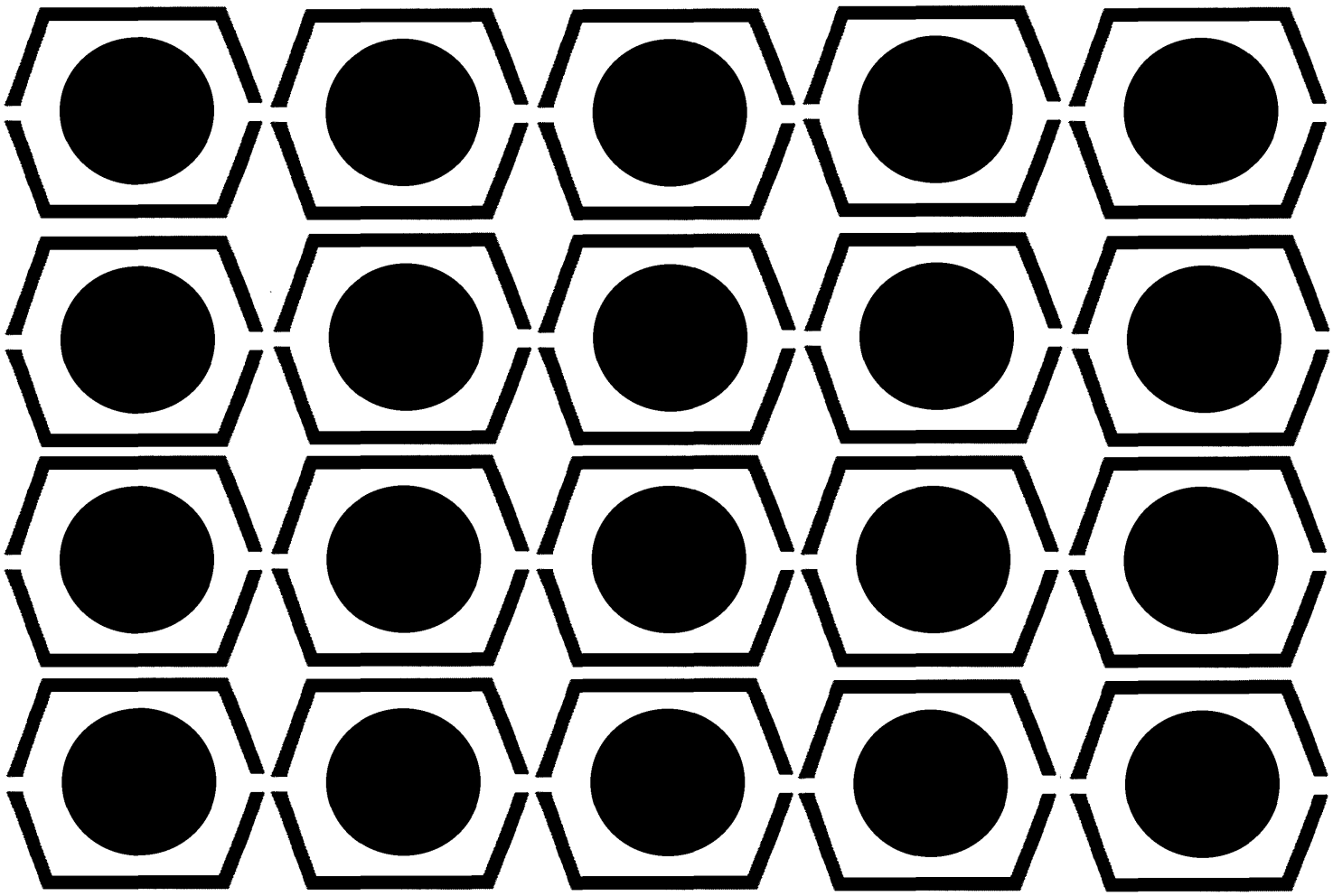


DECUS PROCEEDINGS

FALL 1965

PAPERS AND PRESENTATIONS

of

The Digital Equipment Computer Users Society

Maynard, Massachusetts

FALL
1965

PAPERS AND PRESENTATIONS

of

The Digital Equipment Computer Users Society

FALL SYMPOSIUM

November 29, 1965

Stanford University
Stanford, California

ACKNOWLEDGMENT

DECUS gratefully acknowledges the help of the Technical Publications Department, Digital Equipment Corporation in the preparation of these Proceedings.

CONTENTS

PREFACE

This symposium, like the one this spring, was scheduled to coincide with the Joint Computer Conference in the same part of the country in order to accommodate those traveling from afar. There seems to be no reason to change this procedure as long as DECUS continues to find gracious hosts at the right places at the right times. It was also the first attempt at a double session, single-day symposium and it appeared to be well received.

Of particular interest was the talk given by guest speaker David R. Brown of Stanford Research Institute on the logical design and implementation of future computers.

The sophisticated use of display scopes for data acquisition and general man-machine communication techniques was most impressive as were the tours of the Stanford Computation Center, the Medical Center, and the Linear Acceleration Center.

The delegates meeting was a surprise to many in that it included a discussion of the controversial subject of what to do with symposium papers which describe copyrighted programs which are for sale. The impact of exchanging software for money between users and outside professionals could cause some sweeping changes to DECUS, and it was therefore decided to encourage further discussion of the subject via DECUSCOPE and by a panel discussion session at next spring's symposium.

In summarization; the symposium was another step towards fulfilling the DECUS goal of information exchange.

John T. Gilmore, Jr.
DECUS Meeting Chairman

THE PDP-1 PROGRAM GRASP

R. Gocht, S. Sharpe

United Aircraft Research Laboratories
East Hartford, Connecticut

ABSTRACT

GRASP is a general purpose PDP-1 digital computer program. It uses the Digital Equipment Corporation's Type 340 Scope Display to provide effective and direct graphical communication between a problem in the computer and the operator. This GRaphics Assisted Simulation Program gives the "simulation" user even more versatile problem control than would an analog computer. The generality of GRASP allows it to accommodate many different types of problems with unique input-output control requirements. All solutions are displayed in graphical form on the scope. Variation of problem parameters and input data, including arbitrary function generation, is accomplished under light pen control. A short example is used to illustrate the general method of GRASP operation.

INTRODUCTION

Somewhere in the course of solving an engineering design problem, a mathematical model is constructed for the system under study. Two principle operations usually are involved in using a computer to study this model. The first of these is programming the model for the computer, and the second is running or using the simulation. These operations form the two loops in the flow diagram of Figure 1.

Much effort has gone into making it relatively easy for the programmer to program his model. Special languages such as MIDAS[1] (not to be confused with the PDP-1 assembly program MIDAS) have made digital simulation setup at least as easy as analog computer programming. Once the programming is done, however, the engineer must run the simulation. This paper describes one attempt to aid the engineer in the areas of controlling his calculation and looking at his answers.

The product of a simulation laboratory such as ours is a working or usable computer model of the physical system under study. Adequate communication with this simulation requires that it be accurate, reliable, repeatable, and easy to modify. It must operate inexpensively enough to allow the designer to work with the system continuously, i.e., on-line. The answers or solutions must be available to the engineer immediately; in fact for real-time flight simulation, a designer may fly for hours and the computer response must be in real time. In any case, the ability to adjust parameters and to see the response quickly is important.

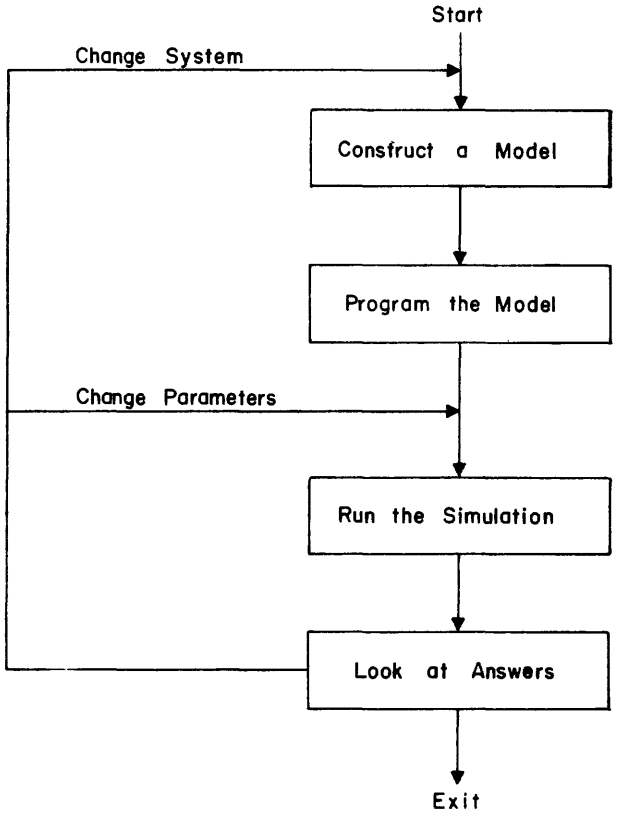


Figure 1 Typical Simulation Process

One of the traditional advantages of the analog computer has been the convenience with which an engineer can

communicate with the programmed model. Strip chart recorders, x-y plotters, input potentiometers, etc., provide a link between the engineer and the simulation that has been unique with this computer. There is a certain ease of use that the digital computer does not possess of itself. That is, the man-simulation relationship is a property of the hardware design and layout in an analog computer, while it is a property of the programming system in a digital computer.

Let us briefly examine the communication with a simulation that has been programmed on an analog computer. The purpose of a day on the computer is often a parametric study. After the problem has been initially set up and the x-y plotter calibrated, the first solution is generated. The response is seen and recorded as it is calculated. When the first solution has been completed, the engineer is usually interested in another solution with different values for one or more problem parameters. If so, he will label the first solution with its parameter values, proceed to change the desired parameters, and make another run. Having repeated this process several times, the x-y plotter will have recorded a family of solutions; and the corresponding parameter values will have been noted. At this point it may be desirable to examine some other variable. After a small program change, and a recalibration of the x-y plotter, the engineer is ready to record a new series of solutions.

In addition to varying parameters, arbitrary functions which are a part of the programmed model may be changed. A new function can easily be entered and solutions found using this modified model. The next level of change required may be to the system itself; this, of course, requires reprogramming and may be accomplished only to a limited extent on-line.

DEFINITIONS

GRASP has been written to effect a good communication system between a digital simulation and the operator of the simulation. The GRaphics Assisted Simulation Program uses the PDP-1 digital computer and associated scope display. It accepts output from any calculation which defines certain symbols and values which GRASP can recognize, and provides the linkage between the simulation and the user via the scope. The generality of GRASP allows it to accommodate many different types of problems with different input/output and control requirements. All solutions are displayed on the scope, typically in the form of x-y plots or graphs, and GRASP handles the scaling of the variables for display and the recording of parameters. Under typewriter control, different variables may be selected for plotting, with the display and modification of input functions accomplished by the use of the scope and light pen. Parameter values can be displayed numerically. The result is a communication between the simulation and user that is very direct and effective.

To see just how a user's program fits into the GRASP system we must have an understanding of the "handles" by which GRASP grips the calculation and of the entities upon which it operates. We shall begin with some definitions.

A light pen variable is a problem parameter in the user's calculation over which GRASP has control. GRASP controls the values of light pen variables by adding preset light pen increments to the current values of one or more of the light pen variables at user command (usually when a solution is touched with the light pen). A problem may contain as many light pen variables as the user wishes. He indicates the initial value and light pen increment of each and selects which light pen variables are to be modified during any particular operation.

A solution is a series of points (x-y data) that represent some output of a calculation for various values of an independent variable in the calculation. GRASP allows the user to generate a solution by calling for sequential x-y data output from the calculation. GRASP accepts the point data as it is generated; scales and stores the values for scope display; and when the solution is complete, displays the result on the scope. Thus the output need not only be engineering plots, but may be pictures which can be drawn from point to point as well.

A solution is defined by the collection of particular values of light pen variables which existed at the time the solution was computed. Since the current values of all its light pen variables are stored with the display data for a solution, a displayed solution which is touched by the light pen is easily identifiable. For example, when the light pen sees a solution, GRASP loads values of the light pen variables which define that solution into the current values of the light pen variables, and interrogates the user for the operation to be performed.

A bundle is a collection of solutions which can be considered belonging together in a group or set, each solution having its unique set of light pen variable values. A bundle may contain any number of solutions and is defined by name only. The name is assigned by GRASP; the user identifies a bundle by remembering which of the solutions currently being displayed belong together. To operate on a bundle, one touches a solution of the appropriate bundle with the light pen.

GRASP operates on the entities just described as it performs its various functions. As the simulation proceeds, the user must indicate which operations are to be performed and upon what, doing this by Macro programming; then by typewriter, sense switch, and light pen control.

OPERATIONS

To begin, the user must write the program to accomplish his particular calculation in the form of a subroutine called by GRASP. A special exit from the subroutine is used when the calculation decides that no more points are to be plotted. A second subroutine is required for

initializing the user's calculation. Operations such as setting the initial value of the independent variable and clearing program flags would be entered here.

In addition to these two routines, certain other information must be supplied in the user's program. For example, values pertaining to the control of light pen variables are listed following each variable. These are:

1. the initial value
2. the light pen increment (the amount which is added to the current value of a light pen variable on light pen contact)
3. an indicator which determines whether or not this particular light pen variable is to be modified during a given operation.

Also, variables which may be displayed along either of the axes are followed by scaling information which GRASP uses to transform the value of the variable in problem scaling into a number in proper display format. Finally, a list which specifies such things as which variables are to be displayed along the axes and the size of the grid on the coordinate display must be included.

Examine how this information is put together to obtain solutions. Figure 2 is a flow chart showing how the calculation proceeds. On it the user's subroutines are enclosed by double lines and GRASP operations by single lines.

GRASP initializes the calculation by executing the user's initialize subroutine and setting the light pen variables equal to their initial values. Also, the table of actual display data for this first solution is begun by depositing the current light pen variable values in the table for future reference, and the calculation proceeds point by point until a solution is complete. For each point issued by the calculation GRASP processes the variables which have been indicated for display, scaling the values and connecting sequential points with straight lines. When the solution is complete, the results are displayed on the scope.

Having one solution, we may be interested in another with different values for one or more problem parameters. GRASP allows us to modify any or all of the light pen variables and create a new solution under light pen control. We have set up the parameters of interest as light pen variables and an indicator in the problem with each of the light pen variables we wish to modify. We now point the light pen at the solution on the scope (i.e., we touch the solution with the light pen). GRASP senses this operation and identifies the solution we have touched by setting the light pen variables equal to the corresponding values which have been found in the list of display data for that solution. Then, GRASP looks up the light pen variables which are to be modified at this time, sets the current value of each to the current value plus the value of the light pen increment, and proceeds to compute a new solution as before.

We can operate on the solutions in the display in the above manner to add, move, or delete them. We can display both the solution we have just touched and the new solution (we call this operation add a solution); or we can delete the old solution and display only the new one (move a solution). A third alternative merely deletes the solution we touch without computing a new one. Add, move, or delete is selected under sense switch control.

The user can use these three basic operations to watch the response of his simulation as a parameter is moved through its range of values or as a family of solutions is built up. The fact that the operations occur under light pen control makes the simulation respond very quickly. In fact, the limiting factor on how fast we can perform operations on the simulation usually amounts to how fast we can think of things to do.

If one wishes to examine the effect of a parameter change on a whole family of solutions, bundle manipulation is convenient, as the same three basic operations: add, move, and delete, are allowed for bundles of solutions as well as for single solutions. Some input function could be changed, for example; and we could quickly see the effect on a whole family of curves. We might also recompute a bundle of solutions, displaying some new variables along the axes.

In addition to the basic operations on solutions and bundles outlined above, we have provided certain other features which enhance our ability to talk to the simulation. Probably the most important of these is the ability to display and modify input functions under light pen control.[2] Almost every simulation we do has input functions or curves associated with it. We can call up any particular input function (this can be a family of curves) to be displayed on the scope and the curves can be read or arbitrarily modified with the light pen. Either we can modify the function in a sequential point-by-point fashion by drawing in the curve desired, or we can shift the entire curve at once. Again, operations are selected under sense switch control.

The scope scaling routines in GRASP provide a versatile transformation from problem coordinates into scope coordinates. The user's calculation operates within a fixed-point coordinate system, the user selecting that portion of this fixed-point system which he wishes to display. This portion is a rectangular window through which is viewed any or all of the fixed-point coordinates. He may then specify both the size of the window on the scope and the position in which it is to appear.

The numerical value of the contents of certain registers can be displayed on the screen as a six-digit number. These "DVMs" are often used to display the current value of some problem parameter.

For reading points from a plot on the scope, a marking cross, which can be moved around where the light pen appears on the screen is provided. The coordinates of

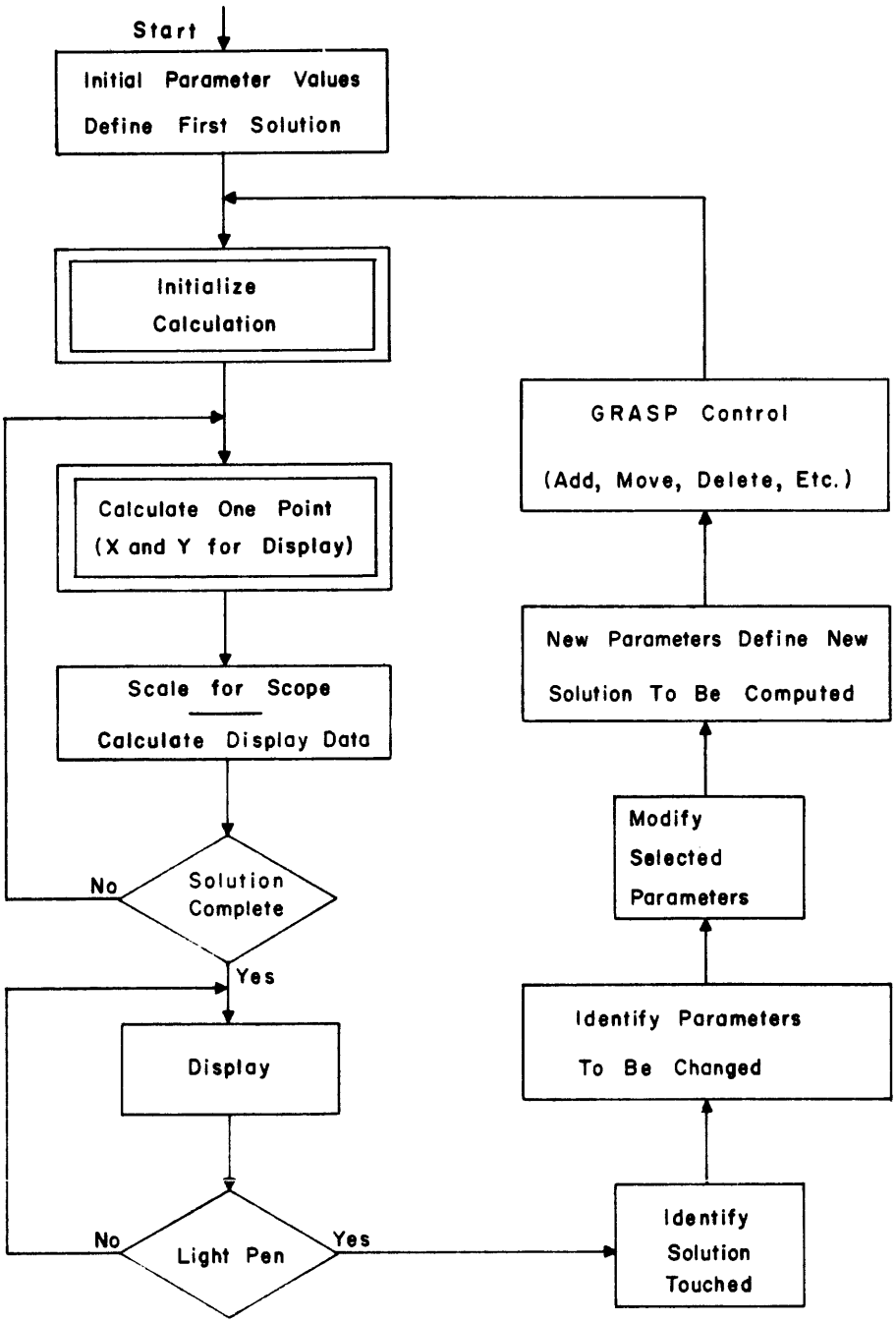


Figure 2 GRASP Operation on User's Calculation

the center of the marking cross appear in DVMs in a corner of the screen. Hard copy of the solutions can be generated on an x-y plotter if desired. This is usually used to keep a final copy of results.

So far, use of GRASP has shown that the control of solutions and bundles in the manner described in this paper provides us with the basic tools for an effective communication with the problem at hand. One is allowed quick access to the answers as they are generated, examination of several problem variables, and convenient variation of parameters and functions. In short, GRASP has proved to be a very useful simulation tool.

REFERENCES

1. R. N. Linebarger and R. D. Brennan, A Survey of Digital Simulation: Digital Analog Simulator Programs Simulation, Vol. 3, No. 6, December 1964, p. 22.

2. S. F. Jackson, Digital Function Generate and Display, UAC-14A, United Aircraft Corporation PDP-1 Program Library, January 1965 (Similar to DECUS No. 65).

DIRECT DIGITAL CONTROL TO HIGH ENERGY PARTICLE ACCELERATORS

Donald R. Machen

Lawrence Radiation Laboratory
Berkeley, California

ABSTRACT

The Bevatron, at the Lawrence Radiation Laboratory in Berkeley, exhibits many of the unmistakable characteristics of a multivariable process. Automatic control of machine parameters is of major interest to not only the operating management but experimental physics groups as well.

In order to fully explore the possibilities of stored-program digital computer control in the accelerator field, a PDP-5 computer, together with the necessary interface, telemetry channels, and remote sense logic, is being installed at the Bevatron.

Closed-loop automatic control of beam position and intensity between the linear accelerator and the main accelerating ring will be accomplished through the PDP-5.

INTRODUCTION

Generation of highly energetic elementary particles has progressed from the small-scale initial concepts on the physicist's laboratory bench to the realm of the large-scale "process plant"--a plant requiring the coordinated skills of many scientists, engineers, and technicians. As the energy imparted to the elementary particle increases, the form and breadth of the control problem posed by these accelerators begins to take on the unmistakable appearance of a multivariable process control system.

In a recent design study (Ref. 1) of a 200 Be V Proton Synchrotron, undertaken by the Lawrence Radiation Laboratory for the Atomic Energy Commission, it was noted that the amount of control information required for efficient accelerator operation would preclude the practice of hard-wire transmission and continuous presentation of these control variables which is presently common practice in operating accelerators. Table 1 indicates the type and number of control variables and information expected in a machine of this energy range.

A solution to this problem, as offered in the 200 Be V Machine Design Study, is centered about the unique abilities of the stored-program digital computer. The digital computer can, with the proper algorithms, perform the function of time-shared direct digital controller and operating-information processor for the central control area of such a machine.

TABLE 1 ESTIMATED NUMBER OF CONTROL VARIABLES, PROPOSED 200 BE V PROTON SYNCHROTRON

Type of Information or Control	Number of Points	
	Main Synchrotron	Injector
Status information	11500	6500
On-off control	3000	500
Analog monitoring	5300	2200
Set point control	2200	300

Several accelerator laboratories throughout the world are beginning to touch on the digital computer as a process controller for machine operation, notably, Argonne and Brookhaven National Laboratories, Stanford Linear Accelerator Center (SLAC), and the European Organization for Nuclear Research (CERN) in Geneva, Switzerland.

DEVELOPMENTAL RESEARCH AT THE BEVATRON

The prospect of nearly complete accelerator operation via a digital control computer has been viewed with some skepticism by accelerator people. The concepts for control put forth in the 200 Be V Design Study are new to the accelerator field, though several laboratories are planning to enter or have actually entered into limited automation with the digital computer as the central processing element.

The Berkely Bevatron, a 6.2 Be V proton synchrotron, was selected as the archetype for development research in the area of direct digital control and operator-information processing and display. A scaled-down computer control system operating on the Bevatron would not only serve to corroborate the proposals made in the 200 Be V Design Study, but would also allow the Bevatron scientists to investigate the worth of computer-aided control as an adjunct to the present control system.

For the initial effort, a section of the Bevatron, the inflection system, was selected for computer control. The inflection system located between the linear accelerator and the main accelerating ring serves to match to the main ring the proton beam emerging from the linac. Figure 1 depicts the physical location of the inflection system and the various devices attached to it.

As noted in Figure 1, there are steering magnets for both vertical and horizontal beam positioning, position monitors for vertical and horizontal beam position indication, triplet quadrupole magnets for beam matching and orientation in phase space, and inflector magnets for large-angle horizontal beam bending prior to beam entry into the main ring.

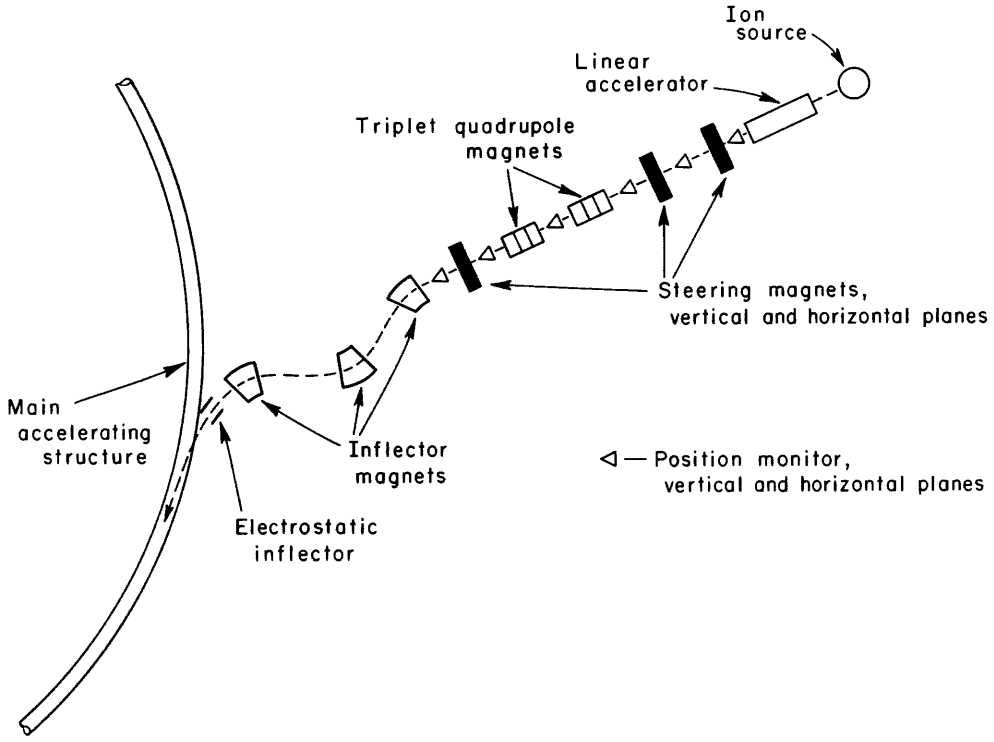


Figure 1 Bevatron Inflection System

The preliminary control problem to be attacked is that of closed-loop regulation of the beam position between the exit of the linear accelerator and the entrance of the first inflector magnet. A program of this nature requires that steering-magnet currents be monitored and controlled and that beam position in two dimensions be monitored as feedback information. In addition, the quadrupole magnet currents and inflector magnet currents must be monitored and used in a logical check routine along with the main control algorithm. As experience is gained on the beam regulation program, the more difficult adjustment of the inflector magnets and the quadrupole magnets will be attempted. With these programs, the charge in the main ring must also be monitored as a final performance index to the control program.

DETAILS OF THE BEVATRON COMPUTER CONTROL SYSTEM

A system similar to that proposed for the 200 Be V synchrotron, but sealed down in size and complexity, was desired for the Bevatron experiments. Figure 2 depicts the form of the control system proposed for the 200 Be V machine. The interface elements entitled "analog control" and magnet set point control, and "analog monitoring" serve to disseminate and collect information necessary for logical control of the process. To provide complete system information to the operator and the computer algorithms, interface elements entitled "status monitor" and "on-off control" are also included.

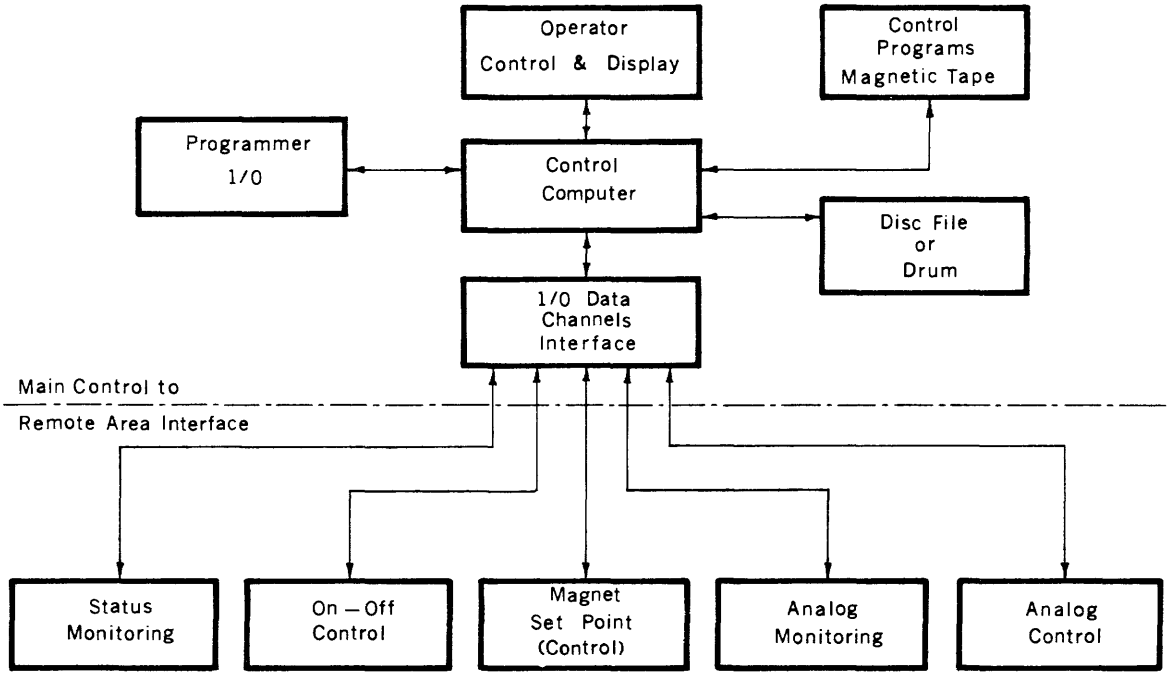


Figure 2 Control Computer and Telemetering Subsystems

Earlier this year, a PDP-5 computer was purchased for use as a direct digital controller at the Bevatron. Figure 3 shows schematically the details of interface for the Bevatron system.

This system will utilize two of the major interface subsystems proposed for the 200 Be V synchrotron—the analog monitor and the analog control subsystems. "Status" and "on-off" information will be handled within these two interface subsystems, as the Bevatron inflector control problem is much smaller than that foreseen in the 200 Be V synchrotron study. The logic will be essentially a time-shared multiplex system for both analog monitoring and control. Time sharing was dictated by the number of monitoring and control points (25 to 30 each) and the limited time involved in each acceleration cycle. Figure 3 also shows a typical acceleration cycle. The injection time (the time in which beam is within the inflector system) amounts to only 600 μsec out of a 6-sec acceleration period. Critical items such as beam position are monitored upon computer command by sample and hold amplifiers and then converted to a digital word in a sequential way during the rise time of the Bevatron field. Magnet currents are next monitored sequentially--some during acceleration and some after acceleration is completed, but before the next injection cycle.

As noted on Figure 3, system interrupts to the computer generate at injection, or the beginning of acceleration, and at completion of inversion, or the end of acceleration. Various control algorithms are initiated depending upon the interrupt involved.

Upon execution of the initial beam-injection interrupt, the interrupt denoted as 1st in Figure 3, the computer monitors all beam position signals and scales them to the proper normalized values. The quadrupole magnet currents are then set to a predetermined value and an initial check of beam deviation from the path center line is made. Should there be a deviation at the entrance to the first inflector magnet, a systematic check is made of each position, in each dimension, going upstream to the exit of the linac. When the offending steering magnet is located, its current is set first to a positive maximum and the beam position is read, then to a negative maximum and the position is read. Three injection pulses are required for this operation. The assemblage of beam position indications is then used in a convergence calculation to determine the appropriate magnet current for zero beam deviation from center line. This routine is repeated for each steering magnet until the beam is positioned properly. After completion of tuning, position monitoring continues, with the capability of retuning when necessary.

In addition to the control algorithms a group of service routines will be available to the operator. These routines will allow selections of groups of variables for visual display on a storage tube oscilloscope or on the Teletype. The operator may also change the course of the control program flow by appropriate commands, e.g., enabling or disabling the regulation routine, changing the variables monitored on each interrupt, etc.

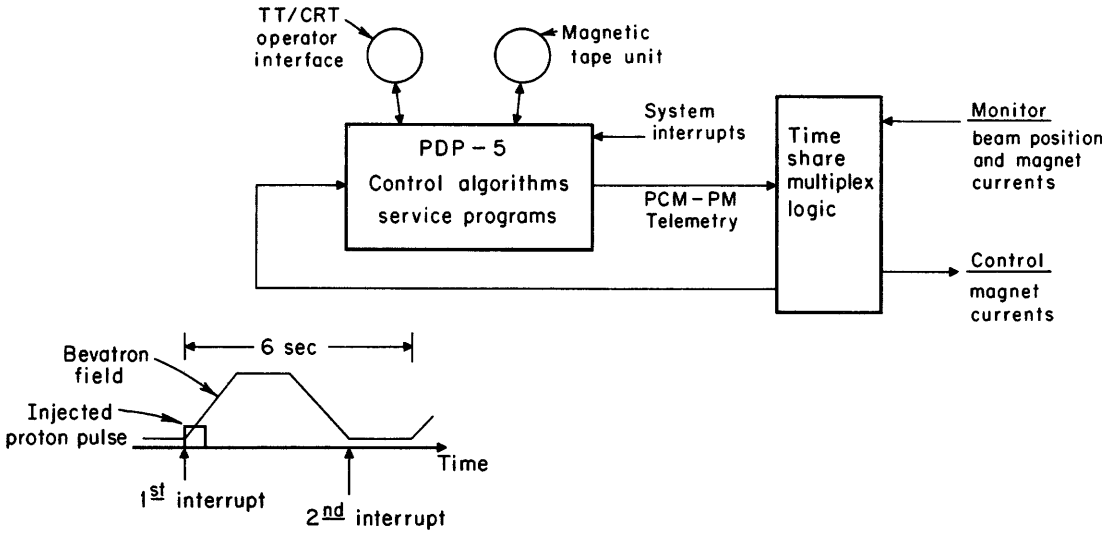


Figure 3 Closed-Loop Computer Control of Beam Position: System Schematic and Timing (Berkeley Bevatron Injection System)

INTERFACE DETAILS

For the initial control program (beam-position regulation) all computer input and output will be passed through the accumulator. In a later experiment, in which control and information processing will take place within a single acceleration pulse, data break mode will also be utilized.

The necessary logic for addressing both the monitoring and control devices, converting and sending monitored information to the computer, and sending control information to the remote area will be constructed from Digital Equipment Corporation FLIP CHIP modules. A few items, such as the read-relay multiplexers and read-relay digital-to-analog converters, are constructed by the Lawrence Radiation Laboratory.

All information sent to and from the computer will be in serial words from 7 to 13 bits long, depending upon the function. Telemetering between the computer and the remote area is over twisted-pair shielded cable, transformer-isolated at each end and essentially in pulse-coded phase modulation form. There is, therefore, a 180° phase shift between a transmitted 1 and 0. Because of the type of decoding required to re-form each bit of the receiving end, this type of bit encoding allows for a greater degree of noise immunity.

CONCLUSIONS

At present, tuning the Bevatron inflection system for beam position requires a great deal of time and effort on the part of the operating personnel. It was felt that computer-aided control of this portion of the Bevatron would not only reduce the tune-up time, thereby aiding the Bevatron operation, but also provide an experimental facility for developmental research on computer control of accelerators in general.

The lessons learned and experience gained through a project of this nature will allow future machines of higher energy to be controlled in the most effective manner by digital computer systems.

ACKNOWLEDGMENT

The help of R. W. Allison, J. B. Greer, S. C. Robison, and C. D Pike of the Radiation Laboratory Staff is gratefully acknowledged.

REFERENCES

1. 200 Be V Accelerator Design Study, Lawrence Radiation Laboratory Report UCRL-16000, June 1965.

CURRENT TRENDS IN HYBRID COMPUTER ORIENTED SOFTWARE

Stephen F. Lundstrom

Consultant to Applied Dynamics Inc.
Ann Arbot, Michigan

ABSTRACT

Many interpreters and compilers have been developed recently to assist the engineer in the complex job of programming a hybrid computer problem. After a shoft discussion of the hardware involved in hybrid computers and some of the unique problems therein, discussion centers on the functions of some of the more outstanding representatives of the current software developments.

Some of the software developments discussed include SPOUSE, MAID, TAOIST, and FOCHUS. The interpreter portion of SPOUSE (Stored Program OUtput Setup Equipment) is designed to assist the engineer in the setup and check-out of the analog subsystem. The MAID software package is used by engineers and technicians in the automatic diagnostic check-out of the analog subsystem. TAOIST, a program originally developed for a 12K core PDP-8, aids the engineer in the initial phases of hybrid and analog computer problem solution. It accepts a description of the problem in mathematical terms and provides the engineer with a list of all problem variables, coefficients, scaling factors, and other useful information. Upon assignment of analog equipment, the program will automatically do all static check-out of every analog component used in the analog subsystem. The last main point of discussion is the FOCHUS (FOrtran Compiler for Hybrid USers) program. This is a compiler designed to accept FORTRAN code, but also to be useful to the engineer in the solution of hybrid problems since a fairly close control over communication between the analog and digital subsystems is allowed.

The paper concludes with some short mention of possible future uses, modifications, and developments in software.

Recent interest in simulation of complex missile and spacecraft missions, simulation of transport delay systems, and the study of plant system control processes are a few of the many areas of investigation spurring the current interest in hybrid computer systems. The technique of coupling analog computer systems and digital computer systems has resulted in a new system with high accuracy and a large logical capability, especially for discrete variables, and high speed and wide bandwidth, especially for continuous variables. Unfortunately, as has often been the case in the past, the software tools to utilize such a system efficiently were not immediately available. However, today we can consider some recent developments which are of as much interest to people in the field as some of the implications of this work are to others. Some of the major problems facing an engineer designing a hybrid problem solution are considered below with an explanation of areas in which these problems are being overcome.

Consider mechanization of the general simulation description onto the available hardware. Three steps are necessary: First, an initial decision must be made as to how to partition the problem between the digital and analog subsystems. Mechanization of the two parts must then be accomplished as the second and third steps before any results are obtained. Recalling the inherent problems with and advantages of each subsystem, the digital subsystem has a high precision and a large logical decision capability. However, the digital subsystem is usually unable to rapidly simulate systems which may be described with a set of differential equations, for example, or to simulate in real time systems involving large numbers of interacting continuous variables. Consideration of the analog subsystem immediately evidences the opposite as far as advantages and disadvantages are concerned. Thus, in a simulation of a digital process control application, it is obvious to simulate the process on the analog subsystem and the digital controller on the digital subsystem.

A somewhat less obvious result occurs in the simulation of a spaceship mission. Here the variables which must be known precisely are the position coordinates of the spacecraft. These variables are not changing rapidly relative to other variables. This fact allows the accurate, but somewhat slow, digital numerical integration procedures to work here. Variables of velocity vector and acceleration changes during midcourse trajectory correction maneuvers are of interest also. These variables obviously change very rapidly, but the great accuracy of the position of the spacecraft is not required which allows the analog subsystem to simulate these variables.

Once it is decided which portions of the solution are to be on which part of the system, it remains to implement these portions. Consider the digital portion first. Due to the physical structure of hybrid computers and the relationship of the variables in the digital subsystem to those in the analog subsystem, it is necessary in any program to transfer a considerable amount of information between the two subsystems. This transfer of information, which is dependent on the needs of the progress of the solution, implies that the engineer must have close control over the interface between the subsystems. To date many advances have been made in the area of compilers for digital computers; however, the trend in these advances has been to carry the programmer farther and farther from direct control of digital computer hardware. For most applications this is advantageous. However, in a hybrid computer situation it is a definite disadvantage. When the analog subsystem requires data, that data request must be handled immediately, and in a way unique to that problem. Likewise, digital interrogation must be done precisely when required because the time base of the two subsystems must coincide. Thus, the requirement of close control over at least the interface hardware is unavoidable. The engineer would still prefer to couch the terms of his solution in compiler language rather than assembler language. The above problems have been the motivation for the FOCHUS (FOrtran Compiler for Hybrid USers) compiler and operating system which has been developed by Applied Dynamics, Inc. Although this problem has been developed for PDP-8 systems, the concepts obviously are applicable on other systems. FOCHUS is a FORTRAN compiler and operating system. The main features of the FOCHUS system are the flexible handling of subroutines coded either in FORTRAN or in machine code, the accommodation for arrays in up to three dimensions, and variable DO loops. The object program produced is normal machine code. The object program runs under the supervision of an operating system which handles all standard I/O and library manipulation. Interrupts caused by peripheral equipment are noted by the operating system, and the device generating the interrupt is put into a hold situation. Program control is interrupted only during execution of the main program and upon transfer of control from subroutines. The FOCHUS compiler is a two-pass compiler. Subroutines may be compiled separately allowing creation of a standard library. The appropriate subroutines from the library are introduced during the second pass to generate the object program. This implies relocatable subroutines and some form of intermediate storage. The intermediate storage currently may be either core, paper tape, or magnetic tape. The use of FOCHUS allows effective, more direct control ot the digital subsystem by the engineer.

Shifting attention to the analog subsystem we will see a number of problems. Consider the previous problem in implementing the digital solution. Mechanization of a solution on the analog subsystem requires the engineer to reduce the model of the system to be solved to an interconnection of analog elements (summers, integrators, multipliers, coefficient devices, etc.). He must then scale the problem so that the analog solution lies within the useable amplitude and frequency response of the analog elements. This is susually done intuitively, followed by trial and error manipulation with the equipment before meaningful results are possible. To assist the engineer in the realization of the solution he seeks, Applied Dynamics, Inc. has developed the TAOIST (Treatment of Analog Operation by an Integrated Simplified Translator) program. The TAOIST program accepts the set of differential equations which models the system from the engineer. After checking the equations mathematically, it requests from the engineer information about assignment of analog elements to each of the variables requiring them. When this is done, the TAOIST program automatically performs scaling on the analog mechanization. TAOIST, which is a resident program in the digital subsystem, then sets the coefficient devices to the appropriate settings and performs a complete problem check of the initial conditions of the problem setup. If required, TAOIST can also compute check points for the dynamic solution so that the engineer can verify quickly that the simulation on the analog subsystem does indeed model the system of interest. Changes in the model are easily made using TAOIST. Such changes are entered at any time and setup proceeds as previously.

Although TAOIST program was originally realized on a PDP-8 with a 12K word magnetic core bank, it is available in a multipass version for the standard PDP-8 with 4K of memory or a single pass version on a standard PDP-8 with DECtape option. Such a program is obviously useful on larger digital subsystems also.

Before development of the TAOIST system, Applied Dynamics had developed the SPOUSE (Stored Program OUtput Setup Equipment) system. This system consists of a digital computer, often a PDP-8, a SPOUSE interpreter program, a library of hybrid and input-output subroutines, and the interface required to properly link the analog and digital subsystems. SPOUSE equipment, controlled through the interpreter, allows the engineer to control setup and check-out of analog simulation di-

ectly through the input console on the digital computer. Prepared descriptions of analog setups may be punched on paper tape or stored on other storage media, to be read automatically by the interpreter and to cause the proper analog setup without operator intervention. The SPOUSE development was historically the first commercially available stored program of its kind. It had the major advantages of allowing the analog system to be treated much the same way as a digital system. The engineer now could come to use the analog system, read in his prepared setup tape, and be ready for operation in a very short time. Previous manual methods dictated that once a problem was set up, all data had to be obtained before anyone else could use that system. Using SPOUSE, this restriction has been lifted, and much more efficient use may be made of the equipment. The final problem to be considered is the one of system maintenance. Digital systems usually have a set of diagnostic programs provided by the manufacturer to check the accuracy and functional operation of the digital system. Unfortunately, such system maintenance on analog systems has been generally a manual chore. Applied Dynamics has available now a new system, called MAID (Maintenance AID), to perform automatic check-out of almost all of the analog components in the analog subsystem as well as the interface part of the hybrid system. Any elements found to be performing incorrectly are indicated by appropriate comments on the on-line typewriter of the digital system. MAID basically consists of a set of prewired program boards for the analog computer and a control program resident in the digital subsystem which controls the check-out through a SPOUSE system. Both MAID and SPOUSE are described in greater detail in the paper "The Use of a Digital Computer in the Setup, Check-out, and Maintenance of Hybrid Computer Systems" presented to the 1965 DECUS Spring Symposium by this author.

The previous discussion has shown how SPOUSE, MAID, TAOIST, and FOCHUS allow better and more efficient usage of the hybrid system. To conclude, consider a few possible improvements which might be projected for the future of such systems. One of the dreams of anyone who has ever worked with analog systems is that the patching of the analog component interconnection be done automatically. Undoubtedly, technology is advancing to a point where such a system will eventually be economically feasible. If this is implemented with an automatic switching system, potential use of analog subsystems as rapid differential equation solvers associated with large scale digital machinery would be reasonable. An extension of TAOIST, the analog setup aid, is obvious along these lines also. Implementation of algorithms to assign interconnection of analog elements would be advantageous. Obviously, both of these can combine into an interesting system which has the set of differential equations to be solved as the input and the operating physical analog of the system as the output.

SPOUSE, MAID, FOCHUS, and TAOIST systems have proved to be definite assets in the area of hybrid simulation. Undoubtedly further use will further expand their abilities as new techniques in the field are used. Such expansion will lead to areas not presently conceived. Two guiding principles will, however, be common to the programs already discussed and future developments: the generation of software, and equipment to allow faster and more powerful solutions ot the problems of interest.

SPIRAL READER CONTROL AND DATA ACQUISITION WITH PDP-4

Jon D. Stedman

Lawrence Radiation Laboratory
Berkeley, California

ABSTRACT

The spiral reader is a semiautomatic, film-digitizing machine that is used by the Alvarez Physics Group at the Lawrence Radiation Laboratory to measure photographs of elementary nuclear particle interactions which are created in a hydrogen bubble chamber. This man-machine-computer system has been measuring these event interactions at an average rate of 75 per hour. As well as being the fastest overall measuring system in operation, it has proven to be highly reliable and easy to maintain.

This system incorporates several unique features that contribute to its efficient operation and high processing rate. A PDP-4 computer controls the following functions: procedure sequencing, data acquisition, data conversion and validity checking, data display, machine diagnostics, machine control, and reaction feedback to human interventions. The operations performed by the spiral reader are: digitizing the event on the film, digitizing the fiducial marks, advancing the film frames, and responding to human control of its event centering stage via the "speed ball." Finally, and the most important part of the system is the human operator who is relied upon to monitor and control the other two parts by looking at the data display CRT, reading the indicative and diagnostic messages from the Teletype, verifying the automatic film frame position using the control button, recognizing and centering on events by means of the speed ball finger control, and giving measure commands with a control button. This balanced combination of man and machine has proven to be an excellent impedance match with the other analysis procedures encountered in experimental bubble chamber physics.

INTRODUCTION

The spiral reader is a semiautomatic, film-digitizing machine used by the Alvarez Physics Group at the Lawrence Radiation Laboratory, Berkeley, to measure photographs of elementary nuclear-particle interactions created in a hydrogen bubble chamber. This statement requires some explanation for those not acquainted with this specialized area of research.

One example of a hydrogen bubble chamber is the 72-inch chamber currently in use at Berkeley. This chamber, constructed of steel and glass, resembles a bathtub whose interior dimensions are approximately 72 x 14 x 20 inches. A glass window is mounted on top of this chamber so that its interior can be photographed. Inside is liquid hydrogen at high pressure and low temperature. Surrounding the chamber are coils of a large magnet whose 18,000 gauss field causes moving charged particles traveling through the chamber to be deflected into circular trajectories.

Pulses of high-energy nuclear particles produced at the Bevatron are collimated and focused so that they enter the hydrogen chamber at one end. With the pressure on the liquid hydrogen momentarily reduced, each charged particle of the beam passing through the chamber causes a wake of small bubbles along the length of its circular trajectory. These bubble tracks are clearly visible and are photographed to capture the fleeting evidence of the particle motion.

Three cameras mounted above the 72" chamber photograph each beam pulse that passes through the chamber. A roll of film with a capacity of 600 frames is threaded through the cameras in

series, so that the stereo triplets will be on the same physical piece of film. These bubble chamber pictures are taken at a rate of 6 frames/minute.

During an operating year, some 4,000 rolls of film are exposed. These stereo photographs constitute the raw data to be analyzed.

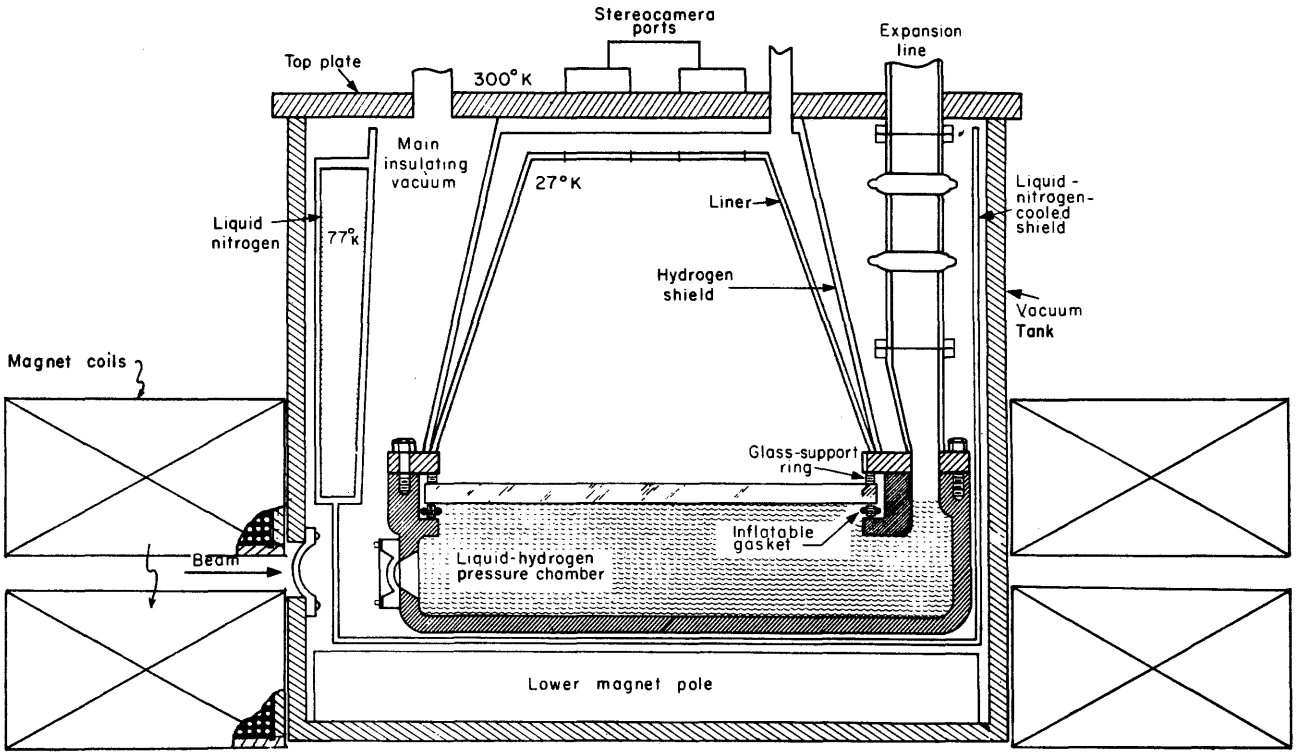


Figure 1 72-Inch Bubble Chamber

Occasionally, one of the beam particles makes a direct hit with the nucleus of a hydrogen atom. This collision may result in a simple ricochet or the production of numerous other particles which in turn produce their own tracks. Physicists are interested in the mechanism which produces new particles, and the bubble chamber is a basic tool for discovering them and studying their dynamic properties.

DATA ANALYSIS PROCEDURES

The first step in the analysis procedure is to scan each frame to locate, identify, and record the occurrences of interacting beam particles. Trained technicians perform this task by scanning magnified projections of the film and recording the indicative data on each event they find. This information forms the first history of each event and consists of such items as the roll and frame number, the type of event, and the approximate location of the event within the chamber. This data is punched into cards used to update a master library file.

The next step involves digitizing coordinate points along particle track images. These coordinates are used to compute such physical quantities as charge, energy, direction and range of particles involved in an interaction. Technicians manually perform this digitizing with the aid of measuring engines that record rectangular coordinates along the particle tracks with respect to the coordinates of known fiducial marks within the picture. A stereo reconstruction in three dimensional space of a particle track within the chamber can be computed from the rectangular coordinates measured along such a track in two corresponding stereo views of that track.

The basic designs of the measuring engines are those developed at Berkeley and known fondly as the Frankenstein. These machines, complete with a number of elaborate automatic features are in the $100,000 price range, and can measure at a rate of eight events per hour. With these rough figures, you can see that it requires a good number of these machines, each with its own human operator, to keep up with the flood of film from the bubble chamber. The measuring phase of the analysis of the bubble chamber pictures becomes a bottleneck in the system due to the track oriented nature of the Frankenstein measuring procedure and the lack of sufficient automatic data handling and control.

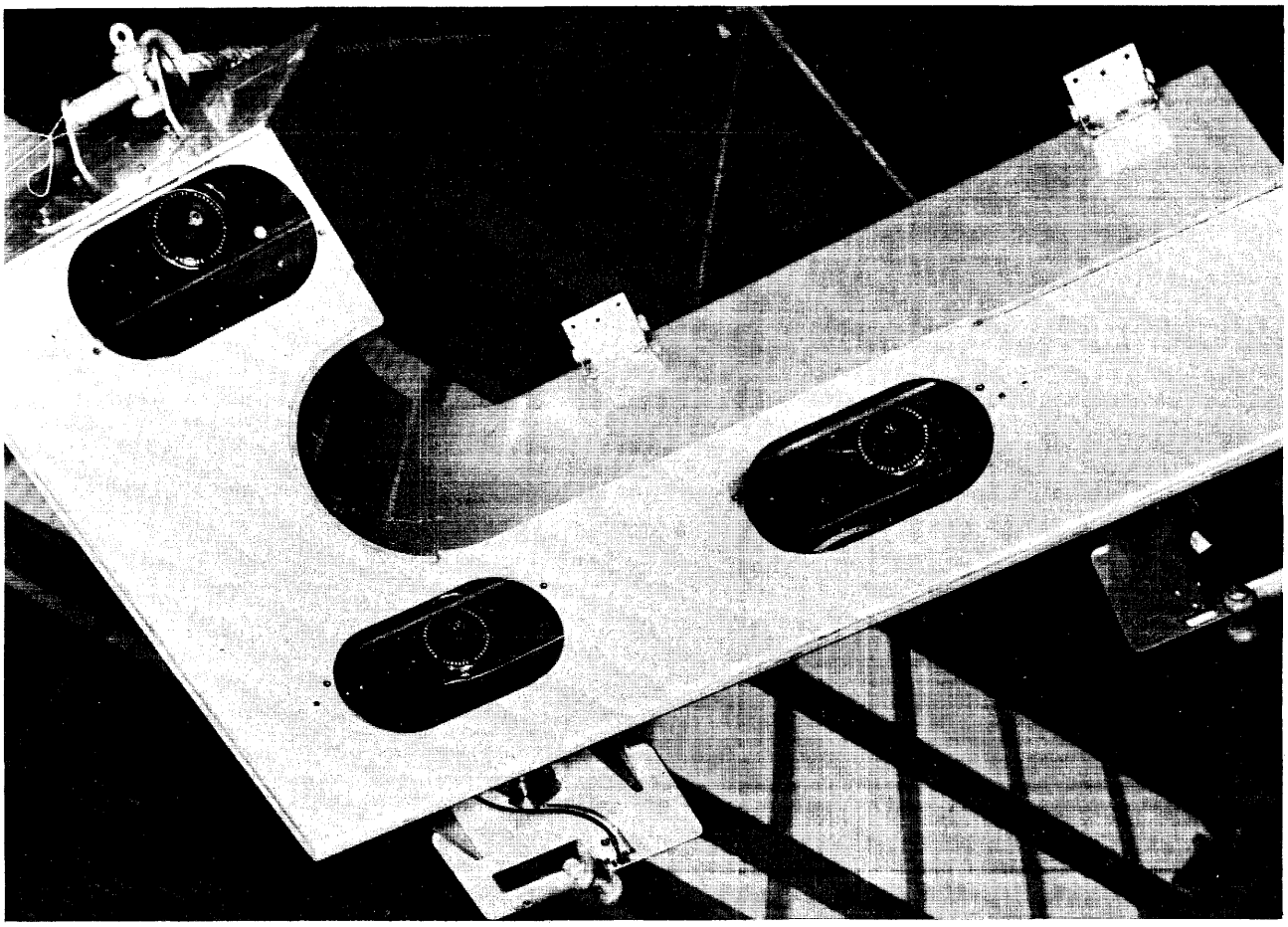


Figure 2 Three Bubble Chamber Cameras

SEMIAUTOMATIC MEASURING MACHINE

The spiral reader is a measuring machine whose design was motivated by the characteristic pattern of most bubble chamber events; that is, the particle tracks associated with an event appear to radiate from a central point of origin called the event vertex. Once centered on the vertex origin of an event, the SR scans the film along a spiral path starting at the vertex. A hybrid system of pulse discrimination circuits processes the analog signal generated from the phototube used to make this scan. Only a select variety of pulses is filtered from the analog signal. These pulses strobe the current polar coordinates of the scanning element. In this way, all dark spots on the film that intersect the spiral scan digitize into polar coordinates relative to the vertex origin. Hence, the SR is not limited in speed by the number of tracks associated with a vertex, thus overcoming the inherent limitation of track-by-track oriented measuring procedures. The spiral reader is a vertex-oriented measuring machine.

It is apparent that the coordinate data resulting from measuring an event with the SR contains points which do not lie on the tracks associated with the event. To compute the space reconstruction of an event, it is necessary to filter the SR data to extract only the desired points associated with event tracks. This pattern recognition is a simple matter for the human mind, but requires a very large and complicated FORTRAN program for an IBM 7094 computer.

The filtering program POOH exploits the fact that the event tracks start from zero radius and are arcs of circles. If the reader examines a display graph of the polar data plotted in rectangular space with the azimuth taken as the X-axis and the radial coordinate as the Y-axis, he can see that the desired track points are colinear and extend down into the region of small radius. A histogram of the data in this region of small radius provides a powerful technique for localizing the desired track points. Once a track is roughly determined in this way, very selective fitting criteria are sufficient to isolate the remaining length of the track. For the case of very short tracks, this procedure is inadequate. In this case, the operator who is using the SR furnishes some additional data points called "crutch points" that aid the POOH program in filtering short and/or confusing tracks.

Even though the spiral reader's operational debut dates back to 1960, it has been limping along since that time held together by circumstance. Once a vertex could be centered, the SR made quick work of it, but the numerous manual steps intervening were too complicated to be performed by an operator quickly and reliably. Hence, whenever the SR could be made to operate at all, its performance was a disappointing 15 events per hour. It was the awkward nature of controlling and sequencing the spiral reader that proved to be its Achilles heel !

SPIRAL READER AND PDP-4

In May 1964, a plan was developed whereby a PDP-4 computer would become a controller of the sequencing and mechanical movements of the periscope, stage, and film drive mechanisms of the SR. Implementing this new system required the design and development of the logical interfacing circuits that would allow the computer to become a part of the SR device servo loops This design involved the planning of the control type instructions needed for the mechanical devices and balancing the tradeoffs between software programming and hardware logic. Some of the more important design objectives were as follows:

All devices should have a manual control and computer control mode.
All devices should have identical hardware with similar control characteristics.
All devices should have mechanical and electronic interlock protections.

The SR system can be seen in Figure 9 which shows the optical and physical layout of the devices as well as the communication and control interconnections. The various devices are listed below for reference.

1. Film motion and frame selection control.
2. Stage motion control and position readout.
3. Periscope motion control and SR data input channel
4. Magnetic tape data output channel
5. Operator console control buttons
6. Auto-fiducial sensing elements
7. Scope display channel

The SR also makes operational use of the standard Teletype and paper-tape reader.

Most of the devices of the SR listed above can be treated essentially the same by the programs residing within the computer. Each device has associated with it a status register containing information bits that indicate the current status of the particular device. Several of the status bits tie into the interrupt facility of the computer to allow real-time processing of the device conditions by the controller.

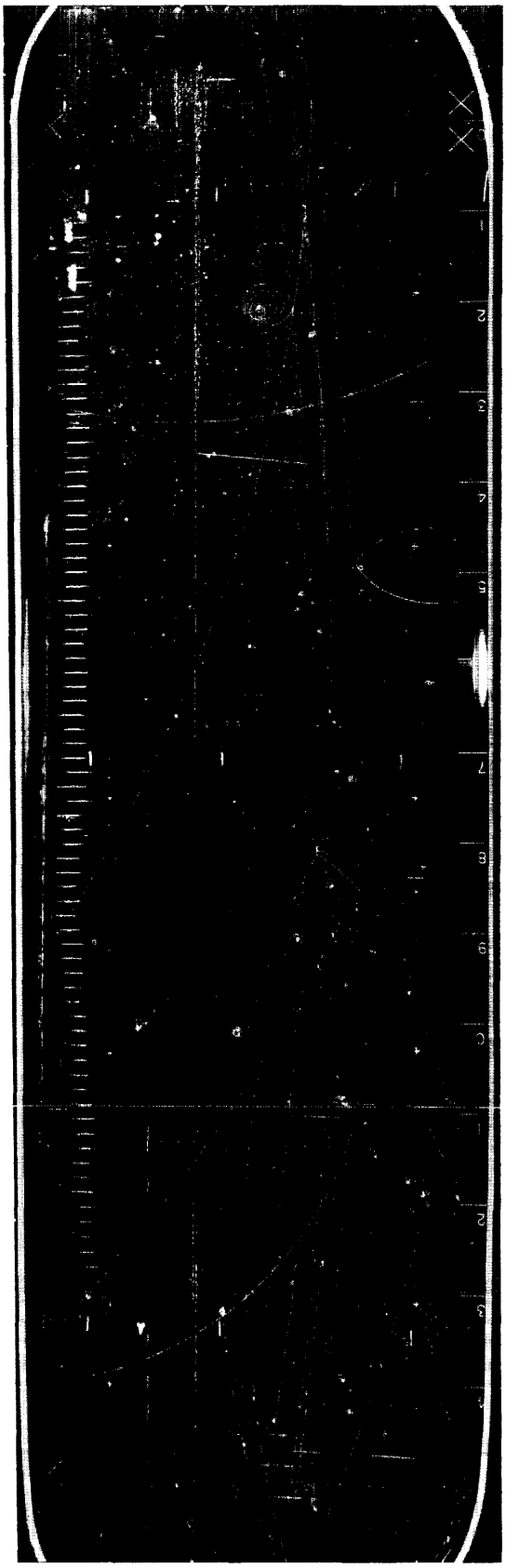


Figure 3 Full Length View of 72-Inch Chamber

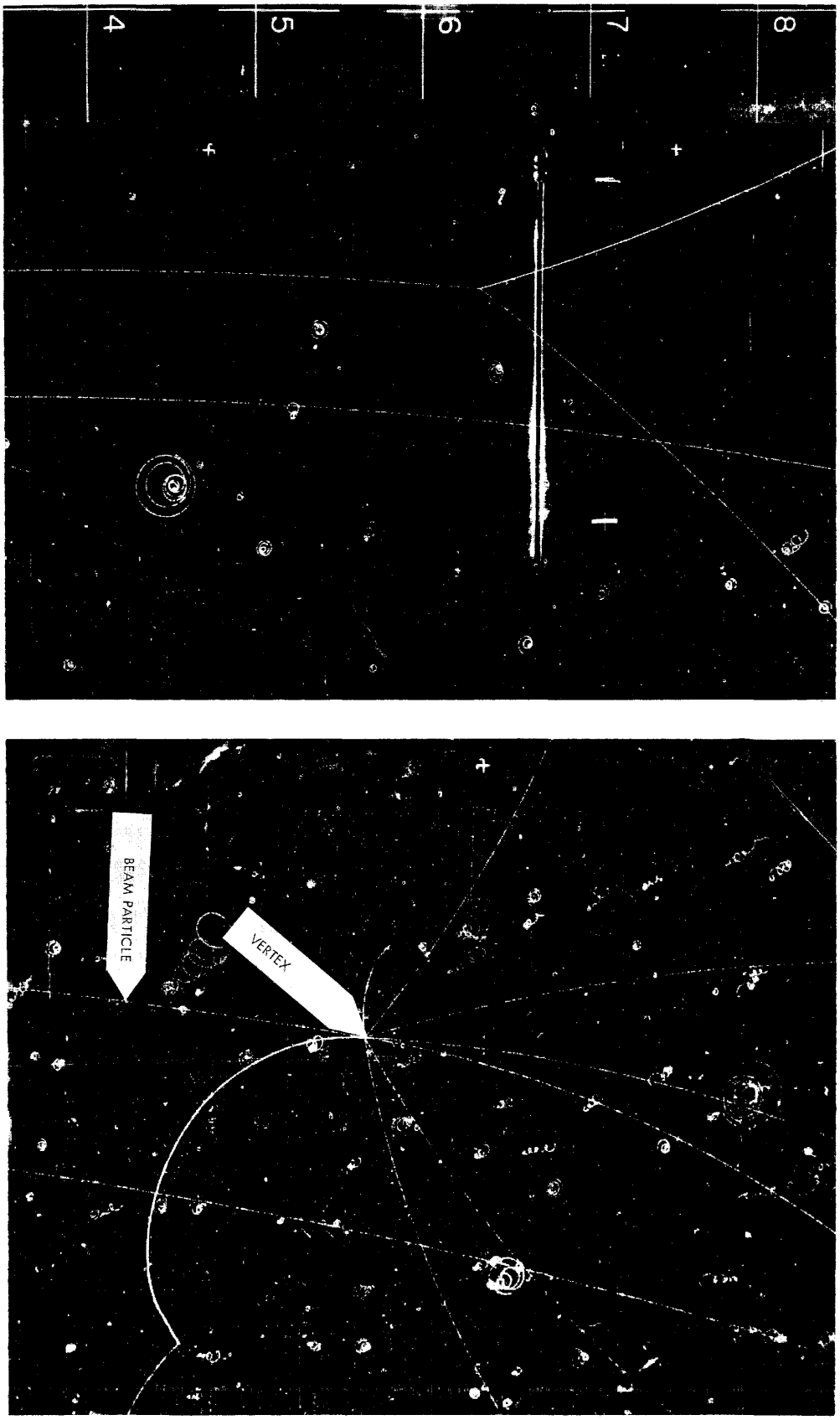


Figure 4 Bubble Chamber Events

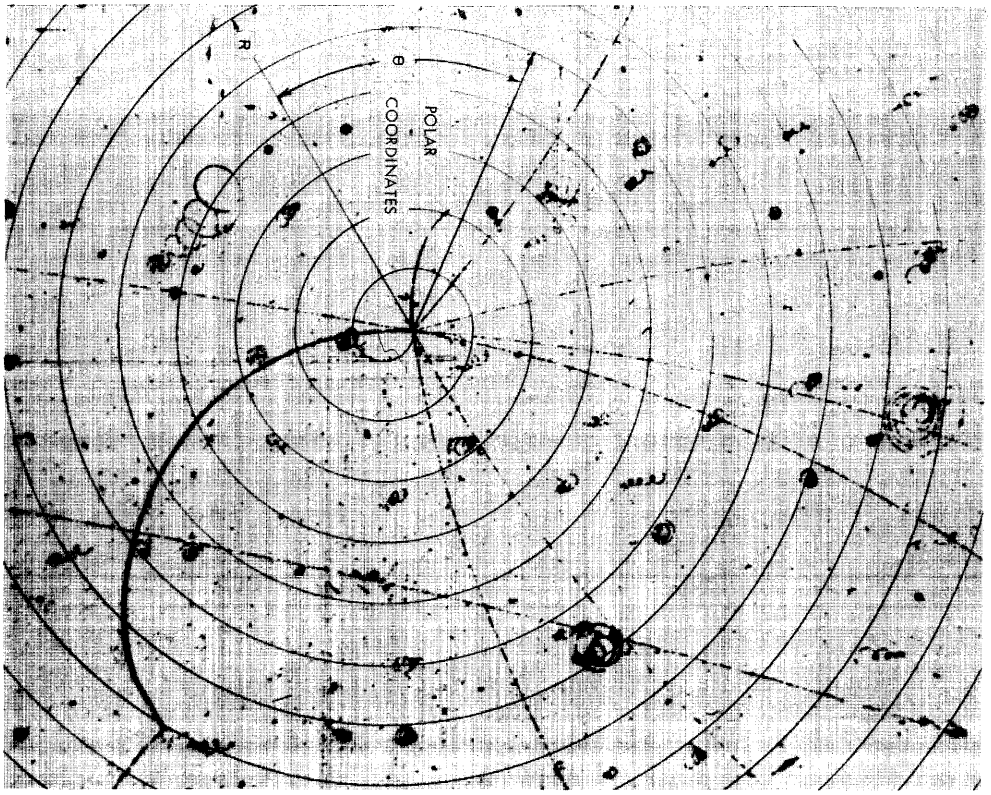


Figure 5 Spiral Scan Path

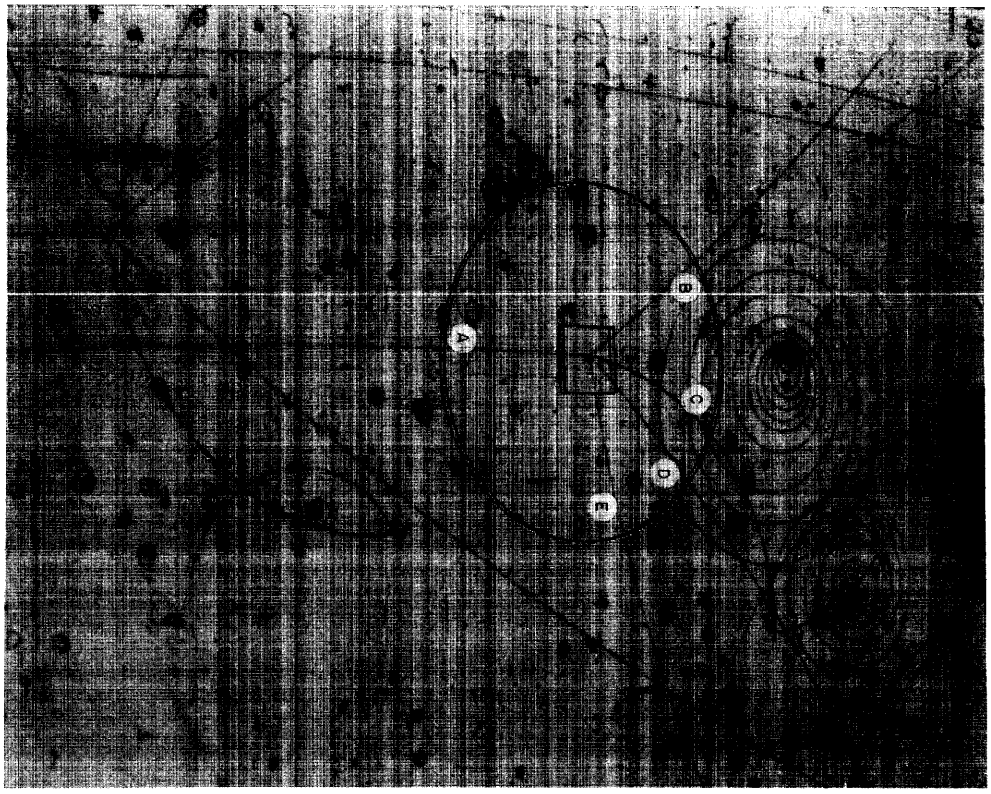


Figure 6 View of an Event As Seen by Operator

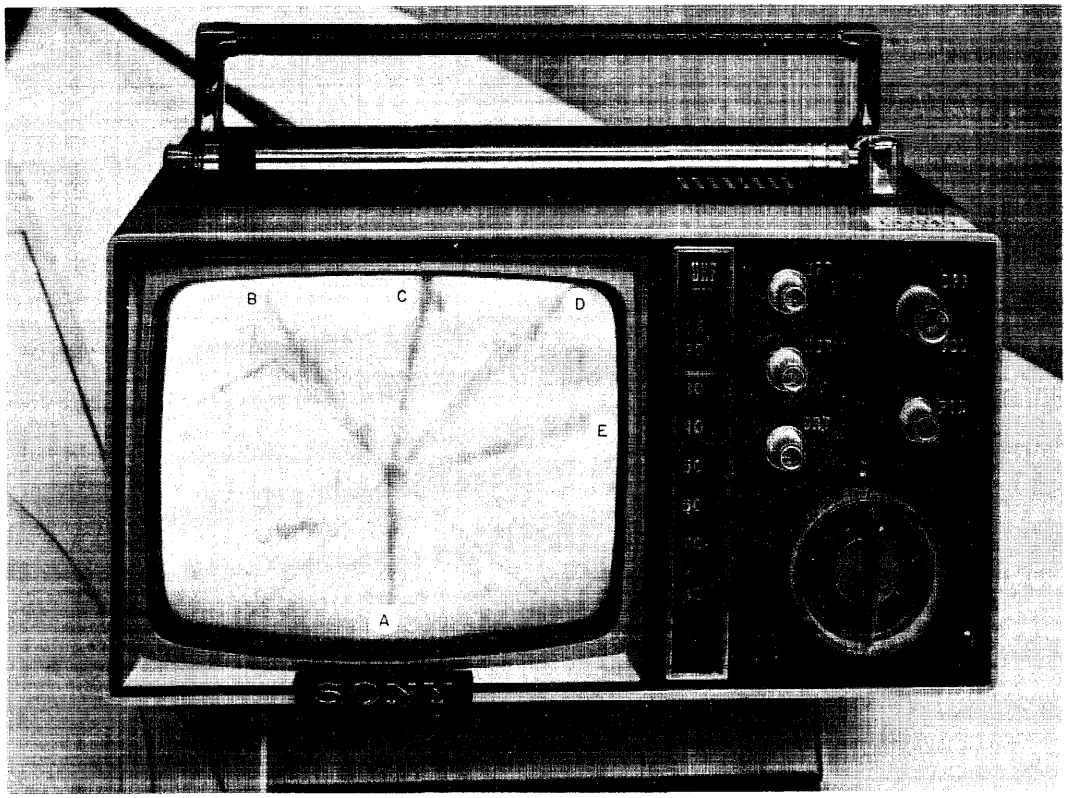


Figure 7 TV Monitor with Crosshair Centered on a Vertex

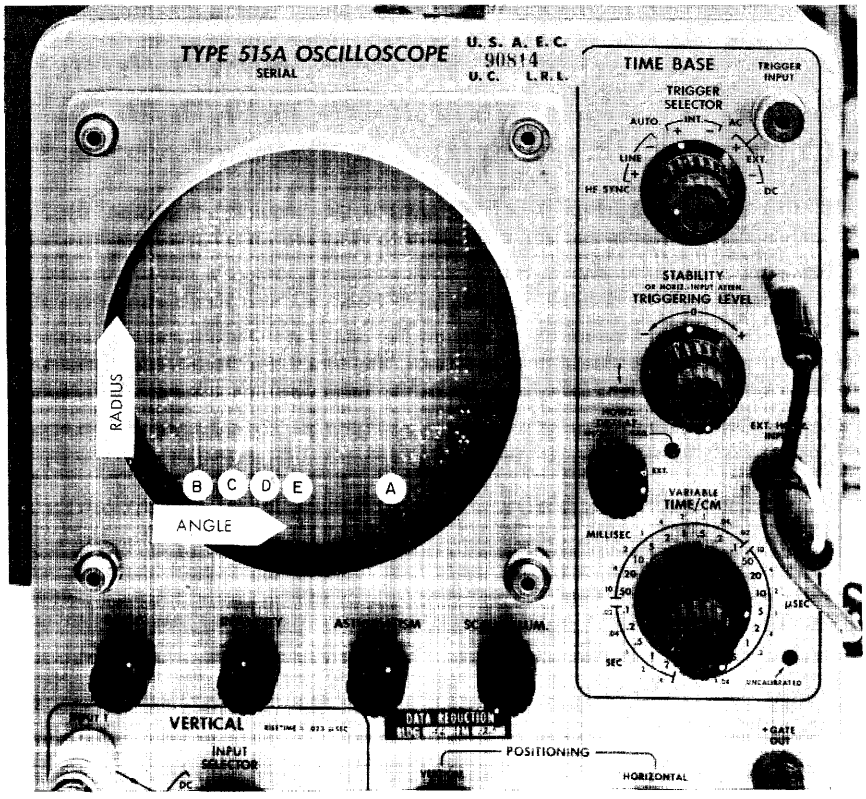


Figure 8 Spiral Scan Data Display

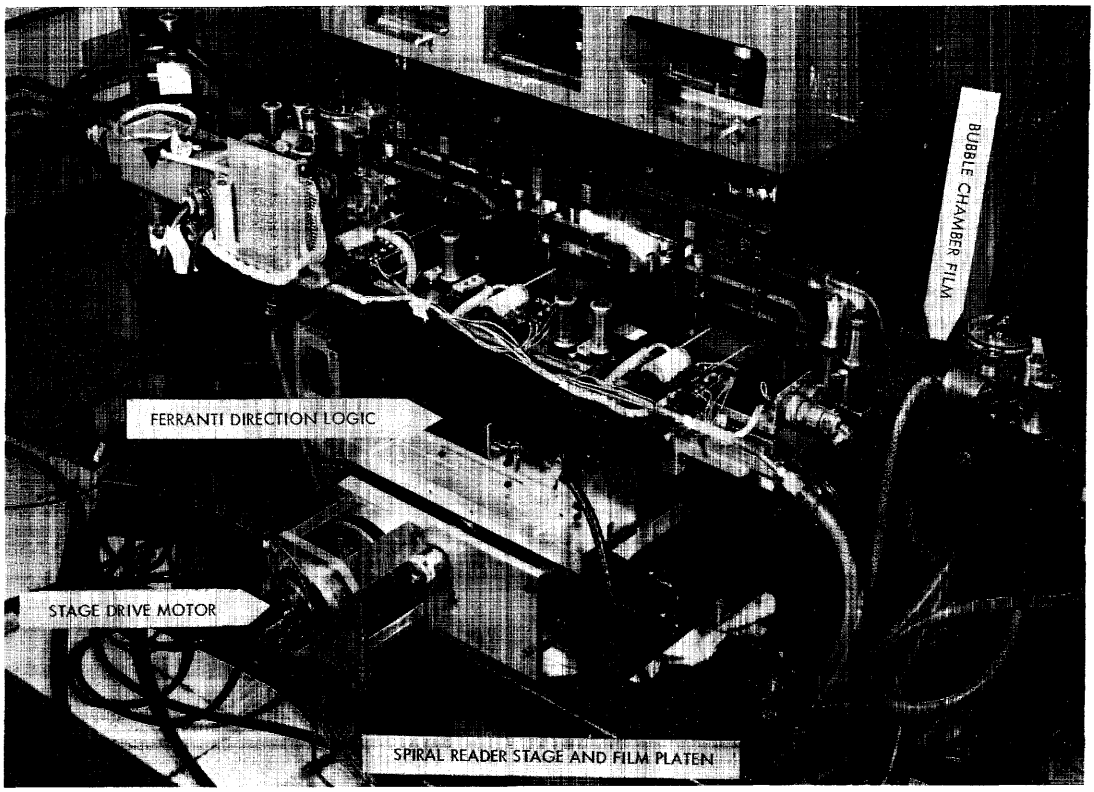


Figure 9 Spiral Reader Stage and Film Platen

STAGE MOTION CONTROL WITH PDP-4

A detailed description of the stage motion device follows, because it is typical of all the devices, and the human operation of this device through the use of the manual SPEED BALL illustrates the importance of man-machine interaction.

The X and Y motions of the stage are two independent devices but are controlled jointly as a single logical operation. The Y stage is mounted on a ground support and moves in one direction. The X stage is mounted upon the Y stage and moves perpendicular to the Y motion. Figure 11 shows only the X component for simplicity.

A DC motor drives the X stage. The motor signal is derived from either a digital-to-analog converter in computer control mode or from a signal converted from an integrator wheel on the SPEED BALL in the manual control mode. Either the operator or the computer may switch the control mode. Mechanical stops limit the X stage motion at both ends and micro-switches connected to the X stage status register electronically sense this motion. An up-down scaler that is fed counting pulses from the Ferranti direction logic holds the current position of the X stage. The lower part of the X status register holds the digital velocity loaded by the control program. The arrows interconnecting the four registers indicate the simple register transfers effected by the corresponding computer instructions. With these few instructions, a computer program can control the positioning of the stage to any predetermined point.

A flow chart of the stage motion control subroutine POST shows the simple logic required to position the stage to the coordinates specified by the parameters XPOP, YPOP. After setting the appropriate control switch to automatic, and reading the current position of the stage, the program computes the remaining distance to its destination, loads the computed velocity into the stage status register, and monitors continuously the current position versus the destination and reduces the velocity appropriately as the remaining distance becomes smaller. In this way the subroutine loops until the stage is within a given tolerance of the desired position. The effectiveness of this system greatly enhances the speed of operation by performing automatically the gross positioning of the stage and at the same time providing visual guidance to the operator.

The speed ball gives the operator positive finger tip control for fine positioning of a vertex under the crosshair. By giving the ball a hard push to start it spinning, the operator can direct the stage to more distant points. For each quantum of rotation that the ball spins along the X integrator wheel, a quantum

of charge is added to the capacitor memory. For each quantum of motion that the X stage moves, a quantum of charge is subtracted from the capacitor. Hence, the total charge on the capacitor produces a voltage whose magnitude and sign provide a dc signal to control the X stage motor.

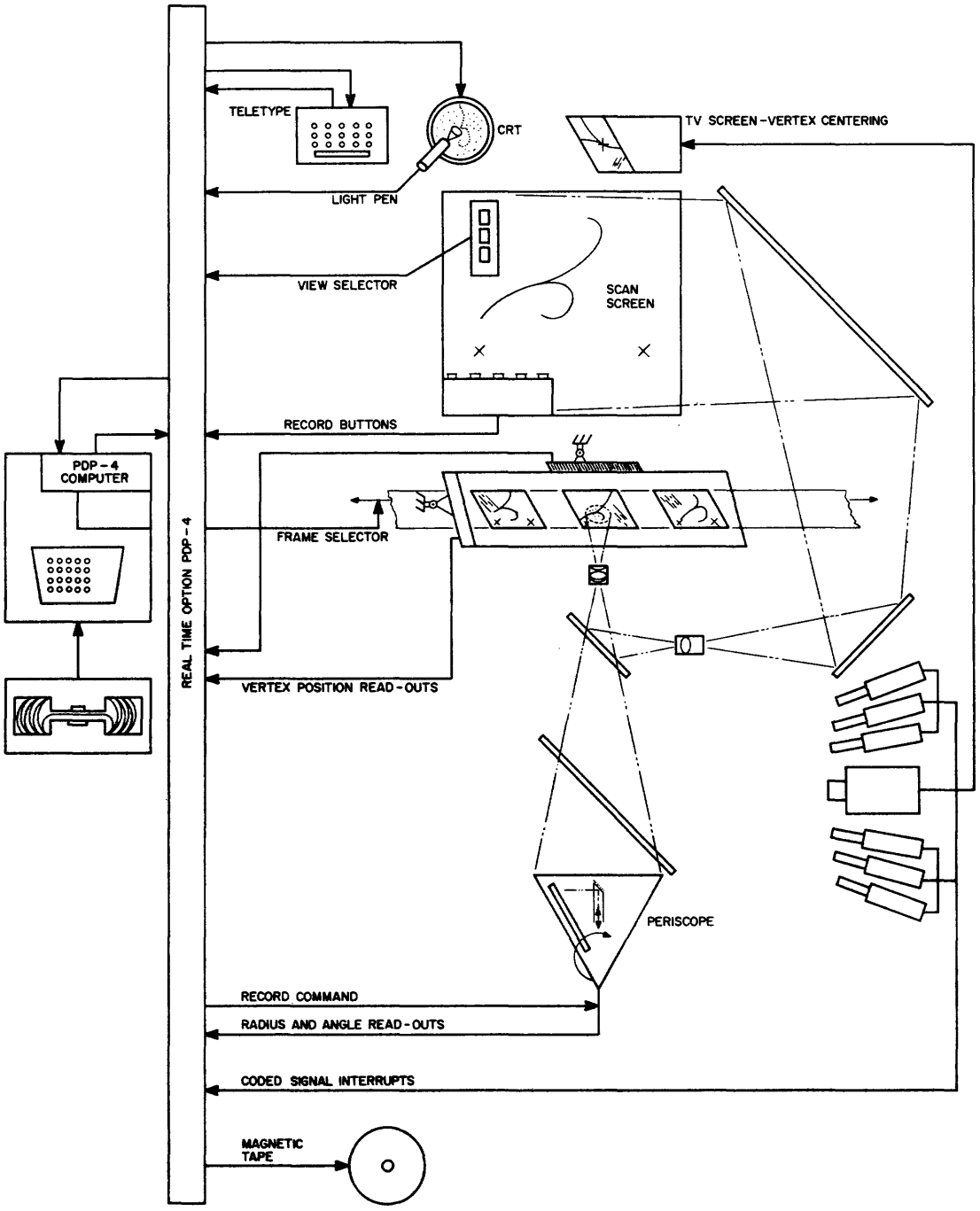


Figure 10 Information and Control Sequence Measuring Projector

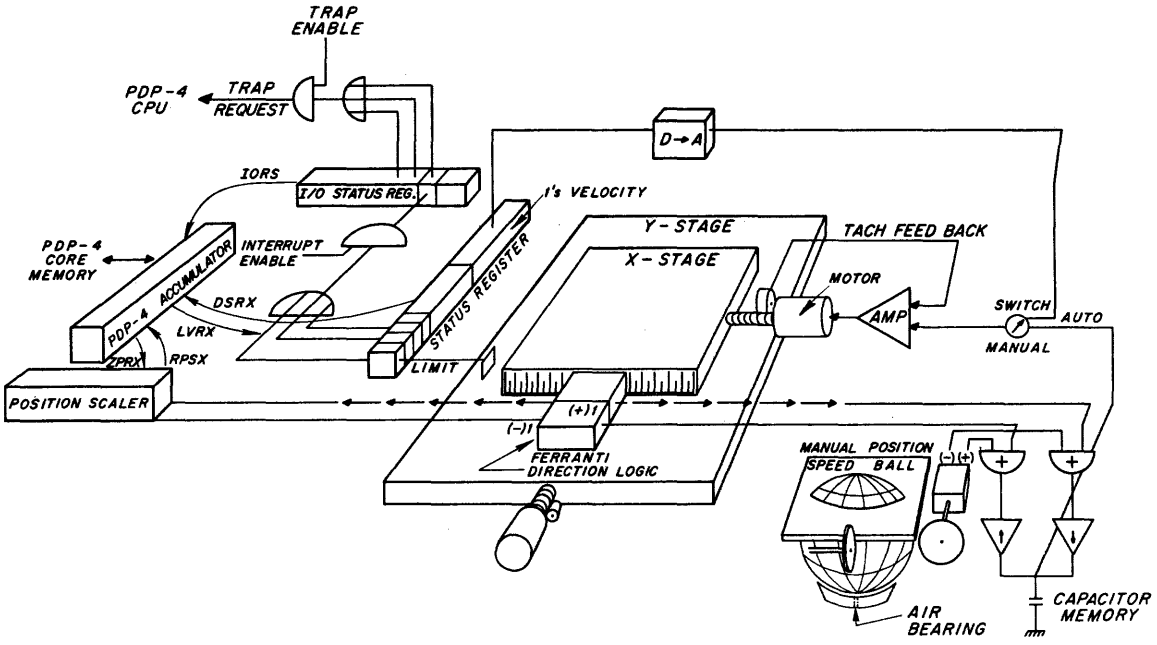


Figure 11 Spiral Reader Stage Device Schematic

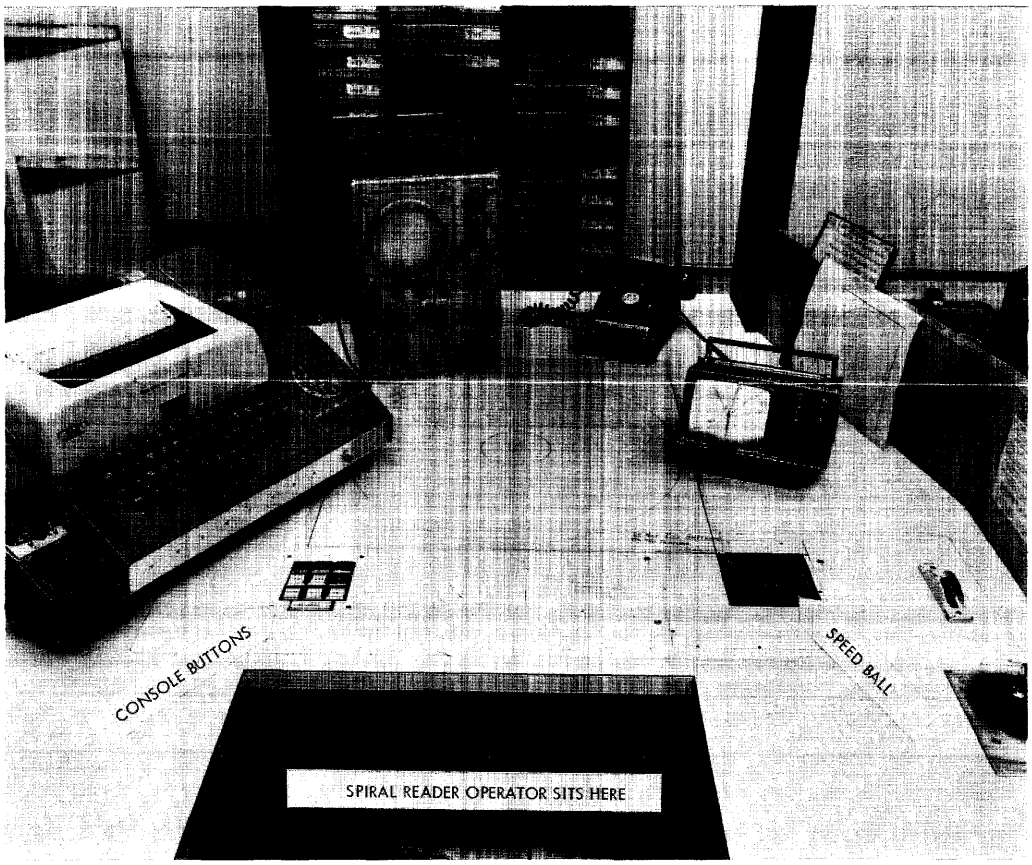


Figure 12 Spiral Reader Controls

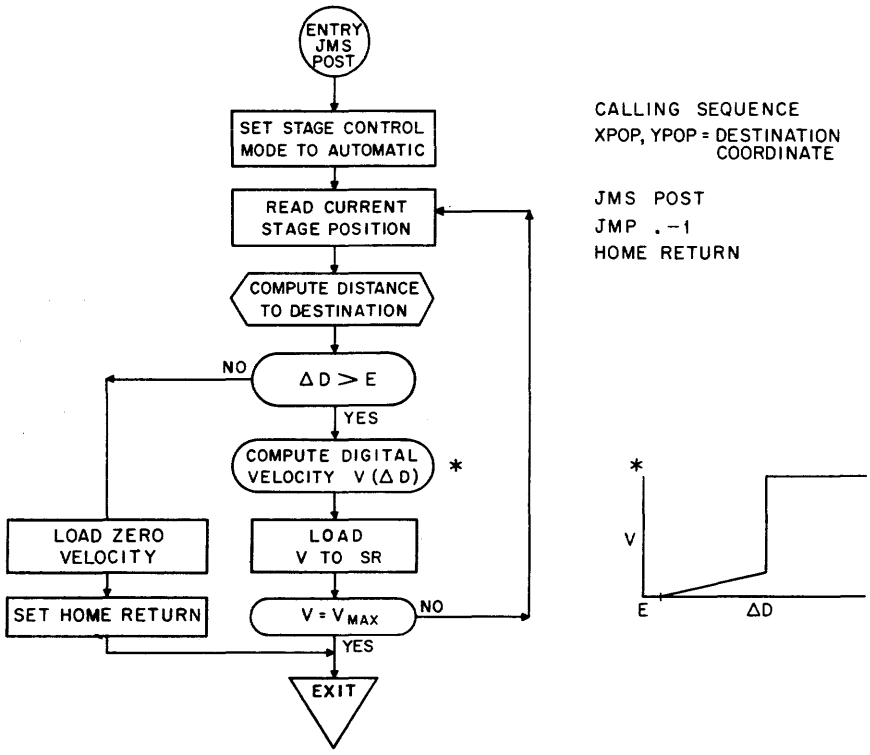


Figure 13 Stage Position Subroutine Flow Chart

MEASUREMENT CONTROL PROGRAM

Figure 14 is a simplified flow chart of the general operating control program ACORN. The steps shown are those encountered while measuring single vertex events. Observe that the process is mostly serial and hence effective overlapping of functions is necessary in only a few places. This happy circumstance simplifies the trap supervisor and avoids the need for elegant parallel operations.

After the control program ACORN has been loaded, the Teletype enters such initial constants as the date, operator number, roll number, and current frame position. The computer reads the measurement control tape to obtain the indicative information for the next event that consists of frame number and the rough coordinates of the event as well as the event type. While the program is moving the film to the specified frame, it types out the indicative data to the operator. Unless otherwise notified by control button, the computer assumes that the frame is positioned correctly and proceeds to scan for the auto-fiducial marks.

Auto-fiducial marks are scanned by moving the stage so that the large fiducial marks pass under photo-detecting slits that are slanted so that they are most sensitive to these fiducial marks. Interrupts from these photo-detecting slits cause the program to store the current stage position. From this data, the program attempts to isolate the location of the fiducial marks by testing the data for correct fiducial separations. If the computed fiducial does not pass the consistency checks, the program positions the fiducial in question under the crosshair and requests the operator to center accurately with the speed ball.

If the film quality is poor, the auto-fiducial sequence is not very successful. The control program can be instructed to delete this procedure from the sequence and revert to positioning the fiducials for the manual measurement. Since the frame positioning is approximately the same from one frame to the next, the program remembers the previous location of the manual fiducials and thereby aids the operator in the rough positioning of the fiducial marks. Measuring the fiducial marks consumes about 1/3 of the total measuring time and hence has been an important area for automated techniques. Auto-fiducials are measured in 7 seconds compared with 15 seconds required by the operator.

After measuring the six fiducial marks, the computer drives the stage to the approximate position of the event to be measured, thus visually instructing the operator to center the vertex under the crosshair.

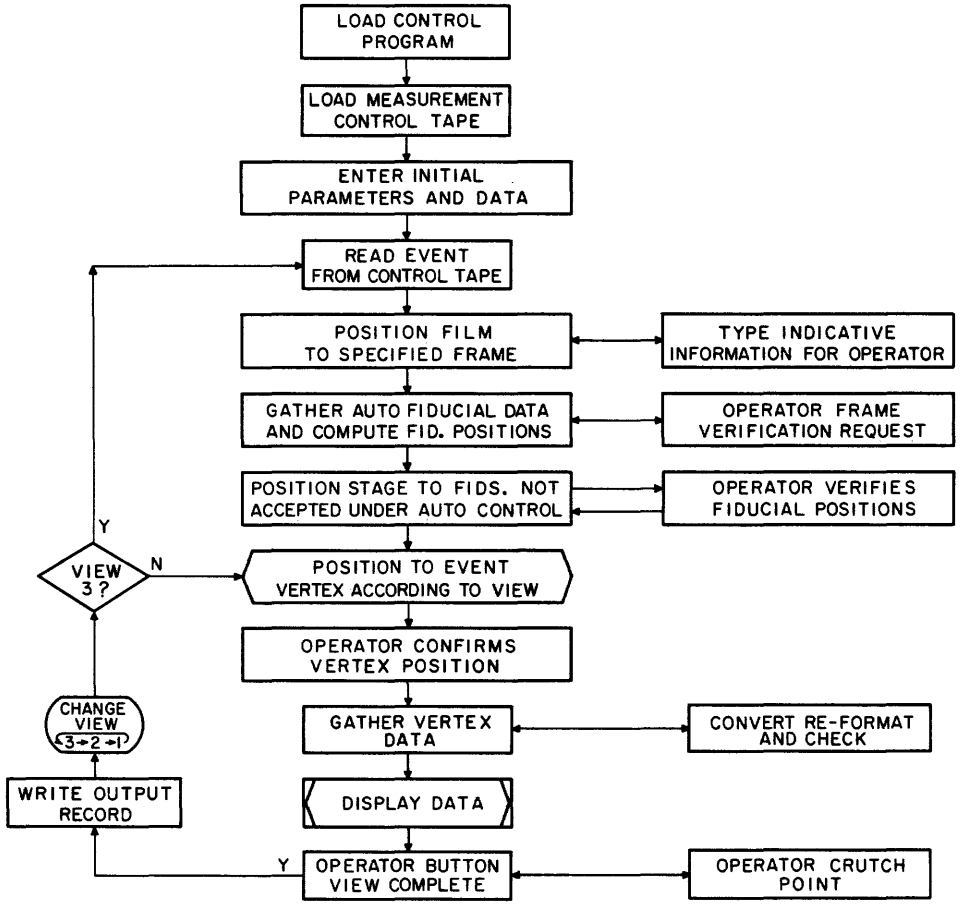


Figure 14 Normal Program Sequence for Single Vertex Events

The operator presses the measure control button to confirm that the event has been centered. The computer then reads the position of the vertex, enables the data-break channel to accept data from the pulse height detector logic, and instructs the periscope to commence its spiral scan. The computer monitors the progress of the data input and performs reformatting, checking, and conversion from gray code to binary as the data is being collected. If the computer detects any error conditions at this time, such as pulse height drift, angle scaler check, or insufficient data rate, it types a note to that effect to the operator and usually the program recycles to make another scan on the vertex.

If the operators deems crutch points necessary, he moves the stage by the speed ball to the end point of short tracks and presses the CRUTCH POINT button. The program records these single points and appends this data to the spiral scan data. During this period in which crutch points may be taken, a scope displays the spiral scan data so that the operator can see the quality and quantity of the vertex digitization. (This provides another powerful diagnostic check on the electronics.) The vertex data and crutch points are written onto magnetic tape whenever the operator pushes the ADVANCE button.

The ADVANCE button also signals the completion of that view and the computer proceeds to change to the next view and position and crosshair near the vertex of the current event in that view. This measuring cycle continues through all three views and is complete on the third view. The computer then reads the next event information from the measurement control tape and starts to advance the film to the desired next frame by counting each frame as it passes.

The timing breakdown for the SR measuring sequence is as follows:

15 seconds for fiducial measurements
15 seconds for vertex measurements
15 seconds for the film positioning, indicative data, and miscellaneous.

Our operators have been measuring with the SR at an average rate of 75 events per hour.

The cost of building another SR system is approximately $200,000 which includes the PDP-4 computer. Comparing these figures with the rough cost and performance figures of the traditional Frankenstein system, we conclude that one operator with one spiral reader can measure at a rate equivalent to ten operators with ten Frankensteins.

MAINTENANCE AND CHECK-OUT WITH PDP-4

Since the addition of the PDP-4 computer to the spiral reader system, the maintenance problem has steadily improved. There are several reasons for this. First, the redesign of the various electronic control devices has lead to mass replacement of old transistors. Second, the modular nature of the design which relies on the PDP-4 as a controller has reduced the amount of hardware logic. A significant result of these inovations is that our electronic technicians have learned to use the computer as a debugging tool. Using some standard test programs and applying their programming skills with DDT, they can isolate most malfunctions to a single circuit board without conducting signal tracing.

For some years B.C.*, there was no effective way of checking the reliability of the data produced by the spiral reader's hybrid pulse-discrimination circuitry. Some important systematic errors were discovered when the data was checked with the aid of the computer and displayed for visual inspection. For example, it was discovered that hot air currents from the projection lamps were passing around the projection lens and causing subtle image distortions. This effect was first seen on the display scope when the track data near the vertex was magnified with the on-line control of an elegant display program. It required the on-line capability to uncover the defect and the genius of a man to recognize the cause !

CONCLUSIONS

It must follow from what I have said so far that if one wants to build a better mouse trap, one had better automate it with an on-line computer. This has certainly been the case for the spiral reader. This balanced combination of man and machine is the fastest overall measuring system in operation and has proved to be an excellent impedance match with the other data-analysis procedures encountered in bubble chamber physics. Considering the low cost of this system, other universities, wanting to break into the big time by discovering "new particles," might do well to study this scheme.

CREDITS

Although it is difficult to give individuals their just acclaim for contributions to a group effort, the following people have contributed to the success of the spiral reader.

Physics	Electronics
Dr. Luis Alvarez	Jerry Butler
Jack Lloyd	Joseph Strople
Dr. Jerry Lynch	Thomas Taussig
Dr. Frank Solmitz	
Programming	**Mechanics and Optics**
James Baldridge	Norman Andersen
Jim Burkhard	Robert Smits
Frank Hodgson	
Jon Stedman	

Finally, the staff of technicians who operate the spiral reader must be congratulated for their patience, efficiency, and high processing rates.

*Before Computer

PARAMET, A PROGRAM FOR THE VISUAL INVESTIGATION OF PARAMETRIC EQUATIONS*

Kenneth R. Bertran

Lawrence Radiation Laboratory
Livermore, California

ABSTRACT

Paramet is an operator oriented program which accepts parametric equations from a typewriter and produces a graphical presentation on a CRT. Equations are of the form $x = f_1(t)$, $y = f_2(t)$; where $f_i(t)$ is any combination of the operations available (i.e., addition, subtraction, multiplication, division, exponentiation, logarithm, sine, and cosine). After the equation is initially entered from the typewriter, control is transferred to the light pen. With the light pen the operator can change the values of the various constants which have been used in the equations. He can also control operations such as scaling, tape storage, and retrieval.

Paramet provides a convenient method of finding first approximations in curve fitting. The investigation of particular equations is readily pursued. As an educational tool, Paramet provides the student with a feeling for his subject more effectively than conventional methods.

INTRODUCTION

The Livermore site of the Lawrence Radiation Laboratory maintains a PDP-1 computing facility for three primary functions:

1. To digitize graphical data.
2. Card-to-tape conversion for the LARC.
3. To test new equipment and ideas to be utilized later on a larger computer.

Investigations are proceeding on on-line man-machine interactions using visual communication.

PARAMET is a program which permits visual observation and control via the light pen. The study being made allows a user to input equations in two or three parametric equations. Initial conditions are also matters of input. The choice of parametric forms is primarily influenced by current engineering applications. A Type 30 CRT Display shows the progress of calculation. Using a display has two advantages:

1. Knowing how the chosen class of equations evolve is often more important than the ultimate result they present.
2. Since the actual calculation takes several minutes in many cases, the ability to observe proceedings adds in maintaining user-interest.

The program development is influenced by two major attitudes; the necessity for simple language, geared as much as possible toward engineering notation; and the limiting of program and data to 4K words of core.

The first consideration is the choice of a method for presenting the given equations to the computer. Examination of the I/O available (see Figure 1) affords three possible means of input to consider with the above requirements in mind: the Rand Tablet, the light pen, and the typewriter. The Rand Tablet is perhaps the most attractive, but the programming problems involved force us to consider a less ambitious course. The typewriter is a good choice from a programming viewpoint because there is a minimum of translation. The Rand Tablet of course would lead us into the complexities of character recognition. While this has been done by several others,[1] it is felt that delays are possible in connecting a character recognition program to the other programs and keeping the entire code within a 4K word limit. The typewriter presents its characters in a convenient form compatible to input speed, but the user, in choice of the typewriter, must accept a more involved notation because of the restrictions of the typewriter character set. The particular format chosen and the reasons for the choices of notation are given in the section on operating instructions.

*Work performed under the auspices of the U.S. Atomic Energy Commission.

[1] Teitleman, W.; Marrill, T.; B.B. and N; Shaw, J.C., Rand Corporation; Cralle, R.K.; Michael, G.A., Lawrence Radiation Laboratory.

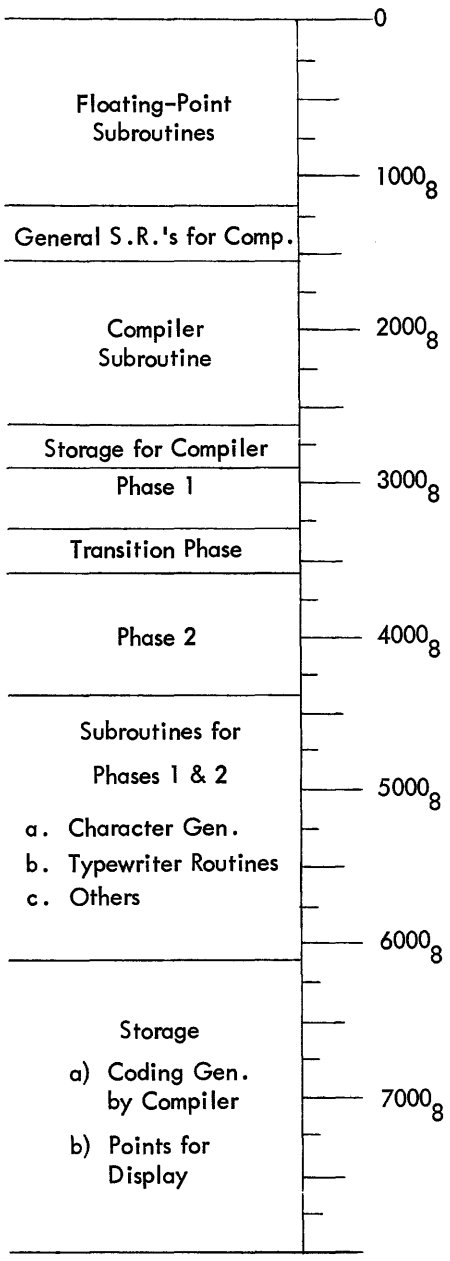


Figure 1 The Distribution of PARAMET in Memory

One then has a means of entering equations into the machine in a form not too unlike the way an engineer usually indicates them. The compiler operates on the input string and produces the coding necessary to generate and display a graph of the function. There is also the need of assigning numbers to any constants that are given, of scaling, and since we are dealing with parametric equations, the range of the independent variable. Since the machine has no way of interpreting what would be of interest, the user must specify these values himself. The calculations however, because of the limited nature of the machine, take an appreciable time to complete. It is much more effective to present the data as it is generated rather than to collect the total set of points before producing a display. A user with nothing to do for the two minutes necessary to complete a calculation tires more quickly than one who watches the graph unfold as the calculations proceed. Since the final graph appears almost motionless, the order of calculation is lost; so, as already noted, the user can gain extra information by watching the display.

Upon completion of the display, attention is focused on the CRT, and at this point the light pen is the best choice for communicating with the computer. The method chosen to effect the desired communication is explained later in this paper. If in fact the light pen could input the information, a better continuity would result. But all the arguments against the Rand Tablet apply to the light pen also. The next version of the code will allow full re-examination of the use of pens.

We consider now the method adopted to change numbers which are involved in the display. The number is selected for modification by merely pointing to it. The technique used to modify the value is somewhat synthetic compared with this rather natural method of selection; i.e., changing the number involves the use of a "gas pedal" effect. A line displayed on the CRT represents this gas pedal. As the light pen is moved to the right, the value of the selected number increases, and the farther the movement to the right the greater the rate of increase. The left half of the line produces a similar effect but decreases the value selected. In practice the gas pedal has proven to be less effective than expected, due to the inability of the user to respond to the indicated changes, because of the irrelevance of these changes to the property in which he is really interested. The changes are shown first as the current decimal value of the chosen parameter. An immediate alteration of the graphical output makes the gas pedal a more effective device, suggesting that the more immediate the response, the more efficient the communication becomes.

One notes that there is very little flexibility allowed in PARAMET operation, as only input or data may be of a variable nature. This inflexibility is a handicap for more experienced users who already are familiar with a FORTRAN approach to computer use. It must be remembered that this program is designed to make available the unique capabilities of a digital computer to a user with little or no experience with such devices. By reducing the flexibility of the program, the user is required to make a minimum of decisions. Those decisions which he must make concern the nature of the equations which are under consideration and not the operation of an unfamiliar device. A more general and flexible version is being developed allowing for the needs of users as they become more familiar with digital computer abilities.

We now have a useful tool for the investigation of parametric equations whose principal feature is that it attempts to combine the maximum aspects of the user and the machine. Some of the jobs done by the operator can be

done by the machine and conversely, but the division of tasks is best accomplished by allowing either one to perform the task best suited to his or its capabilities; in other words, operator intervention should not necessarily be avoided. Close control by the operator very often produces an efficiency which could not be realized by machines alone.

DESCRIPTION

Paramet is an experimental program for the Lawrence Radiation Laboratory's PDP-1 computer designed to aid the study of certain classes of parametric equations. In particular it provides a continuous display of an associated graph which represents the equations and permits user-machine communication. The program also allows for storage of information on magnetic tape so that sets of results may be cross-compared. There are two primary modes of operation, two-dimensional, and three-dimensional. In the two-dimensional mode, mode 1, x and y are expressed parametrically in terms of the independent variable, t. The three-dimensional mode, mode 2, has the additional variable, z, also expressed as a function of t.

The parametric equations are supplied by the operator via the console typewriter. The range of the independent variable, t, and the scaling for the CRT are also specified on the typewriter. After the initial entries have been supplied on the typewriter, a display based on these values is produced and control is passed to the light pen. With the light pen the operator may vary the constants in the equations and the scaling of the CRT, and also perform various tape control operations.

OPERATION

The following I/O equipment is required:

1. Typewriter
2. Visual and precision CRT's
3. Light pen
4. IBM tape unit 3

The program is entered into memory either as binary cards or from the system tape (the ID is PAR). The starting address is $10)_8$.

Phase 1, Typewriter Control

The typewriter supplies the beginning of each line indicating what information is required for that line. In the list below, parentheses are used to mark off input information which the operator must supply. The phrase enclosed by the parentheses tells the kind of input required. The sequence of typeouts is as follows:

x = (An arithmetic statement and then a carriage return)

y = (An arithmetic statement and then a carriage return)

* z = (An arithmetic statement and then a carriage return)

|x| < (A number)

|y| < (A number)

* |z| < (A number)

t > (A number)

t < (A number)

** (Operator inserts comment, if any, and then a carriage return)

a = (A number)

b = (A number)

etc.

The two classes of statements to the typeouts are: arithmetic statements and number statements.

1. Number statements may not exceed five digits with a decimal point optional. If a negative is desired, a minus sign may preceed the number.

2. Arithmetic statements are similar to the conventional notation used in mathematics texts. They are used to specify the relationship between the dependent variable and the independent variable, t. An unassigned variable may be used in place of any constant which appears in the equation. The unassigned variable can be any letter which is not a function or does not have an assigned value. These restrictions eliminate the assigned variables x, y, z, t, e (the base of natural log) and p (Pi) and also u, v, w which are arbitrarily reserved.

Addition and subtraction are indicated in the usual way by plus and minus signs. Division is indicated by a slash in the same manner as seen in texts where it is desired to write an equation within a single line, for example, 2-3/4-1. Exponentiation is a case where the conventional notation requires that the exponent appear displaced from the line. In order to keep within the line, the upward arrow is used to indicate the displacement of the exponent. This symbol is used in the same manner in which the slash is used to indicate division. As an example, $y = x^{2-1}$ would be written as y=xt(2-1). Multiplication is indicated by juxtaposition as in the usual notation or by the use of parentheses. In order to indicate the operation of x multiplied by three, 3x or (3) (x) is used.

In addition to the arithmetic operations there are also three functions available: sine, cosine, and logarithm.

---

*Indicates mode 2 only.

**The comment will be displayed along with the associated curve.

To indicate a function the user types the usual three letter symbol (sin, cos, or log) followed by a left parenthesis. The argument is then enclosed with a right parenthesis thus sin(at-b).

Errors in typing may be corrected by using the backspace to return to the point at which the error was made and proceeding from there.

The termination of a line is signaled by a carriage return. When all the information listed in the first part of the table above has been supplied, the typewriter will wait for any comment the operator may wish to have displayed on the CRT as an annotation of the display. To allow more than one line of typing in the comment, the comment will not be terminated until two consecutive carriage returns are given. When the comment is terminated, the typewriter will request that the operator supply the initial values for all the unassigned variables used in the parametric equations. These initial values must be in the number format. When all the initial values have been given, the set of information required for computation is complete and computation begins.

Phase 2, Light Pen Control

The curve, or stereo pair if using mode 2, now appears on the CRT with the comment and one light pen sensitive point located at the bottom right hand portion of the visual CRT. When this point is activated, the control functions are added to the display.

The control functions consist of three elements. The first element is a single point located slightly to the right of the point used to obtain the control function display. Activating this point causes the control display to be deleted. The second element consists of a list of unassigned variables and their values. Associated with this display is a line which may be used to alter the value of any of those variables displayed. The light pen is used to select the letter variable whose value is to be modified. The pen is then moved over the adjustment line. As the pen is moved to the right along this line, the value assigned to the variable selected is increased. The farther the pen is moved along this line, the faster the rate of increase. To decrease the value the pen is moved along the left portion of the line in a like manner. When the desired value has been reached, a new variable may be selected. The process may be repeated until all the variables have the desired values. Note that only those unassigned variables which were used in the parametric equations will appear in the table.

The third control element is the control word, S C R A B U M, each letter of which activates a particular operation.

- S Store the existing function and display.
- C Compute the function using the modified values.
- R Restart at phase 1.
- A Advance to the next stored function and display.
- B Backspace to the previous stored function and display.
- U Unwind tape 3.
- M Modify the scaling.

Activating M causes a new display to be presented which gives the present values of the scaling and allows them to be modified in the same manner as unassigned variables are changed. A single point at the bottom right hand portion of the screen is also displayed and when activated causes a return to the previous display. The letter 'C' of the control word is now activated if it is desired to compute using the modified values.

Sense Switches

Two sense switches are utilized by the program.

Sense Switch 1

Up - Mode 2 (stereo)

Down - Mode 1

Sense Switch 6

Up - Terminates the calculations immediately and displays the partial results.

FACILITIES

The Lawrence Radiation Laboratory PDP-1 has the following computing facilities at hand:

a. A standard PDP-1 computer with 4096 words of 18 bits each.

b. Six tape units; two IBM units; four Potter units.

c. Two CRT display systems; DEC Types 30 and 31.

d. Light pen for the Type 30 Display.

e. Thousand line per minute printer.

f. IBM card reader.

g. Calcomp digital XY plotter.

h. Console typewriter.

i. Rand Tablet stylus.

j. Paper-tape reader and punch.

k. An 11-bit A-D converter.

l. Telephone handset for voice input.

m. Multichannel amplifiers that are switchable for audio output.

The Compiler

The compiler generates coding to perform the operations indicated by an arithmetic statement. The main program

supplies the compiler with the necessary information about the statement and transfers control to the compiler. In order to keep problems of storage allotment to a minimum, arithmetic statements are compiled as subroutines. The compiler makes two passes over the statement. On the first pass all operations in the order in which they occur are listed in a table. On the second a similar table is constructed for all the operands.

The operations are encoded with one operation code per word in the following manner:

Operation	Code	
Addition	0	000
Subtraction	1	001
Multiplication	2	010
Division	3	011
Exponentiation	4	100
Logarithm	5	101
Sine	6	110
Cosine	7	111

Since only one operation is stored per word (not overly wasteful considering that there are not likely to be very many operations in a single statement), only the last six bits are necessary to specify the operation and the most significant bits may be used to indicate alterations in hierarchy. That is multiples of 10_8 may be added to the coded instruction to indicate deviations from the usual hierarchy. Each time a left parenthesis, "(", is encountered as the compiler scans the statement from left to right for operations, a unit of 10_8 is added to an indicator. For each right parenthesis, ")", 10_8 is subtracted from this indicator. As each operation is encoded, the value of the indicator is added to the value of the operation. With completion of the table of operations the compiler has all the information it needs to assign storage.

The compiler sets up the entrance and exit instructions for the subroutine and begins a second pass where the constants and variables are treated. A variable is treated by identifying the value of the address it represents and then storing the address in the next position in the address table which corresponds to the operation table. For constants, the number is evaluated and the value stored in the next available location after the subroutine which is being generated. The address of this location is then added to the table of operands. With the completion of the table all the information to the right of the equal sign has been extracted from the statement and the linking of operations and addresses begins.

The operation table is scanned in order until a maximum is found. (A maximum is defined as any value which is, after a single right shift, as large as or larger than either of its immediate neighbors.) The last six bits of the operator are extracted to determine the operation. If the operation is arithmetic, the corresponding two addresses are extracted from the operand table. If, however, the operation is a function, only one address is necessary. In either case the appropriate instructions are generated. In writing these instructions it is necessary to select an address for storing the result of the operation. The address of the next available space following the subroutine which is being generated is selected. The value of the address replaces the value of the addresses (or address in the case of functions) in the operand table and the table adjusted accordingly. The operator involved is deleted and the operator table adjusted. The operator table is now shorter by one operation, and the operand table may also be shorter by one. The process is repeated until there are no operations in the operations table. All the coding to perform the operations indicated on the right side of the equal sign has now been generated. The variable to the left of the equal sign is then identified and the subroutine completed. The compiler then transfers control back to the main program.

No particular attempt has been made to evaluate compilation time. During the typing-input pass however, compilation of a reasonably complicated equation is completed in the time required for a typewriter carriage return.

ACKNOWLEDGMENTS

The author's thanks to George Michael, whose encouragement and suggestions led to the development of Phase 2 (light-pen control) and the documentation of this work; Ervie Ferris, for his help in answering questions concerning the idiosyncrasies of the various pieces of I/O for the PDP-1 and for his character generating subroutine; Ray DeSaussure, who supplied the workhorse subroutines for typeouts, etc., and who served as a sounding board for new ideas; and Ed Carr, for the stereo-projection scheme used.

ON-LINE REDUCTION OF NUCLEAR PHYSICS DATA WITH THE PDP-7*

Philip R. Bevington

Department of Physics, Stanford University
Stanford, California

ABSTRACT

In the continuing debate over the relative merits of fixed-wire analyzers vs. computer systems for the acquisition of nuclear physics data, the PDP-7 has emerged as an excellent compromise between cost and capability. The advantages inherent in a computer system are illustrated with specific ways in which the Stanford Computer for the Analysis of Nuclear Structure (SCANS) is used to reduce, point-by-point, on-line data from nuclear physics experiments, as well as to analyze such data with real-time control. Specifically, programs to identify and sort charged particles with thick (E) and thin (dE/dx) solid state detectors, to stabilize the scale of pulse-height spectra, and to control the acquisition of two-parameter data are discussed. The dead time contributed by these programs is shown to be negligible. The equally important ability to control the subsequent analysis of these data in real-time is also discussed with reference to the methods adopted by SCANS to accept, record, and display the data.

In this era of proliferation of small, scientific, specialized, on-line computers, it seems unnecessary to justify the use of such a computer for nuclear physics research. But as recently as May, 1965, an article appeared in Nucleonics** which stated, "For maximum flexibility a large-memory analyzer with delayed-time write and sort capability appears to be the most suitable system. Experimental data should be accumulated and retained before data reduction is undertaken." The article then extolled the virtues of off-line analysis of taped data rather than on-line manipulation of the data before storage. It is the contention of this report that such a system bypasses the use of the scientist's most powerful tool, himself; and that, on the contrary, the system with maximum flexibility and control is one which records a minimum amount of data, with a maximum density of useful information.

The data of nuclear physics research is generally in the form of multichannel pulse-height spectra indicating the number of pulses from a detector as a function of pulse-height, which is a measure of the energy of the detected particle. (See, for example, Figures 7 and 11. The abscissa is the pulse-height or energy, and the ordinate is the number of pulses of that height from the detector.) Fixed-wire, multichannel, pulse-height analyzers have become more and more sophisticated over the years and now include the capability of storing multiparameter spectra for observing simultaneously the correlations between pulse-heights from several detectors. These devices are, however, limited in memory as well as flexibility, and even for two parameters it is necessary to sacrifice considerable detail in precision to store a matrix whose size equals the product of the dimensions of the two individual spectra.

The technique for off-line analysis advocated above therefore utilizes these analyzers only to record data point-by-point on magnetic tape for subsequent reduction by large computers, providing expanded memory space and flexibility of manipulation of the data. This technique, however, tends to insulate the physicist from his experiment, and substitute statistical analysis for his judgment.

The SCANS (Stanford Computer for the Analysis of Nuclear Structure) system was developed with the philosophy that on-line reduction of the data point-by-point would be more economical and efficient and would provide a higher degree of control and understanding of the physics involved in experiments. Figure 1 shows the

*Sponsored in part by the National Science Foundation
**T.J. Kennett and W.V. Prestwich, Nucleonics 23, 65 (May, 1965).

PDP-7 computer comprising the heart of the system, with 8192 words of 18 bits, teletypewriter, paper tape, DECtape, oscilloscope, light pen, IDIOT box, X-Y plotter, and card reader. To the right is a dual-parameter pulse-height analyzer without memory, which is connected on-line; three other satellite single parameter analyzers with memories have provisions to dump their memories into the computer. The PDP-7 was chosen for its speed and flexibility of input/output at a reasonable price: 1.75 μsec cycle time with direct access to the accumulator from external devices and 6 μsec multiply time. The computer, its input/output devices, and the associated on-line pulse-height analyzer and satellite analyzers with memories, are used in conjunction with the Stanford University 3 mev and 15 mev Van de Graaff accelerators for data acquisition, reduction, and analysis.

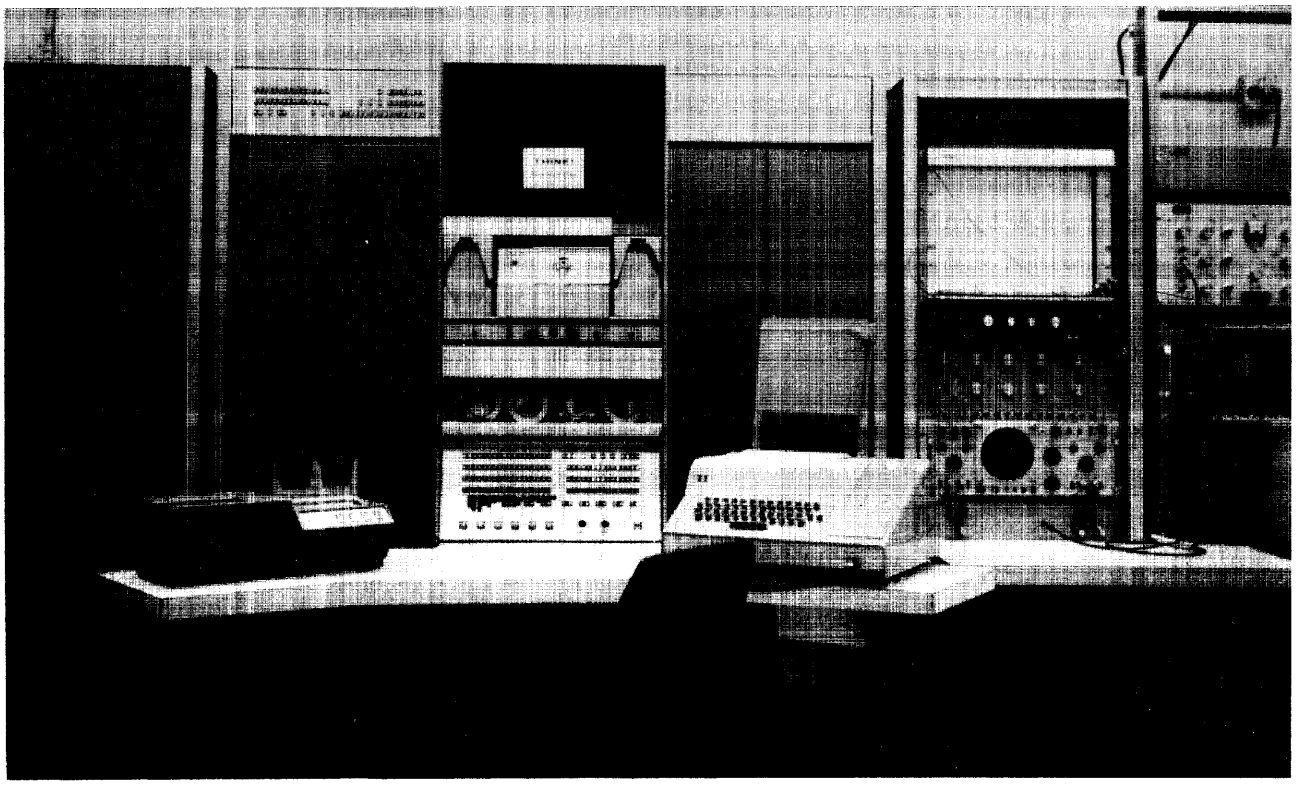


Figure 1 SCANS system control console showing (left-to-right) interface cabinet, card reader, PDP-7 computer, plotter, IDIOT box, scope, and dual parameter multichannel pulse-height analyzer

As an example of the advantages of such a computer-oriented system, consider an experiment in nuclear physics in which the reaction products may be a mixture of protons, deuterons, tritons, alpha particles, neutrons, and gamma rays, with energies up to 20 mev. A standard technique for separating the charged particles is to use a counter telescope consisting of a thick solid-state detector in which the particle stops, preceded by a thin detector in which the particle loses only a small fraction of its energy. The signal from the latter detector is a measure of the energy loss per unit distance, and hence is denoted the dE/dx signal. Analysis of the experiment requires a knowledge of the energy and identification of each particle. This information can be extracted from a combination of the signals from the two detectors.

For a given energy, the dE/dx signal is larger for heavier particles, but the magnitude of the signal is a function of the energy. Figure 2 shows a live display of the signals from the two detectors for 4 mev deuterons incident on Al on a C backing. To a first approximation, the dE/dx signal (the ordinate) is inversely proportional to the E signal (the abscissa) and there are two well defined curves; the upper contains deuteron groups and the lower the proton groups. The resolution of each of the signals is 512 channels; therefore, it is not possible to store all of this information in a 8000 word memory.

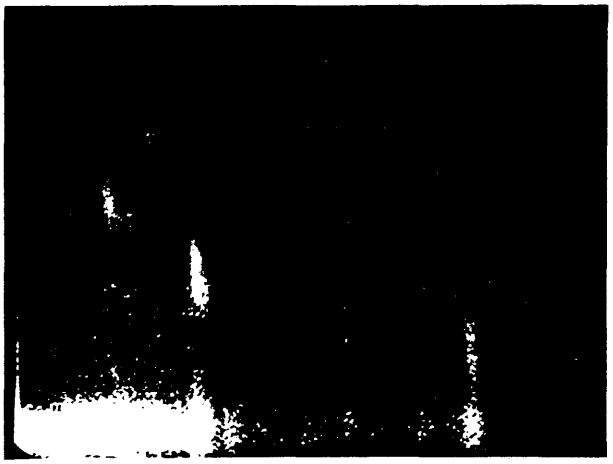


Figure 2 Live display of signals from a thin (dE/dx) detector (ordinate) and a thick (E) detector (abscissa) for charged particles

Figure 3 shows a contour display of the same data stored in a 64 by 64 matrix, with a resolution 1/8th that of the input data for each detector. The intensified channels are those with more counts than a specified baseline. In this case, the largest proton group is sliced at about 4% of peak height, the second proton group at 8%, the larger deuteron group at 16%, and the smaller deuteron group at 50%. Although the deuteron groups are well seperated from the proton groups, it is impossible to distinguish them with a simple discriminator setting, since the ordinate for the weaker deuteron group equals that of the stronger proton group.

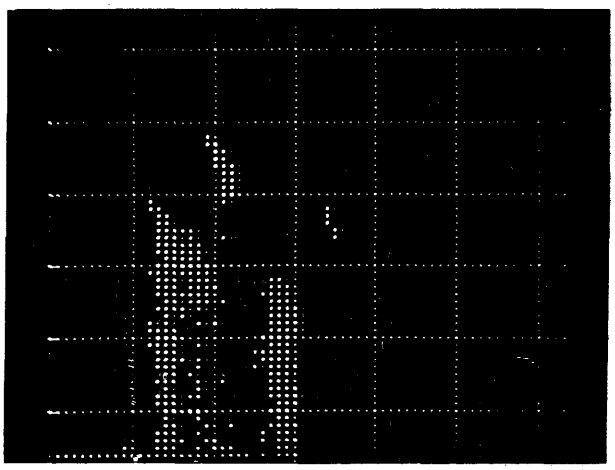


Figure 3 Contour display ot stored data ot Figure 2

Figure 4 shows an isometeric display of the same data, illustrating the difficulty of extracting information from such a display. The abscissa is the signal from the thick detector; the axis at 45° is the signal from the thin detector, and the ordinate is the number of counts in each channel.

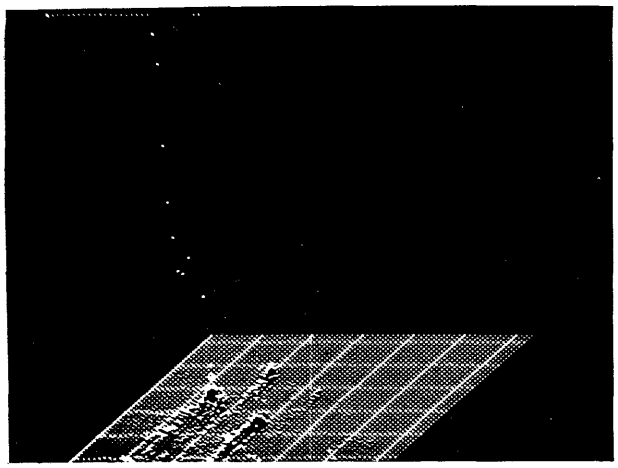


Figure 4 Isometric display of data of Figure 3

As shown in Figure 5, the product (dot-dashed line) of the energy E of the particle and dE/dx varies less than dE/dx alone (dashed line) over much of the energy range but not at low energies. By adding constants to the parameters before multiplication, it is possible to construct a function (solid line) which is fairly energy independent.

Figure 6 shows a live display of the data of Figure 5 treated in this manner. The abscissa is the total energy of the particle: the normalized sum of the signals from both detectors. The ordinate is the function described above. All of the deuteron groups have larger values of this function than the proton groups, and it is possible to discriminate between them according to the value of the function alone.

Figure 7 shows the final data for the proton and deuteron groups taken simultaneously. The top curve shows the total yield in the thick detector as a function of total energy of the particle. The middle curve shows the proton yield; the neutron and gamma ray contamination have been reduced considerably by requiring a coincidence with the dE/dx detector. The botton curve shows the data for the deuteron groups alone. The smaller peak can be found well resolved in the upper spectrum, and the larger peak (elastic scattering from C at 90° results in a loss of over 1 mev) is detectable in the upper curve in a valley between two proton groups. The difference between the cleanly resolved groups in the lower spectrum and those in the upper spectrum is justification enough for so complex a system.

By thus reducing the data point-by-point before storage, it is possible to store the spectra for several types of particles simultaneously with high resolution (e.g., with 1000 channels each) in a reasonably sized memory. There are other ways in which data reduction point-by-point may be highly desirable.

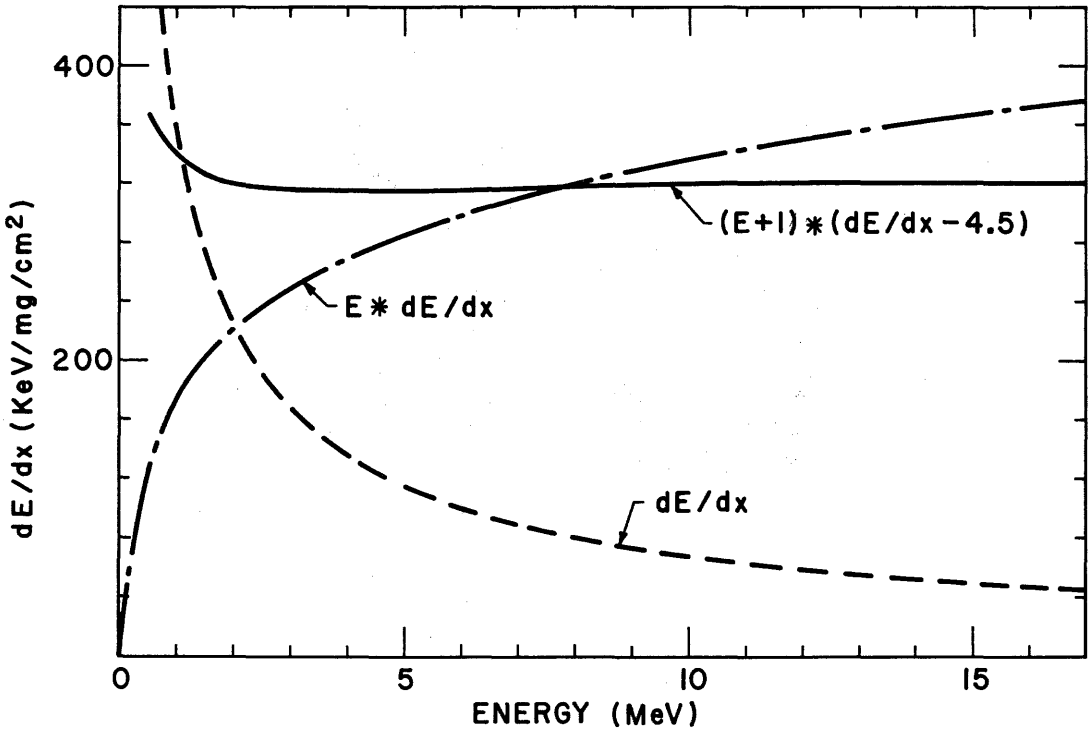


Figure 5 Variation with energy E of incoming particle of
a) energy loss dE/dx in thin detector (dashed line)
b) product of E and dE/dx (dot-dashed line)
c) function (E + 1 mev) x (dE/dx - 4.5 kev-cm^2/mg) for protons

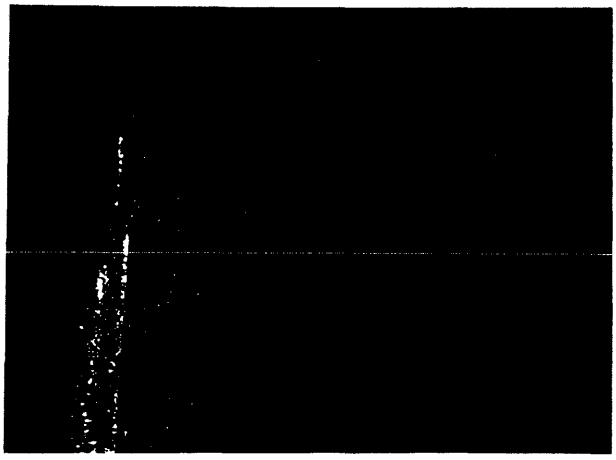


Figure 6 Live display of the function described in Figure 5

For single parameter data, for example, we are developing a spectrum stabilizer which operates internally on the data digitally to counteract changes in the amplification of the input signal. A portion of the spectrum straddling a peak is selected using the ACCUMULATOR switches and intensified for identification. The centroid of the counts falling within this window is monitored and the incoming data multiplied by the same factor by which the centroid is shifted. In case of considerable shifts, the window follows the centroid with a constant relative width. Multiplication of the data by the calculated scale factor is accompanied by a random number check of the insignificant portion of the product to circumvent the errors introduced by the round-off.

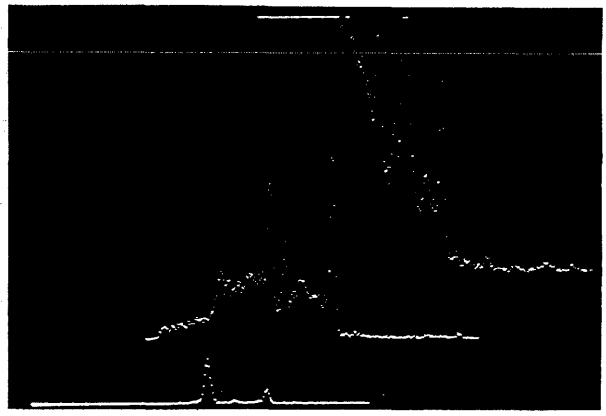


Figure 7 Stored data giving counts per channel (ordinate) vs. energy of particle (abscissa) for:
a) all particles entering thick detector (top curve)
b) protons only (middle curve)
c) deuterons only (bottom curve)

One of the objections to such immediate reduction of the data is the dead time contributed by the computer. During the analysis of each incoming pulse, the analyzer cannot accept another pulse; its input is dead. This results in a reduced counting rate, and the data must be corrected for the loss. If then, the computer must also analyze each pulse in a complex manner, how much will this increase the dead time and decrease the precision of the experiment?

Figure 8 shows the efficiency of an analyzer with a dead time of 40 μsec per pulse. This represents the time for analysis of a pulse in channel 1000 by an analyzer with a decoding rate of 25 mc/sec, and is an optimistic figure for most experiments. The dashed curve indicates the efficiency of a system including both the analyzer and a computer which services each data point for 160 μsec and does not permit interruption during that time to service a succeeding data point. The upper curve is the ratio of these two efficiencies and indicates the degree to which the computer dead time increases the analyzer dead time.

The negligibly small decrease in efficiency over most of the range is a result of the fact that the analyzer is freed to accept a new pulse at the start of the computer dead time. The dead times are, to a large extent, parallel rather than serial and are thus not additive. The dead time illustrated for the computer (160 μsec) is approximately that required for analysis of E and dE/dx signals, and is larger than that required for most on-line data routines. The addition of a hardware instruction for housekeeping purposes to deposit both accumulators and the link in temporary storage in one cycle, and subsequently retrieve them in one cycle, reduces the time between a priority interrupt request and resetting of the analyzer to less than 10 μsec, which is approximately that required by fixed-wire analyzers.

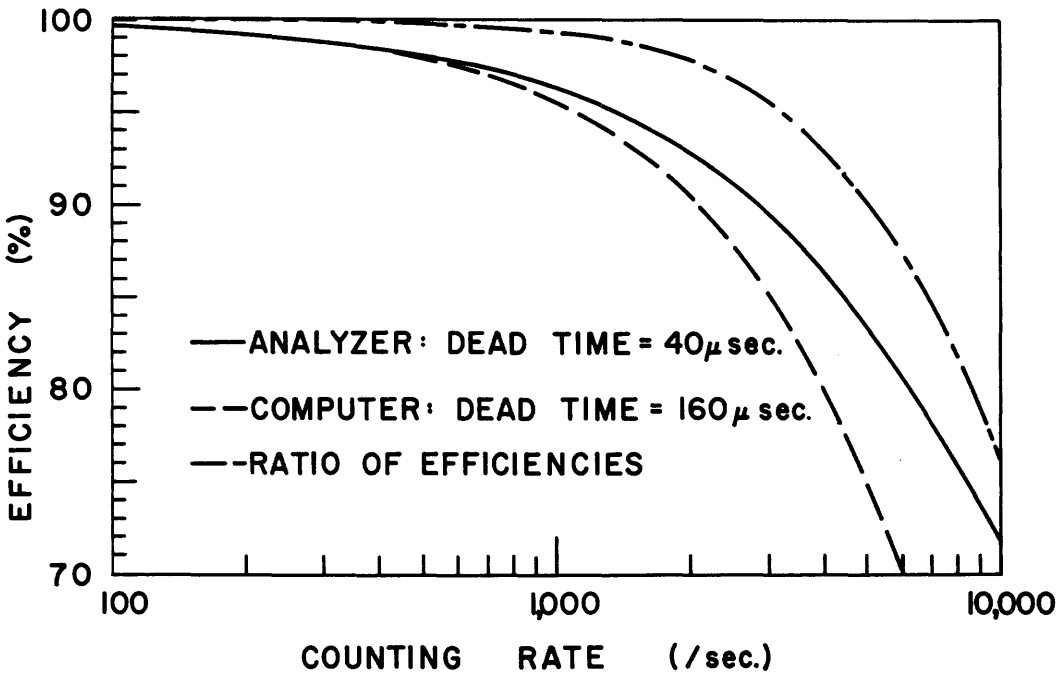


Figure 8 Efficiency curve for multichannel pulse-height analyzer as a function of stored counting rate for:
a) no time spent in storing count (solid line)
b) 160 μsec service time spent by computer on each data point (dashed line), and the ratio of these two curves (dot-dashed line)

Another objection to computer-oriented systems rather than fixed-wire analyzers is that the flexibility of the computer is provided at a sacrifice in simplicity of operation. Figure 9 shows the panel of our IDIOT (Indicating Digitizer for Input/Output Transformations) box which controls the selection of data and display modes for on-line operation. The manipulations are performed by programming, and are therefore easily altered to fit the needs of the particular situation, but the choice of mode is made with knobs and switches to facilitate control. The IDIOT box is essentially a complex sense switch, providing 36 bits of program branching flags.

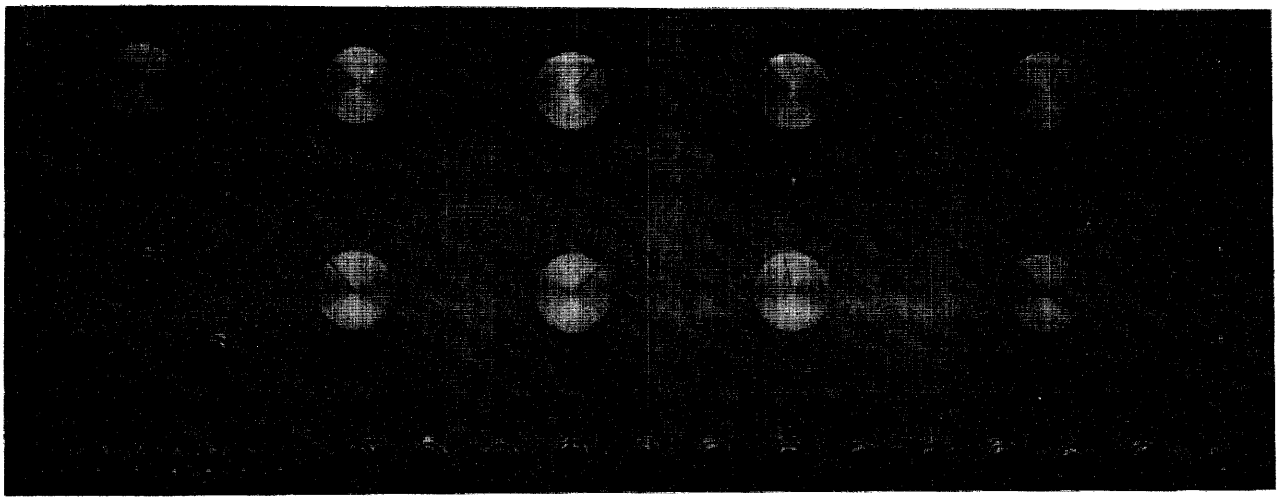


Figure 9 IDIOT box for controlling dual parameter analyzer program with knobs and switches

This particular panel presents a choice of single parameter, dual parameter, multiplexed, or E-dE/dx data; either live, E-dE/dx, contour, isometric, or log display; a choice of any of four quadrants for storing different spectra; a choice of various resolving powers by ignoring insignificant bits of the input addresses; provision for inputting up to 32 different constants for setting matrix size, discriminator windows, etc.; options to invert axes, intensify grids, or stabilize; and a choice of several modes of output or users' subroutines by pushbutton. Input/output is by interrupt, through a priority patch panel, so that there may be several devices operating simultaneously, with the oscilloscope display interspersed.

Different panels may be constructed very inexpensively for special purpose applications or modified by changing labels and programming. For an operation as complex as E-dE/dx data reduction described above, the convenience of switching easily from one mode of operation to another is of great importance.

The computer can enhance the control of the operator over his experiment in other ways. In order to monitor the number of incoming charged particles from our Van de Graaff accelerator, is converted into frequency in the form of pulses with a proportionality of 10 kc/sec per microampere. A 6-decade scaler with a 3-decade prescaler counts the pulses to yield a direct reading of the total charge accumulated, to the nearest 0.1 micro-Coulomb. This scaler and an associated electronic timer can be read into the computer for monitoring or normalization of data.

We are also undertaking two methods of control of angular distributions. The first uses up to eight detectors and eight pulse-height analyzers (without memory) to take data at eight angles simultaneously. Since the largest contribution to dead time is generally from the pulse-height analysis, this method is superior to multiplexing several detectors into one analyzer. This is an example of the way in which the flexibility of the computer memory interfaced with individual analyzer inputs provides capabilities not possible with fixed-wire analyzers with memories. The second method of controlling angular distributions is by installing shaft encoder and remotely controlled motors on our rotating detector tables. With access to the monitoring charge integrators and detectors and control over the angles subtended by the data devices, the computer can run the experiment and present to the experimenter a display of angular distributions for various groups, as well as the individual spectra at each angle.

Equally important for a computer-oriented data acquisition system, however, is the fact that the computer can be used for real-time analysis of the data. The FORTRAN capabilities of the PDP-7 are sufficient to permit reasonably complex theoretical calculations, such as the angular distribution shown in Figure 10, calculated with a plane-wave Born approximation program for deuteron stripping (written by Mr. Albert Anderson). This is the distribution for protons emitted by deuterons of 15 mev incident on Mg, in steps of 1 degree between 0° and 180°.

This program, like others written by members of our group for specific purposes, utilizes the ability of the operator to search the many parameter space for the best set of parameters needed to fit the data, weighted with an understanding of what portions of the data should be presented most precisely. An off-line fitting program is only as good as the programmer makes it, and generalized fitting programs can often lead to serious errors when provision is not made for weighing the data properly.

Eyeball fits to data, though necessarily less precise than statistical analysis, are often more accurate representations of the physics involved.

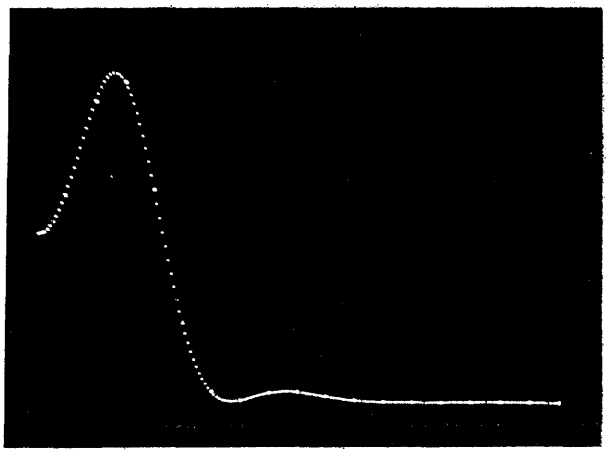


Figure 10 Theoretical calculation of angular distribution of protons from the Mg (d,p) stripping reaction in the plane-wave Born approximation

The bulk of our computer time is spent in real-time data reduction, and it is here that the greatest control over the experiment can often be achieved. The programming is done with FORTRAN, utilizing a library of subroutines to input data from our satellite analyzers, either via punched paper tapes or dumped directly from their memories. Other subroutines plot or display spectra with feedback from the oscilloscope via a light pen. Figure 11 illustrates the use of a typical data reduction program (written by Mr. Gene Sprouse). The main curve is a portion of the spectrum from a Li-drifted Ge solid state detector looking at gamma rays from ^{57}Fe and background from the tandem Van de Graaff accelerator. The narrow peak rising above the broad background represents data of interest. The curved line at the base of this peak is the background to be subtracted, determined in the following manner: Using the light pen, the cross in the lower right is moved successively to identify four points on the curve, two to the left of the peak and two to the right. The program then makes a least squares fit to a specified order to the data between the first two points and between the last two points, on the assumption that these points delimit background only. After background subtraction, the centroid and sum of counts in the peak are calculated.

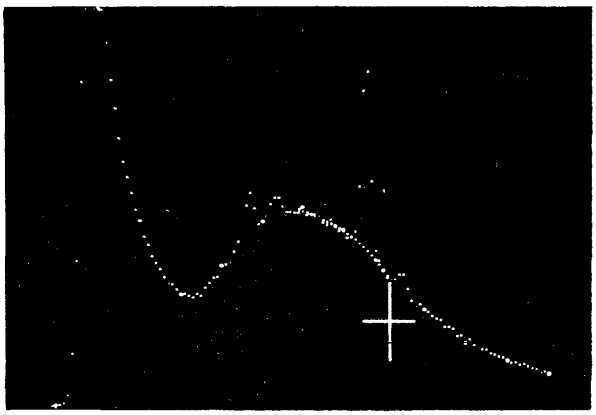


Figure 11 Pulse-height spectrum of gamma rays incident on a Li-drift Ge solid state detector illustrating real-time determination of background (smooth curve under prominent peak)

The most important feature common to these data reduction programs is that they involve interaction between computer and physicist. The operator uses the computer as a tool to display salient features of the data in a meaningful way and to do those calculations which he specifies. Too often, in off-line analysis with no feedback, the computer is used in such a way as to obscure the nature of the data.

The success of such real-time analysis programs is indicated by the wide acceptance they have had by our group. The combination of the power, simplicity, and flexibility of FORTRAN and the convenience of our system monitor, FAST Start, for retrieving FORTRAN or users' programs from DECtape, have made the computer easily accessible to all members of our group.

In conclusion, computers presently have the capability of reducing on-line nuclear physics data point-by-point without introducing intolerable losses in efficiency, and this can be done reasonably economically and with a high degree of control by the physicist over his experiment. Furthermore, the benefits accrued by installing a computer in the laboratory for real-time analysis and further reduction of this data are an even more compelling justification of computer-oriented data systems.

I would like to acknowledge the financial support of the National Science Foundation, the invaluable assistance of Mr. Albert Anderson in all phases of this project, and the ingenuity of the members of the nuclear physics group.

APPLICATION OF THE LINC COMPUTER TO OPERANT CONDITIONING

Alan Boneau

Duke University
Temporarily Studying at Stanford University
Stanford, California

ABSTRACT

Operant conditioning is a procedure in which selected behavior of individual organisms is controlled by a judicious choice of reward contingencies. This paper describes some of the work of the three psychologists in the LINC Evaluation Program and how they adapted LINC to their operant conditioning projects. The use of the computer in dealing with temporal response variables is highlighted.

Since LINC is relatively new and unfamiliar, I will begin by giving a brief description of LINC and an account of its history.

LINC is an acronym for Laboratory Instrument Computer, a fact which should imply a good bit about the design philosophy underlying it. LINC is a small computer with a memory of 2,048 12-bit words and an 8-μsec cycle time. It has various features which make it particularly appropriate for use in a laboratory.

The LINC concept evolved from the experience of several years interaction between members of the Digital Computer Group of MIT Lincoln Laboratory and the Communication Biophysics Group of MIT's Laboratory of Electronics. Some of these people were computer designers and engineers; others were essentially biologists. It was apparent to all that the existing general-purpose computers had deficiencies when used in the context of a biological or medical research program. In such programs problems arise when handling large quantities of data from various sources, particularly when one is in the process of exploring various ways of dealing with the data. There are problems of flexibility caused by interfacing with a variety of inputs and outputs for controlling apparatus and taking records. Existing machines were too large and emphasized those design features which exploited their ability to handle well-defined problems with circumscribed input and output modes. LINC was designed to overcome these deficiencies. LINC was also designed with the hope that access to a computer would stimulate development of new approaches for the biologists, approaches which previously had been in the real sense unthinkable.

The concept of LINC thus was of a small, low-cost computer which would be a permanent but mobile fixture in a biomedical laboratory. It would have a variety of input and output possibilities and be sohisticated enough to deal with the complexities of experiment control and to process data and make routine calculations. It should be convenient, flexible, and reliable enough so that it would not require constant maintenance. It should be simple enough so that routine maintenance could be performed by the investigator himself, or at least by someone in his laboratory.

The realized LINC consists of a mobile main frame which plugs into an ordinary 110-volt socket. The console is four separate units connected to the main frame individually by a set of 30-ft cables. One of the units is the usual control console. Another unit contains an oscilloscope with programmable display; a third is the LINC-tape unit, a two-unit addressable magnetic tape device, very similar to the DECtape units. The fourth unit is called the data terminal box. This is the main channel for tying external equipment to LINC. It contains two large plug-in units into which can be wired appropriate logic and buffers and interfacing using commercial plug-in units. A variety of timing pulses and registers from the main frame are delivered to the data terminal box through cables so that supplementary functions and connections are relatively easy, and in fact are encouraged. The data terminal box is also the input terminal for 8 of 16 analog input channels, the LINC being designed to act as an A-D converter with a sampling speed as small as 32 μsec between samples. Also in the data terminal box are six relays which display the contents of a programmable register in the main frame. Another feature of the data terminal box is a set of 16 sense lines connectable to external switches. These can be examined on command. A Teletype keyboard and printer provides routine communication with LINC. LINC was designed by members of the MIT group consisting of among others,

Wesley Clark, Charles Malnar, Mary Allen Wilkes, and William Simon. A prototype was completed by the group in 1962, and, on the basis of its performance, funds were sought from the Public Health Service for a program which would evaluate the computer and provide a little cloud-seeding by getting computers into the hands of a small group of investigators. Funds were provided, and the Center Development Office for Computer Applications in the Biomedical Sciences was established at MIT to administer the project. The first twenty LINCs were subcontracted by CDO.

Early in 1963, an announcement was circulated requesting proposals from investigators in biomedical fields for the use of a small-scale digital computer to be used in their laboratories. On the basis of the submitted proposals, the Evaluation Board selected twelve investigators to participate in the LINC Evaluation Program. At issue was the validity of the philosophy underlying the design: would the computer be useful in a biomedical research setting; and could it function in the way which was intended? The investigators were largely from physiological fields operating within a medical center and interested in such problems as cardiovascular functioning and neurophysiology, but there were a few others including three psychologists, myself included. We three were working in the general area of operant conditioning, which can be loosely defined as the use of judicious patterns of rewards and punishments to control otherwise spontaneous behavior. Later I will briefly discuss some of the things done by the three of us using LINC.

Each participant in the evaluation program was required to attend a four-week-long training period at Cambridge in late summer of 1963, during which he completed the assembly of his machine. In addition, this period was devoted to learning programming and operation of LINC as well as to a detailed analysis of design. The purpose was to provide an adequate enough background so that the investigator could perform routine maintenance tasks and, most importantly for the evaluation program, so that he could understand the computer well enough to make the necessary interlinkage with other laboratory equipment.

As one of the participants, I must give a personal reaction to this phase of the program. I never realized that so much could be crammed into 100 hours a week. Like the other two psychologists, I spent the four weeks at Cambridge by myself. All the other participants had brought with them a technical person from their staff. This fact is not so much a statement about the work and social habits of psychologists as it is about the size of their budgets and the relative magnitude of their projects. We simply had no one to bring with us. This meant, of course, that the dirty work involved in adapting the machine to our own use had to be done, in the face of academic schedules and commitments with no technical assistance. I feel it is a testimony to the elegance of the LINC concept and design and to the excellence of the training we received that all three of us within a year of receiving the machine had managed to incorporate it into our research in a major way, in spite of the handicaps and lack of background we shared. Most early utilizations we made of LINC occurred in the context of things we were already doing, but there emerged some differences in our plan of attack as a result of LINC. We did things that never would have occurred to us had LINC not been available, and we are now beginning to do things that could be done in no other way but by on-line computer control. With a little time to relax with the machine, I'm sure we will do more.

At the time of my initial acquaintance with LINC, my students and I were engaged in a project of teaching pigeons to discriminate colors so that we might observe the effect of various factors on their decision to peck or not to peck. In the course of the training, a pigeon was placed in a small dark box with a hole on one side through which he could peck a sheet of translucent plastic. The plastic was lighted from behind by means of a monochromator. On a sequence of trials the pigeon was rewarded for pecking to some wavelengths and not to others. While initially not very good at this task, the pigeon was ultimately coaxed into making a sharp discrimination between wavelengths as close as one millimicron. I say ultimately because such training typically required several hundred hours.

We had originally worked on this problem in a rather relaxed way, exposing each color for 30 seconds and counting the number of pecks which occurred during the period. Some of the wavelengths were never rewarded, while the others were rewarded on what is called a variable interval schedule. In this schedule, the apparatus is set to deliver a reward for the first peck after a variable interval of time, averaging perhaps every 20 seconds. On this regimen, a bird will respond to rewarded colors at a relatively high rate, and will respond to unrewarded colors at a very low rate (or never) except when the discrimination is difficult, in which case, the rate is some intermediate value.

There are some disadvantages to a rate measure for our purposes, and I will return to this problem later. Our main theoretical concern, however, was with probability of occurrence of a response; so we decided to utilize rapid sequences of short presentations of the colors, for example, 2 seconds in duration. Then we could measure directly the probability of occurrence of a peck. To do this we made use of a servo-system to drive the monochromator fed by a signal which appeared as the output of a relay-tree decoder. After being provided with a supplementary set of 24 relays, LINC was hooked into the apparatus to furnish signals for the relay tree, to open and close shutters, provide rewards, perform all the timing functions and record the occurrences of pecks, as well as randomly scramble and rescramble the sequence of colors and decide which were to be rewarded.

To keep the pigeons honest, LINC rewarded randomly about 1 peck in 20, but only to a predetermined set of wavelengths.

To get around the problem of the long training which our birds required, we decided to run the experiments overnight. The control system had developed to the point that we felt free to leave LINC in charge without monitoring. Surprisingly enough this worked very well. LINC proved to be remarkably reliable. We have no solid evidence that LINC made a single machine error in the first 6,000 hours of operation. After that, of course, followed the complicated aftermath of an attempt to clean all the accumulated pigeon dust out of memory.

At any rate LINC was running 24 hours a day for several months, during the night acting as experimenter and during the day helping us clarify the gathering data. Since LINC had the necessary capability, we decided to record all the pertinent information on every trial for each bird, a major change in procedure possible because of LINC. This information included the wavelength, the availability of a reward, and the length of time between the onset of a trial and the occurrence of a response if there was one. In a period of a few weeks, we had generated a sizable amount of data on four birds, several million chunks of information which would not have been recorded or analyzed without LINC: it would not have occurred to us to do so.

In the course of the experiment, several effects appeared which smeared the results we were looking for. One such effect was the consequence of a rewarded trial. Our suspicions were confirmed when we had LINC skip through the mass of data, selectively picking out critical trials. We decided to assess the effect of a reward by comparing performance on trials preceding rewards with that on trials following a reward. Consequently LINC selected trials on which a reward was available and obtained, amounting to two or three trials per hundred. LINC then extracted information from the trials immediately preceding and following that one, and formed a tabulation giving the number of occurrences of each of the colors before and after a reward as well as the number of pecks to each.

For example, consider a set of data obtained from one bird. The bird received sequences of presentations of the ten wavelengths from 530 to 539 millimicrons spaced at one-millimicron intervals, all presented equally often. The five values at the left, 530 through 534 millimicrons, were never rewarded. The values 535 through 539 were occasionally rewarded.

Before occurrence of a reward the pigeon discriminated remarkably well, responding consistently to these stimuli for which a reward is sometimes forthcoming. The region of transition between responding and non-responding seems to be in the neighborhood of three millimicrons. On the trial immediately following the reward, the whole curve shifted to the left about one millimicron. Since the resulting curve is almost the same shape, I could not characterize this as a breakdown of discrimination, although the effect is clearly an increase in responding to non-rewarded stimuli. Since I am discussing applications of the LINC, I do not intend at this time to get involved in a discussion of what this phenomenon might mean, except to remark that it is consistent with the notion that the pigeon is wired to act as a statistical decision maker.

Unforturnately, although this procedure results in a necessary condition for random emission, it is not sufficient. The pigeon tended to track the reward timing and to give a majority of responses around that inter-response time which currently was being rewarded. This of course produced problems.

This procedure is an example of LINC's ability to produce a very complex reward schedule which is contingent upon the behavior of the bird and which necessarily has to be handled in real time. Let me discuss another such example this time by the third of the LINC psychologists, Dr. Bernard Weiss, now at the University of Rochester School of Medicine.

I mentioned earlier that rate of responding to an extended presentation of the stimulus might be an alternative to probability of response as a measure of what is going on in experiments of this type. This rate measure has been analyzed by Don Blough of Brown University, another of the psychologists of the LINC Evaluation Program. In a situation somewhat similar to that which I have just discussed, he had LINC look at the interval of time between successive responses, the inter-response time or IRT. These were tabulated by wavelength for different IRT's. Dr. Blough found that shorter IRT's were less dependent upon the stimulus than were the longer. It seemed to be the case that once the pigeon started pecking, the probability of his making another response quickly was more a function of whether he had just made a response than it was of which stimulus was present. These sequential dependencies turned out to be a function of motivation and the time since the last reward, among other things. This tended to confound any underlying relationships.

The way out of these difficulties would seem to be a training regimen which would tend to do away with the sequential dependencies and make the pigeon approximate an ideal random emitter. One of the characteristics of such an emitter would be that the distribution of inter-response times for each stimulus would be expotential. Dr. Blough attempted to produce such a distribution by a selective reward. He divided inter-response time into bins so that if the same number of pecks were in each bin, the resulting distribution of pecks would be exponential in form. LINC was programmed to monitor IRT's on-line, categorizing pecks into the appropriate IRT bins. On each peck it looked at all IRT bins and rewarded the pig-

eon if the IRT for that peck occurred in that bin with the smallest number of pecks. Since the pigeon is sensitive to all sorts of manipulations of reward contingencies, the tendency to respond to the least frequent inter-response time should be increased.

This procedure is an example of LINC's ability to produce a very complex reward schedule which is contingent upon the behavior of the bird and which necessarily has to be handled in real time. Let me discuss another such example this time by the third of the LINC psychologists, Dr. Bernard Weiss, now at the Univeristy of Rochester School of Medicine.

Among other things, Dr. Weiss has been interested in the affect of drugs on timing functions in monkeys. One of his projects was to devise a reward schedule which would take as much variability as possible out of the inter-response time by differentially rewarding low variability. His approach is what he called an "autoregressive" schedule. Operating on-line, LINC was programmed to take the quotient of the last two inter-response times and then reinforce the monkey according to a schedule. If the quotient is close to one, indicating that the last two IRT's are nearly identical, the probability of a reward is high. The greater the difference between the last two IRT's, the more the quotient deviates from unity and the smaller the probability of reward. The actual decision to reward or not was based upon the value of a random number which was generated by LINC at this time.

In another kind of analysis of the data, Dr. Weiss has had LINC plot successive inter-response times in the form of an expectation density plot. This plot is formed by letting successive instances of an event, a peck, take the value T_0. Succeeding events are plotted as deviations in time from this moving reference.

SUMMARY

This paper presents specific applications of LINC to a circumscribed area of psychology. These applications all involve control of an experiment in which LINC is an integral part of the process, operating various pieces of equipment, sensing external events making decisions in real time according to a schedule of contingencies, and recording data.

LINC has made it possible to record raw data extensively to be analyzed as the inspiration leads rather than merely record preselected synopses of the behavior. From my own experience this has meant that data which formerly had to be extracted from a new experiment was often already available. Thus, LINC has furnished the opportunity to explore the possibilities of data analysis in depth.

LINC has also made it possible to program complicated interactions of the organism with its environment. This means that we need no longer conceive of the environment as something passive to be acted upon by the organism; it can be programmed to fight back. Possibilities implied by this are yet to be realized.

A. L. I. C. S. - ASSEMBLY LANGUAGE

Joseph A. Rodnite

Information Control Systems, Inc.
Ann Arbor, Michigan

ABSTRACT

This paper describes the algorithms for the development and implementation of an assembly language for the PDP-5/8 which makes the machine appear to the user to have every core location directly addressable. Since all the the bookkeeping is done by the assembler, relocation becomes practical. A special relocatable loader which allows subroutine relocation is also described.

For the first time it becomes practical for a user to build up a library of binary subroutines and to load them as needed. This feature alone justifies the development and use of the new language. The syntax of the new assembly language is similar to PAL, so there will be little if any difficulty in making existing routines relocatable.

Information Control Systems, Inc., primarily provides custom software for the PDP-5/8. Consequently, programmer efficiency and rate of training are of vital concern to us. A powerful assembly language greatly increases programmer productivity, training rates, and yields more serviceable documentation; these considerations and the desire to make the PDP-5/8 have every core location directly addressable lead us to develop ALICS (Assembly Language by Information Control Systems). We feel manual page turning is necessary in reading a book, but not in a computer language.

In algorithm construction needed to implement ALICS, the two features considered in most detail are the automatic page turning and relocatable output.

Program assembly is just a large bookkeeping task. Any programmer, with sufficient effort and time, can produce the same code as any language. The more the assembler assists a programmer by assuming bookkeeping tasks, the more "powerful" the assembler is said to be. One example of an assembler assuming a bookkeeping function occurs in a macro expanding assembler. After defining a macro as a sequence of instructions, all occurrences of the macro name are replaced in the program by the prespecified instruction sequence.

One very time consuming bookkeeping task in programming the PDP-5/8 is determining the proper direct or indirect address. On the PDP-5/8 a programmer can only directly address 200_8 or 128_{10} locations on the current page and an additional 128_{10} on page 0. To address other than the above locations, he must use one of the directly addressable words as an indirect reference.

The problem of automatic page turning is a subtle one. Even determining if a symbolic location is on the current page is difficult. If subsequent to the instruction assignment many locations are assigned on the current page for indirect references by the assembler, a symbolic location that appeared to represent a location on the current page may be pushed down to the next page. This would cause the previously assigned direct reference to be incorrect. This problem is avoided by the algorithm for automatic paging illustrated in Figure 1.

Instructions are read in one at a time and the location field, instruction field, and address field symbols are obtained. Each symbol, except operation codes, is placed on a page symbol table where symbol definitions for the current page are noted. When the number of instructions and the number of off page symbols + 2 is greater than or equal to the page size, reading of the current page is complete. Then an indirect reference location is created for each off page symbol and reference is made indirectly through this location. The two locations allocated in the above formula allow an exit to be made from the current page to the next, one location for a JMP instruction and one for its indirect reference to the first location of the next page.

If only the above was necessary for automatic paging, it would have been accomplished long ago. Certain extraneous details removed for clarity must also be considered.

One complication in the paging algorithm arises when the ALICS programmer makes an indirect reference. This presents no problem if the symbol is on the current page,

but this is not the case for an off page symbol. Because of hardware limitations one (unfortunately) cannot go indirectly more than one level. Consider the general indirect instruction OP I SYMBOL. One solution is to insert the following sequence of instructions if SYMBOL is not on the current page.

```
DCA     TEMP1   /SAVE THE ACCUMULATOR
TAD I   LOC     /PICK UP CONTENTS OF LOC SYMBOL
DCA     TEMP2   /PLACE IN A TEMPORARY
TAD     TEMP1   /RESTORE THE ACCUMULATOR
OP I    TEMP2   /PERFORM DESIRED OPERATION
  .
  .
  .
LOC, SYMBOL
```

A second solution is to use a subroutine which would place the contents of any machine location in a directly addressable register. For example OP I SYMBOL now becomes:

```
CALL    ISUB    /CALL TO INDIRECT SUBROUTINE
PAR     SYMBOL  /PARAMETER TO SUBROUTINE
OP I    7       /ASSUME CORE LOC 7 TO BE PRO-
                 PER REGISTER
```

Many readers will notice such powerful interrogative statements as "if SYMBOL is on the current page." How is one to know how many locations to reserve for an indirect reference when the end of the page has not been reached during the input phase? Assume only one lo-

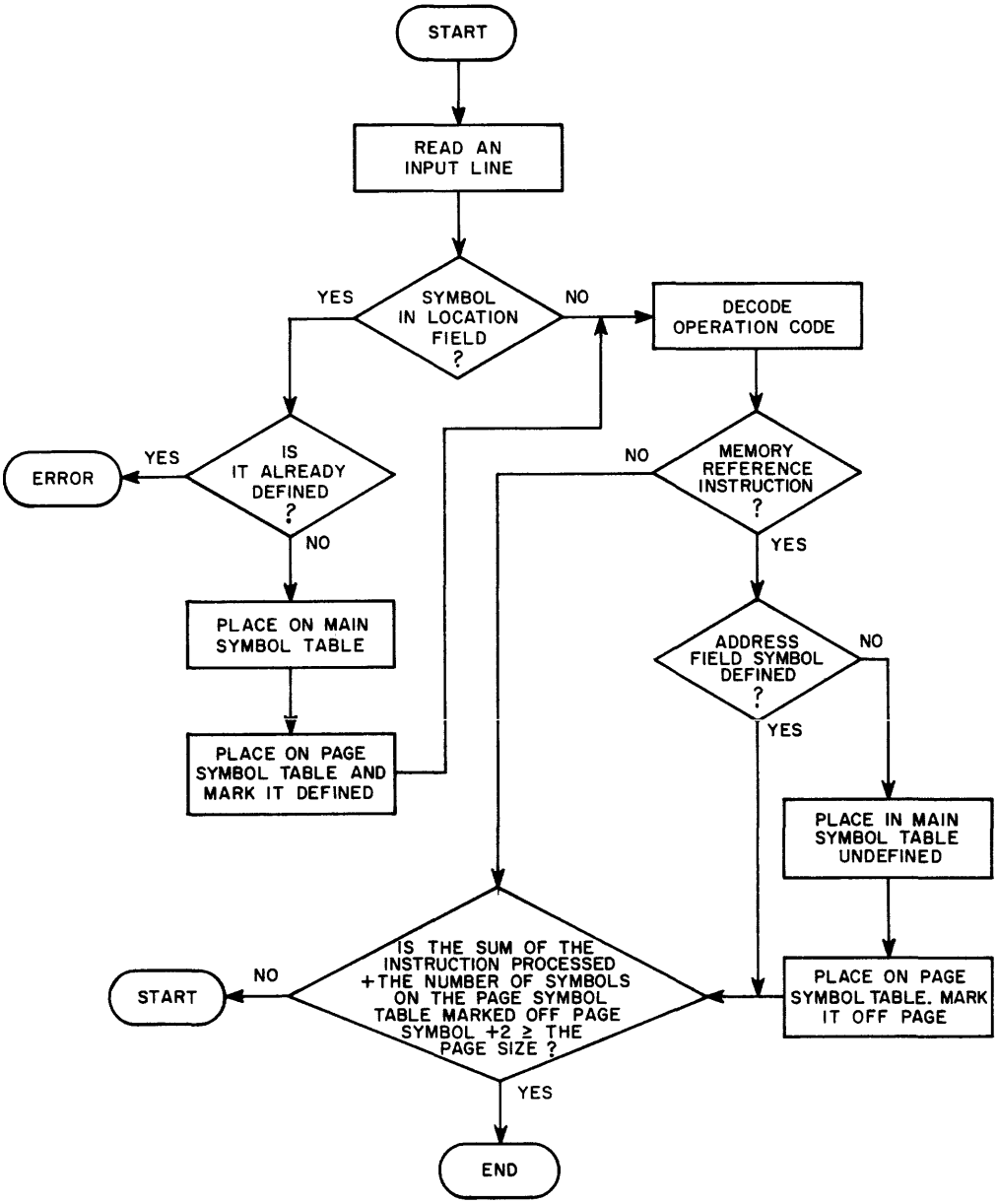


Figure 1 Flow Chart

cation is reserved. How does the assembler "back up the input tape" if indirect references are off page? Assume many locations are reserved, how does the assembler prevent gross storage use inefficiency?

Another problem in automatic page turning arises when physically breaking the code to continue on the next page. One may not merely insert a JMP instruction to a new sequence indiscriminately in a sequence of instructions.

EXAMPLES OF ILLEGAL INSERTIONS

Breaking into a skip instruction:

```
WAIT, KSF          WAIT, SKF
JMP WAIT           JMP I NPAGE
  .                NPAGE
  .                  .
  .                  .
  .                  .
                   NPAGE, JMP WAIT
```

Breaking between a subroutine call and a parameter list:

```
JMS SUB            JMS SUB
ALPHA              ALPHA
BETA               JMP I NPAGE
GAMMA              NPAGE
                     .
                     .
                   NPAGE , BETA
                   GAMMA
```

Many more examples may easily be constructed. The constructing of examples and the correct implementations to handle the examples falls into two types, non-breaking instruction groups of two and non-breaking instruction groups of $n > 2$. All non-memory reference instructions (IOT's and OPERATE) fall into the first group, while the second group consists of subroutine calls with long argument lists and special tables or data areas.

Specifying that a code sequence may not be broken is one reason for qualifiers in ALICS. It is also one of the reasons for the use of pseudo operation codes for certain functions.

The PDP-5/8 hardware makes every page relocatable to every other page. If all programs were no longer than one page, the only difficulty would be in linking programs. The absolute program philosophy of the current software severely limits the generality and usefulness of routines written in it.

If a program is desired to be relocated, the only words that must be modified or relocated before loading are the inter-page indirect addressing locations.

RELOCATION

The following examples illustrate relocating a program from starting location 1000_8 to 2000_8.

```
1000    TAD     377      2000    TAD     377
        TAD I   376              TAD I   376
          .                        .
          .                        .
1176    1200             2176    2200
1177    1234             2177    1234
1200    5670             2200    5670
```

Note that only the contents of location 1176 had to be modified in moving it to location 2176, since the contents represented an address and not a data word or instruction.

On large machines, every subprogram is linked to other subprograms by a "transfer vector" which specifies where other programs are located. There is an internal transfer vector for each subprogram in storage. This scheme may not be practically implemented on the PDP-5/8 because of the inability to directly address every core location. The transfer vector must be directly addressable from anywhere in the machine. This is the reason for reserving page 0, to create a transfer vector.

Since the assembler allows any location to appear directly addressed, the usefulness of page 0 is now not critical to the ALICS programmer. For this reason the loader requires that all routines never use page 0 with the exception of locations $0\text{-}17_8$ which contain the interrupt exit and flag cell, the autoindex registers and a few scratch locations.

To prevent confusion we use the term "connection region" instead of "transfer vector." In an assembly, all external subprograms are referenced by symbolic names. These symbols are recognized, specially flagged by the assembler, and during loading each symbolic name is assigned one location in the connection region. When the symbolic name is defined, during loading, its value is put in the proper location in the connection region. The correct core location for any symbol may now be referenced by using a location in the connection region as an indirect reference. Note that external symbols assigned to the connection region may be subroutines or possibly common data areas.

The assembler must convey certain information to the loader for each location in a program to be relocated. It does this through the use of relocation bits. Since most I/O on the PDP-5/8 is from 8-channel paper tape, two 8-bit words are necessary to specify one computer word. This leaves four extra bits. These four bits are used to specify relocation. There are 16 possible combinations of which 7 are used and 9 are reserved.

The relocation codes and their meanings follow:

Code	Description
0000	No relocation is necessary. Place the word in the next sequential location intact.
0001	Relocate the word by the current amount.
0010	Link the work to the connection region. The low order bits of the word reference an external symbol.
0100	Reset the origin to the relocation constant + the word.
0011	An external symbol is defined as the word + the current relocation. The symbol is the next 6 characters.
1111	A list of external symbols referenced by the program follows. The word contains a count of the symbols.
1000	Leader code. Same meaning as before.

Possible uses of the reserved relocation bits are for using extended memory or a time-shared system. In a time-shared system, bits may specify a user or differentiate between executive and user programs.

LOADING AND ASSEMBLY

The actual process of loading is straightforward. The connection region starts at location 100, and it may contain up to 64 symbols. Required symbols are assigned to locations in the connection region. When a required symbol is defined, its external symbols are placed in the connection region and the location it occupies in the region is given its definition.

If a subroutine is presented that is not yet required, SWITCH REGISTER options control its acceptance or rejection. This allows users with high speed I/O equipment to scan a file which contains a library of subroutines and pick out only the routines required to be loaded.

Most assemblers, such as PAL, require two passes over the input instructions before they can produce machine instructions. The only function of the first pass is to define all the symbols for use by the second pass. One pass assemblers "remember" each instruction which references a symbol defined later and belatedly produce machine instructions at symbol definitions time. This requires the assembler to remember every forward reference.

During assembly, ALICS operates on almost one page of code at a time. Having this large block of information available reduces the number of forward references required. This facilitates making ALICS a one-pass assembler.

A one-pass assembler on a machine with I/O problems such as a PDP-5/8 with only a Teletype ASR-33 is extremely useful. Assembly time is reduced by the time it takes to read through the instruction tape once. On large programs this time reduction of one-third to one-half can be considerable.

SUMMARY

In conclusion, ALICS is a complete system for the PDP-5/8 which frees the user from all restrictions of the machine except its size. ALICS output is relocatable, a feature long desired by users. Now it is practical to build up a library of binary relocatable subroutines and macros which may be loaded when necessary anywhere in the machine instead of reassembling with a new origin every time a routine is required.

This paper is not intended to convince the reader of ALICS desirability but rather is an effort to illustrate the more interesting problems arising in its implementation. It is also obvious that we feel strongly about the worth of ALICS on the basis of the features mentioned here and others that have not been discussed such as macro capability.

SOME NEW DEVELOPMENTS IN THE DECAL COMPILER

Richard J. McQuillin

Inforonics Inc.
Maynard, Massachusetts

ABSTRACT

This paper describes work being carried on by the author for the Decision Sciences Laboratory, ESD, as well as other work that has recently been completed on the development of DECAL.

The basic 1-core version of DECAL was released in its present form as DECAL-BBN in September, 1963. Since that time there has been a general increase in memory capacity of many PDP-1's, and DECAL has likewise been expanding. The first development was the moving of the compiler's symbol table into the second core field. This allowed large programs with many symbols to be compiled. Next came a 2-core modification to allow the compilation to be carried out in sequence break mode. This doubled the speed of compilation. Recently there has been a further modification to compile programs written in ASCII source language.

The author has been carrying on work to extend the DECAL language. In particular the work has been in the area of allowing real (floating-point) variables and constants to be handled automatically in algebraic statements. The other area of work has been to implement input/output facilities in the DECAL language. This has lead to the development of a language similar to the FORTRAN input, output, and format statements.

Finally the author will discuss the desirability of implementation of DECAL on the other PDP computers. DECAL has been developed into a powerful and elegant language through many man-years of effort. Perhaps consideration should be given to other machines, specifically the PDP-6 and PDP-7.

INTRODUCTION

DECAL, Digital Equipment Corporation Algorithmic Language, in its original form, came into being in 1960 as the first algebraic compiler for a DEC computer. It was designed with some very advanced features still applicable to today's compilers, and has been well proven as a compiler for the professional programmer. In fact, as an assembler-compiler, DECAL can be used when a mixture of both machine codes and problem-oriented language is required. The problem-oriented language is similar to ALGOL in call-by-value procedures and conditional statements, and in statements and subscripted variable handling. Dynamic array handling is also allowed. The compiler does not process recursive procedures, call-by-name procedures, or its own variables. The object code produced by DECAL is an intermediate code which is then loaded into core in binary by the DECAL Linking Loader. All source programs are compiled relative to a common origin. At load time they are processed in any order by the DLL and automatically relocated with symbols cross-linked between the various programs. The DLL can accept a library tape on which all utility subroutines may be stored.

The present generation of DECAL compilers was initiated in 1962, by the author and his associates at Bolt, Beranek and Newman, Inc., and is known as DECAL-BBN. This is a one-core version which contains all the features of the language described in the DECAL-BBN Programming Manual which is used today. It was soon decided, however, that one core did not allow enough room for future system expansion or for compilations of large programs, and was too restrictive because most users had more than one core memory on their computers. It was decided to move the symbol table to the second core, and this new version became known as 2-core DECAL-BBN. A subsequent version, Sequence

Break DECAL, provided for the reader, punch, and typewriter to operate under program interrupt control, making compilation about twice as fast.

The next major development came from BBN under sponsorship of the Decision Sciences Laboratory. The DSL PDP-1 Complex had grown to include ASR-33 Teletypes and a Type 164 Line Printer, both of which used ASCII code. DECAL and DECAL Loader were modified to accept ASCII code as source codes as well as to compile strings into ASCII code. The nonalphabetic codes, mostly for the instruction generators, were modified to be compativle with the ASCII character set. For example, the FIODEC codes representing instruction generators such as "∧" and "V" were changed to "AND" and "OR"; the symbol "AND" was made to have meaning both in an algebraic statement (A AND B ⇒ C) or in an instruction statement (AND B). This Teletype DECAL will be embedded into a DSL Monitor System able to compile, load, and go through a sophisticated monitor built around the Type 24 Serial Drum.

RECENT EXTENSIONS OF THE DECAL LANGUAGE

Two basic deficiencies have existed in the DECAL language until now. The first is the automatic handling of real (floating point) quantities, and the second, specification of input/output. Indeed, ALGOL itself has suffered this latter deficiency until recently.

Statements Containing Both Integer And Real Quantities

Older versions of DECAL handled real variables through a floating point interpreter that did the proper floating point operation at run time. The programmer used the operator EFM to "enter floating mode" at that part of the program. He would then write instructions that would be compiled exactly as fixed point instructions, but be interpreted as floating point at run time. Mixed statements were not allowed. In fact, all coding would be interpreted as floating point until a "leave floating mode, LFM" operator was used, or until transfer was given to a subroutine. Particularly difficult was the handling of floating point constants. For example, 3.1416 is represented in two registers, normalized floating binary as 311036,2. These numbers are very tedious to compute by hand, and some computer scheme is generally used, for example, typing in a number, having the floating point input routine convert it, and then reading the converted number from the console lights.

In the new approach, all real variables and constants must be declared before use by the REAL action operator:

```
REAL(A,B,3.1416,2.0,C).
```

The compiler generates two word locations in memory for these declarations. The compiler converts real constants into floating binary and stores these values in the allocated memory location. When the symbols are used in a statement,

```
A + 3.1416⇒B,
```

the real constant is handled just as a real variable A or B. In effect, the compiler creates a new symbol S3.1416.

Variables not declared as real are considered integers. Statements may contain mixed symbols,

```
A + 3.1416 + e + S ⇒B.
```

The compiler keeps track of the state of the accumulator and compares that with the state of the new symbol. Except for a deposit instruction, all quantities are floated whenever there is a choice between fixing or floating a number.

Real variables can also be used in procedures, conditional statements, for statements and subscripted variables:

```
If A > B then beg for i <=1 stepu 1 until n do beg.
        A[i] + b[i+3] ⇒C[i] end end
else beg funct (a, A) ⇒ ans;
        AC + A[j] ⇒b[i];
        lac b[i];
        dac b[i+1] end;
```

Here all capital letter symbols are considered real.

The Handling of Input/Output Information

The other main area of this work is concerned with handling input/output devices through the compiler language. Specifically, we are concerned with the following devices that comprise part of the DSL configuration.

1. Console typewriter
2. Reader
3. Punch
4. Type 164 Line Printer
5. Type 24 Serial Drum
6. Type 52 Magnetic Tape Control (3)
7. Type 850 DECTAPE (2 transports)
8. Type ASR33 Teletype (4 units via a Type 630 Data Communication System)

These devices will be referenced in DECAL language through INPUT, OUTPUT, and FORMAT statements. The general forms of these statements are:

```
INPUT (DEVICE, F, DEVICE PARAMETER LIST,
INPUT LIST)
OUTPUT (DEVICE, F, DEVICE PARAMETER
LIST, OUTPUT LIST)
F: FORMAT (FORMAT LIST)
```

INPUT and OUTPUT statements are compiled like procedure statements, except that terminators are generated at the end of the input/output list. FORMAT statements are special operator statements that compile character by character and generate appropriate calls to an input/output interpreter which resides in core at run time. At run time, the INPUT/OUTPUT statements are encountered in the normal sequence of the program. These are short procedures that simply pick up the device number and the terminating location in the program, and then transfer immediately to the format list. Format list elements drive the subroutines which manipulate the input/output devices.

The Formatting Language

Single characters denote control codes. There is a similarity to FORTRAN, but with extensions which hopefully give more power to the language. We have the following representations for four types of numbers: octal, decimal, floating point without exponent, and floating point:

Of. octal. Here f refers to the field size. The numbers are zero suppressed and right adjusted.

If Integer decimal. Same as octal, except digits are converted to decimal.

Ff.g. Floating point without exponent. Here f is the total field size, including sign and decimal point, and g the number of digits to the right of the decimal point.

Ef.g.(h) Floating point. Here f is the total field, g the number of digits to the right of decimal point, and h the scaling factor. It is assumed that h = 1, meaning one digit to the left of decimal point. If the specification, the scaling factor will be changed accordingly.

In addition we have the modifying codes Y and C. The Y code, for example YE8.2,causes the numbers to be left adjusting. The C code, for example CE10.2, causes commas to be inserted after every third digit, starting with the decimal point and working left.

Strings are handled as either implicit (S) or explicit (' ') strings. In addition they may be modified by the Z code that specifies the terminator. Thus:

```
OUTPUT (PRINTER, F, TAB, A)
   :
F: FORMAT (Z, S)
```

In this example the string starting at location A is printed until the terminator [tab] is encountered. As an explicit string, we may have:

```
OUTPUT (PRINTER, F, B)
   :
F: FORMAT ('THE ANSWER IS', E8.3)
```

Transfer of information is usually carried on through a buffer. When the output statement is given the information is transferred into a buffer. When the buffer is filled, it is automatically transferred to the device.

Two types of binary information are transferred --buffered and unbuffered; only in the unbuffered binary is information directly transferred from core to device.

The typewriter, Teletypes, and printer have page formatting control codes. They are:

X	Blanks. Generates blanks (spaces) in the output.
T	Tabs. Inserts tabs in the output.
↑	Page advance.
/	End of line.
P	Page. Will start numbering pages with the number specified.
H	Heading. Provides a pointer to a heading string that will be output on every page advance.
L	Number of lines on a page.
K	This operator gives values of tabs.
M	The number of spaces equivalent to the left margin.

As an example of the use of these codes, the following statement could be used to set up the parameters for the printer:

```
OUTPUT (PRINTER, F, 1, F2, 60, 30, 10, 20, 30
        40, 50)
   :
F: FORMAT (P, H, L, M, 5K)
F2: OUTPUT (PRINTER, FF, A, B)
FF: FORMAT (S, E5.2)
```

Here the first output statement contains the information:

1. Start numbering the pages with p = 1.
2. There will be 60 lines per page.
3. The left margin is equivalent to 30 spaces.
4. The tab settings are at 10, 20, 30, 40, and 50.
5. The heading can be obtained through statement F2.

Statement F2 is another output statement with associated statement FF. The heading is made up of a string(S),

starting at location A and a floating-point variable (E5.2), B. This rather lengthy process allows dynamic headings.

Under certain conditions the DECtape, mag tape, and drum are referenced the same way. That is, all reference files and records within the file. The following codes are appropriate for these devices:

W	Rewind selected unit
G	Create an end-of-file
N	File reference
Q	Record reference
R	Return on an end-of-file

A typical file handling output statement might be:

```
OUTPUT (MAGTAPE, F, UNIT, FILE,
        RECORD, IA, FA)
   :
F: FORMAT (N, Q, U, G)
```

Here the unbuffered binary information between core locations IA-FA is written onto magnetic tape starting with the specified file and record numbers. After the transfer an end-of-file mark is written. Likewise an input statement might be:

```
INPUT (DECTAPE, F, UNIT, FILE, RECORD,
       EOFR, IA, FA)
   :
F: FORMAT (N, Q, R, U)
```

These statements specify a return, EOFR, if an end-of-file mark is encountered on the transfer. Thus the user may transfer a variable amount of information, terminating on an end-of-file.

The system also has considerable power in array handling. Single elements of an array may be handled by statements such as:

```
OUTPUT (PRINTER, F, A, B, C[i, j])
   :
F: FORMAT (E8.2, 5X, E10.1, 5X, I8, /).
```

More generally, whole arrays may be handled by a symbol ARR which the format interpreter system uses to signal the reference to the entire array. Thus statements like:

```
OUTPUT (PRINTER, F, A, B, C, ARR D)
   :
F: FORMAT (4E8.3, 2F10.4, /)
```

give the output

A B C D_1 (at E8.3) D_2 D_3 (at F10.4)

D_4 D_5 D_6 D_7 (at E8.3) D_8 D_9 (at F10.4)

:

etc. until all D's are printed.

Finally, the language allows a variable replicator. If a V is used in a format list, the value of this code is determined at run time as the next element in the input/output list. For example,

```
3 ⇒ N
OUTPUT (PRINTER, F, N, A, B, C, D)
   :
F: FORMAT (3X, VE8.2).
```

For a more complete discussion of this principle, the reader is referred to Reference 2. Incidentally, if the value at any time is 0, the element is ignored. Replicators may be used to any depth as well, as long as the statements are logically correct. For example:

```
OUTPUT (PRINTER, F, A, B, C, D, E)
   :
F: FORMAT (E8.3, 4(3X, E8.1), /).
```

DECAL AS A LANGUAGE

Presently, DECAL is written for just one computer, the PDP-1. This computer is old by manufacturers' standards and is considered obsolete. The question arises whether or not DECAL should be considered a machine independent language and should be implemented on newer computers, particularly newer DEC computers. Many man-hours have gone into the development of this language, and it would seem worthwhile to examine its assets.

1. The DECAL Language is Powerful

First, DECAL contains the basic machine instructions as a subset of the language. Basic machine instructions may be algebraic statements. There can be a very strong interaction between basic machine code and problem oriented language with a program, even within a statement. Second, the problem oriented language contains most of the features of ALGOL, which is considered more powerful than FORTRAN. For the novice, the DECAL language is more complicated to learn than FORTRAN; however, it gives the professional programmer more power to solve complex numerical, logical, and symbol manipulative problems. Third, it contains a FORTRAN-line input/output facility, which is easy to learn, and gives the programmer easy access to the various input/output devices.

2. The DECAL Compiler is Self-Extending

The DECAL compiler is written with a bootstrapping facility. The compiler system consists of a basic, skel-

etal compiler, and a series of symbolic source language tapes that contain all the word definitions and operator definitions the system needs. The compiler then compiles these definitions into itself to become the general version. This facility is available to the user to define his own operators and compile them into the system. These operators are like macros, but more powerful in the sense the full power of the compiler is available to the user to call upon and even modify at will. The DECAL compiler is organized as a number of subroutines, with an executive subroutine, that the user may call upon with his operator definitions.

With this bootstrapping feature, the language is easily extendable to powerful, special-purpose languages.

3. The DECAL System is Well Documented and Field Tested

Since the first version of DECAL appeared about five years ago, the compiler and the language have been in an almost continuous state of development. Elaborate diagnostic routines have been written to thoroughly test the system, and every new version has had to compile these routines correctly before release. After going into the field, DECAL compilers are used in a variety of programming situations, and all installations have reported essentially error-free operations for several years.

The DECAL system has been documented by both a Programming Manual and a Technical Manual and, as new developments have come along, they have been well documented.

ACKNOWLEDGEMENT

This work has been supported by the Decision Sciences Laboratory, ESD, United States Air Force, under contract AF19(628)-5061. The author wishes to acknowledge the many fruitful discussions with contract monitor Major Frank Sulkowski leading to the final specifications of the formatting language.

REFERENCES

1. McQuillin, R. J., The DECAL-BBN Programming Manual, DECUS, September 1963.

2. Ranelletti, John E., "Dynamic Format Specifications," Comm. ACM, 8, 8, pp 508-510.

3. McQuillin, R. J. The DECAL-BBN Technical Manual, DECUS, September 1963.

4. Krueth, D. E., et al, "A Proposal for Input-Output Convention in ALGOL 60," Comm. ACM, 7, 5, pp 273-283, (May 1964).

APPENDIX I

HARDWARE CONFIGURATION FOR FORMAT CONTROL

1. Console Typewriter

2. Reader

3. Punch

4. Line Printer (Type 164)

5. Serial Drum (Type 24)

6. Magnetic Tape - 3 (Type 52)

7. DECTAPE - 2 (Type 550)

8. ASR33 Teletypes - 4 (via 630 Data Communications System)

THE GENERAL INPUT, OUTPUT, FORMAT STATEMENTS

INPUT (DEVICE, F, DEVICE PARAMETER LIST, INPUT LIST)
OUTPUT (DEVICE, F, DEVICE PARAMETER LIST, OUTPUT LIST)
F: FORMAT (FORMAT LIST)

FORMATTING LANGUAGE

Numerical

Of.	Octal
If.	Decimal Integer
Ff.g.	Floating Point, No Exponent
Ef.g. (h)	Floating Point

Numerical Modifiers

Y.	Left Adjust
C.	Commas

String Handling

S.	Implicit Strings
' '.	Explicit Strings
Z.	String Terminator

Page Control

X	Blanks
T	Tabs Insert
t	Page Advance
/	End of Line
P	Page Number
H	Heading
L	Lines per Page
K	Tab Values
M	Margin

File Control

W	Rewind
G	Create End-of-file
N	File Reference
Q	Record Reference
R	End-of-file Return

Binary

B.	Buffered
U.	Unbuffered

Miscellaneous

V.	Variable Replicator

MESSAGE SWITCHING SYSTEM USING THE PDP-5

Sypko W. Andreae

Lawrence Radiation Laboratory
Berkeley, California

ABSTRACT

At the Center for Research of Management Science at the University of California, Berkeley, recently a message switching system was constructed for research of human behavior in game situations.

The system uses a PDP-5, a 630 System with ten Teletypes, a DEC tape unit, and an IBM compatible tape unit. The basic laboratory layout entails many soundproof rooms in each of which the subject can use one of the Teletypes. The controlling program is essentially a simply time-sharing system of which the organization is outlined in this paper. Among the many advantages of the existing system over the previous manual method are improved traffic intensity, better record keeping, network control by the experimenters, etc. Two examples of games used in these experiments are given.

In the Department of Business Administration on the campus of the University of California, Berkeley, California, a laboratory was constructed recently named "Center for Research of Management Science," (CRMS). Most of the research in this laboratory is aimed at the behavior of people in business situations. In the experiments being conducted, questions are asked such as: On what does a businessman base his decisions? Does he listen to qualified advice, and if he doesn't, why not? What causes one person in a team of businessmen to be dominating?

To shed some light on how experimenters go about setting up their experiments, I will give the following hypothetical example: A team of five people run a fictional company in a simulated market situation. The five functionaries are a Chief Executive, a Purchasing Manager, a Comptroller, a Sales Manager and a Production Manager. Beforehand, the subjects (in general students), are supplied with information about the status of the company they are going to run. In general, the company sells three products and they start from a situation which is illustrated in financial statements. One game may take a whole afternoon, and consists of half-hour periods. During each period all team members communicate with each other via communication channels. The communication network may or may not be under certain constraints. There are several reasons for these constraints. For instance, it would be very difficult to find causes for certain kinds of behavior when communication was completely free. In a real situation, the businessman bases his decision on many facts coming from many sources, not only his letters, the telephone, but also the radio, TV, the newspaper, cocktail parties, etc.

In the experiments, the subjects are not able to see or hear each other; they can only communicate via the written word. First, this took the form of written notes which were picked up and delivered by research assistants, who kept a kind of running mail service between all the subjects. Now this is being done with Teletypes connected to a PDP-5.

The experimenters are certainly aware of the fact that the constraints being put on the communications medium creates a situation which does not compare with the reality of business life. However, when one's aim is to search for certain well-defined and specific phenomena, these constraints do not necessarily invalidate the results of the research.

The properties of the market in which the company is being run are well known, and therefore an optimal solution for the operation of the company exists. The subjects, in general, do not know this solution. Every half-hour the team has to make a decision about purchasing, production, etc. The decisions are then fed into a computer program which prepares financial statements as a result of the decisions of the team and the properties of the market. During the next period the subjects can see the results of their previous action.

The standard model of this experiment is very often modified to study certain specific effects. For instance, sometimes a stooge is inserted. Unknown to the others, one of the team members is informed about the key to the optimal operation of the company. Once the game is started he is not to inform the team members of his specific knowledge, but nevertheless he is to convince the team members how to make certain decisions. It is in-

teresting to see that in most cases the team members do not accept his advice, however well-founded it may be.

During the summer an experiment was performed in which an operations research group was included in the team. The five executives were student subjects; the operations research group (which was in touch with all team members via Teletype), consisted of three highly skilled professionals in the business trade. Still, it was obvious that most of their advice was not being accepted, and in many cases their suggested decisions were vetoed by the Chief Executive who happened to have different "feelings" about the matter. I don't intend here to elaborate on the research itself. These remarks were merely an introduction to the atmosphere of this project.

The use of the Teletypes over the use of little slips of paper was an obvious improvement. Of course the costs of a small computer and an interface for Teletypes, the Teletypes themselves, the cabling in the building, etc., is considerable, and therefore it is obvious there must be very good reasons to offset these expenses. Without going into detail, I could mention a few of those reasons.

The time sequence of the messages needs to be recorded. There is also a need to process communication patterns as they emerge from the experiment. Also, several efforts have been made at CRMS to use informal language. This is a language of which all the rules are known and in general, this takes the form of a predetermined set of sentence elements which a subject is to use only according to a predetermined syntax. Messages furthermore need to be stored and later recorded together with all pertinent experimental data.

Previously, much of the preparation for analysis of the experimental data was done by scores of secretaries who transcribed all the communications, counted communications phenomena, and tried to find the proper time sequence.

Figure 1 shows the configuration of the equipment now being used, and the remainder of this paper will develop several reasons for the acquisition of this particular configuration. Instead of a small computer like the PDP-5, one could probably use a simple electronic switching network when communication is the only function to be performed. A stored program processor as a part of the system obviously can perform many more desired tasks, and will make the whole system more versatile. Unlike the normal telephone system (one-to-one communication), this system was required to have fanned-out communication; that is, any one Teletype should be able to communicate to one or more others. Once this is allowed for, one has to deal with the difficulty of unavailability of the receiving stations. The computer can take care of this by storing the message temporarily and transmitting the message as soon as the unavailable stations are available.

Storage on IBM compatible magnetic tape is especially convenient in view of future off-line processing on a larger machine.

Figure 2 shows a memory map and in the following I will elaborate on the construction of the software that controls the message switching system.

The memory is roughly divided in two parts of which one-third contains the different control programs and roughly two-thirds is reserved for temporary storage of messages of all of the stations. Each station has its own message area which can contain about 450 characters. The control programs take care of a time-shared processing of characters received from the Teletype stations. The characters received from the Teletype stations can be the result of a keyboard action at the station or from the transmission of one character from the computer to the station since all Teletypes are connected to the PDP-5 in a half-duplex fashion. The interface for the ten Teletype stations is made by DEC and is called 630 Data Communication System.

In the use of a half-duplex system, one must remember in the program whether a station is being transmitted to or is being received from. This problem is solved by the use of status words. Each station has a status word from which the program learns what to do with a received character. But this is not the only function of the status word. It also serves as a memory for the particular phase of the procedure in which the station happens to be. I will elaborate on this procedure later.

The total program can be divided in two main parts: SWITCHYARD and all consequential routines, and DISTRIBUTOR. On the upper part of the memory map is SWITCHYARD, which is responsible for processing incoming characters according to the information in the status word and also for transfer of control to the proper part of the remaining software system. DISTRIBUTOR tries to initiate the transmission of stored messages to these Teletypes, in case no characters are coming into the PDP-5.

FILTER is a routine which executes certain prepared constraints on the communication patterns between stations.

A software clock updates the time every second. On the same page is an interrupt service routine for other devices. The routine specifically responsible for the transmission of messages is TX#SUB, the transmitting subroutine. The routine responsible for the construction of a message as it is received from a station is RX#SUB, the receiving subroutine. There is a program which consists of the controls for the D-2020 tape unit. It includes all the timing and length control of records that can be both read and written, and it can generate end-of-files. The remainder of the memory is used for the ten station message areas.

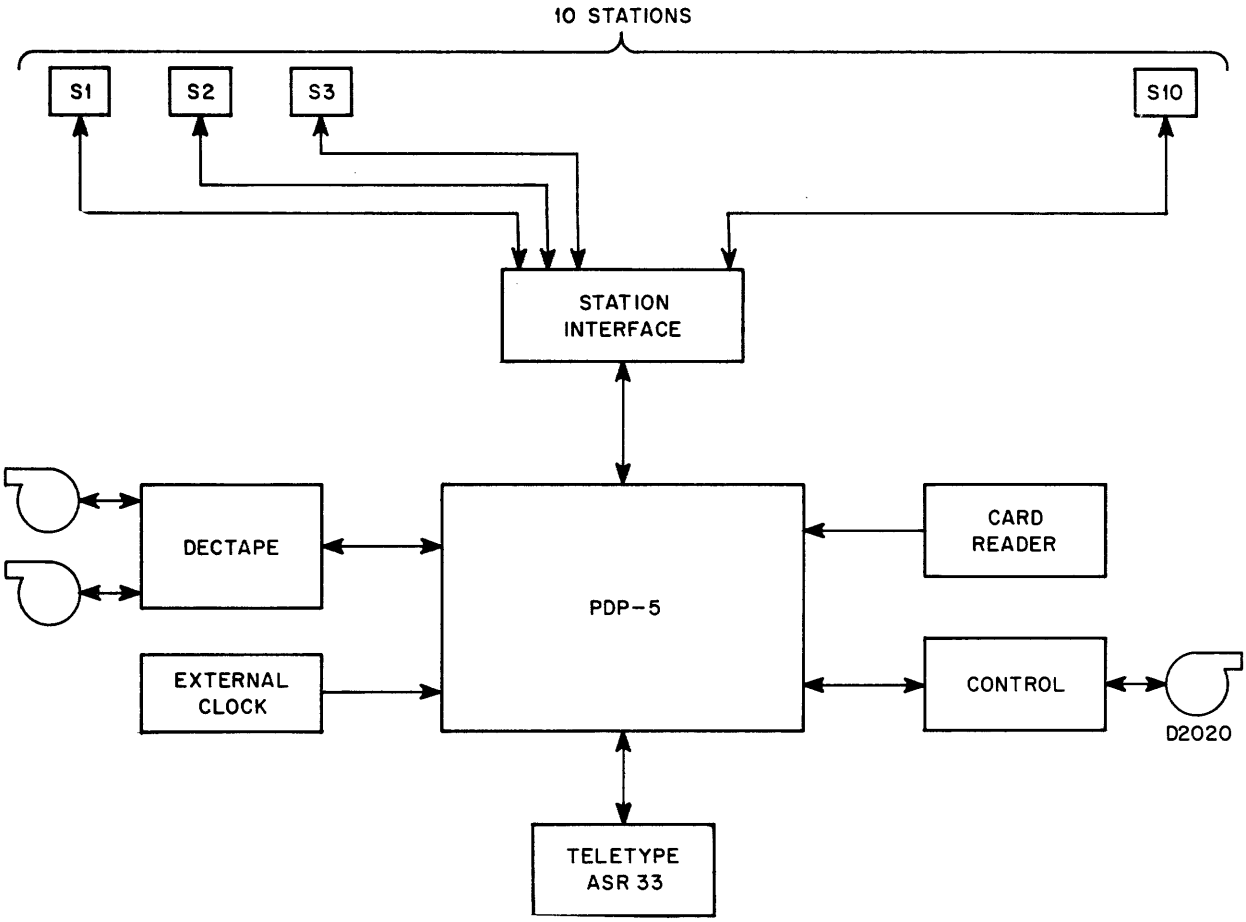


Figure 1 Management Science Laboratory System

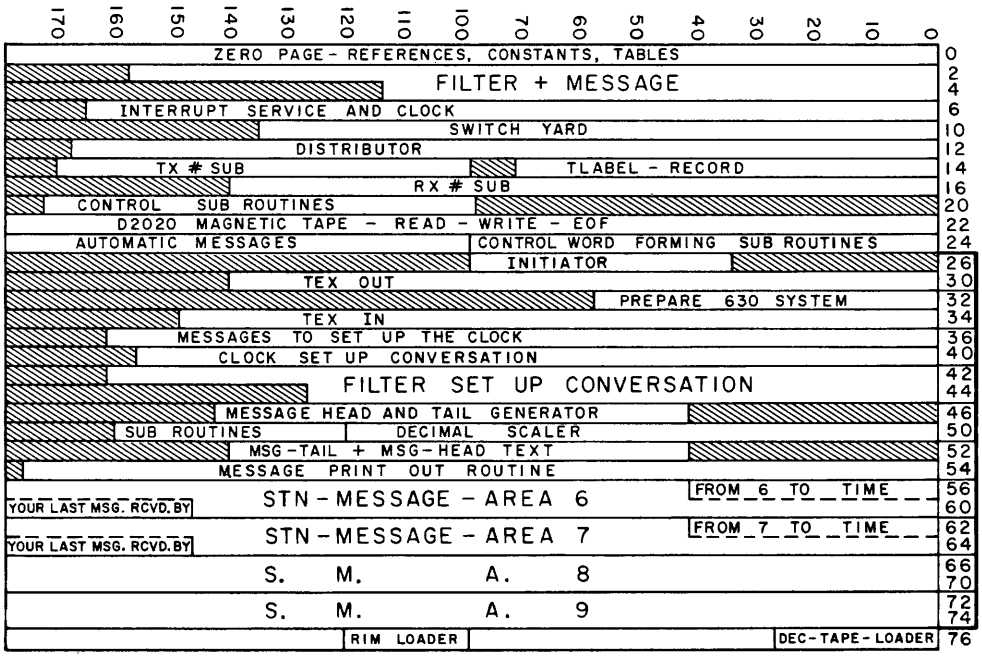


Figure 2 Memory Map

Before the experiments start, however, the experimenter has an opportunity to set them up as he likes. He therefore enters a conversation with the PDP-5: the PDP-5 asks for certain information from the experimenter, which, after his response, is stored in the memory. For instance, the program will ask the experimenter to specify with which stations one particular station is allowed to communicate, whether a station can only communicate to one station at a time, whether each station should be sent back a copy of his own message, and whether certain monitoring stations should be added to the requested destination as provided by the station of origin. The PDP-5 will ask the experimenter to type in the time at which the clock should start and then finally to start the clock (see Figure 3).

All the programs necessary to control this conversation and to store information from the experimenter are loaded in the part that is used later by the stations as message areas.

Once the clock starts, all conversational routines are destroyed and the message switching program is in control. All subjects at the stations start their communications according to a certain procedure.

```
TYPE 1 IF A MESSAGE MAY HAVE ONLY 1 ESTINATION - TYPE 9 OTHERWISE.

1

TELETYPE NO. 0 MAY SEND TO

01234↑

TELETYPE NO. 1 MAY SEND TO

012334///// 01234↑

TELETYPE NO. 2 MAY SEND TO

01234,PLEASE.↑

TELETYPE NO. 3 MAY SEND TO

43210↑

TELETYPE NO. 4 MAY SEND TO

0,1,2,3,4↑

TELETYPE NO. 5 MAY SEND TO

↑

TELETYPE NO. 6 MAY SEND TO

↑

TELETYPE NO. 9 MAY SEND TO

56789↑

TYPE STNS TO WHICH ALL MESSAGES GO

0,5↑

TYPE 0 TO START OVER-TYPE 1 TO CONTINUE

1
```

Figure 3 Teletype Message Destination

```
GIVE HOUR AT WHICH CLOCK SHOULD START

13

GIVE MINUTES

04

GIVE SECONDS

23

CLOCK WILL START AT    130423
TYPE 0 TO START OVER-TYPE 1 TO START CLOCK

1

CLOCK HAS STARTED
```

Figure 3 Teletype Message Destination (continued)

The next figure shows what the subject sees on his Teletype as a consequence of conversation with the message switching program. After hitting carriage return on the Teletype, the computer answers, "TO WHOM"? The subject types in the station numbers that he requests as destinations for his message, delimiting the string of station numbers with a vertical arrow (used throughout this system as a MESSAGE DELIMINTER). The program ignores anything other than numerals during this part of the procedure. As soon as the program detects the delimiter, it invites the subject to start his message by the response "GO AHEAD." He can now enter his message in standard English and completes it by typing the delimiter. The program then adds certain information to the message now residing in the core memory (from what station, to what stations the message is going, and the time), after which the program proceeds to write a record on magnetic tape containing this message.

```
*** TO WHOM? ***
7↑
*** GO AHEAD ***

THIS TIME IT SHOULD GET THROUGH!!!!!!!!!   ↑


FROM 8 TO 0 5 7 8    TIME 133618

THIS TIME IT SHOULD GET THROUGH!!!!!!!!!

YOUR LAST MSG RCVD BY 0 5 7 8




FROM 7 to 0 5 7 8   TIME 133915

INDEED, STN # 8, YOU GOT IT.
THIS IS TO SHOW YOU THAT I DO NOT EVEN NEED ONE MISTAKE TO GET
THROUGH TO YOU!!!!!!
```

Figure 4 Teletype Conversation with Message Switching Program

Figure 5, a simplified flow-diagram of the program, shows these actions. After initiation, the system continuously checks the flag of the 630 System. As soon as a character arrives, the status is checked, and when the character is recognized as a carriage return and the status word for this station equals 0, the message "TO WHOM"? is transmitted to that station.

Note that SWITCHYARD only initiates the message. Transmission subroutine TX#SUB takes care of the transmission of the message until it finds the delimiter. The status word is then modified to indicate completion of this part of the procedure.

The DESTINATION ROUTINE assembles the incoming numerical information and stores the result in the REQUESTED DESTINATION WORD (RDW). After the delimiter is detected, the message GO AHEAD is initiated and the status word is modified accordingly.

The next character coming in is processed by RX#SUB, the receiving subroutine. When RXSUB detects the delimiter, control is transferred to the routine called TLABEL.

In the top of the flow diagram next to node D is the DISTRIBUTOR which starts transmission of messages residing in the station message area if the destinations are available. Once it is successful in initiating a message to an available station, a special word called the DESTINATION WORK WORD is updated. When DWW is updated, the bit corresponding with the station to which the message was being initiated is subtracted from the DWW. This word (DWW) contains all destinations for one particular message. Another function of the DISTRIBUTOR is to send a message (YOUR LAST MESSAGE IS RECEIVED BY...) when all destinations of a particular message have been reached.

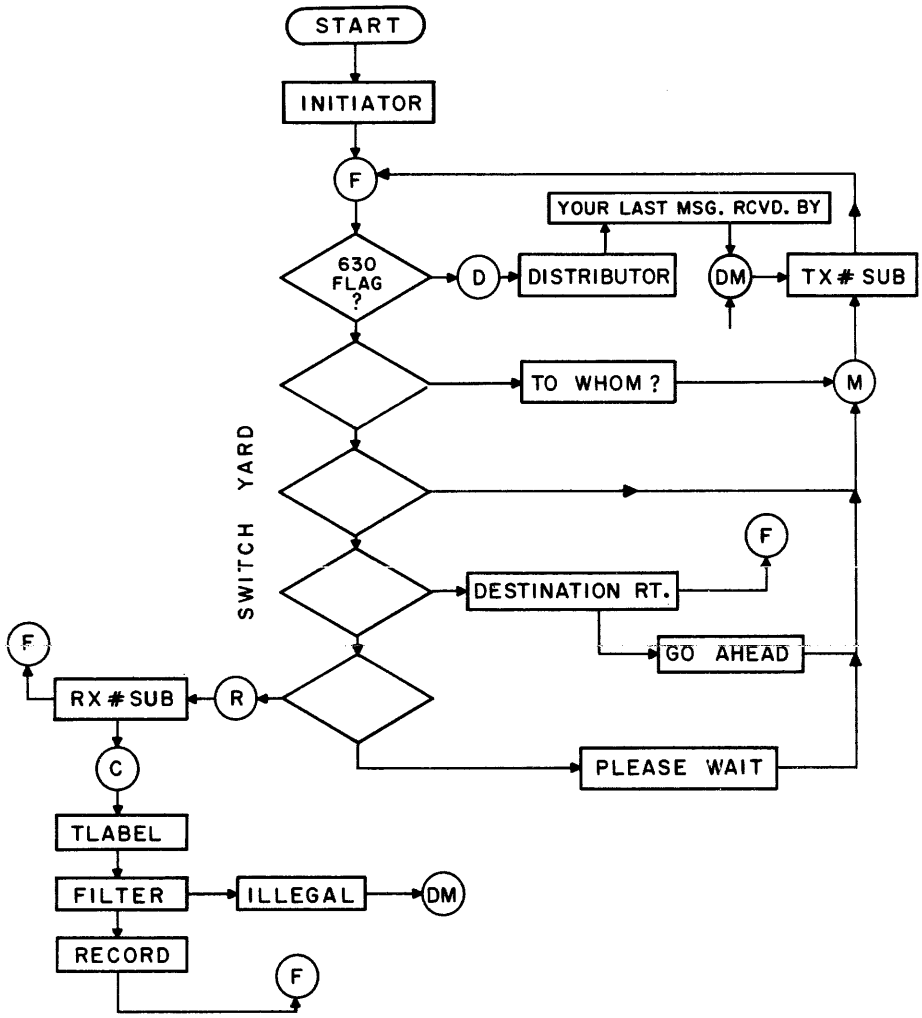


Figure 5 System Flow Chart

The receiving subroutine, RX#SUB stores the characters, two characters per PDP-5 word, in the message areas of the stations. It accounts for the load pointers, pack words, etc., guards against overflow of the message area, handles special characters, and corrects the status word at the end of the message. Most characters are trimmed to a code which consists of six bits per character. The 6-bit code is formed by subtracting 240_8 (octal) from each Teletype character. The first character to be processed by the RX#SUB is placed in the pack word which belongs to that particular station. In the meantime RX#SUB can handle characters of many other stations before it returns to the previous station to handle its next character. The pack word is retrieved, the next character is placed in the left half of the pack word, and this word containing the two characters is stored in the message area. This packing method is of great importance for economical utilization of the available memory space.

A character set of 64 characters may not be enough, however, although the frequency of the use of characters which do not fall within this group of 64 is very low. Therefore, RX#SUB as well as TX#SUB have a special arrangement to process characters not belonging to this group of 64 (for example: carriage return and line feed). One complete word is storing these special characters.

The program FILTER is essential for many of the proposed experiments in this Laboratory. FILTER modifies the destinations of each message before the distributor initiates transmission of the message to one of the destinations.

The following are functions of FILTER.

1. Requested destinations are checked for the presence of one or more destinations, if the experimenter has specified that only one destination per transmission will be legal. If, nevertheless, the station has requested more than one destination, control is transferred to a routine called ILLEGAL. The ILLEGAL routine destroys the message built up in the message area and returns to the violating station a message telling the subject why his message did not come through. Note that one station may be allowed to transmit to several other stations but, if so specified by the experimenter, only to one other station per communication.

2. When the experimenter has so specified in the beginning, the message from the station of origin is returned by the computer to the station of origin.

3. When the subject at the station has requested a group of destinations among which several are illegal, FILTER subtracts those destinations from the requested group of destinations. Thus the REQUESTED DESTINATION WORD (RDW) is modified into the LEGAL DESTINATION WORD (LDW). In this case the message will come through when there are still some legal destinations left and the routine ILLEGAL will not be used.

4. Certain destinations specified by the experimenter beforehand can be added to LDW. This facility is being used by experimenters who want to have a Teletype station available on which all communications can be monitored instantly.

A real-time clock arrangement labels all messages from the stations with a time tag containing hours, minutes, and seconds. This is useful for later analysis, and helpful during the communications in referring to previous messages. The 1-mc computer clock is used as a time reference base. Six micro decades are used in a 6-digit scaler arrangement. This scaler produces a carry every second. This carry pulse enables two flags; each of the flags connects to the program interrupt bus and the I/O skip bus. If one or more flags are set, a program interrupt occurs, and the information that one or more seconds have passed is processed. There may be reasons for which the interrupt is temporarily turned off. In that case, the hardware is able to remember how many seconds the software has missed. To accomplish this, the two flags are arranged in a scaler configuration. If the program interrupt is turned off, and the flags are therefore not being sensed or reset, the flag scaler is able to count up to 3 seconds. When the interrupt is turned on again, the program interrupt occurs and the program that services flag-interrupts determines how many seconds the software clock is behind and updates the software clock accordingly. As shown in Figure 6 the clock can be started and stopped under program control. As an additional feature it is possible to read a small part of the 6-decade scaler into the accumulator.

The flow diagram of the clock explains how the real-time clock is simulated. The essential part of the routine is a scale of 60 which transfers its carry to another scale of 60 which transfers its carry to a scale of 24.

One of the interesting technical questions that one can ask about the system is if the system is capable of handling all the traffic at its highest possible intensity. Since the equipment arrived in August, 1965, and the software was completed two months later, there has been little opportunity to do measurements. The first indications are that during a normal experiment the PDP-5 idles about 95% of the time. Nevertheless, certain queuing effects generate a potential danger of garbling messages under certain circumstances. In the existing system, every character that comes in is immediately and fully processed. The length of the process varies quite a bit according to the status of the station. Certain probability of garbling is due to the constraint that the character of a particular station has to be processed within a limited time after its arrival. When a Teletype transmits characters at full speed, the rate will be one character each 100 msec. Somewhere near the end of the character cycle, the

Teletype flag is raised, and if the flag is not acknowledged within a period of about 20 msec, part of the next character may be superimposed on the previous character in the Teletype buffer, with disasterous results.

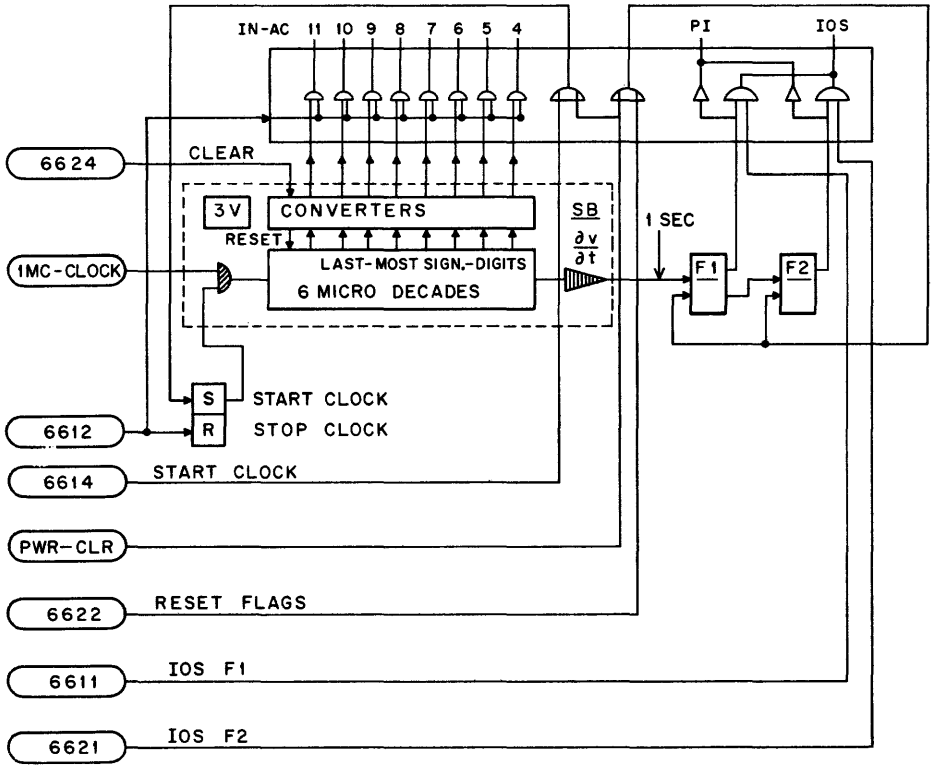


Figure 6 Hardware Clock

One obvious way to resolve this type problem is the use of a buffer area in core (see Figure 8). As soon as a character arrives, it is stored in a buffer consisting of 24 bits, of which 12 bits are used for the character and 12 bits are used for the station number. Each buffer element could consist of one word since the station number will never use more than four bits, and a character will never use more than eight bits, but for simplicity we choose one pair of 12 bit words as the smallest buffer-element. The total buffer has a length of 20 word pairs. As soon as a character arrives, it is stored in a buffer element under control of an input pointer. SWITCHYARD retrieves characters plus station numbers from this buffer under control of an output pointer. SWITCHYARD now checks if the output pointer equals the input pointer and if not a new character can be processed. Once the pointers reach the bottom of the buffer, they are transferred to the top again, so that in effect the buffer is a cyclic software device.

Now other interesting measurements can be made. For instance, at predetermined time intervals a record can be made of the position of input and output pointers. The maximum distance between the input and output pointer dictates the length the buffer should have. This configuration also affords a reliable method to measure average maximum and minimum times to process each character after its arrival.

The following is a short description of another type game for which the system program was completed. Apart from functions the computer performs as described above, several special functions are added. The experimenter may have several versions of his game prepared and stored on the library tape. The versions differ in the established communication patterns. In this game there are nine subjects, and the experimenter uses one of the Teletypes, station No. 0.

At the beginning of every trail all subjects are given a piece of information, for example a character, or a word. During the trail period all subjects are to put all pieces together in cooperation with each other. Each subject is to give a solution for the problem. For example, the experimenter may give each of the subjects one word and ask them to form a sentence of the nine words. As soon as one of the subjects thinks he has the answer he transmits his answer to the experimenter. Every transmission to the experimenter is being remembered by the computer program. As soon as all nine subjects make a communi-

cation to station 0 (whether they had the right answer or not), the trial period automatically ends and the computer sends a message to that extent to all subjects. Both the experimenter and the computer program have control over the completion or starting of all phases of the experiment, the pretrial period, the trial period, the intertrial period, etc. All pertinent information is recorded on tape.

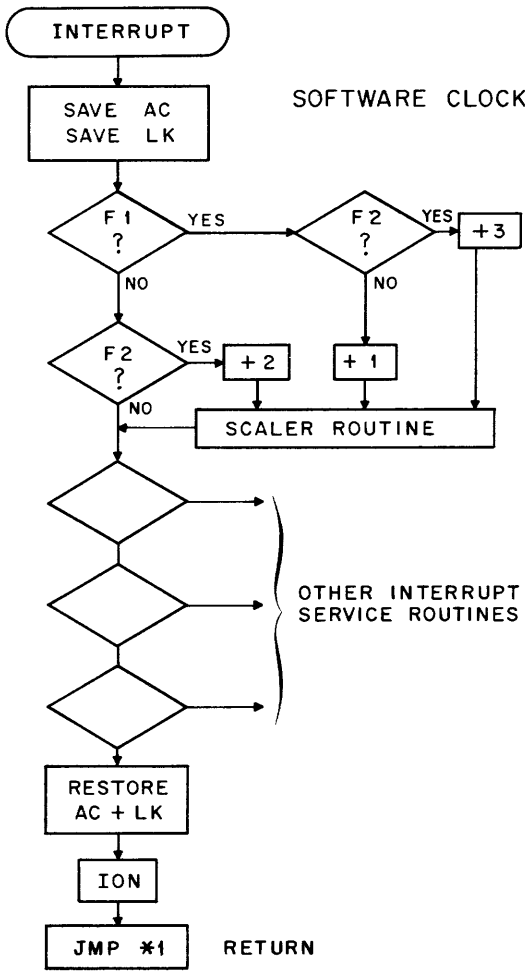


Figure 7 Software Clock

Several devlopments are undertaken for experiments to be performed in the future. A special effort is aimed at the implementation of formal language. A formal language may be built of a set of predetermined sentence elements. All the sentence elements are grouped in columns and each column may have from 10 to 50 sentence elements. Elements of each column can be used according to the rules set forth by a sentence pattern. A simple example is the sentence consisting of elements from the columns 1-6-3-4-5-12-14-11. (See Figure 9 for formal language.)

The computer can now do several things. It may store in its memory all available sentence elements. The subject, who has a list of all possible sentences in front of him, can select certain code numbers which result in an assembly of a sentence of his message. The computer can check if the subject kept the rules of the sentence pattern or syntax.

The latest development requires only a reasonable knowledge of sentence elements to be used. The subject need only type in the beginning of each sentence, and as soon as the computer recognizes the sentence element it finishes it automatically. When the computer, on the other hand, finds that the sentence element doesn't exist, it may either add the new sentence element in the proper column in its memory or it may reject the information from the subject and let the subject know about the rejection. When the subject violates the rules of the sentence pattern, the computer may not only reject the next sentence

element, but may make suggestions about what to do. Another development concerns gathering sentence elements by the system. In other words, the system starts out with a completely blank memory about what sentences to use. Research assistants then play the game, conscientiously adding "experience" to the system. Experience, of course, can be trimmed after a review of the experimenter.

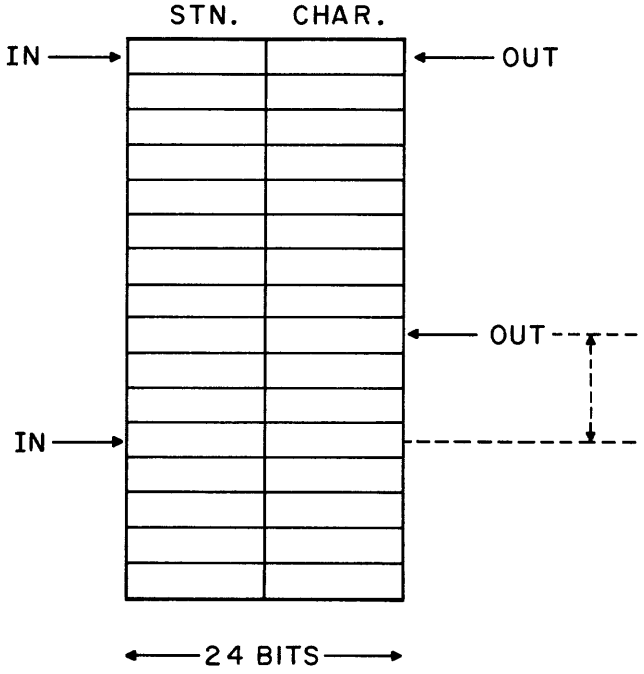
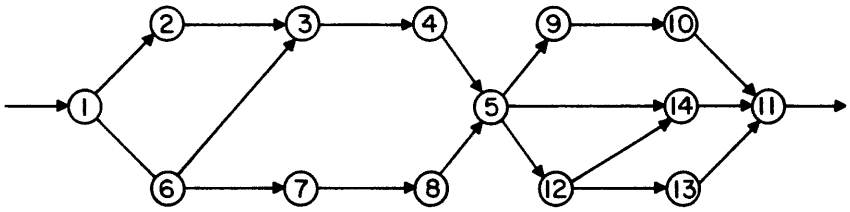


Figure 8 Input Buffer



CONNECTION RULES FOR SENTENCE ELEMENTS IN A FORMAL LANGUAGE.

Figure 9 Pattern Diagram

Another development consists of the simulation of people. No more than five subjects will play a game (one subject with four simulated subjects, present as "robots" in the system). Here we hope to use a similar approach of teaching the system how to respond to the message as before, in formal language. It is obvious that many very difficult problems lie ahead; one can teach a system how to respond to certain assemblies of message elements, but it would be desirable to let the system review previous parts of the conversation. The question is then how far back, let alone how. One can imagine a situation in which the subject communicates to the computer program and, in case the computer program does not know the answer, it asks for help from the experimenter who uses a separate Teletype. The computer could store each response made by the experimenter and relate it to the subject message.

THE LEARNING RESEARCH AND DEVELOPMENT CENTER'S COMPUTER ASSISTED LABORATORY *

Ronald G. Ragsdale

University of Pittsburgh
Pittsburgh, Pennsylvania

ABSTRACT

This paper describes the operation and planned applications of a computer-assisted laboratory for social science research. The laboratory centers around an 8K PDP-7 and its special peripheral equipment, with most of the system already in operation. Special devices include random-access audio and video, graphical input, touch-sensitive and block-manipulation inputs. The control programs for these devices are incorporated in an executive system which permits simultaneous operation of six student stations. The system may be used for presenting instructional material or for conducting psychological experiments.

In April of 1964, the Cooperative Research Branch of the United States Office of Education established a research and development center at the University of Pittsburgh as part of the Office of Education's program designed to concentrate major effort in various areas in education. The Center at the University of Pittsburgh, called the Learning Research and Development Center (LRDC), directs its activities to design and development of instructional practices on the basis of experimental research on learning.

One activity has been construction of a computer-assisted laboratory as a first step of a project on computer-assisted instruction. The Computer-Assisted Instruction Project has two principal objectives which guide the scope of the effort undertaken within this group. The first objective is to provide those facilities and services needed to support the research and development effort of experimental psychologists and others in the field of instructional technology. The facilities include apparatus and controls used in learning experiments which use computer-related equipment. The group provides engineering and programming assistance in design and conduct of experiments. This service, primarily for the Center staff, may be used by other faculty members.

The second objective of this group is to conduct experimental work in the development of computer-based instructional systems. Examples of such work include the development of supervisory programs for controlling many independently operating stations; the development of languages that educators can use for subject matter programs on the computer; and the development of student stations that will provide a high degree of interaction between the student and the subject matter.

* The research and development reported herein was performed pursuant to a contract with the United States Office of Education, Department of Health, Education, and Welfare under the provisions of the Cooperative Research Program.

HARDWARE

The main piece of hardware involved in this project is a Digital Equipment Corporation PDP-7 computer, installed in mid-June of this year. In addition to the standard PDP-7 configuration, this computer has 8,192 words of core memory, 16 levels of automatic priority interrupts, the extended arithmetic element, 100 card-per-minute reader, variable time clock with speeds up to 1000 cycles per second, 2 output relay buffers and terminals for connecting student devices to the computer. The core memory will soon be increased to 16,384 words, with the additional 8,192 already on order. It is anticipated that magnetic drum storage will soon be added to the system as well.

In addition to the computer room, shop area, and an auxiliary equipment room, there are eight separate laboratory areas which average about 180 square feet. The labs range in size from 140 square feet to 285 square feet, with two of the larger labs having observation areas separated from the lab room by one-way glass.

Electrical ducts connect each laboratory area to the computer room. This allows a great deal of flexibility in assigning display and response equipment to the student station. The most basic device is the keyboard, which is

a modified Type 33 Teletype keyboard. Because no printer is associated with the keyboard, this function is usually served by an oscilloscope screen (a Tektronix RM 564 connected with a Type 34 interface). This scope can be operated in stored mode in which information on the screen is preserved and does not have to be refreshed. When the scope is in dynamic mode (selected under computer control) it may be used with a light pen.

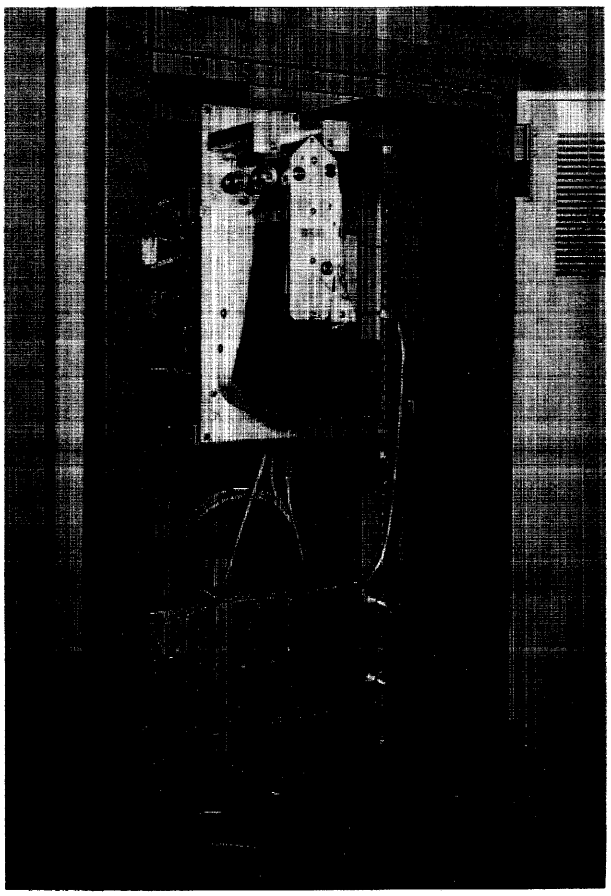


Figure 1 The Random Access Audio-Unit Built by Westinghouse

Another basic device is the audio speaker (or head set). Audio information is stored on loops of 6-inch wide magnetic tape. Each loop has 128 tracks, with 16 play and record heads each servicing 8 tracks. The size of the loop is variable, but at present each holds 1,024 seconds of information with each 1-second block individually addressable. Although it is not a standard item, there may also be a microphone at the student station so that he can record on certain areas of the tape reserved for this purpose (through software). Basic equipment for a student station consists of a keyboard, a cathode ray tube with light pen, and an audio speaker or head set (see Figure 2).

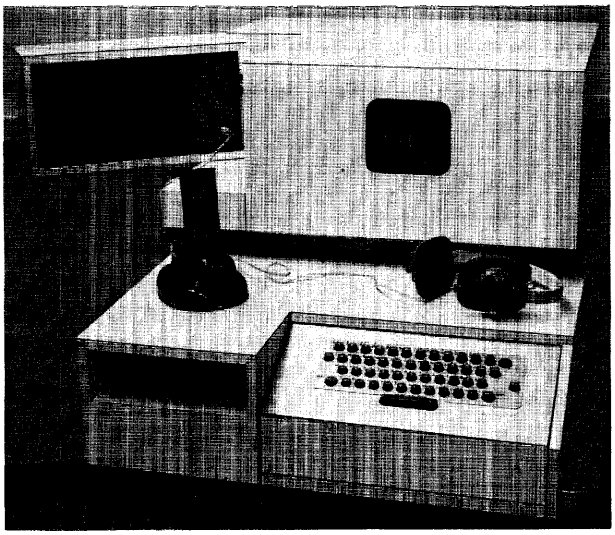


Figure 2 The Basic Student Station with Keyboard, Scope, Speaker, Earphones, and Microphone

In addition to the basic student station devices, a number of special devices, some experimental, exists. One special device already in operation is the "touch-sensitive display" (Figure 3). A random access slide projector is focused on a back lighted screen which displays the information for the student. Many fine wires within the screen allow detection of an object, such as a finger or pointer, striking the screen. Detection of such a response also specifies the location of the response upon the screen. Since the projector and the screen are both under computer control, the presentation of visual stimuli may be sequenced according to the position of the preceding response(s).

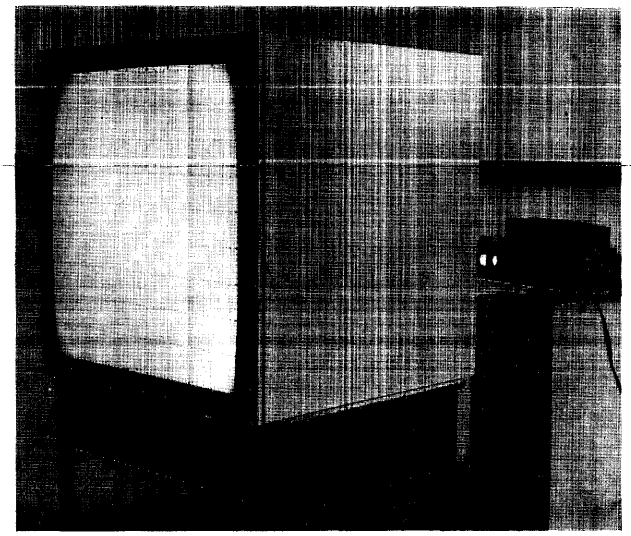


Figure 3 The Touch Sensitive Display with Random Access Slide Projector

Other devices under development include graphic tablets like the "RAND tablet" and "manipulation boards." The graphic tablet is a device through which a student may input graphical information directly to the computer with an electronic pencil (Figure 4). The tablet, used in conjunction with appropriate diagnostic routines, permits the teaching of printing and drafting, to name only one example, in a manner which allows easy detection of student errors.

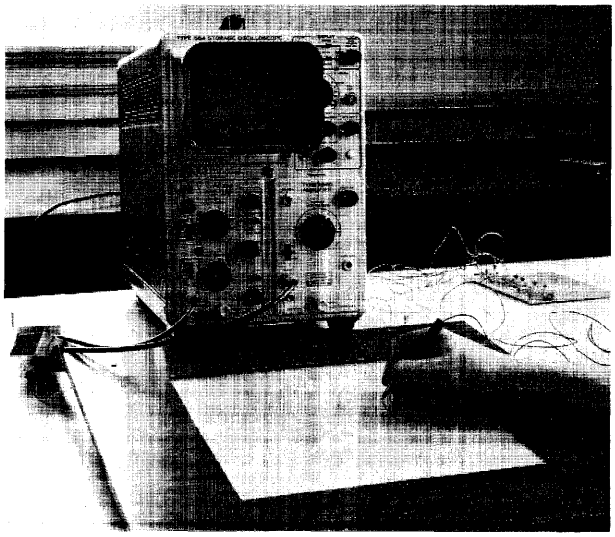


Figure 4 The Graphical Input Surface

At the RAND Corporation, the graphic input tablet is used for debugging programs. A scope displays the contents of several standard locations plus some memory location. The programmer can search memory upward or downward by pointing the electronic pencil at the appropriate spot. He can change the contents of a displayed memory location by merely writing over the display with the electronic pencil.

Figure 5 shows the manipulation board which detects the placement of objects on its surface and relays the corresponding bit pattern to the computer. The problem here is very similar to that of the RAND tablet in that a bit pattern must be recognized. However, in this case, the set of objects such as blocks representing different number quantities, can be restricted to those easily identifiable. With such a restriction, the set of objects can be identified both as to item and location upon the board. This device should be particularly effective in working with young children, or the mentally retarded, since it puts very little demand on the student insofar as the structure of the response is concerned.

A device for magnetic storage of video information is planned for the near future which will permit storage of approximately 500 pictures combining video camera or computer-plotted output. The selection of pictures will be under computer control and additional points may be plotted on the selected picture at any time. This system will also permit the use of a light pen.

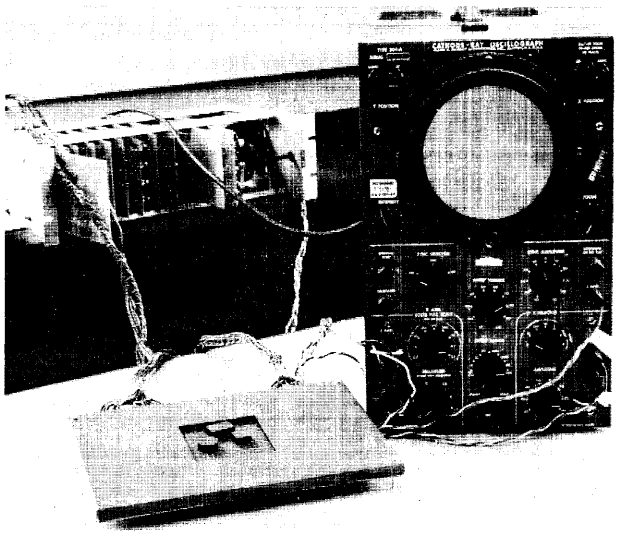


Figure 5 A Small Developmental Model of the Manipulation Board

SOFTWARE

The present software system, still in the final debugging state, consists of a set of control programs for each of the various devices as well as an executive system to control the sequencing of jobs, timing, memory allocation, etc.

All teaching or experimental programs being contemplated are of a type usually waiting for a student response or for some time delay to run out. When this happens, the program returns control to the scheduler and some other job is initiated. In general, this system is capable of servicing six devices of each type, although at present only one of each is in use.

Most of the 16 priority levels are already used by the system. The 60 cycle and 1000 cycle clocks each use 2 levels. Keyboards, light pens, touch-sensitive displays, microphones, and audio tapes also require priority levels based upon the speed of the response they demand. In addition, the executive routine responds to a priority level which may be activated through software.

This interrupt, which has the lowest priority level, is set whenever an interrupt signals the end of some job's suspension. Thus, when all other interrupts have been processed, the scheduler regains control and can reinitiate this job.

Programming of learning experiments, teaching routines, etc., can be accomplished only by machine language at

present. A large part of the coding would, of course, be in the form of subroutine calling sequences which reference the control programs. This type of programming may be more difficult and time consuming, but it also offers a maximum in flexibility.

One major goal of the Computer-Assisted Instruction Project is the implementation of a language which educators can easily use for programming subject matter on the computer. Although the final goal in this area is a compiler which accepts lessons written in some behaviorally oriented language and translates them into the proper codes, several steps are involved. The first, and probably the most important, is the definition of terms which denote particular instructional subsequences, etc. From this foundation, one can proceed until a final compiler is defined and operating.

INDIVIDUALIZED INSTRUCTION

A project which is planned to eventually merge with the Computer-Assisted Laboratory is now being carried out in a suburban Pittsburgh elementary school. In this school, students are allowed to progress at their own rate through curricula in mathematics, reading and science, as they might in a computer-assisted classroom. Each unit of instruction is composed of skills which must be mastered with appropriate instructional materials and progress and quality control tests provided for each skill to be learned. There is a considerable library of instructional sequences and relevant information provided by the materials for a particular skill, and the teacher must prescribe those materials which she feels are most appropriate. It is clear that a curriculum of this type lends itself well to computer instruction, the major difficulty being simulation of the teacher's decision process.

LEARNING EXPERIMENTS

The development of a library of programs relevant to the running of learning experiments is less structured. There are certain basic paradigms which provide a basis for such a library, but the need is for more flexibility, in addition to implementing the old routines.

There are two major objectives in the learning experiment area: experimental design and real-time data analysis. In the design and control of learning experiments, programs would assign subjects to groups on the basis of certain control variables; they would monitor information such as "anxiety level," for example, and modify it if necessary; and they would equate the subject's performance on a certain task prior to the application of the experimental treatment by the assignment of appropriate training or by statistical adjustment. These programs would perform those tasks usually requiring lengthy subject selection prior to the experiment or the conduct of preliminary base-line pilot studies.

Closely related to these programs are programs for real-time data analysis which provide data for the control and design section. This would permit use of complex criterion measures in equating the performance of subjects, to mention only one example. This type of data analysis would be especially helpful in a pilot study, since one could start out with a large number of variables and as the analysis indicated, discard those obviously relevant or obviously irrelevant, and concentrate on the more marginal variables.

Eventually the system might assist in the selection of the variables of interest and the assignment of the experimental treatments. The system would also analyze the data, and terminate the experiment when sufficient precision was attained.

TIME SHARING THE PDP-6

Thomas N. Hastings

Digital Equipment Corporation
Maynard, Massachusetts

ABSTRACT

The PDP-6 time-sharing effort has been undertaken in two distinct phases. The first phase is a multiprogramming system in which all programs are resident in core and are controlled by users at Teletype consoles. Up to 256K of memory may be used with I/O devices which include a card reader, a line printer, 8 IBM compatible magnetic tapes, 8 addressable magnetic tapes (DECtapes), a paper tape reader, a paper punch, and up to 64 Teletypes. The addressable DECtapes permit storage and retrieval of programs and data by file name. All I/O operation is overlapped with computation first for the user program requesting the input/output, and secondly, for other user programs. In this way, the central processor and the I/O devices are kept as busy as possible. All user programs run relocated and protected from each other. Nine phase-one systems have been in operation in the field since May, 1965.

The second phase extends the multiprogramming capabilities by providing a fast secondary storage device for program swapping by the monitor. Thus, more user programs may be run simultaneously than can fit into core at the same time. This I/O device is also available for storage of user programs and data. Since device independence will be maintained, user programs will not need to be modified to make use of the phase-two system.

Most new medium and large scale computers come equipped with time sharing software of one form or another. The PDP-6 is no exception. Twelve time-shared installations have been in operation all over the world since May 1965.

This paper describes four phases of DEC's PDP-6 time-sharing effort:

1. The Operational Time-Sharing System
2. The Real-Time Option
3. The Future Swapping System
4. Time-Sharing Monitor Debugging Techniques

OPERATIONAL TIME-SHARING SYSTEM

Before describing the Operational Time-Sharing System, it is appropriate to describe the PDP-6 as a general purpose computer. Because so much programming effort is required to produce a time sharing system, it is important to pick a machine which is a powerful general-purpose computer in its own right. The PDP-6 is a 36-bit word, 18-bit, single-address machine. It has 16 accumulators, 15 of which also serve as index registers. These 16 accumulators also appear to the program as the first 16 locations of memory. All instructions have the same format and specify an accumulator, an index register, and a memory location. Hence, no special-purpose registers are necessary for program use. The machine has a large number of well organized instructions (353), which are easy to learn. Since the same instructions which operate on the accumulators operate on the index registers and memory locations, the effective number of instructions is two to three times 353. The instruction set includes floating point, Boolean, bit testing, push-down list, and byte manipulating instructions.

Both instruction execution and memory references are performed asynchronously so that different speed components may be used in the same machine. This means that as faster components are developed, they may replace the slower ones without requiring replacement of the entire machine. Also, each customer has a wider choice of components and may upgrade his particular system as he wishes.

The priority interrupt system for handling input/output program interrupts is a key design feature of the PDP-6. Without it, handling many I/O devices by the monitor would be almost impossible. Each device is assigned to

one of seven priority channels under program control. More than one device may be assigned to the same priority channel. While a device is being serviced on one channel, all other interrupts on that channel and lower priority channels are automatically disabled. When the channel is dismissed, any temporarily disabled channels are automatically serviced. Only interrupts on higher priority channels are accepted while servicing lower priority channels. Thus, the priority channel scheme provides a convenient way of disabling some devices while servicing others. Low speed devices, requiring a good deal of computation, are assigned low priority. High speed devices, requiring little computation, are assigned high priority.

A wide variety of I/O devices may be attached to the PDP-6. They include a card reader, line printer, 8 industry-compatible magnetic tapes, 8 addressable magnetic tapes called DECtapes, a paper tape reader, a paper tape punch, up to 64 Teletypes, and a display scope.

The PDP-6 monitor is written completely in machine language. This needs some defense since it runs counter to current trends in monitor writing. Instead of designing a higher level language well suited for writing monitors, we feel the PDP-6 machine language offers most of the advantages of a higher level language. The reasons many people prefer compiler language to machine language are:

1. Machine language is not machine-independent, so that a program written on one machine cannot be run on another.
2. Machine language is harder to learn.
3. Machine language programs are harder to understand mainly because data items held in accumulators are not named.
4. Machine language programs require 2 or 3 times as many statements as equivalent compiler programs.
5. Machine language programs take longer to write.
6. Machine language programs take longer to debug.

The last five objections have been partly eliminated on the PDP-6. The PDP-6 instruction set is readily learned and remembered after a day or two of study. The programmer can name the multiple accumulators, helping him to identify the information in them. After coding in machine language on the PDP-6, many people find they prefer it to FORTRAN, especially for nonnumerical computations. The rich order code of the PDP-6 means shorter programs. Finally, the machine language debugging programs for the PDP-6 have been developed from long experience with smaller on-line computer debugging aids and provide a quick and efficient method for program debugging.

Several of our customers have modified parts of the monitor to suit their own special purposes. They have found this relatively easy to do, thus vindicating the decision to use machine language.

Software resources of the time-sharing system include a growing collection of Commonly Used System Programs called CUSPs. These CUSPs include FORTRAN II (FORTRAN IV will be added in 1966), a macro assembler, a relocating linking loader, a text editor, a desk calculator, and a symbolic debugging program (DDT). Another called PIP (Peripheral Interchange Program) transfers data from one device to any other, eliminating the need for off-line satellite computers in card-to-tape and tape-to-line printer operations.

The monitor itself is a collection of reentrant subroutines which reside permanently in 4000 to 6000 words of memory and which provide overall coordination and control of the total operating system. These monitor subroutines are called either by user programs or in response to commands typed into the monitor by users at their consoles. The monitor loads several user programs into different protected areas of memory in response to Teletype commands. These user programs may be user-written or CUSP programs. Thus, the monitor is context-free in the sense that it can run any program written in any language. A relocation register allows the monitor to place these programs anywhere and move them at any time. The monitor schedules jobs to run using a modified round-robin queuing discipline. Higher priority is given to programs which have just finished waiting for an I/O device to complete operation. Switching is frequent enough so that all programs appear to run simultaneously. This job scheduling is performed by a routine assigned to the lowest priority channel. Monitor commands typed in by users are also processed on this lowest priority channel. User programs run whenever no interrupts are in progress and so have the lowest priority of all. This priority level, on which the machine spends most of its time, is called user level.

In addition to controlling the user programs, the monitor also controls the I/O devices. All devices appear identical to the user's program, thus simplifying program coding. A second advantage of device independence is that a user may substitute one device for another at run time by simply typing a command to the monitor. The monitor makes full use of the PDP-6 hardware priority interrupt system combined with multiple ring buffering to overlap input/output operations with computation. The monitor achieves further system utilization through multiprogramming. Thus, if any program does catch up with its I/O devices, the monitor automatically switches control to another program.

REAL TIME

Although the phrase "real-time program" implies total preemption of the computer's time, many such programs

in fact require far less than 100 percent of the machine's computing capacity. A typical application is an airplane flight simulation using analog equipment which must be set and sampled at frequent intervals in real time. For these applications, a typical program might require repetition every 100 msec for a burst of up to 25 msec. Once the computation has begun, it may not be interrupted since a one-to-one relationship between program execution time and real time must be maintained.

Unfortunately, the monitor overhead time for scheduling programs and for processing console commands blocks out too much time from the user level to be satisfactory for real-time programs. In fact, some of the longer interrupt service routines on the lower priority channels also cannot be tolerated. This problem required the modification of the PDP-6 monitor to provide real-time capabilities. Real-time user programs are assigned to a fairly high priority channel instead of user level. Thus, when they run, all lower priority channels are automatically prevented from causing interrupts.* This modified monitor is currently operating at the United Aircraft Corporation.

FUTURE DEVELOPMENTS

At present the Time-Sharing Monitor does not handle I/O for disc or drum. Also, it does not swap programs in and out of core when the total demand for core exceeds the size of physical core memory. The second phase of our monitor development is to include a disc or a drum for both data storage and program swapping.

In setting down the objectives of our swapping system, we benefited a great deal from the experience of Project MAC at MIT. Our most important objective is to continue the concept of device-independence developed in our present operating system. Not only does this mean that all existing programs will be able to operate using the disc without requiring any recoding, but also that much of the coding for handling I/O devices has already been written and debugged.

Each user will be able to read and write files of data on the disc by name. The monitor will keep track of the physical location of users files.

Because disc and drum technology is undergoing such rapid development, it is impossible to anticipate the I/O interface of future devices. Consequently, the routine which is concerned with interfacing to the device has been kept as simple as possible making it easy to write a new interface routine for a new device.

One big advantage of a time-sharing system is the ability to share programs and data. Thus, the disc file system will allow one user to read or modify a second user's files provided that the second user has given his permission. The programming for granting access to files will also be device-independent so that it may be shared by all file-structured devices.

The technique for representing a linear file on a random access device like a disc was studied at great length. It was desired to use a scheme which would be efficient for relative reading and overwriting as well as the more common ring-buffered and unbuffered linear reading and writing. This included the ability to optimize the operation of the I/O device with respect to minimizing disc arm seek time.

The program swapping part of the monitor relies heavily on the disc routines. Fortunately, the monitor has been written so that it uses absolute addresses only when I/O devices are actively transmitting data. Thus, a program can be swapped out of core at any time devices are not active. It can then be brought back into core in a different absolute location at a later time.

DEBUGGING TIME-SHARING MONITORS

One of the interesting problems of writing time-sharing monitors is developing effective techniques for debugging them. Octal core dumps seem completely useless. Instead, a special version of the symbolic machine language debugging program, DDT is being used. This version does its own Teletype I/O on the operator's console after saving the state of all the other I/O devices. DDT lets the programmer examine and modify the monitor using the same source symbols used in writing the monitor program. DDT also allows a programmer to insert up to eight break points in the monitor. Control is automatically transferred to DDT when any of the break points are encountered. The programmer may examine the monitor and then proceed with execution from the break point. When a programming error is discovered, a patch is made simply and quickly using DDT. The patched version is then saved, a process which takes less than a minute using DECtape. Debugging can then continue.

Since experience has shown that monitors approach the completely debugged state asymtotically rather than suddenly, one cannot actually say that a monitor does or does not contain bugs. For this reason, we divide time-sharing time into three categories:

1. Monitor check-out
2. Remedial time sharing
3. Open time sharing

During monitor check-out, the system programmers are in complete control of the machine using DDT with many break points. Obviously, the monitor cannot be used for useful time sharing when it is full of break points. So, it is debugged until it appears to run successfully. However, the real truth comes when many users get on

*For further information see the article on pages 28-29 of the October, 1965 issue of "Computers and Automation" entitled "Real-Time Computing While Time-Sharing."

simultaneously. Remedial time sharing serves this purpose but is strictly a use-at-one's-own-risk proposition. Furthermore, whenever a bug occurs in the monitor, the system programmers take over the machine to find out what went wrong. Open time sharing is for doing useful work and employs a monitor which has run successfully for a week or more. If a problem occurs, its symptoms are written down and the bug is removed at a later time. Every few weeks the complete monitor is edited incorporating the patches and is reloaded. Finally, after two to four months of this procedure, the monitor is ready for distribution to customers.

A DATA ACQUISITION SYSTEM FOR SUBMARINE WEAPON SYSTEM EVALUATION UTILIZING A PDP-8

Robert H. Bowerman

U.S. Naval Underwater Ordnance Station
Newport, Rhode Island

ABSTRACT

This paper traces the data recording system requirements for shipboard weapon system tests in terms of increased complexity. A newly designed system, presently under procurement, is described in terms of the above requirements and also with regard to PDP-8 hardware utilized. In addition to the basic computer and magnetic tape system (repackaged for portability and environmental protection), there is a high-speed 20-character line printer and a set of multiplexed synchro-to-digital, and voltage-to-digital conversion modules to receive the shipboard signals. All components of the system are designed to function as PDP-8 peripheral devices, allowing expansion and modification by program changes rather than by hardware redesign.

The place of this data system in the overall information gathering operation is described, and the division of tasks between the shipboard data system and the shore-based reduction programming is developed.

The system described has shown advantages of cost and flexibility, and is expected to be a powerful tool in future weapon system evaluations.

INTRODUCTION

The U.S. Naval Underwater Ordnance Station is a research and development activity of the Bureau of Naval Weapons with responsibilities including torpedoes and fire control systems.

In order to verify performance and accept these hardware systems for procurement and general fleet use, it is necessary to provide a shipboard instrumentation system. This device should be equipped to measure and record all performance parameters in a suitable manner for efficient entry into a large computer system capable of coordinating this data with range, target, and weapon information into a single record.

Similar tasks have existed with every weapon system ever built, but each successive system has demanded greater capabilities from the instrumentation.

HISTORY

Table 1 indicates some of the advances in the last 20 years in terms of two submarine weapons systems, their torpedoes, and their data transmissions. The cost figures are indicative of the basic computational elements and the basic weapons, and do not include installation and support costs. For this reason they are given as comparative approximations. However, the figures do indicate the relative complexity of the hardware and the economic factors involved.

Recording procedures for earlier systems were primarily photographic in nature and consisted of a camera focused on the display panels of the Torpedo Data Computer. Every quantity involved was indicated on a dial, even time and the automatically transmitted inputs of course and speed. Data reduction consisted of reading the dials from the photographs and comparing the setting of the weapon, as computed, with its measured track on the test range or the function settings in the weapon as recovered.

Since several firing vessels might be available and weapons can be fired repeatedly, this system used a relatively large sample size to verify performance.

To update from 1946 and to the present, a brief description of some of the major advances in fire control system technology will be described in terms of their impact on data systems.

Since most internal and external transmissions in old computers were mechanical, no attempt was made to measure any of the intermediate quantities in the computing loop except those that were already brought to dials.

TABLE 1 SUBMARINE WEAPON SYSTEMS

	1946	1966
SYSTEM	"TDC" (TORPEDO DATA COMPUTER)	"MK 113"
COMPUTATION METHOD	MECHANICAL ANALOG	ELECTROMECHANICAL ANALOG AND DIGITAL
DATA TRANSMISSION TO WEAPON	MECHANICAL	ELECTRICAL (ANALOG AND PULSE)
INPUT METHOD	ALL HAND CRANKS EXCEPT OWN COURSE AND SPEED	ALL OWN SHIP AND TARGET DATA RECEIVED ELECTRICALLY
APPROX SYSTEM COST (RELATIVE)	11	180
APPROX WEAPON COST (RELATIVE)	1	9

As more and more electrical data transmission and computing elements were utilized, the need for monitoring these became evident. Two solutions appeared: servo-driven dials placed in the field of the camera, or recording oscillographs. Oscillographs became the basic weapon-borne data package; the camera was still the shipboard standby.

FIRST USE OF DIGITAL TECHNIQUES

These techniques were basic until about 1960, when the sheer volume of photographic data prevented effective and timely reduction of data. Among the contributions to this growth which must be considered are system complexity, dynamic problems requiring repetitive measurements, and the use of computing equipment for data reduction. A significant cause was the tendency to perform several independent experiments in some runs to reduce cost and weapon requirements.

An immediate solution was to add shaft encoders to the servo-driven repeater dials and use punched tape as an output medium. Photographic techniques were still used as a back-up mode and for "quick-look" verification. The additional complexity of this system is greater than implied, as all quantities must be brought to servo modules in order to drive encoders, even though the quantity might be on an equipment dial. Some means of data storage was also required to generate a data block on paper tape from simultaneous measurements. A further complexity was the requirement to convert all data to special coding compatible with the particular computer utilized for data reduction.

Several such systems have been built for various naval laboratories. These systems differ mainly in method of storage and output medium, and have been successfully used in evaluation of current weapons systems.

ANTICIPATED REQUIREMENTS

In late 1964, requirements were defined for a data system to be used in evaluation of a new weapon system. This data system was to be part of a three-level hierarchy as shown in Figure 1.

Runs will be made on a range facility, such as the AUTEC (Atlantic Undersea Test and Evaluation Center) range in the Bahamas, capable of measuring and recording pertinent performance data. This range also includes a time synchronization system. Each of the bodies being tracked also includes its own instrumentation system and is synchronized in time by this means.

These records will be reduced individually to develop the individual tracks as the second level of data processing, and these results will be correlated with the range tracking results to develop actual error distances and to reconstruct the overall problem geometry for final analysis. The performance characteristics of the shipboard system required are noted in Table 2.

Originally, modification of existing units was considered. However, examination of the requirements, capability, and availability of existing systems found this impractical.

The first approach followed previous systems: convert to digital form, store, and record in computer-compatible

form. It included several significant advances including solid-state synchro-to-digital conversion, core storage, magnetic tape output, and sufficient arithmetic capability to scale input quantities for display on a direct readout panel. This panel could be photographed for backup, and for verification.

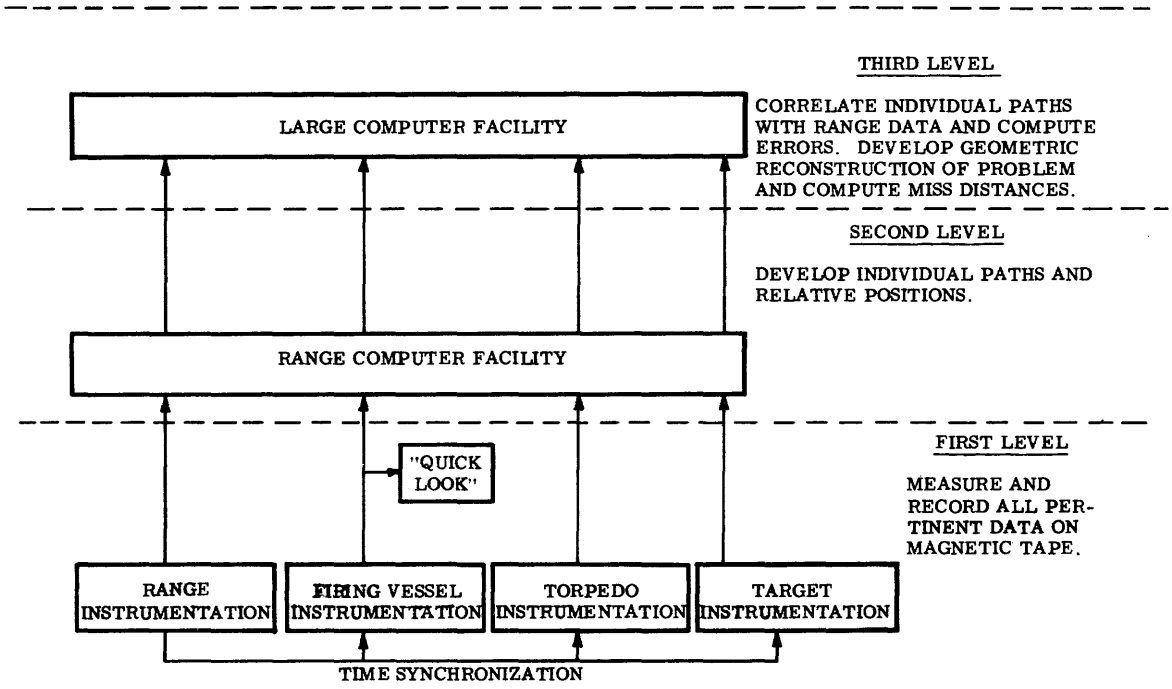


Figure 1 Data Processing Heirarchy

TABLE 2 DIGITAL DATA ACQUISITION SYSTEM, PERFORMANCE CHARACTERISTICS

Synchro Inputs

Up to 32 single speed channels
0.1 accuracy single speed
0.01 accuracy multiple speed
1.0 yard accuracy on multiple speed range functions

AC/DC Scaled Voltage Inputs

Up to 16 ac or dc channels
0.05% ratio accuracy

Mark Signal Inputs

8 mark signal channels (includes tone marks)
0.1-second maximum mark signal timing error

Sample Times

All functions sampled within 0.001 second
Maximum sample rate 10 times per second

Record Rates

Magnetic tape continuous at maximum sample rate
Print out up to 40-20 character lines per second

SMALL COMPUTER CONSIDERATIONS

Upon examination, this proposed system contained all the basic elements of a small computer, except its flexibility. In addition, it required a significant amount of design effort and was a wired program unit which would not lend itself to reprogramming for different types of data acquisition without extensive wiring changes.

As the problem was further examined, other advantages became evident. Output magnetic tape transport and control became off-the-shelf units. The input control panel became the computer input typewriter. The output display panel could be reduced to a set of three decimal readouts (under program control) using a small line printer which prints out input quantities at a reasonable rate, and which also eliminates the photographic

backup record. Except the actual input circuits interfacing with the shipboard signals and the multiplexing circuitry, the entire system became standard computer components or units that could be readily constructed from modules rather than special hardware.

Concurrently with this analysis the PDP-8 was announced, and as it fulfilled the basic requirements of packaging--subassemblies able to pass through a 25" diameter hatch and sufficiently rugged to withstand the noncombat shipboard environment--it was examined in terms of programming and input/output circuitry.

MINIMUM PROGRAM REQUIREMENTS

The basic program for the system requires that 64 words be received from the input signal conditioning and conversion circuits and be stored ten times per second, and that once per second this 640-word data block be written on magnetic tape in pure binary form, with a time code group for identification. In addition, the program must provide a record of all input quantities on the line printer, at a one-per-minute rate. Subsidiary program tasks include typewriter inputs to change sample times for the printer and input circuits, typewriter outputs of special event codes, defects, and entry verification.

Programming shifts several tasks to the second level of data processing to reduce execution time. Typically, synchro signals are sent in groups of two or three with differing scale factors. This provides high resolution as well as full scale values, but requires the signals to be scaled and merged to a single value. For this system all synchro signals are considered singly, and are converted and recorded as normalized 12-bit signals. The scaling and merging task is done in the second level data processing from magnetic tape records.

The task is not totally avoided, however, because the display panel and printer require directly indicated values of the functions, but at a relatively low rate. Recording voltages in a comparable manner to ratios was a second saving. This allows maximum resolution and in most cases, better accuracy, as the ratios are generally more significant than actual values of signals within the weapon system computers. Where absolute values are required, a fixed reference is provided.

SYSTEM OPERATION

The input/output circuit requirements are shown in the system block diagram Figure 2. The left half represents the special hardware, while the right portion consists primarily of standard computer elements, repackaged for submarine handling and installation.

The basic sequence of events is initiated by an external time input pulse, synchronizing the system with the overall range clock and causing the input modules to read all incoming signals, convert them to dc levels, and store them in capacitor trap and hold gates. These modules are capable of conditioning and storing 16 signals each. Various types of modules are utilized depending upon the relative number of signals of each type. The modules are interchangeable to provide system flexibility, although the initial configuration will be limited to two synchro modules, one voltage module, and one mark signal module. The system will accept additional modules with the presently planned programming.

The actual multiplexing is done in two stages: by selecting one of the 16 inputs in all input modules and by gating the outputs from the analog-to-digital conversion registers to the computer sequentially. This provides sufficient time for the conversion process. Upon completion of the reading from all four registers, the next of the 16 lines is selected and the process repeated. After 10 blocks of 64 inputs have been read and stored, a single data block is written on the magnetic tape transport. Since the data break capability of the PDP-8 is utilized, this can be done in the interval following the last conversion of the tenth group of 64 words, and before the next time pulse.

LINE PRINTER AND DISPLAY PANEL

The display panel and line printer are programmed to receive selected scaled and merged data as designated by typeins on the on-line keyboard. The output rate is limited to one or two per minute on the line printer and slightly faster on the display. Computation is done in the off-time following conversion and is stopped upon receipt of the next time pulse.

The line printer is presently a 20-character per line, 40-line per second, numeric printer with DEC interfacing logic. This printer would be capable of much faster rates than the one or two data sets per minute proposed, but the scaling and merging load on the computer and the amount of paper used justify the restriction. An alternate printer is under consideration that will provide 32 character alphanumeric outputs at 100 lines per second, on photo sensitive paper. This printer will allow a more descriptive alphanumeric format and a quieter system.

ANALOG-TO-DIGITAL CIRCUITRY

Figure 3 shows another item of special interest, the analog-to-digital circuitry utilized in the input modules. These circuits, designed by the Librascope Group of General Precision Inc., are capable of performing conversion in 60 μsec. The incoming synchro signal is comparable to three-phase ac signal, but with the angle representing a shaft position rather than time. Thus we have three ac signals of magnitudes corresponding to the cosines of the transmitted angle, and the angle plus and minus 120°. These signals are converted to sine and cosine signals by a transformer connection analogous to the Scott Tee circuit used for three phase to two phase

power conversion. These signals are sampled at the peaks of the reference supply voltage, rectified, and signed in accordance with the instantaneous values. These signals are then strobed into the trap and hold gate and sampled by the time pulse. These signed functions are multiplexed into an absolute circuit and identified as to the quadrant by the signs. This data appears as the highest order pair of bits of the accumulator input word.

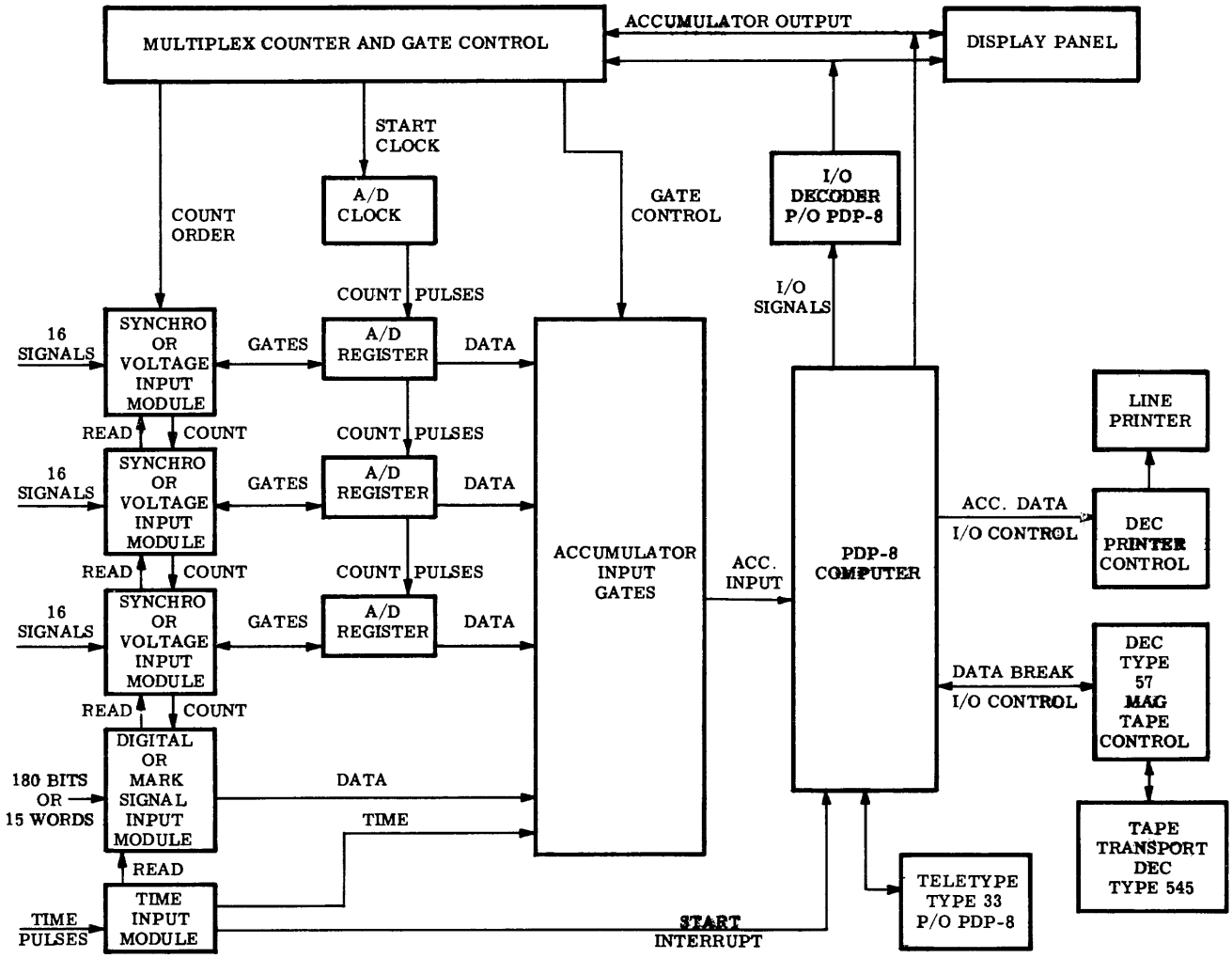


Figure 2 Nuos Digital Data Acquisition System
(Based on PDP-8 Computer)

The lower order 10 bits are generated by the gate control register and buffer of a 10 bit A/D converter. Instead of a fixed reference to the ladder, the sum of the absolute values of the sine and cosine signals is used, and the output of the ladder is compared with the sine signal by the comparator. Upon synchronization, the buffer contains a 10-bit representation of the function:

$$\frac{\sin \theta}{\sin \theta + \cos \theta}$$

This function has a range of 0 to 1 over 90° quadrant and readily converts to the actual angle in the second level data reduction or in the PDP-8 for the display panel or printer. The voltage ratio conversion utilizes the same basic elements but applies the reference voltage and the signal to the D/A circuits in the usual manner. The overall system is packaged into 14 containers of which Figure 4 is typical. This particular section, the display panel, contains both standard DEC FLIP CHIP modules and special Librascope circuitry.

ADDITIONAL CAPABILITIES

The approach described above was verified by a paper design of the special circuits required and by examination of the standard components from a repackaging

standpoint. Upon completion of the design several additional capabilities were realized in the system at no additional cost except programming. Most significant of these was the ability to verify the data record tape on an off-line basis immediately following a run. This is not as efficient as on a larger machine, but the possibility of determining the validity of a run while still in the range area can conceivably allow runs to be repeated or eliminated without the delay imposed by use of a remotely located computer system.

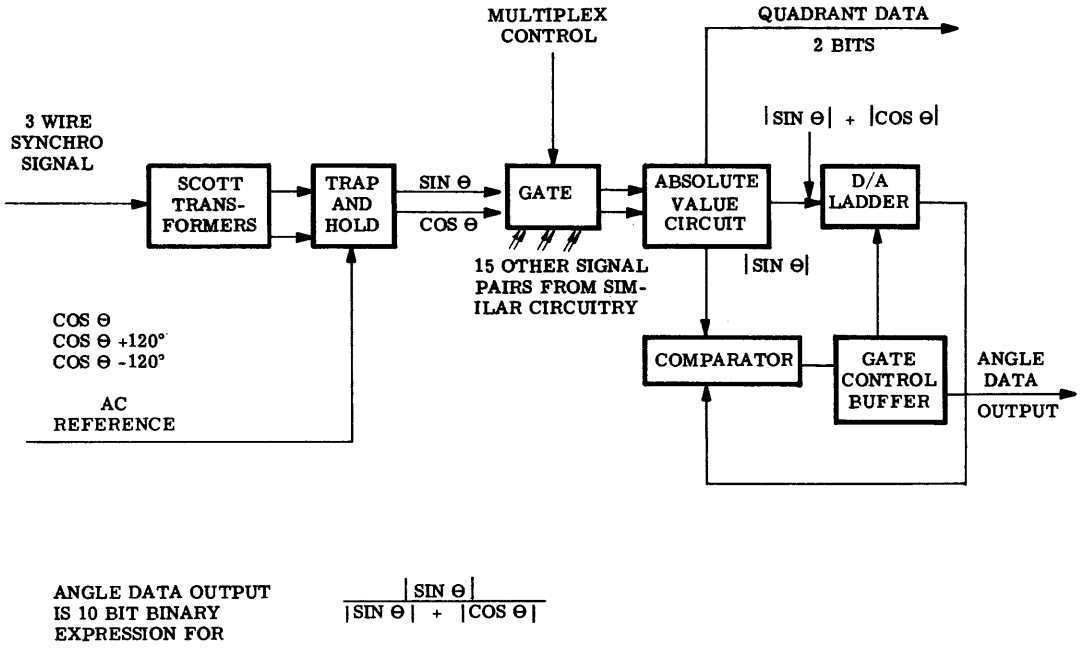


Figure 3 Librascope Synchro-to-Digital Conversion

A second feature is the "conversational" procedure for entering the input instructions into the keyboard. In the set-up procedure, it is necessary to define the input channels in terms of their interconnections with the ship, the quantities measured, scale factors, and merging instructions for multispeed data. In addition, it is necessary to define sample rates, printout and display rates, and quantities to be printed and displayed. The "conversational" procedure allows the computer to verify that all these variables have been defined and to request entry of any missing item. It also provides a hard copy of such actual data and enters it into an initial data block on the output tape. Another feature is the capability of typing warning data corresponding to tape parity failure or similar defects that might affect one data block only. This type of information allows second level rejection of single data blocks without entering bad data into the higher level programming. The final benefit is the self-checking capability that allows the system to go through its calibration procedures without requiring continuous surveillance.

CONCLUSION

The net result of using a commercial machine for this application has been a significant reduction in hardware cost, which is more than sufficient to cover the programming efforts required. In addition, the system is still a standard PDP-8 with some special peripheral equipment, and is completely "open-ended" for both programming and hardware.

ACKNOWLEDGMENTS

Much appreciation is due to the following persons and their organizations for assistance and support during the above efforts:

James O. Banks of NAV UNDERWATER ORDSTA, who examined the PDP-8 subassemblies in shipboard environmental conditions.

Henry Burkhardt of DEC who provided invaluable assistance in programming and application of the PDP-8.

Stewart C. Davis of NAV UNDERWATER ORDSTA, who did the actual programming to determine feasibility.

Jerry Dietz of Librascope, who provided valuable suggestions on applications of Librascope circuitry.

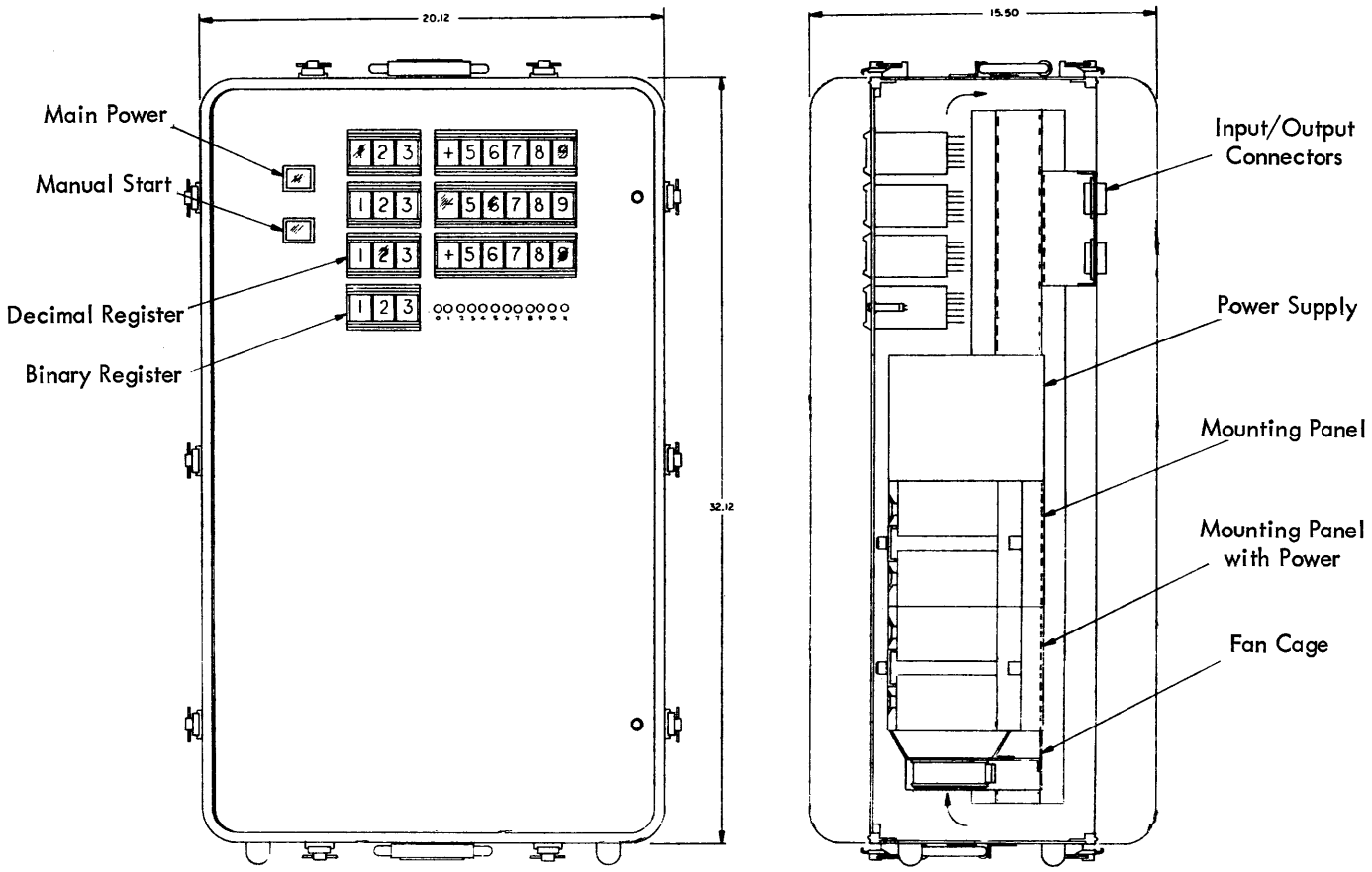


Figure 4 14 Containers of the Overall System

PLANT WIDE COMPUTER CAPABILITY

James Miller

Dow Badische Chemical Company
Freeport, Texas

ABSTRACT

This paper describes the current application of the computer configuration at Dow Badische which includes a PDP-5, DECtape 555, and 630 Communication System with six remote stations.

The paper describes how the off-line digital computer plays a role at Dow Badische. The time-shared programming system, which allows all stations immediate access to the computer and a wide selection of programs, is described. The language used to facilitate programming time-shared programs is outlined. The methods to permit remote stations to compile and run FORTRAN programs remotely are also presented.

Dow-Badische Chemical Company manufactures chemical intermediates which include caprolactam, acrylic monomers, and butanol. The plant is located 50 miles south of Houston, Texas, about five miles from the Gulf of Mexico. In 1964, Dow-Badische purchased a PDP-5 computer with 4K memory, a Type 555 DECtape having one dual transport, and a 630 Data Communications System with six remote stations.

A remote station, placed in each control area, provides computer capability throughout the plant and the results have been most satisfying. The distance from the farthest station to the central processor along the route of the cable is one-half mile. Each station consists of one ASR-33 Teleprinter operating on full duplex with reader-stop capability. This requires a six-conductor cable to each station and makes possible FORTRAN compiling, which calls for intermittent tape reading and punching.

Dow-Badische has equipped each teleprinter with a roll-around pedestal which allows the stations to be moved with considerable ease from office to office, or to a control room in the same building.

Periodically an operations man makes a tour of the control instrument area to log in specific items of data such as temperature, pressure, flow rates, composition of streams, and tank levels. Normally, as many as 60 entries are checked to see that the values are within specified limits.

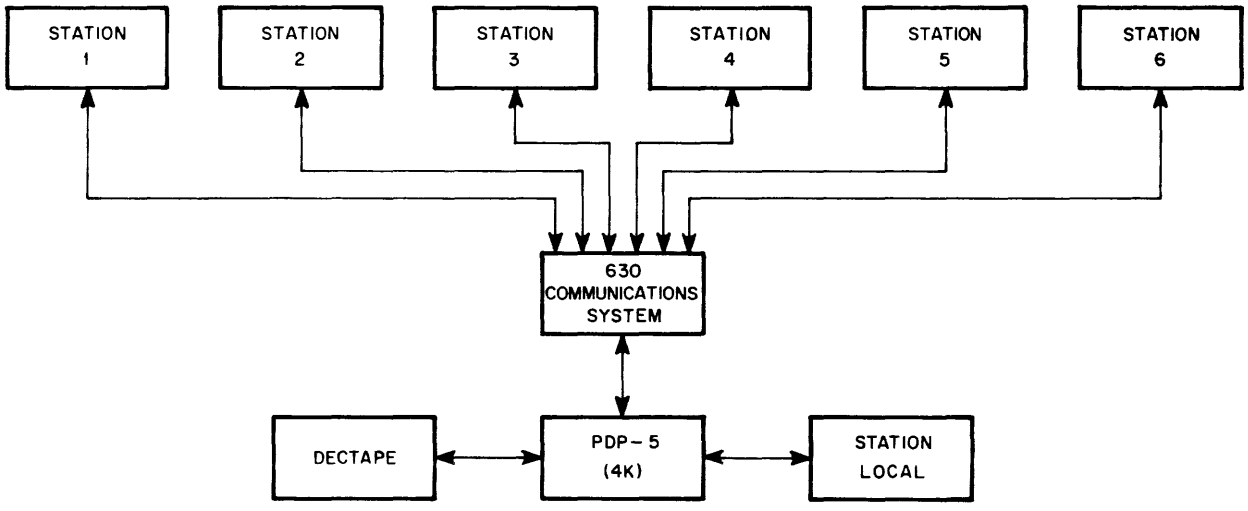


Figure 1 System Block Diagram

With the aid of the computer and suitable input/output devices, much more information is extracted. An actual survey is made of what has happened since the last set of recorded data, allowing the operations personnel to take action if product distribution is poor. Response is realized over periods of 4 or 8 hours, with such results as:

1. Assurance that the flow of materials into and out of equipment has proceeded as desired.

2. Rapid evaluation of the effect of deliberate process changes.

3. Rapid and exact measurement of the effect of unintentional process changes.

Without the computer such evaluations require tedious hand calculating which is time consuming and cannot be performed more than five times a week.

A special time-shared programming system has been developed providing each control area with immediate access to the central processor computer, and allowing ease of interpretation of data by computer users.

THE TIME-SHARED PROGRAMMING SYSTEM

All the double precision mathematical routines are gathered into one compact area along with a test conversion routine and a DECtape write and read routine. The executive routine does not use the interrupt because in this system no main program exists which gathers data or sends information to remote stations. Instead, each teleprinter operates a complete and independent program of its own, with selection of new programs occurring simultaneously with the operation of other station programs. This system allows each teleprinter 17 msec out of every 100 msec; it also responds to all flags of the DECtape system. Surrender of the computer occurs on input or output functions.

Each station has an area three pages long for its program plus one page which serves to link the station to the common mathematical text and DECtape routines. Programs requiring more than three pages are chained together by calling subsequent portions from DECtape into the same three-page area under program control. No interruption of other programs in progress occurs. Very long-programs can be devised in this fashion.

Messages can be sent from one station to another except when a station is on-line.

By means of a special DECtape routine located in the top page of memory, a remote station may take over the entire memory and compile or operate FORTRAN programs in design calculations. This alternating between "remote mode" and "time-sharing mode" is performed without assistance from personnel at the central processor. To prevent serious conflicts a schedule of hours is posted indicating when remote mode is permitted.

By means of the new PAL, a time-sharing "language" was developed. The majority of technical people write programs in this language. Programmers need not understand the details of basic time-sharing and DECtape programming; with PAL III it is possible to store all the pseudo instructions with the assembler, making it easier for the random programmer.

In addition, more than half the salaried technical people and some hourly employees are capable of writing time-shared and FORTRAN programs. This is the result of an intensive programming instruction project developed within the concept of plant-wide computer capability.

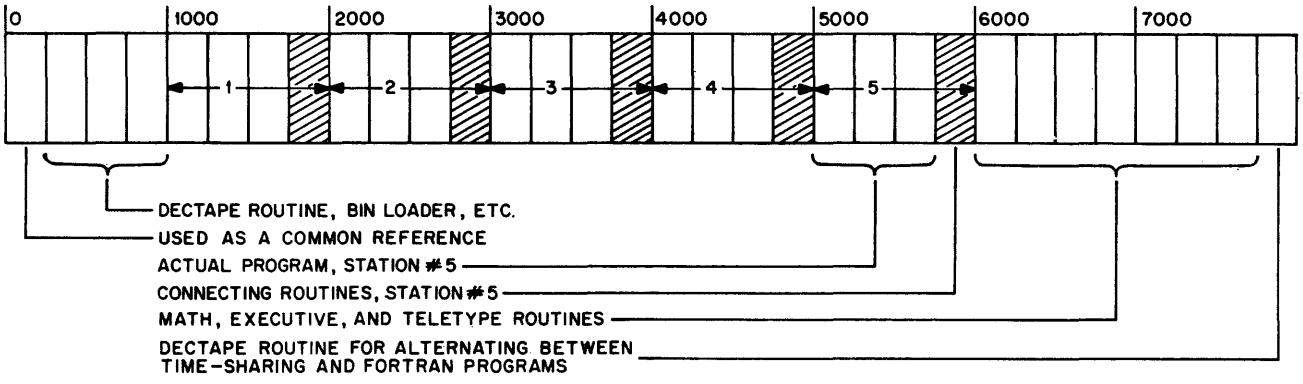


Figure 2 Time-Shared Programming System

SUMMARY

Some chemical companies have the computer do automatic data logging by tying into direct sensing devices in the plant. Still others "close the loop" by not only sensing data but also executing changes in plant operation under computer control.

The Dow-Badische system has the following advantages:

1. It costs less. It is expensive to convert existing data-gathering equipment from the more conventional air-operated style to digital signal-producing equipment or to provide converters.

2. Dow-Badische processes are constantly being upgraded; the present system demands only simple program changes.

3. The computer makes short-term "brush fire" projects quite simple to undertake. With other more complicated systems, such flexibility is often difficult and at times unavailable.

Briefly, with a total investment of $70,000, Dow-Badische has placed in five control areas a computer capable of:

1. Routine calculations
2. Routine records keeping
3. Plant performance evaluation
4. Design calculations
5. Additional user's commands.

APPENDIX 1
DECUS FALL SYMPOSIUM PROGRAM

Tresidder Union Hall, Stanford University
November 29, 1965

8:30 Registration

9:35 Opening of Meeting - William A. Fahle, DECUS President - John T. Gilmore, Jr. DECUS Meetings Chairman

9:45 Welcome Address - Philip R. Bevington, Stanford University, Physics Department

10:00 Computers of the Future - Their Logical Design and Implementation - David R. Brown, Guest Speaker, Stanford Research Institute

3:30 - 5:00 Tours

Stanford Computation Center - PDP-1 Time Sharing
Stanford SCANS System and Medical Center - PDP-7, PDP-8 and LINC
SLAC - Stanford Linear Accelerator Center

4:30 DECUS Delegates Meeting

5:30 Cocktails and Dinner at Ricky's Hyatt House, Palo Alto

SESSION A - Applications and Experiments

10:45 The PDP-1 Program Grasp
Ronald Gocht, Stuart Sharpe
United Aircraft Research Laboratories

11:15 Direct Digital Control to High Energy Particle Accelerators - Donald R. Machen
Lawrence Radiation Laboratory (Berkeley)

11:45 Current Trends in Hybrid Computer Oriented Software - Stephen F. Lundstrom
Consultant to Applied Dynamics Inc.

12:15 - 1:00 Lunch

1:15 Spiral Reader Control and Data Acquisition with the PDP-4 - Jon D. Stedman
Lawrence Radiation Laboratory (Berkeley)

1:45 Paramet, A Program for the Visual Investigation of Parametric Equations - Kenneth Bertran
Lawrence Radiation Laboratory (Livermore)

2:15 Coffee

2:30 On-Line Reduction of Nuclear Physics Data with the PDP-7 - Philip R. Bevington
Stanford University

3:00 Application of the LINC Computer to Operant Conditioning - C. Alan Boneau
Duke University - Stanford University

SESSION B - General and Utility

10:45 A.L.I.C.S. - Assembly Language
Joseph A. Rodnite
Information Control Systems

11:15 Some New Developments in the DECAL Compiler
Richard J. McQuillin
Inforonics Inc.

11:45 Message Switching System Using the PDP-5
Sypko Andreae
Lawrence Radiation Laboratory (Berkeley)

12:15 - 1:00 Lunch

1:15 The Learning Research and Developement Center's Computer Assisted Laboratory -
Ronald G. Ragsdale
University of Pittsburgh

1:45 Time Sharing the PDP-6
Thomas N. Hastings
Digital Equipment Corporation

2:15 Coffee

2:30 A Data Acquisition System for Submarine Weapons System Evaluation Utilizing a PDP-8 -
Robert H. Bowerman
U. S. Naval Underwater Ordnance Station

3:00 Plant Wide Computer Capability (PDP-5)
James Miller
Dow Badische Chemical Company

APPENDIX 2
AUTHOR AND SPEAKER INDEX

APPENDIX 3
ATTENDANCE

Name	Affiliation
Abrams, Les	Digital, Palo Alto
Allison, Robert W.	Lawrence Radiation Laboratory, Berkeley
Anderson, Harlan	Digital Equipment Corporation
Andreae, Sypko W.	Lawrence Radiation Laboratory, Berkeley
Barker, Donald	Digital Equipment Corporation
Berte, Bill	Dymec
Bertran, Kenneth	Lawrence Radiation Laboratory, Livermore
Bevington, Philip	Stanford University
Bishop, N.S.	Nortronics, Anaheim, Calif.
Bjerke, Conrad	University of Wisconsin
Black, John L.	Stanford University
Boneau, Alan	Duke University, Durham, N.C.
Bowerman, Robert H.	USN Underwater Ordnance, Newport
Brown, David R.	Stanford Research Institute
Brown, Roger M.	Lawrence Radiation Laboratory, Berkeley
Chalmers, Robert	Lockheed Missiles & Space Co.
Conn, Richard	Lawrence Radiation Laboratory, Livermore
Cossette, Angela	Digital Equipment Corporation
De Saussure, R.	Lawrence Radiation Laboratory, Livermore
Fahle, William A.	Systems Research Laboratory, Dayton
Fanshier, Donald R.	Lawrence Radiation Laboratory, Berkeley
Fisk, Irving	Hanan & Kiebler
Fitzgerald, John	Stanford University
Fleck, Laurice	MIT Lincoln Laboratory
Fleck, Phil	MIT Lincoln Laboratory
Gilmore, Jack	Adams Associates
Glazer, Eli	Brookhaven National Laboratory, Upton, N.Y.
Goldstein, Robert C.	Lawrence Radiation Laboratory, Livermore
Goodenough, John	USAF, Bedford, Massachusetts
Grandstaff, Dr. Netta	Stanford University
Greer, James B.	Lawrence Radiation Laboratory, Berkeley
Hance, Dr. A. J.	Stanford University
Harmon, Charles, J.	Wright Patterson AFB
Harvill, James R.	Lawrence Radiation Laboratory
Hastings, Thomas N.	Digital Equipment Corporation
Heikkinen, Dale	Stanford University
Holland, Jack H.	C.A.R.D.E.
Horen, D. J.	USNRDL, San Francisco
Ingram, Franklin E.	Beckman Systems
Jensen, Mark	Stanford Linear Accelerator Center
Jones, John A.	Digital Equipment Corporation
Keenan, John	University of Wisconsin
Kent, Douglas	Lawrence Radiation Laboratory, Livermore
Kiiskinen, Jon	Lawrence Radiation Laboratory, Berkeley
Klesmer, Stanley L.	Lawrence Radiation Laboratory, Berkeley
Larsen, Kenneth	Digital Equipment Corporation
Larson, Arvid	Stanford Research Institute
Lee, Daniel	Jet Propulsion Laboratory
Lothrop, Fred H.	Lawrence Radiation Laboratory, Berkeley
Lund, Paul	Lawrence Radiation Laboratory, Livermore
Lundstrom, S. F.	Information Control Systems
Machen, Donald R.	Lawrence Radiation Laboratory, Berkeley
McClure, John	Lawrence Radiation Laboratory
McQuillin, Richard J.	Inforonics, Maynard, Mass.
Miller, Jim	Dow Badische

Name	Affiliation
Myers, Myron	Lawrence Radiation Laboratory, Berkeley
Nelson, Dorothy	Stanford Research Institute
Pike, Chester D.	Lawrence Radiation Laboratory, Berkeley
Pyle, R.C.	Stromberg-Carlson Corp.
Ragsdale, Ronald	University of Pittsburgh
Rising, Donald R.	Lawrence Radiation Laboratory, Berkeley
Robinson, L.B.	Lawrence Radiation Laboratory, Berkeley
Robinson, Richard	Stanford University
Rodnite, Joseph	Information Control Systems
Roth, Richard J., Jr.	United Aircraft, Pratt & Whitney
Rudden, Robert W.	Lawrence Radiation Laboratory
Russell, Jerome	Institute of Medical Science
Schaeffer, Anthony J.	Lawrence Radiation Laboratory, Berkeley
Sharpe, Stuart	United Aircraft Research Labs.
Siegel, Louis	University of Rochester
Stedman, Jon D.	Lawrence Radiation Laboratory, Berkeley
Storch, F. David	Lawrence Radiation Laboratory, Livermore
Storey, L.R.O.	National Centre of Studies
Stubenitsky, Frank	University of California, Berkeley
Stylos, Paul	MIT, Lincoln Lab
Suffert, Martin	Stanford University
Taussig, Thomas	Lawrence Radiation Laboratory, Berkeley
Taylor, Raymond A.	USNRDL, San Francisco
Tubbs, Walter	Stanford University
Wagner, David E.	U.S. Naval Weapons Laboratory
Weir, Kenneth	Digital Equipment Corporation
Wigington, R.L.	Dept. of Defence, Fort Meade
Wilburn, Dr. N.P.	Battelle-Northwest, Richland
Wilkinson, Richard	Digital Equipment Corporation
Wirsching, Joseph	Lawrence Radiation Laboratory, Livermore
Zajec, Emery	Lawrence Radiation Laboratory, Berkeley