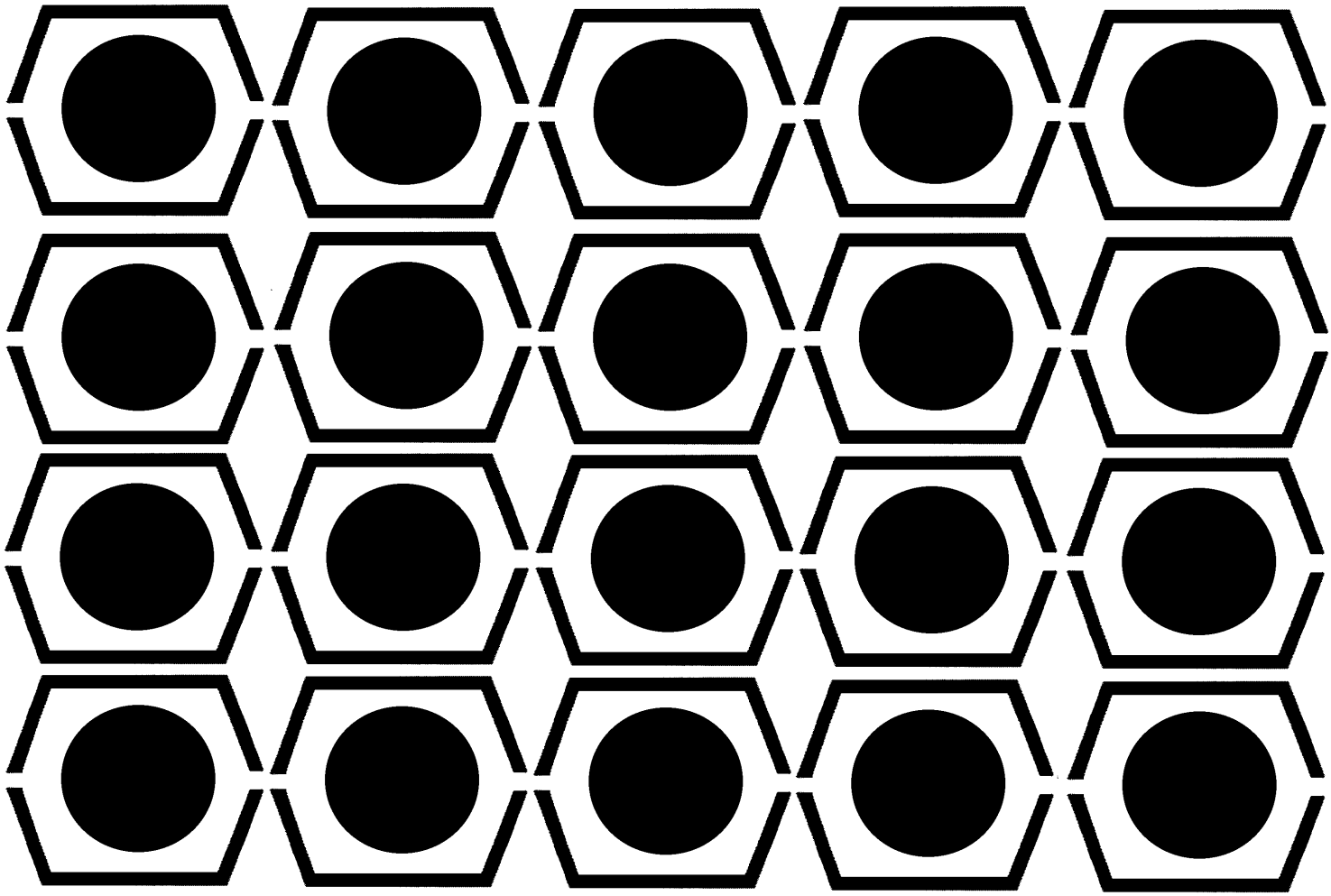


DECUS PROCEEDINGS

SPRING 1965

PAPERS AND PRESENTATIONS

of

The Digital Equipment Computer Users Society

Maynard, Massachusetts

SPRING
1965
PAPERS AND PRESENTATIONS
of
The Digital Equipment Computer Users Society
Maynard, Massachusetts

ACKNOWLEDGEMENT

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

PREFACE

The DECUS Spring Symposium of 1965 was hosted by the Center of Cognitive Studies, Harvard University. Our thanks to Dr. Donald Norman and his staff for a well organized meeting.

There was no particular theme for this symposium because of the desire to welcome many papers that might be of interest to the DECUS membership. The quality of papers presented attests to this fact.

The sound-off session, which was the first to be held at a DECUS Symposium, was a great success, and in the words of our DECUS President, William Fahle, "Its major fault seems to have been that it was about an hour too short."

Of particular interest was a luncheon talk given by Dr. James Oliver, Dean of the Graduate School of Sourthwestern Louisiana and President of the 1620 Users' Group. Dr. Oliver told us of his experiences with users' groups and passed on many helpful suggestions for a better and more effective users' group.

Tours and demonstrations of their systems were conducted by M.I.T.'s Project MAC and Keydata Corporation. A special evening demonstration was conducted by the Air Force Cambridge Research Laboratories. The demonstrations proved both interesting and informative.

In summarization, the two-day meeting was well attended, well received, and another step toward fulfilling one of the DECUS goals of information exchange.

John T. Gilmore, Jr.
DECUS Meetings Chairman

CONTENTS

*Abstract only

CONTENTS (continued)

DECUS ADDRESS

by

Harlan E. Anderson

Digital Equipment Corporation
Maynard, Massachusetts

Mr. Chairman, DECUS members and guests, I want to thank you for inviting me to speak at your meeting today. While considering various topics that I might talk to you about, I ran across the winning papers of a magazine contest in which the contestants were asked to predict what will happen in the computer industry during the next three years. It occurred to me that it might be interesting to speculate what the next three years hold for DECUS members. This seemed like an interesting topic because three years is a sufficiently short interval that any predictions have to be fairly tangible ones. This interval would rule out the blue-sky predictions that have so much appeal to the popular press and sometimes are opportunities for confused thinking even within the computer industry.

Although this topic seemed intriguing, it also had some potential problems. What if I predicted specific new equipment that had so much appeal that you members wanted to order them for your computers, but DEC never ended up manufacturing the particular item? To avoid this, of course, I could make only predictions about what you, the users, would be doing with your equipment during the next three years. This, of course, would be presumptuous of me and would have a different set of hazards associated with it. A third possible aspect of the topic that would avoid both of these problems, would be to talk only about future DEC supplied software. As you all well know, this is probably the most treacherous aspect of the whole topic. In view of these problems with this topic, I have decided to talk about all three parts.

One of the things that gave me the courage to do this was the review of the unique relationship that I believe exists between DEC and DECUS members. DECUS is slightly over three years old and has been an interesting organization. One of the reasons for this is that many of you are working on the frontiers of computer applications, and we, in DEC management, are proud to be associated with you.

The traditional sharing of computer programs has, of course, been an important part of DECUS. But the unique part of DECUS, I believe, has been the technical conferences such as this, in which you have had an opportunity to report on your work and exchange ideas with a group having many common interests. DEC has greatly benefited from the constructive criticism that has originated from DECUS meetings such as this. We have attempted when feasible to respond to your needs with action. In particular, some of our new products have been direct responses to these needs.

For example, our Type 340 Flicker-Free Display is an outgrowth of an experimental system done for Project MAC. Another example of this same type of thing has been the role that DECUS members have played in pioneering with DEC in the field of time-sharing. The world's first time-sharing system to be operational was a PDP-1 computer in the Cambridge laboratories of Bolt, Beranek, and Newman. This was soon followed by PDP-1 time-sharing systems at the MIT Electrical Engineering Department and at Stanford University. These were all instances where you, our customers, provided the system concepts and programming, and DEC provided standard and special hardware to accomplish the goals. These examples typify the unique relationship that I feel exists between DEC and DECUS members.

Let us turn now to the first aspect of the next three years for DECUS members; namely, future hardware.

I believe that large core memories are likely to become slightly faster and less expensive. This is not likely to be a result of any new revolutionary memory device since there is none on the horizon which is likely to be available in the next three years. Instead, these improvements are likely to come about from new electrical driving techniques and will make this all-important part of modern computers even more important. The current interest in large, slow, inexpensive core memories will not be with the industry long because of limited applications for which they are well suited and because fast memories are likely to decrease in cost in the future. The arbitrary dividing line that I am using for fast and slow is a 2-μsec cycle time.

The second hardware area that I believe will show remarkable change in the next three years is the CRT display. The changes will not come about so much from a new technique standpoint as from a wider understanding of what are reasonable uses for our displays. I feel that we have

just scratched the surface of display applications. This experience will possibly lead to some logical changes aimed at making the flicker-free quality of the picture better and reducing the amount of memory and central processor capacity required to drive displays. More widspread use of displays, particularly those driven by small satellite computers, will pave the way for less expensive units. Time-sharing concepts will become an important part of the growth of display applications. Already we see an example of this in the new Type 338 Display, recently announced by DEC, which incorporates a PDP-8 computer as an integral part of the display.

The third hardware area is that of the satellite computer. Machines like the PDP-1's, PDP-4's, and PDP-5's are finding new uses as satellite computers to larger machines in many of your organizations. This trend I foresee being accelerated with the PDP-7 and PDP-8. Some of these satellites will be remote and others will be local satellites. The applications of these satellites will eventually include solution to simple problems and forwarding of more complex problems to a larger computer. Satellites will also perform data collection for on-line experiments causing occasional interrupts to a central computer. Scopes will be used more extensively for monitoring the data collection process.

The fourth hardware area is that of discs. DEC will add new disc units to the 35-million-character unit presently available with the PDP-6. These newer units, which are in the planning stage at the present time, will have lower capacity and will be significantly less expensive. DECtape and IBM-compatible tape will become the major off-line storage media for data and programs and will gradually replace much of the paper tape.

I would like to turn now to the area of user software. Due to the pioneering nature of the work done by you with your PDP computers, it appears to me that you will continue to develop many of your own specialized software systems such as the on-line hospital system developed by Bolt, Beranek, and Newman, the film-scanning system developed by Information International, and others. It would be presumptuous of DEC to claim to know your respective area of application sufficiently well to undertake providing this type of program. One person has suggested to me that we shouldn't do it even if we did have the experience because we might cause technical unemployment of some of our customers. I see a continued compatibility between DEC's hardware capability and your application capability. New application possibility

for computers are far from exhausted, and I foresee the members of DECUS making a continuing and important contribution in these areas. In the process of exploring new areas, I believe that more and more of your organizations, whether you are commercial or nonprofit, will find yourself offering a specialized service to your customers. Examples of this already include the Keydata facility offering business and engineering services to the Boston community; Bolt, Beranek, and Newman, offering patient record keeping for Massachusetts General Hospital; and now the RAND Corporation planning to offer an internal on-line JOSS computing service with its new PDP-6. I believe that the final test of some of the applications will be the rendering of pilot services such as those I have mentioned. The challenge that faces many of you is to reduce some of the blue-sky highly experimental application plans to specific tools usable by people other than those directly involved in the computer field and requiring a minimum of training for these people. Some of you might find it tempting to move on to other blue-sky application areas before the present ones have been reduced to practice. I would strongly urge you to resist this temptation so that your pioneering work does not wither on the vine. In short, I see an increasing amount of user-generated software.

Now I would like to say a few words about manufacturer-supplied software. As some of the old timers in DECUS remember, the software provided by the computer industry as a whole was not extensive in the early days. It was not unusual for a computer's one assembly program to have been generated by the first customer to purchase a computer. This was true of the largest to the smallest computer manufacturer in the very early days. The industry as a whole and DEC in particular during the past year have come a long way on this subject. This was in response to needs expressed by you, the DECUS members. All current DEC computers have FORTRAN Compilers available for them. All of the software provided by DEC, of course, has been intended for generalized use as distinguished from that specialized software I mentioned earlier. In the next three years, I foresee two areas in which more extensive software is likely to be available. The first of these is in monitoring or operating systems. These will be aimed at making the input-output facilities of the computer more usable with less effort on the part of the programmer. The ability to automatically call library functions to save programs on tape are examples of the kind of thing that might be done. Small amounts of this type of thing are already being done, of course, on the PDP-7. The second software area likely to show change in the next three years is software to permit the easy use of a small computer as a satellite to

a larger computer. The software may permit the small computer to do on-line editing, desk calculator operation or small problem solving, display driving, and similar functions. In addition, it will provide the means of communicating problems to the larger computer. In the graphic field, the work load might be divided between the two machines as follows: The large computer might do the arithmetic operations and the assembling of the picture while the small computer's role would be the continual regeneration of the flicker-free display and the sensing for light pen interrupts.

I would now like to take a moment to give you a status report on the PDP-6 time-sharing software system as an indication of the importance being placed on software. I understand that our technical staff either has or is about to cross that mythical barrier where the programmers outnumber the hardware engineers.

The PDP-6 software system is a multiprogramming system in which several users can coexist in core memory at one time. The users are prevented from interfering with each other by memory protection and in-out traps, which are controlled by the Monitor program. This Monitor is permanently in memory and occupies about 5,000 words. It takes care of all input-output operations for a user freeing him of the necessity of having the detailed I/O programs in his area of memory.

In addition the Monitor allocates the system facilities such as core memory and tape units to the users. The Monitor also schedules short bursts of running time for each user on a round-robin basis. It automatically relocates a user's program to work in the area of core memory that is assigned. To accomplish this, all user programs are written with respect to memory location 0.

All of the normal service programs can, of course, be operated under control of the time-sharing Monitor. These include traditional DEC programs such as DDT and an editor, along with the more conventional programs such as the FORTRAN Compiler and the MACRO 6 Assembler, both of which are designed to work with our Linking Loader.

A more unusual service program is our Peripheral Interchange Program called PIP. It provides for card-to-tape, tape-to-line printer conversions, which at a large computation center have normally been done on an off-line satellite computer.

Any combination of those service programs can be in use at one time subject only to the limitation that enough core memory and in-out facilities are available in the system. These features, of course, can be used and controlled from a Teletype console that is either local or remote.

This software system is by far the most ambitious ever undertaken at DEC, and I am pleased to report to you that it is now operational and in use by PDP-6 customers everywhere, including such distant locations as the University of Western Australia in Perth. Last week I was visiting Professor McCarthy at Stanford, and we made an on-line transcontinental demonstration of the preparation, compilation, and execution of a FORTRAN program with the PDP-6 in Maynard. For those of you who are planning to attend the IFIP meeting in New York next week, we will have remote demonstrations at our exhibit.

The next three years for PDP-6 software will integrate large backing stores such as the million-word drum and the disc into the system. It may include N.P.L. if it appears to achieve a sufficiently universal appeal. In addition, the multiprocessor hardware capabilities that are now an inherent part of the PDP-6 design may be utilized in the time-sharing software system.

In short, I believe the next three years are going to be exciting ones for DECUS members in the area of new hardware, new user applications and specialized software, and expanded DEC supplied software. I look forward to a continuation of the cooperative atmosphere in which DEC and DECUS have been carrying out their work.

COMPUTER-AIDED DESIGN*

Steven A. Coons

Massachusetts Institute of Technology
Cambridge, Massachusetts

Abstract A project directed toward creating a symbiotic relationship between a man and a large general-purpose digital computer, so that the two can perform the design process in concert in a way far different from present-day methods, is discussed.

Part of the problem involves development of more convenient means for communication with a computer. Graphical communication, coupled with symbolic and verbal communication, will lead to a revolutionary system in which a handful of engineering designers in the future can perform the work of many technicians in a fraction of time now required.

*This paper was not submitted in time for publication.

DECADE - COMPUTER-AIDED DESIGN ON LOW COST HARDWARE

by

Charles W. Stein

Digital Equipment Corporation
Maynard, Massachusetts

Abstract DECADE (Digital Equipment Corporation's Automatic DEsign System) is the result of an effort to perform engineer/computer communication via a display and light pen. This system provides the interface between man and machine using the general language of schematic drawing. Coupled with the system are analysis programs which can produce wire lists, automatic wiring machine cards, and parts lists. The system is general enough to allow user-written analysis programs. This paper describes the use and programming of DECADE.

INTRODUCTION

MIT, Boeing, IBM, CDC, United Aircraft, and Lockheed, among others, are all doing what they call "computer-aided design." Magazine articles show a man sitting at a computer-driven scope, holding a light pen. Displayed on a screen is a bridge, automobile, airplane wings, or some such complex mechanical device.

This type of system is fine for an aircraft or bridge-building company. However, there are many more firms that need a smaller version of computer-aided design. Their problems are more specific. Many companies are devoted to schematic drawing almost exclusively. Other companies spend 80 to 90% of their time producing schematic drawings. We feel that if a man could give such a drawing to a computer, rather than a generalized line drawing, he could reduce his company's drafting effort considerably. Furthermore, if some programs existed which would analyze the schematic and produce a listing about it, other areas of the company's design effort would be reduced.

DECADE permits such input on low cost hardware. Schematic drawings can be inputted through the use of a Teletype, light pen, and push-button control. This paper describes the system and its use.

We define a schematic as the graphic representation of a system in terms of a set of standard symbols. Lines connect groups of symbols to indicate some relationship between them. Text

is used to describe the symbols and connections. A programming flow chart, a PERT chart, a logic diagram, and a wiring diagram all fall into our definition of schematic.

Schematic drawing was chosen because many of the problems inherent in the general system are eliminated. In the past, CAD systems have required large-scale, general-purpose computers. However, when we limit the input to a schematic, less core storage is needed. Furthermore, replacement of magnetic tapes with DECtapes reduces cost of the system further.

Less core storage is required in DECADE than in the general system for the following reasons:

1. The standard symbols are topologically defined within themselves. The fact that a symbol is a resistor says many things about it that the general system must state in full each time a resistor is called.

2. Since the set of symbols is predefined, we need only to prescribe it once in core. For example: It is not necessary to duplicate all the vectors which draw a resistor each time one appears on the screen. We only need a call to the resistor subroutine which draws and defines it.

There are other advantages to using a small computer system. CAD is now within the reach of many smaller firms which could not afford the large system. DECADE will sell for approximately $200,000. Indeed, it performs a smaller task, but this task now costs industry millions of dollars. By speeding up many of the jobs now done by hand, both the cost and the frequency of error is reduced.

THE HARDWARE

Figure 1 shows a sketch of the basic hardware configuration. A discussion of each of the components follows.

PDP-7 Computer

The word size of the PDP-7 is 18 bits. There are 8192 words of core. The cycle time is 1.75 μsec with an add time of 3.50 μsec. The computer used has no hardware multiply/divide feature.

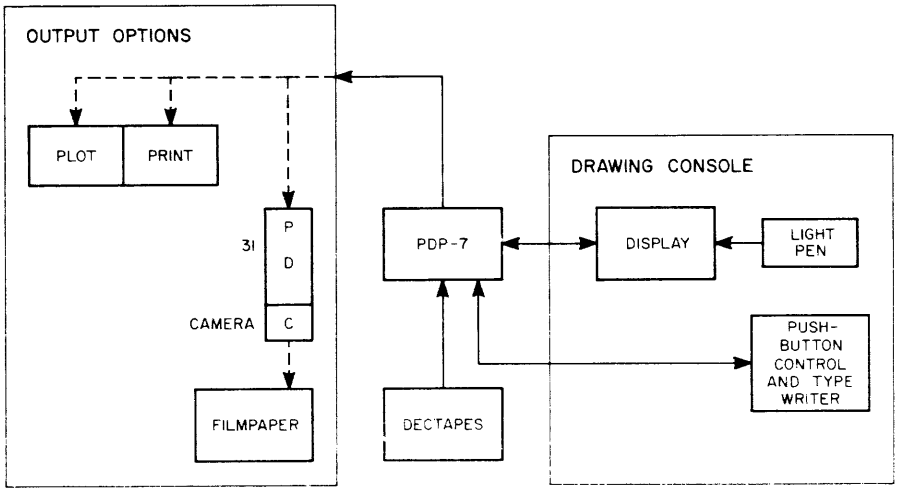


Figure 1 DECADE Hardware

DECtape

There are four DECtape drives attached to the system. Each tape has 576 blocks of 256 words each. Software exists which permits the search and read/write of any block or blocks. This format is highly desirable for information retrieval systems since a directory can provide the specific address of information.

Console

The console consists of a display and light pen, Teletype, and push-button controls. They are described individually.

Display and Pen

The display used is a modified 340 Incremental Display with the 347 Subroutining Option. The pen is the 370 High-Speed Light Pen. The subroutining option is one of the most powerful devices in the system. It permits the display to be thought of as an on-line processor. Just as the computer's central processor performs operations on memory, so the display processor takes a word from memory and performs some operation. Some of these operations result in a line or curve being displayed on the screen. Others change the location in memory from which the display is selecting words or changes the sense of interpretation of the words.

There are five modes of interpretation which the display uses: parameter, point, subroutine, vector, and increment. The first, the parameter mode, sets controls in the display. Beam brightness, scale factor, light pen status, and mode of the next word are all set in the parameter mode. The display may also be stopped by the parameter mode. Each "instruction" specifies its operation and the mode of the next word to be received.

In point, vector, or increment modes, the instructions specify display events such as plot a vector, set the X and Y register, increment the X register, etc. The subroutine mode, however, specifies the location in core from which the next word will be taken and the mode of that word.

The display references some particular location in memory. The address of this location is contained in the display address counter or DAC. Normally, the DAC is incremented by one after each instruction. Using the subroutine mode, we have three instructions which change or save the DAC. They are:

DJP, MODE, ADDRESS	This instruction sets the DAC and the MODE to the specified values.
DJS, MODE, ADDRESS	This instruction saves the DAC in another register called the address save register or ASR. It sets the mode as specified and sets the DAC to the proper address.
DDS, MODE, ADDRESS	Unlike the other two instructions, DDS does not change the DAC. Rather, it sets the specified mode and deposits a DJP in the parameter mode with the contents of the ASR into the specified core location.

The use of the subroutine mode can best be shown through an example.

Example 1

Display the characters "ABCABCAB..
BCABC" across the screen.

Normally, we would have to build a display table in core which contained the vectors to draw the characters A, B, and C many times. However, we will do the same job with the given display file.

```
P       S                       /GO TO SUBROUTINE MODE
        DJS     S       A       /SET MODE TO SUBROUTINE
                                /SET ASR TO P + 2
                                /SET DAC TO A
        S                       /RETURN IN PARAM MODE, GO BACK
                                /TO SUBROUTINE MODE
Q       DJS     S       B       /SET MODE TO SUBROUTINE
                                /SET ASR TO Q + 1
                                /SET DAC TO B
        S                       /RETURN IN PARAM.  SET TO SUB
R       DJS     S       C       /SET MODE TO SUBROUTINE
                                /SET ASR TO R + 1
                                /SET DAC TO C
        S                       /RETURN IN PARAM, RETURN TO SUB
        DJP     S       P + 1   /SET MODE TO SUB, SET DAC TO P + 1.  LOOP
                                /THROUGH P, Q, AND R AGAIN
```

CHARACTER SUBROUTINES:

```
        A,  DDS     V     AEXIT     /SET DJP IN PARAM MODE
                                    /TO C(ASR) = P + 1 IN AEXIT

            VECTORS TO DRAW
            LETTER A

            S                       /GO TO SUBROUTINE MODE

        AEXIT,      0               /WILL GET SET TO DJP P + 1
```

B AND C SIMILAR IN FORMAT.

Thus with only eight words plus the vectors to draw the characters once, we have achieved an otherwise long and memory absorbing table.

The light pen is used for two functions: to point at some place on the screen or to draw lines. Displayed on the screen at all times is a cross which is used to plot the pen. This cross represents the current location of the pen. To change this, place the pen over the cross and move the pen. The cross will follow. A more thorough discussion of the pen and its use follows.

Teletype - This is used in connection with the information retrieval system and the drawing function. Both will be described more fully later. The Teletype is a KSR Model 35 Teletype.

Push Buttons - These buttons are used to control the drawing effort. Figure 2 shows a sketch of the button box. The box is movable for more convenient use. The action of the individual buttons will be discussed later.

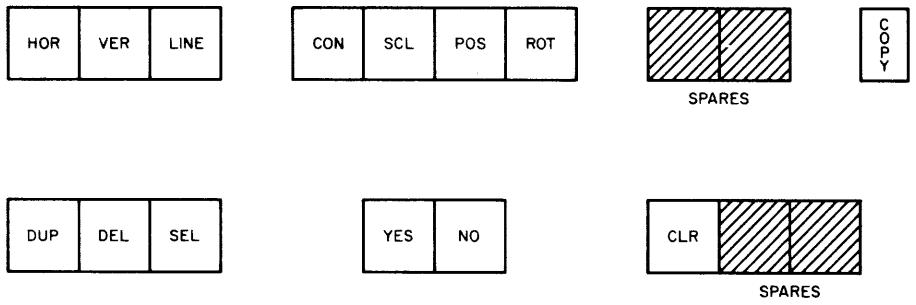


Figure 2 Push-Button Console

Output Option

The system must provide some means for hard-copy output. The desired output is in two forms: schematic drawings and printed output. There are two output options offered. The first consists of the line printer and a display-driven Calcomp plotter. The second uses the Type 31 Ultra-precision (4096 x 4096 grid) Display, computer-controlled camera and automatic film processor, and film-to-hard-copy equipment. The initial cost of the first system is less, but when the amount of time required to process the output on the first system is considered, the operating cost of the second is much lower. Also, the quality of print produced on the plotter is less than that of the film.

SOFTWARE

The software package will be discussed in two parts. The first part is the information retrieval system (IRS) which is used to keep track of the drawings already done on the system.

There are four DECtape drives which are allocated as follows:

System Tape	Contains DECADE plus all analysis programs to be used.

Drawing Tape	This tape contains the drawing which we are now working on. One tape can hold up to 28 different drawings. If this is the start of a new tape, this unit must be initialized by the system.
Scratch Tapes	The system uses these for temporary storage. The analysis programs may also use these.

A drawing may be given any number of names as long as the total number of characters does not exceed 500. The IRS can retrieve a drawing from the tape on any key. At any time, we can add a key or list and edit the existing keys for any drawing in core. We can copy a drawing or set of drawings on either of the scratch tapes. In fact, the procedure used to get hard-copy output might be to copy the drawings desired on a scratch tape during the day and to process them at night during off hours.

The second part of the system is the actual graphical input program (GIP). Using the light pen, typewriter, and push buttons, the user can "draw" some type of schematic.

The Light Pen

The pen is used to perform two tasks, drawing and pointing. Since the orifice of the pen is fairly large, it is difficult to specify some exact point. For that reason, there are two points which are displayed with the pen cross. They are the location and drawing points or LPL and LPD.

At all times, there is a cross displayed on the screen. This cross represents the last known location of the pen. Above and to the left of the cross is a point which is the LPL. If no push buttons are being used, the LPD is coincident with the LPL. To change the location of the cross, the pen is placed over the cross and moved. The cross will follow the pen.

There are three push buttons which are involved with the LPD and LPL. They are HOR, VER, and LINE.

HOR

Constrain the LPD in such a way that its vertical coordinate never changes. That is, only allow LPD to move horizontally. In general, this will cause separation of the cross and LPL from the LPD.

VER

Constrain the LPD in such a way that its horizontal coordinate never changes. That is, only allow the LPD to move vertically.

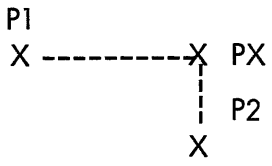
Both HOR and VER are permanent switches. Pressing once will set the condition on. Pressing again will clear these conditions. It is obvious when a switch is on because it glows brightly. Notice that pressing both HOR and VER will cause the LPD to remain stationary.

LINE

There are three steps in the LINE switch.

Pressed Down	This initializes a line. The program saves the location of LPD.
Held Down	While the switch is held down, a line is constantly being drawn for the start point to the current LPD. It's as if a rubber band were stretched from the start point to the LPD.
Released	The current value of the LPD is saved and a line from the start point to the last LPD value is added to the picture.

Example - Connect P1 and P2 together by a horizontal and then vertical line as shown below by the dotted lines.



1. Position LPD at P1 and depress HOR.

2. Depress LINE, hold down and move LPL to P2.

3. Release LINE. This draws a line from P1 and PX and LPD is now at PX. Notice the PX is directly above P2 now since the LPL is at P2.

4. Depress LINE.

5. Depress HOR again to release.

6. The LPD quickly jumps to P2, drawing the vertical line.

7. Release LINE. The two lines are now drawn.

Symbol Generation

Each symbol has been given a tag or name. To call a symbol on the screen, the user types "G." The system will type back three spaces. Then the user types in the tag and a line feed. If a mistake is made, the RUB OUT key will ignore that message. When a symbol is called, it appears in the lower left corner of the screen.

The system provides for the use of a 22 x 34 in. drawing. It is obvious that it would be impossible to draw this size drawing on a 9-in. square screen. Thus, we have two scales of operation. Scale 0 shows the entire 22 x 34 in. drawing, greatly reduced. Displayed in the picture is a 2-1/4-in. square. This can be thought of as a picture frame. Positioning the frame over some section of the drawing and depressing SCL causes that section to be blown up four times. This is scale 1. Depressing SCL again returns to scale 0. The standard drafting procedure is to pick a section of a drawing and blow it up. Then work on that until all has been done. Then move the frame and resume.

A symbol can be called on the screen at both scales. Typing G and then the tag causes the symbol to be made "live." There is a table which keeps track of the live symbols. The CLR button clears this table. The live symbols are affected, as a group, by the POS, ROT, DUP, DEL, YES, and NO buttons. It's obvious which symbols are live since they are brighter than the others.

Another way to make a symbol live is with the SEL or select button. Place the LPL over the symbol and press SEL. The symbol becomes live. If a mistake has been made, merely depress NO. The symbol will be reset to nonlive. If it is desired to make the symbol live, YES must be depressed. All buttons are ignored until either YES or NO is depressed. The buttons POS, ROT, DUP, DEL, YES, and NO operate on live symbols. Their functions will be described individually below:

POS	Position the first live symbols at the LPD. Other live symbols move maintaining their relative positions with respect to the first live element.
ROT	Rotate all live symbols 90° counter-clockwise. The symbols maintain the same start point and rotate independently.
DUP	Duplicate a set of live symbols on top of the current live symbols. Reset the live table to make the new symbols live. Notice that symbols only may be duplicated. Any connections are dropped.
DEL	Delete the live symbols and their connections from the picture. If a mistake is made, depress NO. The live symbols and connections will reappear. YES completes the action and removes the symbols from LIVE. All buttons are disabled after DEL has been pressed until either YES or NO have been pressed.

It may also be desirable to delete part of a connection. To do this, put the LPD over that part of the connection, and depress DEL. If any symbols are live at this time, they will also be deleted. If this happens, press NO, CLR and repeat the process.

Connection

Since connections between symbols imply some relationship, they are handled differently than just lines. To indicate that this symbol is connected to something at this point, the CON

button is used. Connections can be made only at previously defined hot points on symbols but can be made anywhere on a connection.

To make a connection, place the LPD at the desired connection point and then press CON. At that time, all other buttons except HOR, VER, LINE, and CON are disabled, until CON is pressed again.

Through normal line drawing operations, draw the connection, then depress CON again. A connection can be made at some point not on a symbol or old connection. However, if this is desired, the system enters the text mode, and all buttons are disabled until a text terminate is given. This guarantees that there will be no undefined connections.

Text

Generally, there are three types of text information, that text which describes symbols, that which describes connections, and that which describes the drawing. It is necessary that the system understand which type of text the user wants. Again, we use the LPD to point either to a symbol, a connection, or nothing (space).

To enter text, place the LPD over the object to be described and type T, the system will type back S, C, or N for symbol connection, or nothing. If the system types back the wrong character, type carriage return, move the LPD to a more obvious spot on the element, and retype T.

Once the system understands the operator, he may begin typing text information. Text is displayed as a block of characters which are positioned left justified to the LPD. If the user moves the LPD, the text received thus far will move with it. A carriage return key returns to a point directly under the LPD. When the text is in and positioned properly, a line feed terminates the message. From then on, the text block is dependent on the item it references. If the text references a symbol and the symbol gets moved or deleted, the text moves or disappears also. In the case of text which references the drawing, it is stationary and should be thought of as a symbol.

CONCLUSION

A man-machine interface system which operated on a small computer has been described in this paper. This system is currently under development at DEC. However, this is by no means the final version of the system.

At this time, there are a few refinements which are already obvious.

1. One of the problems is that most companies have large amounts of information now on microfilm. An obviously desirable program would be one which reads microfilm and recognizes patterns to generate semiautomatically the internal topological structure of that drawing.

2. Some general-purpose routines which would work on the data need to be written. Some examples of such programs might be a PERT/COST system, a flow chart compiler, a wire list generator, a parts list generator, and many others. As one works with the system, new applications become more clear. The information necessary is all in core or on tape for any of these systems.

3. If more core were added, a time-sharing system could be written which would swap the internal list structure in and out of core and handle many consoles. This task is easier than it may sound if the 338 Buffered Display is used. All that is necessary is to make the system sensitive to more than one control console. Then the required description can be loaded from drum, modified, and shipped to the 338.

If enough of these systems are sold, a users' group, like DECUS, could add considerable power. The system is general enough that a multitude of problems could be solved.

THE USE OF A PDP-5 COMPUTER IN THE COLLECTION OF MOSSBAUER EXPERIMENTAL DATA

by

Ronald H. Goodman and John E. Richardson

Department of Mines and Technical Surveys
Ottawa, Ontario

Digital Equipment Canada Limited
Carleton Place, Ontario

Abstract The nature of Mossbauer experiments and their application in solid-state physics is described briefly. The two types of experimental situations, constant velocity and constant acceleration, and the relative merits of each are presented. An outline of the conversion of the experimental data to digital form and the use of the PDP-5 as a Mossbauer data acquisition device will be discussed. Typical sets of experimental spectra indicate the overall performance of the system.

This paper discusses the Mossbauer effect and some of the chief experimental situations. The use of a small on-line computer for the collection of this experimental data is discussed and some typical spectra are illustrated.

What is the Mossbauer effect? In normal optical situations the resonant absorption of photons is quite well known; whereas, in the case of the nuclear gamma-ray, fluorescent-resonant absorption is rare. Why this should be so is shown in Figure 1. In the optical case, the distance between the peaks, which is due to recoil energy, is much less than the width of the peaks (typically 10^{-8} ev.). In the nuclear case, the separation due to recoil is much greater than the widths. Thus when a gamma-ray from the decay of one nucleus strikes another nucleus, it will not be resonantly absorbed. This is a property of nuclear gamma emission, and the performance of fluorescent resonance experiments are difficult since the energies which are required to cause fluorescence are in the order of 10^{-5} ev.

What was Mossbauer's discovery? It was found that if a nucleus was imbedded in a crystalline material, its recoil energy would not be transferred to a single nucleus but rather to the whole assemblage of nuclei making up the crystal. Since the mass enters into the formula for the recoil energy, this energy will be much smaller and resonance absorption can occur. This

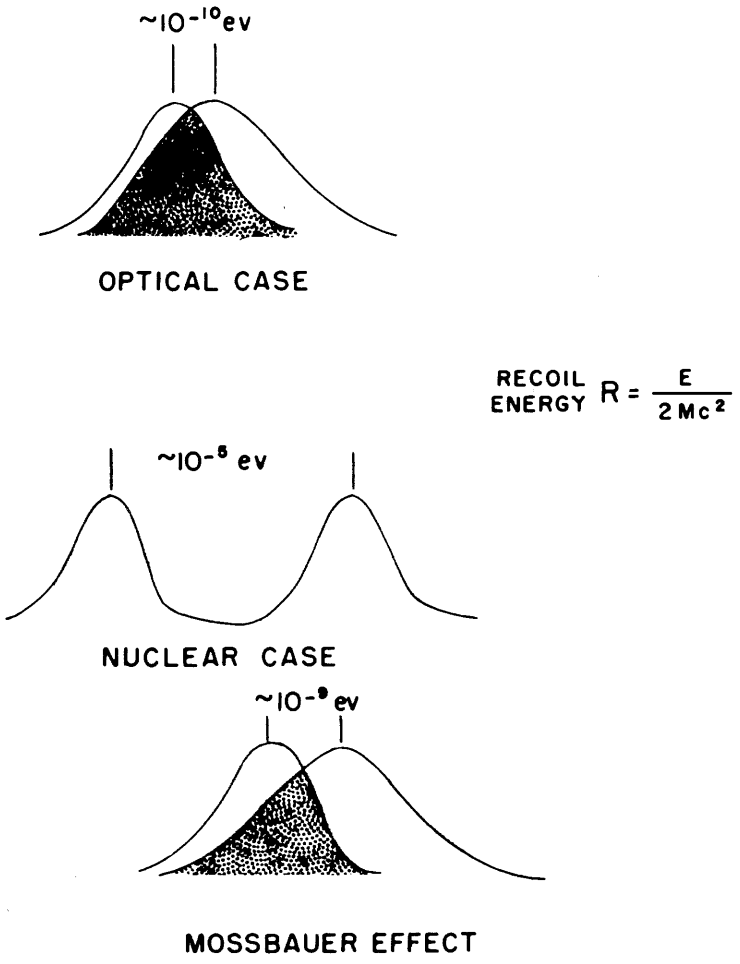


Figure 1 Various Resonance Situations

absorption of the energy by the total crystal is reasonable from the point of view of quantum mechanics. If we assume a rather simple model of lattice vibrations; namely, that proposed by Einstein, (Figure 2) we see that when the recoil energy is less than the Einstein energy, no phonon emission will occur and the gamma-ray must be emitted with its total energy. This is essentially the Mossbauer effect. A more exact model of lattice vibrations is that proposed by Debye which is also shown in Figure 2. A distribution of lattice energies is possible but the low energy phenomena are due to the motion of more than one nucleus. Since the recoil energy is distributed over all the nuclei again, recoilless emission can occur. This is the Mossbauer effect. Now, what is its use in the study of the properties of solids?

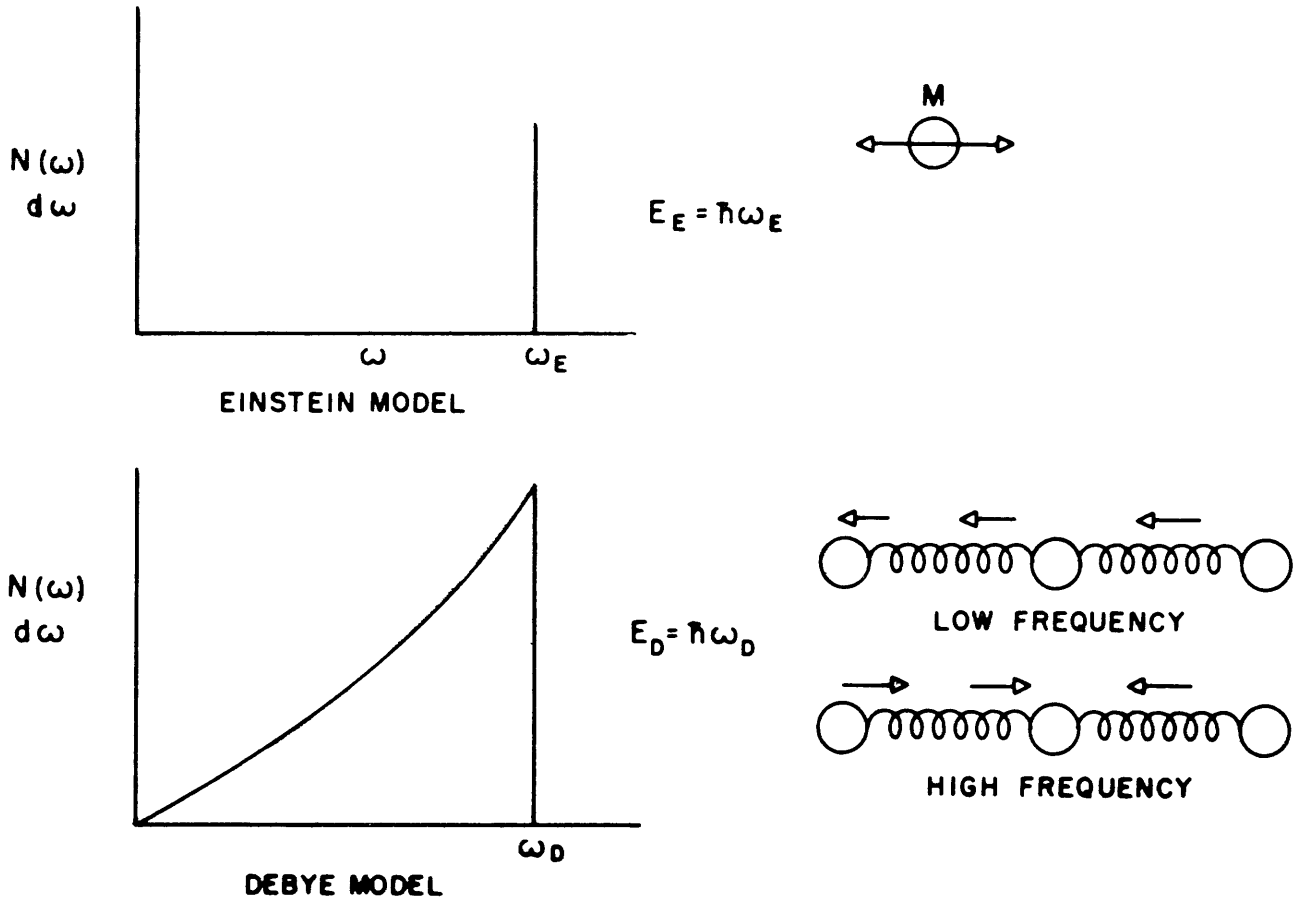


Figure 2 Lattice Models

In Figure 3, a nuclear Zeeman effect is schematically illustrated. The Zeeman effect, as observed in optical spectra, is due to the splitting of levels in a magnetic field. In the nucleus the same phenomena occurs where the splitting is primarily due to the internal fields of a solid, which are about 5×0^5 gauss. Thus there is a diagram for a magnetic field as shown on the left-hand side of the figure, and with the selection rule $\Delta M = 0, \pm 1$, the 6-line Mossbauer spectrum is predicted. If there is an electric field gradient and a quadruple moment of the nucleus, there will be a level diagram as shown to the right. Since in any given solid material, there may be more than one type of site available, these Mossbauer spectra can become complicated.

The experimental apparatus used for Mossbauer experiments may be classified into two types. The first is a system of shifting the energy so as to scan across the energy spectrum, and the second is a system of collecting this data. The energy-scanning system may be reduced to a

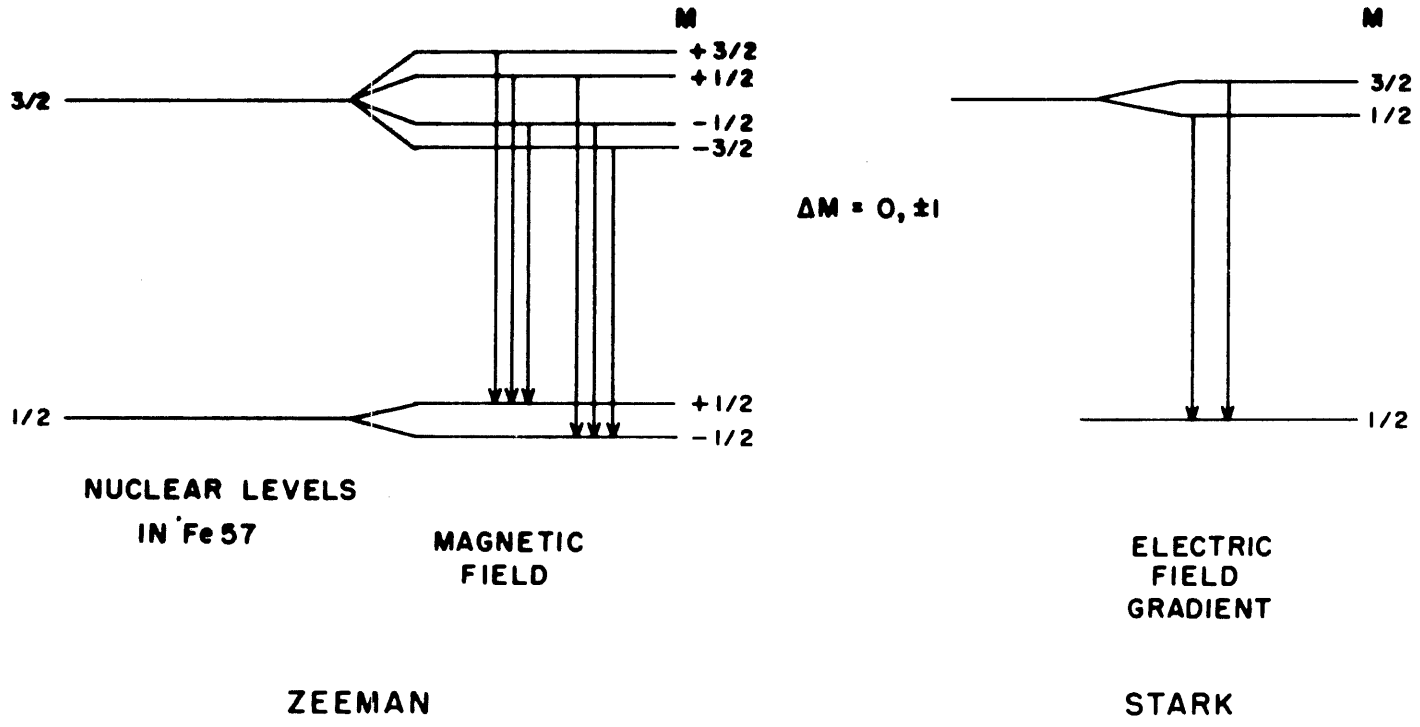


Figure 3 Nuclear Zeeman and Stark Effect

velocity-scanning system since the energies involved can be produced by a Doppler shift at a velocity of a few centimeters per second. There are two possible ways of doing this: a constant-velocity system and a constant-acceleration system. The constant velocity system has mechanical devices, such as lathes or a swinging pendulum, which move at a constant velocity and the spectrum is scanned point by point. This system has the advantage of being able to move heavy weights but is subject to electronic drifts and changes in efficiency of the system. For many experiments, a constant acceleration or a linearly increasing velocity system is used. A typical device for producing such a system is a loudspeaker which is driven by a parabolic-shaped wave. The derivative of such a parabola is a triangle and thus the loudspeaker continuously scans from a minimum velocity through 0 to a maximum velocity. A typical system is shown in Figure 4. These systems are useful for experiments that do not involve moving a large amount of apparatus since they continuously scan and are less likely to be susceptible to drifts in the efficiency from changes in electronics. The PDP-5 was used in this type of system.

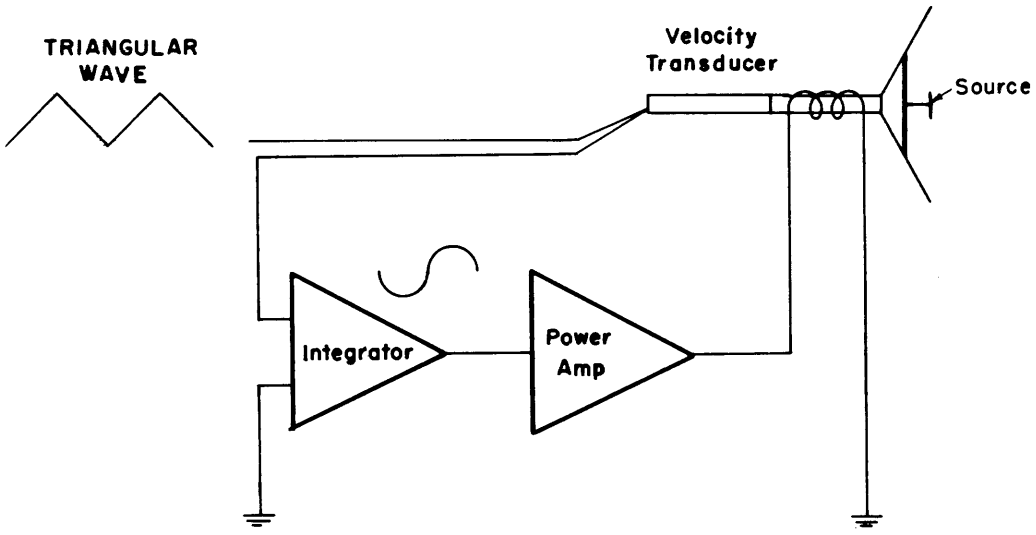


Figure 4 Mossbauer Velocity Drive

The PDP-5 was used as a storage and calculating device for the Mossbauer experiments. The overall experimental setup is shown in Figure 5. The drive and energy selection devices are shown on the left-hand side. The CADC or continuous analog-to-digital converter will be described later. The output of the two counters, "data" and "time," is strobed into the

computer as are the contents of the output register of the CADC. These are then stored in appropriate memory locations as a spectra is accumulated. It requires from 10 to 50 hrs to accumulate a significant set of data.

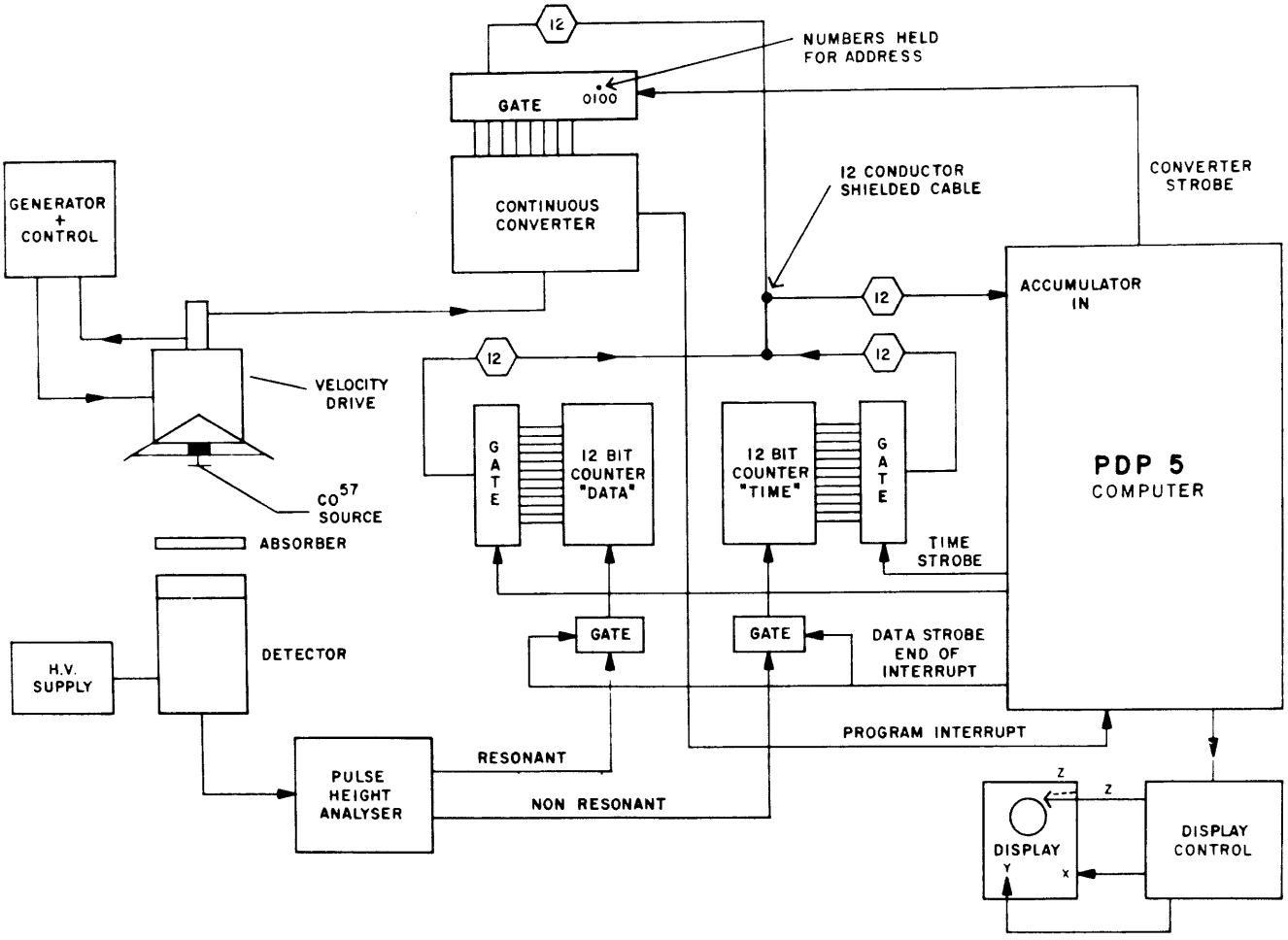


Figure 5 Electronics Block Diagram

The operation of a CADC is shown diagrammatically in Figure 6. The comparator switches state when the difference between the input analog or velocity signal and the output of the digital-to-analog converter changes sign, and thus the up-down counter continuously follows the input velocity. The digital number stored in the CADC register always represents the velocity. This converter is a 10-bit device of which the eight most significant bits are used as an effective address. When the eighth most significant bit changes, a pulse is generated which causes a program interrupt. This interrupt reads the contents of the CADC and the two

registers, "data" and "time," into the AC of the PDP-5 and they are stored in appropriate places in the memory. The system is reset and starts another velocity interval until the velocity has changed by 1/256 of the maximum excursion in velocity. The program-interrupt routine requires 324 μsec with a PDP-5; the average time between interrupts is about 400 μsec. The main reason for the length of time required to service the program-interrupt is because double-precision arithmetic must be used.

Initial experiments were performed using the program-interrupt routine system and were found to give satisfactory results. However, this was an inefficient use of the computer so that another system using hardware and the high-speed, data break channel was devised. This system is shown in Figure 7. In this system, the contents of a register determined by the contents of the CADC are strobed into a parallel adding circuit and are added to the contents of the incident registers "data" or "time." If an overflow occurs, the CADC is decremented by 1, and 1 is added to the contents of this register. A logic diagram is shown in the lower part of Figure 7. Thus the high-speed, data break channel continuously accepts information from the experimental system, modifies the existing registers and restores the new number in the computer. One data break cycle is required for each channel, and one cycle is free while the carry and add of the parallel adder are performed. The total time has been reduced from 324 μsec to 12 μsec. This high-speed, data break system has been found to be much superior to the program-interrupt system in that many other possible functions are possible with the computer at the same time that Mossbauer spectra are being accumulated.

The programs for these systems are different from program-interrupt routine systems, since the relatively restricted time intervals meant that no other devices could be used. With the program interrupt, the only other function performed by the computer was the display of the registers accepting data on an oscilloscope using a Type 34 Display. The typeout and normalizing routines were separately activated after data accumulation had been finished. The normalizing routine took the ratio of data to time and this was the desired spectrum. With the high-speed, data break channel, there is a great deal of time available for performing other experiments. This allows the use of the keyboard for controlling the display, typeout, and normalization routines as well as many other small routines which were quite independent of the Mossbauer spectrum. A map of the computer memory is shown in Figure 8. Most of the instructions are

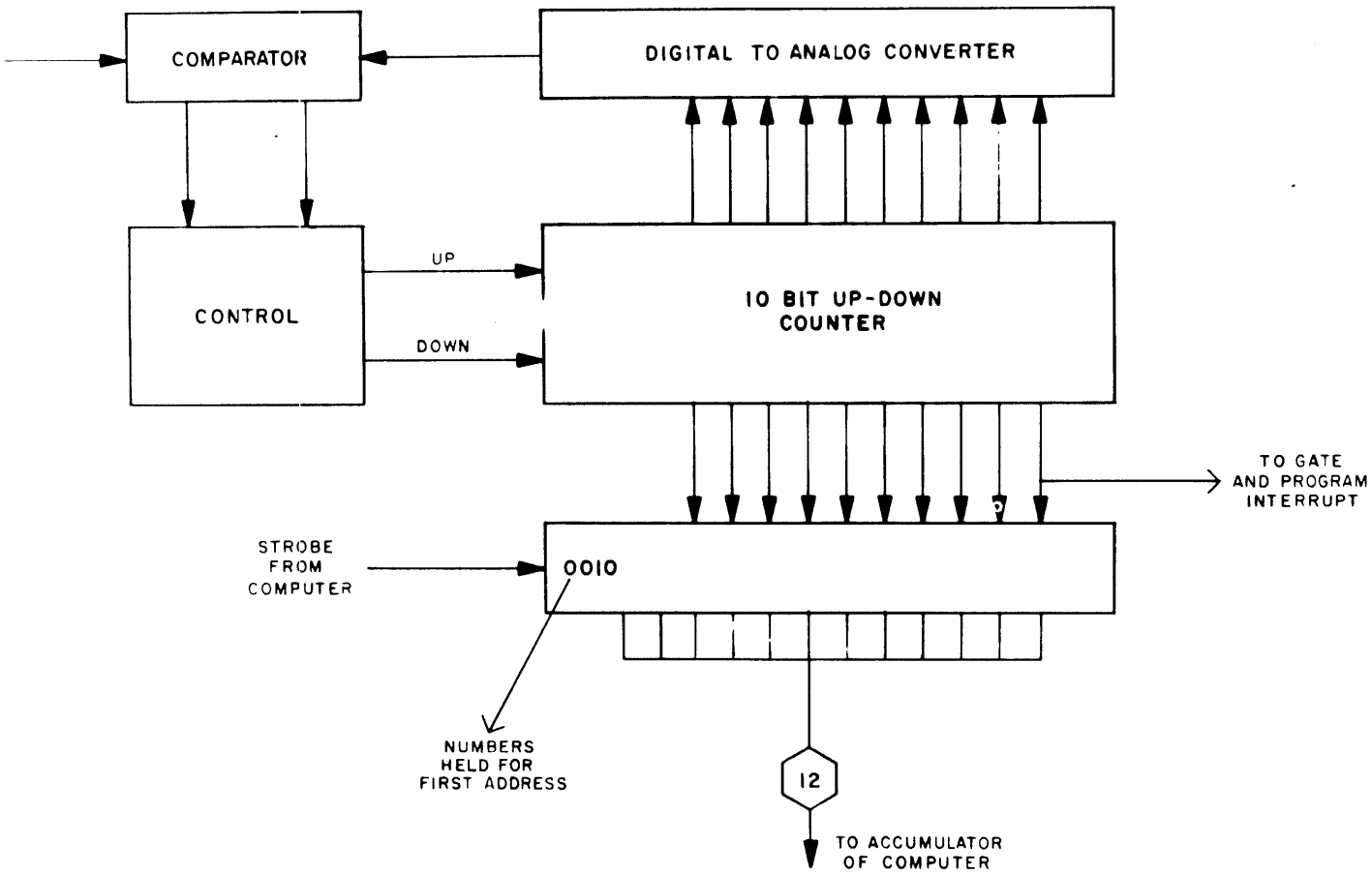


Figure 6 CADC Block Diagram

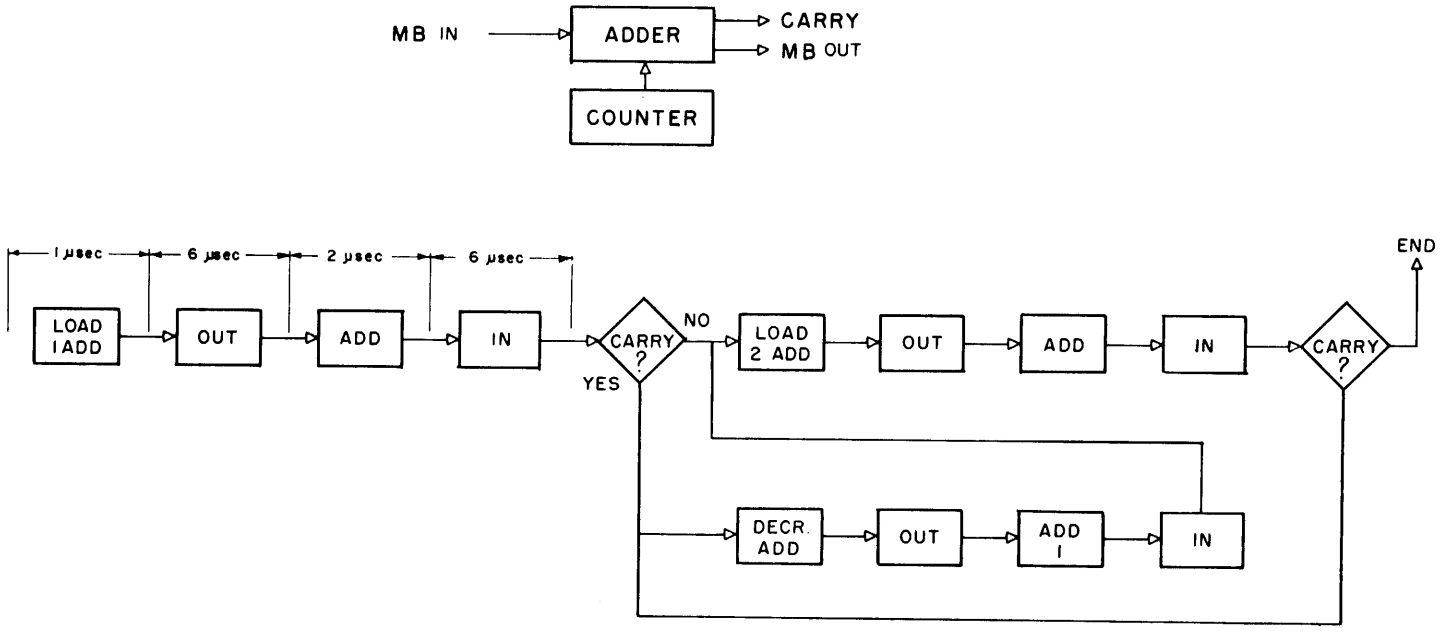


Figure 7 High-Speed Data Break

involved in the typeout and display of data rather than in the accumulation of data from the experimental system. Even in the program-interrupt routine, the amount of data manipulation before storage is quite small. The total storage required for the program and data is approximately half of the total memory. Thus there is room for additional routines which run simultaneously but are in no way connected to the Mossbauer experiment.

FUNCTION	LOCATIONS	
	PROGRAM INTERRUPT	DATA BREAK
DISPLAY	214-267	1000-1120
OUTPUT	400-577 600-777	1200-1377 1400-1577
DIVIDE	5000-5200	1600-1777
SHIFT		2000-2100
SCAN AND MONITOR		200-740
DATA	1000-3000	5000-7000

Figure 8 Table of Memory Locations

The results of a typical Mossbauer spectra are shown in Figure 9. The hyperfine structure of iron is illustrated in this figure. The typical 6-line spectrum is readily observable.

Figure 10 shows the Mossbauer effect in iron sulphide (FeS_2) a mineral known as pyrite. The Mossbauer effect spectra are taken as a function of orientation in the crystal. These spectra are very different because the electric field gradient which is causing the splitting lies along the 111 direction. From these results, the magnitude and direction of both the electric and magnetic fields at the nucleus may be determined. This information is very useful in the extension of knowledge of solid state properties.

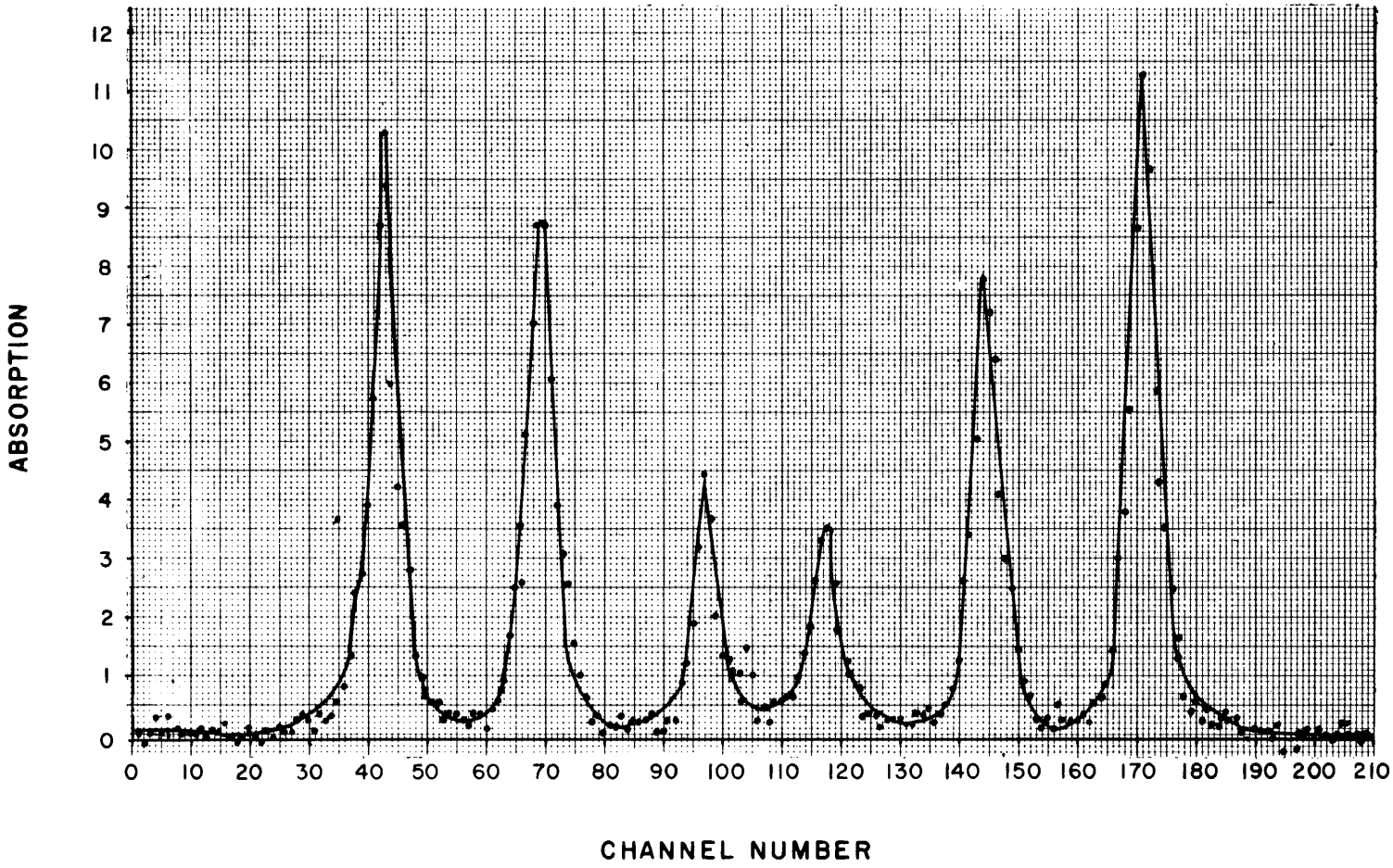


Figure 9 Mossbauer Spectrum of Iron Metal

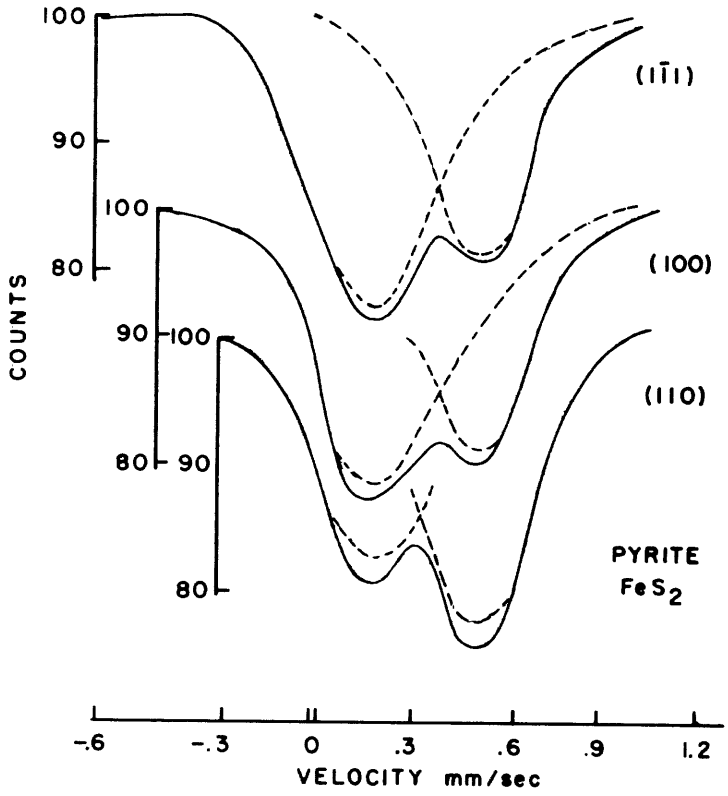


Figure 10 Mossbauer Spectra of Pyrite

A HIGH-SPEED MAN-COMPUTER COMMUNICATION SYSTEM

by

Earl H. Brazeal Jr. and Taylor L. Booth

University of Connecticut
Storrs, Connecticut

Abstract This paper describes a high-speed, man-computer communication system which is presently being employed on a research program at the University of Connecticut. This program is aimed at determining the best way to integrate a human operator and a small computer (PDP-5) in a signal processing and detection system.

The man-computer communication system consists of an electrostatically deflected cathode ray tube, a light pen tracking unit, and a display control unit. The I/O facilities of the PDP-5 are used to full advantage to control the basic operation of the system. This paper describes the peripheral equipment used and its unique operational features. The research applications are also discussed.

INTRODUCTION

One of the current areas of research at the University of Connecticut is directed at determining the best way to integrate a human operator and a small data processor of the PDP-5/8 class into a signal processing and detection system. For this research it is necessary to have a highly flexible, high information density communication link between the operator and the data processor. This paper describes a cathode ray tube display system which has been developed at the University of Connecticut to meet this need. In developing this display system the following design requirements were established:

1. High-density information display capability.

2. Flicker-free operation for all displays.

3. Minimal memory storage requirements for information to be displayed.

4. Minimal direct program control for display and light pen tracking operations.

5. Complete flexibility of display options under computer control.

6. Automatic light pen tracking unit that is able to follow light pen at normal writing rates.

7. Read and writing of information under light pen control.

To achieve these design requirements we found it necessary to develop a separate display system which operates in parallel with the PDP-5. The system consists of a high-speed display oscilloscope, a display logic control unit, and a light pen tracking unit. Control of the display unit and information interchange is maintained through use of the IOT and data break capabilities of the PDP-5. A block diagram of the basic system configuration is shown in Figure 1. The present operator display consists of a 72 x 85 matrix of points which is presented at a rate of up to 32 frames per second. The position of the light pen is also displayed on the same screen by multiplexing the operation of the display control unit and the light pen tracking unit. Operational modes such as those which are available on the DEC Type 340 Display unit can be implemented by use of programmed subroutines stored in the PDP-5.

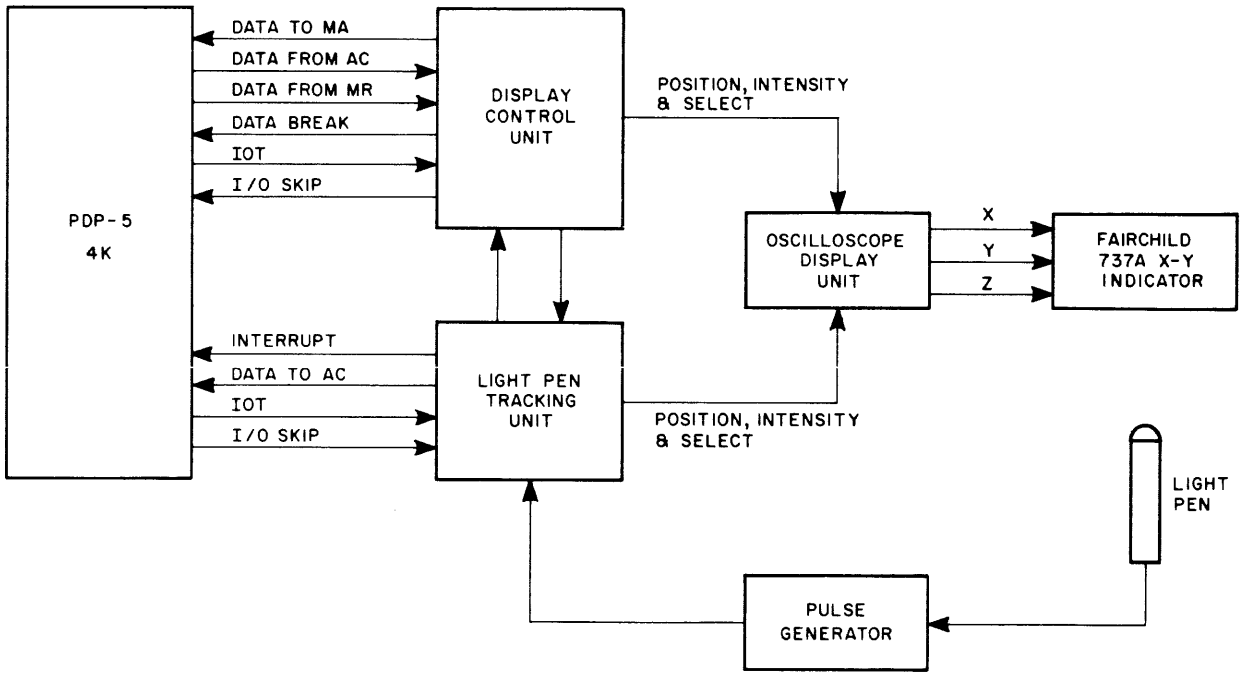


Figure 1 High-Speed Display System

The following sections describe the operation of this unit and compare its operational capabilities with other display systems.

THE DISPLAY PROBLEM

Conventional digital display systems require at least one data word to write a single point at a given X-Y location on the display oscilloscope. For example, the 34B Display associated with the PDP-5 requires two 10-bit words to specify a single display point. This is a good example of information redundancy, since 20 bits are required to specify each point of the display. This also means that the programming required to display a point usually runs into several instructions, which further limits the information density and plotting speed which can be handled by the system.

Another point which determines the usefulness of digital displays is that of flicker. If no flicker is to be noticeable to the viewer, the total display must be presented at a rate of 14 frames/sec or higher. In our system we have 6120 points per frame. Thus, we must be able to display points at a rate of at least 12 μsec/point if our display is to be flicker free. It is impossible to obtain this plotting rate with the present 34B Display system.

Thus, in designing our system we faced the problem of developing a method which could display vast amounts of data as fast as possible. This was accomplished by developing minimum redundancy techniques for storing display information in core memory and designing several auxiliary display control units which reduce the amount of programming required to display this information.

The main properties of each unit of the resulting display system are described below.

THE DISPLAY OSCILLOSCOPE

The heart of our display system is a Fairchild Type 737A X-Y Indicator. This unit is a high-speed, 17-inch, electrostatically deflected oscilloscope, which is capable of plotting points at a speed of 2 μsec/point.

With this high-speed capability, we encounter no problems in displaying at any rate that the display control unit can provide. The 2-μsec/point capability is a vast improvement over the 50-μsec/point writing rates of the small-bandwidth, large-screen magnetically deflected display indicators used in most of the present display systems.

The 737A X-Y Indicator is controlled by two 10-bit D/A converters which supply X and Y position and an input gating circuit which controls the displaying of a point. These inputs are generated by the display control unit which serves as the interface unit between the PDP-5 and the display indicator inputs.

DISPLAY CONTROL UNIT

In designing the display control unit, we realized that the most redundant information in any display is the position of the data point. The real information in the data is not the position of a particular point but whether it is to be displayed or not (considering a 2-level display only). So, we decided to use synchronized counter registers in the display control unit to determine point positions external to the computer. Thus, it was no longer required to store all of the redundant positional information in the core memory of the PDP-5. This resulted in a reduction of 96% in the amount of core memory required to store a given display.

The first experimental display control unit incorporated some computer control in order to maintain synchronization. Display information was stored in an "information matrix" (core memory locations 6000_8 - 7000_8). An IOT checked to see if the display unit was ready for a new data word. If it was, a small subroutine would call up the word from the "information matrix" and transfer it to a shift register in the display unit. Another IOT would then allow the display control unit to display each bit of the word as it was shifted out of the register. At the same time the horizontal position counter was advanced to provide the proper X location for the displayed point. Other IOT's maintained synchronization by clearing the horizontal and vertical counters. The position counters drove a CRT through D/A ladders to provide the position for the data points. From this it is seen that coding of data consisted of using each bit of a data word to tell the external logic whether or not to intensify some point defined by a previously determined X-Y position counter. As one can easily see, 1000_8 data words (512_{10}) defined an over 6000_{10} point display matrix! However this system still required a considerable amount of program control.

The new system accomplished the elimination of computer programming. The data break facility of the PDP-5 was incorporated to reach this goal. A method was devised to use one register (external) to specify data address information to the PDP-5 MA and at the same time determine

the X-Y position on the oscilloscope. Initial position was Y-X_1 where X_1 was the X position of bit 0 of the data word. Figure 2 shows the logical design of this address generating system. As can be seen, the lowest address is 6000 which occurs when all counters are clear implying that bit 0 of the data word located at 6000 will appear in the corner of the display. We arbitrarily chose the corner to be upper left. Now, as H CTR is incremented, the MA doesn't change until H CTR reaches 12 or some integer multiple of 12. At this time, the mode 6 counter is incremented so that the MA becomes 6001. Data break calls up the contents of 6001 and the process is repeated. Note that no computer control is necessary since all control signals are generated externally. We call this the "fully automatic" mode.

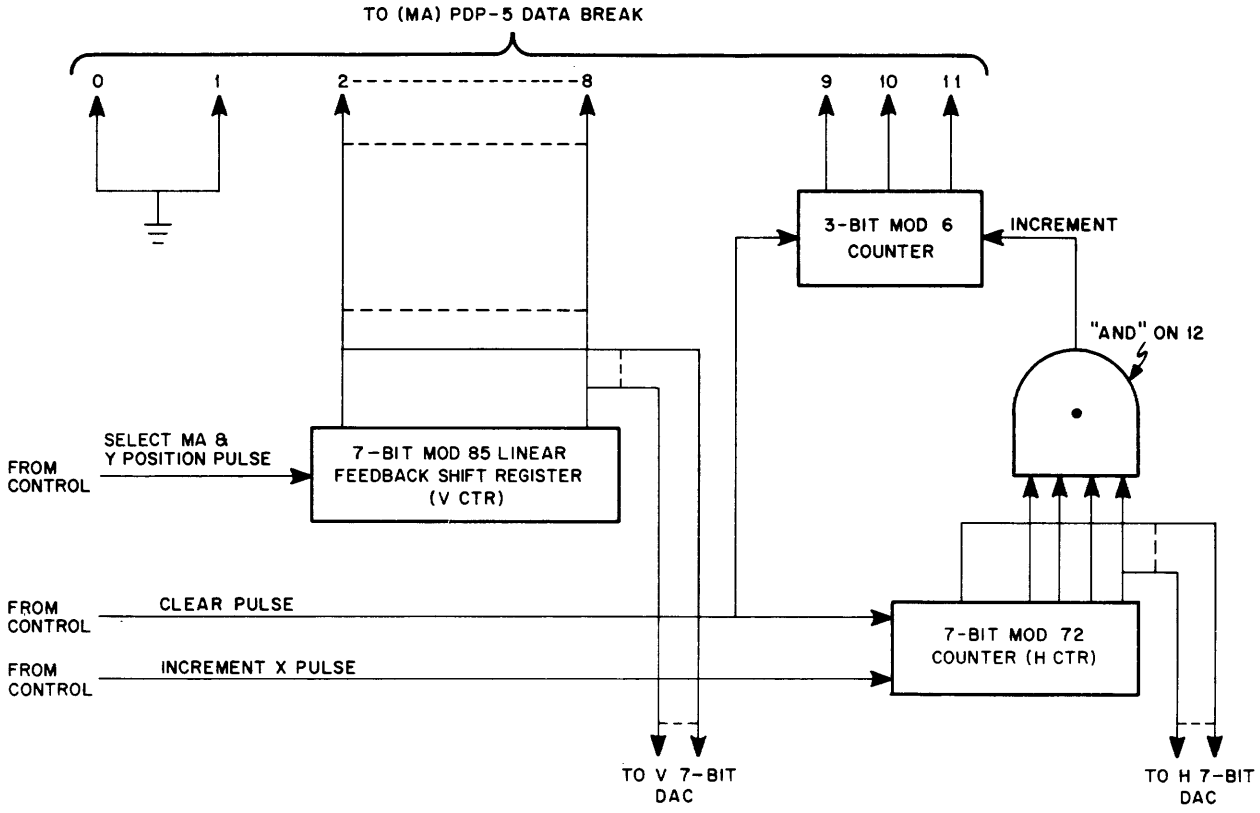


Figure 2 Memory Address and X-Y Position Logic

Since the purpose of the research is to strive for an optimum system, we felt it important to maintain a high degree of flexibility in the display control unit. The explanation above describes only one of the optional operating modes. We have designed three optional features (actually six) into the system that can be computer chosen under program control or switch selected by an operator. These options are designated A, B, C and are described below:

A Option - Allows the display control unit to halt after the contents of one data word has been displayed. A flag informs the PDP-5 of this condition.

A' Option - The display control unit will automatically restart a new cycle after displaying the contents of a data word.

B Option - Allows the display control unit to generate the memory address of the next data word by a pseudo-random sequence (automatically providing X and Y position). This option will allow investigation of random scan in an effort to reduce display flicker. This can only be used with "full automatic sweep" (C option).

B' Option - Allows the display control unit to generate the memory address of the next data word in sequential fashion (line-by-line sweep) i.e., automatically increment the Y-position counter.

C Option - Full automatic sweep. Utilizes the data break facility of PDP-5 as described above. This option frees the PDP-5 to perform other tasks.

C' Option - Semiautomatic sweep. When utilizing this option, control (via programming) must be provided by the PDP-5. The computer supplies the data word to the display control unit via the AC along with IOT signals. A greater level of control over displayed information is established with this option. To insure synchronization of data and X, Y location, B option is not used with C' option. Several other IOT's from the PDP-5 must be interpreted by the display control unit, such as:

1. Clear vertical position register to zero (used with C' option).
2. Clear horizontal position register to zero (used with C' option).
3. Remove display control clock inhibit (start).
4. Interrogate state of display control (A option).

The entire control unit incorporates one mounting panel of 60 FLIP CHIP modules. A block diagram is shown in Figure 3.

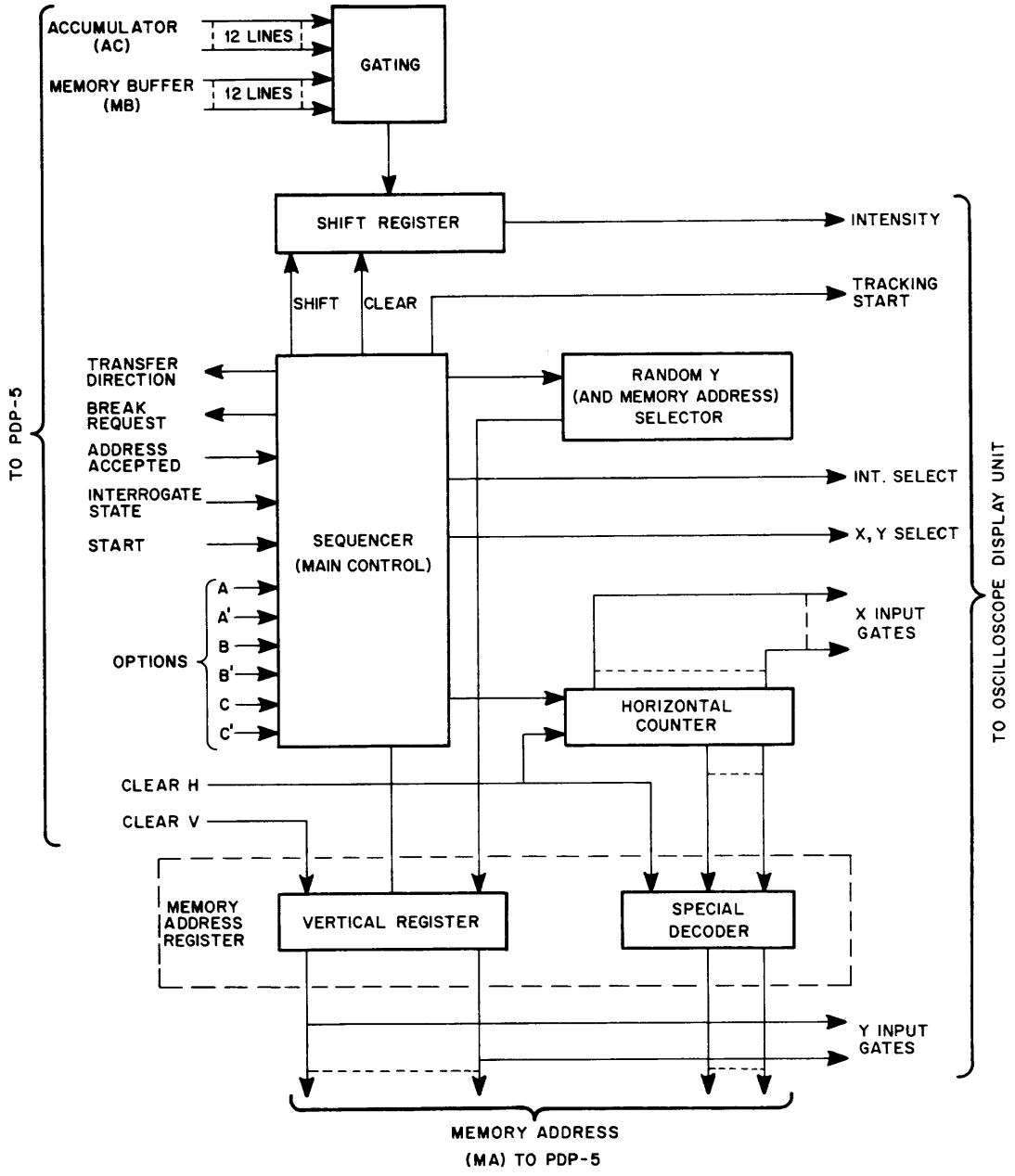


Figure 3 Display Control Unit

The actual operation of the display control, with its many operating modes, may be more easily understood by referring to the flow diagram (Figure 4). At the present time we are displaying a matrix of 72 x 85 points, as the flow diagram indicates. This can be expanded, however, since the vertical counter has a 7-bit resolution. If we expanded horizontally it would have to be in groups of twelve bits at a time.

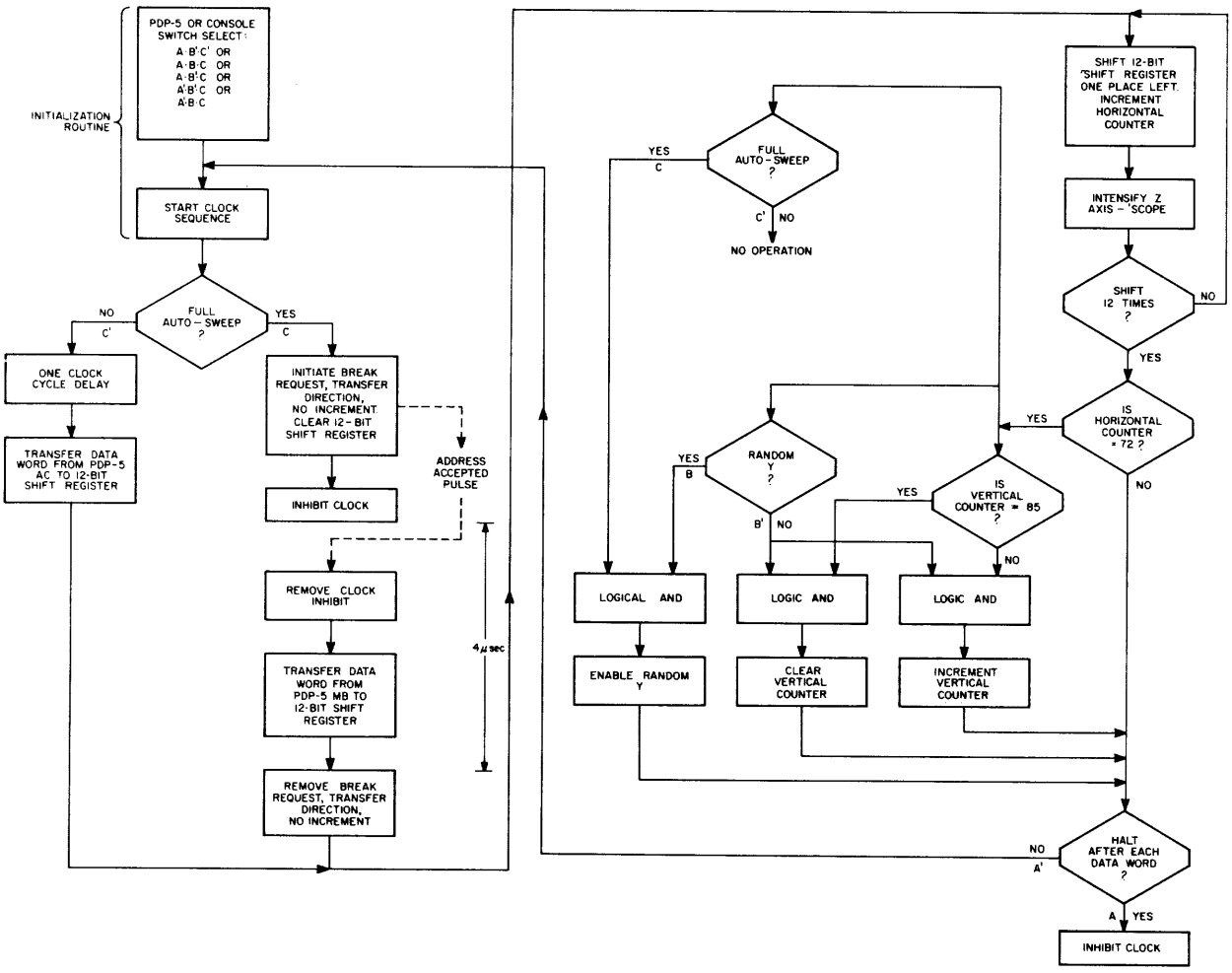


Figure 4 Flow Diagram-Display Control

It might be mentioned here that this is a constant frame rate display with an upper bound on information or point density. Figure 5 graphically illustrates the comparison of a constant frame rate system with a conventional point-by-point system. The major advantage of constant rate is that of flicker reduction. Note that the conventional system enters the "flicker region" at about 2000 points/frame. This value was calculated assuming that 36 μsec is required to display one point. Our new system can display a word in about 60 μsec, which includes 6 μsec for data break. At this word rate, the system is able to display the entire information matrix (510 words) in about 30-33 msec maximum. If time-sharing with the light pen tracking unit is included, the frame rate drops by a factor of 2 or so, or 15-17 frames/sec.

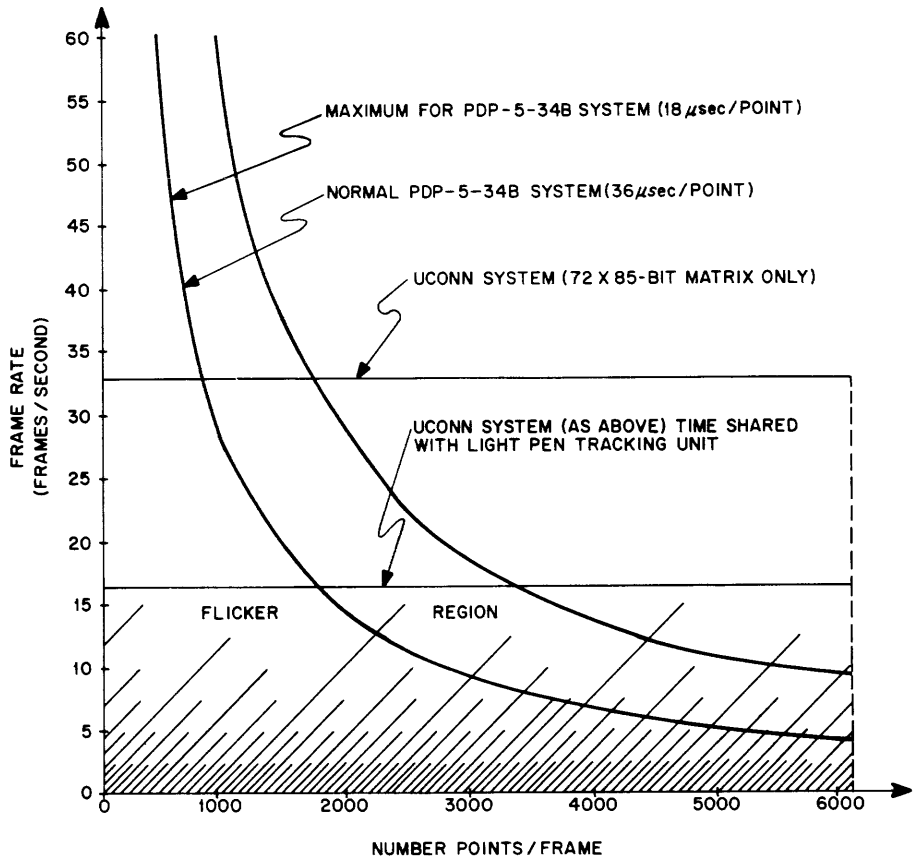


Figure 5 Comparison of Data Displays

LIGHT PEN TRACKING

A light pen feedback system alone can "close the loop" in a man-machine system. However, many practical problems arise which limit the usefulness and the information rate of the device. For instance, it is difficult for an operator to remember the last point the light pen picked up unless the machine tells the operator by modifying the displayed data. Also, some conventional systems display special symbols just for light pen use, which results in limiting information feedback to the machine.

In our display system, we want the capability of knowing the exact location of the previous point which the light pen picked up without modifying the displayed data. To accomplish this aim, a light pen tracking unit was designed and constructed from FLIP CHIP modules.

Upon command, the tracking unit generates a small square (made up of four dots) on the oscilloscope screen. When the operator holds the light pen up to the square, a portion of it is

detected and, based on the location of this portion, new coordinate positions of the square are calculated by the tracking unit. Thus, the operator can move the square to any part of the oscilloscope screen within the range of the tracking unit's vertical and horizontal registers.

Using this system, displayed data is modified by the operator by placing the tracking square over the data point(s) in question and pushing a button (wired to program interrupt or an I/O flag) which instructs the PDP-5 to perform the desired modification (store, erase, etc.).

The slow response time of the human eye allows time-sharing of the oscilloscope by the tracking unit and the display control unit. The resulting image appears as a superposition of the tracking square upon the data matrix (see Figure 5).

Upon reception of the light pen pulse, the tracking unit clock is inhibited and generation of the square is halted. An "enable" pulse is generated at this time which decodes the registers used for square generation into a new set of X- and Y-coordinates. This new set of coordinates corresponds to the new center position of the square. If a light pen pulse is not generated, the tracking unit halts automatically after n, $0 < n \leq 63$, number of squares has been written. The X and Y positions of the square are not changed under this condition.

Other I/O commands interpreted by the tracking unit are:

1. Start tracking.
2. Sense state of tracking unit.
3. Transfer X-coordinate into PDP-5 AC.
4. Transfer Y-coordinate into PDP-5 AC.

Commands 3 and 4 are carried out by a subroutine after detection has occurred since at that time an interrupt is sent to the PDP-5 by the tracking unit.

The block diagram of Figure 6 and the flow diagram of Figure 7 show more clearly the operation of this unit.

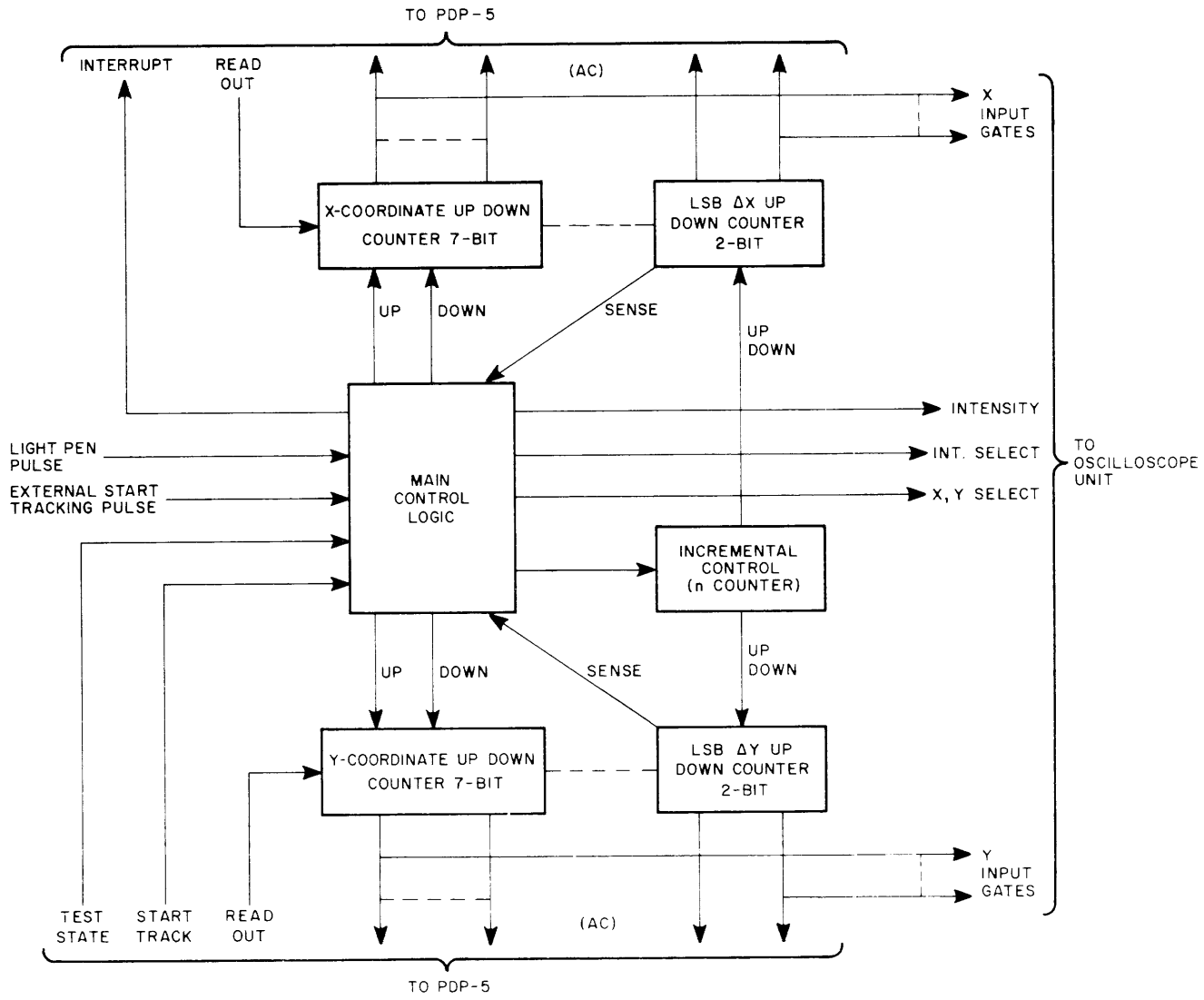


Figure 6 Light Pen Tracking Unit

CONCLUSIONS

This paper described a high-speed CRT display system which has been developed at the University of Connecticut for use with our PDP-5. The development of this system was necessary because the conventional display systems which were available could not handle the high-information densities encountered in our research work in the area of man-computer information processing systems. By fully utilizing the IOT and data break facilities of the PDP-5 we were able to obtain a flicker-free display while at the same time achieving a 96% reduction in the amount of core memory required to store display data. Since our system operates in parallel with the PDP-5 we also were able to drastically reduce the amount of program time required to service the display system.

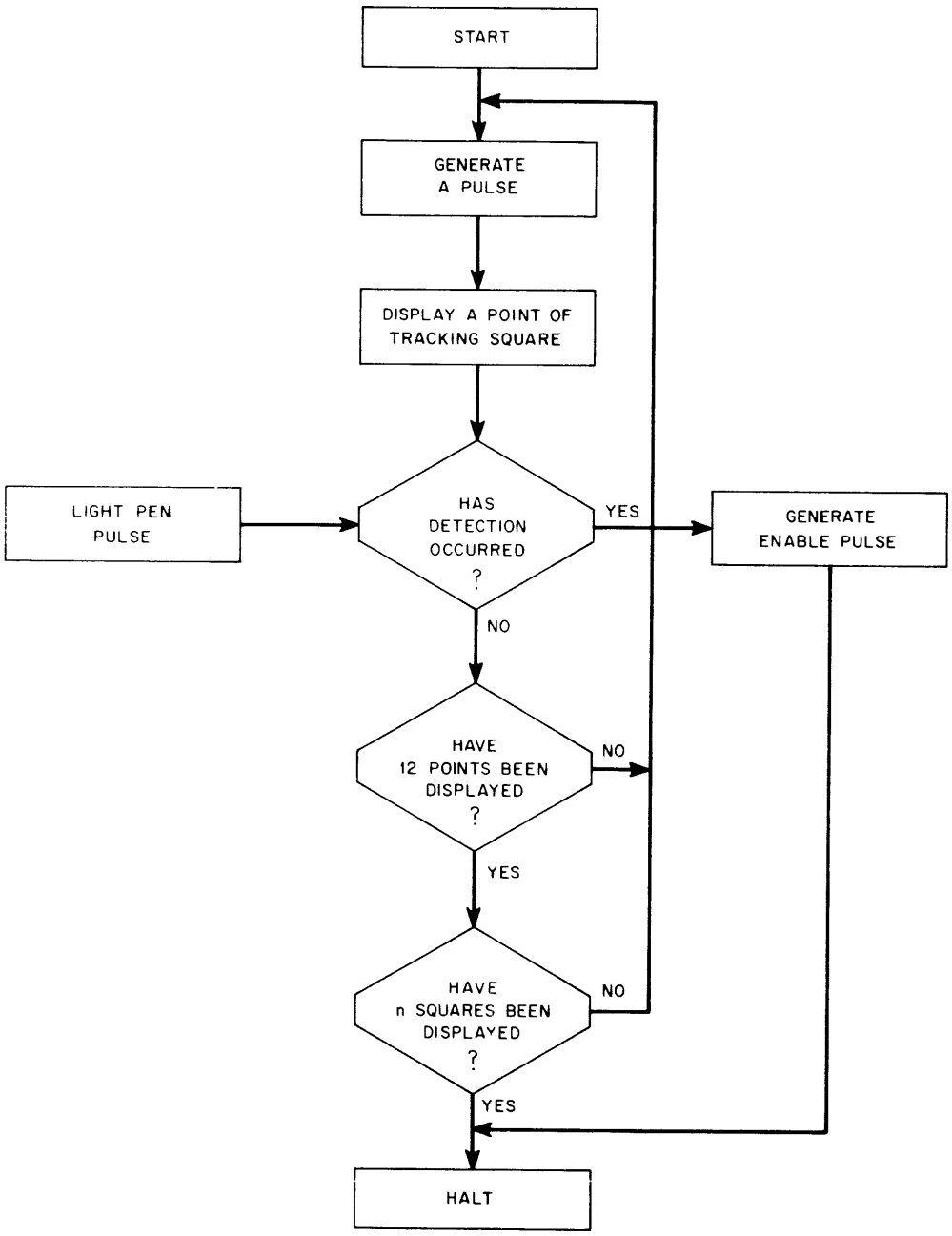


Figure 7 Flow Diagram Light Pen Tracking Unit

Another feature of the system is that the complete display is stored in core memory in the same manner as it is displayed on the CRT. Thus we can process the display data under program control while it is being displayed. In addition, we are able to use subroutines to provide display options such as plotting vectors, writing alphanumeric characters and reading and writing of information under light pen control.

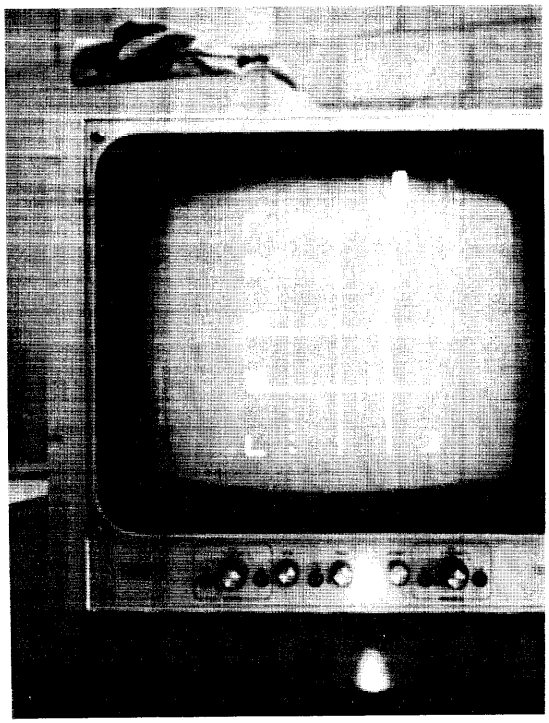


Figure 8 Test Pattern on 737A Indicator

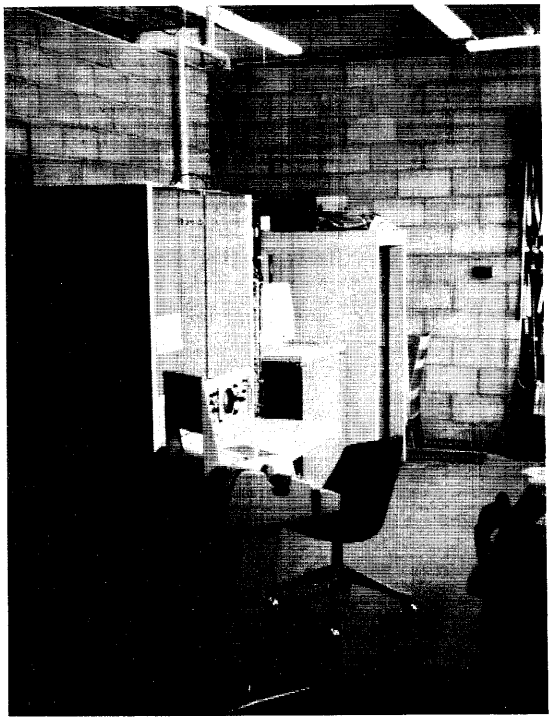


Figure 9 Experimental Facility PDP-5, External Logic, Display Booth

There is very little dependency between the display system and the main computer program except when the display system is operating in the semiautomatic mode. In fact, in the fully automatic mode, the execution of a computer program is completely independent of the display system.

The display system has been very useful in our research work (see reference), and it is operating well within its maximum capabilities. Thus it would be an easy matter to expand the display size or to add other aids to help the operator in the man-computer communication problem.

ACKNOWLEDGEMENT

Acknowledgement is due the Office of Naval Research which supported this research through a prime contract (NOnr 2512(00) with General Dynamics/Electric Boat as a part of the SUBIC (Submarine Integrated Control) program.

REFERENCE

Booth, T., Glorioso, R., Kaufman, H., Levy, R. "An Experimental Investigation of An Integrated Man-Computer Signal Detection System" 6th Annual Symposium of IEEE PTG on Human Factors, Boston, Massachusetts, May 1965.

PDP-5 PROJECTS AT LAWRENCE RADIATION LABORATORY

by

Sypko Andreae

Lawrence Radiation Laboratory
Berkeley, California

Abstract PDP-5's at LRL are now being used in several different applications, the main one being data acquisition and limited data analysis of experimental data. Those PDP-5's work on line with the experiments in the accelerators. Examples of instruments and interfacing used in these experiments are given. Progress in the development of simple techniques for data transmission from the PDP-5 to the computer center and back are discussed. The main configurations of two other PDP-5 systems are explained. One is a PDP-5 with many I/O devices working on line to experiment in the 88 Cyclotron, and another is a PDP-5 with a 630 System which will be used as a message switching center during laboratory experiments in the study of human behavior.

Many PDP-5's are now used in a variety of functions at Lawrence Radiation Laboratory in Berkeley, California. I would like to describe several systems of which PDP-5's form a part. All systems that we are concerned with have a data-acquisition function. A secondary function is data analysis. In cases where the data flow is very large, some processing is performed to achieve a comprehensive display of reduced data on the oscilloscope. But, in other cases, there is sophisticated data reduction, which is controlled by an experimenter in the true on-line configuration.

An example of this is the PDP-5 system in the 88" Cyclotron, which was built and programmed by Lloyd Robinson. Figure 1 shows the layout of this system. During an hour's run, data is gathered in the 4096-channel pulse-height analyzer. During that hour, data of a previous run can be processed. The switch box is used to achieve convenient access to the different software systems. The switch box performs two functions:

1. It selects a program which is then read from the library tape into the lower 4K memory.

2. Once a program is loaded into core, it selects certain functions from that program.

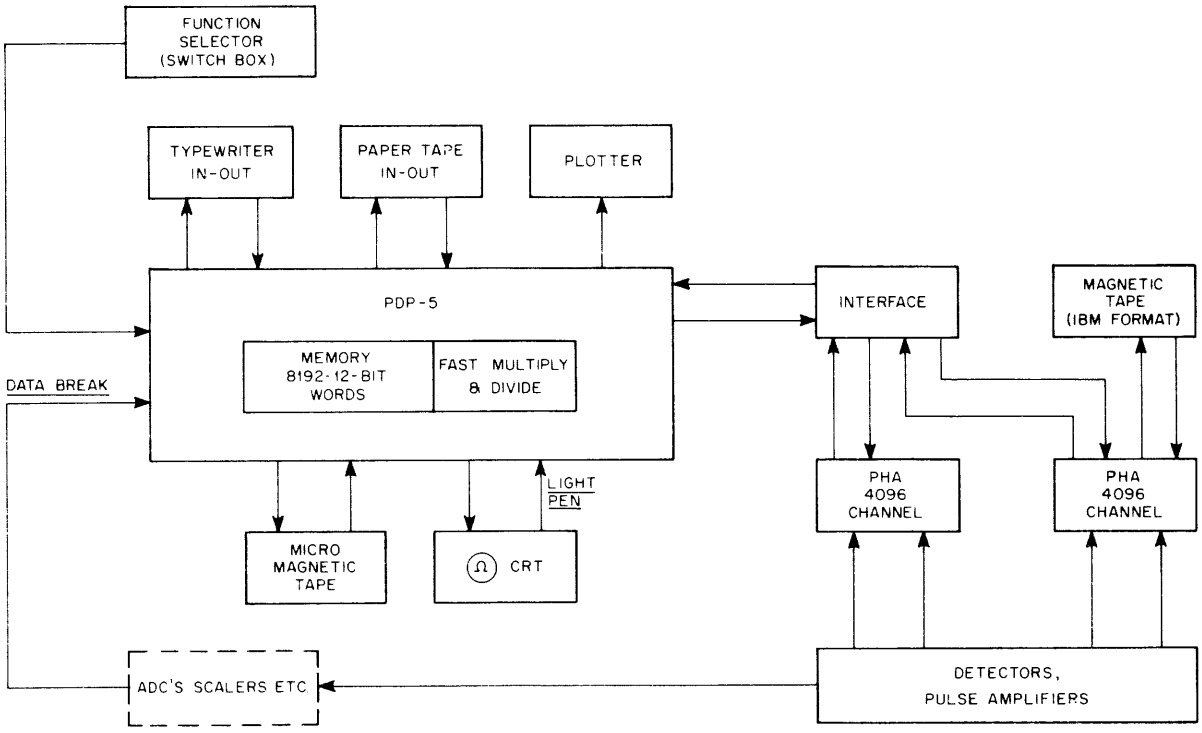


Figure 1 Block Diagram PDP-5 88" Cyclotron System

In the first case, a bootstrap program of 28 wired-in instructions are stored in the core using the data break mode. This bootstrap teaches the PDP-5 how to load in a larger loader from the micro system tape, which in turn loads the program selected with the switch box. This is done automatically by setting the switch in the desired position and depressing the load button. After the program is in core, the switch can be set to select one particular function. When the function push button is depressed, an interrupt is caused in a small executive program, which calls and starts the selected function.

The data is stored in the upper 4K of memory and is also recorded on the second microtape. The data consists of 4096 words in which numbers that are seldom larger than twelve bits (maximum size of words is 20 bits) are stored. The overflow of the few numbers that exceed 2^{12} is stored in overflow tables located in the lower 4K memory.

With light pen and keyboard, the experimenter can guide the data processing of information displayed on the scope to wherever he desires. Of all images--mainly energy spectra--the y-coordinates are supplied by the PDP-5, but the x-coordinates are supplied by an external

sweep generator to achieve less flicker and easy y-scale manipulation (magnification and origin relocation).

Another PDP-5 system that is under construction is shown in Figure 2. It is a system for the Center of Research for Management Science of the Department of Business Administration of the University of California campus in Berkeley. In fact, their experiments have little to do with the experimental work going on at the Lawrence Radiation Laboratory, except that they will use similar hardware.

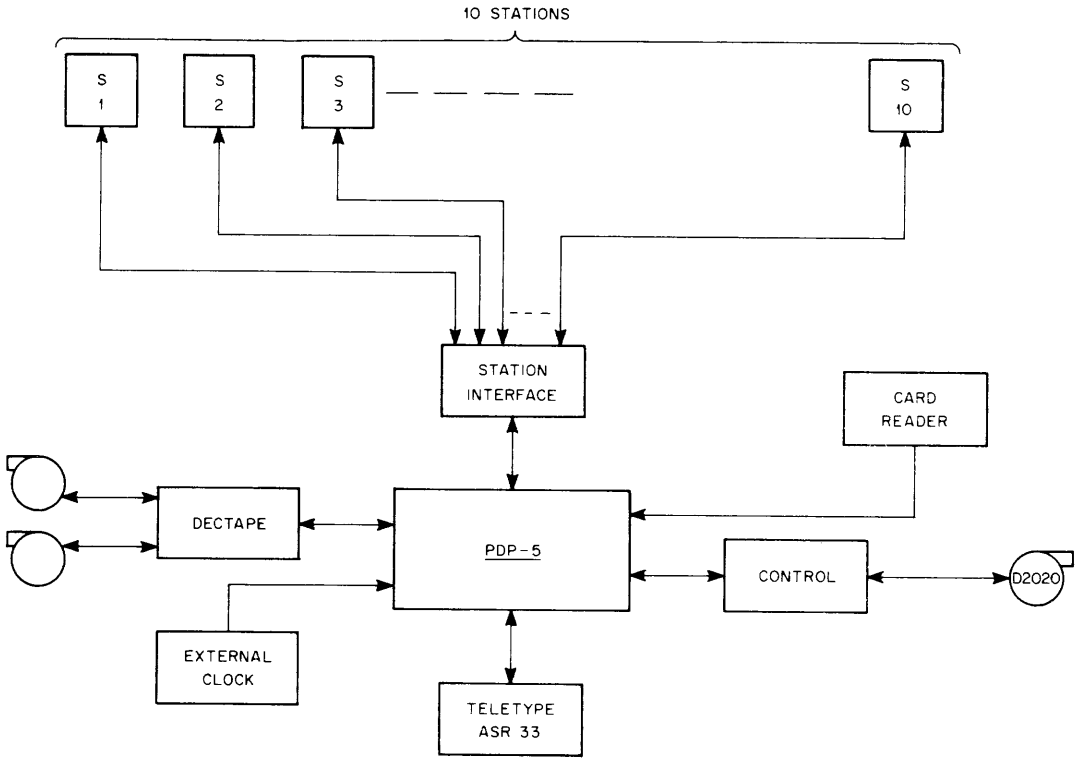


Figure 2 Management Science Laboratory System

This system will be used for communication experiments among subjects placed in a specific business game situation. The main aspect of each game situation, from the system's point of view, is that the communication is limited to channels of well-known characteristics, in this case, Teletypes.

The main functions of this system are:

1. Data collection of all internal communications.

2. Data transfer control between any station to any other station or group of stations.

3. Off-line communication with the computer center or limited data processing in the PDP-5 itself.

The challenge is to design the system so that it is tolerant to human failure and that the subjects like to use it. In the first period it will function mainly as a message-switching center where free English can be used by the subjects. Another more limited mode of communication between subjects is formed by the use of Tree language, where subjects are forced to use prearranged parts of sentences according to a prearranged set of sequences.

A more ambitious plan is an attempt to measure human decision abilities and judgment. It calls for a game where a subject is confronted in a competitive situation with an artificial counterpart. The counterpart is a software robot of which the decision behavior pattern is well known.

A 630 System will be used to implement the message-switching center. The construction of the whole system, including all of the software, is still in its initial phase.

Figure 3 shows the general layout of a system used in counting physics experiments. To the left are the data sources; namely, scintillation counters, spark chambers that feed into a vidicon system, and spark chambers of the wire variety that feed into a wire-chamber core scanner. In the top left is a box called "fast electronics decision logic" where decisions are made, whether the event is one of interest or not. All the information is fed via interfaces into the PDP-5; information can be displayed on the oscilloscope and the data can be stored on magnetic tape.

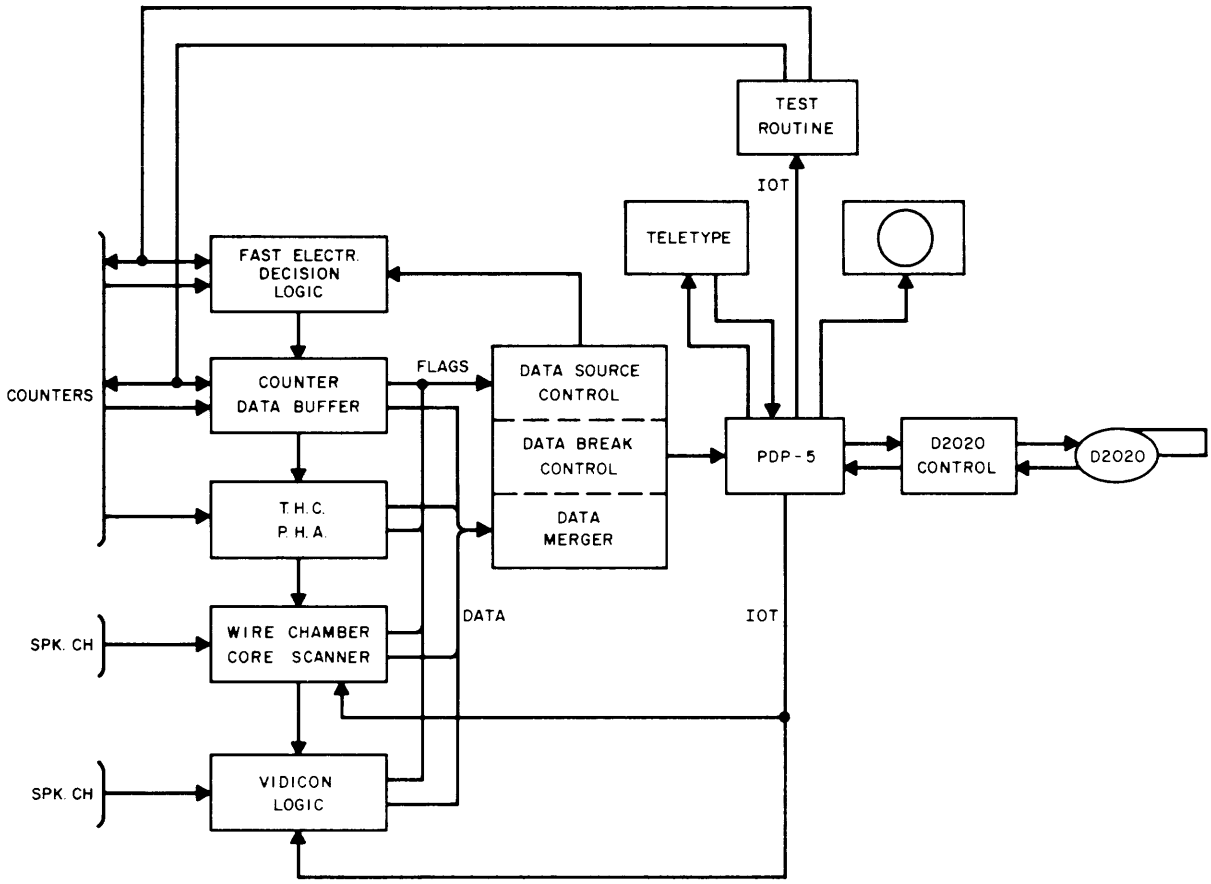


Figure 3 PDP-5 System

Figure 4 gives an impression of the cycle of the data flow, which in terms of time, may take several weeks or even a year because the data processing is off line. The evaluation by the experimenters may then originate a new experiment.

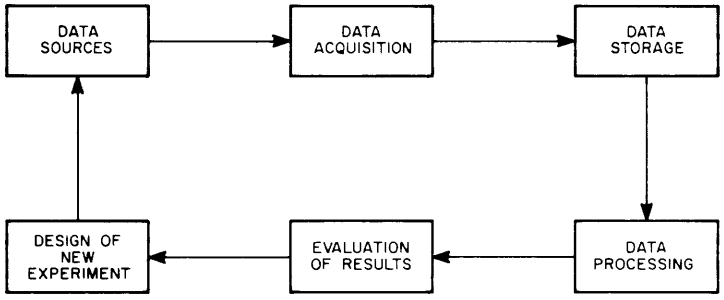


Figure 4 Data Flow Cycle

Figure 5 shows the same cycle, but now there is an additional inner cycle added to it which allows the experimenter to be in more direct contact with this experiment. The data sources here again are the vidicon, the wire-chamber core scanner, and the scintillation counter-memory buffers.

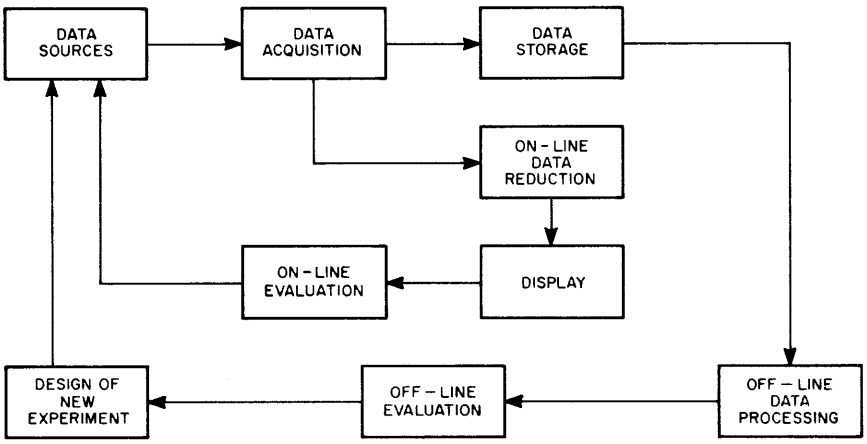


Figure 5 Direct Data Flow Cycle

The data acquisition is now done by the PDP-5 and its interfaces, and the storage is taken care of by magnetic tape units so that information can be fed to the computer center for off-line processing, etc. But there is an additional feature in the on-line data reduction which results in a comprehensive display and an on-line, low level, evaluation by the experimenter on the spot. To achieve the data deduction necessary to produce a number of comprehensive data displays on the oscilloscope, software is built which has many advantages over a pure hardware system with the same function. This is especially true when an experiment remains in the development phase for most of its lifetime.

The value of an on-line evaluation of the data is especially high in counting physics, where the experiments are often designed with specific goals in mind; namely, the detection of a limited number of phenomena. Since such experiments are built to allow events related to these phenomena only, it is very important to know if the data is of interest. If this were not the case, at least the experimenter can readjust his data sources immediately and try again. The feed-back loop is now shortened from weeks or years to the time it takes the physicist to form his judgment.

The choice of the size of the computer to be used on-line with the experiment depends on many factors. Some of them are:

1. The amount of data to be processed as one event and the number of events per time unit.

2. The level of sophistication one desires in the on-line data reduction in relation to the level of the off-line data processing.

3. Experience with on-line systems of the technical support groups; costs, etc.

Another choice to be made is whether to serve different experiments with one on-line computer or to assign one computer to each experiment. The first approach seems feasible only when 1) the experiments have settled down to a stable situation requiring little debugging time after an initial checkout stage, and 2) when the experiments run in phase. If the experiments run "out of phase" with respect to each other, difficulties arise in debugging one experimental program in relation to its hardware, while another one needs to be run to produce data. The consequence may easily be a complicated time-sharing organization, of which the executive program would be very difficult to realize satisfactorily.

It is important to realize that in many cases the people responsible for instrumentation have to acquire their experience on-line, too. With such a complicated system, it is very possible that the system workers have numerous detail problems and miss the opportunity to teach themselves how to handle this new tool, which is the computer.

The "data sources" consist of the vidicon system, which is a camera with associated logic that scans the image of a plate spark chamber. Similar spark chambers were used in experiments where the film camera was the only device to record the data.

When a nuclear event occurs, high voltage is applied across the plates and discharges (sparks) will develop wherever a charged particle left an ionized track behind in the controlled gaseous atmosphere. At the same time fiducial lights are flashed. A usable image of fiducials and sparks will remain only for 10 or 20 msec as a pattern of charges on the semiconductor face of

the vidicon tube. As soon as possible after the event, the image is scanned as parallel to the spark chamber plates as possible. The fiducials on the left side (the start fiducials) are indicators of the middle of each gap. The video signals originating from the sparks will be processed only after such a start fiducial is detected. The first spark found will start the first of a set of four or eight 12-bit scalers, each fed by a crystal-controlled oscillator. The second spark will enable the next scaler, etc. When the fiducial to the far right side (the stop fiducial) is detected, all scalers are halted. This means that the distance of each spark to the stop fiducial is measured and expressed in a 12-bit word. Several horizontal sweeps will go by before the next start fiducial is detected, at which moment the process repeats itself. But in the meantime, the information from all scalers is sent to the PDP-5. The PDP-5 will reset the scalers after absorption of all the data. In most systems that we have built, one out of five horizontal sweeps results (on the average) in a digitizing sweep. Of course, it is important to discriminate the fiducial signals from the spark signals, this discrimination is made possible by a system of time gates. The working of the time gates is based on the fact that we know whether to expect fiducial signals or spark signals during specific periods of time.

A monitor scope supplies the operator with the image seen by the vidicon camera. Superimposed on that, the image of the digitizing sweeps and the time gates can be made visible, making it possible to adjust the time gates for the selection of various video signals. In an experiment now running in the Bevatron, one of the two vidicon systems digitizes a composite picture of six views from a set of three spark chambers. The complete image shows 84 spark gaps, in each of which four sparks can be digitized. Since a spark location is expressed in 12-bit words, the vidicon produces $4 \times 84 = 336$ words of twelve bits per event. Where less than four sparks per gap are detected, zeros will appear in some of the 12-bit words of the corresponding group of four words. This may seem wasteful; but it should be pointed out that no additional identification of each spark address word is needed. This identification is already contained in the location of the word in each block of 336 words. This simplifies the processing of the vidicon data.

Another kind of spark chamber readout is the use of wire spark chambers in combination with a core store. The wire spark chamber discharges take place between planes of parallel wires or between a plane of wires and a metal surface. Each wire is strung through a ferrite core. In this way, it is possible to store at the same time, all information on wires hit by sparks.

Apart from the spark chamber wire, two "half-current" wires and a sense wire are strung through each core similar to a normal core memory. The total memory has a capacity of 4096 bits, and it is organized in words of 32 bits. After the event is over, the complete memory is read out, whereby the stored information is destructed (cleared). The readout procedure is organized as follows:

> Each word of 32 bits is read out and inspected for possible bits that equal 1. If all bits are 0, the next word is read out. If one or more bits are equal to 1, the word is stored in a 32-bit register, which is scanned from one end to the other by a word scanner. The word scanner will stop at every "1 bit," registering the location of that bit in the word. The location in the word can be expressed in five bits. The location of the word in the core stack can be expressed in seven bits. This information is assembled in a 12-bit wire-address word.

The address words are stored in a 3-word buffer which signals the PDP-5 when it holds three addresses. The PDP-5 then reads the buffer and after the completion of the data transfer, the wire core scanner searches for more "1 bits," until the core memory is exhausted; this fact is also signaled to the PDP-5.

The complete readout takes about 2 msec, depending on the number of wires hit by a spark. In contemporary experiments 2,000 wires are read out. The wires are distributed over six wire planes.

In the following, an essential outline is given of the interfacing used between the PDP-5 and the data sources. The first group of units is formed by the data break control, data source control, and data merger. It serves as a fast data channel between the data sources and the computer. The complete data break interface performs the following functions:

1. It brings the PDP-5 from the program mode into the data break mode, in which all memory cycles are available for data transfer to the memory.

2. It specifies the memory address for each data word to be stored.

3. It guards against memory overflow.

4. It recognizes flag signals and assigns priority to data sources.

5. It controls the sequential data gathering in words of twelve bits at a time by a scanning mechanism.

The data break control and data merger form standard pieces of equipment. The data source control is tailored to the specific needs of each experiment and is therefore not a standard piece of equipment. Figure 6 shows the flag-scanning procedure. The vidicon has the highest priority because it has the weakest memory since it is the image face on the camera.

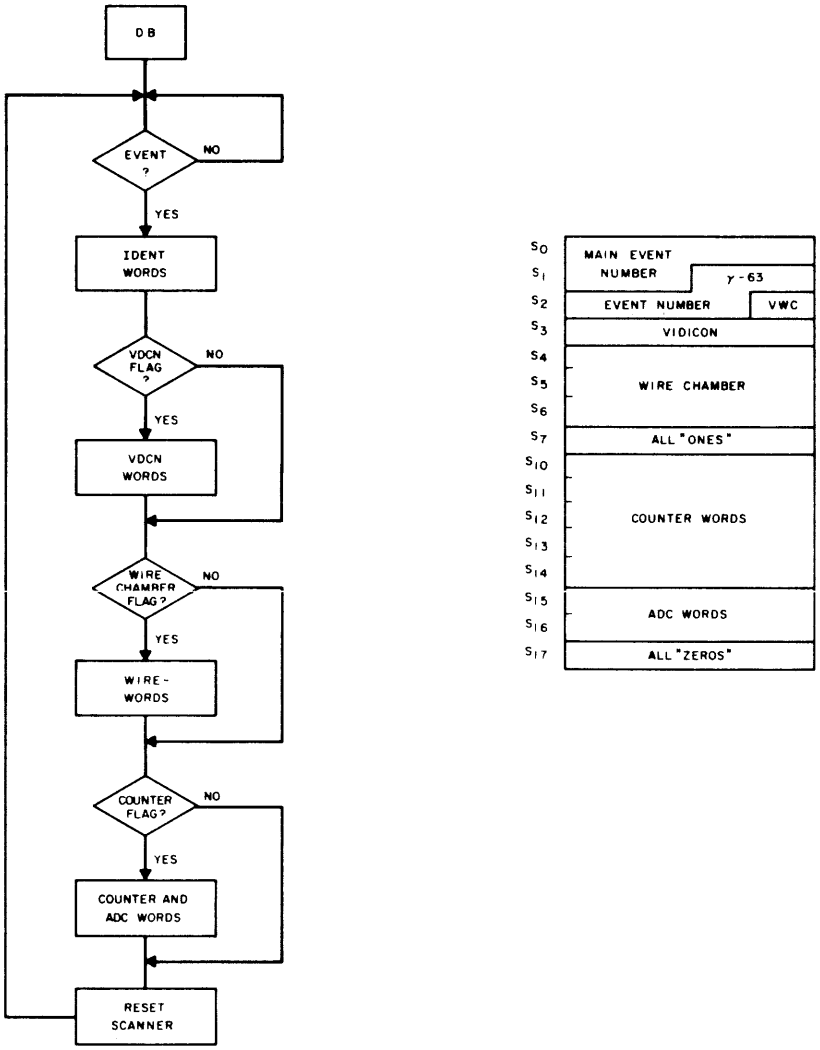


Figure 6 Data Source Control Scanning Procedures

As Figure 7 shows, the data sources, like the vidicon and the wire core scanner, produce two flag signals; namely, the long and the short flag. Due to the nature of these data sources, the data arrives in groups of several words separated by pauses in the transmission. The availability of such a word group is signaled by the short flag, which will be cleared after the word group is stored. The long flag, however, stays up as long as word groups may be expected and will be cleared after all data of the particular data source is transmitted.

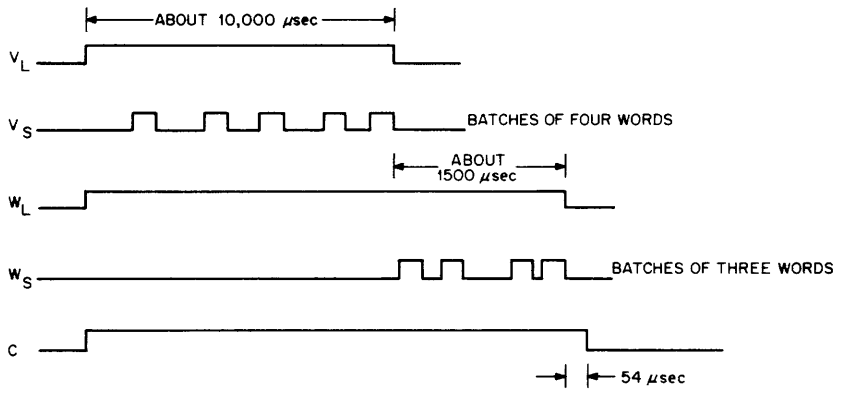


Figure 7 Double Flag System for Data Sources

In Figure 8, an example of data format is shown. Each event has an identification word of 36 bits that indicates which sources supplied the data. The vidicon supplies a constant number of words as do the counter logic and the analog-to-digital converters. The amount of data from the wire core scanner is dependent, however, on the number of wires hit by sparks in a particular event, so that it is necessary to insert a flag word (7777_8) to indicate the beginning of the counter data.

Other interface equipment consists of an oscilloscope display control (provided by DEC), a tape transport control, and a general-purpose facility called the "utility bin." The tape transport used is the Datamec D2020. The character transfer rate is 16.6 kc at a density of 556 cpi. All functions necessary to write and read records, to time the proper gap delays, to write "end of file," etc., are software controlled by a program of 125 instructions.

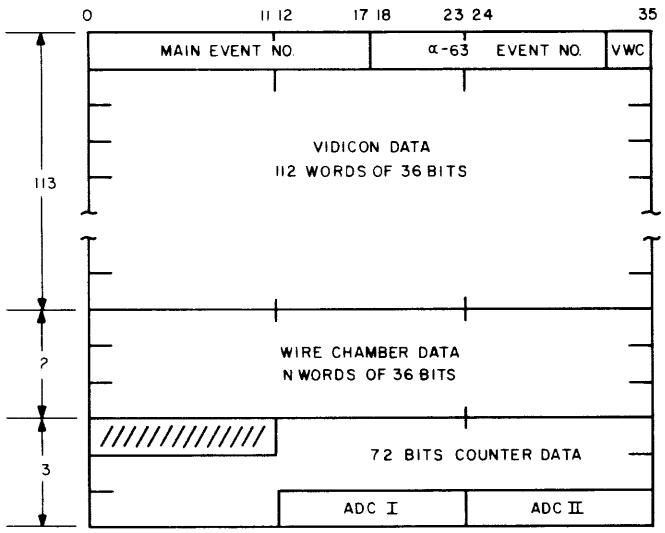


Figure 8 Data Format of One Event (VWC)

The utility bin provides control logic for the paper tape reader and punch, an incremental magnetic tape unit, and a readin facility for an unlimited number of 10-mc scalers. Furthermore, it has five interrupt flag inputs and 15 program-controlled pulse outputs for general use.

To keep a complicated system as this one running, it is necessary to organize effective operations and maintenance procedures. During a run, at least four programs are available (see Figure 9). The program of highest sophistication is called "Monitor," and occupies more than three-quarters of the available memory space, being 4096 words of twelve bits. Monitor therefore does not produce much data on tape, but is mainly used to investigate experimental parameters during the set-up time or after a change setup or for a limited amount of time during the normal run. It affords the experimenter to scrutinize the incoming events by statistical means.

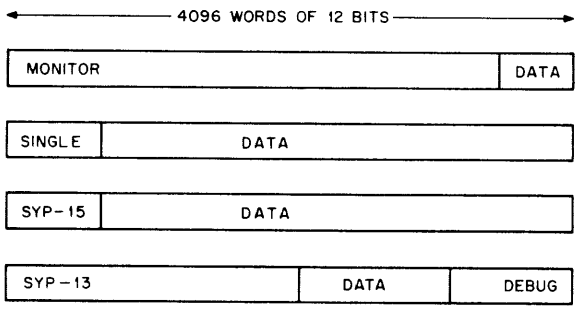


Figure 9 Memory Map

The next program is of lower sophistication and is called "Single." This program occupies only one-quarter of the memory space so that considerable space is available for data accumulation before the tape record is written. It only allows for two modes of display.

Syp-13 and Syp-15 are maintenance-oriented programs. Syp-15 affords the operator to produce a useful amount of data and to make performance tests of the system simultaneously. Syp-13 is more elaborate; it has facilities to test the interfaces, the tape unit, and particular circuits. It also contains a debugging package and a set of 16 input-output subroutines to facilitate the construction of a new test program in a matter of minutes.

There are now several PDP-5's being used at the Lawrence Radiation Laboratory in Berkeley, three of which are assigned to the counting physics experiments. Figure 10 shows a diagram of the control area and the "cave" of an experiment now running in the Bevatron in Berkeley. All the interfaces, with the exception of the oscilloscope display control, were constructed by facilities in the Lawrence Radiation Laboratory.

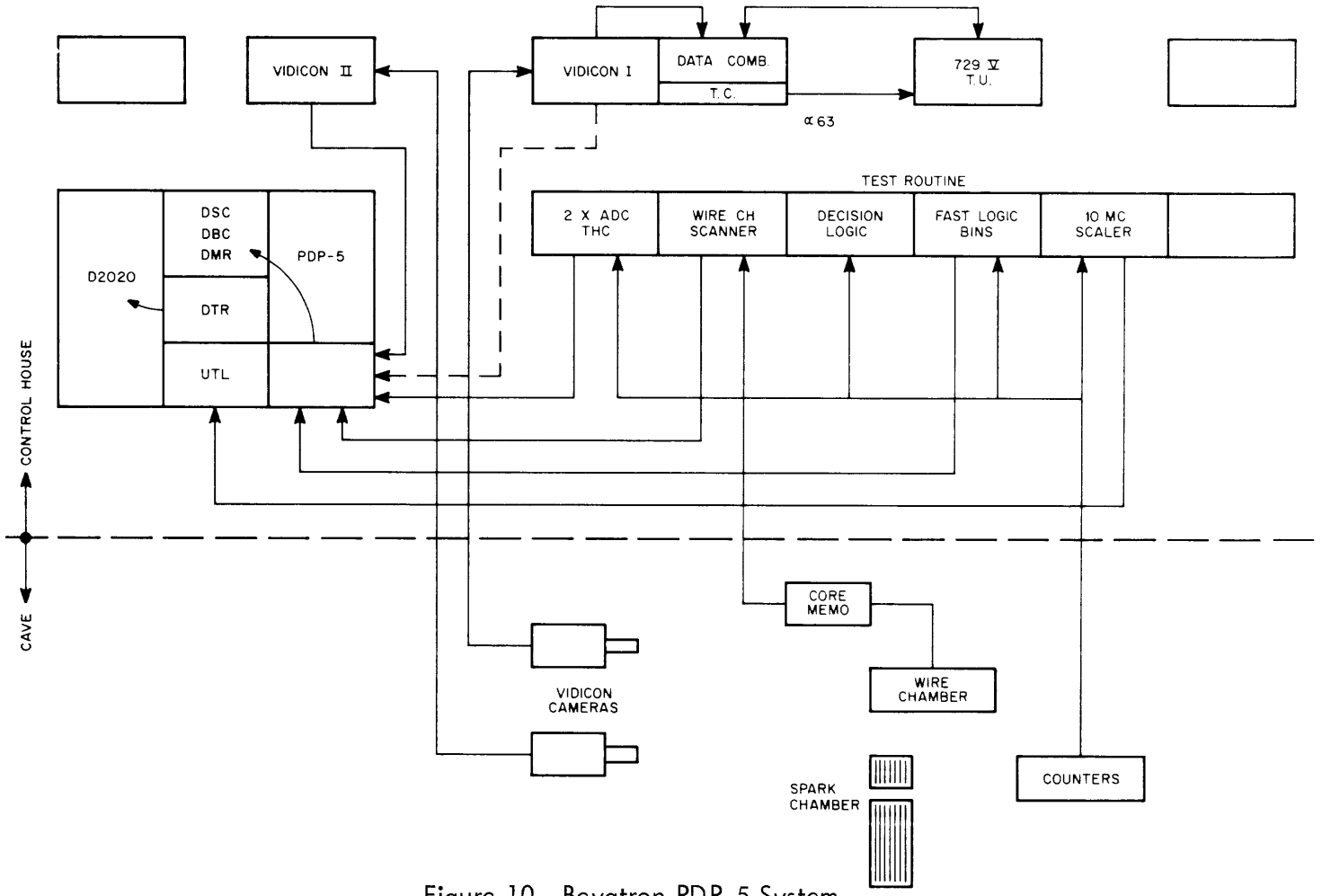


Figure 10 Bevatron PDP-5 System

COMPUTER SYSTEMS FOR RECORDING, RETRIEVAL, AND PUBLICATION OF INFORMATION

by

Richard J. McQuillin and
Joseph T. Lundy

Inforonics, Incorporated
Maynard, Massachusetts

Abstract A comparative description of the various types of typesetting machinery; hot metal, photo composition, and cold-type composition is explained. The areas of applicability of each of these methods can overlap, but publishing applications are described where one type of machine is more suitable than others.

Justification and hyphenation are fundamental problems of automated typesetting. The standards of measurement used by the printing industry must presently be incorporated into the automatic system. In addition, the problems of complex typography, such as mathematical formulae, and the problems of page makeup are discussed.

Typesetting machines are viewed from the aspect of a computer-based installation. This paper discusses solutions to various problems arising in preparation of text in computer readable form, in methods of editing and correcting this data, and the flow of information into typeset material and/or retrievable machine data files.

INTRODUCTION

The range of printed information runs from weekly newsletters dealing with specialized information areas to scientific journals, from newspapers to books, and from parts lists to dictionaries. Every type of publication has its own unique problems. Scientific journals need to cope with the ever-expanding body of knowledge--getting the right information into the hands of those who need it at the right time. Newspapers have the constant requirement of meeting time deadlines. Catalogs, parts lists, and dictionaries face the problem of keeping their information timely. The computer industry has now entered the field of publication. The basic problem that computer systems must address themselves to is the preparation of information to be printed in a computer interpretable form which can simultaneously satisfy the needs of scientific information retrieval, changing catalogs and lists, and speed of publication while maintaining traditional typographic standards.

The purpose of this paper is twofold: to acquaint the reader with the various typesetting equipment and to introduce some of the problems and methods of solution of computer-based typesetting

and publishing. The first section will be a tutorial on typesetting equipment, and following this will be a development of computer typesetting from elementary problems to advanced, and unsolved, problems.

Any composition process in which the margins are even on both sides will be considered. That is, the columns are justified. This could mean typesetting on a variable space typewriter as well as on a Linotype machine.

TYPESETTING TUTORIAL

This section is presented to give the computer-oriented reader some familiarity with the problems that have to be solved between the computer-typesetter interface. Representative examples of three different types of typesetting will be given, such as: cold type (or typewriter), hot metal, and phototype. Typewriter composition is not too pertinent to computer typesetting, but it is described for the sake of completeness.

Typewriter Composition

The essential feature of justifying typewriters is that they must have proportional spacing. That is, the characters must be of variable width.

Typewriter composition of this type tends to be very tedious, but it is advantageous because of the relatively low investment on typesetting equipment. This kind of typesetting is typical in small (weekly) newspapers and company in-house publications. Typical composition times are one page per hour. Some typical composing typewriters will be described below.

The IBM Executive

This has a single set of proportional spacing characters. The operator types the copy once and calculates the deficit in units for the line. The operator then divides this value by the number of spaces between words. The line is then retyped, and this value of increments is added into each interword space, with the result of an even right margin.

The Friden Justowriter

The Justowriter uses a 2-unit system to achieve justification--a recorder unit and a reproducer unit. Both units have the physical appearance of a Flexowriter, but they are modified to achieve justification through a mechanical counting system. During tape preparation on the recorder, the operator types until the justification light is turned on indicating the end-of-line zone. The operator presses the JUSTIFYING CARRIAGE RETURN key, which causes an end-of-line code and numerical value to be punched into the tape.

The reproducer unit has two readers. One reader recognizes only the end-of-line code and the numerical value punched by the recorder unit. The other reader copies the textual material. Both readers are mounted on the machine, the end-of-line code reader behind the copier reader. The tape from the recording unit is passed through both readers. The end-of-line code comes after the line. When the process is started, only the end-of-line code reader is activated. It reads until it meets an end-of-line code. The numerical value it reads at this point is translated in the machine in such a way that every time a space is encountered by the copy reader, extra space is added to make up the justification. After this value is set up, the copy reader begins to read and type out the line. It reads until it comes to an end-of-line code. Here it stops and the other reader starts again.

The Varityper

The Varityper is a single-unit justifying typewriter with the added feature of interchangeable type fonts. Varityper differs from other cold-type composition machines in its method of typing. The characters are on a type plate which rotates into position when a key is pressed. A hammer then forces the paper against the character. Character sets are changed by inserting new type plates. Justification is accomplished in a similar way to the Justowriter. This time the line is typed and the end-of-line key is depressed. The Varityper then calculates the required inter-word spacing to achieve justification. The line is then rekeyed, along side the first, and the justification is achieved.

Hot-Metal Casting

Most of the commercial composition in the world is done by hot-metal machines. Typical of these machines are the Linotype and the Monotype. These machines are run either from manual

keyboard or paper tape. The Monotype caster is always run from paper tape that can be produced from either manual keyboard or by computer. Both these machines can be run by computer-generated paper tape; however, the Linotype is used far more commonly for computer typesetting.

In the operation of the hot-metal machines, the text is composed with characters cast in molds. These molds are ordered by the machine to produce lines of type, with words separated by spaces. Now these lines may be set on a character-by-character basis or the separate characters, or matrices, may themselves be assembled as a complete line and then cast as a whole line, giving a thin, solid block of metal (known as a "slug"). This latter method illustrated the Linotype method, and the former, where the line is composed on a character-by-character basis, illustrates the Monotype method. In both cases the output of the casting machine are lines in which the characters are raised. This illustrates the letterpress style of printing. In the Monotype method the raised characters are produced individually by casting hot metal (lead) into individual characters of a matrix. In the Linotype method, the individual molds (mats) are assembled into a line. When the line is full, the hot metal is forced into the molds, casting an entire line.

The Linotype Machine

The Linotype composes with small brass units called matrices, or mats, with the characters indented in the edges. These matrices are assembled into justified lines. When the line is justified, it is cast into a solid bar or line of type.

The Linotype has four major divisions:

1. The magazine which contains the matrices. This is a thin, flat box near the top of the machine. The matrices are stored in vertical rows in the magazine. When a particular key is struck, a hatch opens at the bottom of a particular channel, and a matrix is dropped.

2. The keyboard and its related parts. The mats are dropped under control of the keyboard. Linotype machines can also work via paper tape readers. In this discussion the tape operation is synonymous with the keyboard operation.

3. The casting mechanism. The line of matrices is presented to the casting mechanism. Here molten type-metal is forced into the indented characters of the matrices, the line is cooled, trimmed, and delivered to the galley as output.

4. The distributing mechanism. When the line of matrices is finished casting, it is lifted and automatically carried to the top of the magazine. Each matrix has a coded notch, so that it always falls back into the proper magazine.

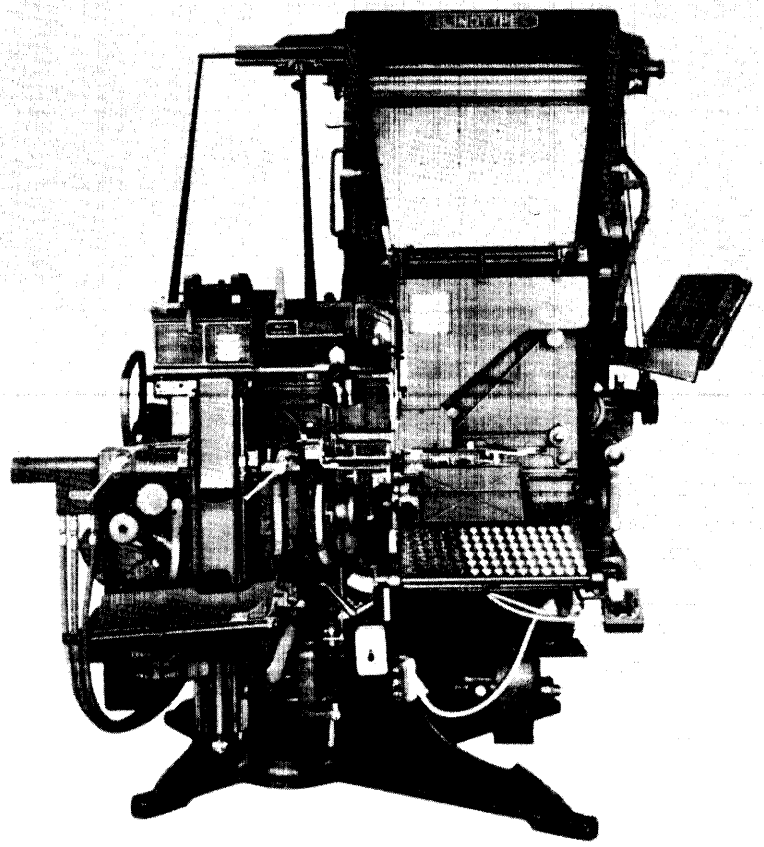


Figure 1 The Linotype Machine

Figure 1 shows a Linotype machine. An essential feature of the Linotype machine is that the line widths are of a fixed dimension. In order to achieve justification, the spaces between the words are variable. This is achieved by an ingenious method of using spacebands. The space-band is a wedge-shaped unit whose width varies with the distance along it. When the line is to

be cast, we say that the line is "elevated." What happens here is that the spacebands are forced upward. This motion expands the line, pushing the end matrices against the jaws of the machine. When this happens, the line is held rigidly in place, ready for casting.

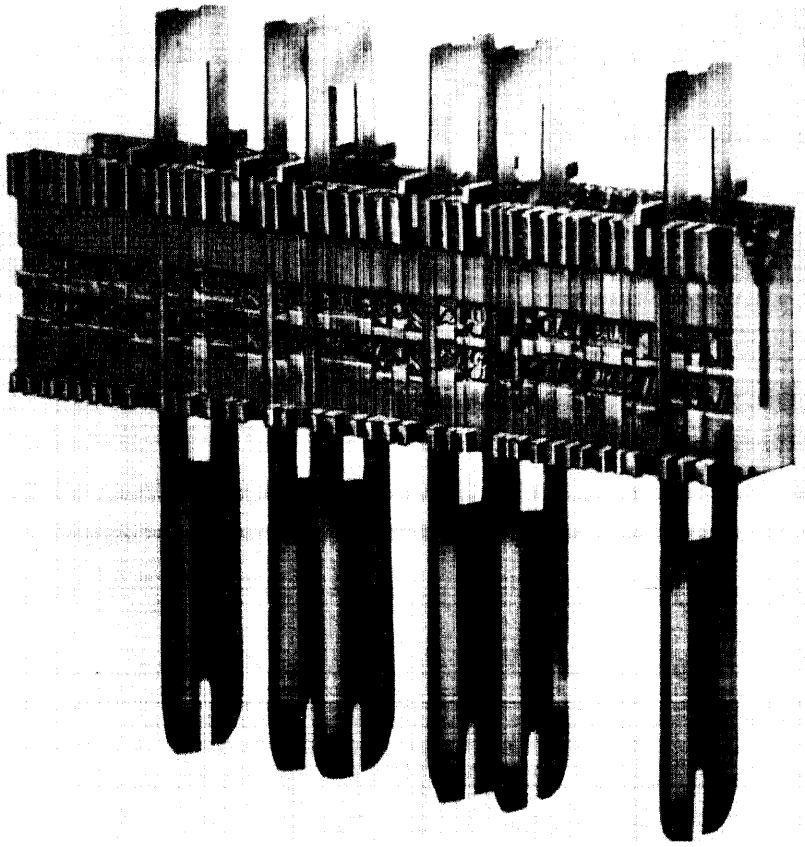


Figure 2 Line of Matrices with Spacebands

Figure 2 is a picture of a line of matrices with spacebands inserted between words. There are two rows of characters on the matrices. The matrices are notched in such a way that either the top character or the bottom character may be positioned in front of the casting window. The notches rest on one of two bars, called rails, on the caster. One row of characters is referred to as being on lower rail and the other row as being on upper rail. In practice, the upper rail characters are usually the bold face or italic equivalents of the standard lower rail characters.

Generally Linotype machines are run from either the keyboard or from paper tape. Paper tape is widely used because it may be prepared off line giving maximum efficiency to the machine itself. The Linotype casts about 10 cps. Two of the most common off-line preparation devices

are the teletypewriter and the Justowriter. Both of these devices prepare paper tape like a Flexowriter, but they are capable of keeping track of the justification range mechanically. They are set up for a particular line width and spaceband width. They accumulate total width as the line is typed and give some indication when the line is full. The teletypewriter is just a tape perforator and gives no visual indication of the copy. The Justowriter, on the other hand, produces copy as it is typed, but the lines do not appear justified because of the variability of the spacebands. While there is no universal standard for the widths of the matrices, there is a trend toward the "unit" system in which each character is considered a relative width, relative to the size of the "M" character. In this system all characters have the same relative width regardless of the type size. The trend to standardization has been brought on to a great extent by the wire services, in which precast lines are sent over the wires. At the wire terminal is a tape punch which processes a paper tape suitable to be read by the Linotype machine directly. In the unit system the relative widths vary from 6 for "i" to 18 for the "M." Figure 3 is a picture of a galley of lead produced as output from a Linotype machine.

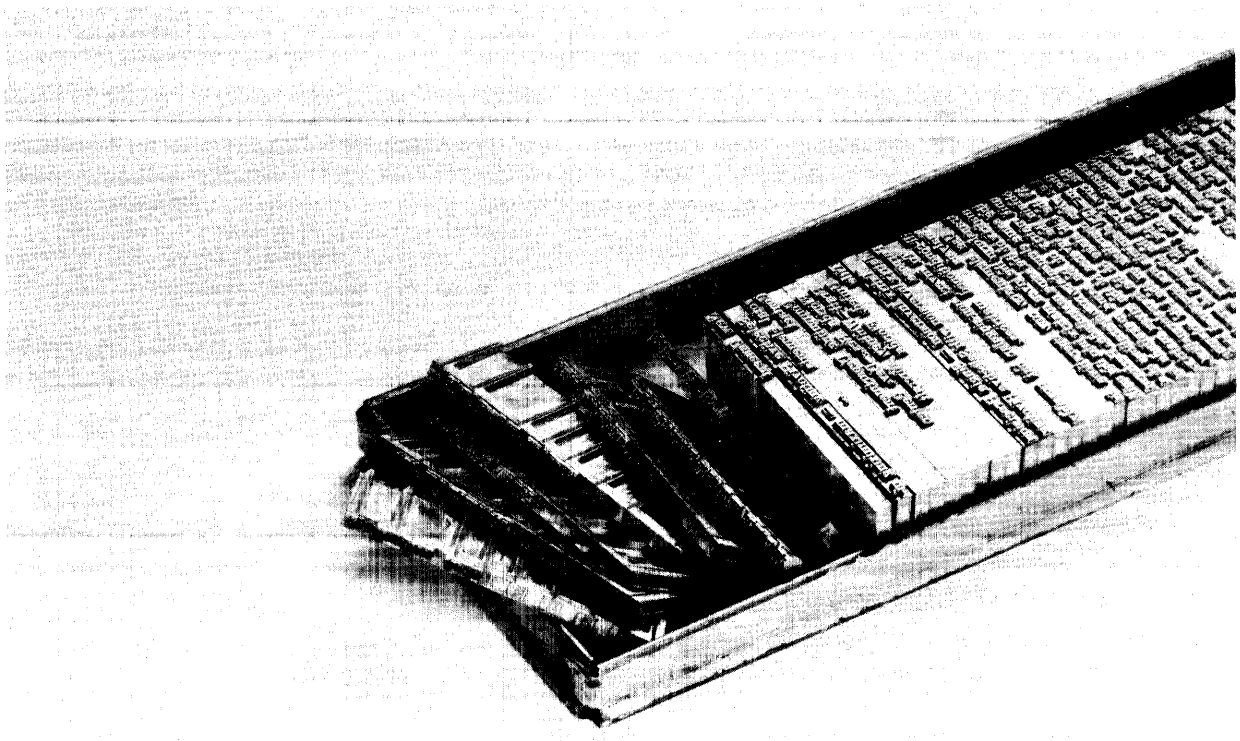
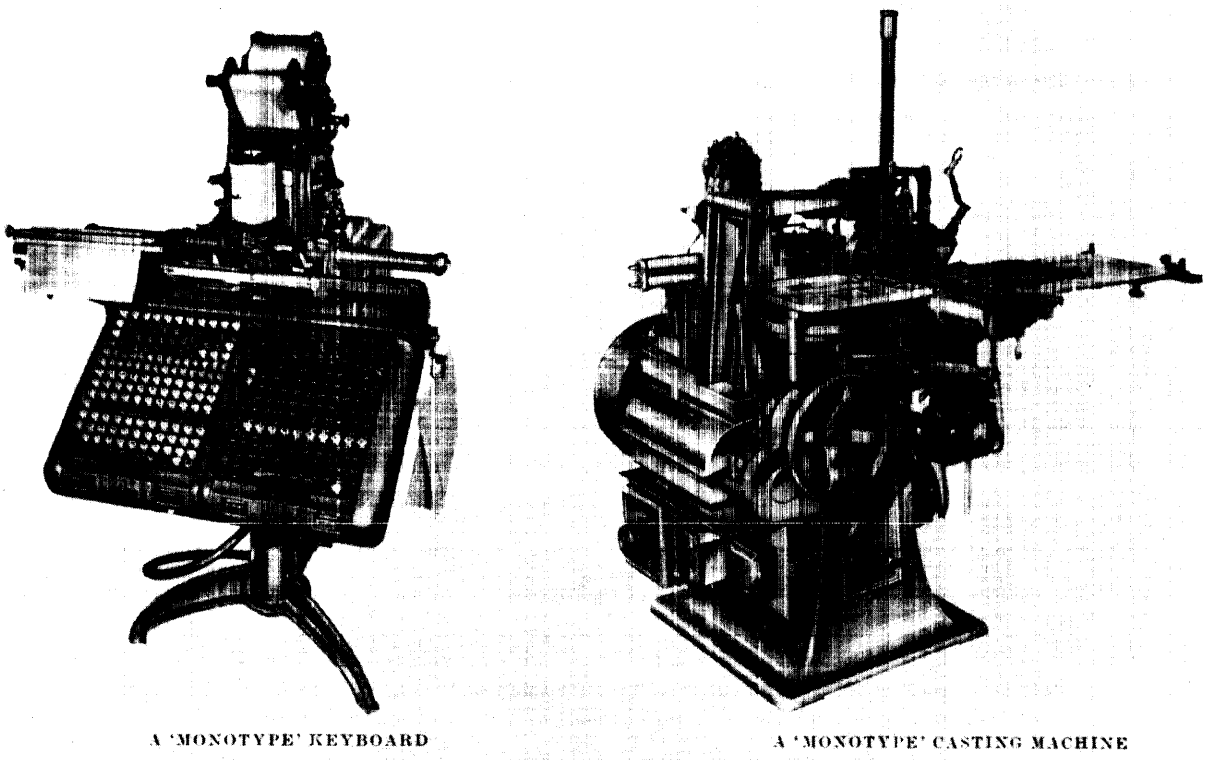


Figure 3 A Galley of Lead Type from a Linotype

Linotype machines are generally used for straight text matter. They are common in newspaper typesetting and book typesetting, where the text matter is well ordered, such as in novels. Its operation is very linear, and it does not lend itself to complicated typography, such as mathematical equations. In addition, its character set is very limited, and if it is desired to introduce a different symbol, say a Greek letter, the machine must be stopped while a new magazine holder is loaded. This is quite tedious in practice.

The Monotype Machine

For more complicated typesetting, the Linotype becomes inadequate, and the Monotype may be used. The Montype system consists of two parts: the Monotype keyboard and the Monotype castor. Figure 4 is a picture of these units.



A 'MONOTYPE' KEYBOARD

A 'MONOTYPE' CASTING MACHINE

Figure 4 The Monotype System

The operation consists of preparation of a paper tape on the keyboard unit (off line), and the running of this tape through the castor, setting the type. Let us take each component in turn.

The Monotype Keyboard Unit - The Monotype keyboard has 256 keys. Besides the keyboard itself, there is a mechanism to keep track of the justification of the line. The operator chooses the width of the line to be set or the "measure." He adjusts the scale above the keyboard to the number of "ems" in the line. As he types, a pointer moves across the scale, being incremented by the number of units of the character. The widths of the characters are determined by their position in the matrix case. Figure 5 gives a schematic of a typical matrix case. All the characters of a given row are of the same relative width, and characters of different widths are placed as shown in the left-hand column of numbers. In the casting operation, this matrix moves over the mold under paper tape control. The width of the mold is varied mechanically as a function of y-position of the matrix. As each character is set, hot metal is forced into the mold up against the matrix case. The character in the matrix case determines the character, and the mold determines the width.

	NI	NL	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
Units 3		)	[	]		/	l	t	,	□	'	ı	t	.		,	·	1
5	z	c	r	i	z	c	r	s	i	□	l	t	,	.	i	l	'	2
6	q	q	b	θ	τ	γ	τ	(	)	□	[	]	|	/	j	f	!	3
6	1	2	3	4	5	6	7	8	9	0	¼	½	¾	’	‘	}	{	4
6	1	2	3	4	5	6	7	8	9	0	-	:	;	j	f	i	§	5
7	a	k	x	y	a	k	x	y	z	s	r	c	e	r	t	s	°	6
8	n	p	n	p	α	α	q	v	b	g	o	θ	I	z	c	e	λ	7
9	+	-	1	2	3	4	5	k	y	d	h	a	g	a	o	ω	α	8
9	+	-	6	7	8	9	0	x	-	□	¼	½	¾	†	√	.	β	9
10	fl	fi	u	n	S	x	q	k	v	y	b	p	h	d	u	n	η	10
10	√	P	π	R	A	m	μ	ψ	φ	?	p	m	fl	fi	J	I	χ	11
12	Γ	Θ	Ω	Φ	ff	w	J	S	Δ	ff	Z	C	P	L	F	T	ω	12
13	V	Z	Q	G	O	L	C	Q	V	B	G	O	E	A	w	Y	◇	13
14	P	R	B	F	E	T	D	A	ffl	ffi	m	Y	U	R	N	D	∑	14
15	→	←	∞	X	∝	K	U	N	H	ffl	ffi	X	K	M	H	m	◇	15
18	±	+	−	×	=	⩽	⩾	W	M	~	>	<	≡	—	≠	W	□	16
	NI	NL	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	16th row only
	K		KM										KN	KM		MN	KMN	

Figure 5 A Typical Monotype Matrix Case

As characters are typed on the Monotype keyboard, codes are punched in the paper output. This paper is 4-5/16 in. wide, and the holes are punched at 1/8 in. intervals. There are

31 perforations on the tape; 28 controls the movement of the matrix case, 2 control the movement of the justification wedges, and 1 brings into use the space transfer wedge when justifying spaces are cast. The codes may be read visually from the tape and translated into characters by character position in the matrix case. The Monotype output tape is divided in half, each specifying an x-y coordinate in the matrix case. There are 14 levels in each dimension. Additional levels are obtained by punching combinations of holes in each dimension.

In order to achieve justification, a mechanical calculator is used. As the line is typed, relative widths keep accumulating. When the total accumulated widths become four "em" spaces less than the total line width, a bell rings to warn the operator. At this point he may either continue with a few more characters or do the justification operation immediately. The justification is calculated mechanically by the wheel or cyclinder at the top of the machine. Each time the space bar is struck during the line preparation, a pointer on the revolving drum is advanced.

What the machine does is to calculate mechanically the amount that must be added to each space between words in order to justify the line. Mounted on the wheel is a table of numbers, with a tiny pair of numbers in each block on the drum. The machine has, in effect, divided the total remaining space on the line by the number of spaces on the line. The two numbers pointed to on the drum tell the operator that he must now depress two keys to set up the justification properly. These two rows of keys control the width of the spaces between the words. One row is coarse adjustment, being in units of .0075 in., and the second row is in units of .0005 in. Thus depressing a 6 in the top row and a 5 in the bottom row increases the minimum 4-unit space between the words by $6 \times .0075 + 5 \times .0005$ in. This will be just right to justify the line. Incidentally, in the casting operation the paper tape is read backwards so the spaces are set up before any characters are cast.

In order to achieve more characters than the 256 on the keyboard, it is common practice to use "overlays" on the keyboard. In order to do this, the operator punches a key which will stop the castor momentarily. He then writes a note on the paper tape strip, telling the castor operator to insert a different matrix case in the castor at this point. The typing is then resumed but interpreted in a different alphabet, say Greek. In a given Monotype operation, all point size of the type is the same. The widths vary, but not the height of the slug. New molds are required for new type sizes. It is noted that small characters, say subscripts, are set in slugs of equal vertical dimension as larger characters.

The Monotype Castor - After the paper tape is prepared, the casting operation begins. The castor reads the perforations in the paper tape by blowing compressed air through the holes. The air causes pins to be raised in a pin block matrix. The character matrix slides over this block until it is restrained by the raised pins. Thus the matrix is positioned.

The molds in which the new character will be cast are about 4 in. square and 2 in. deep. Two fixed blocks determine body size and form two walls of the cavity in which the type is to be cast. A moving cross-block forms the third wall, and an adjustable mold blade forms the fourth wall and determines the width of the type to be cast. On the underside of the mold there is an orifice through which the molten metal is pumped. At the moment of casting the open top is covered and closed by one of the characters in the character matrix. After the character is cast, it is assembled as part of a line. The caster is capable of producing about 160 characters each minute.

The fact that each character is cast individually makes the Monotype a very versatile typesetting machine. It would seem that the machine would not be so versatile in handling complex typography because it only sets one size type. This is overcome because its output can be edited by hand. Since all the characters are individual, any one may be pulled out and replaced by something else. In the case of mathematical composition where a large summation sign was used:

$$\Sigma \text{ An Cos nx}$$

the Monotype operator would leave room for the sigma by inserting fixed spaces. The castor would cast these as fixed space and later the compositor would remove the fixed spaces and insert a large sigma from his matrix box. Complicated expressions that cannot be set too well from the keyboard may be similarly "fixed up" by hand before the printing takes place.

The Monotype system is the method most used to set scientific journals and texts. It is widely used in situations of complex typography.

Photocomposition

Photocomposition frees the typesetter from some of the restraints imposed by hot-metal composition. Characters are produced by a ray of light focused anywhere on a sheet of film and are not physically constrained by a metal wall. With this freedom the photocomposition machine

can mix sizes in a line; juxtapose lines of different sizes; overlap images to form composite characters, such as accented letters; and make any number of page layout designs either from the keyboard or from tape. In many recurring print jobs, such as dictionaries, the publisher prints periodic, updated editions. To avoid the cost of total recomposition, the made-up pages are stored. A lead page weighs quite a few pounds and a full book of lead pages occupy considerable space. In photocomposition, the made-up film pages are kept instead and these can be stored in file cabinets.

Figure 6 The Model 200 Photon Machine

Figure 6 is a picture of a Photon machine. This is the Model 200, which is operated by keyboard as well as paper tape. Behind the operator is the optical unit, which sets the type. The rest of the machine consists mainly of relay logic to provide justification in the manual operation. Figure 7 shows the optical system of the Photon.

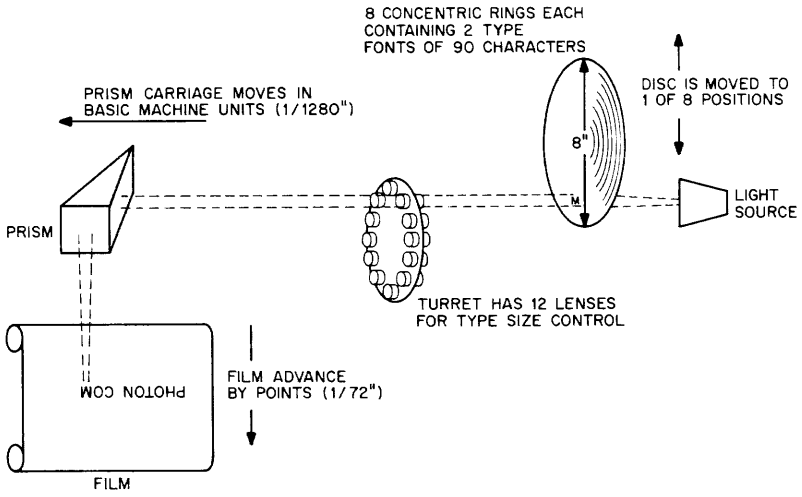


Figure 7 The Optical System of the Photon

The Photon generates its characters from a rapidly rotating glass disc. The characters are depositied photographically on the disc. Each disc has 1440 characters arranged in 8 concentric circles, 180 characters per circle. Generally each font is placed on one circle, or level. The rotating disc rests on a lever arm so that it can be moved in a vertical direction effectively to change disc levels. Behind the disc is a fixed-light source that flashes when the desired character is in position. The light passes through the disc and through a lens. There are twelve lenses on a turret, and a new lens may be rotated into position. This controls the size of the type. The light then passes through a prism, which bends it at right angles and focuses it on a piece of film or light-sensitive paper in the film magazine. The prism moves back and forth along the page, in a similar way as the action of an IBM Selectric typewriter. The Photon is capable of setting about 10 cps. Both the font shift and lens shift mechanism are accessible both from the keyboard and by computer produced tape. In addition to character flexibility for a disc, the discs themselves are interchangeable; a short manual operation.

When the Photon is operated automatically by a computer-produced tape, it interprets the tape codes as function codes, character codes, and width codes. A function code causes a change in disc level; a change in lens to effect a change in character size; film advance in points (1/72 in.) to effect line spacing; and an end-of-line code which returns the prism carriage to the left margin.

The character codes and the width codes are grouped together. That is, for each character code generated there is a corresponding width code. The first level selects a character on the disc by position. The second level gives the width of the character in Photon Basic Machine Units (about 1/1280 in.). The BMUS are translated into the horizontal stepping of the prism carriage across the film.

Figure 8 The Output from a Photon

ELEMENTARY COMPUTER TYPESETTING

Justification, in metal typesetting, is the process of fitting the movable type securely in a container or galley so that it can be moved about without displacing pieces of type. By extension, this definition is also applied to phototypesetting, although no physical boundaries constrain the typeset line. The traditional method of holding the letters in the galley is uniform expansion (or contraction) of all interword spaces within acceptable limits. In Linotype this is achieved by pushing up the wedge-shaped spacebands before the line is cast. Other typesetters including

Monotype and Photon calculate the value of each space. In any case, the result is an even right margin. The limits of expansion are imposed by aesthetics and by typographic traditions, many of which have a solid basis in legibility of type. Look at the pages of a good book from a distance and notice that they appear uniformly gray, without white spots and clusters of white spots. To a reader, the words flow along smoothly, drawing the eye without interruptions in the form of a sudden white void. Lines which are too tight--the spaces too narrow--blur the distinction between words.

A companion problem to this is hyphenation. When a line cannot be justified without defying these limits imposed by standards of legibility, the line must be spaced out on a new basis, including a now divided word.

Computer Techniques In Justification-Hyphenation

A computer program that is to set justified lines must do a table lookup on each input character as it is read in. Furthermore the real unit of information is the word, not the character. Therefore it should input a word at a time and add its width to the accumulated width of the line. Since the spaces between the words have to be variable to achieve this justification, whether it is physically variable as a spaceband in a Linotype, or a calculable variable as in the Photon or Monotype. The computer program must keep track of the number of these variable spaces on a line as well as the limits of this variability.

Figure 9 is a schematic of this operation. The words are added to the line and the total accumulated width is computed using minimum expansion between the words. The line is also computed using maximum expansion. Now there are several things that may happen when the line gets filled up:

1. The maximum expansion calculation does not reach the column width.

2. The range between maximum and minimum expansion may fall within the column width.

3. The minimum expansion calculation gives a value which exceeds the column width.

The three corresponding courses of action are:

1. Get another word.

2. Either try to fit another word on the line or set the line immediately.

3. Investigate the possibility of hyphenating the last word.

Just how each path is taken is determined by the particular typesetting job. Generally, it is desirable to try to get as many words on the line as possible. Eventually this leads to many hyphenated words, which is also undesirable. On the other hand, text which is set loosely will tend to have "rivers of white" down the page. In this case, text which is held back from the eye will exhibit long, connected white space areas running down the page. This effect is unpleasing to the eye and is to be avoided. The amount of hyphenated words appearing in the text is directly related to the number of spaces (variability) on the line, which is related to the column width. Newspapers, for example, necessarily have many hyphenated words, while books with wide columns seldom use hyphenated words.

When the computer decides to try hyphenation there are several possible paths to take:

1. Hyphenation is not possible, so put the word on the next line.

2. The word can be hyphenated, but it still won't fit properly if the hyphen is put in one of the possible hyphenation spots.

3. The word may be hyphenated. If the word simply cannot be hyphenated, it must be put on the next line, and the current line must be expanded to achieve justification. This is done by adding fixed spaces in between the words without limitation until the line is justified.

There are several ways of hyphenating a word, and the method used is again dependent on the type of work. These methods may be divided into four categories: manual, dictionary, logic, and logic with dictionary. It would be very nice if the computer could hyphenate every word correctly by logical algorithm. However there are so many exceptions to rules in the English language that this is not possible. Various logical rule schemes will give correct hyphenation

in excess of 80% of the time, but none will give 100%. In some situations it is desirable to sacrifice some accuracy for speed, and 80% is good enough. Another way of handling the hyphenation problem is by dictionary lookup. This would theoretically give 100% accuracy if entire dictionaries were put in computer memories, but this would imply very large memories and could mean slow operations. A compromise between logic and dictionary is a system which would contain a small dictionary of exceptions that would be searched before the logic is applied. This method is particularly useful in specialized publications, such as scientific journals, where the same words keep coming up.

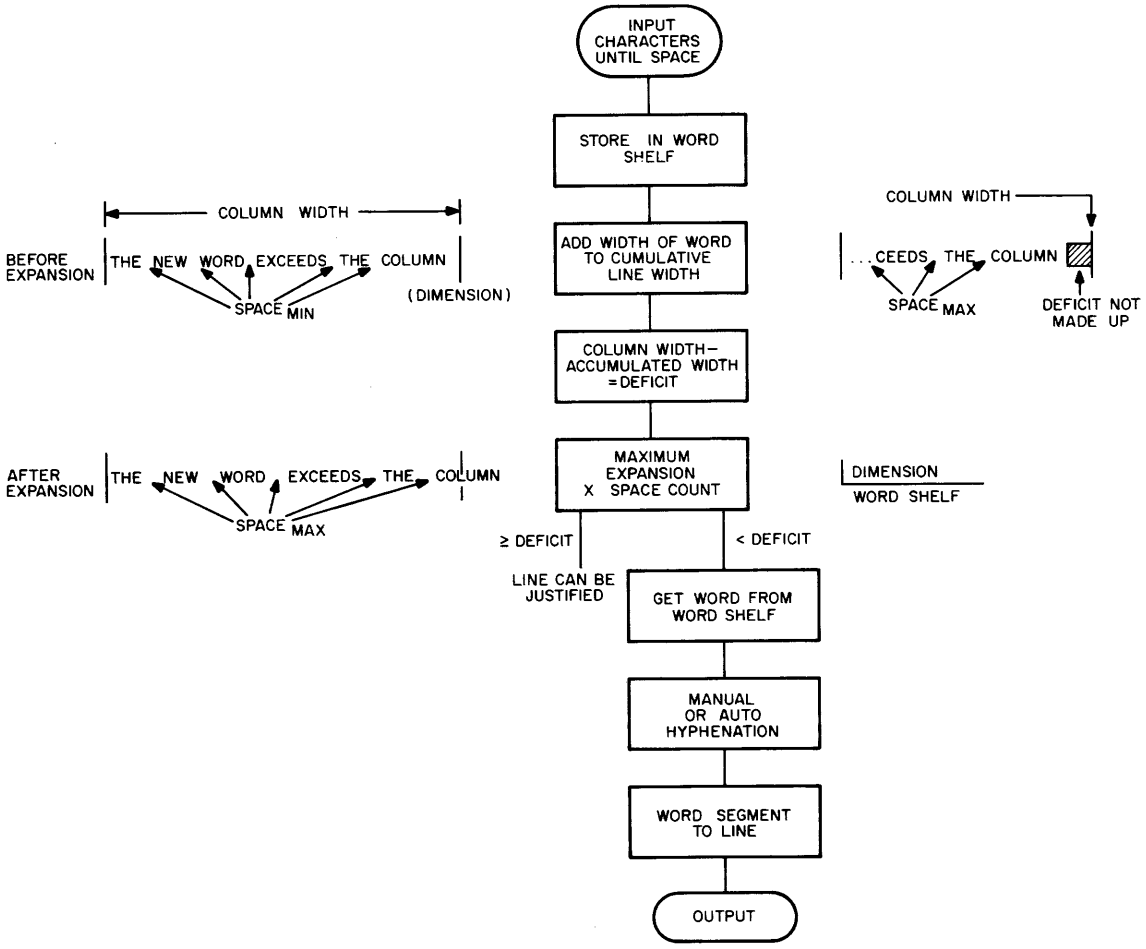


Figure 9 Justification-Hyphenation Schematic

At the opposite end of the spectrum from the fast, logical decisions is the manual hyphenation where the word to be hyphenated, and perhaps the surrounding words, are presented to the human operator by being typed out at the console typewriter or displayed on a scope. The

operator chooses the spot where the hyphen is to occur, and the typesetting proceeds. This method is slow, but if the columns are wide the human operator isn't called very often, and when he is called, the hyphenation accuracy will be 100%. There is also the advantage of the hyphenation subroutines taking far less computer memory than in the other methods.

Other Problems In Elementary Typesetting

Quadding

A common function of any typesetting program is to handle quadding. That is, the program must be able to flush a line left or right, or center it in a column. To do this, it must compute the difference between the total column width and the accumulated width up to that point and fill in this deficit with blank spaces to justify the line. In particular, the scheme for finishing a paragraph is to insert a flush (quad) left code after the last word in the paragraph. If the start of the new paragraph is to be indented, the operator begins by typing in some fixed spaces, usually an em, en combination.

Code Conversion

Typesetting systems generally need code conversion subroutines at both input and output. The input tapes are commonly prepared on Flexowriters of arbitrary codes--usually on what is available. These input tapes are then read by the computer and converted to some standard internal code. When the output is generated, it must be punched in such a way as to be compatible with a particular typesetting machine. Some Photon tapes, for example, are 8-level with two rows of holes representing a character. The first row represents the character itself, and the second row the width, or "escapement" of the character. The exception to this code conversion problem is perhaps the Linotype typesetting program. Here the input is prepared off line in a 6-level Teletype code. The output is in the same form. Teletype is used as the internal code as well.

ADVANCED TYPOGRAPHIC CONTROL

In the last section, the typesetting problems common to every typesetting system were discussed. Now some of the more advanced features of computer typesetting systems will be discussed.

These advanced features, when implemented, tend to reduce the total typesetting costs either by simplifying the initial keying, reduction of editing, ease of reprinting, or in some other area. The reduction of that all-important dollars per page figure is the most important goal of commercial computer typesetting.

Setting of Tables

Tables frequently occur in typesetting and it is desirable to implement some scheme to set them. When tables are hand set, the operator must do considerable calculation to make sure all the columns are aligned. The essence of a computer table setter is that the format of the table is first fed into the system. After the format is specified, the input tape need only contain a sequence of elements, and the computer will do the rest. The time saving here can be considerable.

One scheme for table formatting is to give the computer the widths of the elements of each column as well as the width of the white space between columns. The computer then reads in the next entry from the input tape and does a template matching on the format, supplying missing characters by fixed spaces. There are two common problems with table setting: missing entries and nonstandard entries. Both of these situations may be handled by a special control code to the system. A common one is the dollar sign with the rule that a dollar sign followed by a number is really a number, but a dollar sign followed by a letter is a control code. In the input tape there must always be an entry for every column, even if the entry is to be blank. Thus a code combination $b would tell the table setter to generate a "blank" entry from the template. Similarly a code of $v for "verbatim" would tell the table setter to set the entry as is, disregarding the template matching. Since the template is a width-matching scheme, other characters could be substituted for what normally would be there. As an example, the system would set "abc" properly as well as "123" in a table of numbers, provided the widths were the same.

Extensions to the Input Keyboard

Most typesetting jobs of book or journal quality require at least a full family of type; that is, a set of roman, bold, and italic characters of the same style, as well as numerous special symbols. Since the tape preapration keyboard has a maximum of only 90 characters, a method

must be devised to signal to the computer program that a shift, with its accompanying typesetter control commands, is required. For Linotype, a rail shift is required; for Photon, a disc level shift.

A black square, dollar sign, or some other distinctive character is chosen as departure signal. The character following this signal can describe the new type font. For example, when the typist is typing text in roman and encounters a series of italic characters, he strikes the sequence $i"italic word"$n--the latter returns the keyboard to a normal state.

Keyboard Arrangement

The tape punching typewriter contains a basic or primary key set whose symbols appear in the type copy. The problem is to transliterate easily into other key sets. A keyboard notebook is the operator's guide to the available character sets. The notebook consists of a keyboard layout for each font designator. The primary page in the keyboard notebook serves all of the alphanumeric fonts, i.e., roman, italic, bold, etc., in which the input keys have a 1:1 correspondence with the output characters. A transliteration from roman to bold italic would use the primary keyboard arrangement although two different fonts are involved. A transliteration to Greek would require the operator to flip over to the Greek keyboard page in the notebook. A basic keyboard configuration (standard typewriter) can be used for all alphabetic fonts--the various families of roman, bold, italic where the input key to output character is 1:1. For math and special symbols, a keyboard overlay of arbitrary design is used although it is convenient to group characters together by type, e.g., all math symbols on one overlay, all chemical symbols on another, all punctuation on a third, and so on.

Mathematical Displays

One of the most difficult of computer typesetting problems is how to handle the mathematical display. For discussion, assume that the typesetting machine in question is the Photon. The first problem is how to represent the mathematical display as a linear string of symbols capable of being generated on a Flexowriter and how to proofread the string intelligently before processing. The computer must also generate a linear string because the Photon operates in a linear way. There is no "rolling the carriage" to get subscripts, for example. The carriage

must move all the way across one line of text before moving to the next line. Thus, the Photon must set the symbols on the line, do a carriage return, and then space over to pick up the subscripts.

In addition to the linear typesetting problem, the size of the symbols must be context dependent. For example, symbols on the main line of the display are set the same size as the body of the text, while subscripts and superscripts must be automatically reduced in size. The size of the large integral sign must be dependent on the function within.

Since the input string is artificial and unproofreadable, one powerful tool is the cathode ray tube. Using this device, the operator may view some resemblance of the display although scope coordinates and printing coordinates are not exactly the same. He may also view symbols, say Greek letters, that do not appear on the input keyboard.

SOME ECONOMIC ADVANTAGES OF COMPUTER BASED SYSTEMS-- MULTIPLE USE OF DATA

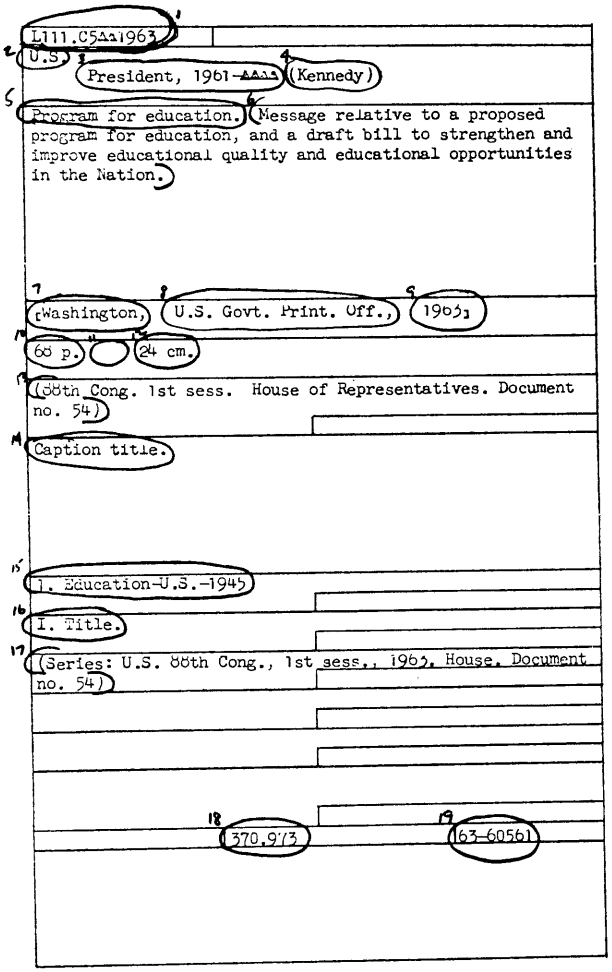
Creation Of Permanent File Records

An important feature of computer-based publishing is that once the original data is put into a machine file, it is available for many uses. It is indeed a tedious and costly matter to key the original manuscript into Flexowriter tape, but once this tape is available, and if the keying scheme is properly set up, this information is available to typesetting programs to do a number of things. In many situations, the information is set differently than the author, affiliation, footnotes, etc. If this information is itemized, it can also be searched in the preparation of indexes, etc.

Item Identification

In a typesetting system described here, the entire system is divided into three parts: The analyzer, the typographic control, and the output. The analyzer may recognize items in three ways: explicit tags, block boundaries, and delimiter sequences. Explicit tagging is easier to proofread and implement on the computer. On the other hand, it leads to more typing, and in high-volume typesetting, this could mean thousands of keystrokes. Recognition of items by

their position on a page is a very good method when the information can be typed into a form, such as library cards, schedules, and parts lists. An item may be terminated by two things in this type of recognition: by impinging on area boundaries in the form and by sequences of certain delimiting characters. The space, tab, and carriage return are the usual delimiting characters. For example, item 6 could begin at a certain coordinate in the input form. Item 8 could begin at the next block boundary. However, item 7 could exist (or not) with the item 6 block, and it would be recognized by three spaces. In Figure 10, there are certain items identified by delimiter sequences (items 8, 9, 11, 12, etc.) and certain items identified by their position on the page (items 1, 2, 5, 7, etc.). In this method, the typist need only type in a natural manner, and the analyzer will pick up the items properly. This method tends toward easier typing than the explicit tag method, but proofreading is more difficult because nonprinting delimiter sequences have to be read.



Card recording form showing the items which can be identified and located in the machine record.

Figure 10 The Items of a Library Card Input Form

The analyzer, having itemized all the information, is dismissed, and the typographic control is called to set in the specific format for the job.

Typographic Identification

When the limits of an item have been identified, it is a simple matter to relate its number, or label, to a stored table of control words containing typographic characteristics and page design commands. Specifically, these control words indicate the style of type to be used, point size, white space before and after a line or before and after an item, and intercharacter spacing. The page layout functions are flush left, center, flush right, and justify. The latter implies an indention of the first line of a text paragraph.

A single input record can be used to set type for the primary journal and various indexes referring to the primary journal. These indexes almost always refer to the number of the first page on which the cited article appears. The page number, however, is not known and currently, not entered, until the first pages to be printed have been made up. This is because figures, tables, and display equations are last to be inserted onto a page; and they are often reduced or expanded to fit a given space so that their precise dimensions, for page length counting, are not known until they are actually inserted. Determination of the size of figures and tables could be keyed in at time of primary data preparation.

With the initial page number included as part of the primary record, all secondary forms can be derived from the basic record and typeset without any intermediate processing.

One of the traditional typographic offenses which must be avoided by programming is the "river of white." This is usually caused by assigning too wide an expansion value to the variable spaces. Another prohibition is the "widow" which is a single word or word segment standing alone as the last line of a paragraph or the first line of a page. The best way to avoid rivers is to prevent lines with near maximum expansion from occuring successively. Widows can be avoided by testing the list of conditions causing a widow, i.e., last lines, not an isolated line, a single word (a word segment).

Secondary Uses of Primary Information

Library Cards

Given the identified items from the analyzer, one is now in a position to write programs to process this information. One such example is the library card. Figure 11 is a schematic of the operations that might be required to process the set of data. Figure 10 shows how the input information was prepared. This information is then fed through the analyzer to get it itemized. Then several typographic control programs were written to process this information in six different ways: Card catalog entries, author catalog entries, subject book catalog entries, Flexowriter cards, and line printer created cards. These forms are illustrated in Figures 12 and 13. In addition, the itemized information can be put on magnetic tape, suitable to be called out at some later time.

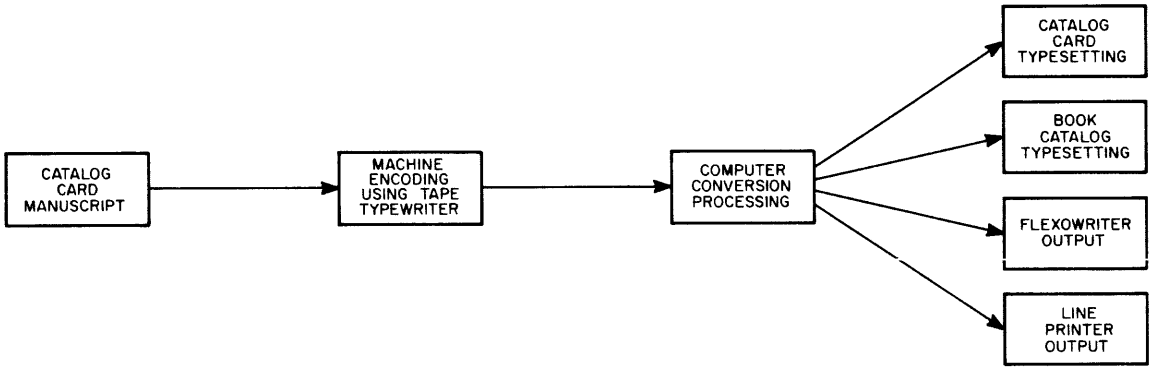


Figure 11 Flow Diagram of Catalog Data Recording

Flow diagram of catalog data recording experiment. Catalog card manuscripts are machine encoded by typing on tape punching typewriter. The machine record is converted by the computer to a variety of forms to suit different end uses.

Scientific Journals

Primary identification of text segments by content allows the basic document record to be used in a number of ways. Figures 14 and 15 show the input text of a scientific journal and its resulting typeset form. Here we can see that the various items are set differently depending on what they are.

U.S. *President*, 1961— *(Kennedy)*
Program for education. Message relative to a proposed program for education, and a draft bill to strengthen and improve educational quality and educational opportunities in the Nation. [Washington, U.S. Govt. Print. Off., 1963]

68 p. 24 cm. (88th Cong. 1st sess. House of Representatives. Document no. 54)

Caption title.

1. Education—U.S.—1945 I. Title. (Series: U.S. 88th Cong., 1st sess., 1963. House. Document no. 54)

L111.C5 1963 370.973 63—60561

Library of Congress

Copy of Library of Congress style catalog card typeset automatically from the input perforated tape record. The tape produced from typing the recording form was processed by a digital computer, producing a second tape which controlled the phototypesetting machine. The output of the phototypesetting machine is the card shown.

U.S. *President*, 1961— *(Kennedy)* Program for education. Message relative to a proposed program for education, and a draft bill to strengthen and improve educational quality and educational opportunities in the Nation. [Washington, U.S. Govt. Print. Off., 1963]

68 p. 24 cm. (88th Cong. 1st sess. House of Representatives. Document no. 54) Caption title. 1. Education—U.S.—1945 I. Title. (Series: U.S. 88th Cong., 1st sess., 1963. House. Document no. 54)

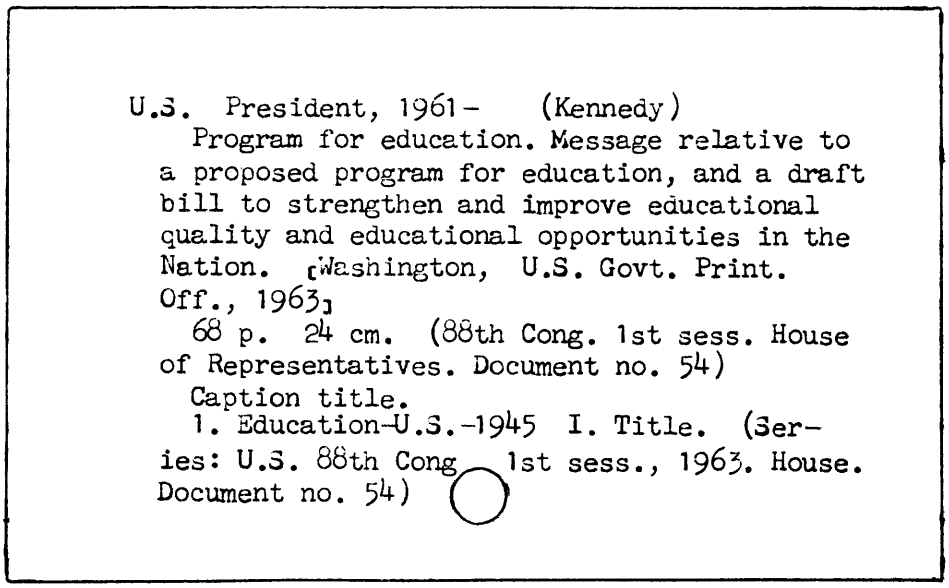
L111.C5 1963 370.973 63—60561

A copy of an author catalog entry. The same input tape was reprocessed to produce a book form entry using smaller type and a different format with the objective of saving space.

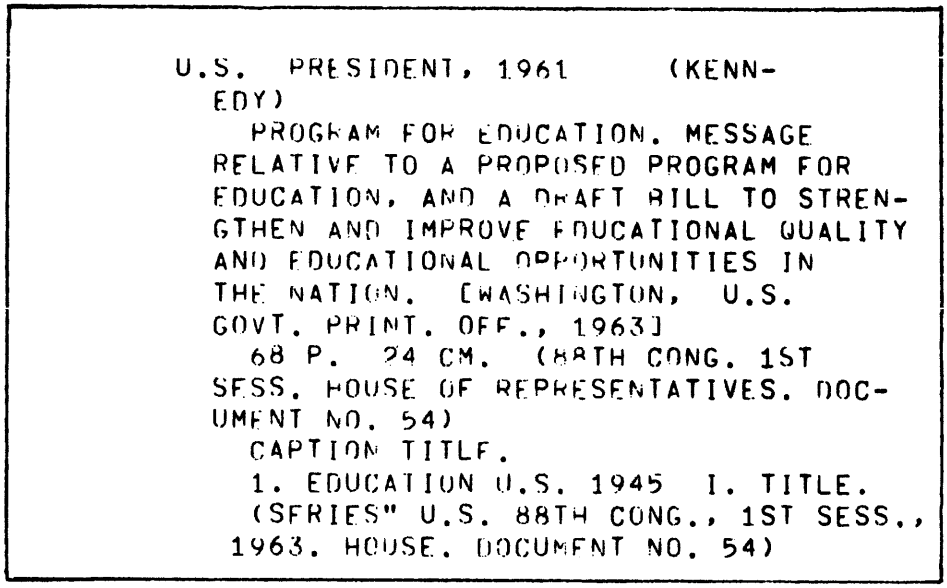
U.S. *President*, 1961— *(Kennedy)* Program for education. U.S. Govt. Print. Off., 1963] 68 p.

A copy of a subject catalog entry. The input tape is processed by the computer so that only selected items are reproduced on the phototypesetter.

Figure 12 Automatically Typeset Library Cards



A catalog card copy automatically produced on a tape typewriter from the same input record processed in a different way by the computer. For this type of output the computer does not need to produce the complex typographic control needed by the phototypesetter.



A catalog card produced on a teletypewriter from the original input data. In this type of output the computer screens out all case shift information.

Figure 13 Library Cards Produced on Tape Typewriters from Input Data

THE JOURNAL OF THE ACOUSTICAL SOCIETY OF AMERICA

VOLUME 34, NUMBER 1

JANUARY, 1962

On the Width of Critical Bands

4.7 Masking (See also 5.5)

John A. Swets and David M. Green

Psychology Section and Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Massachusetts

Wilson P. Tanner, Jr.

Cooley Electronics Laboratories, University of Michigan, Ann Arbor, Michigan

(Received August 24, 1961)

A different technique of analysis is applied to the experiment suggested by Harvey Fletcher for measuring the width of the critical band. This experiment determines the ability of noise bands of different widths to mask a pure tone centered in the band. The analysis considers two filters in series, one outside and one inside the observer. The width of the second filter (the critical band) can be estimated from measurements of the reduction in the noise power at the detector which is effected by the pair of filters. The width of the critical band is estimated under four different assumptions about the shape of the band. The results provide a context for discussing the reasons that may underlie the widely varying estimates of the critical bandwidth which have been obtained in previous studies.

INTRODUCTION

For the better part of a century, attempts to specify the process of auditory frequency analysis were based almost exclusively on anatomical and physiological evidence. Then, in 1940, Fletcher presented psychophysical data that gave a new form to the problem. He reported an experiment showing that only noise components in a narrow region about a pure tone are effective in masking the tone. This region he termed the ncritical band.nhmlwr

The existence of a critical band of frequency has since been clearly established in a variety of psychophysical experiments, including experiments on masking by tones or noise, on loudness summation, on phase sensitivity, and on detection of multiple tones and tones of uncertain frequency. Estimates of the width of the critical band, however—even those

Figure 14 Input for an Article of a Scientific Journal

THE JOURNAL OF THE ACOUSTICAL SOCIETY OF AMERICA VOLUME 34, NUMBER 1 JANUARY, 1962

On the Width of Critical Bands

John A. Swets and David M. Green
Psychology Section and Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Massachusetts

Wilson P. Tanner, Jr.
Cooley Electronics Laboratories, University of Michigan, Ann Arbor, Michigan
(Received August 24, 1961)

A different technique of analysis is applied to the experiment suggested by Harvey Fletcher for measuring the width of the critical band. This experiment determines the ability of noise bands of different widths to mask a pure tone centered in the band. The analysis considers two filters in series, one outside and one inside the observer. The width of the second filter (the critical band) can be estimated from measurements of the reduction in the noise power at the detector which is effected by the pair of filters. The width of the critical band is estimated under four different assumptions about the shape of the band. The results provide a context for discussing the reasons that may underlie the widely varying estimates of the critical bandwidth which have been obtained in previous studies.

INTRODUCTION

For the better part of a century, attempts to specify the process of auditory frequency analysis were based almost exclusively on anatomical and physiological evidence. Then, in 1940, Fletcher presented psychophysical data that gave a new form to the problem. He reported an experiment showing that only noise components in a narrow region about a pure tone are effective in masking the tone. This region he termed the "critical band."[1] an assumption since it was based on very few data, suggested that the critical band could be measured indirectly in masking experiments that used only broad-band noise. Fletcher later reported measurements based on broad-band noise; the critical bands so determined showed a similar dependence upon frequency and, again, the critical band in the region of 1000 cps was estimated to be approximately 65 cps wide.[2]

Examples of first pages of journal articles with their right-hand columns stripped in.

Figure 15 Output of a Scientific Journal Article

Figure 16 shows the role of secondary publications in journal publishing. Here many by-products are obtained from a single input source. Information items can be selectively extracted along with an associated set of typographic characteristics. An author line may occur one way on the first page of a journal article and in a different way in an author index, but since the item is identifiable per se and not by any secondary characteristic, the assignment of typography, by stored tables of control words, to each form presents no problem. The same concept holds true for indexes by title and for abstract journal entries.

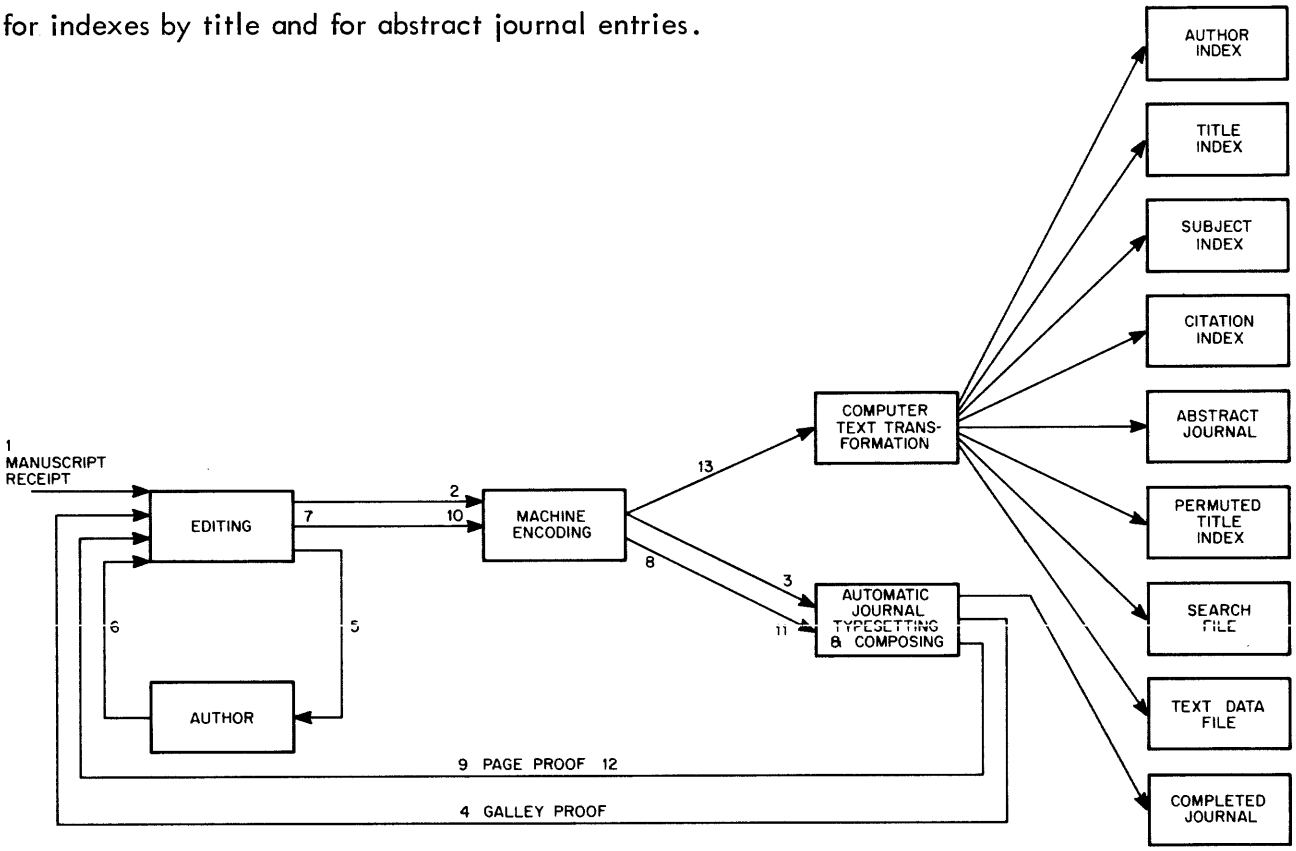


Figure 16 Secondary Publications in Scientific Publishing

Various computer-editing systems have been devised to speed the process of creating clean data. One such method is typewriter-controlled editing such as "Expensive Typewriter" for the PDP-1. This program allows addressing of a line, a set of change commands, and feedback of selected areas of text both before or after change.

A CRT editing program offers similar sets of change commands, but allows continuous feedback of selected portions of the data at higher speeds and allows data addressing by the light pen as a programmed moving cursor. Figure 17 is an example of CRT editing.

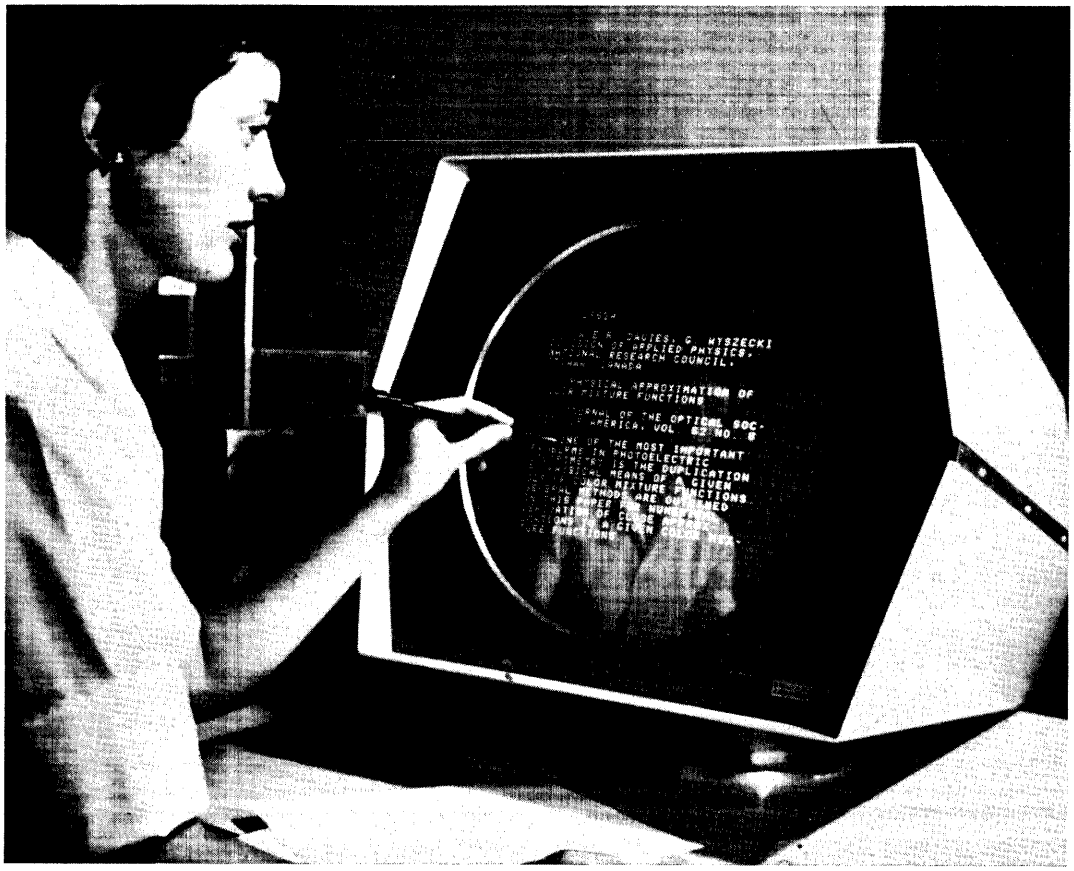


Figure 17 CRT Editing

Both these methods consume on-line time. In the case of typewriter-controlled editing, the operator tends to overuse the feedback capability as a confidence booster. In the case of text display, operator fatigue occurs more quickly than in reading a sheet of paper and editing with pencil. Time-sharing would reduce the cost of both these methods; but a text-producing system would be more economical with an off-line prepared merge system. The forte of both display-controlled and typewriter-controlled systems still is source program editing where fast turn-around time is required. One of the anomolies in correcting errors in serially prepared tape data is that the good data must be reprocessed to get at the bad. It is also necessary to make editing a function in itself--requiring a separate pass either in the tape puncher or on the computer.

A system which allows corrections to be merged into the main body of text as a preprocessor to any data handling system, whether file creation or automatic typesetting, has the obvious advantage of input-output time which is common with the data input-output time for the master

program. In addition, although correction data is entered, the corrected master tape is presented for processing having been inputed only once. Corrections can be prepared off line by clerical personnel. Some of the requirements for a good merge/edit system are:

Ease of Addressing - This means keeping the number of editing commands to a minimum to reduce ambiguous references. Programmers enjoy complex addressing schemes, but these tend to be subjective and usually do not work well in an operational system.

Proofreading - Errors will be made in preparing copy. The format should be designed so that format errors stand out and data errors will not be observed by control information.

A basic rule for systematic data handling is: "Let the program make concessions to the human operator and not the reverse."

REFERENCES

Buckland, L.F., "The Recording of Library of Congress Bibliographic Data in Machine Form," Council on Library Resources, Inc., Washington, D.C., 1965.

Buckland, L.F. "Recording Text Information in Machine Form at the Time of Primary Publication," Proceedings of the 26th Annual Meeting of the American Documentation Institute, Chicago, Ill., October 1963. pp 309-310.

Buckland, L.F., "System for the Recording, Processing and Exchange of Library Card Data," 1964 DECUS Proceedings.

Buckland, L.F., "The Use of a Cathode Ray Tube Display Console for Editing Textual Information," Proceedings of the 26th Annual Meeting of the American Documentation Institute, Chicago, Ill., October 1963. pp 179-180.

Buckland, L.F., and Lundy, J.T., "Study of the Use of Man-Machine Consoles for Text Processing," Report to International Business Machines Corporation, December 6, 1963.

Chaundy, T.W., Barrett, P.R., and Batey, C., The Printing of Mathematics, Oxford University Press, London, 1957.

Linotype Maintenance Manual, Mergenthaler Linotype Company, Brooklyn, New York, 1951.

Lundy, J.T.,"Representation of Complex Character Sequences on an Expandable Input Keyboard," Proceedings of the 26th Annual Meeting of the American Documentation Institute, Chicago, Ill., October 1963, pp 115-116.

The Monotype Casting Machine Manual, The National Committee of Monotype Users' Associations, London, 1952.

The "Monotype" Keyboard Operators' Manual, The National Committee of Monotype Users' Associations, London, 1956.

Nugent, W.R., "The Automated Editing of Text Via Computer and Off-Line Devices," Proceedings of the 26th Annual Meeting of the American Documentation Institute, Chicago, Ill., October 1963, pp 125-126.

Piner, S., and Samson, P., "Expensive Typewriter," DECUS Program Library, Digital Equipment Corporation, Maynard, Mass., February 1962.

Principals of Operation of the Photon S-560 System, Photon, Inc., Wilmington, Mass., 1963.

THE SAVAGE SYSTEM

A MONITOR SYSTEM

by Bernard I. Savage, Consultant*

Abstract The Savage System is a monitor system which is fully compatible with all previous DEC software for the PDP-4. The system requires one DECtape unit and occupies 2K of memory. It is controlled by typein commands that permit a user to load any program from DECtape, dump data onto DECtape, restore to memory previously dumped data, write data into memory, or print data from memory. The Monitor initializes, bootstraps, and housekeeps itself. It also writes programs onto DECtape for later loading. It includes a Teletype I/O package and a DECtape I/O package which respond to one-instruction macro-calls, and thereby perform all necessary Teletype and/or DECtape I/O programming functions.

INTRODUCTION

This paper describes the DEC PDP-4** Monitoring System (Savage System). Monitors generally perform certain functions such as loading a program, accepting corrections, emergency dumping, etc.; this Monitor contains these provisions and others for use by the machine operator via console key ins. A monitor is basically a way of automating the use of sundry software aids by program control reacting to the commands of the operator. The commands and their subsequent actions are determined by the uses for which the machine is programmed. This Monitor was designed and written for the uses of the Center for Cognitive Studies at Harvard University; however, the features of the Monitor are sufficiently broad and flexible as to be applicable elsewhere.

The rest of this paper is divided into three major sections. The first of these contains a detailed description of the Monitor and all peripheral software that is allied to it, such as the I/O

**Although the Monitor was originally created for Digital Equipment Corporation's PDP-4, it is completely applicable to DEC's PDP-7 as well. Thus, wherever the paper mentions the PDP-4, the reader may add "and/or the PDP-7."

System and Just-Loader, which will be explained later. Although this section is not vital to actually using the Monitor, it would help users in understanding other sections of the paper and in gaining insights for the most efficient use of the Monitor.

The second section is primarily a programmer's handbook. It indicates how each Monitor function may be used to enhance a program's design and/or operation and how certain combinations of Monitor functions may enhance the development of programming systems. Many of the functions and subroutines of the Monitor may be accessed by an internal program to save memory space and to bypass the need for operator intervention.

The third section is primarily an operator's handbook. It instructs an operator on the procedures for initiating and for using the Monitor. It describes a step-by-step procedure for each of the Monitor-Control key ins and for emergency or debugging situations. Although this section is designed for the use of operators, it is useful for programmers to know the standard operating procedures.

Included within the Monitor System is DECTRIEVE (modified) from DEC programming systems, and the Two-Teletype Package (modified) from DEC and the Center for Cognitive Studies at Harvard. This Monitor is completely compatible with previous software for the PDP-4.

SECTION 1

DETAILED DESCRIPTION OF THE MONITOR

This section, which is divided into three parts, contains a detailed description of the Monitor System. The first is a description of the Monitor, the second describes the I/O Package and Just-Loader, and the third is a memory map of space taken by the various parts of the Monitor and a format of the Monitor on DECtape. At least one DECtape unit must be a part of the user's configuration to use this Monitor. The other configuration requirements are a paper-tape reader, a Teletype-printer, and a Teletype-reader (keyboard). The Monitor occupies 2K of memory, including DECTRIEVE and the Two-Teletype Package.

We now introduce the term fixed-start. A fixed-start is merely a location which contains a JMP (or GO TO) instruction whose address accesses a certain Monitor function. By this method the user may hit START with any of several addresses in the ADDRESS switches, and thereby enter a given Monitor function or routine. There are four fixed-starts called F1, F2, F3, and F4 that exist at octal locations 50, 51, 52, and 53, respectively. They access Monitor functions as follows:

F1: Go To Initialize and Bootstrap
F2: Go To Monitor-Control
F3: Go To Create
F4: Go To Expand

These functions are defined in the description which follows. There are two spare locations at octal locations 54 and 55 with which the user may set up additional fixed-starts of his own.

PART ONE

The Monitor is divided into the following logical parts: Loader, Monitor-Control, DECTRIEVE, Two-Teletype, Create, and Expand.

Loader

The Loader, which is accessed by an F1, bootstraps the Monitor from tape (all uses of "tape" mean DECtape). The Loader itself must be in memory via a previous loading. The Loader is identical with the Just-Loader (described below) except for one difference—the Loader is not self-loading. The operation of the Loader is as follows:

> Turn off the interrupt. Clear flags and deselect all tape units. Rewind tape 7 and get the first block. The first block must be block one, the first word must be FLEX KIE, the fifth word must be FLEX PGM, the sixth word must be FLEX MTR, and no error flags must ever be raised or else the Loader will halt. If the foregoing legality checks are successful, the Loader brings the fixed-starts into memory, bypasses overlaying itself into memory, loads the bootstrap routine into memory, and gives control to the bootstrap (which some would consider a subset of the Loader).
>
> The bootstrap routine gets the next block from tape 7. The next block must be block two, the first word must be FLEX KIE, the fifth word must be FLEX PGM, the sixth word must be FLEX MTP, and no error flags must ever be raised or else the bootstrap will halt. If the foregoing legality checks are successful, the bootstrap will load the remainder of the Monitor into memory, halt tape 7 (which is ordinarily moving continuously), initialize a carriage return on the Teletype, display LODED, and give control to Monitor-Control.

Monitor-Control

To the user, Monitor-Control is actually the heart of the Monitor, for it is here that rapport, via key ins, exists between the system and the user. Monitor-Control is accessed by an F2 and always initializes a carriage return, displays BEGIN MTR, clears flags and deselects all tape unit selections, and awaits commands via key ins. Each key in will cause a particular action which returns to Monitor-Control (except for E and B) and a description of these actions follows; a command which cannot be recognized will yield a display of "?" as the Monitor-Control awaits a correct key in.

W - Write

The Write command is the way by which information is entered into memory under Monitor-Control. An actual address must follow which indicates the receiving location; next must come an A, O, or D for alpha, octal, or decimal, respectively, to indicate the class of information about to be received; next comes the information in three alpha characters, octal, or decimal, which is placed by Monitor-Control in the previously indicated address. The exact procedures for entering the data and for error-recovery options are detailed in Section 3, Operator's Handbook.

P - Print

The Print command is the way by which information is displayed from memory under Monitor-Control. An actual address must follow, indicating the source location; next must come an A, O, or D for alpha, octal, or decimal, respectively, to indicate the way in which the information is to be printed; Monitor-Control displays the information from the previously indicated address in the requested format. The exact procedures for retrieving such information and for error-recovery options is detailed in Section 3, Operator's Handbook.

D - Dump

The Dump command is a way of realizing the advantages of terminal and dynamic dumping. If a user must leave the machine and does not want to lose his uncompleted run, he may use Dump to place his portion of memory on tape for later restoration to memory (see below, Restore) and the continuation of his run. This is a terminal dump. A user may determine, perhaps as an aid to debugging or as a restart provision, to periodically dump a significant portion of memory onto tape for later examination or restoration. This is called dynamic dumping.

The Dump routine must receive the tape and block number onto which the dumped information will be placed, the starting and ending addresses of the dump from memory, a code by which the user will call his dump for restoration, the location within the user's program at which an interrupt must occur for dumping, and the number of times to dump. The foregoing is essential to provide for dynamic dumping and modified key ins of the foregoing are received for terminal dumping. In either case, the Dump routine uses the key ins to provide for dumping, be they

terminal or dynamic. The procedures for dumping and for error-recovery options are detailed in Section 3, Operator's Handbook. Some suggestions for the way in which a programmer may use the Dump routine in communication with his own program are detailed in Section 2, Programmer's Handbook.

R - Restore

The Restore command complements dumping and is the way by which a terminal dump is replaced into memory for continued processing. The Restore routine receives the code (three alphanumeric characters) which identifies a given dump and the tape unit number on which the dump exists. Restore will search the tape for the dump and replace it into memory at the exact location from which it first came. The Dump routine precedes the information being written onto tape with certain checkwords, and Restore performs legality checking for these checkwords in selecting the block(s) of dumped information for restoration. The Restore procedures and error-recovery options are detailed in Section 3, Operator's Handbook.

L - Load

The Load command is the way by which a user's program is brought into memory for subsequent execution from tape. The Load routine must receive a three-character alphanumeric code and subsequently searches the program tape for the program. The Expand routine (described below) has originally written the program onto tape 7, at some point beyond those blocks occupied by the Monitor, in a given format. As the Load routine searches the tape, it performs legality checking for format and machine errors and keeps track of its block position on the tape. By this tracking, the Load routine is able to search for another program from the place where Load last completed a program loading. In this way, a user may sequentially call for programs (possibly those that form a system) without time-consuming rewinds and searches from the beginning of the tape and without the need to remember the first block number for each program. The Load routine may also be used as a subroutine by an internal program for automated loading within a system of program communications.

If Load does not find the requested program on the first pass, it will go back to the beginning of the tape and search forward, but will not proceed beyond the point on the tape at which it initially started. Load will not load a program over Monitor. DECTRIEVE is used by Load

to finally bring the requested program into memory. After successfully loading the program, Monitor-Control will display LODED PPP where PPP represents the program name. The executable starting address of the program will be stored for use by the Execute routine (discussed below). The exact procedure for using Load and all accompanying error printouts is more fully discussed in Section 3, Operator's Handbook. Some programmer suggestions for using Load as a subroutine are discussed in Section 2, Programmer's Handbook.

E - Execute

The Execute command initiates the operation of the last program brought into memory. Monitor-Control will display BEGIN PPP where PPP is the name of the last program called for, and branch to the beginning address of that program as it was saved by Load (see Load, above). Control passes from Monitor to the user's program. (See Accumulator, below.)

B - Begin

The Begin command is the way by which a user may start instruction execution at any location in memory. The Begin routine must receive an octal address and in essence performs a JMP (GO TO) to the indicated address. Before this JMP, Monitor-Control will display BEGIN PPP where PPP is the name of the last requested program for loading. (See Accumulator.)

Accumulator

This is not a Monitor-Control command, but a feature. It is often desirable to set the accumulator to some value before starting or continuing a program. The tag ACC has a guaranteed address, and by using the Write command (see above), the user may set any value into ACC. The Begin and Execute routines set the accumulator from ACC before branching and relinquishing control.

Create

Create, which is accessed by an F3, is the routine which writes the Monitor System (including Create) out onto tape for later use. The Monitor, as it appears on tape, has a particular format which Create guarantees. If any error occurs in writing the tape, Create comes to a halt.

The Monitor System is always written at the beginning of a tape, beginning in block one. One additional block is written—the end block, which indicates that there is no more information on this program tape.

Expand

Expand, which is accessed by an F4, is the routine which places programs onto the tape that initially contains the Monitor System (see Create). In this way, a program tape is developed which has the Monitor at its front and succeeding programs following in the user's predetermined order. The programs that go onto tape by using Expand get into memory via the Load or Restore functions of Monitor-Control, or via DEC's RIM Loader. In essence, memory must contain a binary program and the user must indicate the starting address, ending address, name, and beginning address of the program in memory. An F4 afterwards allows Expand to write the program on tape by writing over the end block and rewriting a new end block at the updated end of the tape (see Create). Actually, Expand keeps track of the last block which was used, but if this tracking count is zero, Expand searches for the end block's position. Thus, by using a fresh monitor in which the tracking count is zero, a user may expand previously expanded tapes. A description of key ins to use Expand is detailed in Section 3, Operator's Handbook.

DECTRIEVE and Two-Teletype

The foregoing constitutes a description of what may be called the basic Monitor, as written by the author. The following is a description of two routines that are also a part of the Monitor System—DECTRIEVE and Two-Teletype—which were written by DEC and Dr. Donald Norman and Michael Stein of the Center for Cognitive Studies, and which have been modified by the author.

DECTRIEVE

The DECTRIEVE program was written by DEC to allow a user to write and read tapes by manually setting the accumulator switches. DECTRIEVE incorporates and uses the other DEC programs for searching, reading, and writing tapes. These other programs are unaltered within the Monitor System and may be called as closed subroutines by a user in his internal program. For a more detailed explanation of the search, read, and write subroutines, see Digital

Equipment Corporation's DECtape manuals and page 20 of this paper. DECTRIEVE may also be used as a closed subroutine, as described below. The modifications to DECTRIEVE were the addition of housekeeping routines to preserve the search, read, and write subroutines for use individually, and changes that made DECTRIEVE automatic under program control, rather than manual. The Monitor System uses this modified DECTRIEVE extensively, as does the I/O Package.

DECTRIEVE may be accessed by JMS MTREAD, for reading data from tape into memory, or by JMS MTWRT, for writing data onto tape from memory. Both MTREAD and MTWRT use a location called UNT BLK, which contains the specified tape unit and block number for selecting the proper tape and for searching for the requested block. MTWRT also uses a location called FROM, which contains the starting addresses of memory to be written, and TO, which contains the ending address of memory to be written. These tagged locations are identical in format to that of the manual ACS. The user's program should set the necessary locations and perform the appropriate JMS for reading or for writing. Upon returning from DECTRIEVE, the user may sense a location called ERROR—zero implies a successful read or write, and nonzero implies an error. For a complete explanation, see DEC's manual on DECTRIEVE.

Two-Teletype

The Two-Teletype program routines consist of a modified I/O Teletype program as supplied by DEC, with additions to control a second Teletype. Approximately 150 locations could be saved if only a one-Teletype package were used. Dr. Donald Norman of the Center for Cognitive Studies not only modified DEC's package, but also added some very useful routines for page line and spacing control. The Monitor System uses the Two-Teletype Package extensively for all Teletype I/O, even though but one physical Teletype is used.

Two additional routines have been added to the Two-Teletype Package by the author. The first is OCTNMB, for accepting an octal number from the keyboard. The key in procedures are identical to those for DECNMB (Dr. Norman's decimal number key in routine), except that only octal key ins are valid. The second is RCHXX1 (or RCHXX2 for second Teletype), which accepts three-character key ins. In essence, RCH1 (or RCH2) is performed three times, and this routine is subject to the same constraints as Norman and Stein's RCH1. However, three characters must be keyed in.

PART TWO

The Monitor System is designed to reside on tape, as the first program on tape 7, and be called in from tape 7 by an F1 whenever memory is clobbered or whenever the user wishes to renew the Monitor in memory. An initial bootstrap routine is needed which can be loaded by the RIM Loader via the paper-tape reader for a first-time initialization or for cases in which the Loader (F1) is clobbered. The program which serves this purpose is called Just-Loader. The Just-Loader is identical to the Monitor Loader. The description in Part One of the Loader would serve here to describe Just-Loader. There is only one difference—the Just-Loader does not check for itself to bypass reloading itself, but loads the complete loader from tape into its assigned locations before giving control to the Monitor-Bootstrap routine (see Loader), which completes loading the Monitor. Therefore, the Just-Loader must be SETLOCed to some address above octal three hundred. If an error halt takes place during the loading, the Just-Loader must be reloaded via the RIM Loader. Within Just-Loader there is a tag GOLDR which is equal to the tag LDR (bootstrap begin) in the Monitor. If LDR's location changes within the Monitor, by a reassembly perhaps, Just-Loader must be reassembled to equate GOLDR to LDR.

The entire Monitor System occupies about 2K of memory, and it would be unfortunate if many of its routines could not be used by running programs. Actually, in Section 2 there is a list of subroutines and constants which are defined in DEC's Assembler for a programmer's use. The Input/Output Package is a program that has sundry routines for Tape I/O manipulations; all of which rely upon the I/O routines already in the Monitor. The I/O Package may prove useful to a user because the programmer does not need to be a DECtape expert to use DECtape programming features. Moreover, the very same DECtape programming which must be reprogrammed into each newly written program may be avoided by assembling with the I/O Package, just as Teletype reprogramming is avoided by assembling with the Teletype Package. In fact, most of the I/O consists of routines in the Monitor which do not involve assembly.

The I/O Package writes data tapes. A program tape is defined as one which begins with the Monitor, is followed by more binary programs of the user, and ends with an end block. It is always on tape 7. The format for a program tape follows in Part Three of this section. A data tape may be created or exist on any tape drive. The first block on tape is a special

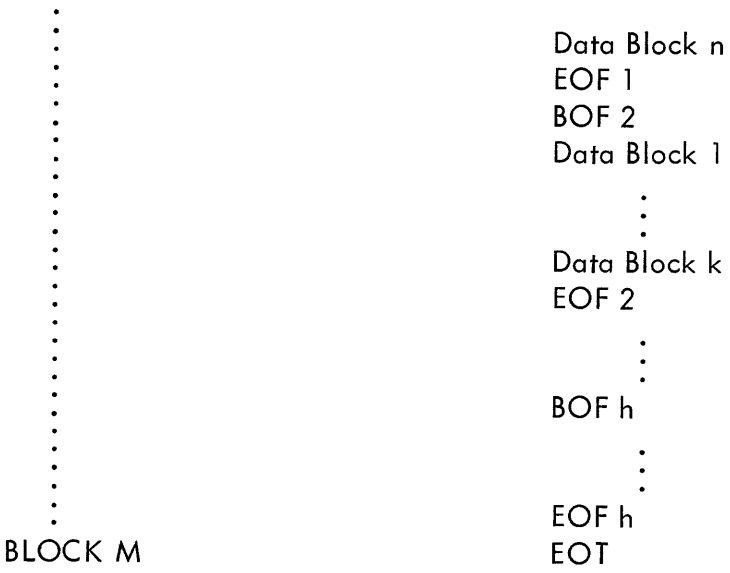
leader block called the beginning-of-tape (BOT). The last block on the tape is a special trailer block called the end-of-tape (EOT). Data files exist between the BOT and the EOT. There may be any number of files of varying lengths; each file contains a leader block called the beginning-of-file (BOF) and a trailer block called the end-of-file (EOF). There may be any number of data blocks between the BOF and the EOF. Each of the aforementioned blocks has a unique value in its first two words. All blocks are written onto tape by DECTRIEVE, which adds four words at the beginning of each block. Therefore, a data tape may be read by DECTRIEVE as well as by the I/O Package.

The routine of the I/O Package may be used to write a BOT, BOF, EOT, EOF, and data blocks; they may be used to read a block from tape and/or to position the tape to EOT or BOF; and they may be used to check that the proper file has been found. They will keep track of sequential blocks during reading and writing so that the user need not concern himself with this, and during reading they will check for the occurrence of an EOF. The I/O Package is designed to use up to one input tape and one output tape. However, with only slight program housekeeping and manipulation, any number of tape units may be used for input and/or output.

The user is required, depending upon his I/O needs, to define six parameters which supply the I/O Package with such things as tape unit numbers, initial block numbers, file names, and starting and ending locations of the memory area from which or to which data will pass. He must also define his own error exits. These parameters may be set by key ins or at assembly time or by the internal program itself at execution time. Manipulations of the parameters permit the use of many tape units and varying I/O operations that are limited only by the programmer's imagination.

Tape Format—Data Tape

BLOCK 1	BOT
BLOCK 2	BOF 1
BLOCK 3	Data Block 1
BLOCK 4	Data Block 2
⋮	⋮



Block Format*—Data Tape

BOT:
- Word 1: FLEX BEG
- Word 2: FLEX BEG

BOF:
- Word 1: FLEX BEG
- Word 2: FLEX (FILE NAME)

DATA:
- Word 1: FLEX DAT
- Word 2: FLEX (FILE NAME)

EOF:
- Word 1: FLEX END
- Word 2: FLEX (FILE NAME)

EOT:
- Word 1: FLEX END
- Word 2: FLEX END

A typical use of the I/O Package would be as follows. Write a BOT on the output tape. Read BOT from input tape and check that it is a BOT. Position the input tape to the desired BOF and read in BOF. Read the first data block and check that it contains the proper code (file name). Write a BOF on output tape, perhaps after positioning it to some BOF or EOT. Perform

*There are four DECTRIEVE control words at the beginning of each block.

operation on the input data and write data block onto output tape. Read the input tape...and so on until an EOF is sensed on the input tape. Write last data block onto output tape, followed by EOF and EOT on output tape. You now have an updated output tape. Aside from defining a few parameters and executing the JMS for the appropriate routine, all I/O processing was done automatically by the I/O Package.

PART THREE

The Monitor System occupies the first 2K of decimal memory. All addresses indicated in this part are in octal. The first 47 locations are available to the user except for the following: location 21 is set to a halt whenever the Loader (F1) is executed; location 10 is the index register used by DECTRIEVE and by OCTNMB and DECNMB within Two-Teletupe, but 10 is available in between the uses of the foregoing; and locations 41-47 are used during the execution of Create. Location 1 is also initialized (see page 20). These lower locations in memory must be set internally for a programmer's uses, for Load does not set anything in a location below 4000. The segments of the monitor and the memory area occupied by each part are:

Loader	0050-0316
Control	0317-1147
DECTRIEVE	1150-2227
Constants	2142-2227
Two-Teletype	2236-3505
Create	3506-3645
Expand	3646-3775
User's Program	4000-

Throughout the Monitor there exist constants and routines that are defined in the Assembler and may be used by a programmer, as outlined in Section 2. The rest of memory is available to the user. He may wish to use the I/O Package, which would eclipse approximately another 165 decimal locations of memory, equal to the savings of using the One-Teletype Package.

The program tape is defined as a tape which has the Monitor System at the beginning and is followed by user programs, the last of which is followed by an end block. The format of a program tape which must be on tape 7 is (in octal notation):

Block 1	Loader and bootstrap
Blocks 2-11	Rest of Monitor System
Blocks 12-m	Program 1
Blocks m-k	Program 2
⋮	⋮
Block j	End block

Blocks 1 and 2 have four DECTRIEVE control words added to them. However, the Monitor cannot be read by DECTRIEVE but only by Just-Loader or F1. The four DECTRIEVE control words which appear at the beginning of a starting block for programs, dumps, and data blocks on program tapes and data tapes are:

Word 1	FLEX KIE
Word 2	Beginning address of memory, minus one
Word 3	1's complement of the number of words for the program, dump, or data block, plus one.
Word 4	Saved contents of location one.

The additional control words that are added by the I/O Package to data blocks are described in Part Two of this section. The additional control words that the Monitor System adds to program and dump blocks are:

Programs		Dumps	
Word 1	FLEX PGM	Word 1	FLEX DMP
Word 2	FLEX PROGRAM NAME	Word 2	Address of Dump
Word 3	Location of first instruction	Word 3	FLEX DUMP NAME

DECTRIEVE may be used to read programs and/or dumps from the program tape in addition to Load and Restore, respectively.

SECTION 2

PROGRAMMER'S HANDBOOK

In this section some of the ways by which the Monitor System as a whole, or some of its features, may be used to enhance programming and system design are explained. This explanation is based upon two premises. The first is that most of the suggestions that follow are not readily self-evident, and rather than wait until time and experience force their evidence it is best to highlight them now. The second premise is that many of the suggestions which follow come from an intimate knowledge of the Monitor, a knowledge that could come to the user only after detailed study of the coding itself, and rather than require the user to gain this knowledge it is best to immediately suggest the procedures that could exploit such knowledge. It is advisable for the programmer to be familiar with the information contained in Sections 1 and 3. In fact, these sections contain some implications for programmer use and consideration. It must also be noted that the following suggestions are not at all exhaustive, and that some imaginative programmer may view them as merely a preamble to far more extensive uses. This section is designed to be reasonably independent of the total documentation; therefore, we first briefly review the Monitor System.

The Monitor System is divided into three parts. The first part is the Loader and Bootstrap routines (F1), which, if they are in memory, will load the entire Monitor System into memory from tape 7. The second part is Monitor-Control (F2), which responds to key ins or fixed starts to perform the following functions:

Write information into memory from key ins.
Print information from memory from key ins.
Execute the operations of a program.
Begin at any location in memory.
Dump memory, or a portion of memory, onto any tape.
Restore to memory information that had been dumped.
Load a program from tape 7 into memory.
Create (F3) a program tape by putting the Monitor onto tape 7.
Expand (F4) a program tape by adding programs to tape 7.

The third part consists of the Two-Teletype Package and DECTRIEVE (including the read, write, and search routines). Adjuncts to the Monitor System are the Just-Loader, a paper tape bootstrap to get the Monitor into memory from tape, and the I/O Package, a generalized method for writing and reading data tapes by macro cells.

Packing

The Monitor takes up 2K of memory. The rest of available memory may hold programs and/or data. Programming projects may be planned that involve several programs cooperating with each other. Any one program may occupy core at a given moment and determine by itself, or leave determination to the operator, of which program to next call in for operation. The following sequence of code should be used for a program to automatically call in the next program:

```
LAC (FLEX XXX); where XXX is program name
DAC PPP
JMS LOAD1
```

To immediately execute Program XXX, the next instruction should be JMP EXCUT.

If the accumulator is preset to some value at the beginning of Program XXX, the coding should be preceded by:

```
LAC...; contents for accumulator
DAC ACC
```

It may be desirable to create a string of programs that are to follow in sequence. Without the necessity of planning for calling each other by name (except for the first), each program may call for the next one by:

```
LAC NOPE
DAC ITSELF
JMS LOAD1
LAC SELF1
DAC ITSELF; restore Monitor to original
```

The above comments for execution and accumulator setting apply.

NOTE: Caution must be exercised that the program being loaded does not clobber the above sequences of coding.

Of course, if memory is not exceeded, several programs may occupy core and communicate with each other. The programs could be loaded without risk of one clobbering the other by manual Load commands. A given program may require many kinds of tables for its operation, and the totality of the tables may exceed memory. If the tables were originally written out onto a program tape via the Expand function, they could be accessed as needed by the program through the use of any of the appropriate methods of automatic loading as outlined above.

Loading

The Monitor will not load any program below octal location 4000. However, if the user has no need for the Monitor after a program has been loaded, he may elect to clobber lower memory for the creation of tables or for the use of temporaries. If possible, nothing below octal 317 should be clobbered, for when the given run is over an F1 would restore the Monitor as the Loader/Bootstrap routines below octal 317 have been preserved. If any parts of the Monitor are to be clobbered, nothing can be guaranteed about the intact preservation of DECTRIEVE, uses of the I/O Package, or the Two-Teletype routines. Subject to these misgivings, even manual or automatic loading below octal 4000 may take place by manipulating the contents of location TOP, whose content is the location minus one below which loading may not occur (it currently is set to 3777).

Parameterization

The key in and printout features may be used in several ways. The first use would be as a debugging tool. The Write function of the Monitor may be used to modify locations, to set breakpoints, to insert patches of code, to display the contents of locations, and to manually perform the other actions associated with debugging. These same functions may be performed by DDT, of course, but for programs that work in conjunction with the Monitor and are too large to share memory with DDT, these manual methods should prove quite useful. Furthermore, it may be desirable during running time, when DDT is not in memory, to modify a location or display a location. This may be done by using Write and Print, respectively, rather than employing the console switches; especially since a written record is left on the flexowriter.

The key in features may prove useful as a programming design consideration. A program or system of programs may be designed that is broad and general, and takes direction from a set of parameters (i.e., a variable constant). This means that at execution time the program(s) takes direction from a set of locations which may contain variable information and that these locations are set, within predetermined bounds, by key ins before the program begins its run. The parameters may take on any of a host of values, each determining or dictating a different course of action by the program. The facility of setting and changing such parameters via key ins and printouts enhances such design considerations and, again, a written record remains on the flexowriter.

The key ins, lastly, are a means of setting sundry fixed starts in either the spares set aside for this purpose within the Monitor (i.e., locations 54 and 55, octal) or some specified location within the body of a user's program. The fixed starts may serve any of many purposes, depending upon a programmer's inclination.

Dumping - Restoring - Restarting

The Dumping and Restoring features of the Monitor System are described in Sections 1 and 3, and the reader should be familiar with the ideas and operating procedures for Dump/Restore. Terminal dumping is readily understandable as to its uses, and the procedures in Part Two of Section 3, for debugging and maintenance, may well be exhaustive. However, for terminal and dynamic dumping, it may be desirable to prepare for automatic dumping rather than depend upon operator key ins. This is especially appropriate in a system of programs where each program has different dumping requirements. Automatic dumping may be had in one of two ways:

1. Set the Monitor Dump routine parameters. In essence, the Monitor Dump routine is set as if key ins were manually inserted and dumping will automatically take place. The following locations must be set to the indicated values:

Dump2X+0	Tape unit and starting block number of UNTBLK.
Dump2X+1	Beginning address of Dump.
Dump2X+2	Ending address of Dump.
Dump2X+3	Code (three-character) for Dump.

Dump2Y	(Minus the number of times to Dump) +1.*
Dump2Z	The address of the location at which a JMP to the Dump routine will be placed.*
RESTOR	The contents of the address in DUMP2Z.*

Then place a JMS DUMP2R into the address specified by DUMP2Z.*

2. Set your own parameters internally to your program and use the output portion of the Dump routine. The procedure for dyanmic and terminal dumping is:

a. JMS DUMP 1
b. ...
c. ...
d. ...
e. ...

These values are the same as those indicated for DUMP2X+0 through DUMP2X+3, respectively, above.

The programmer must keep track of how many Dumps he wants.

To use the foregoing routine for terminal dumping, it is only necessary to set up a fixed start temporary as a JMP to the location occupied by (a) in the above routine.

The Restore function will restore only the first occurrence of XXX, where XXX is the requested code in an RXXX operation. Therefore, if the user is performing dynamic dumping and wants to use Monitor Restore to access any one of the Dumps, he should continuously change, manually or automatically, the code so that each one is unique. If he is doing automatic dumping this involves modifying the contents of either DUMP2X+3 or (e) above.

Another useful approach would be that of Restart. A program or system may operate over a long period of time and continuously yield intermediate results that are reused or elaborated upon. It would be very unfortunate if toward the end of such a run a machine malfunction or previously unknown programming error is suddenly manifest and effectively nullifies hours of running time. Periodic dumping would serve as a safety measure, so that only the time and/or effort between the last Dump and the nullification is really lost. Even then, it may be

---

*For eventual terminal dumping possiblities, these values are: minus zero, octal 51, JMP F2, 51.

possible to Restore later and continue and finish successfully in a short time what would have taken much longer to redo. Restart provisions are probably best provided for by automatic means, as suggested under method 2 above. The same portion of tape is reused as each succeeding Dump is rewritten over the previous one. Thus, only one and the latest Dump remains for possible Restart uses.

One final point is that after a Dump is restored to memory, the location tagged ADDRES contains the starting area minus one of memory where the Dump began.

I/O Package Notes

In the discussion which follows, it is assumed that the reader is familiar with the uses and format of the package as outlined in Section 1. To write information from memory, the I/O routines must know where the data begins (WRBGDT), ends (ENDAT), and the tape unit and starting block number (WRBLOK). The I/O routines use DECTRIEVE to read the same information back into memory, but require their own sets of parameters; therefore REBLOK is the same as WRBLOK, and RDBGDT is the same as WRBGDT. If the input positioning routines are used first, then WRBLOK need not contain a block number as it will be set into WRBLOK after the specified BOF or EOT is found. The user may want to ascertain that the block just read into memory contains his code name for it (i.e., it is his data), and the CHKCOD routine uses DATCOD for this purpose. Often situations arise in which data is not found or something goes wrong, and for these situations the I/O routines provide halts and/or a printout. These error exits may be modified by the programmer for his own uses.

Except for POSBOF and POSEOT, the I/O routines that cause tape movement always exit with the tape halted and with the interrupt OFF. The user must remember that when he directly uses the DECtape READ or WRITE routines in DECTRIEVE, he continues with the interrupt ON. The only printout from the I/O Package is NOTFND, which results when a BOF or EOT could not be found after a search.

Interrupt and DECtape Routine Notes

The DECtape I/O of the Monitor and the independent use of the Search, Read, and Write routines require the use of the interrupt and location 1. Location 1 is initialized as a JMP to a

body of code which interrogates the raising of DECtape data or error flags. If either flag has been raised processing continues in an appropriate routine—data or errors and the interrupt are properly serviced. Thus, it is not necessary for the programmer to alter the contents of location 1 and arrange his own code to check the raising of DECtape flags. In fact, to avoid trouble and unnecessary work, he should not do so.

If neither flag has been raised, the routine exits via a JMP10, after turning the interrupt back ON. If the programmer wishes to check other flags he should do so by code within his own program and access the code by placing a JMP to that code into a location tagged INTEX within the Monitor. The user may then continue to service his flags—the interrupt will be off. Hopefully, NOPE will be replaced into INTEX at program completion.

Subroutines and Constants

Any symbolic tag which appears in this document as a part of the Monitor is defined by the assembler and is also, therefore, in the definition list. Of course the user may take advantage of any of the values or routines within the Monitor System by using "equals" or absolute addresses; and he is urged to peruse the listings to gain the familiarity required to do this.

The definition list contains many tags that are known because they come from the Teletype Package and/or DECTRIEVE. Again, it would be useful to peruse the Teletype package as it is well documented as to its uses and functions. In what follows, we now specifically mention certain constants and subroutines within Monitor which the programmer may need and find useful.

TABLE 1 CONSTANTS

Tag	Value	Comment
BLKMSK	007777	Used for block number masking.
END	FLEX END	
END LST	777777	Minus zero.
HALT	HLT	
KIE	FLEX KIE	DECTRIEVE control word.
MONE	777776	Minus one.

TABLE 1 CONSTANTS (continued)

Tag	Value	Comment
MOVEBK	070060	MMLC constant for tape movement backward.
MTP	FLEX MTP	Loader control word.
MTR	FLEX MTR	Loader control word.
NOPE	NOP	
ONE	+1	
PGM	FLEX PGM	Loader control word.
PPP	——	Contains program name.
READFD	070042	MMLC constant for tape movement—read forward.
SRCHFD	070041	MMLC constant for tape movement—search forward.
THREE	+3	
TWO	+2	
UNTMSK	170000	For masking tape unit selection.
WRTFWD	070043	MMLC constant for tape movement—write forward.
ZERO	+0	

TABLE 2 SUBROUTINES (IN ADDITION TO THOSE ALREADY MENTIONED)

JMS	Function
CLRFLG or CLFLAG	Clears all flags and deselects all tape units, but does not turn OFF the interrupt if it was on.
S1	Puts a selected tape into the search forward mode and moves it.
S2	Moves a selected tape backwards to beginning of tape and exits with tape moving backward.
S3	Puts a selected tape into read-forward mode and moves it.
S4	Checks if a physical end of tape flag was raised; if not, it exits and if yes it exits to the exit address plus one (i.e., a two-word calling sequence is required).
S7A	Deselects all previous tape unit selections.

SECTION 3

OPERATOR'S HANDBOOK FOR USING THE MONITOR SYSTEM

This section details the step-by-step procedures for using the Monitor System. Although it is basically written for operators, it is helpful to programmers.

The section is divided into two logical parts, the first of which delineates the various key in and set-up procedures for each of the Monitor Commands or functions discussed in Part One in Section 1. The second part suggests and discusses ways of using the commands or functions of Part One to perform sundry tape maintenance procedures, and to aid and systematize the testing/debugging of programs into final versions for a programmer's use.

PART ONE

To use or execute each Monitor-Command function, follow this procedure:

Function	This details what is being done and its purpose.
Format	Specifies the key ins and their order.
Error Messages	Explains each possible error message.
Recovery	Offers alternatives of procedure for error messages.
Result	Indicates the results of successful execution of a function.

The Monitor always housekeeps itself so that nothing significant is altered or destroyed. One recovery procedure that is applicable for all key ins in octal or decimal is:

To key in an octal number:

1. All digits must be less than 8.
2. A maximum of six digits may be keyed in.
3. Leading zeros need not be keyed in.
4. Line-feed must terminate the key in.

5. To begin the octal number key in again (presumably because the wrong number was about to be inserted) before line-feed has been used, hit any nonoctal character.

6. Negative values for octal numbers must be entered in their complemented form.

To key in a decimal number:

1. All digits must be less than 10.

2. The value of the number must be less than $2^{17}-1$.

3. Leading zero need not be keyed in.

4. Line-feed must terminate the key in.

5. To begin the decimal number key in again (presumably because the wrong number was about to be inserted) before line-feed has been used, hit any nondecimal character.

6. To indicate negative value, precede the number by minus sign.

In all of the following formats that indicate the key in of octal or decimal numbers, it is never necessary to hit figures before keying the numbers or to hit letters after keying the numbers. However, doing this will not affect the information. The letters and figures keys are exceptions to the two 5 rules given above.

The following definitions hold throughout (octal addresses):

F1	Set the ADDRESS switches to 50 and START
F2	Set the ADDRESS switches to 51 and START
F3	Set the ADDRESS switches to 52 and START
F4	Set the ADDRESS switches to 53 and START

FUNCTION - To Initialize the Monitor System from Paper Tape to Tape

FORMAT - Perform the following steps:

1. With the RIM Loader in upper memory, place the binary Monitor in the reader, set the ADDRESS switches to 17770, and hit START. Wait for the reading to be completed.

2. Place a blank tape on unit 7.

3. F3.

ERROR MESSAGES - None. If the system halts, the function failed.

RECOVERY - Try again at step 3 above or do an F2 to regain Monitor-Control.

RESULT - The entire Monitor System is written onto tape 7 at the beginning of the tape. Control is given to Monitor-Control with the display BEGIN MTR; the binary paper tape Monitor need never be used again.

FUNCTION - To Create a Program Tape by First Placing the Monitor on Tape

FORMAT - Identical to the above function, beginning at step 2.

FUNCTION - To Expand a Program Tape by Adding Programs to it - The program is in memory either by using the RIM Loader or by using Load.

FORMAT - Perform the following steps (see the Write command):

1. W40OXXXXXX; XXXXXX is octal starting address of program.

2. W41OYYYYYY; YYYYYY is octal ending address of program.

3. W42AAA; AAA is three-character name of program.

4. W43OZZZZZZ; ZZZZZZ is octal location of first instruction. Terminate each sequence of digits with line-feed.

5. F4.

ERROR MESSAGES

1. BAD implies that this program could not successfully be placed onto tape. EXPAND will attempt to preserve the program tape by rewriting the end block where it used to be.

2. A halt implies that the end block could not be written.

RECOVERY

1. If BAD occurred with no halt, the tape is good and another attempt at writing the program should be made, beginning at step 5 above.

2. If BAD occurred with a halt, the tape is semi-good but does not have an end block. An immediate attempt should be made to write the program by beginning the program at step 5.

3. If a halt alone occurred, the program is successfully on tape but the tape does not have an end block. This is not serious if the user is to continue EXPAND by adding more programs, as the Monitor remembers at what block to begin the next program. An F2 will restore Monitor-Control, and the user may continue at step 1 under Format. However, if the user does not intend to immediately continue expanding the program tape, the tape:

   a. cannot be expanded in the future unless BLKTRK is manually set to its next available block; and

   b. loading of programs from this tape may be exceedingly slow.

4. As a last ditch effort to write an end block under the previous circumstances, the user may do an F2 followed by a BXXXX, where XXXX is the octal equivalent of tag EX3 (see Begin).

RESULT - The program as defined in locations 40-43 is written onto tape and the tape is closed by an end block.

FUNCTION - Bring the Monitor into Memory from Tape Reloading System

FORMAT - Perform the following:

1. F1, if the loader is intact in memory; or

2. Place the binary Just-Loader in the reader, set the ADDRESS switches to 17770 with the RIM Loader in upper memory, and hit START. (A program tape must be on tape 7.)

ERROR MESSAGES - None. A halt indicates an error.

RECOVERY - Try again at steps 1 or 2 above.

RESULT - LODED, BEGIN MTR are displayed as the Monitor is brought into memory, and control resides in Monitor-Control. BLKTRK is equal to zero. It should be standard practice for each new user to do an F1 when he gets on the machine.

Monitor Control

FUNCTION - To Get Monitor Control Whenever the Monitor is in the Machine

FORMAT - F2.

ERROR MESSAGES - None.

RECOVERY - Reload the Monitor System if anything goes wrong.

RESULT - BEGIN MTR is typed and control passes to Monitor-Control.

Monitor-Control will respond to a key in letter of W, P, D, L, R, E, or B in the formats which follow. Any other key in will cause the display of "?" as Monitor-Control awaits a correct letter to be keyed in as a trigger to the appropriate function.

FUNCTION - To Write Information into Memory

FORMAT - WXRY, where:

X equals an octal address representing the receiving location;

R is O, D, or A for octal, decimal, or alpha, respectively, and represents the class of information to follow; and

Y is an octal number, a decimal number, or three alphanumeric characters, depending upon R.

ERROR MESSAGES - None.

RECOVERY - To interrupt the function, do F2.

RESULT - Y is set into X.

FUNCTION - To Print Information from Memory

FORMAT - PXR, where:

X equals an octal address representing the source location;

R is O, D, or A for octal, decimal, or alpha, respectively, and represents the way in which the data is to be represented on the printer.

ERROR MESSAGES - None.

RECOVERY - To interrupt the function, do F2.

RESULT - The contents of X is displayed in octal, decimal, or alpha.

FUNCTION - To Dump Information onto Tape During Execution Time

FORMAT - T(6) AECBT, where:

T(6) is for six octal digits representing the tape unit and intial block number upon which to Dump, in the same format as a manual ACS for DECTRIEVE;

A is an octal starting address of Dump area;

E is an octal ending address of Dump area;

C is your own three-letter code for naming the Dump;

B is an octal location at which point the running of the program is interrupted to perform the Dump. The instruction normally in B is executed after the Dump and the program continues.

T is a decimal number representing the number of dumps requested. If T is zero, there will be continuous dumping; otherwise, dumping will cease after T dumps.

ERROR MESSAGES

1. "?" implies that E is not greater than A.

2. BAD DMP during execution time implies that the program is doing its own internal dumping, but has set A not greater than E.

RECOVERY

1. Try again, with E greater than A.

2. None. The internal program has been modified not to Dump.

RESULT - See result under next function.

FUNCTION - To Dump on an Emergency or Terminal Basis

FORMAT - Same as above function, but:

1. B should be 51.

2. T should be 1.

3. The last step should be followed by an F2.

ERROR MESSAGES - Same as above.

RECOVERY - Same as above.

RESULT - For all dumping, DECTRIEVE writes the area out onto your specified tape as many times as requested (one for Terminal Dump). If there are several dumps, the routime keeps track of your block numbers and restores normal processing after the last Dump; this also takes place on a Terminal Dump. The DECTRIEVE typeouts should be monitored to see if any tape error occurred while dumping.

FUNCTION - To Restore Dumped Data to Memory from Tape

FORMAT - T(6) RAAA, where:

AAA is the three-letter name originally keyed in for C under Dump (see Dump function);

T(6) is six octal characters which represent the tape unit block number manual ACS used in DECTRIEVE. Only the unit number is necessary, as the block number given is disregarded.

ERROR MESSAGES - NOTFND, which implies that the Dump named AAA could not be found on the specified tape. Always check the DECTRIEVE typeout to see if the dumped data did come in successfully, even if NOTFND does not appear.

RECOVERY - None. Try again.

RESULT - The dumped information under AAA is restored into memory.

FUNCTION - To Load a Program from the Program Tape into Memory

FORMAT - L AAA, where:

AAA represents the name of the program as it was put out onto tape (equivalent to AAA in step 3 under Expand function).

ERROR MESSAGES

1. NOTFND means that the program could not be found under AAA on this tape 7.

2. LOW may accompany NOTFND and means that this program is on the tape but will not be loaded because it would clobber the Monitor.

3. ERX indicates that tape errors are encountered on tape 7 while searching for AAA.

RECOVERY - None, except for error 3; try again.

RESULT - AAA is loaded onto memory, and the location of the first instruction is saved for the Execute function.

FUNCTION - To Start the Execution of the Last Successfully Loaded Program

FORMAT - E.

ERROR MESSAGES - None.

RECOVERY - None.

RESULT - A LAC ACC takes place, followed by a JMP to the first instruction in the last loaded program (which would be the equivalent of step 4 under Expand)

FUNCTION - To Begin Instruction Execution Anywhere in Memory

FORMAT - BS, where: X is an octal address.

ERROR MESSAGES - None.

RECOVERY - None.

RESULT - A LAC ACC takes place, followed by a JMP to X.

PART TWO

This Part discusses suggestions for ways of:

1. performance of sundry tape maintenance procedures; and

2. organizing the systematic addition of final version programs to a program tape after testing and debugging on other tapes, by manual operations by the operator while using the command function of the Monitor and Monitor-Control.

Tape Maintenance

The reader should be familiar with the operations pertinent to the required Monitor functions, and that the manual switching of tape unit selections is done when required. Tape maintenance involves the manipulation of information on tape by deleting unwanted information, by inserting information between two groups of normally continuous information, and by combining deletion and insertion to merge two tapes and/or alter the sequence of information on tapes. There is usually an input tape to be changed called the old master, an input tape or paper tape which will be used to update the old master (called the updating tape), and a resultant output tape which represents the completed updating of the old master by the updating tape and is called the new master. The tape maintenance herein described is for program tapes only.

FUNCTION - To Delete a Program from Tape

FORMAT - Assume A is old master and B is new master:

1. F1 on A; bring in Monitor System.
2. F3 on B; write Monitor onto new master.
3. L AAA; Load first (next) program into memory from A. Do F4 key ins.
4. F4 on B; write AAA onto new master.
5. Continue steps 3 and 4 until the program to be deleted is encountered and bypass it for steps 3 and 4.
6. Continue steps 3 and 4 until the old master, minus the deletions, is copied onto the new master. By varying the order of calling programs in step 3, a new order of programs will exist on the new master.

RESULT - An updated new master.

FUNCTION - To Insert a New Program Onto Tape (Not Add Onto End)

FORMAT - Assume A is old master, B is new master, C represents an input tape or paper tape input via RIM Loader.

1. F1 on A; bring in Monitor System.
2. F3 on B; write Monitor onto new master.
3. L AAA; load first (next) program into memory from A. Do F4 key ins.
4. F4 on B; write AAA onto new master.
5. Continue steps 3 and 4 until point of insertion.
6. L XXX from C; or use RIM Loader for paper tape.
7. Continue at step 4 above until end of A or next insertion.

RESULT - A new master with updated insertions.

FUNCTION - To Merge/Update Program Tapes

FORMAT - This is any combination of insert and/or delete to create a new master from several input old master tapes.

RESULT - A new master with a different order of programs than its input old masters.

Comments

1. It may be advisable, in a system of programs, to place the same program on tape more than once to reduce loading time by reducing search time.

2. It is important that accurate records be kept of the programs on a tape, their starting and ending locations, the address of their first instruction, and their order on a tape. This information is important for F4 key ins and for facilitating tape maintenance procedures.

3. It may be advisable, if possible, that all programs be SETLOCed to the same location and have the same first instruction address, as a way of facilitating tape maintenance and the Expand (F4) function. If the records of programs on a tape are lost, the following function would help.

FUNCTION - To Log a Program Tape

FORMAT

1. F1, get fresh Monitor.
2. PJO, where J is the octal address of the tag SELF1.
3. WJOM, where M is an octal value for NOP, i.e., 740000.
4. L xyz, where xyz may be any three arbitrary characters.

RESULT 1 - The first (next) program will be loaded, followed by LODED NNN, where NNN is the actual name of the program on tape.

5. PhO, where h is the octal address of the tag ADDRES.
6. PfO, where f is the octal address of the tag STARTX.

RESULT 2 - The printout from step 5 is the octal address minus one of the start of the program. The printout from step 6 is the octal location of the first instruction. (If desired, a P 10 O would yield the last address of the program.)

7. Repeat steps 4 through 6 until the Monitor displays LOW.

RESULT 3 - The flexo (i.e., typeouts) is a log of the program on the program tape.

8. The contents of j as altered in step 3 must be restored to the value indicated from step 2.

FUNCTION - To Automatically Log a Program Tape

FORMAT

1. F1.

2. WkOm, where k is the address of the tag ITSELF and m is a NOP, i.e., 740000.

3. Write the following program into memory (30 is a convenient location) by referencing the definitions.

```
JMS     LOAD1
TCR
JMP     30
```

4. Manually halt the machine when LOW is printed.

5. F1.

RESULT - The actual program names in their order on tape will be continuously typed on the flexowriter as LODED NNN_i.

Organizing a System for Final Versions

In the following discussion, it is assumed that the reader is familiar with the Dump and Restore operations and has read the discussion of these in Section 1. A brand new program, say AAA, is loaded into memory for testing. The debugging session may uncover certain errors that are octally corrected or are corrected by DDT. At the end of the debugging session, the program may be dumped as AA1; at a future time, AA1 can be restored and debugging continue, yielding AA2 at the end of this next session. This process continues while a Dump tape grows with past temporary versions of program AAA. When the program is finally debugged, AAn may be restored and expanded onto a program tape as AAA.

Thus, one may have a library of checked out program tapes, and a library of programs undergoing checkout. The expansion of a final version onto a program tape may take place under any of the previously described tape maintenance procedures. The history of past debugging sessions may also be a handy reference for debugging.

PDP-5/PDP-1 ON-LINE DATA COLLECTION AND EDITING SYSTEM

by

Pablo Larrea

Princeton-Pennsylvania Accelerator
Princeton, New Jersey

Abstract On this system, a 12K PDP-5 accepts data from up to ten film-measuring machines. As it comes in, data is edited for a number of possible errors, e.g., label duplication, correct sequence of measurements not being followed, machine or interface malfunctions, etc. All correct data is collected in a table from which, at specific times (end of measuring a track or a point in all its views), it goes to a 16K PDP-1 which will perform a simple spacial reconstruction to determine whether certain criteria as to location of measurements (X and Y coordinates) has been met. When all checks have been made and/or event is complete, it will be written on magnetic tape for further processing.

After a particle reaches a certain energy level on the accelerator, it leaves the accelerator and enters the chamber, which is filled with liquid hydrogen at conditions near its boiling point. The particle in its travel then leaves a "wake" of bubbles in the hydrogen. Inside the chamber the particle is slowed down due to the denser medium and because of the short life span of many of the particles it decays. Its motion is governed by the influence of a magnetic field to which it is subjected; traveling as follows: almost uniform speed into the chamber; with the field causing it to turn in a circular motion (direction depending on particle's polarity) so the trajectory is in the form of a helix (corkscrew). There are three cameras outside the chamber which are triggered by the particle entering the chamber. These cameras then record the event. These cameras are in fixed locations, and with two views of the event it can be reconstructed in space. The third one is for improved accuracy. To correct for any movement of a camera, film shrinkage, etc., the surface of the glass has fiducials etched on it which become the controls. The film as produced then is measured on scanning machines which digitize the information (X, Y coordinates of points).

For programs which reconstruct the event to obtain energy and mass of the particles, several items have to be given. First an identification of the event; then a set of fiducials to establish scale reference, and then measurements on each of the tracks and of the identifiable points or

vertices. The tracks and points need some identification or label associated with them to distinguish each.

Past experience indicates that on the average 4 to 7% of events are rejected because of errors such as mislabeling of tracks or points, measuring too few points on a view etc.; also about 14% of events are rejected because of wrong measurement. This by itself does not seem significant but over ten frames have to be bypassed to get to an event, making remeasurement a slow process.

For this reason, the following on-line system has been designed. The system consists of up to ten measuring machines feeding data on an interrupt basis to a 16K PDP-5 (it has been upgraded by 4K memory due to program size). The PDP-5 checks the logic in the event, such as track and point labeling to avoid duplication, check validity of data, check X-Y encoders, check sequence of measuring (order: identification or parameters, then fiducials, then measurements), check completeness, etc. If any error is found, two actions are taken: 1) an error light is lit on a board in front of the operator (14 lights are available); and 2) a more complete diagnostic is printed on the typewriter which pinpoints the error even more (over 60 different errors indicated).

If the data is correct, a green light will go on in the panel indicating to the operator that everything is working properly. The operator also has control of the program through any of the ten commands that can be given. The ones envisaged at present are: delete last measurement, delete last view, delete all views of present track or point, delete event, event complete, save data. All of which are obvious in their meaning except for the last two. The event complete is needed since the program does not assume at any time that all data is in until it is told to do so; since the operator could notice an error and cancel some data. The save data is (in cases of more than one event) in one frame to save the part of data common (even if only fiducials) from having to be remeasured. Finally when event complete is pressed, the program will check the event for completeness meaning all data in, all tracks identified, event type agrees with that specified in identification, number of tracks from each vertex agrees with number that should come from it, etc. When all checking is finally successful, the event will be written on magnetic tape in the format required by the next program.

A description of the hardware and its use follows. All data starts from the scanning machines. Each machine has 16 wires connected to the PDP-5; twelve of which are for data transfer and the other four for the reply messages from the computer. The data sent from the machine consists of either 15 digits of identification (60 bits); or a measurement consisting of a 4-digit label, a 1-digit view number, 5-digit X-coordinate and 5-digit Y-coordinate; or a command. For the identification or the measurements 60 bits suffice; but it is necessary also to be able to identify one from the other, so for this reason a command buffer exists also. In this buffer when a command is to be executed, its number (1-10) is stored in the buffer; also for measurements, the buffer contains 11 and for identification 12. The machine contains: 72 bits of information, the first twelve giving an indication of the meaning of the rest. When one of the three buttons is pressed, the machine sets up a flag in a shift register corresponding to the machine number. This flag is connected to the program interrupt system. When the PDP-5 accepts the interrupt, it shifts the register until it makes coincidence, at which time the measuring machine becomes locked to the computer. Now the program will read the data from the machine twelve bits at a time by giving the same IOT six times. Each time it reads one, a pointer advances to the next 12-bit block. When the data has been read in, the first twelve bits are analyzed for contents and the program goes to the routine that handles the type of data indicated. After analyzing the data, a replay message is sent to the machine in the form of one of the 15 possible replies (14 errors and an O.K.) at which time the machine is released and the program goes back to interruptible.

If an error is detected, one of the 14 error lights goes on, plus the message is typed again in an interruptible mode so that in between typing characters it can accept data from machines, write on mag tape, etc.

A special set of IOT's was provided for the mag tape writing. The data going from the accumulator on a 12-bit block which can be broken down into either 2-6 bit characters (for alphabetic information) or into 3-4 bit characters (for numerical information). Also, either the whole twelve bits or only the first character can be used. With one more IOT, the parity of the writing is checked to insure that the writing was correct.

A link is also provided to connect to the PDP-1's high-speed data channel (131) on an interrupt basis. For this, two items of information need to be supplied (by use of two IOT's), which are

the word count and the starting address of the data in the PDP-5. When the PDP-1 acknowledges the signal, it reads the data in a data-break mode that is stealing some cycles from the PDP-5 to read from its memory. Replies from the PDP-1 are from an IOT that sets up an interrupt. When the interrupt is accepted, data will be read in (via three IOT's) from the 36 bits of PDP-1's accumulator and I/O; the last IOT releasing the PDP-1 from its I/O halt. This method has been selected since large amounts of data are transferred to the PDP-1 but only low amounts are transferred from it.

Thus the program can be considered as a delay loop for idle time from which the PDP-5 is interrupted to perform any of a number of tasks and to which it returns after performing its task.

A future expansion on the system will be the use of the PDP-1 to do a spacial reconstruction (probably modeled after Dr. Taft's program operating at Yale) on each vertex point and for each track as part of the acceptance test for each track.

The PDP-1 is now being used to check the measuring machines once a day by measuring a straight line and performing a least-squares fit of the points and checking for worst point, average error, and chi-squared distribution of the two least-significant digits to check encoders. This program has been written for the PDP-1 by Dick Zacher.

Although it is still too early to evaluate results, the following observations have been made. The total time taken by the PDP-5 to collect information and check it 90 times was less than 2 seconds. Machine time required for preparing data to be written on mag tape was less than 4 seconds. Mag tape writing time was under 10 seconds (incremental tape recorder at 300 cps). An event takes about 15 minutes to be measured; so based on this time, the PDP-5 is busy approximately 10% of the time with ten machines.

A VERSION OF LISP FOR THE PDP-6

by

William A. Martin

Massachusetts Institute of Technology
Cambridge, Massachusetts

Abstract LISP input-output has been extended to include a dataphone and the Type 640 Oscilloscope with character generator and light pen. A macro language for describing pictures has been embedded in LISP. This system makes possible the coupling of pictorial displays with problems conveniently programmed in LISP, as well as the study of recursive pictures. The choice of concepts for a picture language and tradeoffs involved in its implementation are discussed.

A SYSTEM FOR DISPLAY LANGUAGE CONSTRUCTION

The system provides a language macrosal for the generation of picture parts called objects. An object can be any combination of points, lines, and characters. An object is generated by calling the function macrosal [NAME; DESCRIPTION] described in the second section. The most common way to describe an object is to establish a set point. The set point established is utilizing PARAMETER, LOCY, and LOCX statements. The exact format of these statements is discussed below. An object description is terminated with a STOP statement. If NAME is T, the current description will be appended to the description of the last object generated. The first example, disp, generates a large object in this manner.

The user communicates with a display through a light pen. As the light pen sweeps across the screen, its trajectory can be used in many different ways. For example, it may be used to determine a point, a set of points, or a line. Or if a subpart of the display has been defined as an object, the trajectory may be simply interpreted as a pointer to this object or a point on the object. The LISP functions embedded in the display language facilitate acquiring the data needed to make these different levels of interpretation.

One problem in utilizing the light pen is to determine when it is near the screen and not just being moved into place. This is solved by using the width of the field of view of the pen as

measured by a tracking cross. This width decreases as the pen approaches the screen, and a center dot is displayed whenever the field of view is less than a specific limit. This pen distance is available to LISP.

The approach of the pen to an object is considered significant. Just how close the pen must come before being noticed is a program variable. Its most recent position within this distance is recorded; in addition, so are the last five such positions, each at least a prescribed distance from the preceding one. This represents a crude way of gradually forgetting the details of the past. It is also possible to get the current coordinates of the pen; obtain a list of all objects currently seen by the pen; to report when the pen sees an object with a name other than a given name; or to require an object to move with the pen.

The example function sketch is a LISP function using several of these features. The function uses a subroutine to display five different light buttons. If the light pen is held near one of these buttons, the tracking cross will be centered about the point where the pen is seen. The program interprets pointing at these buttons to mean:

1. Draw a line.
2. Move a line.
3. Delete a line.
4. Suppress the cross.
5. Return control to the Teletype.

To draw lines touch the first button; a new line will then be drawn whenever the pen leaves the screen and then returns. This process is terminated whenever the pen returns near one of the light buttons. Additional LISP functions could be written to expand sketch into a program similar to SKETCH PAD.

Often one wants to communicate to the display certain basic forms which are to be used in constructing larger units. These inputs might be a set of letters which are to be parsed into a sentence, or they might be a set of circuit elements or music symbols. In SKETCH PAD this is done by indicating the type of form and then moving the light pen so that the parameters of the form, the end points of a line segment for instance, can be abstracted from the trajectory. Alternatively, one could abstract both the type and the parameters from the trajectory. The example

program argus uses a method (developed by Teitelman) which enables the user to teach the machine to replace a single line by a known form. A line is a single movement (however complex) on the surface of the display without lifting the pen. The LISP data structure is convenient for storing properties of the forms to be recognized.

The parsing of very large displayed expressions, such as LISP S-expressions for example, can be difficult for people. Furthermore, there may be alternative parsings. People can be aided by intensifying, upon request, grammatical subexpressions or subobjects containing referenced segments or by providing additional displays meaningfully related to the first display. These might be rotated views of an object or shaded objects. Further development of the system is needed in this area.

It is the task of the programmer to organize a program and data base in such a way that the most needed inputs to a machine will have short representations and the most needed computations will be efficient. The combinatorial aspects are such that this must be done through a series of levels of concepts. An important point is that it is not possible to complete an entire level at a time. The most useful concepts at a given level only become clear with the exploration of higher levels. The exploration of higher levels without intermediate concepts is, however, almost impossibly tedious. It is important in an experimental situation to have a system where one can make changes to any desired depth and provide for the irregular growth and reorganization of the data base.

In the present display system, experimentation will probably indicate that new statements for macrosal are needed or that certain objects occur so often that more programming effort could well be spent in generating them efficiently. To provide for these possibilities macrosal has been programmed as a syntactic extension of the scope assembly language, sal. Sal is a LISP function which creates objects from lists of octal numbers. It is described in detail under Implementation. Provision has also been made for the addition of machine language subroutines which alter the objects as they are displayed. Furthermore, the system is organized so that no statements need be made about features of the display language which are not needed.

By embedding these display facilities in LISP, one makes available a wealth of mechanisms which have proved useful in the analysis and generation of language and in the development of systems which can be increased incrementally in complexity.

IMPLEMENTATION

The use of LISP in this system has two distinct disadvantages. First, it is not possible to interrupt the LISP system at any point in time and immediately employ its full power. It may be in the midst of garbage collection. Garbage collection with the current version of PDP-6 LISP requires a noticeable time. Second, the data types are too limited. It is not convenient to set up the type of list structure used in SKETCH PAD, but this can be approximated. A serious problem is the inability to set aside blocks of registers to contain display instructions and to store information about light pen actions.

To get around these problems, a fixed buffer of 2048 words has been set aside for description of the display. All communication between LISP and the display goes through this buffer. This buffer contains two kinds of data structures: display lists and headers. One header is associated with each display list, which is a list of half-word commands for the display. The headers build down from the top of the buffer and the display lists build up from the bottom.

During display an interrupt routine cycles through a dispatch table. Dispatches can occur to a pen track routine, a routine which displays the contents of the display buffer, a line drawing routine, and a routine which terminates the display.

The pen track routine displays a cross as was described earlier. In a crude effort to give the routine enough display time each cycle, it is called between display of each object in the display buffer. Use of the clock would be better. The format of the headers in the display buffer is as follows:

NAME (Pointer to an atom)	
-(Display list length	Pointer to start of display list - 1
Pen hit distance	Pen hit count
Subr address	A B C
Most recent Y	Most recent X
Y	X
Y	X
Y	X
Y	X
Y	X

A = ON-OFF bit
B = Tracking cross jump bit
C = Move with pen bit

Figure 1 Header Format

Each display list has a name which is kept as the first word of its header. When an object is referenced by a LISP function, the headers are searched for one with the name mentioned.

The interrupt program cycles through the headers. It picks up a pointer for a BLKO instruction from the right half of the second word of each header. This BLKO is terminated by a STOP instruction at the end of each display list. If the subr address is not 0, the subr at this address will be executed when the STOP is reached.

If the light pen is seen during display of some display list, control goes immediately to an interrupt program. Several conditional branchings can occur within this program. The interrupt program first redisplays the display list as a check against light pen noise. If the pen is seen a second time, the pen cross movable bit is checked. If this bit is a 1, the pen tracking cross is centered about the point where the light pen was seen and the rest of the display list is displayed. Otherwise, a check is made to see if the center of the pen cross is within a specified minimum distance of the point seen. If not, the display list is continued with the light pen reenabled. If the pen is close enough, the coordinates of the point seen are stored in the fifth word of the header. The last five words of the header contain the coordinates of points seen by the pen in the past. Each of these history points is at least a specified distance from the preceding one. This distance is in the left half of the third word of the header. When a new point is seen, it is added to the history points if it is far enough away from the one most recently stored or if there are none. The number of history points is kept in the right half of the third word of the header. This number can be set to zero by LISP. Whenever there are more than five history points, the oldest one is lost. After the point seen has been appropriately stored, a check is made to see if the object should move with the pen. If bit C is set, a pointer to the display list is transmitted to the pen track routine. The display list is then finished.

Display lists are put into the buffer by the sal language. If a line is to be drawn with the pen, a set point for the line is created with the sal language. Its display list is then incremented by the line drawing routine. This incrementing is terminated when the pen leaves the screen.

macrosal [X;Y]

Macrosal interprets its arguments and then calls sal on the result. X is the first argument for sal. Y is a description of a display list for sal. Y is a list of lists, each of which is a macro. The first word of each macro is an atom which has under the property MACROSAL a function of one argument. Macrosal gets this function and applies it to the remainder of the macro list. The result of this function is a list for sal beginning and ending in mode 1. This list is prefaced by two integers which give, respectively, the mode that the display will end up in and the mode in which it must begin, if it is to interpret this list correctly. Macrosal appends the successive macro expansions, using the prefacing integers to create the proper linking of modes.

The following macro word formats are in the system:

(PARAMETER pen enable scale intensity)
Normally the first word in a set point, this statement creates a parameter word. If pen enable, scale, or intensity is NIL, the corresponding field of the parameter word is not enabled.

(LOCY n)
Creates a nondisplaying Y-point word which sets the scope Y-coordinate to n. This is normally the second statement in a set point.

(LOCX n)
Creates a nondisplaying X-point word which sets the scope X-coordinate to n. This is normally the third and last statement in a set point.

(STOP)
Creates a parameter word with stop enabled. This is normally the last statement in a display list.

(ISTOP)

Creates a parameter word with stop enabled and a flag that the previous word has a breakout bit. The use of ISTOP is explained in the description of sal which follows.

(LOCYD n)

Creates a displaying Y-point word which sets the display Y-coordinate to n and displays a point.

(LOCXD n)

Creates a displaying X-point word which sets the display X-coordinate to n and displays a point.

(LINE X_1 Y_1 --- X_n Y_n)

Creates a sequence of nondisplaying line segments from vector words. The display starts at the last point displayed or set by LOCX, LOCY. Each increment has size $\Delta X = X_{i+1} - X_i$, $\Delta Y = Y_{i+1} - Y_i$.

(LINED X_1 Y_1 --- X_n Y_n)

Creates a line like LINE, but displays it.

(LONGLINE X_1 Y_1 X_2 Y_2)

Creates a nondisplaying vector continue word. A line will be drawn from the current display coordinates to the edge of the display. The slope of the line is $(Y_2-Y_1)/(X_2-X_1)$.

(LONGLINED X_1 Y_1 X_2 Y_2)

Creates a line like LONGLINE, but displays it.

(CHAR X_1 ----- X_n)

Creates a display of character made words which are the PNAME's of the atoms X_i. No spaces are inserted between the PNAME's.

sal [X;Y;Z]

X is the name of the display list to be created, or T. If X is T, this list is appended to the last one created. If the previous list ends in STOP (3000_8), the STOP is removed. If it ends in ISTOP (403000_8), it not only removes this but zeros the breakout bit (400000_8) in the previous half word. Y is a description of the list to be created. There are two forms for the elements of Y corresponding to two modes for the assembly function sal. The function is initially in mode 1. In mode 1 sal removes lists of atoms, two at a time, from Y. Each atom is a number or is bound to one on the dotted pair list Z. For each pair of lists sal forms one display half word instruction by shifting the numbers on the second list the number of places specified by the corresponding number on the first list. When it encounters NIL on Y, it goes into mode 2. In mode 2 sal takes numbers or non-numerical atoms one at a time from Y. Each number is a display half word. If a non-numerical atom is encountered, sal looks at the previous number to see if it puts the display into increment or character mode. If in character mode, sal assembles the PNAME of the atom as characters. If in increment mode, sal gets a list of full words off the atom's property list with the indicator SCHAR. It assembles these as increment mode half words. Consecutive non-numerical atoms are assembled together in the same mode. When sal finds a number, the breakout bit is set. If sal finds NIL, it returns to mode 1. Mode 1 is more flexible and mode 2 is more economical. NIL is not a legal object name.

MORE FUNCTIONS

In the descriptions below S stands for the name of an object with a set point. W stands for the name of an object with or without a set point.

sx [S]

Returns the current X-coordinate of S.

sy [S]

Returns the current Y-coordinate of S.

smv [S]	Makes object S follow the light pen whenever S sees it. Returns S.
sunmv [S]	Negates smv. Returns S.
sxyinc [S;X;Y;]	Increments the set point coordinates of S by X, Y. Returns S.
sclr [ ]	Clears the display buffer. Returns NIL.
sdlt [W]	Deletes object W from the display buffer and returns W.
ptrk [X;Y]	Starts the pen tracking cross at X, Y and returns NIL.
puntrk [ ]	Stops the display of the pen track cross. Returns NIL.
px [ ]	Returns the X-coordinate of the pen.
py [ ]	Returns the Y-coordinate of the pen.
ptchp [ ]	Returns NIL if the pen is far from the screen.
srn [ ]	Starts the display build up in the buffer. Display will continue until sunrn [ ] is called.
sunrn [ ]	Stops the display and returns NIL.
pldw [X;Y]	To use this command, use macrosal to set up a set point at the current pen position followed by a zero length relative line and terminated with ISTOP. Then call pldw. Increments of length at least X and scale Y will be added to this object until the pen leaves the screen. Returns NIL.
phclr [W]	Clears the pen approach history points of object W and returns W.
ph [W]	Returns a list of the pen approach history points for object W.
phcl [W;X]	Sets to X the minimum distance between pen approach history points for object W. X is an integer between 0 and 1024.

phc2 [X]	Sets to X the minimum distance between pen approach history points which is assumed for newly created objects.
phc2 [X]	Sets to X the maximum distance on the screen at which the pen can see an object. X is an integer between 0 and 40.
sint [W;Y]	Sets to Y the intensity of object W. Returns W. W is an integer between 0 and 7.
scopy [W1, W2]	Copies object W1 and names the copy W2. Returns W1.
plsh [ ]	Returns a list of all objects currently being seen by the pen.
sscl [W;Y]	Sets the scale of object W to Y. Returns W. Y is an integer between 0 and 3.
plhw [W]	Returns the name of the first object other than W seen by the pen.
ptchw [ ]	Waits until the pen is near the screen and then returns NIL.
puntchw [ ]	Waits until the pen is not near the screen and then returns NIL.
pjp [W]	If the tracking cross is not following the pen and pjp [W] has been executed, then when the pen sees object W, the tracking cross will be centered about the point of W seen. Returns W.
punjp [W]	Negates pjp and returns NIL.
poh [ ]	Returns the name of the most recent object to be seen by the pen or NIL if none have been seen since pohc was executed.
pohc [ ]	Sets the last object seen by the pen to NIL. Returns NIL.
sline [S]	Returns a list of the coordinates of the end points of the line segments which make up object S.

ssubr [W;Y]	Causes the subroutine beginning at location Y to be executed each time object W is displayed.
sloc [W]	Intensifies a subexpression of object W which is indicated by the light pen. Subexpressions are marked in the object by pseudo parentheses which are not displayed. A pseudo left parenthesis is indicated by the parameter word 600001_8 and a pseudo right parenthesis by the parameter word 600002_8.
sadd [W]	Counts the number of pseudo left parentheses to the intensified subpart of W.

A REPORT ON THE IMPLEMENTATION OF THE KEYDATA SYSTEM

by

Charles W. Adams

C. W. Adams Associates, Inc.
Bedford, Massachusetts

Abstract The KEYDATA System utilizes a PDP-6 (with 48K core, a million-word drum, a 33-million-character disc file, and a full-duplex Type 630 Interface) as the control facility for on-line, real-time data processing services offered to business and engineering users through KEYDATA Stations (teletypewriters) located on their premises. Both packaged services, such as the preparation of invoices and the entry and correction of FORTRAN programs and data, operate through the KOP-3 executive routine. To prepare invoices, for example, the operator keys the quantity and stock number and the computer provides the description, prices, extensions, etc. on an invoice form in the users' teletypewriter as well as inventory control and credit checking on line, and related statistical and accounting reports off line.

In the KOP-3 system, the drum serves as the primary storage for all data and programs (except KOP-3), and also acts as the file directory, output buffer, and repository for the most active file records. Working storage is provided through automatic allocation of core memory in 32-word pages.

KOP-3 is a collection of programs written for the PDP-6 computer, which comprises the central facility of the KEYDATA system. The name is an acronym for KEYDATA On-Line Processor, the number indicating that it is third in a series, KOP-1 and KOP-2 having been developed and implemented on a PDP-4 computer in 1963. KOP-3 serves as the executive routine, monitor, and scheduler, and also provides an interpretive processor in the foreground and batch portions of the KEYDATA system. The overall system divides into three essentially independent parts:

Foreground, in which messages received from KDS keyboards (or other data sources) are processed, usually with reference to one or more data files, resulting in the updating of control totals and file records as appropriate and in the generation of output messages which are transmitted to KDS printers (or other output media) and journal items which are recorded on magnetic tape for later processing in batch mode.

Background, in which the central processor is made available (whenever it is free from its primary responsibilities to the foreground mode) for use in any

of a variety of computation tasks, such as FORTRAN compilation or engineering computation, on a time-sharing basis in which a moderate number of users are scheduled onto the system for periods ranging from a small fraction of a second to a few seconds at a time according to an established scheduling algorithm.

Batch, in which during runs made as the system is taken out of service at night and as it is put back in service in the morning, the random access files are tidied up and recorded on tape for safekeeping, the day's activity records (journal items) are sorted into proper sequence, and sequential files on magnetic tapes are updated and periodic status and activity reports required by users are produced.

INPUT AND OUTPUT

All of the communication between the KEYDATA system and the outside world is handled through the foreground mode. All data to be processed in background and batch mode is collected through the foreground mode into the random-access file (comprising magnetic drum and magnetic disc units) and/or the sequential-access file (comprising industry-compatible magnetic tapes and small block-addressable DECtapes). All output to KDS printers (as well as line printers, etc.) is likewise handled through the foreground mode. Thus, the processing of data for users of KEYDATA's packaged management information services is performed partly in foreground and partly in batch mode, the responsive on-line real-time processing being performed in foreground mode with information needed to produce periodic activity and status reports being recorded at that time in the form of journal items on magnetic tapes which are later processed in batch mode to actually produce the reports.

In foreground mode the basic operations are the receiving of input messages, the processing of them to update totals and file records and generate output and journal items, the transmitting of output messages, and the recording of journal items. Input from keyboards is received character by character and collected into messages of variable length by the "receiver" routine which scans all character input buffers at least once every fiftieth of a second. Data from higher speed input devices is assembled a word at a time through the priority interrupt system

of the PDP-6. Once a complete message has been collected, that fact is noted in a list stored in core memory called the "end-of-message queue." Messages are processed one at a time, first come first served, according to prescribed procedures which may be unique to one user or the same for many different ones, as required.

KOP LANGUAGE

The processing procedures are described in a programming language known as "KOP language." KOP language is designed both to facilitate the description of the procedures likely to be required in KEYDATA message processing and to make the description of these procedures as concise as possible, thereby minimizing the requirement for storage and for machine time used in interpretation.

KOP language is to some degree a problem-oriented language, yet it retains many of the characteristics of a machine language. Aside from the fact that the arithmetic operations in KOP language are performed on 36-bit binary operands, it bears no resemblance and has little relation to the logical design and instruction repertoire of the PDP-6 computer on which it is implemented--it could equally well be implemented on virtually any computer, though some minor redesign would perhaps be desirable if a different word length were involved.

KOP language differs from conventional program language in three important ways:

1. It is a pure procedural language in which instructions may not modify themselves (with many users being processed by the same programs, out of phase with one another; this is of course mandatory).

2. It deals with nonhomogeneous storage areas, that is with words and fields of a variety of lengths, structures, and longevity.

3. It incorporates several macro-operations which permit performance of a variety of functions in relatively few instructions. To distinguish programs written in KOP language from conventional programs, they are frequently referred to as "logic tables."

Programs written in KOP language are executed interpretively by the KOP-3 system. In other words, the logic tables are stored in the computer in their original format as 24-bit words and are translated into the appropriate sequence of PDP-6 instructions each time they are executed, rather than being translated once and for all as would be the case if KOP-3 processing were accomplished by a compiler instead of an interpreter.

The principal reason for interpreting rather than compiling logic tables is to save storage capacity at the expense of machine time. Secondly, the need for relocation and protection of dynamically allocated storage, both core and file, makes the use of interpretive rather than compiling techniques relatively more attractive and less wasteful of machine time than would be the case if dynamic allocation were not required.

While as many as several hundred different KEYDATA Stations may be sending messages into and receiving messages from the central facility at the same time, the processing of messages within the system is done one at a time, with the KOP-3 system keeping track of what information belongs to what user and guaranteeing that each message for each KEYDATA Station is processed using the procedures and data appropriate to that particular station.

Thus, when a programmer sets out to write a logic table to define a particular procedure, he need not concern himself with the fact that many messages are being processed from many stations concurrently. He may, with absolute impunity, limit his thinking to dealing with the messages expected to be received from a single KEYDATA Station, one at a time, with no regard whatever to the time-shared environment in which the procedure he describes will be executed by the KOP-3 system.

A CHALK RIVER PDP-5 PULSE-HEIGHT ANALYZER

by

Dallas C. Santry

Chalk River Nuclear Laboratories
Chalk River, Ontario, Canada

Abstract Hardware has been added to a PDP-5 computer at Chalk River which enables it to operate simultaneously as two multichannel pulse-height analyzers. Data obtained from scintillation spectrometers and semiconductor detectors are analyzed in an analog-to-digital converter, and the pulse-height spectra obtained are stored in memory. Programming is used to obtain complete data reduction, i.e., process the stored data by doing background subtraction, spectrum stripping, integrations on any portions of the spectra and corrections for radioactive growth and decay. External equipment such as timers, automatic sample changers, and choice of detector are also under program control. Operating experience gained with this integrated system will be discussed.

The use of computers to perform pulse-height analysis is now becoming widespread. The first step in this direction began about eight years ago when pulse-height analyzers which used computer-type memory for data storage became available. These were of 100 address or channel storage with a capacity of 2^{16} per channel. Later these were expanded to 256 and 400 channels. The need has since developed for larger data storage (1000 and even 4000 channels). With the production of relatively low-priced computers, the conversion to computer pulse-height analyzers became economically attractive. In addition, the ability of the computer to perform immediate data reduction was an added selling point.

In this paper a prototype pulse-height analyzer that was built at the Chalk River Nuclear Laboratories using a PDP-5 computer is described. The necessary hardware was added using standard solid-state instrumentation techniques to produce a relatively simple but extremely reliable system. Although the unit will perform most forms of pulse-height analysis, this description is limited to the identification and measurement of gamma rays emitted by radioactive materials. Figure 1 shows an overall view of the complete system. On the left is a lead castle which houses the detectors. Fitted to the castle is an automatic sample changer which stores the samples outside the castle and sends them in one at a time. The extra storage bay on the PDP-5 contains the necessary units for pulse-height analysis. A standard ASR Teletype unit is used for data output. Figure 2 shows a block diagram of the system with the associated hardware.

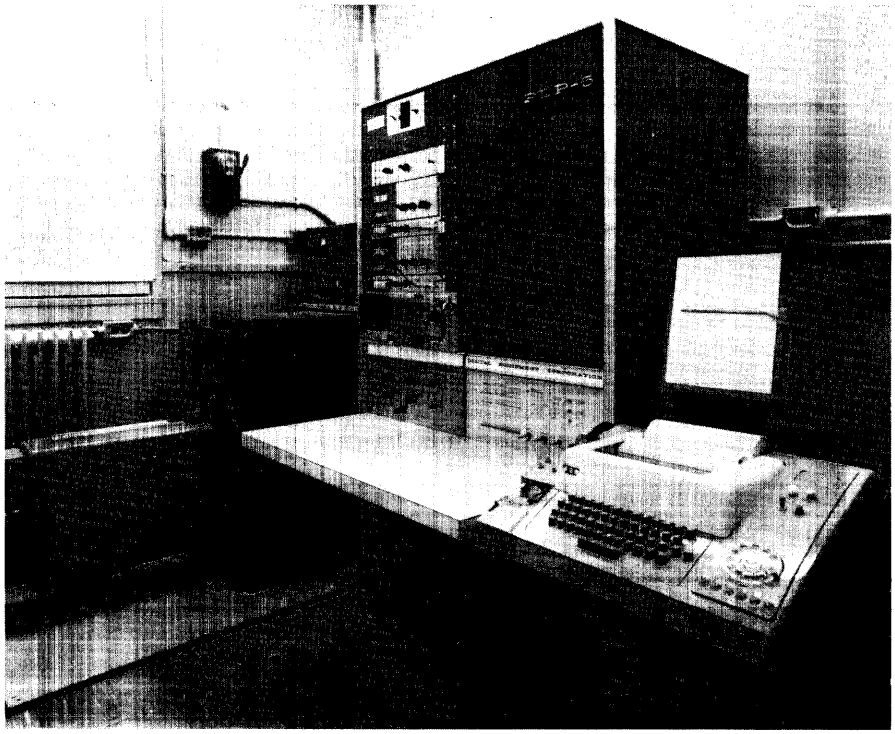


Figure 1 PDP-5 Pulse-Height Analyzer Gamma Ray Spectrometer

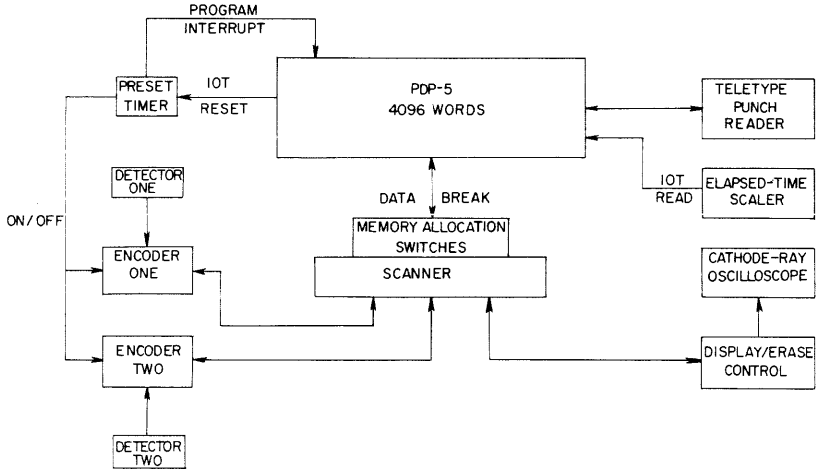


Figure 2 PDP-5 Pulse-Height Analyzer

DETECTORS

The detectors were either a pair of sodium iodide crystals or a lithium-drifted, germanium, solid-state detector. These detectors give pulses the height, which is proportional to the energy of the irradiation which they are measuring.

ENCODERS

An analog-to-digital converter is used to analyze the pulses from the detectors and produce a digital output which is acceptable to the computer. For this system a 1024-channel, solid-state, pulse-height encoder is used. The input pulse is stretched by charging a memory capacitor to the peak amplitude of the signal. An amplitude-to-time conversion is executed by discharging the memory capacitor with a constant current. The time is measured and converted to digital form by counting the cycles of a 5 mc/sec crystal-controlled oscillator during the discharge. The counting of the oscillator cycles is performed in a register within the encoder, which presents a binary number to the computer for storage or processing. A status line is set when the encoding process is complete. This is the flag that instructs the computer to store the word. The register must be reset from the computer when storage is complete. The encoder remains dead to incoming signals until this reset pulse is received. The dead time is recorded as % loss on a panel meter. In operation, the state of the encoder is sampled at 0.1-second intervals by a 10 c/sec clock; and if the encoder is live, a count is stored in channel one. If the encoder is dead, the clock waits until the encoder becomes live and a count is stored in channel two.

$$\% \text{ counting loss} = \frac{\text{counts in channel two}}{\text{sum of counts in channel one plus two}} \times 100$$

The 10 c/sec clock pulses are derived from a tuning fork stabilized oscillator. The encoder presents a binary number to the computer, and its corresponding address in core memory is incremented by one each time that address is given. A section of the computer's memory is, therefore, used to store the number of times a pulse of a certain height has been analyzed. The resultant data constitutes a spectrum which is characteristic of the nuclide producing it. Figure 3 shows typical spectra from a sodium iodide crystal and a solid-state detector. The position along the x-axis gives the energy of emitted gamma rays, and the area under the peak gives a measure of the intensity or counting rate of the gamma rays.

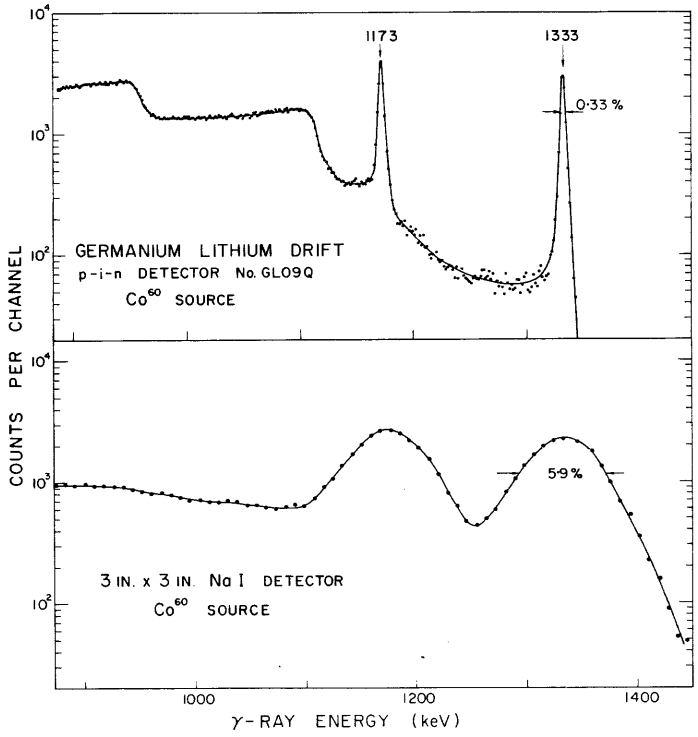


Figure 3 Sodium Iodide Crystal Spectra

SCANNER

A scanning unit (see Figure 4) is used to enable the computer to service more than one encoder and a display unit. When the encoder presents an address to the computer, it makes a break request. The computer completes execution of the instruction in progress and then enters the data break mode. This permits the transfer of data directly between the encoder and core memory, via the memory buffer register. This occurs as follows: The address is placed in the memory address register, the contents of this address in core memory are read into the memory buffer register, an increment request adds one to the contents of the memory buffer, and the new contents of the memory buffer are written back into core memory at the original address. The computer signals and the address is accepted. The scanner checks the memory buffer and if an overflow had occurred (i.e., gone over 4096_{10} counts), an increment address is requested. The original address plus one is set up and during another data break mode the high-order part of a double-precision number is incremented. The address accepted is given and the computer resets the register in the encoder and permits it to resume pulse-height analysis. With the exception of the increment memory address, which is used on data overflow to go into double-precision data storage, the input-output control elements already exist in the PDP-5.

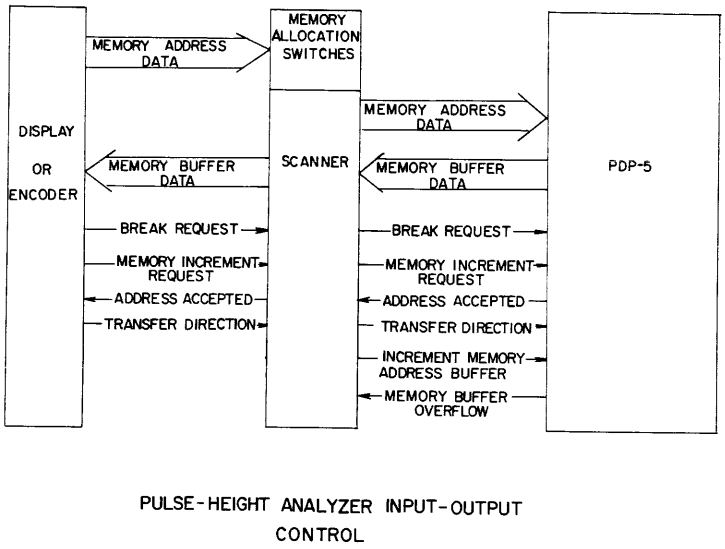


Figure 4 Pulse-Height Analyzer Input-Output Control

A set of memory allocation switches are an integral part of the scanner. These provide the most significant bits to the address supplied by the encoder and determine which block of memory will be used to store data from each encoder.

DISPLAY

The scanner also services a display unit in which data or spectra stored in memory can be viewed on a cathode-ray oscilloscope. An x-axis scaler in the display provides an address to the computer, and again through a data break request the contents of this memory location are read into a y-axis scaler in the display unit. The contents of each of these scalers are separately fed into a digital-to-analog converter and the output is displayed on a 5-in. diameter Tektronix scope. The resultant linear display represents total counts as a function of channel number similar to the spectra shown in Figure 3. Since this unit operates on double-precision data storage, two addresses are assigned to each channel, then two computer cycles are required per data point displayed. Associated with the display unit is a data erase button which permits the section of memory viewed on the scope to be set to zero manually. The PDP-5 operates on a read cycle and a write cycle. Data at a given address in memory becomes erased after it is read into the memory buffer. Normally, in the write cycle the data in the memory buffer is written back into core memory. Therefore, to erase a given location, it is only necessary to inhibit the write cycle.

TIMER

A Sodeco impulse register, which receives time pulses from an encoder clock, is used as a preset timer. It turns the encoders on when reset, and off when it goes to zero. The preset time can be reset manually or under program control. Also, when the timer goes to zero, it sets a flag and can cause a program interrupt. Figure 5 shows the preset timer, a pulse generator, and the encoders.

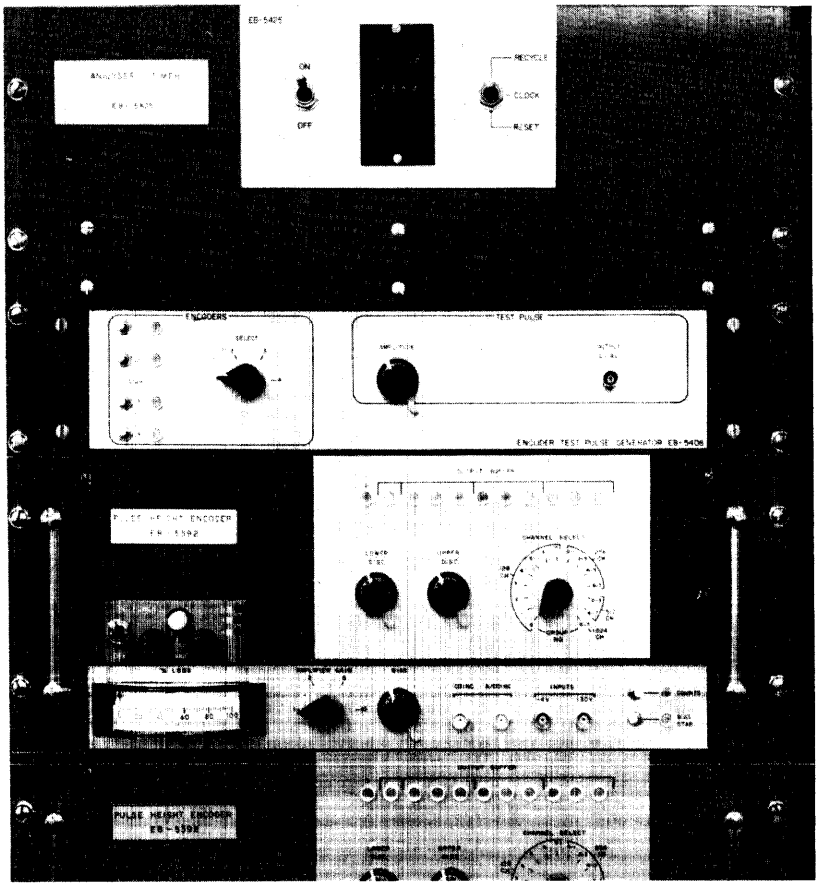


Figure 5 Preset Timer, Pulse Generator and 1024-Channel Encoders

The pulse generator is useful in setting up the pulse-height analyzer system. The address corresponding to the pulse analyzed from the pulse generator can be read directly from the lights of the output buffer on the encoders as well as from the oscilloscope display. A cut amplifier exists at the input to the encoder which enables bias to be applied, then gain can be added

after the bias. A single-channel analyzer has also been incorporated in the encoder. Analyzed pulses below a given "energy" can be cut out using the lower discriminator, and pulses above a given "energy" can be cut out using the upper discriminator. Figure 6 shows the cathode-ray oscilloscope and display unit.

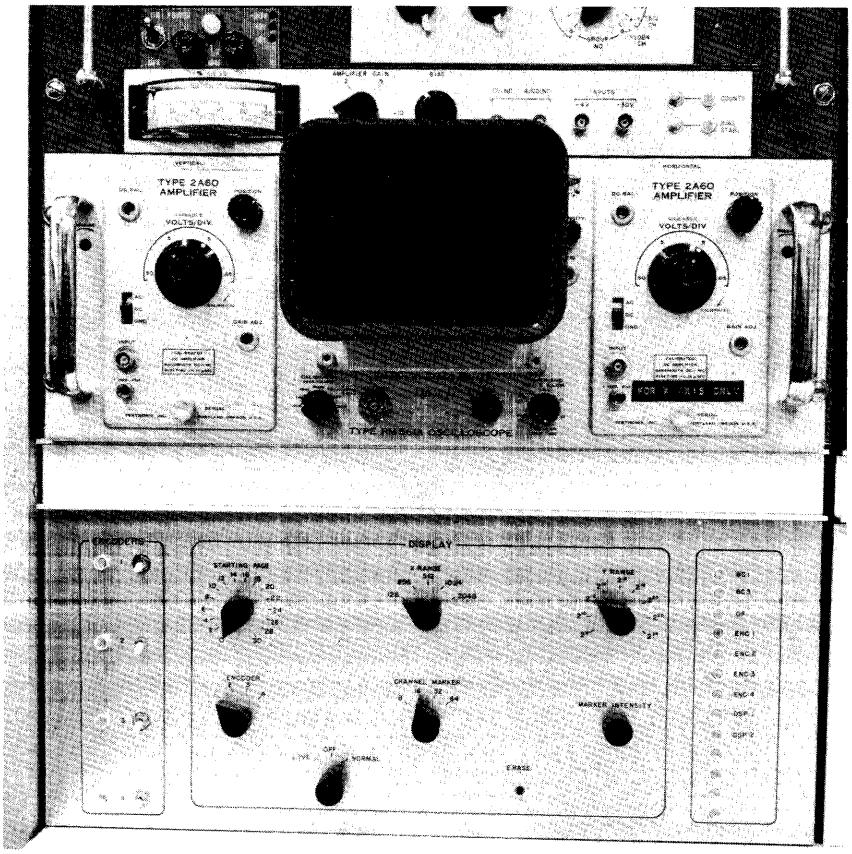


Figure 6 Oscilloscope Display and Display-Ease Control

One feature of this system should be emphasized. Without having any pulse-height analysis program in memory, it is possible to erase a data storage area, collect data for a preset time, and have a display of data while and after accumulating it. That is, pulse-height analysis can be performed by manually setting a few switches, as is the case with any conventional, non-computer pulse-height analyzer. The nature of the data break input requires the computer to be in a run mode which may be a simple (NOP, JMP .-1) loop or an execution of some irrelevant but useful program (a crude form of time-sharing).

One other piece of hardware, which is very desirable, is an external "time-of-day" scaler, which can be set to zero at any appropriate time and which then continues to count in seconds. On an IOT instruction, the computer reads this scaler and stores an elapsed time. The scaling of time outside the computer has the advantage that the clock need not stop when the computer stops.

A more advanced and sophisticated stage of pulse-height analysis with data reduction does involve programming, and one which was designed for automatic data collection is described below. The program was written as a series of subroutines. Other users with different requirements may easily incorporate only those subroutines which will yield the features of pulse-height analysis which they require. The comprehensive program is under keyboard control. The normal sequence of operation is first to type an I and enter an initialization routine. Questions are typed out which are to be answered by the operator, providing limits and format for the data output (Figure 7).

```
DATE                    210565
ZERO TIME               600
DURATION OF COUNT       100
ENCODER                 1-2,
SPECTRUM                YES,
HIGH                    100
LOW                     0
INTEGRATE               5
HIGH                    125
LOW                     0

HIGH                    100
LOW                     90

HIGH                    80
LOW                     60

HIGH                    50
LOW                     40

HIGH                    40
LOW                     20


PRINT                   YES,
PUNCH                   NO,
CHANGER                 YES,
SAMPLES                 3
NICKEL 65, .075093,
SODIUM 24, .0128712,
BACKGROUND, .,
```

Figure 7 Pulse-Height Analysis Data Initialization Format

The name of each sample is listed together with its radioactive decay constant. By typing B for begin, the data storage areas in memory are erased, all flags are cleared, the interrupt is turned on, the timer is reset to its preset value, and counting begins. When the timer goes to zero, a program interrupt is initiated. The interrupt routine first checks the cause of the interrupt. If it was not the timer, the program returns to a simple wait loop. If it was the timer, the elapsed time scaler is read to establish the time of day at which the data was accumulated. An instruction is given to activate the sample changer and the printout of data begins. A typical output is shown in Figure 8.

```
DATE                  0210565
SAMPLES               NICKEL 65
DURATION OF COUNT      0000100
ELAPSED TIME           0002145
ZERO TIME              0000600
DECAY TIME             0001545
EXPONENTIAL           001.1230183
LOSSES                0378423
ENCODER               1-2
SPECTRUM

 0000962  0000038

000  0000000 001  0000000 002  0000145 003  0001784 004  0001424
005  0001214 006  0001170 007  0001099 008  0001121 009  0000980
010  0001104 011  0001120 012  0001052 013  0001246 014  0001332
015  0001368 016  0001360 017  0001417 018  0001320 019  0001376
020  0001287 021  0001249 022  0001207 023  0001191 024  0001227
025  0001190 026  0001154 027  0001016 028  0000738 029  0000600
030  0000573 031  0000520 032  0000505 033  0000489 034  0000485
035  0000432 036  0000423 037  0000469 038  0000504 039  0000916
040  0001983 041  0003998 042  0006161 043  0007787 044  0006525
045  0004467 046  0002238 047  0000909 048  0000347 049  0000256
050  0000205 051  0000210 052  0000215 053  0000195 054  0000221
055  0000220 056  0000223 057  0000226 058  0000215 059  0000213
060  0000245 061  0000279 062  0000245 063  0000246 064  0000225
065  0000249 066  0000207 067  0000234 068  0000250 069  0000222
070  0000223 071  0000215 072  0000236 073  0000223 074  0000226
075  0000241 076  0000238 077  0000249 078  0000232 079  0000248
080  0000249 081  0000216 082  0000230 083  0000247 084  0000258
085  0000225 086  0000216 087  0000194 088  0000199 089  0000191
090  0000162 091  0000131 092  0000136 093  0000133 094  0000099
095  0000089 096  0000093 097  0000095 098  0000095 099  0000094
100  0000108

INTEGRATE
 125 000 0091128  0094576  0106210
 100 090 0001235  0001281  0001438
 080 060 0004982  0005170  0005806
 050 040 0034876  0036195  0040647
 040 020 0018158  0018845  0021163
```

Figure 8 Data Output

The two series of numbers after SPECTRUM are the clock pulses which were stored in the live-time and dead-time channels.

Total counting time is (962 + 38) ÷ 10 seconds.

Losses are 38 ÷ (962 + 38) = 0.038 or 3.8%.

INTEGRATE first lists the sum of counts between the specified channels, then lists this sum corrected for losses, and finally lists the sum corrected for losses and radioactive decay. Following the printout, the program will erase the stored data, clear flags, reset times, etc.,

and start counting the activity in the next sample. After all samples have been counted, the program can be set to stop or recycle the series of samples.

The pulse-height analysis program was also designed to read paper tape from conventional pulse-height analyzers in which spectra are stored in a binary-coded-decimal format. Similarly, spectra in memory can be punched in binary-coded-decimal format on paper tape for plotting on existing automatic plotters or for more elaborate processing either in a larger computer or in the PDP-5 using a FORTRAN program. Spectra stored in memory can be normalized, that is, multiplied by any factor. The normalized spectrum may then be subtracted from a multi-component spectrum, and the difference spectrum may be displayed or printed out (spectrum stripping).

The comprehensive pulse-height analysis program occupies addresses 0-3777 and 6000-6777. Spectra storage is at 4000-5777 and will provide space for four 128 channels, two 256 channels, or one 512 channel. To operate simultaneously as two 1000-channel, pulse-height analyzers with a capacity of 2^{24} counts per channel, it is necessary to have a very small program in memory which can be used to punch the spectra out on paper tape. Later this tape could be read in and processed in sections by the main program. Normally, 7000-7777 is used to store a very useful Service and Debugging Program which has been written by Mr. A.D. Hause of Bell Telephone Laboratories (DECUS No. 5-1 and 5-2). Besides containing a binary loader, the program is useful when applied to programming or data processing. It permits movement of data either into or out of memory via the keyboard, the movement of data from one location in memory to another, the erasing of data in sections of memory, and the punching on paper tape in RIM or binary format, the contents of specified sections of memory.

Although the pulse-height analyzer which this paper describes was designed to meet a personal requirement, it does contain standard features desirable in any computer pulse-height analyzer of this size. Because of the existing useful input-output control elements of the PDP-5, hardware rather than programming was used to execute the pulse-height analysis, the display, and timing. This constitutes a considerable saving in memory space which can then be used either for more data storage or for programming to obtain further data reduction.

The PDP-5 has been operating for 16 months (2900 clock hours) and has never required servicing. The auxiliary pulse-height analysis units have been operating for eight months, and these also have required no servicing. The only weak point of the system appears to be the ASR Teletype which does require frequent servicing. The acquisition of a high-speed, paper-tape reader may partially solve this problem by reducing the necessary operating time of the Teletype.

Although the unit was a prototype, it is adequate, and there are no plans for alterations or modifications.

A DESCRIPTION OF THE PEPR SYSTEM*

by

David Friesen

Laboratory for Nuclear Science
Massachusetts Institute of Technology
Cambridge, Massachusetts

Abstract A system called PEPR (for Precision Encoding and Pattern Recognition) is being constructed at the Laboratory for Nuclear Science of M.I.T. The system is designed to automatically scan and measure photographs of bubble chamber "events." The PEPR system consists of a special-purpose digital-analog film scanning device capable of high-precision measurements, connected on line to a PDP-1 computer. The scanning of film is directed by programs running in the PDP-1. A general description of the scanning hardware is given, and the current state of program development is described.

THE GENESIS OF PEPR

Few areas of science are as dependent on modern high-speed computing power as the field of high-energy physics. In 1964 over 40,000 hours of computing time were devoted to the handling and analysis of experimental data. Trends indicate that the computing load may double within the next two or three years. Much of this data is transcribed from the many millions of bubble chamber and spark chamber photographs taken each year.

Therein is the genesis of PEPR. The data taken from the photographs is the result of very precise measurements, needing many people, much time, and expensive measuring equipment. Scanning technicians need a $50,000-to-$100,000 measuring machine and 10-to-20 minutes to measure each frame of bubble chamber film, and this must be done for millions of frames per year before the data processing can really begin. This bottleneck has, of course, generated many efforts to automate the measuring process (such as the SPASS project for spark chambers under Professor Deutsch at M.I.T.). This is a report on one such project--the Precision Encoder and Pattern Recognizer (PEPR) being developed for bubble chamber film measuring.

---

*The work described in this paper has been supported by the United States Atomic Energy Commission.

THE PEPR SCANNING AND MEASURING APPARATUS

To use a computer in the scanning and measuring of bubble chamber photographs, the information on photographs must be digitized. One method of accomplishing this would be to represent the photograph as a 2-dimensional array, with each element of the array being 1 or 0, according to whether the film was light or dark at the point represented by the array indices/coordinates. Using suitably fine coordinates, an accurate representation of the photograph could be obtained. With such a representation, however, there are formidable programming difficulties in "recognizing" the patterns of basic interest in a bubble chamber photograph. The patterns of interest are "tracks"--essentially lines or curves--and even if these tracks were smooth and continuous, the recognition problem in a point-coded representation would not be easy. The pattern on the film is more complicated, however, because the tracks are in fact a string of bubbles, often separated by spaces equal to or greater than their own diameters.

The approach taken in the PEPR scanning device is to utilize the spatial arrangement of the bubbles along the tracks in the process of digitizing the information. The basic concept in the PEPR device is that a line segment provides a good filter for recognizing a line or track. An illuminated line segment is generated on a high-precision cathode ray tube by defocusing a spot into a short line. This line segment is swept across the film, and information is obtained by placing a photomultiplier behind the film. Two advantages result from using a line segment in the scanning device. The first is that the line segment, having extension in one dimension, automatically averages over many bubbles along a track. Second, the form of the light pulse seen from a track depends upon the relative orientation of the illuminated line segment and the track. Additional circuitry is used to filter the output from the photomultiplier so that only information about tracks at nearly the same orientation of the line segment are "seen" by the scanning device. The net result is that the digital representation of the film consists of locations and orientations of sections of tracks, rather than point locations of bubbles.

The hardware of the PEPR system consists of a digital computer (a PDP-1), a special-purpose digital computer called a controller, and the electronics and optics used to generate, move, and detect the illuminated line segment. The overall configuration of the PEPR system is shown in Figure 1.

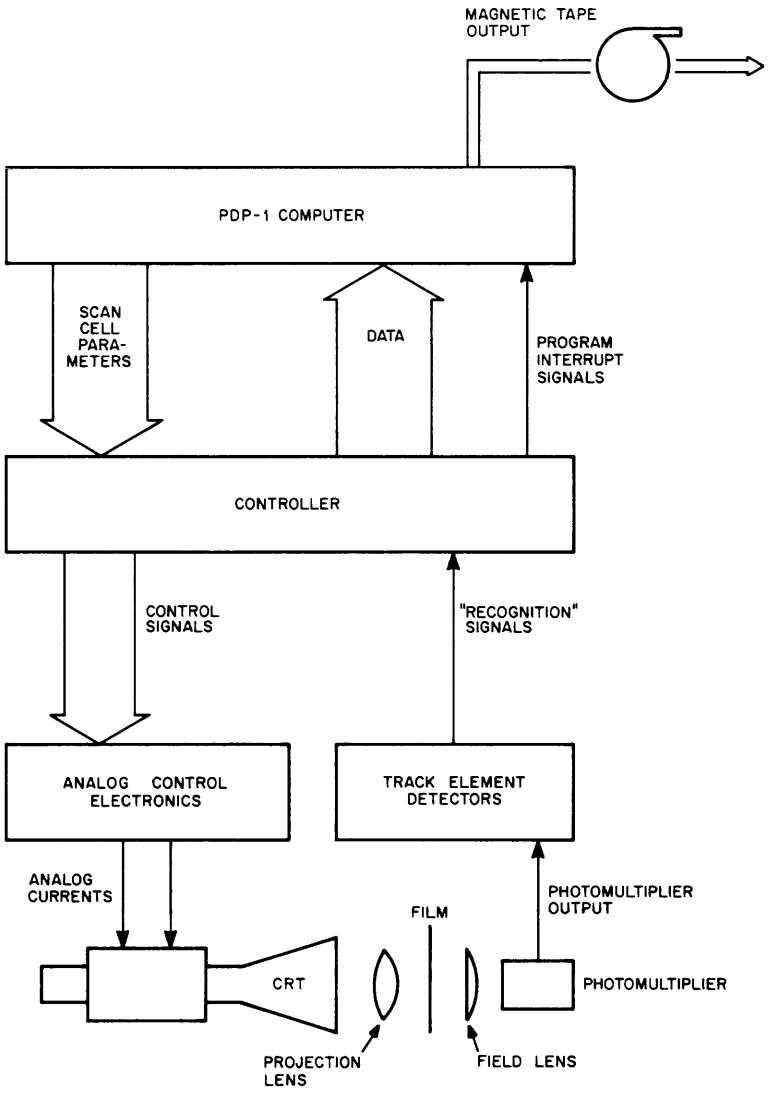


Figure 1 PEPR System Hardware Configuration

Since a complete digitization of a single frame of bubble chamber film would require excessive amounts of memory, the film itself is used as the primary "memory" for the image, with the computer memory storing only that amount of information needed for its purposes. The computer is programmed to retrieve selective information from the film and to use that information for pattern recognition or measurement. The computer communicates directly with the controller, setting parameters in the controller to govern the film scanning, and then instructing the controller to execute the film scanning. The controller performs the scanning as directed, stores any data generated, and then signals the computer by means of a program interrupt. The computer can then transfer this information to the computer memory.

The parameters which govern the scanning define a "scan cell." The PEPR precision scanning CRT has 4096-by-4096 addressable locations which are used to define the center of a scan cell. The line segment can be swept through a small distance about this point. Additional parameters specify the length of the illuminated line segment and the initial and final angles of orientation of the segment. Two further parameters specify the sweep rate and the maximum clock count (discussed below) allowance. The parameters are summarized in Table 1.

TABLE 1 PARAMETERS DEFINING SCAN CELL AND SCANNING PATTERN

Parameter	Allowable Range of Settings
Location of center of scan cell	$0 \leq x, y < 4096$
Initial, final orientation angles of line segment	$0° \leq \phi < 180°$
Sweep rate and maximum clock count	Settings allowing combinations of: Cell Size (mm) / Least Count (μ) 2.0 / 20 0.4 / 2 0.2 / 2 0.2 / 1
Length of line segment	2 mm, 1 mm, 1/2 mm, 1/4 mm, spot

When the controller starts a sweep, the line segment is oriented at its initial angle, a clock is started in the controller, and the line segment begins a sweep. The direction of the sweep is either horizontal or vertical, depending on the angular orientation of the line segment. The orientation conventions and sweep direction are explained in Figure 2. Whenever the filtering circuits (called TED's for Track Element Detector) indicate that the photomultiplier has seen a "hit" on a track, the current clock reading is saved in a buffer in the controller. When the clock reaches its maximum allowed count, the sweep is stopped. If no hits have been detected in the sweep, the orientation angle of the line segment is incremented by one degree, the clock reset, and the sweep and clock restarted. If data was present at the end of a sweep, the sweep is not restarted, and the computer is signaled by a program interrupt. The computer may then transfer the data to its own memory and restart the sweeping. After scanning has taken place at the final angle, the controller also signals the computer by a program interrupt.

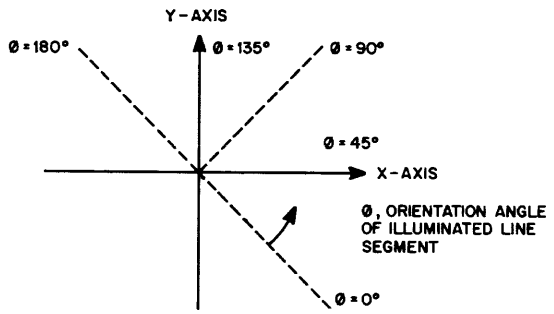


Figure 2a Definition of Angular Orientation of Illuminated Line Segment

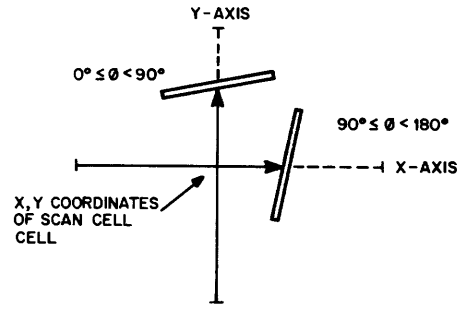


Figure 2b Scanning Pattern for Different Line Segment Orientations

The clock count allows an interpolation to be made between scan cell centers, making possible measurements finer than one part in 4096. There are several combinations of maximum clock count and sweep speeds, allowing for least counts equivalent to 20, 2 or 1μ on the film. The distance on the film swept by the line depends on the choice of clock and rate parameters and varies from 0.2 mm to 2.0 mm.

The scanning hardware is quite rapid, with the basic sweep time at one orientation being either 10 or 20 μsec, depending on the maximum clock count allowed. There is a setup time between sweeps of about 2 μsec. Thus, not counting time needed to transmit parameters to the controller and retrieve data from it, a scan cell 2 mm by 2 mm on the film could be scanned at all angles in slightly over 2 msec.

PEPR AS A MEASURING MACHINE

Measurements made on bubble chamber film serve as the starting point for a long chain of calculations. The measurements must be accurate to tolerances of less than 3 or 4μ on the film. The attainable least count of PEPR is easily adequate for this purpose, but the real accuracy depends upon the stability and the reproducibility of the system. Careful measurements have shown that the PEPR system possesses reproducibility to 1 or 2μ and stability extends over periods of many hours.

There are, of course, systematic distortions inherent in the hardware. These distortions are due to the pincushion distortion of the CRT and to the interactions of the focusing and deflection magnetic fields. One result is that a straight line on film is measured as a shallow curve. Another effects that a specified line segment orientation results in a slightly different actual orientation at various places on the scope. Further, measurements made with different length line segments are systematically different.

These distortions, since they are systematic and well know, do not affect the basic accuracy of the PEPR system. They are minimized in the hardware as far as practical analog correction functions and careful alignment of optical components. The remaining distortions can be accounted for by calibration of the hardware with a precisely drawn grid.

The results of many measurements on such a calibration grid are used to generate mapping functions. These functions, map measurements made with the PEPR system into a "perfect" coordinate system, preserving the basic accuracy of the hardware. In principle, such corrections could be made at any time before the data from PEPR is used in further caluclations. They need not be made during the scanning or measuring. It has been more convenient, however, to make the corrections at the time of measuring, so that programs can operate with a perfect coordinate system.

The method of making the corrections is to use an input/output routine--called Black Box--for all communication between computer and controller. Scan cell parameters are generated by the programs in the perfect coordinate system, and Black Box makes any necessary adjustments before transmitting them to the controller. Similarly, data returning from the hardware is corrected to the perfect coordinate system. Besides correcting for distortions, Black Box converts

clock counts to units homogeneous with the cell center units and returns coordinates of hits as x, y, angle. The use of Black Box by all programs permits hardware changes and new calibration constants to be inserted without changing the logic or coding of the bulk of the programs.

PATTERN RECOGNITION IN THE PEPR SYSTEM

Before measurements can be made, the points to be measured must be found. This is a problem of pattern recognition. For bubble chamber photographs, the pattern of interest is an "event," which consists essentially of several tracks joining at a common vertex. The process of recognizing an event, however, involves "pattern recognition" at several levels.

The lowest lying pattern recognition problem is element recognition. The analog filtering done by the TED's (Track Element Detectors) allows hits from several angles around the actual angle of the track. Since a scan cell may contain several tracks at similar angles, the data returned from the scanning will consist of three or four times as many bits as there are tracks. The job of element recognition is to select the best set of coordinates for each track from this information and return only one such set for each track. Such a set of coordinates (x,y) constitutes an element of a track.

Figure 3a shows an illustration of a typical track configuration in a bubble chamber photograph. A scan cell is indicated, and scanning would occur at angles between initial and final. The data returned from Black Box is shown on a w-o plot in Figure 3b. Here each hit is represented by a point, with the w-axis representing the measured coordinate along the sweep direction, and the o-axis representing the angle at which the measurement is made. The dotted lines represent the connections made by element recognition, which classified the seven hits into two track elements, and supplies the average measurements for each element.

When the film area represented by Figure 3a is scanned completely, a number of track elements will have been found, as shown in Figure 3c by the heavy lines. This scanning procedure--finding all of the track elements in a given film area--is called area scanning. The next step in pattern recognition is to match the track elements into track sections. For most real tracks, at least some will be missing. Thus the matching process will, in general, produce track sections, with each track represented by several sections.

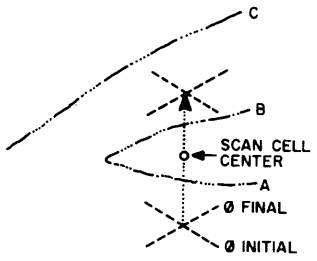


Figure 3a Illustration of typical bubble chamber event, showing scan cell.

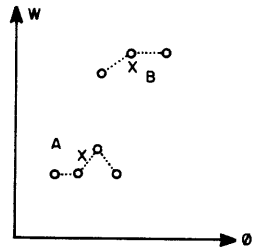


Figure 3b Illustration of Element Recognition. o = data returned from controller, x = measured element of track.

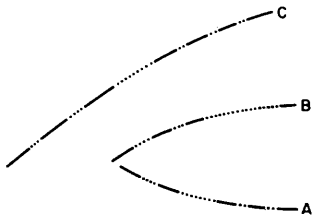


Figure 3c Solid lines represent track elements "seen." Dotted lines represent tracks as determined by track following.

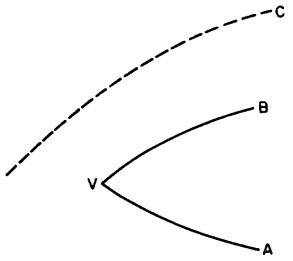


Figure 3d Final stage of pattern recognition links tracks A and B as an event; computes vertex point V.

Figure 3 Stages of Pattern Recognition

To complete the recognition of track sections must be joined across the gaps. This is being done by a process known as track following. A track section or even an element, is selected and an attempt is made to follow the associated track as far as possible. Track following operates in four phases. Phase 1 is the start-up phase, this uses the selected element or section to make several linear predictions of locations of adjacent elements. When these predicted elements are found, they are placed in a "track bank," and the track follower goes to Phase 3. Here a least square parabolic fit is made to the points already in the track bank. The resulting fit parameters are used to predict another point along the track. When the predicted point is found, it is added to the track bank. After enough points have been found (six) in this manner, the track follower switches to Phase 2. Predictions are now made by extrapolating from the last few points (eight) in the track bank using a method that essentially assumes the track to be

a circle on the film. Phase 2 continues to operate until a found element differs from its predicted value by more than an acceptance parameter, and it then reverts to Phase 3.

The usual course of track following is an alternation of Phase 2 and Phase 3. If either phase enters a "confused" region, however, Phase 3 is called. A confused region might consist of a missing element or a scan cell containing more than one element satisfying the prediction. Phase 4 establishes a pointer marking the last accepted element in the track bank, and then makes a parabolic fit to a number (ten) of accepted elements. This fit is used to predict beyond the confused region. If no elements are found on a given step beyond the confused area, prediction and checking occurs for the next area over, up to a maximum distance. If an element(s) is found, it is provisionally accepted, and a fit is made using the provisional element(s) and the accepted elements. This procedure continues (to a maximum length) until a series of "safe" elements have been established. These elements are added to the accepted elements already in the track bank, and the track follower returns to Phase 3. In Phase 4 all predictions are made only on the basis of accepted elements; provisional elements are not used. If the confusion persists after predicting for the maximum length allowed in Phase 4, the track bank is closed. This would usually occur because a track actually stopped or left the field of view of the photograph but it could also occur because the film became very messy and the track is not in fact measurable.

The net result of track following is a track bank containing all safe elements associated with a particular track. The bank may contain gaps, but the elements lying beyond the gaps have been certified by the track following procedure as being consistent with lying on a single track. The track following is done in both directions from the starting element, so the track bank spans the identifiable length of a track. The effect of track following is to provide the connections (dotted lines) between the elements in Figure 3c. The performance of the track follower is shown in Figure 3a.

When tracks have been identified on a photograph, the remaining pattern recognition task is to connect the tracks into "events," determining which tracks intersect in a vertex (rather than crossing) and measuring the location of the vertex. This is not an easy job to do by scanning because a vertex area is by nature a confused area. The event recognition, therefore, relies on the data already in the track banks. Likely candidates for an event are chosen, and

a program is used to find the point which minimizes the sum of the squares of the distances from all associated tracks. This provides both the measurement of the vertex and, using the residual distance, a means of testing the reasonableness of the assignment of tracks to events. The result of the event recognition is illustrated in Figure 3d.

The process of pattern recognition has provided a map, giving locations along all tracks belonging to events. This map itself is not yet a precision measurement because, in the interest of speed, the pattern recognition has been done in the low precision mode of operation. It is now a relatively simple and rapid process to measure points along the tracks. The measured tracks are at this stage only planar projections of a 3-dimensional event in a bubble chamber. To complete the job of digitizing the real event, three views of each event are measured, and later this information is used to make a geometrical reconstruction.

A PRODUCTION FILM PROCESSING SYSTEM

The basic modules for a film measuring system have been described. The hardware of the PEPR system has been built and is working. The major software components--precision calibration, element recognition, track following, event recognition--have been written and tested singly and in combination. What remains is to organize a production system.

Two considerations are important in setting goals for a production system. The first is that it must be fast to justify its use economically. The second is that it must be effective. The criteria for effectiveness in this case is that the system must be able to process most of the photographs given it. It is certainly expected that the first systems in operation will fail to recognize and measure many photographs that can be "reclaimed" by human scanning. But, considering the amount of film involved, a rejection rate of over 10% of the photographs would be rather high.

Operating speed depends on both the scanning hardware and on the program running time. The actual hardware scanning time needed to completely digitize a section of film 50 mm x 50 mm is only a little over 1 sec. The necessary nonscanning time used by the computer is (for the PDP-1) greater by a factor of 100 or so. This time is used simply to direct scanning (setting parameters in the controller and retrieving data) and to perform element recognition. Thus, a complete area scan of a photograph would require 1 or 2 min and would be only the prelude

to the next steps of pattern recognition. On the other hand, track following requires about 3 to 5 sec to follow 10 cm of track. A prerequisite for a production system is to minimize the amount of area scanning needed to select starting points for track following.

Our experience has shown that the effectiveness of the system depends primarily on the track follower. Unless the tracks can be measured for a reasonable length, the event cannot be recognized or measured. In turn, the success of track following depends on getting a good start. If enough elements have been measured along a track, prediction past confused areas is usually successful.

The first production system--called Point Guidance--will incorporate guidance features to increase speed and reliability. Part of the guidance consists of rough digitizing and scanning of the film before it is run on the PEPR System. Human scanning with rough digitizing is a relatively rapid procedure, taking a technician about 5 min to scan and digitize three views of a photograph with equipment costing far less than precision measuring machines. The guidance information provided will include a starting point on each track, an indication of the end point for each track, and of the vertex point for each event. Additional information is provided where the track passes through confused areas on the film.

This form of guidance eliminates extensive area scanning of the film, as the tracks are known to be in relatively small regions. Point guidance should also increase the reliability of track following because the starting points for the track following are chosen to be in "safe" areas. Such guidance does not eliminate the job of pattern recognition, but it does restrict the pattern recognition to smaller areas and (hopefully) less difficult problems.

Some difficult problems will still arise, and for these the Point Guidance System incorporates on-line human aid. When an event cannot be recognized or measured, a picture of the film as seen by the scanning hardware will be displayed on the computer display scope. The operator can cause additional information to be displayed, such as the elements in a track bank, or the guidance points. He can then help the system by adding or detecting information using the light pen and scope display, and the programs can try again. We expect that this feature will reduce the number of rejected photographs and also serve as a powerful debugging aid.

FUTURE PROSPECTS FOR PEPR

Although no production has been done with PEPR, we feel that the hardware and basic programming ideas are sound. The first production system, Point Guidance, is now being written, and we expect to begin measuring events with it sometime during the summer. We expect it to begin by processing a few hundred events per day, and as experience suggests improvements, the rate should increase to about a thousand per day. Since logic operation and arithmetic in the computer take much longer than the actual scanning of film, the easiest improvement is to replace the PDP-1 with a faster computer. Thus we are acquiring a PDP-6, and we will move the Point Guidance System to the PDP-6 as soon as it is running on the PDP-1.

Beyond the increase in speed, we hope to reduce the amount of guidance supplied in the prescanning. How this is done will depend on our experience from operating the Point Guidance System. There are also other pattern recognition strategies, depending on the nature of the experiment being analyzed. For experiments done with a charged particle beam, one might follow all beam tracks until one arrives at a vertex, thus eliminating extensive area scanning. Another strategy would be to prescan, digitizing only one point near each event vertex, and then use a restricted area scan near that point to select tracks to be measured.

ACKNOWLEDGEMENT AND REFERENCES

The development of PEPR has been a large project, owing its present state to the devoted effort of a number of people from M.I.T. and other cooperating universities. The general idea of PEPR was conceived by Professors Pless and Rosenson, of M.I.T. and the development of PEPR at M.I.T. has been directed by Professor Pless. The hardware of the PEPR system has been developed by a team of engineers led by Mr. Wadsworth at M.I.T. including Mr. Kenyon from University of California, Berkeley; Mr. Sartori, CERN; and Mr. Brooks and Mr. Cormack, M.I.T. The programming has been done by Professor Rosenfeld, University of California, Berkeley, Prof. Taft and Dr. Bogart, Yale; Dr. Crouch, Brown; Dr. Bordener, Harvard; and Drs. Bastien Watts and Yamamoto, and Mr. Dunn at M.I.T. And in a project where so many people have contributed ideas and effort, the list of contributors could be lengthened still.

This report has been brief and readers interested in a more detailed discussion are directed to:

Bastien et al. "Programming for the PEPR System," Methods in Computational Physics (1965) Academic Press.

DEXTER—THE DX-1 EXPERIMENTERS TAPE EXECUTIVE ROUTINE*

by

Jerome M. Cohn

Wolf Research and Development Corporation
West Concord, Massachusetts

Abstract DEXTER is an executive routine designed to facilitate the operation of the DX-1 dynamic data processing system during experimental runs. It affords the experimenter full communication with the system from the console of one processor while maintaining the advantages of dual computer processing.

The software for DEXTER consists of a two-part executive routine and several maintenance routines. Other programs are written in a DEXTER Compatible Format and stored on the executive magnetic tape. At run time, DEXTER is loaded into two computers which are connected through an information exchange, a one-word communication link. Each processor is equipped with a magnetic tape on which are stored the necessary routines. The order in which these are loaded and executed by the executive program may be specified either by the experimenter from the on-line typewriter or by the program itself as a result of computation, etc. This gives the desired flexibility without sacrificing speed or efficiency. When a program is called, the executive routine restores itself and searches for the correct block. If the program is not found, control is returned to the experimenter at the keyboard. If it finds the program, DEXTER loads the routine, over itself if necessary, repositions the tape and transfers control to the program. The program must return control to DEXTER when it is finished so that other programs can be called.

Other DEXTER features are: minimum programming restrictions, minimum storage limitations (self-restoring), full debugging features, and comprehensive editing programs for preparing the executive tape. A basic version of DEXTER for a minimum PDP-1 is available and will be described.

THE INSTALLATION

The Experimental Dynamic Processor (DX-1) at the Data Sciences Laboratory, AFCRL, centers around the use of two PDP-1 computers. The two processors can communicate through the "information-exchange" channel and share much of the auxiliary equipment. The machines are usually kept in the "normal" configuration as shown in Figure 1. When required, units can be switched between processors to make up another configuration for a special program.

---

*Work supported by the Data Sciences Laboratory, Air Force Cambridge Research Laboratories, Bedford, Massachusetts.

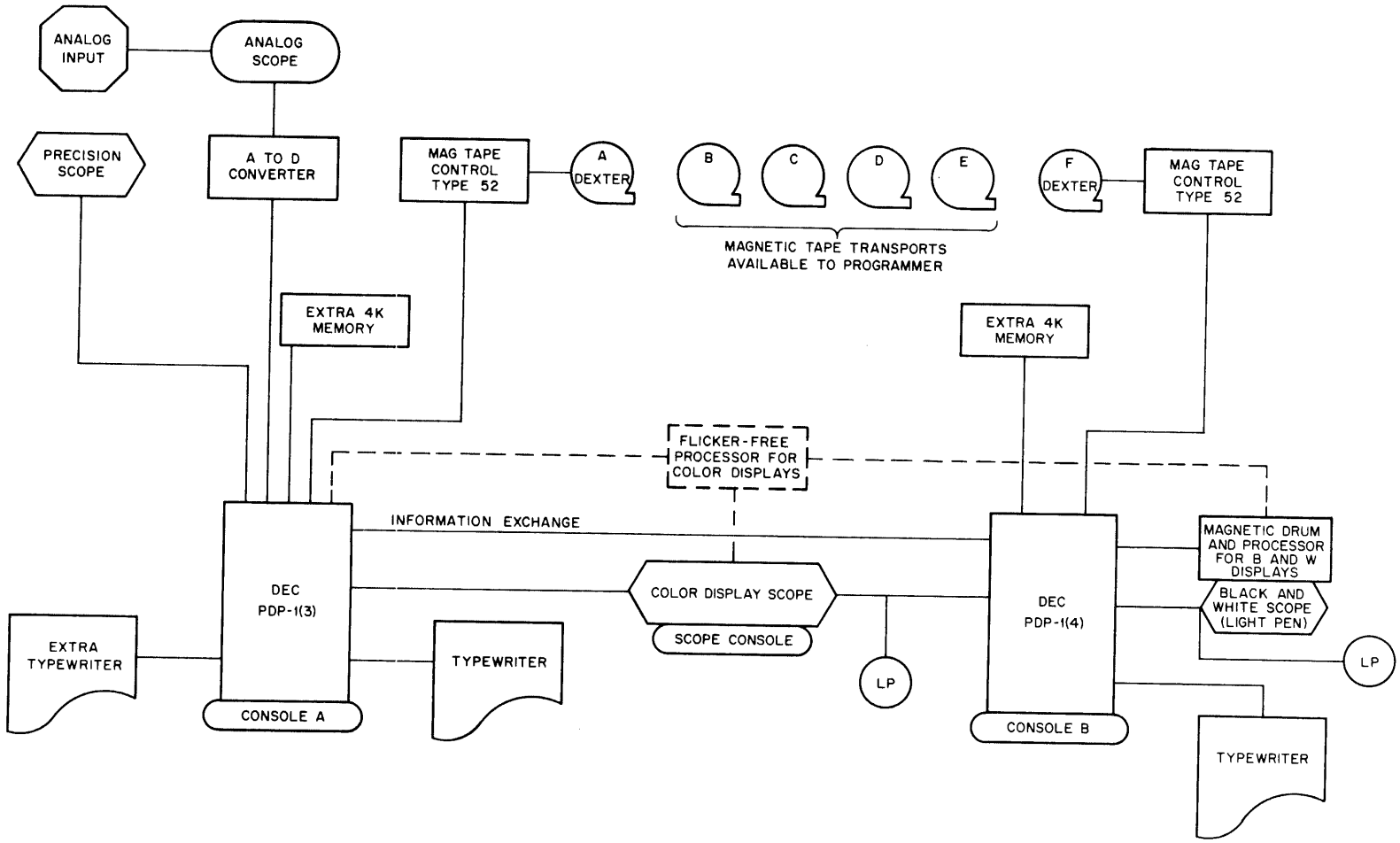


Figure 1 Dynamic Data Processing System (DX-1)

The programming language is AMP, an assembler with provision for macro programming. Symbolic programs are entered via paper tape and the AMP system uses two scratch tapes plus its own system tape for assembly. The system tape contains library routines which can be pulled in by program references or by macro instructions.

THE PROBLEM

In an on-line experimental computing facility many programs are written, tested, and then used only infrequently. Many runs are made up of segments of several programs used in various sequences. The experimenter should have a wide variety of programs at his fingertips so he should not have to weigh the inconvenience of finding and loading a program against its possible benefit. He should not have to "make do" in order to avoid the bother of getting a program into the machine.

The experimenter should be spared the handling of paper tape as much as possible. Loading paper tapes is a time-consuming chore, prone to error and requiring constant attention. The scientist may be so occupied by the task that his attention is diverted from on-line innovation.

In the DX-1 installation an additional problem is present. Many programs utilize both processors working together on a single problem. Use of paper tapes forces the operator to walk back and forth between the two consoles to change the program. This is distracting and time consuming.

Occasionally a program may be too large to fit in core all at once. A possible way around this is to segment into two smaller programs. If the nature of the problem is such that the two segments must alternate many times in memory, the constant reloading of paper tape is very inconvenient. A subroutine might be written to use magnetic tape for the swapping procedure. This method is convenient for the operator but imposes an extra burden on the programmer and uses scarce storage space.

It is not possible for a program to automatically read in another program from paper tape. If the sequence of programs is determined by the data, one program must type out the name of the next and allow the operator to load it manually.

DEXTER

Purpose

In view of the problems inherent in paper tape handling, WRDC has designed and implemented a magnetic tape executive system, DEXTER, for the DX-1. A resident routine selects programs from a magnetic tape by means of three character identifiers. When the program is finished, it returns control to the executive, instead of halting, so that the next program can be called. All programs currently in use at the installation are stored on the system tape. The DEXTER system tape, in turn, is stored as a separate file at the end of the AMP assembler system tape. With this arrangement, a user can use the assembler and DEXTER without changing tapes.

An editing program for the system tape was also written. This allows for additions, deletions, or substitutions and also allows the programs to be rearranged on the tape. Thus an operator can assemble a symbolic program with AMP, append it to the DEXTER file with the editor, and then run the program under DEXTER control all in one run without magnetic tape change and with no paper tape handling other than reading in the original symbolic.

Organization

In the DX-1 system, programs that use both processors always use one as the controlling machine and the other as a satellite. Ideally, the operator should be able to exercise all control from the main console. DEXTER was designed with this in mind. There are two versions of the resident monitor, DEXTER A and DEXTER B. DEXTER A, designed for the primary processor, permanently occupies locations 100-177 of core 0. In addition, it uses locations 200-1000 when loading programs. The rest of memory is untouched. This means that no program can use the block 100-177 for any purpose. The area 200-1000 can be used by programs but is destroyed by subsequent program calls. All the rest of memory can be used and is never destroyed by the executive: therefore data can be left by one program for the next in these areas.

The resident program in the satellite machine is DEXTER B. This program permanently occupies locations 7000-7777 of core 1. This area cannot be used by any program but all other locations are untouched.

DEXTER A permanently occupies locations 100-177 but must restore itself from 200-777 when used. The part of the program that must be restored is kept on the system tape, just like any other program, with an identifier of 000. When a program returns control to the executive, the first thing that is done is to restore this segment.

The arrangement of programs on the system tape has no effect on DEXTER. However, to reduce tape passing time, the most frequently used programs should be near the beginning. The tape is backspaced to the start of the file after each search. The format of each program block is as follows:

[ID]	3-character name of program
[IA]	initial loading address
[FA]	final + 1 loading address
[SA]	starting address
[program]	

The last program on tape is followed by an end of file mark. If the file mark is read before a program is found, a message is typed telling the operator the program is not on the tape. The first three programs in the file are usually the restore segment of DEXTER A (ID=000), DEXTER B (ID=B i.e., 746272 octal), and the tape editor (ID=edi).

Use

The system tape is mounted and the paper tape of the AMP monitor is read in. This is the only paper tape that has to be handled. The DEXTER system is called by a type-in to the AMP monitor. If a dual-processor system is used, DEXTER B must be called into the second computer in a similar fashion. The operator calls the program that he wants by typing in its 3-character ID. This program is brought into the controlling processor and can send the ID of the program to be loaded into the satellite processor through the information exchange. The control and satellite programs then run to end of job at which time each returns control to its resident executive. Another pair of programs can now be called by type-in of the main program ID.

Further convenience is built into the system by allowing a type-in of a list of up to five program ID's at one time. In this case, the first program will be called and executed. When it

returns control to DEXTER, the second program will be called immediately without operator attention. This will continue until the list is exhausted. This allows the operator to set up a run, consisting of several programs at one "sitting" at the typewriter. The same program may appear in the list several times if desired.

The normal return to DEXTER control consists of a CAL with a cleared accumulator. Another option, however, allows the program to return with the ID of another program in the accumulator. In this case DEXTER will call in that program automatically. This is extremely useful when programs must be segmented due to lack of storage or when the next program to be called depends on computation or decisions in the present program. This type of automatic call takes precedence over a list of ID's from the keyboard. That is, the next program on the list is called only when a return to DEXTER is made with a cleared accumulator.

If a program executes the CAL with minus zero (777777) in the accumulator, this is considered an error return. This is used whenever the program could not proceed to its normal end of job. In this event, the executive types an error message. It also assumes that any continuous chain of computation has been broken and accordingly cancels ID's on the list entered from the keyboard, if any.

If a program does not return control to DEXTER for any reason (e.g. an undebugged program does not run to end of job), the executive may be restarted at 100. This has the effect of cancelling any list of ID's.

Several features are available to the operator for convenience in debugging. If sense switch one is turned on, a halt will occur before the jump to a loaded program is executed. This allows program modification before running. In addition, by typing an overbar and an octal number, the operator can set all of memory to that number. This provides "background" for any program subsequently loaded. This is useful in locating transfers out of the program's area or faulty table lookups, etc.

Compatible Programs

The restrictions on programs to be run under DEXTER control have been made as unobtrusive as possible. The philosophy has been that if the system is to be used, it must not interfere seriously with programming.

A program cannot use magnetic tape drive 1 for any purpose. This drive is reserved for the systems tape. Also a compatible program cannot use any locations reserved for the executive. If it is a main program, this means that locations 100-177 cannot be used. A satellite program cannot use 17000-17777. In addition, a main program cannot use locations 200-777 to pass data to another program.

These restrictions have been found to be a negligible obstacle in obtaining the full power of the DX-1.

DECUS VERSION OF DEXTER

A version of DEXTER A has been written which will run on an 8K PDP-1 with Type 52 Control and one drive. This version contains all the features mentioned above except all provisions for control of a satellite processor. The system tape editor is also available in a DECUS version requiring two tape drives. Listings, symbolic and rim tapes, and a complete writeup have been submitted to the DECUS Program Library.

THE USE OF A SMALL COMPUTER WITH REAL TIME TECHNIQUES* FOR OCEANOGRAPHIC DATA ACQUISITION, IMMEDIATE ANALYSIS, AND PRESENTATION

Robert M. O'Hagan

Digital Equipment Corporation
Maynard, Massachusetts

Abstract A compact, high-speed digital computer has been programmed to immediately analyze and evaluate data gathered from in situ data collecting instruments. This paper describes the system and its output.

The computer can be located either aboard an oceanographic vessel or at a land-based station. Using the computer aboard ship, the oceanographer can change the sampling depth intervals to obtain the most significant incremental changes in temperature, salinity Sigma-T, and sound velocity, all of which have been computed from primary data. The computer also obtains and stores data at internationally accepted standard depths.

With the computer separated from the sensors, a telemetry link to handle data from buoy systems permits the computer to become a data monitoring and buffering device between the sensors and data storage, as well as a tool to make possible immediate decisions.

INTRODUCTION

Currently the digital computer is generally thought of as a very fast and efficient method of processing collected oceanographic data. However, its recent use in industrial process control has shown that it is also a valuable tool in acquiring information and making logical decisions concerning the acquisition of data while concurrently storing, processing, and displaying information. Certainly a method of data manipulation similar to industrial control techniques could be advantageous to the oceanographer either aboard a vessel while it is in the marine environment or ashore when the marine environment is sampled via a telemetry link.

INTERFACING MARINE SENSORS

Marine sensing instruments are often considered unique devices that do not lend themselves to being connected directly to a computer; however, by the use of basic standard interface types, most instruments can be connected directly to the computer with little additional equipment. Let us examine the basic types of interfaces that would be necessary to interconnect the computer to oceanographic sensors.

INTERFACE TYPES

Parallel-Parallel Input Signal Buffer

This buffer permits the direct parallel insertion of a digital number into the computer, a method in which the number becomes a computer word immediately. Examples of devices feeding data in by this method are shaft encoders used with Loran and Decca receivers, ship's heading and speed encoders, certain current meters, fathometers and other allied devices. Figure 1 shows the general method.

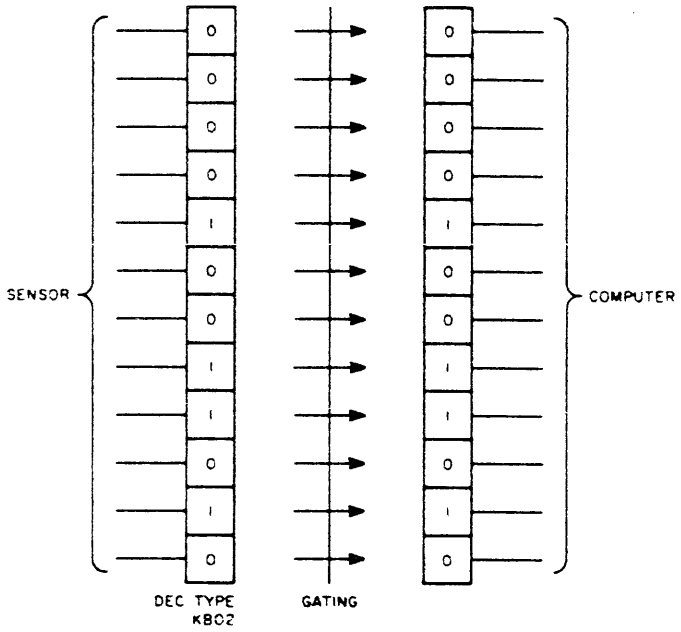


Figure 1 Digital-Parallel Signal Buffer

This method can accommodate signals or pulses from a minimum range of 0 to minus 10 millivolts up to a maximum of 20 to minus 15 volts. The system can include one buffer for each of several dozen variables. Each buffer generates a separate pulse to tell the computer when a new word has been assembled.

Serial-Parallel Input Signal Buffer

This method would be used principally in telemetry systems, where the transmitter is either aboard the ship or on a buoy, and the system is able to transmit a number of input words one bit at a time, along a single conductor cable or by radio. See Figure 2. Examples of the sources of this type of input are pressure, temperature, and salinity sensors, current meters, and strings of sensors.

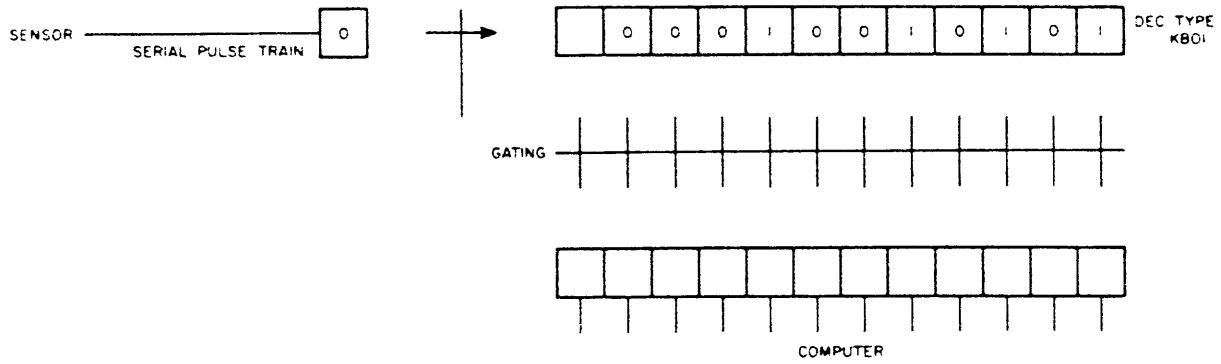


Figure 2 Serial Buffer Input

This buffer can convert a serial pulse train to a 12-bit word in 6 microseconds, or 1 bit every 500 nanoseconds. It accommodates ranges of input levels from a minimum of 0 to minus 10 millivolts, up to a maximum of 20 to minus 15 volts. The flexibility provided in this buffer lets the engineer or oceanographer format data before they are finally assembled as computer words. That is, he can insert octal constants in the specific serial word of his choice. It provides the programmer with the following additional instructions in the computer:

1. Skip if data flag = 0
2. Skip if start flag = 0
3. Clear data flag and start flag
4. Read data into the accumulator

Multiplexed Analog-to-Digital Conversion Input

This method is particularly advantageous in data acquisition when many devices such as thermistors, pressure sensors, and conductivity sensors are working together. One example of where this method would prove advantageous is a thermistor chain in which each thermistor, pressure sensor, or conductivity sensor could be individually sampled by the computer. Figure 3 indicates this relationship.

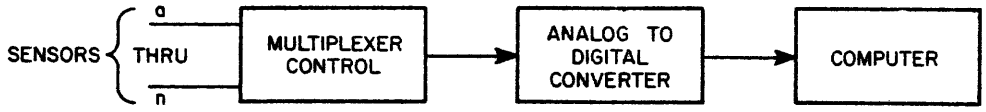


Figure 3 Multiplexed Analog-to-Digital Conversion

Switching from one input to the next in the multiplexer is accomplished in 2 microseconds, and up to 64 separate inputs can be sampled directly by the computer. The analog-to-digital converter uses the range of 0 to minus 10 volts. This system costs $4,500.

PROGRAMMING

The task of writing a special-purpose program for each installation could be a formidable problem involving a great deal of time and money. In order to alleviate this problem, Digital Equipment Corporation has written an algebraic compiler that allows the oceanographer to format his data acquisition problem in a simple language similar to algebra. By using this language, he is given a great deal of flexibility concerning the interface hardware that he has. This program also allows him flexibility in choosing the frequency and conditions under which he samples the marine environment. It also makes possible the adding or changing of sensors without the major task of reprogramming in machine coding. This program is available for the PDP-8, a compact 12-bit computer with a 1.5-microsecond cycle time, or the PDP-7, a medium size 18-bit machine with time sharing capability and a 1.75-microsecond cycle time.

Using this program, the investigator could sample the following inputs:

Up to 96 independent data variables using digital-parallel input
Up to 64 independent data variables using a multiplexed A to D converter
Up to 25 independent data variables via serial buffer input

Each independent variable can be sampled at a rate of up to 100 times per second.

Input

Data can come to the computer from the digital-parallel input buffer, the serial input buffer, and the multiplexed analog-digital converter. Each of these devices includes in its message a symbolic name that tells the computer what device is transmitting information.

The symbols are:

DGIN: Digital-parallel signal buffer; input can be converted from Grey code to binary

BUFR: Serial buffer input

ADCV: Multiplexed analog-digital converter

Output

Data output can be distributed to a number of specifically named devices to allow immediate presentation as well as permanent storage. The following output symbols and their associated devices are presently available:

TYPE: On-line teleprinter. Variables can be typed in decimal or octal. Decimal is specified by ↑ immediately following the variable name.

PNCH: High speed paper tape punch. Variables can be punched in decimal or octal. Decimal is specified by ↑ immediately following the variable name.

DCTP: Digital's compact DECtape. Variables are recorded magnetically on DECtape in binary with identifying words.

PLDT: X-Y plotter. The plotter pen is moved to a new position each time output to it is specified.

DIG 1, DIG 4: Up to four digital outputs are available through parallel buffers to other devices such as relays, buffers, sense lines, and range switching devices.

Variables

Input variables to the computer are assigned alphanumeric symbols by the oceanographer. They can be one to four characters long, and must begin with a letter. Examples of these are as follows:

T 123, SURF, DEEP, AIR, X, Y, TEMP, H20, H202

In addition, variables that are inputted through the multiplexer have specified channels; that is, T 123 (1) would be input through channel 1 of the multiplexer.

Time

Four types of time can be used by the computer: basic, program, variable, and reference.

Basic Time

Basic Time represents the basic interval in which a clock interrupts the program. In the PDP-7 and PDP-8, the program is interrupted every .01 second.

Program Time

This time represents the basic rate at which the investigator desires to interrogate the sensors. It is some multiple of the basic time and is under program control. Its symbology is simply expressed as follows:

QUNT: 62 — 62 is the octal equivalent of 50 decimal. Thus the investigator has specified that the Program Time will be $(62_8) \times (.01) = 0.5$ seconds; that is, each sensor will be sampled every 0.5 seconds. If he desires the fastest rate possible he would have expressed the following:

QUNT: 1 in which case each variable will be sampled every 0.01 seconds.

Variable Time

In order to allow more flexibility in timing, digital inputs can be sampled at a slower rate than the program time specifies. For example, the following expression specifies that the digital input variables L1 and L2

DGIN:L1 (4, L2 (4

should be sampled once in four cycles of the program time or every (4) x (0.5) = 2 seconds.

Reference Time

It is often desirable to know the reference time in order to associate data with time. Within the program is a three-word variable, CLOK, which counts the number of seconds, minutes, and hours that have elapsed since start-up time, and it can be used as an output variable to reference data with time.

Arithmetic Operations

Addition, Subtraction

Variables can have constants added to or subtracted from them as they are sampled, or the variables can be added to or subtracted from each other.

All arithmetic operations are done in 2's complement arithmetic, with the operands being considered signed fixed point numbers. The following examples mean that the variables will have constants added or subtracted before output:

T2 + 137 add a constant to the variable
T2 - 3 subtract a constant
T2 + T3 add a second variable

Arithmetic Comparison

Variables can be compared against constants, compared against other variables, or compared against themselves with respect to sample time. The basic comparison instructions are:

IFEQ X, Y	if |X| is equal to |Y|
IFLS X, Y	if |X| is less than |Y|
IFGR X, Y	If |X| is greater than |Y|

An example of the comparison of a variable against a constant is as follows:

```
IFEQ X, 1000;
```

Meaning that if X is equal to 1000, execute the operation following the semicolon. Otherwise go to the next line.

Every time a variable is recorded and outputted, its value is preserved and is given the name of the original variable. Thus, X and@X are the same variable recorded and outputted at succeeding times.

An example of the comparison of a variable and its predecessor is as follows:

```
IFLS X, @X;
```

Meaning that if X is less than it was when last recorded and output, execute the operation following the semicolon. Otherwise go to the next line.

Gray Binary Conversion

Gray binary code can be converted to simple binary under program control if the input method is digital (DGIN). This provides the investigator a rapid means of conversion in order to inter-compare the usual shaft encoded Gray binary numbers, if the shaft encoder does not convert from Gray Code to simple binary prior to buffering.

The conversion is accomplished by inserting ↑ immediately before the time multiple.

```
DGIN  L1↑4
```

This means that the Gray binary variable L1 is sampled every fourth time through the program and is converted to a simple binary number before comparison or storing.

Format Statements

Format statements are numbered from 1-17 (octal). They contain the names of variables and their output forms (octal or decimal for the Teletype or punch). Format numbers appear along with a device name in every output statement. Thus, the statement

FORM: 1,X,Y ↑,Z

indicates that the variables X, Y, and Z are to be outputted to the Teletype, with X typed in octal, Y typed in decimal, and Z typed in decimal.

GOTO Statement

Program control can be unconditionally transferred through the use of the GOTO statement.

PROGRAM EXAMPLES

Problem 1: A Simple Programming Problem

An in situ pressure, temperature, and salinity sensing instrument is lowered into the ocean. Data are transmitted along a single conductor cable and are brought into the computer using a serial buffer input.

We want to sample the ocean in the following manner:

1. From the surface to 100 meters, record at each meter the pressure, temperature, and salinity.

2. From 100 meters to 1000 meters, record the pressure, temperature, and salinity whenever the absolute change of temperature is greater than .05°C or the absolute change of salinity is greater than .02 ‰. Also record the pressure, temperature, and salinity every 100 meters from 100 meters to 1000 meters.

Let us assume that the oceanographic sensors have the following precision, that is, unity is equal to the following:

1 unit of pressure = 1 meter

1 unit of temperature = .01°C

1 unit of salinity = .01 ‰

A program to accomplish this sampling is written as follows:

```
BUFR :    PRES, TEMP, COND
FORM:     1, PRES, TEMP, COND
[      :  OUTP (1, DCTP
1      :  IFGR PRES, 144; GOTO 2
       :  IFGR (PRES -@PRES), 1; OUTP (1, DCTP)
       :  GOTO 1
2      :  IFLS (PRES -@PRES), 144; IFLS (TEMP -@TEMP), 5;'
          IFLS (COND -@COND), 1; GOTO 2
       :  OUTP (1, DCTP); GOTO 2
       ]
          END
```

This program says, in effect:

BUFR: PRES, TEMP, COND

Three variables named PRES, TEMP, and COND are to be sampled using the serial buffer.

FORM: 1, PRES, TEMP, COND

Three variables named PRES, TEMP, and COND are to be outputted together.

:OUTP (1, DCTP)

This says output is to be recorded on magnetic tape.

1:IFGR PRES, 144; GOTO 2

This states that if the absolute change of pressure is greater than 100 (144_8), the control of the sampling will be transferred to statement number 2; otherwise it will go to the next line.

:IFGR (PRES -@PRES), 1; OUTP (1, DCTP)

This line states that if the absolute change of the pressure between two successive readings is greater than 1, then output onto magnetic tape according to Format 1; that is, OUTP (1, DCTP) which means store data on DECtape using Format 1; otherwise, go to the next line.

```
GO TO 1
```

This says to go to statement number 1 and test the environment again.

```
2:  IFLS (PRES -@PRES), 144; IFLS (TEMP -@TEMP), 5;'
    IFLS (COND -@COND), 1; GOTO 2
```

This states that if the absolute change of pressure is less than 100 meters or the absolute change of temperature is less than .05°C, or if the absolute change in salinity is less than .02 ‰ then go to statement number 2 which begins the tests over again. Otherwise go to the next line.

```
:OUTP (1, DCTP); GOTO 2
```

This says to output data onto magnetic tape and transfer control to statement number 2. The sampling and testing procedure begins again.

```
END
```

This last instruction is self explanatory.

As shown in the above description, the computer has been programmed to make logical decisions specified by the investigator in sampling the marine environment. It also has been used as a means of storing data. In the above instance, data have been stored on magnetic tape and can be used in other programs to determine variables such as Sigma T, anomaly of specific volume, and sound velocity. Figure 4 shows a portion of the calculated output from stored data on magnetic tape transport number 1 that can be run immediately after the sample program.

```
INPUT SOURCE?  T
   OBSERVED VALUES
```

DEPTH	TEMP.	SALIN.	SIGMA-T	DELTA-A	SOUND-VEL
0000	8.35	34.17	+26.590	+145.53	+1483.4
0001	8.28	34.19	+26.616	+143.08	+1483.2
0002	8.20	34.21	+36.644	+140.45	+1482.9
0003	8.13	34.24	+26.678	+137.20	+1482.7
0004	8.05	34.26	+26.706	+134.59	+1482.4
0005	7.98	34.28	+26.732	+132.19	+1482.2
0006	7.91	34.31	+26.766	+128.98	+1482.0
0007	7.83	34.33	+26.793	+126.38	+1481.7
0008	7.76	34.35	+26.819	+123.94	+1481.4
0009	7.69	34.37	+26.845	+121.45	+1481.2
0010	7.61	34.39	+26.873	+118.89	+1480.9

Figure 4 Reduced Data

Problem 2: A More Sophisticated Program

For a better demonstration of the flexibility of this programming technique, consider the following program.

An investigator desires to use a thermistor, pressure, and conductivity chain towed from an oceanographic vessel. He will also sample at the same time a telemetering buoy that transmits data from three current meters. In addition, he desires to obtain Loran lines of position and sample the ship's speed and ship's heading. These can be summarized as follows:

1. Log the time on magnetic tape every 50 meters of distance traveled. Ship's speed is 10 knots; it will cover 50 meters in approximately 9.70 seconds.

2. Sample each thermistor, conductivity, and pressure sensor in the chain every half second. This defines the program time thus, QUNT: 62.

3. Sample the Loran, ship's speed, and ship's heading every 2 seconds thus, L1(4, L2(4, SP(4, HEAD(4.

4. Conditional Output - Since the near-surface values of temperature and conductivity will fluctuate the most, it might be most desirable to set thresholds so that relatively large changes of temperature and salinity will be stored. However, deeper values will not change as significantly, so small incremental changes have more meaning and thus should be outputted and stored. Arbitrary values have been chosen and are shown in Figure 5.

Determination of Octal Constants to be Used in Testing - In order to test the variable we must know to what its unit value corresponds. This is found by dividing the range of the thermistor, pressure transducer, or other device by the precision of measurement; thus if the range of the thermistor is 20°C and the precision of measurement is 1 part in 2000, then each unit equals .01°C.

Summarizing then:

Variable	Range	Precision	UNITY Corresponds to
Thermistor	20°C	1:2000	≅ .01°C
Pressure	100 meters	1:2000	≅ .05 meters
Conductivity	20 ‰	1:2000	≅ .01‰
Vane	360°	1:120	≅ 3°
Compass	360°	1:120	≅ 3°
Rotor			≅ 1 centimeter per second

5. General Output Requirement

   Every variable should be recorded on magnetic tape if the specified conditions are met.

   Plot T0 versus S0 if it is recorded.

   Type Current Meter Data in Decimal if it is recorded.

   Punch T0, P0, and S0 if they are recorded.

By outlining the problem, the investigator will have thresholds established for recording changes in the variables listed in Figures 5, 6, and 7. Figure 8 shows the equipment needed to do the work.

Depth in Meters	Thermistor Variable Name	Multiplexer Channel #	Record if Absolute Change of	Pressure Variable Name	Multiplexer Channel #
0	T0	(0)	.1° C	P0	(6)
10	T1	(1)	.07°C	P1	(7)
20	T2	(2)	.05°C	P2	(10)

Depth in Meters	Thermistor Variable Name	Multiplexer Channel #	Record if Absolute Change of	Pressure Variable Name	Multiplexer Channel #
30	T3	(3)	.03°C	P3	(11)
40	T4	(4)	.02°C	P4	(12)
50	T5	(5)	.01°C	P5	(13)

Depth in Meters	Record if Absolute Change of	Conductivity Variable Name	Multiplexer Channel #	Record if Absolute Change of
0	.5 meter	S0	(14)	.05 ‰
10	.5 meter	S1	(15)	.05 ‰
20	.5 meter	S2	(16)	.04 ‰
30	.3 meter	S3	(17)	.03 ‰
40	.2 meter	S4	(20)	.02 ‰
50	.1 meter	S5	(21)	.01 ‰

Figure 5 Multiplexer A-D Converter Assignment (ADCV)
(Data Originating in Thermistor Chain)

	Vane Variable Name	Compass Variable Name	Record if	Rotor Variable Name	Record if
Current Meter 1	V1	C1	V1 + C1 = 9°	R1	R1 > 10
Current Meter 2	V2	C2	V2 + C2 = 6°	R2	R2 > 10
Current Meter 3	V3	C3	V3 + C3 = 3°	R3	R3 > 10

Figure 6 Serial Data Input Buffer Assignment (BUFR)
(Data Originating in Moored Current Meter String)

	Variable Name	Time Multiple	Gray Code ?
Loran Line	L1	(4	
Loran Line	L2	(4	
Ship's Speed	SP	(4	
Ship's Head	HEAD	↑4	yes

Figure 7 Digital Input Buffer Assignment (DGIN)

(Data Originating in Ship's Instrumentation)

The program listing to sample these variables and record those which exceed the established thresholds is given below.

```
ADCV:   T0(0), T1(1), T2(2), T3(3), T4(4), T5(5), P0(6),'
        P1(7), P2(10), P3(11), P4(12), P5(13), S0(14), S1(15),'
        S2(16), S3(17), S4(20), S5(21)
BUFR :  V1, C1, R1, V2, C2, R2, V3, C3, R3
DGIN:   L1 (4, L2(4, SP(4, HEAD ↑(4
QUNT:   62
FORM:   1, T0, T1, T2, T3, T4, T5, P0, P1, P2, P3,'
        P4, P5, S0, S1, S2, S3, S4, S5
FORM:   2, T0, S0
FORM:   3, V1 , C1 , R1 , V2 , C2 , R2 , V3 ,'
        C3 , R3
FORM:   4, T0, P0, S0
FORM:   5, L1, L2, SP, HEAD, CLOK
        <5, DCTP, 22>
[   :   OUTP(1, DCTP); OUTP(2, PLDT); OUTP(3, TYPE); OUTP(4, PNCH)
3   :   IFEQ (V1+C2), 3; GOTO 1
        IFEQ (V2+C2), 2; GOTO 1
        IFEQ (V3+C3), 1; GOTO 1
        IFLS R1, I2; IFLS R2, I2; IFLS R3, I2; GOTO 2
1   :   OUTP (3, TYPE)
2   :   IFLS (T0 - @T0), I2; IFLS (P0 -@P0), I2; IFLS (S0 - @S0), 5;'
        IFLS (T1 -@T1), 7; IFLS (P1 -@P1), I2; IFLS (S1 -@S1), 5;'
        IFLS (T2 -@T2), 5; IFLS (P2 - @P2), 2; IFLS (S2 -@S2), 4;'
        IFLS (T3 -@T3), 3; IFLS (P3 -@P3), 6; IFLS (S3 -@S3), 3;'
        IFLS (T4 -@T4), 2; IFLS (P4 -@P4), 4; IFLS (S4 -@S4), 2;'
        IFLS (T5 -@T5), 2; IFLS (P5 -@P5), 2; IFLS (S5 -@S5), 1;'
        GOTO 2
    :   OUTP(1, DCTP); OUTP(2, PLDT); OUTP(4, PNCH); GOTO 3
]
        END
```

SUMMARY AND CONCLUSIONS

Three basic types of interfaces are used with the PDP-7 and PDP-8 computers in order to receive data from marine environment sensors. These interfaces allow data to enter the computer under control of a simple real-time symbolic compiler that gives the investigator the following flexibility when he samples the environment:

1. A variety of preprogrammed interface devices that easily connect the computer to oceanographic instrumentation.

2. Simple symbolic assignment of identifying names to input variables such as pressure, temperature, salinity, current meters, and navigation equipment.

3. Absolute control in sampling the marine environment with respect to time and distance.

4. Computer compatibility with respect to rapid response marine sensors using up to 176 separate sensing devices.

5. Capability to make logical decisions concerning the acquistion of data while sampling the environment.

6. Storage of data for future computations as well as immediate output for checking quality while obtaining data.

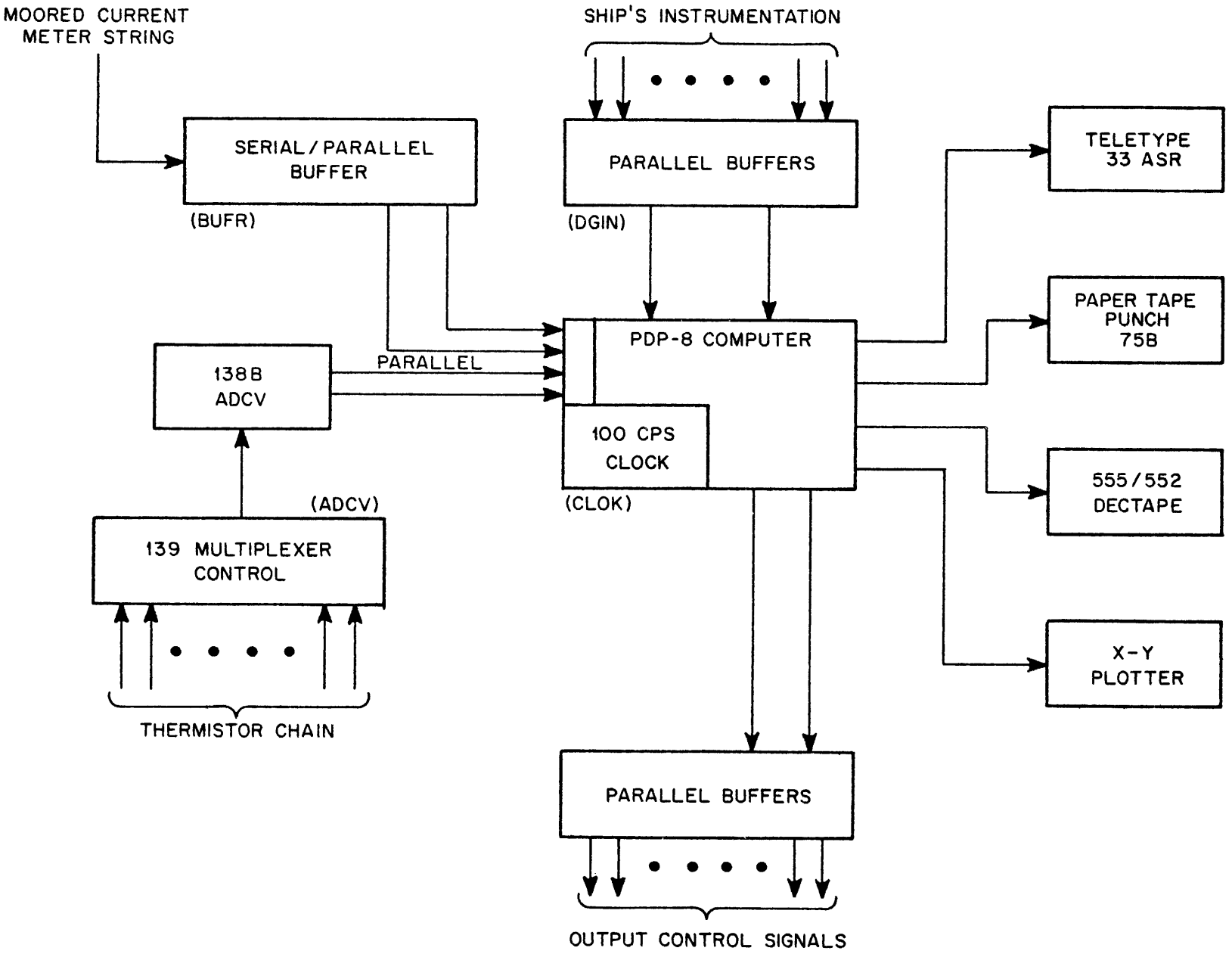


Figure 8 Equipment Configuration for Problem 2

THE USE OF A DIGITAL COMPUTER IN THE SETUP, CHECKOUT, AND MAINTENANCE OF HYBRID COMPUTER SYSTEMS

by

Stephen F. Lundstrom

Applied Dynamics, Inc.
Ann Arbor, Michigan

Abstract The use of a digital computer as a tool of the engineer and technician in setup, checkout, and maintenance of hybrid computer systems has been prompted by the development of a large-scale analog system. The SPOUSE (Stored Program OUtput-Setup Equipment) and MAID (Maintenance AID) systems have been developed at Applied Dynamics, Inc. for such a use.

SPOUSE is a hardware-software combination which is used to control the automatic problem setup and checkout specified by the operator. The software is problem-oriented and may be utilized in a hybrid operating system. A SPOUSE system which controls a multiple analog console hybrid facility with multi-teletype inputs will be discussed.

MAID is a hardware-software combination which is used as a diagnostic tool during checkout of the analog computer subsystem. MAID uses many of the software and hardware elements of SPOUSE.

One of the recent advances in computation involves the development of large-scale hybrid computer systems. This development has come from the basic limitations of analog computers and digital computers. The hybrid configuration combines the best characteristics of the analog and digital computers into one system. A major disadvantage has remained with the analog subsystem. Problem setup and equipment checkout on the analog computer has required manual procedures. The following is the approach taken by Applied Dynamics, Inc. to solve these problems using the SPOUSE (Stored Program OUtput-Setup Equipment) and MAID (Maintenance AID) systems.

The type of hybrid computer considered in this paper will be that consisting of an analog computer, a digital computer, and an interface between the two. Figures 1 and 2 show the configuration of a typical large hybrid computer system. Figures 3 and 4 show a configuration of a typical small hybrid computer system. A basic familiarity with the digital computer will be assumed here; however, a description of the analog computer follows.

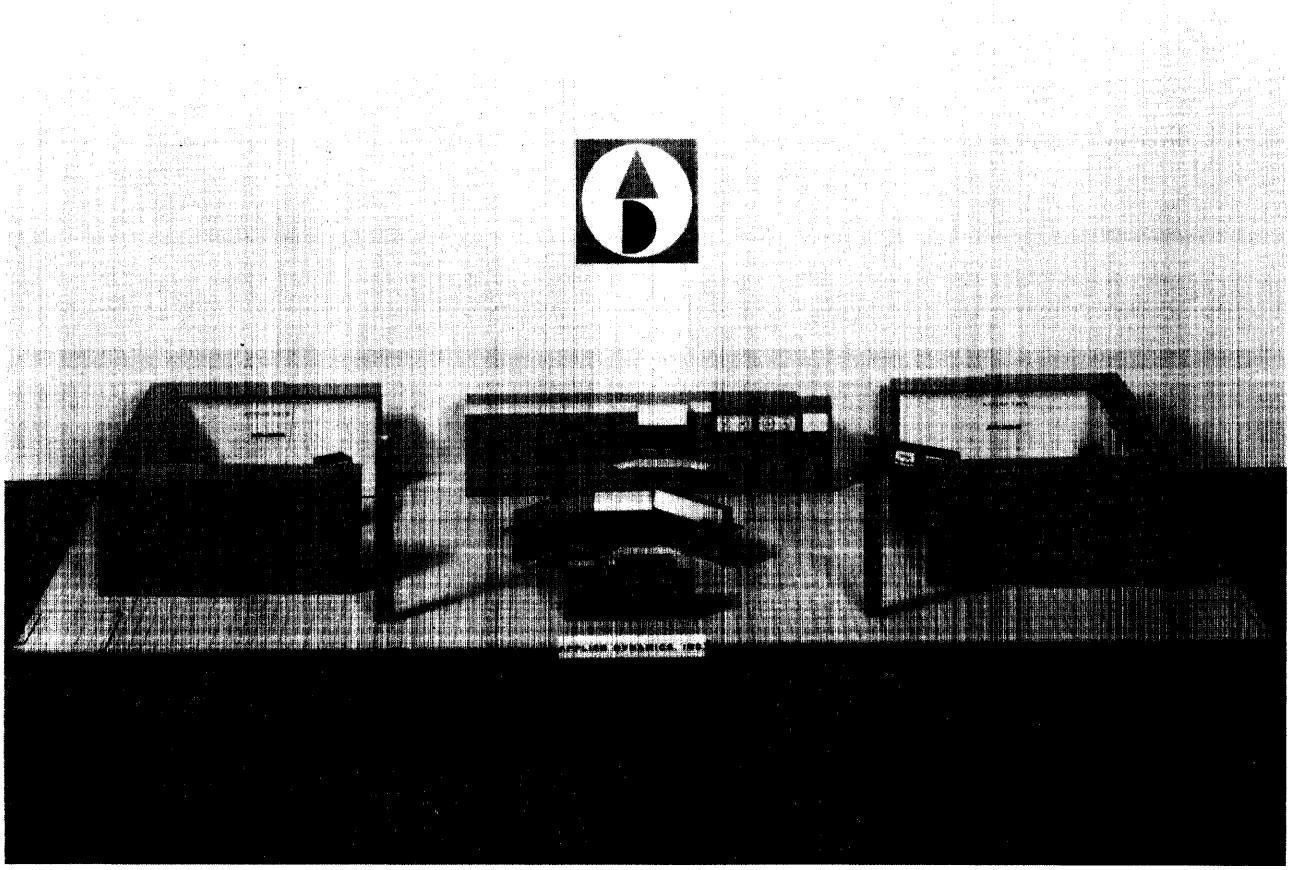


Figure 1 Model of Hybrid System with Four AD-256 Consoles and PDP-6

The analog computer is basically an electronic device that solves differential equations by setting up an electrical analog to the differential equations. The electrical elements required to set up such an analog are many, such as: coefficient devices, summers, integrators, differentiators, multipliers, and many other special-purpose elements. All elements have the inputs and outputs terminated on a patchboard which is used by the programmer to connect elements appropriately. To allow ease of connecting the analog system to a digital system, a number of hybrid elements such as comparators, A/D, D/A, and track-hold devices are usually included. Often a group of patchable logic elements are also included.

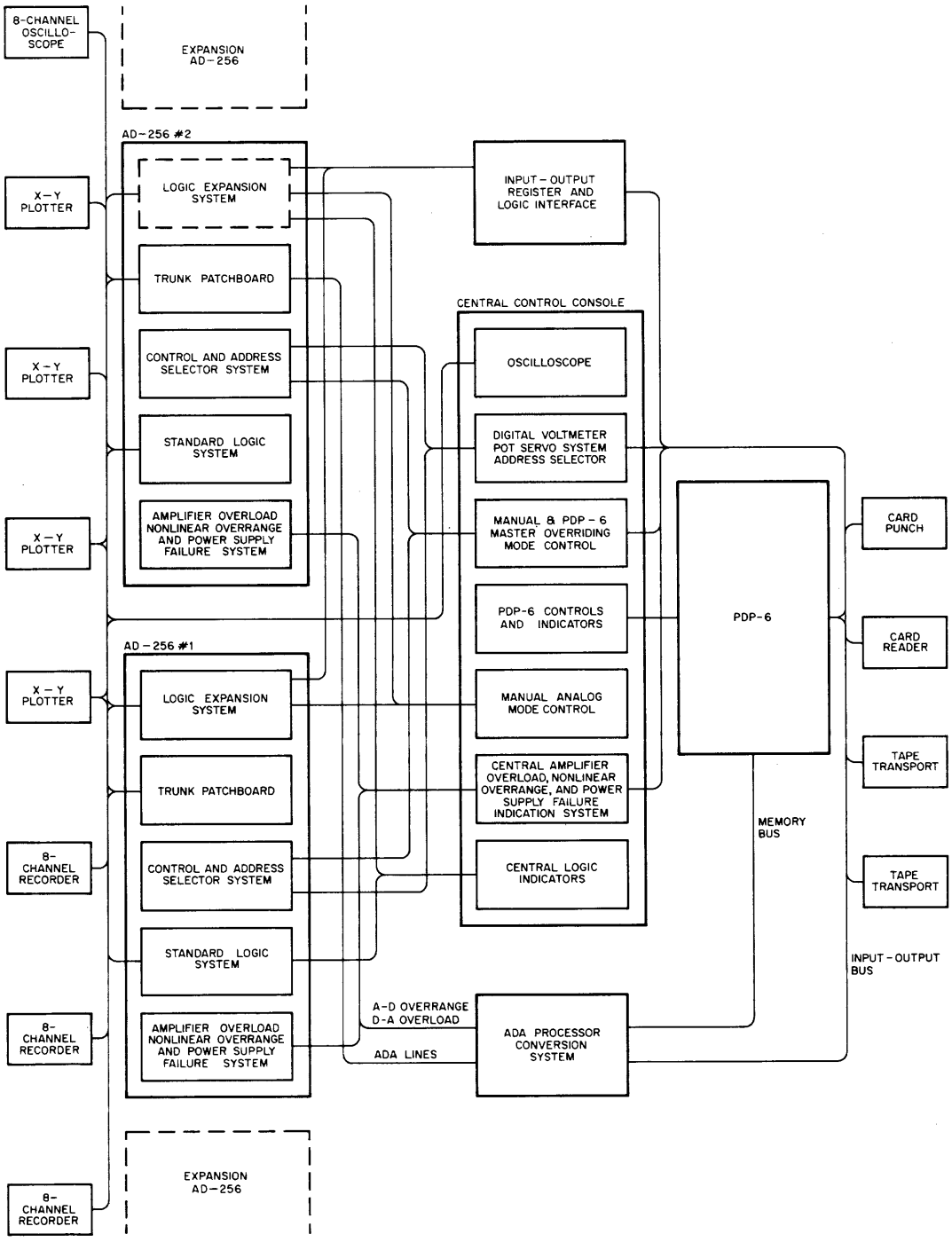


Figure 2 Functional Interconnection of Hybrid System

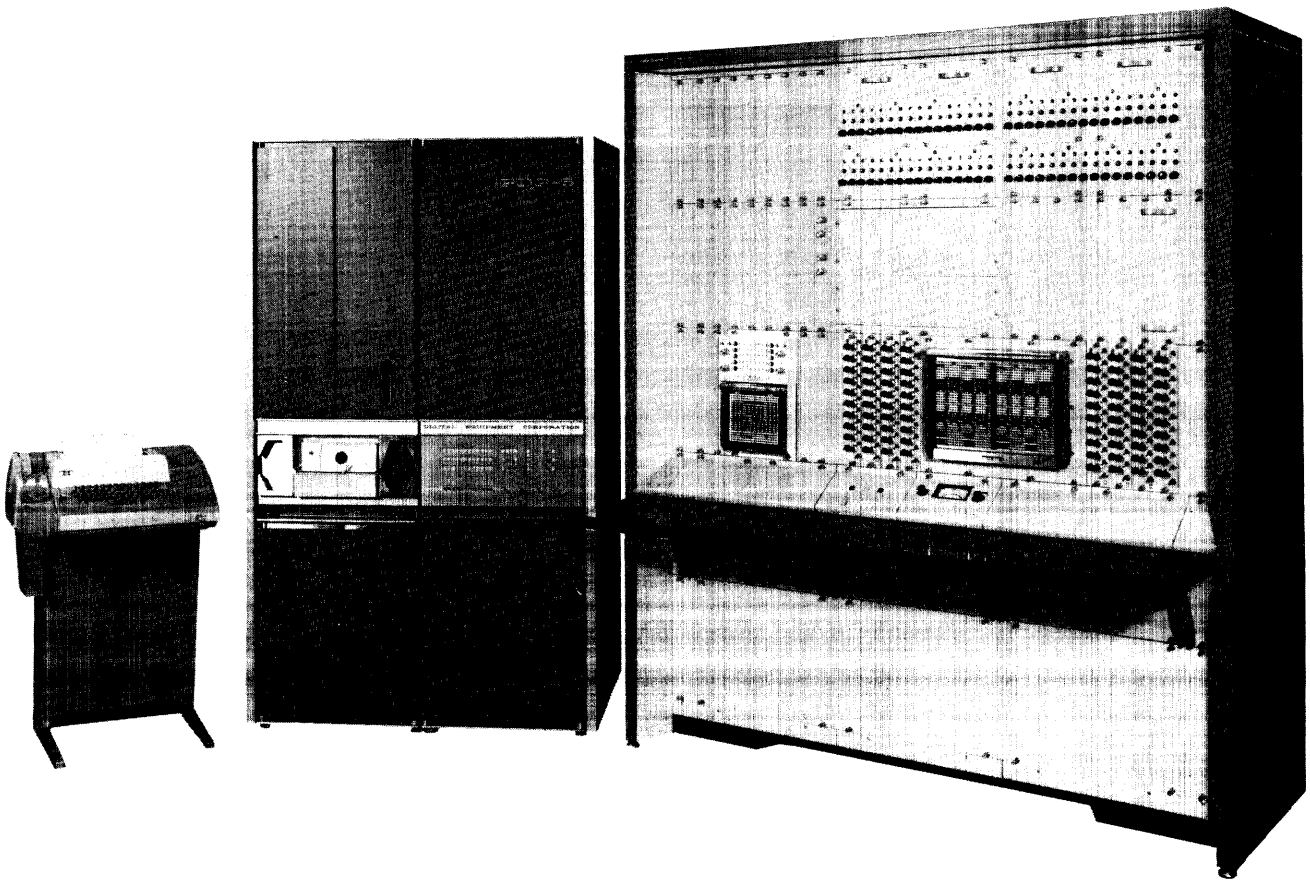


Figure 3 AD-80 and PDP-5 Hybrid Computer System

A typical large-scale analog console with a basic SPOUSE control system using a PDP-5 is shown in Figure 5. The coefficient devices in this computer are mostly servo-set potentiometers. The functions of the integrators (operate, hold, initial condition) may be under individual control of the patchable logic provided in the console. Facility to interconnect many consoles is provided through patchable trunks. Each of the computing elements is assigned an address, much as addresses are assigned to each word in the core of a digital computer. These elements may be accessed through an addressing system on the console and their voltage displayed on an oscilloscope, a voltmeter, recorder, or other output device.

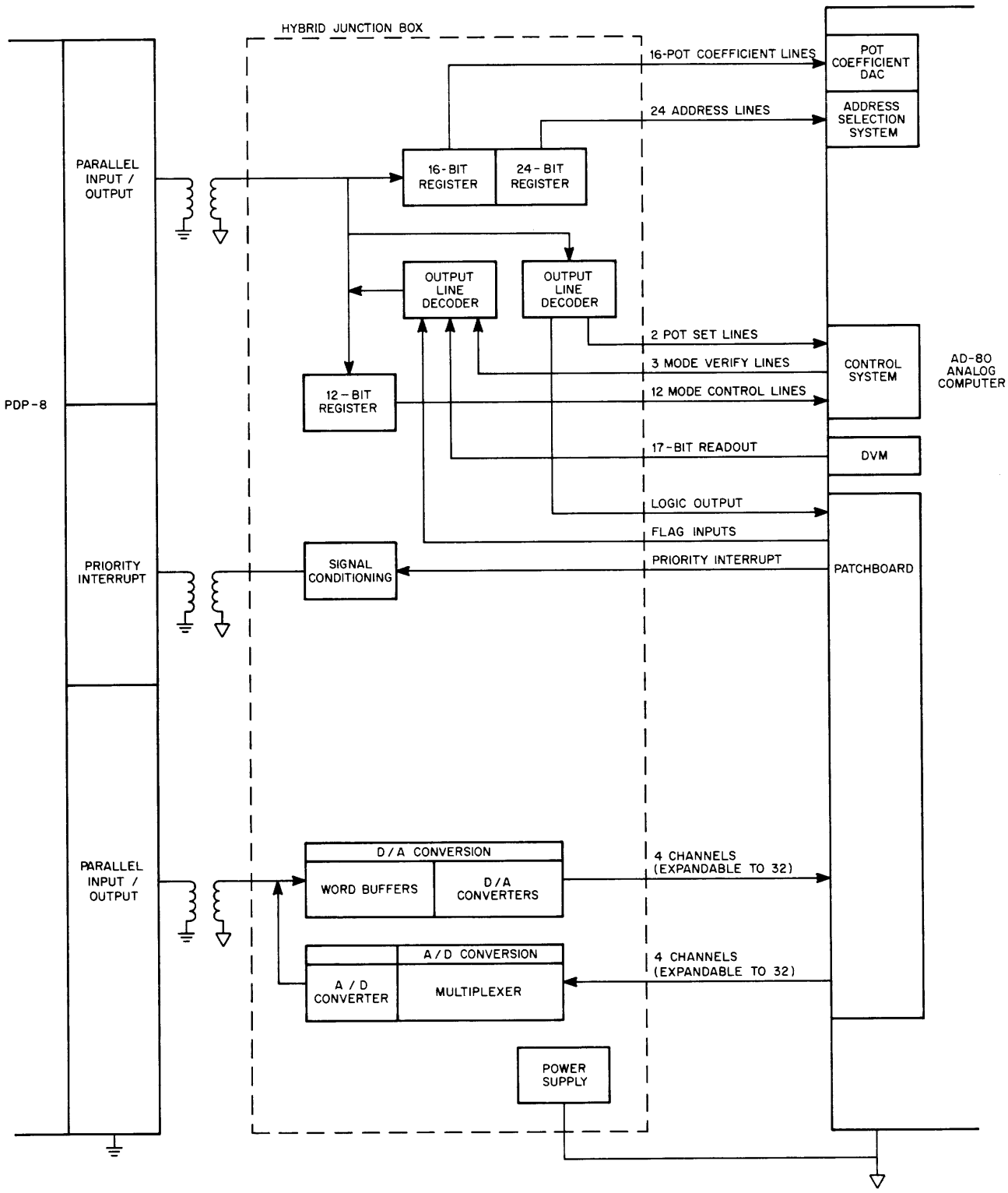


Figure 4 Linkage Subsystem of a Small Hybrid Computer System

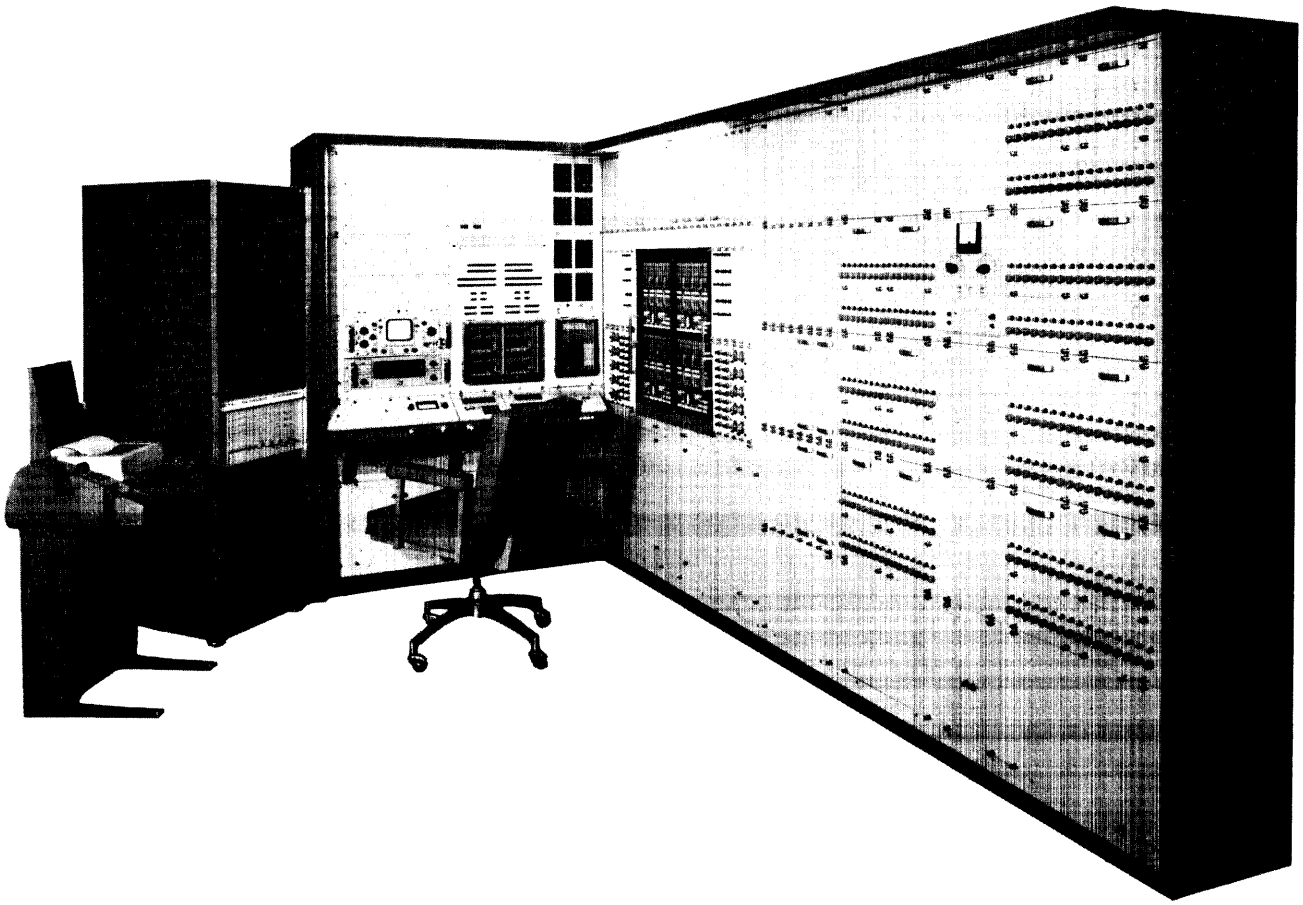


Figure 5 AD-256 and PDP-5 Hybrid Computer System

After patching the problem, the engineer must set all coefficients and initial conditions. The output of each element must be checked to verify proper initial operation. This setup and checkout procedure is time-consuming and subject to human error and oversight. To verify proper operation of the equipment, the technicians are required to perform various diagnostic tests, also involving manual check of the outputs of each element in the computer for known inputs. Again, this procedure is slow and subject to operator error. To aid the engineer and technician in their setup, checkout, and maintenance chores, Applied Dynamics, Inc. has developed the SPOUSE (Stored Program OUtput-Setup Equipment) and MAID (Maintenance AID) systems. Both systems are a combination of hardware and software. The hardware required is simply the basic elements in the interface of the hybrid computer system and a digital computer. The software required consists, for the most part, of a library of hybrid operatins system sub-routines and appropriate system programs for control purposes.

SPOUSE consists of a digital computer, a SPOUSE system program in the digital computer, a library of hybrid and input-output subroutines, the input-output devices connected to the digital computer, and an interface between the digital computer and the analog computer. The current SPOUSE configuration allows up to ten analog consoles to be controlled. The hybrid interface necessary for SPOUSE consists of an analog-digital (A/D) channel, a digital-analog (D/A) channel, and various control lines. A system currently being installed by Applied Dynamics, Inc. uses a PDP-8 digital computer to control two AD-256 consoles, each with Teletype input-output communication. The SPOUSE software discussed below is designed to handle multiple console input-output.

The SPOUSE system software is an interpreter oriented to accept instructions in a form familiar to those using analog computers. It allows the user to interrogate any point on any console from the keyboard he is using by typing the appropriate instructions. In addition, he may prepare a paper tape for input through one of the system paper tape readers. SPOUSE can give automatic typeout of voltage values or of errors from specified voltage values or ground. The user is able to set any servo-controlled pot from the keyboard and has the option of requesting a readback of the pot set to allow verification of the problem setup.

One of the obvious advantages of a programmed system is the ease of carrying out various courses of action depending on conditions. SPOUSE allows the user to specify a number of optional courses of action in case of various conditions arising. One of the options provided is that if the error on a particular reading is greater than the error allowable (as specified by the user), the SPOUSE system will halt further processing until the user gives an indication that he has observed the error. A similar option is available for the condition of pot-set failure. Another option allows the user to request a paper tape record of his operation. Such a record is in a format compatible with the SPOUSE system and may be used at a later date to reset an entire problem. The user may request the deletion of typeouts of voltages on elements meeting the specifications as another option.

The instructions recognized by the SPOUSE interpreter are in three general classes: operating instructions, specifications and requests, and utility instructions. The operating instructions are instructions which actually perform the setup and readout of the analog console. Instructions to read-out the voltage on individual elements, sequential blocks of elements, or random

blocks of elements are provided in this class. Coefficient setting and the check of the outputs of elements by comparison against specified values are also carried out by instructions in the operating instruction class. The specifications and requests are those instructions which are used by the SPOUSE interpreter to set up flags and comparison switches internally to satisfy the various options requested by the user. The utility instructions are provided for the convenience of the octal dumps, binary and RIM paper tape reading and punching, and the typein of octal programs, all under keyboard control. When SPOUSE has more than one keyboard for input, a special class of instructions allows an operator to lock out other keyboards and connect particular analog consoles to his keyboard only, as well as to release all possible constraints.

Many of the subroutines used by the SPOUSE interpreter may be used by hybrid programmers. All the necessary routines for A/D and D/A conversion and discrete controls are provided in the library. Also programs to perform functions such as reading and typeout of the A/D channels and comparison of the A/D channels to specified values are included. As mentioned earlier, one of the disadvantages of analog computers has been the lack of automatic diagnostic procedures. Manual checkout techniques have proved to be time consuming and prone to human error and oversight. Applied Dynamics, Inc. has developed the Maintenance AID (MAID) system to provide a nearly complete automatic diagnostic check on the analog console. The MAID system consists of a checkout maintenance system (CMS) and the SPOUSE system described above. The CMS consists of two special prewired analog patchboards along with two prewired logic patchboards and a trunk patchboard. MAID provides a functional check on nearly every analog component in the analog console. Continuity checks for each component and each patchboard hole, static accuracy checks and dynamic gain checks for analog components and operational tests for analog components and operational tests for hybrid elements are performed. MAID, through the use of SPOUSE, controls the checkout of the analog system automatically. A standard system test tape consisting of SPOUSE instructions and some special programs is used for control. Test results are automatically confirmed and printed out and tapes are punched for verification and comparison.

The following mentions some typical areas of hybrid computer application. This discussion conveys the importance of hybrid systems in order that the preceding discussion of the SPOUSE and MAID systems will not be considered ends in themselves. They are definitely expeditionary devices to increase the percentage of time which the hybrid system can devote to hybrid computation.

Sampled data or computer-controlled system simulations requiring the combination of continuous and discrete variables or data is one typical class of problems which could be solved easily on hybrid computers. For example, simulation of a large power distribution system which is under control of a digital system. Another class of problems is systems of differential equations to be solved at very high speed for prediction or optimization. The high-speed simulation of a space-craft on an interplanetary mission would be an example of this class, as well as the simulation studies involving systems with both high-and low-frequency characteristics. Notice that on an interplanetary simulation exercise, the absolute position is a function that changes slowly but which must be known accurately (this is computed in the digital subsystem). The structural vibrations caused by mid-course maneuvers are high frequency and must, in general, be calcu-lated on the parallel analog subsystem to keep the time base as high as possible. Hybrid com-puters are also useful in the study of systems with transport delays because of the ease of creating various length temporary storage tables in the digital subsystem for variable length delays. Sys-tems requiring on-line, real-time statistical analysis, data filtering, smoothing and editing, and systems described by very large simultaneous sets of ordinary differential equations are also among those systems which find hybrid computation techniques valuable in their analysis. Notice that the high accuracy and logical capability of the digital subsystem and the high speed and wide bandwidth of the analog computer complement extremely well in all of the cases mentioned above.

To conclude, it should be mentioned that both the SPOUSE and MAID systems are operational systems. As with all systems involving software, they are undergoing continuous revision to expand their capabilities and their flexibility. This revision will undoubtedly soon lead into areas not mentioned herein. However, the general use to speed up and make more accurate the setup, checkout, and maintenance of hybrid computer systems through the use of digital computers will remain paramount.

APPENDIX 1

SOUND-OFF SESSION

Moderator: J. Ridgeway

Panel Members: L. Hantman, J. Shields, C. Frazier, D. Cotton, T. Johnson, and J. Jones

Because of the difficulty in recording the questions and answers during the discussion, we are not able to report the session verbatim. However, we did not want to omit the information and have included the questions and answers as close to the actual panel discussion as possible.

Question No. 1

Is the Engineering Department planning on doing anything about lowering the noise level on the 570 Tape Transport?

Answer

We have put in a smaller compressor which has cut down on the noise considerably. We also put in some mufflers on the overflow valves. Presently, however, because it really isn't causing that much of a problem, our Engineering Department is not pursuing it any further.

Question No. 2

I was wondering about the compatibility between the PDP-4 and PDP-7 programs. We are having trouble assembling the PDP-7 programs on our PDP-4.

Answer

We are currently involved in transforming PDP-4 programs to PDP-7 and have assembled 4 and 7 programs on both the PDP-4 and PDP-7. Specifically, the only thing that has to be changed are the timing loops and the FIODEC to ASCII conversions. We are also in the process of changing all of the documentation for the PDP-7 so that there will be a complete new PDP-7 library. In regard to assembling, there is a switch note that is wrong on the directions in the PDP-7

Assembler Manual. An instruction in the front of the manual does not agree with the comment in the back of the manual regarding the use of switch 9 and 10 for the input of ASCII and FIODEC code. Bit 10 should be used for FIODEC and bit 10 up for ASCII. Using FORTRAN, bit 10 has the same function in the Compiler and Assembler. In the Compiler, bit 9 has the same function (up for ASCII, down for FIODEC) for output.

Question No. 3

What is the status of PDP-4 documentation? PDP-4 program manuals?

Answer

We recently rewrote the documentation for the Editor, DDT, FORTRAN, and the Assembler. We printed different manuals for the PDP-4 and PDP-7 basically because of the FIODEC and ASCII code difference. We have a large supply of manuals for both machines, which are available upon request.

Question No. 4

One of the major problems with FORTRAN is the way it is set up. There may be empty parts of core, but they are usually in the middle and we have no way of getting to them.

Answer

FORTRAN is now in the process of being cleaned up. One problem some users have is that they do not know where the low end of FORTRAN is. Right now, I believe, FORTRAN stops at 12000 on the way down and that is the beginning of the Common Storage Area. Anything you define as being in Common will work its way down from location 12000 which is defined as End.

Question No. 5

We are having some difficulty reading in our programs for DECtape; what do you recommend?

Answer

The DECTRIEVE Program is designed to write out over itself and to read in over itself. No matter where your program is in memory, if you want to write it out in one continuous bunch

(I'm not sure that this is always possible) you can use the DECTRIEVE Program to get something in or out. They will read in complete core, that is from location 1 to 17777. As a matter of fact, DECTRIEVE for the PDP-7 will eventually transfer information in and out of extended memory. The write-up for DECTRIEVE also states that all of core beginning from location 0 to 1 can be read in.

Question No. 6

Are there going to be any more programming classes on the PDP-1?

Answer

There are no plans for PDP-1 programming classes at this time; however, we suggest that questions regarding PDP classes be directed to Mr. Robert Pate, who is in charge of customer training at Digital.

Question No. 7

Does DEC have any intention of supporting a sophisticated assembler for the medium and larger machines like the PDP-4 and PDP-7, for example: Midas, Macro, Conditional Assembler or Relocating Assembler?

Answer

At present, there isn't any plan for a sophisticated assembler on either the PDP-4 or 7. There is a lot of interest on the outside and internally by our own programmers, but we haven't decided to develop one. On matters such as these, it is important that the users express their interests regarding software and hardware requirements. DEC is interested in your comments and would be helpful in developing future work.

Dr. Oliver pointed out in his luncheon talk that special projects such as developing a sophisticated assembler, compiler, etc. can be one of the functions of the users' group, and DECUS can serve as a means for assembling those people interested in using such a compiler, etc. and then sponsor it.

Question No. 8

How do I know I have the latest DEC library tape?

Answer

The practice has been that when a revision is made in the program the tape goes to the library with the proper documentation and the librarian is to destroy or not issue the previous copy. Periodically, customers are notified by mass mailings of changes or revisions in the DEC library programs. We are also considering using the DECUSCOPE as a means of informing the users of the status of software at DEC.

Question No. 9

Is there any internal documentation for the PDP-5 FORTRAN System?

Answer

It is presently being written for the PDP-8. The internal documentation is being done by Larry Portner. We now have a DECtape system and we have a FORTRAN on the PDP-5/8 which is usable on high-speed readers and punches.

Question No. 10

Are there any plans in the future of DEC coming up with an updated version of the PDP-1?

Answer

So far, the answer appears to be no. I might say that we might fill what appears to be a gap in the product line with either two considerations; a souped-up PDP-7 or a stripped down PDP-6. It could be done rather easily, however we don't see the market place for it at this time. The PDP-7 pretty well does what the PDP-1 did at the same speed; admittedly, it does not begin to have the same instruction set or the I/O hardware that the PDP-1 had. One of the things we are trying to do essentially is to help PDP-1 users to develop an easy transition between the PDP-1 and PDP-6. This is one of the places our interest has turned. Of course, for obvious reasons, some of our -1 users are about to get a -6. This looks like the way many of our -1 users would like to go.

Question No. 11

I understand there is a market for second-hand PDP-1's. Can you tell me what the selling price would be?

Answer

On the basis of a couple of quotes we made recently on used PDP-1's, a PDP-1 with 4K of memory would cost somewhere in the neighborhood of $50,000 or $60,000 reconditioned.

APPENDIX 2

SPRING SYMPOSIUM PROGRAM

Place: Center for Cognitive Studies
William James Hall
Harvard University

Date: May 20-21, 1965

PROGRAM

THURSDAY

8:30 - 9:20 Registration and Coffee

9:30 Opening of Meeting

9:35 Welcome - Dr. Donald Norman, Harvard University

9:45 Guest Speaker - Harlan Anderson, Digital Equipment Corporation

10:15 "Computer-Aided Design"
Professor Steven A. Coons, Massachusetts Institute of Technology

10:45 "DECADE - Computer-Aided Design on Low Cost Hardware"
Charles W. Stein, Digital Equipment Corporation

11:15 "Use of a PDP-5 for the Collection of Mossbauer Experimental Data"
Ronald H. Goodman, Department of Mines and Technical Surveys (Canada)
John E. Richardson, Digital Equipment of Canada

11:45 Lunch
Luncheon Speaker - Dr. James R. Oliver, Dean of the Graduate School, University of Southwestern Louisiana

1:30 "A High-Speed Man-Computer Communication System"
Earl H. Brazeal, Jr. and Taylor L. Booth, University of Connecticut

2:00 "PDP-5 Projects at Lawrence Radiation Laboratory"
Sypko Andreae, Lawrence Radiation Laboratory (Berkeley)

2:30 "Computer Systems for Recording, Retrieval and Publication of Information"
Richard J. McQuillin and Joseph T. Lundy, Inforonics, Inc.

3:00 "The Savage System: A Monitor System"
Bernard I. Savage, Consultant

3:30 Coffee

3:45 Tour and Demonstrations - Center for Cognitive Studies, William James Hall

4:00 Business Meeting

5:00 - 6:30 Social Hour

FRIDAY

8:30 - 8:45 Registration and Coffee

8:55 Opening of Second Day's Session

9:00 "PDP-5/PDP-1 On-Line Data Collection and Editing System"
Pablo Larrea, Princeton-Pennsylvania Accelerator

9:30 "A Version of LISP for the PDP-6"
William A. Martin, M.I.T. Project MAC

10:00 "A Report on the Implementation of the Keydata System"
Charles W. Adams, Adams Associates, Inc.

10:30 Tours and Demonstrations - Keydata Facilities and Project MAC

12:00 Lunch - Harvard Faculty Club
Sound-off Session

2:00 "A Chalk River PDP-5 Pulse-Height Analyzer"
Dallas C. Santry, Chalk River Nuclear Laboratories (Canada)

2:30 "A Description of the PEPR System"
David Friesen, Laboratory for Nuclear Science, M.I.T.

3:00 "DEXTER - The DX-1 Experimenters Tape Executive Routine"
Jerome Cohn, Wolf Research and Development Corporation

3:30 Coffee

3:45 "The Use of a Small Computer With Real-Time Techniques for Oceanographic Data Acquisition, Immediate Analysis, and Presentation"
Robert M. O'Hagan, Digital Equipment Corporation

4:15 Film: "The NRU Atomic Reactor Control Computer Experiment - Part III Display Techniques"

4:35 "The Use of a Digital Computer for Automatic Setup, Checkout, and Maintenance of a Hybrid Computer System"
Stephen F. Lundstrom, Applied Dynamics

APPENDIX 3

AUTHOR AND SPEAKER INDEX

APPENDIX 4

ATTENDANCE

Name	Affiliation
Akolk, Friedmann	Deutsches Elektronen-Synchrotron (DESY)
Alexander, John R., Jr.	United States Air Force, WPAFB, Ohio
Anderson, Harlan E.	Digital Equipment Corporation
Andreae, S.W.	Lawrence Radiation Laboratory
Avery, D.	Digital Equipment Corporation
Ball, N. Addison	Department of Defense
Balzer, Frank L., Jr.	United States Air Force, Bedford
Bates, Edgar A.	Stromberg-Carlson Corporation
Benton, D.J.	Foxboro Company
Bonar, L.C.	Massachusetts General Hospital
Booth, Taylor L.	University of Connecticut
Bove, Roger E.	Air Force Cambridge Research Laboratory
Bower, Charles	Boeing Company
Bradley, Joyce	Harvard University
Bradshaw, John H.	Boston College
Brazeal, Earl H., Jr.	University of Connecticut
Brown, J.B.	Bolt, Beranek and Newman
Campbell, Douglas	Foxboro Company
Canning, C. Norman	Digital Equipment Corporation
Chase, Nick	Adams Associates
Cohn, Jerome	Wolf Research and Development Corporation
Colomb, Robert M.	Systems Research Laboratories
Coons, Steven	Massachusetts Institute of Technology
Cossette, Angela	Digital Equipment Corporation
Cotton, David Baer	Digital Equipment Corporation
Cullen, T.R.	Itek Corporation
DeStefano, Anthony	Inforonics, Incorporated
Fable, W.	Systems Research Laboratories
Finch, Geoffrey	DEC (UK) Ltd.
Fishman, Jack J.	Foxboro Company
Foxworthy, Vincent	Systems Research Laboratories
Friesen, David	Massachusetts Institute of Technology
Gaines, Eugene C., Jr.	Consultant to Adams Associates
Gibson, Richard S.	Decision Sciences Laboratory
Gillis, D.	University of Glasgow
Gilmore, John T., Jr.	Adams Associates
Glaser, Robert	University of Pittsburgh
Glazer, Eli	Brookhaven National Laboratory

Name	Affiliation
Goldstein, Ira	Decision Sciences Laboratory
Goodman, R.H.	Department of Mines & Technical Surveys
Graetz, J.M.	
Greatorex, Frank S., Jr.	Adams Associates
Grover, C. Stuart	Digital Equipment Corporation
Handy, Roger	Digital Equipment Corporation
Hantman, Leonard M.	Digital Equipment Corporation
Hermistone, John S.	Adams Associates
Hornbuckle, Gary D.	University of California
Hubbard, Howard	Digital Equipment Corporation
Hurlbut, Charles E.	Itek Corporation
Hurley, Nancy L.	Bolt, Beranek and Newman, Inc.
Hyman, Harris	
Johnson, Ted	Digital Equipment Corporation
Keim, Robert D.	Wolf Research and Development Corporation
Kent, Douglas A.	Lawrence Radiation Laboratory
Killeen, Peter R.	Harvard University
King, Murray Vernon	Massachusetts General Hospital
Larrea, Pablo	Princeton-Pennsylvania Accelerator
Levy, Harold	Scientific Engineering Institute
Liggins, Lorraine	Massachusetts General Hospital
Lithgow, Webster	Dekko Films, Inc.
Longee, Steve	Raytheon
Lund, Paul	Lawrence Radiation Laboratory
Lundstrom, Stephen F.	Applied Dynamics, Inc.
Mahoney, Gerald E.	Foxboro Company
Mantel, Charles	Digital Equipment Corporation
Martin, William A.	Massachusetts Institute of Technology
Maxcy, Robert F.	Digital Equipment Corporation
Mayer, Sylvia R.	United States Air Force
McGoldrick, C.C.	Decision Sciences Laboratory
McLaughlin, John	Radio Corporation of America
McQuillin, Richard J.	Inforonics, Inc.
Meltzer, J.	Bell Telephone Laboratories
Miller, Elaine	Massachusetts Institute of Technology
Miller, George A.	Harvard University
Morello, Marie V.	Adams Associates
Newman, E.	Harvard University
Newman, Elsa	DECUS
Nicol, Elizabeth H.	United States Air Force, Bedford
Norman, D.A.	Harvard University
O'Hagan, Robert	Digital Equipment Corporation
Oliver, James R.	University of Southwestern Louisiana

Name	Affiliation
Paquette, Gerard A.	United Aircraft Research Laboratories
Pirtle, Mel	University of California
Ragsdale, Ronald G.	University of Pittsburgh
Richardson, J.E.	Digital Equipment of Canada, Limited
Ridgeway, John A.	Digital Equipment Corporation
Rudloe, Harry	Harvard University
Ryan, Robert H.	Foxboro Company
Sampson, Philip B.	ESD United States Air Force, Bedford
Santry, Dallas C.	Chalk River Nuclear Laboratories
Savage, Bernard I.	
Sharpe, Stuart	United Aircraft Research Laboratories
Shields, John	Digital Equipment Corporation
Smith, David M.	United States Air Force, Laurel, Maryland
Sordillo, Donald	Harvard University
Stapleford, Kenneth R.	Foxboro Company
Stedman, Jon D.	Lawrence Radiation Laboratory
Stein, Charles	Digital Equipment Corporation
Stolz, Walter	Harvard University
Storey, John R.	Defence Research Board
Sulkowski, F. J.	United States Air Force, Bedford
Tenenbaum, Joel	Harvard University
Tillson, John T.	Foxboro Company
Wagner, David E.	U.S. Naval Weapons Laboratory
Walter, Charlton M.	Air Force Cambridge Research Laboratory
Weene, Paul	Decision Sciences Laboratory
Weisberg, David E.	Adams Associates
Wells, N.S.	Atomic Energy of Canada, Limited
Westfield, Robert L.	Decision Sciences Laboratory
Wisonaty, M.	Wolf Research and Development Corporation
Wolfberg, Michael S.	University of Pittsburgh
Wood, James M.	Computer Usage
Wood, Thomas F.	Aragon Geophysics
Woodmansee, G.H.	Edgerton, Germenhauser and Grier
Zaradil, Robert J.	AFTAC, Alexandria, Virginia
Zurlinden, Donald H.	Lawrence Radiation Laboratory