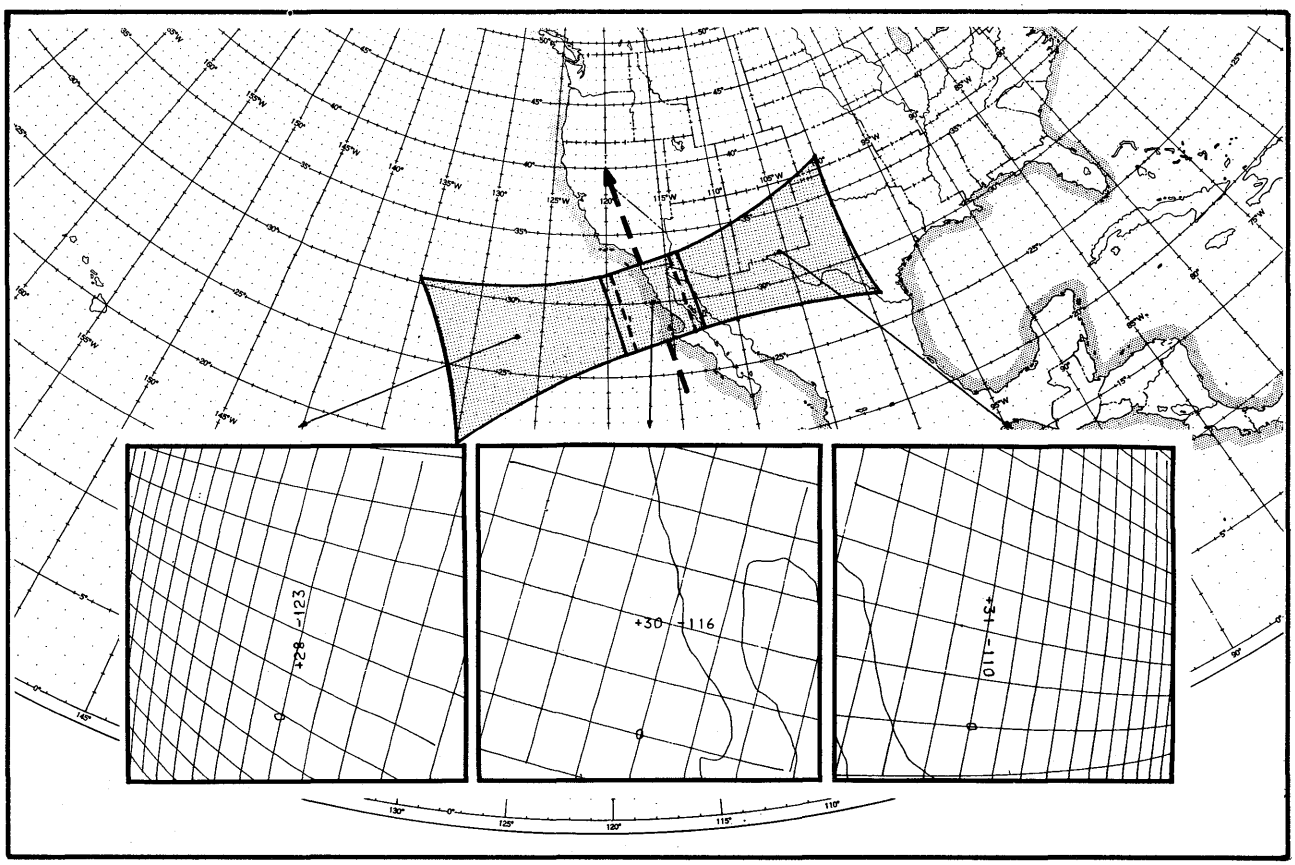


Figure 6. Perspective grids and mapped coverage of Nimbus camera cluster as seen from a 500 nautical mile orbit. The central camera looks directly downward at the sub-satellite point. Side cameras are tilted 35 degrees to either side of the track.

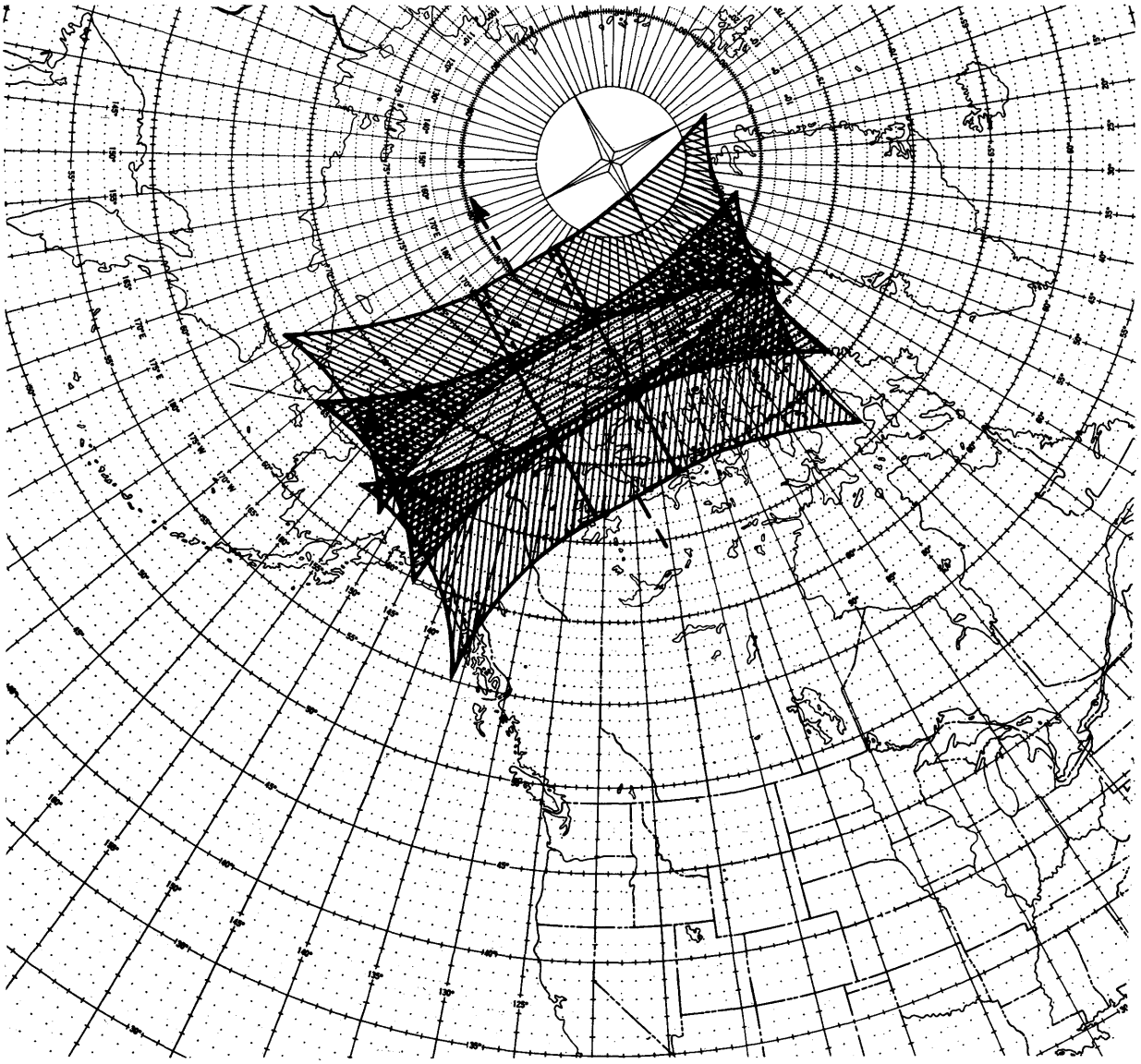


Figure 7. Geographic coverage to come from Nimbus showing overlap between adjacent three camera clusters.

major effort is underway to create a hierarchy of data processing programs to activate the system and produce a variety of outputs in a flexible manner. These may be grouped as ingestive, digestive, and productive.

The ingestive programs are more than simple input routines in that some preprocessing of the data is accomplished. In the case of picture data, the entire volume mentioned previously is to be fed into storage in the computer. Some sorting is required before storage so that separate disk files are created containing data from each of the three cameras. As time permits, other pre-processing activities will also be accomplished. Light intensity signals over the face of each photo require normalization for angle of view before quantitative comparisons are valid. Also, for the same reason, solar aspect variation from equator to pole must be removed.

In the case of the incoming MRIR and HRIR data, the ingestive process is partly one of data editing. By recognition of pulses which provide knowledge of scanner shaft angle, almost two thirds of the incoming data which is non-earth viewing can be eliminated.

Other raw housekeeping input information such as attitude error signals and sensor environment temperatures must be unpacked and translated through calibration in the ingestive process before they can be used in processing the meteorological data.

Final checkout of these programs must await activation of the complete hardware complex since only limited simulation is possible.

The digestive process takes the pertinent incoming data and converts it to a meteorologically usable form. A major task is the melding of this data with the orbit and attitude information to geographically locate the sensor information elements. In the case of the photos, part of this work is accomplished as an adjunct to the earlier mentioned program which produces the picture gridding tape. An open lattice of points selected by scan row and spot number are geographically located within each image. From these location "bench marks" the digestive program transforms the foreshortened, perspective photo image into a rectified equivalent on a standard map base. Figure 11 is an experimental

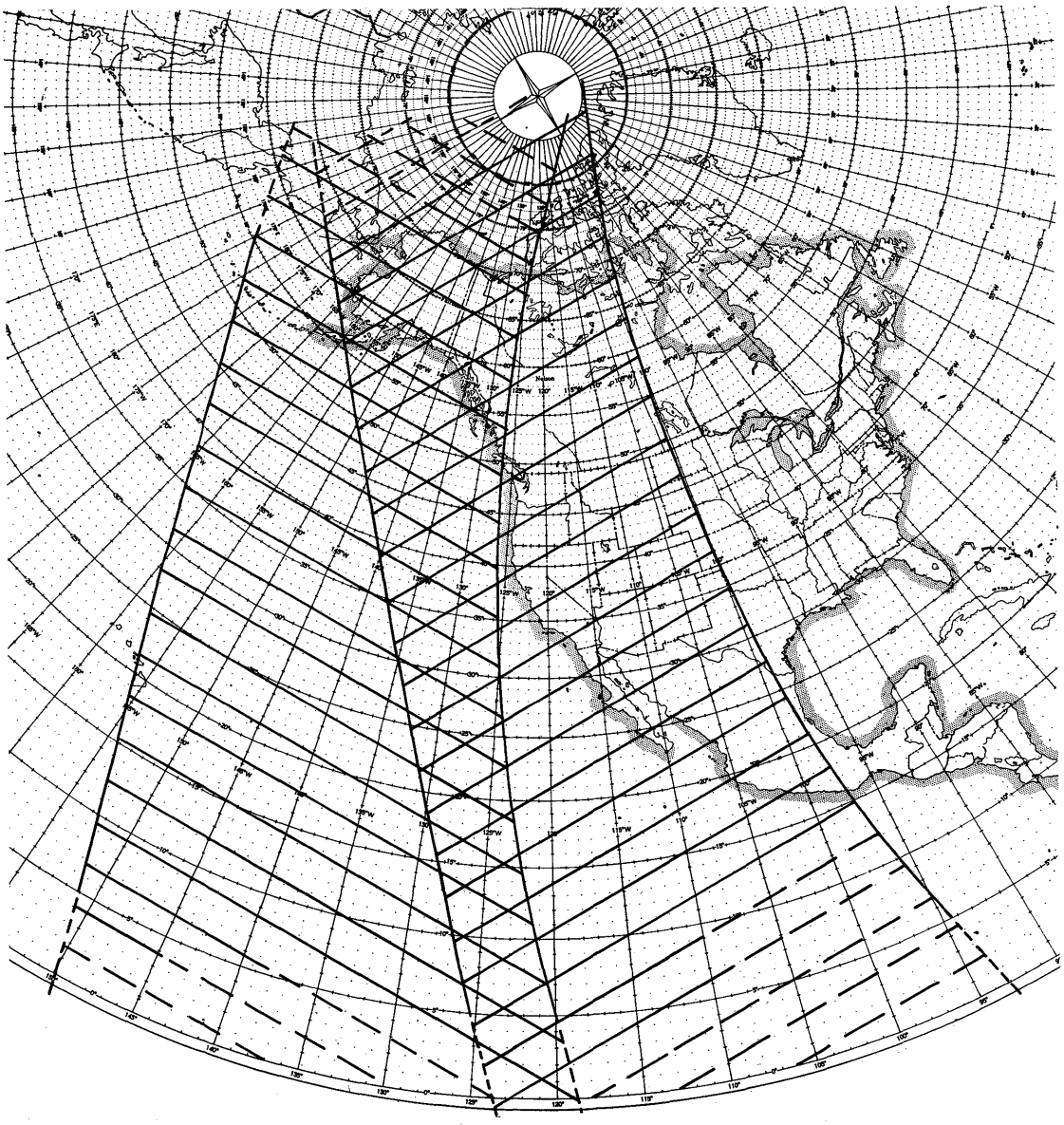


Figure 8. Geographic coverage envelopes to come from Nimbus showing overlap from orbit to orbit.

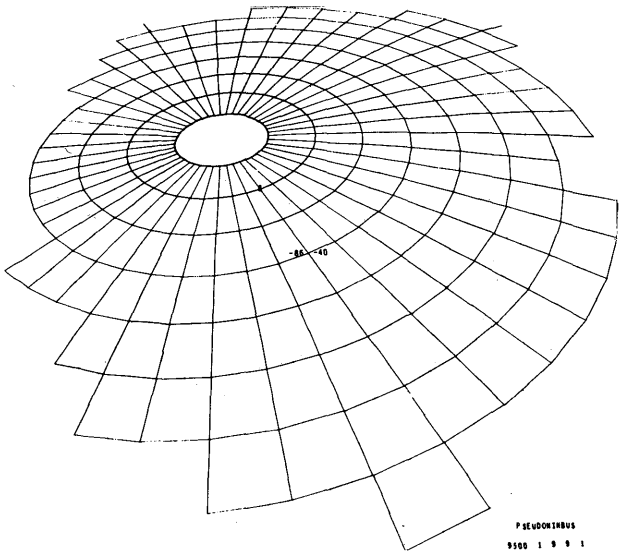


Figure 9. Sample perspective grid showing the polar region to be viewed by Nimbus.

example. The rectified image appears on a mercator map projection—in one view as a replotting of the original picture elements only. It demonstrates the futility of extending this process into extremely foreshortened image areas where a realistic rectified image would consist largely of interpolated filler.

After this step the rectified images are fitted together into a mosaic strip which is then available as a product source.

The digestive infrared data program is being patterned after that mentioned earlier which has been produced for the processing of TIROS radiation data. The calibrated and earth located data will similarly provide a product source through the archivable final meteorological radiation tape.

Programs for production of usable output material present the most problems. Full resolution photo mosaics rectified to polar stereographic or mercator maps are expected to find application over limited regions in connection with hurricane detection and tracking, for example. For other broad-scale analysis problems, products having reduced resolution may be adequate. This implies searching these images by machine, editing and summarizing them as to percent cloud cover, brightness and pattern. Some interesting patterns are revealed in the TIROS photos of Figure 12. Although, as mentioned earlier, quantitative interpretations are only gradually emerging, the rings, spirals and streets seen in these photos will likely be subjects for identification through pattern recognition techniques. Cloud heights, provided indirectly from the IR data through

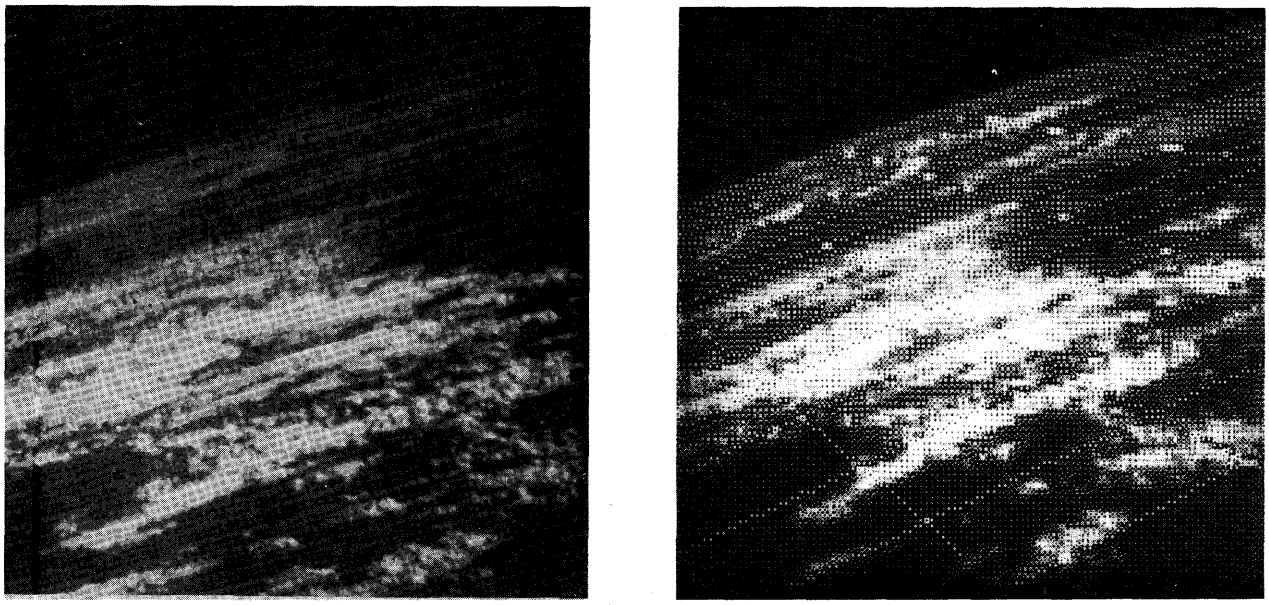


Figure 10. Cloud photo with melded grid (simulated): Original Hugo rocket photo (left) looking toward Bermuda from Wallops Island, Virginia at 85 miles altitude and 100 scan line digitization of the original picture (right) played back through a digital CRT (SC-4020) with 15 unit gray scale produced by programmed time modulation. Certain picture elements have been replaced by grid signals before playback to produce latitude/longitude lines.

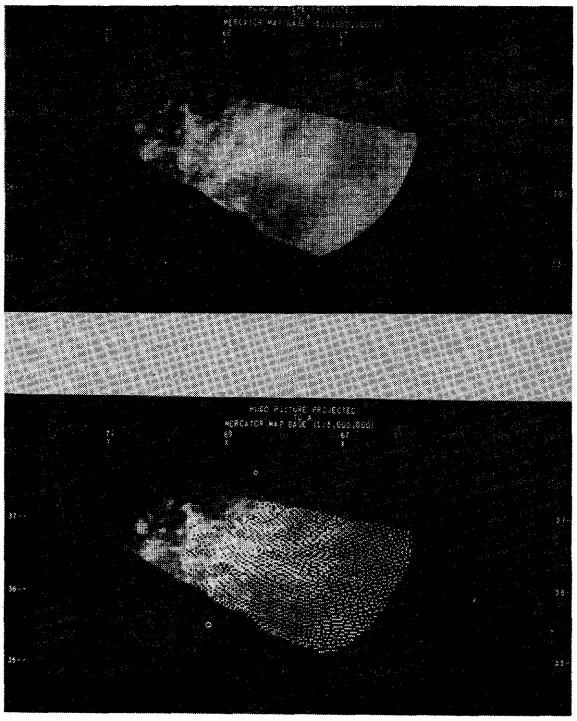


Figure 11. Rectified cloud photo: Digitized picture elements from figure 10 replotted on a Mercator map base without filler (below) and with filler (above) to produce a rectified pictorial image.

cloud top temperatures, present an added output. The MRIR package will yield other derived products such as maps of the net radiation flux. There is thus a family of derived products available from the digested material. A variety of output equipments including prototype cathode ray tube photo recording devices which are driven by digital tapes and somewhat similar photo quality facsimile machines, require additional conditioning of the output products to suit the formats specified.

The variety of production type programs are indicated in the appendix. It is likely that all such production varieties cannot be produced in real time from the data received on all passes of the satellite. The intent is that these products will be available for experimental utilization and that variations and modifications of those which prove to be most useful will assume an operational role.

CONCLUSION

This has been a brief attempt to present a background to the non-meteorologist explaining the need for more weather data, and the present and likely future role of weather satellites. The need for computers and automatic data processing is explained in terms of the kinds of data involved. Computer support of semi-hand methods is discussed along with current efforts toward a truly automated effort for Nimbus satellite data. As the variety of sensors and the volume of such data increases, a maximum degree of automatic processing and utilization of the data is indicated.

The scope has been limited to the data processing job as seen from the computer programmers viewpoint. Other groups within the Goddard Space Flight Center of NASA, the Weather Bureau's NWSC and their contractors have vital roles in the design, launch, command and readout of the satellite and the supplying of other important data in the form of sensor calibration and orbital information from tracking station data before the sensor data can be rendered meteorologically useful.

Only scant mention has been made of the entire data processing system. The second paper in this series will give additional details of the digital and non-digital data processing machine complex—again from the standpoint of the computer programmer. The role of the computer as manager of the process will be amplified in terms of command and control.

APPENDIX

The main program modules are listed below together with some details concerning each subroutine portion. The main section of each program module is indicated by an asterisk. Status of various portions is indicated as of September, 1962.

Executive Program

Details of the Executive Program are provided as part of the text of Part 2 of this paper.

Time - Attitude - Calibration Ingestion Program

*Time/Attitude Sort: Engineering housekeeping data on "A" channel including pitch, roll and yaw attitude signals and certain vehicle temperatures used in IR sensor calibration are transmitted as pulse code modulation (PCMA). Shutter times from the

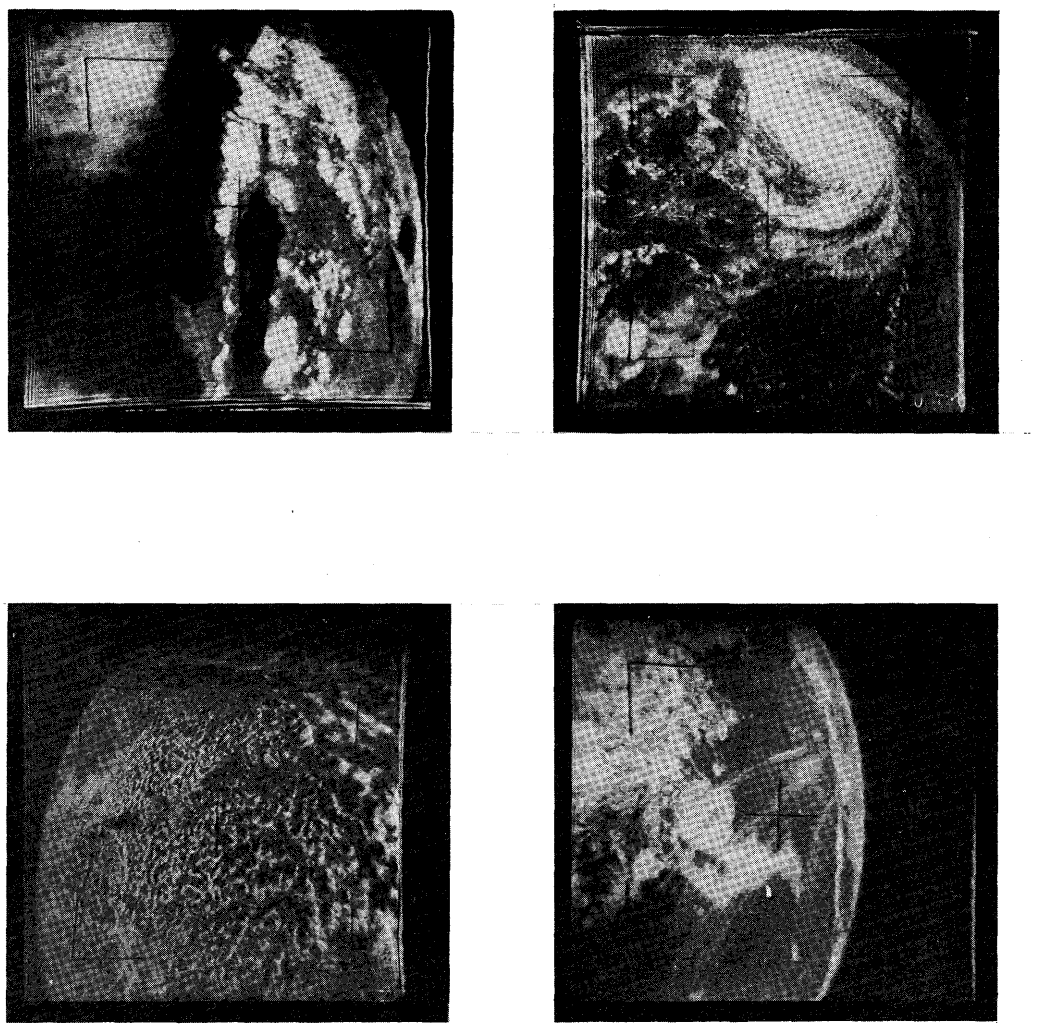


Figure 12. Sample TIROS cloud patterns. Convective clouds over Lower California (upper left) August 21, 1961. Classic hurricane symbol from cloud pattern of Hurricane Betsy (upper right) near 36°N, 59°W on September 8, 1961. Field of cellular clouds (lower left) near 25°S, 10°E on July 31, 1961. Cirrus cloud streamers off the Andes (lower right) passing eastward off the Argentine coast, August 3, 1961.

Advanced Vidocon Camera System (AVCST) are sent in similar format on another channel. This program will accept such information and sort it from an intermixed input format.

PCMA Unpack and Monitor: Unpacks the separate 7-bit raw count measurements and translates selected quantities into meaningful temperatures or angles. Items to be used are examined for quality and format with optional outputs for visual inspection.

PCMA Output: Organizes attitude, calibration temperature, and picture time information into tables and issues the information in a form suitable for use by the main data processing programs.

Time/Attitude Editor: Optionally accomplishes some of the above duties as required in the event that this information is made available in semi-processed form as a direct digital message.

This section is in an active design status awaiting final format of PCMA data and decision on items to be transmitted from Fairbanks, Alaska.

Picture Grid Tape and Rectification Program

Orbit: Based upon a specified time request, this subroutine supplies satellite altitude and latitude/longitude of the subsatellite

point. The information is generated as a prediction based upon periodically updated fundamental orbital elements which are supplied by the main NASA orbital determination through minitrack data.

Picture Attitude: Converts pitch, roll and yaw error signals into nadir and azimuth angles of each camera's principle line and also provides a radial displacement correction to the orientation of each raster.

Distortion: On the basis of prelaunch target photos, produces radial and tangential distortion corrections for a pre-selected family of image raster points so that, through interpolation, any image X, Y point can be expressed in terms of two component angles in object space.

Geography: Provides a large catalog of latitude/longitude points along all major coastlines of the world. The subroutine provides ordered groups of such points in short segments for quick selection. Such coastlines are optionally included with latitude/longitude lines in grids melded to the photos.

*Grid Meld and Rectification Locator: This is the main program segment. It includes the basic calculations which produce latitude and longitude from an X, Y image point. The subroutines above serve as input support. The primary output is approximately 1000 latitude/longitude locations from a pre-selected open lattice of image locations. These locations are available in table form for later interpolative rectification of the entire picture raster.

Grid Meld Output: For every sixth scan line of each picture raster, the locations of latitude/longitude line crossings are calculated. This information from one simultaneous three picture cluster is logically combined into a set of binary tape records containing a series of three-bit code groups and nine-bit count groups which tell where over-ride signals are to replace the picture signal and produce a dotted line grid.

One such orbit routine has been produced for TIROS. Revision awaits coordination with NASA orbital computation group as to mathematical model to be used for Nimbus. Geographic coastline tables from TIROS have been expanded to global coverage and are available for Nimbus. Other portions are active.

Line Drawn Grid Program

*Grid Line Locator: A program similar to the above but intended primarily for emergency use. It computes X, Y image points from pre-selected latitude/longitude intersections.

Line Output: Generates a special format tape for a model 3410 Electronic Associates Data Plotter.

Cathode Ray Tube Grid Program

*CRT Grid Locator: Essentially a duplicate of the Grid Line Locator above.

CRT Output: Generates a special format tape to guide the cathode ray tube beam to produce grids recorded on microfilm from devices such as the Stromberg Carlson Model 4020 Microfilm Recorder.

Both the line drawn and CRT grid programs have been completed as generalized versions of TIROS packages and are being used experimentally.

Digital Picture Ingestion

*Picture Sort: Digitized pictures arriving from the analogue to digital converter through the external format control unit will enter the computer in packed words. Each 36-bit word will contain 4-bit intensity measurements from nine consecutive scan spots all from the same picture. A cyclic commutation intermixes such words from the three cameras. This program sorts the information for output into separate files each containing information from only one camera. The following subroutines support this task and carry on added preprocessing functions.

Picture External Communicator: Picture data is being recorded at 7-1/2 inches per second into a bin tape recorder and the digital conversion process consults this tape intermittently at 30 inches per second. The external communicator is really an extension of the executive routine which sends out commands to stop and start the read capstan on the bin tape recorder.

Picture Monitor: Provides superficial checks to see that a signal is present, that raster line synch marks are clear, etc.

Unrectified Print: Produced by IBM 1401 printer will produce a visual check of the raster and its relationship to the fiducial marks, a single character corresponding to each scan spot.

Solar Ephemeris: With time of photo, provides the latitude/longitude of the subsolar point from which usable sun angles may

be generated for later interpretation of brightness, reflection properties and other attributes of the image.

Sun Glint: Used in conjunction with the Solar Ephemeris routine will earmark that part of any image where the response is primarily caused by sun glint.

Output to Storage: Will consist of routine output commands to the two disk channels but output of information is important insofar as efficient positioning of the write arm is concerned since a maximum net transfer rate is required.

Most parts of this module are active. The Solar Ephemeris has been completed as a more efficient version of a similar TIROS package. Input format and means of detecting ends of scan lines are being worked out in conjunction with final design specifications of the Format Control Unit.

Picture Digestion and Production

*Picture Rectification: Utilizes the output of the rectification locator program. Separate picture scan spots are repositioned in sub-blocks of storage according to grid squares on a standard map base. The following supporting packages are utilized.

Picture Selector: Provides input/output selection capability. A picture will be specified by exposure time and as left, right or center camera. A specification of core buffer location and picture segment will result in movement of the required item to or from disk storage.

Brightness Normalizer: Adjusts the image response for variations due to the scan electronics and also adjusts for pole to equator illumination differences.

Background: Provides an updated background response from which current responses will be treated as anomalies. In this way partial discrimination between cloud and background will be possible.

Interpolate: Provides an efficient quadratic interpolation within a two dimensional array. This package will be used extensively in connection with transformations from x, y image locations to i, j map grids.

Indexing: A flexible subroutine which permits identification of storage location as a function of i, j location in square mesh grid which is to be superimposed on a map projection.

Mosaicker: A routine which will combine rectified, summarized data in an overlap region based on priority selection rules.

Cloud Cover: Some 400 picture elements falling in a ten nautical mile grid square will be ranked as background, cloud or doubtful. Percentage cloud cover and average cloud cover brightness will be expressed as edited output.

Disjunction: Further interpretation of the data used for cloud cover analysis will express the areal variability of cloud cover thus distinguishing between scattered or broken cloud arrays in large contiguous masses as compared to other cases similar in net cloud cover but distributed in a more specular array.

Orientation: By comparing profiles of response within a ten mile square using samples taken from different radial orientations, certain streakiness and other features of the pattern can be deduced.

Stereo Map: Computer i, j coordinates on a specified square mesh grid on a polar stereographic map base for a given latitude/longitude point on earth.

Mercator Map: Similar to above but using a Mercator map base.

Grid Print Output: Prints out on standard IBM printer the various summarizations discussed above by using a character for each 10 mile mesh interval (square type and ten line per inch carriage control are desirable). By coding character selection, both quantitative and pictorial output can be obtained.

Line Drawn Output: Contoured fields are produced from magnetic tape on an Electronic Associates Data Plotter, Model 3410. Cloud height analyses will likely be produced by this device.

CRT Output: Similar to grid print output but utilizing a device such as the SC 4020 microfilm recorder.

Fax Output: Similar to the above but utilizing digital tape directly to drive a facsimile scan device.

Most program segments are active. The interpolation routine is in operation. The background package will be self generating after Nimbus launch in that clear air earth views will be accumulated as background information. Stereo and Mercator mappers have been produced. An experimental unrectified print package has also been produced.

MRIR Ingestion and Digestion Programs

Scan Rate: The scan shaft angle corresponding to a specific sensor sample can be deduced from a shaft angle reference pulse but is also dependent on knowledge of scan shaft spin rate and sampling frequency. This subroutine will be available on an optional basis to compute the spin rate by counting shaft reference pulses over a given number of cloud pulses.

*MRIR Ingestion: Manages input, partial processing and places raw product in intermediate storage.

Scanner Attitude: Similar to picture attitude routine but supplies a series of nadir and azimuth angles along a scan swath.

Space Cropper: From height supplied by orbit routine and roll correction, provides identification of IR samples with respect to scan shaft reference pulse thus permitting rejection of all but earth viewing sample.

Earth Locator: An adaptation of the picture locator package which furnishes latitude/longitude information from input provided by orbit and attitude routines.

Solar Sector: By using the solar ephemeris and location of viewed spot, provides solar angles for interpretation of data.

MRIR Data/Format Monitor: Inspects the raw data to detect format errors and to judge the general quality of the data (noise). Failure to pass acceptance tests causes visual output for further inspection.

*MRIR Format and Output: Creates the archivable intermediate source tape from which various output products are derived. This main portion utilizes the routines below and some of those above which cannot be utilized for want of time during the ingestive phase.

Calibration: A step-wise two dimensional array interpolation which produces effective black body temperatures from raw sensor counts as a function of environmental temperatures adjacent to the sensors and in the electronic data transmission equipment.

Documentation: Places appropriate identification on the archivable product including orbit number, date, time, etc.

Parts of this package that are also used with HRIR are active. Earth Locator and Calibration will be minor revisions of TIROS routines.

MRIR Production Programs

*MRIR Mapper: Consults the final Meteorological Radiation tape produced by the digestive programs and generates fields of derived quantities as indicated below. Also supervises the various output packages.

Cloud Height Analyzer: With the aid of a temperature height analysis based upon existing observations and climatology, provides a map of height information based on water vapor window measurements. This information is now available in consort with cloud photo information for further interpretation.

Limb Darkening: Provides corrections to sensor response as a function of viewing angle (path length).

Net Flux: Creates a map indicating the net radiative flux (incoming short wave vs. outgoing long wave) through a functional combination of sensor responses.

Albedo: Produces a map of reflectivity of the cloud patterns.

MRIR Print Output: These output programs are minor revisions of those mentioned for cloud photos.

MRIR Line Drawn Output:

MRIR Fax Output:

MRIR CRT Output:

This portion is generally not active pending decisions on availability of portions of data in real time.

HRIR Ingestion and Digestion Programs

*HRIR Ingestion and Format: A CDC 160A computer program which accepts packed raw count information, unpacks and edits the data with the help of the two routines below.

HRIR Space Cropper: A preliminary separation of earth and space viewing response is accomplished without specific height or attitude input in order to eliminate unwanted response without using a highly complex program on a small computer.

HRIR Format Monitor: Detects unsatisfactory quality of input data and optionally generates output for visual inspection (see similar MRIR routine).

*HRIR Digestion: Provides intermediate calibrated and geographically located data as indicated above for MRIR. Many of the subroutines cited above for MRIR are also applied directly to HRIR.

HRIR Calibration: A simplified version of the similar MRIR routine.

HRIR Format and Output: Generates the archivable product source tape. Single channel sensor output is arranged in a format somewhat different from that used for the multi-channel MRIR.

This module is active. The ingestive portions using the 160A is being carried out by contract with National Computer Analysts (NCA), Princeton, N. J. An internal segment of the HRIR Digestion package which precisely defines the earth viewed data sample is in check out.

HRIR Production Programs

These programs borrow heavily from the MRIR cloud height analysis and the photo cloud cover routines described above. Output routines will also be minor variations of those discussed.

Some output routines await word format specifications and instruction sets for prototype output hardware. Special character chains for computer printer output are being considered.

Picture Grid Melding Program

*CDC 169A Grid Meld: Provides synchronous recording of digital grid signals produced by IBM 7094 and the analog picture raster.

Time Check: Insures correspondence between gridding signals and pictures by input of PCM time groups direct from the analog picture tape and the comparable time information which accompanies the gridding signals.

Panel Documentation: Provides documentation information from the 7094 produced tape in proper format for output to the multitrack analgue picture tape such that a documentation panel is activated as the gridded picture is produced for film recording.

This segment is completed and awaiting non-digital equipment for final checkout. Details of Panel Documentation await final design specification of panel display device.

Simulation Support Programs

Certain non-operational programs are useful as feasibility and timing experiments while others produce interface input or output product samples which serve to check out segments of operational programs. Some of these have been produced:

AVCS Photo Rectification Study
HRIR FMRT Output Simulation
MRIR Raw Data Simulation
Executive Routine Test

Various phases of the photo rectification study have been completed including gray scale experiments on a digital CRT, filler experiments and obtaining timing figures.

Other Simulation Programs Test Hardware:

Passive Switching Exerciser (7094)
Active Switching Exerciser (7094)
Control Logic Communicator (for 7094 and 160A)
Format Control Test (for 7094 and 160A)
Analog to Digital Test (7094)
AVCS Picture Tape Test (160A)

These routines are awaiting final design specifications and specific control formats.

REFERENCES

American Meteorological Society, 1962: Statement on Weather Forecasting. Bulletin A.M.S., Vol. 43, N. 6, June 1962, 251.

Bandeen, W. R., 1962: TIROS II Radiation Data User's Manual Supplement. A & M Div., GSFC, NASA, May 15, 1962.

Dean, C., 1961: Grid Program for TIROS II Pictures. Allied Research Associates, Inc. Contract No. Cwb 10023, Final Report, March 1961.

Frankel, M. and C. L. Bristor, 1962: Perspective Locator Grids for TIROS Pictures. Meteorological Satellite Laboratory Report No. 11, U. S. Weather Bureau, 1962.

Fritz, S. and J. S. Winston, 1962: Synoptic Use of Radiation Measurements from TIROS II. Monthly Weather Review, 90 (1), January 1962.

Johnson, D. S., 1962: Meteorological Measurements from Satellites. Bulletin A.M.S., Vol. 43, N. 9, September 1962.

National Aeronautics and Space Administration and U. S. Weather Bureau, 1962: Final Report on the TIROS I Meteorological Satellite System. NASA Tech. Report No. R-131.

National Aeronautics and Space Administration and U. S. Weather Bureau, 1961; a: Abstracts and figures of Lectures and Reprints of Reference Papers. The International Meteorological Satellite Workshop. Washington, D. C., Nov. 13-22, 1961.

National Aeronautics and Space Administration and U. S. Weather Bureau, 1961; b: TIROS II Radiation Data User's Manual, August 1961.

Phillips, N. A., 1960: Numerical Weather Prediction. Advances in Computers, Vol. I edited by Franz L. Alt, Academic Press, 1960, 43-51.

Stampfl, R. A. and H. Press, 1962: The Nimbus Spacecraft System, to be published in Aerospace Engineering, 21 (7).

Winston, J. S. and P. K. Rao, 1962: Preliminary Study of Planetary Scale Outgoing Long Wave Radiation as Derived from TIROS II Measurements. Monthly Weather Review, 90, August 1962.

PROCESSING SATELLITE WEATHER DATA - A STATUS REPORT - PART II

Laurence I. Miller
U. S. Weather Bureau
Washington, D. C.

SUMMARY

Experience gained from earlier meteorological satellites provides a firm background for the basic design of the data processing center. Nevertheless, the almost limitless nature of the sampled data and some uncertainty as to the optimum forms of the final products dictate the need for providing the basic system with extreme flexibility and good growth potential. To achieve the desired versatility, the operation of the various portions of the system are being designed so that their functions are almost entirely programmable to facilitate rapid conversions to handle new types of data and cope with changing situations.

Maximum utilization of a computer's logical capabilities are stressed to avoid redundant construction of analog hardware and/or special "black boxes." An executive monitor program is designed to provide the necessary link between computer and external hardware. Emphasis is placed on the centralization of control and the modular design of the main programing packages.

INTRODUCTION

In Part I of this report reference has been made to the site of the data-processing center with only passing comment on the communication network and the system being designed to manage, edit, process and output the enormous volume of data. The data processing plan for the operational meteorological satellite, Nimbus, is the result of a continuing research and development program begun after World War II with German and American rockets and more recently includes the highly successful TIROS satellites. It is beyond the scope of this report to provide a detailed description of the TIROS satellites; however, Table 1 provides a ready comparison between some of the more salient features of the two systems and furnishes a foundation for the ensuing more detailed description of the Nimbus data-processing system.

Limited computer processing of TIROS data was discussed in Part I, and details of the difficulty of "real-time" computer processing of the information have been given elsewhere, along with an engineering description of the first TIROS satellite and a meteorological analysis of some of the data [4]. Equally as important a consideration in not preparing elaborate data-processing codes to handle the TIROS data was the limitation in speed and storage capacity of existent digital computers when the TIROS design was considered. The time required to compute a reprojected image of one complete photograph approached the elapsed time of one entire orbit [5]. Although attention will be given to this problem in a subsequent section, it hardly seems redundant to point out that computers of the present generation are still barely adequate to this task.

Table 1

Comparison of Nimbus and TIROS

	TIROS	Nimbus
Height (inches)	19	118
Diameter (inches)	42	57
Weight (pounds)	300	650
Orbital Altitude (Nautical miles)	380	500
Orbital Inclination	48° Equatorial	80° Polar
Stabilization	Spin-Stabilized	Earth-Seeking (3 axes)
Earth Coverage (%)	10-25	100
Camera Raster (lines per frame)	500	833
TV Resolution (miles)	1	1/2
Maximum Power Available (watts)	20	400
IR Sensors (resolution, miles)	MRIR (30)	MRIR (30)-HRIR (5)
Period (minutes)	App. 100	App. 100
No. of Cameras	2	3
Command Stations	2	1

The second part of this paper serves three purposes: to examine the logical layout of the central computer with associated peripheral equipment and external hardware; to describe the functioning of the data processing system, emphasizing the logical capabilities of the computer; to discuss the vital link between computer and external hardware provided by an executive monitor program.

DATA TRANSMISSION

Figure 1 is a generalized schematic representation of the flow of data from Nimbus to the National Weather Satellite Center (NWSC), Suitland, Md., via the command and data acquisition (CDA) station at Fairbanks, Alaska. The proposed transmission facility between Alaska and Suitland will utilize two 48 Kc lines, known commercially as Telpak B. The telemetry aboard the satellite provides information on the spacecraft environment and attitude as well as information from the three meteorological experiments. Data recorded on magnetic tape recorders aboard the vehicle are telemetered to the ground station using an FM-FM system to accommodate the considerable information bandwidth.

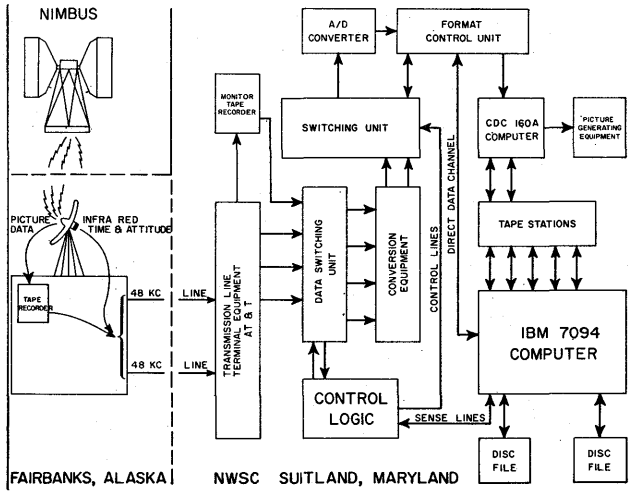


Figure 1. Schematic representation of the flow of data from Nimbus to the National Weather Satellite Center.

Somewhat different considerations apply to each of the multiple sensor and environmental signals as they are initially recorded on the spacecraft, telemetered to the ground and finally received at the transmission terminal equipment. These features are summarized as follows:

AVCS

Each of the three video cameras are simultaneously exposed for 40 milliseconds, scanned for 6.75 seconds and recorded on magnetic tape at 30 i.p.s. Although each exposure of the thirty-three frames (three picture set) are 108 seconds apart, only 3.7 minutes of actual recording time is required. Playback to ground is maintained at 30 i.p.s. but is recorded, still in FM form, at 60 i.p.s. Since the long line bandwidths are not sufficient to accommodate the frequency range,

the ground tape is rewound and then relayed to the NWSC in 30.85 minutes at 7.5 i.p.s.

HRIR

The narrow angle high resolution radiation sensor is active only during the dark southbound portion of the orbit of approximately 64 minutes. During this time data is recorded at 3.75 i.p.s. and then telemetered to the CDA station in 8.1 minutes at 30 i.p.s. The transmission is received at Suitland in 8.1 minutes; however, the data are recorded at 60 i.p.s.

MRIR

Five medium resolution radiation sensors scan from horizon to horizon during the entire orbit. An endless tape loop records the data continuously (except during readout) at 0.4 i.p.s.; increasing the playback speed by a factor of 30 reduces the readout time to 3.6 minutes. The data is recorded at Suitland at 30 i.p.s.

PCM

Space craft environmental signals, including attitude signals, vehicle temperatures and other housekeeping data, are transmitted as pulse code modulation (PCM). This information is also recorded during the entire orbit in a similar manner to the MRIR, discussed earlier.

The "real-time" aspects of the operation are accentuated by the undelayed transmission of the PCM and infrared data directly to the NWSC computers over the leased microwave facilities. The total time required for complete satellite interrogation is 8+ minutes; therefore, all but three to four orbits a day can be recorded at Fairbanks.* Transmission of the video data to Suitland is delayed about 10 minutes while the computers convert the raw PCM data to useful parameters; therefore, all the data is not received at the center until approximately 40 minutes after the start of interrogation. Direct access of the data to the IBM computer is accomplished by means of a Direct Data Connection (DDC), which permits real time transmission between 7094 storage and external devices at rates up to 152,777 words per second.

The NASA Space Computing Center at the Goddard Space Flight Center supplies a set of orbital elements, which are periodically updated by information received from the world-wide Minitrack network. Prior to satellite interrogation these elements are converted to satellite latitude, longitude and height as a function of the orbit time.

INPUT DATA

Before turning to a consideration of the high data rates as they pertain to the "real-time" system, let us briefly outline the presently proposed computer complex. The primary computer will be a 32,000 word core memory IBM 7094 equipped with the following elements: fourteen MOD V magnetic tape drives shared between two channels, two 1301 disk files each connected to a separate channel, one DDC attached to a tape channel, a core storage clock and interval timer, an on-line printer and card-reader. Two smaller scale computers will also be available, an IBM 1401 to serve primarily as an input-output device to the 7094, and a CDC 160A to be used in the picture-gridding program and to some degree as a preprocessor for the less voluminous MRIR and HRIR data.

Table 2 provides a summary of the volume and real-time rates (equivalent to 60 i.p.s. playback) of the experimental data, and Table 3 provides the data rates of the 7094 input-output equipment. From a consideration of the simultaneous input-output computing abilities of the 7094, and the effective use of optimum buffering techniques, it appears at first that the severest constraint to operational use of the data is imposed by the acceptance rates of the DDC and the temporary storage devices. However, closer examination of the basic machine cycle time (2.0 microseconds) and the frequency of main frame cycles borrowed by the input-output equipment reveals that insufficient editing, buffering and operational programming time would be available even if the basic acceptance and transfer rates could be appreciably increased.†

*The east coast of North America is being considered as a site for a second CDA station.

†The 7094 was selected as the result of a staff study which considered among other things delivery dates, performance and reliability, software, user groups, and especially speed and storage capacity.

Table 2

Satellite and Station Recording Rates

	Satellite			
	AVCS	HRIR	MRIR	PCMA
Record (min.)	3.7	64.8	108	108
Speed (i.p.s.)	30	3.75	0.4	0.4
Playback (min.)	3.7	8.1	3.6	3.6
Speed (i.p.s.)	30	30	12	12
Fairbanks				
Speed (i.p.s.)	60	60	60	60
Playback (min.)	30.85	Direct	3.6	Direct
Speed	7.5	-	60	-
NSWC				
Speed (i.p.s.)	7.5	60	7.5	-
Playback (min.)	30 (Batch)	8	.5	-
Speed (i.p.s.)	30	60	60	-

Table 3

Volume and Real Time Data Rates

	Binary Bits	Bits/second
AVCS	275,000,000	1,402,920
HRIR	14,700,000	App. 59,000
MRIR	3,600,000	134,400
IBM 729 Mod IV (high density)		375,000
IBM 729 Mod VI (high density)		540,000
IBM 1301 DISC		App. 500,000
IBM DDC		App. 1,000,000

The required high rate of data transmission is obtained by maintaining a continuous flow between the transmission line and the computers. The analog signals are detected, multiplexed and introduced to an analog-to-digital converter which encodes the sampled values in digital form while preserving the integrity and rate of the data. In the case of the video signals the data are recorded at 7.5 i.p.s. in a special bin storage recorder which permits the information to be read into the computer in batches at 30 i.p.s., well within the data handling capabilities of the computer.

DESIGN CONSIDERATIONS

During all phases of the system design it has been vital for us to consider both the high degree of flexibility and growth potential inherent in the Nimbus Research and Development program and the implications of future programs of international cooperation in weather satellites. Further, as the system passes from the experimental phase to the truly operational stage the degree of automation will increase and eventually replace manually performed functions. The required balance between these practical considerations and the need to assume an immediate operational posture has been achieved by designing the structure of the combined digital-analog complex as machine, not hardware, orientated.

To achieve the desired versatility, the operation of the various portions of the system are being designed so that their functions are almost entirely programmable to facilitate rapid conversions to handle new types of data and cope with changing situations. Emphasis has been placed on the modular concept so that substitution of one package for another does not have ramifications throughout the entire system. Maximum utilization of the computers logical capabilities have been stressed to avoid redundant construction of analog or special hardware. Wherever possible, major hardware units are standard, dependable general purpose equipment; and where it has been necessary to build special equipment, these are of the patch board variety.

CONTROL PHILOSOPHY

The Nimbus system has a common base with many other complex systems where computers are employed for such vital functions as information storage, retrieval and display. Inherent in most of these systems (e.g., BMEWS, SAGE, MERCURY) is a complex information processing problem which

requires intervention of skilled personnel to make the ultimate decision. These systems serve to provide a broad basis of facts on which the dominant information processor, man, can make his decision. Whereas these systems have been designed because it is possible to differentiate between the normal and abnormal, no such clear-cut definition exists in our weather system. Logical uses of pattern recognition theory and meteorological research may well negate this last remark, but such techniques are beyond the state-of-the-art at this time.

A second difference arises when we consider that the ingestive program is not engaged throughout the entire processing cycle, i.e., the time between successive readouts. During the ingestive phase (phase I) the external hardware may be completely active or passive or any combination of the two; during the non-ingestive process (phase II) the external hardware is predominantly passive. At any time during a processing cycle both diagnostic and management interrupts may occur, but the type of program control invoked must be considered in light of these two phases. Management interrupts which may occur at any time are caused by the normal transfer of data through the computer and must be given immediate priority. A component of the external hardware which monitors the system to prevent loss of quality or integrity of the information may also provide a diagnostic interrupt at any phase; however, the right to take action is reserved to the computer. During phase I the computer must be programmed to take immediate action; however, during phase II the suspected malfunction may be beyond the present logical flow of information, and the computer may merely advise a superviser and refuse to disturb the present operation. The monitoring and diagnostic control programs must be optimized as a function of the two phases.

At the time of initial launch when complete understanding of all possible system malfunctions is lacking, problems may arise which have not been anticipated. To cope with this situation a special manual mode of operation is provided which permits human intervention to apply recovery techniques. As a further "guard to the guards" a real time programmable clock senses the status of each phase and signals the present mode of operation.

It appears that the regularity of the data and uniformity of time scale should best be served by an automated system with minimum human intervention. This philosophy is controlled by an executive program which also provides the link between the computer and external equipment.

EXECUTIVE PROGRAM

The actual machine program consists of five main sections:

1. Internal Control: Coordinates and ties together the other portions of the executive monitor. It also requests other program modules from the system file and provides for operator override.
2. Schedule: Accepts pre-readout information concerning the data to come and establishes the time schedule and sequence of program modules to be consulted for that orbit.
3. Interrupt Interpreter: Diagnoses the interrupt from the standpoint of source and reason and directs the computer to the appropriate action. Interrupts may come from the clock, from the direct data connection interrupt wire, from the external interrupt or from regular channel commands.
4. Logical External Communicator: On the basis of clock alarms or otherwise, sends commands to control the mode of operation of the nondigital hardware. This routine is linked to the interrupt interpreter.
5. Clock Manager: Provides the means for setting the interval timer and causing clock interrupts and also fulfills program requests for time information.

However, the executive program is more than a series of machine instructions which controls the flow of information through the computer and the interaction between the main program modules. It is, in fact, the guiding philosophy of the entire data processing system. The program consists of a rigid set of rules and controls which determine the manner in which the various resources available are utilized in the satisfaction of the system design characteristics. At first glance, it seems paradoxical that the Nimbus system, always on the side of growth and flexibility should make such precise demands at the very heart of the system. Nevertheless, without such a firm foundation our system would be at best unstable and at worst completely unable to meet the specific

requirements of growth and flexibility from within the physical and environmental constraints imposed by the system. The original form of the executive program will be overlaid by many accretions, some of which may be major before we are through. The executive program will be the subject of a future paper.

SYSTEM DESCRIPTION

The functioning of the data processing system is best illustrated through a description of the events that occur during one orbital cycle. As the information is received at the common carrier terminal equipment, the data are directed into three main channels.

1. Into a monitor tape recorder which at all times records the input from the transmission lines at appropriate speeds. All data are stored as received providing a safeguard against loss of data in case of breakdown of the processing equipment. This tape also serves as an archive copy until replaced by the CDA master tape.
2. Into a picture gridding and reproduction branch, in which the analog AVCS signals can be directly reproduced in pictorial form, with the insertion of computed latitude, longitude and geographic boundary grids and appropriate legends.
3. Into a digitizing subsystem where the incoming data are formated, converted from analog to digital and transferred to the computers.

Twelve separate modes of operation appear at least once during each complete cycle.* Modes 1 through 11 occur (with considerable overlap) during the data ingestion phase when the primary role of the master computer is one of system command and control and only editing and minimum computations are performed. Mode 12 represents the time allocated to the main data processing programs which are described in the appendix to this report. During this phase the executive program continues to provide the link between the program modules. However, the master computer relinquishes control of the external hardware and peripheral computers to allow manual control for special functions, e.g., archival operations, preventative maintenance. Thus, it can be seen that approximately 60% of each cycle is available for computation during which system software is minimized so as not to interfere with the program's capacity to perform the basic function. The modes for this system are shown in Figure 2 and are as follows:

Mode 1. Initial load - receive and process preinterrogation message and compute orbital track. Activate monitor tape recorder and generate modes 2, 3, 4.

Mode 2. Receive HRIR picture and time data and switch this information to tape bin recorder #1. This mode is terminated by the computer upon receipt of an end of transmission code.

Mode 3. Receive PCM-A data and switch to demodulator and decoding circuits which convert the data to digital form. Transfer the information through the Format Control Unit (FCU) to the 7094. The computer senses the end of data to terminate the mode.

Mode 4. Receive AVCS time and direct the information to demodulator and decoding circuits which convert the amplitude modulated time information to digital form. The data is switched to the computer via the FCU. Upon receipt of all pictures the computer ends this mode.

Mode 5. Playback the HRIR data as soon as it is recorded and dumped into the bin (Mode 2). The information is converted to digital form and routed through the 160A computer to produce an edited digital tape.

Mode 6. Receive MRIR data and record on bin recorder.

Mode 7. Playback MRIR data to digitizing sub-system.

Mode 8. Receive AVCS data and record on bin recorder at 7.5 i.p.s. allowing tape to fill up bin.

Mode 9. Playback AVCS data from bin recorder in short bursts at 30 i.p.s. through digitizing sub-system to 7094.

Mode 10. Receive AVCS data and switch information into picture gridding and reproduction branch.

Mode 11. Transfer HRIR digital tape from 160A to 7094.

Mode 12. Relinquish automatic control of the external equipment. Process the data.

Although, the limited goals of data storage and display are accomplished in quasi-real

*An extra burden is placed on the system when two orbits are stored aboard the spacecraft and simultaneously acquired by the CDA station. An alternate mode is provided but will not be treated in this paper.

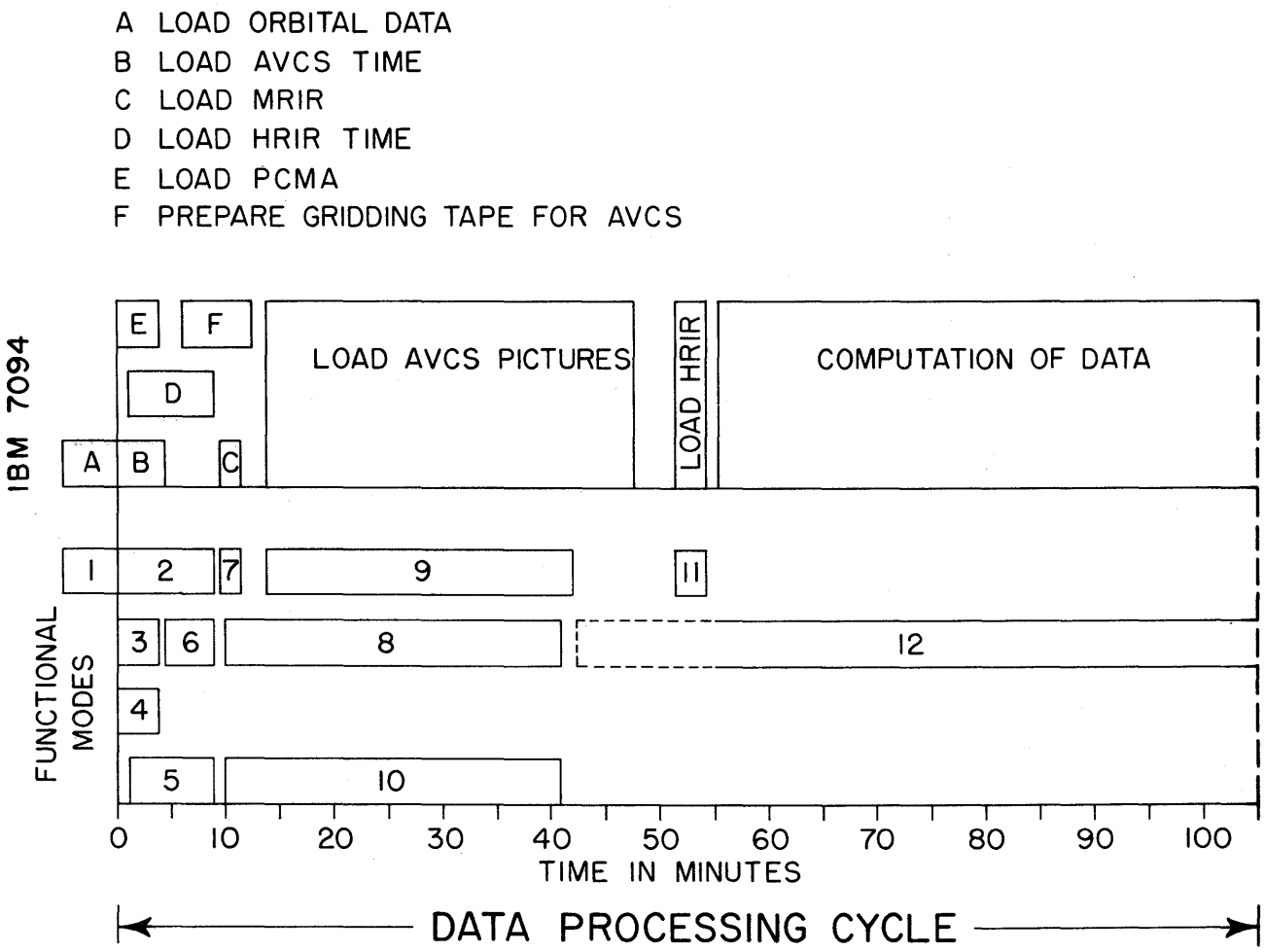


Figure 2. Functional modes and computer usage diagram.

time this represents only a partial fulfillment of the system design. The most important function of the computer will be the summarization of the data into convenient products for use in weather analysis and forecasting. The meteorological information will be disseminated as photographs, maps and charts, and coded teletypewriter analyses over domestic and international networks. True justification for the system design is possible only if we include this capability to obtain upon programmed demand these desired outputs, properly formated and to communicate this information to the outside world.

It is planned to distribute data to meteorologists in the following forms:

1. Gridded photographs (gridded meaning having latitude and longitude lines).

2. Mosaics, one for each orbital swath on a scale of about 1:10,000,000. The resolution of these mosaics will be about 10 miles. Probably three base maps will be made: polar stereographic for northern and southern hemispheres and Mercator for equatorial regions.

3. Mosaics similar to above for North Atlantic, North America and possibly other areas. This item has lower priority than 2 and it may prove easier for stations to prepare their own mosaics. Scale may possibly be 1:20,000,000.

4. Infrared maps similar to 2, from HRIR data.

5. Infrared maps showing cloud heights from MRIR data. Scale possibly 1:20,000,000.

6. Graphical nephanalyses for stations lacking capability of receiving more detailed data.

7. Coded nephanalyses for stations having only radio telegraph or radio teletype.

Distribution will be on a selective basis so that to the greatest extent possible each user will receive only the data he desires. Although additional communication links will be provided distribution to many overseas sites will necessarily be limited to radio, including radio facsimile.

REFERENCES

1. Davis, R. M., "Methodology of System Design."
2. Gass, S. I., Scott, M. B., Hoffman, R., Green, W. K., and Peckar, A., "Project Mercury Real-Time. Computational and Data Flow System," Proceedings of the Eastern Joint Computer Conference, Dec. 1961.
3. Hosier, W. A., "Pitfalls and Safeguards in Real-Time Digital Systems," Datamation, April, May 1962.
4. National Aeronautics and Space Administration, U. S. Weather Bureau, "Final Report on the TIROS I Meteorological Satellite System," NASA Tech. Report No. R-131, 1962.
5. Frankel, M. H., and Bristor, C. L., "Perpective Locator Grids for TIROS Pictures," Meteorological Satellite Laboratory Report No. 11, U. S. Weather Bureau, 1962.
6. Hall, F., "Weather Bureau Preliminary Processing Plan."
7. Ess/Gee, Inc., "Nimbus Data Digitizing and Gridding Sub-System Design Study," U. S. Weather Bureau Cwb 10264.

DESIGN OF A PHOTO INTERPRETATION AUTOMATON*

W. S. Holmes
Head, Computer Research Department
H. R. Leland
Head, Cognitive Systems Section
G. E. Richmond
Principal Engineer
Cornell Aeronautical Laboratory, Inc.
Buffalo 21, New York

INTRODUCTION

The extremely large volume of photographic material now being provided by reconnaissance and surveillance systems, coupled with limited, but significant, successes in designing machinery to recognize patterns has caused serious consideration to be given to the automation of certain portions of the photo interpretation task. While there is little present likelihood of successfully designing machines to interpret aerial photographs in a complete sense, there is ample evidence to support the conjecture that simple objects, and even some complex objects, in aerial photographs might be isolated and classified automatically. Even if machinery, produced in the near future, can only perform a preliminary sorting to rapidly winnow the input volume and to reduce human boredom and fatigue on simple recognition tasks, the development of such machinery may well be justified.

The supporting evidence for the conjecture that simple objects can be identified in aerial photographs is based on work which has shown experimentally that present pattern-recognition machinery—indeed that which existed several years ago—can be applied to the recognition of silhouetted, stylized objects which are militarily interesting. Murray has reported just such a capability for a simple linear discriminator.† Since the information required to design more capable recognition machines is readily available, it might seem that there is no problem of real interest remaining to make a rudimentary photo-interpretation machine an accomplished fact. This, unfortunately, is not so. One of the most difficult problems is that which is referred to as the segmentation problem. The problem of pattern segmentation appears in almost all interesting pattern recognition problems, and is simply stated as the problem of determining where the pattern of interest begins and ends (as in speech recognition problems) or how one defines those precise regions or areas in a photo which constitute the patterns of interest. The problem exists whenever there is more than one

*This work was sponsored by the Geography Branch of the Office of Naval Research and by the Bureau of Naval Weapons.

†See "Perceptron Applications in Photo Interpretation," A. E. Murray, Photogrammetric Engineering, September 1961.

simple object in the entire field of consideration of the pattern recognizer. The situation appears almost hopeless when one finds patterns of widely varying sizes, connected to one another (in fact or by shadow), enclosed within other patterns, or having only vaguely defined outlines.

This paper constitutes a report on a system which has been conceived to solve some of these problems. It is being tested by general-purpose computer implementation. The system discussed represents one of several possible approaches to the problem and had its design focused towards the use of presently known capabilities in pattern recognizers. No special consideration has been given, at this time, to methods of implementing the device; however, the entire system can be built in at least one way.

System Principles

Figure 1 is the basic block diagram for the system. It has evolved from evaluation of possible approaches suggested by research

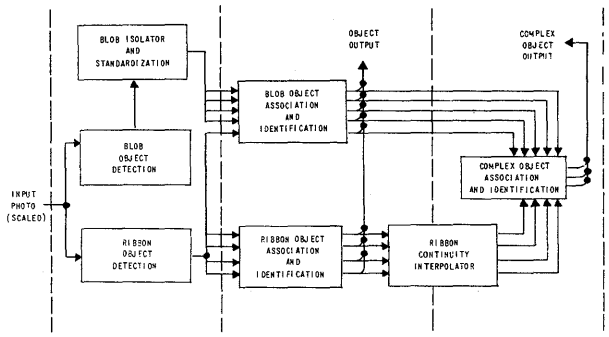


Figure 1. Photointerpretation system block diagram.

conducted at CAL, pattern recognition work of others, and techniques successfully used in other problems.

As is evident from Figure 1, objects of interest have been categorized in two different ways. First, simple objects, such as buildings, aircraft, ships, and tanks have been distinguished from complexes, or complex objects. Second simple objects have been categorized, according to their length-to-width ratios, as being either blobs (aircraft, storage tanks, buildings, runways) or ribbons (roads, rivers, railroad tracks). As shown, the detection of simple objects is accomplished separately for ribbons and for blobs. In the work reported here the blob channel—from the input end through the identification of a few complex objects—is receiving the major attention.

The preprocessing which is carried out in the first portion of the system solves several of the problems inherent in the use of a simple pattern-recognition device to aid in the photo interpretation problem. Briefly, objects are to be detected, isolated, and standardized so that they can be presented separately (not necessarily sequentially) for identification.

The function performed at the object identification level is that of identifying the blobs which have previously been detected, isolated, and standardized. The input material to this level or state consists of black-on-white objects. As has been previously indicated, existing devices are fundamentally capable of accomplishing the identification task.

At the complex object level, the location and identification information available from the simple object-level outputs is combined and appropriately weighted to identify objects at a higher level of complexity. An illustrative example is the combination of aircraft (simple objects) near a runway (another simple object) and a group of buildings (each a simple object) to determine the existence of an airfield.

In the following sections the basic steps in the preprocessing sequence will be described in more detail and some illustrations from current computer studies will be discussed. The most difficult part of the problem, by far, is that of detection.

Object Detection

A study of sample aerial photography suggests three ways in which images of objects of interest differ from their backgrounds:

a. points on objects may differ in intensity from the intensity characterizing the background.
b. objects may be (perhaps incompletely) outlined by sharp edges, even though the interior of the image has the same characteristic intensity as the background.
c. objects may differ from background only in texture, or two dimensional frequency content.

Examples of the first two kinds of objects are shown encircled in Figure 2. There

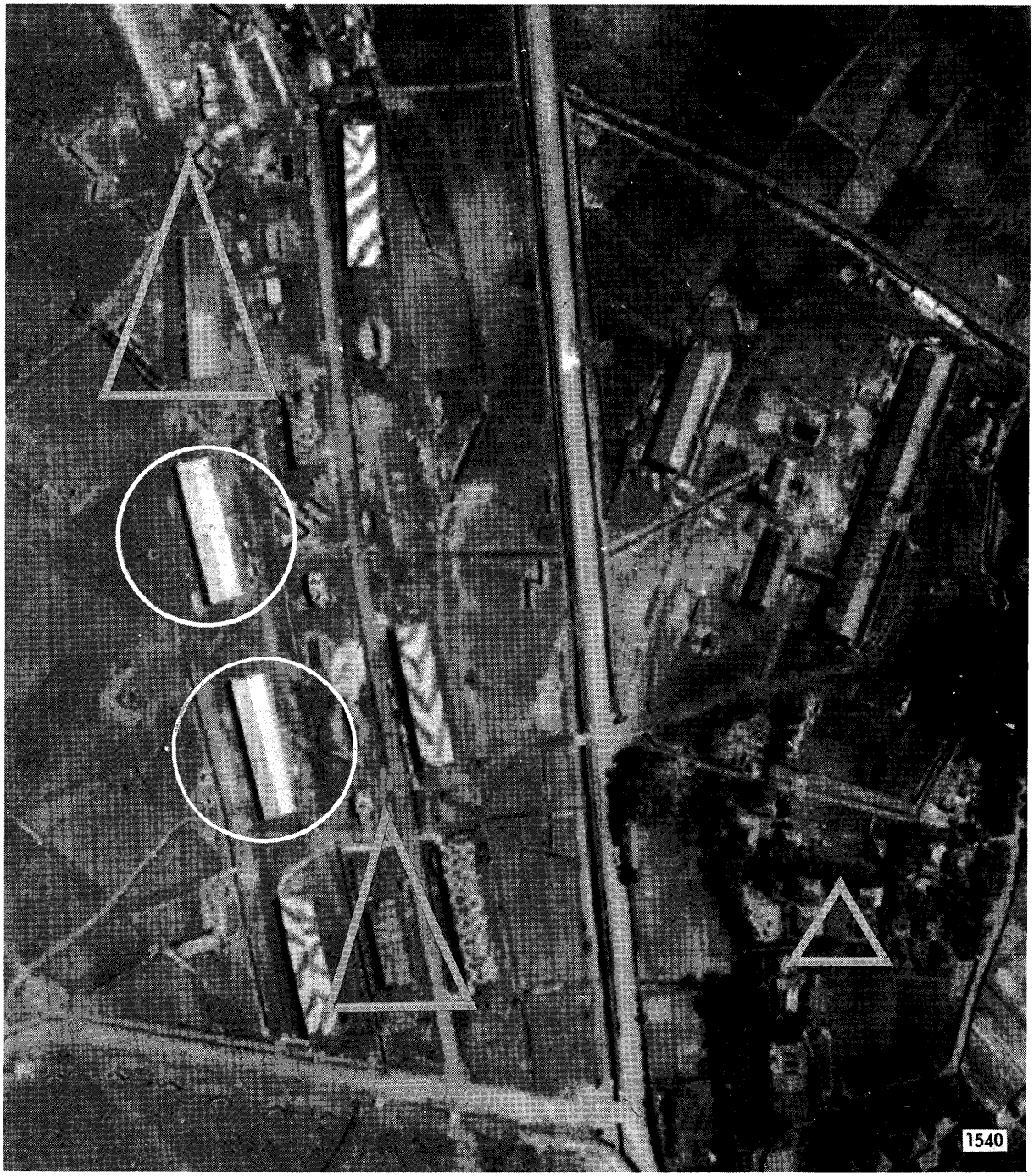


Figure 2. Examples of objects defined by intensity contrast (0) and by edges (Δ).

seem to be many fewer examples of objects which differ from background solely by texture. This class of objects would be much larger if our definition of object were broader, including, for example, corn fields. Perhaps the most useful area in which spatial frequency content can be put into use is that of terrain classification. Terrain classification, as will be noted again later, can play a significant role in the final identification of our narrower class of objects.

For detection of objects in classes a. and b., we have been proceeding experimentally to determine the capabilities of simple,

two-dimensional numerical filters, some nonlinear and some linear.

For initial experimentation,* the object filters for discrimination based on intensity contrast (class a objects) were designed as shown in Figure 3. Square apertures ("picture frame" regions) were used to compute

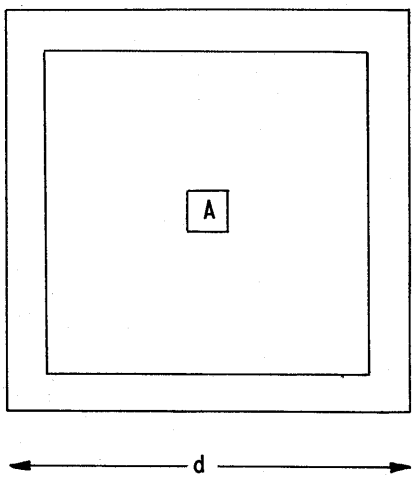


Figure 3. Filter for detection on the basis of intensity contrast.

*The experimental work reported was carried out using IBM-704 computer programs which were prepared to process photographic material. An input device was constructed to scan and quantize photographic information for input to the computer through the "real-time" package, and the computer printer was used to provide pictorial output.

A commerically available facsimile transmitter capable of 50 lines/inch resolution and a commercially available analog-digital sampler and encoder form the basic input device package. In addition, the necessary isolating and synchronizing circuitry has been designed, and constructed to permit the output of the facsimile machine (through the encoder) to be read by the "real-time input package" on the 704 computer. Quantization and processing computer programs have been written and are in operation. These programs cause photographs to be sampled every 50th of an inch, quantized in sixteen intensity levels, and stored on magnetic tape. A relatively crude, interim-nature, output has been arranged by using combinations of symbols available in the computer printer. The available symbols allow representation of four different levels of intensity.

intensity information which was then compared with the intensity of the point at the center of the square, A, to determine if the central point differed sufficiently in intensity from its background to qualify as being a point of an object.

A computing method equivalent to the following was used. Each point in the input photograph was surrounded by a frame one point thick, and of width d (Figure 3). The mean, m, and standard deviation, σ, of the intensity of the points in the frame were then computed.

If

$$A > m + \max(1, K\sigma)$$
$$\text{or} \qquad (1)$$
$$A < M - \max(1, K\sigma)$$

the point was recorded as an object point. Several different frame sizes were used in order to detect objects of different sizes.

Figures 5, 6, and 7 indicate the results of applying three filters of the type described above to the photograph shown in Figure 4. The frame widths were 8, 16, and 32 points,

Figure 4. Original photograph.

and K of equation (1) was 2. The points which satisfied the inequalities of (1) were printed as asterisks.

The three figures illustrate that objects of different sizes are detected best (with least shape distortion) by filters of different size. This is especially noticeable for the

building complex in the lower half of the photograph. In Figure 6 (d = 16), the buildings are reproduced in perfect contrast about as well as can be expected considering the coarseness of the input information. In Figure 5, the buildings are broken up into segments, while in Figure 7, they tend to run together.

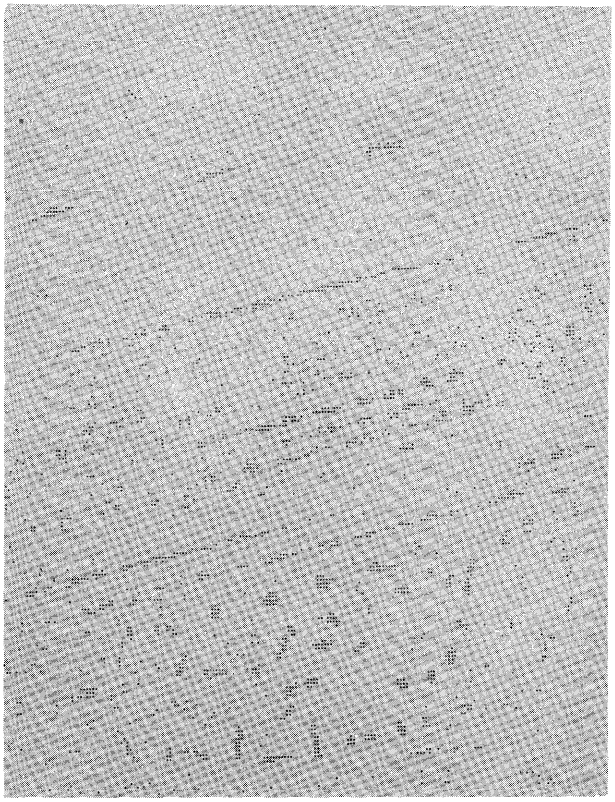


Figure 5. Filtered photo, d = 8.

The seaplane launching ramp at left center is missing completely from Figures 5 and 6, while the filter which matches it well in size reproduces it in Figure 7.

It is important to note that the recognition logic used requires only that an object be detected by a single filter. Distorted versions which are detected by other filters will be rejected.

Simultaneously with experimentation in detecting objects using the object-point-intensity criteria, similar experiments are being carried out to detect objects by outlining them. There are three steps in this process; (1) object edges must be "detected," (2) gaps in the outlines of objects must be filled in, and (3) for compatibility with the first method of object detection in later system stages processing all detected objects, the outlined objects must have their interior space filled in to produce silhouettes.

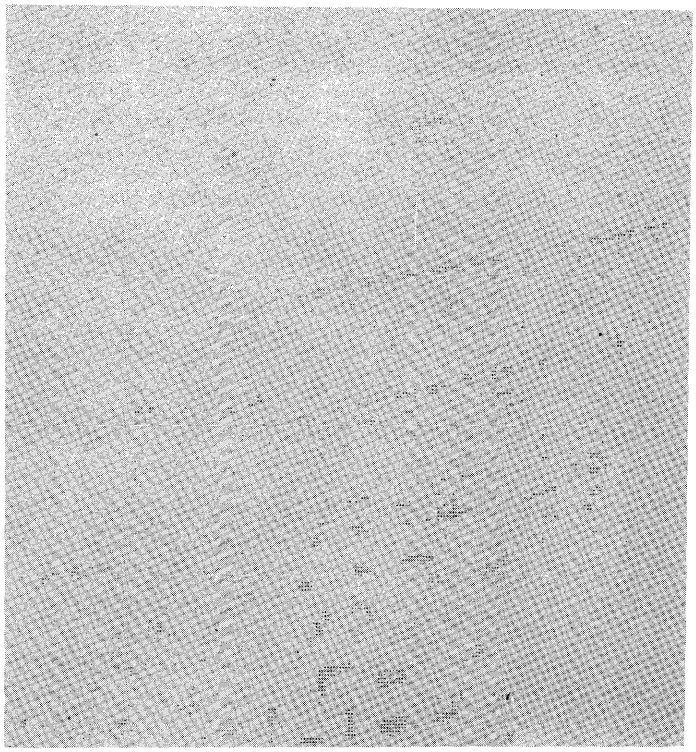


Figure 6. Filtered photo, d = 16.

Figure 7. Filtered photo, d = 32.

The basic operation in edge detection is, of course, differentiation. The earliest

results were obtained by centering a numerical filter of the shape shown in Figure 8 about each image point. The values of intensity,

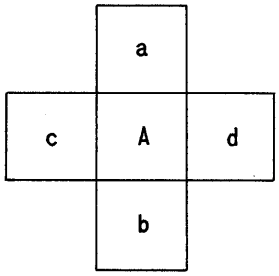


Figure 8. Basic filter for edge detection.

(d - c) = Δx and (a - b) = Δy were determined and the sum of their magnitudes was taken as the gradient associated with the center point, A. A similar filter with nine elements is now being tested with superior results. This filter has the form shown in Figure 9.

a	b	c
d	e	f
g	h	i

Figure 9. Improved filter for edge detection.

Now the difference in the x direction is taken to be

$$\Delta x = \left(\frac{c + f + i}{3}\right) - \left(\frac{a + d + g}{3}\right) \qquad (2)$$

and the difference in the y direction as

$$\Delta y = \left(\frac{a + b + c}{3}\right) - \left(\frac{g + h + i}{3}\right) \qquad (3)$$

Thus first differences are being used, as before, but a three-point average of intensity is used to establish the intensity on either side of the central point. The magnitude of the gradient associated with point e should, of course, be

$$|\text{grad}| = \sqrt{\Delta x^2 + \Delta y^2} \qquad (4)$$

The previous approximation to the true form $\sqrt{\Delta x^2 + \Delta y^2}$ has been improved over the simple sum of magnitudes in that we now use

$$|\text{grad}| = (\text{larger of } |\Delta x|, |\Delta y|) + 3/8\ (\text{smaller of } |\Delta x|, |\Delta y|) \qquad (5)$$

If object detection by identification of edges is to be successful, one must plan on completely outlining objects of interest. In many cases, of course, there will be gaps in the outlines of objects as derived by edge detection. One procedure currently being evaluated for this gap-filling job is described below. It accounts for the two factors which are most important in deciding whether to fill in a point or not; that is, such an action requires both proximity in intensity to the threshold value and proximity in space to at least one other super-threshold point.

After gradient computation, as described above, the complete image, made up of points computed by Eq. (5), is thresholded, eliminating low gradient points. The "influence matrix" (Figure 10) is then centered over every point in the thresholded gradient image (i.e., it is centered over high gradient points),

I_1	I_2	I_3
I_4	I_5	I_6
I_7	I_8	I_9

Figure 10. "Influence" matrix for gap-filling.

and the numbers $I_1, I_2, I_3, \ldots$ are added to the value in the pre-thresholded form of the gradient image. If any point covered by the influence matrix now exceeds the previously used threshold, that point is "filled in" as a high gradient or edge point.

It would, of course, be possible to train a recognition device to identify outlines of simple objects, but a much simpler system will result if outlined objects can be simply converted to solid objects similar to the silhouettes produced by the annular filter detectors. This can be accomplished very simply by forming the logical complement of the thresholded, edge-detected, binary picture and then operating on the complemented picture with the object isolator programs.

Object Isolation

Originally computer routines which traced along the edges of silhouetted objects were

planned for use in object isolation. This technique for isolation, however, does not solve the problem of how to extract the interior portion of the traced-out object from the background in any neat fashion. A different technique, devised by Mr. Richmond, simultaneously traces through the interior of objects and records these elements in a frame for separate storage. At this stage, that is, after isolation, all images of objects are stored in binary form, in separate frames, and in their original size, orientation, and location within the frame.

Object Standardization

Standardization involves simply the translation of the binary image of the object so that its center of gravity coincides with the center of the frame and rotation of the image so that one of its principal axes of inertia is vertical. Recently, the programs being used in feasibility studies have been modified so as to provide scaling of all objects to the same maximum dimension.

Object Recognition

In the system being discussed, recognition of simple binary images, after detection, isolation, and standardization, will be accomplished by a linear discrimination device, i.e., by comparing the weighted sum of a set of property values to a threshold. The weights used are determined by exhibiting a sequence of patterns whose classification is known and adjusting the weights when classification is incorrect, according to prescribed algorithms, until all patterns in the sequence are correctly classified. Thus, the device is "adaptive" and "learns." The properties may be thresholded sums of intensity at randomly selected points in the preprocessed image, or they may be more "objective" properties, that is they may be measured values of such determinable features of the pattern or image as maximum extent, area, or moments about principal axes. Certainly, use will be made of the size, area, and moment information derived during the standardization process.

The non-determinable properties mentioned earlier (thresholded sums of intensity at randomly selected points in the image) have the appeal of being very simple to derive and of being of demonstrated usefulness in classification problems. The ability of a system using such properties to generalize over pattern distortion and small translations has not yet been defined to our satisfaction. A recognition system using these non-determinable properties has been referred to by Rosenblatt as a simple perceptron. In our experimental work to date on this particular problem we have attacked the multiple class recognition problem by performing a set of dichotomizations. Some data on recognition capability have been gathered for synthesized patterns of the type to be produced by the preprocessor, but they are insufficient to make an explicit statement of capability at any reasonable confidence level.

Two types of information are fairly readily available in the system and have apparent use in classification but have not yet been used. Thus, we could use the silhouetted image of an object to mask out all but that object in the original full gray scale image and then derive one or more object properties available in the original image. (For example, one might scan the interior of an object, to derive some measure of its interior complexity.) This technique makes available recognition properties other than those based on shape. The second possibility is to use terrain classification information for the immediate background of an object to aid in the classification of that object. Certainly a final system might find such information useful.

Illustrative Results

The entire sequence of preprocessing operations (including detection by intensity contrast, but not object detection using edges) is illustrated in Figures 11-14. These results were derived from experiments which use general-purpose digital computer programs to implement the entire sequence of operations. The first, Figure 11, shows the original aerial photo. It has been quantized spatially and fed into CAL's IBM-704 computer through the facsimile input device. After processing by the intensity contrast filter, the binary photo of Figure 12 is produced. (The output of the 704 printer; an asterisk is printed in each 1/50" × 1/50" cell of the original photo which is a point on an object.) We found that some additional low pass filtering is very useful in making

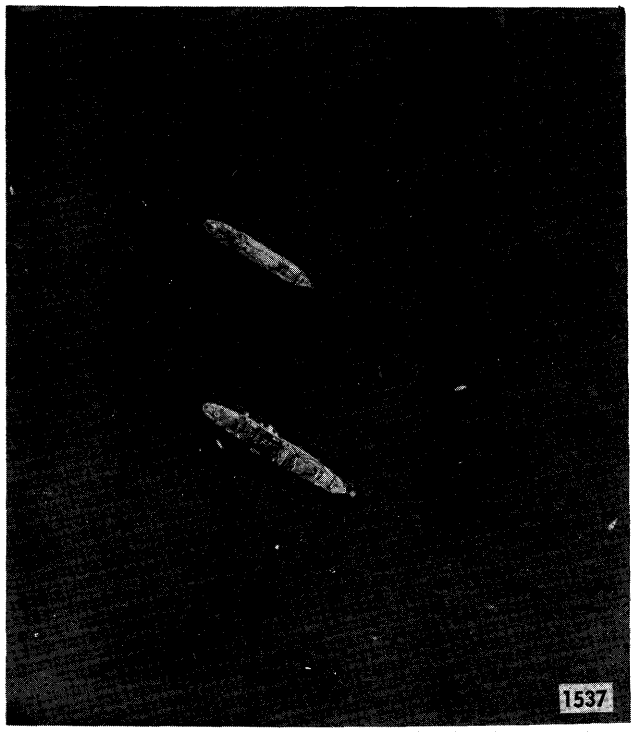


Figure 11. Original photograph.

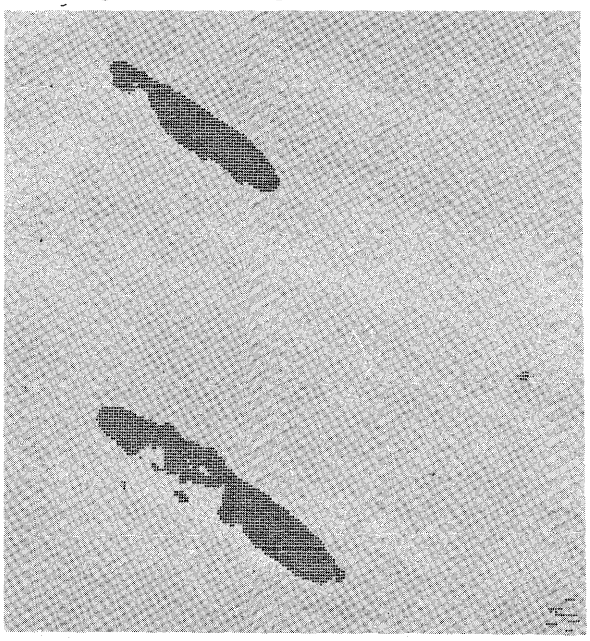


Figure 12. Processed photograph after object detection and low-pass filtering.

objects "hang" together, in filling in imperfections in silhouettes, and in reducing the number of small collections of points which appear but which are really not objects. After this filtering (a simple, low-pass, two-dimensional filter is used) and after eliminating collections of very few points, the binary photo was subjected to the isolation programs. This operation produced the frames shown in Figure 13. Each of these

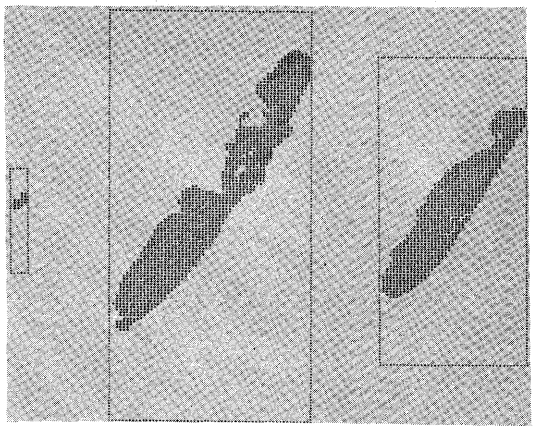


Figure 13. Isolated silhouettes from processed photograph.

several frames from the silhouetted photo was then subjected to the rotation and translation programs and the corresponding frames of Figure 14 produced.

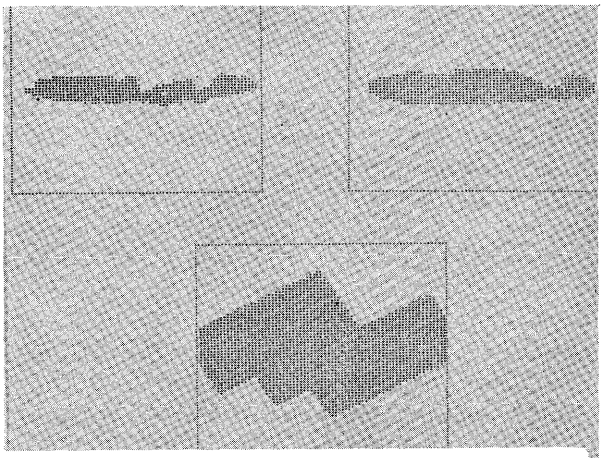


Figure 14. Standardized form of isolated silhouettes.

Now recall that standardization processing (1) fixes the center of gravity of a blob within the frame, (2) rotates it to align a major axis of inertia with the vertical in the frame, and (3) adjusts scale factor to roughly fill the frame. All information about how much translation, rotation, and scale change has of course been preserved for use in the recognition process (size is an input to simple target recognition, while location and orientation are inputs to target-complex recognition). Examining Figure 14 we find one odd looking building-shaped blob which must be

explained. Referring back to Figure 13 we can locate the source of this standardized object, a small collection of points. Mentally treating each of these points as a square, rotating and enlarging the resulting shape shows that the standardization routine functioned properly. Scale change information would prevent recognition as a building.

SUMMARY

In this paper approaches to the preprocessing portions of a photo interpretation automaton have been discussed. Clearly, some extremely important evaluative work remains:

1. Detection capability must be quantitatively defined. This first requires that some plausible criterion or criteria for this capability be defined.
2. Recognition capability must be quantitatively defined. Here there exists a body of evidence that recognition of silhouetted objects is within the recognition capability of current state-of-the-art systems. We are currently applying measures similar to probability of detection and false alarm rate to the definition of recognition capability for a property-list, linear discriminator type of system.
3. Implementation problems for a prototype system must be solved. Our IBM-704 work is for feasibility only. We have kept implementation problems in mind during the current system studies and have carefully avoided using system elements which are unduly complex. As an example, more complicated two-dimensional filters for object detection represent a very real temptation, yet we have exercised restraint and used only the simplest ones which we felt held any hope.

What has been achieved is a demonstration that a plausible system, combining current state-of-the-art pattern recognition capability and simple two-dimensional preprocessing operations, can be stated in specific terms and that it represents the very real and likely prospect of providing automated aid to photo interpreters.

EXPERIENCE WITH HYBRID COMPUTATION

E. M. King, Jr., and R. Gelman
Missile and Space Division
General Electric Company
Philadelphia 1, Pennsylvania

INTRODUCTION

In recent years, there has been increasing emphasis on the use of electronic analog and digital computers in the fields of science, engineering, and education. Interest has also been generated in computational methods not readily available from either of these classes of computers alone, but rather from the combination of them provided by the so called hybrid computer system. This interest has been a direct result of the increase in the complexity of present day systems and hardware, and the increased accuracy required of them. Based on two years of experience with a large hybrid facility, this paper will review the main characteristics of digital and analog computers, compare their capabilities and their limitations, and show some of the reasons for and advantages of mating the two to form the hybrid computing system. We will describe some of the problems to which we have successfully supplied hybrid solutions, and predict the direction of future hybrid progress.

Simulation

In general, there are two distinct modes of simulation; mathematical and physical.

Mathematical simulation utilizes a mathematical model of the physical system under study. This model is excited by certain ordered inputs and provides information about the behavior of the physical system, to the extent that it is correctly represented by the mathematical model.

Physical simulation requires the excitation of the system under conditions which are representative of those encountered in actual system operation. This testing can involve anything from an inclined plane to large multi-million dollar ventures like the Space Environmental Simulator located at General Electric's Valley Forge, Penna., Space Technology Center. These two types of simulation can be combined by mating physical hardware with a mathematical model.

The general purpose computers available today are primarily designed for mathematical simulation. They are ordinarily quite capable of simulating our physical systems with a depth and accuracy comparable to that obtainable by building and testing the actual equipment. As such, the general purpose analog and digital computers have allowed engineering to advance further and faster than ever before. It is, therefore, a definite requirement that any new entrant to these fields become familiar with these basic machines; their abilities, their limitations, and their possible use in his endeavors.

The Analog Computer

An electronic analog computer is an array of computational building blocks, or modules, each being able to perform a particular mathematical operation on an input voltage signal and provide a specific output response. These building blocks normally provide the functions of summation, integration with respect to time, multiplication by a constant, multiplication and division of variables, function

generation, generation of trigonometric functions, and representation of system discontinuities. All quantities are represented on the analog by continuously varying voltages, restricted on almost all analog computers to the range between -100 and +100 volts.

The ease with which this computer is able to perform the operation of integration makes it ideally suited to the solution of differential equations, as the differential equations can be integrated a suitable number of times to form integral equations, and these integral equations can be represented on the computer. Data are fed into the analog computer in the form of parameter settings, which are usually associated with the coefficients that exist in the mathematical equations. Data are extracted from the computer in the form of voltages, either as steady-state values which can be read out on a voltmeter, or as varying values which can be recorded on a strip chart recorder or a plotting table. Some of the analog characteristics pertinent to our discussion are:

1. The analog is a parallel machine. All the variables are computed simultaneously and continuously. Thus, the speed with which the calculations are made is completely independent of the size or complexity of the problem.

2. The bigger a problem is, the more equipment is needed, as each piece of equipment works on one part of the problem.

3. Numbers on the analog are fixed point. Every variable must be scaled. The scaling will greatly affect the accuracy of the results.

4. The analog is best suited for solving systems of ordinary linear differential equations, although it can handle many other types of problem in a very satisfactory way.

5. There is no such thing as a computational cycle with the analog, because of characteristic No. 1. The analog can be set to calculate at any rate desired, but in practice there is an optimum time base associated with any particular problem, and attempts to run the problem much faster or slower will severely degrade the accuracy. The analog, generally speaking, is faster than the digital.

6. Analog outputs are almost always accurate to within 1%, but seldom better than 0.1%.

7. It is very easy, with most problems, to introduce extensive changes in the simulation in a matter of minutes.

Principal Areas of Use

Although the analog computer was designed primarily for the solution of problems in the aircraft field, its area of application has broadened considerably over the years. It has become a major analysis and synthesis tool in the electrical, mechanical, and chemical industries, and as a tool of instruction in our leading universities interested in graduating well-rounded engineers.

While a list of applications of the analog computer would provide a sizeable volume itself, suffice it to say that the analog computer can be applied profitably to almost any problem that can be specified in the form of mathematical equations. The exact form may require a special programming technique, but information is readily available on most of these. In addition, these special cases are welcomed by the applications engineer as a continuation of his education.

The Digital Computer

The digital computer works by a counting technique and obeys logic rules exactly. The solutions are at discrete points dependent on the size of the time increment used. The smaller the mesh size, the more we approach the continuous solution. In contrast to the analog computer, which uses continuous variables in the form of voltages, the digital computer uses discrete variables, and operates with numbers as opposed to voltages. The digital computer is essentially a very fast calculating machine. It is able to perform simple arithmetical operations of addition, subtraction, multiplication, and division. By re-writing the mathematical equations representing a physical system or a physical problem in a special form, it is possible to program a digital computer to solve the same kind of problems that are solved on the analog computer by simulation. The power of the digital computer lies in the fact that theoretically it is capable of solving any problem that can be written in the form of simple arithmetical operations.

There are a number of digital computer characteristics that are of particular interest in connection with hybrid simulation. These are:

1. It will deal only with numbers. Any problem must be reduced to a series of numerical operations before it can be handled

by the computer. This is not to say that every step must actually be written each time. All sorts of aids to compiling programs are available. A program is nothing more than the entire sequence of instructions given to the computer to solve a problem. In actual practice, the machine itself will write most of its own instructions.

2. It will do exactly what it is told. All changes involve writing new instructions. The easier it is to make a change, the more complicated the original instructions have to be to include the option.

3. The results are exactly repeatable, but their accuracy is dependent on the numerical methods used to solve the problem.

4. The computer will perform only one operation at a time. That is, if the instruction reads, "Move number N from location A to location B," the machine will, for a given period of time, be doing nothing but that.

5. The computer works with increments. None of the variables are calculated continuously. Generally speaking, the larger the calculation increment of the digital computer, the faster and the less accurate is the computation. There is absolutely no drift with a digital computer.

6. Compared with an analog, the digital is very much better equipped to make decisions. These can be made on the basis of comparison, time, reaching a point in the program, or almost any other criterion chosen by the programmer.

7. The digital can store very much more information than the analog. It can store tables, functions of several variables, whole programs, and many other things.

Principal Areas of Use

It is almost impossible to list the areas of application of the computer because of the diversity involved. We can say, however, that the digital computer lays sole claim to those problems which store a lot of information, use much logic, or require extreme accuracy. It will calculate trajectories, solve problems in astronomy, simulate mental processes such as learning and memory, analyze games, do translations, help design new computers, and do untold numbers of other tasks. The major effort to discover new computer applications is devoted to the digital area, with the analog a poor second, and the hybrid far behind.

Table 1 is a comparison of the relative performance of analog and digital computers with respect to some operating characteristics which might interest a user. The

Table 1

Characteristic	Analog			Digital		
	Ex	Good	Poor	Ex	Good	Poor
Accuracy (See discussion)						
Speed	X				X	
Absence of long term drift			X	X		
Adaptability to change in program	X					X
Information storage			X	X		
Decision making			X	X		
Ease of programming	X				X	
Ability to solve differential equations	X			X		
Ability to simulate control functions	X					X
Ability to tie in with hardware	X					X
Ability to operate in real time	X				X	
Repeatability		X		X		
Amount of output data available		X		X		
On line outputs	X				X	
Accuracy of outputs			X	X		
Changing outputs	X				X	
Changing inputs	X				X	
Systematic parametric variation		X		X		

information included is intended only as a rough guide to general tendencies, and a stimulus to discussion. It can't possibly be more than that, for several reasons. First, there is a very great variation in characteristics from computer to computer. This is particularly true of digital computers, where the speed, for instance, can vary by a factor of a million, and great disparities exist between the principles of operation. There are significant variations in analog computer characteristics as well. The table was compiled on the basis of experience with large, up-to-date installations, both digital and analog, and undoubtedly reflects this fact. Second, there are no generally accepted standards by which the listed characteristics can be evaluated. Obviously, one per cent accuracy will be excellent for some problems, while one part in a billion will be poor for others. This particular list is highly subjective, and it's very unlikely that anybody else would put his X's in all the same boxes we have. One can always think of a special case which will move any X into any box. Even so, we feel fairly confident that most people who have worked with both types of computer would be in sufficient agreement with this chart to justify its use as an aid to someone considering hybrid simulation for the first time.

The subject of accuracy is so complicated, and dependent on so many factors, that it just didn't seem possible to summarize it by a mark in a box. While this is to some extent true of all the other characteristics listed, we believe considerations of accuracy fall into a special case.

On an analog computer, the result is usually within 0.1% and 1% of the value inherent in the equations. Whether this is excellent or poor depends on the nature of the problem. In many engineering investigations, this is much more precise than the data upon which the problem is based. The use to which the answer will be put also affects the accuracy required. Determination of the region of stability of a control system to within a millionth of the control range would be valueless, as the nature of the input could affect it much more than that. On a digital computer, the ultimate limit of accuracy is the number of bits in a word. This accuracy is seldom attained by the output variables of a problem, due to the approximations involved in almost any mathematical model, the idiosyncrasies of programming, and the practical necessity of taking reasonably large computing steps. The question concerning accuracy is more often, "How much cost and effort is needed to obtain the required accuracy?", than "What accuracy is obtainable?" The answer has to be determined separately for each individual problem.

Combined Analog-Digital Computer Systems

Having duly noted most of the characteristics of analog and digital computers independently, we come to a recurrent question in the industry today, "Is there a need for a new computer system, and if so, what should it be?"

Actually the first part seems to be unanimously answered in the affirmative, while the second leads to all sorts of answers, such as:

1. More digitized analog computers
2. More flexible analog type modules for digital computers
3. Digital differential analyzers
4. Hybrid system coupling the present day analog and digital computers.

This last seems to have the largest support and will receive ours as well. As justification for this support, we can cite our own experience with the hybrid facility at GE MSD; the installation itself, some of the reasons for developing it, and specific examples of its profitable use.

This experience has been with a hybrid installation which combines two large, general purpose computers; one digital, and one analog. It is used primarily for simulation of missile and space vehicle systems and components. Other hybrid facilities are considerably different. Each one is a custom installation, designed for particular needs. Some, as ours, connect a general purpose digital computer to a general purpose analog. Some use a digital differential analyzer rather than an analog. At least one installation uses all three types of computer. The hybrid used at the GE Defense Systems Department in Syracuse, New York, is semi-specialized, as it is used for many types of problems, but only in the field of electronic system simulations. Despite the breadth of the field, there are enough uses and characteristics common to most hybrid installations to make much of what is said here generally applicable, although limited.

There are three obvious reasons for the development of a new type of computation. These are:

1. It will provide a more satisfactory means of performing some calculations than is presently available.
2. It will do computations that would not be practical at all with existing methods.
3. It appears to have potentialities that are worthwhile exploring, if only to keep abreast of competition.

This last reason has, in some cases, undoubtedly been a large factor in the decision to develop a hybrid system, competitive pressures being what they are. However, considerations of this kind do not seem to be pertinent to this particular presentation, and so will not be considered further.

The advantages of a hybrid that we felt to be of most value to the work of the department were in the area of increasing the size and variety of the problems we could solve. The things a hybrid can do to help in that endeavor are:

1. Assign different sections of a problem to each computer. For instance, in simulating a missile, the trajectory calculations can be assigned to the digital, because of the available precision, and the control simulation put on the analog because of its flexibility.
2. Assign different functions to each computer. For instance, all integrations might be assigned to the analog computer, in order to save time and get a continuous output. Or, all function generation might be assigned to the digital computer (where it is known as table look-up).
3. Provide analog plots of digital variables. This is particularly useful in observing the behavior of selected variables while the simulation is in progress. In one case, a stop was put on a 7090 after the first 15 seconds of what would otherwise have been a 10 minute run because it was easy to tell from the behavior of a continuous analog output that a key variable was not behaving quite as desired.
4. Let the digital provide logic for the analog. Things such as switching, scale changing, ending the program, choosing tables to examine, can be readily programmed into the digital and can greatly simplify and possibly even speed up an analog simulation.
5. Allow real hardware to be part of a simulation. Most hardware can readily be connected into the analog, and hybrid operation would allow it to connect to the digital just as easily. Similarly, digital devices can be included in analog operation the same way. Real hardware could also be considered to include people, as part of a control loop.
6. Provide accurate digital printouts of analog variables. Normally, the accuracy with which the analog variables are plotted is less than the accuracy that actually exists in the equipment. Hybrid operation enables selected variables to be converted to digital form and printed out from a digital tape. This gives an accurate printout of an analog variable in any of the forms available for digital variables.

There are many other things that can be accomplished by hybrid computation, but perhaps it would be best at this point to give some examples of problems which were actually done this way.

Examples of Hybrid Simulations

The examples given here are all taken from work done at the hybrid facility at General Electric's Missile and Space Division in Philadelphia. Each illustrates a particular type of use to which such a facility can be put. The explanations will necessarily be brief, but it is hoped they will be sufficient for our illustrative purposes.

Voice Data Reduction: This task consisted of reconstituting, from a digital tape, the original signals from which the tape had been made. The hybrid problem was the last link in a larger study, investigating the conditions under which a spoken message could be converted into digital signals and still contain enough information to be reconverted into intelligible speech. The main study was conducted by another company (Sylvania) which supplied the digital tape to us. The input tape was used as an input to a digital program which scaled the signals and sent them to Hycol (HYbrid COmputing Link) in the proper time sequence. The output of the Hycol was recorded on tape, which when played back produced the original message intelligibly. The program was a straightforward one involving only a one way transfer of information and not making use of the analog computer at all. Actually, part of the program did send the input data back to the digital computer for checking purposes, but the primary information flow was in one

direction; D-A. The program was run in real time. Had it been necessary to transmit information at a greater rate, the data comparison portion of the program might have had to be curtailed or omitted.

Telemetry Data Reduction: This task was very similar to the voice tape program described, in that data were supplied to the program by a digital tape which was used as the primary input of a digital program which, in turn, sent data to the Hycol in real time. In this case, the purpose of the program was to provide a tape which could be used to check the operation of telemetry data reduction equipment. The original tape contained coded signals corresponding to known time variations of all the quantities to be measured. The hybrid program received this data, placed it in the core memory of the digital computer, and performed the operations necessary to dole it out to Hycol in real time, properly coded. The Hycol output was sent through a filter which put the signal in the form required by the telemetry data reduction equipment. This procedure had two advantages over other methods of testing the equipment. One, the input signal was just what it would be expected to receive in actual operation. It even included errors at chosen points. Two, the output could be compared directly with the inputs to the original computer program.

Missile and Telemetry Simulation: The object of this hybrid simulation was, as with the preceding one, to test a telemetry data reduction system. The output of the program consisted of twenty voltages, equal to those that would be produced by the transducers on a missile under the conditions being simulated. In order to produce this output, two separate computer programs were required. The first was a six degree of freedom all-digital simulation which generated the parameters that would be measured by on-board instrumentation in a real flight. The complexity of the program made it necessary for it to generate the data at less than the real time rate. A second program used these data as inputs, scaled them, and converted them to analog signals which were used as real time inputs to an analog simulation. The analog provided the inputs to the telemetry equipment by simulating the behavior of the transducers aboard the vehicle. As in most hybrid simulations at this facility, all the D-A signals were not only sent to the analog, but were reconverted and sent back to the digital for checking purposes. In this simulation, the final output of the data reduction system could be compared directly with the output of the all-digital missile dynamics program.

Missile Control and Trajectory Simulation: This simulation was part of a preliminary design effort whose object was to develop a very accurate short range tactical missile. The part of the design effort involving hybrid simulation was the study of the effect of parametric variations and control system design on the impact accuracy of the missile. This was investigated by a full hybrid simulation, involving the simultaneous operation of both computers. The missile control system was simulated on the analog, and the trajectory, dynamics, and aerodynamics on the digital. The machines did their calculations at the same time, with each one sending and receiving information during an assigned portion of each digital computing cycle. The usual check of the data transfer was made by sending the D-A data back, via A-D, to the digital, and printing out both sets of numbers. This use of the hybrid enables us to have the flexibility and convenience of the analog for investigating the control system, and at the same time use the data storage capacity and precision of the digital for the dynamics and the trajectory calculations. This particular simulation could not be run in real time because of the complexity of the six degree of freedom digital program.

SUMMARY

Some hybrid computer applications with which we have had personal experience have been described. Other hybrid installations have reported work on chemical processes, flight analysis and control, air traffic control, reaction jet control, solution of partial differential equations, and function generation, among others.

The advantages of hybrid simulation are based on the disparate nature of analog and digital computers. The characteristics of the analog that are most useful in hybrid operation are:

1. The variables are in the form of easily available d.c. voltages, continuously varying in time.
2. The program can be changed very rapidly and easily.

3. The output is recorded as the problem is run.
4. Transfer functions and feedbacks can be represented very easily.
5. It is very easy for most engineers to get the physical "feel" of a problem on the analog.
6. The analog can usually operate very fast, and the speed is independent of the size of the problem.

The aspects of the digital which are very useful in hybrid simulation are:

1. Precision
2. The ability to store information
3. The ability to make logical decisions
4. Stability and repeatability
5. Printing out large amounts of data in tabular form.

It is obvious that any problem which would benefit by the availability of computer characteristics from both lists could probably benefit from being set up as hybrid problem.

As far as the future of hybrid operation is concerned, we feel that the biggest contribution will be made by the digital computer manufacturers, although this has not been the case so far. Eventually, some digital computers will come with optional packages which will enable at least these three instructions to be added:

1. Stop until time t on clock, or priority interrupt at time t, then proceed.
2. Transfer to analog, from core location X, to output channel Y, scale value Z.
3. Transfer from analog, to core location X', from input channel Y', scale value Z'.

The package will contain an accurate source of time signals, n output jacks, and m input jacks. The jacks could be connected directly to the analog computer.

The first instruction will either stop the computer until a specified time signal is received, or send it to a chosen instruction at that time.

The second instruction will take the (presumably) floating point number in core location X, convert it to fixed point, and convert it to a voltage which will be established at output jack Y. The value of the voltage will depend on the scale value. For instance, if a quantity 2.3×10^5 is stored in core location X, and the scale value Z (that is, the value represented by full scale voltage, which is 100 volts on most analogs) is 5×10^5 per 100 volts, then 46 volts would appear on output jack Y. As many as n transfer out instructions could be given during one computer cycle.

The third instruction would transfer analog data into the digital computer in a manner similar to the way the second instruction transfers out. As many as m transfer in instructions would be possible per computing cycle.

With these three simple instructions, a great variety of hybrid simulations could be readily performed. It would not be necessary to limit the information transferred in this manner to problem variables, as a voltage appearing on a trunk line of an analog can be used for any purpose desired, such as starting, changing modes of operation, and operating relays. It would be expected, however, that the transfer of problem variables would utilize more of the transfer package than any other function, for most problems. The simplicity of the programming procedures would probably be a greater factor than any other one thing in encouraging people to avail themselves of the advantages of both types of computer in a single problem, instead of limiting themselves to either type alone.

We feel that a listing of installations provided as part of the first lecture at a recent hybrid symposium at Electronic Associates Computation Center in Princeton, New Jersey is well worth including as an indication of the variety of systems in use, and as a source of further information.

Table 2

Location	Digital Computer	Linkage
IBM - Yorktown Heights	IBM 704	EAI
University of Minnesota	Univac 1103	EAI
Ramo Wooldridge originally IBM for Ft. Belvoir - Fieldata	PB 250	EAI
Douglas Aircraft	G - 15	EAI
White Sands Missile Range	IBM 704 (7090 later)	EAI
General Motors Goleta, California	CDC 160 A	EAI
Minneapolis Honeywell Pottstown, Pennsylvania	MH 290	EAI
Detroit Tank Arsenal	Datatron 205	EPSCO A-D EAI D to A only
Western Electric	Univac thru Bell Tel. I/O	EAI D to A
Westinghouse Electric	Special West. Computer	EAI D to A Plotters
Convair Astronautic Vacuum Tube	IBM 704	EPSCO
Space Technology Lab Vacuum Tube	Univac 1103	EPSCO
Canadian National Research Council	G - 15	EPSCO
Grumman Aircraft	IBM 704	ADAGE
General Electric Philadelphia - MSD	IBM 7090	Homemade
NBC	SEAC	Homemade
North American Aircraft		PB
Bell Labs		
IBM - Owego	IBM 7090	PB
General Electric Syracuse	PB 250	GE
Applied Science Corporation of Princeton (ASCOP)		
PMR, Pt. Mugu	IBM 7090	PB
Aeronautic Research Associates of Princeton (ARAP)	LGP 30	
Westinghouse Research		
M.I.T.		
L.T.V. Chance Vought	PB 250	EAI
Martin Orlando	IBM 7090	EAI
McDonnell Aircraft	IBM 7090	McDonnell

DATA HANDLING AT AN AMR TRACKING STATION

K. M. Hoglund, P. L. Phipps, E. J. Block, R. A. Schnaith, J. A. Young
UNIVAC
Division of Sperry Rand Corporation
Military Operation
Univac Park, St. Paul, Minn.

SUMMARY

The downrange tracking station at Ascension Island (AMR Station 12) includes, as part of its instrumentation, a real-time data-handling system. Using digital techniques primarily, this system collects, records, and transmits data describing the trajectory of selected ballistic objects within range of the FPS-16 radar on Ascension Island. Among its functions, the data-handling system performs the following tasks:

- Aids the FPS-16 radar in initial acquisition of the ballistic vehicle, and in reacquisition following any subsequent loss of track.
- Records all raw data collected by the FPS-16.
- Maintains communication with Cape Canaveral (and with other tracking stations when required) prior to, during, and after a flight.
- Provides stations further downrange with real-time acquisition data in the form of updated orbital parameters.
- Provides trajectory data to plotting boards and other instrumentation at Ascension Island.

The UNIVAC 1206 Military Real-Time Computer is used as the central data processor. Associated with the computer are various data buffers, A/D and D/A converters, data recorders, and communication equipment.

Information concerning a planned flight (nominal trajectory parameters and other data) is transmitted from the Cape to Ascension Island prior to lift-off. Shortly after burn-out, actual, measured, parameters are similarly transmitted, along with timing data.

Within the computer, the dynamics of ballistic flight are simulated in real-time in order to predict the coordinates of the vehicle when it comes within range of the Ascension radar. The radar uses these data to acquire the target. After lock-on, the condition is reversed, with the radar supplying track data to the computer, and via the data-handling system to the entire range network. The mathematical treatment of the problem involves editing and smoothing the data (for which a "combination" technique is used), integrating the data to obtain a predicted position of the vehicle, and then extrapolating beyond the integration to obtain a still further look-ahead. The interval of integration and the manner of implementing it have been tailored to fit the particular conditions encountered at Station 12.

The computer program may be considered as comprising two major routines: the initialization routine and the acquisition and track routine. The former routine is executed prior to lift-off and sets the initial conditions prevailing for a specific mission. The latter performs the real-time acquisition and tracking functions during the flight.

INTRODUCTION

The purpose of this paper is to report on the use of a high-speed, general-purpose

computer in the data-handling system of a downrange radar tracking station—Ascension Island. The real-time data handling system for Ascension Island consists of equipment, primarily digital, to collect, record, and transmit data describing the trajectory of selected ballistic objects within range of the FPS-16 radar. The primary functions of the system are:

(1) To aid the FPS-16 radar in initial acquisition, and in reacquisition following any loss of track;
(2) To record all raw data collected by the FPS-16 radar;
(3) To enable transmission of FPS-16 data from Ascension Island to Cape Canaveral in real-time and/or post-flight mode;
(4) To provide stations farther downrange with acquisition data in the form of updated orbital parameters;
(5) To provide real-time trajectory data to plotting boards and other local instrumentation.

The equipment consists of three functional groups: namely, the digital computer and associated peripherals, buffers and converters, and the data recovery system. (See Figure 1.) The computer group includes the UNIVAC 1206 computer, a magnetic tape handler and control unit, and an AN/FGC-20 teletypewriter unit. The buffers and converters comprise the necessary analog-to-digital and digital-to-analog converters, impedance matchers, and format control buffers. The data recovery group consists of two subsystems: the first is located on the Island and is made up of a raw data recorder and an output buffer for high-speed RF transmission to the Cape. The second subsystem is at the Cape, and consists of a high-speed RF receiver, followed by an input buffer and

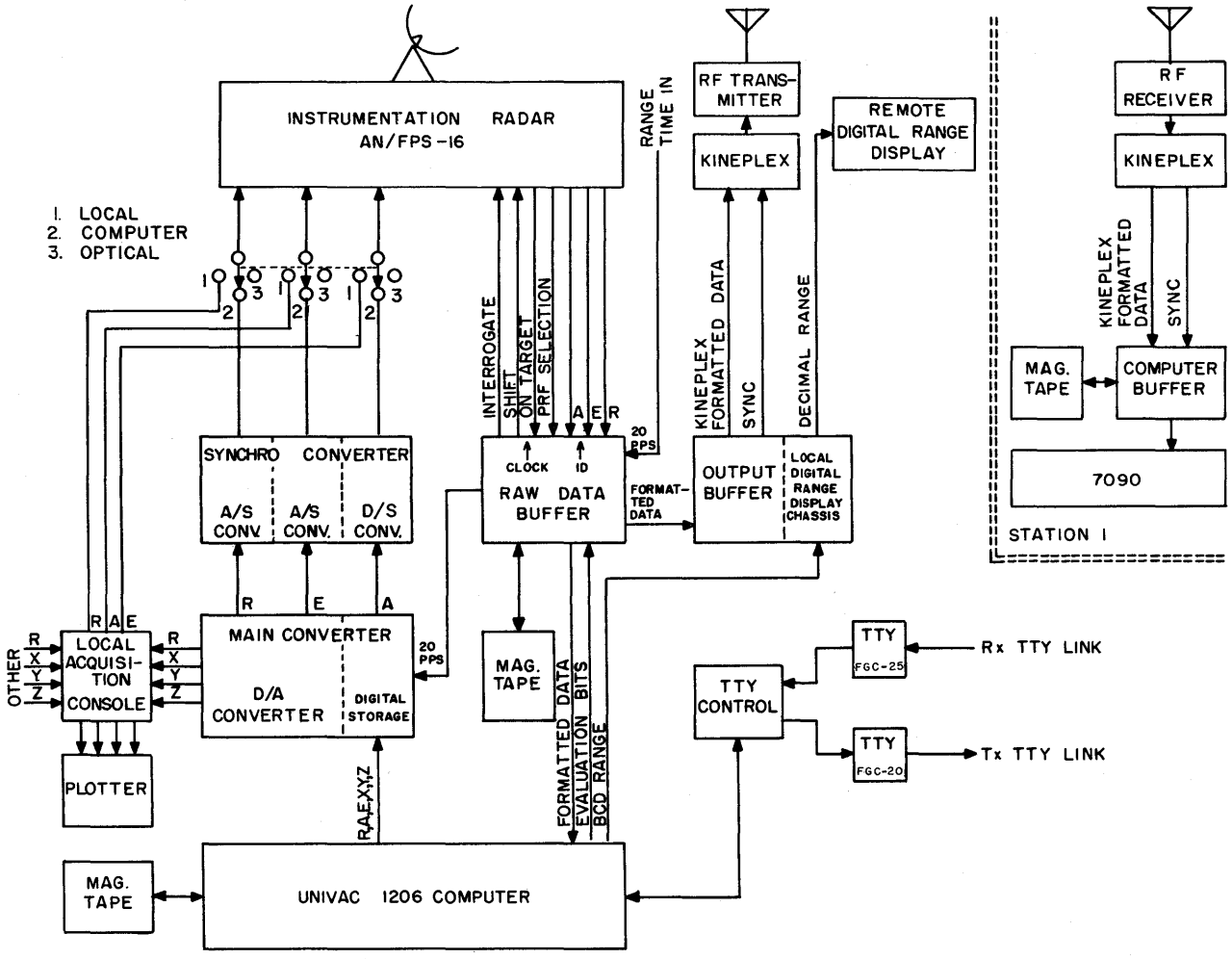


Figure 1. Ascension Island Real-Time Data Handling System.

a magnetic tape unit which records in the same format used by the raw data recorder on the Island.

The Tracking Problem

In order to provide, in real-time, acquisition data to a local tracking device and to downrange stations, and to provide current position data to a plotter, some form of prediction is necessary. An accurate prediction scheme, based on well known dynamic principles governing ballistic motion in the vicinity of the earth, can be implemented in real-time by a fast digital computer with a memory of moderate size.

Within the computer, the dynamics of ballistic flight can be simulated in real-time and used to predict the position and velocity of the vehicle. This simulation must take into account the non-sphericity of the earth and the atmospheric drag. The prediction information must then be converted to radar coordinates and fed to the radar continually. If the radar has not yet acquired, it will accept the information. After lock-on, the radar will ignore the prediction data, but will quickly switch back to the acquisition mode if track should be lost. In the following discussion, the physics and instrumentation of the problem are discussed; system, mathematical and programming considerations subsequently follow; and finally, the results are discussed and certain extrapolations made.

Physics of the Problem

The specification of a method of orbit determination (or more properly in this case, orbit improvement) embraces a number of items, all of which are interrelated. These are:

- The parameters to be used to describe the orbit
- The coordinate system(s) to be used
- The physical features to be included in (and excluded from) the simulation model
- Mathematical representation of the model
- Mathematical methods to be utilized in manipulating the model
- The form and rate of available input data
- The form and rate of the required output data
- The time available for the computations

Since Ascension Island is one of several stations widely spaced over the earth, a geocentric inertial coordinate system is the most convenient choice for purposes of communication among them. This coordinate system is also a convenient one in which to express the dynamic parameters of orbit improvement and prediction. Furthermore, the position coordinates and their time derivatives, together with the time at which they are valid, constitute a convenient set of orbital parameters since they uniquely specify the orbit.

The dynamic principles governing the motion of the object to be tracked are expressed in a set of simultaneous differential equations. Newton's laws of motion applied to the two-body problem provide principal terms in these equations. Smaller terms arise out of the effect of the asphericity of the earth's gravitational potential. These terms depend only on the position coordinates of the bodies involved. In addition, when the altitude of the orbiting vehicle is low enough, the retarding effect of the atmosphere gives rise to drag terms which are velocity-dependent. Drag terms depend upon the density of the atmosphere, the shape of the vehicle, the orientation of the vehicle with respect to its velocity vector and the orientation of the velocity vector with respect to the computational coordinate system. Logic must therefore be provided to insert the drag terms whenever the altitude is below a specified limit. An assumed-atmosphere model and the drag characteristics of the vehicle must also be included in the program.

Prediction essentially means solving the set of differential equations for a certain epoch (time), given a sufficient set of initial conditions. The presence of terms other than the principal terms in the differential equations precludes a solution in closed form. Numerical integration methods must be used to approximate the solution. These methods advance the solution in stepwise fashion, predicting ahead by a 2-second time increment. The results obtained from one step in the process become the inputs to the next step.

Instrumentation

The choice of particular devices wherewith to implement the system must take into account many ancillary factors, most of which

can be satisfactorily handled either by the computer program itself or by hardware buffers. As in many other systems utilizing a digital computer, there exists the problem of converting from analog to digital form and vice versa, as well as the problem of matching the data rates demanded by external equipments.

Some peculiarities of the external equipment are accommodated by the computer program. For example, after the predicted position has been found in the computational coordinate system, it must be represented to the external buffering devices both in the format and coordinates demanded by the devices and with the required scaling. The radar and the X-Y plotter both work in topocentric ("topos" = place) coordinate systems, the radar in spherical polar coordinates, and the plotter in Cartesian. In addition, data for the radar must be modified by inclusion of the refraction errors. It has not proven necessary, however, to compensate for possible errors in the dynamic response of the radar servo systems (although this may be done and is done on the other systems).

Similarly, at the input end, a major task of the computer is to interpret the input data properly. In particular, a pulse radar, if it has the capability of tracking beyond the range which corresponds to the time between pulses, has an ambiguity in its range data because the equipment has no way of associating a return with the initial pulse which caused the return. Consequently, range data both to and from the radar are given modulo R_0 where $2R_0$ is the speed of light divided by the pulse repetition frequency (prf) of the radar. In the Ascension Island system there is the additional complication that the prf can be selected at will by the radar operator from among a number of alternatives. The computer program must contain logic to add R_0 the correct number of times in order to resolve the ambiguity.

Communications

The system must communicate with the operator and with the outside world at both input and output ends. As mentioned previously, the system must receive nominal orbital parameters before launch. These data are used as back-up in the event of a communications failure later on. After burn-out, the system receives orbital parameters based on measured up-range track data. These are de-formatted and checked for reasonableness in the computer, as a guard against error in the communications system. Initial acquisition data for the radar are based on them. If they are garbled, or not received at all, the nominal values are used.

As output, the best set of orbit parameters is transmitted downrange. During this phase of the program, the computer formats the data for transmission.

Because of the relatively slow rate of data transmission, input-output functions consume little computer time. The narrow bandwidth of the communications system makes it desirable that only one "point" be transmitted from up range and lends importance to the computer's function of stepwise prediction from a distant point to yield many acquisition points for the radar.

Man-machine communication is achieved by means of a radio Teletype unit attached to the computer or to the radio link, the selection being made manually. This device permits entering the required data into the computer and also allows the operator to monitor, to a limited extent, the operation of the computer. However, the switching from pre-acquisition mode to acquisition mode to tracking mode is fully automatic.

As indicated above, communication between the computer and the radar is via input and output buffers. The buffers control the data rates also, sending an "interrupt" to the computer to signal the presence of input data or a demand for output data. These demands are quickly handled by the program, which accepts and stores the input data and provides output data from previous computations. Between interrupts, the mathematical calculations are carried forward in preparation for future demands. The mathematical formulations and methods and their coding are such that the computation always stays ahead of these demands.

System Operation

The interface between the data-handling system and the other parts of the tracking instrumentation includes points of contact at the radar acquisition inputs, the radar outputs, the local acquisition console (for displays), the RF transmitter at Ascension, the RF receiver at the Cape, and, finally, the

7090 computer at the Cape. Figure 1 lists the specific kinds of data exchanged across this interface.

In order to explain the operation of the system, a hypothetical mission is described below in terms of the function of each equipment shown in the system block diagram.

Prior to a mission, usually at least a day before, the nominal or planned trajectory parameters, drag tables, and refraction data are transmitted to Ascension Island and introduced into the computer via radio Teletype link for use during the actual mission. These parameters are stored on magnetic tape along with a copy of the program to be used during the mission. Many subsystem and system confidence checks are also run at this time.

During countdown, the complete system is exercised by feeding the raw radar data stored on the raw data tape unit into the raw data buffer and thence to the rest of the system. The Kineplex transmission link to the Cape may be tested in this manner. Radar calibration values (zero sets) are also given to the computer at this time.

Shortly after burn-out of the last stage of the missile, tracking instrumentation at the Cape determines a final set of orbital parameters ($X, Y, Z, \dot{X}, \dot{Y}, \dot{Z}$), all valid at some time, $\underline{t}$. This set—together with lift-off time and time $\underline{t}$—is transmitted to Ascension Island, entered into the computer, tested for reasonableness, and copied onto the back-up program magnetic tape. The computer program substitutes the second set for the first set as soon as the second set is received. In the event of communication failure, however, the program will continue to operate on the first set.

The parameter set is in an inertial, geocentric system. Time $\underline{t}$ is an instant in the flight occurring after burn-out. Based on this information, the UNIVAC 1206 integrates the trajectory by stepwise integration to that portion of the flight regime within range of the Ascension Island radar. The computer also establishes a time, t_o, at which acquisition will occur and sends the associated look angles and range to the radar.

When t_o occurs, acquisition computations are updated. Predicted values of slant range (R), azimuth (A), elevation (E), and local x, y, and z are transmitted, at the rate of 20 samples per second to the output converters, which change the digital values to analog form. Local x, y, and z are used by the local acquisition bus and the plotters. The analog (polar) R, A, and E values are converted to synchro signals and fed to the radar servo system, where they maintain the radar dish in position for acquisition. At the same time, the computer generates a recommended mode of radar scan (circular or raster) and displays it to the radar operator. This recommendation is based on the probability of target presence as determined in part by the reasonableness of the measured set of orbital parameters.

R, A, and E values appearing at the output of the radar are sampled by the raw buffer at the rate of 20 samples per second. These data are formatted along with range time, ID (identification) bits, and parity checks, and recorded on the raw data magnetic tape. Figure 2 illustrates the recording format. Concurrently, the data are given to the computer in a similar format.

When the radar succeeds in locking on the target, it generates an on-track signal. This signal is detected by the computer, and if the accompanying position data pass a reasonableness test, the computer program switches from acquisition mode to track mode. If RF propagation conditions permit, these same data from the raw data buffer are converted to three 40-bit words by the output buffer and transmitted via Kineplex to the Cape at 80 words per second. There, a Kineplex receiver terminal reconverts the three 40-bit words back to the original 28-character format. The data are then recorded on a second raw data magnetic tape, making the information available to the Impact Predictor at the Cape.

If while tracking the radar should lose the target—during re-entry for example—the computer will respond to the absence of the on-track signal and immediately switch back to acquisition mode. Acquisition data at such times are based on past tracking data. Further, if the missile is low enough in altitude to be affected by them, drag forces must necessarily be taken into account in calculating the acquisition data.

For missions where the vehicle will impact within the radar's line of sight, the computer automatically prints out a predicted impact point. In the case of flights of longer range, the computer calculates acquisition data, which are then transmitted downrange for use by other tracking stations.

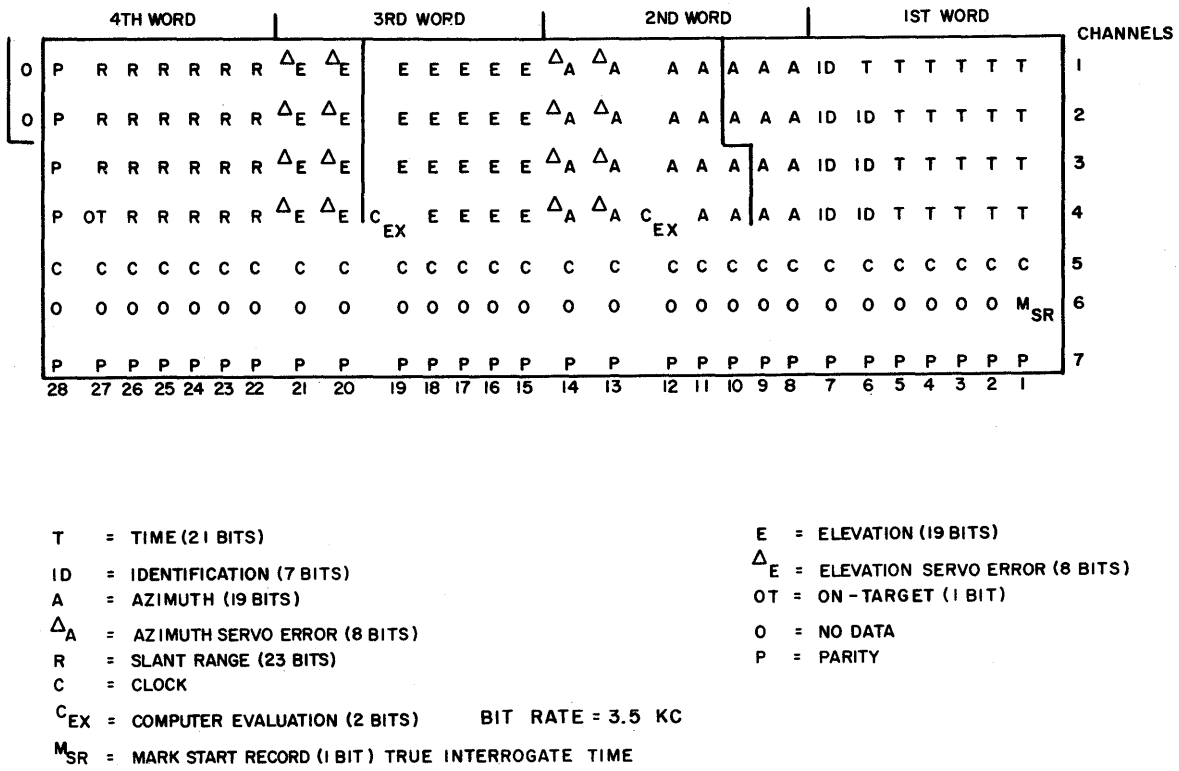


Figure 2. Format of Recorded Data.

Univac 1206 Computer

The computer used in this installation is the UNIVAC 1206 Military Real-Time Computer. This unit is a general-purpose, high-speed, digital machine of modular construction and ruggedized to meet military specifications. In the configuration employed at Ascension Island, the computer has the following characteristics:

- Internal core memory of 16,384 30-bit words (expandable to 32,768),
- 8-microsecond cycle time
- Single-address, indexed instructions
- Wired-in bootstrap memory for initial loading and recovery purposes
- Fixed-point, one's complement, subtractive arithmetic
- Seven input and three output channels
- Internal 7-day clock, with granularity of 2^{-10} second
- Over-all size of $30 \times 37 \times 65$ inches

Program Philosophy

The main concern of the computer program is the meshing of each component of the tracking system into a single, operable, real-time unit. The specific objects which must be mathematically related are the earth, the vehicle being tracked, and the radar. Past experience by many groups has shown that computer simulation of a vehicle in ballistic flight can be made very precise since gravitational, atmospheric, and drag parameters can be determined quite accurately. However, missile simulation is notoriously time-consuming; therefore, much effort was spent in tailoring the simulation exactly to the particular real-time problem at hand. Numerical integration of the equations of motion of a vehicle in ballistic flight about an oblate spheroid, with drag terms when applicable, was chosen as the most desirable method of simulation.

With the use of numerical integration, the program can be made very symmetrical—that is, numerical integration can continue at all times, with the only changing factor being the input or initial conditions. If the radar is delivering high-quality track data, these are used as the input to the numerical integration; however, if high-quality track data are unavailable, the integration will continue using the results of the previous

integration step as shown in Figure 3. Radar data must pass a reasonableness test, be on-track, and must have variance within a specified range to be considered of high quality.

A 2-second interval of integration was chosen after a study of the accuracy of numerical integration versus the integration interval. Two seconds was found to be a reasonable interval over all portions of the flight; however, the integration interval could have been much greater in the exo-atmosphere phase, and perhaps smaller during re-entry.

The time sequence of the calculations is: collect and smooth radar data; combine with past data, and then use the results as input to the numerical integration. Since all this consumes time, extrapolation is used to yield data three seconds "ahead of the clock." If the radar should lose track, the extrapolated data are available for predicting future look angles. If the radar is not transmitting high-quality track data, the combination sequence is bypassed, but the integration and extrapolation continue.

Mathematical Model

The importance of having a proper mathematical description of the earth and vehicle, particularly during re-entry, is apparent. Although the vehicle has traversed thousands of miles since it was last observed, its position must be pinpointed within the beamwidth of the radar. Therefore, one system design goal was that it should never be necessary to scan the radar to acquire the vehicle, although scanning is utilized to increase the probability of acquisition.

Studies revealed that, for trajectories of intercontinental range, negligible error was introduced by neglecting the fourth harmonic term in the earth's gravitational equations [1]. The attraction of the sun and moon was also neglected.

Atmospheric drag terms enter the computations when the vehicle descends to 300,000 feet. The values of these terms are derived from the drag characteristics of the particular nose-cone and the 1959 ARDC model atmosphere [2]. Radar range and elevation are corrected for atmospheric refraction based on the surface index of refraction computed prior to the flight [3].

An inertial reference system was chosen for all major computations since in this system the differential equations have their simplest form. The familiar equation of motion in vector form becomes:

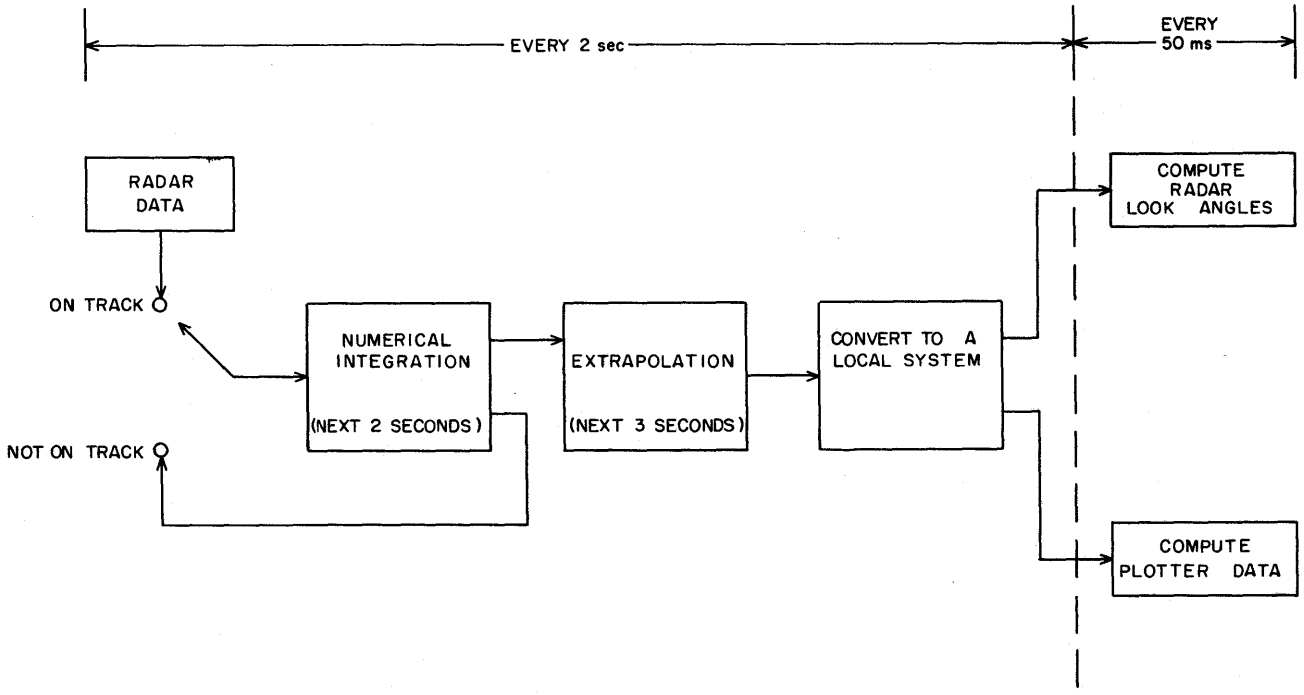


Figure 3. General Outline of Prediction Technique.

$$\ddot{\bar{R}} = \bar{G} - \bar{F}_D/m$$

where $\bar{R}$ is the position vector for the center of the earth

$\bar{G}$ is the earth gravitational force vector as "seen" by the vehicle at position $\bar{R}$. In this application $\bar{G}$ includes the second harmonic term.

$\bar{F}_D$ is the drag vector and has the form $-1/2 C_D \rho A Va \dot{\bar{R}}$

C_D is the drag coefficient, ρ is air density, A the cross-sectional area, Va the absolute value of vehicle velocity relative to the atmosphere, and $\dot{\bar{R}}$ the inertial velocity vector.

m is the mass of the vehicle.

Note in this equation that no lift or thrust terms are provided. Also, the vehicle is assumed to be both a point mass and a constant mass.

A fourth-order Runge-Kutta-Gill integration method was selected because of its self-starting ability and ability to change integration interval in a simple manner [4]. Since the method is basically a truncated Taylor-series expansion, the truncation error in the fourth-order integration is proportional to the fifth derivative of the integrated quantity. The fourth-order scheme makes four complete passes through the differential equations.

Processing of Raw Data

Raw radar data are collected for two seconds and fitted to a second-order polynomial by means of least-squares. This filtering and smoothing technique delivers midpoint values of radar range, azimuth, and elevation and their velocities, all valid at the filter midpoint.

Midpoint range and elevation are corrected for atmospheric refraction (the refraction error is considered constant over the 2-second period) and used in a "combination" technique. Before combination, two additional computations must be performed.

a. The variance (noise) of the raw range, azimuth, and elevation is estimated using second-order variate differences [5]. The velocity variance is considered proportional to position variance.

b. During the previous 2-second computation, the predicted position and velocity at the time t were computed by numerical integration and extrapolation. The variance of each of these quantities is also computed.

Two estimates of each parameter and the variance of each parameter are now available. If X_p, X_f are the predicted and filter midpoints respectively, and σ_p^2, σ_f^2 the variance of the predicted and filter midpoint, then the combined value and its noise are:

$$C = \frac{\left(1/\sigma_F^2\right) X_F + \left(1/K\sigma_P^2\right) X_P}{\left(1/\sigma_F^2\right) + \left(1/K\sigma_P^2\right)}$$

$$\sigma_C = \frac{1}{\left(1/\sigma_F^2\right) + \left(1/K\sigma_P^2\right)}$$

K multiplies the predicted variance, and was chosen as 1.35. Increasing the predicted variance tends to put more weight on the data from the filter.

The combined value is used as input to the numerical integration after conversion to inertial space. The results of the integration yield a new predicted parameter to be used on the following cycle; that is, after integration and coordinate conversion, C becomes the new X_p. The integration coordinate conversion also yields the σ_p that will accompany X_p.

It is thus evident that a particular X_F will indirectly enter into the combination scheme over and over again, with less weight each combination cycle because the data are getting older. Thus, the K constant controls the memory of the combination scheme; K = 1.35 weights the oldest data (18 filters old) at 1% of the entire span.

Program Organization

The computer program is organized into two parts: the initialization routine and the acquisition and tracking routine.

The initialization routine:

1. Accepts preliminary data via Teletype link;
2. Writes recovery tape;
3. Accepts in-flight message or lift-off time;
4. Develops the acquisition point.

The acquisition and tracking routine:

1. Accepts tracking information from the radar through the raw data buffer;
2. Provides designation data for the radar through the output converter;

3. Provides real-time trajectory data to the plotting boards and other instrumentation through the output converter;
4. Provides updated orbital parameters for downrange stations via the radio Teletype link;
5. Provides visual display of recommended scan mode and target range.

Each routine has its own executive routine which controls the subroutines necessary to perform its functions. As in all real-time systems, one basic problem is the sharing of time among different functions. The purpose of the executive routine is to allocate time to the various subroutines in such a manner that all functions may be performed in their proper sequences and within their time restrictions.

The basic time requirements considered in the program organization were:

1. To provide radar designation, plotter data, and visual display every 50 ms
2. To accept radar data every 50 ms
3. To maintain a 2-second prediction cycle
4. To be capable of transmitting up-dated orbital parameters downrange every 30 seconds

The computer is synchronized to range time through an external interrupt generated by the raw data buffer and output converter every 50 ms. Immediately after the program has been loaded into the computer from magnetic tape, the initialization executive routine assumes control and the following functions are performed:

1. Accept Preliminary Data

The following Teletype inputs are called for and accepted by local insertion:

a. Nominal Trajectory Message containing an inertial set at some point soon after burn-out for the theoretical trajectory; the delta time since lift-off at which the point is valid; and the physical characteristics of the nose cone.
b. Calibration Data consisting of corrections to be applied to radar data, as determined by pre-mission calibration.
c. Refraction Data consisting of temperature, barometric pressure, aqueous vapor pressure, from which the index of refraction is computed.
d. Plotting Board Scaling Data consisting of the origin and maximum values for the plotting board.

2. Write Recovery Tape

The entire program, including above inputs, is now written on magnetic tape and check-read for recovery purposes.

3. Accept In-Flight Message or Lift-Off Time

The program accepts either an in-flight message for some point soon after burn-out (which can be introduced directly through the Teletype linkage), or a locally inserted lift-off time and delta time will give a nominal starting point for the inertial set. A "ball park" check is made on the in-flight message, and if it fails, the operator has the option of continuing or inserting lift-off time to use the theoretical inertial set for a starting point. It should be noted that any recovery procedure will cause the program to jump to the point where the in-flight message or lift-off time can be introduced into the computer. Therefore, the initial steps through writing the recovery tape can be done many hours before the shot if it is advantageous to do so.

4. Develop Acquisition Point

The inertial set of the starting point is integrated ahead by 30-second steps. After each integration step, the range, range rate, and elevation of the point with respect to the radar are calculated as is the altitude of the point with respect to the subsatellite point. Various checks are made to determine whether the point is within acquisition range of the radar, or if the target will splash before it comes into acquisition range, or will not come into acquisition range for other reasons. If the program determines no acquisition is possible, the operator is notified and the program terminates. When the point is determined to be in the acquisition range of the radar, the input to the last 30-second integration step is used to integrate the point into the acquisition range of the radar by 2-second steps. The time of the last inertial set is then checked to determine if it is at least 4 seconds ahead of t_o. If the time of this point should fail this test, the integration is continued in 30-second steps if the point is outside the atmosphere (300,000 feet), or 5-second steps if it is in the atmosphere, until either the condition is satisfied or the program terminates due to a splash or

out-of-range condition. The output of the last integration becomes the point of acquisition, and its associated time to is the acquisition time. Through integration, extrapolations coordinate conversion, and interpolation, all initial conditions required for the transition to the acquisition and tracking routine are developed. The designation data to the radar, plotter data, and visual displays for the acquisition point are then continuously buffered out to the output converter while t_0 is being checked against acquisition time, T. When t_0 and acquisition time are equal, control is transferred to the acquisition and tracking executive routine.

In analyzing the mathematical functions of acquisition and tracking, it becomes apparent that three basic repeating cycles of performance are involved:

1. A data cycle that occurs every 50 ms
2. A prediction cycle that occurs every 2 seconds
3. A transmission cycle that occurs every 30 seconds

The acquisition and tracking routine was therefore organized into three major tasks or routines to correspond to these cycles and perform the functions required:

1. 50-ms task
2. 2-second task
3. 30-second task

The three major tasks are controlled by the acquisition and tracking executive routine, which allocates time to the various subroutines in the proper order and at the proper time to permit the performance of these tasks without interference with each other. The Task Flow Chart in Figure 4 indicates the logic connecting the three tasks.

The data flow revolves around the mechanics of the filter and the combination of smoothed and predicted data. The filter takes every other piece of radar data over a 2-second span. The last point into one filter becomes the first point of the next filter. The output of the filter is at the midpoint of the span. The integration is arranged so that each step is time-correlated to the output of the associated filter. The predicted values for radar designation and plotting are interpolated between 3-second extrapolations of the outputs of the integration steps. At the time of acquisition (T), after control has been turned over to the acquisition and tracking executive routine, the following quantities are provided by the initialization routine:

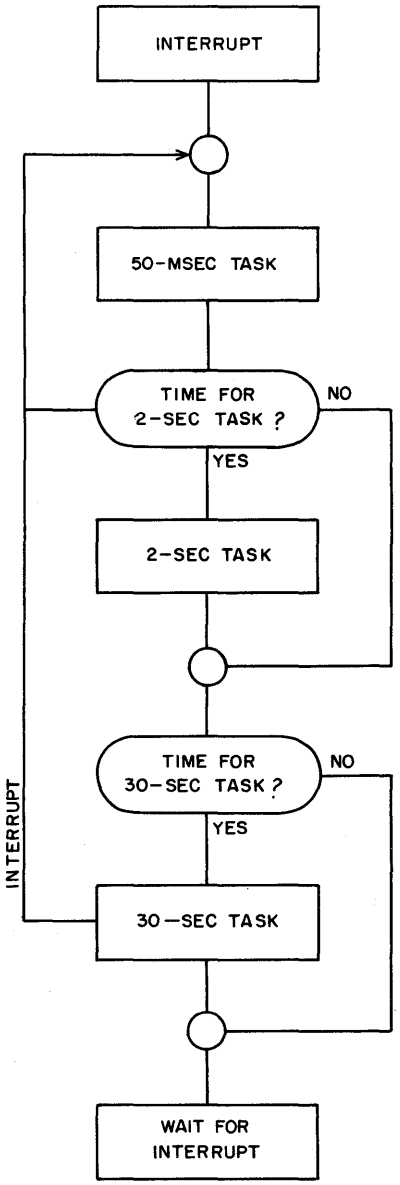


Figure 4. Task Organization of Acquisition and Tracking Routing.

1. A starting point for the integration valid at T-1 second.
2. Extrapolated points for both T and T+2 seconds.
3. Interpolated designation and plotting data at 50-ms intervals between T and T+2 seconds.

During the first 2 seconds in the acquisition and tracking routine, the following events occur:

1. Every other sample of radar data is applied to the filter;

2. The interpolated designation and plotting values between T and T+2 are outputted to the output converter at the time they are valid.

and concurrently:

3. An integration step from T-1 second to T+1 second is performed.
4. The output of the integration is extrapolated from T+1 second to T+4 seconds.
5. The extrapolated values for T+4 seconds and T+2 seconds are converted and interpolated for designation purposes.

At T+2 seconds, a new filter is started. The output of the previous filter is valid at T+1 second and corresponds to the output of the integration. The starting point of the new integration cycle is determined and the cycle is determined and the cycle repeats itself. At T+2 seconds, and every 30 seconds thereafter, the output of the integration is checked for transmittability, and if so, the downrange transmission is initiated.

Taking advantage of the range-time interrupt, the acquisition and tracking routine is able to use this feature as its sole source of control and timing. The internal computer clock is used to check for extraneous or missing interrupts. Each interrupt signals the start of a block of radar data through the raw buffer and automatically transfers control to the executive routine.

The various functions of the routine are ordered according to the following priority:

1. The interrupt routine (executive)
2. The 50-ms task
3. The 2-second task
4. The 30-second task
5. Wait (a passive subroutine executed after all other tasks are completed and the computer is awaiting the next interrupt).

The 50-ms task is initiated every time an interrupt is received. The 2-second task is initiated after every fortieth 50-ms task. The 30-second task is initiated after every fifteenth 2-second task. As noted below, the three tasks are performed in far less time than their nominal cycle times. Thus, the 2-second task can be performed in the time remaining from the 50-ms task.

Similarly, the 30-second task can be executed in the time remaining after the 50-ms and 2-second tasks have been completed.

The 50-ms task requires a maximum of 27.4 ms and an average of 17 ms. The 2-second task requires a maximum of 300 ms, and the 30-second task requires a maximum of 86 ms. Thus, an effective maximum of 35 ms per 50-ms interval is used. In other words, the computer is said to be loaded by no more than 70 per cent. The combination, integration, and extrapolation computations previously described require a maximum time of 155 ms exo-atmosphere and 186 ms endo-atmosphere. The total operational program is 12,000 instructions in length.

CONCLUSIONS

To date, three such data handling systems have been installed: at Station 12, as already described; at Station 13, near Johannesburg, South Africa; and at Station 9.1 on the Island of Antigua.

At this writing (Nov. 1962), the station at Ascension Island has successfully performed computer-aided acquisition, tracking, and reacquisition before and after reentry. Station 13, more recently installed acquired the Aurora 7 capsule on its last pass. Station 9.1 has also been tested successfully.

Many other functions and greater generality could be incorporated in the data-handling system. In the future, additional sophistication will not be optional but will be required because of the ever-increasing demands of both our space and military missions.

The most recent advance in range-tracking coverage results from the placing of tracking stations aboard ships. Here, in the calculation of acquisition data, the data-handling system must consider all ship motions which can affect the radar data. These include roll, pitch, navigation maneuvers, and even flexure of the ship which occurs between the inertial platform of the navigation equipment and the radar pedestal.

An obvious extension of downrange tracking stations is the ability to detect and measure the beginning or termination of thrust during orbit transfers and maneuvers. The implementation of this ability must await further study of signal-to-noise ratios and the sampled-data characteristics of the instrumentation. To detect low thrusts by means of tracking data alone requires very sensitive digital filters, but these take time to settle before giving an indication that thrust is detected (or terminated). This extra time (5 to 10 seconds) may easily be comparable

to the thrust duration itself; hence one is fortunate to detect it, let alone measure it.

A possible solution is to receive telemetry measurements of acceleration from the Cape, but this entails decommutating and sorting a great mass of data and possibly converting analog-to-digital words suitable for computer input. These communication and data extraction problems are not known to have been solved on a real-time basis as yet.

Another attribute which will be realized some day is the assignment of some portion of the command and control function to the remote site, e.g., orbit transfer. With an "intelligent" program, certain parameters of the orbit can be measured and transmitted in error-correcting code back to the Cape via radio Teletype link. On the basis of these, the go-ahead can be teletyped back to make final measurements, check for additional deviations, check the control link, and finally to issue the command itself. The expected effects of the command can then be monitored, and predictions of the resulting orbit quickly made and relayed to other stations as well as the central control at the Cape. These are but a few of the enhancements that someday will be mandatory at such tracking stations.

ACKNOWLEDGMENTS

A system of this complexity is necessarily the creative work of a number of individuals whose contributions, though unacknowledged, are no less important to the success of this project. This paper is based on work performed for the Air Force Missile Test Center, Patrick AFB, Florida, under contract number AF08(606)-4455. Acknowledgment should be made to the Milgo Electronics Corporation who built the buffers and converters associated with the data-handling system. Special acknowledgment is due Mr. J. B. Willmann, who assisted in the mathematical analysis and provided the ideas for the data "combination" technique. Also, many thanks are due to Dr. C. J. Homan, Dr. W. J. Kenny, and Mr. C. J. Pence, program manager.

BIBLIOGRAPHY

1. Bush, K. J., and Hoglund, K. M., "Missile Simulation Using the Digital Simulator Program," Bell Telephone Laboratories memorandum MM-60-4423-5, 29 August 1960.
2. Minzner, Champion, and Pond, "The ARDC Model Atmosphere, 1959," Air Force Surveys in Geophysics, No. 115, August 1959.
3. Dryden, W. A., "Refraction Corrections to Optical and Electronic Tracking Data," Second Joint AFMTC-Range User Data Conference, 7 March 1961.
4. Gill, S., "A Process for the Step-by-Step Integration of Differential Equations on Automatic Digital Computing Machines," Proceedings of the Cambridge Philosophical Society, 1951, 57:96-108.
5. Kendall, M. G., The Advanced Theory of Statistics, London: C. Griffin & Co., Ltd., 1945.
6. Ehricke, Krafft A., Space Flight, Environment and Celestial Mechanics, Van Nostrand, 1960.

INFORMATION PROCESSING FOR INTERPLANETARY EXPLORATION

T. B. Steel, Jr.
System Development Corporation
Santa Monica, California

Payload limitations, environmental constraints, ignorance of the relevant human factors, cost and the generally hazardous conditions surrounding manned vehicles in interplanetary space have combined to elect machines as the probable predecessors of men in the exploration of the Solar System. This, of course, is not to deny that man will soon follow—we may rest assured that he will. Indeed, manned space flight has been called "the ultimate biological experiment." It is a fact, however, that the capability to place a substantial quantity of machinery in the vicinity of the planets will exist well before a human being can make such a trip in safety. Prudence and the urgency of scientific curiosity join to suggest that this capability be exploited in full.

The design of appropriate exploratory devices—planetary probes—is already under intensive investigation in a number of laboratories. Although many of the basic problems of propulsion, guidance, encapsulation, sensing, recording and reporting are well understood and their treatment is in capable hands, the vital question of providing real time direction to the resultant complex of equipment, surprisingly, has been almost totally ignored. For example, Reference 1, a widely used text on the subject of space technology, devotes less than three of its eleven hundred pages to the subject of computers—furthermore, over half of those three pages are given over to photographs of componentry! This seems an undue de-emphasis. The principal objective of this paper is the exploration of this gap in the development of the science of astronautics.

Certainly few would doubt that the proper substitute for a human brain in this context of space exploration is a digital computer, a thesis fully supported in the sequel. If the literature is representative, most astronautical engineers and instrumentation designers seem to expect the employment of a device similar in character to a missile guidance computer, although, as was observed above, the subject does not seem to have been given very serious consideration. It is argued below that the resulting picture of a lightning-fast, fixed program computer as a planetary probe control organ is almost certain to be incorrect.

It must be kept in mind throughout, of course, that this question of the sensible design of an information processing system for a planetary probe cannot, realistically, be divorced from a study of the total probe system. A complete systems analysis is essential if optimum allocation of resources is to be obtained and, in view of the enormous expense of interplanetary exploration and the absolute limitations on available scientific manpower, a serious pursuit of this ideal of optimization is clearly imperative.

In addition to the need for a systems analysis of individual probes, it is important to view each particular probe as a subsystem of the larger complex of equipment and experiments whose total mission is the advance

of human knowledge about the physical and biological character of the Solar System. Study of the interaction among soft-landing vehicles, fly-by probes, the Orbiting Astronomical Observatory, eventual manned space travel and a variety of other experimental activities is an integral part of the design process for each.

No attempt has been made to cover in depth the several points noted in the preceding two paragraphs. This paper is less ambitious. What has been done is to provide a rough estimate of the influence of those characteristics of a total probe system that appear to have a bearing on the information processing subsystem, discuss appropriate configurations of hardware and software in the light of requirements imposed by the task and environment, and present some preliminary conclusions concerning the design and programming of a nerve center appropriate to an unmanned vehicle intended for exploration of the Solar System. The fact that the conclusions of this analysis, superficial as it is, are at considerable variance with the tacit assumptions current in the literature is excellent evidence of the vital need for a complete systems analysis prior to any design effort.

In order to develop the theme, it is necessary to re-examine the raison d' etre for planetary probes and infer the general characteristics of a probe as a system. In its broad outlines, of course, the mission of such a vehicle is straightforward. A planetary probe must carry appropriate instruments to the vicinity of a selected celestial object, record physically observable data and report this data accurately to sites on Earth. On the one hand, a planetary probe may be merely an instrumented comet which is placed in a near-miss orbit; i.e., a fly-by probe. Alternatively, it may be subjected to capture by the target planet. In this latter case it may be designed either to enter a circum-planetary orbit as a satellite or else to make a landing, hard or soft, on the surface of the target body.

In the event that the option of a soft landing is exercised, the payload package of the probe may be mobile as a unit, possess mobile sub-units or be totally and permanently bound at the point of landing. The data collection instrumentation may be wholly passive, sensing only signals that are generated naturally by the environment, or may include active elements, dynamically interfering with the environment in order to elict informative responses.

The data gathered as a result of these observations may be reported to Earth either through some form of interplanetary broadcasting or by actual recovery of some portion of the probe. The recovery tactic, of course, requires a two-way mission plan. Direction and supervision of the behavior of the probe and its instruments may be Earth based—in which case there is a requirement for Earth-to-probe communication—or this control may be internal to the probe. In the event that control is self-contained, reactions may be pre-programmed in their entirety or left subject to some degree of adaptive modification.

A wide variety of possible destinations for planetary probes may be contemplated, ranging from the Moon to the outermost planets. The ambient conditions to which these probes and their instruments will be subjected are extremely varied. On the bright side of Mercury the temperature is sufficiently high to liquify some of the less refractory metals, while some of the satellites of the trans-Jovian planets are cloaked in methane and ammonia snow. Pressure may be virtually absent, as in interplanetary space, or heavier than that found on the terrestrial ocean floors, as in the atmospheres of the giant planets. The local chemistry may involve the three atmospheres of CO_2 anticipated at the surface of Venus or a corrosive poison of free radicals such as expected in the atmosphere of Jupiter. Omnipresent hazards during transit include hard radiation and micrometeorites. It should require little imagination to realize that probes of many different designs are essential.

In direct contrast to this clear requirement for multiple designs in the case of instrumentation, shielding, and the like, the hardware of the control and information processing subsystems of planetary probes seems to be rather free of dependence on the precise mission, excepting, of course, protective covering and, possibly, some component design. This point may be made most emphatically when the control center envisioned is a stored program computer. Certainly there will be differences of detail, especially in the linkages to the instrumentation, but this is a matter of only incidental concern to the considerations of this essay.

While the substance of the remarks below is speculative and quite general in scope, it seems wise to create a specific image to serve as an anchor for the imagination. Accordingly, in the remainder of this paper attention will be focused on a particular mission: that mission most likely to first benefit from these considerations. The period unmanned exploration of the Moon has already begun and, if the present Apollo schedule is realistic, may have given way to manned exploration by the time the suggestions of this paper could be significantly implemented. Instrumented comets have already begun to explore the inner Solar System and others are expected momentarily. On the other hand, a brief reflection on the energy requirements indicates that direct examination of the outer Solar System, beyond Mars, will remain infeasible until exotic propulsion systems become practical.

It seems reasonable, therefore, to center attention upon a second generation of relatively sophisticated probes aimed primarily at Venus and Mars. As a consequence of intense study over many years and the fortunate circumstance that its surface is accessible to visual observation, a good working hypothesis is available concerning conditions on the surface of Mars. The situation regarding Venus is far less happy. Accordingly, this paper will concentrate on a specific Mars probe. It must be kept in mind, however, that the general design strategy espoused herein is catholic, applying equally well to probes of Venus and other parts of the Solar System.

Prior to detailed examination of a proposed Mars probe, however, it will be instructive to reflect briefly on the nature of Venus [2], as it provides an excellent example of the uncertainties which may be overcome only by expensive repetition or inherent flexibility. Four rather different views of the nature of the Cytherean surface have their serious advocates, and present evidence is, as yet, unable to decide with certainty among them. The classical picture of Venus shows a steaming swamp similar to the geologist's conception of the Earth in its Carboniferous era [3]. Recently this view has begun to give way to a picture of an arid, wind-swept desert overlain by clouds of dust [2]. Fred Hoyle [4] has proposed that the entire planet is covered by an ocean of oil with the cloud layer a giant umbrella of smog. Finally, Menzel and Whipple [5] have argued in favor of a global ocean of soda water. Although the desert picture is rapidly gaining adherents, the jury is still out. Small wonder that the hypothetical planner of a manned expedition to Venus would be in a quandry over whether to send along a paleobotanist, a mineralogist, a petroleum geologist, or a deep-sea diver with a bottle of Scotch.

Man has long speculated on the nature of the planet Mars, and the experiments which will be conducted from planetary probes, such as the subjects of this discussion, will go far toward giving definitive answers to many often-asked questions. Much has been deduced already, however, about the structure and general character of Mars on the basis of terrestrial observations [2, 6, 7, 8, 9]. The resulting descriptive picture, together with some relevant numerical data, is summarized in the next few paragraphs.

Mars is the fourth planet in the Solar System and the next out from Earth. Its orbit is relatively eccentric and about half again as big as the orbit of Earth, resulting in a Martian year almost twice as long as an Earth year. A small planet, Mars has a radius about half that of Earth. This small size combined with a density lower than that of Earth leads to a low surface gravity and, consequently, a thin atmosphere. The internal structure of the planet is estimated to be analogous to that of Earth but with a less well defined core and a more equitable distribution of radioactive material throughout its various layers. This accounts for the apparent absence of substantial tectonic activity and the nearly—although not totally—smooth surface.

The Martian atmosphere is almost entirely nitrogen with traces of argon, carbon dioxide, water vapor and (presumably) oxygen. Atmospheric pressure is 85 millibars at the surface of Mars—approximately that of the Earth's atmosphere at an altitude of 16 kilometers. Clouds are observed in the Martian atmosphere at four altitude levels. Yellow clouds near the surface are probably dust clouds raised by desert storms. They may on occasion cover a substantial fraction of the planet's surface and are seen to be carried for several thousand kilometers by winds. At an altitude of 15 to 25 kilometers white clouds are observed. Polarization studies suggest that these clouds are composed

of tiny ice crystals and have much in common with cirrus clouds on Earth. Just above the white clouds are found blue clouds—again presumably tiny ice crystals, analogous to terrestrial mother-of-pearl clouds. Finally, in the 75-100 kilometer region another layer of blue-violet clouds is observed; these are presumed to be minute carbon dioxide crystals—clouds of dry ice.

The electromagnetic properties of Mars are unknown from direct observation, but physical theory suggests the existence of a mild planetary magnetic field, van Allen belts of less strength than those found around the Earth and a problematical ionosphere which is likely to be temporary and is certainly situated almost at the planet's surface.

Four surface phenomena are important to recognize; pole caps, bright areas, dark areas and canals. Mars has pole caps which are observed to form during the autumn and winter in each hemisphere under a thin icy mist in the atmosphere. The caps are white in color with occasional yellowish tints and are probably analogous to terrestrial hoar frost with an admixture of dust. At maximum a cap will extend to about 65 degrees latitude and is not more than five or six centimeters thick at its center. During the spring the cap will melt from its edge (actually hoar frost sublimes directly to vapor at the ambient conditions on Mars) and a wave of humidity is observed to pass along the surface of the planet, across the equator and to the other pole.

Three quarters of the surface of Mars is taken up by the bright areas which are responsible for the planet's characteristic reddish color. These areas are sandy deserts of silicated dust colored red by ferrous oxide and other red metallic minerals. Visual and polorimetric observations are consistent with a moderate relief of the surface. Irregular retreat of the pole caps suggests a topographical variance of about 1000 meters [10]. The remainder of the surface is covered by the dark areas, greenish brown in color, fluctuating in intensity with the seasons. Current theory holds, for the most part, that the phenomena observed in these dark areas are best explained by the existence of a living plant community.

The possibility that Mars possesses an indigenous biosphere has, of course, long provided one of the most exciting vistas in scientific inquiry. Speculation on the reality and nature of a Martian biological system has been rife for fifty years [11]. Current thinking [9] on this subject draws a picture of higher plants with broad, flat and very thin leaves oriented to the sun in the daytime and rolled up into a small cylinder at night. Fast warming during the daylight hours is facilitated by pigments black to ultraviolet, visible and infrared radiation, possibly accompanied by a color change toward white at night. During the night the plants would either noctivate or freeze solid, only to resume full metabolic activity with the coming of dawn.

The poor oxygen availability may be supplemented by micro-organism stimulated reduction of iron oxides. In any event, Martian plants should be extremely efficient by terrestrial standards in conserving oxygen. Given the existence of higher plants, there is the distinct possibility that individuated, mobile biota (usually called animals) exist.

Finally, the ubiquitous canals are now known to striate the planet. Their existence is no longer in question as there is now available direct photographic evidence, but their nature remains obscure. Determination of the true nature of these canals will be one of the major tasks of early Martian exploration, although the scientific importance of them is liable to be small—the explanation for the canals will probably be obvious a posteriori. Nevertheless, so much speculation has been expended on these canals that it is certain that human curiosity will insist on satisfaction.

The mean surface temperature on Mars is found to be -40°C in contrast to +15°C on Earth. The diurnal variation of temperature near the equator in the desert areas has a run from -55°C to -3°C with the minimum just before sunrise and the maximum about 2 P.M. local time. The dark areas average 8°-10° higher both day and night. The atmosphere is quite dry. Indeed, the mean amount of precipitable water vapor expected in the atmosphere at the equator is 0.1 mm and the maximum over the pole caps is only 0.7 mm! Although clouds are sometimes present, as indicated above, they are quite rare by terrestrial standards, there being many days when no clouds at all are observed anywhere on the planet, except what appears to be a sunrise fog. The Martian wind pattern is similar to that of Earth except that irregularities are mitigated by the absence of an

ocean-continent division. Wind velocities of 15 kilometers per hour are normal but velocities of up to 60 kilometers per hour have been observed.

The above account of the general characteristics of Mars has been given overconfidently. So little is certain and so much is hypothetical that it seems best to give a positive description and then provide a general disclaimer. It is likely, however, that this picture is correct in most essentials, and it is certainly adequate for the remainder of this discussion. Tables 1-6 below summarize and quantify the preceding discussion.

Table 1

Orbital and Rotational Elements of Mars

Epoch 1920 Jan $0^d.0$ GMT

major axis	2.281×10^{13} cm	1.524 ⊕
eccentricity	0.09333	
inclination	1° 51' 33"	
perihelion	334° 35' 12"	
period (year)	5.9355×10^7 sec	1.88 ⊕ *
rotation (day)	8.8775×10^4 sec	1.015 ⊕

*Martian year = 668.6 Martian days.

Table 2

Areoid

Mean radius	3.313×10^8 cm	0.520 ⊕
Oblateness	0.005215	
Surface area	1.379×10^{18} cm^2	0.270 ⊕
Volume	1.522×10^{26} cm^3	0.141 ⊕
Mean density	4.24 gm cm^{-3}	0.768 ⊕
Mass	6.45×10^{26} gm	0.108 ⊕
Surface gravity	3.92×10^2 cm sec^{-2}	0.400 ⊕

Table 3

Internal Structure

Composition	70% Silicate phase		30% Iron phase	
Depth (cm)	Radius (cm)	Pressure (dyne cm^{-2})	Density (gm cm^{-3})	Composition
0	3.313×10^8	0	3.28	Mg_2SiO_4 Olivine I
3.00×10^6	3.283×10^8	3.0×10^9	3.30	
2.08×10^7	3.105×10^8	2.5×10^{10}	3.36-3.87	
5.00×10^7	2.813×10^8	6.4×10^{10}	3.93	$(Mg,Fe)_2SiO_4$ Olivine II
1.000×10^8	2.313×10^8	1.3×10^{11}	4.03	
2.000×10^8	1.313×10^8	2.3×10^{11}	4.17	
2.583×10^8	7.30×10^7	2.8×10^{11}	4.25-10.1	
3.313×10^8	0	3.5×10^{11}	10.3	(Fe,Ni)

Table 4

Martian Seasons

Heliocentric Long.	N. Hem.	S. Hem.	Duration (Mars days)
87°-177°	Spring	Autumn	194
177°-267°	Summer	Winter	177
267°-351°	Autumn	Spring	142
351°-87°	Winter	Summer	156

Table 5

Martian Atmosphere

Composition	Gas	Thickness (cm NTP)	Volume	
	N_2	1.8×10^5	0.985	
	A	2200	0.012	
	CO_2	440	0.0025	
	O_2	100	0.0005	
	Kr	0.2	10^{-6}	
	Xe	0.02	10^{-7}	
Structure	**Height (km)**	**Temperature (°C)**	**Pressure (mb)**	**Phenomena**
Convective Equilibrium	0	-20	85	Yellow Clouds (dust)
	5	-40	64	
	10	-60	48	
	15	-80	36	
	20	-100	27	White Clouds
	25	-110	20	Blue Clouds
	30	-120	15	(H_2O ice)
	35	-130	11	
	40	-145	8	
Tropopause Temperature min	45	-160	6	
Radiative Equilibrium	50	-145	5	
	55	-120	5	
	60	-110	4	
	65	-105	4	
	70	-100	4	Blue-viloet Clouds
	75	-100	3	(CO_2 ice)

Table 6

Martian Surface Temperature

Seasonal temperature - °C - local noon mean

	Perihelion	Aphelion
equator	+20	0
tropics - summer	+20	+10
tropics - winter	-10	-20
temperature - summer	+20	0
temperature - winter	-30	-50
poles - summer	+10	-10
poles - winter	-80	-100

Consider, now, a descriptive exposition of the voyage of an hypothetical Mars probe. As both the circum-planetary orbit and the hard landing (impact) missions are relatively limited in their information processing requirements, the obvious choice of an example for this discussion is an unmanned, one-way, soft landing probe whose target is the surface of the planet Mars. Although the two-way mission is perhaps more challenging to the astronautical engineer, it is the one-way mission that presents the more complex set of problems to the designer of the information processing subsystem. If data reporting were to take place by recovery, as might be expected for the two-way mission, the information processing system would not have to organize the data collected; it could merely

store it for data reduction on Earth. In the one-way situation, however, there is a need for broadcasting data with the attendant, and potentially involved, requirement for encoding prior to transmission. Further, the high energy needs of the two-way mission suggest that consideration of the problems of the one-way case is realistic in the light of present constraints on payload mass and volume.

Due to the present state of the propulsion art, serious consideration must be given to the energy requirements of any mission in interplanetary space [12]. First and foremost, of course, this translates into the mass limitation on payload noted above. It also limits the possible launch dates as the relative positions of the source and target planets must be appropriate if a ballistic trajectory between them is to have minimum energy requirements. Table 7 below lists those time periods in the remainder of this decade during which it will be possible to launch a probe into a minimum energy transfer ellipse in such a way that the Earth-Mars constellational requirements are satisfied for a planet-to-planet transfer. There are, of course, precise points in time when the exact conditions for a true minimum are extant, but anytime within roughly a month on either side of such a minimum point may be considered acceptable in the Earth-Mars case.

Although the development of non-chemical propulsion systems will reduce and, perhaps, eventually eliminate the significance of minimum energy transfers, and various system considerations such as communication power requirements may yield optimum orbits different from minimum energy transfer ellipses, Table 7 will serve as a satisfactory guideline to the type of constraints that will be in force for the remainder of this decade. For the purpose of illustration, then, this table will constitute a fundamental limitation.

While it is possible that some of the concepts discussed below could find their way into a system whose operational date is late in 1964, past experience with the lead times necessary for the effective combination of hardware and software, together with current vehicle planning, suggests the end of 1966 as a reasonable target date for initial serious implementation of the ideas considered in this paper. The date itself is important here only for the indication it gives of the techniques that may be available for incorporation in the information processing subsystem of the probe. In addition, of course, selection of a particular date permits the discussion of design criteria to be given substance in terms of a specific example. It must be re-emphasized, however, that the principles involved are quite independent of the quantitative material that is exhibited for this special case.

Consider, then, the following program for an exploratory Mars Probe; an admittedly artificial plan, but one whose parameters are well within the range of the feasible. The vehicle, a three stage device with a separable payload—perhaps a Saturn C-4, currently programmed for the Voyager project [13]—is launched around Christmas in 1966[1]. The first two stages exhaust their propellant and are detached from the third stage and payload after placing them in a ballistic trajectory toward the orbit of Mars. (It is possible that this is accomplished in steps by first entering an orbit around the Earth, rendezvousing with a supply rocket, refuelling in space and then entering the excape maneuver, This is irrelevant to present concerns.) Current estimates suggest that about 2500 kilograms can be placed in a circum-Martian orbit in 1967. Figure 1 illustrates a typical launch profile.

The probe remains in the transfer ellipse for about 252 days, the precise time being dependent on the launch date, before arriving in the vicinity of Mars. During this time it is in free fall, with the possible exception of brief periods when the vernier rocket motors are activated for minor trajectory correction,

[1]The Saturn C-4 project was subsequently cancelled.

Table 7

Possible Earth-Mars Launch Dates

1962	1963	1964	1965	1966	1967	1968	1969
Oct-Nov	—	Nov-Dec	—	Dec	Jan	—	Jan-Feb

TYPICAL LAUNCH PROFILE

1 Vertical ascent
2 First stage powered flight
3 Second stage powered flight
4 Second stage coast
5 Third stage powered flight
6 Coasting in orbit
7 Final boost into interplanetary trajectory
I First stage rocket shell trajectory
II Second stage rocket shell trajectory

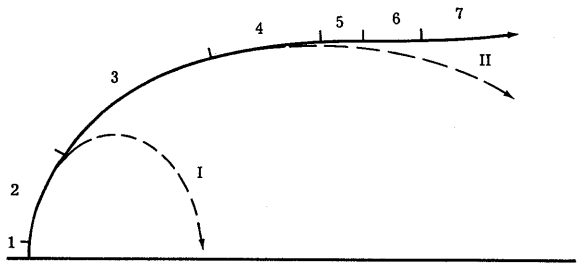


Figure 1. Typical launch profile.

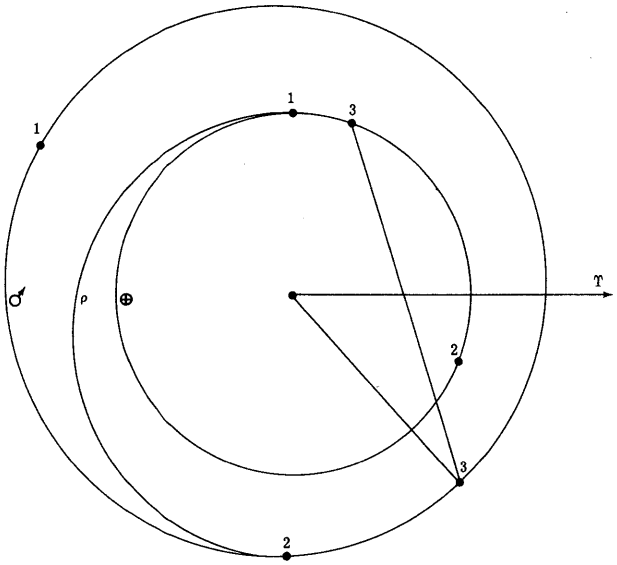
and in the radiation and micrometeorite infested environment of the interplanetary vacuum. The orbit of the probe is diagrammed in Figure 2.

If everything has gone according to plan, the probe will enter a terminal maneuver about September 1, 1967 (Earth date), powered by the remaining third stage, and penetrate the atmosphere of Mars, probably in a grazing trajectory that decays into a high altitude orbit from which final letdown is made by aerodynamic means, Mars having sufficient atmosphere to permit an aerodynamic decay trajectory. (Again, as in the case of launching, the precise procedure is not germane.) Figure 3 suggests possible landing profiles.

In September 1967, spring has just begun in the Martian southern hemisphere [6]. This is a relatively satisfactory time for data gathering, insofar as terrestrial observation can determine, as many of the known phenomena are in their most active state in the springtime. A likely point of landing is in one of the dark—possibly vegetative—areas in the southern tropics such as the Sinus Sabaens or the Aurorae Sinus. These areas are suggested because of their proximity to confluences of dark and bright regions with the problematical canals.

In a siderial coordinate system the Earth gains just under one half degree per day on Mars. As a consequence, after about eighty

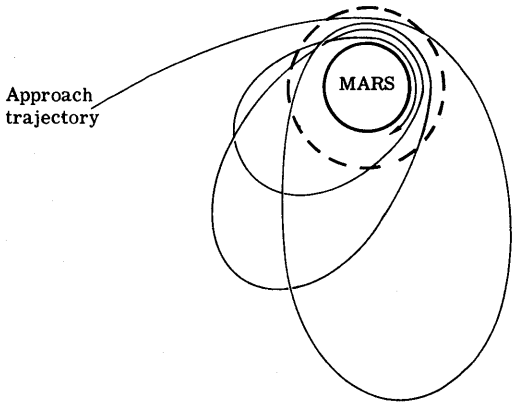
ORBITS OF EARTH, MARS AND PROBE



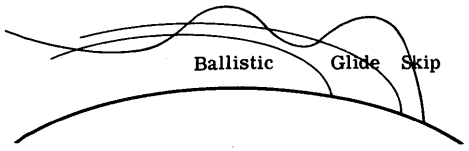
1 December 21, 1966
2 September 1, 1967
3 November 20, 1967

Figure 2. Orbits of Earth, Mars and Probe.

TYPICAL LANDING PROFILE



Braking ellipses



Final descent variations

Figure 3. Typical landing profile.

days the Earth will be within thirty degrees of the Sun as seen from Mars. At this point communication difficulty may well develop as a result of Solar radio noise and the need to transmit signals through the high energy electron field of the Solar corona. It is as yet unknown what effect the corona will have on communication across interplanetary distances, but it is not likely to be beneficial. The Earth will not reappear from behind this pall until nearly eleven months later. Accordingly, no more than the first three months following the probe's arrival on Mars may be available for the active performance of the mission. Figure 2 illustrates the relevant planetary configurations.

In view of the stringent mass limitations that will still be in force in 1967 (barring the unexpected), it is not likely that the main instrument package of the probe will be mobile. Biological sample collector tapes may cover a few square meters of Martian surface and an antenna lifted, say, two meters above ground level on an extensible rod will make possible communication with small mobile devices containing data collection instrumentation and radio equipment. This rod might also mount a television camera. The horizon for this equipment ensemble on Mars is about four kilometers, although the possibility of extending the horizon by attaching an antenna to a tethered balloon should be studied. What evidence there is suggests that any Martian ionosphere is so near the surface that communication will be limited to line of sight means [6]. Exotic communication systems that might be feasible on Mars, such as those employing the phenomena of tropospheric scatter and ground wave propagation, are too demanding of power and mass for consideration here [14].

Thus, the exploration capability of a probe such as is described above is limited to fifty square kilometers and, perhaps, half of one season (southern spring on Mars lasts 142 days). To cast this point in familiar terms, it might be asked, how much can be learned from an examination of, say, a New England meadow during the month of April and half of May? The answer is clear: a great deal in absolute terms, but relatively little in the context of an entire planet. It follows that it is imperative to extract as much information as possible from this fifty square kilometers of Martian countryside and, in particular, not to miss anything of interest, for repeated return to the same site would be prohibitively expensive with so much total ground to cover. Minor extensions of coverage by novel procedures such as the balloon antenna in no way disturb the substance of the argument.

The instrumentation required to gather this intelligence about the Martian landscape and its inhabitants, if any, is complex and multifarious. It will include thermocouples, photometers sensitive to infrared, visual and ultraviolet radiation, seismographs, particle radiation detectors and other passive recording devices. In addition there will be active instruments that sample and analyze biological and mineralogical specimens by chemical, visual and spectra-photometric means.

For example, it has been suggested [15] that samples be collected by a transparent, travelling ribbon combined with a soil auger; the sampled material separated by flotation in a dense liquid, and the separated matter brought to the aperture of a microscope. In addition, a cytochemical processor can be employed relatively easily if a travelling tape is used for collection as reagents can be applied sequentially. Use of appropriate specific enzymes and fluorescent stains combined with microscopy in the ultraviolet (2600Å and 2800Å) will result in selectivity for nucleoproteins as well as RNA and DNA.

To the information processing subsystem, however, all of these devices will be digital signal generators with variable data rates and characteristic parameters subject to alteration by the central control system. These parameters include such things as scan rates and geometrical orientation. Presumably, the information processing system can be designed without giving too much thought to the semantic content of the data to be gathered. The instrumentation does have some impact on the information processing subsystem, however. The UV microscope may be designed as prefocused and rigidly supported or subject to servo focusing, either with special equipment or through a central data processor.

A brief glance at the expected surface conditions of Mars, as reviewed above, shows that they will not stress the design of the probe or its information processing subsystem. To recapitulate, the diurnal variation of temperature in the tropical springtime is from about -35°C to +10°C and the atmosphere, 99.7% nitrogen and argon, is free

from complicating factors. The surface atmospheric pressure, 85 millibars, is adequate to provide protection from the bulk of serious radiation from space, although absence of a permanent ionosphere implies that the instruments will have to contend with the Solar ultraviolet. Certainly the atmosphere is sufficient to provide shielding against any major threat from micrometeorites. Although it is clearly submarginal for men, the Martian surface environment is relatively benign to machinery. As has been noted above, this happy situation is far from general in the Solar System.

From the above outline, sketchy as it is, several important conclusions may be drawn regarding the design philosophy of the information processing subsystem for a planetary probe. Three cardinal points that will underlie such conclusions stand out: (1) the time scale of a mission is of the order of months or years, (2) for some part of the time at least, the environment is, at best, suboptimal, and (3) the desirability of employing more sophisticated procedures than blind data gathering cannot be overemphasized.

A recognition of the importance of this long time scale is vital. It adds a degree of freedom to the computer design as the capacity for very high speed computation can be traded for storage capacity. During the entire process of data gathering, analysis and reporting there is no requirement for computational speed, provided the data can be recorded as fast as it is obtained. In most of the desirable observations the rate of data collection can be adjusted until it is commensurate with the capability of the computing system. The central computer must monitor its own data rate requirements, however. This argues for flexibility and militates against a fixed program.

The one phase of the process that might seem to require high speed calculation is guidance. During launch, the guidance computation may be performed on Earth and handled in the vehicle by special equipment housed in a dispensable pod attached to the second stage frame. The launch activity need not involve the payload computer at all. Trajectory correction during the eight and a half months of transfer do not impose any requirement of high speed on the system. Accurate control of the correction impulses is far more important than haste in execution. It is only during the terminal approach and landing maneuver that some serious stress might be placed on the capability of the system. Care must be exercised at this stage of the operation to determine a suitable touchdown spot and this is a matter that cannot be adequately preplanned. In order to obtain the full horizon potential, it is necessary to land on a rise or, at least, on level ground. Terrestrial observation is unable to determine topographical differences of a few meters on Mars, and it is unlikely that fly-by probes launched prior to a soft landing try will be able to produce sufficient accuracy. Indeed, it is possible that wind-blown sand could shift the terrain in this amount over the two-year span between probes. The probe must, therefore, be prepared to make this determination for itself. Even here, however, the braking ellipse process lasts for many hours, obviating the need for nanosecond logic and wired programs.

An important consequence of this analysis of speed requirements is that it permits the use of contemporary logical design and, more to the point, present-day components in the information processing subsystem. Existing solid-state components such as diodes, transistors and cores are relatively gross and, consequently, are not very sensitive to particle radiation—in particular they are not prone to permanent damage as a result of ionization tracks. The ambient cosmic radiation dosage in interplanetary space is about 10 roentgens per year [16], some four orders of magnitude below the critical threshold for damage to the components under consideration. This situation no longer prevails for the microminiaturized components seemingly necessary for extremely high speeds. Thin films and other products of molecular electronics technology are liable to give considerable trouble after months in space unless protected, and the high energy of cosmic radiation precludes adequate shielding.

The use of components with a high threshold for radiation damage reduces the probability of component failure, but there <u>is</u> considerable hard radiation in interplanetary space and the failure probability is nonzero. In particular, during periods of high solar activity, relatively intense beams of corpuscular radiation as well as X-rays sweep through the Solar System, being gradually attenuated as they recede from the Sun [17]. Also the well-known radiation belts centered on the terrestrial magnetic equator and their

problematical counterparts near other bodies in the Solar System form potential hazards to electronic devices.

As the probe departs from the vicinity of the Earth, preparatory to entering its interplanetary transfer ellipse, it must, as noted above, pass through the van Allen belts of particle radiation [18]. Although the vehicle remains in the neighborhood of these belts only briefly, the ambient radiation is of sufficient intensity to form a hazard to improperly designed and protected electronic equipment. Figure 4 shows the radiological geometry of the belts.

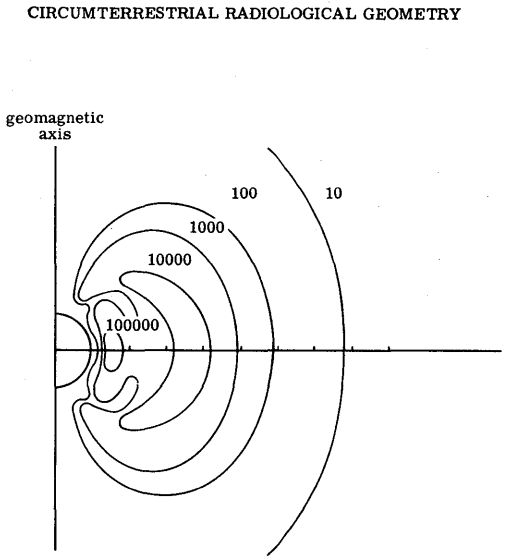


Figure 4. Circumterrestrial radiological geometry.

In addition, there are problems associated with thermal extremes. These problems can be mitigated by appropriate capsule design but are unlikely to be entirely eliminated. For example, in the vicinity of the Earth's orbit, a space vehicle with a black surface will have its skin temperature range between -70°C and +80°C, depending on the extent of exposure to Solar radiation. A white surface results in a skin temperature range of -90°C to -60°C. An appropriate structural configuration for the vehicle, combined with the proper skin finish, can produce almost any desired skin temperature within 50°C of room temperature almost everywhere in the inner Solar System. Skin temperature is, of course, not the whole story. Proper encapsulation and internal thermal shielding and insulation can further reduce the temperature range. A case in point is the internal temperature range of Explorer I (1958α) which was maintained between -5°C and +35°C [19]. Further, thermal sensitive servo-mechanisms can provide active compensation for temperature variations resulting from radial displacement relative to the Sun.

Finally, a certain amount of system malfunction will result from normal degradation of component performance as a consequence of the long time scale of the mission. Responsible calculation of the expected number of component failures is not possible, but it is clear that some failures will occur. In view of the difficulty of attaching maintenance personnel to the probe, the system must be designed to function adequately in spite of the failure rate. Both component redundancy and alternate signal routing capabilities are required, and adequate monitoring information must be provided to the information processing subsystem itself so that it can maintain a continuous watch on its own performance.

The foregoing has tacitly assumed that solid-state devices are preferable to vacuum tube systems in this context of interplanetary exploration. There are two reasons for this preference in addition to the basic factor of general reliability. First, solid-state components have the advantage of comparatively small size—an advantage that can be increased by appropriate development efforts. Miniaturization is important not only for the intrinsic minimization of mass but also for the associated reduction in support, shielding, and power requirements.

Second, the tolerance of vacuum tubes for acceleration-induced stress is low. The launch and escape maneuvers will require accelerations of six to eight gravities and the atmospheric braking operation may result in up to twenty gravities and very high rates of change of acceleration. Solid-state devices can be shock mounted to withstand such treatment with relative ease—not so with vacuum tubes.

The magnitude of the problem of micrometeorites in interplanetary space is not

well known [20], and the difficulty of obtaining accurate measurement of the particle sizes by the grid-wire detection devices that have been employed in satellite experiments [21] is considerable. What evidence there is, however, suggests that micrometeorites will be a minor hazard to a reasonably shielded probe.

The requirement for sterilization of the probe has an important bearing on the physical structure of the information processing subsystem. Considerable study, spearheaded by the international committee on Contamination by Extraterrestrial Exploration (CETEX), has made it abundantly clear that interplanetary vehicles can be thoroughly sterilized but only if careful attention is paid to the problem at every stage of design and construction [22]. The most satisfactory sterilization procedure currently known is treatment with ethylene oxide. Fewer types of materials are damaged by this technique than by any other known sterilization method. A concentration of 400 milligrams of ethylene oxide per liter of air will cause sterilization in about six hours at room temperature.

Experiments have shown, however, that ethylene oxide sterilization is not adequate for components manufactured by typical industrial procedures. Although the component surfaces become sterile, viable microbiological contamination remains in gas-tight pockets of conventional electronic components. In addition, the plastic structures often used to provide mechanical strength in electronic systems may contain microorganisms. Investigations have shown that hardy bacteria of certain types are quite capable of withstanding the polymerization process employed in the manufacture of these plastic bases. The danger, of course, is that a component might fracture when subjected to the heavy mechanical stress of acceleration, or be damaged by radiation or a meteorite in such a way that internal biota are released. The slightest crack may be sufficient to release biological contaminants. Table 8 [23] shows the fraction of various components internally contaminated even after six hours of ethylene oxide treatment.

In connection with this question of component sterilization it is worth examining the consequences of failure. Considering the typical generation time of 30 minutes for common terrestrial bacteria, together with the observed wind velocities on Mars, it would be possible for a nutrient medium the size of Mars to be occupied in less than a week [15, 24]. The only limit is the available nutrient, an unknown quantity on Mars. At worst, such a proliferation of terrestrial biota could overwhelm and completely destroy a Martian biosphere and, at best, would leave the biological environment hopelessly confused. The undesirability of this is clear, not only on scientific grounds, but also in view of the possible utility of alien biota; e.g., the possibility of medically applicable biologicals.

The fundamental limitations of communications capability form a major constraint on the design of a planetary probe. Three of these limitations have a pronounced effect on the design of the information processing subsystem. They are: (1) the maximum information rate, (2) the lag time for signal transit, and (3) periodic blackout.

The information rate is, of course, a function of the capability of the space-borne transmission system and this is largely a matter of the combined mass of the transmitter and antenna. Studies [25] have shown that if a 250-foot diameter antenna is used on Earth, 100 kilograms of equipment on Mars can provide a little more than 100 bits per second at the Earth-Mars separation pertinent to the current example. While some improvements may be made in this area, the order of magnitude of this rate in application is not likely to change significantly in the next decade.

It is clear that the information capacity of such a small bandwidth must be carefully

Table 8

Microbiological Contamination of Electronic Components

Type of component	Transistor	Capacitor	Resistor	Diode
Percent contaminated	6%	20%	15%	0%

husbanded. All redundancy not immediately related to reliability considerations must be eliminated from the encoding scheme. Indeed, many encoding schemes may be employed, the choice being dictated by the nature of the data to be reported. The best solution to this problem seems to be to allow the information processing system to select its own encodings, dynamically, as the circumstances require. In this event, the system must be able to transmit descriptions of the encodings prior to their use in actual data reporting.

Nothing that has been said so far would preclude the use of an Earthbound data processing system. However, the fact that the transmission of signals is limited to the speed of light by the laws of nature does so. For the probe under discussion, the minimum separation of Earth and Mars is eleven and a half light minutes. Thus, at best, a response to a signal generated by the probe would take twenty-three minutes to report. This is clearly out of bounds for the contemplated landing maneuver, and it is quite liable to be unacceptable for many of the data collection activities. For example, should some element of the instrumentation system detect surface motion—perhaps a tumbleweed analogue, not necessarily a Martian—it would be highly desirable for several other instruments to focus on this source of motion and follow its progress in detail. A delay of more than twenty minutes in initiating this activity might well be fatal to the success of the observation. If the system had built-in wiring designed to handle this particular situation, there is likely to be some other, unanticipated occurrence that turns out to have more interesting properties. Also, it might be the case that the surface of Mars was covered with tumbleweeds and a preplanned program of observation would result in the instruments simply spinning their wheels. While the illustration above is fanciful, it clearly shows the kind of problems that must be solved in order to have a profitable mission.

Brief consideration suggests that the inclusion of sufficient circuitry to accept complicated modifications in real time for the instrumentation program is not a satisfactory solution to this problem. If the accepted range of modifications is adequate to the situation, the required on-board equipment is almost as much as would be necessary for the inclusion of the whole computer system in the first place. Also, while the unanticipated, repetitive occurrence could be handled by broadcast modifications, an unexpected, unique or rare event might well be inadequately recorded. In view of the huge expense associated with the dispatch of each such probe and the number of probes essential to provide adequate coverage, even superficially, it is imperative that each experiment be as complete as possible. The best known way of achieving this goal is to include on board a general purpose computer containing a highly flexible program.

The third limitation mentioned above—periodic blackout—merely adds some punctuation to the situation outlined in the preceding paragraphs. Even if the Earthbound receivers were scattered widely enough to afford twenty-four hour surveillance of Mars, the fact that the target planet itself rotates with a period of about twenty-four hours and thirty-eight minutes is sufficient to destroy all communication links between the probe and Earth for periods of up to twelve hours at a time. Clearly, for a substantial fraction of the mission, a Mars probe must be out of contact with its home base. This circumstance, of course, is not a phenomenon that is peculiar to Mars; it is a general problem on any body in the Solar System. In order to make full use of its capabilities, it is clear that the probe's computer must be on board.

The information processing requirements of a probe are hybrid. Four general areas may be recognized: (1) guidance, (2) data reduction, (3) decision making, and (4) message handling. Only the first two—and of those, guidance in particular—pose any requirement for arithmetic capability. Guidance computations must be carried out with considerable precision—of the order of 0.1" in some of the angular variables [26]—suggesting that a reasonably long word—perhaps 48 bits—is desirable to eliminate, or at least reduce, multiple precision calculations. This point should be subjected to thorough study, however, for word length can be quite expensive in circuitry. The range of values of the variables entering into the guidance computations are so well known that floating point capabilities are not likely to be needed—prelaunch analysis can provide the scaling necessary for fixed point calculations.

The data reduction requirements supply nothing over and above the guidance

requirements. The speed with which data must be ingested for the recording of observations is much less than that necessary for guidance. As was implied above, a microsecond clock is more than adequate. Part of the data reduction problem is, of course, data storage. High capacity serial storage is what is needed here. Miniature magnetic tape systems [27] such as have already been employed in satellite experiments are certainly adequate. There is no requirement for ultra-high speed data rates, and very rapid start-stop acceleration is unnecessary.

The decision making and message handling aspects of the information processing system require bit and byte arithmetic [28] and a full complex of logical instructions. The programming of decision making calls for the logical capability of an associative store [29], but the absence of a need for high speed suggests that a programmed associative store will do nicely. The cost in mass of adding a hardware capability for associative storage is not necessary.

The kind of decision making envisioned here is the type currently being studied in the investigations of heuristic programming. What is needed is an adaptive system that is able to adjust to variations in the environment. Careful reflection will lead to general rules or methods of decision making that can be applied to particular situations as they arise. As has been pointed out above, so little is known of the details of Martian conditions (and even less of the circumstances elsewhere) that preplanning the precise course of every observation is unsatisfactory. In general terms this situation is analogous to that faced by the chess-playing program: the goal is explicit, the overall pattern of behavior clear and the details dependent on factors that are uncontrollable and are unveiled over time. Pursuance of a study into the nature of such a program might well be more fruitful than some of the current investigations into heuristic programming.

The message handling is, of course, principally message composition. Thus, in addition to byte arithmetic, table lookup instructions are vital. Considerable care must be exercised in designing these instructions so that random access storage requirements are not too high. With care these requirements can probably be held to a few thousand words. Indeed, study might show that the optimum solution to the problem of the main store is the use of a high speed magnetic drum.

The above analysis has been highly speculative. Its intention has been to highlight some of the problems rather than to offer clear-cut solutions. Perhaps the most important message is that the design of the information processing subsystem of a planetary probe is a vital and complex matter which must not be left as an exercise for the last minute. The view presented here might be read as expecting such a system as a miniaturized close relative of the 7094, 1604 or S-2000. Other possibilities are certainly present, but they can only become known through thorough study of all the relevant factors.

Indeed, it is this study requirement that is the main message implicit in this discussion. In view of the enormous amount of money, manpower and talent (some 40 billion dollars is projected in the long run for Apollo) that is being put into the space program in the United States alone, to say nothing of the rest of the world, it is clearly incumbent on the information processing fraternity—engineer and programmer alike—to contribute serious effort toward the optimization of this space effort. It can only be done through thorough awareness of the total picture of space science, a truly interdisciplinary study involving virtually every aspect of modern science.

REFERENCES

1. Seifert, H. S. (ed.), Space Technology (Wiley, New York, 1959).
2. Sagan, C., The Planet Venus, Science, 133, 3456 (1961).
3. Russell, H. N., Dugan and Stuart, Astronomy, Vol. 1, The Solar System (Ginn, New York, 1955).
4. Hoyle, F., Frontiers of Astronomy (Harpers, New York, 1955), 68-72.
5. Menzel, D. H. and Whipple, F. L., Pub. Astron. Soc. Pacific 67, 161 (1955).
6. de Vaucouleurs, G., Physics of the Planet Mars (MacMillan, New York, 1955).
7. Kuiper, G. P., Planetary Atmospheres and Their Origin, ch. 12 in Kuiper, G. P. (ed.), The Atmospheres of the Earth and Planets (Univ. of Chicago Press, Chicago, 1952).

8. Kuiper, G. P. and Middlehurst, B. M. (eds.), Planets and Satellites (University of Chicago Press, Chicago, 1961).
9. Salisbury, F. G., Martin Biology, Science, 136, 3510 (1962).
10. Barabashov, N. P., Soviet Astronomy, 2, 814 (1958).
11. Lowell, P., Mars as an Abode for Life (MacMillan, New York, 1910).
12. Ehricke, K. A., Interplanetary Operations, ch. 8 in Reference 1.
13. Space Programs Summary 37-15, Vol. VI, Jet Propulsion Laboratory June 1, 1962.
14. Vogelman, J. H., Propagation and Communication Problems in Space. Proc. IRE, 48: 4 (1960).
15. Lederberg, J., Exobiology, Experimental Approaches to Life Beyond the Earth, in Bijl, H. K. (ed.), Space Research, Proceedings of the First International Space Science Symposium, Nice (Jan. 11-16, 1960), pp. 1153-1170.
16. Newell, H. E. and Naugle, J. E., Radiation Environment in Space. Science, 132, 3438 (1960).
17. Nonweiler, T. R., Effect of Solar Flares on Earth Satellite 1957β, Nature, 182, 468 (1958).
18. van Allen, J. A. and Frank, L. A., Radiation Around the Earth to a Radial Distance of 107,400 km, Nature, 183, 430 (1959).
19. von Braun, W., The Explorers, Astronautica Acta, 5, 126 (1959).
20. Whipple, F. L., The Meteoric Risk to Space Vehicles, 115-124 in Vistas in Astronautics (Pergamon Press, 1958).
21. Manning, E. R., Micrometeorite Measurements from 1958α and 1958γ Satellites, Planetary and Space Science, 1, 27 (1959).
22. Nature, 183, 925 (1959).
23. Phillips, C. R. and Hoffman, R. K., Sterilization of Interplanetary Vehicles, Science, 132, 3433 (1960).
24. Davies, R. W. and Comuntzis, M. G., The Sterilization of Space Vehicles to Prevent Extraterrestrial Biological Contamination, Proceedings of the 10th International Astronautics Congress, (London 1959).
25. Mueller, G. E., A Pragmatic Approach to Space Communication. Proc. IRE, 48: 4 (1960).
26. Wheelon, A. D., Correction of Interplanetary Trajectories, ch. 26-3 in Reference 1.
27. van Allen, J. A. (ed.), Scientific Uses of Earth Satellites (Univ. of Michigan, Ann Arbor, 1956).
28. Brooks, F. P., Jr., Blaauw, G. A., Buchholz, W., Processing Data in Bits and Pieces, Proceedings of the International Conference on Information Processing, 375, (UNESCO, Paris, 1959).
29. Kiseda, J. R., Peterson, H. E., Seelbach, W. C., Telg, M., A Magnetic Associative Memory, IBM Journal of Research and Development, 5:2, 106 (April, 1961).

EDP AS A NATIONAL RESOURCE

The impact of Electronic Data Processing is amplified by the speed with which it has entered and spread throughout our national life. To those who are working in the computer sciences, these vast and sudden changes may seem natural and expected. But to others, whose only contact is, perhaps, a cardboard check with rectangular holes and the legend, "Do not fold or spindle," EDP is at least strange, if not awesome. Amidst their general feeling that it represents progress and is therefore good, there lurks a residual doubt, a vague and undefined fear of the unknown, the mysterious. As a technology, EDP is neither good nor evil, but rather a resource to be employed wisely for the common good.

From the standpoint of technology, the future is likely to be more exciting and wondrous. Machine translation, character recognition, self-organizing systems—the very words stir the imagination. As never before, what is possible appears limited only by what can be imagined. The possibility of very large memories, orders of magnitude larger than those presently available, implies an increased problem-solving ability for processing systems. Increased speed, increased capacity, increased reliability all seem to be technically within reach. But to what uses can these increases be put? What are the needs that must be satisfied? What are the problems to be solved? Norbert Wiener has said that "machines can and do transcend some of the limitations of their designers, and that in doing so, they may be both effective and dangerous." Since a computer can really only do what it is told to do, the danger, if any, lies in our not knowing what to tell the computer.

It is interesting to speculate on the possibility of achieving artificial intelligence and on whether or not it constitutes a menace. A. L. Samuel points out that "throughout history, man has discovered many properties of nature which he has not understood, and he has proceeded to use these properties both for good and for evil in spite of lack of comprehension. We must, therefore, reckon with the possibility that man may yet create a Frankenstein monster in the form of an intelligent machine more or less by accident and long before he has developed the detailed knowledge required to control his own creation." Samuel goes on to state, that "such a development is extremely unlikely" . . . and . . . "our chance of constructing a device resembling the brain of man is less than something like 1 in 10^{12}." He concludes that "we have nothing to fear from the machine in terms of domination."

Interesting as it may be to speculate on the machines of the future foreshadowed by Perceptrons and Checker-playing Programs, we must not lose sight of the vital force EDP has become in our society today. The availability of electronic digital computers in many colleges and even in some secondary schools will require changes in traditional curricula as classes of problems previously inaccessible become soluble. The industrial demands for computer scientists and programmers in large numbers leads to new interdisciplinary courses and programs. The educational system must train those who will design and operate the large-scale complex and composite systems which now become feasible.

All of this is but one part of the impact on education. A second area of great potential

---

NOTE: This brief introductory is intended only to call attention to some of the interesting questions which the panel may discuss. It is not intended to anticipate the panel discussion nor is it intended to reflect the views of any of the panel members.

impact is the application of the digital computer to automated instruction. Certainly one expects there will always be the need for personal contact between teacher and pupil. It is difficult to conceive of a satisfactory education, entirely synthetic and without the direct interplay of human minds. Nevertheless, Programmed Learning and Computer-based Instruction will find a useful role in future education. A proper balance is required. What is this balance and how it can best be achieved are questions open for discussion.

At the same time that our educators are training their students for and with digital computers, they must also provide the cultural and social understanding necessary for properly assimilating the profound changes which are taking place.

The threat of technological unemployment induced by automation is now being widely discussed. The problem is not new and predates the digital computer. The technological displacement occurring today is part of a continuing process which began with the Industrial Revolution. The net effect has been an increased requirement for highly educated and trained people and a corresponding decrease in the need for the uneducated and unskilled. We must determine those functions which are best performed by people, and structure training and educational programs to prepare our citizens for these jobs. Machines will continue to take over the performance of simple repetitive tasks as well as some complex and sophisticated jobs. But there is the need to preserve human values and human dignity.

It is not only the unskilled blue collar and clerical worker who is affected. The nature and number of managerial positions will also change. The management process is significantly influenced by EDP. Many decision processes can now be programmed and thus centralization of control is encouraged. For example, inventories and schedules can be controlled centrally for an entire corporation even where the warehouses or plants are geographically widely dispersed. Managers now establish the decision rules from which the computer generates decisions. At the same time, greater technical or scientific skills and insight will be demanded of managers as the areas, in which decisions are required, move more closely towards the problems associated with new technologies.

Problems and change accompany the introduction of EDP, but with it also comes a vastly expanded ability to solve problems, to control processes, and to handle information. America's defenses are strengthened by the extension EDP facilities of the Department of Defense. Command Control, logistic support, fiscal management, intelligence coordination, and research and development, all require large-scale data processing.

With hardly an area of national activity untouched by EDP, informed discussion by leaders in many fields is essential if the potential of EDP is to be realized. These discussions should define the problems, expose the impediments and explore a variety of solutions. A number of difficult socio-economic problems have already arisen. The continuing technological explosion will almost certainly raise other problems of equal difficulty. In some circles, the word "Automation" has replaced Mephistopheles as the connotation of evil. Indeed, it would be foolish to deny the growing concern over technological displacement. However, it is important that our energies be directed towards positive goals. We live in dangerous times with complex and difficult problems. The speed with which events occur emphasizes the need for full utilization of our national resources in electronic data processing.

PLANNING THE 3600

Charles T. Casale
Control Data Corporation
Minneapolis 20, Minnesota

INTRODUCTION

The planning of any large computer system invariably requires a framework within which the planners may explore numerous alternatives. Such a framework is itself the result of much planning, but need not be finely detailed. The basic 3600 planning framework will be briefly presented in this paper. The bulk of the paper will concern itself with the specific organization of the 3600 system and planning considerations affecting the organization. Details of the 3600 system will be included where pertinent.

Planning Framework

The planning framework for the 3600 system was solidified in late 1961. A computer system was to be designed for delivery in the spring of 1963 and was to have several broad characteristics. The category of the computer was to be extra large, ranging somewhere between the 1604-7090 class and the 6600. The machine was to employ solid-state circuitry, and was to be easy to program. Some degree of compatibility with the Control Date 1604 was to be maintained unless total abandonment would demonstrably improve the performance of the system. Other requirements, such as ease of manufacture, moderate cost, maintainability, and clarity of design were to be no less than presently found in practice. The computer was to have a reasonably long life for the user, and to be easily adapted to the major problem-oriented languages. Within this basic framework, the planning was initiated.

Initial Planning

To the basic frame of reference, several corollary statements were added. These had the effect of restricting the range of possibilities with the further effect of hastening the planning process. The major corollary was that the system machine language be the same as or comparable to the 1604 language. The effect of this would be to avoid the temptation of initiating a closed loop of new machine language-to-new machine oriented language-to-new problem oriented language-to-new machine organization and back. Such a loop would have resulted in a considerably longer planning period, but with no assurance that the outcome would indeed be more favorable. In addition, it was considered that the present tri-lingual capabilities of most computers (FORTRAN, COBOL, ALGOL) would be extended and expanded for most new computers in all classes. Recognizing the sometimes fleeting quality of problem oriented languages, it was considered unwise to slant the system design toward any one language. To avoid the possibility of a system which would not be efficient in any problem-oriented language, however, it was decided to incorporate additional operational features on the machine code level.

Examination of the 1604 machine language and instruction set showed that nearly all present generation computer operations were available in the computer. In some cases, however, two or more operations would be required to perform the equivalent function of another computer and vice versa. With such being the case, it was decided to retain

the structure of the 1604 machine language, but append to it where required, and excise from it individual operations if needed. The major corollary, then, was that the 1604 machine code structure be used.

A second corollary was derived from the requirement that the system have a reasonably long life: some sort of modularity or expandability feature would be required. The hardware must be designed so newer hardware could easily be added to the system at a later date without undue unbalance or disruption of program continuity. There were several choices of various merits to be examined once the decision to take the modular approach was made. The most important choice was the determination of the size of modules and module inter-relations. A working decision was reached in the initial planning stages to arbitrarily restrict the number of module types to less than eight. This initial constraint was later found to be a reasonable one.

As a guide in determining the module inter-relationships, module interfaces were required to be as simple as possible. A working limit of less than 200 signals per interface was chosen as a further guide.

A third corollary, also derived from the long life requirement, was that the system be highly independent of specific peripheral computers, external data storage mediums, and paper handling devices. This meant that maximum data transfer rates must be much faster than those presently possible, and that a large number of equipments be permitted to process data simultaneously.

Choice of Module Types and Inter-relationships

Considering the constraints of 1604 compatibility, long equipment life, and independence from external devices, the problem of defining the module types was thus well defined. Several approaches were eventually rejected in favor of the one presented here. The largest single constraint proved to be the 1604 compatibility. This constraint was not particularly restrictive for internal processing, but was quite restrictive for input-output. It was determined that abandonment of compatibility in the input-output data would be beneficial in several ways: increased speed of input-output operations; a greater number of operations; ability to add to the system and ease of programming. For these reasons it was planned to make the computer non-compatible in the input-output area. This had a secondary benefit of permitting the choice of module types to be less restricted.

Once the limited compatibility approach was taken, the choice of module type became a relatively clear one. It was decided to make the major module the one which performed all arithmetic and logical operations. This module would execute all 1604 operations with the exception of the input-output operations. The module would also execute new instructions, including the new input-output operations.

With this choice made, the question of storage was considered. It was seen that the need for additional and faster magnetic core storage was one of the principal factors in obsoleting computer systems. To permit expansion of the magnetic core in an economical manner required a design with electrical and physical characteristics that would permit easy integration into the system. Computer systems ranging from 32,768 words to 252,144 words were anticipated in the near future. The larger storage capacity should not suffer in access time brought about by excessive distances from the central processor. To accommodate the range of storage required, and to gain over-all cycle time, a basic magnetic core storage unit of 16,384 51-bit words was chosen. The unit was to have all circuitry required to make it a completely independent unit. Two such units, each with independent access control, were to be packaged together as a single storage module.

The resultant storage module was to be very easy to communicate with. The only information needed by the storage module was a start signal, a specification of read or write operation, and an address. For write activity, data was required. Upon completion of the storage cycle, a completion signal would be given to the computer module.

The interface between the magnetic core storage module and the computer module is thus seen to be a simple one, and quite conventional. From the computer module's point of view, the storage module is an integral part of the computer.

To permit expansion of the storage system, other modules are added to the bus connecting the computer module and the storage module as shown in Figure 1. Up to

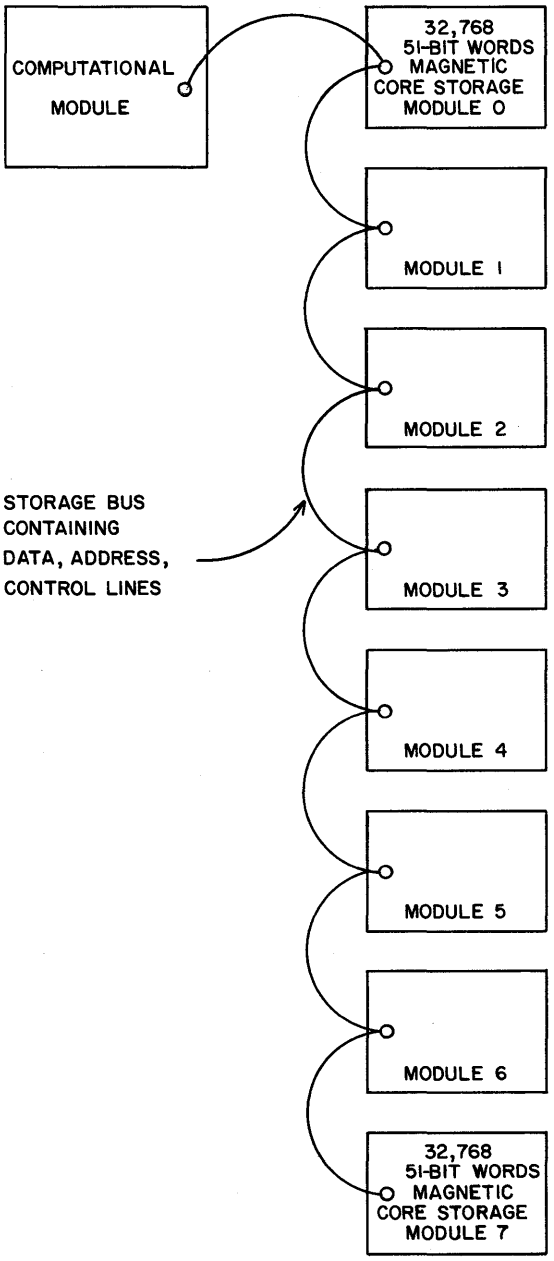


Figure 1. Expansion of storage system.

eight modules of 32,768 words each could be included in the systems as thus far illustrated. Access to the total storage array of 262,144 words was to be accomplished in two ways. The first way employed two 3-bit registers to contain the module number of the operands and the module number of the program. Each operand storage reference would append the 3-bit operand module address to the base 15-bit address (for one of 32,768 words within a module) contained in the instruction as the computer module requested service from the storage module. When the program address counter requested the next instruction, the 3-bit program module address would be used in a similar manner. Since one module could be used for operands and another for programs, a large amount of overlapping of storage references could be obtained. Since each 16,384 unit of storage is independent, the total storage system may be thought of as being composed of 16 modules of 16,384 words each. System programs are written on this basis.

The second way in which the total storage array was to be addressed employed an 18-bit address within the instruction. This feature could be used in the newer, non-1604 compatible instructions inherently, and in the 1604 type instructions by the appendage of an instruction "augment." It was considered undesirable to use the 18-bit address as a mandatory feature in the instructions; relocation of programs from module to module, particularly sizeable subroutines, would require changing the 3-bit module address for each relocated instruction. To avoid this, a 1-bit designator was to be used in each instruction, permitting the 3-bit module address within the instruction to be ignored in favor of the current address contained within the 3-bit operand module register. Thus, all programs of less than 32,768 words in length could be located within any one of eight modules without modification.

With two modules and their interrelationship specified, there remained the choice of the input-output area and the peripheral equipment. It was seen that a serious drawback in present computers was the inability to expand the input-output section independent of the remainder of the computer system. The way chosen to avoid this drawback was to make the linkage between the computer module and the input-output section a weak one. A weak link would assure some control, but would not impose long-range expansion problems arising from space, electrical, and timing considerations. The computer module interface chosen for the as yet unspecified input-output section was: nature of activity, i.e., input, output; establishment of equipment configuration; detection of status; and storage location where additional information concerning data transmissions could be found. As a further means of assuring a weak linkage, all computer attempts to communicate with the input-output section would

result in one of two actions: the desired effect would occur, and the normal program sequence continue; or, the desired effect would not occur because of previous use, illegal code, etc., with the result that the program sequence would jump to an address specified in the attempted input-output instruction. This latter course of action could be used to loop back to the branch point if a "wait until available" action was desired. Thus, the computer merely initiated action with no direct intervention permitted. The computer module was given housekeeping and emergency capability to cancel selected previous activity, or to examine the status of any input-output activity.

The planning of the input-output section of the 3600 disclosed some major problems. With a small number of peripheral units operating in the system, swamping of storage with numerous requests was not likely to occur unless very high speed devices were in use. With numerous such decives, swamping could occur on an averaged basis, and synchronous data could be lost due to queueing delays. As swamping increased, the problem of available storage time allocation arose with the attendant necessity to arbitrate. With a computer sharing the same storage unit, the problem was increased even further. At this point in the planning stage, a serious examination of previous solutions was undertaken. Prior art solved the problem in two similar ways. The first employed an exchange to sort out requests from all devices requesting storage, and to allocate a particular storage reference to one of the requesting devices. Several schemes were employed to arbitrate simultaneous requests and to anticipate needs in the immediate future. The other method employed a single or multiple storage unit scheme which also employed a sort of exchange. The allocation of storage cycles was on a time-slot basis, however, and usually not under program control. Both approaches, and the several variations on them, did not permit true simultaneity of multiple storage cycles. A degree of simultaneity was possible with time slotting, but with much inefficiency resulting from non-optimum usage of all time slots.

The solution to the exchange problem and the problem of storage request queueing was resolved by increasing the interrogatory capability of the storage modules in the system. To each storage module in the system were added four additional access positions with associated registers, gates, and line drivers. Each of these five access positions is truly independent, and each can request a storage reference from any address within it. The storage module honors the first request seen; in case of simultaneity, a very high speed resolver circuit arbitrates. This solution requires each storage module to operate completely asynchronously with respect to any of the five apparatus requesting service. Special high speed resynchronizing circuits, employing tunnel diodes, are used to prevent excessive delay from resynchronization.

In a minimal system, it was decided that the computer module would use one of the five access positions, with the other four allocated to the input-output (see Figure 2). This allocation must be such that neither

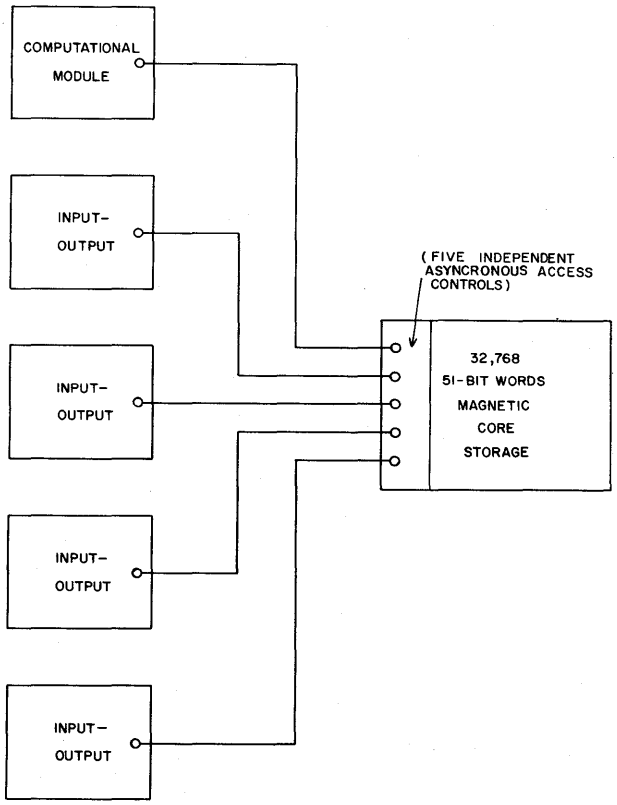


Figure 2. Minimal 3600 system illustrating independent storage access.

swamping nor loss of bursts of data occur. Noting the requirement that a large number of peripheral equipments of diverse nature would be attached to the system, it was decided to organize the input-output into two

types of modules. The two were to be chosen so that economical systems could be designed for high data-rate applications, for many-equipment applications, and for any reasonable combination of these two.

The module chosen as the basic input-output medium was the data channel. The data channel, on one side, would speak the common peripheral equipment language. On the other side, the data channel would communicate with the storage module, and in a loose fashion, with the computer module, both via an intermediary. The data channel, as a minimum, would be required to store basic information about its activity. Information stored would include: equipment on the channel currently in use; starting or current address in storage where the data was situated; number of words in a block; number of blocks in an initiated activity; location of the word in storage where control data is situated for the next block; direction of data flow; status of interrupts and assorted conditions. Such a module would require more than a small amount of hardware, but would require no housekeeping by the storage or computing modules. Certain housekeeping activities, of course, would be required, and these were to be performed by a housekeeping module interposed between the data channel and the storage module. This housekeeping module would also serve as the media through which the data channels received initial instructions from the computer module.

For applications where a multiplicity of medium-speed devices were to be serviced on a regular basis, a number of data channels could be used. Up to eight data channels could be attached to an intermediary housekeeping module, with up to four such housekeeping modules attached to the basic system. Each of these four modules would attach to one of the five access positions of the storage module. Thus, within the maximum limit of four housekeeping modules with up to eight data channels each, any lesser combination would be possible initially. Installation of additional modules, up to the maximum, could be made in the field at any time. It was planned that additions could be made without rewiring, and be tested in a very short period of time.

It was planned that the data channel be able to communicate with a maximum of eight peripheral devices including card readers, card punches, printers, magnetic tape controllers, etc. Since the data channel would contain only one set of control information, only one peripheral device could communicate with it at any instance in time. Figure 3 shows the maximum input-output configuration for a single computer module.

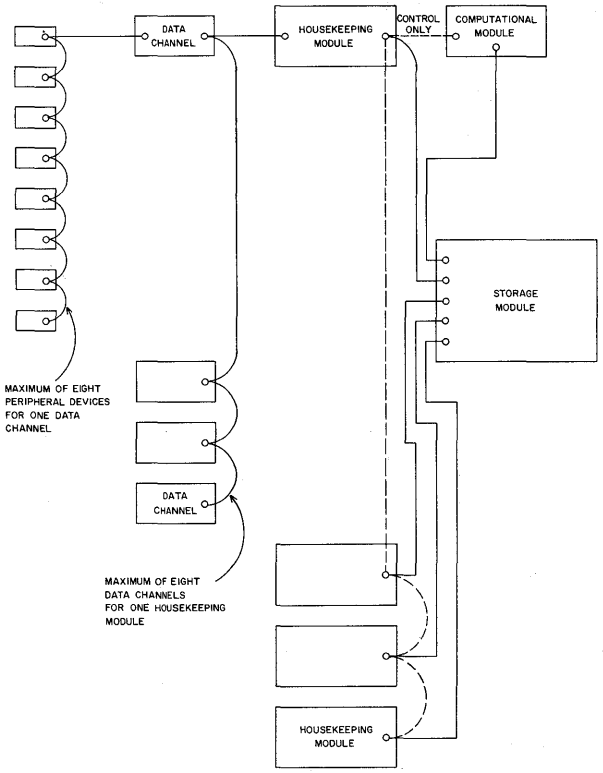


Figure 3. Maximum input-output configuration.

Planning the Computer Module

After the major system framework was relatively well defined, planning effort was concentrated upon the details of the computer module. At this point the constraint of 1604 internal compatibility was reckoned with. The coding structure of the 1604 left little space for additional codes, and the constraint of compatibility precluded the possibility of excising codes or using a somewhat larger instruction word. The solution finally planned made extensive use of full word instructions which were individually quite powerful. Each such instruction had several options, permitting a wide range of new operations.

To permit inclusion of the newer instructions, the three 1604 input-output instructions

were removed and replaced by a single 48-bit instruction group. The resultant two instruction locations, plus two unused ones, were adequate to permit inclusion of several new instructions. Some of the newer instructions were half-word, some full-word; the choice of instruction size was dictated by the nature of the function performed. No instructions required the use of more than one computer word.

During the planning stages of the computer module it was decided to improve the speed of several of the instruction algorithms relative to the 1604. Speed could be achieved principally through sophistication of the logical design. As an example, the timing of the single precision multiply instruction algorithms, exclusive of storage references, could be decreased to approximately a quarter of the 1604 time. This could be roughly multiplied by the hardware speed increase to give an over-all increase in speed of about 10 times or more. Other areas, such as shifting, and floating point addition and subtraction were increased in speed by more powerful logic.

In addition to increasing the relative speed of certain instruction groups, it was considered important to increase the speed of groups of instructions or of larger functional units. For example, examination of a list of words for one that lies between two values takes two or more conventional instructions. With nearly no increase in hardware, a group of two-valued search instructions were included so that a single instruction could perform this function. Another often performed function was that of double precision arithmetic. By utilizing the arithmetic hardware more efficiently, it was possible to include floating add, subtract, multiply and divide commands which have a 10-bit-plus-sign exponent and an 84-bit-plus-sign fraction at fast operating speeds.

Other capabilities were added wherever they could be generalized. Special instructions were avoided wherever possible. It was decided, for example, to permit division of data words into bytes ranging in size from one bit through full word size. It was thought that arbitrary division of the word into fixed byte sizes, such as halves, quarters, or thirds, would unnecessarily limit the use of the computer while not materially reducing the price of the equipment. In addition, fixed partitions would be most inconvenient to change if the nature of the problem being run changed.

A brief description of system characteristics will be set forth here so that the general nature and size of the system may be understood. It should be recognized that the capabilities of the system are dependent upon the number and types of modules of which it is composed. Arithmetic speeds and capabilities, however, are not dependent upon the other modules, and may be considered constant. The computer module is basically a 48-bit, one's complement, parallel binary computer. It has six 15-bit index registers, two principal operational registers, a 48-bit auxiliary sense register, and several registers to process program interrupts. The instruction list contains all 1604 non-input-output operations plus several newer instructions. These include provision for addressing bits and bytes. Transfer instructions may be conditioned upon a single bit in any one of several registers, and the criterion bit may be altered as a result, if desired. On fetch and store operations, provision is made for extracting any portion of the operand and positioning it anywhere in the destination. This feature is extended through the use of search instructions which permit bytes to be searched across words in long arrays. Bytes may be any size from one bit through full word size of 48 bits. Data transfer and editing provisions permit lists to be inverted, expanded, contracted, or added to.

Instructions which permit inter-register transfers and arithmetic and logical manipulations are provided. Provision is made for extending the capabliity of the 59 basic 1604 instructions by such options as double indexing, direct addressability for all 262,144 storage locations, complementation of results, unrounded and/or unnormalized arithmetic. In addition to the two-valued search instructions mentioned, a list processing instruction is included for threaded list processing. Several new jump instructions are added, some permitting braching upon the condition of any bit in any operational register. A full set of input-output instructions are included which permit all operations found in present day computers including such modes as chaining, skipping, and zero insertion on write.

A major system feature is an interrupt system which permits isolation and action

upon any interrupt in the system within 10 microseconds. Instruction times for typical executions are: floating add and subtract, 3.4 μsec; double precision floating add and subtract, 4.3 μsec; single precision multiply, 5.4 μsec; fetch, 1.5 μsec; store, 1.5 μsec; conditional transfer, 1.5 μsec; shift instructions regardless of places shifted, .8 μsec; logical, 1.5 μsec; single precision divide, 13 μsec.

Hardware features include: parity checking on all input-output transmissions and storage references; 1.5 μsec core cycle time; 16 megacycle phase logic, all solid state; modular packaging; built in preventative maintenance features. Cabling over 100 feet is permitted, and all electronic components are accessible while the computer is running.

System Integration

The planned interfaces for the modules were such that a fully assembled system could be treated as a conventional digital computer. For example, a storage module, a computer module, a housekeeping module, and a few data channels are the exact equivalent of present day systems organization (Figure 4). Such a minimal system is programmed by conventional techniques. With

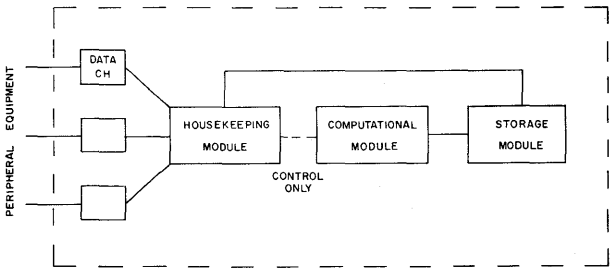


Figure 4. A 3600 simple system.

the availability of many operating programs, the system would be ready for use upon first delivery. Any programs operational or planned for the 1604 could be readily adapted in input-output to the new conventions.

Expanded systems, such as those with increased storage capacity, are programmed in similar manner. Special consideration is given in the address structure of the computer module for the full range of 262,144 words of storage. The over-all system control program would be used to allocate storage modules for optimum usage. Such allocation would consider data rates, sizes of data blocks, and whether maximum usage of each 16,384 storage unit could be obtained by proper program segmenting. Figure 5 shows the maximum system using a single computer module.

Further use of the multiple input-output and storage facility is provided by the ability to preload data for effective scheduling of main computer time. The master control program was considered a more efficient and economical way in which to handle many programs. A true hardware multiprogramming computer was not considered to be feasible in this generation of computers for several reasons. The principal reason was that a multi-programmed computer requires, to be efficient, a large number of instruction registers and manipulative units. Compromises in this direction would be possible, but only at the expense of degraded system performance arising from a high proportion of time and hardware sent in switching and housekeeping.

Multi-Computer Systems

A by-product of the modular approach taken is the ability to construct large, easily controlled, multi-computer systems. Many possible usable combinations of systems exist. A natural system is the one which evolves from a basic one into one which has additional input-output and additional storage. In this type of system, usually no more than two housekeeping modules are used, leaving two unused access positions on the storage modules (Figure 6A). To one of these positions another computer module may be added, and to the other, another housekeeping module may be added (Figure 6B). This latter module would have its own data channels associated with it. This additional complex of computer and input-output would share the storage of the previous system. Additional storage could be added to the common system at this point (Figure 6C), or to either independently (Figure 6D). Since the second system, consisting of computer and input-output, shares only the storage of the initial system, its only real connection with the initial system is via the access position on the storage module. It has no

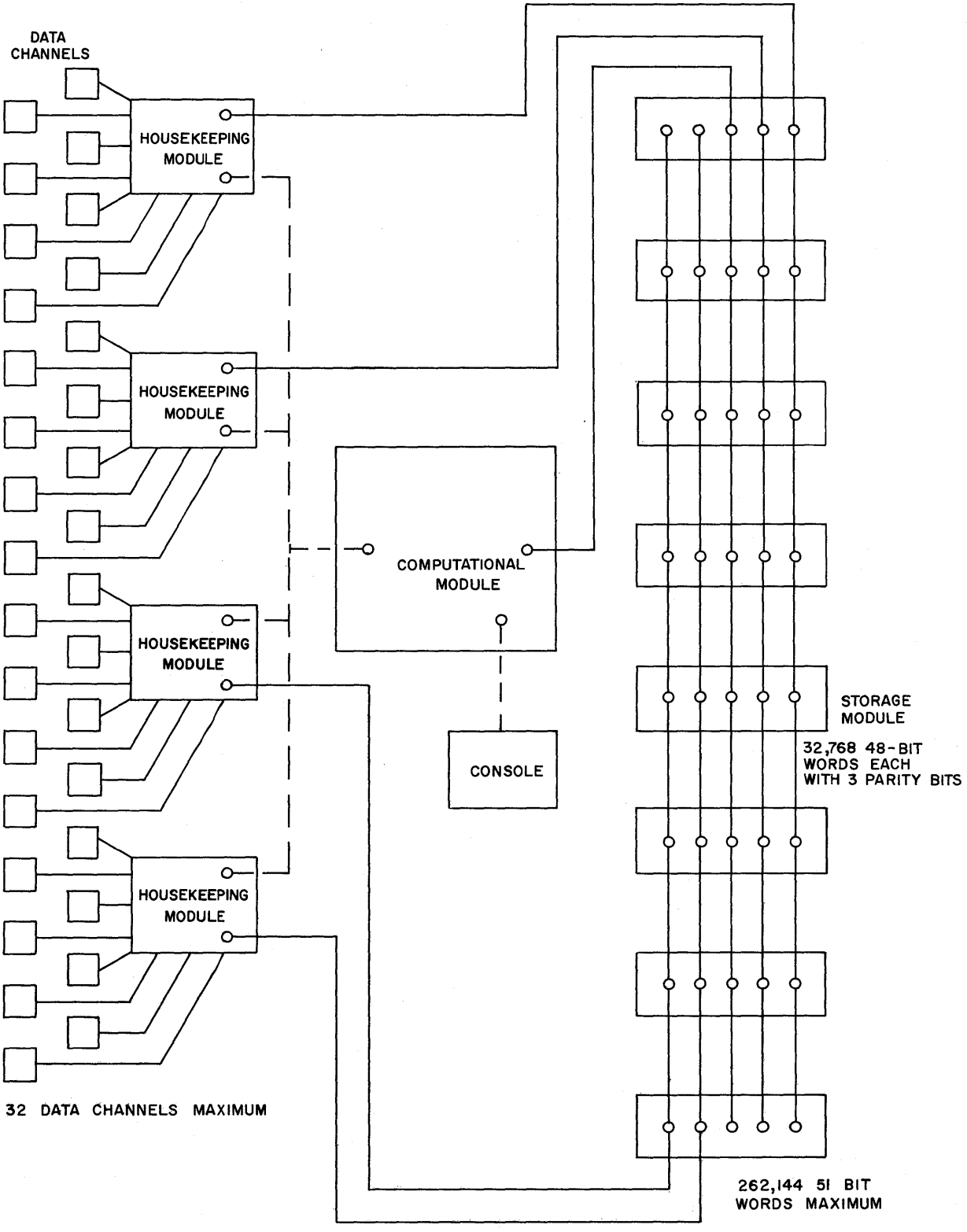
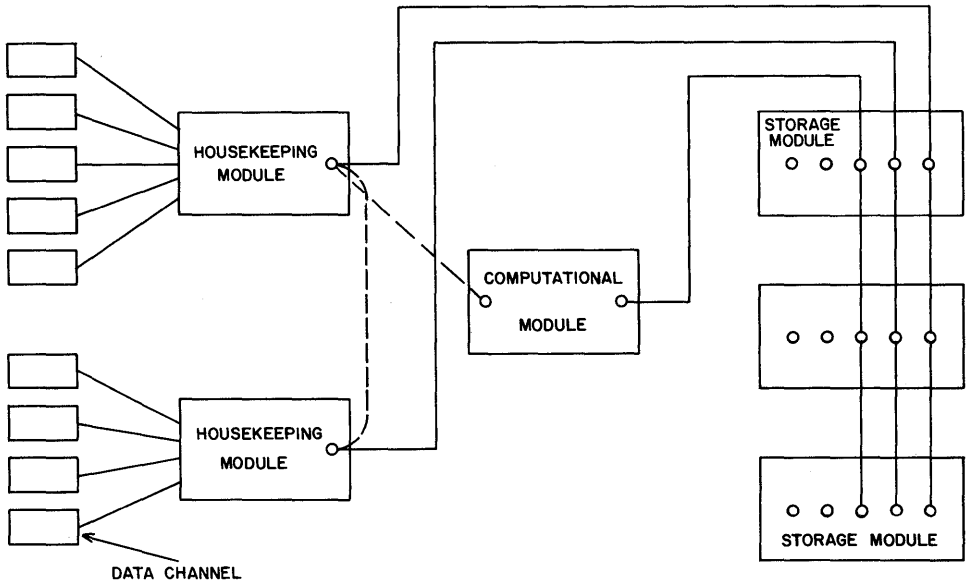


Figure 5. Maximum 3600 simple system.



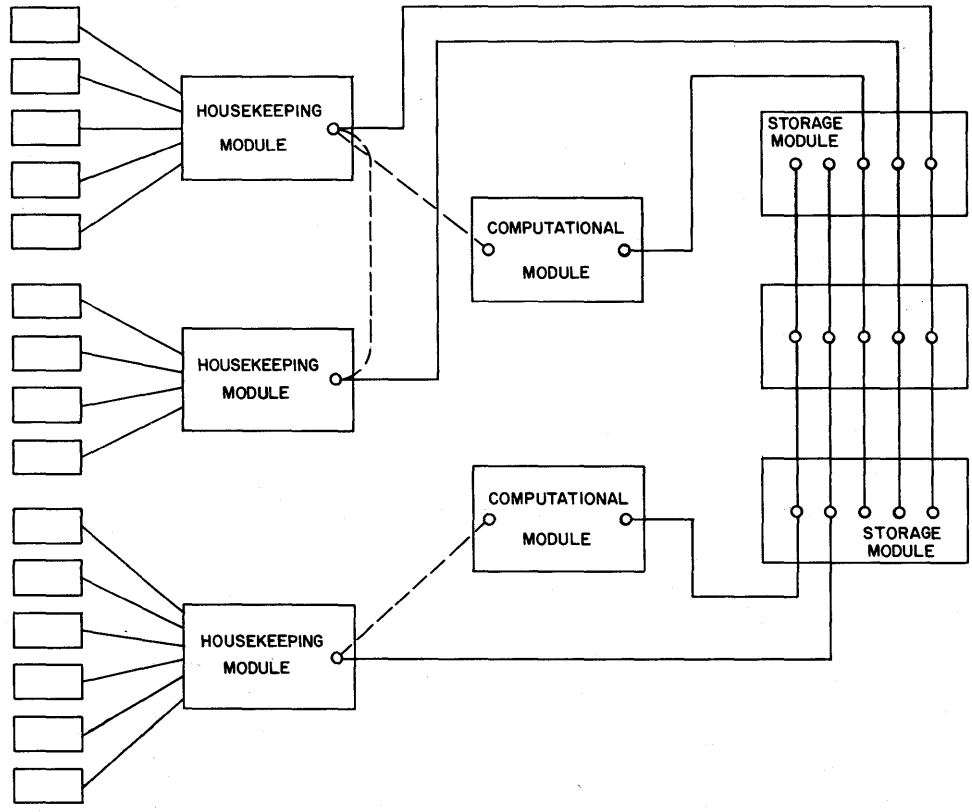


Figure 6A. Partially expanded 3600 system.
Figure 6B. Two 3600 systems sharing commom storage.

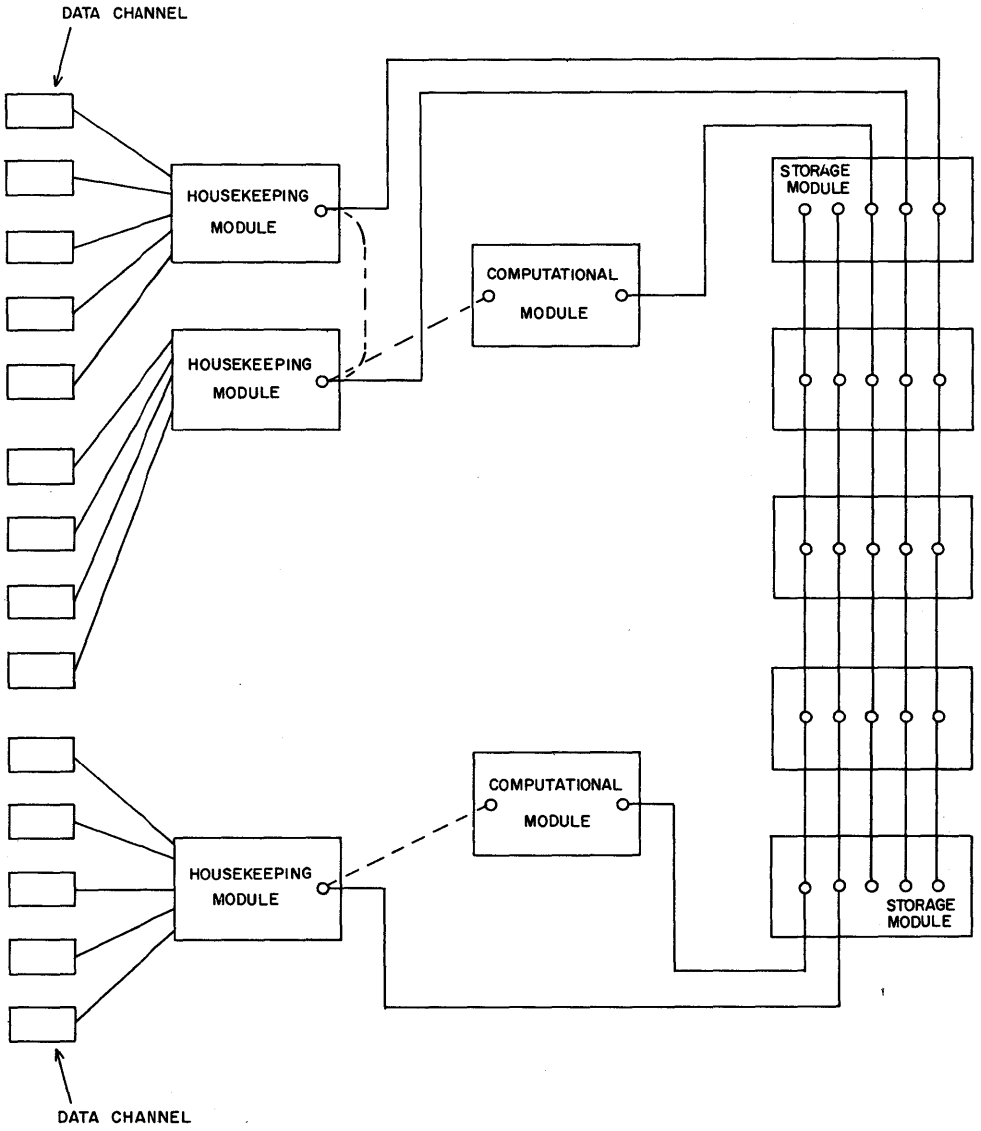


Figure 6C. Two-computer system with additional common storage.

awareness of the existence of the initial computer and input-output complex. The same holds for the initial system. Thus, each system, from an operating point of view, is completely independent of the other. The effects of mutual existence are detectable, however, but these only indirectly. If both systems are using the same storage modules extensively, to the exclusion of their own private modules (if any), over-all operations may be slowed. For such operations, a more reasonable approach would give each computer-input-output complex its own storage module or modules and reserve the common storage area for data of more permanent nature.

In a similar manner, other multi-computer complexes can be constructed within the interconnecting limitations imposed on individual modules.

Real-Time Multiplexed Systems

In multiplexed real-time systems, a high degree of control is required over the entire system. The modular approach planned permits this as an extension of the multi-computer complexes mentioned. In addition to two or more completely independent systems sharing a common storage pool, a system is used whereby the independent systems are also interconnected via their data channels. Each data channel is designed so it may be connected directly, without intervening black boxes or cable adapters, into any

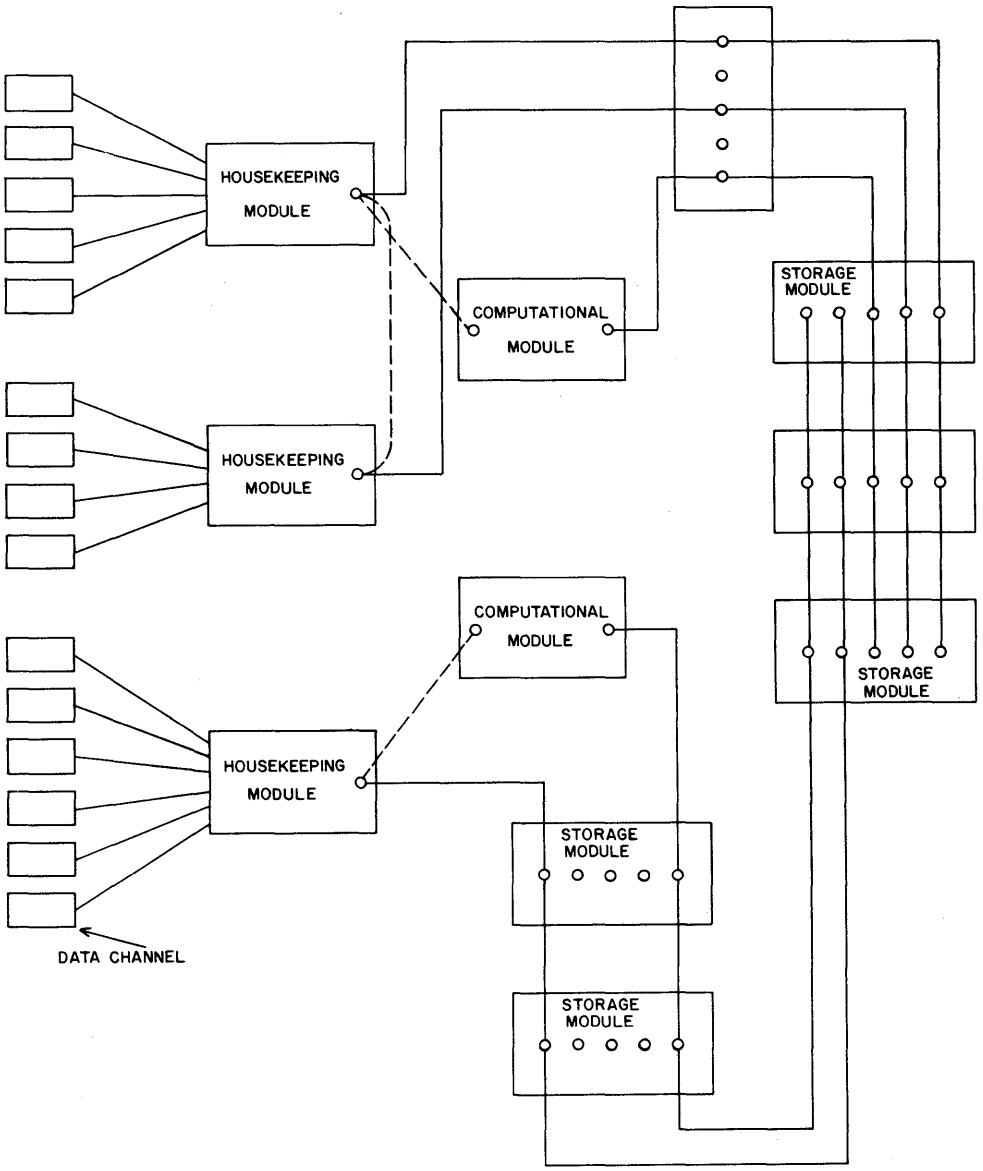


Figure 6D. Two-computer system with private and common storage.

other data channel. These other data channels may be on the same or different systems. The interconnecting linkage supplies data paths and coupling information. In addition, each interconnecting link permits interruption in either direction, and presents major fault information such as parity error in storage or illegal operation code. Thus, one computer may run in real time with data and an initial reference program in a common storage pool. Another computer may be in standby status and processing low priority problems. The on-line computer may interrupt the standby computer at any time and request it to resume the problem. The new on-line computer, depending upon the status of the remainder of the system, may merely serve as a substitute computational unit or as a total computational facility. Either option is under program control. The central controlling program would be stored in a common storage unit with possible interchange with standby units. The computer units are so designed that any one of a group may act as the dominant force with option to transfer the responsibility at any time. Figure 7 shows a typical multiplexed system.

Future Expansion

One of the initial planning goals required long life through addition of newer equipment. With simple interfaces chosen between

modules, it is a relatively simple matter to change the system by the substitution of a new module type for an older one. Newer storage modules or special purpose computing modules may be easily added to the system. With the five access positions on each storage module, new and unrelated module types may be added to the system and controlled via the storage medium.

As an aid in adding new features to the present design without the necessity of disrupting current or future units, a limited micro-programming facility was designed into the computer module. All control elements which the logical designer has at his disposal when he designs instruction algorithms are available for further use. Very high speed transmission lines terminate in

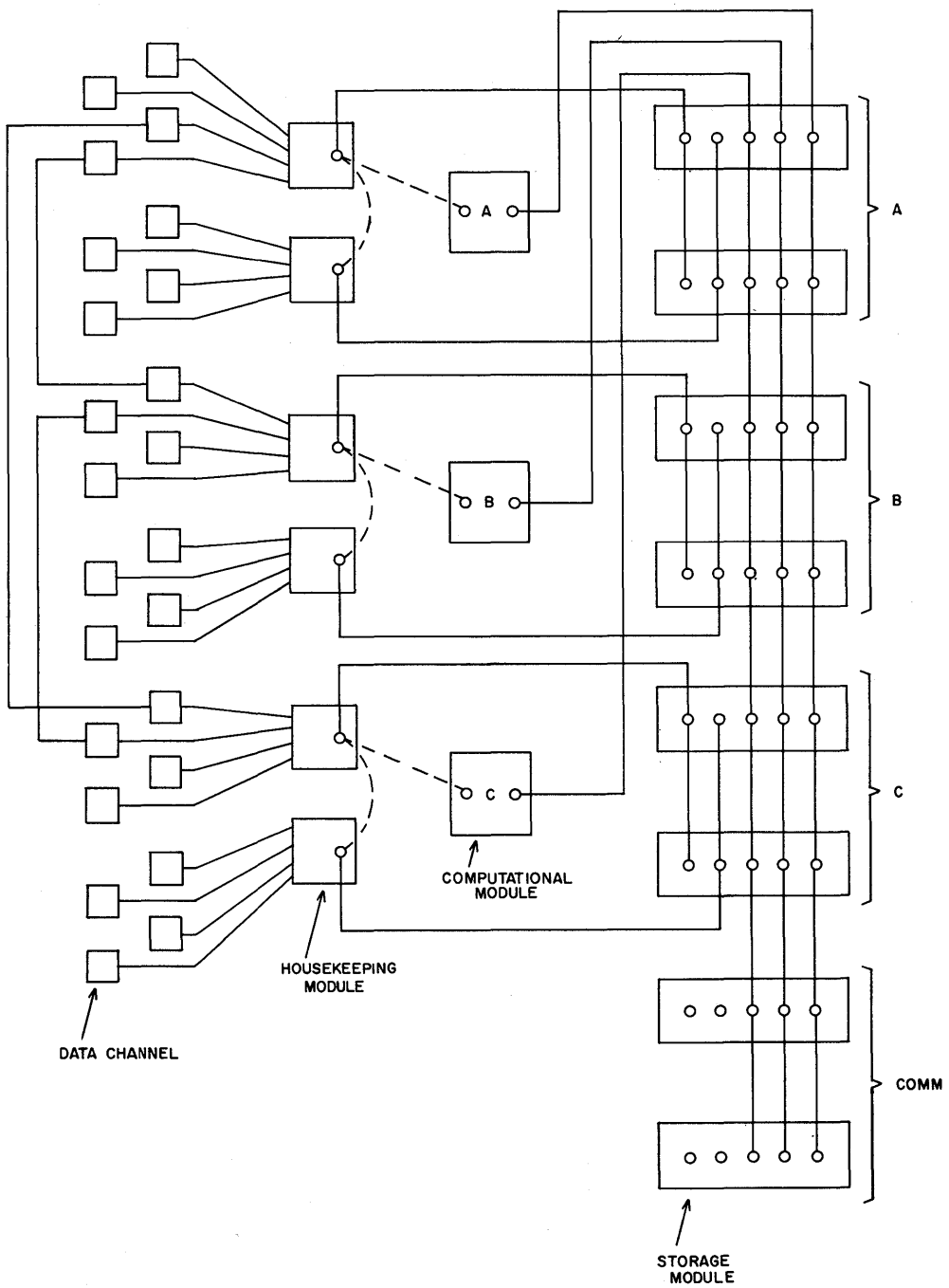


Figure 7. A multiplexed 3600 system.

each of these many control elements. The other ends of the transmission lines are attached to a timing and control device which selectively pulses the lines at the same frequency as the basic computer instructions do. Thus, the cable is a logical extension of the computer module control circuitry. Instruction algorithms performed via this method perform at the same rate of speed as if they were incorporated into the original computer module design itself. Algorithms under active consideration are a square root and a generalized polynomial evaluator.

This facility permits later inclusion of new instructions, hardware subroutines, or modification of existing instructions without modification to the computer system. This facility is used by attaching an external unit to the computer module. The external unit contains timing elements, a function translator, and a small command structure. When connected to the computer module, the external unit is considered an integral part of the computer module. It is referenced by a special instruction which gives total control of the external unit. The unit then performs the specified instruction or subroutine, and then gives control of the computer back to the main program.

The advantages of this facility are many: it is possible to include specialized instructions where very heavy usage is encountered; subroutines such as square root can be constructed avoiding a multiplicity of storage references; and instructions can be given to special equipment attached to the computer module.

SUMMARY

A large-scale computer system is planned in which modular and flexible system design assures a reasonably long machine life. Relationships between modules are shown to be highly important, particularly for future expansion.

D825 - A MULTIPLE-COMPUTER SYSTEM FOR COMMAND & CONTROL

James P. Anderson, Samuel A. Hoffman, Joseph Shifman, and Robert J. Williams
Burroughs Corporation
Burroughs Laboratories
Paoli, Pennsylvania

INTRODUCTION

The D825 Modular Data Processing System is the result of a Burroughs study, initiated several years ago, of the data processing requirements for command and control systems. The D825 has been developed for operation in the military environment. The initial system, . constructed for the Naval Research Laboratory with the designation AN/GYK-3(V), has been completed and tested. This paper reviews the design criteria analysis and design rationale that led to the system structure of the D825. The implementation and operation of the system are also described. Of particular interest is the role that developed for an operating system program in coordinating the system components.

Functional Requirements of Command and Control Data Processing

By "command and control system" is meant a system having the capacity to monitor and direct all aspects of the operation of a large man and machine complex. Until now, the term has been applied exclusively to certain military complexes, but could as well be applied to a fully integrated air traffic control system or even to the operation of a large industrial complex. Operation of command and control systems is characterized by an enormous quantity of diverse but interrelated tasks—generally arising in real time—which are best performed by automatic data-processing equipment, and are most effectively controlled in a fully integrated central data processing facility. The data processing functions alluded to are those typical of data processing, plus special functions associated with servicing displays, responding to manual insertion (through consoles) of data, and dealing with communications facilities. The design implications of these functions will be considered here.

Availability Criteria: The primary requirement of the data-processing facility, above all else, is availability. This requirement, essentially a function of hardware reliability and maintainability, is, to the user, simply the percentage of available, on-line, operation time during a given time period. Every system designer must trade off the costs of designing for reliability against those incurred by unavailability, but in no other application are the costs of unavailability so high as those presented in command and control. Not only is the requirement for hardware reliability greater than that of commercial systems, but downtime for the complete system for preventive maintenance cannot be permitted. Depending upon the application, some greater or lesser portion of the complete system must always be available for primary system functions, and all of the system must be available most of the time.

The data processing facility may also be called upon, except at the most critical times, to take part in exercising and evaluating the operation of some parts of the system, or, in fact, in actual simulation of system functions. During such exercises and simulations, the system must maintain some

(although perhaps partially and temporarily degraded) real-life and real-time capability, and must be able to return quickly to full operation. An implication here, of profound significance in system design, is, again, the requirement that most of the system be always available; there must be no system elements (unsupported by alternates) performing functions so critical that failure at these points could compromise the primary system functions.

Adaptability Criteria: Another requirement, equally difficult to achieve, is that the computer system must be able to analyze the demands being made upon it at any given time, and determine from this analysis the attention and emphasis that should be given to the individual tasks of the problem mix presented. The working configuration of the system must be completely adaptable so as to accommodate the diverse problem mixes, and, moreover, must respond quickly to important changes, such as might be indicated by external alarms or the results of internal computations (exceeding of certain thresholds, for example), or to changes in the hardware configuration resulting from the failure of a system component or from its intentional removal from the system. The system must have the ability to be dynamically and automatically restructured to a working configuration that is responsive to the problem-mix environment.

Expansibility Criteria: The requirement of expansibility is not unique to command and control, but is a desirable feature in any application of data processing equipment. However, the need for expansibility is more acute in command and control because of the dependence of much of the efficacy of the system upon an ability to meet the changing requirements brought on by the very rapidly changing technology of warfare. Further, it must be possible to incorporate new functions in such a way that little or no transitional downtime results in any hardware area.

Expansion should be possible without incurring the costs of providing more capability than is needed at the time. This ability of the system to grow to meet demands should apply not only to the conventionally expansible areas of memory and I/O but to computational devices, as well.

Programming Criteria: Expansion of the data-processing facility should require no reprogramming of old functions, and programs for new functions should be easily incorporated into the overall system. To achieve this capability, programs must be written in a manner which is independent of system configuration or problem mix, and should even be interchangeable between sites performing like tasks in different geographic locales. Finally, because of the large volume of routines that must be written for a command and control system, it should be possible for many different people, in different locations and of different areas of responsibility, to write portions of programs, and for the programs to be subsequently linked together by a suitable operating system.

Concomitant with the latter requirement and with that of configuration-independent programs is the desirability of orienting system design and operation toward the use of a high-level procedure-oriented language. The language should have the features of the usual algorithmic languages for scientific computations, but should also include provisions for maintaining large files of data sets which may, in fact, be ill-structured. It is also desirable that the language reflect the special nature of the application; this is especially true when the language is used to direct the storage and retrieval of data.

Design Rationale for the Data-Processing Facility

The three requirements of availability, adaptability, and expansibility were the motivating considerations in developing the D825 design. In arriving at the final systems design, several existing and proposed schemes for the organization of data processing systems were evaluated in light of the requirements listed above. Many of the same conclusions regarding these and other schemes in the use of computers in command and control were reached independently in a more recent study conducted for the Department of Defense by the Institute for Defense Analysis [1].

The Single-Computer System: The most obvious system scheme, and the least acceptable for command and control, is the single-computer system. This scheme fails to meet the availability requirement simply because the failure of any part—computer, memory, or I/O control—disables the entire system. Such a system was not given serious consideration.

Replicated Single-Computer Systems: A system organization that had been well known at the time these considerations were active involves the duplication (or triplication, etc.) of single-computer systems to obtain availability and greater processing rates. This approach appears initially attractive, inasmuch as programs for the application may be split among two or more independent single-computer systems, using as many such systems as needed to perform all of the required computation. Even the availability requirement seems satisfied, since a redundant system may be kept in idle reserve as backup for the main function.

On closer examination, however, it was perceived that such a system had many disadvantages for command and control applications. Besides requiring considerable human effort to coordinate the operation of the systems, and considerable waste of available machine time, the replicated single computers were found to be ineffective because of the highly interrelated way in which data and programs are frequently used in command and control applications. Further, the steps necessary to have the redundant or backup system take over the main function, should the need arise, would prove too cumbersome, particularly in a time-critical application where constant monitoring of events is required.

Partially Shared Memory Schemes: It was seen that if the replicated computer scheme were to be modified by the use of partially shared memory, some important new capabilities would arise. A partially shared memory can take several forms, but provides principally for some shared storage and some storage privately allotted to individual computers. The shared storage may be of any kind—tapes, discs, or core—but frequently is core. Such a system, by providing a direct path of communication between computers, goes a long way toward satisfying the requirements listed above.

The one advantage to be found in having some memory private to each computer is that of data protection. This advantage vanishes when it is necessary to exchange data between computers, for if a computer failure were to occur, the contents of the private memory of that computer would be lost to the system. Furthermore, many tasks in the command and control application require access to the same data. If, for example, it would be desirable to permit some privately stored data to be made available to the fully shared memory or to some other private memory, considerable time would be lost in transferring the data. It is also clear that a certain amount of utilization efficiency is lost, since some private memory may be unused, while another computer may require more memory than is directly available, and may be forced to transfer other blocks of data back to bulk storage to make way for the necessary storage. It might be added in passing that if private I/O complements are considered, the same questions of decreased overall availability and decreased efficiency arise.

Master/Slave Schemes: Another aspect of the partially shared memory system is that of control. A number of such systems employ a master/slave scheme to achieve control, a technique wherein one computer, designated the master computer, coordinates the work done by the others. The master computer might be of a different character than the others, as in the PILOT system, developed by the National Bureau of Standards [2], or it may be of the same basic design, differing only in its prescribed role, as in the Thompson Ramo Wooldridge TRW400 (AN/FSQ-27) [3]. Such a scheme does recognize the importance, for multicomputer systems, of the problem of coordinating the processing effort; the master computer is an effective means of accomplishing the coordination. However, there are several difficulties in such a design. The loss of the master computer would down the whole system, and the command and control availability requirement could not, consequently, be met. If this weakness is countered by providing the ability for the master control function to be automatically switched to another processor, there still remains an inherent inefficiency. If, for example, the workload of the master computer becomes very large, the master becomes a system bottleneck resulting in inefficient use of all other system elements; and, on the other hand, if the workload fails to keep the master busy, a waste of computing power results. The conclusion is then reached that a master should be established only when needed; this is what has been done in the design of the D825.

The Totally Modular Scheme: As a result of these analyses, certain implications became clear. The availability requirement

dictated a decentralization of the computing function—that is, a multiplicity of computing units. However, the nature of the problem required that data be freely communicable among these several computers. It was decided, therefore, that the memory system would be completely shared by all processors. And, from the point of view of availability and efficiency, it was also seen to be undesirable to associate I/O with a particular computer; the I/O control was, therefore, also decoupled from the computers.

Furthermore, a system with several computers, totally shared memory, and decoupled I/O seemed a perfect structure for satisfying the adaptability requirements of command and control. Such a structure resulted in a flexibility of control which was a fine match for the dynamic, highly variable, processing requirements to be encountered.

The major problem remaining to realize the computational potential represented by such a system was, of course, that of coordinating the many system elements to behave, at any given time, like a system specifically designed to handle the set of tasks with which it was faced at that time. Because of the limitations of previously available equipment, an operating system program had always been identified with the equipment running the program. However, in the proposed design, the entire memory was to be directly accessible to all computer modules, and the operating system could, therefore, be decoupled from any specific computer. The operation of the system could be coordinated by having any processor in the complement run the operating system only as the need arose. It became clear that the master computer had actually become a program stored in totally shared memory, a transformation which was also seen to offer enhanced programming flexibility.

Up to this point, the need for identical computer modules had not been established. The equality of responsibility among computing units, which allowed each computer to perform as the master when running the operating system, led finally to the design specification of identical computer modules. These were freely interconnected to a set of identical memory modules and a set of identical I/O control modules, the latter, in turn, freely interconnected to a highly variable and diverse I/O device complement. It was clear that the complete modularity of system elements was an effective solution to the problem of expansibility, inasmuch as expansion could be accomplished simply by adding modules identical to those in the existing complement. It was also clear that important advantages and economies resulting from the manufacture, maintenance, and spare parts provisioning for identical modules also accrue to such a system. Perhaps the most important result of a totally modular organization is that redundancy of the required complement of any module type, for greater reliability, is easily achieved by incorporating as little as one additional module of that type in the system. Furthermore, the additional module of each type need not be idle; the system may be looked upon as operating with active spares.

Thus, a design structure based upon complete modularity was set. Two items remained to weld the various functional modules into a coordinated system—a device to electronically interconnect the modules, and an operating system program with the effect of a master computer, to coordinate the activities of the modules into fully integrated system operation.

In the D825, these two tasks are carried out by the switching interlock and the Automatic Operating and Scheduling Program (AOSP), respectively. Figure 1 shows how the various functional modules are interconnected via the interlock in a matrix-like fashion.

System Implementation

Most important in the design implementation of the D825 were studies toward practical realization of the switching interlock and the AOSP. The computer, memory, and I/O control modules permitted more conventional solutions, but were each to incorporate some unusual features, while many of the I/O devices were selected from existing equipment. With the exception of the latter, all of these elements are discussed here briefly. (A summary of D825 characteristics and specifications is included at the end of the paper.)

Switching Interlock: Having determined that only a completely shared memory system would be adequate, it was necessary to find some way to permit access to any memory by any processor, and, in fact, to permit

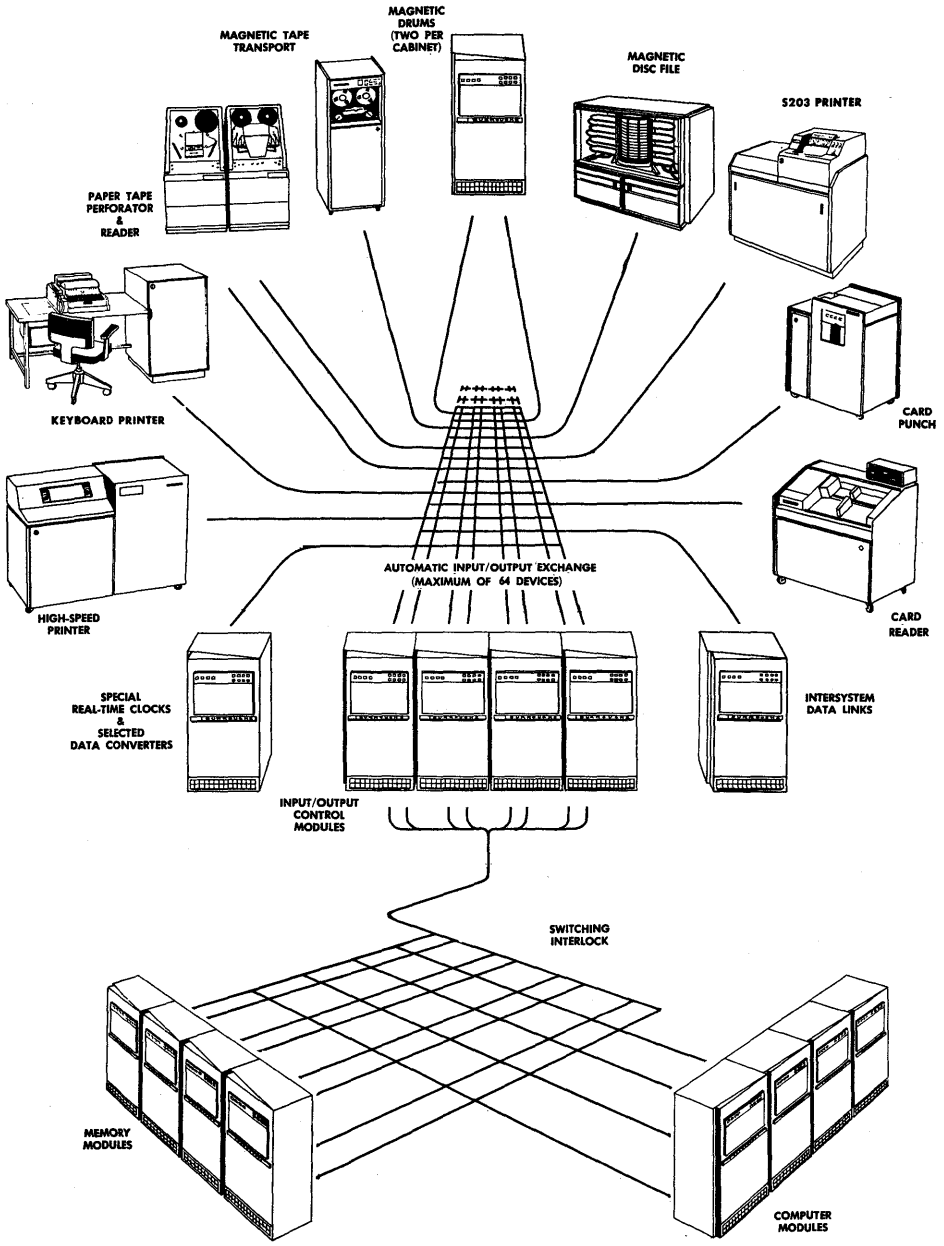


Figure 1. System Organization, Burroughs D825 Modular Data Processing System.

sharing of a memory module by two or more processors or I/O control modules.

A function distributed physically through all of the modules of a D825 system, but which has been designated in aggregate the switching interlock, effects electronically each of the many brief interconnections by which all information is transferred among computer, memory, and I/O control modules. In addition to the electronic switching function, the switching interlock has the ability to detect and resolve conflicts such as occur when two or more computer modules attempt access to the same memory module.

The switching interlock consists functionally of a crosspoint switch matrix which effects the actual switching of bus interconnections, and a bus allocator which resolves all time conflicts resulting from simultaneous requests for access to the same bus or system module. Conflicting requests are queued up according to the priority assigned

to the requestors. Priorities are pre-emptive in that the appearance of a higher priority request will cause service of that request before service of a lower priority request already in the queue. Analyses of queueing probabilities have shown that queues longer than one are extremely unlikely.

The priority scheduling function is performed by the bus allocator, essentially a set of logical matrices. The conflict matrix detects the presence of conflicts in requests for interconnection. The priority matrix resolves the priority of each request. The logical product of the states of the conflict and priority matrices determines the state of the queue matrix, which in turn governs the setting of the crosspoint switch, unless the requested module is busy.

The AOSP: An Operating System Program: The AOSP is an operating system program stored in totally shared memory and therefore available to any computer. The program is run only as needed to exert control over the system. The AOSP includes its own executive routine, an operating system for an operating system, as it were, calling out additional routines, as required. The configuration of the AOSP thus permits variation from application to application, both in sequence and quantity of available routines and in disposition of AOSP storage.

The AOSP operates effectively on two levels, one for system control, the other for task processing.

The system control function embodies all that is necessary to call system programs and associated data from some location in the I/O complement, and to ready the programs for execution by finding and allocating space in memory, and initiating the processing. Most of the system control function (as well as the task processing function) consists of elaborate bookkeeping for: programs being run, programs that are active (that is, occupy memory space), I/O commands being executed, other I/O commands waiting, external data blocks to be received and decoded, and activation of the appropriate programs to handle such external data. It would be inappropriate here to discuss the myriad details of the AOSP; some idea of its scope, however, can be obtained from the following list of some of its major functions:

1. configuration determination,
2. memory allocation,
3. scheduling,
4. program readying and end-of-job cleanup,
5. reporting and logging,
6. diagnostics and confidence checking,
7. external interrupt processing.

The task processing function of the AOSP is to execute all program I/O requests in order to centralize scheduling problems and to protect the system from the possibility of data destruction by ill-structured or conflicting programs.

AOSP Response to Interrupts: The AOSP function depends heavily upon the comprehensive set of interrupts incorporated in the D825. All interrupt conditions are transmitted to all computer modules in the system, and each computer module can respond to all interrupt conditions. However, to make it possible to distribute the responsibility for various interrupt conditions, both system and local, each computer module has an interrupt mask register that controls the setting of individual bits of the interrupt register. The occurrence of any interrupt causes one of the system computer modules to leave the program it has been running and branch to the suitable AOSP entry, entering a control mode as it branches. The control mode differs from the normal mode of operation in that it locks out the response to some low-priority interrupts (although recording them) and enables the execution of some additional instructions reserved for AOSP use (such as setting an interrupt mask register or memory protection registers, or transmitting an I/O instruction to an I/O control module).

In responding to an interrupt, the AOSP transfers control to the appropriate routine handling the condition designated by the interrupt. When the interrupt condition has been satisfied, control is returned to the original object program. Interrupts caused by normal operating conditions include:

1. 16 different types of external requests,
2. completion of an I/O operation,
3. real-time clock overflow,
4. array data absent,
5. computer-to-computer interrupts,
6. control mode entry (normal mode halt).

Interrupts related to abnormalities of either program or equipment include:

1. attempt by program to write out of bounds,
2. arithmetic overflow,
3. illegal instruction,

4. inability to access memory, or an internal parity error; parity error on an I/O operation causes termination of that operation with suitable indication to the AOSP,
5. primary power failure,
6. automatic restart after primary power failure,
7. I/O termination other than normal completion.

While the reasons for including most of the interrupts listed above are evident, a word of comment on some of them is in order.

The array-data-absent interrupt is initiated when a reference is made to data that is not present in the memory. Since all array references such as A[k] are made relative to the base (location of the first element) of the array, it is necessary to obtain this address and to index it by the value k. When the base of array A is fetched, hardware sensing of a presence bit either allows the operation to continue, or initiates the array-data-absent interrupt. In this way, keeping track of data in use by interacting programs can be simplified, as may the storage allocation problem.

The primary power failure interrupt is highest priority, and always pre-emptive. This interrupt causes all computer and I/O control modules to terminate operations, and to store all volatile information either in memory modules or in magnetic thin-film registers. (The latter are integral elements of computer modules.) This interrupt protects the system from transient power failure, and is initiated when the primary power source voltage drops below a predetermined limit.

The automatic restart after primary power failure interrupt is provided so that the previous state of the system can be reconstructed.

A description of how an external interrupt is handled might clarify the general interrupt procedure. Upon the presence of an external interrupt, the computer which has been assigned responsibility to handle such interrupts automatically stores the contents of those registers (such as the program counter) necessary to subsequently reconstitute its state, enters the control mode, and goes to a standard (hardware-determined) location where a branch to the external request routine is located. This routine has the responsibility of determining which external request line requires servicing, and, after consulting a table of external devices (teletype buffers, console keyboards, displays, etc.) associated with the interrupt lines, the computer constructs and transmits an input instruction to the requesting device for an initial message. The computer then makes an entry in the table of the I/O complete program (the program that handles I/O complete interrupts) to activate the appropriate responding routine when the message is read in. A check is then made for the occurrence of additional external requests. Finally, the computer restores the saved register contents and returns in normal mode to the interrupted program.

AOSP Control of I/O Activity: As mentioned above, control of all I/O activity is also within the province of the AOSP. Records are kept on the condition and availability of each I/O device. The locations of all files within the computer system, whether on magnetic tape, drum, disc file, card, or represented as external inputs, are also recorded. A request for input by file name is evaluated, and, if the device associated with this name is readily available, the action is initiated. If for any reason the request must be deferred, it is placed in a program queue to await conditions which permit its initiation. Typical conditions which would cause deferral of an I/O operation include:

1. no available I/O control module or channel,
2. the device in which the file is located is presently in use,
3. the file does not exist in the system.

In the latter case, typically, a message would be typed out on the supervisory printer, asking for the missing file.

The I/O complete interrupt signals the completion of each I/O operation. Along with this interrupt, an I/O result descriptor is deposited in an AOSP table. The status relayed in this descriptor indicates whether or not the operation was successful. If not successful, what went wrong (such as a parity error, or tape break, card jams, etc.) is indicated so that the AOSP may initiate the appropriate action. If the operation was successful, any waiting I/O operations which can now proceed are initiated.

AOSP Control of Program Scheduling: Scheduling in the D825 relies upon a job table maintained by the AOSP. Each entry is identified with a name, priority, precedence requirements, and equipment requirements. Priority may be dynamic, depending upon

time, external requests, other programs, or a function of many variable conditions. Each time the AOSP is called upon to select a program to be run, whether as a result of the completion of a program or of some other interrupt condition, the job table is evaluated. In a real-time system, situations occur wherein there is no system program to be run, and machine time is available for other uses. This time could be used for auxiliary functions, such as confidence routines.

The AOSP provides the capability for program segmentation at the discretion of the programmer. Control macros embedded in the program code inform the AOSP that parallel processing with two or more computers is possible at a given point. In addition, the programmer must specify where the branches indicated in this manner will join following the parallel processing.

Computer Module: The computer modules of the D825 system are identical, general-purpose, arithmetic and control units. In determining the internal structure of the computer modules, two considerations were uppermost. First, all programs and data had to be arbitrarily relocatable to simplify the storage allocation function of the AOSP; secondly, programs would not be modified during execution. The latter consideration was necessary to minimize the amount of work required to pre-empt a program, since all that would have to be saved to reinstate the interrupted program at a later time would be the data for that program and the register contents of the computer module running the program at the time it was dumped.

The D825 computer modules employ a variable-length instruction format made up of quarter-word syllables. Zero-, one-, two-, or three-address syllables, as required, can be associated with each basic command syllable. An implicitly addressed accumulator stack is used in conjunction with the arithmetic unit. Indexing of all addresses in a command is provided, as well as arbitrarily deep indirect addressing for data.

Each computer module includes a 128-position thin-film memory used for the stack, and also for many of the registers of the machine, such as the program base register, data base register, the index registers, limit registers, and the like.

The instruction complement of the D825 includes the usual fixed-point, floating-point, logical, and partial-field commands found in any reasonably large scientific data processor.

Memory Module: The memory modules consist of independent units storing 4096 words, each of 48 bits. Each unit has an individual power supply and all of the necessary electronics to control the reading, writing, and transmission of data. The size of the memory modules was established as a compromise between a module size small enough to minimize conflicts wherein two or more computer or I/O modules attempt access to the same memory module, and a size large enough to keep the cost of duplicated power supplies and addressing logic within bounds. It might be noted that for a larger modular processor system, these trade-offs might indicate that memory modules of 8192 words would be more suitable. Modules larger than this— of 16,384 or 32,768 words, for example—would make construction of relatively small equipment complements meeting the requirements set forth above quite difficult. The cost of smaller units of memory is offset by the lessening of catastrophe in the event of failure of a module.

I/O Control Module: The I/O control module executes I/O operations defined and initiated by computer module action. In keeping with the system objectives, I/O control modules are not assigned to any particular computer module, but rather are treated in much the same way as memory modules, with automatic resolution of conflicting attempted accesses via the switching interlock function. Once an I/O operation is initiated, it proceeds independently until completion.

I/O action is initiated by the execution of a transmit I/O instruction in one of the computer modules, which delivers an I/O descriptor word from the addressed memory location to an inactive I/O control module. The I/O descriptor is an instruction to the I/O control module that selects the device, determines the direction of data flow, the address of the first word, and the number of words to be transferred.

Interposed between the I/O control modules and the physical external devices is another crossbar switch designated the I/O exchange. This automatic exchange, similar in function to the switching interlock, permits two-way data flow between any I/O control module and any I/O device in the system. It further enhances the flexibility of the

system by providing as many possible external data transfer paths as there are I/O control modules.

Equipment Complements: A D825 system can be assembled (or expanded) by selection of appropriate modules in any combination of: one to four computer modules, one to 16 memory modules, one to ten I/O control modules, one or two I/O exchanges, and one to 64 I/O devices per I/O exchange in any combination selected from: operating (or system status) consoles, magnetic tape transports, magnetic drums, magnetic disc files, card punches and readers, paper tape perforators and readers, supervisory printers, high-speed line printers, selected data converters, special real-time clocks, and intersystem data links.

Figure 2 is a photograph of some of the hardware of a completed D825 system. The equipment complement of this system includes two computer modules, four memory modules (two per cabinet), two I/O control modules (two per cabinet), one status display console, two magnetic tape units, two magnetic drums, a card reader, a card punch, a supervisory printer, and an electrostatic line printer.

D825 characteristics are summarized in Table 1.

SUMMARY AND CONCLUSION

It is the belief of the authors that modular systems (in the sense discussed above) are a natural solution to the problem of obtaining greater computational capacity—more natural than simply to build larger and faster machines. More specifically, the organizational structure of the D825 has been shown to be a suitable basis for the data processing

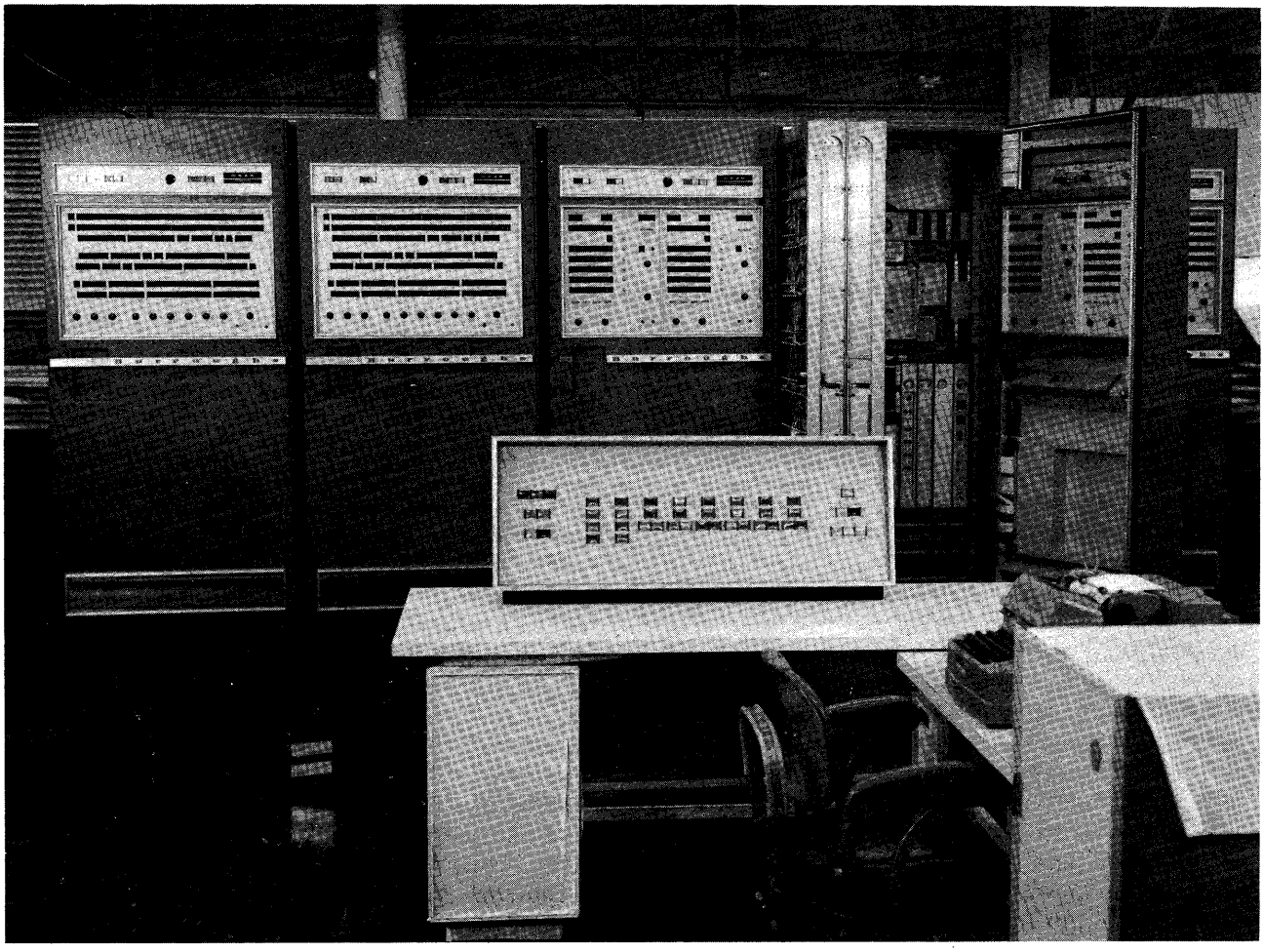


Figure 2. Typical D825 Equipment Array.

facility for command and control. Although the investigation leading toward this structure proceeded as an attack upon a number of diverse problems, it has become evident that the requirements peculiar to this area of application are, in effect, aspects of a single characteristic, which might be called structural freedom. Furthermore, it is now clear that the most unique characteristic of the structure realized--integrated operation of freely intercommunicating, totally modular elements--provides the means for achieving structural freedom.

For example, one requirement is that some specified minimum of data processing capability be always available, or that, under any conditions of system degradation due to failure or maintenance, the equipment remaining on line be sufficient to perform primary system functions. In the D825, module failure results in a reduction of the on-line equipment configuration but permits normal operation to continue, perhaps at a reduced rate. The individual modules are designed to be highly reliable and maintainable, but system availability is not derived solely from this source, as is necessarily the case with more conventional systems. The modular configuration permits operation, in effect, with active spares, eliminating the need for total redundancy.

A second requirement is that the working configuration of the system at a given moment be instantly reconstructable to new forms more suited to a dynamically and unpredictably changing work load. In the D825, all communication routes are public, all modules are functionally decoupled, all assignments are scheduled dynamically, and assignment patterns are totally fluid. The system of interrupts and priorities controlled by the AOSP and the switching interlock permits instant adaptation to any work load, without destruction of interrupted programs.

The requirement for expansibility calls simply for adaptation on a greater time scale. Since all D825 modules are functionally decoupled, modules of any types may be added to the system simply by plugging into the switching interlock or the I/O exchange. Expansion in all functional areas may be pursued far beyond that possible with conventional systems.

It is clear, however, that the D825 system would have fallen far short of the goals set for it if only the hardware had been considered. The AOSP is as much a part of the D825 system structure as is the actual hardware. The concept of a "floating" AOSP as the force that molds the constituent modules of an equipment complement into a system is an important notion having an effect beyond the implementation of the D825. One interesting by-product of the design effort for the D825 has, in fact, been a change of perspective; it has become abundantly clear that computers do not run programs, but that programs control computers.

ACKNOWLEDGMENTS

The authors wish to acknowledge the outstanding efforts of their many colleagues at Burroughs Laboratories who have contributed so well and in so many ways to all stages of D825 design, development, fabrication, and programming. It would be impossible to cite all of these efforts. The authors also wish to acknowledge the contributions of Mr. William R. Slack and Mr. William W. Carver, also of Burroughs Laboratories. Mr. Slack has been closely associated with the D825 from its original conception to its implementation in hardware and software. Mr. Carver made important contributions to the writing and editing of this paper.

REFERENCES

1. Marlin G. Kroger et al, "Computers in Command and Control" (TR61-12, prepared for DOD:ARPA by Digital Computer Application Study, Institute for Defense Analyses, Research and Engineering Support Division), November 1961.
2. A. L. Leiner, W. A. Notz, J. L. Smith, and A. Weinberger, "Organizing a Network of Computers to Meet Deadlines," Proceedings, Eastern Joint Computer Conference, December 1957.
3. R. E. Porter, "The RW-400--A New Polymorphic Data System," Datamation, Vol. 6, No. 1, January/February 1960, pp. 8-14.

Table 1. Specifications, D825 Modular Data Processing System

Computer module:	4, maximum complement
Computer module, type:	Digital, binary, parallel, solid-state
Word length:	48 bits including sign (8 characters, 6 bits each) plus parity
Index registers: (in each computer module)	15
Magnetic thin-film registers: (in each computer module)	128 words, 16 bits per word, 0.33-μsec read/write cycle time
Real-time clock: (in each computer module)	10 msec resolution
Binary add:	1.67 μsec (average)
Binary multiply:	36.0 μsec (average)
Floating-point add:	7.0 μsec (average)
Floating-point multiply:	34.0 μsec (average)
Logical AND:	0.33 μsec
Memory type:	Homogeneous, modular, random-access, linear-select, ferrite-core
Memory capacity:	65,536 words (16 modules maximum, 4096 words each)
I/O exchanges per system:	1 or 2
I/O control modules:	10 per exchange, maximum
I/O devices:	64 per exchange, maximum
Access to I/O devices:	All I/O devices available to every I/O control module in exchange
Transfer rate per I/O exchange:	2,000,000 characters per second
I/O device complement:	All standard I/O types, including 67 kc magnetic tapes, magnetic drums and discs, card and paper tape punches and readers, character and line printers, communications and display equipment

THE SOLOMON COMPUTER*

Daniel L. Slotnick, W. Carl Borck, and Robert C. McReynolds
Air Arm Division
Westinghouse Electric Corporation
Baltimore, Maryland

INTRODUCTION AND SUMMARY

The SOLOMON (Simultaneous Operation Linked Ordinal MOdular Network), a parallel network computer, is a new system involving the interconnections and programming, under the supervision of a central control unit, of many identical processing elements (as few or as many as a given problem requires), in an arrangement that can simulate directly the problem being solved.

The parallel network computer shows great promise in aiding progress in certain critically important areas limited by the capabilities of current computing systems. Many of these technical areas possess the common mathematical denominator of involving calculations with a matrix or mesh of numerical values, or more generally involving operations with sets of variables which permit simultaneous independent operation on each individual variable within the set. This group is typified by the solution of linear systems, the calculation of inverses and eigenvalues of matrices, correlation and autocorrelation, and numerical solution of systems of ordinary and partial differential equations. Such calculations are encountered throughout the entire spectrum of problems in data reduction, communication, character recognition, optimization, guidance and control, orbit calculations, hydrodynamics, heat flow, diffusion, radar data processing, and numerical weather forecasting.

An example of the type of problem permitting the use of the parallelism is the numerical solution of partial differential equations. Assuming the value of a function, u, is known on the boundary, Γ, of a region, the solution of the Laplace equation† can be calculated at each mesh point, x, y in the region as illustrated in Figure 1.

Since the iteration formula is identical for each mesh point in the region, the arithmetic capability provided by a processing element corresponding to each point will enable one calculation of the equation; i.e., a single program execution, to improve the approximation at each of the mesh points simultaneously.

Figure 2 illustrates a basic array of processing elements. Each of these elements possesses 4096 bits of core storage, and the arithmetic capabilities to perform serial-by-bit arithmetic and logic. An additional capability possessed by each processing element is that of communication with other processing elements. Processing element E can

*The applied research reported in this document has been made possible through support and sponsorship extended by the U.S. Air Force Rome Air Development Center and the U.S. Army Signal Research and Development Laboratory under Contract Number AF30(602)2724: Task 730J. It is published for technical information only, and does not necessarily represent recommendations or conclusions of the sponsoring agency.

†Jeeves, T. A., et al; "On the Use of the SOLOMON Parallel-Processing Computer." Proceedings of the Eastern Joint Computer Conference, Philadelphia, December 1962.

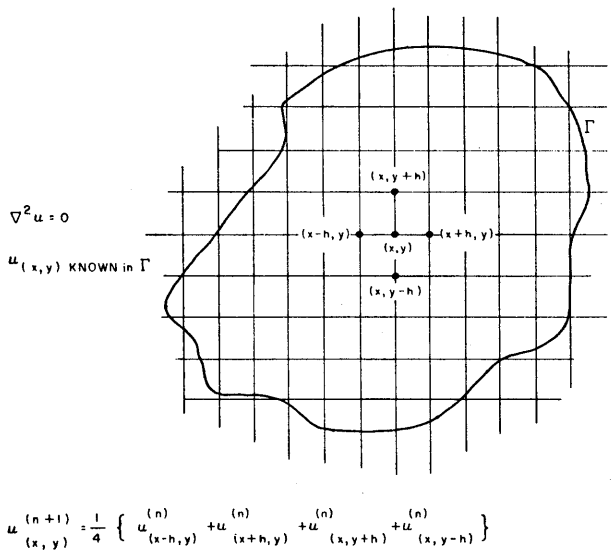


Figure 1. Iterative Solution of Laplace's Equation

transmit and receive data serially from its four nearest neighbors: the processing elements immediately to right, A; left, C; above, B; and below, D.

A fifth source of input data is available to the processing element matrix through the "broadcast input" option. This option utilizes a register in the central control to supply constants when needed by an arbitrary number of the processing elements during the same operation cycle. This constant is treated as a normal operand by the processing elements and results in the central control unit becoming a "fifth" nearest neighbor to all processing elements.

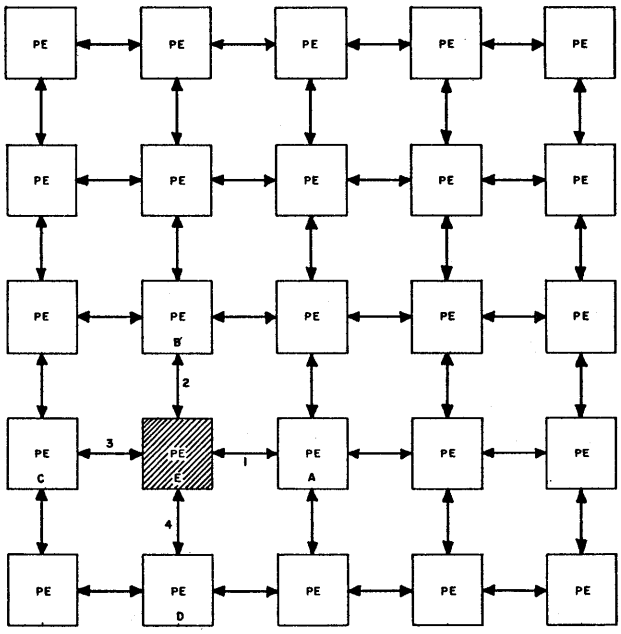


Figure 2. Basic Array of Processing Elements

The processing element array is the core of the system concept; however, it is the method of controlling the array which turns this concept into a viable machine design. This method of control is the simplest possible in that the processing elements contain a minimum of control logic - the "multimodal" logic described below.

Figure 3 illustrates how the processing element array, a 32 × 32 network, is controlled by a single central control unit. Multimodal control permits the processing elements to alter control signals to the processing element network according to values of internal data. They are individually permitted to execute or ignore these central control signals.

Basically, the central control unit contains program storage (large capacity random-access memory), has the means to retrieve and interpret the stored instructions, and has the capability, subject to multimodal logic, to cause execution of these instructions within the array. Thus, at any given instant, each processing element in the system is capable of performing the same operation on the operands stored in the same memory location of each processing element. These operands, however, may all be different. The flow of control information from the control unit to the processing elements is indicated in Figure 3 by light lines. An instruction is retrieved from the program storage and transmitted to a register in central control. Within central control, the instruction is interpreted and the information contained is translated into a sequence of signals and transmitted from central control to the processing elements. Since this information must be provided to 1024 processing elements, it is necessary to branch this information and provide the necessary amplification and power. This is accomplished by transmission through branching levels, which provide the necessary power for transmission.

As described above, each processing element in the network possesses the capability of communicating with its four adjacent elements. The "edge" processing elements,

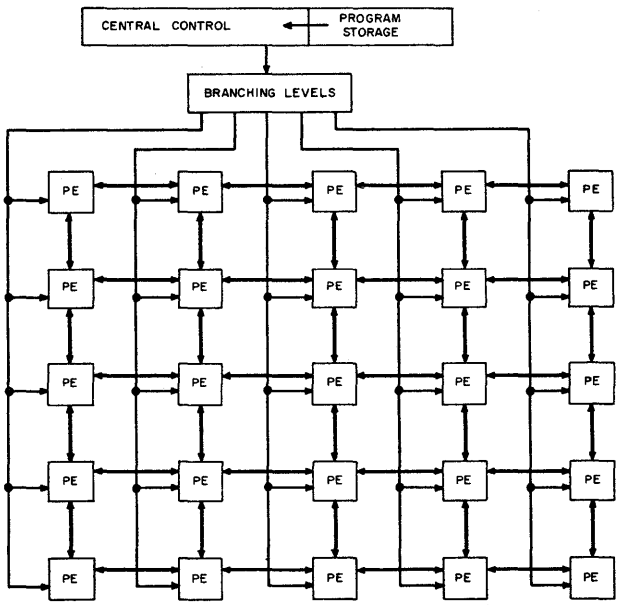


Figure 3. PE Array Under Central Control

however, do not possess a full complement of neighbors. The resulting free connections are used for input-output application. This makes possible very high data exchange rates between the central computer and external devices through the input-output subsystem. These rates could be still further increased by providing longer "edges"; i.e., by the use of a nonsquare network array.

Two input-output exchange systems are used by the input output equipment. The primary exchange system is a high speed system operating at a data rate near that of the processing element network. This system consists of magnetic tapes and rotating magnetic memories and serves the network with data storage during large net problems.

The secondary exchange system provides the user with communication with the primary exchange system through conventional high speed printers, and tape transports. The data at the output of this system is compatible with most conventional devices.

The Processing Element

The processing element (PE) logic, illustrated in Figure 4, basically consists of two parts: the processing element memory, and the arithmetic and multimodal control logic.

The multimodal control within each processing element provides the capability for individual elements to alter the program flow as a function of the data which it is currently processing. This capability permits the processing element to classify data and make judgments on the course of programming which it should follow. Whenever individual elements are in a different mode of operation than specified by central control, they will not execute the specified command.

During each arithmetic operation, one word will be read serially from each of the two memory frames associated with a unique processing element. The operand in frame one will be transmitted by central control command, either to the internal adder or to that of one of the four adjacent elements which are its nearest neighbors in the network array. The five gates labeled A in Figure 4 control the routing of information from frame one. Since only one of these may be activated during a single operation, a word in frame one can be entered in the operation select logic of only one of the five processing elements. The frame-two operand can be routed only into the unit's full adder.

Each PE in the system will communicate with a corresponding unit, thereby producing a flow of information between processing elements during network operations.

Word addressing of the memory is performed by the matrix switches in the central control unit. These switches convert the address from the binary form of the instruction to the one-of-n form required for addressing the memory frame. Provision is made for special addressing of specific memory locations for temporary storage of multiplier and quotient during multiplication and division. Successive bits are shifted into the PE logic by two digit counters in central control.

Three different types of storage are permitted: (1) the sum can be routed into frame one while the original word in frame two is rewritten; (2) the sum can be routed into frame two while the word in frame one is rewritten; and (3) information can be interchanged between frames. Note that in the first two operations, the word which was located in the memory frame into which the sum is routed is destroyed. No information is altered during the third type operation.

Multimodal Operation: Multimodal operation gives the processing element the additional capability for altering program flow and tagging information on the basis of internal data. Any command given by the

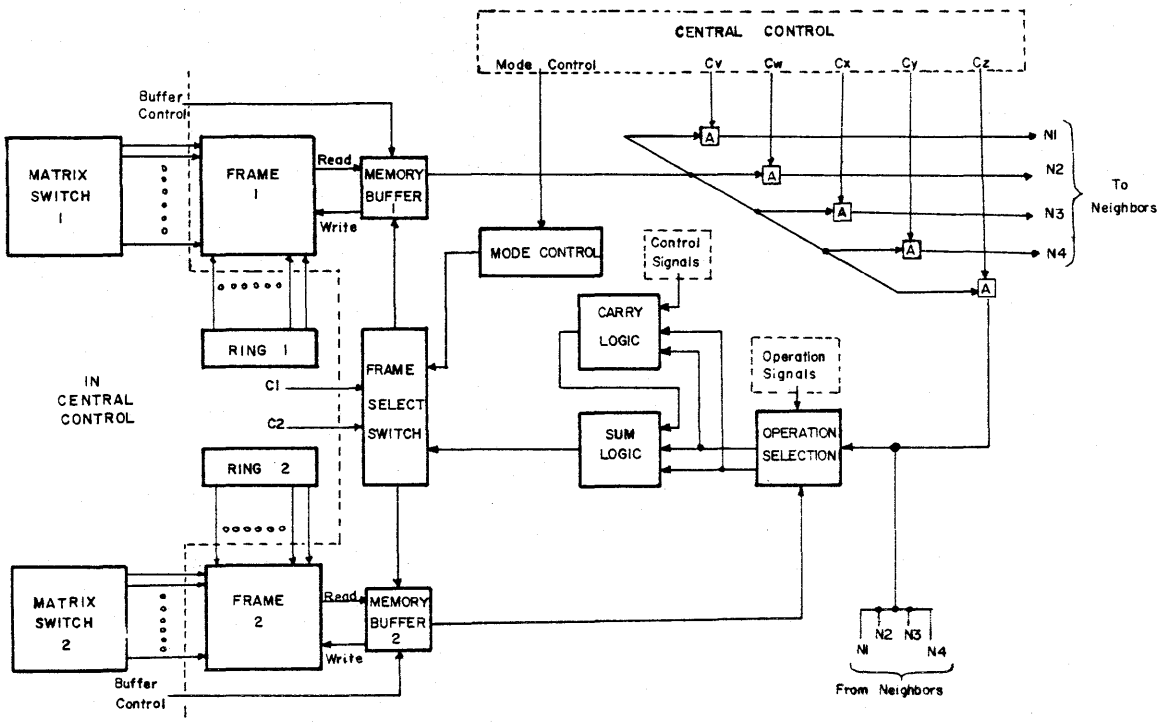


Figure 4. Processing Element Block Diagram

central control unit to the PE matrix is executed by the processing element only when the mode control signals from the control sequencer are identical with the coding of the processing element mode register. The central control unit may activate any combination of four states permitting a given command to be executed by individual elements in different multimodal states.

Upon comparison of a command field with the multimodal state, the execute signal is energized enabling the processing element to execute the command. When this signal is not energized, the command is not executed. If external routing is specified, the numbers in frame one are routed to the specified neighbors, regardless of mode state. The processing elements in a nonselected mode will not accept information; they will return bits of information read from memory to their respective frames unaltered.

Internally controlled mode advancement will take place as "condition met" signals are received from the arithmetic sum network. These signals transfer those processing elements which have satisfied the condition for transfer to a mode of operation specified by central control.

Modes can also be changed by the programmer by using special commands. The set mode command will automatically set all addressed processing elements into the directed mode state.

Commands for loading modes operate on the mode control flip-flops loading into or loading from two bits in memory specified by central control. The store mode command does not alter the contents of the mode control flip-flops. By programming, this capability can be used to tag information or results of calculations.

Row and Column Selection: In a number of matrix calculations, the use of a series of load and store mode commands to do row selection becomes quite cumbersome. Including the capability for commands to select particular rows or columns for the operations saves both time and processing element memory storage.

During row-column operations, the mode control logic is altered by the selection control. Processing elements execute commands in a manner identical to the ordinary mode control operations. Nonselected rows will transmit required operands, but will not alter their memory contents. The selected

processing elements operate in the conventional manner.

In combination with the multimodal operation, row-column selection is a useful programming tool. The hardware in central control consists of a holding register (switches) whose output is gated to either rows or columns. Two holding registers could permit simultaneous row and column selection.

Arithmetic Operation: Each of the two memory frames communicates with a two-bit memory buffer shown in Figure 4. Each buffer holds the bit just read from memory, along with the result of the logical operation which is about to be written into memory.

The frame-one memory buffer includes a control flip-flop which can sense a bit set into the buffer during division or multiplication, as well as the condition of the summing network during a logical or arithmetic operation.

The frame-select switch controls the routing of the sum and memory bits according to the command from central control.

The arithmetic portion of the PE consists of two parts: operation selection logic and a modified full adder. Subtraction is performed by the addition of the complement of the subtrahend to the minuend. SOLOMON uses 2's complement arithmetic in the serial processing element to eliminate the end-around carry required when using 1's complement. The 2's complement is formed by an addition of 1 during the first bit cycle by initially setting the carry to a 1 and gating the complement of each bit. Therefore, a 1 is automatically added during a cycle when no carry is normally present. Negative numbers are stored in 2's complement form within the processing element memory.

Multiplication is accomplished by a series of shifts and additions. Prior to the start of the multiplication, the multiplier is stored in a specific memory location in frame one. The multiplicand is also stored in frame one. When the multiplication signal arrives, the matrix switch is set so that the first multiplier bit is read out of memory into the flip-flop. Gating then modifies the central control signals according to the value of the multiplier bit. They will provide either a set of zeros, or permit addition of the multiplicand to the partial product which is stored in frame two. To maintain the alignment of bits, the second ring is started one bit time earlier than ring one. Note that the product need not be stored in the same processing element as the multiplicand and multiplier, since the multiplication can be implemented in conjunction with any of the four nearest neighbors.

Division is implemented through a nonrestoring technique. When a divisor is smaller than or equal to the dividend, a test is applied to determine if the quotient will not possess any significant bits. When this occurs, the processing element will enable the overflow signal and not divide. The central control unit can then choose to ignore the overflow or stop the computer for the operator to make corrections.

Logical operations such as "and," "or," and "exclusive or" are included in the command repertoire. Other operations such as comparisons are implemented by varying combinations of the control signals that permit flexibility within the processing element logic.

Overflow (when addition or subtraction results in a value greater than the range of the computer) can be sensed and may initiate programmed corrective routines in central control. The execute signal is also transmitted to central control for sensing to determine that some processing elements have met or have not met the transfer conditions.

Processing Element Memory Organization: Each processing element includes two memory frames each with its own read and write circuitry. Each frame has a capacity of sixty-four 32-bit words. These frames are physically organized into stacks as shown in Figure 5. The frame-one planes and

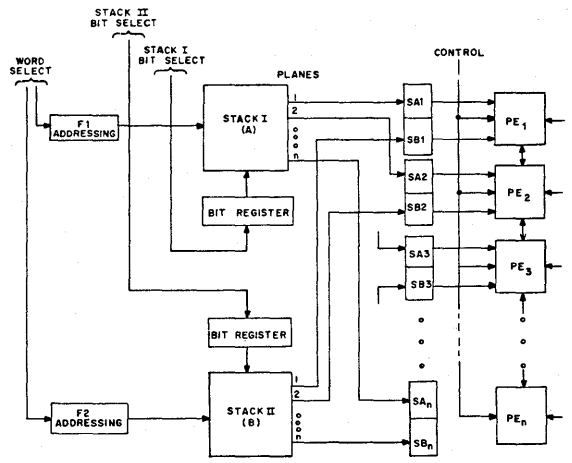


Figure 5. Sharing of Memory Stacks

frame-two planes make up separate stacks (stack I and stack II). These stacks are driven in parallel from central control.

Both sets of X-drivers are controlled from central control and select one of 64 words in each frame (see Figure 6). The word selected from frame 1 can be different from the word from frame 2. While the selected X-drivers are turned on (read-write), the Y-drivers will sequence through the bit positions with a series of read-write pulses, and thus cycle the bits of both selected words serially into the sense amplifiers. The Y-drivers of stack I can be offset by any number of bit positions from the Y-drivers of stack II. The outputs of the sense amplifiers are sent to the buffer flip-flop register where they are gated by the processing element either into its own adder or to a designated nearest neighbor.

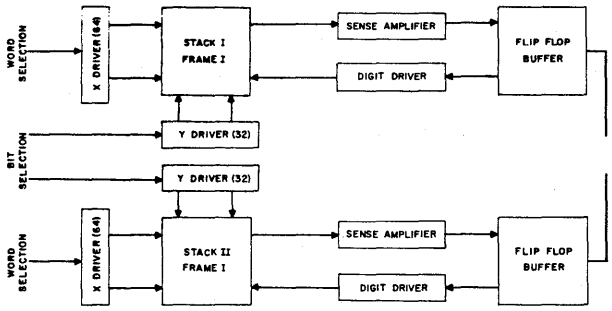


Figure 6. Basic Operation of Memory Unit

Writing the information back into the memory will be accomplished in a similar fashion, with the exception that the flip-flop buffer will control the digit driver, which, in turn, determines whether a one or a zero is written.

Drivers of convenient size and complexity can be built to drive up to 128 frame pairs. Therefore, a 1024-frame memory requires eight sets of X- and Y-drivers. When expanded systems are desired, an additional unit is provided with each module of 128 processing elements.

Central Control

The Control Unit: The purpose of the Control Unit, Figure 7, is to control the operations of the SOLOMON computer complex and to maintain the proper command distribution. This unit is the only one which addresses the program memory. All indexing, whether it is to be performed on an input-output command or a processing element command is completed within this unit. The unit has control over all broadcast and index registers. Loading, transferring, and other operations upon these registers are accomplished while processing element matrix sequencing and input-output control is taking place.

The control unit has sufficient command decoding logic to discriminate between four basic types of commands: (1) processing element commands, (2) input-output control commands, (3) program control commands, and (4) commands which transfer information between the matrix and the input-output equipment.

In processing element commands, the control unit addresses the program memory sequentially, and receives the instruction. The control unit ascertains that the instruction is intended for the sequencer, and then does the required indexing. When the sequencer has completed its previous instructions, the indexed command is transferred in parallel to the network sequencer. The control unit then addresses the program for the next instruction.

The input-output control commands are partially decoded and the addresses are indexed. When the input-output control has completed its previous instruction, the information from the program memory is transferred to the input-output executor.

The program control commands are completely executed within the control unit unless inputs are required from other control complexes. If the former is true, the control unit will complete the instruction and then return to the program memory for new instructions. When inputs from the other units are necessary, the control unit must wait until the device is not busy. Then the command will be executed. This class of instructions includes console control interrupt tests, switch test and control program, transfer instructions, and register controls.

The final class of instructions are those where both the input-output control and network sequencer are required to act together. With this type, both the sequencer and the input-output executor must have completed their previous instructions, then, with indexing completed, the command is transferred in parallel to the network sequencer, the

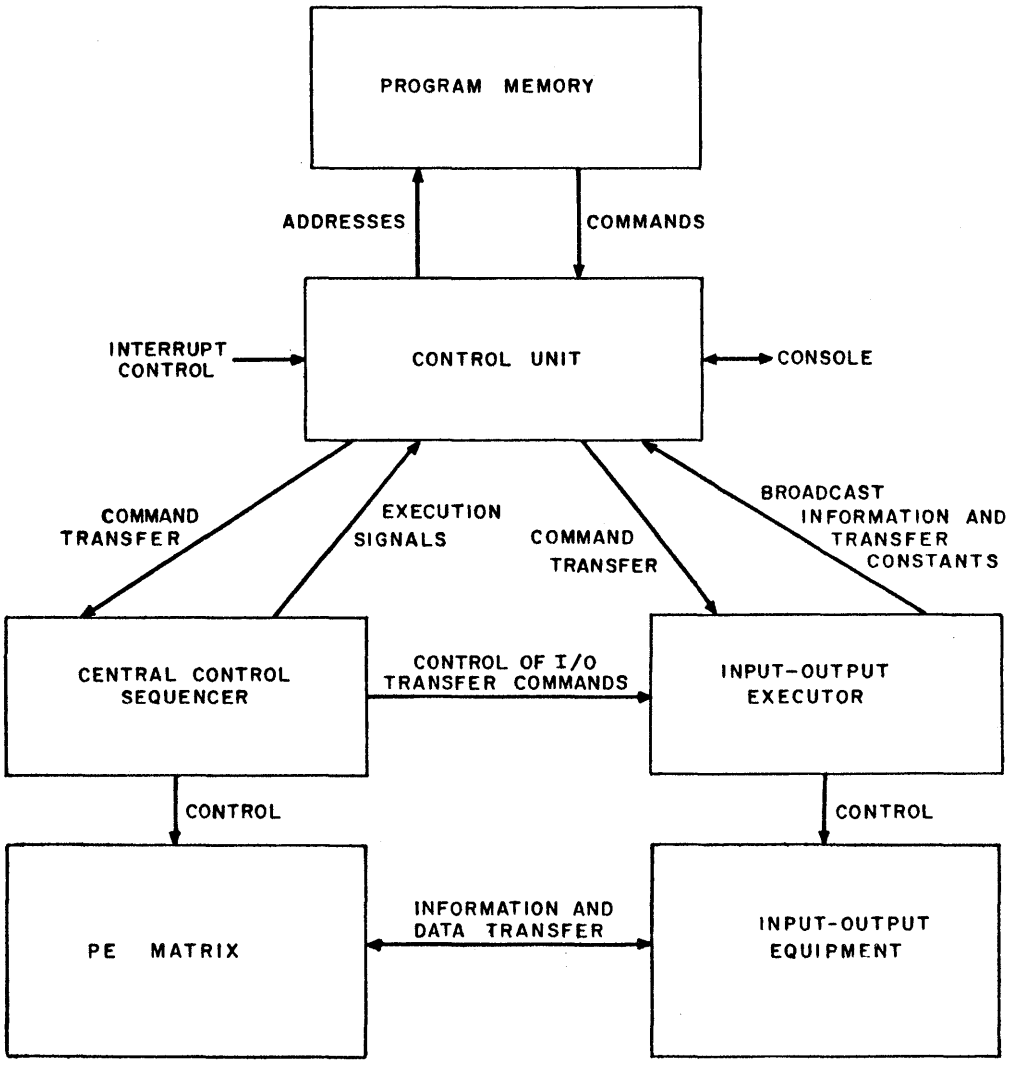


Figure 7. Control Organization

control of the input-output executor is transferred to the sequencer, and the control unit is released to obtain the next instruction.

This organization of control will provide the programmer with the flexibility to maintain a maximum utilization of all control equipment, see Figure 8. Interweaving instructions will provide a maximum amount of command overlap and results in the highest speed.

Network Sequencer: The sequencer, shown in Figure 9, is the portion of the SOLOMON system whose major objective is to provide control signals for commands involving the processing element matrix. The device provides control signals to the input-output unit during information transfers.

The unit has two major functions, one to decode commands and provide control pulses and levels to the processing element matrix, and the second to control memory addressing.

Memory address control is provided by two digit counters and two matrix switches. These counters are loaded by the control unit and will advance or decrement under control of the operation register and controller.

While memory addressing is mainly accomplished by counters, the processing element controls are more varied. After the operation register is loaded from the control unit, it is decoded and supplies command signals to the controller. The controller supplies all the time-varying signals to the processing element matrix and transmits control pulses to other sequencer registers.

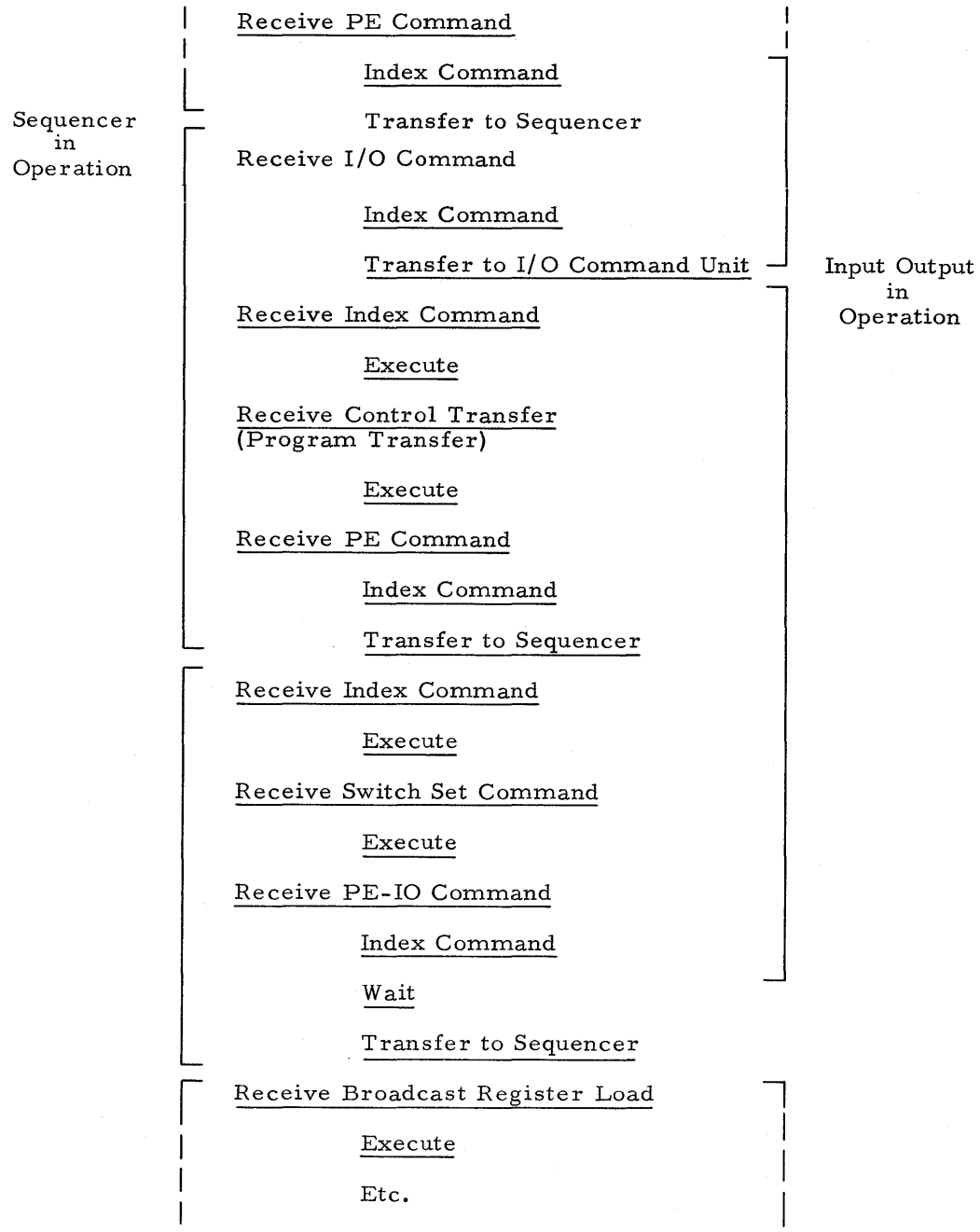


Figure 8. Control Sequence

The controller has the capability of selecting a broadcast option which will enable the central control to act as a "fifth" nearest neighbor to the network. The frame one operand is replaced by one of the broadcast registers in the Control Unit. This option will be selected by the programmer whenever a single constant is required by a large number of processing elements during the same operation cycle.

During input-output operations, the controller sends the coded signals to the input-output control unit for the duration of the operation. The input-output unit then transmits the control pulses to the input-output equipment to maintain exact synchronization between the two subsystems.

The Input Output System: The input output system, Figure 10, is organized to facilitate information transfer between the network

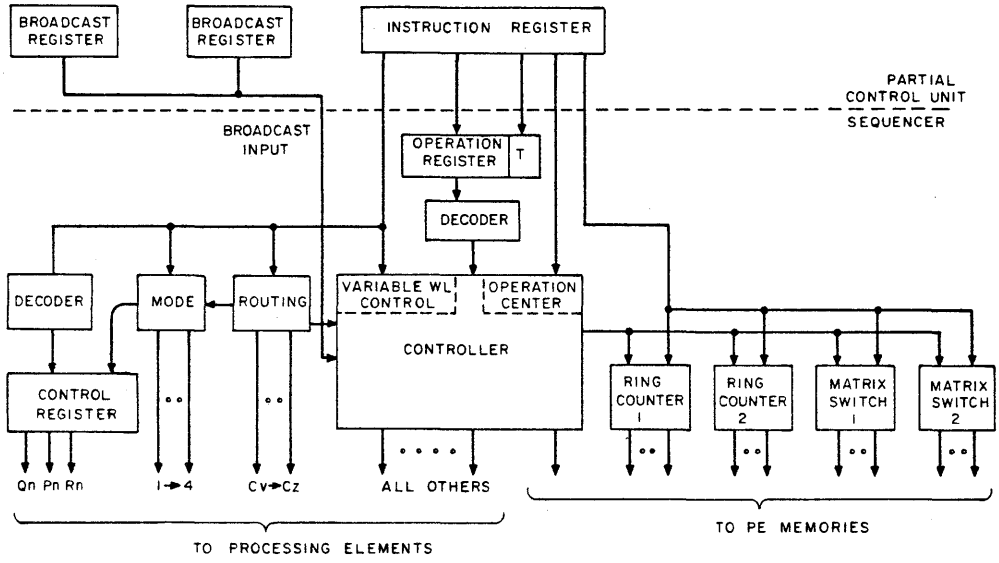


Figure 9. Sequencer Block Diagram

and auxiliary equipment with high efficiency.

The primary information exchange is used as auxiliary network storage in large mesh problems. The exchange consists of wide multi-channel magnetic tape whose function is to keep a high speed drum loaded. This drum operating with parallel channels can provide a bit rate comparable with the network rate. Other advanced storage devices are being considered in this area.

This primary exchange system communicates through the input output buffer with the secondary exchange system. This secondary exchange system provides the user with compatibility with existing systems such as conventional magnetic tape, card reader and punch, high speed printers, paper tape units and other devices. The Format Converter provides conversions from the binary form required by the network to the format required by the peripheral devices.

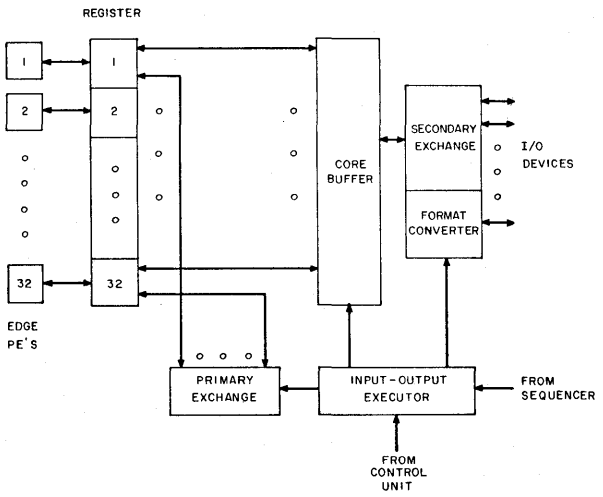


Figure 10. Input Output Organization

Both the primary exchange and the core buffer can communicate with the processing element network in two ways, the "edge" elements and geometric control.

In the first method, since each element has the capability of communication with its four adjacent neighbors a processing element located on the "edge" of the network has one or more free connections. These unused connections are connected to the input-output register.

Figure 11 shows the 32×32 matrix array with the capability to select the particular edge into which information is to be written. This flexibility increases the ease of loading the processing element network.

Additional flexibility is obtained through "geometric control." This technique employs special information transfer commands that enable the programmer to select specific rows or columns which are to communicate directly with the input output register. This eliminates the necessity of multiple transfers to obtain or load information directly into the interior of the matrix.

Programming

In the overall evaluation of any computer system, consideration must be given both to

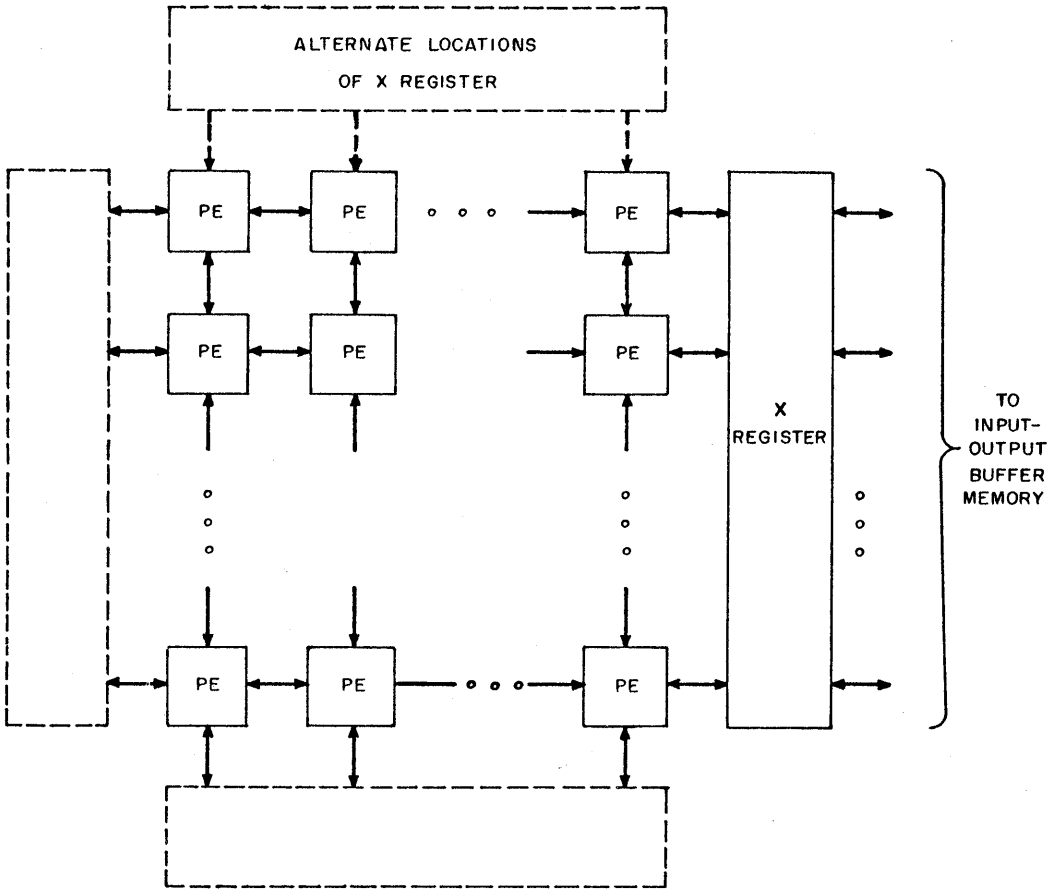


Figure 11. Input Variations Possible With 32 × 32 Array

the mathematical formulation of the problem to be solved and to the way in which this is reduced to a sequence of computer operations. Both of these aspects have a particular significance in the SOLOMON system. It is becoming increasingly apparent that the traditional methods of numerical analysis will be largely inapplicable to the new generation of computers represented by SOLOMON.

Present analytical and programming methods have evolved directly and with little change from centuries of hand calculation and later from use of desk calculators. For this reason, such methods have been considered "natural," and the organization of present computing systems may be directly compared to the computing complex consisting of a human operator, an explicit computational algorithm, and mechanisms (or persons) capable of performing the required sequence of calculations. Heretofore, no serious attempt has been made to employ an approach to computer design which is based not on human capability and "natural" computation methods, but on potential computer capability and entirely new methods "natural" to the computer innovation.

Current developments demand an alteration of this state of affairs. The SOLOMON system, as a consequence of its radically different organization, requires new techniques in numerical analysis and programming for its effective utilization.

The main question is the amenability of basic mathematical functions and processes to the parallel approach. Specifically, in the SOLOMON system a fixed number of processing elements under common control must be efficiently used to perform calculations which will vary both in size and in basic type. A fixed number of processing elements can both speed the solution of a given problem of fixed size and perform substantially different but mathematically related calculations simultaneously. The former capability takes advantage of the parallelism which is intrinsic to many problems but which has not been previously exploited.

The latter stems from the capability to represent broad classes of functions and operators mathematically in normal forms which distinguish between the represented quantities only by the values of constants that occur in their representations; examples of this are the representation of continuous functions by polynomials, of analytic functions by Taylor series, and of linear operators by matrices.

A previous paper* describes in detail the application of the SOLOMON system to problems in partial differential equations and certain matrix calculations. Results to date† establish a performance advantage between 60 and 200 for the SOLOMON Computer compared to currently available large scale digital systems.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the assistance received through many discussions with their associates during the conception and development of the SOLOMON system, and, in particular, E.R. Higgins, W.H. Leonard, and Dr. J.C. Tu.

*Slotnick, D. L., W. C. Borck, and R. C. McReynolds; "Numerical Analysis Considerations for the SOLOMON Computer." Proceedings of the Air Force Rome Air Development Center-Westinghouse Electric Corporation Air Arm Division Workshop in Computer Organization. To appear.

†Jeeves, T. A., op cit.

THE KDF.9 COMPUTER SYSTEM

A. C. D. Haley
English Electric Co., Ltd.,
Kidsgrove,
Stoke-on-Trent, England

SUMMARY

The English Electric KDF.9 computing system has a number of unusual features whose origins are to be found in certain decisions reached at an early stage in the planning of the system. At this time (1958-59) simplified and automatic programming procedures were becoming established as desirable for all programming purposes, whereas previously they had been regarded as appropriate only to rapid programming of "one-off" problems because of the drastic reductions of machine efficiency which seemed inevitable.

Many early interpretive programming schemes aimed to provide an external three-address language, and for a time it appeared that a machine with this type of internal coding approached the ideal. Increasing interest in translating programs, particular for problem languages such as ALGOL and FORTRAN, showed the fallacy of this assumption. It became evident that efficient translation could only be achieved on a computer whose internal structure is adapted to handle lengthy algebraic formulae rather than the artificially divided segments of a three address machine.

The solution to the difficulty was found in the use of a "nesting store" system of working registers. This consists of a number of associated storage positions forming a magazine in which information is stored on a "last in, first out" basis. It is shown that this basic idea leads to development of a computer having an order code near to the ideal for evaluation of problems expressible in algebraic form.

A number of other significant advantages arise directly from the nesting store principel, chief among them being a striking reduction in the program storage space required. This is due to elimination of unnecessary references to main store addresses and to the implicit addressing of operands in the nesting store itself. Many instructions are therefore concerned with specifying only a function, requiring many fewer bits than those instructions involving a main store address. Instructions are therefore of variable length to suit the information content necessary and on average three instructions may occupy a single machine word (48 bits). This again reduces the number of main store references, allowing the use of a store of modest speed while still allowing adequate time for simultaneous operation of a number of peripheral devices on an autonomous main store interrupt basis.

Fast parallel arithmetic facilities are associated with the nesting store, both fixed and floating-point operations being provided. A further nesting store system facilitates the use of subroutines, and a third set of special stores is associated with a particularly comprehensive set of instruction modification and counting procedures.

Operation of the machine is normally under the control of a "Director" program. A number of different Directors cover a variety of operating conditions. Thus a simple version is used when only a single program is to be executed and more sophisticated versions

may be used, for example, to control pseudo-off-line transcription operations in parallel with a main program, operation of several programs simultaneously on a priority basis, etc.

INTRODUCTION

For almost a decade after the first computers were put into service, developments in system specifications were almost exclusively a by-product of local engineering progress. No radical changes took place in machine structure, except insofar as engineering changes led directly to operational ones. Thus the emergence of the ferrite core store as the most reliable and economic rapid access store for any significant quantity of information led to the abandonment, now virtually complete, of the various types of delay line store. In consequence, "optimum programming" is now used only in certain systems which exchange speed for economy and use a magnetic drum for the working store.

The majority of systems continue to be basically simple single address order code machines, in which most orders implicitly specify an arithmetic register as one participant in every operation. It was the subsequent proliferation of transfers between this register and the main store, purely to allow its use for intermediate arithmetic operations on instructions, which led to the introduction at Manchester University of the "B-tube." This in turn has been extended and elaborated to provide the automatic instruction modification and/or counting features which are now universal.

A further reduction in housekeeping operations, with a consequent increase in speed, can be obtained by providing not a single register associated with the arithmetic unit, but a number of registers. Sometimes all facilities are available on all registers, and in other machines the facilities are divided.

Changes and elaborations such as these have, of course, had the primary aim of increasing the effective speed of the machine. The penalty to be paid is the complexity of programming. The increased quantity of hardware required is, of course, more than compensated by increased computing power.

There is no corresponding compensation for the increased programming costs, particularly for problems of infrequent occurrence. This factor was the provocation for much of the early work on simplified programming schemes. Major programs and those to be used repeatedly were written in machine code, but the remainder were written in problem-oriented languages either obeyed interpretively (as if a set of subroutines) or translated into the necessary machine code program by highly sophisticated translator routines. Typical of the former approach are English Electric "Alphacode" [1] and Ferranti/Manchester University "Autocode" [2, 3] and of the latter method I.B.M. "Fortran" [4] and English Electric "Alphacode Translator" [5, 6]. The penalties of using these schemes are again different in nature. Interpretive routines lead to inefficient use of machine time during every run, factors of 10 to 100 covering most of the systems in common use. The translator routines, in contrast, may ideally produce a 100% efficient program, although a loss of speed by a factor of 1-1/2 to 5 is more usual. The translation operation, performed only once, may however occupy long periods and may swamp the subsequent gain over the interpretive method for a rarely used program.

In the late nineteen-fifties it was evident that a successful new general-purpose system should ideally have two programming features:

(a) It should have an order code designed to allow easy efficient coding in machine language after a minimum training period:

(b) The language should be such as to allow the preparation of a number of rapid translators (for many categories of work) from problem-oriented languages into efficient machine programs.

Studies based on these fundamental requirements culminated in the machine organization selected for KDF.9.

Choice of an Order Code

Many interpretive schemes used up to this time had been of three-address type, where the typical instruction is of the form

$$C = A \text{ (function) } B,$$

and A,B,C are main store addresses. Experience had shown that this type of code was well adapted for design calculations and scientific usage, expecially if a wide range of fixed and floating point functions and well-chosen loop counting and instruction modifying facilities were made available.

At a time when potential arithmetic speeds were overtaking storage access rates the use of a three-address machine code had the drawback of requiring four main store references for each operation (including extraction of an instruction), and with the constant demand for stores of 32,000 or more words the necessary instruction length approaches 60 bits when provision for a large number of functions and modifiers is made.

A single address system, on the other hand, requires greater care in programming and a multi-accumulator machine is perhaps worse still from this standpoint.

None of these conventional structures is particularly well suited to preparation of efficient translation routines, and a study was therefore made of a special form of multi-accumulator system using automatic allocation of addresses for the working registers. This depends on presentation of data to the system in a form closely analagous to the "reverse Polish" notation. (This notation merely involves writing the sign of any arithmetic operation after, rather than before, the two operands.

Thus a + b is written ab+

a ÷ b is written ab÷, etc.)

Fundamentally the procedure is to arrange the machine structure in such a way as to allow operations to be performed in a natural order. Thus, in carrying out the operations involved in calculating E = A.B. + C.D, the natural sequence is

obtain A:

obtain B:

multiply & retain A.B:

obtain C:

obtain D:

multiply & retain C.D:

add:

store E.

Given a suitable "user code" (i.e., a convenient form in which to prepare a program of instructions), it was evident that such a machine structure, if attainable at reasonable cost, would meet requirement (a) above. The belief in its appropriateness for simplicity of coding was coincidentally supported by the appearance of an interpretive scheme [7], producing a similar problem-language, written for DEUCE. This proposal, however, attracted little attention, mainly because the multi-level storage system of DEUCE prevented attainment of a reasonable working efficiency.

The potential virtues of this type of problem language were again supported by a subsequent proposal to use a reverse Polish notation as a computer common language (APT). For a number of reasons this proposal was not adopted, but two of the reasons advanced in its favor are worthy of note in the present context. They were:

(a) That the language is one in which problems can be formulated with little effort after a short period of training:

(b) That the process of automatic or manual translation from any conventional problem language to most of the existing machine languages requires an effective expression in reverse Polish notation as an intermediate step in the procedure. Using such a machine code therefore eliminates a substantial part of the process.

The Nesting Store

The evaluation of a formula above can be seen to consist of successive operations either of fetching new operands (or storing results) or of performing operations on these most recently named. An intermediate result is immediately reused (as in the seventh step) or temporarily pushed "below the surface," as when the product A.B is superseded by the fetching from store of C.

A mechanical analogue of such a system is shown in Figure 1. It consists of a magazine, spring loaded to maintain the contents at the top, with only one point of entry and exit. Objects stored can therefore only enter and leave on a "last in, first out" basis.

The electronic equivalent is shown in Figure 2. There are n separate registers, each capable of accommodating one machine word, and corresponding bit positions of each register are connected so that the assembly can also be treated as a number of n-bit reversible shifting registers. Information can be

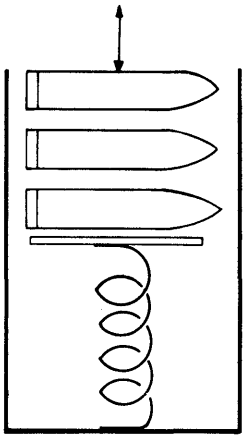


Figure 1

transferred to the top register N1 from a conventional main store buffer register (a parallel machine is assumed), and simultaneously with any such transfer a shift pulse causes a downward shift of any existing information. Thus the new word appears in N1, the previous content of N1 in N2, etc. This process is called "nesting down." Reversal of the process causes "nesting up" with transfer of the content of N1 towards the main store. "Nesting up" and "nesting down" can occur in any sequence, provided, of course, that not more than n words are in the store at any time.

Associated with the top two registers is a "mill" or arithmetic unit [8]. Serial transfers to and from the mill are shown for clarity, but in a fast machine parallel working is again used.

To allow the evaluation of a formula such as the simple one of section 2, the mill must be made capable of the two arithmetic operations needed. One of these is addition, and at this point it must be noted that in general no operand is considered as necessary after being used by the mill. The operation "add" therefore uses the words in N1 and N2, and places the sum back in N1. Since N2 is now unoccupied, "nesting up" occurs, the content of N3 moving to N2, etc.

The natural effect of most of the conventional arithmetic operations can now be visualized. Thus, "multiply" produces a double-length product in N1 and N2 (the more significant half in N1 for obvious reasons). No nesting is involved in this operation, but the more elaborate "multiply and round" produces a single length result in N1 and therefore requires nesting up.

Two important points emerge at this stage. The first is that an operation sequence written as

YA, YB, x, YC, YD, x, +, = YE

evaluates the formula given earlier (YA is to be interpreted as "fetch from the main store address containing A into N1," and = YE as a converse operation). This sequence is the one involving a reduction to a minimum number of main store operations.

The second important point is that arithmetic operations have implied addresses, so that only the function need be specified. On the other hand, all instructions referring to a main store address require many bits for this purpose unless flexibility and convenience are sacrificed by addressing relative to a local datum.

Variable Length Instructions

It is clearly advantageous to economize in instruction storage space, and hence, for reasons stated above, to allow instructions to vary in length according to their function and the additional information they require. Obviously any instruction must carry within itself a definition of the number of bits included in it. Analysis shows that complete variability is unprofitable (as well as complicating the hardware), since five bits would be needed to specify the length. Three possible lengths of 8, 16 and 24 bits are therefore made available, including in each case bits to designate length. In connection with instruction length, a unit of eight bits is referred to as a "syllable." Most arithmetic operations are therefore one-syllable instructions; memory fetch and store operations together with jumps are three-syllable, while two-syllable instructions include a number requiring parameters of less length than memory addresses (shift instructions, input/output instructions, etc.).

The word length of the computer is 48 bits, and this is the smallest unit in which information can be extracted from the main store. Instructions are stored continuously and obeyed serially; i.e., the store area concerned is regarded as a continuous series of eight-bit syllables, rather than of 48-bit words, and two or three-syllable instructions may overlap from one word to the next. Associated with the main control there is

therefore a two-word register. This at any time holds the word containing the current instruction together with that from the next higher main store position (if the current instruction overlaps a word boundary both words are of course in use). As soon as all syllables in one word have been used, this word is replaced at the first available main store cycle by a further word in sequence from the main store.

Because of the economy in instruction storage space achieved by these means, it is frequently possible to contain important inner program loops in two words of instructions. Provision is made to mark such loops by a special jump instruction, whose effect is to inhibit the extraction of a new instruction word until the condition for leaving the loop is satisfied. This saves two main store cycles (12 microseconds) for extracting instruction words on every circuit of the loop, and incidentally saves a syllable since again no main store address needs specifying completely.

Some additional complexity arises due to the separation of control into two parts, one associated primarily with arithmetic operations (known as "Mill Control") and the other, which runs normally at least one instruction in advance, controlling most other internal operations and in particular controlling access to the main store (this is called "Main Control"). The object of this is, of course, to increase effective speed by allowing, for example, a new instruction word to be extracted while an arithmetic operation proceeds in the nesting store. Such overlaps are completely automatic and no special actions are required of the programmer.

Further Consideration of Nesting Store

At this stage it is convenient to examine the nesting store and the consequence of its use in a little more detail.

It should first be stated that the representation of Figure 2 while possible is uneconomic if more than a two or three words of storage capacity are required. Examination of a large number of programmes shows that only occasionally is storage for more than about eight words needed in the evaluation of an expression, and that a sixteen word limit to capacity will cause inconvenience only on very rare occasions.

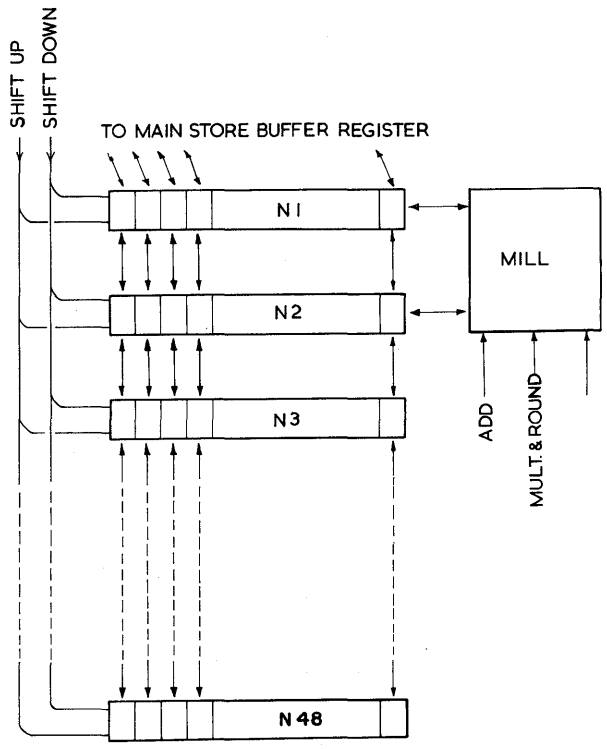


Figure 2

A set of 16 registers of 48 bits each is an expensive assemblage and it is natural to examine the possibility of using core storage in some form. The use of a core store for all positions carries penalties in speed of operation (this remains true as long as the speed is similar to the main store speed), and a satisfactory solution is reached by the use of the configuration shown in Figure 3.

The top three registers are now conventional flip-flop registers, and a 16-word core plane makes up the remainder of the store. This gives a total of 19 words which is advantageous for reasons shown below. The flip-flop registers are inter-connected by a number of gated transfer paths (each for parallel transfer of 48 bit words) which also include transfers to and from the arithmetic unit. An unconventional transformer coupling system allows these paths to be provided economically while permitting the transfer of a word between any two registers in under half a microsecond. Below the registers the core store operates in a manner differing in form but identical in principle with the lower registers of Figure 2. No shifting of words between store positions occurs as nesting up or down is called. Instead successive positions are emptied or filled by successively

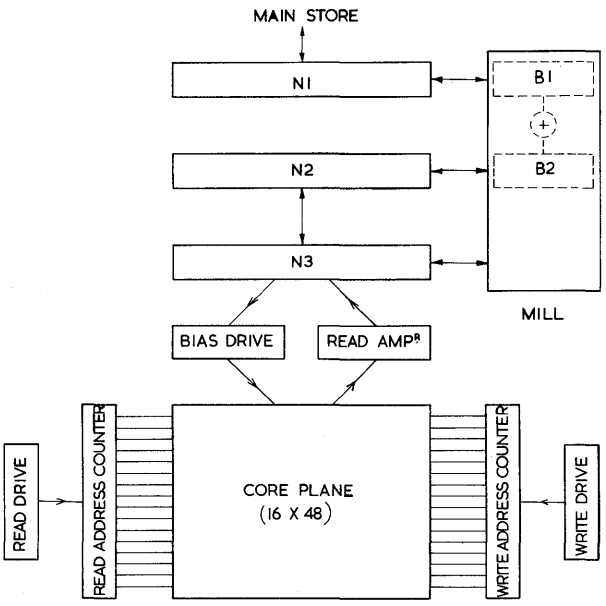


Figure 3

addressing levels in the plane. Suppose, for instance, that the whole system is empty, and successive words are fed into N1 (by a series of main store fetch instructions). The transfer paths between N1, N2, N3 and the mill are gated as required by control.

The only complication arising here is that the transfer system does not allow N2, for example, to send to N3 in the same half microsecond period in which it receives from N1. The two buffer registers of the mill are therefore used, and the transfers are, in the first half microsecond, N1 to B1 and N2 to B2.

The write address counter is at zero and the read counter at 15 for a reason which will appear shortly. In the first half microsecond N3 is also allowed to set up bias currents in the vertical core lines where digits are to be written.

The next half microsecond sees the word in N3 written into the top core plane position by operation of the write drive, and the subsequent period covers the operations needed to complete the first fetch. These are main store to N1, B1 to N2, and B2 to N3. Simultaneously the read and write address counters are advanced by one to zero and one respectively.

A second and third fetch may now be executed in exactly the same way, but, as the system was assumed empty originally, blanks (as distinct from data zeroes) are written into levels 0, 1, 2 of the core plane, leaving the cores cleared. Only on the fourth and subsequent fetches does a genuine word appear in level 3.

Continuation of these operations will on the sixteenth occasion fill the lowest level of the core plane, and the write counter has now carried round to address zero while the read counter is at 15. The programme is interrupted if further entries are attempted (except under circumstances mentioned in a later section).

It will have been noted that the read counter is always one position behind the write counter, so that if a nesting down operation is followed by nesting up, the read drive is used and no correction is needed to the counter position in order to read out the last word inserted.

Note also that all operations are either to read out, leaving a clear storage position, or to write into an already clear position. The normal read/write cycle is therefore unnecessary and would be wasteful of time.

Special Nesting Store Instructions

The simple example given earlier illustrates the general nature of the instructions provided. It is soon discovered, however, that a few instructions are desirable which have no real counterpart in more conventional systems.

Straightforward evaluation of an expression in algebraic form will usually produce the desired result without special manipulation. Occasionally it will be found that two operands, for example prior to a division, are in reverse order. An instruction "REVERSE" has the effect of interchanging the contents of N1 and N2.

The instruction "DUPLICATE," which nests down and leaves a copy of N1 in both N1 and N2 has many uses. Followed by MULTIPLY it produces the square of N1, and it also allows an operand shortly to be lost in some other operation to be preserved for later use without requiring a main store reference.

Instructions ZERO and ERASE bring an all zero word into N1, nesting down, and erase N1, nesting up, respectively.

Many instructions are also available in double-length form. Thus double length addition (mnemonic +D) treats N1 and N2 as one 96 bit number, and N3, N4 as a second.

It produces a double-length sum in N1, N2, nesting up two places.

Single and double precision floating point representation is also permitted, the corresponding mnemonic forms for addition being +F and +DF.

With these instructions it is possible to introduce an example showing the power of the system and incidentally the speed and the economy in instruction storage space.

The example to be given is perhaps somewhat artificial, but it has been selected to illustrate not only the essential simplicity of evaluation of any expression from an algebraic statement, but to illustrate also that even in a sophisticated system there is scope for an occasional elegant twist. The formula to be evaluated is

$$f = \frac{a\,(a^3\,b^2 + 1)}{b + 2c^2\,d^2}$$

where a, b, c, d are single length fixed point numbers stored at Ya, Yb, Yc, Yd, non-adjacent addresses in the main store.

The table shows the successive steps in the calculation, together with the contents after each step of the top few cells of the nesting store.

The following points should be noted:

(a) Again main store references are reduced to the minimum practicable.

(b) A count shows that only 30 syllables, or five instruction words, are used (there are no two-syllable instructions in this particular example).

(c) It is advantageous to evaluate the numerator in expanded form in this particular case.

(d) The use of DOUBLE DUPLICATE at step three neatly anticipates future requirements for the parameters. An automatically programmed version would fetch these parameters again or could perhaps use a less satisfactory temporary storage process to avoid this.

The complete evaluation on KDF.9 takes less than 170 microseconds.

It is interesting and instructive to compare this performance with that of a conventional one or three-address machine. The same actual arithmetic and main store speeds, and a single accumulator only for the one-address machine are assumed. It is also assumed that instructions are packed two to a word for the single address system and one to a word for the three address type.

The single-address programme then occupies eight words of storage and takes about 250 microseconds (increases of 50% in both factors). A three-address system uses 9 words of storage and takes 340 microseconds.

This example requires few operations of a "housekeeping" nature and the savings arising from use of a nesting store are less prominent than is frequently the case. On the other hand, it is also possible to find cases where there is little difference between the various systems.

The very poor relative speed of the three-address machine is accentuated because of the assumption of the same basic internal speeds as for KDF.9 (typically 6 microsecond store cycle, 15 microsecond multiplication). It is, however, true that these represent speeds attainable at corresponding levels of economy. A substantially faster store with 1 microsecond cycle time could be used, at a significant cost penalty, and would bring down the problem time to around 160 microseconds. There is, of course, no corresponding saving in storage space.

Treatment of Sub-Routines

It will have been observed that in the example above the condition of the nesting store at the end was identical with that at the start. A system of preparing sub-routines can be based on this fact. At any stage in a program there may be information in one or more cells of the nesting store. At this point the parameters required by the sub-routine must be planted by the main program (or of course by a lower order sub-routine). A note must also be made of the next instruction in the program for re-entry purposes. These two points will be considered separately.

Just as an instruction such as MULTIPLY expects to find operands in the top cells of the nesting store and terminates leaving the product in place of the operands (nesting as necessary), so a sub-routine can also be arranged to function. It then becomes in effect a machine instruction in that it does not influence the nesting store contents below the level of the operands made ready for it.

It will also be obvious, as soon as multi-level sub-routines are considered, that the operations involved in storing the return instruction address for use after each

Table 1

OPERATION	N1	N2	N3	N4	N5
FETCH	b	-	-	-	-
FETCH	a	b	-	-	-
DOUBLE DUP.	a	b	a	b	-
DUPLICATE	a	a	b	a	b
MULTIPLY	a^2	b	a	b	-
MULTIPLY	a^2 b	a	b	-	-
DUPLICATE	a^2 b	a^2 b	a	b	-
MULTIPLY	a^4 b^2	a	b	-	-
ADD	a^4 b^2 + a	b	-	-	-
REVERSE	b	NUM	-	-	-
FETCH d	d	b	NUM	-	-
FETCH c	c	d	b	NUM	-
MULTIPLY	cd	b	NUM	-	-
DUPLICATE	cd	cd	b	NUM	-
MULTIPLY	c^2 d^2	b	NUM	-	-
DUPLICATE	c^2 d^2	c^2 d^2	b	NUM	-
ADD	$2c^2$ d^2	b	NUM	-	-
ADD	DENOM	NUM	-	-	-
DIVIDE	$\frac{\text{NUM}}{\text{DENOM}} = f$	-	-	-	-
STORE		-	-	-	-

sub-routine in turn are again of a last in, first out nature. Another nesting store is therefore provided for this purpose, but there is here, of course, no necessity for a number of inter-connected registers or any arithmetic facility. This "sub-routine jump nesting store" (S.J.N.S.) therefore has one register as its most accessible cell, with again a sixteen-word core plane below. As in the case of the main nesting store, only a total of 16 words, not 17 as might have been expected in this case, are available. Since one is reserved for a special purpose (see below), only 15 may be used by the programmer.

The instruction "Jump to Sub-routine" has the effect of planting its own address in the top cell of S.J.N.S. and of causing an unconditional jump to the entry point of the required sub-routine. It will be noted that

S.J.N.S. must receive not only the word address but also the syllabic address of the return point.

Any sub-routine is terminated by one of the variants of the basic EXIT instruction, which transfers control to the instruction whose address is in the top cell of S.J.N.S. (nesting it up one cell). These variants augment this address by units of three syllables before performing the basic EXIT operation, and thus return control to the instruction immediately following the "Jump to sub-routine" instruction (itself three syllables long) or to one of the succeeding three-syllable instructions. These are usually a string of unconditional jump instructions, corresponding to failure exits or multiple exit points preceding the normal return instruction.

The Q-Stores

The computer organization is completed by addition of a further set of storage registers (not of nesting type) known as the Q-stores, and by provision of an input/output system.

The first of these features is based on conventional practices, and will be treated briefly. A set of fifteen registers (formally 16, one of which is identically zero) is used for address modification, counting and a number of other purposes. Each register is a full 48-bit word, but for address modification is considered as having three 16-bit independent sections for modifier, increment and counter.

Any main store reference has associated with it a Q-store number (or an implied Q0), and refers to the address specified, augmented by the content of the modifier section. If the letter Q is added to the mnemonic form of the instruction then after the address has been calculated the modifier is changed by addition of the increment and the counter is reduced by one. Jump instructions testing the counters are, of course, provided.

The Q-stores may also be used if desired as 48-bit registers, or as independent 16-bit registers, in each case with accumulative or direct input. The "counter" part of any Q-store may be used to hold the amount of shift (positive or negative) in shift instructions. There are sufficient facilities of this kind in the machine to remove the need for any kind of programmed instruction modification.

The other main use of the Q-stores is in connection with input/output operations, outlined in the next section.

Input/Output Operations

Provision is made for use of the normal peripheral devices, each one operating completely autonomously and simultaneously with other peripherals and with computer operations. In the standard system up to 16 peripheral devices, with a total transfer rate in excess of a million characters per second, may be handled.

To call any peripheral transfer, an instruction specifies the nature of the operation, referring also to a Q-store in which the other required parameters have been planted. These are the device number, and the limiting addresses of the main store area concerned, since peripheral transfers are of variable length.

Assuming that the device is available (i.e., not already busy), a check is now carried out by the input/output control system that the main store area specified does not overlap that involved in any peripheral transfer already in progress. The parameters are restored within the control, so that the Q-store register is freed for further use by the program.

The transfer now proceeds in a manner which, for economic reasons, differs depending on whether the device concerned is fast (magnetic tape, for example) or slow (punched card or paper tape, etc.).

In the case of the former, six-bit characters are assembled (taking a read operation by way of illustration) into machine words. When a word is complete, it is placed in a single word buffer and a signal to the main control system seizes a store cycle as soon as possible to transfer the word into store. Such calls have priority on the time of the main store, but any one peripheral device may have to wait until other devices have been dealt with.

Since such a buffering system uses a substantial quantity of equipment, more economical procedures are adopted for the slower devices. A common unit is shared by all these, and no assembly into words takes place outside the main store. Instead, as any device has a character ready, the required main store word is extracted, the character inserted in the next character

location and the word is returned to store.

This process is somewhat prodigal of main store time, but the penalty of handling devices with character rates up to a few kilocycles per second in this way is quite negligible.

An attempt by the computer to make use of a peripheral device or any part of a main store area concerned in an uncompleted peripheral transfer results in a "lock-out." The programme operation is interrupted until the prior operation is completed. The user is thus freed from any obligation to check and guard in his program against such conflicting operations.

KDF.9 "User Code"

Up to this point it has been implied that programs are written directly in machine code, albeit in mnemonic form. A very simple compiler routine can clearly translate the mnemonics into instructions. This would, however, leave the programmer with certain tedious tasks, such as calculation of addresses for jump instructions. The syllabic instruction form increases this problem.

The opportunity is therefore taken to incorporate in the compiler a number of additional features to eliminate such difficulties. One or two specific facilities will be mentioned.

The mnemonic code handled by the compiler is called "User Code" and has the following important characteristic: every User Code instruction is either a directive to the compiler, not appearing explicitly in the compiled version of the program, or is compiled into one machine instruction proper. Thus there are no macro-instructions in User Code which become sequences of instructions in basic machine code, the correspondence between User Code and machine code instructions being essentially one to one.

Calculation of jump addresses is handled by the compiler, the user being required only to label the entry point with an arbitrary reference number and to specify "jump to reference number ... if ..." (for example JrCqZ is the mnemonic meaning "jump to reference r if the counter section of Q-store q is zero").

Further elaboration of this process allows a similar labelling of sub-routines. When it encounters an instruction JSLi, the compiler ensures that a copy of the library sub-routine i will be made from the library tape and incorporated into the program. It also writes the appropriate entry address into the jump instruction in the main program.

It will be remembered that in section 3 an instruction YA was used as a mnemonic for "Fetch into N1 the word in main store address A." It is clearly possible for the programmer to treat as his data store a continuous main store area to be addressed as Y0, Y1, etc. The compiler is again used to convert these relative addresses to absolute addresses, allocating as the starting point the first word available after assembly of the programme.

An obvious extension is to allow the programmer to use a number of groups designated YA, YB, etc., each having an area commencing at zero. Thus YA0, YB75 are permissible addresses.

Other areas of store are similarly allocated by the compiler. Constants appear to the user as a set of stores known as V-stores. To incorporate a required constant for a program requires only the user code statement that, for example, V27 = F3.14159. This has the effect of converting the numeric value in standard floating binary form and allocating a store word to it. Any subsequent program reference to V27 fetches this constant into the nesting store.

Similarly the programmer can refer to main store areas reserved by the compiler for working store (storage of intermediate results or data). Exactly as for Y and V stores these are referred to with prefix W.

When the compilation is performed, the resultant program commences with the transcribed main program, followed by any sub-routines used. Areas for V and W stores (and for other functions of this type) are provided up to the highest numbered of each type in the original program. Finally the remaining area becomes the Y store area, thus allowing maximum generality.

Clearly it is possible to prepare special versions of the compiler with any degree of elaboration or desired characteristics. The incorporation of a routine for conversion of constants (which may be expressed in a number of ways) is a requirement for all uses, but it is evident that the principle can be extended as desired.

Thus the standard compiler, which is appropriate for preparation of programs for a variety of applications, may be supplemented by additional special purpose compilers.

Interruption and the Use of a Director Routine

KDF.9, like other computers of its generation, has built-in interrupt facilities. Specifically, this means that at virtually any time the normal sequence of instructions may be suspended and control transferred to a fixed address (syllable 0 of word 0 for obvious practical reasons). Such a transfer of control is called an Interruption. It is arbitrary only in the sense that it is generally outside the control of the programmer. It will take place only when one of a certain number of quite clearly defined situations arises in the machine. When an Interruption occurs, the interrupted program is left in such a state that it may be subsequently resumed and will then continue exactly as if nothing had happened - unless the reason for the interruption was some obvious abuse by the program of the facilities of the machine.

The purpose of the Interruption facility is to make "Time-sharing" possible. Here it is necessary to distinguish between "parallel operation" and "time-sharing." The former implies that the machine is doing more than one operation at a particular moment. On KDF.9 such occasions include the ability of one or more peripheral transfer operations to proceed at the same time as computation as long as no lock-out violation occurs (see above). If such a violation does occur, a transfer of control is necessary in order to enter some other sequence of instructions which can proceed unaffected by the lock-out.

This switch of control is automatic, and so implies the necessity of interruption, in order that the programmer shall not be required to anticipate and take action over lock-out violations. The ability to switch from one instruction sequence to another is called "Time-sharing"; it will be evident that time-sharing and parallel operation go hand-in-hand, the former enabling the most efficient use to be made of the latter.

There are two versions of the KDF.9 system. One of them has an elaborate time sharing facility which enables up to four independent programs to be stored within the machine at once (together with a supervisory program or "Director"). They are obeyed on a time-shared priority basis - that is, each program is allocated a priority, and the hardware and the supervisory program together ensure that the program of highest priority is always operating subject only to peripheral lockout conditions.

The other version of the KDF.9 system only permits one program to be stored, in addition to a supervisory program. This version can be converted to "Full Time-sharing" by addition of equipment. The extra equipment includes:

(a) Extra core planes which enable each program to have its own Nesting Store, Subroutine Jump Nesting Store and Q-Store. To switch Nesting Stores, for instance, it is necessary only to nest down three places, so that the entire contents of the nesting store are in the bottom sixteen cells. These are all in the core plane, which can then be disconnected from the top registers and replaced by another core plane. This is a very satisfactory compromise between loss of time during changeover and volume of extra equipment required.

(b) A register corresponding to each priority level, which is set whenever that program is held up by a peripheral lock-out and which records the details of the lock-out. Interruptions are caused whenever a hold-up occurs and also whenever a lock-out is cleared which was holding up a program of higher priority than the one currently operating. For this purpose a register noting the current priority level is also necessary.

Features common to both types of machine provide for relative addressing and for address checking. The relative addressing feature causes the contents of a "Base Address" register to be added to any main store address used by a program before access to the main store is made. This allows programs to be coded always as if stored at location 0 onwards, but to be stored and obeyed in any segment of the store.

The address checking actually precedes augmentation of the relative address by the base address, and includes a check that the relative address is not negative and does not exceed the size of the store area allocated to that particular program. (Core storage allocation is completely flexible and is in the hands of the supervisory program. Programs may be moved about bodily in the store and may have their priorities interchanged).

If a program tries to go outside its allocated storage area a "Lock-in Violation" is said to have occurred, and an interruption follows. The same thing happens if a program tries to use a peripheral device which

has not been allocated to it - a register of currently allocated peripheral devices is automatically referred to every time a peripheral transfer is called.

In addition, interruption will occur if an ordinary program tries to use one of a number of instructions which are reserved for use by the Director. Such instructions provide access to the various hidden registers concerned with the interruption facilities.

The over-riding objective is to ensure that no program is capable of doing anything which can upset the operation of any other.

Other interruptions can be instigated by the machine operator, in order to allow input of control messages on the console typewriter. The program itself may also include instructions which cause entry to the Director in order, for example, to ask for allocation of peripheral units.

One other reason for interruption, of particular significance, is that which occurs whenever a peripheral transfer called by the Director itself comes to an end. This enables a certain amount of "programmed time-sharing" within the Director. There are many interesting and valuable possibilities. Thus, the feature is used to allow programs to output "lines of print" which the Director can send direct to a printer or to a magnetic tape for subsequent off or on-line printing - the latter again Director controlled. This allows standard programs to be written so as to be capable of running on systems having different output device configurations.

The only limitation on the facilities offered by such supervisory routines lies in the size of program which can be allowed without restricting the "main program" unduly. It is therefore expected that in addition to the "standard Directors" a number of others will appear for various ranges of use.

Any Director must satisfy a number of requirements. It will be stored throughout normal operation of the machine in word 0 onwards of the main store and will be entered at this point by any interruption. On each such entry, it must:

(a) preserve, as far as is necessary, the state of the interrupted program;

(b) discover the reason for interruption and take appropriate action;

(c) return to program.

These requirements will be considered in turn.

(a) This involves very little work on the Director's part. The address to which control must be returned when the program is resumed is automatically planted in the S.J.N.S. by the interruption process. This makes use of one of the "spare" cells of this store, leaving one for use by the Director itself.

Similarly the Director makes use of the three spare cells of the Nesting Store and therefore does not have to do anything to preserve the sixteen cells used by programs. It must, however, store away the contents of any Q-stores which it is going to use itself (usually three) and of two special registers provided to record arithmetic overflow and peripheral device states.

(b) The Director has access to a "Reason for Interrupt" register, different digits of which are used to indicate the different reasons. It is, in fact, the setting and subsequent detection of non-zero bits in this register, as various situations arise, which triggers off the interruption sequence. Once interruption has occurred it cannot occur again until control has been returned to program; however, digits corresponding to reasons for interruption may appear in the register at any time (they are cleared out as soon as the register is read by the Director).

(c) In the case of a KDF.9 system with full time-sharing facilities, this requires the Director first to determine the priority level to which control will be returned, and then to arrange for connection of the appropriate Nesting Stores, etc.

On any machine, the Director must also restore any Q-stores which were temporarily parked away, and must reset the two special registers mentioned in (a) above. The Base Address register, which was at zero while the Director was in operation, must then be set before finally using a special version of the EXIT instruction to return to the main program (this special instruction removes the inhibition preventing interruption while the Director operates).

CONCLUSIONS

In the space available it has been possible only to outline some of the distinctive features of the KDF.9 system. Many of these arise from the novel arrangement of working registers, whereas others are unspectacular

in terms of hardware but are the product of close investigation of operational needs. No attempt has been made to catalogue the performance factors of the KDF.9 system or even to include a specification. It is, however, confidently anticipated that it may point the way to even more striking improvements in the ratio of performance to cost.

Apart from the standard versions of the Director already mentioned, efficient use of a system of the proper KDF.9 presupposes the availability of an adequate "software package." Prominent among the individual items associated with the system are the following:

(a) A fast-compiling "load and go" ALGOL Compiler [9].
(b) A fast object-program ALGOL translator [10, 11].
(c) ALGOL program testing and operating systems.
(d) An ALGOL procedure library.
(e) Translators for FORTRAN, COBOL and other languages.

Detailed descriptions of some of these items appear elsewhere.

ACKNOWLEDGMENTS

In preparing this paper, the author is privileged to report the work of enthusiastic teams of designers and users led by R. H. Allmark and C. Robinson respectively. Most of the members of these groups have made significant original contributions and mention of individuals in such a team enterprise is out of place. The writer wishes to express his thanks to them and also to the Manager of the Data Processing Division of the English Electric Company, Ltd., for permission to publish this paper.

REFERENCES

1. S. J. M. Denison, E. N. Hawkins and C. Robinson: "DEUCE Alphacode." DEUCE Program News, No. 20, January 1958 (The English Electric Co. Ltd.).
2. R. A. Brooker: "The Autocode Programs developed for the Manchester University Computers." Computer Journal, 1, 1958, p. 15.
3. R. A. Brooker, B. Richards, E. Berg and R. Kerr: "The Manchester Mercury Autocode System." (University of Manchester, 1959).
4. FORTRAN Manual: (International Business Machines Corporation).
5. F. G. Duncan and E. N. Hawkins: "Pseudo-code Translation on Multilevel Storage Machines." Proceedings of the International Conference on Information Processing, p. 144 (UNESCO, Paris, June 1959).
6. F. G. Duncan and H. R. Huxtable: "The DEUCE Alphacode Translator." Computer Journal, 3, 1960, p. 98.
7. C. L. Hamblin: "GEORGE: A Semi-translation Programming Scheme for DEUCE." Programming and Operation Manual. University of New South Wales, Kensington, N.S.W.
8. R. H. Allmark and J. A. Lucking: "Design of an Arithmetic Unit Incorporating a Nesting Store." Proceedings of the I.F.I.P. Congress, Munich, August 1962.
9. B. Randell: "The Whetstone KDF.9 ALGOL Translator." Proceedings of the Programming Systems Symposium, London School of Economics, July 1962.
10. E. N. Hawkins and D. H. R. Huxtable: "A Multi-pass Translation Scheme for ALGOL 60 for KDF.9." Annual Review of Automatic Programming, Vol. 3, 1962, Pergammon Press.
11. F. G. Duncan: "Implementation of ALGOL 60 for KDF.9." Computer Journal, 5, 1962, 130.

A COMMON LANGUAGE FOR HARDWARE, SOFTWARE, AND APPLICATIONS

Kenneth E. Iverson
Thomas J. Watson Research Center, IBM
Yorktown Heights, New York

INTRODUCTION

Algorithms commonly used in automatic data processing are, when considered in terms of the sequence of individual physical operations actually executed, incredibly complex. Such algorithms are normally made amenable to human comprehension and analysis by expressing them in a more compact and abstract form which suppresses systematic detail. This suppression of detail commonly occurs in several fairly well defined stages, providing a hierarchy of distinct descriptions of the algorithm at different levels of detail. For example, an algorithm expressed in the FORTRAN language may be transformed by a compiler to a machine code description at a greater level of detail which is in turn transformed by the "hardware" of the computer into the detailed algorithm actually executed.

Distinct and independent languages have commonly been developed for the various levels used. For example, the operations and syntax of the FORTRAN language show little semblance to the operations and syntax of the computer code into which it is translated, and neither FORTRAN nor the machine language resemble the circuit diagrams and other descriptors of the processes eventually executed by the machine. There are, nevertheless, compelling reasons for attempting to use a single "universal" language applicable to all levels, or at least a single core language which may be slightly augmented in different ways at the various levels.

First, it is difficult, and perhaps undesirable, to make a precise separation into a small number of levels. For example, the programmer or analyst operating at the highest (least detailed) level, may find it convenient or necessary to revert to a lower level to attain greater efficiency in eventual execution or to employ convenient operations not available at the higher level. Programming languages such as FORTRAN commonly permit the use of lower levels, frequently of a lower level "assembly language" and of the underlying machine language. However, the employment of disparate languages on the various levels clearly complicates their use in this manner.

Second, it is not even possible to make a clear separation of level between the *software* (metaprograms which transform the higher level algorithms) and the hardware, since the hardware circuits may incorporate permanent or semipermanent memory which determines its action and hence the computer language. If this special memory can itself be changed by the execution of a program, its action may be considered that of software, but if the memory is "read-only" its action is that of hardware—leaving a rather tenuous distinction between software and hardware which is likely to be further blurred in the future.

Finally, in the design of a data processing system it is imperative to maintain close

communication between the programmers (i.e., the ultimate users), the software designers, and the hardware designers, not to mention the communication required among the various groups within any one of these levels. In particular, it is desirable to be able to describe the metaprograms of the software and the microprograms of the hardware in a common language accessible to all.

The language presented in Reference 1 shows promise as a universal language, and the present paper is devoted to illustrating its use at a variety of levels, from microprograms, through metaprograms, to "applications" programs in a variety of areas. To keep the treatment within reasonable bounds, much of the illustration will be limited to reference to other published material. For the same reason the presentation of the language itself will be limited to a summary of that portion required for microprogramming (Table 1), augmented by brief definitions of further operations as required.

MICROPROGRAMS

In the so-called "systems design" of a computer it is perhaps best to describe the computer at a level suited to the machine language programmer. This type of description has been explored in detail for a single machine (the IBM 7090) in Reference 1, and more briefly in Reference 2. Attention will therefore be restricted to problems of description on a more detailed level, to specialized equipment such as associative memory, and to the practical problem of keying and printing microprograms which arise from their use in design automation and simulation.

The need for extending the detail in a microprogram may arise from restrictions on the operations permitted (e.g., logical *or* and negation, but not *and*), from restrictions on the data paths provided, and from a need to specify the overall "control" circuits which (by controlling the data paths) determine the sequence in the microprogram, to name but a few. For example, the basic "instruction fetch" operation of fetching from a memory (i.e., a logical matrix) $\underline{M}$ the word (i.e., row) $\underline{M}^i$ selected according to the base two value of the instruction location register $\underline{s}$ (that is, $\underline{i} = \perp\underline{s}$), and transferring it to the command register $\underline{c}$, may be described as

$$\underline{c} \leftarrow \underline{M}^{\perp \underline{s}}.$$

Suppose, however, that the base two value operation (i.e., address decoding) is not provided on the register $\underline{s}$ directly, but only on a special register $\underline{a}$ to which s may be transferred. The fetch then becomes

$$\underline{a} \leftarrow \underline{s}$$

$$\underline{c} \leftarrow \underline{M}^{\perp \underline{a}}.$$

Suppose, moreover, that all communication with memory must pass through a buffer register $\underline{b}$, that each transfer out of a memory word $\underline{M}^i$ is accompanied by a subsequent reset to zero of that word (that is, $\underline{M}^i \leftarrow \overline{\epsilon}$), that every transfer from a register (or word of memory) $\underline{x}$ to a register (or word of memory) $\underline{y}$ must be of the form

$$\underline{y} \leftarrow \underline{x} \vee \underline{y},$$

and that any register may be reset to zero, then the instruction fetch becomes

1 $\underline{a} \leftarrow \overline{\epsilon}$

2 $\underline{a} \leftarrow \underline{s} \vee \underline{a}$

3 $\underline{b} \leftarrow \overline{\epsilon}$

4 $\underline{b} \leftarrow \underline{M}^{\perp \underline{a}} \vee \underline{b}$ }

5 $\underline{M}^{\perp \underline{a}} \leftarrow \overline{\epsilon}$ }

6 $\underline{M}^{\perp \underline{a}} \leftarrow \underline{b} \vee \underline{M}^{\perp \underline{a}}$

7 $\underline{c} \leftarrow \overline{\epsilon}$

8 $\underline{c} \leftarrow \underline{b} \vee \underline{c}.$

In this final form, the successive statements correspond directly (except for the bracketed pair 4 and 5 which together comprise an indivisible operation) to individual register-to-register transfers. Each statement can, in fact, be taken as the "name" of the corresponding set of data gates, and the overall control circuits need only cycle through a set of states which activate the data gates in the sequence indicated.

The sequence indicated in a microprogram such as the above is more restrictive than necessary and certain of the statements (such as 1 and 3 or 6 and 7) could be executed

concurrently without altering the overall result. Such overlapping is normally employed to increase the speed of execution of microprograms. The corresponding relaxation of sequence constraints complicates their specification, e.g., execution of statement k might be permitted to begin as soon as statements h, i and j were completed. Senzig (Reference 3) proposes some useful techniques and conventions for this purpose.

The "tag" portion of an associative memory can, as shown in Reference 2, be characterized as a memory $\underline{M}$, an argument vector $\underline{x}$, and a sense vector $\underline{s}$ related by the expression

$$\underline{s} = \underline{M} \triangleq \underline{x}$$

or by the euivalent expression

$$\underline{s} = \overline{\overline{\underline{M}} \overset{\vee}{\neq} \overline{\underline{x}}},$$

obtained by applying De Morgan's law. Falkoff (Reference 4) has used microprograms of the type discussed here in a systematic exploration of schemes for the realization of associative memories for a variety of functions including exact match, largest tag, and nearest larger tag.

Because the symbols used in the language have been chosen for their mnemonic properties rather than for compatibility with the character sets of existing keyboards and printers, transliteration is required in entering microprograms into a computer, perhaps for processing by a simulation or design automation metaprogram. For that portion of the language which is required in microprogramming, Reference 5 provides a simple and mnemonic solution of the transliteration problem. It is based upon a two-character representation of each symbol in which the second character need be specified but rarely. Moreover, it provides a simple representation of the index structure (permitting subscripts and superscripts to an arbitrary number of levels) based upon a Lukasiewicz or parenthesis-free representation of the corresponding tree.

METAPROGRAMS

Just as a microprogram description of a computer (couched at a suitable level) can provide a clear specification of the corresponding computer language, so can a program description of a compiler or other metaprogram give a clear specification of the "macro-language" which it accepts as input. No complete description of a compiler expressed in the present language has been published, but several aspects have been treated. Brooks and Iverson (Chapter 8, Reference 6) treat the SOAP assembler in some detail, particularly the use of the open addressing system and "availability" indicators in the construction and use of symbol tables, and also treat the problem of generators. Reference 1 treats the analysis of compound statements in compilers, including the optimization of a parenthesis-free statement and the translations between parenthesis and parenthesis-free forms. The latter has also been treated (using an emasculated form of the language) by Oettinger (Reference 7) and by Huzino (Reference 8); it will also be used for illustration here.

Consider a vector $\underline{c}$ representing a compound statement in complete* parenthesis form employing operators drawn from a set $\underline{p}$ (e.g., $\underline{p} = (+, \times, -, \div)$). Then Program 1 shows an algorithm for translating any well-formed statement $\underline{c}$ into the equivalent statement $\underline{l}$ in parenthesis-free (i.e., Lukasiewicz) form.

Program 1 - The components of $\underline{c}$ are examined, and deleted from $\underline{c}$, in order from left to right (steps 4, 5). According to the decisions on steps 6, 7, and 8†, each component is discarded if it is a left parenthesis, appended at the head of the resulting vector $\underline{l}$ if it is a variable (step 9), appended at the head of the auxiliary stack vector $\underline{s}$ if it is an operator (step 10), and initiates a transfer of the leading component of the stack $\underline{s}$ to the head of the result $\underline{l}$ if it is a right parenthesis (steps 11, 12). The behavior is perhaps best appreciated by tracing the program for a given case, e.g., if $\underline{c} = ([, [, x, +, y,], \times, [, p, +, q,],])$, then $\underline{l} = (\times, +, q, p, +, y, x)$.

APPLICATIONS

Areas in which the programming language has been applied include search and sorting procedures, symbolic logic, linear programming, information retrieval, and music

*In complete parenthesis form all implied parentheses are explicity included, e.g., the statement $((x + y) \times (p + q))$ is represented by $\underline{c} = ([, [, x, +, y,], \times, [, p, +, q,],])$.

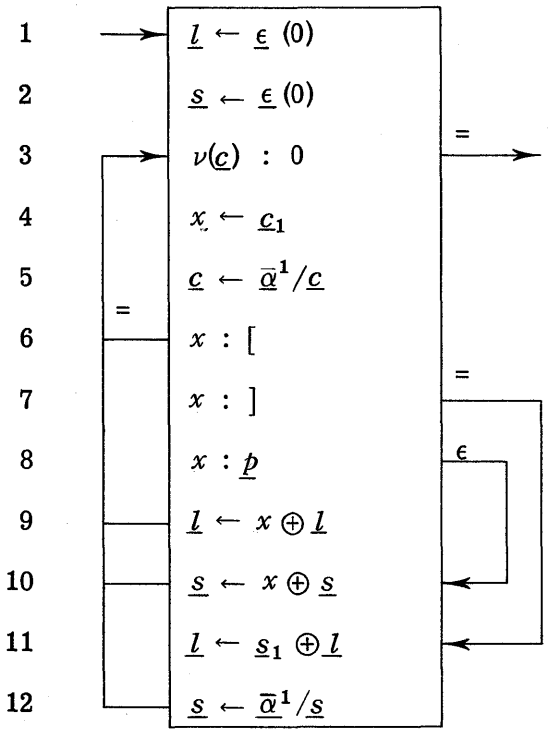
†A branch is followed if the relation with which it is labelled holds between the quantities separated by the colon. The relation "ϵ" denotes set membership.

Table 1

	Operation	Notation	Definition	Examples
OPERANDS	Scalar	a		$\underline{a} = (3, 4, 5, 6, 7)$, $\underline{b} = (8, 9)$, $\underline{c} = (3, 2, 1)$
	Vector	$\underline{a}$	$\underline{a} = (\underline{a}_0, \underline{a}_1, \ldots, \underline{a}_{\nu(\underline{a})-1})$	$\underline{p} = (1, 0, 1, 0, 1)$, $\underline{q} = (1, 0, 1)$
	Matrix	$\underline{A}$	$\underline{A} = \begin{pmatrix} \underline{A}^0 \\ \vdots \\ \underline{A}^{\mu(\underline{A})-1} \end{pmatrix} = (\underline{A}_0, \ldots \underline{A}_{\nu(\underline{A})-1})$ — $\underline{A}^i$ is i-th row vector; $\underline{A}_j$ is j-th column vector	$\underline{A} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \end{pmatrix}$ $\underline{P} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$
BASIC OPERATIONS	Floor	$k \leftarrow \lfloor x \rfloor$	$k \leq x < k+1$ (k, m, n, q integers)	$\lfloor 3.14 \rfloor = 3$, $\lfloor -3.14 \rfloor = -4$
	Ceiling	$k \leftarrow \lceil x \rceil$	$k \geq x > k-1$ (k, m, n, q integers)	$\lceil 3.14 \rceil = 4$, $\lceil -3.14 \rceil = -3$
	Residue mod m	$k \leftarrow m \mid n$	$n = mq + k$, $0 \leq k < m$ (k, m, n, q integers)	$7 \mid 19 = 5$, $7 \mid 21 = 0$, $7 \mid -3 = 4$
	And	$w \leftarrow u \wedge v$	$w = 1$ iff $u = 1$ and $v = 1$	
	Or	$w \leftarrow u \vee v$	$w = 1$ iff $u = 1$ or $v = 1$	
	Negation	$w \leftarrow \overline{u}$	$w = 1$ iff $u = 0$	$(3 \geq 2) = 0$, $(5 \neq 2) = 1$, $(i = j) = \delta_{ij}$,
	Proposition	$w \leftarrow (x R y)$	$w = 1$ iff x stands in relation R to y	$(u \neq v) =$ exclusive-or of u and v, $(u < v) = \overline{u} \wedge v$.
			All basic operations are extended component-by-component to vectors and matrices, e.g., $\underline{z} \leftarrow \underline{x} + \underline{y}$, $\underline{z} \leftarrow \underline{x} \times \underline{y}$, $\underline{W} \leftarrow \underline{U} \wedge \underline{V}$, $\underline{w} \leftarrow (\underline{x} \leq \underline{y})$	
SPECIAL ARRAYS	Full vector	$\underline{w} \leftarrow \underline{\epsilon}(n)$	$\underline{w}_i = 1$ (All 1's)	$\underline{\epsilon}(5) = (1, 1, 1, 1, 1)$, $\overline{\underline{\epsilon}} =$ zero vector
	Unit vector	$\underline{w} \leftarrow \underline{\epsilon}^j(n)$	$\underline{w}_i = (i = j)$ (1 in position j)	$\underline{\epsilon}^0(5) = (1, 0, 0, 0, 0)$, $\underline{\epsilon}^3(5) = (0, 0, 0, 1, 0)$
	Prefix vector	$\underline{w} \leftarrow \underline{\alpha}^j(n)$	$\underline{w}_i = (i < j)$ (j leading 1's)	$\underline{\alpha}^3(5) = (1, 1, 1, 0, 0)$, $\underline{\alpha}^2(4) = (1, 1, 0, 0)$
	Suffix vector	$\underline{w} \leftarrow \underline{\omega}^j(n)$	$\underline{w}_i = (i \geq n-j)$ (j final 1's)	$\underline{\omega}^3(5) = (0, 0, 1, 1, 1)$, $j \downarrow \underline{\omega}^j = \underline{\alpha}^j$, $j \uparrow \underline{\alpha}^j = \underline{\omega}^j$
	Infix vector	$\underline{w} \leftarrow i \downarrow \underline{\alpha}^j(n)$	See Rotation (j 1's after i 0's)	$2 \downarrow \underline{\alpha}^3(9) = (0, 0, 1, 1, 1, 0, 0, 0, 0)$
	Interval vector	$\underline{k} \leftarrow \underline{\iota}^j(n)$	$\underline{k}_i = i + j$ (Integers from j)	$\underline{\iota}^0(3) = (0, 1, 2)$, $\underline{\iota}^1(3) = (1, 2, 3)$, $\underline{\iota}^{-6}(4) = (-6, -5, -4, -3)$
			Of dimension $\nu(\underline{w}) = n$. The n may be omitted if clear from context	
	Full matrix	$\underline{W} \leftarrow \underline{E}(m \times n)$	$\underline{W}^i_j = 1$ (All 1's)	$\overline{\underline{E}}(m \times n) =$ zero matrix
	Identity matrix	$\underline{W} \leftarrow \underline{I}(m \times n)$	$\underline{W}^i_j = (i = j)$ (Diagonal 1's)	
			of dimension $m \times n$ (may be omitted)	

SELECTION	Compression	$z \leftarrow u/x$	z obtained from x by suppressing each x_i for which $u_i = 0$	$p/a = (3, 5, 7)$, $\overline{p}/a = (4, 6)$, $a^3/x = (x_0, x_1, x_2)$
	Row compression	$Z \leftarrow u/X$	$Z^i = u/X^i$	$p/A = \begin{pmatrix} 0 & 2 & 4 \\ 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$, $q//A = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 & 6 \end{pmatrix}$
	Column compression	$Z \leftarrow u//X$	$Z_j = u/X_j$	
	Mesh	$z \leftarrow \backslash x, u, y \backslash$	$\overline{u}/z = x$ and $u/z = y$ (Select in order from x or y as $u_i = 0$ or 1)	$\backslash b, p, c \backslash = (3, 8, 2, 9, 1)$
	Mask	$z \leftarrow /x, u, y/$	$\overline{u}/z = \overline{u}/x$ and $u/z = u/y$ ($z_i = x_i$ or y_i as $u_i = 0$ or 1)	$/a, p, A^0/ = (0, 4, 2, 6, 4)$
	Expansion	$z \leftarrow u \backslash y$	$\overline{u}/z = \overline{\epsilon}$ and $u/z = y$ (Mesh using zero vector for x)	$p \backslash c = (3, 0, 2, 0, 1)$
	Catenation	$z \leftarrow x \oplus y$	$z = (x_0, x_1, \cdots, x_{\nu(x)-1}, y_0, y_1, \cdots, y_{\nu(y)-1})$	$a \oplus b = (3, 4, 5, 6, 7, 8, 9)$, $x \oplus y = \backslash x, \omega^{\nu(y)}, y \backslash$
MISCELLANEOUS	Base 2 value	$z \leftarrow \perp u$	Value of u as a base 2 number	$\perp q = 5$, $\perp p = 21$
		$z \leftarrow \perp U$	$z_i = \perp U^i$	$\perp P = (3, 5)$
		$z \leftarrow \perp\!\!\perp U$	$z_j = \perp U_j$	$\perp\!\!\perp P = (1, 2, 3) = \iota^1(3)$
	Left rotation	$z \leftarrow k \uparrow x$	$z_i = x_j$, $j = \nu(x) \mid (i + k)$	$2 \uparrow a = (5, 6, 7, 3, 4)$, $5 \uparrow a = a$
	Right rotation	$z \leftarrow k \downarrow x$	$z_i = x_j$, $j = \nu(x) \mid (i - k)$	$1 \downarrow a = (7, 3, 4, 5, 6)$
	Reduction	$z \leftarrow \odot/x$	$z = (\ldots(x_0 \odot x_1) \odot x_2) \odot \ldots \odot x_{\nu-1})$	$+/p = 3$, $\times/c = 6$, $\wedge/p = 0$, $\neq/q = ((1 \neq 0) \neq 1) = (1 \neq 1) = 0 = 2 \mid +/q$
	Row reduction	$z \leftarrow \odot/X$	$z_i = \odot/X^i$	$+/A = (10, 15, 20)$, $\vee/P = (1, 1)$, $\neq/P = (0, 0)$
	Column reduction	$z \leftarrow \odot//X$	$z_j = \odot/X_j$	$+//A = (3, 6, 9, 12, 15)$, $+/(\neq//P) = 2$
	Matrix product	$z \leftarrow X \overset{\odot_1}{\odot_2} Y$	$Z^i_j = \odot_1/(X^i \odot_2 Y_j)$, where $\odot_1, \odot_2$ are binary operations or relations	$X \overset{+}{\times} Y$ is ordinary matrix product, $P \overset{+}{\times} A = \begin{pmatrix} 3, 5, 7, 9, 11 \\ 2, 4, 6, 8, 10 \end{pmatrix}$
				$c \overset{+}{\times} A = (4, 10, 16, 22, 28)$, $P \overset{+}{\times} q = (0, 1)$,
				$x \overset{+}{\times} y$ is ordinary scalar product, $b \overset{+}{\times} b = 145$
	Maximum prefix	$w \leftarrow \alpha/u$	w is the maximum length prefix (suffix) in u	$\alpha/(1, 1, 0, 1, 0) = (1, 1, 0, 0, 0)$, $\omega/(1, 1, 0, 1, 0) = (0, 0, 0, 0, 0)$,
	Maximum suffix	$w \leftarrow \omega/u$		$\alpha/p = (1, 0, 0, 0, 0)$, $\omega/p = (0, 0, 0, 0, 1)$,
				$\alpha/\alpha^j = \alpha^j$, $(+/\alpha/(x = \overline{\epsilon})) \uparrow x = x$ left justified
	Representation	$z \leftarrow \rho(x)$	z is the vector representation of the character x	In 7090 $\rho(d) = (0, 1, 0, 1, 0, 0)$, $\rho(e) = (0, 1, 0, 1, 0, 1)$
				In 8421 code, $\rho(0) = (0, 0, 0, 0)$, $\rho(1) = (0, 0, 0, 1)$

Basic Operations for Microprogramming (selected from Iverson, A Programming Language, Wiley, 1962)



PROGRAM 1

Translation from complete parenthesis statement $\underline{c}$ to equivalent Lukasiewicz statement $\underline{l}$

theory. The first two are treated extensively in Reference 1, and Reference 9 illustrates the application to linear programming by a 13-step algorithm for the simplex method. The areas of symbolic logic and matrix algebra illustrate particularly the utility of the formalism provided. Salton (Reference 10) has treated some aspects of information retrieval, making particular use of the notation for trees. Kassler's use in music concerns the analysis of Schoenberg's 12-tone system of composition (Reference 11).

Three applications will be illustrated here: search procedures, the relations among the canonical forms of symbolic logic, and matrix inversion.

Search Algorithms. Figure 1 shows search programs and examples (taken from Reference 1) for five methods of "hash addressing" (cf. Peterson, Reference 12), wherein the functional correspondent of an agrument x is determined by using some *key transformation* function t which maps each argument x into an integer i in the range of the indices of some table (i.e., matrix) which contains the arguments and their correspondents. The key transformation t is, in general, a many-to-one function and the index i is merely used as the starting point in searching the table for the given argument x. Figure 1 is otherwise self-explanatory. Method (e) is the widely used open addressing system described by Peterson (Reference 12).

Symbolic Logic. If $\underline{x}$ is a logical vector and $\underline{T}$ is a logical matrix of dimension $2^{\nu(\underline{x})} \times \nu(\underline{x})$ such that the base two value of $\underline{T}^i$ is i (that is, $\perp \underline{T} = \iota_0(2^{\nu(\underline{x})})$), then the rows of $\underline{T}$ define the domain of the argument $\underline{x}$, and any logical function $f(\underline{x})$ defined on $\underline{x}$ can be completely specified by the *intrinsic vector* $\underline{i}(f)$ such that $\underline{i}_j(f) = f(\underline{T}^j)$.

Expansion of the expression $\underline{p} = \underline{T} \triangleq \underline{x}$ shows that $\underline{p}$ is the vector of minterms in $\underline{x}$, and consequently

$$f(\underline{x}) = \underline{\gamma}(f,\vee) \underset{\wedge}{\vee} \underline{p} = \underline{\gamma}(f,\vee) \underset{\wedge}{\vee} (\underline{T} \triangleq \underline{x}),$$

where $\underline{\gamma}(f,\vee)$ denotes the characteristic vector of the function f in the disjunctive canonical form. Formally, $\underline{\gamma}(f,\vee)$ may be defined by the relation

$$\underline{i}(f) = \underline{\gamma}(f,\vee) \underset{\wedge}{\vee} (\underline{T} \overset{\triangle}{\neq} \tilde{\underline{T}}), \qquad (1)$$

where $\tilde{\underline{T}}$ denotes the transpose of $\underline{T}$. It is easily shown that $\underline{T} \overset{\triangle}{\neq} \tilde{\underline{T}}$ is equal to the identity matrix, and hence that $\underline{\gamma}(f,\vee) = \underline{i}(f)$.

In a similar manner, the expression for the exclusive disjunctive canonical form may be written (Reference 1, Chapter 7) as

$$\underline{i}(f) = \underline{\gamma}(f, \neq) \underset{\wedge}{\neq} (\underline{T} \underset{\diamond}{\triangle} \tilde{\underline{T}}), \qquad (2)$$

and the relation between the intrinsic vector and the exclusive disjunctive characteristic vector $\underline{\gamma}(f, \neq)$ (first derived by Muller, Reference 14) is given directly by the square matrix $S = T \underset{\diamond}{\triangle} \tilde{T}$. The properties of the matrix S are easily derived from this formulation. Moreover, the formal application of De Morgan's laws to equations (1) and (2) yields the two remaining canonical forms directly (Reference 1, Chapter 7).

Matrix Inversion. The method of matrix inversion using Gauss-Jordan (complete) elimination with pivoting and restricting the total major storage to a single square matrix augmented by one column (described in References 14 and 15) involves enough selection, permutation, and decision type operations to render its complete description by classical

k = (Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday)

$t(k_i) = 1 + (6 \mid_0 n_i)$, where

n = (19, 13, 20, 23, 20, 6, 19)

(n_i is the rank in the alphabet of the first letter of k_i)

z = (2, 2, 3, 6, 3, 1, 2), where $z_i = t(k_i)$,

d = (1, 2, 3, 6), and $s = 6$.

Data of examples

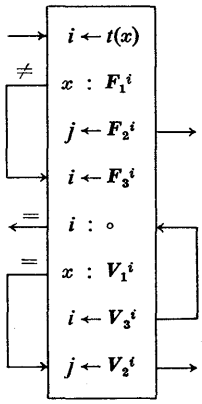
1	Friday	6	∘
2	Sunday	1	1
3	Tuesday	3	2
4	∘	∘	∘
5	∘	∘	∘
6	Wednesday	4	∘

$F =$

1	Monday	2	3
2	Thursday	5	∘
3	Saturday	7	∘

$V =$

Overflow with chaining
(*b*)



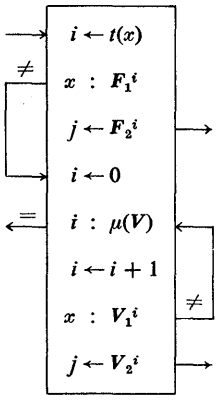
1	Friday	6	∘
2	Sunday	1	1
3	Tuesday	3	3
4	∘	∘	∘
5	∘	∘	∘
6	Wednesday	4	4

$F =$

1	Monday	2	∘
2	Thursday	5	∘
3	Saturday	7	∘

$V =$

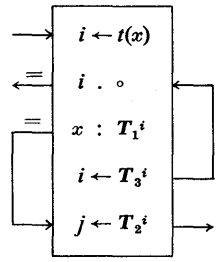
Overflow
(*a*)



1	Friday	6	∘
2	Sunday	1	4
3	Tuesday	3	5
4	Monday	2	7
5	Thursday	5	∘
6	Wednesday	4	∘
7	Saturday	7	∘

$T =$

Single table with chaining
(*c*)

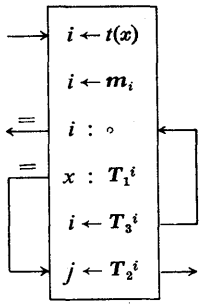


1	Sunday	1	2
2	Monday	2	7
3	Tuesday	3	5
4	Wednesday	4	∘
5	Thursday	5	∘
6	Friday	6	∘
7	Saturday	7	∘

$T =$

m = (6, 1, 3, ∘, ∘, 4)

Single table with chaining and mapping vector
(*d*)



1	Friday	6	
2	Sunday	1	
3	Monday	2	
4	Tuesday	3	
5	Thursday	5	
6	Wednesday	4	
7	Saturday	7	

$T =$

Open addressing system-construction and use of table
(*e*)

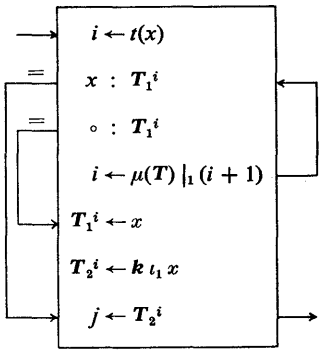


Figure 1. Programs and examples for methods of scanning equivalence classes defined by a 1-origin key transformation t

matrix notation rather awkward. Program 2 describes the entire process. The starred statements perform the pivoting operations and their omission leaves a valid program without pivoting; exegesis of the program will first be limited to the abbreviated program without pivoting.

Program 2 - Step 1 initializes the counter k which limits (step 11) the number of iterations and step 3 appends to the given square matrix $\underline{M}$ a final column of the form (1, 0, 0, ..., 0) which is excised (step 12) only after completion of the main inversion process. Step 7 divides the pivot (i.e., the first) row by the pivot element, and step 8 subtracts from $\underline{M}$ the outer product of its first row (except that the first element is replaced by zero) with its first column.* The result of steps 7 and 8 is to reduce the first column of $\underline{M}$ to the form (1, 0, 0, ..., 0) as required in Jordan elimination.

```
    1   k ← ν(M)
*   2   p ← ι⁻¹(k)
    3   M ← M ⊕ α¹
*   4   j ← ((α^k ⌈| M₁ |)/ι⁻¹)₁
*   5   p₁ ↔ pⱼ
*   6   ω¹/M¹ ↔ ω¹/M^j
    7   M¹ ← M¹ ÷ M¹₁ε
    8   M ← M - (ω¹ × M₁) ∘× M¹
    9   M ← ε ↑ (ω¹ ⇑ M)
*  10   p ← 1 ↑ p
   11   k ← k - 1
   12   M ← ω¹/M
*  13   M ← M_p ι ι⁻¹
```

PROGRAM 2

Matrix inversion by Gauss-Jordan elimination

*The outer product $\underline{Z}$ of vector $\underline{x}$ by vector $\underline{y}$ is the "column by row product" denoted by $\underline{Z} \leftarrow \underline{x} \overset{\circ}{\times} \underline{y}$ and defined by $\underline{Z}^i_j = \underline{x}_i \times \underline{y}_j$.

The net result of step 9 is to bring the next pivot row to first position and the next pivot element to first position within it by (1) cycling the old pivot row to last place and the remaining rows up by one place, and (2) cycling the leading column (1, 0, 0, ..., 0) to last place (thus restoring the configuration produced by step 3) and the remaining columns one place to the left. The column rotation* $\underline{\omega}^1 \Uparrow \underline{M}$ rotates all columns save the first upward by one place, and the subsequent row rotation $\underline{\epsilon} \uparrow$ rotates all rows to the left by one place.

The pivoting provided by steps 2, 4, 5, 6, 10, and 13 proceeds as follows. Step 4 determines the index j of the next pivot row by selecting the maximum† over the magnitudes of the first k elements of the first column, where $k = \nu(\underline{M}) + 1 - q$ on the q-th iteration. Step 6 interchanges the first row with the selected pivot row, except for their final components. Step 5 records the interchange in the permutation vector‡ $\underline{p}$ which is itself initialized (step 2) to the value of the identity permutation vector $\underline{\iota}^1(k) = (1, 2, \ldots, k)$. The rotation of $\underline{p}$ on step 10 is necessitated by the corresponding rotations of the matrix $\underline{M}$ (which it indexes) on step 9. Step 13 performs the appropriate inverse reordering§ among the columns of the resulting inverse $\underline{M}$.

*The row rotation $\underline{k} \uparrow \underline{X}$ is a row-by-row extension of the rotation $k \uparrow x$, that is, $(\underline{k} \uparrow \underline{X})^i = \underline{k}_i \uparrow \underline{X}^i$. Similarly, $(\underline{k} \Uparrow \underline{X})_j = \underline{k}_j \uparrow \underline{X}_j$.

†The expression $\underline{u} = \underline{v} \lceil \underline{x}$ denotes a logical vector $\underline{u}$ such that $\underline{u}_i = 1$ if and only if $\underline{x}_i$ is a maximum among those components $\underline{x}_k$ such that $\underline{v}_k = 1$. Clearly, $\underline{u}/\underline{\iota}^1$ is the vector of indices of the (restricted) maxima, and $(\underline{u}/\underline{\iota}^1)_1$ is the first among them.

‡A permutation vector is any vector $\underline{p}$ whose components take on all the values 1, 2, ..., $\nu(\underline{p})$. The expression $\underline{y} = \underline{x}_{\underline{p}}$ denotes that $\underline{y}$ is a permutation of $\underline{x}$ defined by $\underline{y}_i = \underline{x}_{\underline{p}i}$.

§Permutation is extended to matrices as follows:

$$\underline{N} = \underline{M}_{\underline{p}} \Longleftrightarrow \underline{N}_j = \underline{M}_{\underline{p}j} ,$$

$$\underline{L} = \underline{M}^{\underline{p}} \Longleftrightarrow \underline{L}^i = \underline{M}^{\underline{p}j} .$$

The expression $\underline{a} \iota x$ is called the $\underline{a}$ index of x and is defined by the relation $\underline{a} \iota x = j$, where $x = \underline{a}_j$. If $\underline{x}$ is a vector, then $\underline{j} = \underline{a} \iota \underline{x}$ is defined by $j_i = \underline{a} \iota \underline{x}_i$. Consequently, $\underline{p} \iota \underline{\iota}^1$ denotes the permutation inverse to $\underline{p}$.

A description in the ALGOL language of matrix inversion by the Gauss-Jordan method is provided in Reference 16.

REFERENCES

1. Iverson, K. E., A Programming Language, Wiley, 1962.
2. Iverson, K. E., "A Programming Language," Spring Joint Computer Conference, San Francisco, May 1962.
3. Senzig, D. N., "Suggested Timing Notation for the Iverson Notation," Research Note NC-120, IBM Corporation.
4. Falkoff, A. D., "Algorithms for Parallel-Search Memories," J.A.C.M., October 1962.
5. Iverson, K. E., "A Transliteration Scheme for the Keying and Printing of Microprograms," Research Note NC-79, IBM Corporation.
6. Brooks, F. P., Jr., and Iverson, K. E., "Automatic Data Processing," Wiley (in press).
7. Oettinger, A. G., "Automatic Syntactic Analysis of the Pushdown Store," Proceedings of the Twelfth Symposium in Applied Mathematics, April 1960, published by American Mathematical Society, 1961.
8. Huzino, S., "On Some Applications of the Pushdown Store Technique," Memoirs of the Faculty of Science, Kyushu University, Ser. A, Vol. XV, No. 1, 1961.
9. Iverson, K. E., "The Description of Finite Sequential Processes," Fourth London Symposium on Information Theory, August 1960, Colin Cherry, Ed., Butterworth and Company.
10. Salton, G. A., "Manipulation of Trees in Information Retrieval," Communications of the ACM, February 1961, pp. 103-114.
11. Kassler, M., "Decision of a Musical System," Research Summary, Communications of the ACM, April 1962, page 223.
12. Peterson, W. W., "Addressing for Random Access Storage," IBM Journal of Research and Development, Vol. 1, 1957, pp. 130-146.
13. Muller, D. E., "Application of Boolean Algebra to Switching Circuit Design and to Error Detection," Transactions of the IRE, Vol. EC-3, 1954, pp. 6-12.
14. Iverson, K. E., "Machine Solutions of Linear Differential Equations," Doctoral Thesis, Harvard University, 1954.
15. Rutishauser, H., "Zur Matrizeninversion nach Gauss-Jourdan," Zeitschrift fur Angewandte Mathematik und Physik, Vol. X, 1959, pp. 281-291.
16. Cohen, D., "Algorithm 58, Matrix Inversion," Communications of the ACM, May 1961, p. 236.

INTERCOMMUNICATING CELLS, BASIS FOR A DISTRIBUTED LOGIC COMPUTER

C. Y. Lee
Bell Telephone Laboratories, Inc.
Holmdel, New Jersey

The purpose of this paper is to describe an information storage and retrieval system in which logic is distributed throughout the system. The system is made up of cells. Each cell is a small finite state machine which can communicate with its neighboring cells. Each cell is also capable of storing a symbol.

There are several differences between this cell memory and a conventional system. With logic distributed throughout the cell memory, there is no need for counters or addressing circuitry in the system. The flow of information in the cell memory is to a large extent guided by the intercommunicating cells themselves. Furthermore, because retrieval no longer involves scanning, it becomes possible to retrieve symbols from the cell memory at a rate independent of the size of the memory.

Information to be stored and processed is normally presented to the cells in the form of strings of symbols. Each string consists of a name and an arbitrary number of parameters. When a name string is given as its input, the cell memory is expected to give as its output all of the parameter strings associated with the name string. This is called direct retrieval. On the other hand, given a parameter string, the cell network is also expected to give as its output the name string associated with that parameter string. This is called cross-retrieval.

The principal aim of our design is a cell memory which satisfies the following criteria:

1. The cells are logically indistinguishable from each other.
2. The amount of time required for direct retrieval is independent of the size of the cell memory.
3. The amount of time required for cross-retrieval is independent of the size of the cell memory.
4. There is a simple uniform procedure for enlarging or reducing the size of the cell memory.

Aim and Motivation

We are primarily concerned here with the design of the memory system of a computer in which memory and logic are closely interwoven. The motivation stems from our contention that in the present generation of machines the scheme of locating a quantity of information by its "address" is fundamentally a weak one, and furthermore, the constraint that a memory "word" may communicate only with the central processor (in most cases the accumulator) has no intrinsic appeal. This motivation led us to the design of a cell memory compatible with these contentions.

The association of an address with a quantity of information is very much the result of the type of computer organization we now have. Writing a program in machine language, one rarely deals with the quantities of information themselves. A programmer normally must know where these quantities

of information are located. He then manipulates their addresses according to the problem at hand. An address in this way often assumes the role of the name of a piece of information.

There are two ways to look at this situation. Because there is usually an ordering relationship among addresses, referring to contents by addresses has its merits provided a programmer has sufficient foresight at the time the contents were stored. On the other hand, a location other than being its address can have but the most superficial relation to the information which happens to be stored there. The assignment of a location to a quantity of information is therefore necessarily artificial. In many applications the introduction of an address as an additional characteristic of a quantity of information serves only to compound the complexity of the issue. In any event the assignment of addresses is a local problem, and as such should not occupy people's time and may even be a waste of machine's time.

A macroscopic approach to information storage and retrieval is to distinguish information by its attributes. A local property, "350 Fifth Avenue," means little to most people. The attribute, "the tallest building in the world," does. The macroscopic approach requires only that we be able to discern facts by contents. Whatever means are needed for addressing, for counting, for scanning and the like are not essential and should be left to local considerations.

Doing away with addressing, counting, and scanning means a different approach to machine organization. The underlying new concept is however simple and direct: The work of information storage and retrieval should not be assumed by a central processor, but should be shared by the entire cell memory. The physical implementation of this concept is intercommunicating cells.

Although one of the principal aims of an intercommunicating cell organization is to make the rate of retrieval independent of the amount of information stored in the cell memory, a number of other engineering criteria are no less important. Uniformity of cell design makes mass production possible. Ease of attaching or detaching cells from the cell memory simplifies the growth problem. Also, facilities for simultaneous matching of symbols make complex preprocessing such as sorting unnecessary.

A few intuitive remarks on cell memory retrieval may be appropriate here. Information stored in the cell memory are in the form of strings of symbols. Normally, such a string is made up of a name and the parameters which describe its attributes. Each cell is capable of storing one symbol. A string of information is therefore stored in a corresponding string of cells. In the cell memory each cell is given enough logic circuitry so that it can give us a yes or no answer to a simple question we ask. If we think of the symbol, say s, contained in a cell as its name, then the question we ask is merely whether the cell's name is s or is not s.

In retrieval, for example, we may wish to find all of the parameters (attributes) of a strategy whose name is XYZ. As a first step we would simultaneously ask each cell whether its name is X. If a cell gives us an answer no, then we are no longer interested in that cell. If a cell gives us an answer yes, however, we know it may lead us to the name of the strategy we are looking for. Therefore, we also want each cell to have enough logic circuitry so that it can signal its neighboring cell to be ready to respond to the next question. We then simultaneously ask each of these neighboring cells whether its name is Y. Those cells whose names are Y in turn signal their neighboring cells to be ready to respond to the final question: whether a cell's name is Z. The cell which finally responds is now ready to signal its nearest neighbor to begin giving out parameters of the strategy.

The process of cell memory retrieval provides a particularly good example of letting the cells themselves guide the flow of information. By a progressive sequence of questions, we home in on the information we are looking for in the cell memory, although we have no idea just where the information itself is physically stored. Because generally most cells contain information which is of no use to us, the number of cells which give yes answers at any moment is quite small. We may, if we wish, think of the retrieving process therefore as a process of getting rid of useless information rather than a searching process for the useful information.

Cell Configuration

Each cell in the cell memory is made up of components called cell elements. Each cell element is a bistable device such as a

relay or a flip-flop. The cell elements are divided into two kinds: the cell state elements and the cell symbol elements. In the design of the cell memory to be described here, each cell will have a single cell state element so that each cell has two logical states. A cell may either be in an active state or in a quiescent state. There may be any number of cell symbol elements, depending upon the size of the symbol alphabet.

The over-all structure of a cell memory is shown in Figure 1. Each cell in the cell memory, say cell i, is controlled by four types of control leads: the input signal lead (IS lead), the output signal lead (OS lead), the match signal lead (MS lead), and the propagation lead (P lead). The input signal lead is active for the duration of the input process. The input symbol itself is carried on a separate set of input leads. When a cell is in an active state, and the input signal lead is activated, whatever symbol is carried on the input leads is then stored in that cell.

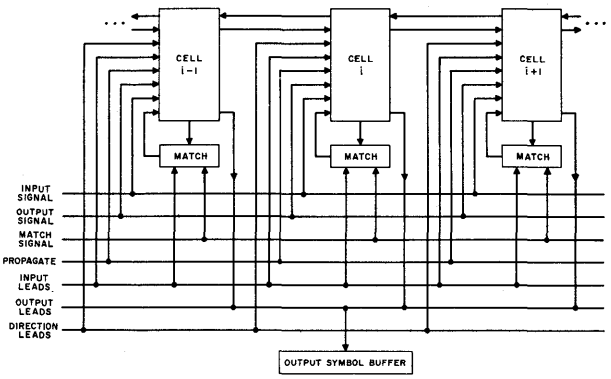


Figure 1. Overall diagram of the cell memory.

The output signal lead controls the flow of output from the cell memory. Each symbol read out from a cell is carried by a separate set of output leads, and is also stored in a buffer called the Output Symbol Buffer. When a cell is in an active state, a pulse on the output signal lead causes that cell to read out its contents to the set of output leads.

An important function performed by the cell memory is the operation of simultaneous matching of the contents of each of the cells with some fixed contents. This operation is controlled by the match signal lead. During matching, the contents of each cell, say cell i, is compared with the contents carried on the input leads. If the comparison is successful, an internal signal, m_i, is generated in cell i. The signal, m_i, is transmitted to one of the neighboring cells of cell i, causing that cell to become active.

The propagation lead controls the propagation of activity in a cell memory. When a cell is in an active state, a pulse on the propagation lead causes it to pass its activity to one of its neighboring cells. The direction of propagation is controlled by two separate direction leads: R and L.

The circuits of a cell employing flip-flops and gates are shown in Figure 2.

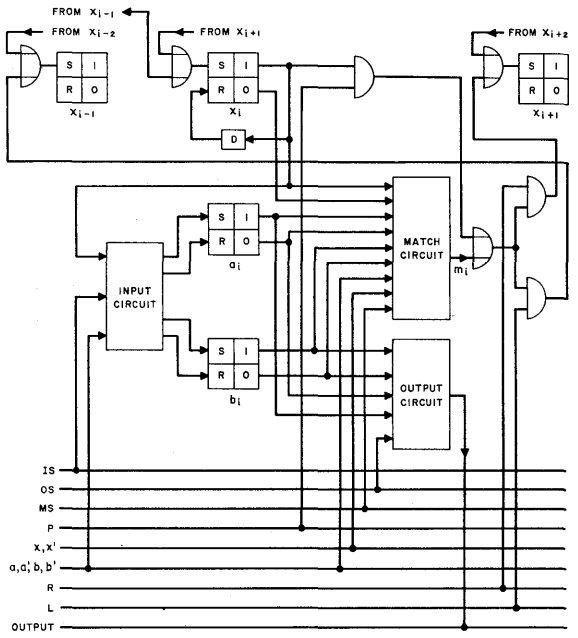


Figure 2. More detailed circuit of cell i.

An Example of Cross-Retrieval

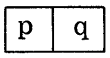
Let there be three separate strings of information stored in the cell memory. Let these strings have the following names and parameters:

Name	Parameter
B	XY
AB	XW
AC	U

The strings are stored in the cell memory in the form of a single composite string. There must be some way, therefore, by which the name and the parameter strings can be told apart and also a means for distinguishing among the three strings of information themselves. To do this we introduce two tag symbols, α and β. Every name string is preceded by a tag of α, and every parameter string is preceded by a name of β. The string stored in the cell memory therefore has the form:

$$\alpha\ B\ \beta\ X\ Y\ \alpha\ A\ B\ \beta\ X\ W\ \alpha\ A\ C\ \beta\ U\ \alpha\ \ldots$$

We have found it convenient to use the diagram



to represent a cell. In this diagram, p stands for the symbol stored in the cell, and q for the state of the cell. Also, q is 1 if the cell is quiescent, and is 2 if the cell is active. Using such diagrams, Figure 3 shows the manner in which the composite string is stored in the cell memory.

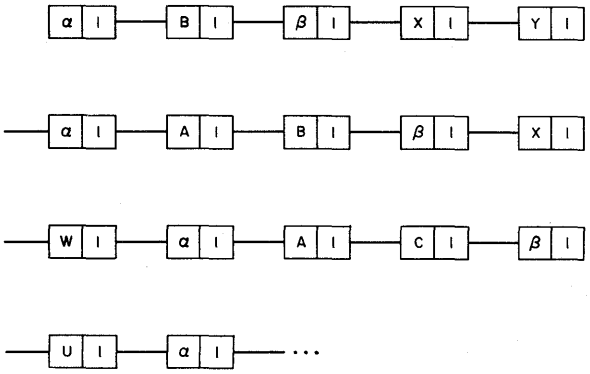
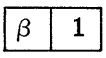


Figure 3. Cell memory configuration at the start of retrieval.

Let us suppose that we wish to retrieve the name of a string whose parameter is XW. We call such a process in which a parameter string input causes a name string output the process of cross-retrieval. The process of retrieving a parameter knowing its name is called direct retrieval.

Initially, we want all of the cells to match their individual contents against the fixed information



Furthermore, we want every cell whose contents happen to be $\boxed{\beta}\boxed{1}$ to send a signal to the neighboring cell on its right. We then have the situation shown in Figure 4, where each arrow indicates that a signal is being transmitted by a cell whose contents are $\boxed{\beta}\boxed{1}$.

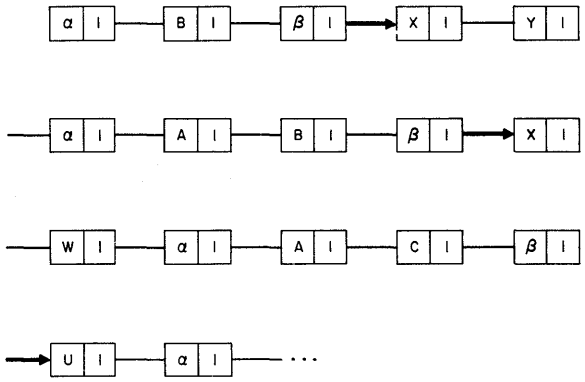


Figure 4. Signals being transmitted by $\boxed{\beta}\boxed{1}$ after matching.

Now every cell which receives a signal from its neighboring cell, whether from the left or from the right, will change from a quiescent state to an active state. Also, the signals transmitted by the cells to their neighbors should be thought of as pulses so that they disappear after they have caused the neighboring cells to become active. The next stable situation is shown in Figure 5; each of the active cells is represented by double boundary lines.

During the next match cycle, we want all of the cells to match their individual contents against the fixed contents:

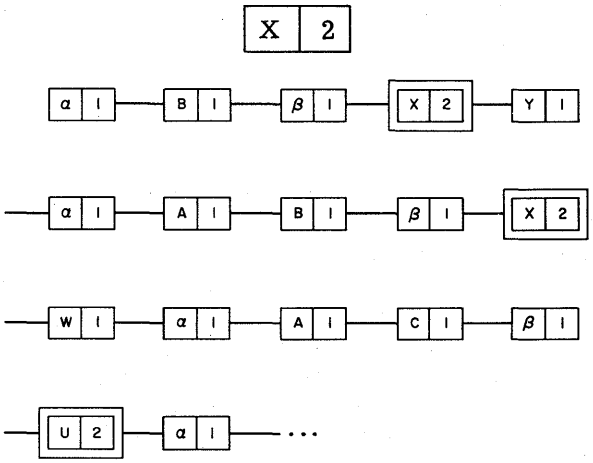


Figure 5. Some of the cells become active after receiving pulses from the neighboring cells.

As before, every cell whose contents are [X | 2] sends a signal to its right neighboring cell, causing that cell to become active. At this point, each previously active cell is made to restore itself to the quiescent state. The stable situation is illustrated in Figure 6.

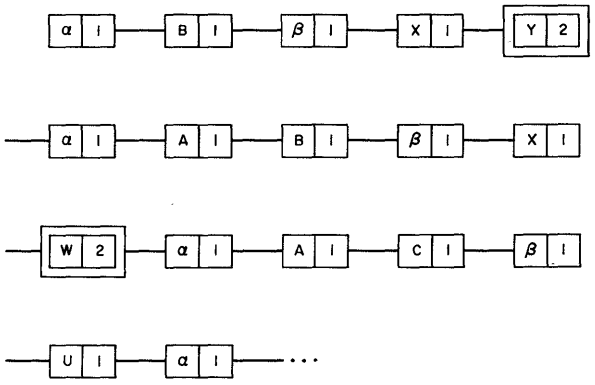


Figure 6. Previously active cells restore themselves to quiescent state as new cells become active.

During the following match cycle, the cells are made to match their individual contents against the fixed contents

W	2

In this example, there is now only one cell whose contents are [W | 2]. That cell first signals the cell on its right, causing that neighboring cell to become active, and then restores itself to the quiescent state.

During the next match cycle, all of the cells are made to match against

α	2

The presence of the symbol α shows that the matching process is at an end, and that the output part of the retrieval process is to begin. The cell whose contents are [A | 2] is the only cell which is active at the moment.

During the output phase, a number of actions take place. First of all, two successive propagate-left signals are sent to the cell memory. The result is a transfer of activity from the cell whose contents are [A | 2] to the cell whose contents are [W | 2], as shown in Figure 7. An output signal is now supplied to all of the cells. The cell which is active

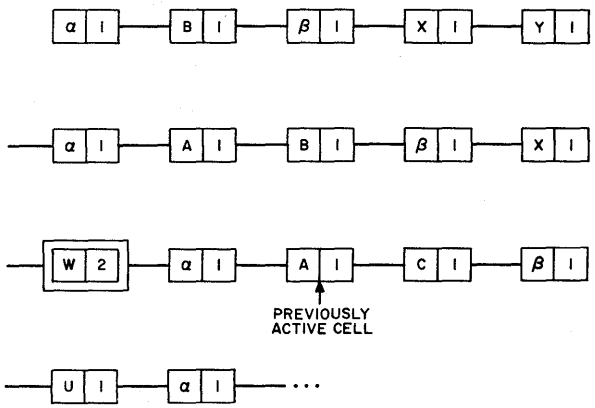


Figure 7. Transfer of activity from cell [A | 2] to cell [W | 2]

then reads out its symbol to the output leads. That symbol is now compared with the fixed symbol α in an external match circuit associated with the control leads. If there is no match, a propagate-left signal is sent to every cell and the external comparison process is repeated. This process eventually terminates when the cell [α | 2] is reached (Figure 8). The purpose of this phase of the output process is strictly to locate the beginning of the information string which is being retrieved.

The actual read out process begins with a propagate-right signal. The cell [A | 2] which contains the first letter of the name string AB is activated. The symbol A is now read out and matched externally with the symbol β. The read out process continues until β is reached and is then terminated.

Figure 9 gives a general picture of the read out part of the cross-retrieval process.

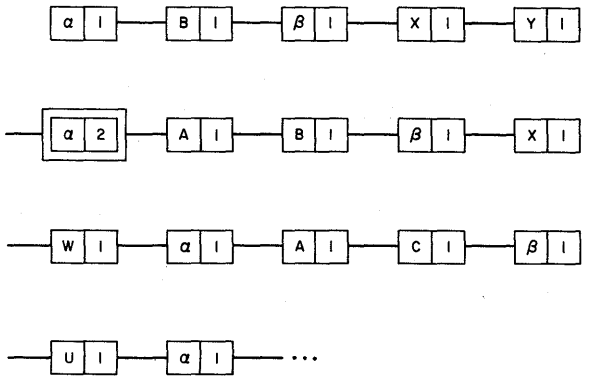


Figure 8. Reaching the initial symbol in the string to be retrieved.

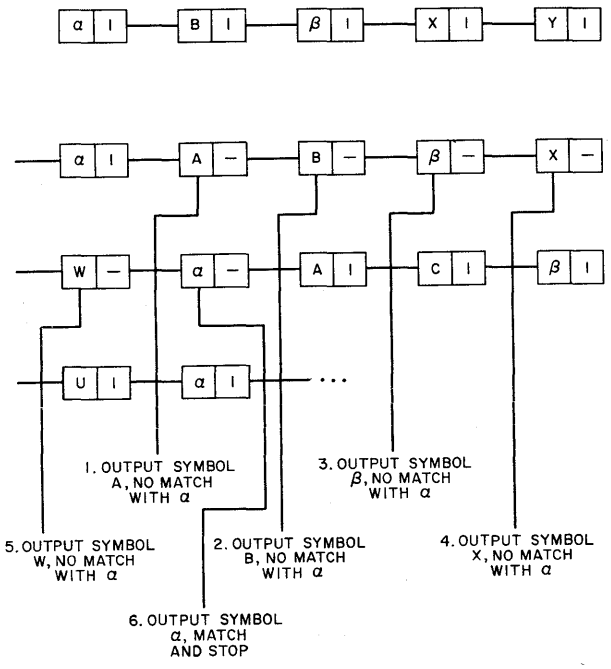


Figure 9. A general picture of the read-out part of the cross-retrieval process.

Storage and Cross-Retrieval

The storage and the retrieval of symbols in the cell memory are both accomplished by letting the cells pass their activities to their neighboring cells and, in this way, guide the flow of information in the cell memory. The process of storing symbols in the cell memory provides a good illustration of the dependence of the cell memory upon the propagation of activity among the cells.

Prior to storing the first symbol, the first cell in the cell memory is made active. When the first symbol appears on the set of input leads, the first cell, being active, becomes the only cell prepared to receive that symbol. The first cell, like all of the cells, plays a dual role however. After taking in the symbol, it also passes its activity to the right neighboring cell. The neighboring cell then becomes the only active cell in the cell memory, and hence becomes the only cell prepared to receive the next symbol when it appears on the input leads.

When we examine the many kinds of information strings which make retrieval difficult, we find that a string is much more likely to have several parameters rather than a single parameter. For such strings the tag system which we have used in the example in the last section is inadequate. If a string consists of a name N and k-1 parameters $P_1, P_2, \ldots, P_{k-1}$, we will now assign to it a set of k+2 tags: $\alpha_0, \alpha_1, \ldots, \alpha_k$, and β. The string will be stored in the cell memory in the following form:

$$\ldots \alpha_0\ \alpha_1\ N \beta \alpha_2\ P_1 \beta \alpha_3\ P_2 \beta \ldots$$

$$\alpha_k\ P_{k-1}\ \beta \alpha_0 \ldots$$

The symbol α_0 indicates the beginning of a string, and the symbol β indicates the end of a component (that is, a name or a parameter). The symbols $\alpha_1, \alpha_2, \ldots, \alpha_k$ are the tags associated with the components N, $P_1, \ldots, P_{k-1}$. Furthermore, it should be noted that a tag is associated always with a given attribute. For example, α_1 is the name tag and should be used as a name tag for all information strings.

Consider now the cross-retrieval problem where the cell memory is given as its input a component string together with its tags:

$$\alpha_j\ P_{j-1}\ \beta$$

The cell memory, for the purpose of cross-retrieval, must give as its output the entire string:

$$\alpha_1\ N \beta \alpha_2 P_1\ \beta \ldots \alpha_k\ P_{k-1}\ \beta$$

In describing a procedure for cross-retrieval, we shall assume for the purpose of this presentation that every component stored in the cell memory is unique. This means that if an input string $\alpha_j\ P_{j-1}\ \beta$ is presented to the cell memory, we can be sure that either (1) there is no string in the cell memory which has $\alpha_j\ P_{j-1}\ \beta$ as one of its components, or (2) there is exactly one string in the cell memory which has $\alpha_j\ P_{j-1}\ \beta$ as one of its components. Under this assumption therefore, no two strings could compete with each other during retrieval.

The basic cross-retrieval procedure is the following. The string $\alpha_j\ P_{j-1}\ \beta$ is first matched with all of the strings stored in the cell memory. When a match has occurred, the cell in which the symbol α_{j+1} is stored (α_{j+1} being, in this case, the symbol immediately following $\alpha_j\ P_{j-1}\ \beta$ in the cell memory) would be activated. Because $\alpha_j\ P_{j-1}\ \beta$ is unique in the cell memory, the cell in which the symbol α_{j+1} is stored becomes the only active cell in the cell memory. This

activity is now propagated towards the left until the symbol α_0, which is the beginning of the string, is reached. Symbols are then retrieved from the cell memory to the right until finally the symbol α_0, which is the beginning of the next string, is reached.

Outlook

We wanted to present here the basic ideas of a distributed logic system without going into many related problems and other technical considerations. The most obvious asset of such an organization is the tremendous speed it offers for retrieval. Suitable programs can also be developed to make the organization extremely flexible. In addition, we believe the macroscopic concept of logical design, away from scanning, from searching, from addressing, and from counting, is equally important. We must, at all cost, free ourselves from the burdens of detailed local problems which only befit a machine low on the evolutionary scale of machines.

On the other hand, the emphasis on distributed logic introduces a number of physical problems. If a cell memory is to be practically useful, it must have many thousands, or perhaps millions of cells. Each cell must therefore be made of physical components which are less than miniature in size, and which must each consume extremely tiny amounts of power. Furthermore, because the cells are all identical, mass production techniques should be developed in which a whole block of circuitry can be formed at once.

Because the coordination of vast amounts of information is essential to scientific, economic, and military progress, the type of organization exemplified by the cell memory needs to be explored and explored extensively. The research on machine organization, however, cannot stand alone; the success of this research will depend also on the success in other fields of research: microminiaturization, integrated logic, and hyper-reliable circuit design.

ACKNOWLEDGEMENT

The writer is indebted to Messrs. S. H. Washburn, C. A. Lovell, and W. Keister for their support in this work. The writer also wishes to acknowledge the benefit he received from many discussions with Mr. M. C. Paull and with his other colleagues.

REFERENCES

1. Albert E. Slade, The Woven Cryotron Memory, Proc. Int. Symp. on the Theory of Switching, Harvard Univ. Press, 1959, p. 326.
2. R. R. Seeber and A. B. Linquist, Associative Memory with Ordered Retrieval, IBM Jour. of Res. and Dev., 5, 1962, p. 126.
3. R. S. Barton, A New Approach to the Functional Design of a Digital Computer, Proc. of the Western Joint Computer Conference, May 9 to 11, 1961, p. 393.
4. S. H. Unger, A New Type of Computer Oriented Towards Spatial Problems, Proc. of the IRE, 46, 1958, p. 1744.
5. P. M. Davis, A Superconductive Associative Memory, Proc. Spring Joint Computer Conference, May 1 to 3, 1962, p. 79.
6. V. L. Newhouse and R. E. Fruin, A Cryogenic Data Address Memory, Proc. Spring Joint Computer Conference, May 1 to 3, 1962, p. 89.
7. J. W. Crichton and J. H. Holland, A New Method of Simulating the Central Nervous System Using an Automatic Digital Computer, Tech. Report, Univ. of Mich., March, 1959.
8. H. Blum, An Associative Machine for Dealing with the Visual Field and Some of its Biological Implications, Tech. Report, Air Force Cambridge Research Labs., February, 1962.
9. R. F. Rosin, An Organization of an Associative Cryogenic Computer, Proc. Spring Joint Computer Conf., May 1 to 3, 1962, p. 203.
10. R. J. Segal and H. P. Guerber, Four Advanced Computers - Key to Air Force Digital Data Comm. Syst., Proc. Eastern Joint Computer Conference, December, 1961, p. 264.

ON THE USE OF THE SOLOMON PARALLEL-PROCESSING COMPUTER*

J. R. Ball†, R. C. Bollinger‡, T. A. Jeeves†, R. C. McReynolds×, D. H. Shaffer†
Westinghouse Electric Corporation
Pittsburgh, Pa.

SUMMARY

The SOLOMON computer has a novel design which is intended to give it unusual capabilities in certain areas of computation. The arithmetic unit of SOLOMON is constructed with a large number of simple processing elements suitably interconnected, and hence differs from that of a conventional computer by being capable of a basically parallel operation. The initial development and study of this new computer has led to considerable scientific and engineering inquiry in three problem areas:

1. The design, development, and construction of the hardware necessary to make SOLOMON a reality.
2. The identification and investigation of numerical problems which most urgently need the unusual features of SOLOMON.
3. The generation of computational techniques which make the most effective use of SOLOMON's particular parallel construction.

This paper is an early report on some work which has been done in the second and third of these areas.

SOLOMON has certain inherent speed advantages as a consequence of its design. In the first place, computers conventionally require two memory cycles per simple command—one cycle to obtain the instructions and one cycle to obtain the operand. Although SOLOMON has the same basic requirement it handles up to 1024 operands with each instruction. Consequently, the time per operand spent in obtaining the instruction is negligible. This results in increasing the speed by a factor of two. In the second place, the fact that the processing elements handle 1024 operands at once greatly increases the effective speed. The factor is not 1024, however. Since the processors are serial-by-bit they require n memory references to add an n-bit word. If n is taken nominally to be 32, then the resulting net speed advantage is 1024/n, that is 1024/32 = 32. These two factors result in a fundamental speed increase on the order of 64 to 1 for comparable memory cycle times.

In addition to these concrete factors, there are other factors whose contribution to speed cannot be as easily measured. Among these are i) the advantages due to the intercommunication paths between the processing elements, ii) the advantage of using variable word length operations, iii) the net effect

*The applied research reported in this document has been made possible through support and sponsorship extended by the U.S. Air Force Rome Air Development Center and the U.S. Army Signal Research and Development Laboratory.

†Westinghouse Research Laboratories, Pittsburgh, Pa.
×Westinghouse Air Arm Division, Baltimore, Md.
‡Presently at Pennsylvania State University, State College, Pa. This work was done while at Westinghouse Research Laboratories, Pittsburgh, Pa.

resulting from either eliminating conventional indexing operations or else superseding them by mode operations, and iv) the loss in effectiveness resulting from the inability of utilizing all processors in every operation. The net speed advantage can only be determined by detailed analysis of individual specific problems.

The task of evaluating the feasibility of the SOLOMON computer has led to investigations of problems which primarily involve elementary, simultaneous computations of an iterative nature. In particular, the solution of linear systems and the maintenance of real-time multi-dimensional control and surveillance situations have been considered. Within these very broad areas two special problems have been rather thoroughly studied and are presented here to demonstrate the scope and application of SOLOMON. The first of these is a problem from partial differential equations, namely the discrete analogue of Dirichlet's problem on a rectangular gird. The second is the real-time problem of satellite tracking and the computations which attend it. These problems are discussed here individually and are followed by a brief summation of other work.

Partial Differential Equations

Introduction: Among current scientific computational problems, the numerical solution of partial differential equations has in recent years made the most severe demands on the computational speed and storage capacity of computing machines. It is therefore natural to investigate the capability and performance of the SOLOMON computer in this area. This discussion describes such an investigation in the area of partial differential equations of elliptic type. In particular, the problem chosen to serve as a standard for comparison with various methods and conventional machines is that of solving Laplace's equation over a square with Dirichlet boundary conditions. Since estimates of rates of convergence of various methods are obtainable for the Dirichlet problem on the square, and since running-time estimates for conventional machines also can be easily made for this problem, it seems a reasonable criterion.

The discussion which follows assumes familiarity with the concept of the SOLOMON Parallel-Processing Computer [1]. There are three parts: (1) an outline of the problem and the numerical methods to be used; (2) a discussion of the organization of the computations using the parallel-processing ability; (3) a presentation of the results of some comparisons of SOLOMON with conventional machines on the basis of the problem mentioned previously.

Finite Difference Approximations: In this section, the main features of the numerical methods to be considered are summarized. A complete exposition is given in the book of Forsythe and Wasow [2]. The statement of the problem given below is by now fairly standard in the literature.

Suppose the region to be considered is the open region, R, of the xy-plane, and suppose R has a boundary, C, which is a simple closed curve. It is required to find the solution $u = u(x,y)$ of the Laplacian boundary value problem

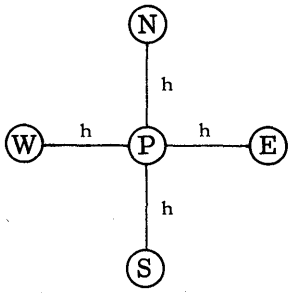
$$\Delta u = 0 \text{ in } R, \qquad u \text{ prescribed on } C, \tag{1}$$

where Δu denotes the Laplacian of u, $u_{xx} + u_{yy}$.

For purposes of numerical computation, the problem (1) is approximated as follows. We first replace the xy-plane by a net of square meshes of side h. For the given mesh constant, $h > 0$, the net consists of the lines $x = \mu h$, $y = \nu h$, $\mu,\nu = 0,1,2,\ldots$. The points $(\mu h, \nu h)$ are called nodes, and the nodes within R form the net region R_h, assumed to be connectable by line segments of the net within R. A node of R_h is said to be a regular interior point if each of its four (nearest) neighbors $(\mu h \pm h, \nu h \pm h)$ is in RUC. All other points of R_h are called irregular interior points. The set of boundary points, denoted by C_h, contains those points which are the points of intersection of C with the lines of the net; these may or may not be nodes. Since the principal purpose of this paper is to illustrate the SOLOMON concept, for purposes of exposition the simplifying assumption will be made that all points of C_h are nodes. That is, it will be assumed henceforth that the region of interest is a bounded plane region which is the connected union of squares and half-squares of the net imposed, so that all interior points are regular. For a detailed treatment of irregular interior points, and errors of discretization and approximation incurred by the use of an approximate boundary C_h rather than C, and by approximations

to the partial derivatives, the references should be consulted, especially [2].

From well-known finite difference approximations for the partial derivatives occurring in (1), a formal difference approximation, defined on the net, may be obtained as follows. For convenience, the neighbors of a point P are designated as shown below, and the net function will be denoted by U to avoid excessive subscripting.



In terms of this diagram, the replacements are:

$$u_{xx} \text{ by } \frac{1}{h^2}[U(W) - 2U(P) + U(E)]$$

$$u_{yy} \text{ by } \frac{1}{h^2}[U(N) - 2U(P) + U(S)].$$

Then (1) becomes

$$\Delta_h U = \frac{1}{h^2}[U(N) + U(E) + U(S) + U(W) - 4U(P)] = 0 \text{ in } R_h. \quad U \text{ prescribed on } C_h. \tag{2}$$

The formula for the Laplacian operator in (2) is the well-known five-point formula or "star" for the numerical solution of field equations. By using the formula (2) to replace the operator in (1) at each point P in R_h, the conditions of (1) may be approximated by a system of algebraic equations. Using the boundary conditions and applying (2) at each interior point P, the problem (1) is replaced by a system of N simultaneous equations for the N unknown values of P interior to C_h. Under appropriate conditions (see [2], for example), it can be shown that for P in R_h, $U(P) \to u(P)$ as $h \to 0$; the discussion here, however, is limited to a brief outline of some iterative procedures used for solving the approximate problem on SOLOMON, and the references should be consulted for the mathematical basis.

Simultaneous Displacements: One straight forward method which comes readily to mind as a candidate for a SOLOMON program because of its use of nearest neighbors is the method of simultaneous displacements. This well-known iterative method makes use of the five-point formula mentioned above, and goes as follows: A trial solution $U_o(P)$, P in R_h, is chosen. Suppose that we are at the k^{th} stage in the iteration, i.e., U_{k-1} has already been determined. For each point P, the value of the new solution $U_k(P)$ at that point is obtained averaging the values of the $(k-1)^{st}$ solution at the four nearest neighbors of P, i.e.,

$$U_k(P) = \frac{1}{4}\left[U_{k-1}(N) + U_{k-1}(E) + U_{k-1}(S) + U_{k-1}(W)\right].$$

This process is continued until the change in the values at every point does not exceed some prescribed tolerance. Sufficient conditions for the process to converge are known, and are given in [2, section 21.4]. This method has much to recommend it for SOLOMON in terms of simplicity and ease of programming. Its major defect, and one which encouraged investigation of other methods, is its inferior rate of convergence; this will be discussed later.

Optimum Successive Overrelaxation: There is a modification of the method of simultaneous displacements in which a new value is used as soon as it has been computed. That is, when solving for a new value, U(P), at a point P, one always employs the latest values of all other components involved in the formula for the new value at P. The procedure is called the method of successive displacements (or successive relaxation). As might be inferred from the brief description, it depends critically on the order σ in which the unknowns are determined by relaxation. Only cyclic orders are considered, in which the new values are obtained in the order $U_{\sigma(1)}, U_{\sigma(2)}, \ldots, U_{\sigma(N)}$, and repeat, where $\{\sigma(j)\}$ is some permutation of the first N integers.

Early users of relaxation often found it profitable to overrelax, that is, to change a component by some real factor w, $w > 1$,

times the change needed to satisfy the equation exactly. Overrelaxation was originally employed primarily in hand computation, and was not usually employed in the regular or cyclic order which is found most convenient when the computation is done on a digital computer. The question was then raised as to whether overrelaxation was profitable when used with a fixed, cyclic order of determining the unknowns. Although it is known [2, section 21.4] that overrelaxation is not profitable in solving the Dirichlet problem by simultaneous displacements, it is true that overrelaxation is highly profitable for many elliptic difference operators in connection with the method of successive displacements (successive relaxation). The theory of overrelaxation in successive displacements is due to Young and Frankel; a detailed exposition is given in [2, section 22], where the original papers and later developments are also referred to.

The practical application of the method is as follows. As before, we use the five-point difference formula. The nodes in R_h are scanned repeatedly in a cyclic order. At each point, the residual, $\Delta_h U(P)$, is formed, where $\Delta_h U(P) = U(N) + U(E) + U(S) + U(W) - 4\,U(P)$. Then, a new value of U(P) is computed by the formula,

$$\text{new } U(P) = U(P) + \frac{w}{4}\Delta_h U(P),$$

where w is the overrelaxation factor. (For $w = 1$, this is the method of successive displacements.) The theory of the method shows that it will converge for $0 < w < 2$, and that the most rapid convergence occurs at a value called w_{opt}, with $1 < w_{opt} < 2$. A good estimate of w_{opt} is important, and it is known [2, section 22] that a value of w somewhat larger than w_{opt} is less costly in computer time than one somewhat smaller than w_{opt}. It is also important that the time required to obtain a good estimate of w_{opt} be small compared to the time required to solve the problem using merely a reasonable guess for w_{opt}.

The problem of determining w_{opt} is a very substantial one. A few methods are suggested in [3, section 25.5], but the results at present are inconclusive, and it is conjectured in [2] that finding w_{opt} may in some cases be as hard as solving the original boundary value problem. For this reason, the program developed here assumes an acceptable value of w has been determined, and proceeds from that point.

One other point should be mentioned in connection with the cyclic order in which the new values at the nodes of the mesh are found. To be acceptable an order must be what is called consistent in the literature [2,4]; the details and definitions require a somewhat lengthy preparation and will not be given here, but are fully discussed in the references just cited. Forsythe and Wasow [2, p. 245] do state a simple criterion (due to Young) for consistency of five-point formulas, which may be reproduced here:

> "Of each pair P, Q of adjacent nodes of the net, one, say P, precedes the other, say Q, in the order σ. If so, draw an arrow from P to Q. Do this for every pair of adjacent nodes. Then, according to David Young (private communication), the order σ is consistent if and only if each elementary square mesh of the net is bounded by four arrows with a zero circulation, i.e., if a directed path enclosing the square travels with the arrows on two sides of the square, and against the arrows on two sides."

In order to make as much use as possible of the parallel nature of SOLOMON, and at the same time observe the requirement that the order be consistent, we have chosen the following scheme. We partition the nodes, P_{ij}, of the mesh into two sets S_1, S_2, by letting S_1 be the set of all points with odd parity (i + j odd) and S_2 be the set of all points with even parity (i + j even). Then an order consistent with this partition is to solve for all components in S_1, and then for all those in S_2. This leads to a checkerboard-like arrangement which can be easily obtained on SOLOMON.

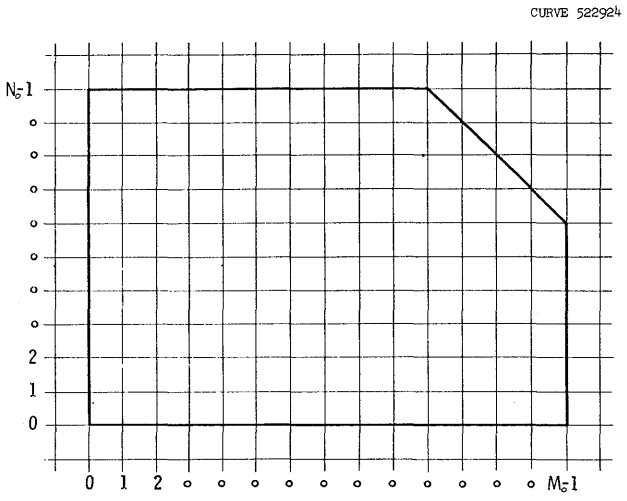
Rates of Convergence: A few words about rates of convergence are in order here to illustrate the reason that successive overrelaxation was chosen for SOLOMON rather than simultaneous displacements, even though the latter method is so simple to program. By "rate of convergence" is meant the average asymptotic number of base-e digits by which the error is decreased per iterative step. Forsythe and Wasow [2, p. 283] list some approximate rates of convergence for various methods for solving the Dirichlet problem on a $\pi \times \pi$ square with n points on a

side. For simultaneous displacements the approximate rate of convergence is given as $h^2/2$; for optimum successive overrelaxation it is 2h. For the $\pi \times \pi$ square, these rates will then be $h^2/2 = \pi^2/2n^2$, and $2\pi/n$, respectively, and the error will be decreased by about one base-e digit per $2n^2/\pi^2$ iterations for simultaneous displacements and per $2n/\pi$ iterations for successive overrelaxation. In [2, p. 374], it is shown that to reduce the initial error by a factor of 10^{-6} (approximately $e^{-13.8}$) takes $13.8n/(2\pi) = 2.2n$ sweeps for successive relaxations; a similar calculation gives $2.8n^2$ sweeps for simultaneous displacements. Some running-time computations based on these figures indicate that because of the large number of iterations required with the method of simultaneous displacements, SOLOMON can only show an order-of-magnitude improvement over conventional machines for n small. For successive overrelaxation, however, the improvement is substantial; this will be discussed later.

The Computational Scheme: The effectiveness of SOLOMON in solving problems will depend to a large extent on the representation of the problem in SOLOMON memory. The particular representation chosen for the Dirichlet problem and presented here permits all 1024 processing elements to be fully utilized. That is, no processing element is required to remain inactive for one or more instruction times, except when that processor contains a boundary or exterior point of the net.

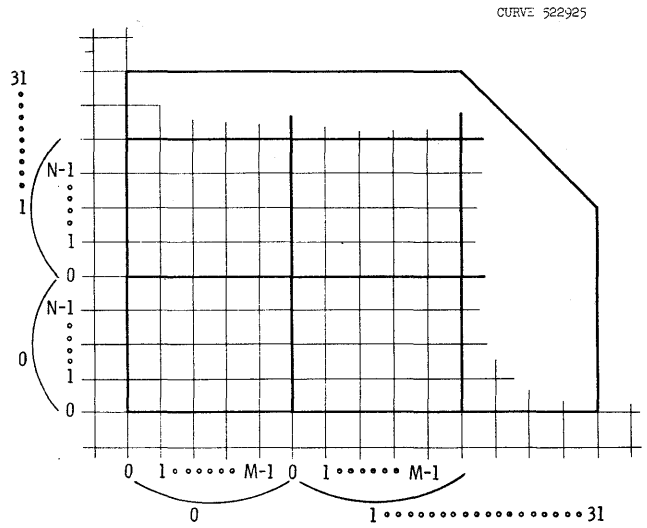
Consider the rectangular net shown in Figure 1 which contains $N_0 \times M_0$ net points. The nodes of the net are partitioned into 1024 rectangular groups of dimension $N \times M$. This is shown in Figure 2. In order to apply the iteration formula with a consistent ordering, it is convenient to restrict N and M to be even integers.

Each group of net points must now be represented in the corresponding processing element memory. As shown in Figure 3, the net points in an $N \times M$ group are ordered and a list is constructed in each processing element memory to contain the net function evaluation at each net point. Since the application of the iterative formula requires the four nearest neighbors of each net point, a net point ordering will be selected which allows the neighbor net functions to be easily obtained.



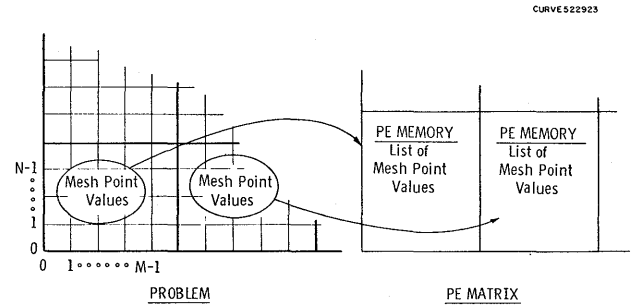
Region composed of squares and half squares

Figure 1



Partition mesh points into 1024 equal groups.

Figure 2



Order mesh points for each group in a corresponding PE memory.

Figure 3

The distinction among boundary, interior, and exterior net points is facilitated by the multi-modal operation of the processing elements. Associated with each net point in each processing element will be mode bits identifying that point. These mode bits will then be used to set the mode state of the processing elements before applying the iteration formula at a net point. Then by only operating on processing elements which are in the mode assigned to interior net points, both boundary and exterior points remain unchanged.

Storage Requirements: Each SOLOMON processing element contains 4096 bits of core memory. This memory is addressable by bit number and the word size may be set and changed by the program. This variable word length structure of the processing element permits net point values to be listed without any wasted bits. All statements about storage capacity for data words will then depend directly on the accuracy needed for the data.

A word in SOLOMON memory will be necessary for each of the $N_0 M_0$ net points. In addition, 2 mode bits are used to identify the points as boundary, interior, or exterior. Therefore, if p is the number of bits in a net point word, the total storage capacity, c, of SOLOMON core memory is:

$$c = \left[\frac{4096}{P + 2}\right] \times 1024.$$

For various p this is

p:	18	24	30	36
c:	~200,000	~160,000	~130,000	~100,000

These figures can be extended through the auxiliary storage system of SOLOMON.

Running-Time Estimates: In order to measure the effectiveness of the SOLOMON parallel organization on the Dirichlet problem an estimate of the time required to process each net point on SOLOMON will be compared to a similar estimate for a conventional computer.*

A program has been written for SOLOMON which applies the Young-Frankel iteration formula. This program requires time T_I given by

$$T_I = (14.5\ p + 3)\ 1.2\ \mu s$$

for one iteration on p-bit words. Since one iteration revises the net point value at 1024 nodes simultaneously, the time, T, required to process a single net point is

$$T = \frac{T_I}{1024} = \frac{14.5\ p + 3}{853}\ \mu s.$$

The time required to test the results of an iteration for a solution within tolerance is very small and may be neglected.

The time required to process one mesh point on a conventional computer is estimated at

$$T_C = 70\ \mu s.$$

This estimate is based on the arithmetic instructions necessary to evaluate the formula which include

6 Additions
1 Multiply
1 Shift
2 Load and Store

plus an assumed 3 index-instruction times to distinguish each point as interior or boundary.

These two estimates may now be compared to obtain an approximate SOLOMON time-advantage ratio, ($T:T_C$);

p	:	18	24	30	36
T_C	:	70 μs	70 μs	70 μs	70 μs
T	:	.31 μs	.41 μs	.52 μs	.60 μs
$T:T_C$	:	225:1	167:1	133:1	117:1

These figures are conservative since they include neither the time to test for a solution nor the time required for a conventional computer to buffer the large number of net points. In addition, it appears to be very convenient and quick for SOLOMON to solve a reduced problem with approximated boundary values in order to provide an initial guess for the full problem. In this manner, it is hoped to further increase the SOLOMON speed advantages given above.

*The computer chosen for comparison is of conventional contemporary design with a multiplication-division time that is one to seven times as long as its 4 μs addition time.

Satellite Tracking

The computing and data processing problem for satellite surveillance is receiving increasing attention as the tempo of the space program increases. The increase in satellite densities over the next few years will increase the computing and data processing problem by several orders of magnitude. Since the presently existing problem of simultaneously tracking relatively few objects requires, in general, high-speed, large capacity computing systems, it is evident that the future problem of satellite surveillance will require highly complex and sophisticated computing systems having capabilities far greater than those of contemporary systems.

The SOLOMON computer with its parallel computational capabilities, large capacity memory system and novel system organization features is capable of meeting the advanced computing requirements for satellite surveillance with respect to both speed and memory requirements. To illustrate applicability of SOLOMON to the problem, the major functions that must be performed in a satellite surveillance system are discussed in some detail. Certain aspects of the problem will be omitted in order to keep the text of this paper unclassified.

The satellite surveillance problem is, in general terms, that of receiving and performing the required processing of input data from a radar system maintaining continuous surveillance over a specified area of coverage.

The raw radar data needs to be converted to digital form and fed into the computing system. The computing system must then establish and maintain track files on all satellites passing through its coverage sector.

The establishment of track files on each target within the coverage involves the elimination of false alarms, resolution of ambiguity, etc., from the input data. Once firm tracks have been established, the computer must ascertain the status of each detected satellite; that is, whether the satellite is a known satellite following a predicted orbit, a known satellite not within its predicted course, or a satellite not included in the computer's available identification data.

The status of each satellite will determine the subsequent processing of the relevant data. If the satellite is known and following a predicted course, it will be tagged and no subsequent processing will be required. If the satellite is known but not within its prescribed course, it will probably be necessary to perform the orbital correction calculations required to update the orbital elements maintained on each known satellite. If a particular satellite is unknown, orbital calculations must be performed, resulting in a set of orbital elements defining the orbit of the newly detected satellite.

The SOLOMON computer, although capable of performing the bulk of the computing required in the system, will probably not solve the entire computing problem. One might visualize the entire computing complex as consisting of a conventionally organized tactical computer and a radar control unit in addition to SOLOMON. The tactical computer would be capable of performing the refined processing on a relatively small number of satellites that require additional processing, a task that would be an inefficient use of SOLOMON. The radar control computer would probably be a special purpose machine capable of performing the specific control function required by the radar system (i.e., frequency control, antenna beam steering control, etc.).

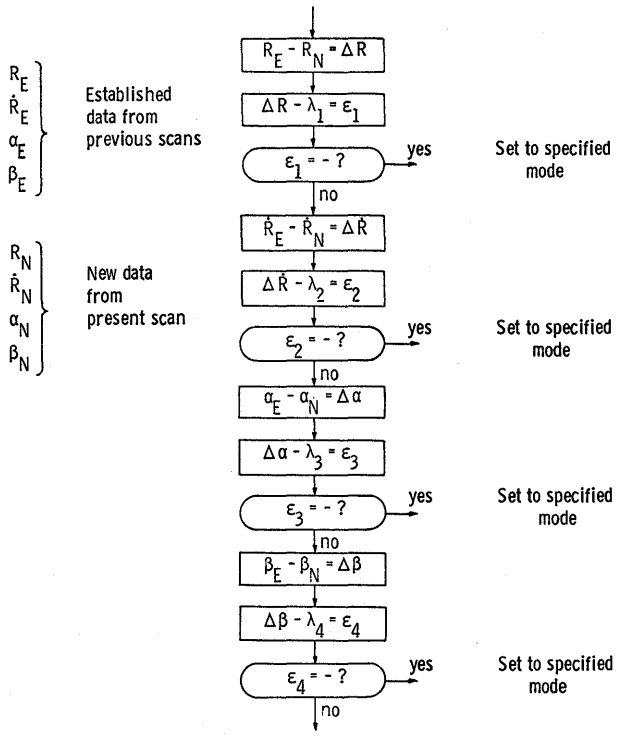
Data correlation is a function having a substantial bearing on computing system requirements. Correlation will be required in two phases of the problem. The first of these is scan-to-scan correlation; i.e., correlation of radar input data on successive radar scans. The second is the problem of matching radar returns on observed satellites with known or predicted satellite orbital positions.

The problem of scan-to-scan correlation becomes most critical when the false alarm rate of the input data is high, because then the computing system must process a much larger number of returns than the number of actual targets. Due to stringent requirements on the radar system, e.g., long range tracking, high accuracy, low signal-to-noise ratio, etc., the false alarm rate will probably be high. It should be pointed out that a trade-off exists between the computing capabilities required and the overall radar performance. With a computing system possessing the inherent capabilities of SOLOMON the radar system performance will be vastly increased. This increased system performance will result from SOLOMON's ability to accept and process a much higher density of return

including false alarms and ambiguous returns as well as legitimate returns) than could be realized otherwise. In addition to eliminating false alarms the computing system must discard returns from non-orbiting objects and recognize multiple returns from a single object. The most reliable techniques for eliminating false alarms and ambiguous returns are based on persistence of the returns from scan-to-scan. This necessarily means that the computing system must store all returns for several radar scans (on the order of 5) before any unique returns can be eliminated as false alarms. Such a technique demands large memory capacity such as that provided by the SOLOMON processing element memories.

SOLOMON is particularly well adapted to perform the scan-to-scan correlation functions as illustrated by the following discussion. Figure 4 illustrates the scan-to-scan correlation of 4 parameters: range (R), range rate ($\dot{R}$), azimuth (β) and elevation (α). The parameters of new returns are compared with those from previous scans. The number of comparisons that can be made simultaneously by SOLOMON is equal to the total number of processing elements in the network, assuming of course that each unique return is routed to a specified processing element in the desired manner. Once the program has cycled through the correlation subroutine there will undoubtedly be returns in various processing elements that did not match the returns with which they were being compared. In these cases, by the use of multimodal control, the returns that did not match are set to a specified mode by the programmer. The subroutine can now be repeated, acting only on the processing elements in this mode and comparing the unmatched returns with other returns either within the same processing elements or in adjacent processing elements. This process utilizes the interconnection between processing elements. The number of comparisons that must be made (i.e., the number of iterations of the subroutine required to perform the scan-to-scan correlation function) to assure that a return has been compared with all possible matches is a function of target density, false alarm rates, target geometry, precorrelation techniques, and the actual SOLOMON configuration.

When a return has been firmly established as a return from a satellite then that



Correlation on Four Parameters

Figure 4

return must be compared with the predicted satellite orbit data to determine if this is an unknown satellite, a known satellite following a predicted course, or a known satellite not within its predicted course. The autocorrelation subroutine required for this function is almost identical to that required for scan-to-scan correlation.

Other systems that have been proposed for the performance of satellite surveillance have relied upon a separate catalog memory containing the orbital elements for each known satellite. Because of the irregular order in which satellites might appear within the radar coverage at any given time, it is probably not feasible to order the catalog of tracks in the machine. In SOLOMON the orbital elements in the catalog memory would be distributed throughout the processing element's memories as a function of total satellite density. That is, if the total satellite count is five times the total number of SOLOMON processing elements, each processing element memory would contain 5 satellite tracks. The correlation is therefore simplified and solvable within SOLOMON in a manner almost identical to that of the scan-to-scan correlation.

To establish the magnitude of the advantages of SOLOMON over conventional computers in the performance of such correlations, the problem is outlined in detail as follows: a computer has in its memory m total track files. Each track file, consisting of several words, defines a particular target. At any given time the computer can receive n new inputs from the radar systems. The problem is (1) to associate each new input with an established track file, (2) establish new track files where required, or (3) eliminate spurious or ambiguous returns. The approach to this problem to date has been quite straightforward. Since existing computers are sequential and only one operation can be performed at a time, each new return is sequentially called from memory and a search of the established tracks is made. The total number of discrete steps required to make the search in a sequential computer on n new returns through m established track files to determine if any n has a match among m is

$$m + m - 1 + m - 2 \text{ --- } + m - n = \frac{(n+1)(2m-n)}{2}$$

where $m > n$.

In SOLOMON the total number of discrete steps required to perform the same function, as previously pointed out, is simply m, since in any given step the computer would be making n comparisons and, at most, comparison of each n with all possible m's would take place in m steps (see Figure 4).

In the satellite surveillance problem, where both m and n will be exceptionally large, the advantages of SOLOMON over conventionally organized computers are obvious.

The problem of coordinate conversion will pose additional requirements on the computing systems employed for this problem. Since the track files as established from the raw radar input will be in radar coordinates and the tracks in the catalog will probably be in the form of orbital elements, some form of conversion must be done prior to the correlation of the detected satellites with the known satellites. These computations, although not especially complex, are quite time-consuming. While in a conventionally organized computer each track would be converted in a sequential manner, in the SOLOMON computer the execution of one conversion subroutine would perform the conversions for all satellites (assuming that the total number of conversions required is not greater than the total number of processing elements). In order to compute new satellite orbits and update existing data on established satellites these satellites must be tracked and sufficient data on the actual orbits must be gathered. Of the total number of satellites detected in the radar search mode, only a small number will require detailed orbital calculations. In a typical satellite tracking system this function would not be performed by SOLOMON, but by a high speed conventionally organized computer.

Other Problem Areas

In the course of studies such as those reported here it has become apparent that the computational techniques most suitable for SOLOMON are not necessarily those which have been popular for conventional high-speed machines. To make proper use of the parallel design of SOLOMON it sometimes seems necessary to employ methods of computation that would be quite cumbersome for a conventional machine.

A case in point can be found in the problem of solving a system of linear equations. To be specific, consider the problem of solving a system of 15 equations in 15 unknowns. Because of the unique construction of SOLOMON, it is possible to solve up to 64 such systems simultaneously for the same cost in time and effort. The activities of the computer will be the same whether one or sixty-four systems are being solved. Consequently, SOLOMON will perform better when many systems of linear equations can be solved at the same time.

We wish to stress that the proper computational scheme must be employed for the most efficient use of SOLOMON's capabilities since our mission is not to establish SOLOMON as the ultimate in computers, but rather to explore those areas in which it is superior.

The problem areas in which SOLOMON has been found to be especially capable are constantly expanding under the impetus of the present investigations. Work is now in process on several other sample problems to demonstrate the general applicability of SOLOMON. We are studying multidimensional functional optimization, where an entirely new methodology for finding absolute

maxima may result. We are studying communication and transportation problems, a sorting problem, a problem in handling a Boolean form, multiple integration, and a sound detection problem. Additional systems studies being considered include photo-reconnaissance, numerical weather forecasting, cryptoanalysis, nuclear reactor calculations, and air traffic control.

REFERENCES

1. Slotnick, D. L., Borck, W. C., McReynolds, R. C. "The SOLOMON Computer," Proceedings of the Fall 1962 Joint Computer Conference.
2. Forsythe, G. E., and Wasow, W. R. "Finite-Difference Methods for Partial Differential Equations," Wiley, N.Y., 1960.
3. Forsythe, G. E. "Difference Methods on a Digital Computer for Laplacian Boundary Value Problems." Transactions of the Symposium on Partial Differential Equations. N. Aronszayn, A. Douglis, C. B. Morrey, Jr. (eds.), Interscience, N.Y., 1955.
4. Sheldon, J. W. "Iterative Methods for the Solution of Elliptic Partial Differential Equations," Mathematical Methods for Digital Computers, Wiley, N.Y., 1960.

DATA PROCESSING FOR COMMUNICATION NETWORK MONITORING AND CONTROL

D. I. Caplan
Surface Communications Division
Radio Corporation of America
Camden 2, New Jersey

The long-haul communications network is the backbone of military communications. It provides the coordination necessary for global military operations and logistic support. For maximum network effectiveness, a central monitoring and control function is necessary. System studies described in this paper have shown that automatic data processing is applicable to network monitoring and control, and provides rapid and efficient network reaction to natural and man-made disturbances.

The basic elements of the long-haul communication network are the switching centers, the trunks connecting them, and the subscribers. Three types of service are provided to the user of the long-haul network. These are:

1. Direct - A direct connection is made on demand between two subscribers, and broken down when the call is completed.

2. Store and Forward - A message is transferred through the network. It is stored at each switching center, and then passed along to the next center until it reaches its destination.

3. Allocated Service - Allocated service is a direct subscriber-to-subscriber connection which remains in effect full time. This "hot-line" service differs from direct service in that the connection is not broken down at the end of a call.

The long-haul network must provide service in the face of many operational difficulties. These difficulties include wide variations in the traffic load and "outages" of equipment due to acts of nature or enemy action. The hot-line service must be restored immediately when any of the hot-line channels are effected by outages. To complicate the situation, peak traffic loads will usually occur at the very time outages are caused by enemy action or severe storms.

There are basically four actions which can be taken to alleviate operating difficulties. These are:

1. Alternate Routes: Backed up store-and-forward traffic can be sent via other routes. Direct calls can also be handled over routes other than the preferred route.

2. Spare Channels: Space channels can be put into service, either to replace down channels or to add transmission capacity.

3. Preemption: Circuits or facilities can be reassigned from low priority users to high priority users.

4. Service Limitations: Maximum message length or call time can be specified, service can be denied to certain classes of subscribers, or other limitations can be placed on the subscribers.

The actions listed above can be taken on a local or global level. Local action will be taken at an individual switching center, while regional or global action will require cooperative performance at a number of switching centers. Obviously the effectiveness of regional or global measures depends on coordination of the switching centers, which

must be achieved through a central control facility.

Based on analysis of the problems involved, a system study of the control center functions and possible implementation has been performed. Automatic data processing was found to be applicable both at the switching centers and at the control center. The results of the system study, described in this paper, are applicable to many long-haul systems, and provide an insight into the network control complex of the future.

Network Monitoring and Control Concept

To provide effective network reaction to varying traffic loads and equipment outages, a closed loop network monitoring and control system is necessary. As shown in Figure 1, the status of the network is monitored at the switching centers and transferred back to the network control center. Network operation is analyzed at the control center and control actions are initiated there. These control actions are carried out through command messages sent to the switching centers.

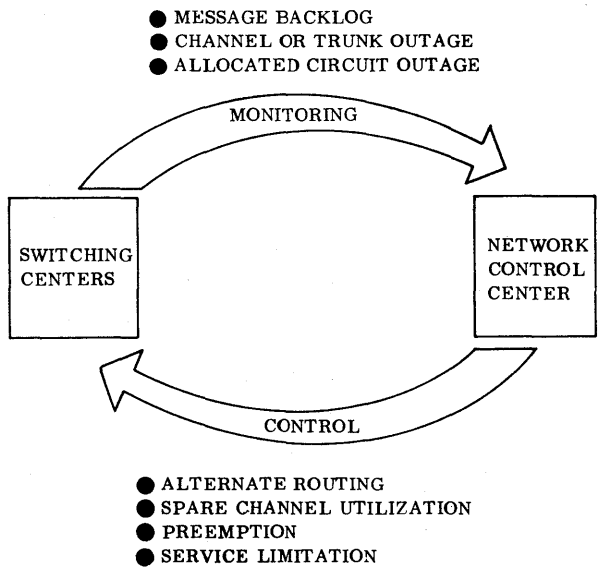


Figure 1. Network Monitoring and Control Concept.

Like any closed loop system, the monitoring and control system can be made ineffective by delay or by inaccuracy caused by data errors. To reduce these two problems to a minimum, automatic data processing should be used at the switching centers for composition of the status messages and at the control center for network status analysis and display.

Communication between the switching centers and the network control center can be accomplished in several ways. First, allocated channels could be provided between the switching centers and the network control centers. Second, direct or store and forward communications can be initiated either periodically on a preassigned schedule or when required. In order to achieve effective use of communication facilities, a common practice is to use store and forward messages for both monitoring and control information, with direct calls used only under emergency conditions. In the case of status messages which are usually long and which contain routine information for the most part, a preassigned schedule of reports is used. Generally, an hourly report is frequency enough for satisfactory reaction time. Emergency reports can be entered at any time.

Status Message Composition

The simplest approach to station status monitoring is manual. The technical controller at the station records the message backlogs, channel outages, and other pertinent data. He then composes a teletype status message which he sends to the network control center.

From the network control center viewpoint, the manually prepared status messages are a special problem. Because of human error, mistakes in format are common. Messages having incorrect formats will be rejected at an automated network control center, and manual intervention and correction will be required. An automated status message composed at the switching center is therefore desirable.

Data ordering is another problem in manually prepared status messages. Each event at the station, such as a channel outage or restoration, has a time of occurence which must form a part of the status messages. These events are usually recorded by the station personnel in order of occurrence. However, the status message format will normally require grouping by channel or trunk, so the events must be sorted into a prescribed sequence before they are transmitted. This operation is time consuming and subject to error when performed manually.

The Status Message Composer concept, shown in Figure 2, was developed to provide automatic status message generation from manual inputs. A block diagram of the proposed unit is shown in Figure 3.

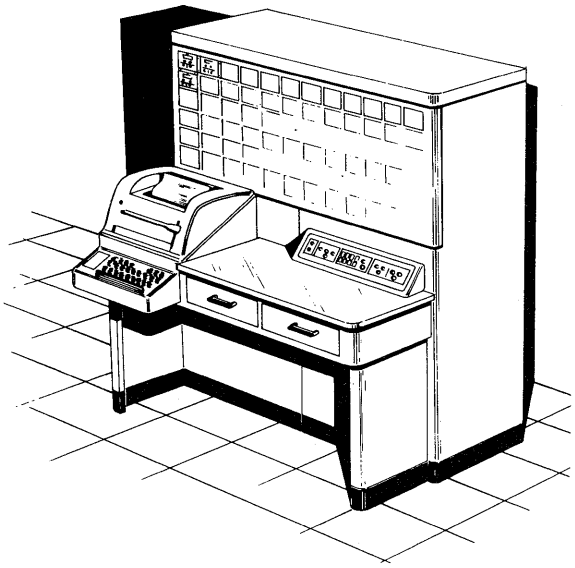


Figure 2. Status Message Composer, Artist's Concept.

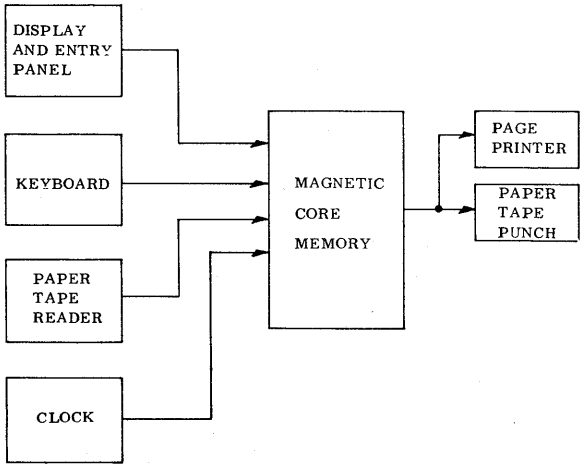


Figure 3. Status Message Composer, Block Diagram.

The display and entry panel at the top of the Status Message Composer provides the means for manual entry of trunk or channel outages. Each of the small display and entry modules corresponds to a single trunk or channel. The color of the display module indicates the last inserted status, and serves as a station status display.

The operator types variable data, such as reason for outage, into the keyboard on the left of the Status Message Composer. The panel on the right of the keyboard indicates to the operator what information is required, and provides overall controls such as unit power.

As shown in the block diagram, the operator-entered information is stored in a core memory. The time of data entry is read into the core memory automatically from a real-time clock. The location of the stored information is predetermined, so that all information about a particular trunk or channel always goes into the same group of words in the memory. Therefore, the status data are always stored in the proper sequence, and will be properly grouped when read out of the memory.

The initial setup of the core memory is performed by reading in a punched paper tape which designates the core locations to be used for each trunk, channel, etc. The paper tape also designates the display and entry module corresponding to each trunk or channel. Changes in station trunks or channels can be accommodated by simply changing the paper tape and relabeling one or more display and entry modules.

A status report can be generated either periodically under clock control or at any time under manual control. The report is generated by a memory read-out which transfers all stored status data to both the paper tape punch and the page printer. The punched paper tape is entered into the communications network for store-and-forward transmission to the network control center. The copy produced by the page printer becomes the station operating log.

Network Control Center Functions

The functions to be performed at the network control center are described in this section. They can be handled manually or automatically. The next section describes an automatic implementation of the network control center.

The network control center operation is shown in the information flow diagram, Figure 4. Status messages from the switching centers are received at the network control

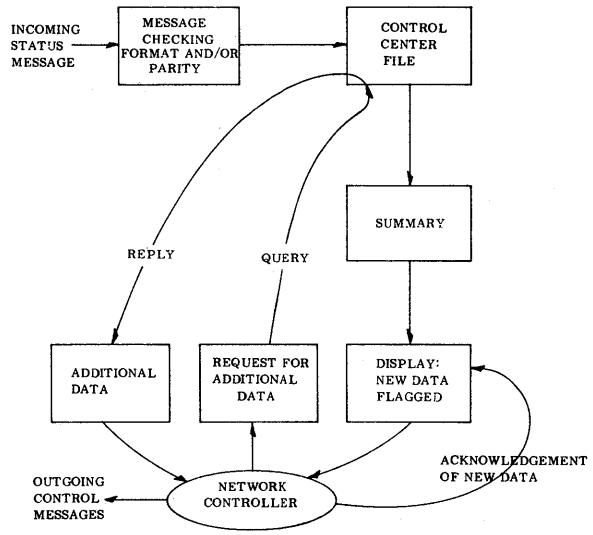


Figure 4. Network Control Center Information Flow Diagram.

center and recorded, either manually or automatically. The information in the incoming messages is checked for errors, using whatever redundancy is available in the messages (i.e., parity bits or format). The status data are then recorded in the control center file.

The status information is summarized for display to the network controller. It is important that the degree of summarization be optimized. Too much detail will swamp the controller; too little will limit his understanding of network status. Also, new information must be flagged in some way to attract the controller's attention and indicate the need for action on his part. The controller will acknowledge recognition of the new information by resetting the flag.

Using the data provided by the display, the controller is able to solve most network problems. On occasion, however, he may need additional detailed data. Any information in the control center files will be on-call, as required. To get additional data, the controller will initiate a query, designating the desired data. This operation is shown in Figure 4 as the query-reply loop.

Automated Network Control Center

To provide the network control center functions automatically, the system concept shown in Figure 5 has been developed. A general purpose digital computer is proposed, together with special equipment and displays, to provide status message processing, storage, and display. The Information Processor suggested for this application is an RCA 304 computer with a record file for bulk storage.

As shown in Figure 5, the data input section of the control center provides terminations for incoming channels. Two channels are used for locally generated data; queries from the network controller on one channel and manually entered data on the other channel. The other channels carry status messages from network switching centers.

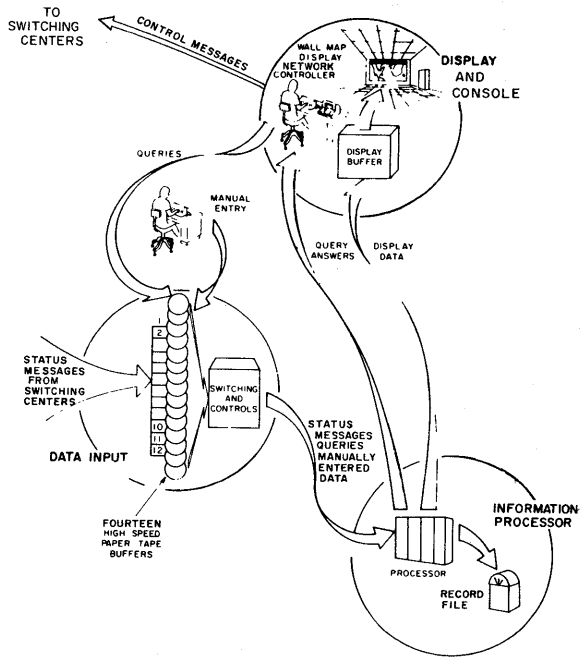


Figure 5. Data Flow in Automated Network Control Center.

Temporary storage for all incoming channels is provided by paper tape. Data can be handled on all channels simultaneously at 100 words per minute. When a complete message has been recorded on tape in one channel, it is read into the Information Processor at 1000 words per minute. The Information Processor checks the incoming messages for correct format. Messages with format errors are rejected and printed out. Control center personnel correct the format errors and reenter the corrected status messages at the manual entry keyboard.

In addition to checking the incoming status messages, the Information Processor maintains a complete file of network status. It continually provides an updated summary of

network status, in suitable code and format, for transfer via the display buffer to the wall map display. New display data are indicated to the controller by flashing lights, which the controller can reset when he desires.

Both trunk equipment status and traffic backlog would be provided in a single geographic type display. These two factors are closely related and form the basis for intelligent system control decision. Above the geographic display would be a tabular display of the status of lines allocated to high priority users. The wall display is shown in Figure 6.

An artist's concept of the proposed network control room is shown in Figure 6. The wall map displays the status of all switching centers, traffic backlogs, and trunks in the network. The tabular display at the top of the map shows the status of allocated channels.

The controller's console is simple and functional. It contains a page printer and keyboard for query entry and reply and a second keyboard for composition of control messages. A small panel is provided for display illumination controls, and for indicators showing status of equipment in the next room.

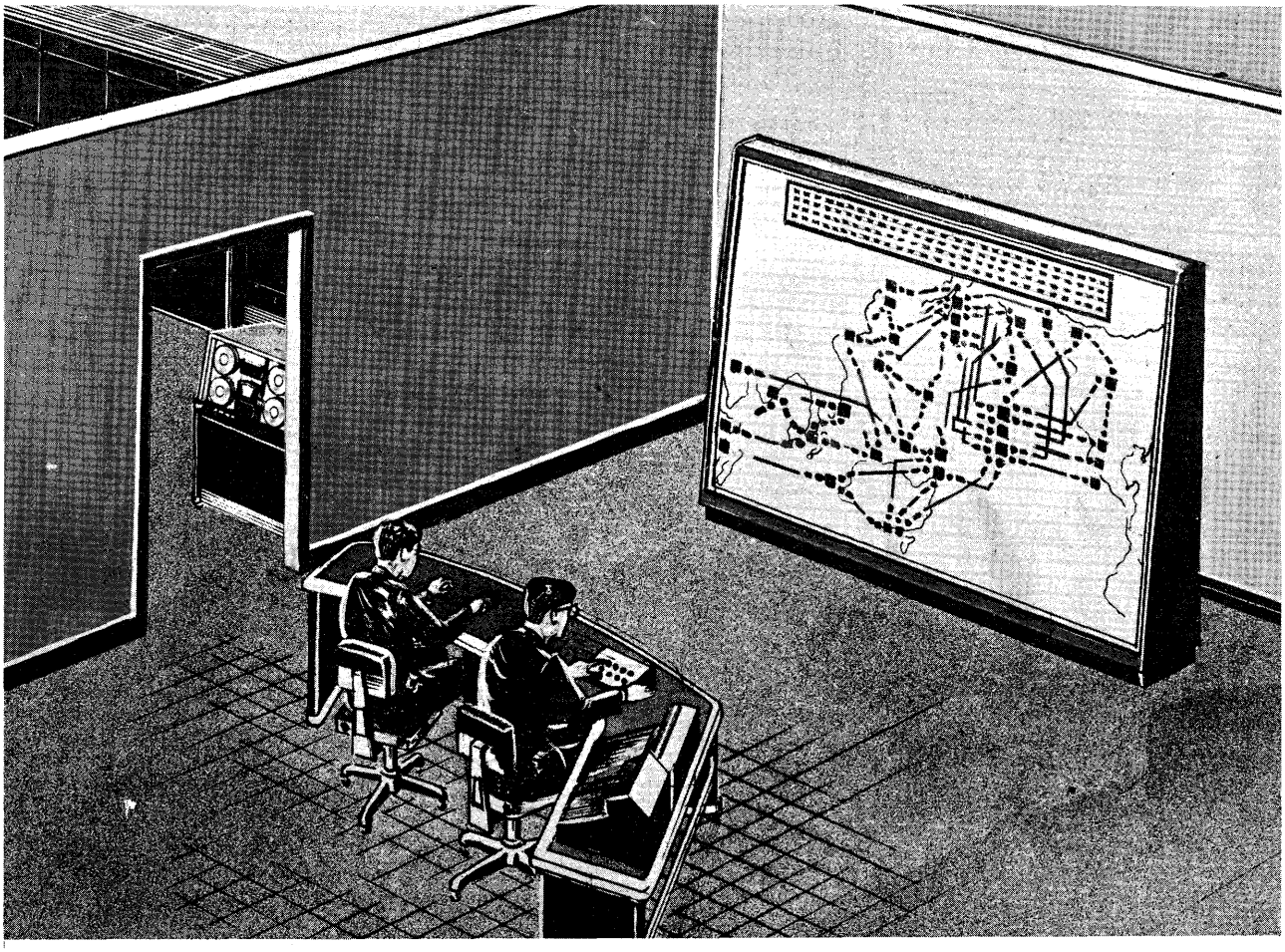


Figure 6. Network Control Room.

If the controller wants data other than that shown by the display, he will enter a query to the Information Processor. The Processor prepares the reply data and transfers it back to the controller. By entering a query, the controller can get any data recorded in the Processor files.

Although one man can operate the console, working space for an observer is provided.

A possible layout of the automated control center is shown in the artist's concept of Figure 7. The equipment room is located next to the network control room. The separating wall has been removed for clarity.

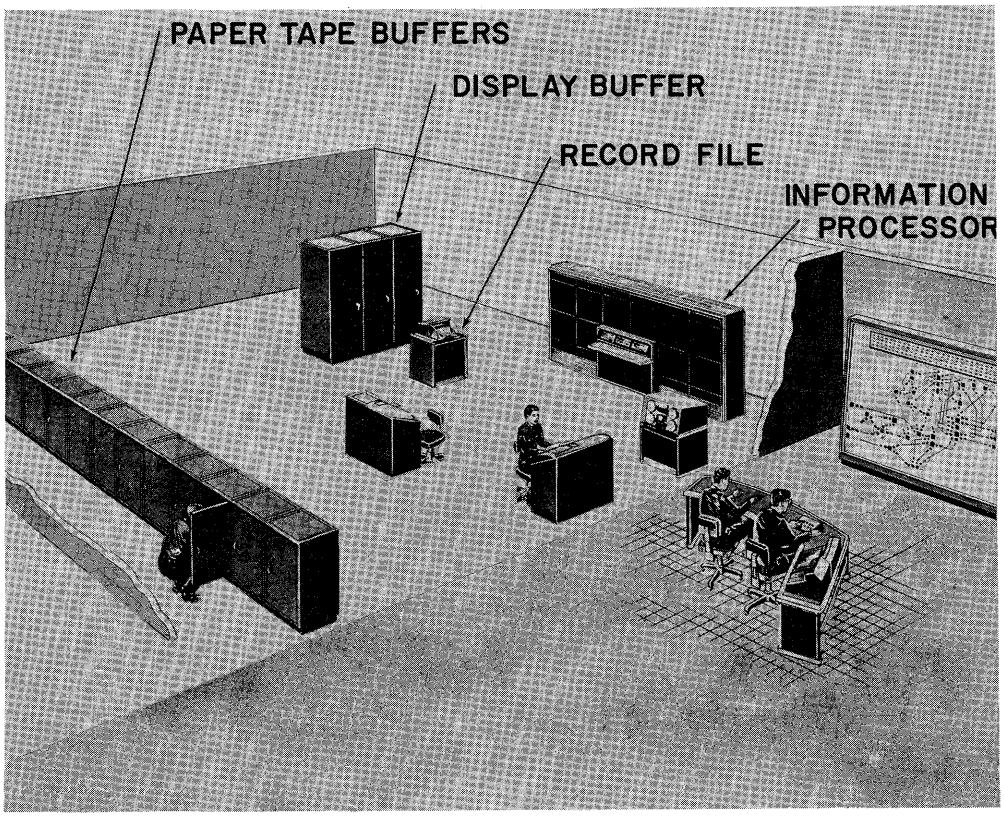


Figure 7. Artist's Concept of Automated Control Center.

At the wall to the right in an RCA 304 Information Processor, with the record file and paper tape inputs located in front. The smaller cabinets contain the paper tape punches and readers for terminating the incoming channels. Two teletype operator positions are provided; one for manual entry and the other for channel coordination with the incoming and outgoing channels. The large racks in the rear house the display buffers and other special equipment.

Data Processing in the Automated Control Center

The Information Processor in the Automated Network Control Center must perform four data processing functions:

1. Incoming Message Check: The processor will check the format of incoming messages and reject those having format errors. The messages in error will be printed out, together with an indication of the detected error.

2. Station File Maintenance: A file of the current status and recent history of each station (switching center) will be kept in the processor memory. As the station status reports come in, the status information will be posted to the station file.

3. Display Data Output: The processor will automatically provide updated status in a form suitable for the control center display.

4. Process Queries: The processor will accept queries from the keyboard at the controller's console. The data requested will be retrieved from the station file and output to the page printer at the controller's console. Each of these tasks must be done on a real time basis, to avoid system delays which would reduce effectiveness. The operating speed of the processor complex should be designed to keep up with the peak work load, and to catch up after periods of scheduled maintenance.

The data processing operations in the proposed control center system are based on the use of a Data Record File for storage of station status. The RCA Data Record File, Model 361, stores information on both sides of 128 magnetic coated records, with 18,000 characters stored on a side. The status of each station is stored on one side of a record in the Data Record File. Included in the stored

data are the status of every trunk terminating at the station, broken down into traffic backlog, status of the trunk, status of the channels in the trunk, and status of the users having allocated channels in the trunk.

As previously described, the incoming status reports are initially stored on paper tape, and then read into the Processor as complete messages. As soon as the Processor recognizes the station identity referred to in a particular tape, it selects the record containing the status of that station and reads the complete station file into the core memory. When the station file is in the core memory and the status message has been read in, the Processor performs the updating operation. After the entire station status has been updated, it is rewritten in the record file as a unit.

As the Processor updates the station file, it abstracts the data that must be displayed. These data are translated into the proper code and format for driving the display, and transferred to the display buffer.

Query processing is done on a station basis. Each query is analyzed by the processor to determine the stations involved. The station file records are then scanned and the appropriate information is extracted and converted to a format suitable for print-out.

SUMMARY

A system study of long-haul network control center requirements, functions, and operation has resulted in a network monitoring and control concept which includes data processing at both the switching centers and the control center. Such high speed data processing will substantially increase effectiveness of the present world wide long-haul system by reducing the reaction time to natural and man made disturbances. It, therefore, is a valuable tool in meeting the increased traffic loads and the vulnerability of communication channels to modern weapons.

ACKNOWLEDGEMENT

Many RCA engineers contributed to the network control center concepts described above. The author wishes to particularly acknowledge the efforts of A. Coleman, S. Kaplan, J. Karroll, M. Masonson, D. O'Rourke, and E. Simshouser.

DESIGN OF ITT 525 "VADE" REAL-TIME PROCESSOR

Dr. D. R. Helman, E. E. Barrett, R. Hayum and F. O. Williams
ITT Federal Laboratories
Nutley, New Jersey

SUMMARY

The ITT 525 VADE (Versatile Automatic Data Exchange) is a medium-scale communications processor capable of handling 128 duplexed teletype lines and 16 high speed data lines. The processor is of the single-address, parallel binary type utilizing a two-microsecond-cycle-time core memory and operating at a single-phase clock rate of four megacycles. The fundamental design approach of the machine is to trade the intrinsic speed of high performance hardware for a reduction in total equipment, through time-sharing. The memory is shared between stored program and input/output functions without the use of a complicated "interrupt" feature. Serial data transfers between the memory and communication lines are performed on a "bit-at-a-time" basis requiring a minimum of per-line buffering. The central processor hardware is largely conventional but has been reduced as much as possible without impairing the power of a basic communications processing instruction repertoire—which includes indexing and character mode operations but not, as yet, multiplication or division. Instruction time is six microseconds and the number of instructions performed per second varies from 63,500 to 81,000 depending on the existing input/output traffic load. Duplexing of the machine is accomplished by a "shadow" system whereby the off-line processor is continuously updated by the on-line processor through one of the normal high speed data links.

INTRODUCTION

At the present time the Design of Real-Time Processors is following the multiprogramming or multisequencing philosopy. Multiprogramming is usually defined as the time sharing of a single central processing unit. The processor responds to the real-time channels by activating a stored program which in many cases is unique to that particular channel. The memory must store not only the individual programs, but also the addresses of the programs for the corresponding channels. Furthermore, the processor must provide some type of priority-interrupt system which will respond to the various service requests of the real-time channels.

Real-time processors designed upon the multiprogramming basis usually provide per-line equipment which converts the serial binary stream to a parallel character. After this conversion, the character enters the Input-Output system where character or word buffering occurs. Finally, the data enters the Main Memory for processing, after the channel service—request has been recognized. To implement such a processing system much special purpose hardware and programming is required. Program, index and supervisory memories may be utilized in conjunction with special purpose priority interrupt hardware.

ITT 525 (Versatile Automatic Data Exchange) is a real-time processor designed upon a radically new philosophy. The objective of the design is to trade-off high internal processing speed with hardware, such that

effective utilization is made of the machine capability. The ITT 525 processor serves as a real-time store and forward message processor, which may serve as many as 16 high speed duplexed data lines and 128 teletype lines operating at a 100% line utilization.

The unique features of the ITT 525 include the sharing of one core memory for input, output and processing functions; the serial bit at a time assembly and disassembly of messages in the shared core memory using some simple in-out hardware; the storage of instruction micro-function logic instead of the standard operation decoder and logic; a minimum register central processor utilizing direct data transfers and providing the facilities for indexing and character mode; a powerful instruction repertoire for the implementation of the operational, utility and diagnostic programs.

System Design

The objective of the ITT 525 design was to produce a versatile message processor, at a minimum cost per line, to perform the function of a local area center handling a reasonable amount of data and teletype lines. It was decided to implement a system capable of interfacing with 128 duplexed teletype lines plus 16 duplexed data lines.

In order to achieve minimum cost per line the first design philosophy established was to make optimum use of the common equipment. Thus, it was decided to time-share one core memory and the major control circuits, between the In-Out unit and the central processor. This was made possible because of the high speed core memory, operating at a 2 microsecond read-write speed, and 4 megacycle logic circuits. If it is assumed that two memory cycles are required to perform a machine instruction, a maximum of 250,000 instructions per second may be executed by the ITT 525.

Since the ITT 525 has such a high internal speed, it was further decided to deviate from conventions and accept data from the real-time lines a bit at a time per line into the one core memory with no per line buffering. The messages are, thus, taken directly from the serial bit stream into the core memory where they are completely assembled. The message remains in this storage area while it is being processed and analyzed by the stored program and finally becomes disassembled one bit at a time for the output line transmittal. This line scanning or bit sampling of the input and output lines requires a total of 62% of the total machine time for the 128 teletype and 16 data line configuration. Thus, a total of 95,000 instructions per second are available for the central processing functions.

The system analysis of the ITT 525 determined that to completely process the assembled messages, with an input line utilization of 100%, would require between 35,000 to 45,000 instructions per second. This processing consists of message validity checking, message decoding for destination and priority, message filing, message journalling, message code conversion and finally output queueing. Approximately 1500 instructions per message are needed to perform the complete processing functions. This processing estimate of 45,000 instructions/sec is rather conservative, since the probability of continuous 100% line utilization is very remote. Thus, the average processing time will be much smaller than the 45,000 instructions per second. However, the total available central processing time for the ITT 525 is 95,000 instructions per second so that, obviously, 50,000 instructions per second remains for future expansion or a further trade-off of time for hardware or flexibility.

To make further use of the extra machine time, it was decided to employ the concept of stored micro-operations or microfunctions. A reserved area of memory contains the microoperations for each machine instruction. This word is retrieved for each instruction before the execution of the instruction can proceed. This extra memory retrieval per instruction uses an equivalent of 31,500 instructions per second, so that a total of 63,500 instructions/per second remain for message processing. The stored microfunction logic replaces the conventional wired logic operation decoder and some corresponding microoperation logic. However, the primary advantage of this approach is not the reduction of hardware obtained, but in increased instruction flexibility and speed of machine check-out. Each microfunction may be tested independently either by a diagnositc program or from the operator's console. This facility greatly reduces the time required to isolate and repair machine failure.

In summary, the ITT 525 system design has resulted in the development of a stored program processor in which the memory is

time-shared between Input/Output wired logic and the program control logic. The processor operates internally on parallel binary words each consisting of thirty-two bits. The instruction cycle, consisting of 6 microseconds, performs single address instructions with an available rate of at least 63,000 instructions per second.

Machine Organization

The block diagram of ITT 525 VADE (Figure 1) illustrates the machine configuration consisting of a Central Processor and In-Out time-sharing the core memory.

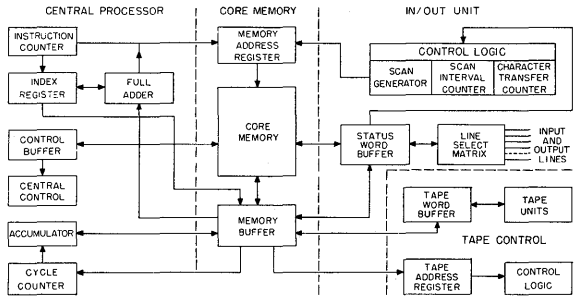
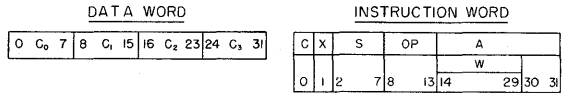


Figure 1. ITT 525 VADE

Central Processor

The central processor of the ITT 525 is a single address, binary, one's complement processor employing stored logic for instruction decoding, one index register and character mode operation. The design is based upon a minimum register configuration, with maximum time sharing, and direct transfer between registers.

The instruction cycle of the processor is six microseconds. This cycle is broken down into three memory accesses: one unload-load to fetch the instruction; one unload-load to obtain the stored microfunction control word; and finally one unload-load to obtain the operand and execute the instruction. The processor word consists of thirty-two bits which may take the form of 4 eight bit characters or an instruction word divided into six fields (Figure 2).



BITS	DEFINITION
0-7	CHARACTER 0 (C_0)
8-15	CHARACTER 1 (C_1)
16-23	CHARACTER 2 (C_2)
24-31	CHARACTER 3 (C_3)

BITS	DEFINITION
0	CHARACTER MODE (C)
1	INDEX TAG (X)
2-7	SENSIBLE DEVICE CODE (S)
8-13	OPERATION CODE (OP)
14-29	WORD ADDRESS (W)
14-31	CHARACTER ADDRESS (A)

Figure 2. ITT 525 Processor Words.

The Memory Unit consists of a high speed linear selection core memory operating at a speed of 2 microseconds per complete cycle. The size of the core modules varies from 4096 words at 33 bits to 32,768 words. An extra bit (33) is furnished, which enables parity checking during the unload cycle and parity generation during the load cycle.

The design of the processor registers may be considered conventional for the instruction counter, index register and memory address register. However, several unique features were employed in the use and design of the Memory Buffer, Accumulator and Control Buffer.

The Memory Buffer is a time-shared register which is concerned with normal memory functions plus other functions such as arithmetic unit buffering, character mode gating, In-Out mode buffering and behaving like a pseudo bus. During arithmetic and logical operation the arithmetic unit may be considered as the Memory Buffer Register and the Accumulator. The reasoning behind this approach is that the contents of the memory buffer may be utilized as the arithmetic unit "B" register, while the data is being loaded into memory. For example, in addition the Accumulator contains the augend and the Memory Buffer contains the augend. These operations may, furthermore, be performed in either the word or character mode. Special character gating between accumulator and memory buffer enable the programmer either to perform the operation on 32 bits or one of the eight bit characters. In the transfer of data from register to register, the Memory Buffer acts as a pseudo-bus, through which all data must pass. This configuration reduces redundant paths and allows one to form any data transfer path as desired. This capability is especially useful in developing new instructions consisting of several register transfers.

The accumulator register of the ITT 525 is the heart of the arithmetic unit. This register in conjunction with the memory buffer performs a parallel, two-step addition and subtraction. One of the unique features of the Accumulator is the carry chain configuration, which has a maximum delay of 425 nanoseconds.

The standard, simple, carry chain configuration consists of a single gate per flip-flop stage. The delay encountered for this arrangement is the number of stages times the gate delay, which for the ITT 525 would have been 32 x 35 or 1120 nanoseconds. In order to take full advantage of the four megacycle clock it was determined that a carry chain delay of less than 500 nanoseconds would be desirable for the ITT 525. One technique available to speed the carry chain is the pass-carry or grouping-carry idea. In this case, several stages are combined to form one large carry gate, thus, reducing the overall carry chain delay. However, in the ITT 525 Accumulator the carry chain design is based upon the group hierachy principle. This concept makes optimum use of the recursive nature of the carry equation by first combining flip-flops into groups and groups into sections. In this way, if a carry has to be passed for 32 bits, it will avoid not only the groups of flip-flops, but also the section of groups.

The functions that the accumulator may perform upon data are as follows:

1. Partial Add (Exclusive Or)
2. Carry
3. Inclusive Or
4. Reset
5. Complement
6. And
7. Cycle Left

The accumulator may be sensed by program for the following conditions:

1. Minus Zero
2. Plus Zero
3. Overflow
4. Any bit of Character 3 (24-31)
5. Plus or Minus Zero

The Control Buffer contains the Instruction Micro-operations obtained during the Processor's "Stored Logic" Memory cycle. Each bit of this register is assigned a specific microfunction, such as "Reset Accumulator," Transfer Index Register to Memory Buffer," etc. If a particular instruction requires that microfunction a "One" appears in that bit position. High fan-out drivers distribute these microfunctions to the various register input gates. In addition to the increased flexibility and some cost reduction, the use of the stored logic technique provides a powerful tool for checkout and maintenance.

INPUT/OUTPUT UNIT

With the single exception of a direct input from a paper tape reader on the console, all processor inputs and outputs are handled by the Input/Output Unit, including transfers between core memory and secondary storage devices. The initial implementation of the ITT 525 system has the following traffic-handling capability:

1. 16 duplexed high speed data lines operating at any speeds up to 2400 bits per second (8-bit code).
2. 128 duplexed teletype lines operating at speeds of 60, 75 or 100 words per minute (5-bit code).
3. Block transfers of computer words to one of eight magnetic tape units operating at a transfer rate of 2500 computer words per second.

This is a maximum capability configuration with regard to teletype and data lines. Smaller machine capabilities are implemented in any combination of modular blocks of 4 data or 16 teletype lines. Also, individual line speeds are completely independent and may be changed without incurring hardware changes.

Although the stated capabilities conform only to the task of communications processing, the unique features of the Input/Output Unit are applicable to other tasks and configuration requirements with a moderate amount of hardware change. The "bit-at-a-time" technique is easily adapted to various forms of serial bit streams, regardless of framing or Synchronization details, and the method used for tape word transfers is directly applicable to any block transfer process, even if the "blocks" are degenerate ones of only a few words of characters.

Teletype and High Speed Data Lines

Incoming serial bits on these lines are transferred directly into core memory. Outgoing serial bits are transferred directly from the memory to one or two per-line output flip-flops. Each output line requires one

flip-flop for pulse-stretching and data output lines require an additional flip-flop to reduce bit jitter. The total line storage required is 160 flip-flops, which compares favorably with the 1536 flip-flops required if each line (input and output) were to terminate in a one-character buffer.

Several fixed core memory locations are permanently assigned to each input and each output line which are used by the input/output logic. These per-line locations contain space for character assembly/disassembly, program flag bits, control bits and timing information. The stored program exercises control over the Input/Output Unit by performing regular scans of these words and changing their contents when necessary, thus modifying the operations of the wired logic of the Input/Output Unit. Specifically, in addition to noting the end of incoming messages or initiating output for outgoing messages, the program must make "bin" assignments to active lines. It is unfeasible to reserve for each line a space in memory adequate for the largest possible message. Alternatively, "bins" of 75 words, or 300 characters, are assigned to active lines as they are required. The Input/Output Unit logic notifies the program of such needs by flag bits and can store temporarily, in the fixed memory locations, as many as twelve incoming characters during the interim between bin assignments. In normal input operation, after a bin assignment is received, characters are transferred to memory soon after completion, independent of the stored program.

In reference to the block diagram of Figure 1, the basic operation of the I/O Unit is rather simple. A "Scan Generator" controls line selection and memory addressing (for control words) according to a fixed cycle of operation. Then, for each line scan, the most important control word for the line, the "status word," is unloaded to the "Status Word Buffer" where it remains for one or two more memory cycles to control operations on the line information through use of the Memory Buffer for examination and modification of other words. Finally, two counters are used for timing purposes indicated below.

The Input/Output Unit obtains control of the memory and performs a "scan cycle" every 280 microseconds. This interval is compatible in two different ways, with the bit periods of the lines. A 2400 bit-per-second data line has a bit length of 417 microseconds and a 100 word-per-minute teletype line has a bit length of 13.46 milliseconds. By scanning all data input and output lines each scan cycle but only one-fourth the teletype input lines and one-sixteenth the teletype output lines, the following rates are obtained:

1. data lines are scanned at least 1.49 times per bit.
2. teletype input lines are scanned 12, 16, or 20 times per bit for 100, 75 and 60 word-per-minute lines, respectively.
3. teletype output lines are scanned 3, 4, or 5 times per bit for 100, 75 and 60 word-per-minute lines, respectively.

These rates permit the sampling of teletype input lines within ±8% of the nominal bit-center, or better, to minimize the effects of distortion. Actual sampling is accomplished on the basis of predicted bit-center sampling times established when the "stop" pulse to "start" pulse transition is detected and stored in the fixed memory space for the line for later coincidence comparison with a real-time (based) counter. Data input lines are sampled on the basis of timing provided by their associated synchronizing signals. The synchronizing signal used has a frequency of one-half the bit rate of the line. The value of this signal (zero or one) is stored on each scan and the line value is not sampled unless the stored and present values of the synchronizing signal differ. Output lines are handled in exactly the same manner as input lines except that teletype output lines do not need the high scan rate provided for input teletype lines since the output process itself controls the waveform distortion.

During the I/O scan operations, one memory cycle is required to scan each line and two additional memory cycles are required for each character transfer between the fixed memory locations for the line and the message bin elsewhere in memory. By limiting the number of character transfers allowed in each scan, minimum and maximum I/O scan time requirements of 144 and 173 microseconds, respectively, are obtained. Since the interval between scans is 280 microseconds, 52% to 62% of total processor time is spent in input/output operations and 38% to 48% remains for stored program use (63,500 to 81,000 instructions per second).

Magnetic Tape Block Transfers

Block transfers of computer words between the memory and the Magnetic Tape Module

of the 525 are initiated by the stored program and executed in detail by the Input/Output Unit. A single fixed location in memory holds a block address and a count of the number of words to be transferred. During a block transfer, the MTM sends requests for word transfers to the I/O Unit at approximate 400 microsecond intervals and these requests must result in a word transfer within 67 microseconds. It is possible, then, that a word transfer must be made when the I/O Unit is not in control of the memory. In this case, the I/O Unit gains control of the memory only for the transfer time and then relinquishes control until the next regular line scan cycle time. While it is in control of the memory, the I/O Unit uses the fixed location MTM word to control the transfer of a word between the specified address and the MTM buffer. Then the address is incremented, the block transfer count decremented and the MTM word is transferred back to its fixed location. Much of the logic performing these operations within the I/O Unit is common to the logic required for teletype and data line operations, since character transfers and bin counts for these lines are handled on the same general basis. The I/O Unit is easily expanded to include a similar block transfer provisions for magnetic drums, card readers, punches, printers and displays.

Except for the implementation of the block transfer process, there is nothing unusual about the operation of the magnetic tapes. One tape at a time may be selected to read, write, backspace one record, advance one record, write end-of-file or rewind, the rewind operation being performed in a quasi-off-line state so that other units may be selected during this operation.

Input/Output Program Requirements

Since the ITT 525 has no interrupt feature, the stored program—input/output interface is an unusual one. Regular scanning of the fixed—location input/output control words is essential to the bin assignment task of the program. Input teletype words must be scanned at least once every 800 milliseconds and input data words at least once every 40 milliseconds. These figures represent the amount of time required for incoming information to fill the twelve-character per-line temporary storage space. Output words are scanned (for bin assignment needs) at whatever speeds the programmer desires since no information can be lost and the only consideration is for efficiency in transmitting messages which are more than one bin in length.

A more complicated problem arises when the program must exert control over I/O Unit operations by modifying the contents of fixed location control words. An input/output line scan cycle may interrupt the program and change the contents of a control word at the same time that the program is preparing to modify the word. Since the program has no natural means of knowing an interruption has occurred, it would tend to force obsolete data into the control word. To circumvent this problem, a flip-flop is provided which can be sensed by the program and which, if on, guarantees the program that the six instruction times immediately following the sense instruction will be free of I/O Unit interruption—this number of instructions being sufficient to perform the control word modification.

Duplexing

To meet the reliability of many real-time problems, the ITT 525 may be utilized in a duplexed configuration. The duplexing design of the ITT 525 has been selected on the basis of maximum reliability and minimum special purpose duplexing hardware.

The Duplexed System configuration is illustrated in Figure 3. System A is in control of the magnetic tape and communication links, all input lines and output lines are accepting and sending data. The standby machine B also has the input lines connected to it and accepts all input messages. Furthermore, machine B assembles, processes the message and sets up the output queue. Machine A regularly sends data to machine B via a

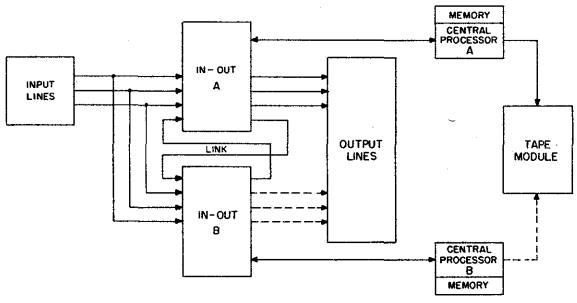


Figure 3. Duplex Configuration.

normal high speed data line concerning the disposition of messages. Once machine A has outputted the message, machine B erases the message and updates its own output queue. Machine B does not file, journal or overflow, or output any messages. Using one of the high speed data links for the regular communication between machines A and B insures a smooth cutover with no loss of data. The worst condition that might occur is that after cutover machine B might output a given message again since it had not received the last disposition data.

Machine A, if in control may by program relinquish control to Machine B and vice versa. Also, manual means are available to establish the duplex configuration via the Operator's Console. Output lines and the magnetic tapes are automatically switched into the proper machine for each configuration.

CONCLUSION

The ITT 525 VADE is intended to do a medium-scale job using only a small-scale amount of hardware. Although testing and program debugging will not be complete for another two or three months, there is little doubt that the system will satisfy this aim.

Further extensions of the VADE approach have been planned which will improve the speed, line-handling capacity and versatility of the machine by modular additions of hardware at selectively increased cost.

ON THE REDUCTION OF TURNAROUND TIME

H. S. Bright and B. F. Cheydleur
Computer Division
Philco Corporation, a Subsidiary of Ford Motor Company
Willow Grove, Pennsylvania

SUMMARY

Objective: To Reduce Delays. It is the intent of this work to permit small computing jobs to run with typical delays of minutes rather than hours, while no jobs, including the largest ones, become appreciably worse in turnaround time than at present.

Background. The basic idea of multiple-break-in operation from many input/output stations is not new. Most authors, however, have proposed either dramatic advances in hardware or software, or computation complexes of conventional hardware so large as to be economically unattractive. McCarthy, for example, proposed serving some dozens of stations for simultaneous on-line debugging of as many programs, by using perhaps a million words of slow magnetic core memory with a very fast computer.

Sources of Delay. "Legitimate" delays, for jobs to be run as entities in the sequence in which received, consist primarily of queue development during periods in which the average workload acquired exceeds the computation rate capacity of the facility. "Illegitimate" causes of delay result mainly from manual job stacking. Artificial delays are inserted at several places in a typical facility, including the sign-in-desk, the card-to-tape facility, the on-line input tape stack, the on-line tape units (gross operational delays from the mounting of file tapes while the computer system idles), and the printer tape stack.

Resources:

(a) Flagging, in the procedure oriented language, of the permissible break-in points on large programs.

(b) Sequential, rather than concurrent operation of programs, by means of fast exchange of more contents with disc.

(c) Use of main core as input/output buffer for short communications with multiple remote stations.

Approach. In principle, the operator system is to be increased in capability for minimizing delays, through application of currently available hardware, together with planned interruption of long runs by short ones.

For small jobs, break-in on large jobs at selected interrupt points. For large jobs, stack input/output on disc. Sequence primarily by estimated run time, with some consideration of priority, and with less attention given to arrival chronology.

The paper describes a typical large relaxation calculation, giving operation parameters as executed on a modern computer, showing for this rather formidable example a break-in-point interval on the order of several seconds.

In jobs of such size, total tape and disc traffic can be comparable in volume to internal data flow. In contrast, the concurrent buffering of set-up information for many problems constitutes a relatively minor contribution to total data flow. Thus the initial input and final output for many jobs may

proceed concurrently, although calculations will in all cases be executed sequentially.

Conclusion. The paper marshals arguments supporting the practicality of greatly reducing turnaround delays without using huge memory or very costly types of communication facilities.

The productivity of all direct users of large-scale general-purpose digital computing centers, and to a lesser extent the productivity of the entire organizations they serve, are significantly affected by the typical time delay between request for and delivery of computer service, which we shall call "turnaround time."

For this reason, reduction of turnaround time has in recent years become recognized as having major economic importance. As machines have become faster, individual problem setup time has assumed larger significance in computing center logistics. Some of the delays for clerical work at setup time have been taken over by operator programs. Much of the effort on delay reduction has been applied to attempts to increase the effective throughput capacity of the computing systems themselves, either by concurrent operations or through increase in sheer speed.

One approach that has been widely used, as a matter of absolute necessity, for the handling of real-time problems within general-purpose facilities, has had surprisingly little attention in "unreal-time" applications. The intent of this paper is to direct attention to the technique of short-run break-in by programmed interrupt, and to show how modern hardware, without the costly special facilities often required for prompt interrupt, can make this method attractive for general-purpose applications.

We believe that the method, which uses only off-the-shelf hardware and software, can permit many short jobs to run with typical delays measured in minutes rather than in hours, while no jobs (including the longest ones) become drastically worse in turnaround time than in conventional first-in, first-out operation.

Throughput Increase

Before proceeding with our discussion, it will be useful to review some of the steps that have been taken to increase the effective capacity of general-purpose computing facilities:

1. Concurrent Schemes (cohabiting programs)
   - 1.1 Micro-segmentation by commutating hardware
   - 1.2 Decentralization by input/output autonomy
   - 1.3 Macro-segmentation (program segment merging by hardware interrupt)
     - 1.31 Merged input/output, sequential execute (several concurrent I/O streams permitted, but only one computation at a time has control)
     - 1.32 Merged input/output and execute (full-blown "multi-programming")
     - 1.33 Sequential input/output, merged execute (early real-time operations on unbuffered machines)

2. Sequential Schemes (programs alone in memory)
   - 2.1 Multi-phase operation (batched input, execute, output ("I, E, O")) [1].*
   - 2.2 Faster machines
     - 2.21 Sequential integral programs
     - 2.22 Short-run break-in by program interrupt

3. Multiple Independent Machines

The concurrent schemes suffer from the serious disadvantage that, even in multiple-computer-unit complexes (whether or not all of the available memory space is accessible by all processors) sufficient main-memory space must be available for all of the programs or program segments that are to be operated together, if the operation is to be economically feasible. This often means that either the program multiplexing is limited to jobs that require very little memory space, or that memory sizes are required that are economically unattractive at present.†

Method 1.1, in present realizations, has the additional disadvantages that both time and memory space segmenting must be

*Batching of input, execute, and output phases of all jobs on an input tape is discussed in detail in reference [1].

†One proposal, [4], called for a single computer system with one million words of magnetic-core memory.

relatively simple and inflexible. It increases turnaround time for all processor-limited problems that are run concurrently, since the single central processor must be time-shared and all such jobs must take longer than when run seriatim.

All of the Concurrent Schemes shown above permit efficient use of multiple on-line input/output devices, but the sharing of a single I/O device by several problems is at present feasible only if the "device" is actually a large random-access auxiliary memory element or if Data Select* hardware facilities are available.

This is particularly significant when scheduling multi-tape problems on a large machine; many of those jobs for which concurrent operation would be most attractive require the use of half or more of the total number of tape units available, especially when tape-oriented operator systems are used. This limitation on time-sharing of a single I/O device is, alas, almost as frustrating for the modest scheme proposed by the present paper as for the most sophisticated time-and-corespace-merging scheme discussed. Its effect is to impose serious limits on the permissible assignments of tape units or other I/O devices, for jobs that are to be run concurrently in any system, and in most cases to prohibit reassignment of a given I/O device until completion of the job to which it was last assigned. The noteworthy exception is the case of tape units used for "scratch" storage of intermediate results; such units may be reassigned as soon as the last Read operation upon a given data string has been completed, although the reassignment problem is a difficult one in the important case when the number of rereads is dependent upon calculation results and must be determined at run time.

In the proposed scheme, the effect of this tape assignment restriction is merely to hold back the start of an interrupting job until adequate I/O facilities can be assigned. Thus, when an interruptable job that has extensive I/O unit requirements is running, only those jobs that can be accommodated on the remaining I/O devices can be permitted to get to the head of the interrupting job stack.

(A comment on semantics is in order here. Many of the early papers on multiprogramming, and a few recent ones, seem to consider concurrency of I/O with computation (Method 1.31 with the restriction that all of the I/O activity relates to the execution of one program) to constitute multiprogramming. We feel that "buffering" is the accepted term to be applied to such concurrency, as long as a single main program (which of course may be controlled by an operator program and from time to time by an arbitrary number of subprograms) has control of the machine. We consider multiprogramming to mean "concurrency of the execution phases of two or more unrelated programs".)

Among the sequential schemes, Method 2.1 was clearly not designed with turnaround time in mind, since it normally increases turnaround time for all jobs in a batch.

This scheme, commonly known as "three-phase" operation, was intended to save time by avoiding repeated loading of large input and output routines. It operates by first preparing ("I" Phase) the input from all jobs on an input tape; then ("E" Phase) executing all of these jobs; and finally ("O" Phase) preparing output (printer tape) for all jobs. The usual operating option of assigning special output tape(s) at run time, in order to take care of priority situations or to take advantage of a temporarily short printer job queue, is not available without a complete change of operating procedure back to "single-phase" operation, the normal scheme in which a single job is processed from start to finish. Thus, in true three-phase operation, the output from all jobs on a given input tape is delayed until the last job has been completed.

Method 2.21 is the one that comes under fire in the classical justification for effort to be expended upon development of operator programs. It is clear that, as times for Input, Execute, and Output become vanishingly small on a well-balanced very-high power machine, one could conceive of a facility in which most of the time was spent in program setup, startup, and wrapup. In a typical large-scale present-day facility, in fact, brute speed alone cannot accomplish much reduction of turnaround time. The scheme

*This feature permits individual data records on magnetic tape to be tagged with control marks so that they can be processed or skipped without detailed examination; it is most commonly used for the writing of multiple reports on a single tape by a single program, so that report selection may be made at the time of off-line printing.

proposed below permits setup information accession to be concurrent for many jobs, while startup and wrapup can occur very quickly under program control.

Because of the remarkable advances that have been made recently in machine power and relative economy, a few words in retrospect will serve to underline the significance of the preceding paragraph. A typical high-power modern machine has 2000% to 5000% more computation capability, and 2500% to 7000% more input/output capability, than the vacuum-tube machines (circa 709 and 1105) that ushered in the concept of the integrated data processing facility using parallel binary arithmetic and buffered magnetic tape. Clearly, such huge increases in capacity, and in computation-per-dollar if the machines can be kept occupied, call for serious re-examination of our operating methods.

Method 3, which is the addition of entire computer systems, has represented sound management practice during the first two generations of large machines (the last of the vacuum tube machines and the first of the transistor machines), at least in those installations where unscheduled delays of more than a few hours might be prohibitively costly; if not having on-site backup hardware can be more expensive than that hardware would be, then the extra hardware is justified irrespective of capacity considerations.

Because third-generation hardware will be much more predictable as to its ready-willing-able condition (i.e., unscheduled down-time will be greatly reduced), and because maintenance experience on second-generation machines has taught design lessons that should dramatically reduce time required for scheduled maintenance, it seems reasonable that hardware unpredictability will in the few years to come offer less justification for parallel facilities. The large user who has had the advantages of more than one machine will, thus, in many cases consider conversion to a single, more-powerful machine in which overall hardware economy (computation per hardware dollar) can be better. There will be, from this class of user, intense interest in means for achieving the excellent traffic-handling behavior of the multiple-machine facility in a larger-single-machine facility. Since this user will not be willing (and in many cases will not be able) to submit to the restrictiveness of the concurrent schemes, we feel that only Method 2.2 will meet his needs with any degree of success.

Macro-Segmentation in Practice

In many installations, the basic hardware configuration is determined by the requirements of a single class of "bread-and-butter" problems. With such problems running in the system, there will not be much excess memory space or processor capacity available. The macro-segmentation scheme is a means for scheduling the available excess capacity.

If problems could be segmented so precisely that the onset and duration of memory, processor(s), and I/O device availability could always be matched precisely with the demands of other problem segments, then parallel operation could permit complete use of the entire machine. This does not appear to be workable in the real world.

In practice, a useful degree of approximation to that ideal can be achieved if the problems, major and minor, are macro-segmented at compilation time so that the incidence of spare capacities in various subsystems, instead of being pre-computed, may be continually tested by control hardware and assigned at execute time, in vivo.

Such a strategem requires the prescripting to every macro-segment of a precis of the I/O and memory requirements of that segment, a signalling of activity-completion from each I/O device to the executive program, and the continued monitoring of the problem programs at the macro-segment level.

This scheme is being implemented for several machines in the U.S. and in England, notably in the English Electric KDF.9 which is described elsewhere in these proceedings.

Particular notice should be taken of the recent work reported in reference [6], by Corbató, et al, describing an application of Method 1.32. Their thoughtful comments on several aspects of multiprogramming system requirements and planning have inspired much of the work reported here, and the reader is referred to that paper for valuable background information on program time-sharing of hardware. Their algorithm for run queue control will be discussed briefly below, and several references will be made to observations in that paper.

Short-Run Break-In

The basic method proposed here is much simpler conceptually, and offers advantages shared by none of the other methods listed except those that utilize sheer power alone. In a sense, it is a simplification of the macro-segmenting concept outlined above.

The segmenting is to be performed in large programs only, under control of flags planted by the programmer. This will require establishment of a programming convention for maximum on-line time interval between flags, which for large machines might be chosen on the order of a few seconds to a few minutes.

Jobs whose maximum machine time requirement is smaller than the maximum permitted interval between flags will not be segmented, and it is these jobs that can be called in by the operator program whenever a break-in flag is encountered.

While we do not have the temerity to essay a rigorous proof that any particular break-in time limit is a reasonable one for all circumstances, it will be helpful to consider one example of a notably forbidding class of problem in which data flow to and from auxiliary memory proceeds concurrently with rather involved calculation and indexing. In the inversion by relaxation methods of large sparse matrices, it is prohibitively expensive of restart time to interrupt the calculation during a mesh sweep. As each new sweep starts, however, a substantial amount of initialization is performed; it is not unreasonable to request that auxiliary memory data flow be organized for efficient interruption at these points. For instance, tape data can start new blocks, so that, at worst, tapes may require simple backspacing in the event of interrupt at such a point; disc data flow may start a new I/O order at these points.

Consider a tridiagonal matrix of order 100,000 that represents an array of difference equations, calculation for each point to consist of eight accumulative-floating-multiply operations together with a few housekeeping operations. On a typical modern large-scale computer* the floating-load-multiply-add sequence may take 7 microseconds. Ignoring the brief housekeeping operations, the time for this mesh sweep would be 5.6 seconds. Thus, imposition of the one-minute rule would afford no hardship to the programmer of this large problem. Clearly in the formalization of problems that are even larger than this one, sectionalization into relatively autonomous parts is a sina qua non of rational construction and rational problem checkout. The run duration for these sections will tend to be far less than a minute.

Thus the feasibility of the strategem (2.22), wherein a major problem occupying most of one critical facility must be displaced, sectionally, to reduce turnaround time for a minor problem, is assumed to be dependent only on the means for dumping core memory, etc., into a high-data-rate "scratch medium" such as drums or discs.

From the standpoint of turnaround time, the availability of modern discs suggests that the complete loading of a number of problems can be made from cards and tape to discs well in advance of actual processing. All short-run segments and all problems such as usually require five minutes of machine room set-up time and one or two seconds of run time, can now be processed ambulando. It should be noted that the loading of information in advance of processing each problem segment can be effected automatically from discs more rapidly and with less entailment of control equipment, via Method 2.22, than would be the case with the macro-segmenting (Method 1.3), for the passage-time of macro-segments is not well matched with the access time of discs and is even more badly matched with the access times of tapes.

Altogether, from considerations of simplicity of Method 2.22 and of the tanking advantages of discs, it seems quite practical to permit small jobs to interrupt large ones and to thereby implement a first level of priority for jobs of short estimated run time.

Memory-Protect Considerations

With regard to the ubiquitous problem of memory protection (which let us discuss in the limited context of protection of the Operator System program from being overwritten by a not-yet-debugged user program), Corbató [ibid.] suggested dynamic relocation of all memory accesses that pick up instructions or data words. This, in a true multiprogramming system, would consume significant machine time on a computer that did not have rather extensive specialized control

*e.g., the Philco 212.

hardware. With the straightforward scheme proposed here, memory protection can be adequately provided by the addition to a conventional machine of simple boundary registers. For the protection of I/O unit assignments, Corbató [ibid.] suggested the trapping of all I/O instructions issued by user programs; under the scheme suggested here, this would be necessary only for the interrupting (small) programs.

Control of Precedence

Assuming that overall review and authorization of problems provides all the filtering needed in a facility except for within-shift scheduling, and further assuming that short-run break-in is adopted and discs are utilized, it becomes necessary to answer the question, "how much running-time should be allowed for small problems before automatic program-reversion to large problems is permitted?" Should the parameter be fixed or variable? Should it be some interval that is greater than a few seconds and perhaps less than five minutes? Is this range too large?

Using modern high-flushing-rate auxiliary memory equipment, one can save and replace the contents of main memory in less than one second, even on a fairly large computer. Consider the example of a 32,000-word core memory machine equipped with a disc backup memory that positions in a maximum of 100 milliseconds and communicates data at the rate of 119,000 words per second (8,192 words per 68-millisecond revolution), with angular delay of no more than one word-time when data words are moved in groups of 8,192 or more. Assume that the disc heads have been prepositioned to a "home" position by convention at a flagged break-in point in a large program. Time to refresh main memory would then be, at most:

$$T = \frac{32{,}768}{119{,}000} \text{ second} + 0.1 \text{ second}$$

$$+ \frac{32{,}768}{119{,}000} \text{ second} = 0.65 \text{ second},$$

approximately.

Among the parameters of priority, those that are dependent on equipment required for each problem segment become less important in a computer complex in which high capacity discs are included, because the tape-drives that would be required to serve as scratch media, in classical complexes, are now replaceable with areas on the discs. Likewise in file updating, even the current-changes may be kept on discs, as well as the problem-program and the library. Thus, except for the propriety of using tapes as a medium for large history files or reference files, the functions of tapes are apt to be supplemental to those of discs, rather than vice versa. One such supplemental preference for tapes with respect to discs inheres in the two or three millisecond access time to the beginning of information blocks that are already in position for the next reading or writing action. On the whole, however, the criticality of I/O availability is considerably reduced when modern discs are available, and the number of essential availability parameters that must be used in a scheduling calculation is very small.

When there are several problems loaded into the tape or disc stack, the selection of the next one to be processed can be based on a calculation that takes into account the estimated run time. An early, perhaps whimsical scheme that considered e.r.t., among other variables, was the North American (Aviation) "Precedence Program" circa 1955.

NAPP controlled the job stack on an IBM 701 by considering the four factors U = Urgency, W = Wait time (since problem submitted, B = Business this customer gives the computing center per month, and r = run time estimated for this problem. For each waiting problem, the program calculated priority and chose the problem having the highest value of P to be run next, according to:

$$P = \frac{WUB}{r}$$

The value of U was set by reference to a table established by laboratory management and changed from day to day or perhaps from hour to hour. The parameter B was inserted in order to provide an appropriate indication of the loudness with which this customer was able to knock on the computing center door. The usual first-come, first-served sequencing convention may be looked upon as a degenerate form of this formula, with U, B, and r held constant.

We propose that two of the above four factors, Wait Time and Estimated Run Time, be considered in addition to I/O units required, in establishing run precedence. We do not propose to consider memory space requirements, since this scheme does not require cohabitation of running programs in main memory. We also propose to provide some weighting other than linear for the two times, thus:

$$P = n \log W - m \log r,$$

where n and m and weights given to Wait Time and Run Time respectively.

So far as turnaround time is concerned, the responsibilities of the Executive Program can be summarized into three classes:

(a) The computation in advance for each problem in the input stack of a precedence number, taking into account the parameters of priority.

(b) The anticipation, through survey of the estimates of running time of the status of the queue, giving advance notice to operators of when the backlog of little and big problems is to be replenished.

(c) The providing of advance notice to operators of need to set up reference file tapes, during the advance of a large problem from segment to segment, in accordance with the computations of (a) and the intrasegment directions to operators provided by the complier.

For the case of a true multiprogramming operation, with some scheme for sequencing small time-segments of user programs, system efficiency can approach zero for heavy workload when for some large programs the loading time becomes large compared to the run time segment length. Corbató [ibid.] proposed a scheduling algorithm that guaranteed an operating efficiency of at least 50% by keeping segment operate time equal to or greater than load time, and pointed out that one may determine the longest loading delay among a number of competing program segments and that, for a given "segment delay," the number of users must be limited. Unfortunately, in a typical computing center environment, it is the completion of a job rather than the start of its execution that is of interest; completion time does not seem to us to be predictable in the general case.

Under the proposed scheme, on the contrary, provided good discipline is maintained with regard to insertion of flags in interruptable programs and to limitation on the duration of interrupting jobs, it is possible to permit prediction of worst completion delay, or turnaround time limit, in terms of the number of short jobs waiting, by merely assigning a weight of zero to the coefficient m in (b) above, as executed by the operator program. This limit, for j jobs waiting, would be simply $j(t_f + t_i)$ where t_f is the maximum permitted time between flags and t_i is the maximum permitted time for any interrupting job.

We feel intuitively that it would be desirable to experiment with weights for Wait and Run time coefficients n and m. For the facility that serves only a few dozen short-run users, it might be best to weight Wait time much more heavily than Run time, thereby approximating what Corbató calls "round robin" service; for the facility that serves a very large number of users, mean turnaround time must be greater and it will be desirable to favor short jobs by weighting Run time heavier.

Since, in this system, the criticality of I/O equipment scheduling (so far as tapes are concerned) is relaxed, some of the complexities that would enter into a general scheduling model are not present. Thus, for any problem in the stack, it becomes feasible to automatically examine the remaining factors, complying with the residue of considerations in a model such as given by J. Heller [2].

In this system, no special reprocessing of object languages is required in order to conform memory allocation to the ongoing problem-mix decisions. Furthermore, we require no real-time solution of linear models of flow or loading such as that reported by Totschek and Wood [3], nor are surrogates for solutions of these models needed when disc storage is present to provide cushioning.

The Executive Routine is relieved of responsibility for micro-monitoring of error concatenations that can thread across a set of problems and a complex of equipment for only one large problem segment is processed at a time; problem independence is fostered. Thus, in modern computers, the interdependence of hardware within any problem segment can be kept at a reasonable level while maintenance and problem debugging are simplified. In partitioned and buffered memory computers, i.e., those incorporating several

memory modules having independent data and address registers, the inherent parallel capability is thereby conserved so as to contribute to speed of processing. This is in sharp contrast to the usual situation in micro-segmentation schemes, where memory partitioning complicates inter-job control and contributes to program control ricochet.

Implications

The basic idea of multiple-break-in operation from many input/output stations is not new. Most proposers, however, have advocated either dramatic advances in hardware or software, or computation complexes of conventional hardware so large as to be economically unattractive. McCarthy and associates [4], for example, proposed serving some dozens of stations for simultaneous on-line debugging of as many programs, by using an enormous slow magnetic core memory with a very fast processor complex. One of their principal concerns was to provide on-line responsiveness in the system to any set of queries or inputs emanating from the array of program-development stations, so that it seemed that all problem materials must be immediately accessable in directly-addressable memory. We believe that a variation of the Strategem of (2.22), adopted for high-speed processors and discs, could serve most of these requirements, particularly when individual groupings of problems can be controlled by individual executive routines, with occasional call-out from one group to another.

In this connection, the peak memory traffic load level reached when transmitting ten characters per second per station to or from 100 stations simultaneously, reaches a character rate of only 1000 per second, or 1/250 of the load that a modern tape unit imposes on a single I/O channel. In a current model* commercially available computer with each of four independent memory modules operating at one microsecond full cycle, and assuming that one full word of memory would be accessed twice for each character incoming from these control stations, this loading would entail 2/250/4 = 1/500 or 0.2% of the full memory capability. Clearly, the on-line query of raw information from a hundred or so stations is not a time-consuming process for this simple memory complex.

*Philco 212.

The Many-Short-Jobs Workload

Clearly, a production-computation workload that consists entirely of one-minute jobs is not going to be expedited by a traffic-handling scheme that emphasizes short jobs at the expense of long ones. This is a real limitation, for there are many organizations in which much of the daytime workload consists of brief compile-and-execute jobs.

Even for such organizations, however, there may be a powerful advantage in the method of short-run break-in. As discussed in the previous section, the economic soundness of use of main memory as a buffer for a multiplicity of input stations appears to be evident.

Most large computing centers would be capable of serving an enormous number of additional users if the minimum time per job were sharply reduced. In a typical present-day operation, jobs much shorter than one minute in extent are relatively few in number, because people having such work to do can get it done more promptly in other ways.

With the possibility of achieving reasonably good efficiency for jobs requiring one second or less of large-scale machine time, a whole new class of user becomes vulnerable to the wiles of the numerical mountain-mover. It is not difficult to conceive of several thousand jobs per day being done for the technical staff of a large laboratory, provided there are a large number of input stations conveniently located in the manner of reference [4].

In passing, it should be noted that one second (net—ignoring setup, startup, and wrapup) of machine time in this age is an item of not inconsiderable potential value. When used for such a mundane task as generation of a table of values for an implicit function, for example, it could accomplish the equivalent of months of hand calculation.

Perhaps more to the point, the availability on a few minutes notice of a tool of such awesome power can encourage "calculation, not guesstimation" for problem sizes which would otherwise not be served at all.

CONCLUSION

We have endeavored to show that the conceptually simple scheme of short-run

break-in can permit turnaround time for brief computing jobs to be reduced drastically without substantial increase in time for any jobs, including the longest ones. In particular, we have pointed out how one of the basic objectives of the McCarthy, et al proposal, to make feasible nearly simultaneous access by many people to a large computer, can be met through the application of presently-available hardware and presently-designable software.

REFERENCES

1. Mock, Owen and Swift, Charles, J., "The SHARE 709 System: Programmed I/O Buffering," J. ACM 6, 2, April 1959.
2. Heller, J., "Sequencing Aspects of Multiprogramming," J. ACM, 8, 3, July, 1961.
3. Totschek, R. and Wood, R. C., "An Investigation of Real-Time Solution of the Transportation Problem," J. ACM, 8, 2, April, 1961.
4. McCarthy, John, et al., "Report of the Long Range Computation Study Group," (private communication), Massachusetts Institute of Technology, Cambridge, Massachusetts, April, 1961.
5. Greenfield, Martin N., "Fact Segmentation," Proc. 1962 SJCC, pp. 307-315.
6. Corbató, F. J., et al, "An Experimental Time-Sharing System," Proc. 1962 SJCC, pp. 335-344.

REMOTE OPERATION OF A COMPUTER BY HIGH SPEED DATA LINK

G. L. Baldwin
Bell Telephone Laboratories, Incorporated
Murray Hill, New Jersey

N. E. Snow
Bell Telephone Laboratories, Incorporated
Holmdel, New Jersey

INTRODUCTION

One promising means of attaining data transmission speeds high enough to be effective with present day computer operation is the use of wide band facilities provided by the Bell System TELPAK service offerings. Almost every industry large enough to make use of a computer also has need of large numbers of voice telephone circuits between centers of operation. Quite often these circuits are provided by a TELPAK channel. Alternate use of the entire channel in a continuous spectrum data transmission system not only makes high speeds possible, but in many cases economically attractive.

With the establishment of a new installation at Holmdel, New Jersey, Bell Telephone Laboratories had an excellent opportunity to make use of and evaluate an experimental data transmission service, using a TELPAK A channel. A TELPAK A service with appropriate terminal equipment, can be used as twelve voice circuits or as an equivalent continuous spectrum wide band channel.

Installation of the system was completed and routine operation begun in February, 1962, giving the Holmdel Laboratories rapid access to an IBM 7090 computer at the Murray Hill, New Jersey, Laboratories.

This paper presents first, a description of the system, and second, the concepts under which it was devised with an evaluation of the operational results. As a part of the evaluation, an attempt is made to point out the limitations in usefulness of such a system and the inherent qualities, both good and bad.

Description of Experimental System

A functional block diagram of the data transmission system is shown in Figure 1. Basically it is a magnetic core to magnetic core system used primarily for tape-to-tape transmission. IBM input-output and transmission control equipment are utilized with Bell System experimental data sets, prototype N-2 telephone carrier terminals, and a specially engineered type N-1 carrier repeatered line facility. A discussion of each of these system components follows.

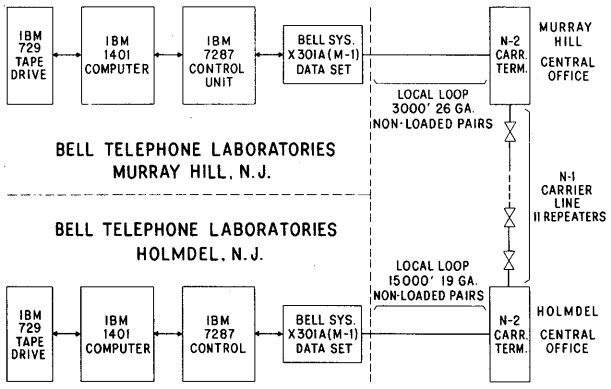


Figure 1. Block Diagram-Experimental Murray Hill-Holmdel Data Link.

Data Link Input-Output Equipment

Tape drives are of the IBM 729 type operating under control of IBM 1401 computers. Both computers and tape drives are used in routine data processing operations when not connected in the data transmission configuration. While transmission is in progress, the tape drive at either end of the system is under control of the local 1401 and subsequently the transmission control unit, an IBM 7287 Data Communication Unit.

The system is designed for transmission of data records or "blocks" with error detection and reply between transmissions. Data is read from magnetic tape one record at a time. Parity checks are made as the data is read into the 1401 core storage, from where it may be clocked at a synchronous rate for transmission.

The IBM 7287 transmission control unit is arranged in the system to clock data from the 1401 storage under control of the timing signal supplied by the data set (timing could be supplied by the 7287 if not supplied by the data set). Upon receiving data from storage the 7287 performs parity check, code translation from seven bit to four-out-of-eight fixed count, serializes the data and delivers a binary dc signal acceptable at the data set interface. In addition, for each record transmitted, it generates and adds record identification, start-of-record, end-of-record, and longitudinal redundancy check characters.

The receiving 7287 receives serial data from the data set, performs character and longitudinal error detection, code translates back to seven bit characters and delivers in parallel to the receiving 1401. Upon completion of receiving and checking a record, a digital control signal is returned via the reverse direction of transmission to the transmitting 7287. The control signal identifies the record received and indicates that the record passed all error checks or failed and should be retransmitted. At this point the transmitting end may be in either of two conditions. In the first operating mode the 1401 may have already read the next record from tape and transmission may continue if no error is indicated. If an error is indicated, it is then necessary to back the tape up two records and read the record again for retransmission. The second operating mode (determined by choice of 1401 program) holds each record in 1401 storage and continues retransmitting until a "no error" reply is received, and then the next record is read from tape. The more efficient mode of operation is, of course, dependent upon the transmission error rate.

When a record passes all 7287 error checks and parity checks in the receiving 1401, the record is delivered to the receiving 729 tape drive, completing the tape-to-tape transmission.

In order to achieve maximum efficiency it is necessary to maintain character synchronization continuously in both directions of transmission, rather than re-establish synchronism for each record or reply transmission. To accomplish this, the 7287 transmits periodically (once each 500 milliseconds) a short interval (approximately 10 milliseconds) of a character synchronization pattern either between record transmissions or between reply transmissions. Continuous repetition of a synchronization pattern must receive careful design consideration (as will be explained) for transmission interference reasons in this or similar systems.

Data Set

Experimental Bell System X301A (M-1) Data Sets used in the system are designed for serial transmission at a synchronous rate of 42,000 bits per second. The principles of operation are the same as in the Bell System 201A Data Set (commonly referred to as a "four phase data set") currently providing DATA-PHONE service on voice circuits.

The data set employs quaternary phase modulation with differential synchronous detection. Data delivered serially to the transmitter is encoded two bits (a "dibit") at a time into a phase shift of an 84 KC carrier. For the four possible dibits (11,00,01,10) the phase of the carrier transmitted during a dibit time interval is shifted by 1, 3, 5, or 7 times $\pi/4$ radians with respect to the carrier phase during the previous dibit time interval. The 21,000 dibit per second modulation results in a line signal spectrum symmetrical about the carrier frequency in the 63 KC to 105 KC band.

At the receiver, dibit timing is recovered directly from sideband components of the line signal. This timing is then used in the demodulation process. Data is recovered by detecting and decoding the phase relationship between the previous dibit interval of line

signal (available from a one dibit delay line) and the present dibit interval of line signal. The recovered data with a synchronized bit timing signal (generated from the recovered dibit timing) is then delivered at the receiver output.

Data, timing, and control circuits all appear on one ganged coaxial connector on the rear of the data set chassis. Interface circuits are designed to drive low impedance (90-120 ohm) loads or to terminate similar circuits.

Operation of the X301A (M-1) Data Set differs somewhat from that of voice band sets using the same modulation technique. It is undesirable, in the idle condition, to transmit a signal corresponding to a repeated bit pattern. Under this condition the line signal spectrum contains high level single frequency components which may result in crosstalk into other telephone carrier systems operating in the same cable. At the same time it is desirable to maintain continuous bit synchronization, requiring continuous transmission of a line signal. A compromise solution is necessary.

The data set interface provides Send Request and Clear to Send control circuits. A Send Request "on" signal presented to the data set results in a Clear to Send "on" signal being returned to the data source device, and data will then be accepted on the Send Data interface circuit. When the Send Request and Clear to Send control circuits are in the "off" condition, the data set will not accept data on the Send Data circuit but generates a line signal automatically, corresponding to a repeated "1000" bit pattern correctly related to the dibit timing signal. This results in the most desirable (lowest level single frequency components) "idling" line signal possible. Continuous transmission also enables the receiver to maintain bit synchronization between data, reply, or character synchronization transmissions.

N-2 Carrier Terminal

Telephone carrier terminals used in the system are prototype models of the type N-2 transistorized system designed to provide twelve two-way voice channels. A prototype N-2 data channel unit replaces plug-in voice channel units. The system then handles a two-way wide band data channel. The data channel unit serves to adjust signal levels and modulate the data signal from the data set into the frequency band vacated by the voice channels removed. The data channel spectrum is then modulated by the N-2 terminal group circuitry into the proper frequency band for transmission to the carrier line.

N-1 Carrier Line

The type N-1 carrier line utilized for this system is of the same type widely used in providing carrier telephone circuits throughout the Bell System, insofar as equipment and cable facilities are concerned. This particular line is specially designed to minimize noise (e.g. short repeater sections). The same design is required in many military services. Not all N-1 carrier lines in service meet the necessary noise requirements, but with additional engineering and construction effort can be made to do so.

The N-1 line between Murray Hill and Holmdel, New Jersey, is approximately thirty miles in length, short enough so that no phase or amplitude equalization is required. It is estimated that equalization will become a necessity in the order of one hundred miles of repeatered line.

Transmitted signal levels into the N-2 terminal and N-1 carrier line must, on a particular system, be a compromise between deriving a signal-to-noise ratio yielding satisfactory data error rates and keeping interference into adjacent systems in the same cable at a minimum.

Local Loops

Data signals are transmitted from and received at the data set over non-loaded telephone cable pairs. The experimental system described here utilizes 3000 feet of 26 gauge pairs between the Murray Hill Laboratories and the Murray Hill central office. At Holmdel 15000 feet of 19 gauge pairs are used. Although it is not expected to be universally true, it proved necessary in this system to use double shielded pairs between the building entrance cable termination and the computing center data set location to avoid inductive pickup of interfering signals.

Need for System—Initial Concepts of Design and Usage

In late 1959, when we were doing the initial planning for computing equipment at Holmdel,

we had been told that within about six months of initial occupancy, the Holmdel buildings would house a total of some twenty-five hundred people. Since these people were to be transferred primarily from our New Jersey Laboratories at Murray Hill and Whippany, many of them at the time of relocation would be in the middle of projects requiring use of a computer. In order to provide a capability roughly equal to that at Murray Hill, we considered the following alternatives:

Installation of a 7090 at Holmdel

This we realized to be an ultimate requirement, and a 7090 is presently scheduled for installation in the latter part of the year. However, the total anticipated load during the first six months or so of occupancy did not justify earlier installation. A computing facility of this size costs of the order of $100 thousand per month, and it is almost out of the question from an economic point of view to install one without a nearly full prime shift load.

Install a Smaller Machine in the Interim Period

This is an alternative which we dismissed at once. Our programming costs are of the same order of magnitude as the computer operating costs, and we just could not afford the reprogramming effort entailed.

Use a Station-Wagon Data Link

This was the most attractive alternative from a point of view of economy, and we have even gone so far as to provide backup for the automatic data link by a truck which makes several regularly scheduled round trips per day between Murray Hill and Holmdel in the event of data link failure. However, the truck scheduling problems here were such that the service, in terms of turn-around time on a typical job, would not be good enough for a continuing operation.

A Voice-Bandwidth Data Link

There were several commercially available automatic data transmission facilities which operated at speeds up to 2400 bits per second on voice bandwidth lines. However, these were all too slow to give adequate service under heavy load conditions. Since we were sure of several hours of 7090 usage per day, the delays in such a facility would result in about the same grade of service as the station-wagon.

A Tape-Speed Data Link

At the time we were making our plans for Holmdel, a Microwave tape-to-tape data link was in operation on the West Coast. Such a system would provide higher speeds and some operational benefits. Since, however, we were really interested in coverage only over a fairly brief interval, installation of a microwave transmission system made this alternative unattractive, from a cost viewpoint alone.

The TELPAK A Data Link

This, of course, was our final choice. Its main attractions were that it operates over transmission facilities which are available in fairly large quantity in the Bell System toll plant throughout the country, its cost reasonable, and its operating speed such as to increase the total job processing time by only about twenty-five percent. Furthermore, being in the business of communication, we felt this to be an extremely worthwhile experiment in the field of data transmission.

In forming our initial concepts of the TELPAK A data link, we knew that at the time of its installation the Murray Hill Computation Center would include a 7090 supported by three 1401 computers as peripheral equipment. At Holmdel we needed at least one 1401 to read and write the tapes to be transmitted over the data link and to read cards, print, and punch. Since the signalling rate of the data link was limited to the order of 40 kilobits per second, we required a buffer at each end to compensate for the difference between tape and transmission speeds. It was quite natural then to examine, together with people from IBM, the possibility of using the 1401 computer at each location for both buffer storage and control.

Since we have stored program computers at each end of the data link, there is a great deal of flexibility in matters of tape format, block size, coding scheme and the like. However, virtually all of our operating experience has been with tapes written in the format peculiar to our 7090 monitor program,

BE-SYS-4. In reference to the particular options possible with IBM tape transports, this format involves binary (odd parity), high density (556 characters per inch) records of length which is variable but do not exceed 1000 characters. As far as the data link is concerned, our only limitation on block size is the buffer space we have reserved in the 1401 memory. We have 8000 characters of core storage in all of our 1401's and the storage required for the transmission program itself is about 1100 characters, so that it is possible for us to work with a much larger block size.

It is typical of a buffered transmission scheme that the average effective data rate is low for both very short and very long records. With very short records a large fraction of time is spent in "overhead" - starting and stopping the tapes and transmitting acknowledgements. Conversely for very long records, there is a higher probability of a parity error in transmission of a record and high penalty in time for retransmission. In between these two extremes there is a fairly broad optimum. In terms of our particular experience a block of 1000 characters is the optimum length for a fairly high transmission noise level, say, one error in fifty thousand characters. On the other hand, even in a noise-free system it gives an average transmission rate of about 90% of the maximum possible. In this sense, then, the block size associated with our normal tape format is quite satisfactory. Secondly, the uniformity of the physical appearance of all information on tape has simplified the 1401 program normally used for operation of the data link to the point where we have made it a part of our standard 1401 program used for card-to-tape and tape-to-card/print operations. This in turn allows us to switch the 1401 from local operation to transmission without loading a separate program.

Operation of System

Let us now look at our method of operation of the data link in more detail. First, at Holmdel, a "batch" of programs is loaded into the 1401 using our standard card-to-tape program, thereby producing a 7090 input tape. This is rewound, and the 1401 program is altered (by sense switches) to transmit this tape to the receiving 1401 at Murray Hill. The tape is then sent to Murray Hill, and at the end of transmission the duplicate tape is rewound and the tape transport is switched electrically from the 1401 to the 7090. When the computer becomes available, the monitor program on the 7090 reads the tape, executing the various programs as they appear and generating results, again batched on a single output tape. At the completion of all jobs the output tape is rewound, switched electrically to the 1401, and transmitted to Holmdel. Finally the tape received at Holmdel is rewound and processed by our standard 1401 program to produce printed output and punched cards. Although operation over the data link in this manner involves the extra tape spinning (twice forward, two rewinds) required for transmission, the entire process can be carried out without mounting or dismounting a tape. In practice, we do move the data link output tape from its tape transport at Holmdel to one on an adjacent 1401, to allow printing and transmission to proceed simultaneously.

One very nice feature of the buffered transmission scheme is that we have complete freedom in the choice of 1401 input/output equipment to be used. At the time of writing we are using 729 Model II tape transports at Holmdel and 729 Model IV transports at Murray Hill; these differ in their operating speeds (75 and 112 inches per second, respectively). It is quite possible, for example, to go directly from cards at one end of the link to tape at the other, although we do not normally do so because of the added time this ties up both 1401's. Indeed our standard program in the receiving 1401 scans the data for certain 7090 monitor control cards and prints these immediately on the 1403 printer while transcribing them as well on tape. This then provides the computer operators with a summary of the jobs to be processed as well as any unusual instructions as to how they should be treated.

In a typical day's use, we transmit over 20 input tapes from Holmdel to Murray Hill, on a twice-an-hour schedule between 9:15 and 4:45, with transmissions in the evening shifts as required by the load. At the time of writing our daily load from Holmdel is about five hours of 7090 time, during which time we process typically 130 separate jobs. Roughly one hundred of these jobs are processed during the prime shift (9 a.m. to 5 p.m.); during this same eight hour period we typically process 150 jobs which originate at Murray Hill. The turn-around time (time from

submission of a program to the Holmdel Computation Center to delivery of printed output) is generally between two and four hours on jobs which require five minutes or less on the 7090. This is limited primarily by the printing capacity of the Holmdel 1401's, since the groups which were transferred to Holmdel are working on jobs which generate a considerable amount of output. We have handled some high priority runs for one part of the Telstar project giving less than half-hour turn-around at Holmdel without disrupting our flow of work through the 7090 (aside, of course, from the 7090 time used for actual execution of the program).

Operating Reliability and Results

As far as reliability is concerned, the link has gone down, in the period February 15 through July 15, a total of about eight times—once for the data set, once for the IBM translator and a half dozen times for the transmission facilities. These facility failures were isolated to one cable section and the problem was eliminated by changing to different pairs in the same cable. When the data link is working correctly, we have experienced an average retransmission rate due to noise of one error per three thousand records (see Figure 2).

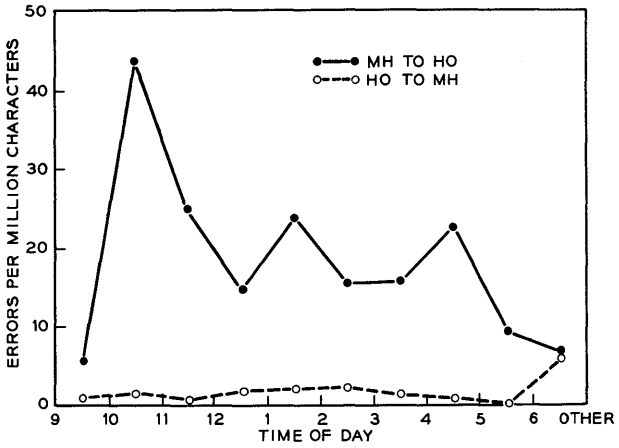


Figure 2. Error Rate Vs. Time of Day.

Preliminary Conclusions—Areas of Usefulness, Qualities

Our primary use of the Data Link was to provide rapid yet economical access to a large computer from a remote location. The desirability of the data link for this use depends on a number of factors:

The Load at the Remote Location

Since this use of the data link requires a 1401 at the remote location, not only to transmit data, but also to do normal card-to-tape and tape-to-print/punch operation, it is expensive. A remote facility, including a 1401, a tape unit, auxiliary keypunches and associated equipment, furniture, staff and space, would cost of the order of $10 thousand per month. Reasonable economy dictates that this cost be spread over a load of at least thirty or forty hours of 7090 usage. On the other hand, a monthly load of over 150 hours would justify the installation of a separate 7090. Therefore, this use of the data link depends strongly on bounding the load within a fairly critical range.

Anticipated Growth of the Load

Clearly, one of the competitors of remote operation of a large computer over a data link is the installation of an on-premises machine which is smaller and less expensive. The desirability of this course of action depends on what future loads are anticipated. That is, if we expect the load to be fairly constant with time the separate smaller computer is probably cheaper and more attractive. On the other hand, if the load is expected to grow to the point where a large machine will be justified within one or two years, then the saving in reprogramming and re-training of programmers might well repay the added dollar cost of the data link many times over. This certainly has been true in our own case. It might be added here that competition from small machines is purely on an economic basis. As soon as the industry can develop really inexpensive peripheral printers, readers, and punches, the data link will be much more attractive.

The Need for Good Service

Certainly the least expensive remote use of a computer, with today's technology, involves the use of cars, trucks, telephones and the U. S. Mail. These, however, give a grade of service whereby it takes a day or more to return results on any given run. In

a situation where the computer is being used primarily for production runs on a predictable schedule, this grade of service is often more than satisfactory. However, when a large part of the load consists of checkout and running of programs which are required in scientific and engineering projects, such economies in operating costs are far outweighed by ineffective use of technical personnel and by project delays.

Upon installation of a 7090 at Holmdel, we plan to continue use of the data link for purposes of load balancing as well as for protection of both computation centers in the event of temporary unavailability of one of the 7090's. Although this is a far less compelling motivation than that of remote computer operation, the price comes down at the same time, for now we have 1401 computers, operating personnel and everything else at both locations anyway.

As with anything in this world except good bourbon, the data link does have some recognized deficiencies. In the first place, because the operation is essentially tape-to-tape, we find ourselves winding and rewinding tapes twice more than we ordinarily do when operating a 7090 locally. These added operations, although quite simple, are enough to encourage us to do more batching of jobs than otherwise. This, in turn, increases the turn-around time and requires more attention of the operators.

As far as speed is concerned, it would be nice to have it a good deal faster, for we find ourselves tying up the two 1401's for roughly thirty hours a month just transmitting tapes. On the other hand, this is not a really serious deficiency, since for each minute we spend transmitting data we generally spend five more in printing it.

The major limitation on the data link as we have used it is its cost, and this is really not a reflection on the cost of data transmission, but rather on the cost of the supporting equipment and personnel at the remote location. If we could bring the total price of a remote location down to one or two thousand dollars per month and yet retain the input-output speed of the 1401 card reader and printer, then we would be able to justify a remote operation with a monthly load of well under ten hours of 7090 time. It is felt to be but a matter of time until the proper equipment is developed, but it certainly is not possible today.

STANDARDIZATION IN COMPUTERS AND INFORMATION PROCESSING

C. A. Phillips and R. E. Utman
Business Equipment Manufacturers Association
New York, New York

The data processing standardization program is a comparatively new effort since it had its genesis in an action by the International Organization for Standards (ISO) in late 1959. On a recommendation from Sweden, ISO decided there was a need for a standards program in connection with computers and information processing. It seems rather remarkable that the need for a standards program was recognized so early in the state of an art whose principal tool, the electronic computer, is only fifteen years old this year. Before getting into the specifics of this program, let us first consider very briefly, standardization as a process, and its affect upon our lives. The Encyclopaedia Britannica describes standardization as a continuing process to establish measurable or recognizable degrees of uniformity, accuracy or excellence, or an accepted state in that process. It goes on to point out that man's accomplishments in this direction pale into insignificance when compared with standards in nature, without which we would be unable to recognize and classify within a species, the many kinds of plants, fishes, birds or animals. Without such standardization in the human body, physicians would not know whether an individual possessed certain organs, where to look for them, or how to diagnose or treat disease. To further quote the Encyclopaedia "without nature's standards there could be no organized society, no education and no physicians; each depends upon underlying comparable similarities." Although we are inclined to think of man-made standards as relating principally to such things as weights and measures, money, energy, power or other material commodities, you will also find standards in social customs, in codes, procedures, specifications and time—to name a few. Standardization is important to geography, photography, chemistry, pharmacy, safety, education, games, sports, music, ethics and religion. The profession of accounting, for example, is largely dependent upon standards—which are generally referred to as "accepted practice."

In fact, it is "accepted practice" that usually generates standards—many of which may be unwritten, simple and crude, while at the other end we have standards that are specified in great detail, nationally accepted and used, and, in many cases, subject to legal definition.

There is no question that industrial activity thrives on standardization. It has been argued with strong support, that industrial standardization is the dynamic force that, in a sense, created our modern Western economy. There is no question that industrial standardization is the cornerstone of our mass-production methods, which, in turn, is such a vital part of our American economy. All of the industrially advanced countries of the world have their own national standards, and in many of them, standardization is whatever the government decrees—at least that is the case in Soviet Russia and its

satellites. Russia has over 8,500 standards in effect; Germany, over 11,000; France about 4,500. The United Kingdom has approximately 4,000 British Standards available for use. The United States has approximately 2,000 American Standards approved by our voluntary national standards body, the American Standards Association.

The multiplicity of standards making groups and the frequent duplication of effort by several groups having a kindred problem, led to the founding of the American Standards Association (ASA) in 1918. During World War I, the need for eliminating conflicting standards and duplication of work became urgent. Several engineering societies, together with the War, Navy and Commerce Departments, established the American Engineering Standards Committee which was reorganized in 1928 and renamed the American Standards Association.

Today, the ASA federation consists of 126 national organizations, supported by approximately 2,200 companies. Over the years, ASA has evolved a set of procedures that apply checks and balances to assure that a national concensus supports every standard approved as an American Standard by ASA. By the terms of its constitution, ASA is not permitted to develop standards, but instead, acts as a catalyst by aiding the different elements of the economy to obtain a desired standards action through the established procedures.

In 1946 some one hundred top leaders in business and industry entered into a formal agreement with the Secretary of Commerce to broaden the scope and activities of the ASA. Along with the American Society for Testing Materials, ASA is now reviewing federal specifications to bring them into line with the best industry practice. Today the federal government is following a policy of using industry standards rather than writing its own and ASA has become a focal point of cooperation in standards work between government and industry.

The Department of Defense, by specific Directive, has authorized its personnel to participate in ASA activities as voting members and, at the present, no fewer than 25 federal agencies and in excess of 600 government representatives are participating in the work of ASA committees. The National Bureau of Standards accounts for many of these committee posts.

As a means of avoiding or eliminating differences among National Standards, which sometimes may be even a greater trade barrier than import quotas or high tariffs, 44 nations have joined together in a world-wide non-government federation of national standards bodies known as the International Organization for Standards, or ISO. The objective is to coordinate the national standards in various fields by means of ISO Recommendations, which are then available for voluntary adoption by all countries. In the electrical field, international standardization is conducted through the International Electrotechnical Commission (IEC) which is an independent division of ISO, made up of national committees in 34 countries. The American Standards Association is the USA member of ISO and the U. S. National Committee of the IEC is an arm of ASA.

With this very general background of standards practices and organization, let us next look at the standards program in the field of computers and information processing under three general headings:

1st - the relationship of this program to the international and national standards organizations and the manner in which the effort has been organized and is being directed;

2nd - the membership of the various groups participating in the program;

3rd - the scope of the overall program and its various subdivisions, along with the approach in each case and a brief report on progress.

Coming out of the 1959 meeting of ISO, previously referred to, was the assignment by ISO to the United States of overall responsibility for the programs conduct. A chart reflecting the overall organizational structure would show at the top of two parent organizations: ISO, from the international level, and ASA from the national level. Following established procedures, ASA assigned the program to a sponsor, which is usually a trade association with a direct interest in the subject and a willingness to undertake the effort. In this case, the Office Equipment Manufacturers Institute, which later became the Business Equipment Manufacturers Association or BEMA, was the logical organization to be given the responsibility as sponsoring activity. Under ASA procedures, the sponsor organizes the project, subdividing it as necessary, and finances

the full time staff and other direct costs incident to the program.

Each major project under a sponsor is referred to as a Sectional Committee. ASA has an identification system of letters and numbers for these Sectional Committees, and under this system the data processing standards project became known as the X-3 Sectional Committee.

It should be mentioned that a concurrent project that is concerned with standards in the office machines area was also assigned to BEMA as sponsor and is identified as the X-4 Sectional Committee. This breakdown into the X-3 and X-4 Sectional Committees, coincides with the organization of BEMA into semi-autonomous groups known as the Data Processing Group, with responsibility for X-3, and the Office Machines Group, with responsibility for the X-4 Project.

Closely related to the X-3 Sectional Committee is another Sectional Committee identified as X-6, which was established by ASA under the sponsorship of the Electronic Industries Association (EIA) for consideration of those aspects of the standards program in data processing which are purely electrical as distinguished from the logical or other physical characteristics which is the responsibility of X-3.

It would be well at this point to consider further the role of ASA in relation to the Sectional Committees. As the various Sectional Committees develop recommendations through various sub-committees, they go through an approval process at the Sectional Committee level and are then submitted to ASA. The ASA will review the proposed standards and the supporting data and reach a judgement as to whether or not a concensus exists for such a standard. It may be refused on a single negative vote or approved with several dissents.

The X-3 Sectional Committee is made up of three major groups with approximately the same number of members in each group. These groups are known as the Users Group, the General Interest Group and the Manufacturers Group and are made up for the most part, by representatives of trade associations, professional or technical societies or other bodies having a direct interest in the subject. The members of the Manufacturers Group are selected from the BEMA membership by the Engineering Committee of the Data Processing Group/BEMA, which is charged with direct responsibility (within BEMA) for general direction of the standards program. At the present time, the X-3 Sectional Committee is chaired by a staff member of the Data Processing Group of BEMA.

The General Interest Group of the X-3 Sectional Committee is made up, for the most part, of organizations or societies related by professional background or interest. They include the Association for Computing Machinery (ACM), the American Management Association (AMA), the Electronic Industries Association (EIA), the Engineers Joint Council (EJC), the Institute of Radio Engineers (IRE), the Association of Management Engineers (ACME), the National Machine Accountants Association (NMAA) and the Telephone Group. The Department of Defense is also represented in the General Interest Group.

The Users Group is made up of associations that have a common interest as to type of business. They include the Air Transport Association (ATA), the American Bankers Association (ABA), the American Petroleum Institute (API), the Insurance Accounting and Statistical Association, the Joint Users Group (JUG), the Life Office Management Association (LOMA), the National Retail Merchants Association (NRMA) with the American Gas Association and the Edison Electric Institute holding a joint membership. The General Services Administration represent the Federal Government in the Users Group.

Representing the Manufacturers Group are ten companies, some manufacturing complete data processing systems, while others manufacture devices used in conjunction with data processing systems. The companies representing BEMA are: Burroughs Corporation, International Business Machine Corporation, Minneapolis-Honeywell EDPD, Monroe Calculating Machine Company, National Cash Register Company, Pitney-Bowes Inc., Radio Corporation of America, Remington Rand Divisoin of Sperry Rand, Royal McBee Corporation, and Standard Register Company.

ASA Procedures require that the terms of reference under which a Sectional Committee operates shall be clearly set forth in a statement of scope, which might be called a "charter." The language used to describe the scope of the X-3 Sectional Committee is as follows:

"Standardization of the terminology, program description, programming

languages, communication characteristics, and physical (non-electrical) characteristics of computers and data processing devices, equipments and systems."

You will note the specific exclusion of electrical characteristics which, as previously mentioned, has been assigned to the X-6 Committee under sponsorship of EIA.

The very broad scope of the X-3 Sectional Committee has been subdivided into seven parts or subcommittees, all having one thing in common—they are dealing with problems of communication. The first subcommittee, X-3.1 is concerned with Optical Character Recognition, X-3.2 is concerned with Coded Character Sets and Data Format, and X-3.7 is concerned with Magnetic Ink Character Recognition. You will note that these three deal primarily with input and output problems which might be described as communications between men and machines. Another concerned with communications between men and machines is X-3.4 on Common Problem Oriented Programming Language. The X-3.3 subcommittee is concerned with data transmission problems which might be described as communications between machines. X-3.5 concerned with Terminology and Glossary, and X-3.6 concerned with Problem Definition and Analysis, represent problems of communication between men about machines. These seven subcommittees are chaired by representatives of the companies that comprise the Manufacturers Group together with one chairman from the Navy Department and one from the General Interest Group representing the Association for Computing Machinery.

In the numeric order let us next examine the scope, the approach and the progress of each of the subcommittees of X-3.

The scope of the X-3.1 subcommittee has been defined as the development of humanly legible character sets for use as input/output for data processing systems and the interchange of information between data processing and associated equipment. Considerable work has been done over the past few years in this field by the Retail Merchants Association and others, which has been followed up and expanded upon by X-3.1. Initially, the work of this group has been concentrated in the numeric area—which is so badly needed, and at the same time is probably easier to develop. As you probably know, there are several optical readers on the market today, all using their own unique character font, and a standard font, both for numbers and letters, could do much to advance the state of the art. This group has a two-pronged problem—if the standards are set low in quality as to format, size, density or other printing characteristics, the optical reader will be comparatively expensive to produce. If, on the other hand, the standards are set high, the reader may be cheaper, but the printing devices and imaging media may be higher in cost. Achieving a proper balance, is the big problem confronting this subcommittee.

The X-3.1 subcommittee approached their problem by dividing the work between three task groups. The first group will determine the proposed measurements, specifications and terminology for the font; the second group is concerned primarily with printing capabilities and the parameters of printing devices; while the third group will study retail requirements and priorities and will evaluate other requirements with similar or different problems. Interest and participation in the X-3.1 effort has been very high with about 50 people working actively. They have been holding meetings every 4 to 6 weeks and are confident of measurable progress within the current year. Hopefully, they will have a numeric font ready for consideration soon.

The scope of the X-3.2 subcommittee provides that they will develop standards for coded character sets and data record formats to facilitate communications both within and between data processing systems. Here the problem is primarily machine to machine communication, rather than man to machine, as with X-3.1 Today there are over 65 different machine codes used world-wide, with over 50 different ones used in the United States. Frequently the difference between these codes may appear to be minor, although such differences may have a major impact. For example, the order in which the string of characters places the numbers and letters. Some codes put the alphabetic characters first, followed by the numbers, then followed by the machine-function codes such as: carriage return, back-space, upper-case, lower-case, etc. Other codes change this order or reverse it. There is also the problem of a standard representation of the characters in the bit structure of the magnetic tape, or the punched holes in cards or tape.

Obviously, if standards proposed by this country are to be accepted internationally, they must make provision for alphabets other than English, and must not differ too greatly from codes used by European or other countries.

The approach adopted by the X-3.2 subcommittee is in three phases—the first task group is to determine the alphanumeric characters, symbols and control characters desired; the second group will develop the detailed code representation for such characters, symbols and functions; and the third group will develop a standard format for utilizing the standard coding. The first two of these groups have met their targets—with what accuracy is yet to be determined—and the third group is now working actively with a joint input/output group.

The X-3.2 subcommittee has completed the development of a recommended American Standard code for information interchange and has submitted it to the X-3 Committee for processing. In turn, X-3 has submitted the recommendation for balloting by the X-3 members. In the meantime, through conferences with groups in Europe, the X-3.2 subcommittee is considering revisions or modifications that would make the proposed standard more acceptable as an international code. The pros and cons are being actively discussed and we may or may not have an American Standard within this area within the current calander year.

The X-3.3 subcommittee on Data Transmission is concerned with the determination and definition of parameters governing the operational action and reaction between communications systems and the digital generating and receiving systems utilized in data processing. From an overall business standpoint, this is a relatively new problem that has been under active development for only four to six years although the military have been working on it for much longer. Previously, translations from data processing codes to codes suitable for transmission have been done manually. Today there are standards in the field of voice and telegraph transmission, but very little has been done beyond this. Radio and television raised questions on line or channel quality and width and the advent of the computer raised questions of relative costs. Many users of data processing equipment believe that the economical use of data processing equipment requires the centralization of the data processing activity. This immediately imposes requirements for economical data transmission and the need for an effective interface with data communication.

Although X-3.3 was somewhat slow in getting under way, they have now organized their effort under five groups. The first group will handle liaison with other interested groups; the second group will develop a glossary of special terms relating to this subject; the third group will document error detection and control techniques; the fourth group will try to specify the system aspects of data processing terminal equipment to be connected with communications equipment, and the fifth group will do research into systems performance characteristics.

In spite of a slow start, X-3.3 has made good progress, largely because of excellent cooperation with the related subcommittee under the X-6 Sectional Committee sponsored by EIA. For several years EIA has had a group known as EIA TR/27.6, working on problems in this area. Through this cooperative approach X-3.3 now has had its proposed standard on signalling speeds for data transmission equipment approved by the X-3 Sectional Committee and the ASA as an American Standard. This is the first American Standard to result from the X-3 program.

The scope of X-3.4 has been described as follows: "Standardization and specification of common programming languages of broad utility, with provision for revision, expansion and strengthening, and for definition and approval of test problems." This area is one of the most difficult and probably of the greatest concern to the data processing community because of the increasing costs of "soft-ware." It will be noted that the scope encompasses both the business-type languages and the scientific-engineering type languages. In fact, it probably also includes the so-called "command and control" languages of the military under the title of "problem-oriented." There have been many users groups active in the development of programming languages and it is expected that X-3.4 will utilize much of this work that has gone before. At the present time the subcommittee is considering COBOL, ALGOL and FORTRAN, the three most widely used programming languages.

The X-3.4 subcommittee has been organized into six working groups, the titles of

which will suggest the areas of work assigned: Working Group 1 is concerned with language theory and structure, WG 2 with ALGOL, and Specification languages, WG 3 with FORTRAN, WG 4 with COBOL and language processors, WG 5 with international problems, and WG 6 with programming language terminology. The complexity of this area makes it probable that there will be some overlap within the subcommittee and with other groups and that the progress may be somewhat slower in spite of the most dedicated and sincere effort of the participants.

The X-3.5 subcommittee is concerned with problems of man-to-man communications under the following two-part scope: (1) to recommend a general glossary of information processing terms, and (2) to coordinate and advise the subcommittees of ASA X-3 in the establishment of definitions required for their proposed standards. This group has recognized an overlap with work done in this field by others, for example, a joint glossary has been compiled by the ACM—the AIEE and IRE consisting of over 2800 words or terms. X-3.5 expects to use this as a base and to coordinate their efforts with those of other groups, including one in the Federal Government under the aegis of the Interagency Data Processing Committee, and a similar joint effort in Great Britain.

X-3.5 is a comparatively small group with strong participation from the User and General Interest groups on X-3. Much work has already been done in this field and it is now largely a matter of collating, refining and editing and of putting the product in prescribed form for submission as a proposed American Standard.

New users, or prospects for electronic data processing systems are frequently surprised to find that there is still no standard accepted ways of defining data processing applications. This is the field in which X-3.6 has defined their scope; as: (1) the development of standardized survey techniques, (2) the standardization of flow charting symbols, and (3) the development of narative - symbolic - quantitative methods of presenting results to top management, data processing customers and data processing operators. Work along this line has been done by the separate companies and a good bit by government agencies. The Federal government, through the Interagency Data Processing Committee has developed guide lines and criteria for feasibility studies, application studies, and flow charting techniques and symbolization. There is good reason to believe that these efforts will have a strong influence on the work of the X-3.6 subcommittee. It is also hoped that active interest and participation by educational institutions can be stimulated.

X-3.6 has subdivided the work into four task group assignments. The first is concerned with methodology, the second with input/output data and file description, the third with data transformation, and the fourth with nomenclature and flow charting. So far, the greatest tangible results have been in the work of Group 4 which is hopeful to having a proposed standard for flow charting symbols ready soon for consideration. In a recent indication of the dynamic nature of industrial standardization, ASA abolished the obsolete X-2 Sectional Committee on office standards and assigned its project on charting paperwork procedures to X-3 and X-3.6.

The latest of the subcommittees is X-3.7 which is concerned with magnetic ink character recognition. Work in this field is well along, and this is recognized in the scope which is described as follows: (1) development of standards for magnetic ink character recognition (MICR) for present and future use, and (2) resolution of problems arising in industry and the market place involving manufacturers and printers. Under the aegis of the American Bankers Association and interested manufacturers, the magnetic ink character recognition font, known as E 13 B, has been adopted by the American banking industry. Therefore, the X-3.7 subcommittee is following an approach which might be thought of as a maintenance program rather than a development program. They propose to (1) determine the best common method for handling miscoded documents, (2) resolve a standard location for check serial numbers, and (3) eliminate extraneous magnetic printing on the clear band of checks.

The X-3.7 subcommittee is hopeful of getting the de facto standard MICR font processed and accepted as an American Standard and thereafter submitted for consideration as an International Standard. It has been approved by X-3 and is now submitted for processing through ASA.

By design this structure of the X-3 Sectional Committee closely resembles the organization of ISO Technical Committee 97

on Computers and Information Processing. There are six subcommittees and one working Group of TC97 at the international level with titles and scopes quite similar to those of the X-3 subcommittees: SC1 for multi-lingual glossaries; SC2 for coded character sets; SC3 on both optical and magnetic character recognition; SC4 on input/output media standards; SC5 on programming languages; SC6 on data transmission, and WG1 on Problem Definition and Analysis including flowcharts.

It should be emphasized that although organizationally similar, the character of national and international standardization differs considerably. National standardization activity can involve development of standards where need exists and accepted practice or appropriate developmental facility does not, as in the aforementioned character code case. International standardization, on the other hand, tends to be legislative in nature, with the work of TC97 and its groups devoted to the processing of national proposals that represent local standards or practice. Little if any standards development is foreseen or considered in the international activities of TC97. In addition to considering national standards proposed for ISO consideration, TC97 also accepts documented proposals from official liaison organizations of an international nature such as the European Computer Manufacturers Association and the International Federation for Information Processing.

Proposed international standards involving electrical characteristics are processed by the IEC Technical Committee 53 - Computers and Information Processing, and its four subcommittees: A for Input/output Equipments; B for Data Communications; C for Analog Equipments in Digital Systems; and D for Input/Output Media.

Counterpart interests are included in the scope of the ASA X-6 Sectional Committee.

Where logical, physical and electrical factors in a standardization proposal cannot be isolated, as in such input/output media as magnetic tape, TC97, TC95 on Office Machines, and TC53 have made provision internationally for joint WG D and SC 53D work. Nationally, X-3 and X-6 have joined forces in three joint task groups for consideration in input/output media standards for magnetic tape, perforated tape, and punched cards. Other cooperative efforts are provided for as need arises, such as in programming languages and character sets, and between all special technical areas and the glossary activities.

Two final comments on the basic interests and intent of the national and international standardization efforts. First, there is no desire to develop or establish standards for the sake of standardizing. The only justifiable reason for standards in information processing is need, as expressed by the users and manufacturers of computers and devices, and those affected by such equipment. Second, in order to assure that such need is properly expounded when it exists, and that resultant standards represent acceptable solutions to such needs, the industry and users and general interests must participate fully and with qualified, active representation. Standards must accurately represent the predominant practices or wishes of the entire information processing community.

Those of you who are active in the data processing community are probably familiar with the magazine DATAMATION and may have read the feature article on the ASA X-3 Sectional Committee in the February 1962 issue. Although the article is rather critical of the lack of progress that has been made through the end of 1961, it is generally factual and, with minor exceptions, gives a good picture of this first year of the Standards Program. In spite of its rather critical tenor, the the article concludes with this statement:

> "A more realistic point of view is that standards activities are, by their very nature, methodical, plodding and, subsequently, quite permanent in their effect. It should be clearly understood that the biases, politics and frictions which come to play and may seem to impede the effort are, in fact, expressions of legitimate interests which comprise one of the most important aspects of the deliberations involved in setting and maintaining a standard."

HIGH-SPEED FERRITE MEMORIES

H. Amemiya, H. P. Lemaire, R. L. Pryor, T. R. Mayhew*
Radio Corporation of America
Camden 8, N. J.

INTRODUCTION

For several years ferrite cores [1,2] have constituted the mainstay of computer storage memories. The typical computer today [3] has a transistor-driven core memory which operates in a coincident-current mode [4] with a 5-to-10-μsec cycle time. Although a coincident-current storage unit with a cycle time approaching 2 μsec has been built, higher speeds have been attained by exploiting so-called partial-switching modes of operation. Word-address memory systems [5] with cycle times less than 1 μsec and as low as 0.7 μsec have been reported [6,7]. To attain these speeds with conventional 50/30 cores (that is, cores of 0.050-inch outer diameter, 0.030-inch inner diameter) drive requirements are necessarily high (approximately 1 ampere-turn), and particular attention must be given to the physical arrangements of conductors, sense windings, and storage elements.

Actually, short cycle times have been realized by the development of fast-switching storage elements which operate in impulse switching modes using high drives, and by minimizing such factors as propagation time, field transients, and mutual-coupling effects to reduce the duration of the unproductive phases of the memory cycle. The reduction of these phases has assumed increasing importance as the operating speeds of memories have increased. Array geometry, timing operation, and the relative positioning of components have become prime factors in the determination of the minimum access time of a memory.

The use of permalloy thin films as high-speed storage elements has recently received a great deal of attention [8,9,10]. A small memory with a cycle time of less than 0.5 μsec has been operated [11], and cycle times shorter than 1 μsec appear generally feasible in memories of larger capacity [12]. In addition to their high-speed potentialities, thin-film memories can be fabricated in large sheet arrays at relatively low cost. (This fabrication technique has not been fully developed at the present time.) Disadvantages of thin-film memories include high drive-current requirements (0.5 to 1 ampere) and low bit outputs (less than 5 mv) [9]. Large arrays operating at high speeds may, in fact, be impracticable, because the discrimination between the low output signal and the stack noise becomes increasingly difficult as memory capacity is increased.

Ferrite pieces utilizing closed magnetic paths of miniature dimensions offer obvious advantages for high-speed memories [13]. Outputs and switching speeds can be kept high, while at the same time drive currents kept low. This paper presents the results of a program aimed at developing ferrites capable of operation in memories with cycle times less than 0.5 μsec and at bit costs competitive with those of slower, more conventional arrays. These goals have necessitated the solution of two associated problems: first,

*RCA, Needham, Mass.

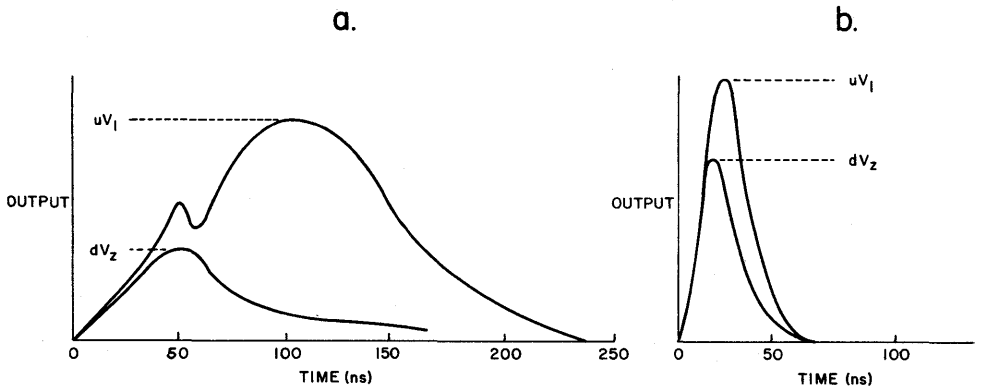
the development of low-drive, fast-switching memory cells; and second, the development of methods for the assembly of these cells into arrays economically.

Memory Organization

High speeds have been achieved utilizing a word-address, two-core-per-bit memory organization. Linear selection (word-address) memory schemes are well established as a means of obtaining increased memory speeds since, in contrast to coincident current methods, readout currents of unlimited magnitude can be used. (Currents are then determined by transistor driver limitations rather than by core or memory organization.) Linear selection provides a second, important means of attaining high speeds by making it possible to use high amplitude, short duration write pulses. Narrow pulses such as these switch a minimum of flux by themselves although their amplitude is substantially greater than the normal switching threshold; when added to the exciter or digit pulses, however, they are capable of switching a significant amount of flux [14].

As memory cycles are reduced, a point is reached where two-core-per-bit operation becomes a necessity if practicable signal-to-noise ratios are to be maintained. This comes about for two reasons; first, as the write pulse is made increasingly narrow, a very small fraction of the core is being switched so that upon readout the difference between a 1 and a 0 is very small; and second; as the read pulse is made narrower and the rise time decreased the contribution of reversible magnetization changes becomes an increasingly significant fraction of the total output. In addition, the peaking time of the core rapidly approaches the time at which the reversible flux peak occurs so that the two peaks merge into one. Figure 1 illustrates qualitatively, the output waveforms for two cases, each involving partial switching modes differing essentially in the amount of flux switched and the switching times. Figure 1a illustrates the performance of a core with a switching time of 200 nsec, usable in a memory with approximately a 1 μsec cycle time. Figure 1b illustrates the case for higher speed situations. Here, less flux was written into the core, the read duration and rise times were decreased and the switching time reduced to 50 nsec. When the switching time is short, even with precise strobing, the difficulty of obtaining sufficient discrimination is evident from Figure 1b.

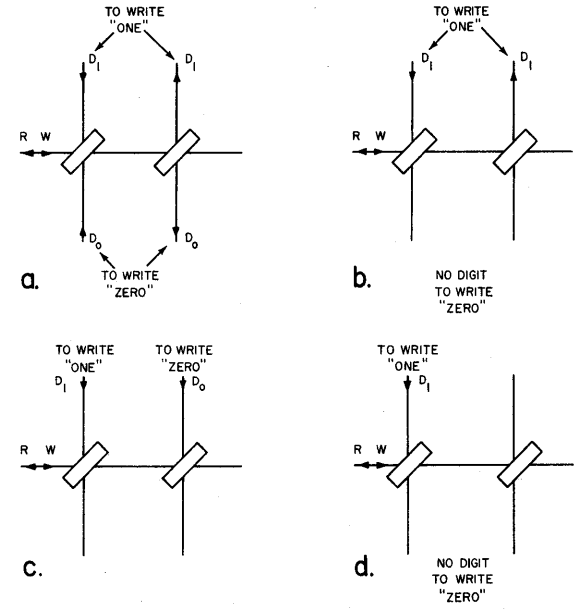
Two-core-per-bit operation provides a means of cancelling out the reversible flux contribution to the total output. Figure 2 illustrates four possible two-core-per-bit schemes which differ only in the way the digit pulses are applied to the bit. In Figure 2a, bidirectional digit pulses pass through both cores in the bit. In Figure 2b, a digit pulse passes through both cores to write a 1; there is no digit pulse when writing a 0. In both 2a and 2b, digit pulses "add" to the partial-write pulse in one core and "subtract" from the partial-write pulse in the other. In 2c and 2d,



a. Switching time, t_s - 200 ns b. t_s - 50 ns

Figure 1. Output Waveforms vs. Switching Times.

DIGIT DRIVERS



SENSE SIGNALS

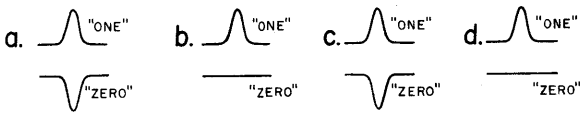


Figure 2. Four Digit Drive Techniques and Sense Signal Waveforms.

digit pulses have but one direction which is the same as that of the partial-write pulse. In cases 2a and 2c, the read outputs are bipolar, whereas in 2b and 2d the outputs are unipolar. Only two of these schemes (2c and 2d) were in fact used in this system.

Unidirectional bit drives were used exclusively when it became evident that bidirectional digiting led to an increase in digit-disturb sensitivity. This effect was particularly evident when each core in the bit was individually examined. Test results showed that a core which received a digit pulse opposite in direction to the partial-write pulse had a lower disturb threshold than a core for which write and digit pulses were in the same direction. This effect and similar phenomena has been reported elsewhere [6]. Henceforth, in this article, Figure 2c will be referred to as the bipolar sensing scheme. The situation drawn in Figure 2d, which differs from 2c in that only one core is digited (output of single polarity) will be termed an amplitude sensing scheme.

Bit Characteristics

Bit evaluation was performed using a two-wire system with common sense-digit wires and common read-write wires. A schematic illustration of the test setup is indicated in Figure 3. In bipolar operation, digit driver "one" is turned on to write a 1 and driver "zero" to write a 0. In the amplitude sensing mode, driver "one" is again turned on to write a 1; driver "zero" is not needed and is simply turned off.

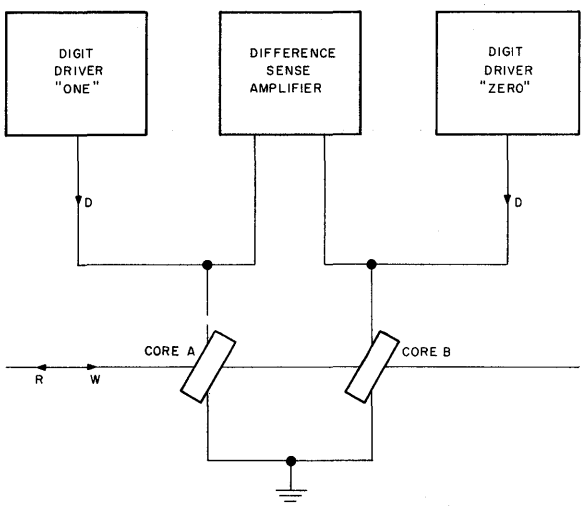
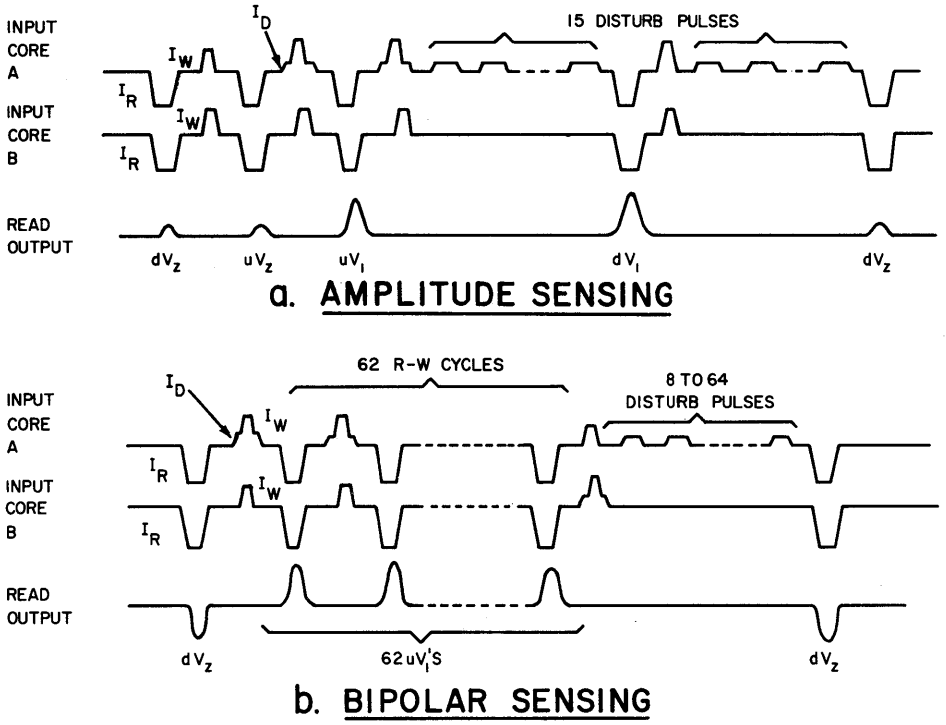


Figure 3. Test Set-Up (Bit Evaluation).

Amplitude Sensing: Cores which were tested with the amplitude sensing mode were subjected to the series of input pulses as shown in Figure 4a. This sequence was chosen to produce the worst signal-to-noise ratio; in this case, this ratio is the ratio of the amplitude of the undisturbed 1 to the amplitude of the disturbed 0 ($uV_1 : dV_Z$). The particular order of the sequence—undisturbed 0 voltage (uV_Z), undisturbed 1 voltage (uV_1), disturbed 1 voltage (dV_1), disturbed 0 voltage (dV_Z), was intended to bring out instability in the remanent state on readout, if instability existed. If readout were incomplete the dV_Z which follows a dV_1 would have its highest value, and a uV_1 following a uV_Z would have its lowest value.

The four read outputs were superimposed to obtain a simultaneous oscilloscope display of the type shown in Figure 5. The waveforms shown were obtained using cores of 0.050-inch outer diameter, 0.010-inch inner diameter, with the drive pulse characteristic shown



a. Amplitude Sensing b. Bipolar Sensing

Figure 4. Pulse Test Programs and Outputs.

TEMPERATURE = 25° C

CORE = 50 MIL O.D., 10 MIL I.D.

	AMPLITUDE ma	DURATION ns	RISE TIME ns
READ	325	100	25
WRITE	250	50	25
DIGIT	60	90	25

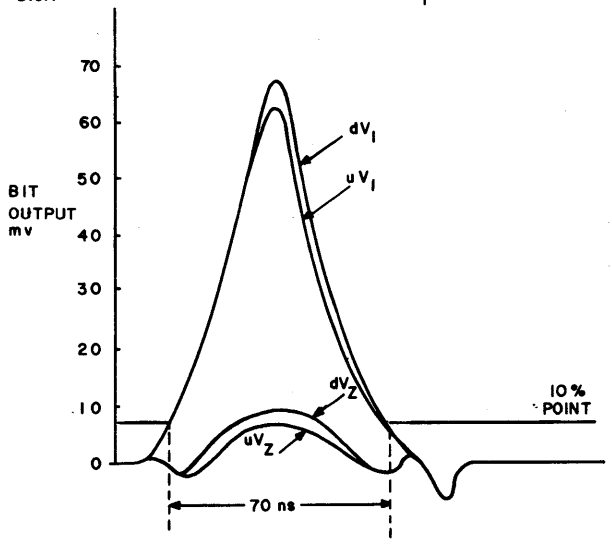


Figure 5. Output Waveforms (Amplitude Sensing).

in each figure. The switching time, taken at the 10-percent points, is about 70 nsec. The relatively small difference between disturbed and undisturbed signals is an indication that the digit-disturb pulse has minimum disturbing effect on the core.

Figure 6a shows the effects of variations in the digit and partial-write current amplitudes for fixed current-read conditions. The 1 and 0 outputs are actually undisturbed 1's (uV_1) and disturbed 0's (dV_Z), respectively, Figure 6b shows the signal-to-noise rotios which serve as a guide in the determination of optimum drive-pulse characteristics. For the particular durations and rise times used in this case, a range of workable digit and partial-wire current levels is available. In Figure 6, for example, it is apparent that a digit current in the 60- to 70-ma range and a partial-write current of 200 to 220 ma would give a 1 output of 40 to 65 mv at signal-to-noise ratios of 9 or 10 to 1. Switching times in this case are in the order of 80 nsec.

The drive pulse and output characteristics indicated in Figures 5 and 6 are typical of cores with a 10 mil inner diameter and 50 mil

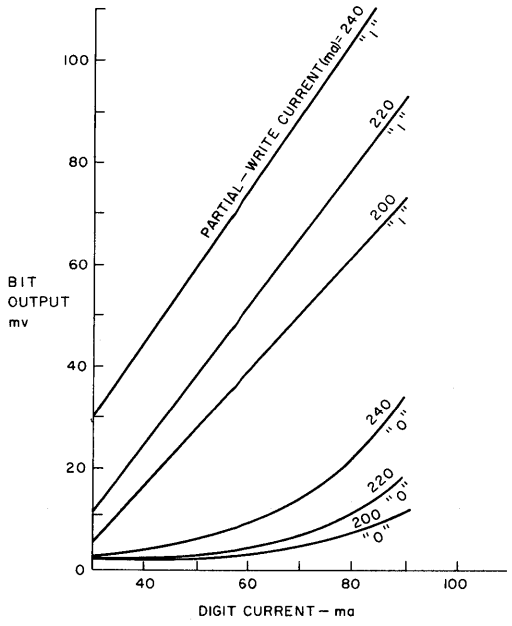


Figure 6a. Changes in Bit Output with Variations of Digit and Partial-Write Current Amplitudes.

CORE = 50 MIL O.D., 10 MIL I.D.

	AMPLITUDE ma	DURATION ns	RISE TIME ns
READ	350	100	40
WRITE	VARIABLE	60	40
DIGIT	VARIABLE	100	44

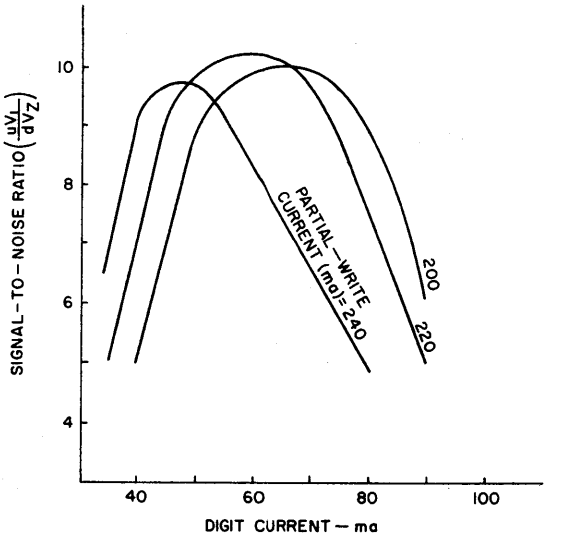


Figure 6b. Signal to Noise Ratio vs. Digit and Partial-Write Current Amplitudes.

outer diameter. Switching times and read-write cycle times can be shortened by increasing the read pulse amplitude and decreasing rise times and durations. Shortened rise times, however, bring about increased back voltages per bit during readout with little change in net signal output. This happens, of course, because as the rise time is decreased the reversible flux contribution to the total output is increased. Figure 7 illustrates the effect of decreasing the rise time of the read pulse while maintaining its amplitude fixed. As is evident from the diagram, the total back voltage per bit decreases from 250 mv to 150 mv when the rise time is increased from 13 to 40 nsec. The decrease in signal is in comparison very small—in fact there is no change in signal when the rise time is changed from 30 to 40 nsec. These facts are of obvious importance if long memory words come into consideration since one can bargain for lower back voltages on the word lines by increasing rise times and memory cycle times but with very little loss in bit output.

Bipolar Sensing: In the bipolar-sensing method (Figure 4b), core A is digited to write a 1, core B to write a 0. Thus, in the test program shown, the first series of read outputs are undisturbed 1's. The last readout

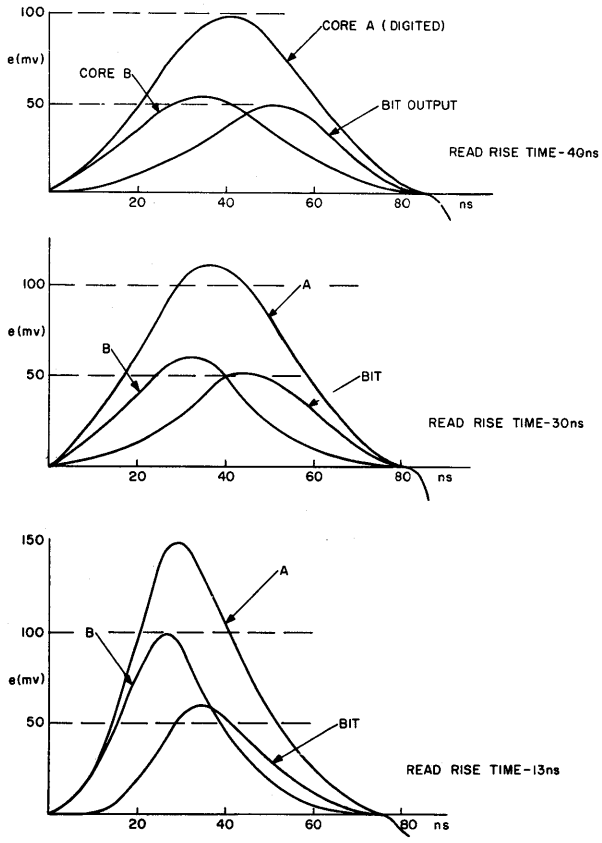


Figure 7. Effect of Read Pulse Rise Time on Core and Bit Output.

on the right is a disturbed 0 which is termed the "lowest 0." In this case, although core B is raised to the higher flux state by superimposing digit and partial-write pulses, the disturb pulses are applied to core A. (If the pulse patterns in the two cores were reversed, the initial series of readouts would be undisturbed 0's and the last readout a disturbed 1.)

Actually, the pulse sequence is a variation of that used for amplitude sensing and was chosen to give the worst-case conditions (i.e., by promoting the lowest 0 or the lowest 1). As before, the reasoning was that if the read-pulse amplitude or duration was insufficient to bring the core back to a stable remanent state, a 0 would have its lowest magnitude if it followed a series of 1's. A disturbed 0 obtained under these conditions, then, should be the lowest 0.

This particular pulse sequence also aggravates the worst-case condition, because the high pulse repetition rate (2 mc) increases core temperature. In one of the materials examined, both cores become heated to about 20°C above room temperature. Core A, however, undergoes more flux reversals per cycle than core B, and becomes slightly warmer than core B. The temperature difference between the cores was found to be in the order of 5 to 7°C. This temperature tends to increase the 1 output and decrease the 0 output. Because core A is warmer, it not only has an increased output, but also a reduced coercivity which tends to lower the digit-disturb threshold.

The temperature effect was easily observed by switching from the program shown to one producing a series of undisturbed 0's followed by a disturbed 1. For about 3 seconds after the program change, the output amplitudes are unstable. Immediately after the program change, the undisturbed 0 is low and gradually increases to a higher stable value. Initially, the disturbed 1 is relatively high and stabilizes at a lower value. If the pulse program is again reversed, the undisturbed 1 is low initially, and gradually increases; the disturbed 0 is high initially and gradually decreases.

Figure 8 shows the effect of digit-current amplitude on output for a given material and for a given set of partial-write and read conditions. The upper curve is a plot of undisturbed 1 (or undisturbed 0) outputs as a function of digit-current amplitudes. The lower curve shows the effect of digit amplitude

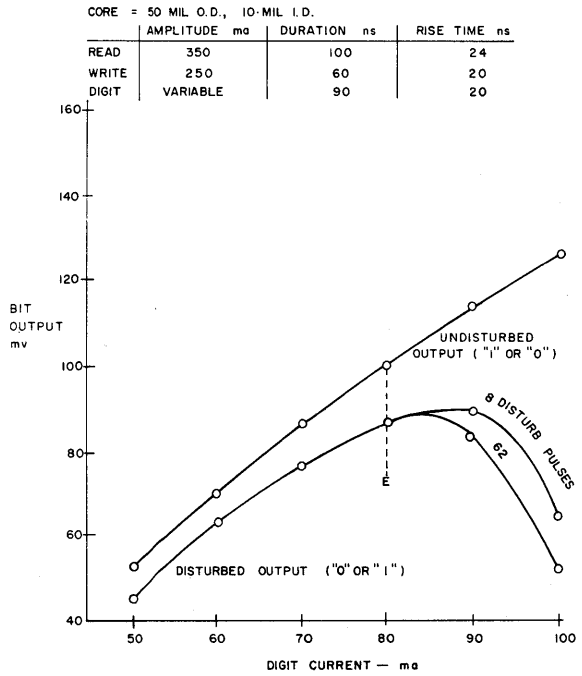


Figure 8. Bit Output vs. Digit Current (Bipolar Sensing).

on a disturbed 0 or disturbed 1. As expected, the undisturbed output gradually increases as the digit current level is increased. The disturbed output, on the other hand, goes through a maximum (in this case at about 90 ma). The decrease in 0 output beyond this point occurs because the disturb threshold of the core has been grossly exceeded at this digit-current level. This is particularly evident from the figure in that the disturbed curve is now "split," depending upon whether 8 or 62 disturb pulses were in the pulse program. Point E in Figure 8 is the current level at which a difference is first detectable between the effects of 8 and 64 disturb pulses. In practice, an operating digit level would have to be below 80 ma to be "safe." At current levels below point E, the disturbed outputs are generally significantly lower than the undisturbed outputs because of the heating effect described earlier. Evidently, the core is quite usable at these frequencies (2 Mc) in spite of this effect. (As indicated in the next section, this heat effect is eliminated in the assembly scheme actually used in the memory.)

Fabrication and Assembly Techniques

The ferrite pieces are fabricated using conventional dry-pressing techniques.

Generally, pieces have been made with circular and rectilinear geometries with aperture diameters ranging from 5 to 10 mils.

The method of assembling the cores or platelets is based upon a word-address, two-wire system. A new printed-winding technique is used to furnish one winding while conventional hand-threading techniques are employed for the other winding. The printed winding serves as either a common sense-digit line or as a common read-write line.

The assembly of platelets into strips is essentially a three-step operation:

1. Metallization of individual ferrite pieces by vacuum deposition.
2. Mechanical assembly of ferrite pieces into strips; at the same time the conductive path between pieces is completed.
3. Electroplating of the strip as a unit to improve contact between platelets and to lower the over-all resistivity of the conductive path.

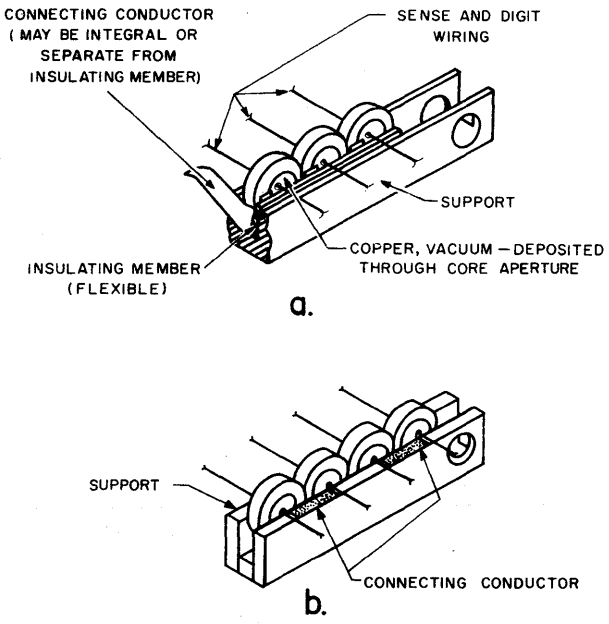


Figure 9. Strip Assembly Schemes.

The first step is a relatively simple operation and involves the vacuum metallization of the appropriate pattern on the individual ferrite pieces. The aperture wall is also metallized at the same time. Toroids (0.050-inch outer diameter, 0.010-inch inner diameter, 0.010-inch thick), have been generally used but in a few cases the pieces have been rectilinear.

Metallized cores, or platelets, have been assembled into strips in a variety of ways, but assembly is generally similar to the methods shown in Figure 9. In Figure 9a, the cores are held in a plastic member in which a photo-etched conductor connects the far side of one core to the near side of the adjoining core. These core connection complete the electrical path along the length of the strip. The plastic member rests in an aluminum support to facilitate handling and assembly.

Figure 9b shows schematically, another mode of "wiring" the platelets and supporting them. Figure 10 shows a plane of 32 words, 30 bits each, which was assembled by this technique. Figure 11 is a close-up of a section of the same array. The electrical connection between platelets is simple and the plastic piece is self-supporting. The return conductor is printed on the underside of the structural support and can be brought close to the actual core winding. This arrangement is important because it minimizes mutual inductances and coupling effects between word lines. The plastic piece serves as a structural support only. The final operation in the assembly of the strips is to electroplate the strip as a unit. This step lowers the over-all resistance of the conductive path.

As indicated previously, each strip may constitute one word of the memory stack, and the printed conductor is a common read-write winding. The strips, stacked side by side and wired as shown in Figure 10, constitute a plane. The planes may then be stacked vertically. Although the non-printed wire is hand-threaded with comparative ease, one of the obvious advantages of the strip concept is that it is easily adaptable to a mechanized threading.

The finished stack is a compact unit (packing density may vary between 1000 and 2000 bits/in.^{3}) and is particularly well suited to high-speed operation. If the printed wire is the read-write wire, then depending on the thickness of the plastic strip support, the sense-digit wire can be made very short and thereby reduce sense delays and stray noise pick-up. If a three-wire system is desirable it is a relatively simple matter to hand-thread a pair of wires through the plated aperture.

A further advantage of the strip assembly design arises from the fact that the metal plating on each core constitutes a convenient heat sink. In assembled strips, the adverse heating effect described earlier which occurs

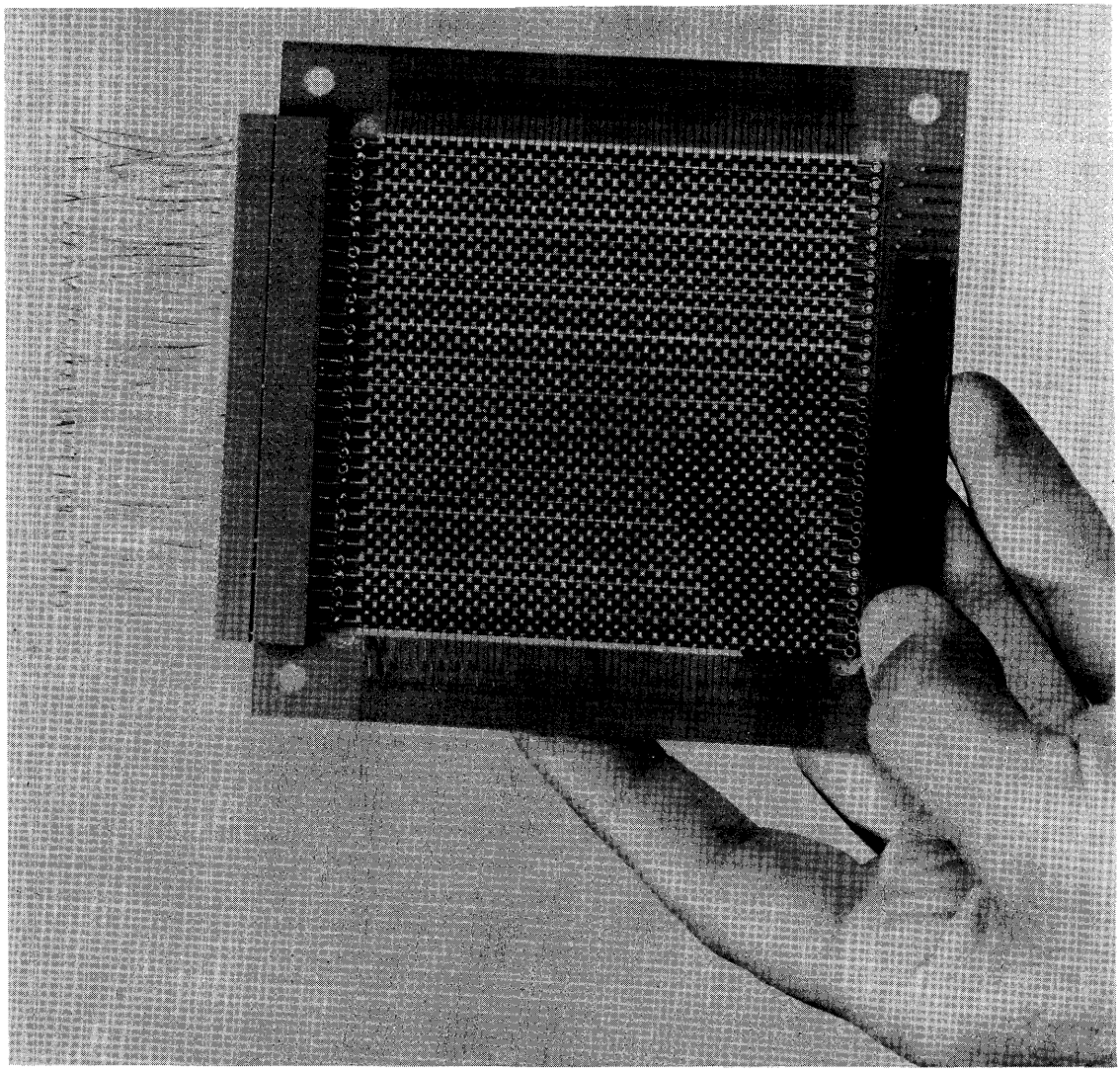


Figure 10. 32 x 30 Memory Plane.

at high repetition rates, no longer exists and the bit performance is correspondingly improved.

A Sixteen-Word Memory System

To prove the feasibility of two-core-per-bit high speed ferrite memories, a sixteen-word, twelve-bit-per-word memory system has been built and tested. This memory system uses a square platelets of 0.080 inch x 0.080 inch with a 0.006 inch hole in the center. The cycle time is approximately 200 nsec.

The memory plane contains all 16 words of 12 bits each and the diodes necessary to form a word selection matrix. In addition, it contains current transformers for measuring word drive current. The components are mounted on both sides of a printed circuit board, with the bottom side of the board also providing a ground plane. A photograph of the memory plane is shown in Figure 12.

Each platelet is threaded with two 0.002 inch enamel wires. One is a word line and carries read and write currents, and the other is a sense-digit line and carries digit current and sense signal. The platelets are placed on 0.15 inch centers in both directions, which are indicated by the particular connector and the diodes used.

The operating characteristics of the memory plane are shown in Table 1.

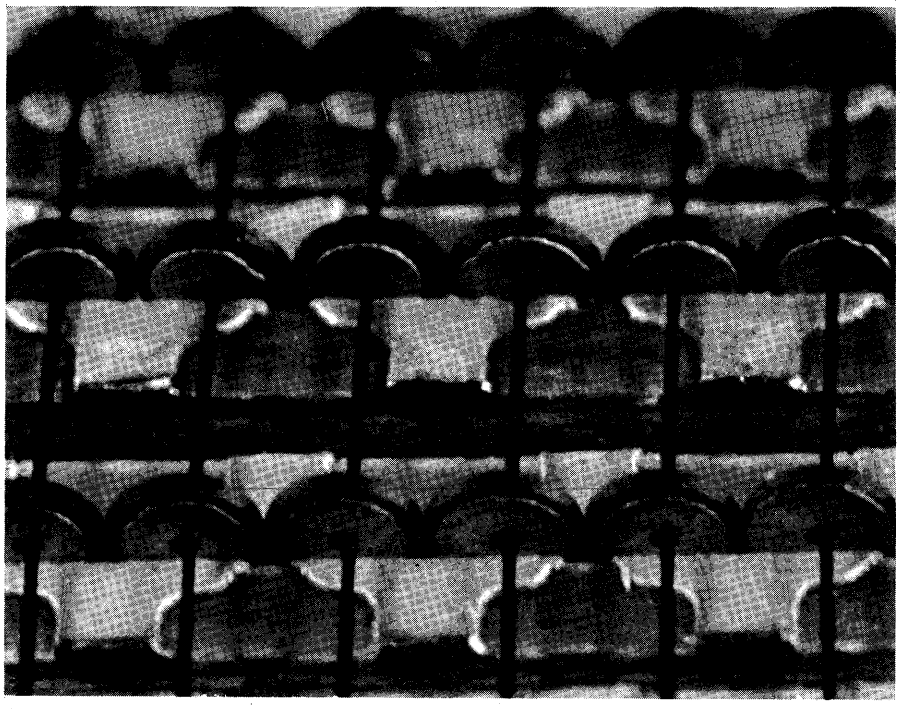


Figure 11. Close-up View of 32 x 30 Memory Plane.

Table 1

	Current Amplitude	Rise Time	Width (50% Point)
Read	320 MA	30 NS	70 NS
Write	200	20	40
Digit	60	10	40

Sense Signal (Bipolar) ± 30 to 60 MV

The memory system includes, in addition to the memory plane, the following electronic circuits:

1. Timing Generator
2. Four read drivers
3. Four write drivers
4. Four read switches
5. Four write switches
6. Four-bit memory address register
7. Address decoder
8. Twelve-bit memory register
9. Twelve sense amplifiers
10. Twenty-four digit drivers

The block diagram is shown in Figure 13, and a photograph of the memory system in Figure 14.

Two modes of memory operation are available. These are "regenerate" and "load." The memory register is reset to all 0's at the beginning of a memory cycle. In the regenerate mode, a strobe pulse is applied to the sense amplifier, and the information from a memory location is transferred to the memory register during the read portion of a memory cycle. During the write portion, the same 1's and 0's are written back into the memory location under the control of the memory register.

In the load mode, the strobe pulse is inhibited and the information from memory is discarded. The memory register is set externally at the desired bits so that during the

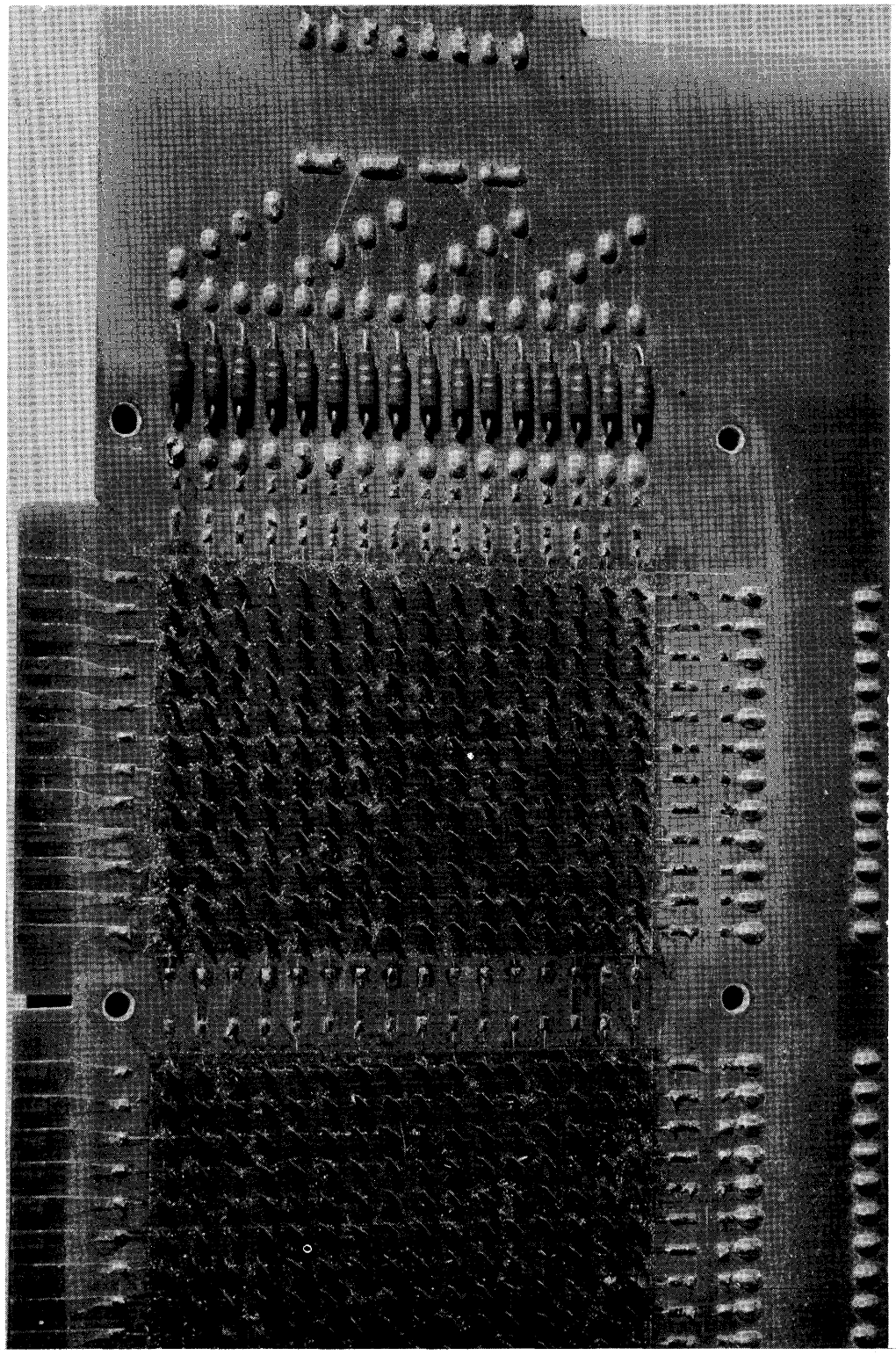


Figure 12. Close-up View of 16 x 12 Memory Plane.

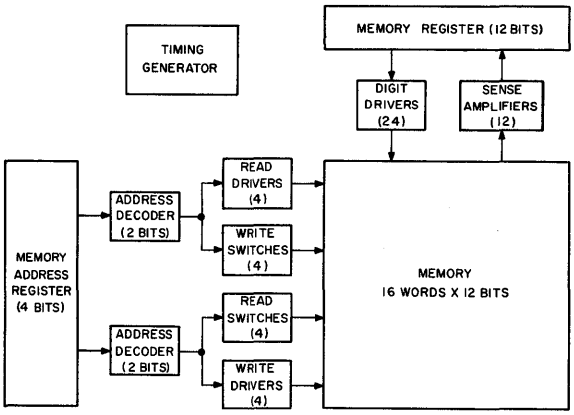


Figure 13. Block Diagram of the 16 x 12 Memory System.

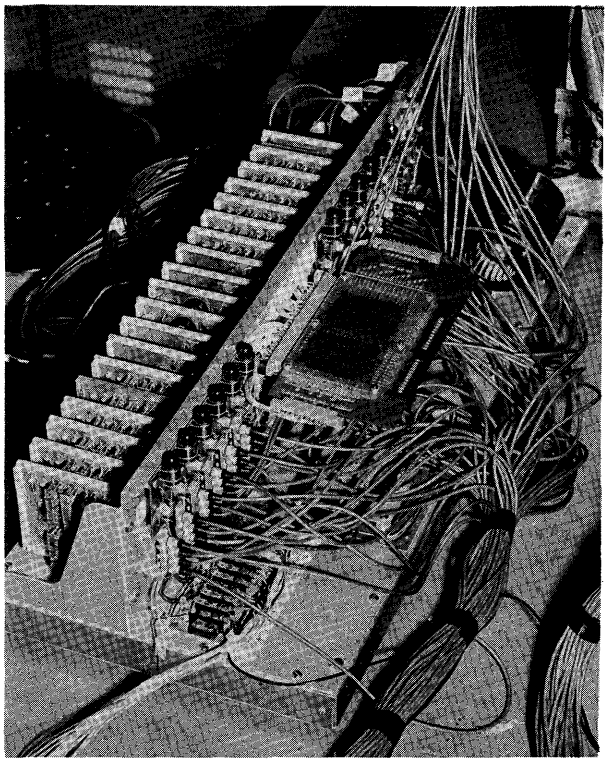


Figure 14. 16 x 12 Memory System.

write portion of the cycle, new information is placed into the memory location.

To test the memory system thoroughly, a memory exerciser has been built. Figure 15 shows waveforms inside the memory system operation under the control of the exerciser. This particular operation has been chosen because it is simple and yet informative. However, the exerciser is not limited to this operation.

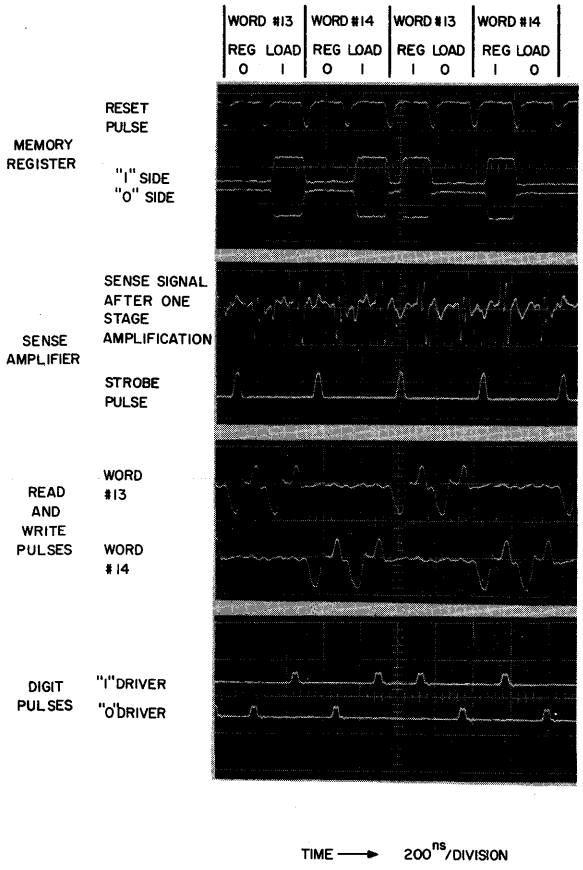


Figure 15. Waveforms Inside the Memory System Performing "Regenerate" and "Load" Operations.

The waveforms show one bit of two words regenerating and loading. Each word is selected in two consecutive memory cycles, first regenerate and second load. The bit loaded in the second cycle is the complement of the bit regenerated in the first cycle. Since the two words are selected alternately, the bit regenerated is the one loaded three memory cycles earlier. Eight memory cycles complete one full operation cycle, which is repeated.

The memory register is reset to 0 at the beginning of a memory cycle. A strobe pulse is applied when regenerating, and if a 1 is detected, the memory register is set. When loading a 1, the memory register is set externally.

The sense signal waveforms shows a positive signal for 0 and a negative signal for 1 at the strobe time. The read and write pulse

waveforms show which word is being selected. They are obtained at the secondaries of the current transformers mounted on the memory plane. The digit pulses occur in time coincidence with the write pulses, and indicate the bit which is written in.

Figure 15 shows a cycle time of 220 nsec. This cycle time can be reduced further by improving the electronic circuits.

CONCLUSIONS

The fabrication of microferrite, high-speed memories with cycle times less than 300 nsec at drive current levels less than 350 ma, has been found to be feasible and practical. Such prototype memories have been operated at memory cycle times as low as 200 nsec.

As for the size of a basic memory module, investigations show that one thousand words of one hundred bits per word are feasible. The delay on a sense-digit line then will be in the order of 30 nsec which will not increase the cycle time excessively. However, the high ratio of the sense-digit line delay to the cycle time will certainly necessitate a more complex pulse timing than that required for a slower memory. Sense-digit lines will have to be treated as transmission lines. The delay on a word line will be still so small compared to the cycle time that it will not require additional timing consideration.

Ferrite devices, as applied to high-speed memories, offer several advantages over other competing materials and devices. Not only are outputs high (50 mv) and current levels low but we have here an established technology of fabrication which by simple modification has been adapted to the economical fabrication of a high-speed device.

An important consideration in this program has been core and bit uniformity. Memory cell uniformity, or the lack of uniformity, in large part determines feasibility and cost of a device. Ferrite cores and platelets have been developed with excellent uniformity in two-core-per-bit application so that in this context there is absolutely no problem.

Ferrites offer further advantages in being generally insensitive to radiation damage. In addition, it appears quite practical to combine this technique with recently developed, temperature-stable ferrites. These materials can perform without current compensation over wide temperature ranges (100°C) and, by varying current drives, up to 400°C.

Thus a high-speed, temperature-stable memory operating at low current levels and which is generally insensitive to radiation damage, appears to be not only practical but actually realizable in the immediate future.

ACKNOWLEDGMENT

The authors acknowledge the active roles played by several coworkers on this project. Special credit is due to the following members of the Memory Products Operation: J. J. Sacco, A. Erickson, B. Frackiewicz, R. Gravel, D. Kadish, P. Lawrence, and H. Lessoff. Additional credit is due to H. C. Nichols of the Electronic Data Processing Division for the fabrication of the memory system. Special acknowledgement is due to RCA Electron Tube Division, Harrison, for help in developing vacuum-deposition methods, and to the RCA Laboratories at Princeton for their applied-research support. A great deal of this development is a result of work in high-speed ferrites previously undertaken at the laboratories.

BIBLIOGRAPHY

1. Jan A. Rajchman, "Static Magnetic Matrix Memory and Switching Circuits," RCA Review, vol. 13, pp. 183-201, June, 1952.
2. J. W. Forrestor, "Digital Information Storage in Three Dimensions Using Magnetic Cores," J. Appl. Physics, vol. 22, pp. 44-48, January, 1951.
3. Jan A. Rajchman, "Computer Memories: A Survey of the State of the Art," Proc. of the IRE, pp. 104-127, January, 1961.
4. C. A. Allen, G. D. Bruce, and E. D. Councill, "A 2.18 Microsecond Megabit Core Storage Unit," IRE Transactions of Electronic Computers, p. 233, (1961).
5. J. A. Rajchman, "Ferrite Apertured Plate for Random Access Memory," Proc. of the IRE, vol. 45, pp. 325-334, March, 1957.
6. W. H. Rhodes, L. A. Russell, F. E. Sakalay, R. M. Whalen, "A 0.7 Microsecond Ferrite Core Memory," IBM Journal, pp. 174-182, July, 1961.
7. M. M. Stern, and H. A. Ullman, "The Maximum Efficiency Concept for High-Speed Random-Access Memories," NEREM Record, November 1961.

8. M. S. Blois, "Preparation of Thin Films and Their Properties," J. Appl. Phys., August, 1955.
9. J. I. Raffel, "Operating Characteristics of a Thin Film Memory," J. Appl. Phys., April, 1959.
10. R. J. Petschauer, R. D. Turnquist, "Nondestructive Readout Film Memory," Proc. Western Joint Computer Conference, 1961.
11. J. I. Raffel, et. al., "Magnetic Film Memory Design," Proc. of the IRE, pp. 155-164, January 1961.
12. R. J. Petschauer, and K. Leenay, "Matrix Memories—A Comparative Evaluation," Electronic Design, August, 1961.
13. R. Shahbender, T. Nelso, R. Lochinger, and J. Walentine, "Microaperature High-Speed Ferrite Memory," Fall Joint Computer Conference (1962).
14. R. H. Tancrell, R. E. McMahon, "Studies in Partial Switching of Ferrite Cores," J. Appl. Physics, vol. 31, no. 5, pp. 762-771, May, 1960.

SUMMARY

High speed ferrites have been developed capable of operating in memory cycle times of less than 300 nsec. Drive current levels are less than 350 ma and well within the capabilities of present-day transistor drivers. High speeds have been achieved utilizing a word-address, two-core-per-bit memory organization. This paper describes the performance of low-drive, fast-switching memory cells, methods for the assembly of these cells into arrays economically, and the operating characteristics of 16-word, 12-bit memory system.

Bit performance is determined by the use of two general methods which both involve two-core-per-bit operation. The essential difference between the two methods is that in one method either core of the bit may receive a digit pulse while in the other only one core receives a digit pulse. Thus in the former situation bipolar signals are obtained upon readout while in the latter case only unipolar signals are obtained. Outputs of ±50 mv are generally obtained in the bipolar sensing scheme; in the amplitude sensing scheme the 1 output is about 50 mv and the 0 output, 10 mv. These are obtained using drive currents of less than 350 ma and read-write cycle times less than 300 nsec.

Ferrite cases and platelets are fabricated using conventional dry-pressing techniques. The ferrite piece is either of circular or rectilinear geometry with aperture diameters ranging from 5 to 10 mils. These are assembled by a new method involving the use of a printed-winding technique to furnish one winding and conventional hand-threading for the other winding. Ferrite pieces have been developed with excellent uniformity in this two-core-per-bit application.

A memory system of 16 words, 12 bits per word has been built and operated successfully. The ferrite piece used in this case was an 80 mil square platelet with a 6 mil aperture. The memory cycle time is about 200 nsec.

MICROAPERTURE HIGH-SPEED FERRITE MEMORY

R. Shahbender, T. Nelson, R. Lochinger and J. Walentine
RCA Laboratories, Radio Corporation of America
Princeton, N. J.

INTRODUCTION

Present day digital computers universally employ ferrite cores for random storage. Available memories, operating in a current coincident mode, have capacities up to approximately 2×10^6 bits and cycle times in the neighborhood of 10 microseconds [1]. Shorter cycle times around 1 microsecond have been achieved in commerical systems by resorting to a word organized mode of operation [2]. Because of the increased cost of the required electronic circuitry, bit capacities are in general less than those of a coincident current store.

Impulse switching techniques [3] together with advanced fabrication technologies are being used to reduce cycle times to the vicinity of 100 nanoseconds. In this paper are presented the results establishing the feasibility of medium size, random access ferrite memories with cycle times of 100 nanoseconds. This goal, sought by the competitive approach based on magnetic thin films [4,5,6], is achieved by the well established technology of ferrites.

The general philosophy adopted for the program is:

1. Miniaturization of the size of the storage element to attain high speed.
2. Use of printed circuit techniques to facilitate fabrication and assembly.

The miniaturization of the size of the element leads to power requirements for high speed switching that are compatible with semiconductor devices. The use of printed circuit techniques permits the use of automatic fabrication processes with potential cost reduction.

Electron beam milling, a recently developed fabrication process, opens up vistas in micro-miniaturization. The use of this technique as described in this paper to form micromagnetic elements is timely since the minimal size limits attainable lie intermediate to those achieved by current techniques and those to be expected from research programs in the area of miniaturization by controlled crystal growth. Further, the automatic programming of milling operations is easily accomplished by controlling the position, energy, and size of the milling tool, viz: the electron beam. The process is particularly suited to automation since the variables are naturally suited to electronic control.

Ferrite Switching Characteristics

The switching characteristics of a toroidal core are described by the well known relationship [7]:

$$T_s \ (H - H_c) = Sw$$

where T_s = Switching time
H = Applied magnetic reversing field
H_c = Coercive field of the core
S_w = Switching constant characteristic of the core composition

This relationship has been experimentally verified over a large range of applied fields and for a variety of material compositions (both ferrites and magnetic metals). The experiments indicate a decreasing value of S_w for high fields [8].

Irrespective of the mechanism involved, experiments have shown that a core may be switched in a few nanoseconds by the application of a sufficiently high drive field. For the case where:

$$H >> H_c$$

the switching relation reduces to

$$T_s I \simeq 2\pi r_o S_w$$

where I = applied reversing current (assuming single-turn windings)
r_o = average core radius

Since a core is a closed magnetic path there are no demagnetizing fields present. This implies that the average core radius is not limited to a range of values by fundamental considerations due to demagnetizing effects. Thus, for a given switching speed, the current required to reverse the flux around an aperture is reduced by reducing the average radius r_o of the aperture, and/or the switching constant of the material.

For a toroidal geometry, the switching current decreases linearly with decreasing radius. The current carrying capacity of a conductor linking the core decreases as the square of the radius. This places a lower limit on aperture radius given by

$$r \geq \frac{S_w S}{K T_s} = 0.55 \times 10^{-6} \text{ meters (= 0.02 mils)}$$

where $K = 10^9$ Amps per square meter = permissible conductor current density
T_s = 30 nanosecond = Switching speed required for 100 nanosecond memory
$S_w = 8.3 \times 10^{-6}$ Amp-Seconds/meter (0.1 oersted-microsecond)
= Lowest switching coefficient measured for ferrites.

For a core of average radius r_o, radial extent Δr, and axial length L, the energy E required to switch it, and the output voltage V derived from it are:

$$E \alpha \frac{S_w L}{T_s} r_o \Delta r$$

$$V \alpha \frac{L \Delta r}{T_s}$$

The ratio of output voltage to energy, which may be considered as a figure of merit for magnetic elements, is:

$$\frac{V}{E} \alpha \frac{1}{S_w r_o}$$

This ratio may be increased, at constant output voltage by reducing the effective core size. The output voltage is determined by the product of L and Δr. For obvious reasons Δr must be kept small (a fraction of r_o). The axial length L may not be made arbitrarily large to obtain arbitrarily large output voltages since this introduces delay and slows the memory cycle. Thus depending on the actual memory element configuration a specified value of L must be used representing a compromise between output voltage and delay.

The expressions given above may be used to estimate the element size required for a given drive current. Drive current limitations exist since the memory is to be operated with associated electronic circuitry capable of delivering the required pulses at a 10 Mcps repetition rate. Assuming a drive current of 200 mA, a reasonable value for solid state devices, and a switching time of 30 nanoseconds leads to an average core radius of

$$r_o \leq \frac{T_s I}{2\pi S_w} = 120 \times 10^{-6} \text{ meters}$$
$$(\simeq 5 \text{ mils} = 0.005 \text{ inches})$$

Memory Element and System Organization

The memory element selected consists of a ferrite wafer with dimensions $80 \times 80 \times 10$ mils. Four apertures with a diameter of 1 mil each and a center to center spacing of 2 mils are electron beam drilled in the center of the wafer as shown in Figure 1. Four separate windings, each linking an aperture in the pattern shown in Figure 2, are fabricated by photo-etching techniques to give a wired element. The elements are assembled in a mosaic and the individual windings are interconnected by mass fabrication techniques to give a wired memory plane as shown schematically in Figure 3.

The memory is operated in a word organized, two core per bit mode [2] under impulse switching conditions. The two X-directed windings are for the word currents: the read and write currents are applied to different

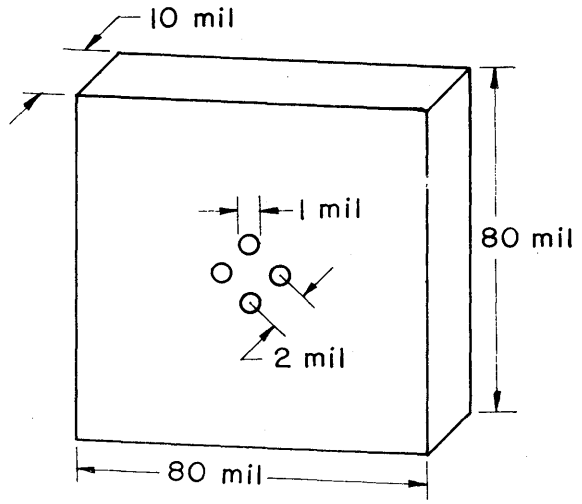


Figure 1. Geometry of a Microaperture Ferrite Element.

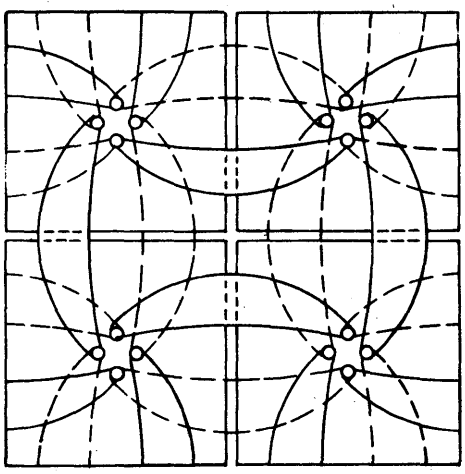


Figure 3. Mosaic of Interconnected Wafers (Four Wire System).

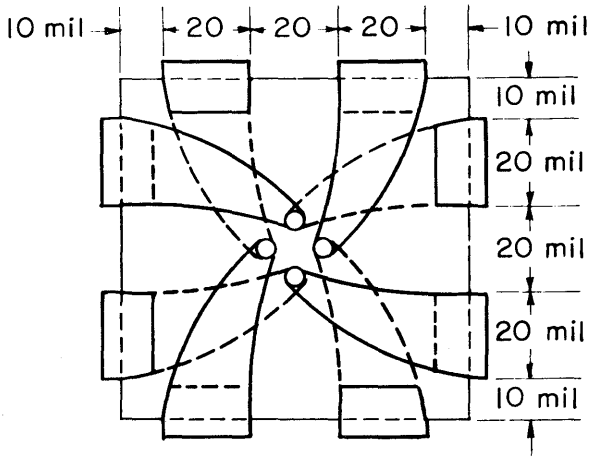


Figure 2. Winding Pattern for Microaperture Element.

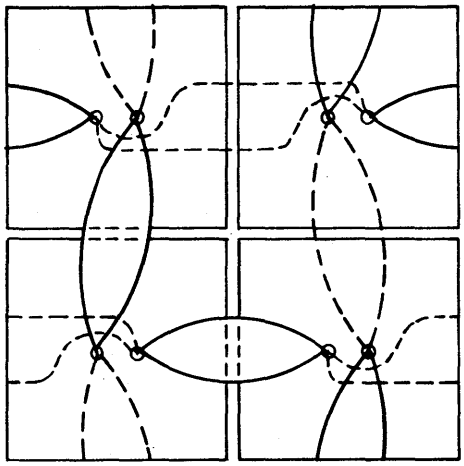


Figure 4. Mosaic of Interconnected Wafers (Two Wire System).

windings. The two Y-directed windings are for digit and sense, respectively. The effective internal diameter of an equivalent core is approximately 4 mils (this is the diameter of the circle circumscribing the four apertures). The effective external diameter is determined by the magnitude of the switching currents.

A two wire system is also possible, and in fact may be preferable, in which bipolar pulses for read-write are applied to the X-directed winding, and the Y-directed winding is used for digit-sense. The ferrite element and winding patterns are suitably modified by having two apertures per wafer (each with a diameter of 1 mil and a center to center spacing of 2 mils) instead of four and the winding pattern shown in Figure 4.

The four wire system reduces the burden on the electronic drive circuitry since unipolar pulses are needed on the word lines. It offers a marginal advantage in having separate digit and sense windings because of the inherent tight coupling between these windings. The two wire system requires bipolar pulses on the word lines. However, it offers an advantage in halfing the number of apertures and interconnections. The reduction in the number of windings per wafer permits the use of rectangular wafers with a reduced dimension in the digit direction. This decreases propagation delays along the

digit-sense windings which is of importance for high speed operation [9].

Memory Fabrication Technology

Ferrite Wafer Fabrication:* Dry pressing techniques are utilized to fabricate the ferrite wafer blanks for the memory system. Investigations were conducted to determine optimum conditions (compositions, particle size, binder content, compacting pressure, firing schedule, etc.) for fabricating the blanks with the required magnetic characteristics and a high degree of dimensional tolerance to permit mosaic assembly. As a result of these investigations blanks are being fabricated with nominal dimensions of 80 × 80 × 10 mils, with tolerances of ±0.5 mils in lateral dimensions, and ±0.1 mils in thickness for an 80 per cent yield. The compositions used are of the conventional variety, with the metal ion content and final firing temperature adjusted to give the required magnetic characteristics. For the composition selected, the coercive force is approximately 1.0 oersted and the switching coefficient is 0.3 oersted-microseconds. A variation in firing temperature of more than ±20°C is sufficient to cause serious deterioration in the high speed switching characteristics.

Microaperture Drilling:† The microapertures are formed in the blank wafers by electron beam drilling. In this recently developed technique a beam of high energy electrons is focused onto the work piece in the manner indicated schematically in Figure 5. The energy of the incident electrons is absorbed at the point of impact by the work piece and converted to heat. Exceedingly high power densities (10^8 watts per square centimeter) are attained resulting in the local generation of very high temperatures and high temperature gradients. To thermally protect

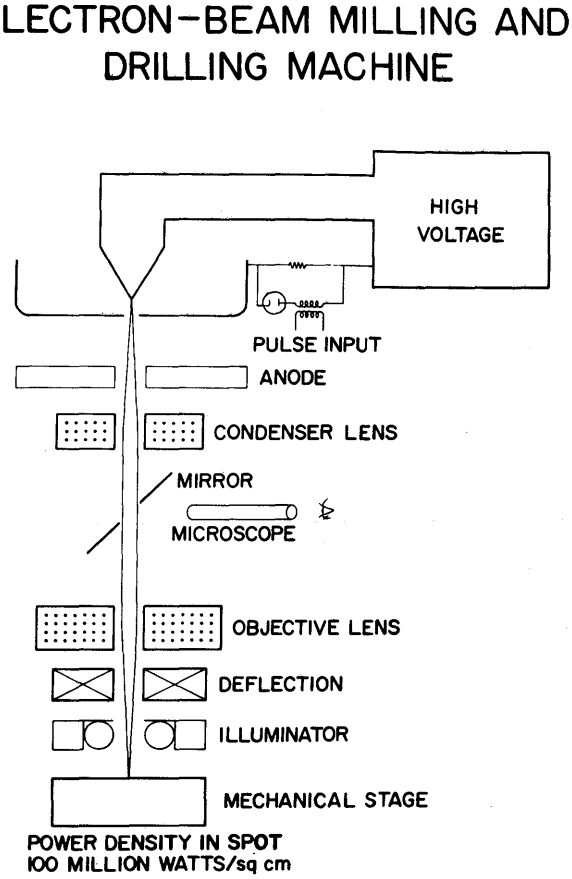


Figure 5. Schematic of the Electron Beam Drilling Machine.

the work piece and confine the elevated temperature zone to the impact region, a pulsed beam is used. For ferrites the locally generated heat is sufficient to sublimate the material without causing excessive recrystallization in neighboring regions.

To drill the microapertures in the ferrite wafers, beam voltages in the range of 80 to 100 KV, peak beam currents in the range of 150 to 200 microamperes, pulse widths of 10 microseconds and pulse repetition rate of 100 per second were found optimum. The beam is focused under these conditions to result in an aperture with a diameter of 1 mil. Smaller diameter apertures (as small as 0.3 mils) were also successfully drilled. The experimental work performed indicates that the optimum drilling conditions depend on the ferrite composition and the thickness of the work piece. An increase in wafer thickness requires a higher drilling voltage. Samples

*This phase of the work was conducted at the RCA Semiconductor and Materials Division - Ferrite Operation, Needham Heights, Mass., under the direction of J. J. Sacco.

†This phase of the work was performed using the electron beam milling facility of L. R. Industries, Mt. Vernon, N. Y., available on a jobbing basis. A description of this facility is included in the "Proceedings of Annual Symposium on Electron Beam Processes" held by the Alloyd Corporation on March 24-25, 1960.

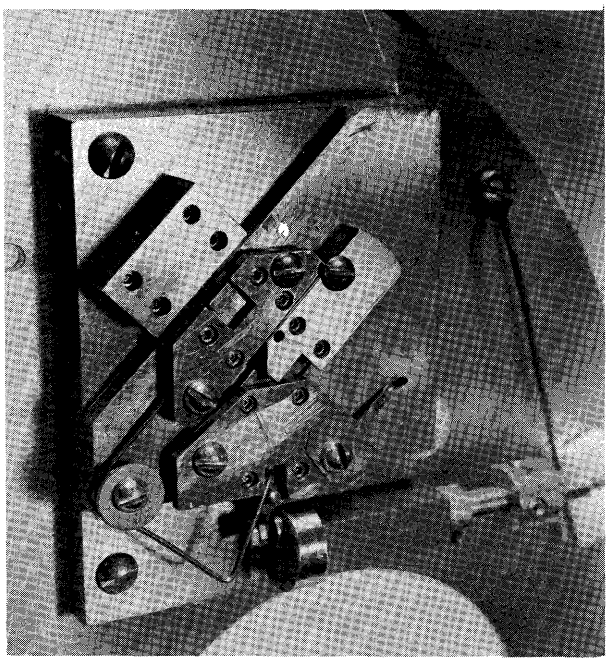


Figure 6. Drilling Jig.

as thick as 30 mils were successfully drilled by increasing the beam voltage to 135 KV.

The electron beam drilling machine used for the experimental work under optimum focusing conditions resulted in apertures with a diameter of 0.3 mils. To drill larger diameter apertures the beam is defocused. This results in a non-optimum power density distribution across the beam, a reduction in the temperature gradient in the work piece and an increase in the size of the recrystallized zone surrounding the drilled apertures. To overcome this difficulty a redesign of the electron-optics is necessary.

To drill the four aperture pattern of Figure 1, the beam is sequentially deflected to the four positions by energizing the deflection coils shown in Figure 5 from a relay control unit.

To facilitate the drilling operations, the semi-automatic jig shown in Figure 6 was designed and built. The function of the jig is to precisely position wafers, supplied from a storage magazine, under the drilling beam and to remove the wafers after they are drilled. The precision in positioning the wafers under the beam is necessary to facilitate the fabrication of the windings. The jig is made of nonmagnetic stainless steel and is actuated via a flexible shaft and rotary vacuum seal by an electric motor mounted outside the vacuum chamber. The jig operates without the benefit of lubricants because of the effects of the high x-ray levels generated during drilling. The supply magazine has a capacity of 120 wafers which are manually loaded prior to insertion in the machine. The drilling rate of 6 wafers per minute achieved is mainly limited by operation of the jig rather than the time required for the actual drilling.

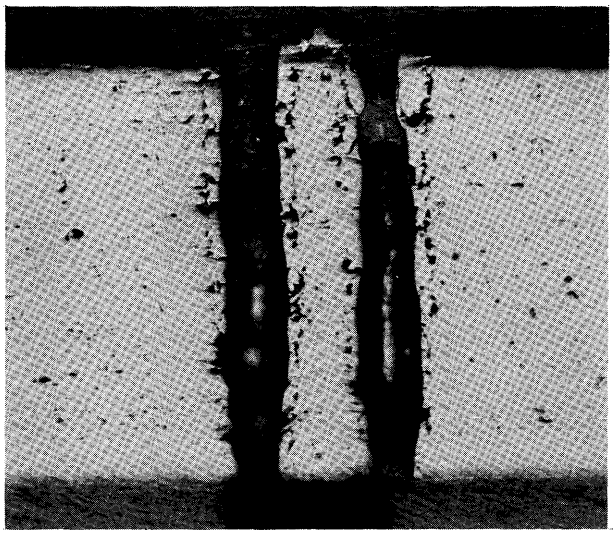


Figure 7. Micrograph of Section through Two Electron Beam Drilled Apertures.

Figure 7 is a micrograph of a polished section through two apertures in an 8.8 mil thick wafer. The recrystallized zone is clearly visible. In addition, a small crater and mound can be seen at the top of the sample corresponding to the entry side of the beam. The crater and mound formation seems to be inherent to electron beam drilling and necessitated lapping the entry surface prior to fabricating the windings. Figure 8 is a micrograph of the surface of a sample after lapping. Again the recrystallized zone can be clearly seen.

Winding Fabrication:* To fabricate the windings on the individual wafers the following three-stage process is used:

(1) The apertures are filled with a conducting paste consisting of a mixture of finely powdered silver and silicone oil. The paste

*This phase of the work was conducted at the RCA-Electron Tube Division, Harrison, N. J., in the Chemistry and Physics Laboratories under the direction of N. Freedman.

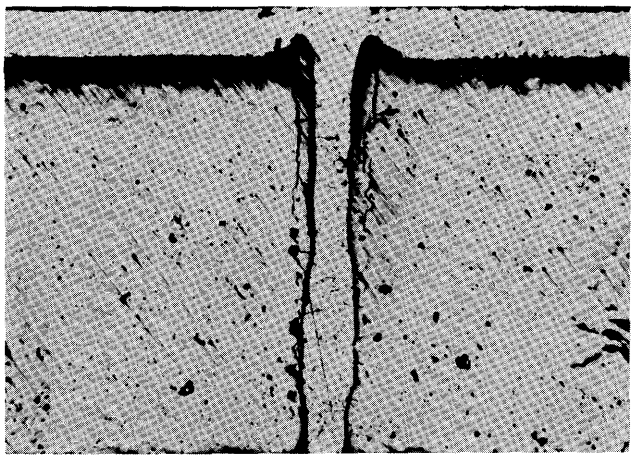


Figure 8. Micrograph of the Surface of a Lapped Wafer.

is placed on a glass substrate and the wafers are pressed into it. This forces the paste into the apertures. The paste is sintered at a low temperature and the surfaces of the wafers are lapped to remove the excess paste and the crater and mound formed during the electron beam drilling. Figure 9 shows a photomicrograph of the cross section of an aperture with the sintered silver paste in it.

(2) The two surfaces of the wafer are metalized by vacuum evaporation with an 0.5 mil thick layer of copper. The deposited copper surface is a replica of the ferrite surface and under proper illumination shows the location of the apertures as depressions.

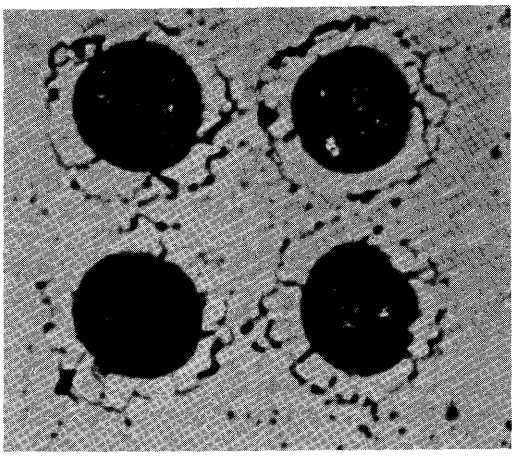


Figure 9. Micrograph of Section through Drilled Aperture Showing Injected Silver Conductor.

(3) The metalized wafers are coated with photo-resist, optically aligned in a high magnification system with a photo master and exposed to UV. The copper is then etched to give the required winding pattern and the photo-resist is then removed.

Because of the high precision attained in drilling the apertures with respect to a reference corner, coarse alignment (within a few tenths of a mil) is achieved by the jigging fixtures used, thus the final alignment requires only small motions of the wafer with respect to the photo master.

The electrical resistance of the silver plugs, as measured by two fine probes, had a minimum value of 0.04 ohms. A total winding resistance of 0.1 ohms was acceptable and those showing a higher value were rejected.

Mosaic Assembly: The winding pattern shown in Figure 2 is designed to permit all interconnections to be performed from one surface. This is accomplished by having the windings overlap the edges of the wafer. To interconnect the wafers a printed circuit board with a pattern of holes and interconnecting tabs is fabricated. The holes are 55 mils in diameter and the interconnecting tabs are photo-etched in the copper. A low temperature solder is electroplated onto the board prior to etching. The wafers are assembled into a mosaic onto the board and are held in position by the application of a vacuum to the back surface of the board. The entire assembly is heated to effect the soldering of the wafers to the interconnecting tabs.

Figure 10 is a photograph of a 4×4 mosaic assembled in this fashion. The corner wafer is removed to show the hole and interconnecting tabs. Because of difficulties encountered in having the windings overlap the edges of the wafers, the interconnections on the top surface of the mosaic were manually soldered. An improved assembly technique is under development to permit the interconnections to be made on both surfaces. This eliminates the need for having the windings overlap the edges of the wafers.

In addition to the 4×4 mosaic shown in Figure 10, a 4×8 mosaic was assembled. Total winding resistance ranged from a low value of 0.2 ohms to a high value of 10 ohms. A 12×16 mosaic is presently being assembled.

Magnetic and Electrical Characteristics

Operating Mode: The mode of operation selected for the memory is word organized

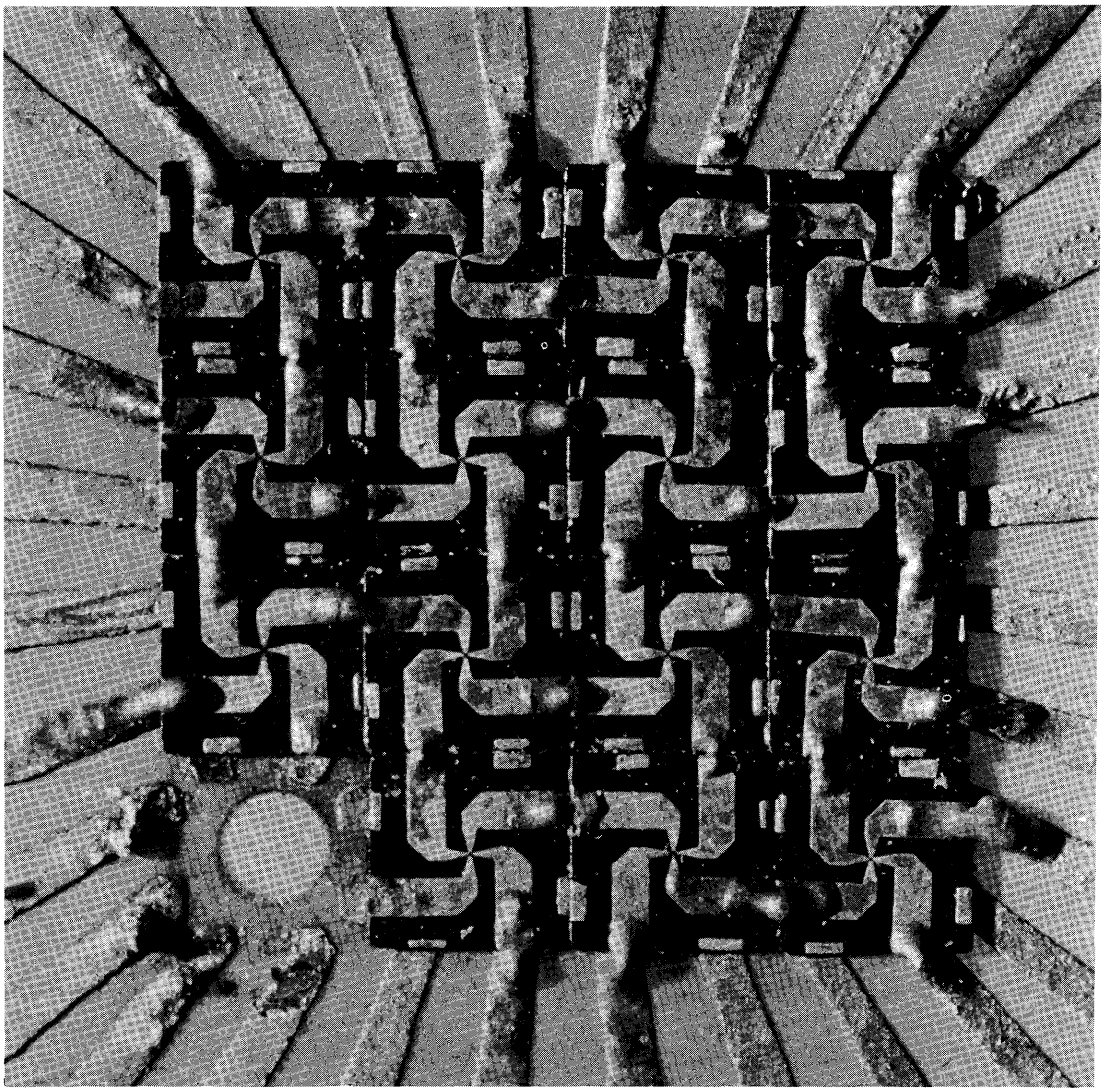


Figure 10. 4 × 4 Mosaic of Microaperture Wafers.

with two wafers per bit. In this mode all bits of a selected word are subjected to the same read-write current pulses. The optimum digit drive technique,* selected on the basis of experimentation, applies a digit pulse to either one or the other of the two wafers of a bit in time coincidence with the write pulse and of such a polarity to always add to it. In other words, if the two wafers of a bit are labelled A and B, to write a binary "1" wafer A is digited and to write a binary "0" wafer B is digited, the polarity of the digit pulses in both cases being chosen to add to the write current pulse.

Sensing is differential in that the output sense voltage obtained during read is the difference between the two voltages induced along the digit-sense windings linking the two wafers of a bit. The sense voltage corresponding to the two binary states is bipolar since the difference in the irreversible flux switched in the two wafers of a bit is positive or negative depending on which of the two wafers received the digit pulse.

The digit-sense windings link corresponding wafers in all words of the memory. Thus the digit pulses, unless controlled in magni-

*This technique was suggested and verified by H. Amemiya of the RCA Electronic Data Processing Division - Advanced Development, Pennsauken, N. J.

tude, will disturb the information stored in the memory. The effect of the digit pulses acting to disturb the stored information is minimized by the digiting technique described above. Alternate digiting techniques, e.g., in which a binary "1" is entered by adding mmf to wafer A and subtracting mmf from wafer B by digit pulses, and the reverse for a binary "0", may in principle result in larger sense voltages for the same magnitude currents; experimentally resulted in sense voltages that were reduced to zero or even reversed in polarity under appropriate disturb conditions. The alternate digiting techniques investigated always resulted in sense outputs considerably smaller than those obtained with the technique described above, when properly disturbed, and with the current amplitudes adjusted to optimum.

The observed disturb effects indicate that a partially switched ferrite element has a considerably higher disturb threshold for current pulses tending to switch additional flux in the element as contrasted to current pulses of the opposite polarity. For the digiting technique selected, the disturb pulses are unipolar and of the same polarity as the write current resulting in minimal changes in the sense output voltages as a result of the application of disturb pulses following writing and prior to reading.

In a random access memory operating with a cycle time of 100 nanoseconds, digit disturb pulses may occur at a minimum time separation of 100 nanoseconds and a maximum repetition rate of 10 mcps after the initiation of writing at a given location and prior to reading at that location. The total number of applied disturbs is essentially unlimited. As is to be expected, the repetition rate at which the disturb pulses occur influences the degree to which they disturb the magnetic states of the two wafers constituting a bit. To a first order this may be explained on the basis of the change in average value of a long sequence of unipolar pulses of fixed width and amplitude and variable repetition rate. The higher the repetition rate, the higher the average value of the pulse train and the greater is the disturb effect. Further, since the digit pulses are time limited, each disturb pulse can produce partial effects only. Thus the full disturb effects can only be assessed by applying a larger number of (the order of a hundred) disturb pulses.

The minimum elapsed time between writing and the application of disturb pulses is of great importance in determining the effect of the disturb pulses. In general, for the ferrite compositions tested, elapsed time of 0.5 microseconds or more resulted in indistinguishable disturb effects. For elapsed time shorter than 0.5 microseconds the disturb effects increase for decreasing elapsed times. Similar effects are reported in the literature [10] and are explained in terms of relaxation mechanisms. The experimental work performed indicates that this effect is material dependent and has been reduced significantly in the composition selected for the memory.

Experimental Data

Ferrite Compositions: Ferrite compositions investigated are of the conventional square-loop type with variable metal ion concentration. Both hand wired electron beam drilled wafers and toroids were pulse tested to determine the high speed switching characteristics. The toroids were also used to obtain low frequency hysteresis loops. The results indicate a dependence of switching coefficient, coercive force, squareness (defined as the ratio of remanent flux to saturation flux) and Curie temperature on the concentration of metal ions. The minimum switching coefficient measured is approximately 0.15 oersted-microseconds, with corresponding coercive force of 0.1 oersteds, Curie temperature of 70°C, and squareness ratio of 0.5. The material selected for the memory has a switching coefficient of 0.3, a coercive force of 1.0, a Curie temperature of 100°C and a squareness ratio of 0.7.

Bit Operation: A number of wafers of different compositions are electron beam drilled with the two aperture geometry of Figure 4 (1 mil diameter on 2 mil centers). Two wafers of a kind are hand wired (using 0.7 mil diameter wire) and mounted on strip line board for operation as a memory bit. A schematic of the circuit configuration is shown in Figure 11. Read, write and digit pulses are provided by transistor drivers (fixed pulse width of 30 nanoseconds at the base and adjustable amplitude) triggered by commercially available 10 Mcps logic building blocks. The sense output transformers are 1:1 turns ratio (10 turns on each side) wound on ferrite beads.

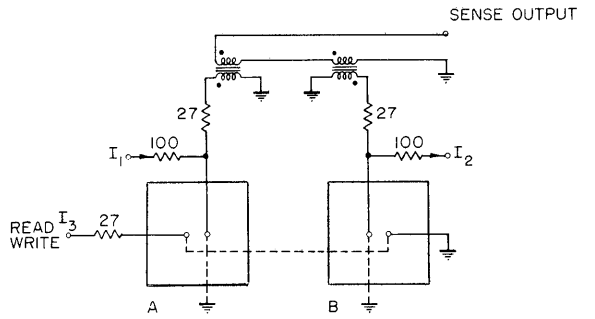


Figure 11. Circuit Used to Test Memory Bit.

Three basic pulse programs are used to evaluate the performance of a bit. These are:

(a) Alternate "One-Zero" read-write

(b) Alternate "One-Zero" read-write with disturbs

(c) Mixed "One-Zero" read-write with disturbs

Alternate "One-Zero" Read-Write: In this program an alternating pattern of "One and Zero" is cyclically written into, and read out of the memory bit at a cycle time of 100 nanoseconds and a repetition rate up to 10 mcps. Figure 12 shows the pulse program used and Figure 13 shows oscillograms of the voltage developed at the sense output terminals under constant drive conditions and at three different repetition rates: 0.05, 2.0, and 10 Mcps. The "one" and "zero" outputs are shown superimposed at the lower repetition rates and separated by 100 nanoseconds at the 10 Mcps repetition rate. Inspection of Figure 13 shows that the sense voltage is independent of repetition rate. The operating current levels and sense outputs are:

Read Current: 600 mA
Write Current: 250 mA
Digit Current: 45 mA
Undisturbed Sense Output: ±85 mv

Alternate "One Zero" Read-Write with Disturbs: In this program an alternating pattern of "One and Zero" is cyclically written into the memory bit. Each write operation prior to reading is followed by the application of disturb pulses aimed at destroying the stored information. Figure 14 shows the pulse program and Figure 15 shows oscillograms of the voltage developed at the sense output terminals for the same current drive levels as above.

The disturbed sense voltages are ±65 mv; a decrease of 20 mv due to the digit disturbs. Experimentally it was determined that 3×10^5 disturb pulses produced no additional disturb effects as compared to only 100 pulses.

Mixed "One-Zero" Read-Write With Disturbs: The pulse program described above is not the most pessimistic. A further deterioration of the sense signal occurs if the following pulse program is applied: a large number of 1's, each followed by disturbs in the 1 direction is written and read from the bit; this is followed by a single zero, which is then disturbed in the 1 direction. This zero is then read out and the output voltage recorded. The mirror image of this program is then applied, and a one read out and recorded. The program is shown in Figure 16, and Figure 17 shows oscillograms of the voltage developed at the sense output terminals for the same current levels as above. The sense voltages obtained are ±55 mv. Comparing Figures 13, 15 and 17, a progressive decrease in output voltage is noted. It is felt that the pulse program of Figure 16 represents a sufficiently stringent test so that a memory bit that yields the acceptable output of Figure 17 under the conditions of the test will perform satisfactorily in a computer under random conditions.

Uniformity of Electron Beam Drilled Wafers: The uniformity of the magnetic characteristics of electron beam drilled wafers is obviously of importance in determining the feasibility of this fabrication process. A measure of this uniformity is obtained by measuring the difference in output voltages between a reference wafer and a test wafer for the same read current pulse and the following write conditions:

(a) Both wafers subjected to equal write pulses (Zero State)

(b) The reference wafer is subjected to the write pulse and the test wafer is subjected to a write plus digit pulse (One State)

The circuit configuration used is shown in Figure 18. The data obtained for four groups with a total of 34 unselected, hand wired wafers is plotted in Figure 19. The four groups were drilled in succession under the same operating conditions with the vacuum released and the machine pumped down after each group drilling. The test current pulses are 50 nanoseconds wide at the base and have the following amplitudes:

Read Current: 400 mA
Write Current: 200 mA
Digit Current: 50 mA

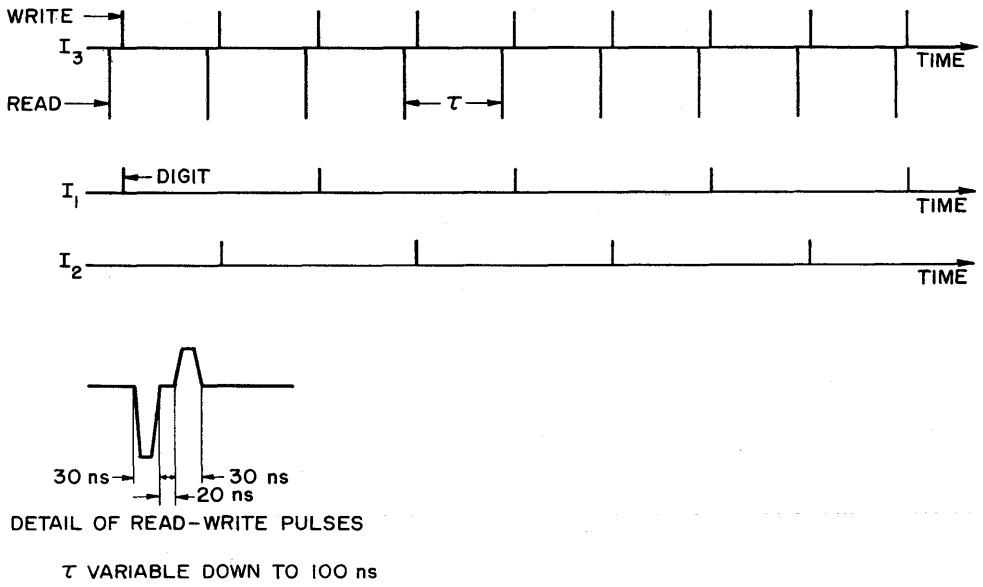


Figure 12. Alternate One and Zero Pulse Program.

The data of Figure 19 shows a spread of approximately 25 mv in the "1" output and a somewhat smaller spread in the "0" output. This spread in outputs is tolerable for operation with the sense voltages shown previously. However, since each wafer is fabricated with its individual windings, automatic pretesting with segregation into batches prior to assembly is feasible.

Temperature Rise Measurement: The average power required to switch a wafer at a 10 Mcps repetition rate is approximately 200 milliwatts. The temperature rise above ambient, experimentally measured with a miniature thermocouple having a 3 mil diameter junction bead in contact with a hand wired wafer switched by a 10 Mcps sine wave, is 35°C for a low coercive force material (0.1 oe) and 24°C for a high coercive force material (0.8 oe) for a drive current peak amplitude of 200 mA.

The temperature rise data given above indicates undesirable heating effects. The situation is alleviated by the presence of the large radiating surfaces of the printed conductors in intimate contact with the ferrite in the region of the microapertures. These surfaces are effective in providing the necessary cooling. In addition, the output sense voltages shown in Figure 13 show no deterioration with increased repetition rate up to 10 Mcps. Thus heating is not a serious problem for a high speed ferrite memory.

Electronic Circuitry

Three basic functions are required of the electronic circuitry for a word organized random access store. These are:

a) Word Drivers
b) Digit Drivers
c) Sense Amplifiers

Word Drivers: The word drivers are to deliver the read-write currents to individual words. A diode matrix is normally used to reduce the number of drivers. For a two wire system, a bidirectional diode matrix is needed. For a read-write cycle time of 100 nanoseconds, the word drivers must be capable of delivering externally triggered pulses with a base width of at most 30 nanoseconds at a maximum repetition rate of 10 Mcps. The maximum word length is determined by the breakdown voltage of the output transistor in the drivers. The back voltage developed along a word line is in the range of 300 to 500 millivolts per wafer depending on the magnitude of the switching current. High speed, high current transistors with collector breakdown of 50 volts are commercially available. This implies that word lengths of 25 to 40 bits (50 to 80 wafers per word) are feasible.

The load impedance presented by a selected word may be considered as lumped and is highly resistive. This is valid since the number of bits per word is normally small so that propagation effects in the word direction are

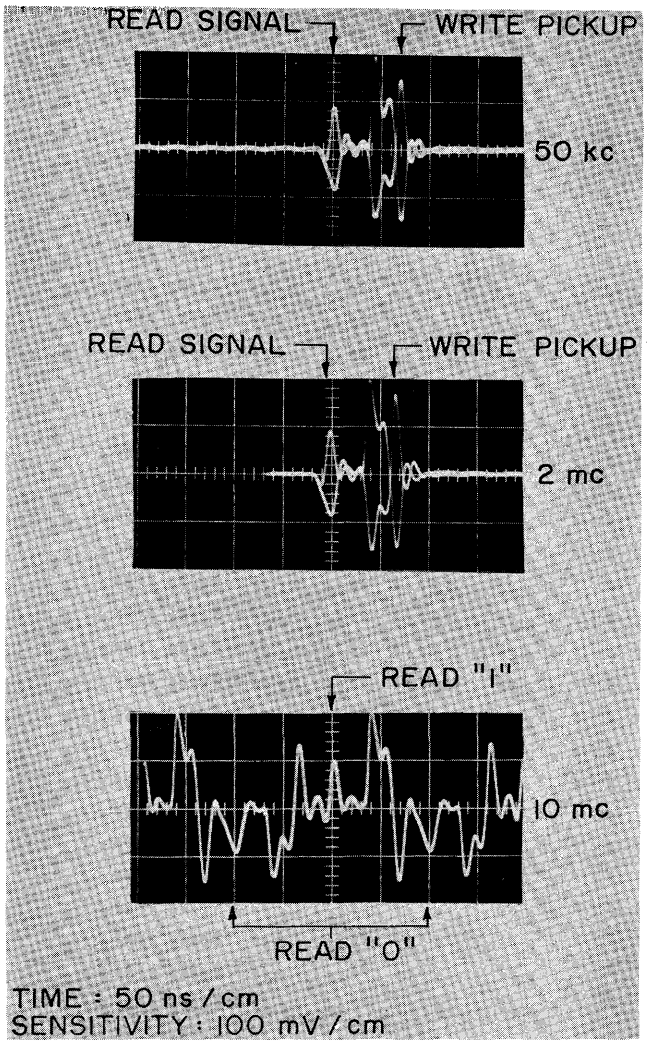


Figure 13. Sense Output Voltages at 0.05, 2, and 10 Mcps (Pulse Program of Figure 12).

negligible. Further, the switching characteristics of a square loop element from an energy point of view are such that it is to a first order approximation equivalent to a resistor. For a two wafer per bit system with bipolar sense output, the load impedance presented by the word line is constant independent of the stored information.

Driver Circuits: The two stage transistor driver shown in Figure 20 was designed, built and tested. Using the RCA developmental type TA2084 transistor (140 V collector breakdown) for the output stage, current pulses of 300 mA into 300 ohms and repetition rates up to 10 Mcps were achieved as shown by the oscillograms in Figure 21.

Positive current pulses with the same characteristics as the negative ones are obtained by modifying the output stage of the circuit of Figure 20 to that of Figure 22. Positive and negative drivers are interconnected in the manner shown in Figure 23 to deliver bipolar pulses into a common load. The two drivers are interconnected via back-biased diodes to permit operation close to the collector break-down, i.e., the voltage transient induced in the load due to the positive pulse is absorbed by the diode in the negative driver circuit and vice versa.

Bidirectional Diode Matrix: A square diode matrix with $\sqrt{N}$ rows and $\sqrt{N}$ columns will drive a memory with a capacity of N words. For a bidirectional matrix the number of unipolar drivers is 4 $\sqrt{N}$. Half of the drivers may be transistor switches, e.g., in Figure 24, Ai and Bi may be drivers and Ci and Di may be switches.

The diodes must have fast reverse recovery, high forward conductance, high reverse voltage breakdown, and low junction capacitance.

The fast recovery is necessary for high speed random access. High forward conductance decreases diode dissipation and allows more of the voltage developed by the driver to overcome the back voltage due to flux switching, i.e., longer words. The reverse breakdown must be approximately the same as the back voltage developed across the load. Low junction capacitance reduces the current drain on the drivers since each Ai or Bi driver (Figure 24) operates into a load consisting of the selected word line in parallel with the capacitance of ($\sqrt{N}$ - 1) back biased diodes. Each Ci or Di driver operates into a load consisting of the selected word line in parallel with (2 $\sqrt{N}$ - 1) back biased diodes.

The RCA developmental type TA1126 diodes, as well as other commercially available diodes have characteristics compatible with the above requirements. The 2 × 2 bidirectional matrix shown in Figure 25 was built and operated at a 10 Mcps repetition rate. Three of the shown loads are 51 ohm resistors and the fourth is effectively a 48 bit (96 wafer) word. (This load consisted of four ferrite plates each having dimensions of 300 × 300 × 10 mils with a 4 × 6 array of electron beam milled aperture clusters. One aperture in each cluster was hand wired to

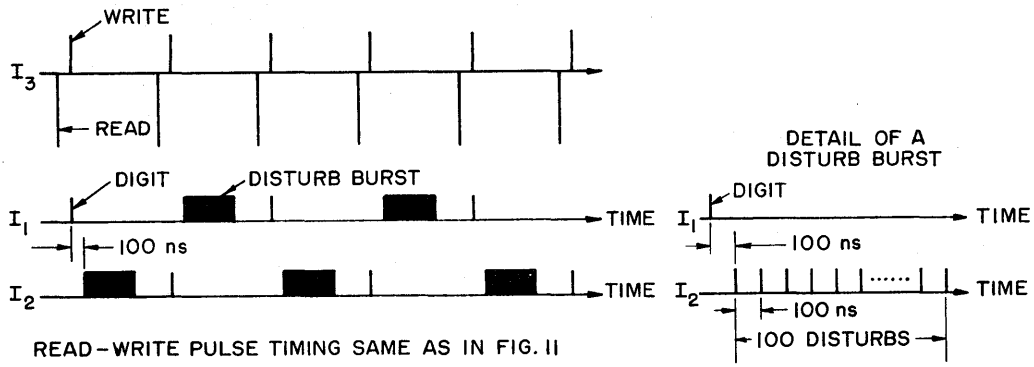


Figure 14. Alternate One and Zero With Disturbs Pulse Program.

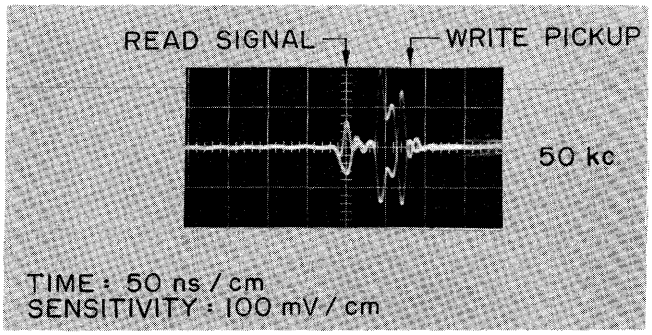


Figure 15. Sense Output Voltages with 10 Mcps Disturbs (Pulse Program of Figure 14).

give a total of 96 apertures connected in series.) In addition capacitors were added to the circuit (C = 120 pF) at the points shown in Figure 25 to simulate the effects of unselected lines. Biasing voltages are supplied to all rows and columns through series R-L networks to maintain all unselected diodes in reverse bias.

For 300 milliamperes through the selected load the total voltage developed across the load (IR drop as well as back voltage due to flux switching) is 30 volts. The capacitive current in an unselected diode is 3 mA, and the current shunted to ground by each R-L bias network is 10 ma.

Digit Drivers: The digit drivers deliver the digit pulses to the digit-sense windings. The lengths of these windings are of the order of ten feet (80 inches for a 1000 word memory) and propagation effects are no longer negligible. The characteristic impedance of a digit-sense line is a function of the geometry of the element winding pattern and the magnetic and dielectric characteristics of the ferrite. Experimental measurements and theoretical calculations indicate a characteristic impedance in the range of 250 to 500 ohms. Thus the digit drivers must deliver current to a relatively high impedance load. The magnitude of this current is 15 to 25 per cent of the write current amplitude leading to voltage requirements in the range of 10 to 25 volts.

The digit pulse normally overlaps the write current pulse. Additional pulse width must be provided to compensate for the delay in the digit pulse due to propagation along the

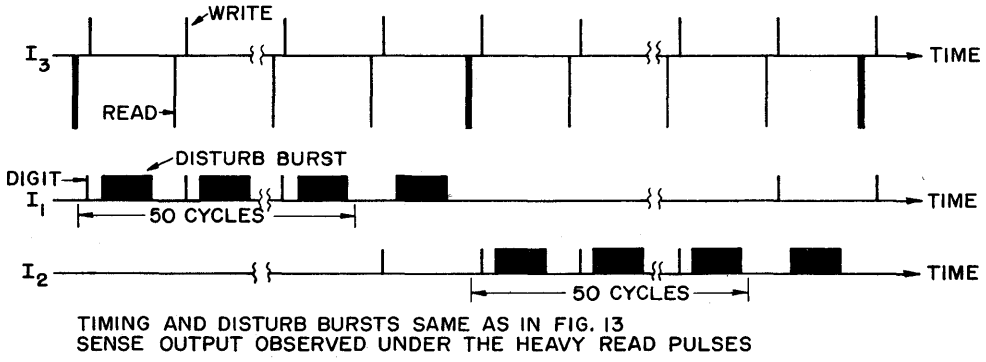


Figure 16. Mixed One and Zero With Disturbs Pulse Program.

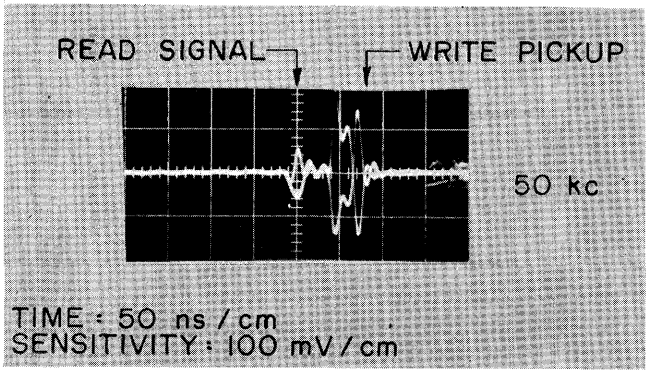


Figure 17. Sense Output Voltages with 10 Mcps Disturbs (Pulse Program of Figure 16).

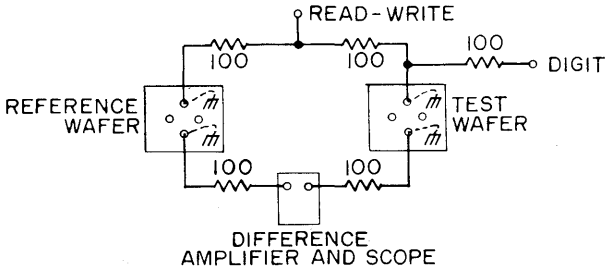


Figure 18. Circuit for Testing Uniformity of Drilled Wafers.

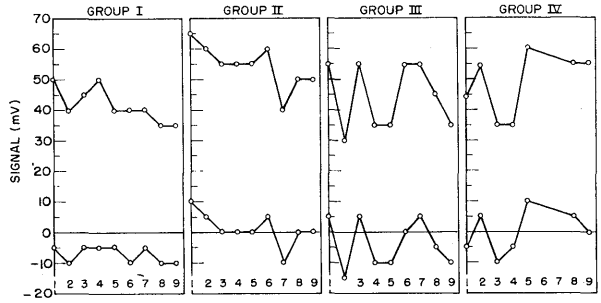


Figure 19. Uniformity of "1" and "0" Outputs of Drilled Wafers.

digit-sense winding to insure coincidence for all words. In the experimental work undertaken the digit and write current pulses are of the same duration and are provided by identical drivers.

Sense Amplifiers:* A two stage transistor amplifier followed by a tunnel diode strobing

*This phase of the work was conducted at the RCA Electronic Data Processing Division in Advanced Development Activity, Pennsauken, N. J., under the direction of A. I. Pressman. A detailed report on this work will be available.

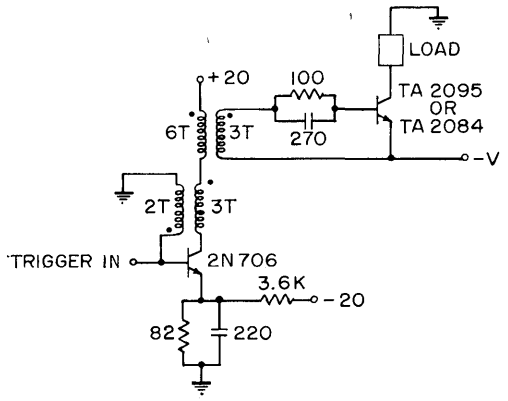


Figure 20. Two Stage Transistor Driver-Negative Polarity Output.

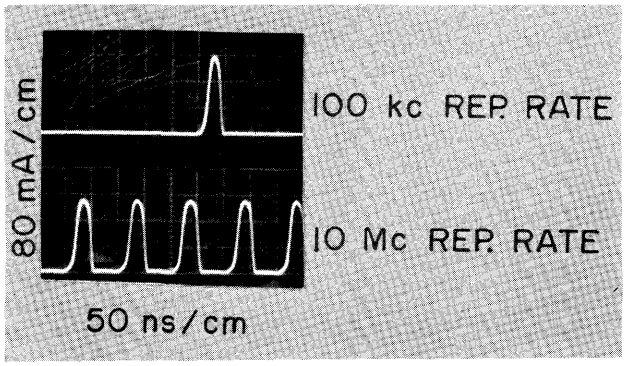


Figure 21. Oscillogram of the Current Output into a 300 ω Load of a Positive Polarity Driver.

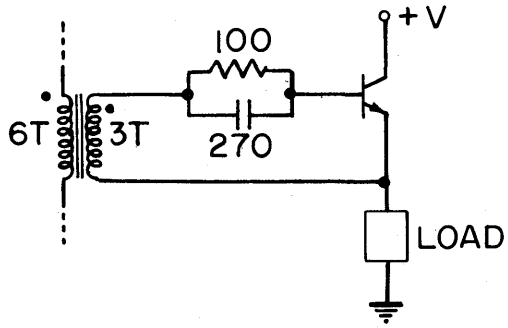


Figure 22. Second Stage of Transistor Driver Connected for Positive Polarity Output.

stage and an additional transistor stage was developed for operation as a differential sense amplifier at high speed. A current limiting diode network is used to interconnect the sense-digit lines and digit drivers to the sense amplifiers. The diode network limits the signal appearing at the input of the sense amplifier to about 1.5 volts during the digit transient.

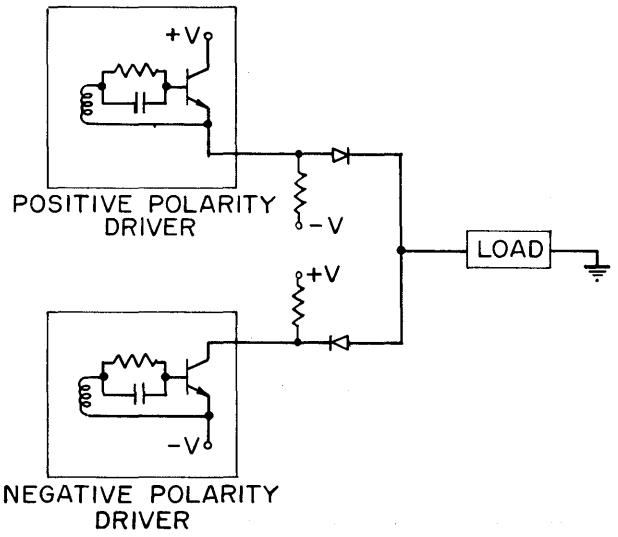


Figure 23. Bi-Directional Driver.

The amplifier is designed to sense a signal of 30 mv, has an output of 5 volts, a delay of less than 10 nanoseconds, and a recovery from the digit transient of less than 40 nanoseconds.

Future Developments

The memory system as described is based on the fabrication of individual wafers assembled into a mosaic. The merits of this approach are:

1. One hundred per cent perfection in fabrication is not necessary and a high yield is adequate.
2. Automatic pretesting of wafers is easily accomplished.
3. Replacement of faulty wafers in a mosaic is possible.

The wafer dimensions of 80 × 80 × 10 mils selected at the beginning of the program represents an educated guess to alleviate fabrication difficulties. The results of the work performed indicate that smaller lateral wafer dimensions are more desirable to reduce the propagation delays along the sense-digit windings and the back voltages induced along the read-write windings without decreasing the sense outputs. Reduction in wafer thickness will reduce the back voltage along the read write windings and the sense output. Measurements on wafers with dimensions of 80 × 30 × 10 mils show a decrease of approximately 30 per cent in back voltage. A new design is being completed based on wafers of 50 × 30 × 10 mil dimensions with the wafers assembled on 50 mil centers in the word direction and 30 mil centers in the digit direction. This will lead to a 30 per cent increase in bits per word and an approximate reduction in the propagation delay by a factor of 2.

Further extensions involving the fabrication of multiple bits per wafer are possible. A modest start is to combine the apertures on two wafers into a single wafer, i.e., use a single wafer with two clusters, each cluster having two apertures in it to store one bit. Experimentally, a separation of 20 mils between clusters leads to sense outputs identical to those shown previously. Figure 26 is a photograph of a "microapertured plate" containing 16 clusters of apertures. The clusters are drilled by mechanically moving the plate under the beam with electronic deflection of the beam used to drill the apertures within a cluster. The feasibility of fabricating such "microapertured plates" depends on the mechanical tolerances that can be maintained in locating the clusters to facilitate the photoetching of the winding patterns.

Cycle Time Considerations

Rajchman [9] has shown that the cycle time T of a memory can be expressed as:

$$T = 2t_s + t_t + t_g + t_a + t_n$$

where t_s is the switching time, t_t the trans-

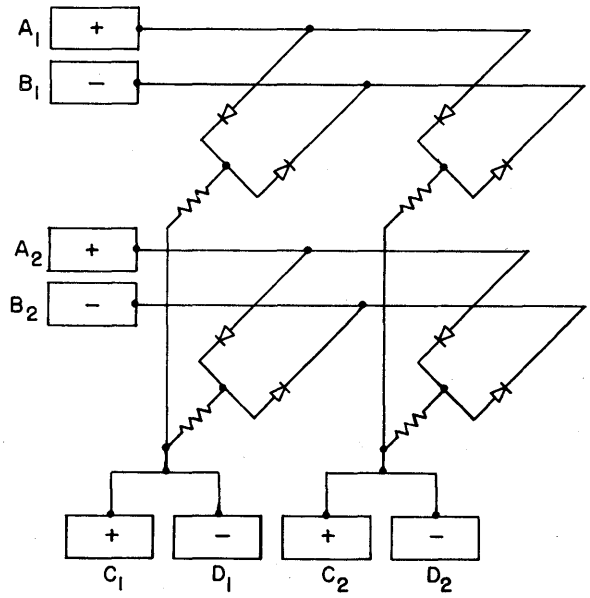


Figure 24. 2 × 2 Bi-Directional Diode Matrix (Simplified Schematic).

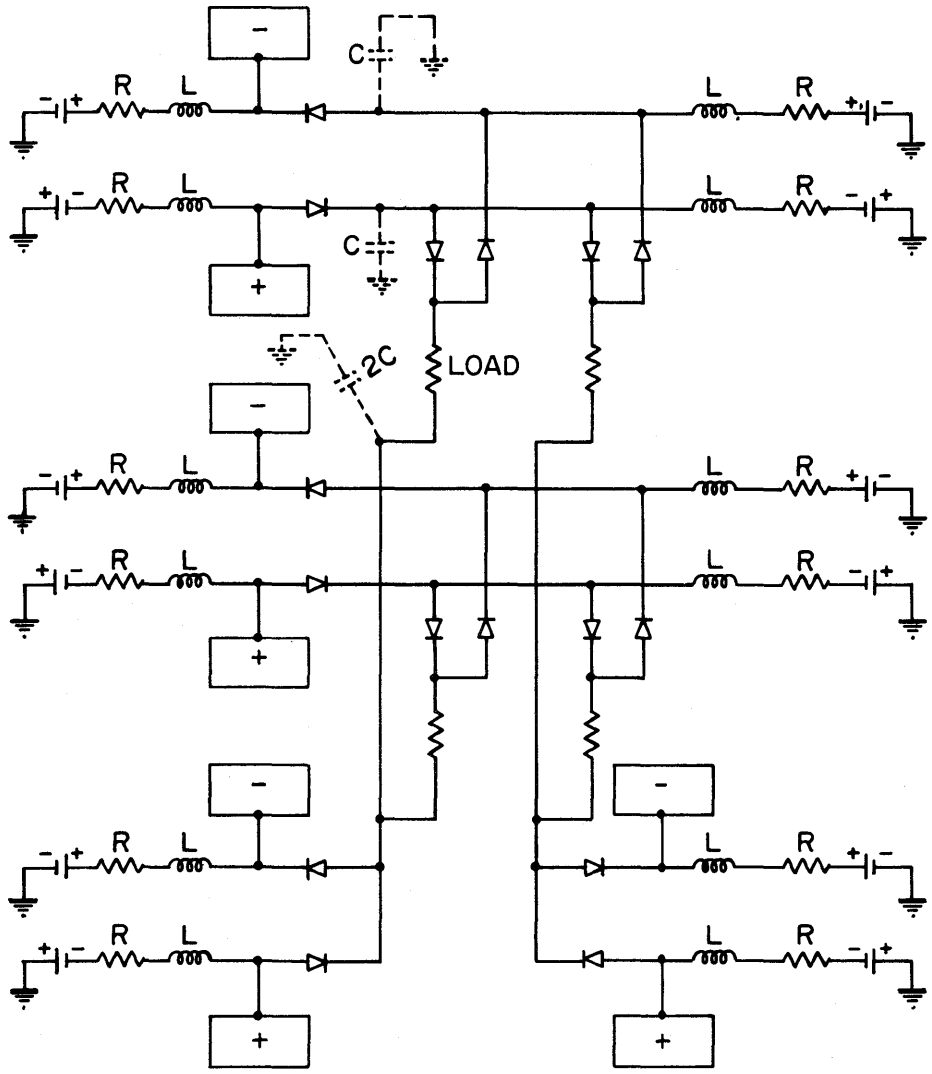


Figure 25. Schematic of Tested 2 × 2 Bi-Directional Diode Matrix.

mission time, t_g the amplification time, t_a the addressing time, and t_n the preread delay time. For the microaperture memory the switching time t_s is 30 nanoseconds. The transmission time t_t along the sense digit winding for the 80 × 80 × 10 mil wafers is 28.3 nanoseconds per 1000 words (assuming a propagation velocity of 1/3 and 1/10 that of light for the paths between clusters and through the clusters, respectively). For the 50 × 30 × 10 mil wafers the transmission time is 15.8 nanoseconds. The amplification time t_g is the sum of the time delays through the sense amplifier and the digit driver. The sense amplifier time delay is 5 nanoseconds and the same value is assumed for the digit driver giving t_g = 10 nanoseconds. For 1000 words the addressing time t_a is the delay through 10 logic levels which is assumed to be 30 nanoseconds. The preread delay in the sense amplifier t_n is 30 nanoseconds. Thus the cycle time for the memory is approximately 150 nanoseconds for a 1000 word capacity.

CONCLUSIONS

The data obtained from the operation of a 4 × 8 mosaic establishes the feasibility of high speed ferrite memories with cycle times of 150 nanoseconds for a 1000 word capacity. A 12 × 16 mosaic is to be tested in the near future. Word lengths of 40 bits are feasible with presently available transistors. Write and digit current requirements are modest and sense output voltages are exceptionally

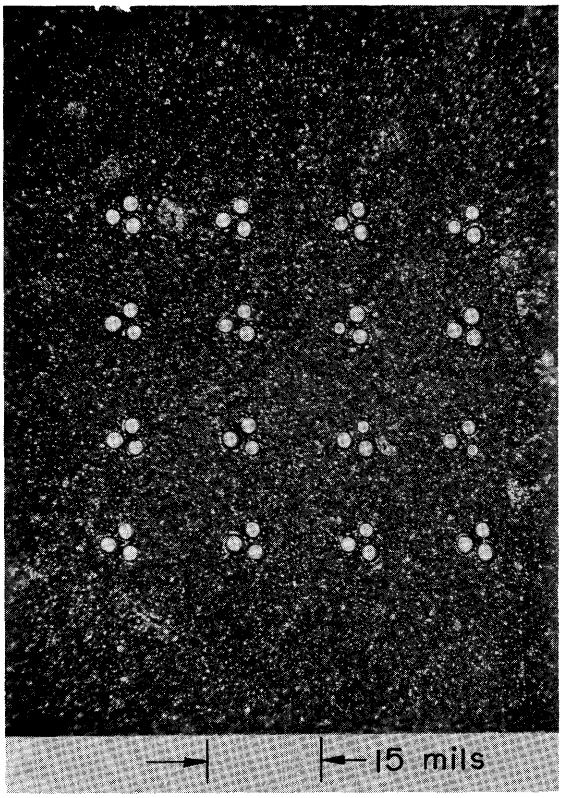


Figure 26. Integrated Microaperture Ferrite Plate.

high. Read current levels are relatively high. Nominally, the read current required should be no greater than the sum of the write and digit currents. The not too well understood effects in the high speed switching characteristics of ferrite required the use of the high read currents. Research in progress in the synthesis of ferrite compositions for high speed application indicates that this is not a fundamental limitation.

The conclusions to be inferred are that microapertured ferrite elements are eminently suited for high speed applications and have advantages that place them in a more favorable position as contrasted to thin magnetic films.

ACKNOWLEDGMENTS

The authors wish to express their gratitude to Dr. J. A. Rajchman, Director, Computer Research Laboratory, RCA Laboratories, under whose direction this work was performed, for the many ideas he contributed, and for his understanding help and guidance.

To their many colleagues at RCA Laboratories and the Operating Divisions of the Corporation who were involved in this program, the authors wish to express their thanks for the help they received. Thanks are also due A. Monsen of the Princeton Laboratories who hand wired most of the samples tested.

REFERENCES

1. J. A. Rajchman, "Computer Memories: A Survey of the State-of-the-Art," Proceedings of the IRE, January, 1961, pp. 104-127.
2. J. A. Rajchman, "Ferrite Apertured Plate for Random Access Memory," Proceedings of the IRE, Vol. 45, pp. 325-334, March, 1957.
3. R. E. McMahon, "Impulse Switching of Ferrites," Digest of Technical Papers, Solid State Circuits Conference, Philadelphia, Pa., pp. 16-17, February, 1959.
4. A. V. Pohm and E. N. Mitchell, "Magnetic Film Memories - A Survey," IRE Trans. on Electronic Computers, Vol. EC-9, pp. 308-314, September, 1960.
5. Raffel, et al., "Magnetic Film Memory Design," Proceedings of the IRE, Vol. 49, pp. 155-164, January, 1961.
6. W. E. Proebster, "The Design of a High-Speed Thin Magnetic Film Memory," 1962 International Solid State Circuits Conference, Philadelphia, Pa., February, 1962.
7. N. Menyuk and J. B. Goodenough, "Magnetic Materials for Digital Computer Components," J. Appl. Physics, Vol. 26, No. 1, p. 8 (1955).
8. W. Lee Shevel, Jr., "Millimicrosecond Switching Properties of Ferrite Computer Elements," J. Appl. Physics, Supplement to Vol. 30, No. 2, p. 475 (1959).
9. J. A. Rajchman, "Computer Memories - Possible Future Developments," RCA Review, 23, No. 2, pp. 137, June, 1962.
10. V. T. Shahan and O. A. Gutwin, "Threshold Properties of Partially Switched Ferrite Cores," J. Appl. Physics, Supplement to Vol. 33, No. 3, p. 1049 (1962).

MAGNETIC FILMS - REVOLUTION IN COMPUTER MEMORIES

C. Chong and G. Fedde
UNIVAC Division of Sperry Rand Corporation
Blue Bell, Pennsylvania

SUMMARY

The development and application of magnetic thin films has been persued vigorously by a number of workers. Some of the practical results are tabulated seven years after the first reported work.

Advances in ferrite memory technology and difficulties in fabrication of film memories have delayed the widespread use of film memories. However, a number of experimental memory systems have demonstrated significant achievements in packing density, 200-900 bits per square inch; read-write cycle time of 100-200 nsec.; reliable nondestructive read out in an electrically alterable memory; in the use of high performance circuits and construction techniques giving an economical memory system.

Several film memories are in production or pilot production at the UNIVAC Division of Sperry Rand Corp. These are the small 128 word 1/2 microsecond film memory used in the "UNIVAC 1107" and the large 6656 word, nondestructive read out, electrically alterable film memory developed for the ADD (Aerospace Digital Development) Computer. Burroughs Corp. is using a small film memory in a military computer and is marketing a memory plane. Texas Instruments announced immediate availability of a memory plane pair at the August 1962 WESCON.

Magnetic thin films differ from ferrite memory elements in two important ways. The nickel-iron alloy used for thin films is deposited under conditions that cause a magnetic anisotropy and the film thinness and geometry is chosen to permit coherent rotational switching. The magnetic anisotropy makes possible the simple operating mode that is widely used, in which coherent rotational switching of the magnetization can be completed in several nanoseconds.

Film memory elements are usually operated with open instead of closed flux paths permitting simple wiring with printed circuits. The film elements are made in arrays of large numbers of bits or in continuous processes and can be economically tested. Large volume production and testing may be able to produce film memory bits for about one-tenth of a cent per bit.

Thin films rate highly in the characteristics required of an ideal memory element. They can be utilized economically, packaged very densely for severe environments, operated with low power, can be used in very fast memories, and operated in either a destructive or nondestructive read out mode.

All of the film memories disclosed to date use a word organized or linear select method of operation. Studies with feasibility models have shown that it should be possible to build large capacity magnetic film memory systems that are competitive with other systems.

Magnetic thin films have been successfully deposited on silver, aluminum, glass, copper and gold, by various workers, making great design flexibility possible. Present

research may result in higher bit density and low current operation that is compatible with integrated and molecular circuits. Further improvements in film properties and manufacturing uniformity may make coincident current selection practical while achieving coherent rotation. The combined impact of low cost, high speed, wide temperature range, low power and simple NDRO operation should result in a revolution in computer memories.

INTRODUCTION

Brief History

The potential usefulness of magnetic thin films has attracted a great deal of research and development effort. To the many people actively seeking practical applications, it is a little like chasing an attractive mirage which continually recedes as one approaches. The significant and steady improvements in ferrite memories and the practical problems of magnetic film elements are about equally responsible for this characteristic. Taking some risk of being premature, it is thought that a revolution in computer memories is now underway and that the extent and impact of that revolution will expand very rapidly. To be more specific, the change from individually formed, tested and wired elements to the deposition of large multi-bit arrays that are tested and laminated with their wiring as arrays, is a major advantage. At the same time it is troublesome because of the close control of uniformity that is necessary. In addition, the switching time of the magnetic film memory element is no longer of major significance in determining the memory cycle time or access time.

Since 1955 [1] when the first thin magnetic film work was reported, until about 1960, practical applications were almost nonexistant. In the last two years, relatively small, moderately fast thin film memories have been put to practical use at Lincoln Labs [2] in the TX-2 and FX-1 Computers and in the commercially available UNIVAC 1107 [3]. Outstanding because of its large size, 166,000 bits, the thin film memory used in the UNIVAC ADD (Aerospace Digital Development) Computer is relatively slow, but represents a milestone in thin film technology [4]. Its application of non-destructive read-out in an electronically alterable memory may be the cornerstone necessary to achieve the long sought breakthrough in practical application of film memories, although its physical realization may change in future designs.

Achievements of Some Film Memories

A number of experimental magnetic film memory systems have been built and their achievements reported. See Figures 1 and 2. This list is not intended to be comprehensive

COMPANY	LINCOLN LAB M. I. T.	IBM	NATIONAL CASH REGISTER	I. C. T.
IDENTIFICATION	FX-1 COMPUTER[2]	W. E. PROEBSTER[14]	D. A. MEIER[7]	E. M. BRADLEY[15]
DATE	FEB. 1962	FEB. 1962	MAY 1961	AUG. 1962
NO. OF WORDS	256	256	128	4096
NO. OF BITS/WORD	13	72	8	
ACCESS TIME		60 ns		
CYCLE TIME	370 ns	100 ns	200 ns	1000 ns
FILM	LONG RECTANGLE	RECTANGLES	CONTINUOUS CYLINDERS	CONTINUOUS FILM
SUBSTRATE	GLASS	SILVER, SiO	WIRE	ALUMINUM
DEPOSITION	VACUUM	VACUUM	ELECTROPLATED	VACUUM
NO. OF PLANES	1	1		
OPERATING MODE	DRO	DRO	DRO	DRO
ORGANIZATION	WORD SELECT	WORD SELECT	WORD SELECT	WORD SELECT
COMMENTS	OPERATING IN A COMPUTER	BASED ON INITIAL TESTS		

7137b

Figure 1. Achievements of some experimental thin film memories.

IDENTIFICATION	ULTRA HIGH[6] SPEED MEMORY	FEASIBILITY MODEL	SEARCH[6]
DATE	AUG. 1962	JUNE 1962	JUNE 1962
NO. OF WORDS	1024	512	128
NO. OF BITS/WORD	24	32	24
ACCESS TIME	50 ns	200 ns	100 ns SEARCH
CYCLE TIME	100 ns	250 ns	100 ns SEARCH
FILMS	PAIRED RECTANGLES	RECTANGLES	BICORE CIRCLES
SUBSTRATE	GLASS	GLASS	GLASS
DEPOSITION	VACUUM	ELECTROPLATED	VACUUM
NO. OF PLANES	4	8	1
OPERATING MODE	DRO	DRO	SEARCH
ORGANIZATION	WORD SELECT	WORD SELECT	SEARCH
COMMENTS	OPERATING IN TEST MACHINE	BEING EXPANDED TO 4096 WORDS	7137a

Figure 2. Achievements of some UNIVAC experimental thin film memories.

and no particular inference is to be drawn from the omission of other film memory work. It is both interesting and worthwhile to examine these results and the problems that have been solved. A list of the principal problems would certainly include the following:

1. Magnetic film uniformity and reproducibility.
2. Connections to and interconnections between the magnetic film memory elements.
3. Memory packaging and organization suitable for large, fast memory systems.

The first problem has apparently been adequately solved by five or six companies. The second problem is probably the most vexing and most likely to resist a completely satisfactory solution. A technique, of limited usefulness, to minimize the interconnection problem is to place all film spots in a single plane. This technique seems to place a practical limit on the number of bits in a conveniently packaged memory of high speed.

The third problem has been partly solved by three companies, judging by hardware that is available. See Figure 3. One of these companies has announced memory planes for sale that use connectors similar to standard printed circuit edge connectors. These memory planes, or similar ones, are used in a military computer [5] built by the company. A second company announced the availability of memory planes at WESCON this year. The other company has two different memories in production or pilot production [6]. Several types of film memories are available as production items or, depending on the exact application, can be produced with a small amount of engineering design and integration [6]. These include destructive read-out and nondestructive read-out. One of these memories is a 128 word, 36 bit DRO memory capable of a 300 nsec. cycle time. A second memory consists of 6656 24-bit words of NDRO and 256 24-bit words of DRO with a 0.7 microsecond access time and a 3 microsecond cycle time. The NDRO memory is electrically alterable and is similar to the memory in the ADD Computer mentioned earlier.

One experimental thin film memory [7] uses a different alloy 97% Fe - 3% Ni and

COMPANY	UNIVAC	BURROUGHS	UNIVAC	TEXAS INSTRUMENTS
IDENTIFICATION	[3]	MEMORY PLANE	ADD[4]	MEMORY PLANE PAIR
NO. OF WORDS	128	128	6656	64
NO. OF BITS/WORD	36	24	24	18
ACCESS TIME	330ns		700ns	
CYCLE TIME	670ns	200ns *	3000ns	200ns
FILMS	CIRCLES	RECTANGLES	BICORE CIRCLES	CONTINUOUS
SUBSTRATE	GLASS	GLASS	GLASS	ALUMINUM
DEPOSITION	VACUUM	VACUUM	VACUUM	VACUUM
OPERATING MODE	DRO	DRO	NDRO	DRO
ORGANIZATION	WORD SELECT	WORD SELECT	WORD SELECT	WORD SELECT
COMMENTS	USED IN 1107 COMPUTER	*OPERATING SPEED OF WIRED PLANES IS STATED TO BE 5mc.	MEMORY FOR ADD COMPUTER TESTED FOR AEROSPACE APPLICATION.	IMMEDIATELY AVAILABLE
				7135

Figure 3. Characteristics of available thin film memories and memory planes.

does not use a magnetic anisotropy common to the other magnetic film work mentioned here. It closely resembles an advanced ferrite memory in that domain wall motion is used and the switching constant of 0.6 to 0.7 oersted-microseconds is approximately the same as for ferrites. Fast coincident current switching is obtained by using a high coercive force material (14 oersteds). The core wiring problem is exchanged for a multiple solenoid winding problem although the connection problem is basically the same as for a ferrite memory stack.

The UNIVAC Division has built a 512 word, 32 bit per word feasibility model thin film memory with a read-write cycle time of 250 nanoseconds. This model is, of course, based on much previous work. This feasibility model demonstrated the praticality of a large thin film memory for commercial computer usage. Some of the circuits and techniques developed by this work are discussed later.

UNIVAC has developed several other significant thin film memory systems under sponsorship of a government agency. These include a 10 mc. DRO memory of 1024 words and 24 bits, and a 100 nanosecond search memory of 128 24-bit words.

A recently announced microferrite memory is an extremely interesting extension of ferrite memory elements into the speed and performance range of magnetic film memories. Information available at the time of this writing is somewhat incomplete, but it appears that manufacturing, testing and wiring techniques have been developed that are suitable for economical production. It is almost certainly true that the relative useage of any of the available memory elements will be decided by balancing economic and performance factors.

COMPARISON OF MAGNETIC FILMS TO FERRITE CORES

Anisotropy: Perhaps the most significant difference between magnetic thin films and ferrite cores for example, is that the thin films are made to have a uni-axial magnetic anisotropy [8]. This is just another way of saying that the direction of magnetization will always be parallel to the preferred, or easy axis, unless an external force acts upon it. It is this anisotropy that makes possible a simple nondestructive read-out mode for magnetic films [9]. The anisotropy is usually "built-in" to the films by the presence of an external orienting magnetic field during deposition whether by electroplating or vacuum evaporation or sputtering of the nickel-iron alloy. Several other factors can cause or effect the magnetic anisotropy, strain [10], for example, but these effects are discussed in the references and won't be discussed further. The presence of a preferred or easy axis of magnetization means that the effect of an external field on the magnetization of the film will be dependent upon the angle between the field and the easy axis. This angular dependence provides an extra degree of freedom in the organization of a memory not widely used in ferrite memories [11, 12].

Coherent Rotation: The very fast switching speed of thin films is a result of the simultaneous rotation of the axes of all the electron spins rather than the sequential reversal of the slower domain wall motion switch. The thinness of the films plays an important role in achieving an ultra fast switching time. More complete discussions of the theory and behavior of thin magnetic films may be found in references 8, 11, 13, 14, 15, and 16.

For practical circuits, it is reasonably accurate to say that switching speed is determined only by the rise time of the drive fields. Films have been observed switching in a few nanoseconds or less in special experimental apparatus [17]. Such speeds cannot be practically utilized in any but the smallest memories. Memory systems of several thousand words exhibit so much delay in their drive and sense wiring that their cycle time is determined by physical size and transmission time and not by switching time.

Comparison to Coincident Current Selection: Nondestructive read-out has been achieved with magnetic films in a variety of ways [4, 9, 10]. If we restrict our attention to electrically alterable memories, we can make some interesting comparisons to coincident current selection memories. Assume a 4096 word, 32 bit per word coincident selection memory. This would probably result in 32 planes of 4096 bits (64×64). If we assume that the X-Y drivers are arranged in 8×8 matrices, we need 32 write drivers, 32 read drivers and 32 bit or inhibit drivers and 32 sense amplifiers. If we now use an NDRO film element in a word organized memory, we might choose 1024 words each

of 128 bits. These are, of course, quadruple length but only one set of 32 bits would be gated to the 32 read amplifiers at one time. The other bits on the selected word line would not have their information destroyed and no recirculation path is needed. Thus, a 32×32 matrix of drivers with 1024 diodes will provide the main read-write drive current. Only one driver is needed for each line of the 32×32 matrix since only one direction of drive current is needed in each word line. This organization requires the same number of drivers, read amplifiers and bit drivers as the coincident selection memory. The film memory would have approximately 800 extra diodes.

All of the film memories reported on to date have used word organization since the requirements for film uniformity are less stringent. Films are made with more than adequate squareness in their easy axis hysteresis loop so that operation is possible in the normal coincident current mode. A slower domain wall motion switch is obtained if the film is used that way so most current work is not going in this direction. Film technology is at or very close to the point where film uniformity is good enough so that coincident current selection with a coherent rotational switching mode may be feasible in the near future.

CHARACTERISTICS OF AN IDEAL MEMORY ELEMENT

The choice of an element for a memory, and the organization thereof, requires sound engineering judgement and a full understanding of the applications and requirements of the memory. For instance, a main store for a large scale commercial computer probably will emphasize large capacity and low cost with fair reliability and speed. A store for a space vehicle will probably emphasize reliability, low power and small physical size. An ideal memory element would fill the requirements for all memories. The thin film, like other memory elements, is not an ideal element, but, in some respects, approaches one. Figure 4 is a condensed chart of some of the characteristics of an ideal element and those of thin films.

Tiny Size: Small size is one of the most important characteristics of an ideal memory element. Lower power and fast access and cycle times are all related to the size of the elements. An ideal element should be as small as can be accommodated by wiring and connection techniques.

Present film memories have elements typically 200 to 1000 to the square inch. Advanced work, however, is trying to increase this density significantly.

Low Power: Low drive power in a memory is extremely important for spacecraft applications. It is also important in other applications as low drive power means less amplification of the sensed output of the element and consequently less delay. Ideally, the drive power would be the same order of magnitude as the output power.

Interrogating currents at present are in the order of 300-800 milliamperes and drive

	HIGHLY DESIRABLE	THIN FILM
PHYSICAL SIZE	AS SMALL AS CAN BE ACCOMODATED	200 – 900 PER SQ. IN.
DRIVE CURRENT DRIVE POWER	COMPARABLE TO OUTPUT POWER	300 – 800 MA DRIVE LINE IMPEDANCE 5 – 30 OHMS
SWITCHING TIME	SMALL COMPARED TO SIGNAL DELAY	NEGLIGIBLE
OUTPUT SIGNAL	MODEST SIGNAL TENS OF MV	1 MV – 100 MV
OUTPUT SIGNAL/NOISE	S/N → ∞	4 – 10
READ OUT	DRO OR NDRO ELECTRICALLY ALTERABLE	BOTH MODES ARE USED IN EXPERIMENTAL AND PILOT PRODUCTION FILM MEMORIES
OPERATING MODE	CAPABLE OF COINCIDENT CURRENT SELECTION	WORD ORGANIZED - COHERENT ROTATION COINCIDENT SELECTION - DOMAIN WALL MOTION
ENVIRONMENT	NOT A LIMITING FACTOR	LARGE MEMORIES DEMONSTRATED -40°C → +80°C, OPERABLE UNDER SHOCK AND VIBRATION
COST	EASILY FABRICATED, COST < $0.01 EACH	SHOWS GREAT PROMISE, COST NOT PROVED IN VOLUME PRODUCTION

7136

Figure 4. Summary of memory element characteristics.

line impedances are in the 5-30 ohm range. As the size of the film and its drive lines get smaller, the drive current may also become smaller. Interrogating currents may be as small as 50 milliampere in the future.

Switching Time: The switching time of an ideal element should be small compared to transmission and amplification delays. Magnetic film switching time is several nanoseconds or less so that it is negligible in all but the smallest memories.

High Output - Fast Switching Speed: A high output from a memory element allows easy sensing and less amplification of the signal and hence, less delay. An ideal element should have an output in the tens of millivolts. The larger the output, the larger the energy required to switch the element so that outputs in the volt range would be awkward for a large memory.

The outputs of present film memory elements are in the order of 1 to 10 millivolts and of 50 to 10 nsecs. duration. Just as important as a large signal is a large ratio of desired signal to "noise." With careful design a ratio of 10 to 1 has been achieved in large film memories [18].

Simple NDRO and Coincident Current Selection: Simple nondestructive read-out and coincident current selection are very desirable features of a memory element. NDRO eliminates the need for rewriting, resulting in shorter cycle time and less drive power. Coincident current selection reduces the cost and complexity of the drive circuitry. Ideally the same circuits and drivers will be used for reading and writing with the addition of a small bit current to write information.

Theoretically, the threshold characteristics of the magnetic thin film allows both simple NDRO and coincident current selection. Practically, the uniformity of the present day films is such that many of the memories built to date are DRO and word organized. NDRO has been achieved by the BICORE* type of construction used in the UNIVAC ADD Computer [4] and in the cylindrical construction developed by T. R. Long at Bell Labs [9].

Insensitivity to Temperature Variations: Both external conditions and internal power dissipation may cause temperature variations in a memory. It is then important for a memory element to be as temperature insensitive as possible.

The Curie Point Temperature (550°C) of the permalloy film is sufficiently high that the film is practically temperature insensitive. Individual films have been tested from -80°C to 100°C while the complete ADD memory has operated in ambients from -40°C to 80°C [4, 6].

Ease of Fabrication of Element and Memory: In order to make a large capacity memory practical, it is important that the elements can be fabricated and tested in a batch or continuous process. It is further necessary that the drive and sense lines can be prepared without high cost. The ideal memory element will not require any critical alignment to its associated wiring.

The next section will present several techniques, circuit designs and fabrication methods developed in our attempt to solve the practical application problems of a 4096, 50 bit word thin film memory for computer usage.

TECHNIQUES USEFUL FOR A LARGE MAGNETIC FILM MEMORY

Organization: As mentioned previously, UNIVAC has built a 512, 32 bit word thin film memory feasibility model. This is part of a program whose objective is to show that a large, fast film memory can be built that has a competitive advantage over other memory systems. A simple destructive read-out operation is used because component variation and drive current tolerances may be relatively wide. A single polarity word drive pulse provides the read-out signal during its rise time and a digit drive field, of appropriate polarity, overlapping the trailing edge of the word pulse restores the film to its proper information state. For a more complete description of the operating mode, the references should be consulted [13, 16].

Fabrication of the Magnetic Thin Films: Electroplating was chosen as the fabrication method for many reasons. Two of these are: (1) the apparent simplicity and low cost of the basic apparatus, (2) a room environment process should make possible a relatively simple automatic or continuous fabrication method.

The electrolyte is a solution containing nickel and iron sulfates and was approximately 0.8 molar in nickel sulfate. A number of additives were present including boric acid, sodium chloride, sodium laurly sulfate and

*Trademark of UNIVAC Division of Sperry Rand Corporation.

saccharin. Phosphorus was introduced into the final alloy that was electroplated by adding sodium hypophosphite to the solution. A more complete description may be found in the references [19].

The plating cell is a simple rectangular vessel inside of a large coil which provides the orienting magnetic field to determine the anisotropy axis. The plating cell geometry has a pronounced effect on film uniformity. The most favorable plating current density seems to be 6 ma per square centimeter and the plating was done at room temperature in a non-agitated electrolyte. The thickness of the deposited film is easily controlled by the amount of charge transferred in the plating cell.

Present work uses electroplated nickel-iron films on metal substrates instead of the glass substrates used in the feasibility model. This is one of several design improvements that is hoped will decrease the cycle time of a much larger memory to 200 nanoseconds, 50 nanoseconds less than the feasibility model.

Digit Write Circuit: The memory is word organized with three separate transmission lines in the memory stack for the sense circuit, the word drive circuit, and the information write circuit. If 4096 film spots (bit number one of all words for example) were arranged 20 per inch, the bit line would be about 20 feet long (allowing 20% for interplane connections and packing inefficiency). Such a line would have a transmission delay of 50-60 nsec. This is much larger than can be tolerated for a 200 nsec. memory cycle time. One practical solution is to use two to four sections connected in parallel rather than reduce the spot size, decrease the output signal, and increase the packing density by the same factor. Typical films can be written with a bit current of 60 milliamperes from a 0.050 inch wide bit line, so the total current required from the bit driver will be two to four times this current. A single driver transistor can easily supply this current. Figure 5 shows an early circuit that was developed to deliver positive and negative

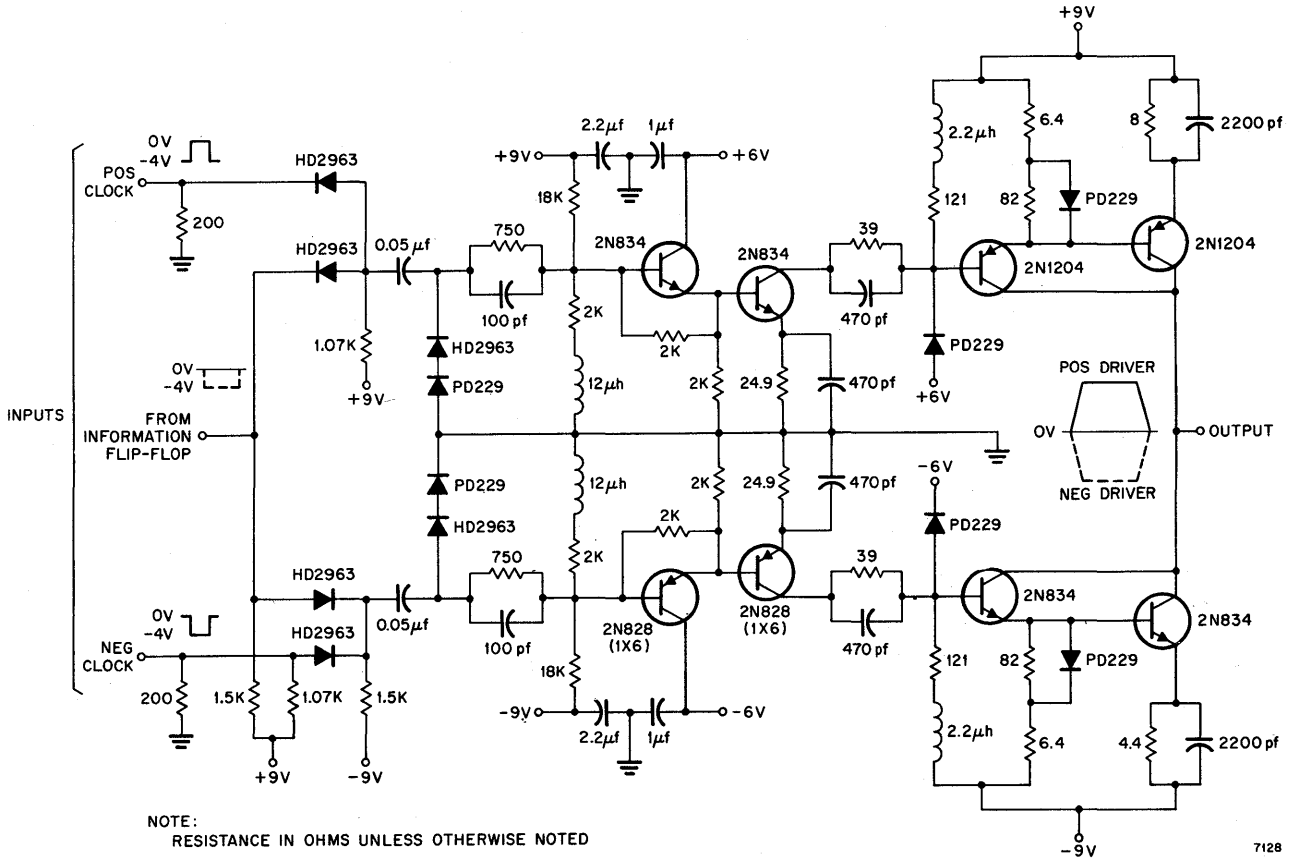


Figure 5. Schematic - digit driver.

pulses of 760 milliamperes with a delay time, rise time and fall time of 10, 15, and 20 nsecs., respectively. For the lower current requirement the paralleled output stages are removed.

Sensing Circuit: Fragmentation of the sense lines, just as that of the digit lines, is used to reduce the delay for the film signals to travel out to the sense amplifier. A means to accomplish the fragmentation of the sense line is shown in Figure 6. Here a fragmentation into eight sections is shown. If the

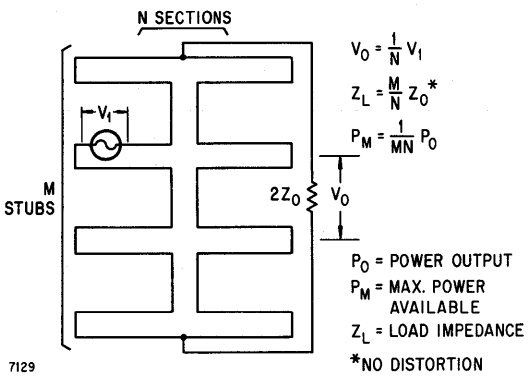


Figure 6. Possible arrangement of sense lines.

delays and characteristics impedances of the line stubs are essentially the same, and if the load to the system is twice the characteristic impedance of a line stub, there will be no distortions. That is, the output has the same shape as one that has come out of a single line. The delay is, of course, 1/8th that of a signal traveling down a long line. This scheme can be extended to the general case of paralleling N sets of M stubs. The terminating impedance is then M/N times the characteristic impedance of a single stub. If lossless lines could be used, the ratio of output signal power to total available signal power from one line is 1/MN. The output voltage is 1/N times the voltage from a single terminated line.

The sense amplifier is the most difficult circuit in the film memory. It accounts for a significant portion of the cycle time. The signal out of a film element is high enough so that sensing it is not difficult. However, the common mode noise on the balanced sense line caused by the word current changing and the uncommon mode noise caused by the digit current changing could be over 1000 times and 10 times the signal, respectively. Rejection of the former and rapid recovery from the latter are essential to the operation of the sense amplifier. The conventional approach for rejecting the common mode noise is to carry the signal in two separate channels followed by a difference amplifier. This approach is costly in transistor count and poor in performance. A better approach is used in the circuit shown in Figure 7. In this circuit, a three-core magnetic common mode rejector (CMR) is used. The common mode signals are attenuated by the high impedance; while the film signal, which is differential, is passed by the transmission line and is not attenuated. This amplifier further uses emitter-gating in the third stage to gate out the differential noise caused by the digit drive. The following is the performance of the circuit:

	Input (15 ns rise time)
1. Output (5 ns rise time):	
Channel A 4V at 12 ma:	0.75 mv min.
Channel B 4V at 12 ma:	-0.75 mv min.
2. Common mode rejection:	60 db min.
3. Gain-gate on/Gain-gate off:	200/1
4. Delay through amplifier:	15 ns
5. Recovery from digit transient of 30 mv (saturation is in the first 2 stages):	50 ns
6. Gate required:	4v, 60 ns wide
7. Gate transient:	0.4v max for 5 ns at output

Typical signals obtained from the 512, 32 bit word memory feasibility model are shown in Figure 8.

Word Lines and Selection Circuits: The word line is next closest to the magnetic film and slit to reduce eddy current damping of the film switching. At the same time the changing field of the digit line can more easily be coupled to the film. Of critical importance to the design of the word lines and their spacing is the amount of drive field that is present at the adjacent, non-selected words. Digit write currents which are not capable of destroying previously stored information by themselves, can cause a loss of information after many thousands of pulses if a transverse field of approximately 5-10% of H_K is present. Since the field generated by

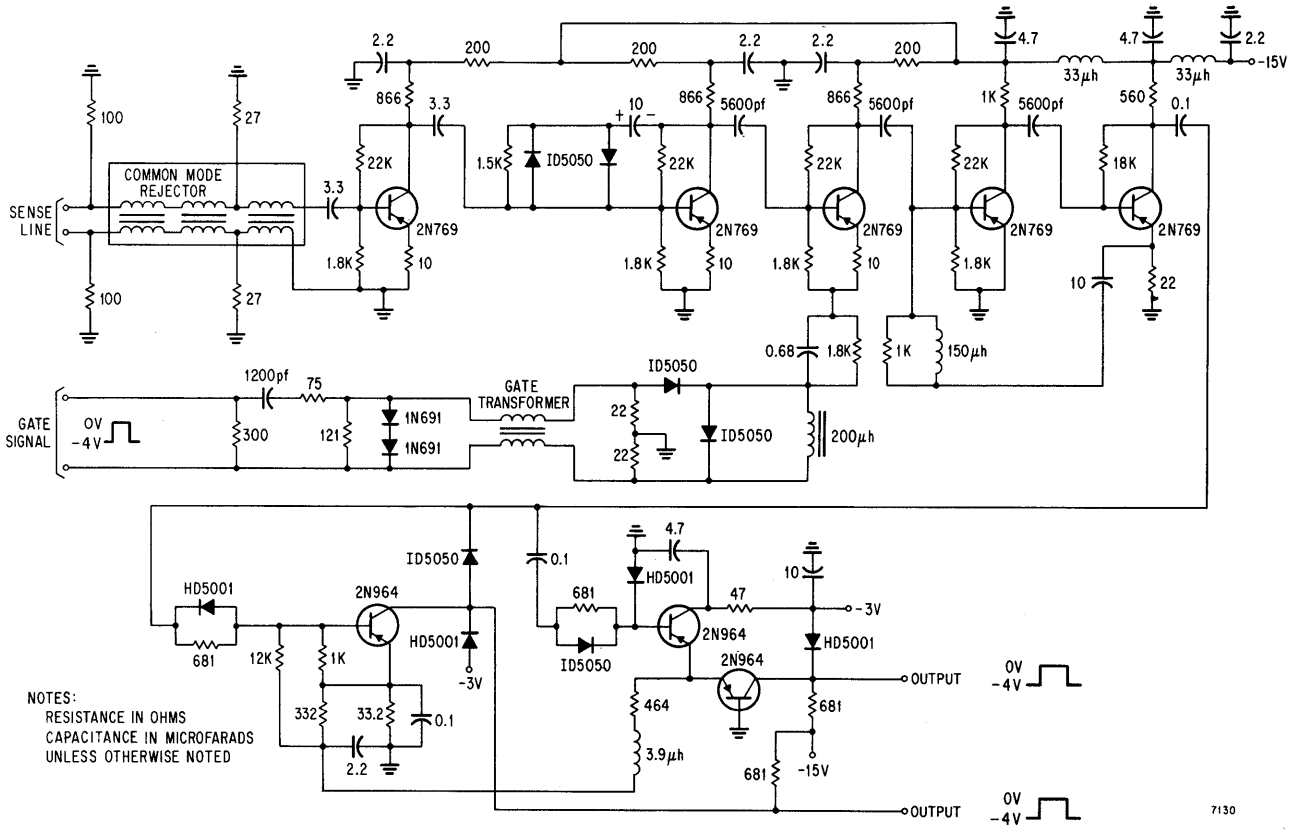


Figure 7. Schematic - sense amplifier.

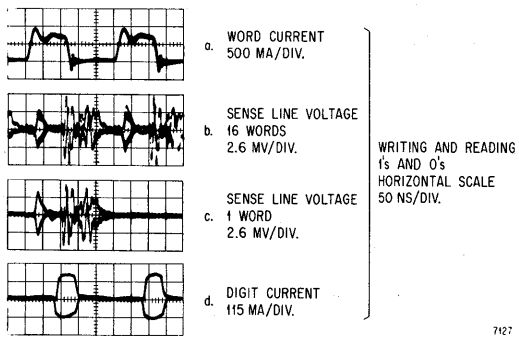


Figure 8. Signals of UNIVAC 512 word thin film memory feasibility model.

the selected word line is usually about twice H_K at the intended film spots, then the spacing between the word line and its return should be about 1/8th to 1/10th of the distance from the edge of the adjacent spot to the edge of the word line to attenuate the word drive field to 2.5-5% of its value.

The choice of the word selection circuit is greatly dependent on the size of the memory. In a small memory (say up to a few hundred words) a straight forward approach of one driver per line with high level matrix decoding may be used. A system like this has the merits of its simplicity and ease of design. However, in a larger system (4096 words), the number of drivers required makes this scheme impractical. A diode matrix scheme may be used. For a 4096 word memory, 4096 diodes, 64 X drivers and 64 Y switches are required. Description of a typical diode matrix may be found in the references [13, 18].

A matrix system has several problems, one of which is the capacitive load on each Y switch. This capacitive load is comprised mostly of the capacity through the digit lines to ground of the 64 word lines connected to each of the Y switches and may be as much as 10,000 picofarads. A circuit has been developed and tested that can charge 10,000 picofarads in 80 nsecs. with peak current of 1.8 amps. The maximum repetition rate is 4 mc. The X driver drives current down the selected word line. Figure 9 shows a circuit that can drive up to one ampere with

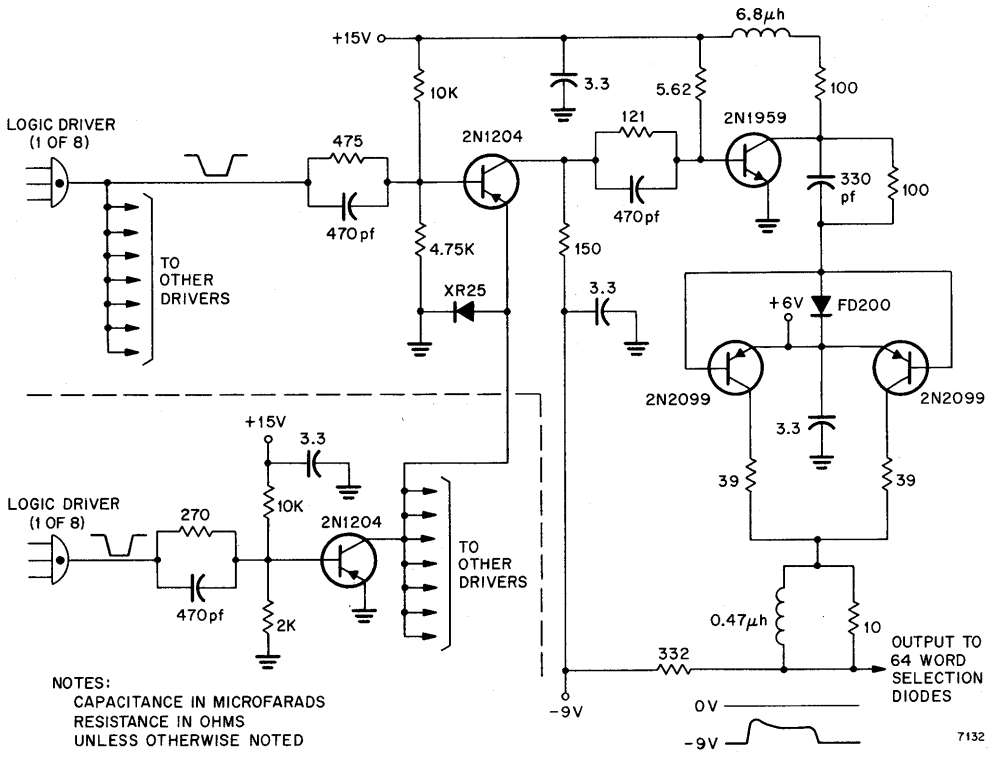


Figure 9. Matrix driver.

rise and fall time of 30 nsec. each. These circuits can select word lines at a rate of 4 megacycles per second.

Figure 10 shows a sketch of the three, closely spaced transmission lines that give access to the film spots. This circuit board was fabricated to avoid the necessity of handling very thin and flimsy sheets of insulating material supporting precision etched circuits. The first layer of wiring is etched and then coated with an insulating epoxy. A sheet of copper is then bonded to the circuit and the second layer etched. The process is repeated for the third layer. One set of index holes in the substantial base serves for registration of all layers. With this technique it has been possible to place the layers of circuits within 1 mil of the previous layer. Registration between layers depends on the original registration of the precision masters and upon the dimensional stability of the base. At no time in the fabrication is it necessary to perform registration of individual circuits to each other.

Figure 11 shows a sketch of the etched circuit low impedance backboard power distribution lines. The 512 word feasibility

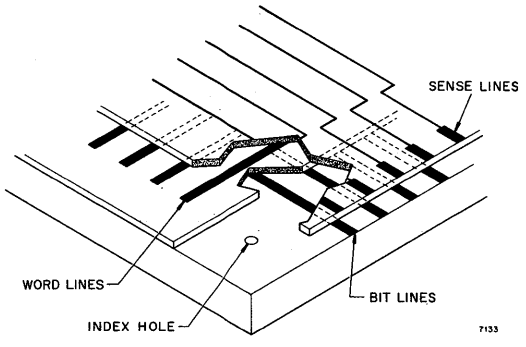


Figure 10. Sketch of closely spaced, three layer strip transmission line.

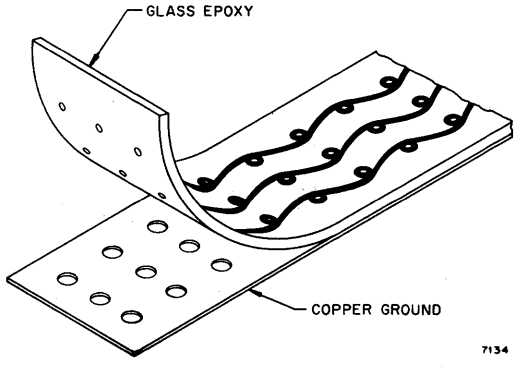


Figure 11. Sketch of etched circuit backboard power distribution.

model uses wire wrap connectors for all of its circuits which are built on conventional cards. The power distribution lines are fabricated from 4 mil double copper clad glass epoxy with a common ground plane and up to eight separate voltage lines. The pins from the connectors go through clearance holes in the ground and the appropriate ones are connected to the voltage lines. After the flexible distribution lines are in place, solder rings were dropped over the pins and a hot air stream used to solder all connections nearly simultaneously. In this case the individual voltage lines have a surge impedance of 5 ohms, although it could be made lower if necessary.

All of the circuits and techniques which have been shown were tested in the feasibility model completed early in 1962. A thin film stack of 512 words and 32 bits was constructed for the feasibility model. Only 16 words were driven and the loading effect on the word selection matrix of the missing words was carefully simulated. All of the drive circuit and logic circuits were spread out in a rack large enough to hold all of the circuits for a complete 4096, 32 bit word memory. Based on the test experience of the feasibility model which operated in a 250 nsec. cycle time, a full 4096, 50 bit word memory with a 200 nsec. cycle time is practical.

CONCLUSIONS

At the present time memories have been built that demonstrate the potential of thin films. In the near future, better solutions to the problems of connections and interconnections to the film elements will make submicrosecond million bit memories practical. These memories are expected to have a competitive advantage over other comparable memories.

Advanced work is leading to greatly reduced size of the thin film elements. Very high packing density (5000-10,000 bits per square inch) and low drive current (less than 50 ma) may be possible. Improved material properties should make coincident current selection with coherent rotational switching possible. With better techniques, high density integrated memory stacks with access lines made on the same substrate with the films seems likely. Microelectronic circuits probably could then be added to or made with the stacks.

ACKNOWLEDGMENTS

The authors extend their "Thank you" to the many co-workers at the UNIVAC Engineering Center, Blue Bell, Penna., that made this paper possible.

REFERENCES

1. M. S. Blois, Jr., "Preparation of Thin Magnetic Films and Their Properties," J. Appl. Phys., Vol. 26, pp. 975-980, August 1955.
2. Lincoln Lab - Division Five - Quarterly Progress Reports for years 1959, 1960, and 1961.
3. UNIVAC 1107 Brochures.
4. UNIVAC Brochure ADD-1, ADD-2-A Computers.
5. W. W. Carver, "Comparing Storage Methods," Electronic Industries, August 1962, pp. 120-130.
6. "UNIVAC Thin Film Memories" - UNIVAC Brochure #MO 5462.
7. D. A. Meier, "Magnetic Rod Memory," Proc. 1960 Electronic Components Conference, pp. 122-128, also "Large Capacity Memory Techniques for Computing Systems," pp. 195-212, edited by M. C. Yovitts - MacMillan.
8. D. O. Smith, "Anisotropy in Permalloy Films," J. Appl. Phys., Vol. 30, pp. 2645-2655, April 1959.
9. T. R. Long, "Electrodeposited Memory Elements for a Nondestructive Memory," J. Appl. Phys., Supplement to Volume 31, pp. 1235-1255, May 1960.
10. U. F. Gianola, "Nondestructive Memory Employing a Domain Oriented Steel Wire," J. Appl. Phys., Vol. 29, pp. 849-853, May 1958.
11. J. I. Raffel, "Operating Characteristics of a Thin Film Memory," J. Appl. Phys., Vol. 30, pp. 605-625, April 1959.
12. R. M. Tillman, "Fluxlok-A Nondestructive, Random-Access Electrically Alterable, High Speed Memory Technique Using Standard Ferrite Memory Cores," IRE Trans. on Electronic Computers, Vol. EC-9, pp. 323-328, September 1960.
13. J. I. Raffel, et al., "Magnetic Film Memory Design," Proc. IRE, January 1961, pp. 155-164.
14. W. E. Proebster, "The Design of a High Speed Thin-Magnetic Film Memory," Proc. 1962 Int'l. Solid State Circuits Conference.

15. E. M. Bradley, "Properties of Magnetic Films for Memory Systems," J. Appl. Phys., Supplement to Vol. 33, No. 3, pp. 1051-1057, March 1962.
16. A. V. Pohm, E. N. Mitchell, "Magnetic Film Memories A Survey," IRE Trans. Electrical Computers, Vol. EC-9, No. 3, pp. 308-314, September 1960.
17. W. Dietrich, W. E. Proebster, "Millimicrosecond Magnetization Reversal in Thin Magnetic Films," J. Appl. Phys., Vol. 31, pp. 2815-2825, May 1960.
18. R. D. Turnquist, C. D. Hogenson, V. E. Christenson, "A Compact 166-Kilobit Film Memory," Proc. IRE Int'l. Conv., New York, March 1962.
19. W. O. Freitag, et al., "Electrodeposition of Nickel-Iron Phosphorus Thin Films for Computer Memory Use," Electrochemical Society Meeting, Boston, Mass., September 1962.

HURRY, HURRY, HURRY

Howard Campaigne
Department of Defense
Jessup, Maryland

SUMMARY

Greater and greater capacity has been achieved in computers, and to a considerable extent this increase has been achieved by faster components. In fact this increase in capacity is the main attraction of still more speed, for this seems the way to get much more work for a little more money.

The state of the art appears to be that large increases in speed will be achieved in the near future, but beyond that it will take extreme ingenuity to break the natural barriers that present themselves. There are a number of competing techniques for the near future; it remains to be seen which of them will be the most useful. These techniques include cryotrons, sub-harmonic phase elements, Esaki diodes, and thin magnetic films. An impediment to progress is inadequate instrumentation.

The increased capacity of very high speed machines emphasizes the need for improved means to get data into and out of the computer. It now takes 5 million nanoseconds to get a tape unit into operation, and ten thousand nanoseconds per character to read it. To make available the new speeds for general application some new input-output inventions are needed.

The Need for Speed

The history of automatic sequenced computers has been one of greater and greater speeds giving greater and greater capacities at greater and greater economies. Look at the landmarks. The first leap in technology was the change from relays to vacuum tubes. This resulted in an increase of capacity by a factor of 2000, and an increase in cost by a factor of ten, roughly. The result is an economy by a factor of 200.

This was such a dramatic break-through that it made the world sit up and take notice. It was this break-through which started the computer industry. For any director of a university computing center who may be trying to find money let me hasten to emphasize that this economy is per computation; unfortunately the capital invested is large. In the same way a ten-ton truck can haul cheaper per pound than can a half-ton truck, provided one has ten-ton lots to haul. Very few situations require nanosecond reaction time for themselves, but almost everyone can use the increased capacity which the speed produces. A machine which is twice as fast can do twice as much work, almost.

Later improvements in technique have led to more speed, and thus more capacity, and more economy. The introduction of magnetic cores doubled the cost of machines, but also increased their effectiveness by an order of magnitude. Transistors also increased both the cost of the machines and their capacity. It now appears that the introduction of thin films will give another large step.

If one has need for large computing capacity one can always get it by paralleling equipment. But it has been cheaper, up to now, to go faster; much cheaper. At the present state of the art it seems that more speed is the cheapest way to get more capacity. Of course there will be an end of this some day, the law of diminishing returns will set in, and we will have to find our economies elsewhere.

State of the Art

How much longer can we hope to find ways to go faster? The early advances were so fantastic that one was tempted to believe that speeds might be increased without limit. But this is not so, as was observed by Mr. Einstein in 1905. According to his principle a signal cannot be sent farther than one foot in one nanosecond. Thus a machine a foot in diameter, designed so that the result of each logical step must be available throughout the machine before the next step, could not exceed a kilo-megacycle. In fact, because of the properties of various conductors, if electricity is used, or of dielectrics if light signals were used, it could not achieve much more than half this rate. The limit set by the speed of light is absolute, but it might be evaded by building very small machines. Whether our technology will be able to build such machines is yet to be determined. Shoulders of Stanford Research Institute has plans to develop a technique with which to build full scale computers as small as this, a "computer in a bottle." But Shoulders may be the Babbage of the twentieth century.

There is another way to avoid the limitation set by the slowness of light and that is to use a logical design in which computational steps are taken without waiting for previous results. In the present designs machines alter their states and then wait for transients to dissipate before proceeding, and this wait seems unnecessarily time consuming. If the wavelength were shorter than the machine then it could truly be called fast. Such a logic, "distributed logic" perhaps, may have severe restrictions on what it can do. So far no one has explored it.

At the present time demonstration machines can be built which have moderate sized memories (one thousand words) with 100 nanosecond cycle time (read and rewrite) memories. This was done with transistors with five nanosecond rise times driving thin magnetic films. The resistance of the lines had to be kept to very low levels, which was done by having very heavy copper buses. This had the amusing affect of making the equipment look like a Hollywood set, as though it were meant for very heavy loads. The engineers who are doing this are confident that with enough practive they can make some with 35 nanosecond cycle time. Beyond this there is little promise.

To accomplish 35 nanoseconds memory cycle would require circuits with rise and fall times of no more than three nanoseconds, physically compact for quick transmission, with power levels high enough to overcome opposition, and cooling to prevent the accumulation of heat. One sees that there are some contradictions inherent in this project, and that it can only be forced so far before these conflicts become critical. New techniques may avoid the difficulties, such as perhaps "molecular engineering," but such techniques must need be completely new, mere refinement of present methods will not do it.

Computers act just as do automobiles, the production machine does not move as fast as the laboratory demonstration. Specially designed cars can go over 200 miles an hour, races are run at 140, but productive traffic moves along at 40, one fifth the record. This drastic reduction is necessary for safety, reliability, and economy. Of course the 200 mile an hour automobile is easily passed by the jet airplane. One concludes from this analogy that the way to get high speed computing is to find completely new techniques. Where to look for such techniques? The jet engine was not invented by an automobile builder, nor by a propeller maker. We will have to cast far and wide to find new approaches.

In recent years exploration has been far and wide. The phase locked oscillator is a device which vibrates on a subharmonic (say half) of the pump frequency. It can therefore run in either of two phases. Once its phase is determined it will continue to oscillate in this phase, so it is bistable. A half wave length of conductor can reverse the phase. Other logical operations can be performed easily, especially that of majority vote. In fact logic is so easily performed by adjusting lengths of wires that the designer's main problem is to prevent transformations he doesn't want; in a big assembly this would become an overwhelming task. An attractive aspect of such a scheme is that the speed of logic is strictly proportional to the pump frequency. Since sine waves can be generated at super high frequencies, and even faster by using lasers, here is a way to go almost as fast as you please. The size of the components is strictly proportional to the wave length, which gets awful small. The very small tolerance on the dimensions of the

components is also proportional to the wave length. No one has been bold enough to build even a half-adder at more than four kilomegacycles. If techniques can be found to assemble such configurations within the very critical tolerances there may yet be a future in computing by microwaves.

Another area of exploration has been super conductivity. A super conductor has no resistance. In a magnetic field materials cease to be super conducting. Therefore magnetic fields can switch super conductors, and since the ratio of resistances is infinite (so far as we can measure) the switch is as positive as the common every day light switch. This switching can take place at fantastic speeds, but the capacitance and inductance of the circuits in which they are imbedded prevent using its highest speeds. If configurations can be found which avoid this impediment then the cryotron may be very useful, especially for very large assemblages, for they can be constructed by what amounts to printing press techniques. As long as one must supply a liquid helium bath for some cryotrons, one might just as well have a large number; it will take very little more helium and no more cooling equipment. Of course there would still be the interesting problem of maintaining such assemblages, steeped as they are in liquid helium, inside a narrow-necked Dewar flask inside a second Dewar filled with liquid nitrogen.

Magnetic cores, the old standby, can be plated thinly on glass or metal, and in this form work faster and can be assembled more easily than the conventional beads. By making the core thin, like half a wavelength of light, it can be made quick. It can be fabricated by evaporating or electro-plating or chemical deposition through a stencil. Such assemblage lends itself to large configurations, which I hope will eventually lead to easy and cheap manufacture. This technique is already available commercially; it remains to be seen how far it can be pushed.

Solid state technology is not to be left behind. Transistors get faster each month; speeds up to 200 megacycles have been achieved. Tunnel diodes are very fast but have limitations of their own, and there are even some three terminal tunnel devices being tried.

A limit on the state of the art is instrumentation. Our laboratory measurements are not made facilely enough. Perhaps this is because if we had techniques fast enough to make an easily usable pulse measurer we could use those techniques in the computer, and thus be left again without adequate instruments. The needs in instruments are pulse generators and detectors of pulse shapes. The experimentor would like to have pulses of prescribed shape at a prescribed rate on demand. He would also like to see the true shapes of the pulses after they have been filtered through a configuration of components. All the instruments now available do some additional filtering of their own. This is undesirable and must be minimized.

New Problems

If we picture a machine with a 35 nanosecond memory, logical units of comparable speeds, and as complex as most computers are today, then it will be able to do a man-year of work each second. How will we plan such work? If we plan it as we now do the work of extant computers, there will be a bottleneck at the input-output. Work will come and go so fast our heads will swim. We do not now have suitable means to feed work this fast. An invention is needed.

If we imagine that we will organize our work into bigger tasks, jobs with less input and less output and more running between, we can do this but it will take more planning and more pains, in other words, more programming. This uses a skill in short supply, so again we need a new development; a method of describing the job to be done which is succinct and flexible. Such methods are now being attempted, ALGOL being one. At the present time these languages use up considerable machine capacity of a specific kind. If one were using a type of "compile and go" system one would need to fetch the compiler and then store the object program. In any case there will be a need for large scale storage, fast enough not to unduly delay the main frame. Today's disc storage takes 100 million nanoseconds to find its place, and seven thousand nanoseconds per bit to read the data. Any waiting for such a device is clearly inefficient. In passing I notice that these slow and frustrating peripheral devices are also getting to be more expensive than the computer itself. That is, a quite reasonable little computer can now be bought for around sixty thousand dollars. But even a couple of tape transports, the

minimum for a sort, will cost eighty thousand.

The two outstanding innovations of the fifth decade were the jet airplane and the digital computer. The jet airplane can go one hundred times as fast as a man can run. A computer can compute ten thousand times as fast as a man can. The devices described here can go ten million times as fast.

BIBLIOGRAPHY

1. Haynes, M. K., "Transient Analysis of Cryotron Networks by Computer Simulation." Computer Issue Proc. IRE, 49 (Jan. 1961) p. 245.
2. Hollander, Gerhard L., "Gigahertz Computer Circuitry." Datamation, 8 Jul 1962, p. 43.
3. H. C. Hwang, R. Marolf, W. Peil, H. Raillard, E. P. Stabler, "Analysis of a Pumped Tunnel Diode Logic Circuit." Trans. PGCT, Sept. 1962.
4. Itner, W. B., III, "The Critical Fields of Thin Superconducting Films." Physical Review, 119 (1960) p. 1591.
5. Itner, W. B., III, "Speed and Power Balance in Cryogenic Circuits." 8th Lightning Quart. Prog. Rep.
6. Itner, W. B., III, "Switching Speeds in Thin Film Cryotron Loops." 9th Lightning Quart. Prog. Rep.
7. Mersel, Jules, "Research in MT at Ramo-Wooldridge." Proc. Nat. Symp. on MT. Ed. Edmundson, p.30, Prentice Hall, 1961.
8. Meyers, Norman H., "An Analog Solution for the Static London Equations of Superconductivity." Proc. IRE, 48 (1960) p. 1603.

THE CASE FOR CRYOTRONICS ?

W. B. Ittner, III
International Business Machines Corporation
Thomas J. Watson Research Center
Yorktown Heights, New York

While cryotronics is, to some degree, a rather special technology, it has not been possible to show that cryogenic circuits possess, on balance, any functional advantages over circuits realizable in competitive technologies. Accordingly, the only real promise for cryogenics would seem to lie in the possible cost advantages which might, in principle, be realized through the batch manufacturing of large integrated circuit modules. That such an advantage can be realized in fact is, at present, somewhat problematic.

The principal costs in the cryogenic technology are found to lie largely in five areas, namely: design, mask layout and checking; mask fabrication; circuit deposition; packaging; and refrigeration. A review of the steps required to carry out the construction and assembly of complete cryogenic processor indicates that, in terms of present technology, considerable cost reductions must be realized in each of the above areas if cryogenics is to be made commercially attractive.

The current state of the cryogenic technology is reviewed and an attempt is made to project the cost structure which would appear to be realizable in the near future.

It is appropriate, in presenting the case for cryogenics, to begin with two judgments which are, a priori, the basis for the central arguments which follow.

A. The technical problems which need be solved in reducing the cryotronic technology to practice are, within reason, developmental rather than fundamental in nature, i.e., no fundamental "breakthroughs" or major innovations are believed to be required.

B. The cryotronic technology, while possessing a number of special features, is functionally equivalent, on balance, to a number of existing technologies and must, therefore, be evaluated basically in terms of the function it can provide for a given cost.

If we admit these judgments as fact, it is now expedient to examine a number of the costs which underlie the production of cryogenic hardware and to then examine a number of applications for which cryotronics has been considered.

One of the major costs in the fabrication of cryogenic integrated circuits is that associated with the design and layout of the composite circuit and the generation of the artwork from which the stencils or masks for deposition are eventually fabricated. For complex logical circuits where redundancy or modularity is low, it has been estimated that about one man hour of labor per cryotron is required to manually produce the detailed designs for a complete set of deposition masks. It is apparent that the mask designs for a circuit containing, say, 5000 cryotrons could easily cost tens of thousands of dollars and that it would be necessary to produce thousands of circuits in order to economically absorb such design cost. Obviously much can be done to automate the design process and computers are already being used to assist in the layout and checking of circuit patterns. It would appear, however, that even with automation, design costs may be high enough to limit the use of large integrated circuits to applications where the circuits are highly redundant or extremely modular, e.g., memory.

The fabrication of precision masks is, at present, a rather costly process which adds significantly to the cost of the finished circuit. Typically, several dozen masks are required for a complete circuit and the precision masks have a rather finite life due to the necessity of periodically cleaning them of evaporated material. It appears that much can be done to lower costs in this area, but it is equally obvious that substantial work must be done to provide large precision masks economically.

Vacuum evaporation is currently employed as the central technique for depositing cryotronic circuits and experience has shown that an automated and sophisticated production facility is not particularly inexpensive either to construct or to operate. Economic production of circuits is possible only if the "throughput" of the system is high and it is possible to produce circuits in volume. Economic production does appear possible, but a cursory examination of most of the present production systems is sufficient to indicate that considerable work must be done to provide for decreased cycle time in the fabrication process and for true on line "batch fabrication."

Packaging of cryogenic circuits is both a technological and economic problem. Obviously, low impedance interconnections are required and it is desirable to hold the number of interplanar connections to an absolute minimum. Demountable connections appear to be necessary for economic assembly and disassembly of arrays. Within limits (and depending on the application) the problem is not insurmountable, but the difficulty of making numerous or lengthy connections between substrates does appear to limit seriously the general applicability of the cryogenic technology. Here again, cost must be borne in mind and, unfortunately, the cost of the "precision" connectors required for high-speed coupling is nonnegligible.

Finally, the cost of refrigeration must be weighed in evaluating the performance of a cryogenic processor. Costs of closed cycle refrigerators are, at the present time, prohibitively high, and virtually rule out the serious considerations of cryotronics for any application. On the other hand, such units are in their commercial infancy and it is obvious that initial production costs will be significantly reduced in time. The cost of servicing and maintaining such refrigerators is still unknown, but hopefully this will not prove to be a serious problem.

With the above factors in mind, it is convenient to now consider the potential of a number of proposed cryogenic applications.

1. *Continuous Film Memory.* This type of memory, wherein storage is provided by circulating supercurrents which are trapped in a superconducting sheet or film, is functionally equivalent to conventional magnetic core or thin-film memories. For very large memories, superconducting storage appears to offer a number of advantages over conventional room temperature storage. Basically, the technical advantages accrue from the fact that the back emfs and signal attenuations on the drive lines are small when the lines are superconductive and from the fact that the signal to noise ratio on the output is virtually independent of the size of the memory array (due to the screening of the superconducting sheet). These advantages result in reduced costs in the memory driving and sensing equipment, particularly as the array size is increased. Further, since the storage array can be batch fabricated together with the memory decode network, it would appear that very large memories could be fabricated quite economically.

In many ways this application appears to be ideally suited to the cryogenic technology. Plane design and layout is simple and all planes in the memory are redundant. The number of edge connections per plane is small in comparison to the number of elements contained on a plane and interconnection is not a serious problem. The duty cycle in such an application is relatively low and power dissipation is the decoder and memory array is trivial (of the order of a milliwatt at a megacycle operating rate). The chief power dissipation comes from the lines leading into and out of the helium bath and very modest refrigeration is required for a sizable memory.

2. *Associative Memory.* This type of application combines some of the simplicity inherent in memory with some of the advantages gained by utilizing the logical capabilities of cryotrons. Like the continuous film memory, planes can be used redundantly; the duty cycle (and hence the power dissipation) is low; and the number of interplane connections is modest. The application is quite obviously an appropriate one for the cryogenic technology, but it is not certain at this time

just how significant an application can be made of associative memories themselves. This question is obviously quite debatable, but it appears probable that cryogenics could be a significant technology for special purpose processors of this general class.

3. *General Logical Processors.* It is in this area that the future of cryotronics is least assured. The projected performance of in-line cryotron circuits is roughly comparable to the performance expected from high-speed transistor circuits. In principle, cryotrons would appear to offer possible cost advantages through the batch fabrication of the integrated circuit planes. Whether this basic advantage will be lost in the costs of design, packaging, and refrigeration is, at present, unknown.

CRYOTRONICS - PROBLEMS AND PROMISE

Martin L. Cohen
Arthur D. Little, Inc.
Cambridge 40, Massachusetts

In the past few years cryotronic technology has matured to the point where one may reasonably discuss basic limitations and important engineering problems. The basic limitations stem from certain specifics of the technology, such as gain bandwidth products, impedance and signal levels and the required low-temperature environment. The engineering problems are the problems associated with taking a device from the laboratory to the production line. Reliability, electrical and mechanical strengths, quality control, yield rates and other economic problems have, at worst, a certain nuisance value in the laboratory, but are of vital importance in a production situation.

Consider first some of the basic limitations. Cryotrons are not exceptionally fast. Speeds on the order of one to ten or twenty megacycles can be obtained from the simplest sort of circuitry. An order of magnitude improvement, that is, stage delays of about five nanoseconds can be obtained from biased cryotrons. The impedance level of superconductive circuits is an interesting source of difficulty. This impedance level, defined here as the resistance of a quenched gate, is only on the order of one to one hundred milliohms. Cryotron circuit impedances do not match those of transistor input-output circuitry or transmission lines. This mismatch means inefficient power transfer between cryotron circuits and the outside world and slow transfer of information in cryotron circuits. Stray inductances of ten to one hundred pico henrys are very significant and prevent full use of a 10^9 sec^{-1} gain bandwidth product.

Finally, cryotrons require a low temperature environment. There is no fundamental problem here since reliable, closed-cycle helium refrigerators are presently available. Unfortunately the cost of these refrigerators is greater than originally contemplated, and places an economic limitation on superconductive circuits. Small systems are uneconomical; a system large enough to absorb the refrigerator cost must be considered. Cryotrons are also temperature sensitive. Typical operating temperatures are about one-quarter to one-half degree Kelvin below the critical temperature of the gate material. Since critical currents are roughly proportional to this temperature difference, temperature rises of hundredths of a degree are significant, and heating rather than the electrical time constants may limit circuit speed. Good temperature regulation of the bath and high thermal conductivity substrates are desirable.

Production problems present a less clear situation. Although the problems are well defined, much of the recent progress toward solving these problems is of a proprietary nature and therefore the present state of the art is obscure. Nevertheless, device parameters must remain stable with time, insulation must insulate even after repeated temperature cycling and connections to the substrates must endure. As far as we know all these problems have been or can be solved by sufficient development effort. For example, mechanical stress due to the different thermal coefficients of expansion of the various materials in a cryotron circuit, can cause crazing of the insulation and short or open

circuits. But we have found that when metal rather than glass substrates are used the problem is much less and that even with glass substrates, a suitable encapsulant will hold the circuit together and solve this problem. Only time will tell how well all these problems have been solved and only through actual experience can reliable data on yield rates and manufacturing costs be obtained. The success or failure of cryotronics hinges on these economic factors, and since the largest circuits made to date contain only a few hundred cryotrons, the future is still uncertain.

Let us now consider how these problems are reflected in certain proposed superconductive systems. Simplest and closest to commercial exploitation is the RCA-Crowe cell or continuous sheet memory. This circuit replaces ferrite core arrays. It is a coincident current, destructive read memory. There are a number of technical problems. That tight tolerances must be met and metal substrates may not be used are peculiar to the circuit. Temperature sensitivity is illustrated by a significant temperature rise at about a one megacycle operation rate even with a high conductivity substrate. Large input currents (tenths of amperes) and low output voltages (millivolts) result from the low impedance level.

If we assume that a suitable refrigerator can be obtained for $30,000, input-output equipment costs are the same as for core arrays and that the continuous sheet memory planes cost $100, one finds that RCA-Crowe cells become competitive with cores at about the 10^6 bit size ($.03 per bit for the refrigerator).

Some very interesting sophisticated memory schemes have been proposed using cryotrons rather than Crowe cells. These include associative memories as well as nondestructive multiple access memories both with and without such additional features as internal transfer or shifting of information. The packing densities obtainable in these circuits are less than those of the RCA-Crowe cell (10^4 bits/square inch) by one or two orders of magnitude. The cost per bit will therefore be greater than that of the continuous sheet memory. Some of the special features such as internal transfer, require switching currents in long loops, a comparatively slow process which may limit speeds to the hundred kilocycle range.

The economics of those sophisticated memories is unclear. Neither the relative desirability of the various features nor the market for any given design are known and furthermore there are no presently existing memories that can be used as a basis of comparison.

The ultimate dream of an all-cryotron computer brings in the problem of information transfer. If a conventional computer design is used, signals must be gated on and off some main transfer bus. Cryotron impedances are too low for these circuits to work at competitive speeds. Perhaps some new device will be found to be used with cryotrons for this purpose. An alternative scheme is to design the computer as a very large memory with sufficient internal logic to perform all the necessary computer functions. But this sort of computer, as yet not designed, falls into the class of sophisticated memories already discussed.

Cryotronics is just one approach to integrated circuits. If we broaden the definition of a component to mean any circuit element or circuit or system made all at once, assembled, and not dissectable, so as to include any thing from a resistor to a whole integrated circuit computer we may consider the concept of the maximum economic size or complexity of a component. A simple component has the advantage of versitility and the consequent large market. A one kilohm resistor can be used everywhere and millions are used. On the other hand, the more complicated component, say a silicon substrate flip-flop in a transistor can, is potentially cheaper than an equivalent flip-flop composed of smaller components because of savings in handling and assembly costs. But this potential can only be realized if there is sufficient market for the more complex component to defray the costs of expensive, specialized manufacturing equipment. The maximum economic size has increased gradually as the entire electronic technology has grown. But in cryotronics a large step, rather than gradual growth is necessary, because of the cost of the special low temperature environment that is required. It is possible that the world is not yet ready for cryotrons, that is, there is not yet sufficient market for even large cryotronic systems to be economical.

SOME EXPERIMENTS IN THE GENERATION OF WORD AND DOCUMENT ASSOCIATIONS

*Gerard Salton**
Harvard Computation Laboratory
Harvard University
Cambridge, Massachusetts

INTRODUCTION

The solution of most problems in automatic information dissemination and retrieval is dependent on the availability of methods for the automatic analysis of information content. It is, in fact, impossible to identify, classify, encode, and organize items of information, or requests for information, without first determining the content or subject matter of the information to be processed. In most proposed automatic systems, this analysis is based on a counting procedure which uses the frequency of occurrence of certain words or word classes to generate sets of index terms, and to prepare automatic abstracts or extracts.

Unfortunately, it is not possible to perform completely effective subject analyses solely by frequency counting techniques. First, if such techniques are to be trusted, it is always necessary to accept the underlying assumption that information content can be expressed completely in terms of the words which occur in individual texts or documents. Second, since most types of relations are difficult to identify by statistical methods, it is necessary to assume that relations between words can be largely disregarded. Finally, because of the variability inherent in the texts of different documents, frequency counts are always used more effectively for individual documents than for complete document collections. Thus, counting techniques, while generally acceptable as a valuable adjunct in automatic content analysis, cannot be relied upon to handle all required tasks. In particular word and document associations are difficult to determine by frequency counts alone.

The simplest solution to the content analysis problem, as indeed to many other problems in information processing, might lie in the construction of a big semantic dictionary, including all relevant data about all subjects, as well as cross-references and relationships between the different aspects of these subjects. However, even if it were possible to wait until such a dictionary could be built, it would not be clear how the material should be organized, nor indeed do any criteria exist for determining what material should be included and what should be left out. For practical purposes, it seems more useful therefore not to count on the existence of complete semantic dictionaries.

The present study is concerned instead with two automatic methods, both of which use information directly supplied by the texts to be analyzed, and neither of which requires extensive dictionaries or tables of the type normally used for automatic

*This study was supported in part by the Air Force Cambridge Research Laboratories, and in part by Sylvania Electric Products, Inc.

classification systems. The first one uses information extracted from certain function words, such as prepositions or conjunctions, and from suffixes attached to many words in the language to generate syntactic indicators. These indicators, in turn, are then used to determine syntactic functions and word associations by a simplified type of syntactic analysis. The second method uses bibliographic citations in the form of a reference index (listing citing documents against cited documents) or a citation index (listing cited documents against citing documents) to determine document similarities. Experimental results are shown for both methods, and suggestions are made for their utilization in conjunction with other standard methods for content analysis.

Word and Document Associations

The most immediate approach to the problem of generating word and document associations consists in extending the frequency counting methods previously used for automatic indexing and abstracting [1-3]. Indeed by assuming that close proximity between certain word occurrences in texts implies a strong association between these same words, and by further postulating that the strength of association between documents varies as a function of the number of high-frequency words which co-occur in these documents, it is possible to compute association or similarity coefficients as a function of the number and type of co-occurring words in phrases, sentences, paragraphs, or complete texts [4,5]. Word and document associations can then be displayed in tabular form in decreasing order of magnitude of the similarity coefficients, or, alternatively, it is possible to exhibit these associations on a two-dimensional map by linking associated words or documents [4,6,7].

While such association maps are valuable tools in content analysis, the need to provide additional information concerning the type of association which obtains between words or documents is nevertheless widely recognized. For example, if the word "information" is found to be associated with the word "documents" in certain texts, it is, of course, not clear whether these texts deal with "information about documents," or "documents about information," or "information in documents." It has been suggested that "with little extra strain on storage, some simple indications of relationships between key words can also be added" (to the two-dimensional association maps) [4]. However, no information is given concerning the type of relation to be displayed, or the manner in which these relationship indicators are to be generated.

Ideally it might be desirable to display complete encyclopedic knowledge about all items appearing in an association map, including, for example, information about the environment, property lists, homographic uses, and so on. In the absence of a complete semantic dictionary, a compromise is usually made in actual systems, either by restricting the relations to be recognized automatically to certain very specific types [8-10], or else by providing extensive lists of relationships which are not, however, recognized fully automatically [11,12].

In a practical automatic system, however, it would seem necessary on the one hand to distinguish more than a few specific types of relations, and on the other to perform the recognition procedure automatically without benefit of extensive dictionaries. These requirements would suggest that a small number of restricted dictionaries be used together with other indications provided by the linguistic context. The following linguistic indicators might provide important information:

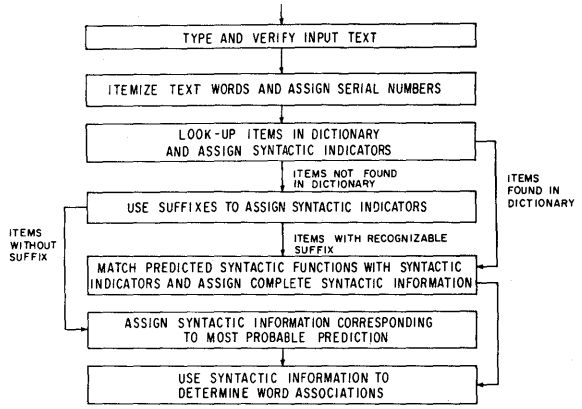
a) prepositions and other function words;
b) affixes of various kinds;
c) special quantifiers such as "many," "some," "every";
d) logical connectives such as "and," "or," "not";
e) special referents such as "like," "these," "those";
f) special linguistic units such as "the fact is," "it is claimed," "it is hoped."

Prepositions and other function words have been used in the past for the extraction of semantic indications [13-15]. More general forms of syntactic analysis may be used similarly for the recognition of word relations [15-17]. In the next section, a method is presented for the recognition of word associations by means of a simple form of syntactic analysis.

A Simple Syntactic Method for the Recognition of Word Associations

Dictionary Look-up and Suffix Analysis: The method to be described takes ordinary

English text and assigns a unique part-of-speech indicator and a unique syntactic function indicator to each word. These indicators are then used to assemble words into phrases and phrases into clauses. The complete program is described in the flowchart of Figure 1.



SIMPLIFIED SYNTACTIC ANALYSIS PROGRAM

Figure 1

The input text is first transferred onto magnetic tape, and the individual words are separated and provided with a serial number. A small dictionary is then used to assign syntactic part-of-speech indicators to those words which are included in the dictionary. A dictionary excerpt is shown in Figure 2. It may be noted that each dictionary item is provided with as many syntactic indicators as there are possible applicable parts of speech. For example, the word "weekly" reproduced as the first item in Figure 2 is furnished with three indicators (HO,AO,N1), representing respectively adverb, adjective, and noun-singular indications. Certain semantic indicators are also included in the dictionary. The "TO" semantic indicator, for example, represents a time indication. Other semantic indications also included, denote motion, location, direction, dimension, value, duration, and so on.

The dictionary presently used contains about four hundred function words, such as prepositions, conjunctions, adverbs, articles, and so on, and about five hundred common nouns, verbs, and adjectives. The dictionary composition is shown in detail in Table 1. Experimental evidence indicates that for the technical texts tested approximately fifty-five percent of the words are found in the dictionary, so that forty-five percent of the words are still left without any syntactic indication after dictionary look-up.

INPUT ITEM	SEMANTIC INDICATOR (2nd SYNTACTIC CLASS)	SERIAL NUMBER	INPUT ITEM AND SYNTACTIC INDICATORS	SEMANTIC INDICATOR (1st AND 3rd SYNTACTIC CLASSES)
WEEKLY		GAS-00816000	WEEKLY %	TO
	TO		HO, AO, N1	
WEEKS		GAS-00802000	WEEKS %	TO
			N2	
WELL		GAS-00098000	WELL %	
			HO, AO, N1	
WENT	"	GAS-00208000	WENT %	MV
			V1	
WERE		GAS-00043000	WERE %	LV
WEST		GAS-00559000	WEST %	DO
	DO		N1, AO, HO	DO
WHAT		GAS-00048000	WHAT %	
		RPR000000000	C1, P3, AO, HO	
WHEN		GAS-00051000	WHEN %	TO
	TO	RPR000000000	C1, HO, P3	TO
WHENCE		GAS-00333000	WHENCE %	
			C1, HO	
WHENEVER		GAS-00334000	WHENEVER %	
			C1, HO	

DICTIONARY EXCERPT

Figure 2

In order to generate parts-of-speech indications for the words not found in the dictionary, a suffix analysis program is used. Specifically, an attempt is made to detect a recognizable suffix for each word by comparing the word endings with a list of suffixes contained in a suffix table. Suffixes of seven letters are tested before suffixes of six letters, and so on, down to suffixes of one letter. Special provisions are made to detect plural noun forms and third person singular present forms for verbs. When a suffix match is obtained, the part-of-speech indicators included in the suffix table are attached to the corresponding text words. An excerpt from the suffix table is shown in Table 2.

It should be noted that many words include "false" suffixes which, when looked up in the suffix table, would provide incorrect information. For example, the words "king," "wing," and "sing" would be provided with "gerund" or "present participle" indicators because of the "ing" ending. Such words are considered to be exceptions; they are included for the most part in the regular dictionary so as to eliminate the suffix analysis. Experience has shown that the suffix

Table 1. Dictionary Composition

SYNTACTIC INDICATIONS	NO OF WORDS IN DICTIONARY*	PERCENTAGE OF TOTAL
NN (no VB)	266	27.7 %
NN, VB (no VBEF[1])	101	10.5
NN, VB (with VBEF[1])	57	6.0
VB1 or VB2 (no NN, no IOBJ[2])	118	12.3
VB1 or VB2 (with IOBJ[2])	44	4.6
VB3 (no V3TI[3])	12	1.3
VB3 (with V3TI[3])	4	0.4
VB4	13	1.4
PO1	5	0.5
PO2	8	0.8
PO3 (no NAMS[4])	21	2.2
PO3 (with NAMS[4])	8	0.8
COM (no CON)	1	0.1
CON (no COM)	49	5.1
COM CON	12	1.3
PCT	7	0.7
ADJ (no NAMS[4])	271	28.2
ADJ (with NAMS[4])	11	1.2
ADV	134	13.9
PRE	71	7.4
PRI	1	0.1
ART	3	0.3
PRP	67	7.0
PAP	72	7.5
TOTAL WORDS	963	

1) VBEF:
VERB PREDICTIONS TAKE PRECEDENCE OVER NOUN COMPLEMENT PREDICTIONS.

2) IOBJ:
VERBS MAY TAKE INDIRECT OBJECT

3) V3TI:
VERBS MAY TAKE AN INFINITIVE VERB COMPLEMENT (OUGHT, NEED, DARE, USED).

4) NAMS:
CANNOT FULFILL ADJ MAS OR NADJ CM PREDICTIONS (THIS, THOSE)

* ITEMS EXHIBITING MORE THAN ONE SYNTACTIC INDICATOR ARE LISTED SEVERAL TIMES

Table 2. Excerpt from Suffix Table

NO. OF CHARACTERS	SUFFIX	SYNTACTIC INDICATOR*
7	SORBING	PRP
	DENTIAL	ADJ
	NENTIAL	ADJ
	RENTIAL	ADJ
	TENTION	NN1
	TURBING	PRP
	SENTIAL	ADJ
6	ANSION	NN1
	ANTIAL	ADJ
	ENSSON	NN1
	ENTIAL	ADJ, NN1
	⋮	⋮

* EXCEPTIONS APPEAR IN THE DICTIONARY

analysis provides syntactic information for an additional thirty percent of the words found in technical texts. After the suffix analysis, less than fifteen percent of the words are thus left without any grammatical information.

A small piece of suffix analyzed output is shown as an example in Figure 3. It may be noticed that all items found in the dictionary, or furnished with a recognizable suffix, are provided with syntactic indicators as a result of the table look-up procedures. The words "atmosphere" and "alarm" in the excerpt of Figure 3 could not be classified by any of the available methods, and are therefore left without grammatical indication.

Predictive Syntactic Analysis: Following the suffix analysis, a simplified predictive syntactic analysis is used to assign to each word a unique part of speech indicator and a unique syntactic function [18,19]. Specifically, each sentence is scanned from left to right, one word at a time. At each point, predictions are made concerning the syntactic structures to be found later in the same sentence. When a new word is considered, its associated grammatical indicators are tested against the unfulfilled predictions available at that time. If a match is found between any of the predicted grammatical functions and the available syntactic indicators, the first matching grammatical function and the associated part-of-speech indication are assigned to the corresponding word in the input text. The accepted prediction is then deleted, and further predictions are made for new structures to be expected later in the same sentence. If no match can be found between any unfulfilled prediction and available grammatical indication, as is the case when words are tested which have no associated grammatical information, the most likely prediction is used to generate acceptable grammatical indicators for the corresponding input items.

The predictive analysis makes use of information stored in two separate tables, the grammar table and the prediction table. The grammar table lists predicted syntactic functions against the syntactic indicators which can fulfill each prediction, and the prediction table lists accepted syntactic indicators

INPUT ITEM	SERIAL NUMBER	INPUT ITEM AND SYNTACTIC INDICATORS		SYMBOLS IDENTIFYING PROCEDURE USED TO GENERATE SYNTACTIC INDICATORS*	
OF	ISH-02540000	OF RO	%		
IMPENDING	ISH-02550000	IMPENDING B1, N1	%		X-LIT.
CRISIS	ISH-02560000	CRISIS N2, V2	%		X-LIT.
	ISH-02570000	. PO	%		
NOW	ISH-02580000	NOW HO, NI, C1	%		
AN	ISH-02590000	AN TO	%		
ATMOSPHERE	ISH-02600000	((((((((((((		X-LIT.	X-LIT.
OF	ISH-02610000	OF RO	%		
ALARM	ISH-02620000	((((((((((((		X-LIT.	X-LIT.
	ISH-02630000 DUPC30000000	, CO, C1	%		
SINCE	ISH-02640000	SINCE C1, RO, HO	%		
IT	ISH-02650000	IT P3	%		

*BLANK / BLANK:
ITEM FOUND IN DICTIONARY

BLANK / X-LIT:
SYNTAX GENERATED BY SUFFIX ANALYSIS

X-LIT / X-LIT:
NO SYNTACTIC INFORMATION AVAILABLE

SUFFIX ANALYZED OUTPUT

Figure 3

against new syntactic functions to be predicted as a result of a match. Excerpts from the grammar and prediction tables are shown in Tables 3 and 4 respectively.

Table 3. Excerpt from Grammar Table

SYNTACTIC FUNCTION PREDICTED	SYNTACTIC INDICATORS SATISFYING THE PREDICTION
SUBJECT	ART, ADJ, NNI, NN2, PRP
ADJ MAS	NNI, NN2, ADJ
OBJT VB	ART, ADJ, NNI, NN2
PRED HD	VB1, VB2, VB3, VB4
NADJ CM	NNI, NN2
VERB CM	VB1, VB3, VB4, PRP, PAP, PRI
ADVB MS	ADJ
PREP CM	ART, ADJ, NNI, NN2, PRP
ADVB ES	ADV
PREP ES	PRE
END SEN	PCT

Table 4. Excerpt from Prediction Table Generated by Accepted Syntactic Indicators

ACCEPTED SYNTACTIC INDICATOR	SYNTACTIC FUNCTIONS PREDICTED
ART	ADJ MAS, ADVB ES
ADJ	ADJ MAS, NADJ CM, PREP ES
NNI	NADJ CM, PREP ES
NN2	NADJ CM, PREP ES
PO1	NADJ CM, PREP ES
PO2	NADJ CM, PREP ES
PO3	NADJ CM, PREP ES
PRP	(NADJ CM),[1] VERB CM, PREP ES, ADVB ES, OBJT VB
COM	LAST PREDICTION SATISFIED BY CERTAIN ESSENTIAL ITEMS, (SCL SUB, PRED HD, INFINTY)[2]
CON	SUBJECT, PRED HD, (DUP CNJ)[3], INFINTY, (ADVB MS)[4]
PCT	SUBJECT, PRED HD, INFIN. REL CLS, END SEN

1) WHEN ACCEPTED AS "PREP CM"

2) WHEN NOT OCCURING BEFORE "CON"

3) WHEN EXHIBITING SUPPLEMENTARY "DUPC" CODE

4) WHEN ACCEPTED AS "DUP CNJ" FUNCTION

Consider, for example, the conditions which obtain at the beginning of a sentence. The functions predicted initially are "subject," "predicate," and "end of sentence," in that order. Suppose that the first word in the sentence has an associated "article" indicator. The grammar table is then consulted to find out whether the first prediction (subject) matches the available part-of-speech indication (article). The first entry in Table 3 shows that the subject prediction is fulfilled by "article," "adjective," "noun singular," "noun plural," and "present participle" indicators. A match is therefore found, and "subject" and "article" are assumed to be the correct grammatical function and part-of-speech indication for the first word in the sentence. The "subject" prediction is then erased from the list of unfulfilled predictions, and a number of new predictions, as indicated by the prediction table, are added to the list.

Since the "article" indicator was accepted as correct for the first word, Table 4 shows that "adjective master" and "adverb essence" predictions are to be added to the list of predictions. The list of predictions operates, with minor modifications, as a pushdown store, so that new predictions are always added at the top of the list. When the second text word is about to be processed, the list of unfulfilled predictions therefore contains "adjective master," "adverb essence," "predicate," and "end of sentence" predictions in that order. Subsequent words are processed in the same manner until the end of the sentence is reached, at which point the list of predictions is cleared and initial conditions are restored.

The analysis of a short sentence is shown as an example in Table 5. The generation code indicates whether any syntactic indicators were available at the start of the predictive analysis, and, if so, how they were generated. The column labelled "predictor serial" contains the serial number of the item which was originally used to generate

Table 5. Analysis of a Sample Sentence

INPUT TEXT	GENERATION CODE	SERIAL NO.	SYNTACTIC FUNCTION	SYNTACTIC INDICATOR	PREDICTOR SERIAL
ON	D	0238	INFINTY	PRE	*237
A	D	0239	PREP CM	ART	238
SUITABLY	D	0240	ADVB ES	ADV	239
SCALED	S	0241	ADJ MAS	PRP	239
GRAPH	NI	0242	NADJ CM	NNI	241
,	D	0243	INFINTY	COM	*237
ANY	D	0244	/PREP CM	ADJ	243
RISING	D	0245	ADJ MAS	PRP	244
EXPONENTIAL	S	0246	/OBJT VB	ADJ	245
CURVE	NI	0247	ADJ MAS	NNI	246
TENDS	D	0248	PRED HD	VB2	*237
TO	D	0249	VERB CM	PRI	248
PRODUCE	D	0250	INF BSE	VB1	249
AN	D	0251	OBJT VB	ART	250
HYPNOTIC	S	0252	ADJ MAS	ADJ	251
EFFECT	D	0253	ADJ MAS	NNI	252
OF	D	0254	PREP ES	PRE	253
IMPENDING	S	0255	PREP CM	PRP	254
CRISIS	S	0256	NADJ CM	NN2	255
	D	0257	END SEN	PCT	*237

D - DICTIONARY
S - SUFFIX ANALYSIS
NI - NO INFORMATION

* INITIAL PREDICTION
/ ERRONEOUS ANALYSIS

the prediction fulfilled by the current word. Thus the word "tends" in Table 5 was recognized as a verb fulfilling the "predicate head" prediction. The predictor serial (237) shows that "predicate head" was an initial prediction originally predicted at the start of the sentence. The "syntactic function" and "predictor serial" columns of Table 5 are used later in the bracketting procedure which generates phrases and clauses.

By checking the syntactic functions of Table 5, it may be seen that an error was made in the analysis. In particular, the subject phrase "any rising exponential curve" was not properly recognized because the comma (item 0243) predicted a parallel construction consisting of another prepositional phrase. This particular error can be caught since the subject prediction, which is considered to be "essential," is left unfulfilled at the end of the sentence. However, other mistakes are made in the course of a normal analysis which cannot be detected so easily. These errors are due to three principal causes:

a) the absence of grammatical information for words not found in the dictionary or in the suffix table;
b) the inadequacy in certain cases of the grammar and prediction tables used;
c) the large number of syntactic ambiguities present in the language.

Clearly, nothing can be done about the last of the three listed causes. In fact, even the most elaborate automatic methods for syntactic analysis, including, in particular, those which make use of complete syntactic dictionaries of the language, produce erroneous output in cases of linguistic ambiguity. The question arises whether the performance of the crude analysis here described is much less reliable than other more ambitious programs.

A list of principal error types found in the current analysis appears in Table 6. The analysis of the output indicates an error percentage in the assignment of syntactic functions of 7 to 8 percent primary errors, and 6 percent induced errors. The latter category consists of errors which arise as a result of some preceding error in the same sentence. If, for example, the word "profound" in the phrase "many profound questions..." is not found in the dictionary, it will be assigned a noun indicator satisfying the subject prediction. When the word "questions" is taken up, the subject prediction will already have been erased, and "questions" is therefore recognized as the third person present singular form of the verb "to question" satisfying the predicate prediction. The latter error is thus induced by the false classification of the adjective "profound."

In general, local structures such as subject phrases, noun complement phrases, preposition complement phrases, participial phrases, verb complement phrases, and so on, are almost always properly analyzed. Errors arise most frequently in the correct recognition of coordinated and subordinated structures, and, in particular, in the accurate prediction of parenthetic and parallel constructions. For example, in the sentence

"The conceptual, rather than equipment, aspects..."

the first comma correctly predicts a parenthetic expression. The word "than," however, generates among other predictions a subclause prediction which dominates the other unfulfilled predictions. When the second comma is reached, it is interpreted in accordance with the latest prediction as signalling a parallel construction, thus repeating for "aspects" the same assignment as for "equipment."

Table 6. Principal Error Types

ERROR TYPE	EXAMPLES
I ITEMS NOT ENTERED IN DICTIONARY	
1. ADJECTIVE ⟶ NOUN	BENEFICIAL, REMEDIAL, ANNUAL, SCIENTIFIC, REAL, OBSCURE;
2. NOUN ⟷ VERB (WORDS IN -S)	EXPENDITURES, TRENDS, PROBLEMS; TENDS, INSISTS;
3. NOUN ⟷ VERB (VARIOUS)	INFORMATION, EDUCATION; GENERATE, ASSIMILATE; NEEDED, OUTSTRIPPED, PUNISHED;
4. ADVERB ⟷ NOUN, ADJECTIVE	SIMPLY, REGARDLESS, INDEED; ABSURD, AWARE;
II ITEMS IN DICTIONARY WITH CORRECTLY ASSIGNED SYNTACTIC FUNCTION	
1. CONJUNCTION ⟷ ADJECTIVE ("THAT")	(EXTRA OBJECT PREDICTION LEFT IN POOL)
2. CONJUNCTION ⟶ RELATIVE SUBJECT	THAT, WHO
3. PRONOUN ⟶ ADVERB ("SO")	SO FAR AS, SO VERBOSE

In order to judge the severity of the handicap arising from the absence of a complete word dictionary, a test run was made with a special dictionary, which included all relevant words occurring in the sample texts. The error rate was found to be reduced by less than forty percent, resulting in a primary error rate of 4.5 percent and an induced rate of about 4 percent. These figures are comparable with error rates produced by other far more complicated syntactic analysis programs. The absence of the dictionary is therefore largely compensated by the effectiveness of the suffix analysis and of the modified predictive techniques. This conclusion is further confirmed by the fact that the bracketting program which produces word associations performs almost perfectly for local structures.

Bracketting Program: A number of computer programs are in existence which use

the output of an automatic syntactic analysis to produce dependency structures in tree form [14,20,21]. The list of all dependent word groups can be obtained from a dependency tree by following all branches of the tree, and generating groups of words, located on the same branch. Alternatively, if the tree structure is not available, the same result can be achieved by using the information provided by the predictive analysis.

Specifically, the syntactic function indicators and predictor serial numbers (columns 4 and 6 of Table 5) are used to produce for each item a chain serial number and a chain function. The chain serial numbers for "subject," "subclause subject," "predicate head" and "preposition essence" are the same as the corresponding predictor serials; for other syntactic functions, the chain serials are given by the predictor serials for the embedding structure. The chain serial is then effectively defined as the serial of the first item in the same phrase, although discontinuous constituents will cause items with the same chain serial to be separated by other extraneous items.

For example, preposition complements are assigned the chain serial corresponding to the predictor serial of the preceding preposition. Verb complements are similarly assigned the chain serial for the predictor of the preceding predicate head. Comparable rules apply to the remaining syntactic functions. The chain function assigned to a given word is the syntactic function associated with the predictor of the embedding structure, except that "subject," "subclause subject," "predicate head," "preposition essence," "preposition complement" and "infinity" functions keep their present function indicator.

In order to bring together all members of the same structure, it is only necessary to sort the items in chain serial number order, while at the same time preserving the word order inside each substructure. The structures generated for the sample sentence of Table 5 are represented by brackets in column 4 of the Table. The chain functions assigned to the five brackets are, respectively, preposition complement, preposition complement, predicate head, object of verb, and preposition complement. The chain function assigned to the second bracket is in error, since the subject of the sample sentence was not correctly recognized, as previously explained. Higher order brackets can similarly be produced by combining subject, predicate, and object brackets. The bracketting procedure is generally performed without difficulty, and the substructures are almost always correctly recognized. In fact, many of the more trivial errors in the syntactic analysis are not reflected by any bracketting errors.

Following the bracketting procedure, it is easy to generate word groups of various kinds together with their associated functions. For example, the brackets of Table 5 will yield four noun phrases ("suitably scaled graph," "rising exponential curve," "hypnotic effect," "impending crisis"), two prepositional phrases ("on graph," "effect of crisis"), and one verb phrase ("tends to produce"). Similarly, a subject-verb-object grouping would yield, assuming proper recognition of the subject, the phrase "curve produce effect."

The output produced by the syntactic bracketting procedure can be incorporated in a word association map in two different ways. First, it is possible to replace each word associated with a node of the map by a complete phrase, and, second, relations between nodes, represented by branches in the map, can be provided with relationship indicators, such as prepositions, conjunctions, verbs, and so on, as determined by the syntactic analysis program. The replacement of individual words by phrases is relatively straightforward. The problem created by the addition of relational indications is more complicated, particularly for those relations which are indicated by the context rather than by specific function words [15]. However, even if a standard set of relational indications is not available, and relational information provided by the context is not explicitly identified, the addition of word groupings and of some simple indicators of relation (e.g., prepositions), furnishes a more accurate description, and permits a more profitable comparison of document content.

The Use of Bibliographic Citations for the Generation of Document Associations

In the preceding section, information was extracted from written texts to obtain word associations. It may be expected that when these association programs are added to

other methods for the identification of document content, similarities and differences between documents will be easier to detect. Another possible approach to the generation of document associations consists in using bibliographic references to determine document content. Specifically, if it could be shown that a similarity in the bibliographic references or citations attached to two documents also implies a similarity in subject matter, then citations might be helpful for the automatic assignment of index terms to new documents, and for the generation of document associations.

An experiment is described in the present section which uses citation tables (listing with each cited document the set of all those documents which cite the original ones) and reference tables (listing with each citing document the set of all documents cited by the original ones) to compute similarity coefficients between documents. The coefficient sets derived from the citations are then compared with another set of similarity coefficients derived from index terms attached to the documents, and an attempt is made to determine whether large coefficients in one set correspond to large coefficients in the other, and vice-versa [22].

The proposed use of citations exhibits the same advantages as the syntactic method previously described, namely that the required input information (in this case bibliographic references and citations) is directly available with most documents, and need not therefore be generated by more or less unreliable methods. Moreover, references can be processed automatically nearly as easily as ordinary running text. Use can also be made, at least for experimental purposes, of existing manually indexed collections for which the assignment of index terms is made under controlled conditions.

Measure of Similarity: Consider first a given collection of n documents. Let $\underline{X}^1, \underline{X}^2, \ldots \underline{X}^n$ be a collection of n vectors, such that each vector element $\underline{X}^i_j$ of a given vector $\underline{X}^i$ represents some property relating to the ith document. Specifically, if the n documents are characterized by m properties each, each document is represented by one m-dimensional vector. To relate the properties which identify two specific documents i and j, it is then only necessary to compare the elements of the two vectors $\underline{X}^i$ and $\underline{X}^j$; moreover, if each document is to be related to each other document in the collection, all possible vector pairs must be compared.

In most cases, it is convenient to restrict the values of the vector elements to 0 and 1, in such a way that element $\underline{X}^i_j = 1$ if and only if property j holds for document i, and $\underline{X}^i_j = 0$ otherwise. This restriction is, however, in no way essential. Two documents i and j are then assumed to be closely related if they share a number of common properties, or, alternatively, if the vectors $\underline{X}^i$ and $\underline{X}^j$ have their nonzero elements in corresponding positions. The desired measure of relatedness must also be a function of the total number of properties applying to each of the documents, that is, of the total number of nonzero elements in each vector, since one shared property out of a possible total of one hundred is clearly less significant than one out of a possible total of two.

A simple measure which exhibits the desired behavior and permits a comparison of magnitudes is obtained by considering the vectors of document properties as vectors in m-space, and by taking as a distance function the cosine of the angle between each vector pair. Specifically,

$$s = \cos(\underline{X}^i, \underline{X}^j) = \frac{\underline{X}^i \cdot \underline{X}^j}{|\underline{X}^i| \; |\underline{X}^j|}$$

$$= \frac{\sum_{k=1}^{m} \underline{X}^i_k \underline{X}^j_k}{\sqrt{\sum_{k=1}^{m} \left[(\underline{X}^i_k)^2\right]} \sqrt{\sum_{k=1}^{m} \left[(\underline{X}^j_k)^2\right]}}$$

If one or both of the vectors is identically equal to zero, s is defined as zero, following the intuitive notion that if a document has no identifiable properties it cannot share any properties with other documents.

If the vector elements take on values between 0 and 1, the angle between any two vectors cannot exceed ninety degrees; the value of s then ranges from 0 to 1. If, moreover, the document properties are represented by strictly logical vectors, the expression for s can be simplified since $(\underline{X}^i_j)^2$ is then equal to $\underline{X}^i_j$. In that case s is directly proportional to the number of common properties and inversely proportional to the total number of identifiable properties of each document.

Comparison Between Cited and Citing Documents: Consider now a given collection of n documents each of which is characterized by the property of being cited by one or more of the other documents in the same collection. The situation may be represented by a graph, as shown in Figure 4, in which each document, denoted by a code name, appears as the node of the graph, and arcs are used to interconnect cited and citing documents. The arrowheads point in the direction of the citing documents. The documents at the top of the graph are not cited by other documents, and those at the bottom do not cite any others; a partial ordering thus exists among the documents as pictured in the figure.

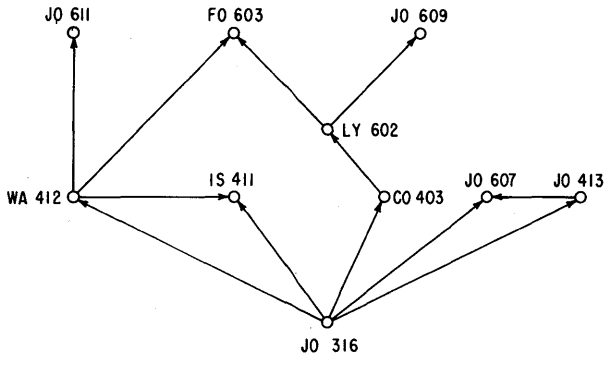
Let each document of the collection be represented by an n-dimensional logical vector $\underline{X}^i$, where $\underline{X}^i_j = 1$ if and only if document i is cited by document j, and $\underline{X}^i_j = 0$ otherwise. If these n vectors are arranged in rows one below the other, a square logical matrix is formed which completely exhibits the desired document properties. The n by n citation matrix $\underline{C}$ for the document set of Figure 4 is shown as an example in Figure 5. A dot represents the value 1, while a blank is used for 0. The number of dots in the row vectors of $\underline{C}$ represents the degree of "citedness" for the documents listed at the head of the rows, and corresponds therefore to the number of outgoing arrows from the document nodes of Figure 4. Similarly, the number of dots in the column vectors of $\underline{C}$ measures the amount of "citing" for the documents listed at the head of the columns, and corresponds therefore to the number of incoming arrows in the graph of Figure 4. Since the document collection of Figure 4 is closed, the set of row identifiers is the same as the set of column identifiers in Figure 5.

A measure of similarity between row (column) vectors can be obtained by calculating the s factor for each pair of rows (columns). The result of such a computation can be represented by a similarity matrix $\underline{R}$, such that $\underline{R}^i_j$ is the value of the similarity coefficient between the ith and jth rows (columns). Since the cosine of the angle between vectors i and j is the same as that between vectors j and i, the similarity matrix will be a square symmetric matrix. Specifically, if the original property matrix has n rows and m columns, an n by n row similarity matrix and an m by m column similarity matrix can be formed. The row similarity matrix $\underline{R}$ corresponding to the citation matrix $\underline{C}$ of Figure 5 is shown in Figure 6.

CITING \ CITED	C0 403	F0 603	IS 411	JO 316	JO 413	JO 607	JO 609	JO 611	LY 602	WA 412
C0 403									•	
F0 603										
IS 411										•
JO 316	•		•		•	•				•
JO 413						•				
JO 607										
JO 609										
JO 611										
LY 602		•					•			
WA 412		•	•					•		

MATRIX $\underline{C}$ EXHIBITING DIRECT CITATIONS

Figure 5



CITATION GRAPH FORMED BY TEN SELECTED DOCUMENTS

Figure 4.

	C0 403	F0 603	IS 411	JO 316	JO 413	JO 607	JO 609	JO 611	LY 602	WA 412
C0 403	1									
F0 603		1								
IS 411			1	$1/\sqrt{5}$						
JO 316			$1/\sqrt{5}$	1	$1/\sqrt{5}$				$1/\sqrt{15}$	
JO 413				$1/\sqrt{5}$	1					
JO 607						1				
JO 609							1			
JO 611								1	$1/\sqrt{6}$	
LY 602								$1/\sqrt{6}$	1	
WA 412				$1/\sqrt{15}$				$1/\sqrt{6}$	1	

$$\underline{R}^i_j = \frac{\sum_{k=1}^{m} \underline{C}^i_k \underline{C}^j_k}{\sqrt{\sum_{k=1}^{m} (\underline{C}^i_k)^2 \sum_{k=1}^{m} (\underline{C}^j_k)^2}}$$

ROW SIMILARITY MATRIX $\underline{R}$ CORRESPONDING TO CITATION MATRIX $\underline{C}$

Figure 6

From the definition of the similarity measure it is clear that the diagonal elements $\underline{R}_i^i$ of the row (column) similarity matrix $\underline{R}$ are equal to 0 or 1 depending on whether row $\underline{C}^i$ (column $\underline{C}_i$) of the citation matrix $\underline{C}$ was identically equal to 0 or not. Since the similarity coefficients between any row (column) and itself ought to be identical, the diagonal elements of a similarity matrix will be disregarded in further calculations. Moreover, because $\underline{R}$ is symmetric, only the coefficients which appear in the right triangular part of $\underline{R}$ need be considered for further processing.

The coefficients of $\underline{R}$ represent a measure of similarity between documents, based on the number of overlapping direct citations. This concept may be extended, by using as a basis for the calculation of correlation coefficients not the existence of direct links between documents (links of length one), but links of length two, three, four, or more. Consider, as an example, the graph of Figure 4. If document A cites document B, or B cites A, the corresponding nodes are directly linked by an arc. On the other hand, if A does not cite B, but A cites (or is cited by) C which in turn cites (or is cited by) B, no direct link exists between A and B. Indeed, A and B are then linked by a path of length two, since an extraneous document node C exists between nodes A and B in the graph of Figure 4. Similarly, if the path between two document nodes includes two extraneous nodes, they are linked by a path of length three, and so on.

Given a square citation matrix $\underline{C}$ it is possible by matrix multiplication to obtain matrices $\underline{C}'$, $\underline{C}''$, etc., exhibiting respectively the existence of paths of length two, three, and so on [23]. Specifically,

$$[\underline{C}']_j^i = \bigvee_{k=1}^{m} (\underline{C}_k^i \wedge \underline{C}_j^k),$$

$$[\underline{C}'']_j^i = \bigvee_{k=1}^{m} (\underline{C}_k^i \wedge \underline{C}'^k_j),$$

and so on. Boolean multiplication is used, since the new connection matrices $\underline{C}'$, $\underline{C}''$, etc., are again defined as logical matrices. $(\underline{C}')_j^i$ is then equal to 1 if and only if at least one path of length two exists between nodes i and j in the citation graph; otherwise, $(\underline{C}')_j^i$ is equal to 0. It may be noted that $\underline{C}'$, unlike $\underline{C}$, can have nonzero diagonal elements, corresponding to the case where two documents mutually cite each other. This situation arises for the document set of Figure 4, where IS 411 cites WA 412, and WA 412 cites IS 411.

As before, similarity matrices $\underline{R}', \underline{R}'', \ldots$ can be obtained from citation matrices $\underline{C}'$, $\underline{C}'', \ldots$ by computing the similarity coefficient s for all pairs of rows (columns) of the citation matrices. These new similarity matrices measure document similarity based on common citation links of length two, three, ..., and so on. While it is theoretically possible to work with similarity matrices $\underline{R}^n$ based on overlapping citation links of length n, it is likely that the similarity between subject matter and citations will diminish rapidly as the length of citation links increases. For present purposes, the investigation of overlapping citations is therefore restricted to the consideration of direct links and links of length two, three, and four.

Comparison Between Indexed Documents: In order to be able to evaluate document similarity based on content, it is necessary to choose some method for the identification of content. It would clearly be desirable to use for this purpose one or another of the existing automatic indexing methods. If this were done, the evaluation of the test results would be difficult to carry out, since it would then be impossible to distinguish a real discrepancy between overlapping citations and overlapping index terms, from a discrepancy due to an inadequate index set. In order to avoid such complications, it is preferable to use a manually prepared and controlled set of index terms at least for the initial experiments.

Given a set of index terms, it is convenient to display the assignment of terms to documents by means of a logical matrix $\underline{T}$, where $\underline{T}_j^i$ is set equal to 1 if and only if term i is assigned to document j and $\underline{T}_j^i$ is 0 otherwise. The row dimension n of $\underline{T}$ is then equal to the total number of documents and the column dimension m is equal to the number of distinct terms. A typical term-document matrix $\underline{T}$ for the previously used document set is shown in Figure 7.

A measure of document similarity based on the number of overlapping index terms between documents can now be obtained as before by using the cosine function s to com-

DOCUMENTS / TERMS	CO 403	FO 603	IS 411	JO 316	JO 413	JO 607	JO 609	JO 611	LY 602	WA 412
CDRUN				•	•	•				
CRECT							•			
LOKUP							•			
EDITG							•	•		
HADIC	•		•				•			
INFLE			•							•
RFVBS									•	
SORTG			•							
STRAN		•								
SUFAN				•						
SYNAR		•	•							
TRNSL		•							•	

TERM-DOCUMENT MATRIX $\underline{T}$ FOR DOCUMENT SET

Figure 7

pute elements $\underline{S}^i_j$ of an n by n symmetric similarity matrix $\underline{S}$. If the documents appear as column headings as in Figure 7, the similarity coefficients are obtained by matching pairs of columns rather than rows.

Since the term-document matrix $\underline{T}$ is not in general a square matrix, matrix multiplication cannot be used to obtain second order effects, similar to the citation links of length two or more. Instead, it is first necessary to compare the index terms by performing a row correlation of the rows of $\underline{T}$. This produces a new m by m symmetric correlation $\underline{T}^*$ which displays similarity between index terms. This matrix can be used to eliminate from the set of index terms those terms which exhibit a large number of joint occurrences with other terms. A reduced set of index terms can then be formed, and a new term-document matrix $\underline{T}'$ constructed, from which a new similarity matrix $\underline{S}'$ is formed. Higher order term similarity can of course be obtained if desired by squaring or cubing $\underline{T}^*$.

The last step needed in the testing procedure is a comparison between the document similarity coefficients obtained from the citations and the coefficients obtained from the index terms. Specifically, it is necessary to compare the elements of matrices $\underline{R}$, $\underline{R}'$, $\underline{R}''$, and so on, with the equivalent ones in $\underline{S}$ or $\underline{S}'$. Since the comparison of individual pairs of equivalent coefficients may not be very meaningful, one coefficient will be computed for each document by comparing equivalent document rows in $\underline{R}$ and $\underline{S}$. The previous measure may again be used for that purpose in the form

$$\underline{x}_i = \cos(\underline{R}^i, \underline{S}^i) = \frac{\sum_{k=1}^{n} \underline{R}^i_k \underline{S}^i_k}{\sqrt{\left(\sum_{k=1}^{n} (\underline{R}^i_k)^2\right)\left(\sum_{k=1}^{n} (\underline{S}^i_k)^2\right)}}$$

to obtain a "cross-correlation vector" $\underline{x}$. Each element of a cross-correlation vector is a measure of similarity for a given document, obtained by comparing similarity coefficients obtained from citations with the corresponding similarity coefficients obtained from index terms for that given document. Large values of the element $\underline{x}_i$ will indicate a close similarity between the measures obtained from citations and those obtained from index terms, since then nonzero terms in $\underline{R}^i$ will correspond to nonzero terms in $\underline{S}^i$. Small values of the element $\underline{x}_i$, on the other hand, will indicate no similarity in the two types of measures.

In addition to the cross-correlation vector, a single "over-all cross-correlation coefficient" can be obtained by considering each of the n by n similarity matrices to be compared, as a single vector of dimension n^2, and computing the similarity coefficient s for these two vectors. The "over-all" cross-correlation coefficient is then a similarity measure for the complete collection. The "over-all" coefficient will be large, if many of the cross-correlation vector elements are large, that is, if, for many documents, large similarity coefficients obtained from citations correspond to large similarity coefficients obtained from index terms.

Computer Experiment: A computer experiment carried out to test the system will now be described. A collection of sixty-two documents dealing with mathematical linguistics and machine translation was chosen. The writers of all documents were, at the

time of publication of their papers, staff members of the Computation Laboratory of Harvard University. While this fact may introduce a certain amount of bias in the experimental results, it was the only reasonable way to guarantee that an adequate number of citations would be found in a relatively small collection of documents.

A set of fifty-six index terms was used for manual indexing of the document collection. No particular effort was made to insure that the terms would be independent, nor was any limitation placed on the frequency with which a particular term could be used.

The two basic inputs used for the computer experiments were logical matrices of dimension 62 by 62 and 62 by 56, listing, respectively, cited versus citing documents, and documents versus terms. These two input matrices correspond in format to the examples of Figures 5 and 7. From the two logical input matrices, a set of three principal and six auxiliary similarity matrices was obtained by performing comparisons between pairs of rows or columns. The similarity matrices all correspond in format to the example of Figure 6. The CITED and CITING similarity matrices of dimension 62 by 62 were obtained from the original citation matrix by row and column comparisons, respectively. The TDCMP similarity matrix, also of dimension 62 by 62, was similarly obtained by column comparisons from the original term-document matrix. Additional citation similarity matrices, designated CTD2, CTD3, CTD4, and CNG2, CNG3, CNG4 were obtained from the squared, cubed, and fourth-power logical citation matrices, as previously explained.

Eight cross-correlation operations were performed by correlating each of the eight citation similarity matrices with the term-document similarity matrix TDCMP. Each cross-correlation operation produced a cross-correlation vector of dimension sixty-two. An over-all cross-correlation coefficient was also computed in each case.

The eight over-all cross-correlation coefficients obtained for the sample document collection are shown at the top of Table 7. The value of the over-all similarity coefficient first rises as the length of the citation links increases, and then drops again as the length of the links becomes still greater. A peak is reached for links of length 2. This behavior is due to two counteracting forces.

Table 7. Comparison of Overall Similarity Coefficients

	TYPE OF CITATION INPUT	OVERALL SIMILARITY COEFFICIENT	TOTAL NUMBER OF LINKS (62 DOCUMENTS)	NO. OF DOCUMENTS WITH ZERO LINKS	MAX. NUMBER OF LINKS FOR ANY DOCUMENT
DOCUMENT INPUT	CITED	0.281	165	13	16
	CTD 2	0.394	335	18	25
	CTD 3	0.392	452	21	36
	CTD 4	0.380	490	25	34
	CITNG	0.322	165	12	11
	CNG 2	0.400	335	16	20
	CNG 3	0.395	452	20	25
	CNG 4	0.357	490	27	28
RANDOM TREE INPUT	RTCTD	0.149	165	12	8
	RTCD 2	0.243	265	16	15
	RTCD 3	0.278	331	20	20
	RTCD 4	0.265	338	26	25
	RTCNG	0.159	165	13	7
	RTCN 2	0.175	265	22	17
	RTCN 3	0.174	331	28	30
	RTCN 4	0.170	338	30	30
RANDOM GRAPH INPUT	RMCTD	0.127	164	6	9
	RMCD 2	0.239	379	6	23
	RMCD 3	0.313	795	6	39
	RMCD 4	0.349	1461	6	51
	RMCNG	0.143	164	6	7
	RMCN 2	0.261	379	6	17
	RMCN 3	0.358	795	6	36
	RMCN 4	0.383	1461	6	50

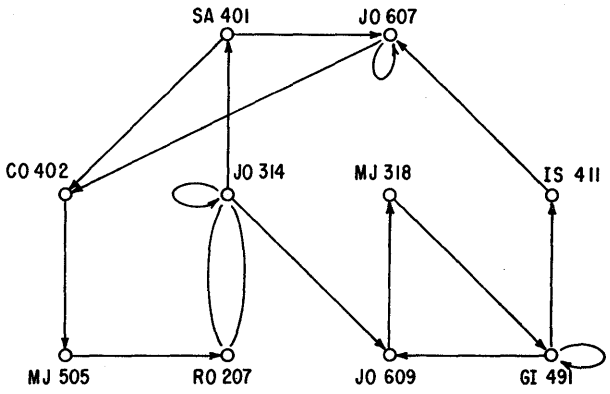
As the length of the links increases, the total number of links of any given length increases also because of the tree-like structure of the citation graph shown in the example of Figure 4. The actual data, shown in column 3 of Table 7 for the document set under consideration, demonstrate that an initial set of 165 links of length one is expanded into a set of 490 links of length four. An increased number of links results in a larger number of ones in the original logical citation matrix, and thus in a higher probability of overlapping ones and a larger over-all similarity coefficient.

However, as the length of the links increases, two factors also exist which tend to decrease the magnitude of the over-all similarity coefficient. First, given any document collection, the number of documents which exhibit citation links of length n, but which do not exhibit links of length greater than n increases as n becomes larger. The actual data for the sample document collection are shown in column 4 of Table 7. Thus more and more documents will exhibit individual similarity coefficients of zero value, since no citations can be found to perform the required comparisons. This tends to decrease the value of the over-all coefficient. Second, as the length of the links increases, and the citations thus become increasingly less accurate indications of document content, the

magnitude of the cross-correlation coefficients obtained from the citation matrix and the term-document matrix would be expected to decrease even for those documents for which a large number of citation links can still be found. It is clear that for the document collection under consideration the larger number of citation links first outweigh these other factors, as the length of the links increases from one to two. Thereafter, the larger number of documents with no citation links of the proper length and the greater change in document content tend to outweigh the increase in the total number of links.

In order to test the significance of the values obtained for the over-all cross-correlation coefficients, the experiment was repeated with a "random-tree" input. The random-tree input consisted of a partially ordered system of the same type as that shown in Figure 4. However, this time, the fictitious "documents" were assigned to the nodes at random; similarly the assignment of citations and of index terms to the "documents" was randomized. The "random-tree" input simulates very closely the conditions which obtain for the actual document collection, in that the total number of links increases, as before, with the length of the links, as does the number of "documents" with a zero similarity coefficient. Nevertheless, the values of the over-all cross-correlation coefficients exhibited in the middle section of Table 7 are about fifty percent smaller for the random-tree input than they are for the actual document collection. The differences can only be due to the fact that the comparison of citation similarities with index term similarities is meaningless for the random-tree input, because of the random assignment of citations and index terms, whereas there do, in fact, exist similarities for the actual document collection.

The analysis was carried still further by using as input an arbitrary abstract graph which no longer preserves the partial ordering typical of a document collection. In such a graph any "document" can cite or be cited by any other "document," including itself. A typical graph is shown in Figure 8. The assignment of citations and of index-term to the "documents" was again randomized, and the data in the lower portion of Table 7 show that for the comparison with direct links (RMCTD and RMCNG), the values of the over-all cross-correlation coefficients are even lower for the random-graph, than they are for the random-tree input. Since the total number of links increases very fast for the random graph, as the length of the links increases, and the number of "documents" with zero links is very small, the values of the over-all coefficients increase approximately in proportion to the length of the links. However, even though the number of links of length four is about three times larger for the random graph input, than it is for the actual document collection, and the number of documents with zero links is four times smaller, the over-all cross-correlation coefficients are still only approximately equal in the two cases. Again the differences must be due to the meaningful similarities between citation input and index term input which hold only for the actual document collection.

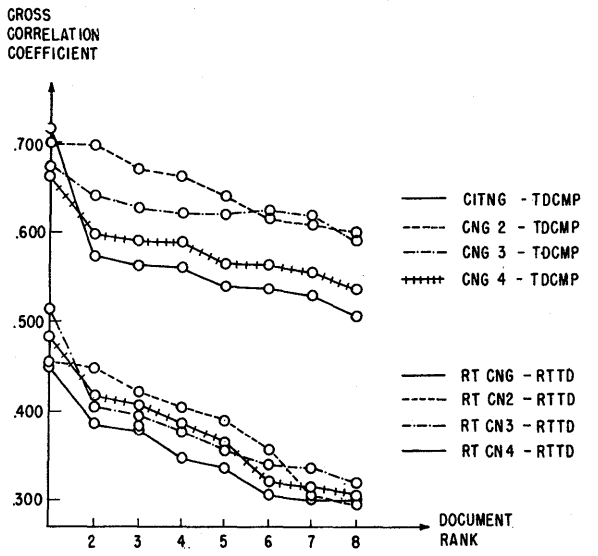


RANDOM GRAPH FOR TEN SELECTED DOCUMENTS

Figure 8

In order to determine how close a similarity is observed between document content and citations, it is not enough to look at the over-all cross-correlation coefficient for the whole collection, but it is necessary also to examine in more detail the values of the cross-correlation coefficients for the individual documents. The values of the eight largest cross-correlation vector elements obtained for each of four document cross-correlations, and for each of four random-tree cross-correlations are shown in Figure 9. The documents corresponding to the various coefficients shown are not necessarily the same in each case; however, a comparison of the magnitudes reveals that the individual cross-correlation elements

are, as a whole, much larger for the cross-correlation operations involving the document collection, than they are for the random-tree input. This observation confirms the results previously given for the over-all cross-correlation coefficient. The largest maximum values are again obtained for citation links of length two.



COMPARISON OF MAXIMUM CROSS-CORRELATION COEFFICIENTS

Figure 9

Consider now the sample list of cross-correlation coefficients shown in Tables 8, 9, and 10, for early documents published in 1957-1958, for middle-range documents published in 1959-1960, and for recent documents published in 1961, respectively. It is seen in Table 8 that the large values of the coefficients for the early documents occur as expected by comparing cited documents. Similarly, Table 10 shows that large values of the coefficients for the recent documents

Table 8. Typical Cross-Correlation Coefficients for Early Documents (Published 1957-1958)

COEFFICIENTS / DOCUMENTS	TDCMP CITNG	TDCMP CNG 2	TDCMP CNG 3	TDCMP CNG 4	TDCMP CITED	TDCMP CTD 2	TDCMP CTD 3	TDCMP CTD 4
CO 208	.000	.000	.000	.000	.455	.587	.610	.635
FO 496	.299	.459	.000	.000	.000	.000	.000	.000
FO 506	.000	.000	.000	.000	.217	.496	.487	.552
FR 205	.436	.483	.559	.563	.433	.542	.628	.615
GI 209	.109	.134	.306	.000	.424	.428	.447	.426
GI 491	.000	.000	.000	.000	.292	.615	.643	.622
GI 495	.273	.000	.000	.000	.210	.469	.523	.556

Table 9. Typical Cross-Correlation Coefficients for Middle-Range Documents (Published 1959-1960)

COEFFICIENTS / DOCUMENTS	TDCMP CITNG	TDCMP CNG 2	TDCMP CNG 3	TDCMP CNG 4	TDCMP CITED	TDCMP CTD 2	TDCMP CTD 3	TDCMP CTD 4
BO 319	.575	.702	.678	.675	.308	.494	.472	.498
BO 407	.438	.549	.593	.490	.411	.456	.000	.000
BO 408	.430	.489	.464	.000	.463	.424	.335	.000
CO 305	.723	.376	.342	.317	.378	.491	.483	.430
FR 406	.000	.000	.000	.000	.250	.286	.334	.000
GE 302	.326	.529	.000	.000	.372	.572	.552	.528
GR 313	.325	.370	.388	.000	.467	.520	.488	.445

Table 10. Typical Cross-Correlation Coefficients for Recent Documents (Published 1961)

COEFFICIENTS / DOCUMENTS	TDCMP CITNG	TDCMP CNG 2	TDCMP CNG 3	TDCMP CNG 4	TDCMP CITED	TDCMP CTD 2	TDCMP CTD 3	TDCMP CTD 4
FO 603	.077	.416	.404	.433	.000	.000	.000	.000
FR 612	.481	.526	.505	.478	.000	.000	.000	.000
GR 604	.142	.285	.338	.279	.000	.000	.000	.000
JO 607	.572	.605	.510	.443	.000	.000	.000	.000
JO 609	.362	.411	.485	.452	.000	.000	.000	.000
JO 611	.000	.000	.172	.196	.000	.000	.000	.000
KU 605	.133	.207	.264	.274	.000	.000	.000	.000

occur as expected by comparing citing documents. These data confirm the expected fact that in a closed collection the amount of citing is not a good content indicator for early documents, and the citedness is not a good indicator for recent documents. Table 9 shows that for those documents which can both cite and be cited, the maximum values of the coefficients occur sometimes by comparing cited documents and sometimes by using citing documents. The data of Table 9 thus confirm those previously exhibited in Table 7, to the effect that no criterion was found to exist for consistently preferring either the amount of citing or the amount of being cited as a content indication.

Turning now to the actual values of the coefficients obtained, it appears that for the majority of the documents the values of the cross-correlation coefficients lie between 0.450 and 0.700, when citation links of length two are used, indicating a substantial similarity between overlapping citations and overlapping index terms. For some documents, the value of the cross-correlation coefficient for links of length two was found to be smaller than 0.400. However, in almost every case the failure to obtain adequate agreement between citations and index terms was due to the total, or almost total, lack of citations. The exact data for 34 documents published in 1959-1960 are shown in Table 11.

The documents with coefficients below 0.400 are listed explicitly together with the number of citation links in each case. The following results are apparent for the correlations between CNG 2 - TDCMP and CTD 2 - TDCMP respectively:

a) number of documents with coefficient above 0.400, exhibiting substantial agreement between overlapping citations and index terms — 23 and 20;
b) number of documents with coefficient below 0.400, exhibiting fewer than three citation links of length one or two — 11 and 13;
c) number of documents with coefficient below 0.400 exhibiting three or more citation links — 1 and 1.

While there exists no scientific reason for asserting that a threshold value of 0.400 for the cross-correlation coefficients is necessarily significant as a similarity indication, it does appear that for nearly all documents which exhibit a reasonable number of citations (more than two), the coefficients obtained reveal considerable agreement between overlapping citations and overlapping index terms. Further experimentation is needed to evaluate more precisely the significance of the numeric results, and to determine to what extent citations can actually be used for the automatic generation of index terms.

CONCLUSION

Two experiments have been described which use information directly provided with written texts to determine associations between words and documents. The syntactic experiment has shown that it is possible to extract information from function words and word suffixes to generate word groups and relations between word groups. These, in turn, may be used to obtain a more effective method for comparing the content of documents.

The citation experiment has shown that for documents which exhibit an adequate number of citations, similarities in citations seem to provide some indication of similarities in content. For early documents, citedness furnishes a better indication than the amount of citing, and vice-versa for recent documents; for documents which can both cite and be cited, equally good indications were obtained by comparing citing and cited documents.

Table 11. Range of Coefficients for 34 Middle-Range Documents

TYPE OF CROSS-CORRELATION	RANGE OF COEFFICIENTS	NUMBER OF DOCUMENTS	SELECTED DOCUMENT CODES	NUMBER OF LINKS OF LENGTH 2 (AND NO. OF DIRECT LINKS)
CNG 2 TDCMP	.600 OR OVER	7		
	.500-.599	8		
	.400-.499	8		
	.300-.399	3	CO 305	11 (2)
			GR 313	2 (1)
			JO 314	6 (4)
	.200-.299	1	MA 301	5 (2)
	.100-.199	0		
	.001-.099	0		
	.000	7	CO 402	3 (1)
			FR 406	0 (0)
			IS 312	0 (0)
			LY 414	0 (0)
			MG 414	0 (0)
			MJ 306	0 (0)
			MJ 318	0 (0)
CTD 2 TDCMP	.600 OR OVER	1		
	.500-.599	7		
	.400-.499	12		
	.300-.399	3	CO 317	1 (2)
			FO 415	1 (2)
			MJ 318	2 (1)
	.200-.299	5	FR 309	7 (2)
			FR 405	1 (1)
			FR 406	1 (1)
			MA 301	3 (3)
			MG 404	1 (1)
	.100-.199	0		
	.001-.099	0		
	.000	6	CO 402	0 (0)
			IS 311	0 (1)
			JO 413	0 (1)
			LY 414	0 (2)
			SA 401	0 (0)
			VC 307	0 (0)

It is suggested that methods of this type, which require neither extensive input dictionaries nor complicated computer processes, may eventually form the basis for practical automatic programs for content analysis. While no simple method will be completely successful on its own, a combination of many techniques of the kind described may well offer not only a speedier, but also a more satisfactory, solution to the content analysis problem than the construction of semantic encyclopedias which are so far away in the future.

SUMMARY

The solution of most problems in automatic information dissemination and retrieval is dependent on the availability of methods for the automatic analysis of information content. In most proposed automatic systems, this analysis is based on a counting procedure which uses the frequency of occurrence of certain words or word classes to generate sets of index terms, to prepare automatic abstracts or extracts, to deter-

mine certain word groupings, and to extend or modify in various ways sets of terms originally given. Unfortunately, it is not possible to perform completely effective subject analyses solely by frequency counting techniques.

Two automatic methods are presented to aid in an effective subject analysis. The first makes use of a simplified form of syntactic analysis to determine associations between words in a text, and the second uses bibliographic citations to classify documents into subject areas. Neither method requires extensive dictionaries or tables of the type normally used for automatic classification schemes; instead, information is extracted from certain function words, from suffixes provided with many words in the language, and from bibliographic citations already available with most documents.

Specifically, the syntactic analysis makes use of a small dictionary of a few hundred function words such as prepositions, conjunctions, articles, and certain nouns. Word suffixes are then isolated, and a suffix table is used to obtain additional grammatical indicators. A type of predictive analysis is then used to assign syntactic function indicators to all words in a sentence by matching predicted syntactic structures against the available grammatical information for the various words. If no grammatical information is available, the most likely prediction is used to classify the given word. The syntactic function indicators are used to group words into phrases of certain types, and phrases into clauses, and to determine certain word associations. Experimental evidence indicates that the error rate is not substantially higher than that found in other more complicated syntactic analysis programs which require full syntactic word dictionaries.

The citation matching program uses bibliographic citations to determine document similarities. A similarity coefficient is first calculated for all document pairs as a function of the number of overlapping citations between them. A second similarity coefficient is then derived using this time the number of overlapping index terms as a criterion. The index terms may be generated by hand, or may be derived by means of word frequency analyses. Finally, similarity coefficients derived from overlapping citations are compared with those derived from overlapping index terms.

The coefficients, computed for a sample document collection, are analyzed to verify the hypothesis that when a closeness exists in the subject matter of certain documents, as reflected by overlapping index terms, there exists a corresponding closeness in the citation sets. It is found that the computed similarity coefficients are much larger than those obtained by assuming a random assignment of citations and index terms. Suggestions are made for using citation sets as an aid to the automatic generation of index terms.

REFERENCES

1. H. P. Luhn, Auto-encoding of Documents for Information Retrieval Systems, IBM Corporation, ASDD Report, 1958.
2. H. P. Luhn, "Automatic Creation of Literature Abstracts," IBM Journal of Research and Development, Vol. 2, No. 2, April 1958.
3. H. P. Luhn, "The Automatic Derivation of Information Retrieval Encodements from Machine-Readable Texts," in Information Retrieval and Machine Translation, Part 2, A. Kent, ed., Interscience Publishers, New York, 1961.
4. L. B. Doyle, Indexing and Abstracting by Association, Report SP - 718/001/00, System Development Corporation, April 1962.
5. H. Borko, "The Construction of an Empirically Based Mathematically Derived Classification System," Proceedings of the AFIPS Spring Computer Conference, San Francisco, May 1962.
6. L. B. Doyle, "Semantic Road Maps for Literature Searchers," Journal of the Association for Computing Machinery, Vol. 8, No. 4, October 1961.
7. V. E. Giuliano, "A Linear Method for Word Sentence Association," private communication.
8. B. F. Green, Jr., A. K. Wolf, C. Chomsky, and K. Laughery, "Baseball: An Automatic Question Answerer," Proceedings WJCC, Los Angeles, May 1961.
9. R. K. Lindsay, "The Reading Machine Problem," Doctoral Thesis, Carnegie Institute of Technology, September 1960.
10. N. S. Prywes and H. J. Gray, Jr., "A Report on the Development of a List

Type Processor - I," University of Pennsylvania, Moore School of Electrical Engineering, 1961.

11. M. Detant and A. Leroy, "Elaboration d'un Programme d'Analyse de la Signification," Rapport GRISA No. 11, EURATOM - CETIS, June 1961.
12. P. J. Stone, R. F. Bales, J. Z. Namenwirth, and D. M. Ogilvie, "The General Inquirer: A Computer System for Content Analysis and Retrieval based on the Sentence as a Unit for Information," Laboratory of Social Relations, Harvard University, November 1961.
13. P. Baxendale, "An Empirical Model for Machine Indexing," Third Institute on Information Storage and Retrieval, February 1961.
14. G. Salton, "The Manipulation of Trees in Information Retrieval," Communications of the Association for Computing Machinery, Vol. 5, No. 2, February 1962.
15. G. Salton, "The Identification of Document Content: A Problem in Automatic Information Retrieval," Proceedings of a Harvard Symposium on Digital Computers and their Applications, Annals of the Computation Laboratory of Harvard University, Vol. 31 (to be published, 1962).
16. S. Klein and R. F. Simmons, "Automatic Analysis and Coding of English Grammar for Information Processing Systems," Report SP-490, System Development Corporation February 1962.
17. S. Klein and R. F. Simmons, "A Computational Approach to Grammatical Coding of English Words," Report SP - 701, System Development Corporation, February 1962.
18. I. Rhodes, "A New Approach to the Mechanical Translation of Russian," National Bureau of Standards, Report No. 6295, 1959.
19. M. Sherry, "Comprehensive Report on Predictive Syntactic Analysis," Mathematical Linguistics and Automatic Translation, Report No. NSF - 7, Section I, Harvard Computation Laboratory, 1961.
20. W. Plath, "Automatic Sentence Diagramming," First National Conference on Machine Translation and Applied Language Analysis, National Physical Laboratory, Teddington, 1961.
21. D. G. Hays, "Grouping and Dependency Theories," Report P - 1910, Rand Corporation, Santa Monica, 1960.
22. G. Salton, "The Use of Citations as an Aid to Automatic Content Analysis," Information Storage and Retrieval, Report ISR - 2, Harvard Computation Laboratory, in preparation, June 1962.
23. F. E. Hohn, S. Seshu, and D. D. Aufenkamp, "The Theory of Nets," IRE Transactions on Electronic Computers, Vol. EC - 6, 1957, pp. 154-161.

A LOGIC DESIGN TRANSLATOR

D. F. Gorman and J. P. Anderson
Burroughs Corporation
Burroughs Laboratories
Paoli, Pennsylvania

INTRODUCTION

The process of logic design of a computer is analogous to programming, using hardware for commands. Logic designers must frequently take imprecise narrative descriptions of computer systems, and, applying experience, ingenuity, inventiveness, and considerable perseverance, transform the description into a prescription for a running system. This process takes an inordinate amount of time. Considerable attention has been devoted to the reduction of programming time through the use of higher programming languages, such as ALGOL and COBOL, and conversion to machine code through translators. In a similar manner, translation of higher languages for logic and system design would greatly reduce the effort now needed to specify and design computer systems.

Just as in the case of programming, changes to the description are tolerated up to a certain point, then are prohibited, or are made with great reluctance and less than perfect efficiency. This paper offers systems and logic designers a means to automate the repetitive and otherwise error-prone detail associated with much machine design, as well as to provide a means for making systems changes acceptable for a longer period of time, without encountering the logic design equivalent of program patches. An additional motivation is to be able to rapidly obtain a canonical description of a computer system in the form of application equations of all registers within the machine.

It is envisioned that a translator such as this would be incorporated within a complete design automation system, and, as such, should be expected to provide input for minimization programs, card layout and assignment programs, backplane wiring programs, and the like. For this reason, a canonical form is essential to a completely automated system and, in addition, will provide a convenient input form for follow-up programs.

From the canonical form, it is possible to obtain an estimate of the relative cost of a system, based upon component costs. As a result, the cost and effects of modifications and changes can be quickly and accurately determined. The canonical form also provides a means of comparing and evaluating various designs by examining the hardware structure. Previously, such comparisons could not be made, because of the prohibitive number of man-hours involved in the detailed logical design. Thus, those operations which are difficult to implement, in terms of complexity or cost, are easily pinpointed for further study. Logical inconsistencies and timing conflicts are eliminated. Thus, the translator will provide a tool for system designers to more accurately measure their systems, and, hopefully, will promote better descriptions of future systems.

System Description Language

A system can be described by giving the functional interrelationships of the various

parts of the system. The description is usually presented in a narrative form, with the drawbacks of ambiguities and otherwise imprecise meanings. Upon examining the notion of system design, as opposed to logic design, it is often found that the emphasis is upon the control structure of a processor, with details on the number of registers to be included, the structure of each, the manner of connection in each case, and the parts to be played by each in the execution of various commands. Particular attention is paid to the selection and description of a command list, especially those commands which are the "particular features" of a machine. These considerations suggested that a language that was a concise and unambiguous description of these objects would, in fact, describe the computer system.

The system descriptive language devised is based upon the informal programmatic notation frequently used to replace clumsy narrative [1], such as replacing the statement

> The contents of A and X are interchanged; the contents of A are then stored in memory at the place addressed by the contents of MA.

by

EXCHANGE A AND X
A → MA*

In general, the language assumes the existence of such hardware entities as counters, adders, registers, clocking systems, and the like. The forms of the language are not unlike constructs found in algebraic languages such as ALGOL [2].

The basic language construct devised to describe the interaction of the various registers is a statement. The principal statement type is associated with interregister transfers. In addition, counting statements, conditional statements, shift statements, memory access statements, and the like are provided for the most frequently invoked system functions.

Examples of some of the various statements are shown below:

Type	Example
transfer	A → B
conditional	IF G(13) = 0 THEN (5.G → B) ELSE (6.B → G)
shift	A MOVE RIGHT OFF 6 → B
arithmetic	A + B → R
counting	C + 1 → C
subroutine	INSTFETCH
memory access	MA* → L

Statements are freely combined to form functional descriptions of instructions and control. These larger constructs are the microsequences used to describe the machine functions. The complete functional description of a computer system is given by the set of microsequences for the instruction set and control (such as, instruction fetch and data fetch), plus the declarations describing the register sizes, and their interconnections.

As an example, the following is a microsequence for the acquisition of the next instruction in some machine:

INSTFETCH
1. PC → MA
2. MA* → B
3. PC + 1 → PC

Declarations are used to specify details of the system. They are used to name registers and describe their size and structure as well as to provide characteristics of various equipment assumed, such as arithmetic units, logical units, etc. In addition, declarations describe permissible data paths in the system as a check on the transfers specified in the microsequences. In general, the microsequences make reference to substructure names for the apparent reason of readability and mnemonic value. As a canonical representation, however, it was deemed desirable to use the parent name of a register, since, in general, the substructure is artificially imposed by the system designer and has no a priori meaning in hardware. Typical of the register declarations is the example of an instruction register description in Figure 1.

It is the intention to draw upon the advances achieved in recent years in the design of system elements, such as adders, counters, comparators, etc., by providing libraries of designs of these specialized units. These designs would reflect various compromises between speed and cost, and would represent diversified implementations of computational algorithms. These units are declared as special registers, and have a format similar

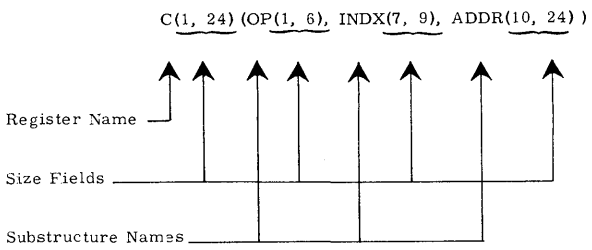


Figure 1. An Instruction Register Description.

to storage registers, differing only by the inclusion of cost and speed parameters.

Translator Structure

The overall structure of the translator is that shown in Figure 2.

Briefly, the translator accepts systems descriptions in the language provided, and transforms the microsequences into an intermediate language known as the design table. After suitable manipulation, the design table may be converted to application equations. The translator is coded as a set of recursive procedures, using the Burroughs 220 ALGOL translator [3]. Each procedure corresponds (more or less) to one of the syntactic definitions of the language. In addition, since the language unit most frequently invoked is a statement, a procedure is provided to array statement elements into a vector, and associate with them a type designator (for example, an identifier, a number, an operator, a delimiter, etc.). A set of link list procedures is used to handle the associative storage for the register and transfer lists, and for various other lists used in the translation process.

The logic design translator has three major parts: the scan and recognizer, the design table analysis, and the equation generator.

The Scan and Recognizer: This segment of the translator takes microsequences, in the system descriptive language, converts them into a series of information transfers, and enters them into the design table. As mentioned above, the design table can be thought of as an intermediate language used during the translation process as a convenient form in which to store and manipulate data. The design table is represented as an m × 12 matrix, where m is the number of operations to be performed within a microsequence and the 12 columns indicate the following information:

Column 1 - source register
2 - most significant bit of the field of the source register under consideration
3 - least significant bit of the field of the source register under consideration
4 - destination register

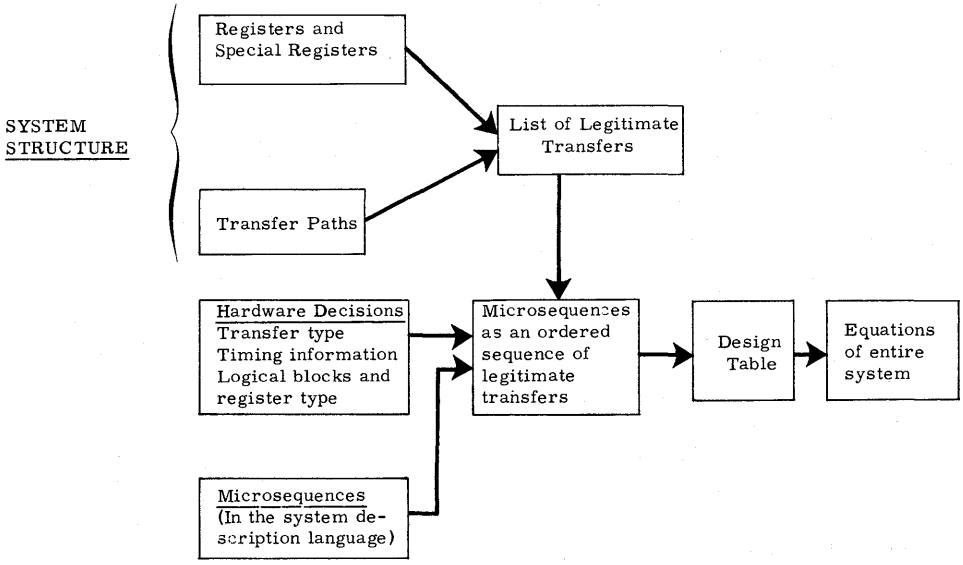


Figure 2. A Logic Design Translator.

Column 5 - most significant bit of the field of the destination register under consideration
6 - least significant bit of the field of the destination register under consideration
7 - equipment used for the transfer of information
8 - most significant bit of the field of the equipment under consideration
9 - least significant bit of the field of the equipment under consideration
10 - control conditions to be satisfied during this operation
11 - relative time at which this microstep is to occur
12 - delay of the destination register

The design table is set up to accommodate information transfers; that is, its basic structure embodies a source and a destination of information, provisions for indicating other equipment which may be used, and the control conditions which must be fulfilled for each transfer. That all routines to be performed by the specified machine can be decomposed into a series of information transfers is fundamental in the design of digital computers.

Each microstep must be individually analyzed in the order of its appearance within the microsequence. The microstep is first examined by the recognizer in order to determine which type of statement it is. At the present time, 12 basic statement types have been defined within the system. Depending upon the statement type, a subroutine is chosen which scans the statement, determining which registers or substructures of registers of the machine are invoked by this microstep. In general, substructure names are used in the microsteps, mainly for their mnemonic value. These substructure names are now replaced by the name of the parent register along with the appropriate bit fields. It is at this time that the basic statement is broken down into a series of information transfers between the specified registers.

The parent register names and bit fields are now entered into the design table in the "source" and "destination" columns, each transfer in a separate row. Since each information transfer between registers may use other equipment in the system (busses, memory address register, etc.), these transfers are compared to the list of data paths which was initially entered as declarations concerning the system, and all additional equipment which is used in this transfer is entered in the "equipment used" column of the design table. The operators appearing in the statement are entered in the "control" column. Flags, to be used in the design table analysis, are entered in the "time" column, if needed. A "0" flag is used to indicate that the following row is part of the operation; thus, the two rows are to be considered as one. A "1" flag indicates that the design table is partitioned below that row.

The Design Table Analysis: This section of the translator determines the length of time each operation will take, the time when it may begin, and sets up the control for the next instruction fetch.

The type of transfer associated with each operation in the design table is determined by the destination register. If the destination register accepts information in parallel, one delay will be required for the transfer. If a serial transfer is required, then the number of delays required for the operation is equal to the number of bits in the register involved. A delay is the time that the destination register requires to process one bit of information.

Since the GO TO statement effectively alters the order in which operations are to be performed in the machine, the design table must be partitioned so that the timing of each branch of the program may be determined independently.

The clock time at which an operation may begin is determined as follows:

1. The first operation in the design table can begin at the first clock pulse, denoted by t_1.
2. The first operation in any partition of the design table can begin at t_{max+1}, where t_{max} is the highest clock time assigned to any previous operation in the design table.
3. The clock time of the n^{th} operation is determined by comparing all registers and other equipment used by an operation against previous operations, within the same partition, beginning with the $(n-1)^{st}$ operation, until the conflict with the maximum time is found. A conflict is said to occur between two operations if any equipment is used in common by both operations. The n^{th}

operation is then begun at a time one greater than the time of the conflicting operation with the highest time. If no conflict is found within the partition, the starting time of the first operation within the partition is also the starting time of the n^{th} operation.

This method of assigning clock times to operations ensures that, for the order of microsteps as declared, equipment does not remain idle needlessly.

An implicit subroutine, called microsequence completion, must be added to each microsequence so that, as each microsequence is executed, control is transferred to the following microsequence. This is accomplished by entering in the $(i + 1)^{st}$ row of the design table the following:

a) "0" in column 1
b) CLOCK in column 4
c) RESET in column 10
d) t_{max+1} in column 11
e) $Delay_{CLOCK}$ in column 12

The Equation Generator: After the operation of the design table analysis, the application equations for all the data storage registers in the machine are generated. Each operation in the design table will form a term of the application equation for each bit of the destination register. This term is formed by taking the conjunction of the microsequence name, the corresponding bit of the source register, all variables in the control column, and the clock time. After all the microsequences to be performed by the specified machine have been processed, the total equation for a particular bit of a register is formed in a separate sort run by taking the disjunction of all terms in which the bit under consideration was the destination bit.

Likewise, the application equations for the control flip-flops are derived from the design table based upon the microsequence name and the clock times associated with each control variable.

An Example

Two microsequences will now be used to illustrate the functioning of the translator. These microsequences do not represent a useful routine in a computer system but were constructed to illustrate at least one example of each type of statement. Assume that all information transfers occur in parallel. The resulting design tables are shown in Figs. 3 and 5. The two microsequences are as follows:

```
MICROSEQUENCE
   WIN
   1. P → Q
   2. K + 1 → P
   3. S + T → W
   4. X MOVE RIGHT · OFF 3 → X
   5. IF                K = P
      THEN              (6. W → X)
      ELSE              (7. W → Y
                         8. R6-10 → W2-6 )
   9. K - 1 → P

MICROSEQUENCE
   LOSE
   1. (K · W ∨ R) → X
   2. WIN
   3. M* → T
   4. EXCHANGE S AND T
   5. T → M*
   6. GO TO 2
```

For microsequence WIN, microsteps 1 and 2 are straightforward entries into the table. The fact that BUS1 was used during these operations was determined by consulting declarations about the system assumed to have been previously entered via the translator. In microstep 3, row 7, the 0 in column 11 is used as a flag to indicate that both operands (S and T) must be brought to the arithmetic unit at the same time. Microstep 5 is a conditional statement and, as such, imposes control conditions upon following statements, as indicated by the FF1 and ~FF1 entries in column 10.

Microsequence LOSE illustrates some different statement types and another feature of the translator. In evaluating the Boolean expression in microstep 1, the translator finds that temporary storage is necessary. The translator assigns a name to this register (TEMP 1) and notifies the designer that this register is necessary if the microstep is to be executed in the prescribed manner.

The analysis section of the translator determines the clock times for the operations indicated in the design table. Since parallel transfers were assumed, only one delay will be required for each transfer; the results are shown in Figures 4 and 6. In accord with the flags in column 11 of microsequence WIN, the beginning times for the operations of rows 1 and 2, 3 and 4, etc., are the same. An interesting situation occurs in rows 19,

Microstep Number k	Row i	Source Register 1	Begin 2	End 3	Destination Register 4	Begin 5	End 6	Eq. Used 7	Begin 8	End 9	Control 10	Time 11	Delay 12
1	1	P	1	n	BUS1	1	n					0	0
	2	BUS1	1	n	Q	1	n						1
2	3	K	1	n	BUS1	1	n					0	0
	4	BUS1	1	n	COUNT	1	n				UP		1
	5	COUNT	1	n	BUS1	1	n					0	0
	6	BUS1	1	n	P	1	n						1
3	7	S	1	n	ARITH (A)	1	n				ADD	0	5
	8	T	1	n	ARITH (B)	1	n				ADD		5
	9	ARITH	1	n	W	1	n						1
4	10	X	1	n	SHIFT	1	n				$\overline{\text{RIGHT}}\cdot\overline{\text{OFF}}$		1
	11	SHIFT	1	n	SHIFT	1	n				RIGHT·OFF		1
	12	SHIFT	1	n	SHIFT	1	n				RIGHT·OFF		1
	13	SHIFT	1	n	X	1	n						1
5	14	K	1	n	BUS1	1	n					0	0
	15	BUS1	1	n	LOGICAL (A)	1	n				EQL	0	3
	16	P	1	n	BUS2	1	n					0	0
	17	BUS2	1	n	LOGICAL (B)	1	n				EQL		3
	18	LOGICAL	1	1	FF1	1	1						1
6	19	W	1	n	X	1	n	FF1	1	1	FF1		1
7	20	W	1	n	Y	1	n	FF1	1	1	~ FF1		1
8	21	R	6	10	W	2	6	FF1	1	1	~ FF1		1
9	22	K	1	n	BUS1	1	n					0	0
	23	BUS1	1	n	COUNT	1	n				DOWN		1
	24	COUNT	1	n	BUS1	1	n					0	0
	25	BUS1	1	n	P	1	n						1

Figure 3. Design Table for Microsequence WIN.

Microstep Number k	Row i	Source Register 1	Begin 2	End 3	Destination Register 4	Begin 5	End 6	Eq. Used 7	Begin 8	End 9	Control 10	Time 11	Delay 12
1	1	P	1	n	BUS1	1	n					1	0
	2	BUS1	1	n	Q	1	n					1	1
2	3	K	1	n	BUS1	1	n					2	0
	4	BUS1	1	n	COUNT	1	n				UP	2	1
	5	COUNT	1	n	BUS1	1	n					3	0
	6	BUS1	1	n	P	1	n					3	1
3	7	S	1	n	ARITH (A)	1	n				ADD	1	5
	8	T	1	n	ARITH (B)	1	n				ADD	1	5
	9	ARITH	1	n	W	1	n					6	1
4	10	X	1	n	SHIFT	1	n				RIGHT·OFF	1	1
	11	SHIFT	1	n	SHIFT	1	n				RIGHT·OFF	2	1
	12	SHIFT	1	n	SHIFT	1	n				RIGHT·OFF	3	1
	13	SHIFT	1	n	X	1	n					4	1
5	14	K	1	n	BUS1	1	n					4	0
	15	BUS1	1	n	LOGICAL (A)	1	n				EQL	4	3
	16	P	1	n	BUS2	1	n					4	0
	17	BUS2	1	n	LOGICAL (B)	1	n				EQL	4	3
	18	LOGICAL	1	1	FF1	1	1					7	1
6	19	W	1	n	X	1	n	FF1	1	1	FF1	8	1
7	20	W	1	n	Y	1	n	FF1	1	1	~FF1	8	1
8	21	R	6	10	W	2	6	FF1	1	1	~FF1	9	1
9	22	K	1	n	BUS1	1	n					4	0
	23	BUS1	1	n	COUNT	1	n				DOWN	4	1
	24	COUNT	1	n	BUS1	1	n					7	0
	25	BUS1	1	n	P	1	n					7	1
999	26	"0"	-	-	CLOCK	-	-				RESET	10	1

Figure 4. Design Table for Microsequence WIN after Timing Analysis.

Microstep Number k	Row i	Source Register 1	Begin 2	End 3	Destination Register 4	Begin 5	End 6	Eq. Used 7	Begin 8	End 9	Control 10	Time 11	Delay 12
1	1	K	1	n	BUS1	1	n					0	0
	2	BUS1	1	n	LOGICAL (A)	1	n				AND	0	3
	3	W	1	n	LOGICAL (B)	1	n				AND		3
	4	LOGICAL	1	n	TEMP1	1	n						1
	5	TEMP1	1	n	LOGICAL (A)	1	n				OR	0	3
	6	R	1	n	LOGICAL (B)	1	n				OR		3
	7	LOGICAL	1	n	X	1	n					1	1
2	8												
	.												
	.	The entire microsequence WIN is entered here.											
	.												
	33												
3	34	M	1	m	MEM (ADD)	1	m					0	1
	35	MEM (DATA)	1	n	T	1	n						1
4	36	S	1	n	TEMP1	1	n						1
	37	T	1	n	S	1	n						1
	38	TEMP1	1	n	T	1	n						1
5	39	T	1	n	MEM (DATA)	1	n					0	1
	40	M	1	m	MEM (ADD)	1	m					1	1
6	41	T(k)	-	-	CLOCK	-	-				RESET	1	1

Figure 5. Design Table for Microsequence LOSE.

Microstep Number k	Row i	Source Register 1	Begin 2	End 3	Destination Register 4	Begin 5	End 6	Eq. Used 7	Begin 8	End 9	Control 10	Time 11	Delay 12
1	1	K	1	n	BUS1	1	n					1	0
	2	BUS1	1	n	LOGICAL (A)	1	n				AND	1	3
	3	W	1	n	LOGICAL (B)	1	n				AND	1	3
	4	LOGICAL	1	n	TEMP1	1	n					4	1
	5	TEMP1	1	n	LOGICAL (A)	1	n				OR	5	3
	6	R	1	n	LOGICAL (B)	1	n				OR	5	3
	7	LOGICAL	1	n	X	1	n					8	1
2	8	The entire microsequence WIN is entered here.										9-17	
	33												
3	34	M	1	m	MEM (ADD)	1	m					14	1
	35	MEM (DATA)	1	n	T	1	n					14	1
4	36	S	1	n	TEMP1	1	n					9	1
	37	T	1	n	S	1	n					15	1
	38	TEMP1	1	n	T	1	n					16	1
5	39	T	1	n	MEM (DATA)	1	n					17	1
	40	M	1	m	MEM (ADD)	1	m					17	1
6	41	"9"	-	-	CLOCK	-	-				RESET	18	1
999	42	"0"	-	-	CLOCK	-	-				RESET	19	1

Figure 6. Design Table for Microsequence LOSE after Timing Analysis.

20, and 21. Since the control conditions for operations 20 and 21 are the inverse of the conditions for operation 19, operation 19 is ignored during the determination of the starting time for operations 20 and 21.

The equation generator will convert the design tables into terms which will be combined in a later sort run to form the application equations. Terms generated by microsequence WIN are:

$$
\begin{aligned}
BUS1_{1\text{-}n} &= WIN \cdot P_{1\text{-}n} \cdot t_1 \\
Q_{1\text{-}n} &= WIN \cdot BUS1_{1\text{-}n} \cdot t_1 \\
BUS1_{1\text{-}n} &= WIN \cdot K_{1\text{-}n} \cdot t_2 \\
COUNT_{1\text{-}n} &= WIN \cdot BUS1_{1\text{-}n} \cdot UP \cdot t_2 \\
BUS1_{1\text{-}n} &= WIN \cdot COUNT_{1\text{-}n} \cdot t_3 \\
P_{1\text{-}n} &= WIN \cdot BUS1_{1\text{-}n} \cdot t_3 \\
ARITH(A)_{1\text{-}n} &= WIN \cdot S_{1\text{-}n} \cdot ADD \cdot t_1 \\
ARITH(A)_{1\text{-}n} &= WIN \cdot S_{1\text{-}n} \cdot ADD \cdot t_2 \\
ARITH(A)_{1\text{-}n} &= WIN \cdot S_{1\text{-}n} \cdot ADD \cdot t_3 \\
ARITH(A)_{1\text{-}n} &= WIN \cdot S_{1\text{-}n} \cdot ADD \cdot t_4 \\
ARITH(A)_{1\text{-}n} &= WIN \cdot S_{1\text{-}n} \cdot ADD \cdot t_5 \\
ARITH(B)_{1\text{-}n} &= WIN \cdot T_{1\text{-}n} \cdot ADD \cdot t_1 \\
ARITH(B)_{1\text{-}n} &= WIN \cdot T_{1\text{-}n} \cdot ADD \cdot t_2 \\
ARITH(B)_{1\text{-}n} &= WIN \cdot T_{1\text{-}n} \cdot ADD \cdot t_3 \\
ARITH(B)_{1\text{-}n} &= WIN \cdot T_{1\text{-}n} \cdot ADD \cdot t_4 \\
ARITH(B)_{1\text{-}n} &= WIN \cdot T_{1\text{-}n} \cdot ADD \cdot t_5 \\
W_{1\text{-}n} &= WIN \cdot ARITH_{1\text{-}n} \cdot t_6 \\
SHIFT_{1\text{-}n} &= WIN \cdot X_{1\text{-}n} \cdot RIGHT \cdot OFF \cdot t_1 \\
SHIFT_{1\text{-}n} &= WIN \cdot SHIFT_{1\text{-}n} \cdot RIGHT \cdot OFF \cdot t_2 \\
SHIFT_{1\text{-}n} &= WIN \cdot SHIFT_{1\text{-}n} \cdot RIGHT \cdot OFF \cdot t_3 \\
X_{1\text{-}n} &= WIN \cdot SHIFT_{1\text{-}n} \cdot t_4 \\
BUS1_{1\text{-}n} &= WIN \cdot K_{1\text{-}n} \cdot t_4 \\
LOGICAL(A)_{1\text{-}n} &= WIN \cdot BUS1_{1\text{-}n} \cdot EQL \cdot t_4 \\
BUS2_{1\text{-}n} &= WIN \cdot P_{1\text{-}n} \cdot t_4 \\
LOGICAL(B)_{1\text{-}n} &= WIN \cdot BUS2_{1\text{-}n} \cdot EQL \cdot t_4 \\
FF1_1 &= WIN \cdot LOGICAL_1 \cdot t_7 \\
X_{1\text{-}n} &= WIN \cdot W_{1\text{-}n} \cdot FF1 \cdot t_8 \\
Y_{1\text{-}n} &= WIN \cdot W_{1\text{-}n} \cdot \sim FF1 \cdot t_8 \\
W_{2\text{-}6} &= WIN \cdot R_{6\text{-}10} \cdot \sim FF1 \cdot t_9 \\
BUS1_{1\text{-}n} &= WIN \cdot K_{1\text{-}n} \cdot t_4 \\
COUNT_{1\text{-}n} &= WIN \cdot BUS1_{1\text{-}n} \cdot DOWN \cdot t_4 \\
BUS1_{1\text{-}n} &= WIN \cdot COUNT_{1\text{-}n} \cdot t_7 \\
P_{1\text{-}n} &= WIN \cdot BUS1_{1\text{-}n} \cdot t_7 \\
CLOCK &= WIN \cdot \text{"0"} \cdot RESET \cdot t_{10}
\end{aligned}
$$

After all microsequences have been processed, the terms from all microsequences are collected and sorted; and the equations for the system are formed. For example, the equation for the first bit of the X register will have the following form:

$$X_1 = \underbrace{\text{WIN} \cdot \text{SHIFT}_1 \cdot t_4 \vee \text{WIN} \cdot W_1 \cdot \text{FF1} \cdot t_8}_{\text{(from microsequence WIN)}} \vee$$

$$\underbrace{\text{LOSE} \cdot \text{LOGICAL}_1 \cdot t_8 \vee \text{LOSE} \cdot \text{SHIFT}_1 \cdot t_{12} \vee \text{LOSE} \cdot W_1 \cdot \text{FF1} \cdot t_{16}}_{\text{(from microsequence LOSE)}} \vee \ldots$$

CONCLUSION

The foregoing outlines a programming system that is another tool for the computer designer. The emphasis has been on the representation of a computer system in a form suitable for further processing by an automated design system, or by programs to evaluate the cost of the machine under consideration. In general, it is envisioned that the user of this system would be the designer concerned principally with the overall control structure of a computer.

It should be pointed out that the algorithms devised are, to a large degree, a function of the class of machines being designed (parallel-synchronous, in the example), and, to a lesser degree, a function of assumptions regarding characteristics of the system elements available to the designer. (For example, the translation algorithm employed would have to be changed to accommodate "shifting type" registers.) Further changes would have to be incorporated to give the designer explicit control over the concurrences in microsequences rather than to arbitrarily exploit the concurrences (based upon utilization of system elements) in the translator.

The system described herein will be used as a vehicle for extending the notation to cover wider classes of designs, and to study the implications of the notational devices to canonical representation of computer systems.

It has not escaped the authors that the language described in this paper is essentially the form necessary for functional simulation of computers, and that it would be a relatively simple task to write a translator that would generate a simulation program representing a proposed machine design, either on a functional level or on an individual clock pulse basis.

ACKNOWLEDGEMENT

The contributions of Roy Proctor, of the Burroughs Laboratories Computation Center, both for programming and for several suggestions in connection with this work, are acknowledged with pleasure.

REFERENCES

1. Barton, R. S., "A New Approach to the Functional Design of a Digital Computer," Proceedings, 1961 WJCC, pp. 393-396; May 9-11, 1961.
2. Naur, P., et al, "Report on the Algorithmic Language ALGOL 60," Communications of the ACM, vol. 3, no. 5, pp. 299-314; May 1960.
3. Burroughs Algebraic Compiler, Bulletin 220-21011-P (Detroit, Michigan: Equipment and Systems Marketing Division, Burroughs Corporation) January 1961.

COMPROTEIN: A COMPUTER PROGRAM TO AID PRIMARY PROTEIN STRUCTURE DETERMINATION*

Margaret Oakley Dayhoff and Robert S. Ledley
National Biomedical Research Foundation
Silver Spring, Maryland

INTRODUCTION

Among the main chemical constituents of the human body—and, in fact, of all living things—are proteins. In addition to serving as component structural parts of many types of living tissues, the proteins are enzymes that are necessary in order that the chemical reactions which comprise the life processes may occur. The protein enzymes act to "decode" the message of the genes, interpreting this message in terms of specific chemical reactions which determine the physical and functional characteristics of the organism. Thus proteins play a uniquely vital role in the evolution, ontogeny, and maintenance of living organisms. It therefore becomes important when studying the basis of life processes to know the structure of the proteins themselves.

In spite of their highly complex role, the molecular structure of proteins is, in principle, relatively simple: they are long chains of only twenty different types of smaller molecular "links" called amino acids (see Figure 1). Each type of protein is characterized by a particular ordering of the amino acid links, and a major problem in finding the exact structure of a protein is to obtain the ordering of the amino acids in the chain. This ordering is of great interest because it is the order of the amino acids in a protein that is determined by the gene. Thus, according to current biological theory, the gene determines which proteins will be made by determining the order of the amino acids in the protein chain and it is these proteins in turn, acting as enzymes, that control the chemical processes that determine the physical and functional characteristics of the organism.

Finding the amino acid order of a protein chain has proved a time consuming process for the biochemist; in fact, only about 6 complete or almost complete protein orderings have been found so far, namely those of insulin [1], hemoglobin [2], ribonuclease [3], tobacco mosaic virus protein [4], myoglobin [5], and cytochrome C [6]. The basic technique used on all these proteins (with the exception of myoglobin) was to breakdown the long chain chemically into smaller fragment chains at several different points, to analyze the amino acids in each fragment chemically, and then to try to reconstruct the entire protein chain by a logical and combinatorial examination of overlapping fragments from the different breakdowns. It is in this reconstruction of the protein that the computer finds its application.

*The research reported in this paper has been supported by Grana GM 08710 from the National Institutes of Health, Division of General Medical Sciences, to the National Biomedical Research Foundation.

	Name	Abbreviation		Name	Abbreviation
1	alanine	ALA	11	leucine	LEU
2	arginine	ARG	12	lysine	LYS
3	asparagine	ASN	13	methionine	MET
4	aspartic acid	ASP	14	phenylalanine	PHE
5	cysteine	CYS	15	proline	PRO
6	glycine	GLY	16	serine	SER
7	glutamine	GLN	17	threonine	THR
8	glutamic acid	GLU	18	tyrosine	TYR
9	histidine	HIS	19	tryptophane	TRY
10	isoleucine	ILU	20	valine	VAL

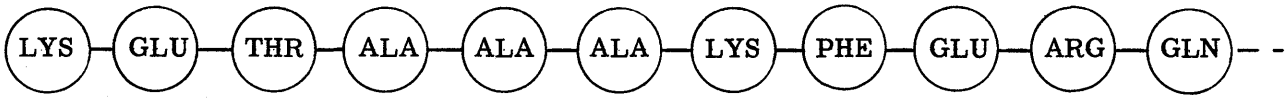


Figure 1. A listing of the amino acids with their abbreviations is shown in the upper section and the lower indicates part of the protein ribonuclease which actually comprises a chain of some 124 amino acids.

As a trivial example, suppose that for a protein one chemical breakdown produced the fragment chains of known ordering,

Breakdown P: AB, CD, and E

Where A, B, C, D and E each occur once and only once in the protein. Let us call this a complete breakdown, and let another breakdown, this time incomplete, produce the fragments

Breakdown Q: BC and DE

where A, B, C, D, and E represent amino acids. Here fragment BC in Breakdown Q (see Figure 2a) clearly overlaps the two fragments AB and CD of Breakdown P, and DE overlaps CD and E of breakdown P, giving as the reconstructed protein

ABCDE.

As another example, consider the more common case where the amino acid components of a fragment are known, but the order of these within the fragment is unknown. Let parentheses indicate that the order they enclose is unknown [e.g., (A, B, C) represents the six permutations of A, B, C; (D, E) represents either DE or ED; (A, B, C) (D, E) represents the $6 \times 2 = 12$ fragments of each of the six permutations in (A, B, C) followed by DE or ED etc.], and suppose that one complete breakdown is

Breakdown P: (A, B, C) and (D, E)

and that another, incomplete, breakdown is

Breakdown Q: (A, B) and (C, D)

Clearly (C, D) of breakdown Q overlaps (A, B, C) and (D, E) of breakdown P but (A, B) is contained within (A, B, C). Hence, since each amino acid has distinct "left" and "right" ends, two possible protein reconstructions result, namely (see Figure 2b)

(A, B) (C) (D) (E) and (E) (D) (C) (A, B)

where in each possibility the order of A, B still remains unknown. Such partial reconstructions frequently occur, and pinpoint for the biochemist that portion of the molecule on which further effort is required.

Unfortunately, however, the problems involved in reconstructing proteins are not as simple as in the examples just given. The largest protein analyzed so far, the tobacco mosaic virus protein, has only 158 amino acids whereas proteins usually have chains of many hundreds of amino acid links. Since the number of combinations on n things taken r at a time ($1 < r < n$) increases more rapidly than does n itself, it is to be expected that the difficulties in piecing together the fragments of a protein will increase proportionally faster than the number of amino acids in the protein. In addition, there may be occasional errors in the fragments reported by the biochemist,

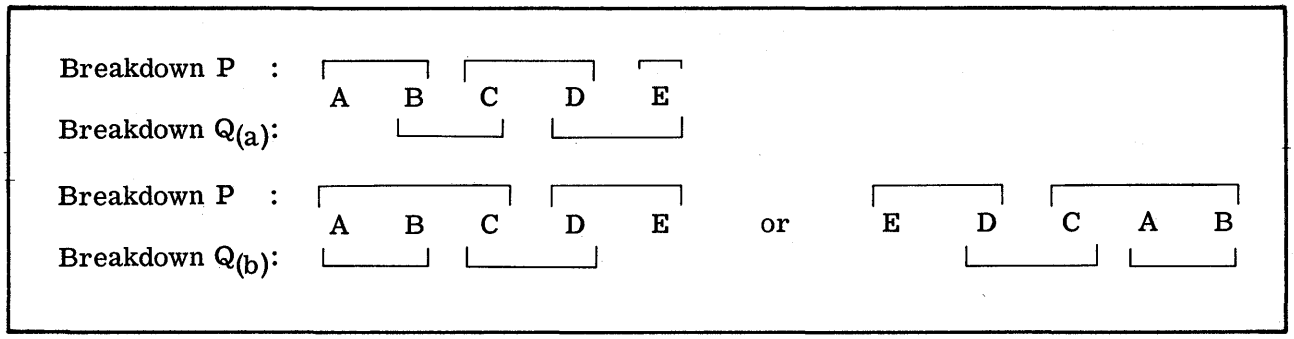


Figure 2. Illustration of two different breakdowns of a protein into amino acid fragments (see text).

as well as other aberrations in the data. Hence the logical and combinatorial problems can become severe, and a computer is then required to assist in the analysis.

The advantage of computer aid is that it may help significantly to extend the current chemical analysis methods of determining the amino acid sequences of proteins to many more and much larger proteins. By exhaustively analyzing the possibilities of protein reconstructions, the computer may assist in determining the best next step to try in the chemical analysis processes. In addition, it should be noted that presently the chemical analysis is carefully planned to produce results that will be logically simple for mental analysis. The use of a computer to perform the logical analysis may thus allow significant simplification and further systematization of the chemical experimental procedures by placing more of the burden on the automated logical and combinatorial analysis and less on the experimental procedures.

In this paper we shall describe a completed computer program for the IBM 7090, which to our knowledge is the first successful attempt at aiding the analysis of the amino acid chain structure of protein [7]. The idea was conceived by us in 1958, but actual programming was not initiated until late 1960. D. F. Bradley, S. A. Bernhard, and W. L. Duda have independently reported, in an as-yet unpublished paper [8], progress in approaching a similar problem. R. Eck has reported on a system for using marginal-punch cards to aid in certain aspects of the logical analysis problem [9].

A SIMPLIFIED ILLUSTRATION

Discussion of the programming methods utilized will be clarified if we first consider a simple illustration. Suppose a complete breakdown P is made by the biochemist as in Figure 3, and that another breakdown Q is also known but not complete (see Figure 3).

Complete breakdown P LIST		Incomplete breakdown Q LIST	
p_1	(R)(A,B)	q_1	(A)(B,B,D)
p_2	(D)(B)(C,A)	q_2	(A)(C,A,C)
p_3	(A)(C)(D)(X,A)(C)	q_3	(X)(A,C,B)
p_4	(B)(D)(B)(D)(A,B,D)	q_4	(B)(A,A,C,D)
p_5	(C)(A)(Z)	q_5	(A)(B,C,D)

Figure 3. Breakdowns of protein fragments for the illustration in the text.

In Figure 4 we show how such breakdowns P and Q can occur from our hypothetical protein, but the problem is to reconstruct this from the fragments given in Figure 3. Since each fragment q_i of breakdown Q must either overlap several fragments of P, or else be included within some fragment of P, let us start by making a list for each q_i of all possible such associated P fragments; Figure 5 shows such lists for our illustration. As an example of how each entry in a list is found, consider the test of whether or not q_4 overlaps p_4p_5 where

q_4 is (B)(A,A,C,D)

and where

p_4 is (B)(D)(B)(D)(A,B,D), p_5 is (C)(A)(Z)

The problem is to determine if each acid of q_4 can be accounted for in p_4 and p_5. First note that the maximum overlap between q_4

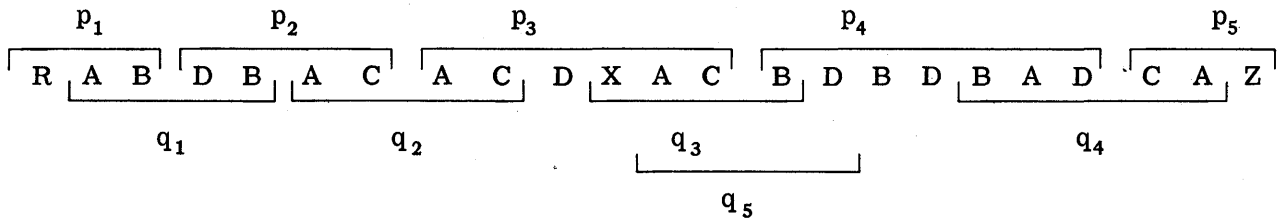
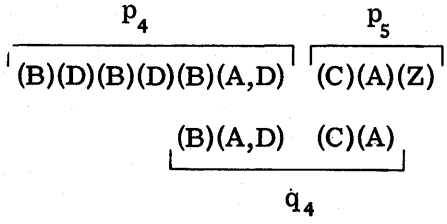


Figure 4. Illustration of sources of peptide fragments from protein molecule illustrated in the text.

q_1	q_2	q_3	q_4	q_5
$p_1 p_2$	$p_2 p_3$	$p_3 p_4$	$p_1 p_3$	$p_1 p_2$
$p_1 p_4$	$p_2 p_5$		$p_4 p_3$	$p_2 p_4$
$p_2 p_4$	$p_3 p_5$		$p_4 p_5$	$p_3 p_2$
$p_4 p_2$				$p_3 p_4$
				$p_4 p_2$
				$p_4 p_5$

Figure 5. The q lists for the illustration in the text.

and p_4 is (B,A,D), on the right of p_4. This leaves (A,C) of q_4 "hanging over" on the right of q_4, to be accounted for in p_5. This is clearly possible, resulting in the overlap:



In order to determine all the entries in all the lists, such trials must be made by the computer for every pair of fragments $p_i p_j$, for each q_k.

However, just forming the lists is but the first step in reconstructing the protein chain. The next step is an elimination process to leave only the consistent possibilities. For instance, q_3 can only arise from p_3p_4; hence p_3 must be followed by p_4, and p_4 must be preceeded by p_3, and therefore all other possibilities in other lists involving p_3 and p_4 can be eliminated—such as p_1p_4, p_2p_4 and p_5p_4 in the q_1 list of Figure 5, p_3p_5 in the q_2 list, p_4p_3 in the q_4 list, and p_2p_4 and p_3p_2 in the q_5 list.* This leaves in the list for q_1 only p_1p_2 and p_4p_2. If we first assume that p_1p_2 is overlapped by q_1, then in the q_4 list only p_4p_5 remains, and hence in the q_2 list only p_2p_3 remains, giving altogether these adjacent fragments,

p_3p_4, p_1p_2, p_4p_5, and p_2p_3

which determine the structure as

$p_1p_2p_3p_4p_5$

On utilizing p_1, p_2, p_3 and p_4 we find as the final structure:

(R)(A)(B)(D)(B)(A)(C)(A)(C)(D)(X)
(A)(C)(B)(D)(B)(D)(B)(A,D)(C)(A)(Z)

Returning to the second possibility in the q_1 overlap list, namely p_4p_2, this leaves only p_1p_3 in the q_4 list, which in turn leaves only p_2p_5 in the q_2 list. Hence a second possibility for adjacent fragments is

p_3p_4, p_4p_2, p_1p_3, and p_2p_5

which gives the structure

$p_1p_3p_4p_2p_5$

On utilizing q_3, q_2, q_4, and q_2, we find

(R)(B)(A)(A)(C)(D)(X)(A)(C)(B)(D)
(B)(D)(D)(A)(B)(D)(B)(A)(C)(C)(A)(Z)

Actually, it is also necessary to eliminate the occurrence of p_4 in the lists by replacing it with p_3, which stands for p_3p_4. This is to insure that an impossible succession of conditions such as p_1p_2, p_2p_3, p_3p_1 is not produced.

COMPUTER HANDLING OF BIOCHEMICAL INFORMATION

The problems involved in writing a computer program, however, are not as straightforward as the above illustration might indicate. The biochemist utilizes enzymes to break up (hydrolyze) the protein into the fragments that we have been considering; these fragments are called peptides by the biochemist. The enzymes commonly used, such as subtilisin and chymotrypsin, produce an assortment of peptides which may overlap each other. Hence a problem arises in actually arriving at a complete set of peptide fragments, as illustrated above in the breakdown P. In addition, for several reasons, the biochemical experiments very often do not result in integer values for the number of amino acids of a particular kind that occur in a peptide fragment. This second uncertainty problem must also be taken into account by the computer program. Furthermore, there may be experimental errors in the amino acid composition and ordering of some peptides.

In the case of overlapping peptide fragments from a hydrolytic breakdown, the computer program tries to reconstruct a complete set of fragments from overlapping subsets of fragments. The procedure of accomplishing this is to look for every group of two, three, or four acids known to be adjacent in some peptide. Then, for each such group, the probability that this particular group will occur again in the protein chain is computed from the amino acid frequency data. For instance, if the ordered pair LYS-PHE occurs, and it is known that there are five LYS residues and four PHE residues in the entire protein chain of, say, 150 amino acid links altogether, then the probability that another such LYS-PHE pair will occur is approximately $4 \times 3/150$.

If the probability is small that another such group occurs, it is most likely that all of the peptides containing this group should arise from the same part of the protein; hence these peptides are sorted out. All possible fragments that can be reconstructed from these (overlapping) peptides are then determined.

It may happen, however, that all these peptides cannot "fit" into any reconstructed fragment; this indicates either that some peptide must arise from a different place on the protein or that there may be an experimental error. In such a case the experimental results are reconsidered from a chemical point of view.

Of course, there is a small but finite probability that a misinterpretation can be made at this point and an erroneous peptide constructed, such as could occur if a highly unlikely configuration actually occurred more than once in the protein or if there were lost peptides from a particular region but the existing peptides fortuitously fit. However, it is likely that in any case, in later building up of the protein, an inconsistency would arise, leading to the rejection of this erroneously constructed peptide.

Some peptides may contain two or more groups on which searches are made. Reconstructed fragments containing these peptides must be joined together themselves. Hence the program merges these fragments to obtain all possible larger fragments. Such procedures will fix the relationships of many amino acids beyond that given in the initial data. These new relationships change the probabilities of occurrence of the two, three, or four amino acid groups. The probabilities are accordingly recalculated, and once more searches on improbable groups are made, leading to further merges of fragments into even larger fragments. This process is repeated by an iterative program until less than about 20 fragments remain.

Further details must be taken into consideration by the program: the set of fragments may still not be complete; there may exist alternative possibilities for a fragment; and there may be gaps in the chain where all the peptides were lost.

After obtaining a complete, or almost complete, set of fragments by iteration of the searching procedure, the program can continue toward reconstructing the entire protein utilizing the remaining peptides not used in the building up of the complete set P as the Q set of peptides (see example above).

It should be noted that in the various phases of the reconstruction of the protein the assumption is made that the total amino acid content of the entire protein and of the fragments is known. There is always, however, some experimental uncertainty in the number of amino acids of each type in the protein, to within a fraction of one amino acid. As a rule, the larger number of amino acids is always chosen initially. If an extra acid is thereby included in the computations, it may

be eliminated at the end, by a procedure described below.

Completing the final reconstruction of the protein again can present further details which must be taken into consideration by the computer program. Some peptides may appear to overlap at only one amino acid. If this occurs it would be unwise to conclude definitely that this represented a true overlap. Hence "pseudofragments" are used which consist of each overlapping fragment without the common amino acid.

Single amino acids to complete the P set, as required by the amino acid constitution of the protein are considered with the larger fragments.

If extra amino acids are so included, the final answer showing which fragments must be attached may place no attachment restrictions on these extra acids. In this case, if the acid arose from a fractional experimental result (see above), one may presume that it does not actually occur in the molecule. Otherwise further experimentation may be required. For example, if amino acid X is added to the P list in Figure 3, the Q lists will be unaffected and the resulting answers will be unchanged. One might conclude that either X really didn't belong in the P list or it could be at either end of the molecule. It is sometimes known which amino acids are on the right and left ends of the protein itself, and this information can further reduce the final possibilities.

DESCRIPTION OF COMPUTER PROGRAMMING SYSTEM

The computer programming system to aid protein analysis has been written in a flexible manner. The computer input and output is in terms of three letter abbreviations for the amino acids, with the parentheses notation for ordered and unordered sets as described above. Intermediate results are printed out for examination by the biochemist; in fact the entire process is geared for a close cooperative effort between the computer and the biochemist during the entire analysis. This is necessary in order to take advantage of the special conditions presented by any particular protein and type of chemical experimental procedures. For example it might be convenient to omit all prolines from the peptides, or not to consider a distinction between Glu and Gln. Special rules might be introduced regarding end-groups from hydrolyses by certain enzymes, etc. Such special considerations can be handled by the programming system, and make it easier to spot errors in the experimental data.

The programming system is based on the following six programs:

(1) MAXLAP: Program to find the maximum possible overlap between any two peptides with any amount of ordering information known.

(2) MERGE: Program to find all possible overlapping configurations of two peptides.

(3) PEPT: Program to find all possible fragments that are consistent with the overlapping of any number of peptides.

(4) SEARCH: Program to search on probabilistic considerations all peptides which contain an unusual group of amino acids.

(5) QLIST: Program to generate the Q-lists of possible associated sets of P peptides over which each q_i fragment can fit.

(6) LOGRED: Program to perform the logical reduction of the Q-lists to obtain all possible protein structures that are consistent with the data.

Since detailed flow diagrams would consume too much space and not be appropriate for the present discussion, we have included here only gross overall flow diagrams of these programs. Each of the six programs will now be described, and simple examples illustrating some of the methods involved will be given. (For further details see "Sequencing of Amino Acids in Proteins Using Computer Aids," Report No. 62072/8710, National Biomedical Research Foundation, Silver Spring, Md., July 1962.)

Program MAXLAP (Figure 6). In this program p and q are peptide fragments, PCOM and QCOM are lists of acids from these peptides respectively which may provide the maximum overlap. After setting up a tentative maximum number of positions (i.e., amino acids) of overlap, three cases may be distinguished as illustrated by the three examples of Figure 7. In the first example there is the successful maximum overlap situation where all of p or q is overlapped. Here MV is the list of all the acids from PCOM and QCOM which match; with this maximum overlap, the complete maximum overlap protein fragment is shown. In

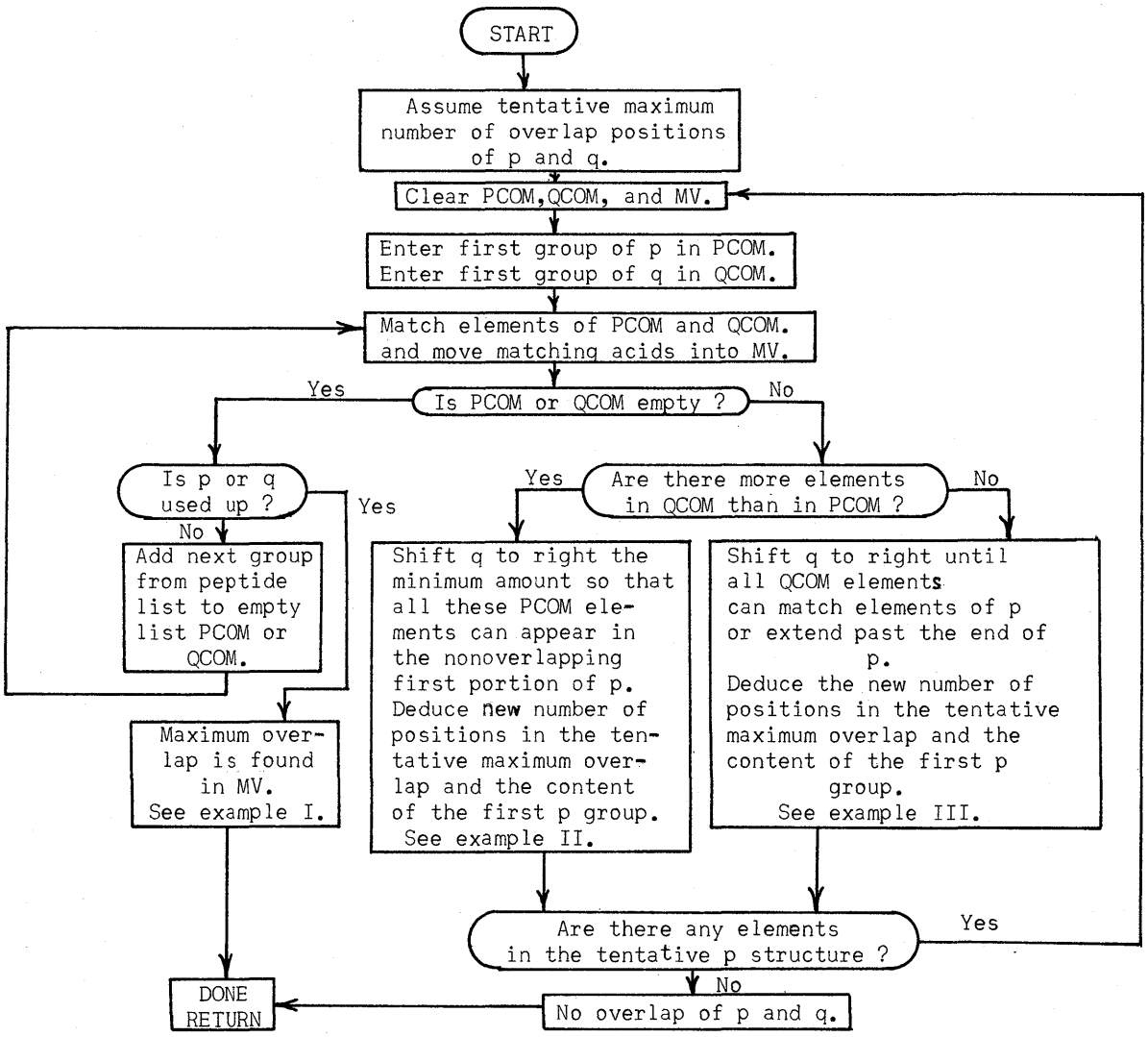


Figure 6. Flow Chart for Program MAXLAP.

Example I	**p:**	**(A,B)(C,D)**
	q:	**(C,B,A)(F,G,D)**
	MV:	**(A,B)(C)(D)**
	Protein fragment with maximum overlap:	**(A,B)(C)(D)(F,G)**
Example II	**p:**	**(C)(D,E,F)**
	q:	**(D,C,E)(G,H)**
	New tentative maximum Overlap structures	**(C)(F,D,E) (D,C,E)(G,H)**
	Protein fragment with maximum overlap:	**(C)(F)(D,E)(C)(G,H)**
Example III	**p:**	**(A,B)(B,E,A)(D)**
	q:	**(B)(A,D,E)(F)**
	New tentative maximum Overlap structures	**(A,B)(B,E,A)(D) (B)(A,D,E)(F)**
	Protein fragment with maximum overlap:	**(A,B)(B)(A,E)(D)(F)**

Figure 7. Examples of the three cases considered in flow chart of program MAXLAP.

the second example all of p cannot be overlapped by q, and hence new tentative overlap positions must be assumed. Here F is the limiting acid since it does not appear in q. In the third example even though D appears in both p and q, new tentative overlap positions must be assumed, because in p the D acid is restricted in position at the right.

Program MERGE (Figure 8). Once the maximum overlap has been determined, all other possible overlaps can be determined. Several cases can occur. First the essentially trivial cases of p and q entirely disjoint, as in

p: (A,B,C) q: (D,E)

or else q a subset of p as in

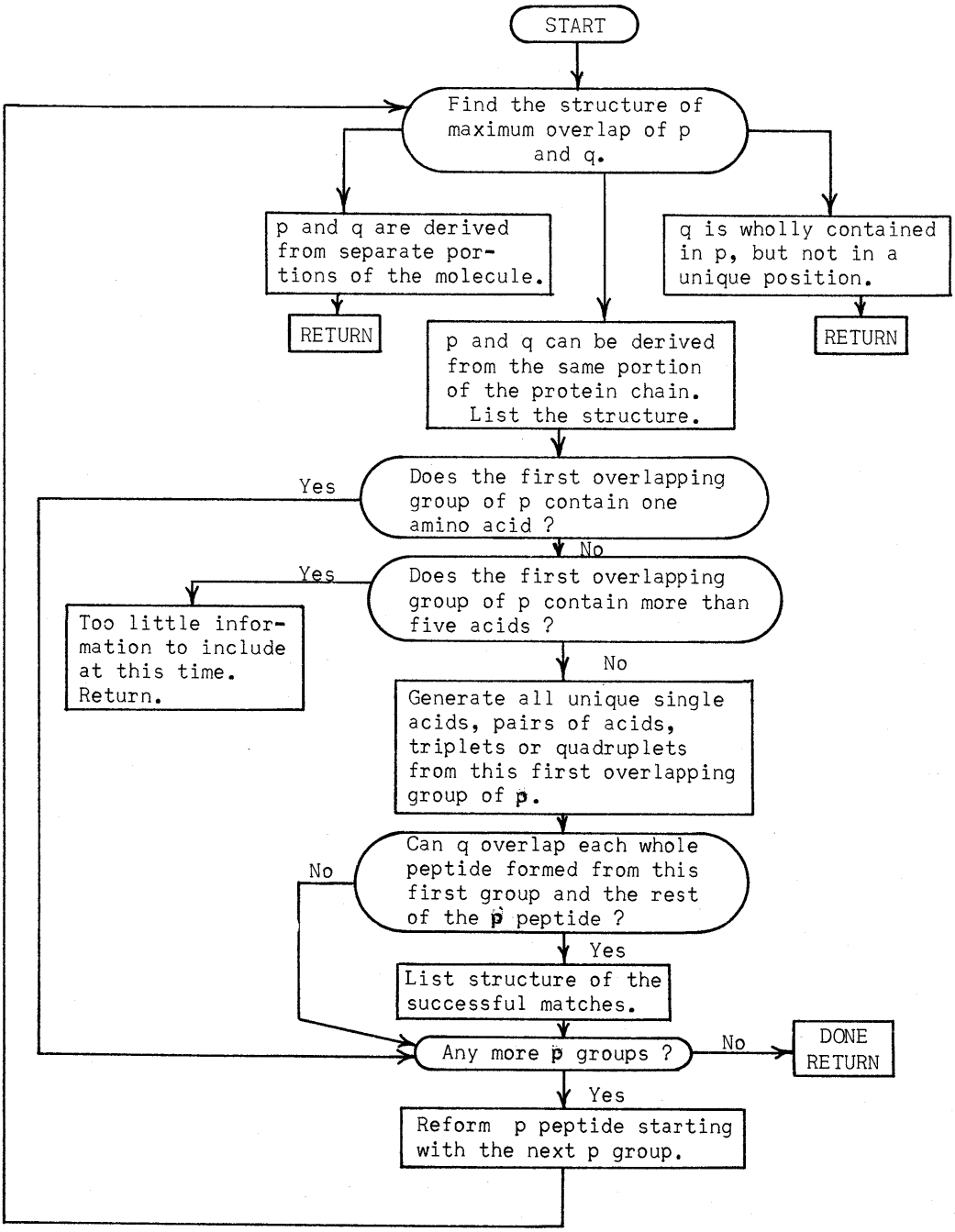


Figure 8. Flow Chart for Program MERGE.

p: (A,B,C,D) q: (B,C)

If neither of these is the case, as for example in

p: (A,B,B,D) q: (B)(B,D)(G)

the first overlapping group of p is considered, which for our later example is

(B,B,D)

If this first overlapping group contains not more than five nor less than two acids, it warrants further consideration. A list is made of all singles, pairs, triplets and quadruplets of acids that can be formed from this overlapping group, which for our example is

(B) (D) (B,D) (B,B)

Next each of these is examined to see if it can overlap with q. For an example we have respectively

(A,B,D)(B)(B,D) G, none,
(A,B)(B)(D)(B)(G), and (A,D)(B)(B)(D)(G)

where we have underlined the overlapping group to the left of which p fits and to the right of which q fits. Finally the peptide is re-formed starting with the next group, and so forth, until all the overlapping groups of p and q have been considered.

Program PEPT (Figure 9). This program extends the previously discussed program MERGE in that it finds all of the possible structures of the protein chain consistent with the overlapping of all the peptides obtained from a search for all experimental peptides with a rare configuration of amino acids. The overlapping portion of these acids must contain the group of rare amino acids, called SAA in the flow chart. The list resulting from the search is called MCO in the flow chart.

Program SEARCH (Figure 10). This program systematically looks at each pair, triplet, and quadruplet of amino acids that are known to occur together from the experimental peptide fragment data. For each such group, the probability of its occurrence is computed from the amino acid frequency data as described above. A list, called Num(L) in the flow chart, is made of these groups of amino acids known to occur together which are improbable of occurrence (i.e., less probable than some chosen value). The letter L is used to index the elements of the list Num(L). Finally the program utilizes PEPT to generate and print out all possible merged structures. For example suppose a search was made on the ordered pair

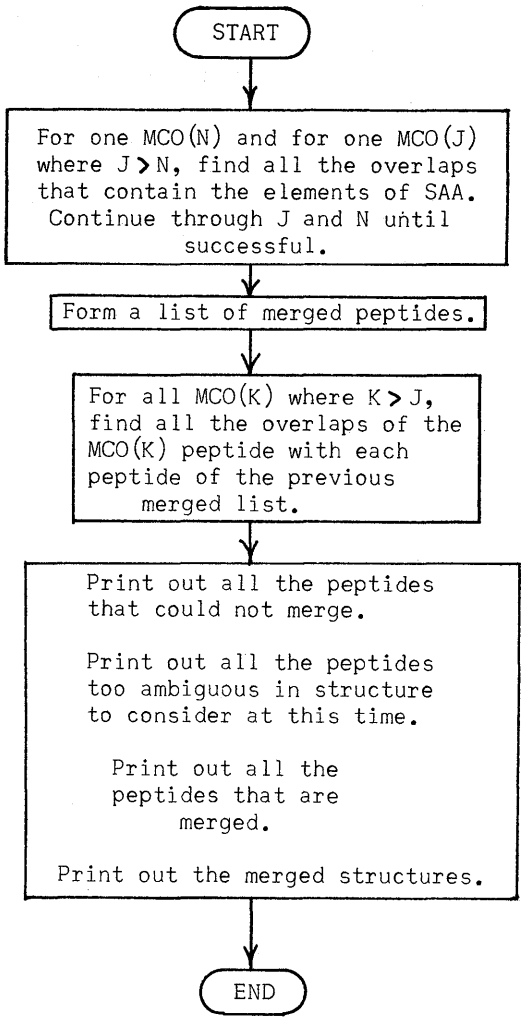


Figure 9. Flow Chart for Program PEPT.

LYS—PHE

and there resulted the following fragments:

(ALA)(ALA, ALA, LYS)(PHE)
(ALA)(ALA, LYS)(PHE)
(ALA)(LYS, PHE, GLU, ARG)(GLU)
(LYS)(PHE)

The merged structure becomes

(ALA)(ALA)(ALA)(LYS)
(PHE)(GLU,ARG)(GLU)

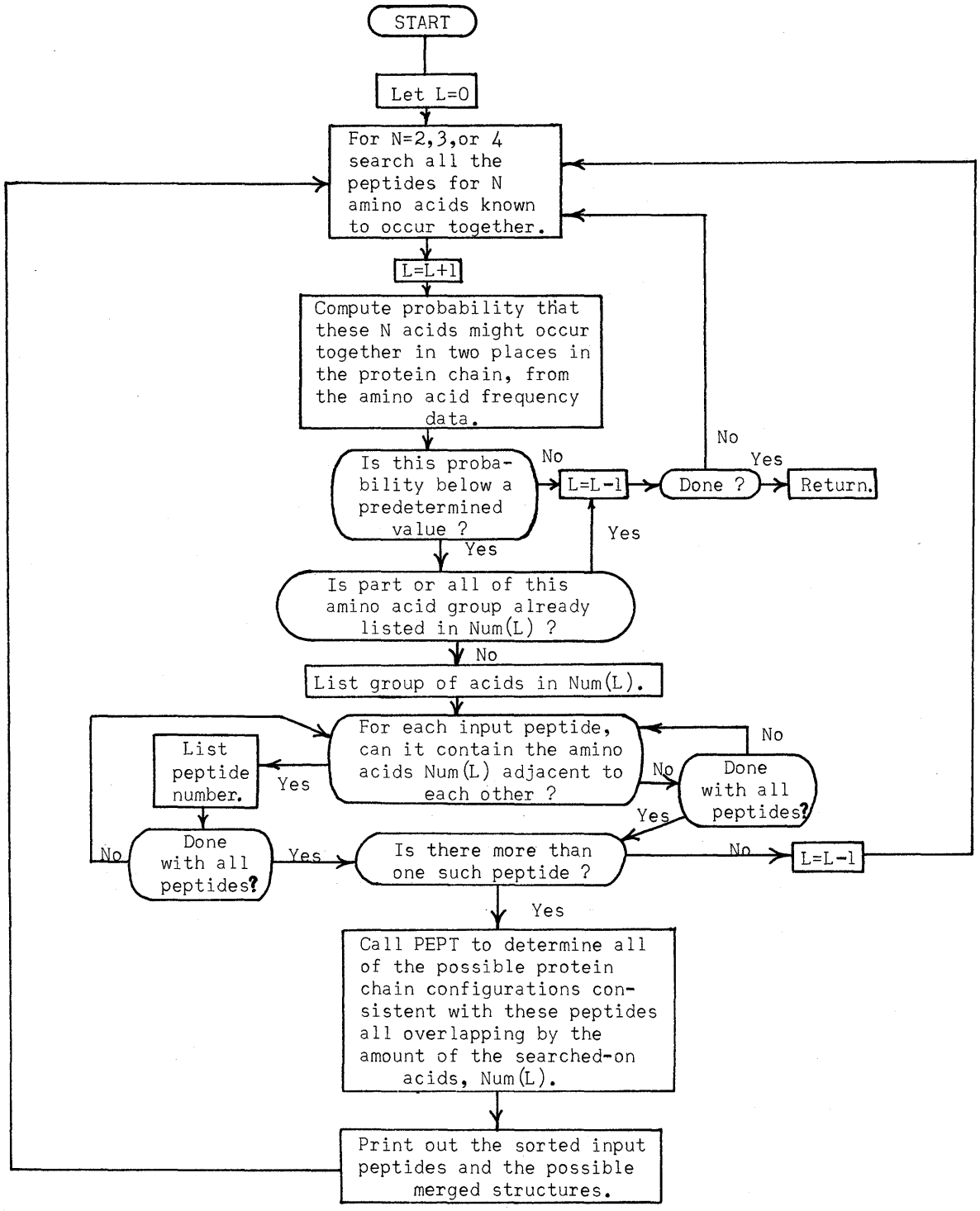


Figure 10. Flow Chart for Program SEARCH.

Program QLIST (Figure 11). This program forms the lists of peptides related by each fragment q_1. It is to be noted that each element of a Q list may contain up to five p fragments (although in our example of section 2 only two peptides appeared in each element of the Q lists). The input to this program is a list P of peptides which in some order will reconstruct the original protein and a list Q of peptides which give additional ordering information about the protein. In the flow chart P' is a hypothetical peptide

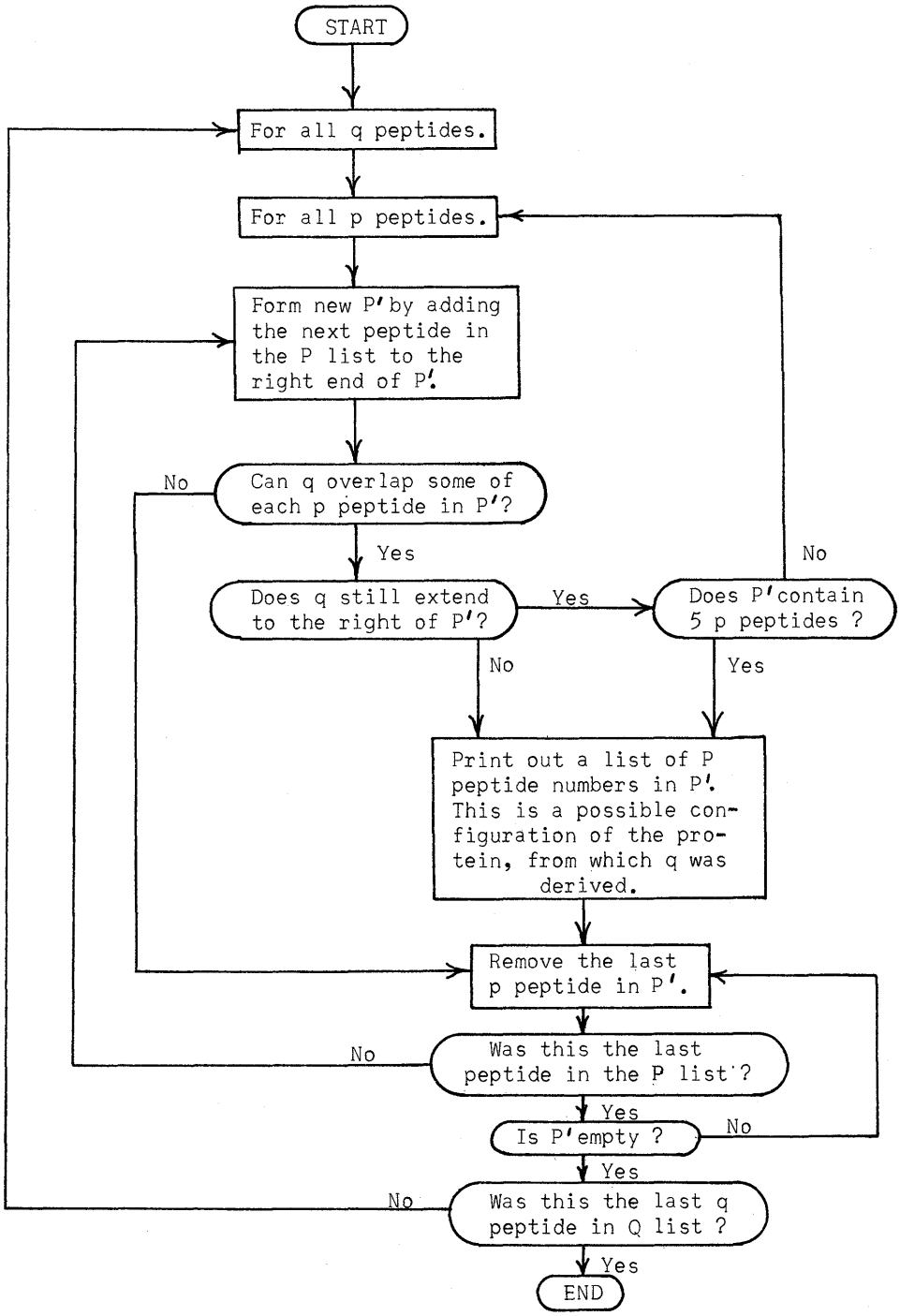


Figure 11. Flow Chart for Program Q LIST.

constructed by the juxtaposition of up to five P peptides.

Program LOGRED (Figure 12). In this program the Q lists are given. Calling each term of a Q lists a condition, the flow chart involves the lists: MQ(M) of conditions on the assumption of the Mth tentative condition; MQI of conditions being considered; and IR(M) of tentative conditions which determine a possible protein structure. The symbol MTI(J) is the last condition considered in the Jth Q list. The program follows a tree structure of possibilities, keeping tract of tentative conditions until a branch is eliminated or comes to a successful conclusion. The program follows with greater generalization

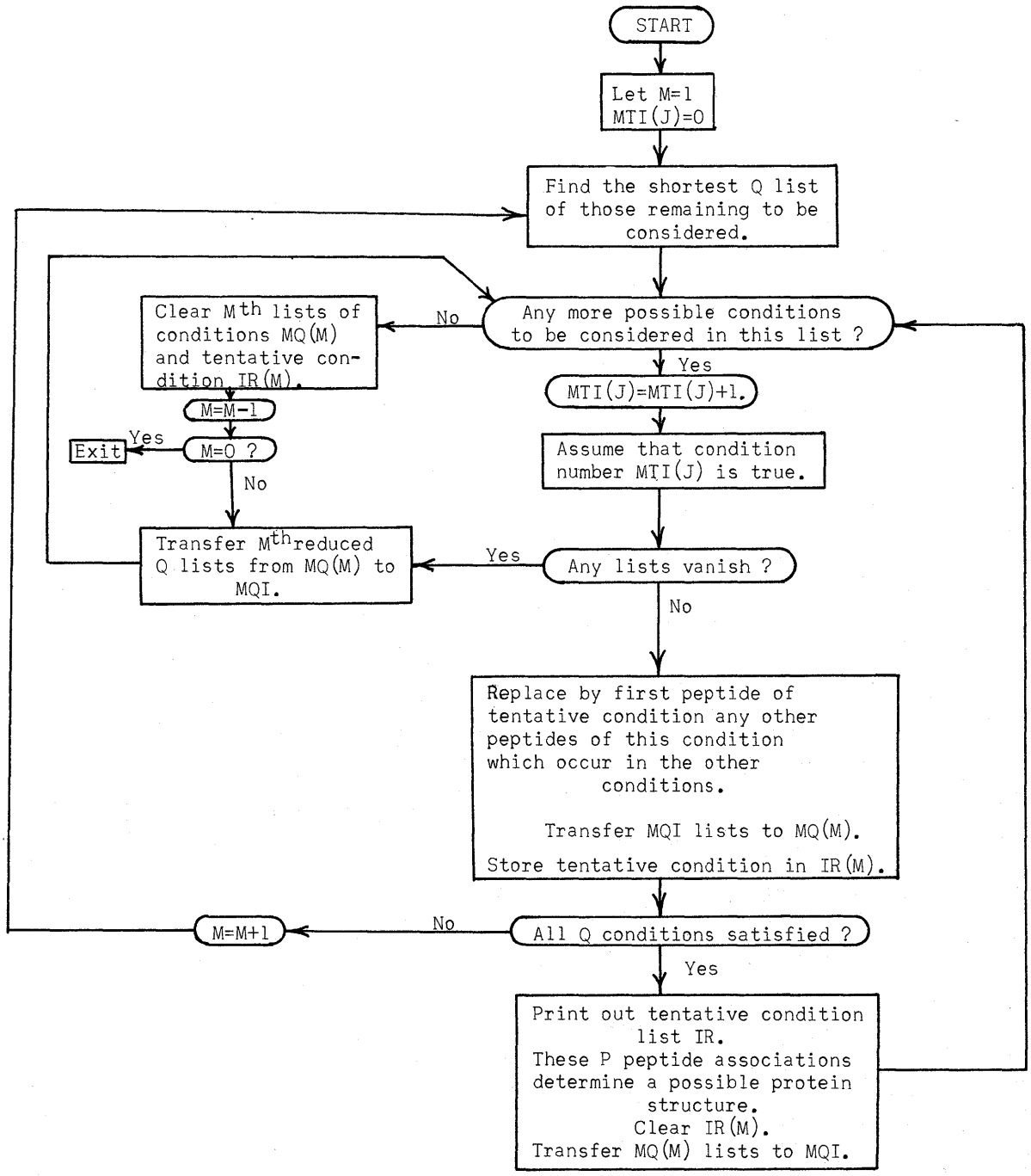


Figure 12. Flow Chart for Program LOGRED.

the method of the "simplified illustration" of the second section of this paper.

SUMMARY AND CONCLUSION

The computer program described in this paper becomes useful when there is a large number of small peptide fragments resulting from the breakdowns which are to be woven into a consistent and unique structure. This is a long tedious process when carried out by hand, and is subject to careless errors and impatient overlooking of all alternative

possibilities.* The completed IBM 7090 Computer program has been successfully tested on a hypothetical subtilysin breakdown of ribonuclease into over eighty fragments.

Just as the proteins are composed of chains of the same types of molecules, the genetic substances desoxyribonucleaic acid (DNA) and ribonucleaic acid (RNA) are composed of chains of only four different types of molecules called the nucleotide bases. It is possible that the order of the molecules in these substances can also be determined by the aid of this computer program and some computer experiments in this direction have been made. However, application of these techniques to DNA and RNA still awaits further development in the chemical experimental methods.

*Dr. Wm. Dreyer, of National Institutes of Health, has developed chemical techniques for isolating a large percentage of the peptides formed in a subtilisin hydrolysis, and for determining the total amino acid content and identifying the right and left ends of the peptide fragments. This experimental technique is very rapid and can be mechanized to a large extent; thus data taking should be reduced to months instead of years. The computer program is ideally suited for analysing this type of data.

The computations in this paper were done in the Computing Center of the Johns Hopkins Medical Institutions, which is supported by Research Grant, RG 9058, from the National Institutes of Health and by Educational Contributions from the International Business Machines Corporation.

REFERENCES

1. Sanger, F., Science, 129, 1340 "1959." Hirs, C. H. W., Moore, S., Stein, W. H., J. Biol. Chem. 235, 633, "1960" Spockman, D. H., Stein, W. H., Moore, S., J. Biol. Chem. 235, 638 "1960."
2. Braunitzer, G., Gehring-Mueller, R., Helschmann, N., Helse, K., Hobom, G., Rudloff, V., and Wittmann-Tiebold, B., Hoppe Seyler Z. Physiol. Chem., 325, pp. 283-6, Sept. 20, 1961.
3. Hirs, C. H. W., Moore, S., and Stein, W. H., J. Biol. Chem. 235, 633 (1960).
4. Tsugita, A., Gish, D., Young, J., Fraenkel-Conrat, H., Knight, C. A. and Stanley, W. M., Pros. Natl. Acad. Sci., Vol. 46, pp. 1463-9, 1960.
5. Kendrew, J., Watson, H., Strandberg, B., Dickerson, R., Phillips, D. and Shore, V., Nature, Vol. 190, pp. 666-70, May 20, 1961.
6. Margoliash, E., Smith, E., Kreil, G., and Tuppy, H., Nature 192, 1121-7, (Dec. 1961).
7. Ledley, R. S., Report on the Use of Computers in Biology and Medicine, Natl. Acad. of Scis.-Natl. Research Council, May 1960.
8. Bradley, D. F., Bernhard, S. A., Duda, W. L., unpublished work.
9. Eck, R., Nature, Jan. 20, 1962, 241-243.

USING GIFS IN THE ANALYSIS AND DESIGN OF PROCESS SYSTEMS

William H. Dodrill
Service Bureau Corporation
Subsidiary of International Business Machines

A process system may be defined as an integrated combination of equipment which functions to produce one or more products, and possibly byproducts, by altering the physical and/or chemical nature of raw materials. Even though there is wide diversity in products produced, raw materials used, and equipment combinations employed, all process systems exhibit certain fundamental similarities. Characteristically, any process system may be subdivided into three operational phases. These are: preparation of raw materials, conversion to products, and recovery and purification. All three phases may not necessarily be included in a specific process system, nor is there always clear distinction among them. Still, there is universality in practice in that only a limited number of types of equipment are used to perform specific types of operations.

As an example of a relatively simple, but fairly typical, process system, consider the flow diagram for high-temperature isomerzation reproduced in Figure 1. This is an operation performed in petroleum refining to improve the octane rating of certain gasoline constituents. It consists of a reactor for conversion, a flash drum and three distillation columns for recovery and purification, and several pieces of auxiliary

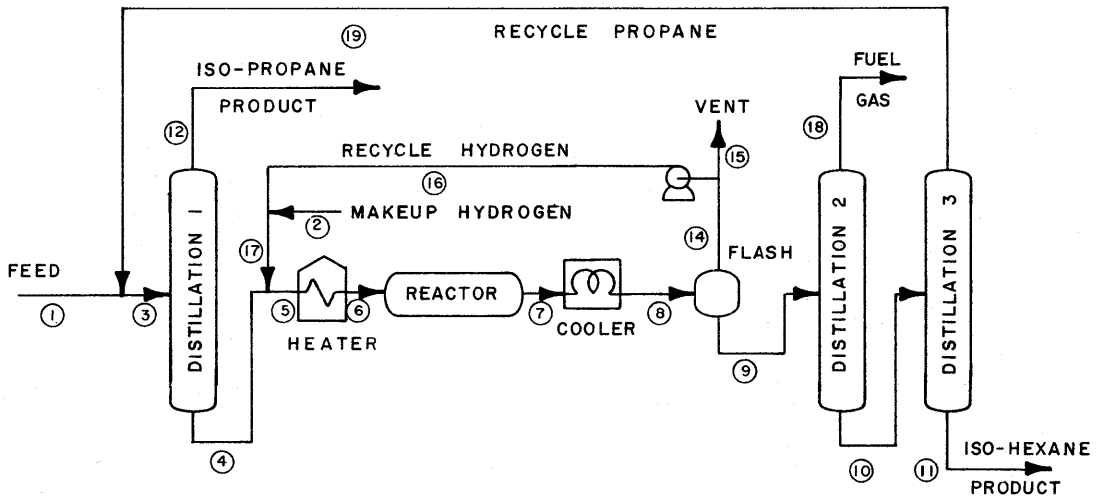


Figure 1. High-Temperature Isomerization

equipment. Distillation 1 also serves in preliminary preparation of the feed. Let us suppose that a unit similar to this is to be built for a particular refinery. Before individual pieces of equipment can be ordered or fabricated, it is necessary that their sizes and capacities be specified. This type of analysis is termed process design.

Prior to the general availability of computer methods, the process design engineer was forced to rely almost exclusively on information obtained from physical systems. Thus, if information about a similar unit, which was operating satisfactorily, was available, the design engineer merely gave the same or slightly modified specifications. For new processes, or processes for which comparison information was not available, he had to resort to scaleup from a pilot unit, construction of the pilot unit, in turn, being based on scaleup from a bench or laboratory unit. Thus, information required for construction of each process system was derived from a bench or laboratory model, through pilot unit, to production unit. This procedure is costly and time consuming. Further, the difficulties involved in experimenting with physical systems, especially of plant scale, all but prohibits investigation of anything more than minor design modifications.

As an alternate method of design, many computational methods are used. However, rigorous computations are too laborious for practical hand solution. As a consequence, only shortcut and approximate methods are used regularly, and their primary application is limited to supplementing plant and pilot information. Now that economical and reliable machine computations are readily available, lengthy, as well as complex, computational procedures are increasing in utility. Design is still ultimately based on comparison information, but this dependence is becoming less critical.

One of the many applications of computers in process design is in implementing the use of mathematical models. A mathematical model may be defined as a series of arithmetical operations performed to compute numerical values which represent certain performance characteristics of a physical system. Required input information are numerical values for design and operating conditions. A wide range in complexity is exhibited by the mathematical models which are programmed for computer solution. Some, such as for the mixing of two streams, are very simple, while others are very complex, requiring many hours of high-speed computer time for solution. The general concept, though, is the same. That is, given a numerical evaluation of operating and design conditions, the objective is to compute an estimate of performance characteristics for a corresponding physical system. Actually, this describes the simulation of a physical system using a mathematical model. An alternate type of mathematical model can be formulated for direct design. Accordingly, given numerical values for operating parameters and performance restrictions, numerical values for required design can be computed directly. Formulation of this type of model is normally much more difficult.

Many simulations for commonly used pieces of industrial equipment are being processed daily. However, this often requires theoretical isolation from the process system of which the piece of equipment is an integral part, and so may result in significant error. That is, the functional interrelationship of pieces of process equipment may be so significant, even with respect to the piece being studied, that gross errors in analysis may result when the effects of proposed modifications on the system are neglected. This is one of the primary motivations behind the development of Generalized Interrelated Flow Simulation, GIFS. That is, even more important than providing a convenient means of using preprogrammed mathematical models, GIFS provides a means of analyzing a process system as an integrated network.

The development of GIFS (Generalized Interrelated Flow Simulation) is based on study of a wide variety of industries. Representative among these are: petroleum refining, metals production, and manufacture of chemicals, pharmaceuticals, pigments, plastics, and pulp and paper. In essence, all of these represent special cases of continuous flow systems. Therefore, the design of GIFS is based not on the analysis of specific production systems, but rather, on the much broader scope of flow systems analysis in general.

The development of a generalized flow network method, which is easy to use as well as applicable to a wide variety of process systems, poses a number of difficulties. Outstanding among these is the ever-present

possibility of system over or under specification. That is, there is a finite number of variables which, when fixed, completely determine a system, and all other pertinent variables can be computed. There is always the possibility that an engineer will attempt to set either more or fewer variables than are required for complete specification of a process system. In order to insure against this possibility, GIFS has been purposely restricted to analysis of cases which correspond to physical systems that are completely determined and which are not overspecified. Thus, GIFS is applicable only to the generation of a mathematical model to simulate the functioning of an entire integrated process system. No attempt has yet been made to incorporate direct design aspects in this model. In short, the method can be used only for the performance evaluation of a fixed process system at fixed operating conditions and with fixed feeds. This is really not a drastic limitation since design as well as optimization can be approached through multiple case studies.

In application, each system under consideration is represented as a network of interrelated stream flows. Figure 1 is an example. Each stream is identified by assigning to it a unique stream number as indicated. Numerical values which are used to describe the nature of a stream are termed stream properties. Stream properties include total flow rate, composition, temperature, pressure, heat content, and phase state. Restrictive relationships among streams, such as are simulated by preprogrammed mathematical models for specific types of process equipment, are indicated by what is termed unit computations. Examples of unit computations include the simulation of process equipment, such as reactors, distillation columns, heat exchangers, etc., as well as factors which have no direct equipment counterparts (pressure drops in lines and ambient heat exchange). The currently available library includes 21 unit computations. These are summarized in Table 1. Complete descriptions are given in the GIFS user's manual.* They represent a collection which is basic in nature and highly versatile. Further, GIFS is constructed so as not to be limited to the library of unit computations available at any one time. At present, four additional unit computations are being prepared for inclusion in the library, and others can be added as specific needs arise.

Figure 2 is a reproduction of one page of the input data sheet which describes Figure 1 in terms of unit computations. Each unit computation is specified by giving the unit computation type, associated stream number or numbers, and arguments if any. A unit computation number is included for identification only. It is usually convenient to number unit computations sequentially. The first unit computation in Figure 2 indicates the addition of streams 1 and 19 to obtain stream 3. The unit computation type is STAD (Stream Add), and the associated stream numbers are 1, 19, and 3. The second unit computation indicates the function of distillation column number 1, the associated streams being 3, 12, and 4. The unit computation used is CRSEP (Component Ratio Separation). This is an approximate simulation of a distillation column which requires, as arguments, component recoveries for all components present in the feed (ratio of component flow rate recovered in the overhead product, stream 12, to component flow rate in the feed stream, stream 3). Since there are eleven components for the illustrative system, values for eleven arguments are entered. Additional unit computations necessary to describe the entire system are entered sequentially. Detailed information on the preparation of input data sheets is contained in the user's manual.*

The objective of each computation is an evaluation of all interrelated stream properties. These are computed as a function of the properties of the given feed streams. Each system is computed as an integrated network and in a manner which satisfies fundamental material balance relationships as well as the restrictions defined by the specified unit computations. Systems are normally non-linear, and a method of solution by iteration is employed. As an example of a computer output, just one portion of the report for the illustrative example is reproduced in Figure 3. This is the portion which describes computed properties for stream 9.

*GIFS, Generalized Interrelated Flow Simulation, user's manual is available through any SBC office or local sales representative.

SBC

GIFS
UNIT COMPUTATIONS

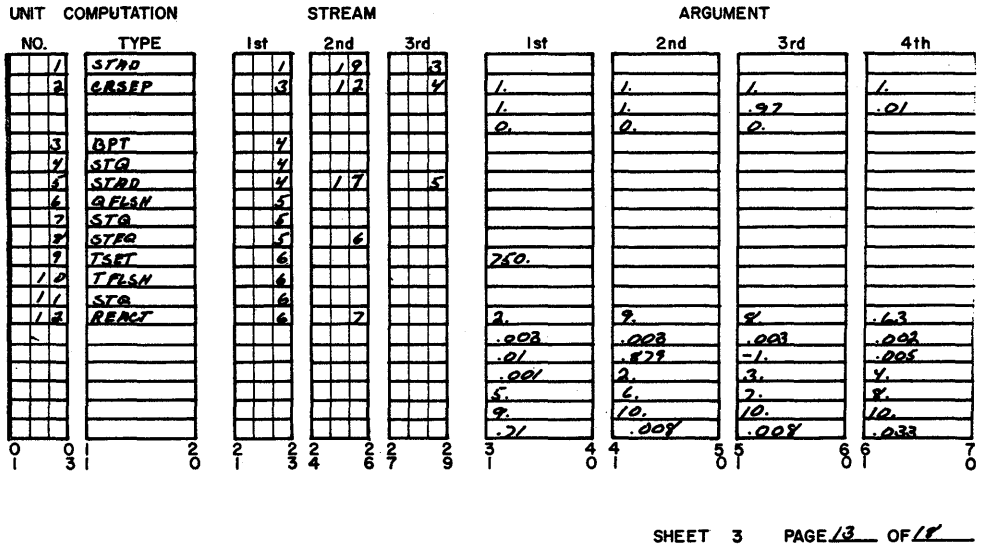


Figure 2. Sample Input Data Sheet

STREAM 9 FLASH LIQ

COMPONENT NO.	NAME	TOTAL FLOW MOLES/HR.	MOLE FR.	LBS./HR.	WT. FR.	VAPOR MOLES/HR.	LIQUID MOLES/HR.
1	H2	1.2926	0.0139	2.5853	0.0003	0.	1.2926
2	C1	0.3115	0.0033	4.9845	0.0007	0.	0.3115
3	C2	0.1001	0.0011	3.0124	0.0004	0.	0.1001
4	C3	0.0682	0.0007	3.0062	0.0004	0.	0.0682
5	I-C4	0.0472	0.0005	2.7415	0.0004	0.	0.0472
6	N-C4	0.2389	0.0026	13.8804	0.0018	0.	0.2389
7	I-C5	22.2530	0.2389	1604.4429	0.2125	0.	22.2530
8	N-C5	3.4096	0.0366	245.8286	0.0326	0.	3.4096
9	I-C6	18.3820	0.1973	1584.5274	0.2098	0.	18.3820
10	N-C6	44.9652	0.4827	3875.9987	0.5133	0.	44.9652
11	C7+	2.0943	0.0225	209.8512	0.0278	0.	2.0943
	TOTALS	93.1626	1.0000	7550.8590	1.0000	0.	93.1626

PRESSURE	300.00	PSIA.	TEMPERATURE	120.00	DEG. F.
VAPOR FLOW	0.	M CU. FT./HR.	LIQUID FLOW	25.04	GPM
(Z = 1.0)	0.	M STD. CU. FT./HR.			
HEAT CONTENT	-0.	MM BTU/HR.	FRACTION VAPOR	0.	

Figure 3. Computer Report

Table 1

Unit Computations Summary

Title	Type	Streams	Arguments
Stream Add	STAD	ST1 ST2 ST3	
Stream Subtract and Zero	STSBZ	ST1 ST2 ST3	
Stream Split	SPLT	ST1 ST2 ST3	F12
Stream Equate	STEQ	ST1 ST2	
Stream Zero	STZ	ST	
Temperature Add	TAD	ST	DT
Heat Add	QAD	ST	DQ
Pressure Add	PAD	ST	DP
Temperature Set	TSET	ST	T
Heat Set	QSET	ST	Q
Pressure Set	PSET	ST	P
Vapor Ratio Set	RSET	ST	R
Bubble Point	BPT	ST	
Dew Point	DPT	ST	
Isothermal Flash	TFLSH	ST	
Adiabatic Flash	QFLSH	ST	
Constant R Flash	RFLSH	ST	
Stream Heat	STQ	ST	
Equilibrium Separation	EQSEP	ST1 ST2 ST3	
Component R Separation	CRSEP	ST1 ST2 ST3	R1 R2 --- ---- RN
Reactor	REACT	ST1 ST2	NR NC1 NK1 CNV1 S1 ---------- SNC1 C1 ---------- CNC1 ---------CNCNR

GIFS has been developed, and is now being offered, as one of SBC's preprogrammed computer services. Characteristics of these services include accurate, inexpensive, and rapid processing for a wide variety of problems, use of convenient input data sheets, and the presentation of computed values in clear, comprehensive reports. Cases which have been processed to date represent applications in such fields as petroleum refining, inorganic and organic chemicals manufacture, and pulp and paper production. These have been processed in a routine manner, and have not indicated any unforseen complications in the approach or the computer implementation. From all indications, companies who are using the service are highly satisfied, and expect to continue using it. Future developmental efforts will depend on industrial response. SBC is already contemplating the implementation of a number of additional features.

ACKNOWLEDGMENTS

Appreciation is expressed to Service Bureau Corporation and to IBM for supplying the materials, equipment, and encouragement needed for this project, and to the SBC personnel who assisted in its completion. Appreciation is also expressed to Prof. L. M. Naphtali for his original developmental efforts, and to Dr. J. E. Brock for his assistance in the preparation of this documentation.

A DATA COMMUNICATIONS AND PROCESSING SYSTEM FOR CARDIAC ANALYSIS

M. D. Balkovic
Bell Telephone Laboratories
Holmdel, New Jersey

*P. C. Pfunke**
A. T. & T. Company
New York City

C. A. Steinberg
Department of Medical and Biological Physics
Airborne Instruments Laboratory
Deer Park, Long Island, New York

C. A. Caceres, M. D.
Chief, Instrumentation Unit
Heart Disease Control Program
U. S. Department of Health, Education, and Welfare
Washington 25, D. C.

Many aspects of medical diagnoses involve extensive, tedious procedures in which the capabilities of a digital computer can provide the physician an invaluable aid.† Because of the complexity and cost of modern computers, systems to aid in diagnostic procedures must generally be centrally located and be capable of serving many physicians. Thus, data communication links between the physicians and the computer location are required. Such a data communication and processing system for cardiac analysis is now in operation and is described in this paper.‡

The cardiac data processing system discussed in this paper is comprised of data acquisition units located throughout the country, data communication links which can be established as required, a data processing unit, and print-out devices.

The data acquisition unit, as shown in Figure 1, provides the capability of recording an electrocardiographic signal simultaneously on a graphical recorder and on a magnetic tape recorder. Prior to recording the electrocardiogram, an 8-digit, serial, binary-coded

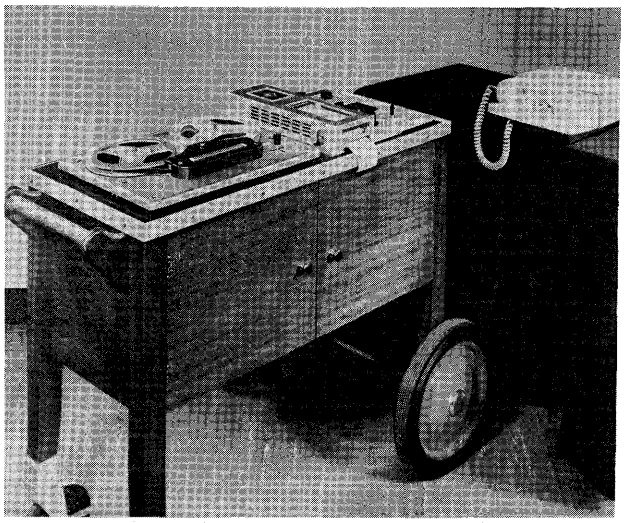


Figure 1. Data Acquisition Unit.

*Will present paper.
†Reference #1.
‡This is a pilot project facility of the Instrumentation Unit, Heart Disease Control Program, Division of Chronic Diseases, U.S. Dept. of Health, Education and Welfare, Washington 25, D. C.

decimal number is generated and recorded on magnetic tape. This 8-digit number is used to identify the particular data that is recorded. Two digits indicate the place where a recording is taken; four digits indicate the patient's number, and the last two digits indicate the particular electrocardiographic lead that is recorded. The electrocardiographic signal is modulated using pulse repetition frequency modulation prior to recording on magnetic tape in order to achieve a frequency response down to 0.1 cycles per second. The upper frequency response extends to 200 cycles per second.

Figure 2 is a block diagram of the data acquisition unit. The input electrocardiogram is amplified prior to recording. A coder generates the 8-digit number corresponding to the location, patient and lead.

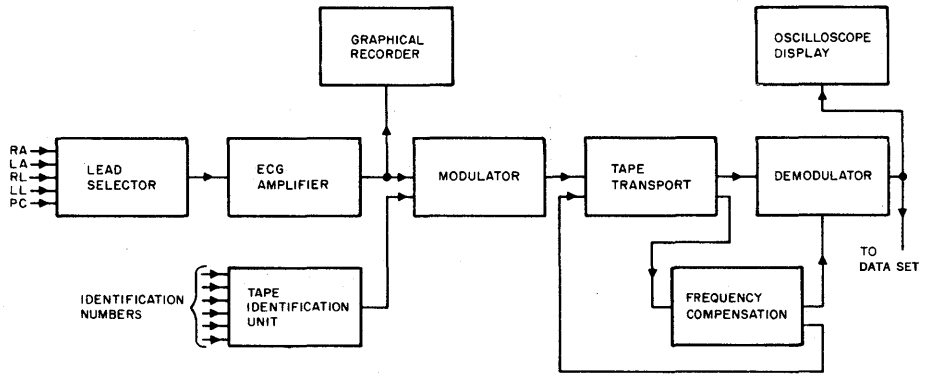


Figure 2. Data Acquisition Unit.

The code number followed by the amplified electrocardiogram is fed to pulse repetition frequency modulator whose center frequency is 3600 cycles per second. The output of the modulator is recorded on magnetic tape. The output of a second head on a tape recorder is demodulated and the demodulated signal is fed to a graphical recorder and/or a DATA-PHONE data set for transmission to the data processing unit. The magnetic tape records generated by the data acquisition unit contain completely identified electrocardiographic signals. These magnetic tape records can be sent via DATA-PHONE service to the data processing unit for subsequent analysis and processing.

Communication between the data acquisition units and the data processing unit is achieved over DATA-PHONE service on switched telephone facilities. Specially designed and constructed analog transmitting data sets are located at the data acquisition units while receiving data sets are located at the data processing unit. These analog data sets incorporate integrated telephone sets which allow, when in the voice mode of operation, the dialing of calls and normal voice communication over the regular switched telephone network. When in the data mode of operation, the data set provides modulation and demodulation circuitry to allow the transmission of analog signals with frequencies from 0 to 200 cps over dialed-up voice telephone facilities. An input to output linearity from transmitting to receiving data sets is maintained to better than 1%.

Figure 3 shows a block diagram of the electrocardiogram data set transmitter. The input circuitry provides a load impedance of 1000 ohms and accepts analog signals which can range from -3 volts to +3 volts and can contain frequency components between 0 and 200 cps. The input circuitry acts to adjust the DC level and gain of the input signal such that it provides proper bias range for the astable multivibrator which follows the input circuit. This astable multivibrator accomplishes the actual frequency modulation. A

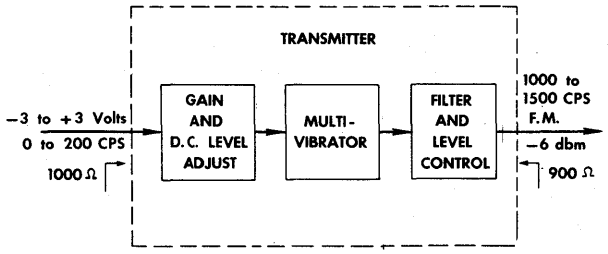


Figure 3. Data Set Transmitter.

given voltage delivered to the input of the data set represents a particular bias on the multivibrator and determines the multivibrator's frequency of oscillation. For the indicated input voltage range, the astable multivibrator will oscillate at frequencies ranging from 1000 cps to 1500 cps when the input circuit level and gain controls are properly adjusted. The square wave output of the multivibrator is filtered and attenuated by the transmitter output circuitry. The output circuitry provides a 900 ohm source impedance to the telephone line. The signal transmitted to the telephone line is essentially a sine wave with a fundamental frequency between 1000 cps and 1500 cps at a level of -6 dbm. A bandpass filter removes harmonics above 2500 cps.

Figure 4 shows a block diagram of the electrocardiogram data set receiver. The input bandpass filter terminates the telephone line in 900 ohms at all transmitting frequencies. It has a slope of 12 db/octave to either side of the 1250 cps center frequency of the telephone line signal. Following the bandpass filter is a combined amplifier and limiter, the gain of which determines the sensitivity of the receiver, -38 dbm. The output of this stage is a 1.5 volt square wave with a fundamental frequency that of the

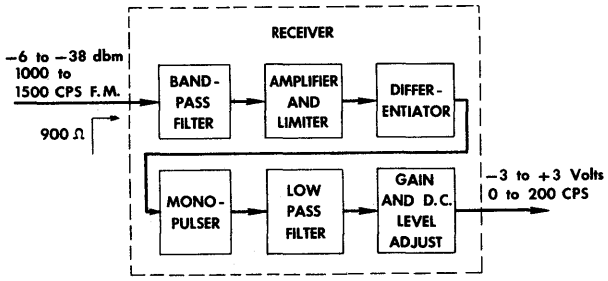


Figure 4. Data Set Receiver.

received signal. The square wave is differentiated and full-wave rectified to give a sequence of pulses which correspond to the zero crossings of the received signal. These pulses are used to trigger a mono-pulser which delivers a pulse of fixed width each time it is triggered. Thus, the mono-pulser delivers an output pulse for every zero crossing of the input signal from the telephone line and provides an average output voltage which is proportional to the received signal frequency. By passing the output of the mono-pulser through a low pass filter, the original baseband signal is reconstructed. The output circuitry serves to amplify the signal and restore the correct DC signal level. This output circuitry is designed to drive a 1000 ohm load impedance.

The telephone line signal spectrum (1000 cps to 1500 cps) is chosen to avoid falling within the same band used by various kinds of single-frequency signalling equipment employed in Bell System switching facilities. If this were not true, automatic circuit disconnect could occur when certain signal patterns are transmitted. The input sensitivity of the data set receiver is chosen to be appropriate for the range of typical dialed-up telephone connections.

Figure 5 is a block diagram of the equipment in the data processing unit. This consists of three basic units; an input console, a digital computer, and an output unit. The input console can accept data that is transmitted over the telephone line or data obtained from playing back magnetic tape records made in the field. The data transmitted over the telephone line is recorded on one of the two magnetic tape recorders. If it is desired, the data transmitted over the telephone line can be recorded on magnetic tape and fed into the computer simultaneously.

The input console is capable of automatically searching the magnetic tape for any pre-selected set of identification numbers.* Once finding the proper set of identification numbers, these numbers are decoded and fed into the computer and serve to identify the result of the data processing. Alternately, the system can accept any electrocardiogram and set of identification numbers, whether transmitted over the telephone line or played back from a magnetic tape recording, decode the identification and feed the identification number into the digital computer.

An oscilloscope and photographic recorder are incorporated into the system to monitor the electrocardiogram while recording and playing back. The electrocardiogram is filtered with a bandpass filter to eliminate high frequency noise. The derivative of the filtered electrocardiogram is then obtained using conventional analog techniques. The filtered electrocardiogram and the derivative of the electrocardiogram are converted to digital form at a rate of 500 samples per

*Reference #2.

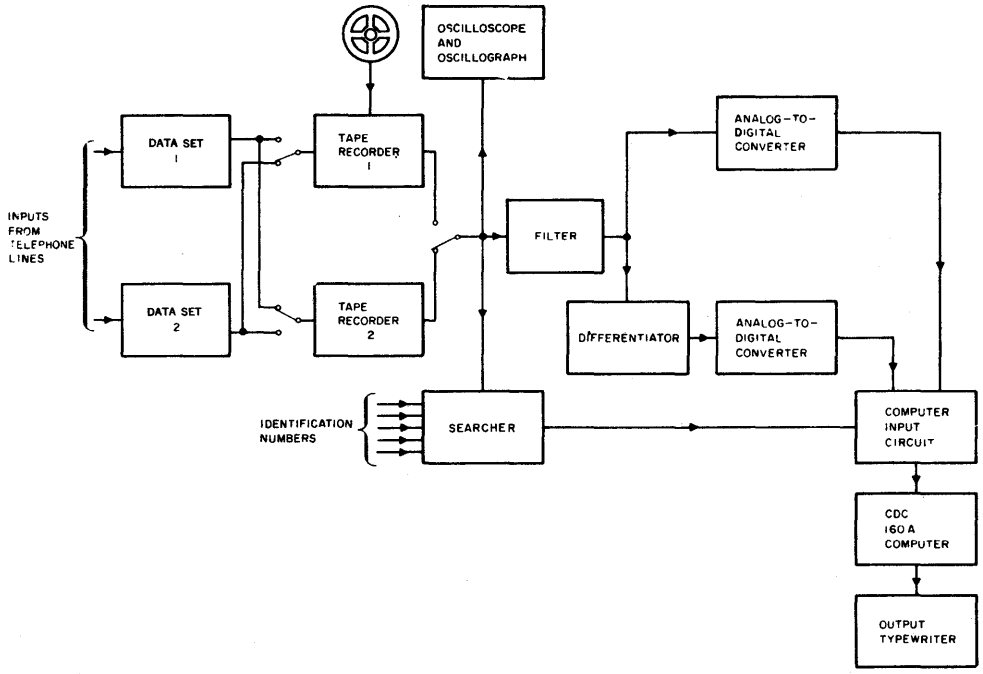


Figure 5. The Data Processing Center.

second using a successive approximation analog to digital converter. The digital representation of these two signals, along with the decoded identification numbers, are then fed directly to the digital computer for subsequent processing. Figure 6 is a photograph of the system. The input console is at the left and the computer is at the right.

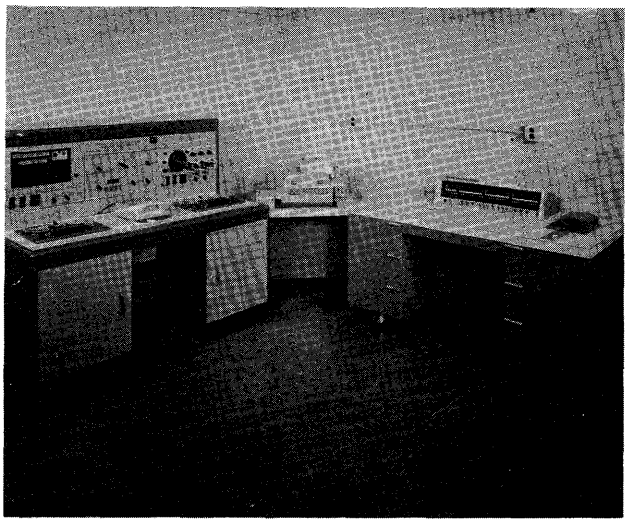


Figure 6. Data Processing Unit.

The digital computer is a Control Data Corporation 160A computer (CDC 160A) and is programmed to automatically recognize and measure the wave forms of the electrocardiogram.* The cardiologist uses these measurements to interpret the electrocardiogram. The measurements are of amplitudes and duration of the waves, and the interval between certain of them. The program recognizes whether some waves are diphasic, bi-fed or tri-fed. The measurements made by the computer were programmed based on conventional EKG criteria for wave onset, termination and slope.†

The computer processes an electrocardiographic lead and arrives at the desired measurements in 12 seconds. The entire input, process, and write-out time is 52 seconds. The output of the computer is printed on an automatic typewriter or punched on paper tape that can be used for telephone transmission of the results back to the sender. Although at present verbal telephone communication is used for return of results, a completely electronic system has been tried.‡

*Reference #3.
†Reference #4.
‡Reference #5.

The data communication link used for output data from the data processing unit has consisted of DATA-PHONE service over switched telephone facilities. Bell System Data Sets 402A and 402B have been used in conjunction with Tally Register Corporation Model 51 tape-to-tape equipment. This communication link is capable of transmitting 8-level parallel digital data at a rate of 75 characters per second.

The approach taken in the system described in this paper can be extended to other medical diagnostic processes such as the analysis of phono-cardiograms, electroencephalograms, etc. There is great potential here for electronics to assist the medical profession.

REFERENCES

1. Rikli, Arthur E. and Caceres, Cesar A., 1960. The Use of Computers by Physicians as a Diagnostic Aid. Reprinted from The New York Academy of Sciences Ser. II, Volume 23, No. 3, Pages 237-239.
2. Paine, L. W., and Steinberg, C. A. A Medical Magnetic Tape Coding and Searching System. Presented to the Spring, 1962 IRE Convention.
3. Steinberg, C.A., Abraham, S., and Caceres, C. A. Pattern Recognition in the Clinical Electrocardiogram. Presented at The Fourth International Conference on Medical Electronics, New York, New York, July, 1961
4. Caceres, Cesar A. How Can the Waveforms of a Clinical Electrocardiogram be Measured Automatically by a Computer? Presented at a meeting of the Professional Group on Bio-Medical Electronics (local chapter) New York, New York, February, 1960.
5. Rikli, Arthur E., Caceres, Cesar A., and Steinberg, C. A. Electrocardiography with Electronics: An Experimental Study (Exhibit) American Heart Association National Meeting October, 1962.

CLUSTER FORMATION AND DIAGNOSTIC SIGNIFICANCE IN PSYCHIATRIC SYMPTOM EVALUATION*

Gilbert Kaskey
Engineering Director, Systems Design and Applications Division
Remington Rand Univac

Paruchuri R. Krishnaiah
Senior Statistician, Applied Mathematics Department
Remington Rand Univac

Anthony Azzari
Senior Programmer, Applied Mathematics Department
Remington Rand Univac

SUMMARY

The tremendous variability in symptom constellations associated with specific psychiatric diagnoses makes the use of statistical techniques a virtual necessity in the rigid formulation of any symptom-disease model. In addition, the large number of symptoms associated with psychiatric disorders suggest that an electronic computer can be used to considerable advantage in certain aspects of psychiatric diagnosis, e.g., in the correlation analysis between symptoms, in the determination of quantitative criteria for diagnosis, and in the evaluation of the reliability of symptom assignment by different personnel.

This paper reports on the results obtained from the analysis of data on 199 subjects collected by the Childrens Unit of the Eastern Pennsylvania Psychiatric Institute. The types of information available, the method of collection, and the statistical methodology and techniques are all discussed along with the results obtained using the tools of correlation analysis and simultaneous multivariant analysis of variants.

INTRODUCTION

The fact that electronic computers can be used effectively in certain phases of medical diagnosis has been recognized for some time. It has been only recently, however, that applications have been extended to include research investigations in psychotherapy generally, and symptom pattern formation specifically. Although the present study has not extended beyond its initial phase, preliminary findings suggest that the merger of computer techniques with clinical observations can do a great deal to shed more light on the problems of emotionally disturbed children.

The extreme variability in symptom constellations associated with specific

*The authors are indebted to Mrs. Janice Schulman, Research Associate, and Dr. Robert Prall, Director of the Children's Unit of the Eastern Pennsylvania Psychiatric Institute for their invaluable aid in formulating the problem areas and in interpreting the results.

psychiatric diagnoses makes the use of statistical techniques a virtual necessity in the rigid formulation of any symptom-disease model. In addition, the large number of symptoms associated with psychological and psychophysiological disorders suggests that an electronic computer can be used to considerable advantage in many aspects of this diagnosis. For example, the objective determination of symptom clusters using correlation analysis and the statistical testing of significance among alternative diagnoses require an excessive amount of computation which is only feasible when handled by a computer.

This report on the initial phase of the investigation describes the clinical information collection techniques, the psychiatric categories assigned, and the statistical methodology applied in the determination of the preliminary results.

The ultimate goal of the project is to evolve, through analysis of the data, a picture of how the various symptoms tend to form patterns or "clusters." It is hoped that these clusters can be related directly to the relevant category as an aid in the problem of differential diagnosis. Associated with this investigation is an evaluation of the diagnostic categories themselves and the determination, from a statistical standpoint, of the significance among the several categories using the "symptom vectors" as criteria. The mathematical details of the technique used in this phase—Simultaneous Multivariate Analysis of Variance—are given in the Appendix.

Data Description

The Children's Unit at the Eastern Pennsylvania Psychiatric Institute has been collecting data regarding symptom formation in emotionally disturbed children since 1956. An enumeration of the emotional problems exhibited by each child seen in the Clinic—which is both an inpatient and outpatient treatment center—is obtained in a highly structured interview with each parent who is seen individually. In addition to whether the child possesses any of the 130 symptoms commonly encountered in clinical practice (see Appendix, Table A-1), information with regard to (1) duration of the symptom; (2) whether the symptom is currently characteristic of the child or existed only in the past; and (3) whether the parent considers it as "serious" or "not so serious," is also obtained.

Several assumptions were necessary in order to make the data amenable to computational techniques. Since additional evidence is now being collected on "normal controls," i.e., on non-patient children of various ages, and since consistency checks between parents are being made, it is expected that the validity of these assumptions will become apparent in later studies; in any case they are such that the final results will not be seriously affected by their acceptance.

A symptom is considered to be present if either parent says it is present. If only one parent sees a particular symptom, the designation as to the severity and duration is determined by the response of the parent who sees it even though it may be listed by the other as not characteristic of the child.

If the information elicited indicates that both parents agree on the relevance of the given symptom, but disagree as to the specific designation, the following rules determine how the response is counted:

a. If one parent says the symptom exists at present while the other says it existed only in the past, the symptom is listed at present.
b. If one parent says present always and the other says present recent, the former designation is used.
c. If one parent considers the symptom serious but the other does not, it is listed as serious.

Other basic sociological data are available on each of the 199 cases being investigated but no attempt at analysis has been made. Presumably such information as age, sex, religion, race, and number of parents interviewed, will provide fruitful areas for future study.

Only four of the fifteen diagnostic categories are being studied in detail since in many instances the data available in the remainder represent too few cases from which to draw meaningful results. Further, the categories finally selected are those of greatest interest to the research personnel at the Children's Unit with whom the study is being made. Descriptive definitions* are given below to enable the reader to relate the symptom patterns, as indicated by the correlation analysis described in the next

*See Diagnostic and Statistical Manual of Mental Disorders, The American Psychiatric Association, Washington, D. C.

section, to the emotional disturbance classifications tested in the section entitled "Diagnostic Significance."

Childhood Schizophrenia - this category, also known as childhood psychosis, is characterized by varying degrees of personality disintegration and failure to test and evaluate external reality correctly. Children in this group fail in their ability to relate themselves effectively to other people and to such tasks as school work. They may exhibit unpredictable behavior, regression to earlier childhood forms of behavior, and uneven rates of development of any area of bodily and mental development. The onset of this condition may be from birth or may come later on in childhood and is often gradual.

Psychophysiological Disorders - such disorders are sometimes referred to as psychosomatic disorders and represent organic disturbances which are induced by mental or emotional stimuli.

Psychoneurotic Disorders - these are characterized chiefly by "anxiety" which may either be felt and expressed directly or may be unconsciously and automatically controlled by various psychological defense mechanisms.

Personality Trait Disturbances - disorders of this sort are characterized by developmental defects or pathological trends in the personality structure, with minimal subjective anxiety, and little or no sense of distress. Usually it is manifested by long standing patterns of action or behavior, rather than mental or emotional symptoms seen in neuroses and psychoses.

The 199 cases under consideration have each been studied by professional personnel at the Clinic prior to having a final diagnosis assigned.

Symptom Cluster Formation

A major purpose of the investigation has been the objective determination of symptom clusters associated with each individual diagnostic group and also with all groups together. The analysis has been aimed, then, at answering the two questions:

1. Is there a tendency for specific symptoms to be more frequently accompanied by other symptoms?
2. Are there specific symptoms or clusters of symptoms associated with certain diagnostic categories?

The intercorrelations between symptoms have been examined statistically and classed into overlapping and non-overlapping clusters.

A cluster is defined to be overlapping if all the symptoms in the cluster are correlated significantly with one another.

A cluster is defined to be non-overlapping if each symptom in the cluster is correlated significantly with one or more, but not all, of the other symptoms.

The correlation between any two symptoms is given by

$$r_{ij} = \frac{(N N_{ij} - N_i N_j)}{\sqrt{N_i N_j (N - N_i)(N - N_j)}},$$

where

N = total number of subjects
N_i = number of subjects with symptom i
N_j = number of subjects with symptom j
N_{ij} = number of subjects with symptoms i and j.

The hypothesis that no correlation exists between symptom i and symptom j can be easily tested by using the fact that

$$t = \frac{r_{ij}\sqrt{N - 2}}{\sqrt{1 - r^2_{ij}}}$$

is distributed as Student's "t" with (N - 2) degrees of freedom. We accept or reject the hypothesis according as

$$t_{ij} \lessgtr t_\alpha,$$

where t_α = the appropriate "t" table value at the α significance level.

It is important to note that the application of this significance test is valid only when the distribution of individuals for a given symptom is normal. Although the symptom data are discrete and binomial—since each individual is assigned a "zero" or a "one" depending on whether the symptom is absent or present—the sample size would seem to be large enough to take advantage of the fact

that the binomial tends to normality as the sample size increases. Moreover, the discreteness of the data is due more to the lack of precision in measuring than to any inherent lack of continuity in the underlying scale.

A simple frequency count of the 20 most frequently occurring symptoms in each diagnostic group and also in all diagnostic groups taken as a single entity has been made with the following results:

Symptoms

Diagnostic group 1: (Psychological Disorders)	2, 9, 11, 13, 14, 15, 18, 25, 26, 39, 45, 47, 50, 52, 60, 61, 64, 65, 68, 79.
Diagnostic group 2: (Personality Trait Disturbances)	1, 10, 11, 13, 14, 15, 26, 28, 33, 37, 39, 40, 43, 47, 56, 57, 60, 61, 66, 68.
Diagnostic group 3: (Childhood Schizophrenia)	10, 11, 13, 16, 17, 18, 23, 25, 26, 28, 31, 33, 53, 59, 60, 64, 68, 73, 76, 80.
Diagnostic group 4: (Psychophysiological Disorders)	1, 9, 10, 11, 13, 14, 15, 18, 21, 26, 39, 40, 47, 56, 57, 60, 61, 66, 68, 69.
All groups:	1, 9, 10, 11, 13, 14, 15, 17, 18, 21, 26, 28, 33, 39, 47, 56, 57, 60, 61, 68.

The correlation between each pair of symptoms listed within the indicated category is shown in Tables I through V where a star in a particular cell of the matrix indicates significance at the 5% level. The critical values for diagnostic groups 1, 2, 3, and 4 are 0.514, 0.344, 0.269, and 0.250, respectively; for all diagnostic groups the figure is 0.138. The non-overlapping symptom clusters, determined on the basis of significant correlations, are formed as indicated below.

Non-Overlapping Clusters

Group 1, Cluster 1:	11, 13, 14, 15
Group 1, Cluster 2:	2, 9, 18, 25, 26, 39, 45, 47, 50, 60, 64, 65
Group 2, Cluster 1:	10, 11, 13, 14, 15, 26, 28, 33, 37, 39, 40, 43, 56, 57, 60, 61, 66, 68
Group 3, Cluster 1:	10, 11, 13, 16, 17, 18, 23, 25, 31, 33, 59, 60, 68, 73
Group 4, Cluster 1:	1, 9, 10, 11, 13, 14, 15, 18, 21, 26, 39, 40, 47, 56, 57, 60, 61, 66, 68, 69
All Groups, Cluster 1:	1, 9, 10, 11, 13, 14, 15, 17, 18, 21, 26, 28, 33, 39, 47, 56, 57, 60, 61, 68

The above listing clearly indicates that virtually all symptoms within each group are indirectly interrelated with the exception of those in diagnostic group 1. In an attempt to study the direct relationships which exist, the overlapping clusters (as defined above) were determined.

Overlapping Clusters

Group 1:	2, 50, 64; 9, 26; 11, 13; 13, 15; 14, 15; 18, 26; 25, 39; 25, 45; 25, 50; 25, 65; 39, 45; 39, 50; 39, 65; 45, 50; 45, 65; 50, 65; 26, 45; 39, 47; 47, 60; 47, 65; 60, 65.
Group 2:	10, 11, 13, 37, 56; 10, 14, 56; 11, 37, 43, 56; 11, 40; 14, 26; 14, 56, 60; 15, 26, 33, 61; 15, 37; 28, 33; 28, 60; 39, 40; 40, 61; 43, 56, 68; 43, 57; 56, 60, 68; 57, 61; 66, 68.
Group 3:	10, 13, 17, 33; 10, 23, 73; 11, 17; 13, 33, 60; 13, 60, 68; 16, 17, 33; 16, 68; 16, 73; 18, 33; 23, 25, 31, 59; 23, 31, 59, 73.
Group 4:	1, 9, 47; 1, 47, 56, 60; 9, 10, 11, 13; 9, 10, 11, 40, 47; 9, 11, 15; 9, 47, 57; 9, 61; 10, 11, 13, 21; 10, 11, 21, 47; 10, 11, 21, 68; 11, 15, 18; 11, 15, 21; 13, 14, 21; 13, 26; 14, 21, 47; 14, 40, 47; 15, 18, 26; 26, 66; 39, 40; 39, 57; 47, 57; 47, 66; 68, 69.
All Groups:	1, 9, 39; 1, 17, 18; 1, 9, 47; 1, 47, 56, 60; 9, 10, 14, 15; 9, 14, 47; 9, 15, 61; 9, 68; 10, 11, 13, 15, 17; 10, 13, 14, 15, 17, 21; 10, 15, 17, 21, 33; 10, 21, 33, 60; 14, 17, 21, 28; 14, 47; 15, 17, 18, 33; 15, 18, 26; 15, 56; 15, 61; 17, 21, 28, 33; 21, 28, 33, 60; 21, 28, 33, 68; 28, 39; 28, 60, 61; 57, 60, 61.

TABLE I

Correlation Matrix for Diagnostic Group 1

	2	9	11	13	14	15	18	25	26	39	45	47	50	52	60	61	64	65	68	79	
2	+1.00	+.354	-.200	-.200	+.100	-.426	+.139	+.100	+.378	+.213	+.000	-.200	+.577*	-.107	+.000	+.289	-.577*	+.000	-.354	-.200	2
9	+.354	+1.00	-.354	+.000	+.354	+.075	+.294	+.000	+.535*	+.075	+.167	+.000	+.272	-.302	+.272	+.272	-.408	+.272	-.250	-.354	9
11	-.200	-.354	+1.00	+.700*	+.100	+.213	-.277	-.200	-.189	-.426	+.000	+.100	-.289	+.213	+.000	-.289	+.289	+.000	+.000	+.100	11
13	-.200	+.000	+.700*	+1.00	+.400	+.533*	-.277	-.200	-.189	-.426	+.000	+.100	-.289	+.213	+.000	-.289	+.000	+.289	+.000	-.200	13
14	+.100	+.354	+.100	+.400	+1.00	+.533*	+.139	+.100	+.378	-.107	+.354	+.100	+.000	-.107	-.289	-.289	+.000	+.289	-.354	-.500	14
15	-.426	+.075	+.213	+.533*	+.533*	+1.00	-.237	-.107	-.161	-.364	+.075	+.213	-.492	+.318	-.185	-.185	+.123	+.431	-.302	-.426	15
18	+.139	+.294	-.277	-.277	+.139	-.237	+1.00	+.139	+.681*	+.207	+.294	+.139	+.080	+.207	+.080	+.480	+.080	+.080	+.294	+.139	18
25	+.100	+.000	-.200	-.200	+.100	-.107	+.139	+1.00	+.378	+.533*	+.707*	+.400	+.577*	+.213	+.000	+.289	-.289	+.577*	+.000	+.100	25
26	+.378	+.535*	-.189	-.189	+.378	-.161	+.681*	+.378	+1.00	+.443	+.535*	+.378	+.327	-.161	+.327	+.327	-.218	+.327	-.134	-.189	26
39	+.213	+.075	-.426	-.426	-.107	-.364	+.207	+.533*	+.443	+1.00	+.075	+.533*	+.431	-.023	+.431	+.431	-.185	+.431	+.075	-.107	39
45	+.000	+.167	+.000	+.000	+.354	+.075	+.294	+.707*	+.535*	+.075	+1.00	+.354	+.272	+.075	-.068	-.068	-.068	+.272	-.250	+.000	45
47	-.200	+.000	+.100	+.100	+.100	+.213	+.139	+.400	+.378	+.533*	+.354	+1.00	+.000	+.213	+.577*	+.289	+.000	+.577*	+.000	-.500	47
50	+.577*	+.272	-.289	-.289	+.000	-.492	+.080	+.577*	+.327	+.431	+.272	+.000	+1.00	-.185	+.167	+.444	-.667*	+.167	-.068	+.000	50
52	-.107	-.302	+.213	+.213	-.107	+.318	+.207	+.213	-.161	-.023	+.075	+.213	-.185	+1.00	-.185	+.431	+.123	+.431	+.075	+.213	52
60	+.000	+.272	+.000	+.000	-.289	-.185	+.080	+.000	+.327	+.431	-.068	+.577*	+.167	-.185	+1.00	+.444	-.111	+.167	+.272	-.289	60
61	+.289	+.272	-.289	-.289	-.289	-.185	+.480	+.289	+.327	+.431	-.068	+.289	+.444	+.431	+.444	+1.00	-.389	+.444	+.272	+.000	61
64	-.577*	-.408	+.289	+.000	+.000	+.123	+.080	-.289	-.218	-.185	-.068	+.000	-.667*	+.123	-.111	-.389	+1.00	-.389	+.272	+.289	64
65	+.000	+.272	+.000	+.289	+.289	+.431	+.080	+.577*	+.327	+.431	+.272	+.577*	+.167	+.431	+.167	+.444	-.389	+1.00	-.068	-.289	65
68	-.354	-.250	+.000	+.000	-.354	-.302	+.294	+.000	-.134	+.075	-.250	+.000	-.068	+.075	+.272	+.272	+.272	-.068	+1.00	+.354	68
79	-.200	-.354	+.100	-.200	-.500	-.426	+.139	+.100	-.189	-.107	+.000	-.500	+.000	+.213	-.289	+.000	+.289	-.289	+.354	+1.00	79

TABLE II

Correlation Matrix for Diagnostic Group 2

	1	10	11	13	14	15	26	28	33	37	39	40	43	47	56	57	60	61	66	68	
1	+1.00	+.000	+.160	+.067	+.021	+.021	+.124	-.020	+.021	-.020	+.250	+.199	+.280	-.134	+.250	+.043	+.160	+.043	+.021	+.067	1
10	+.000	+1.00	+.418*	+.800*	+.373*	+.233	+.167	+.000	+.233	+.535*	-.326	+.000	+.289	+.060	+.373*	+.000	+.060	+.000	+.093	-.100	10
11	+.160	+.418*	+1.00	+.353*	+.273	+.273	+.239	-.144	+.089	+.383*	+.089	+.383*	+.500*	+.057	+.457*	+.311	+.057	+.311	+.089	+.155	11
13	+.067	+.800*	+.353*	+1.00	+.242	+.089	-.083	+.160	+.242	+.454*	-.219	+.013	+.289	+.155	+.396*	+.130	+.155	-.029	+.242	+.010	13
14	+.021	+.373*	+.273	+.242	+1.00	+.283	+.373*	+.187	+.283	+.324	-.004	+.187	+.336	+.089	+.570*	+.336	+.457*	+.336	+.139	+.242	14
15	+.021	+.233	+.273	+.089	+.283	+1.00	+.373*	+.187	+.426*	+.461*	-.004	+.187	+.188	+.089	+.283	+.188	+.089	+.485*	+.139	+.242	15
26	+.124	+.167	+.239	-.083	+.373*	+.373*	+1.00	+.134	+.373*	+.297	+.202	+.297	+.241	-.199	+.202	+.064	+.020	+.417*	-.140	+.100	26
28	-.020	+.000	-.144	+.160	+.187	+.187	+.134	+1.00	+.461*	+.083	+.050	-.048	-.039	+.032	+.187	+.244	+.559*	+.244	+.324	+.307	28
33	+.021	+.233	+.089	+.242	+.283	+.426*	+.373*	+.461*	+1.00	+.187	-.004	+.050	+.040	+.089	-.004	+.336	+.089	+.485*	+.139	+.242	33
37	-.020	+.535*	+.385*	+.454*	+.324	+.461*	+.297	+.083	+.187	+1.00	-.087	+.214	+.386*	-.144	+.461*	+.103	+.208	+.103	+.050	+.307	37
39	+.250	-.326	+.089	-.219	-.004	-.004	+.202	+.050	-.004	-.087	+1.00	+.735*	-.108	+.089	-.004	+.188	+.089	+.188	+.139	+.089	39
40	+.199	+.000	+.383*	+.013	+.187	+.187	+.297	-.048	+.050	+.214	+.735*	+1.00	+.103	+.208	+.187	+.103	+.032	+.386*	+.187	+.013	40
43	+.280	+.289	+.500*	+.289	+.336	+.188	+.241	-.039	+.040	+.386*	-.108	+.103	+1.00	+.121	+.633*	+.389*	+.121	+.083	+.188	+.447*	43
47	-.134	+.060	+.057	+.155	+.089	+.089	-.199	+.032	+.089	-.144	+.089	+.208	+.121	+1.00	+.089	+.311	+.057	+.121	+.273	-.042	47
56	+.250	+.461*	+.457*	+.396*	+.570*	+.283	+.202	+.187	-.004	+.461*	-.004	+.187	+.633*	+.089	+1.00	+.336	+.457*	+.188	+.139	+.396*	56
57	+.043	+.000	+.311	+.130	+.336	+.188	+.064	+.244	+.336	+.103	+.188	+.103	+.389*	+.311	+.336	+1.00	+.311	+.389*	+.188	+.289	57
60	+.160	+.060	+.057	+.155	+.457*	+.089	+.020	+.559*	+.089	+.208	+.089	+.032	+.121	+.057	+.457*	+.311	+1.00	+.311	+.273	+.353*	60
61	+.043	+.000	+.311	-.029	+.336	+.485*	+.417*	+.244	+.485*	+.103	+.188	+.386*	+.083	+.121	+.188	+.389*	+.311	+1.00	+.188	+.130	61
66	+.021	+.093	+.089	+.242	+.139	+.139	-.140	+.324	+.139	+.050	+.139	+.187	+.188	+.273	+.139	+.188	+.273	+.188	+1.00	+.396*	66
68	+.067	-.100	+.155	+.010	+.242	+.242	+.100	+.307	+.242	+.307	+.089	+.013	+.447*	-.042	+.396*	+.289	+.353*	+.130	+.396*	+1.00	68

TABLE III

Correlation Matrix for Diagnostic Group 3

	10	11	13	16	17	18	23	25	26	28	31	33	53	59	60	64	68	73	76	80	
10	+1.00	+.242	+.384*	+.087	+.341*	+.213	+.341*	+.232	+.141	+.004	+.018	+.426*	+.044	+.141	+.153	-.101	+.070	+.293*	-.064	-.015	10
11	+.242	+1.00	+.179	+.095	+.447*	-.009	+.257	-.058	+.100	-.009	-.042	+.123	-.103	-.042	-.103	+.112	-.006	-.024	-.103	+.095	11
13	+.384*	+.179	+1.00	+.213	+.447*	+.233	-.123	-.058	+.242	+.233	-.042	+.271*	+.149	+.100	+.275*	+.112	+.305*	+.095	+.023	+.213	13
16	+.087	+.095	+.213	+1.00	+.334*	+.259	+.059	-.046	+.190	+.084	-.118	+.339*	+.145	-.118	+.145	-.265	+.279*	-.288*	-.037	+.056	16
17	+.341*	+.447*	+.447*	+.334*	+1.00	+.073	+.118	-.162	+.177	+.212	-.152	+.371*	-.043	+.012	+.103	-.207	+.226	+.059	+.103	-.079	17
18	+.213	-.009	+.233	+.259	+.073	+1.00	+.073	+.175	+.213	+.112	+.004	+.472*	+.264	+.004	+.079	-.155	+.186	-.091	-.106	+.171	18
23	+.341*	+.257	-.123	+.059	+.118	+.073	+1.00	+.473*	-.152	-.067	+.341*	+.200	+.103	+.341*	-.043	-.067	-.133	+.334*	-.043	+.196	23
25	+.232	-.058	-.058	-.046	-.162	+.175	+.473*	+1.00	-.004	+.175	+.351*	+.267	+.120	+.469*	+.120	-.127	-.081	+.251	+.016	+.053	25
26	+.141	+.100	+.242	+.190	+.177	+.213	-.152	-.004	+1.00	+.108	-.227	+.171	+.262	-.105	+.153	-.101	+.204	-.015	+.044	-.015	26
28	+.004	-.009	+.233	+.084	+.212	+.112	-.067	+.175	+.108	+1.00	+.108	+.145	-.014	+.108	+.171	-.243	+.072	-.003	+.171	+.171	28
31	+.018	-.042	-.042	-.118	-.152	+.004	+.341*	+.351*	-.227	+.108	+1.00	+.043	+.262	+.509*	-.064	+.108	-.065	+.498*	+.262	-.015	31
33	+.426*	+.123	+.271*	+.339*	+.371*	+.472*	+.200	+.267	+.171	+.145	+.043	+1.00	+.189	+.043	+.302*	-.073	+.093	+.018	+.076	-.089	33
53	+.044	-.103	+.149	+.145	-.043	+.264	+.103	+.120	+.262	-.014	+.262	+.189	+1.00	+.153	+.036	+.079	+.110	+.145	+.229	-.037	53
59	+.141	-.042	+.100	-.118	+.012	+.004	+.341*	+.469*	-.105	+.108	+.509*	+.043	+.153	+1.00	+.262	+.108	+.070	+.498*	+.153	-.015	59
60	+.153	-.103	+.275*	+.145	+.103	+.079	-.043	+.120	+.153	+.171	-.064	+.302*	+.036	+.262	+1.00	+.079	+.348*	+.054	+.132	+.054	60
64	-.101	+.112	+.112	-.265	-.207	-.155	-.067	-.127	-.101	-.243	+.108	-.073	+.079	+.108	+.079	+1.00	-.042	-.091	+.171	+.171	64
68	+.070	-.006	+.305*	+.279*	+.226	+.186	-.133	-.091	+.204	+.072	-.065	+.093	+.110	+.070	+.348*	-.042	+1.00	+.054	-.128	-.058	68
73	+.293*	-.024	+.095	-.288*	+.059	-.091	+.334*	+.251	-.015	-.003	+.498*	+.018	+.145	+.498*	+.054	-.091	+.054	+1.00	+.145	-.030	73
76	-.064	-.103	+.023	-.037	+.103	-.106	-.043	+.016	+.044	+.171	+.262	+.076	+.229	+.153	+.132	+.171	-.126	+.145	+1.00	-.037	76
80	-.015	+.095	+.213	+.056	-.079	+.171	+.196	+.053	-.015	+.171	-.015	-.089	-.037	-.015	+.054	+.171	-.058	-.030	-.037	+1.00	80

TABLE IV

Correlation Matrix for Diagnostic Group 4

	1	9	10	11	13	14	15	18	21	26	39	40	47	56	57	60	61	66	68	69	
1	+1.00	+.255*	+.196	+.132	+.145	-.025	+.097	-.148	+.075	+.059	+.105	+.190	+.295*	+.360*	+.132	+.441*	+.183	+.115	+.219	-.049	1
9	+.255*	+1.00	+.439*	+.312*	+.287*	+.209	+.460*	+.188	+.117	+.073	+.186	+.399*	+.361*	+.065	+.281*	+.065	+.460*	+.215	+.091	-.159	9
10	+.196	+.439*	+1.00	+.516*	+.478*	+.238	+.248	+.215	+.376*	+.154	+.117	+.286*	+.294*	+.089	+.165	+.165	+.168	+.165	+.346*	+.013	10
11	+.132	+.312*	+.516*	+1.00	+.456*	+.185	+.498*	+.312	+.472*	+.141	+.093	+.255*	+.307*	-.034	+.185	-.126	+.209	+.149	+.261*	-.034	11
13	+.145	+.287*	+.478*	+.456*	+1.00	+.325*	+.210	+.205	+.360*	+.288*	+.028	+.224	+.126	+.059	+.164	+.059	+.122	+.059	+.164	+.059	13
14	-.025	+.209	+.238	+.185	+.325*	+1.00	+.179	+.065	+.300*	+.133	-.022	+.268*	+.319*	-.054	+.155	+.092	+.179	+.238	+.194	-.054	14
15	+.097	+.460*	+.248	+.498*	+.210	+.179	+1.00	+.460*	+.381*	+.343*	-.009	+.234	+.178	-.152	+.179	+.008	+.242	+.168	+.110	-.072	15
18	-.148	+.188	+.215	+.312*	+.205	+.065	+.460*	+1.00	+.195	+.354*	+.186	+.113	+.099	+.065	+.137	+.065	+.224	+.290*	+.243	+.065	18
21	+.075	+.117	+.376*	+.472*	+.360*	+.300*	+.381*	+.195	+1.00	+.222	-.027	+.202	+.339*	+.062	+.149	+.140	-.032	+.219	+.324*	+.219	21
26	+.059	+.073	+.184	+.141	+.288*	+.133	+.343*	+.354*	+.222	+1.00	+.118	+.208	+.007	-.006	+.225	+.184	+.042	+.279*	+.106	+.184	26
39	+.105	+.186	+.117	+.093*	+.028	-.022	-.009	+.186	-.027	+.118	+1.00	+.300*	+.173	+.117	+.319*	+.117	-.009	+.117	+.046	-.237	39
40	+.190	+.399*	+.286*	+.255*	+.224	+.268*	+.234	+.113	+.202	+.208	+.300*	+1.00	+.384*	+.069	+.128	+.141	+.006	+.214	+.244	+.141	40
47	+.295*	+.361*	+.294*	+.307	+.126	+.319*	+.178	+.099	+.339*	+.007	+.173	+.384*	+1.00	+.294*	+.405*	+.383*	+.084	+.560*	+.226	+.029	47
56	+.360*	+.065	+.089	-.034	+.059	-.054	-.152	+.065	+.062	-.006	+.117	+.069	+.294*	+1.00	+.092	+.393*	+.088	+.241	+.037	+.089	56
57	+.132	+.281*	+.165	+.185	+.164	+.155	+.179	+.137	+.149	+.225	+.319*	+.128	+.405*	+.092	+1.00	+.238	+.102	+.238	+.194	-.054	57
60	+.441*	+.065	+.165	-.126	+.059	+.092	+.008	+.065	+.140	+.184	+.117	+.141	+.383*	+.393*	+.238	+1.00	+.168	+.165	+.191	+.241	60
61	+.183	+.460*	+.168	+.209	+.122	+.179	+.242	+.224	-.032	+.042	-.009	+.005	+.084	+.088	+.102	+.168	+1.00	+.168	+.110	-.072	61
66	+.115	+.215	+.165	+.149	+.059	+.238	+.168	+.290*	+.219	+.279*	+.117	+.214	+.560*	+.241	+.238	+.165	+.168	+1.00	+.114	+.013	66
68	+.219	+.091	+.346*	+.261*	+.164	+.194	+.110	+.243	+.324*	+.106	+.046	+.244	+.226	+.037	+.194	+.191	+.110	+.114	+1.00	+.346*	68
69	-.049	-.159	+.013	-.034	+.059	-.054	-.072	+.065	+.219	+.184	-.237	+.141	+.029	+.089	-.054	+.241	-.072	+.013	+.346*	+1.00	69

TABLE V

Correlation Matrix for All Diagnostic Groups

	1	9	10	11	13	14	15	17	18	21	26	28	33	39	47	56	57	60	61	68	
1	+1.00	+.235*	-.001	+.045	+.017	+.063	+.039	-.086*	-.124*	+.127	+.052	+.130	+.065	+.233*	+.410*	+.378*	+.133	+.317*	+.119	-.005	1
9	+.235*	+1.00	+.214*	+.191	+.140	+.291*	+.326*	+.083	+.166	+.001	+.165	+.153	+.097	+.218*	+.257*	+.055	+.089	+.028	+.285*	-.082*	9
10	-.001	+.214*	+1.00	+.375*	+.515*	+.205*	+.246*	+.260*	+.105	+.298*	+.115	+.191	+.276*	-.002	+.100	+.049	+.110	+.221*	+.141	+.173	10
11	+.045	+.191	+.375*	+1.00	+.335*	+.162	+.328*	+.332*	+.083	+.233	+.128	+.121	+.119	+.065	+.116	+.130	+.153	-.035	+.154	+.150	11
13	+.017	+.140	+.515*	+.335*	+1.00	+.285*	+.262*	+.392*	+.079	+.291*	+.119	+.183	+.200	+.022	+.153	+.084	+.079	+.153	+.101	+.139	13
14	+.063	+.291*	+.205*	+.162	+.285*	+1.00	+.270*	+.234*	+.123	+.221*	+.111	+.248*	+.179	+.071	+.222*	+.107	+.077	+.080	+.141	+.107	14
15	+.039	+.326*	+.246*	+.328*	+.262*	+.270*	+1.00	+.447*	+.286*	+.247*	+.303*	+.210	+.222*	+.061	+.189	-.093*	+.061	+.115	+.302*	+.061	15
17	-.086*	+.083	+.260*	+.332*	+.392*	+.234*	+.447*	+1.00	+.248*	+.325*	+.219	+.240*	+.282*	+.054	+.061	-.013	+.114	+.085	+.173	+.234	17
18	-.124*	+.166	+.105	+.083	+.079	+.123	+.286*	+.248*	+1.00	+.076	+.343*	+.125	+.217*	+.157	+.076	-.080	+.064	+.006	+.180	+.225	18
21	+.127	+.001	+.296*	+.233	+.291*	+.221*	+.247*	+.325*	+.076	+1.00	+.180	+.265*	+.247*	+.050	+.223	+.189	+.073	+.223*	+.096	+.402*	21
26	+.052	+.165	+.115	+.128	+.119	+.111	+.303*	+.219	+.343*	+.180	+1.00	+.042	+.185	+.219	+.126	-.004	+.100	+.155	+.190	+.131	26
28	+.130	+.153	+.191	+.121	+.183	+.248*	+.210	+.240*	+.125	+.265*	+.042	+1.00	+.276*	+.240*	+.210	+.123	+.179	+.256*	+.218*	+.222*	28
33	+.065	+.097	+.276*	+.119	+.200	+.179	+.222*	+.282*	+.217*	+.247*	+.185	+.276*	+1.00	-.024	+.030	+.089	+.149	+.198*	+.212	+.226*	33
39	+.233*	+.218*	-.002	+.065	+.022	+.071	+.061	+.054	+.157	+.050	+.219	+.240*	-.024	+1.00	+.206	+.031	+.158	+.133	+.125	+.034	39
47	+.410*	+.257*	+.100	+.116	+.153	+.222*	+.189	+.061	+.076	+.223	+.126	+.210	+.030	+.206	+1.00	+.273*	+.174	+.312*	+.181	+.061	47
56	+.378	+.055	+.049	+.130	+.084	+.107	-.093*	-.013	-.080	+.189	-.004	+.123	+.089	+.031	+.273*	+1.00	+.125	+.318*	+.076	+.054	56
57	+.133	+.089	+.110	+.153	+.079	+.077	+.061	+.114	+.064	+.073	+.100	+.179	+.149	+.158	+.174	+.123	+1.00	+.243*	+.225*	+.165	57
60	+.317*	+.028	+.221*	-.035	+.153	+.080	+.115	+.085	+.006	+.223*	+.155	+.256*	+.198*	+.133	+.312*	+.318*	+.243*	+1.00	+.206*	+.188	60
61	+.119	+.285*	+.141	+.154	+.101	+.141	+.302*	+.173	+.180	+.096	+.190	+.218*	+.212	+.125	+.181	+.076	+.225*	+.206*	+1.00	+.134	61
68	-.005	-.082*	+.173	+.150	+.139	+.107	+.061	+.234	+.225	+.402*	+.131	+.222*	+.226*	+.034	+.061	+.054	+.165	+.188	+.134	+1.00	68

The clustering of these groups suggests that for the various diagnoses, and also for all cases regarded as a single entity, certain specific constellations do have a tendency to reappear; in other words, the appearance of a specific symptom indicates the probable appearance of other symptoms associated with the former.

Diagnostic Significance

A typical experiment usually involves taking one or more measurements on the experimental units under test. The experiment is called univariate, or uniresponse, as opposed to multivariate, or multiresponse, when only one measurement is taken on each unit. Since the symptom study data have many determinations associated with each sample item, i.e., individual, the tools of multivariate analysis are required for the analysis.

TABLE VI

Diagnostic Group Means by Symptom Type

Symptom	Diagnostic Group*			
	1	2	3	4
1	0.7581	0.3519	0.9091	0.5333
9	0.6774	0.4444	0.5758	0.8000
10	0.6935	0.8148	0.6667	0.5333
11	0.8226	0.8704	0.8485	0.6667
13	0.7742	0.8704	0.7576	0.6667
14	0.6452	0.6852	0.6970	0.6667
15	0.7419	0.6852	0.6970	0.7333
17	0.6290	0.9074	0.6061	0.5333
18	0.6774	0.7037	0.5152	0.8667
21	0.7258	0.6111	0.6364	0.4667
26	0.8387	0.8148	0.8182	0.9333
28	0.6129	0.7037	0.6364	0.5333
33	0.6290	0.8333	0.6970	0.4667
39	0.8065	0.6296	0.6970	0.7333
47	0.8065	0.5185	0.8485	0.6667
56	0.6935	0.4630	0.6970	0.4667
57	0.6452	0.6481	0.7273	0.5333
60	0.6935	0.7407	0.8485	0.6000
61	0.7419	0.6667	0.7273	0.6000
68	0.7097	0.8519	0.7576	0.8000

*Diagnostic Group 1: Psychoneurotic Disorders
2: Childhood Schizophrenia
3: Personality Trait Disturbances
4: Psychophysiological Disorders

Table VI shows the mean proportion of individuals in each diagnostic group having the particular symptom under consideration. Differences among the diagnostic groups are to be analyzed on the basis of these 20 symptoms - the 20 most frequently occurring symptoms in the entire sample.

Using the notation of the Appendix, we let x_{ijt} adopt the value one or zero depending on whether the jth individual in the ith diagnostic group does or does not have the tth symptom. For this particular analysis "i" takes the values 1 through 4; "j" takes the values 1 through η_i where $\eta_1 = 62$, $\eta_2 = 54$, $\eta_3 = 33$, and $\eta_4 = 15$; and t can adopt any of the 20 symptoms listed in Table VI.

The symbol $\bar{x}_{i.t}$ refers to the value in each cell of the table, i.e., the sample proportion of individuals in the ith group exhibiting symptom t. For example, $\bar{x}_{3.13}$ has the value 0.7576 indicating that almost 76% of the study group classified as having personality trait disturbances exhibit definite signs of rebelliousness. We let μ_{ij} denote the population mean proportion.

The total hypothesis of no difference among diagnostic groups (see Appendix A) is denoted by

$$H : \underline{\mu}_1 = \underline{\mu}_2 = \underline{\mu}_3 = \underline{\mu}_4 ,$$

where

$$\underline{\mu}'_i = (\mu_{i1}, \mu_{i2}, \ldots, \mu_{i20})$$

and

$$\underline{\mu}_i = \text{transpose of } \mu'_i .$$

The hypothesis of no difference between the ith and ℓth diagnostic groups is denoted by

$$H_{i\ell} : \underline{\mu}_i = \underline{\mu}_\ell \quad (i \neq \ell = 1, 2, 3, 4).$$

The Within Group SP matrix (S_e) and its inverse are given in Tables VII and VIII. The sizes of the samples drawn from the several groups are $\eta_1 = 62$; $\eta_2 = 54$; $\eta_3 = 33$; $\eta_4 = 15$. The number of variates (i.e., symptoms) is $p = 20$ and η, the total sample size is 164. The "error degrees of freedom," ν, $= \eta - p - K + 1 = 164 - 20 - 4 + 1 = 141$. Then

$$T_{i\ell}^2 = \nu\eta_{i\ell} (\hat{\underline{\mu}}'_i - \hat{\underline{\mu}}'_\ell) S_e^{-1} (\hat{\underline{\mu}}_i - \hat{\underline{\mu}}_\ell),$$

where

$$\eta_{i\ell} = \frac{\eta_i \eta_\ell}{(\eta_i + \eta_\ell)}$$

and

$$\hat{\underline{\mu}}'_i = (\bar{x}_{i.1}, \ldots, \bar{x}_{i.20}), \; i = 1, 2, 3, 4.$$

TABLE VII

Within Groups Sums of Squares and Cross Products (SP) Matrix

	1	9	10	11	13	14	15	17	18	21	26	28	33	39	47	56	57	60	61	68	
1	+30.146	+7.0441	+0.6551	+2.0138	+2.0153	+1.4165	+1.3348	-0.2537	-2.5970	+5.4517	+2.0870	+5.4656	+4.9597	+8.3581	+8.4570	+10.965	+3.2778	+10.075	+4.8442	+3.3327	1
9	+7.0441	+37.342	+10.249	+7.4415	+5.2010	+11.216	+12.352	+3.8877	+6.4716	-0.8414	+4.4732	+7.8783	+6.7382	+6.9755	+6.5634	+1.9174	+6.1295	+1.7720	+10.821	-2.2448	9
10	+0.6551	+10.249	+32.392	+10.383	+15.413	+7.4432	+6.7486	+7.4257	+2.6413	+11.168	+3.6170	+5.4155	+9.2183	-0.5811	+3.5078	+1.7404	+3.4729	+5.1182	+3.9634	+5.9357	10
11	+2.0138	+7.4415	+10.333	+22.717	+9.7299	+4.7113	+10.109	+9.9682	+2.2866	+6.7768	+2.6871	+3.5163	+2.5709	+2.4299	+4.0764	+3.6880	+3.9368	-1.9434	+4.4643	+3.5573	11
13	+2.0153	+5.2010	+15.413	+9.7299	+26.325	+8.7376	+8.4258	+11.673	+1.8643	+10.863	+3.6578	+5.2641	+6.5489	-2.0598	+5.0412	+3.8595	+2.0541	+3.6827	+2.8719	+3.9590	13
14	+1.4165	+11.216	+7.4432	+4.7113	+8.7376	+36.145	+8.6071	+7.9919	+4.3510	+9.0536	+5.1519	+9.4771	+7.3084	+2.0820	+7.3749	+3.4315	+3.1515	+2.3355	+5.9286	+4.6701	14
15	+1.3348	+12.352	+6.7486	+10.109	+8.4258	+8.6071	+33.422	+14.684	+9.4199	+8.2321	+8.1864	+6.2664	+8.0675	+1.5100	+6.8696	-4.1965	+1.7472	+1.5742	+9.8770	+1.6121	15
17	-0.2537	+3.8877	+7.4257	+9.9682	+11.673	+7.9919	+14.684	+30.617	+6.8628	+13.288	+4.5341	+7.6214	+8.9617	+0.8905	+3.8379	+1.5937	+2.2673	+1.8856	+5.0524	+7.0303	17
18	-2.5970	+6.4716	+2.6413	+2.2866	+1.8643	+4.3510	+9.4199	+6.8628	+34.783	+2.4091	+8.7688	+3.7658	+8.9988	+4.8213	+4.3344	+0.3632	+2.9766	+2.4986	+7.3417	+5.5444	18
21	+5.4517	-0.8414	+11.168	+6.7768	+10.863	+9.0536	+8.2321	+13.288	+2.4091	+36.542	+6.6540	+10.100	+10.291	+3.1622	+7.1137	+6.6095	+1.5728	+9.3277	+2.1402	+14.444	21
26	+2.0870	+4.4732	+3.6170	+2.6871	+3.6578	+5.1519	+8.1864	+4.5341	+8.7688	+6.6540	+22.378	+1.5176	+5.2721	+5.2760	+4.0073	+1.2136	+2.8301	+4.0338	+6.0497	+2.9607	26
28	+5.4656	+7.8783	+5.4155	+3.5163	+5.2641	+9.4771	+6.2664	+7.6214	+3.7658	+10.100	+1.5176	+37.339	+10.060	+7.9259	+8.4996	+4.6829	+5.3148	+7.8788	+8.4004	+6.3528	28
33	+4.9597	+6.7382	+9.2183	+2.5709	+6.5489	+7.3084	+8.0675	+8.9617	+8.9988	+10.291	+5.2721	+10.060	+32.671	+3.0514	+4.0332	+3.8213	+7.2114	+6.9031	+8.1372	+4.9650	33
39	+8.3581	+6.9755	-0.5811	+2.4299	-2.0598	+2.0820	+1.5100	+0.8905	+4.8213	+3.1622	+5.2760	+7.9259	+3.0514	+32.173	+6.1993	+2.4182	+5.1110	+3.0222	+4.9093	+1.3289	39
47	+8.4570	+6.5634	+3.5078	+4.0764	+5.0412	+7.3749	+6.8696	+3.8379	+4.3344	+7.1137	+4.0073	+8.4996	+4.0332	+6.1993	+30.735	+6.1778	+4.8968	+9.8243	+5.8729	+5.4522	47
56	+10.965	+1.9174	+1.7404	+3.6880	+3.8595	+3.4315	-4.1965	+1.5937	+0.3632	+6.6095	+1.2136	+4.6829	+3.8213	+2.4182	+6.1778	+37.306	+3.5938	+10.944	+1.5028	+4.1633	56
57	+3.2778	+6.1295	+3.4729	+3.9368	+2.0541	+3.1515	+1.7472	+2.2673	+2.9766	+1.5728	+2.8301	+5.3148	+7.2114	+5.1110	+4.8968	+3.5938	+36.787	+8.1685	+8.7347	+4.2163	57
60	+10.075	+1.7720	+5.1182	-1.9434	+3.6827	+2.3355	+1.5742	+1.8856	+2.4986	+9.3277	+4.0338	+7.8788	+6.9081	+3.0222	+9.8243	+10.944	+8.1685	+31.390	+7.6665	+7.9977	60
61	+4.8442	+10.821	+3.9634	+4.4643	+2.8719	+5.9286	+9.8770	+5.0524	+7.3417	+2.1402	+6.0497	+8.4004	+8.1372	+4.9093	+5.8729	+1.5028	+8.7347	+7.6665	+34.016	+3.3064	61
68	+3.3327	-2.2448	+5.9357	+3.5573	+3.9590	+4.6701	+1.6121	+7.0303	+5.5444	+14.444	+2.9607	+6.3528	+4.9650	+1.3289	+5.4522	+4.1633	+4.2163	+7.9977	+3.3064	+28.050	68

TABLE VIII*

Inverse of Within Groups Sums of Squares and Cross Products (SP) Matrix

	1	9	10	11	13	14	15	17	18	21	26	28	33	39	47	56	57	60	61	68	
1	+4.7895	-.86387	+.65889	-.19965	-.29130	+.37448	-.08430	+.45582	+.87013	-.27882	-.01717	+.07960	-.54182	-.88753	-.58243	-.89158	+.22999	-.87236	-.16321	-.31069	1
9	-.36387	+4.2844	-1.2788	-.33975	+.29152	-.33422	-1.0461	+.25134	-.35483	+.36807	+.03580	-.32662	-.01724	-.43655	-.13843	-.12170	-.27392	+.30606	-.51847	+.63943	9
10	+.65889	-1.2788	+5.3770	-1.4410	-2.2807	+.06407	+.45554	+.53559	+.14944	-.71938	-.05008	+.08471	-.82438	+.21025	+.23603	+.41460	+.11938	-.71326	+.12864	-.45911	10
11	-.19965	-.33975	-1.4410	+7.1461	-1.0302	+.31518	-1.3250	-1.0364	+.20638	-.19892	+.18313	+.12727	+.75995	-.43834	-.27961	-.93619	-.62546	+1.6151	-.35149	-.40905	11
13	-.29130	+.29152	-2.2807	-1.0302	+6.8221	-.62666	-.15958	-1.3232	+.13984	-.42110	-.24723	-.09088	+.00066	+.69187	-.35779	-.22772	+.02469	-.13696	+.15264	+.50533	13
14	+.37448	-.88422	+.06407	+.31548	-.62866	+3.6094	-.13688	-.15679	+.09965	-.42777	-.33215	-.41733	-.17096	+.10552	-.43472	-.22695	-.01507	+.36342	-.10962	-.24900	14
15	-.08430	-1.0461	+.45654	-1.3250	-.15933	-.13688	+5.3675	-1.5236	-.56668	-.50976	-.87040	-.04387	-.25906	+.50808	-.52870	+1.0774	+.33740	-.21028	-.57503	+.63519	15
17	+.45582	+.25134	+.53359	-1.0364	-1.3232	-.13679	-1.5236	+5.4720	-.32928	-.89612	+.20521	-.29967	-.55140	-.04372	+.26372	-.10442	+.01144	+.15340	-.02996	-.04912	17
18	+.87013	-.35453	+.14944	+.20538	+.13984	+.09965	-.56668	-.32923	+3.8491	+.64048	-1.0109	+.08759	-.78134	-.42223	-.24533	-.20975	+.12182	-.01175	-.22791	-.81407	18
21	-.27892	+.86807	-.71938	-.19692	-.42110	-.42777	-.50976	-.89812	+.54048	+4.9229	-.74180	-.43416	-.68443	-.31915	-.10817	-.27262	+.35766	-.51180	+.47157	-1.5637	21
26	-.01717	+.03680	-.05008	+.18313	-.24728	-.33215	-.87040	+.20521	-1.0109	-.74180	+5.7803	+.61250	-.08773	-.73685	-.00683	-.03233	-.09158	-.28369	-.43446	+.11646	26
28	+.07960	-.32662	+.08471	+.12727	-.09098	-.41733	-.04387	-.29967	+.08759	-.43416	+.61250	+3.5602	-.53728	-.64626	-.35438	-.06220	-.02278	-.32820	-.38625	-.22586	28
33	-.54182	-.01724	-.82438	+.75995	+.00066	-.17096	-.25906	-.55140	-.78134	-.58448	-.08773	-.53728	+4.2980	+.11190	+.27977	-.12855	-.55900	-.03859	-.34554	+.24006	33
39	-.88753	-.43655	+.21026	-.43834	+.69187	+.10552	+.50808	-.04372	-.42223	-.31915	-.73685	-.64526	+.11190	+3.9297	-.36335	+.20168	-.28987	+.18533	-.04978	+.25118	39
47	-.58243	-.13543	+.23603	-.27961	-.36780	-.43472	-.52870	+.26372	-.24533	-.10617	-.00683	-.35438	+.27977	-.36335	+4.2623	-.15616	-.11542	-.84641	+.00732	-.26424	47
56	-.89158	-.12170	+.41460	-.93619	-.22772	-.22695	+1.0774	-.10442	-.20975	-.27262	-.03233	-.06220	-.12655	+.20168	-.15516	+3.5394	+.02394	-.97478	+.16777	+.13447	56
57	+.22999	-.27392	+.11938	-.62346	+.02469	-.01507	+.33740	+.01144	+.12182	+.35765	-.09158	-.02278	-.53699	-.28987	-.11542	+.02394	+3.2273	-.70979	-.45211	-.31327	57
60	-.87236	+.30606	-.71326	+1.6151	-.13696	+.36542	-.21028	+.15340	-.01175	-.51180	-.28369	-.32320	-.03659	+.18533	-.84541	-.97478	-.70979	+5.0131	-.67093	-.55908	60
61	-.16321	-.51847	+.12864	-.35149	+.15264	-.10952	-.57503	-.02996	-.22791	+.47157	-.43446	-.38625	-.34664	-.04978	+.00732	+.15777	-.45211	-.67093	+3.8961	-.19339	61
68	-.31069	+.63948	-.45911	-.40906	+.60038	-.24900	+.63519	-.49125	-.81407	-1.5637	+.11646	-.22686	+.24005	+.25116	-.26424	+.13447	-.31327	-.55968	-.19359	+5.0808	68

* Each element of the matrix is to be multiplied by 10^{-2}

The data on symptoms yield the following values for the T^2's:

$$T^2_{12} = 77.3215,$$

$$T^2_{13} = 15.8515,$$

$$T^2_{14} = 18.7329,$$

$$T^2_{23} = 53.7181,$$

$$T^2_{24} = 36.4148,$$

$$T^2_{34} = 27.0456.$$

It is apparent that $T^2_{max} = T^2_{12} = 77.3215$.

If we are interested in testing the hypothesis at the 6% level, the critical value, T^2_{α} is found in the following way.

$$\text{Probability}\left[\text{largest } T^2_{i\ell} \leq T^2_{\alpha} \middle| H\right] \cong 1 - \sum_{i \neq \ell = 1}^{4} P_{i\ell} = 0.94,$$

where

$$P_{i\ell} = \text{Probability}\left[T^2_{i\ell} > T^2_{\alpha} \middle| H_{i\ell}\right].$$

Since all $P_{i\ell}$'s are equal to (say) P, we have

$$1 - 6P = 0.94$$

and therefore P = 0.01.

Now each $T^2_{i\ell}$ is distributed as the F distribution with (p,ν), that is (20,141) degrees of freedom. The F tables yield a value of $T^2_{\alpha} = 1.88$ when P = 0.01.

Hence $T^2_{max} > T^2_{\alpha}$ and the hypothesis of no difference among diagnostic groups is rejected. In fact, each $T^2_{i\ell} > T^2_{\alpha}$ which indicates that each differs significantly from all others.

Although the application of this particular test is justified only when the data are multivariate normal it seems reasonable to assume that (1) the data for each diagnostic group do approximate normality since the sample sizes are moderately large; and (2) the test itself is not particularly sensitive to deviations from normality.

The results, while preliminary, are extremely encouraging in that they suggest the existence of reasonably clear-cut criteria for diagnostic grouping. Obviously much work remains to be done, but the pilot investigation to date gives every indication that future study will provide revealing information and meaningful results for differential diagnosis of emotional disturbances.

APPENDIX

Tests for Equality of Mean Vectors

Consider K multivariate normal populations with a common unknown covariance matrix Σ and mean vectors $\mu_1, \ldots, \mu_K$, where $\mu_i' = (\mu_{i1}, \ldots, \mu_{ip})$ and μ_{it} denotes the ith population mean on the tth variate. Since a multivariate normal is completely specified by its mean vector and covariance matrix, the above populations are homogeneous if their mean vectors are equal. The hypothesis of equality of mean vectors is denoted by

$$H: \mu_1 = \ldots = \mu_K.$$

Various test procedures are available in the literature to test the total hypothesis H and to make a decision relative to the acceptance or rejection of its subhypotheses in the event H is rejected. For a detailed discussion on the relative merits of these procedures, the reader is referred to [5]. Two of these procedures are briefly reviewed here. The first procedure is known as the "Multivariate Analysis of Variance (MANOVA) test based on the largest root." This test was proposed by S. N. Roy [7] and the test procedure is to accept of reject H according as

$$C_L(S_H S_e^{-1}) \lessgtr \lambda_{\alpha},$$

where $C_L(A)$ denotes the maximum characteristic root of A and λ_{α} is chosen such that

$$\text{Prob.}\left[C_L(S_H S_e^{-1}) \leq \lambda_{\alpha} | H\right] = (1 - \alpha).$$

In the above equation, S_H and S_e respectively denote the sums of squares and cross products (SP) matrices due to H and "error" respectively. (In the univariate case, where p = 1, S_H and S_e respectively denote the sums of squares due to H and error respectively.

The value of λ_{α} for any given value of α can be obtained from the tables of D. L. Heck [1]. If the total hypothesis is rejected, we can make multiple decisions on the acceptance or rejection of the various subhypotheses

by examining the confidence intervals on the "parametric functions" which measure departures from these subhypotheses. These confidence intervals were derived by S. N. Roy and R. Gnanadesikan [8,9]. For an illustration of the MANOVA test with biochemical data, the reader is referred to [13].

The second method is based on the maximum of the $\binom{K}{2}$ Hotelling T^2's. This procedure is applicable to test H and make multiple decisions on the acceptance or rejection of its subhypotheses of the form $H_{i\ell} : \mu_i = \mu_\ell$, $(i \neq \ell = 1, 2, \ldots, K)$. This procedure was formulated by S. N. Roy and R. C. Bose [6] and is a multivariate analogue of Tukey's multiple comparison test [14]. The technique (with trivial modification) is described below:

Let

$$\eta_{i\ell} = \frac{\eta_i \eta_\ell}{(\eta_i + \eta_\ell)}, \quad \gamma = \sum_{i=1}^{K} \eta_i - p - K + 1$$

$$T^2_{i\ell} = \eta_{i\ell} \nu (\hat{\mu}^1_i - \hat{\mu}^1_\ell) S_e^{-1} (\hat{\mu}_i - \hat{\mu}_\ell),$$

$$(i \neq \ell = 1, 2, \ldots, K),$$

where ν is the error degrees of freedom, η_i is the size of ith sample and $\hat{\mu}_i$, $(i = 1, 2, \ldots, K)$ is the maximum likelihood estimate of μ_i. Then we accept or reject H according as

$$\text{largest } T^2_{i\ell} \text{ out of } \binom{K}{2} \text{ pairs } \lessgtr T^2_\alpha,$$

where T^2_α is chosen such that

$$\text{Prob.}\left[\text{largest } T^2_{i\ell} \text{ out of } \binom{K}{2} \text{ pairs} \leq T^2_\alpha \middle| H\right] = (1 - \alpha).$$

If the total hypothesis H is rejected, we accept or reject the subhypotheses $H_{i\ell}$ according as

$$T^2_{i\ell} \lessgtr T^2_\alpha.$$

Recently, it was shown [5] that the above test is better (in the sense of shortness of the lengths of the confidence intervals) than the MANOVA test but the nature of the exact distribution of the "largest Hotelling T^2" is not known. M. Siotani [10-12] has suggested some approximations to this distribution which for moderately large samples, seem satisfactory. When the error degrees of freedom are very large, the following approximation can be used.

$$\text{Prob.}\left[\text{largest } T^2_{i\ell} \leq T^2_\alpha | H\right] \cong 1 - \sum_{i \neq \ell = 1}^{K} P_{i\ell},$$

where

$$P_{i\ell} = \text{Prob.}\left[T^2_{i\ell} > T^2_\alpha | H_{i\ell}\right].$$

But, when $H_{i\ell}$ is true, $T^2_{i\ell}$ is distributed as the F distribution with $(p, \eta - p - K + 1)$ degrees of freedom where

$$\eta = \sum_{i=1}^{K} \eta_i .$$

So, the values of $P_{i\ell}$'s can be obtained from F tables (or incomplete Beta function tables) for any given value of T^2_α. Here, we note that M. Siotani suggested an approximation similar to the above as a first approximation for upper percentage points.

In one way classification, $S_H = (s_{tu})$ and $S_e = (s_{etu})$ are respectively called "Between Groups" and "Within Groups" SP matrices where

$$s_{tu} = \sum_i \eta_i (\bar{x}_{i.t} - \bar{x}_{..t})(\bar{x}_{i.u} - \bar{x}_{..u})$$

$$s_{etu} = \sum_i \sum_j (x_{ijt} - \bar{x}_{i.t})(x_{iju} - \bar{x}_{i.u})$$

and

$$\bar{x}_{i.t} = \left(\sum_j x_{ijt}\right) / \eta_i = \hat{\mu}_{it}$$

$$\bar{x}_{..t} = \left(\sum_i \sum_j x_{ijt}\right) / \eta .$$

Here x_{ijt} denotes the observed value associated with the jth individual in the ith group on the tth variate. $\bar{x}_{i.u}$, $\bar{x}_{..u}$ can be defined similarly.

TABLE A-1

List of Symptoms

DESCRIPTION	ORIGINAL NUMBER	REASSIGNED NUMBER
School Maladjustment:		
Arithmetic disability	1	
Reading disability	2	1
Unsatisfactory school work in general	3	
Truancy from school	4	Delete
Frequent absences from school	5	2
Fear or extreme dislike of school	6	3
Asocial Behavior:		
Destructiveness	7	4
Lying	8	5
Stealing	9	6
Cheating	10	7
Pre-occupation with matches, fire-setting	11	8
Running away from home	12	Delete
Negative Attitudes and Behavior:		
Sullenness, Sulkiness	13	9
Disobedience	14	10
Stubbornness	15	11
Negativism	16	12
Rebelliousness	17	13
Resentment	18	14
Easily aroused anger	19	15
Unprovoked anger	20	16
Temper tantrums	21	17
Excessive whining or crying	22	18
Jealousy, envy of others	23	19
Bearing grudges	24	20
Teasing behavior	25	21
Attacking behavior	26	
Fighting	27	22
Bullying	28	
Cruelty	29	Delete
Other Interferences in Social Relationships:		
Seclusiveness	30	23
Shyness	31	24
Excessive passivity, non-assertiveness	32	25
Over-sensitivity	33	26
Over-conformity	34	27
Over-dependency	35	28
Excessive independency	36	29
Clinging exaggerated display of affection	37	30
General indifference or disinterest in people	38	31

TABLE A-1 (Continued)

DESCRIPTION	ORIGINAL NUMBER	REASSIGNED NUMBER
Other Interferences in Social Relationships (Continued)		
Inability to form strong attachments or relationships	39	32
Inability to get along with other children his own age	40	33
Unpopularity with children	41	
Inability to get along with adults	42	97
Preference for younger children	43	34
Preference for older children	44	35
Suspiciousness, distrust of people	45	36
Excessive assertiveness	46	37
Excessive competitiveness	47	38
"Sissy-like" behavior (boys)	48	Delete
"Tom-boyish" behavior (girls)	48	Delete
Attitudes Toward The Self:		
Lack of self-confidence	49	39
Self-depreciation	50	40
Feelings of bodily inadequacy or defects	51	41
Attempts at physical self-injury	52	Delete
Disregard for his own possessions	53	42
Carelessness	54	43
Messiness	55	44
Worrying	56	45
Emphasis on sameness	57	46
Dawdling, procrastination	58	47
Over-conscientiousness, need for perfection	59	48
Specific fears, phobias	60	49
General fearfulness,timidity	61	50
Unrealistic fearlessness	62	51
General feeling of dissatisfaction, lack of enthusiasm	63	52
Specific compulsions	64	53
Bragging, grandiosity	65	54
Accident proneness	66	55
Exaggerated unconcern	67	56
Interference with Thought Activity:		
Day-dreaming	68	57
Verbalized fantasies	69	58
Delusions, hallucinations	70	
Complete self-absorption	71	59
Inability to concentrate	72	60
Boredom, lack of interest	73	61
Specific obsessions	74	62
Memory disturbances - underdeveloped	75	63
Memory disturbances - overdeveloped	76	64
Motor Disturbances:		
Tiredness	77	65

TABLE A-1 (Continued)

DESCRIPTION	ORIGINAL NUMBER	REASSIGNED NUMBER
Motor Disturbances (Continued)		
Laziness	78	66
Poor coordination, awkwardness	79	67
Restlessness	80	68
Hyperactivity	81	69
Stereotyped mannerisms	82	70
Tics	83	Delete
Facial grimaces	84	71
Head-banging	85	Delete
Disruption in Developmental Pattern:		
Speech disorders		
Doesn't talk at all	86	72
Slow in learning to speak	87	73
Doesn't speak clearly	88	74
Limited vocabulary	89	75
Repetitive speech	90	76
Use of 3rd person	91	77
Other speech disorders	92	Delete
Eating problems		
Won't eat at all	93	78
Doesn't eat enough, "poor eater"	94	79
Likes only a limited number of foods	95	80
Won't try new foods	96	81
Eats only strained foods	97	Delete
Food fads	98	82
Allergies to certain foods	99	Delete
Excessive appetite	100	Delete
Eats with fingers, poor table manners	101	Delete
Other eating problems	102	Delete
Wetting		
Day	103	83
Night	104	
Soiling		
Day	105	84
Night	106	
Excessive masturbation	107	Delete
Sex Play		
Exhibitionism	108	85
Voyeurism	109	

TABLE A-1 (Continued)

DESCRIPTION	ORIGINAL NUMBER	REASSIGNED NUMBER
Disruption in Developmental Pattern (Continued)		
Sex Play (Continued)		
Sexual activities with others	110	Delete
Sexual attitude	111	Delete
Sexual approach	112	Delete
Fearful of sex	113	Delete
Other sexual problems	114	Delete
Thumb-sucking	115	86
Nail-biting	116	87
Mouthing, licking, sucking, biting, objects	117	88
Fetishism	118	89
Sleep disturbances		
Refusal to go to bed	119	90
Fear of the dark	120	91
Difficulty falling asleep	121	92
Nightmares	122	93
Talking or calling out in sleep, crying in sleep	123	94
Sleep-walking	124	Delete
Light sleeper, waking up frequently at night	125	Delete
Prowling at night through the house	126	Delete
Wanting to sleep with one or both parents	127	95
Excessive sleep	128	96
Other sleep disturbances	129	Delete
Other problems not mentioned above	130	Delete

BIBLIOGRAPHY

1. Heck, D. L., "Charts of Some Upper Percentage Points of the Distribution of the Largest Characteristic Root," Annals of Mathematical Statistics, Vol. 31 (1960), pp. 625-642.
2. Johnson, P. O., Statistical Methods in Research, Prentice Hall, Inc., New York (1950).
3. Kaskey, Gilbert and Krishnaiah, P. R., "Statistical Routines for Univac Computers," Transactions of Middle Atlantic Conference of American Society for Quality Control (1962), pp. 289-304.
4. Krishnaiah, P. R., Simultaneous Tests and the Efficiency of Generalized Balanced Incomplete Block Designs (Unpublished Manuscript).
5. _____. "Multiple Comparison Tests in Multi-Response Experiments." Presented at the Annual Meeting of the Institute of Mathematical Statistics, September 1962.
6. Roy, S. N., and Bose, R. C., "Simultaneous Confidence Interval Estimation," Annals of Mathematical Statistics, Vol. 24 (1953), pp. 513-536.
7. Roy, S. N., Some Aspects of Multivariate Analysis, John Wiley and Sons, Inc., New York, 1957.
8. Roy, S. N., and Gnanadesikan, R., "Further Contributions to Multivariate Confidence Bounds," Biometrika, Vol. 44 (1957), pp. 399-410.
9. _____. "A Note on Further Contributions to Multivariate Confidence Bounds," Biometrika, Vol. 45 (1958). p. 581.
10. Siotani, M., "The Extreme Value of the Generalized Distances of the Individual Points in the Multivariate Normal

Sample," Annals of the Institute of Statistical Mathematics, Vol. 10 (1959), pp. 183-203.

11. _____. "On the Range in Multivariate Case," Proceedings of the Institute of Statistical Mathematics, Vol. 6 (1959), pp. 155-165.
12. _____. "Notes on Multivariate Confidence Bounds," Annals of the Institute of Statistical Mathematics, Vol. 11 (1960), pp. 167-182.
13. Smith, H., Gnanadesikan, R., and Hughes, J. B., "Multivariate Analysis of Variance (MANOVA)," Biometrics, Vol. 18 (1962), pp. 22-41.
14. Tukey, J. W., The Problems of Multiple Comparisons (Unpublished Notes), Princeton University.

SPACETRACKING MAN-MADE SATELLITES AND DEBRIS

Robert W. Waltz, Colonel, USAF
Commander 9th Aerospace Def. Div.

and

B. M. Jackson, Captain, USAF
Ent Air Force Base, Colorado

There are many types of satellites in orbit. One revolution requires a minimum of about ninety minutes. This varies, of course, and can be several times this quantity. The only organization in the free world that has responsibility for tracking all man-made objects in space is the 1st Aerospace Control Squadron.

AEROSPACE EVALUATION

The United States got into the aerospace business in 1957 when Russia launched its first Sputnik. At that time there was no operational skill developed in the art of detecting or tracking satellites. The Research and Development Command of the Air Force was assigned the responsibility to develop such a capability. This organization wa established at Hanscom Field. In addition to research and development responsibility, they were tasked with the operational requirements of those days. In November 1960, the Air Force began its training of a group of Air Force personnel to establish a completely operational organization. In February of 1961, Headquarters USAF directed that the 1st Aerospace Control Squadron be in position by 1 July at Ent Air Force Base with an operational capability of detecting and tracking satellites in support of NORAD. At that time we had neither trained personnel nor a physical location to go into, communications facilities, nor a computer. In the next four months our organization increased from the original twelve people to something in the order of over 100. A building was identified. Generals were moved from their offices and the facilities were completely rehabilitated. A computer was designated and installed by the middle of April of 1961. Personnel began arriving on site about that time, and by the 10th of June all of the facilities were ready. Unexpectedly, the computer at our research and development facility had a failure due to air conditioning. The requirement was placed upon our organization to commence operations 20 days ahead of schedule. We accomplished this without much difficulty.

Since that time we have taken on the name of Spacetrack Center. This is the portion of 1st Aerospace Squadron that has the satellite mission. We were established as a contingent of the NORAD Combat Operations Center. The entire space detection and tracking system, which is the responsibility of NORAD, has been dubbed SPADATS.

SATELLITE BACKUP

In order to provide reliability in the system and to provide our research organization with the equipment and tools to further develop the state of the art of satellite detection and tracking, a backup facility was established at Hanscom Field. Equipment identical to that installed at Ent has been provided. The backup installation is called the Spacetrack R&D Facility and is under the control of the Air Force Systems Command. As do we, they have other names, the prime one being 496L Systems Project Office (SPO).

RELATIONSHIPS WITH OTHER ORGANIZATIONS

The 1st Aerospace Control Squadron was established to support the North American Air Defense (NORAD). The United States Air

Force, through its Air Defense Command, is responsible for the support and technical operation of all the facilities of the NORAD Combat Operations Center (COC) located at Ent Air Force Base. The 1st Aerospace Control Squadron is responsible to the Air Defense Command (through the 9th Aerospace Defense Division). NORAD is a unified command composed of all the various military services of the United States and Canada.

SPACETRACK MISSION

The mission of the Spacetrack Center is to detect and track all man-made space objects; maintain an information catalogue on all space objects; determine orbits and ephemerides of all space objects; provide system status and satellite displays; and provide satellite data to NORAD and other military and scientific agencies as required.

SATELLITE SENSORS

To support the mission of Spacetrack, there are sensors located all around the world. There are three basic types of sensors that are used to detect and track satellites. These are optical, radar and radiometric. Some of these are controlled by military organizations; others are scientific instruments which support Spacetrack on a cooperative basis.

An example of the optical type sensor is the Baker-Nunn camera. A Baker-Nunn camera is very similar to a telescope and is equipped for taking pictures of satellites against a star background. The mount for the Baker-Nunn camera is steerable and can be programmed to track the path of the satellite in accordance with predictions provided by the Spacetrack Center. Techniques of the astronomer are used to determine the position of the satellite with respect to reference stars. In this manner, we obtain the right ascension and declination of the satellite's position.

Radars provide our most useful information in that they determine all of the quantities required to fix the satellite's position as one point in space. From the radar we get azimuth, elevation, range, range rate, doppler, and what we call a signature of the satellite. By signature we imply its tumble rate and radar cross section from which we can, in general, determine the size of the object, length, width and general configuration. Radars are of two basic types. One is the fixed fan which has a stationary antenna. It radiates electrical energy in a horizontal or vertical fan-shaped plane. The other type radar is the tracker. It provides a pencil beam pattern and must be pointed at the satellite. The antenna is parabolic and is steerable either by manual control or automatically by computer program. See Fig. 1.

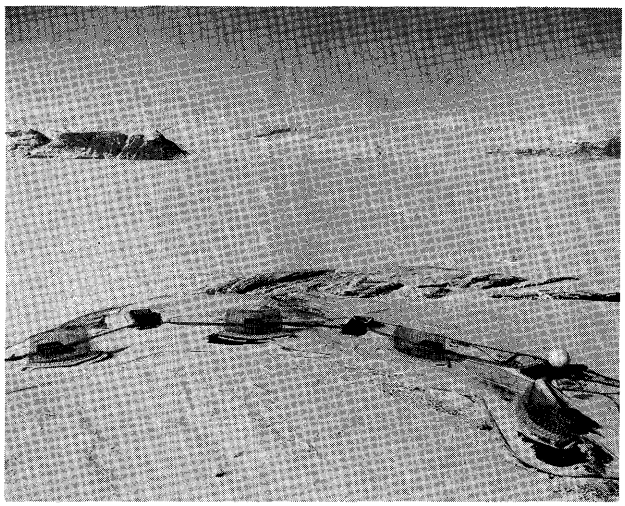


Figure 1. Aerial View of Thule—This areial view of the BMEWS, Thule, Greenland radar site shows the four huge detection radar antennas and the one radome which protects the movable tracking radar antenna. The stationary detection antennas measure 165 feet in height and 400 feet across the top.

A third and last major sensor that contributes significantly to our mission is the radiometric type. It is a passive type sensor that depends upon transmissions from the satellite itself. It is, in effect, a simple but highly directional radio receiver. With this device we can determine the azimuth of the satellite, the time of its closest approach to the facility, and the doppler.

COMMUNICATIONS AND SENSOR LOCATIONS

Teletype or high-speed data link communications connect Spacetrack to all of the major sensors or centers that collect satellite observations. Direct circuits are provided to:

a. The Ballistic Missile Early Warning fixed fan radars at Clear, Alaska, and Thule, Greenland (Thule also has a tracking radar).

b. Shemya Island (Alaska)—both a fixed fan and tracking radar.
c. Laredo AFB, Texas—tracking radar.
d. Moorestown, New Jersey—tracking radar.
e. Spacetrack R&D Facility, Mass,—for computer backup. Data from a radiometric sensor and a tracking radar are received from this area.
f. SPASUR operated by the U. S. Navy—consists of a series of vertical radar fans across southern United States.
g. Patrick AFB, Florida—provides data from the Atlantic Missile Range sensors including all types.
h. Sunnyvale and Point Mugu, Calif.—for data from the Pacific Missile Range.
i. National Aeronautics & Space Administration (NASA) at Greenbelt, Maryland—links Spacetrack with the scientific sensors around the world including those of foreign governments.

SATELLITE DATA INPUTS

Information received from the sensors by 100 wpm teletype circuits is processed to the computer in two modes. In the manual mode a paper tape and hard copy of each message received by the Communications Center is delivered to the Data Conversion Room. If the message is in the proper format, it can be immediately converted from paper tape to punched cards by using an IBM 047 tape-to-card converter. On the other hand, for certain few messages, the observations must be processed manually from the hard copy by entering the information in a converted form onto a standard observation sheet. This sheet, in turn, is hand-punched to provide cards. The cards are then consolidated and read on magnetic tape in an off-line mode.

The second mode is called a semi-automatic mode of operation. Due to equipment and program difficulties, this system has never been used operationally. However, the hardware is designed to allow satellite observations received by teletype to be fed directly to the computer through electrical circuits. The system input starts with the sensor looking at the satellite and determining its position in azimuth, elevation, range, range rate, and any other quantities it can obtain. This information is transmitted over 100 wpm teletype circuits and received in the Communications Center on a page printer which provides a hard copy. If the format is coded beginning with five $ signs, switching action routes the message to the computer. The switching unit is activated by the first three $ signs and connects the incoming circuits to a tape punch. The remainder of the message, which includes two $ signs, the text and the closing signal, is punched on paper tape and stored until ready to be transmitted to the computer electrically. To end the message, a signal of five right-hand parentheses are provided. Three of these activate the switching unit to return it to its normal position of a local tape punch in the Comm Center. Messages not in this coded format are received only as a hard copy and local paper tape in the Comm Center. As mentioned in the manual mode, these two items are delivered to Data Conversion for processing by hand. Upon command from the computer, messages stored in the Comm Center's semi-automatic input equipment can be transmitted through the DMNI and Real-Time System to the Philco 2000. Twenty-four low-speed teletype and three high-speed data link circuits can be connected to the DMNI at present. To provide for storage of data during periods when the 2000 is inoperative, a DMNI recorder will be installed. This will consist of a 410 processor and a magnetic tape unit.

SATELLITE PROGRAM (A-1)

The A-1 program system (see Figure 2) provides for the processing of nearly all of the routine satellite data. It contains its own executive and is completely independent of

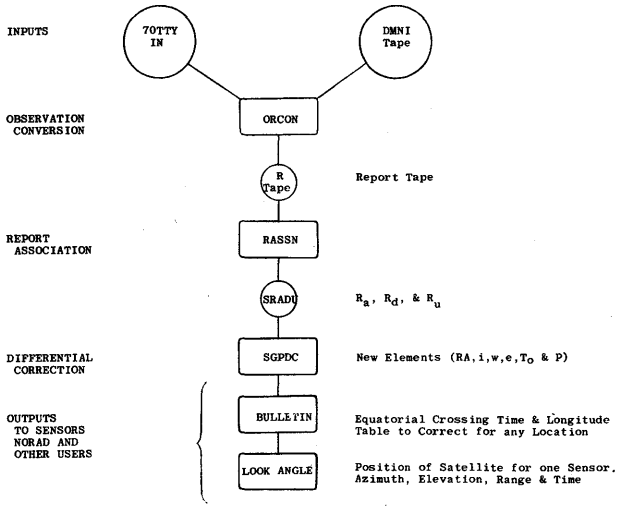


Figure 2. Satellite Computer Program (A-1).

the Philco SYS. The inputs to this system are from magnetic tape and consist of (1) the 70TTY IN tape which contains the manual data provided from punched card and (2) the DMNI tape created through the semi-automatic mode of operation.

These two tapes are the input to the observation conversion (ORCON) routine, where the observations are converted on a report (R) tape to one common standard computer format. The R tape becomes an input to the report association (RASSN) routine. In RASSN, the observations (reports) are compared against the entire satellite inventory and are identified according to their association with the predicted satellite positions. Those that meet very narrow limits established in the program are called fully associated reports (R_a's). Those that do not associate within this narrow tolerance or associate with more than one satellite are called doubtfully associated (R_d). The third category contains those observations which do not associate with any objects in the satellite inventory and are called unassociated reports (R_u). Most of the unassociated reports are the results of electrical noise or inaccurate observations. The SRADU tape (sorted reports associated, doubtful and unassociated) is the input to the Simplified General Perturbation Differential Correction (SGPDC) routine. Here the fully associated observations are used to correct the elements of each satellite.

Elements are the variables required to fully describe a satellite orbit. These consist of the right ascension, the inclination, the argument of perigee, the eccentricity, the time of epoch, and the period of the satellite. The outputs produced from the new elements are forwarded to the sensor in order that they may acquire the satellites on future passes. In addition, outputs are forwarded to NORAD and other users for tactical and scientific purposes.

The primary outputs consist of the bulletin and the look angle. The bulletin is a listing of equatorial crossing times and longitude for each revolution of the satellite. Also included with the bulletin is a table to correct the equatorial crossing to any location in the world. Look angles provide the position of one satellite for one sensor. This information is presented in the form of azimuth, elevation, range, and time for the specific sensor and for the specific satellite to be observed. Other outputs from our system include periodic reports required by NORAD and the other users of our information.

In addition to the A-1 system, there are about 20 other programs in use for satellite computations. These were originally written in FORTRAN and IBM machine language and were later converted to ALTAC for use with Philco SYS. These are now being converted to TAC for attachment to the A-1 executive.

SPACETRACK CONTROL ROOM

The computer outputs are manually checked for accuracy prior to release to the tactical or scientific communities. This is accomplished in our Spacetrack Control Room.

Compared to operational displays, the Spacetrack Control Room for aircraft surveillance operations is quite an unspectacular facility. (See Figure 3.) To date, no dynamic displays have been created which are entirely satisfactory for satellite purposes. In order to provide satellite data to CINCNORAD, static menu boards and a random access slide projector are used. Included are a summary of the satellite population, detailed information regarding the payload of each launch, and sensor data. A world map shows the location of the major sensors and centers that collect observations and the connecting

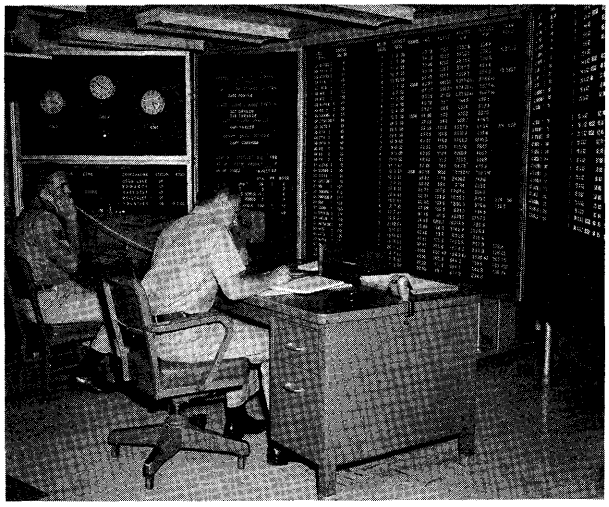


Figure 3. Space Scoreboards—Status boards in North American Air Defense Command's Space Detection and Tracking System Operations Control Center at Colorado Springs, Colorado, display timely tracking information on all man-made objects in earth orbit.

communications circuits. These are displayed to NORAD thru a closed-circuit television system.

BMEWS BACKUP

In addition to the satellite mission, the Philco 2000 has been programmed to provide a backup function for the BMEWS Display Information Processor located in the NORAD COC. The mission of BMEWS (Ballistic Missile Early Warning System) is to display to NORAD impact and launch data on missiles detected by the BMEWS which has radars at Clear, Alaska, and Thule, Greenland.

BMEWS RADARS AND FUNCTIONS

The BMEWS site at Clear consists of several fixed fan type radars. The site at Thule consists of a tracker radar in addition to several fixed fan radars. A future site being installed at Fylingdales, England, will contain all tracker type radars. These "forward" sites provide the coverage required to detect hostile missiles that may be launched against the North American Continent.

With computers located at these sites, predicted missile launch and impact points and the time of impacts are computed. This information is provided to the computer facilities within the NORAD COC for generation of displays.

BMEWS PROGRAM (B-1)

Data flows to the Philco 2000 from the forward site radars over high-speed data link circuits and thru the DMNI and Real-Time System. The B-1 program (see Figure 4) processes this information for display in the COC. Like the A-1 program, B-1 is completely independent of SYS. The B-1 program converts the forward site information into data suitable for drawing slides for a projection system and for driving numerical displays. The output is provided through the DMNO (Device for Multiplexing Non-Synchronous Outputs) to four devices: the impact and launch display decoder for projection system, the threat summary panel, the DMNO flexowriter for status data, and the remote transmitters that provide information similar to that displayed in the NORAD COC to the Strategic Air Command and the Joint Chiefs of Staff.

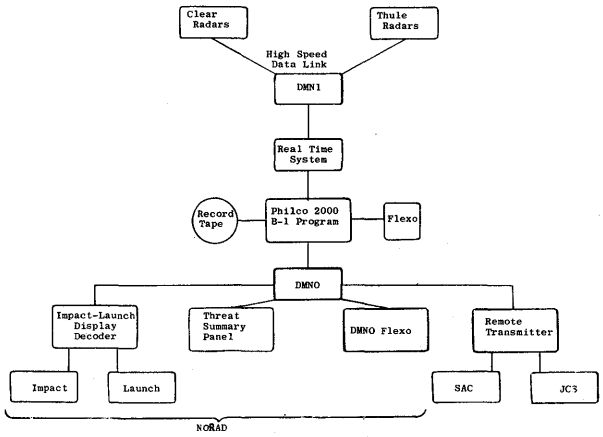


Figure 4. BMEWS Data Flow.

NORAD COC DISPLAYS

The BMEWS data are presented in the Combat Operations Center (COC) on two projection systems and a numerical display. The projection systems consist of two maps: one is of the North American Continent on which predicted impacts are displayed, and the other is a polar projection of the European-Asian Continent on which computed launch points are indicated. Both launch and impact locations are represented by ellipses. Included with each ellipse is a letter-number reference that identifies the forward site which detected the missile and a serial number for correlation between the two maps.

The Threat Summay Panel is a numerical display. "Five-minute windows" provide a measure of the size of the missile raid during the past five minutes. In another portion of the panel are shown the total numbers, for each site, of missiles detected and predicted to impact on the North American Continent. The time of next missile impact is displayed. Lastly, an "Alarm Level" is produced which is a combined measure of the missile raid and summarizes the credence of the threat.

FUTURE

In the near future the B-2 program will be integrated into the system. It combines the functions of both A-1 (for satellite) and B-1 (for BMEWS) under one executive. This will allow for full-time backup of the DIP computer and for processing satellite data simultaneously.

Under study is hardware which will completely automate the satellite inputs and outputs. The present manual teletype system

will be replaced by a computerized communications center.

CURRENT SATELLITE SITUATION

The satellites in orbit as of 24 September 1962 were as follows:

	USA	UK	USSR
Payloads in Earth Orbit	38	1	4
Debris in Earth Orbit	*185	0	4
Deep Space Probes (Heliocentric Orbit)	4	0	2
	227	1	10
Objects Decayed	109		60

*Includes 61 Omicron which exploded and produced 139 known objects

LIST OF REVIEWERS

Mr. S. N. Alexander
Mr. James P. Anderson
Mr. W. L. Anderson
Miss Dorothy P. Armstrong
Dr. M. M. Astrahan
Mr. Robert C. Baron
Mr. W. D. Bartlett
Mr. R. S. Barton
Mr. Aaron Batchelor III
Mr. J. V. Batley
Mr. M. A. Belsky
Mr. Eric Bittman
Mr. Erich Bloch
Dr. Edward K. Blum
Mr. Theodore H. Bonn
Mr. Arthur Bridgman
Mr. Herbert S. Bright
Mr. Edward A. Brown
Mr. L. E. Brown
Mr. W. Brunner
Dr. Werner Buchholz
Mr. James H. Burrows
Mr. R. V. D. Campbell
Mr. B. F. Cheydleur
C. K. Chow (Mr.)
Mr. Robert H. Courtney
Mr. Robert P. Crago
Mr. L. Jack Craig
Dr. T. H. Crowley
Mr. James A. Cunningham
Miss Ruth M. Davis
Dr. Douglas C. Engelbart
Mr. Howard R. Fletcher
Dr. Ivan Flores
Miss Margeret R. Fox
Mr. R. F. Garrard
Mr. Ezra Glaser
Mr. Jack Goldberg
Mr. Geoffrey Gordon
Mr. Joseph K. Hawkins
Mr. George G. Heller
Mr. George Heller
Mr. W. H. Highleyman
Mr. S. A. Hoffman
Mr. Arthur Holt
Mr. E. Hopner
Dr. Grace Murray Hopper
Mr. Richard A. Hornweth
Mr. Paul W. Howerton
Mr. Morton A. Hyman
Mr. George T. Jacobi
Mr. Robert Jayson
Dr. Laveen Kanal
Mr. Herbert R. Koller
Dr. R. A. Kudlich
Mr. J. Kurtzberg
Mr. Michael R. Lackner
Dr. Herschel W. Leibowitz
Prof. C. T. Leondes
Mr. Harry Loberman
Mr. J. D. Madden
Miss Ethel C. Marden
Mr. Walter A. Marggraf
Dr. R. E. Meagher
Mr. Philip W. Metzger
Mr. Albert Meyerhoff
Dr. Robert C. Minnick
Mrs. Betty S. Mitchell
Mr. Ralph E. Mullendore
Mr. Simon M. Newman
Mr. J. P. Nigro
Mr. Glenn A. Oliver
Mr. James L. Owings
Mr. G. W. Petrie
Mr. C. A. Phillips
Dr. Arthur V. Pohm
Mr. Jack Raffel
Mr. M. J. Relis
Mr. A. E. Rogers
Mr. David Rosenblatt
Mr. Arthur I. Rubin
Dr. Morris Rubinoff
Mr. Bruce Rupp
Prof. Norman R. Scott
Mr. I. Seligsohn
Mr. Donald Seward
Mr. J. E. Sherman
Mr. William Shooman
Mr. R. A. Sibley
Mr. Q. W. Simkins
Mr. R. F. Stevens
Dr. Richard I. Tanaka
Mr. Lionel E. Thibodeau
Mr. Robert Tillman
Mr. R. A. Tracy
Mr. R. L. Van Horn
Mr. Kenneth W. Webb
Mr. Gerard P. Weeg, Prof.
Mr. Thomas J. Welch
Mr. Stanley Winkler
Mr. Hugh Winn
Mr. H. Witsenhausen
Mr. William W. Youden

1962 FALL JOINT COMPUTER CONFERENCE COMMITTEE

General Committee

J. Wesley Leas, Chairman
RCA Building 201-3
Camden 8, N. J.

E. Everet Minett, Vice-Chairman
Remington Rand Univac
Blue Bell, Pa.

T. T. Patterson, Secretary
RCA Building 13-2
Blue Bell, Pa.

Finance Committee

Herman A. Affel, Chairman
Auerbach Corporation
Phila. 3, Pa.

Arthur D. Hughes, Vice Chairman
Auerbach Corporation
Phila. 3, Pa.

Public Relations Committee

Thomas D. Anglim, Chairman
Remington Rand Univac
Blue Bell, Pa.

Joseph Hoffman
General Electric Co.
Missile & Space Division
Valley Forge, Pa.

Thomas I. Bradshaw
RCA Building 202-2
Camden 8, N. J.

E. C. Bill
Remington Rand Univac
Blue Bell, Pa.

Proceedings Committee

Joseph D. Chapline, Chairman
Philco Computer Division
Willow Grove, Pa.

Walter Grabowsky, Vice Chairman
Auerbach Corporation
Phila. 3, Pa.

Program Committee

E. Gary Clark, Chairman
Burroughs Corporation
Paoli, Pa.

Arnold Shafritz
Auerbach Corporation
Phila. 3, Pa.

Aaron Batchelor, Vice Chairman
Burroughs Corporation
Paoli, Pa.

Robert S. Barton
1981 E. Meadowbrook Rd.
Altadena, Cal.

T. H. Bonn
Remington Rand Univac
Blue Bell, Pa.

Dr. Stanley Winkler
IBM, Fed. System Div.
Rockville, Md.

B. F. Cheydleur
Philco Computer Division
Willow Grove, Pa.

James L. Owings
RCA Building 82-1
Camden, N. J.

Dr. Hugh Winn
General Electric Co.
Missile & Space Division
Valley Forge, Pa.

Arrangements Committee

Peter E. Raffa, Chairman
Technitrol, Inc.
Phila. 34, Pa.

Robert A. Hollinger
Vice Chairman
Technitrol, Inc.
Phila. 34, Pa.

William McBlain
Minneapolis-Honeywell
Regulator Co., Pottstown, Pa.

Ladies Activity Committee

Miss Mary Nagle, Chairman
RCA Building 204-2
Camden 8, N. J.

Mrs. John M. Bailey
1032 Wayne Rd.
Haddonfield, N. J.

Mrs. John W. Mauchly, Vice-Chairman
Mauchly Associates
Fort Washington, Pa.

Miss Josephine Schiazza
RCA Building 204-2
Camden 8, N. J.

Printing and Mailing Committee

Norman A. Miller, Chairman
Remington Rand Univac
Blue Bell, Pa.

John Coston
Remington Rand Univac
Blue Bell, Pa.

Mrs. Ethel Levinson
Remington Rand Univac
Blue Bell, Pa.

Registration Committee

Louis F. Cimino, Chairman
General Electric Co.
Missile & Space Div.
Valley Forge, Pa.

Richard D. Burke
Vice-Chairman
IBM Corp.
Phila. 2, Pa.

John Schafer
General Electric Co.
Missile & Space Div.
Valley Forge, Pa.

Miss Eleanor Gardosh
General Electric Co.
Missile & Space Div.
Valley Forge, Pa.

Jack Armstrong
General Electric Co.
Missile & Space Div.
Valley Forge, Pa.

Miss Liz Gunson
IBM Corp.
230 S. 15th St.
Phila. 2, Pa.

Sol Steingard
General Electric Co.
Missile & Space Div.
Valley Forge, Pa.

Exhibits Committee

R. A. C. Lane, Chairman
RCA Building 204-1
Camden 8, N. J.

Lowell Bensky
Rese Engineering
A & Courtland St.
Phila., Pa.

W. P. Hogan, Vice Chairman
Leeds & Northrup
North Wales, Pa.

Special Events Committee

Herbert S. Bright, Chairman
Philco Computer Division
Willow Grove, Pa.

Dr. Louis R. Lavine
Philco Computer Division
Willow Grove, Pa.

B. F. Cheydleur, Vice Chairman
Philco Computer Division
Willow Grove, Pa.

R. Paul Chinitz
Remington Rand Univac
Blue Bell, Pa.

Edward H. Nutter
Philco Computer Division
Willow Grove, Pa.

John W. Mauchly
Mauchly Associates
Fort Washington, Pa.

Daniel Ashler
Auerbach Corporation
1634 Arch Street
Phila. 3, Pa.

Dr. Morris Rubinoff
Moore School of Electrical Engrg.
University of Pennsylvania
Phila., Pa.

Harry Bortz
IBM Corp.
230 S. 15th St.
Phila. 2, Pa.

Mrs. Margery League
Remington Rand Univac
Blue Bell, Pa.

Administration Committee

T. T. Patterson, Chairman
RCA Building 13-2
Camden 2, N. J.

John P. Brennan, Jr., Vice Chairman
RCA Building 82-1
Camden, N. J.

Technical Advisor

Dr. Morris Rubinoff
Moore School of Electrical Engineering
University of Pennsylvania
Philadelphia 4, Pa.

AMERICAN FEDERATION OF INFORMATION PROCESSING SOCIETIES (AFIPS)

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

Chairman

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

Executive Committee

Dr. Arnold A. Cohen, IRE
Dr. Harry D. Huskey, ACM
Dr. Morris Rubinoff, AIEE

Secretary

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

Treasurer

Mr. Frank E. Heart
Lincoln Laboratory
P. O. Box 73
Lexington 73, Mass

AIEE Directors

Mr. G. L. Hollander
Hollander Associates
P. O. Box 2276
Fullerton, Calif.

Mr. C. A. R. Kagan
Western Electric Co.
P. O. Box 900
Princeton, N. J.

Mr. H. T. Marcy
IBM Corporation
1000 Westchester Ave.
White Plains, N. Y.

Dr. Morris Rubinoff
Moore School of Elec. Engr.
200 South 33rd St.
Philadelphia 4, Pa.

ACM Directors

Mr. H. S. Bright
Secretary, ACM
Philco Computer Division
Willow Grove, Pa.

Mr. W. M. Carlson
E.I. duPont deNemours & Co.
Mechanical Research Lab.
101 Beech St.
Wilmington 98, Del.

Dr. H. D. Huskey
Computer Center
University of Calif.
Berkeley 4, Calif.

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

IRE Directors

Mr. W. L. Anderson
General Kinetics, Inc.
2611 Shirlington Road
Arlington 6, Va.

Dr. Werner Buchholz
IBM Development Lab.
P. O. Box 390
Poughkeepsie, N. Y.

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

Mr. Frank E. Heart
Lincoln Laboratory
P. O. Box 73
Lexington, Mass.

AFIPS Representative to IFIP

Mr. I. L. Auerbach
Auerbach Corporation
1634 Arch Street
Philadelphia 3, Pa.

Simulation Council Observer

Mr. J. E. Sherman
Simulation Council, Inc.
Sunnyvale, Calif.

STANDING COMMITTEE CHAIRMEN

Finance

Dr. R. R. Johnson
General Electric Co.
P. O. Drawer 270
Phoenix, Ariz.

Planning

Dr. Morris Rubinoff
Moore School of Elec. Engrg.
200 South 33rd St.
Philadelphia 4, Pa.

Admissions

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