A Shared Memory Computer Display System

N. A. BALL, MEMBER, IEEE, H. Q. FOSTER, W. H. LONG, MEMBER, IEEE, I. E. SUTHERLAND, MEMBER, IEEE, AND R. L. WIGINGTON, MEMBER, IEEE

***Abstract*—The display system discussed in this paper was implemented to provide a flexible test bed for man-machine interaction. The discussion includes both hardware and software design concepts, and concludes with an evaluation of the system based on operating experience. The two significant organizational features of the hardware are 1) the sharing of memory between a normal computer and a special picture computer (display device), together with 2) the capability of the latter to execute subroutines. The philosophy of design was that hardware should not limit the programmer's flexibility. This idea has been extended to utility software.**

Introduction

A NEED for a general purpose computer-controlled display system arose in a research and development environment which included work on a variety of information processing problems. Coupling a human decision maker with computer processes via textual and graphic displays seemed to be appropriate in several processes involving data review and manipulation, textual and diagrammatical editing, and monitoring and control of computer processes. The display system[1] discussed in this paper was implemented beginning December, 1963, in order to provide a flexible test bed for man-machine interaction.

The philosophy of design throughout was that hardware should not limit the programmer's flexibility. All hardware conditions should be available to the programmer without restrictions which would force him to use any of the equipment in a preconceived fashion. It is the purpose of this paper to describe some of the ways in which this result was achieved. The discussion includes examples of software concepts made possible by the hardware design and descriptions of several utility programs.

The display system, delivered in August, 1964, consists of a modified Digital Equipment Corporation Type 340 Display attached to a PDP-4 computer. The PDP-4 computer, which serves the immediate computational needs of the user, is connected to a larger computer, a CDC-1604A. Complex computations involving floating point arithmetic or requiring considerable memory may be performed by the 1604A. The small computer and the display, together with manual intervention devices (knobs, pushbuttons, light pen, etc.), provide a general purpose facility for man-machine communication.

There were several significant constraints on the design and implementation of the system. The new equipment and software were to be integrated with an existing host computer using a non-time-sharing job shop monitor. The system was to allow operation:

1) as a special I/O complex for the host computer with control residing in the host computer
2) independent from the host computer for problems of reduced magnitude and complexity, and
3) with some form of job insertion from the new display system into the job shop operation.

Also, the cost and time to implement the system had to be competitive with the acquisition of any standard display facility with internal memory.

Display System

The two significant organizational features of this display are the sharing of memory between a normal computer and a special picture computer, together with the capability of the latter to execute subroutines.

The display system may be divided into three major physical parts: the control computer, the display unit, and the operator console. (See Fig. 1.) In addition, there are the interferface between the host and the display system and several I/O facilities local to the display system.[2]

Control Computer

The control computer provides both system control and system memory. It is a PDP-4, a small, binary, parallel, single-address machine manufactured by the Digital Equipment Corporation (DEC).[3] The memory consists of 8192 words of 18 bits, each with an 8 μs cycle time. Three bi-directional real-time data channels and flexible I/O transfer operations through the accumulator provide coupling to other elements of the system.

Two types of internal timing devices are included. One is a free-running periodic program interrupt which

Manuscript received February 21, 1966.
N. A. Ball, H. Q. Foster, I. E. Sutherland, and R. L. Wigington are with the Department of Defense, Washington, D.C.
W. H. Long is with the Digital Equipment Corporation, Maynard, Mass.

[1] The display has a short name: SCHOOLER. The reader should not attribute any implied meaning to this name; it was assigned in much the same fashion as some programmers choose statement labels.

[2] These include a four-drive DEC tape system, a 300 character per second paper tape reader, a Polaroid camera, and a computer-controlled frame-by-frame movie camera.

[3] Digital Equipment Corporation, Manual PDP-4, F. 45, December 1962.

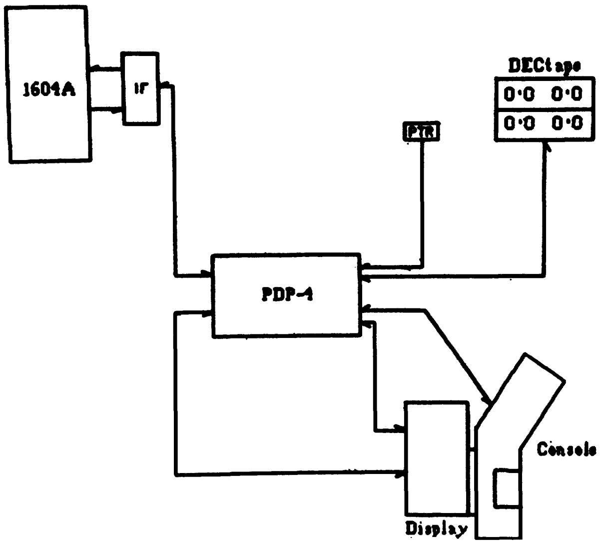


Fig. 1. System block diagram.

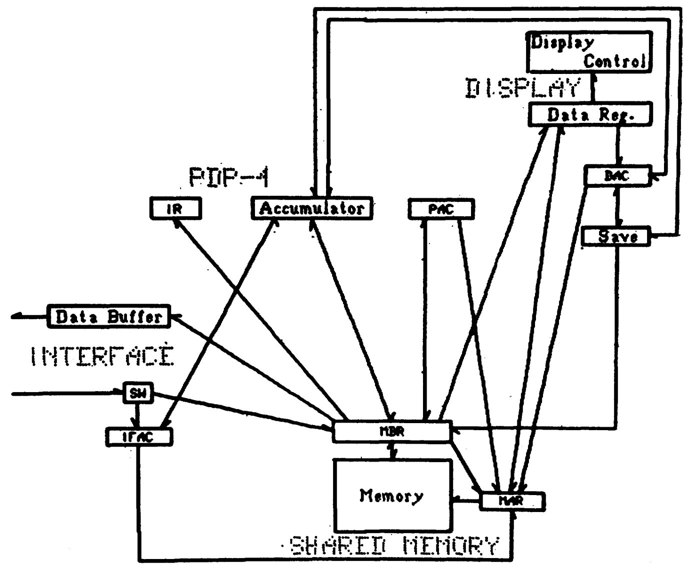


Fig. 2. Interconnection of major registers.

is screw-driver-adjustable over the range of 20 to 10 000 c/s. It is normally set at approximately 100 c/s and may be used to initiate console scanning operations. For example, the EXECUTIVE subroutine, discussed later, is an interrupt-actuated display system monitor which depends on the periodic interrupt to initiate regular checks on the system status. The second is a reserved location in memory which is automatically incremented every 1/60 of a second.

Display Unit

The basic cathode ray tube display is a DEC-340, a magnetically deflected, digitally controlled, CRT display.[4] Pictures are formed on a quantized field of 1024 by 1024 points. Vector specifications of up to 127 dots in $\pm\Delta x$ and $\pm\Delta y$ can be made, and the specified line is generated by a binary-rate-multiplier[5] at the rate of 1.5 μs per point. An incremental specification, for curved lines, causes the CRT to display points adjacent to the preceding point in any of eight directions, at 1.5 μs per point. Arbitrary points can be positioned in 35 μs for an x- or y-axis specification or in 50 μs for a pair (y, x).

The display control unit shares the control computer's memory, as shown in Fig. 2. Thus the two control units must compete for memory cycles. Positioning points and drawing vectors take more time than one memory cycle, so the control computer usually gets the majority of the cycles, even though it has the lower priority. The display then proceeds at a rate determined primarily by the speed of picture drawing, while the control computer retains much of its original memory-access capability. Although memory-sharing in this manner eliminates the necessity for two separate memories, thus reducing memory cost, the real significance is that the display unit and control computer are more tightly coupled than if independent memories were provided.

Since this display can execute subroutines, it is itself almost a computer. Its instructions format is shown in Table I. The subroutine call instruction DJS is equivalent in operation to the 7090 subroutine call TSX. DJS causes control to transfer to the specified address and places the old contents of the display program counter in another register called the save register. The save register plays the same role for DJS that an index register plays for TSX. The deposit save register instruction DDS is analogous to the 7090 instruction SXA, which deposits an index register in the address portion of a word; DDS makes possible multiple levels of display subroutines. A single escape bit in a display instruction causes subroutine exit; this serves the same basic purpose as an indexed jump instruction used for subroutine exit in the 7090.

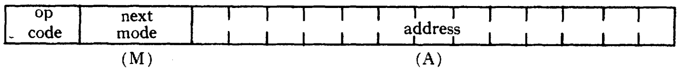
Wherever possible, logical independence of display functions has been preserved. Since the x and y coordinate registers are set independently, it is possible to display rows of dots by first giving the y coordinate for the row and then giving a series of x coordinates. The entire row may be moved by changing the single y coordinate. Since vectors and increments specify motion relative to a given starting point, it is possible to reposition an entire vector or incremental figure by changing only the coordinates of the starting point. Similarly, the intensity and light pen state (on or off) can also be changed independently, which allows selected sections of the picture to be made invisible to the light pen without regard to changes of intensity within the picture. When coupled with the subroutining features, this independence of function makes a very powerful system.

[4] ——, *340 Precision Incremental Display Manual*, H. 340, November 1964.

[5] E. Grabbe, S. Ramo, and D. Wooldridge, *Handbook of Automation, Computation and Control*, vol. 2, New York: Wiley, 1959, pp. 29–35.

TABLE I
DISPLAY INSTRUCTIONS

Control Mode Instructions



DJP: Display Jump: The next display instruction will be taken in mode M from location A.

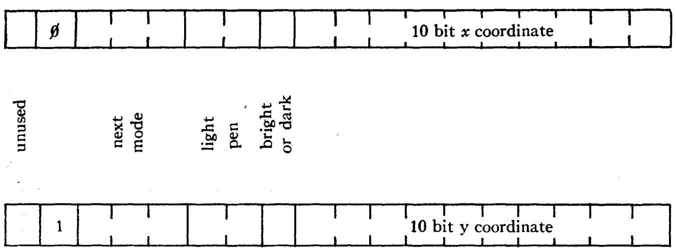
DJS: Display Jump and Save: The next sequential address is stored in the save register and the save flip-flop is set. The next display instruction will be taken in mode M from location A.

DDS: Deposit Display Save Register: The save register is stored (as a Display Jump in control mode) in location A and the save flip-flop is cleared. The next display instruction will be the next sequential word, executed in mode M.

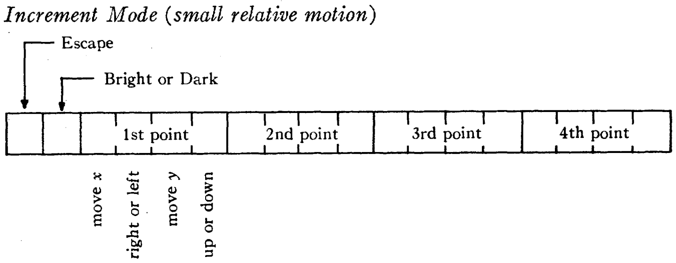
DOP: Display Operate: Operations microprogrammed in the 13 low-order bits are performed and the next sequential instruction is interpreted in mode M. Options are:

Light pen on or off	Escape
Stop	Set scale
Stop and interrupt	Set intensity
Skip	Set or clear temporary dark.

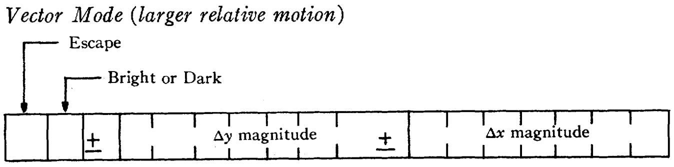
Point Mode (absolute x and y positioning)



Increment Mode (small relative motion)



Vector Mode (larger relative motion)



Vector Continue Mode

Same as vector mode, except line is extended to edge of screen, and escape is forced when the edge is reached.

By using vector and increment mode instructions, one can construct a subroutine which draws any particular shape, for example, an alphabetic character. The position, size, and intensity of a character may be established by setting the appropriate registers before entering the character drawing subroutine. The standard character subroutines cause the x coordinate register to increase by a constant amount equal to the width of a character. Another set of character subroutines decreases the x coordinate, to print from right to left, for easy conversion from binary to other radices. Still another character set has variable width characters. Further, if entire words are to be treated as single characters, such as the words **begin** and **end** in an ALGOL statement, they may be caused to appear by a single call to a subroutine which then calls each letter.

In addition to sharing the same memory, the PDP-4 and the display are closely coupled in other ways. The PDP-4 can start or stop the display at any time. The display can interrupt the PDP-4 to service such error conditions as edge of screen violations. Since the display lacks arithmetic and conditional program transfer capability, it may need to request that the PDP-4 execute processes that it cannot do for itself. For example, assume that an object described by a display subroutine is to be constrained to some bounded subsection of the screen. Suppose also that this object is moved incrementally by the control program and that its absolute location is not easily known to the control program. The display subroutine can request that the PDP-4 check the current object position each time it is displayed.

In many cases, the PDP-4 program is an on-line display program compiler which makes structural changes in the display program. In other simpler cases the PDP-4 program changes display file contents but does not change the file structure.

Console

The specially designed operator's console shown in Fig. 3 contains the following communication devices:

Man to Machine

1) Momentary-contact pushbuttons (108): typewriter-like array (54), keyboard 1 (36), keyboard 2 (18).
2) Analog shaft encoders (6): three-dimensional trackball and three single-axis knobs.
3) Light pen.
4) Rand tablet (GRAFACON Model 1010).

Machine to Man

1) CRT.
2) Lights under program control, mounted in the switches (58).
3) Bell, two-tone gong, telegraph sounder, and other small noise-making devices, operated by relays under program control.

In keeping with the system design philosophy, the programmer defines the meaning of all manual intervention devices; wired-in functions are avoided.

The status of all 108 buttons as seen through contact-bounce filters can be read into the PDP-4 accumulator in banks of 18. The buttons do not cause interrupts. There is no wired connection between the buttons and the lights in them. Fifty-one of these buttons have the physical appearance of typewriter keys, but there is nothing to force the programmer to use them as such. A button may be programmed to complement an internal function either when it is depressed or when it

Fig. 3. Operator console.

is released. The light within the pushbutton may be programmed to show when the internal function is on. A program segment might initiate action only when two or more pushbuttons are down at the same time; it is possible in this way to provide a momentary upper-case shift function. Another segment might continue an action only as long as a button remains down. Providing the computer program with raw information about the positions of these pushbuttons achieves a flexibility appropriate to the experimental application of the display system.

Similarly, the shaft encoder registers provide raw information about the rotation of the shaft encoders. The encoders provide a pulse for each 1/256 of a shaft revolution. These pulses control 5-bit up-down counters which act as buffers for each encoder. The PDP-4 may, at any time, examine the change in the shaft position since the shaft encoder buffer was last read. It is up to the program to make appropriate use of this information. For example, a program can couple the motion of a cursor on the CRT to the motion of the trackball. One such program adds the square of the number of counts received from the trackball during each 25-ms period to the current cursor position, thereby achieving variable speed drive. For slow, fine adjustment of the shaft encoder, the cursor moves very slowly and in direct accord with the motions of the trackball, since at most one count occurs in any 25-ms period. For faster trackball rotation the cursor moves even more rapidly than the increase in rotation speed.

Host Computer

The host computer system is a 1604A job shop operating under a modified COOP monitor.[6] It provides R/D computational support for a variety of projects on the basis of open-shop programming with closed-shop operations. The CDC 1604A is a standard single-address computer with 32 768 words of memory of 48 bits each.[7] Three pairs of input/output buffer channels provide I/O communications through the appropriate I/O controllers. The system has a total of 10 CDC 607 magnetic tape units, a card reader and punch, a printer, and a paper tape punch and reader. A CDC-160 is used for off-line conversion. Available programming languages include ALGOL, FORTRAN 63, SLIP, CODAP II (symbolic assembly language for the 1604A), DYSAC (an analog computer simulator), XAP ("macro-CODAP"), and COBOL.

It is not practical to interrupt the 1604A during the execution of a job, but 45 percent of the daytime jobs run less than $1\frac{1}{2}$ minutes and 60 percent of the jobs run less than $2\frac{1}{2}$ minutes. This is the environment into which the display was introduced.

1604A Interface

The 1604A interface transfers data between the PDP-4 and the 1604A. On the PDP-4 side, the interface accesses memory via the data interrupt system, and on the 1604A side it is connected to a pair of input-output buffer channels. An interface address counter directs transfers to and from the PDP-4 memory, and eight flags provide communication between the two machines. For independent use, the 1604A and the PDP-4 systems can be separated with manual switches.

To the 1604A, the interface appears to be a tape control unit (CDC 1615). An "I/O Driver" programming package was added to the 1604A monitor system to handle transfer to and from the PDP-4. It is compatible with "Drivers" for other I/O units.

System Software

A summary of utility programs for the system is given in Table II. Brief descriptions of some of the more significant ones appear as follows.

TABLE II
Summary of System Software

Absolute Utility Routines

LIBCOR—Loads requested absolute programs from DECtape library.
DDTS—Dynamic, on-line debugging aid.
MICDAR—Allows user to dump onto and retrieve from DECtape.
MRLL4—Relocating linking loader; loads from paper tape and DECtape.
RLLMAP—Displays undefined external symbols and loading map (from MRLL4).
QBID—Allows 1604A job insertion from PDP-4/340.

Relocatable Library Routines

EXECUTIVE—Selectively relieves programmer of burden of handling some functions; a relocatable subroutine under his control.
MICROW—Relocatable subroutine to read and write DECtape.
Characters—Upper and lower case English letters and other alphabets and special characters.
Computational Subroutines
Display Generation and Control Subroutines

Off-Line Support

MIDAS—Conditional, macro, relocating assembly program. Runs on PDP-1 or 1604A, not PDP-4.

[6] Control Data Corp., *COOP Monitor Programmers' Guide*, 60050800, rev. A, April 1964.

[7] ——, *Reference Manual, Control Data 1604A Computer*, 245a, rev., May 1963.

DDTS—An On-Line Debugging Routine

DDTS is similar in many ways to FLIT, DOCTOR, and DDT.[8] However, the output medium for DDTS is the display, which permits dynamic qualities not found in the previous versions using typewriters.

As in DDT, the user can "open" any register by typing its address followed by a slash. The address of each open register appears at the left of the CRT and the *current* contents of that register appear beside it. As many as 20 locations can be displayed on the screen at one time. Newly opened locations are displayed at the bottom of the list, and when the screen becomes full the oldest ones simply roll off the top and disappear. The significant point here is that DDTS completely recomputes what to display each time the display is refreshed. Thus, if the contents of some displayed location change, that change will immediately become visible on the screen. (This is illustrated in Fig. 4. Location 7 is a 60 c/s clock.)

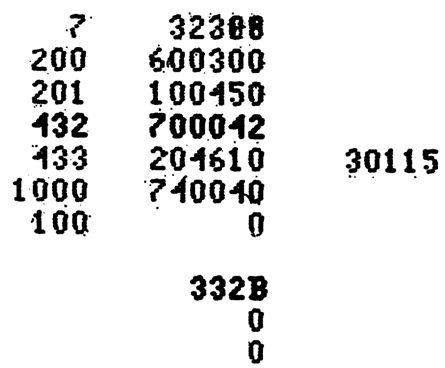


Fig. 4. DDT. Note that the contents of location 7 are changing.

The operator may modify core locations, perform word and effective address searches, set a breakpoint, execute a single instruction, define symbols, and so on, as in other versions of DDT. However, DDTS displays the latest contents of every address on its list each time a breakpoint is encountered. Thus, if DDTS is asked to proceed automatically a number of times (up to 2^{18}) through a breakpoint, the results of program calculations can be observed as they change. This is particularly interesting in observing the effects of manual intervention devices on the operation of programs. DDTS preserves and restores the state of the display so that a breakpoint can be used in a program which is also using the display. In this way, one can monitor both the changing contents of memory and its effect on a display.

The only serious disadvantage of this version of DDT is that space and speed limitations restrict it to displaying only in octal. On the other hand, it does accept symbolic information as typed input.

[8] A. Kotok, *DEC Debugging Tape*, Cambridge, Mass.: MIT, Memo MIT-1, rev., December 1961.

MIDAS—A Conditional Macro Assembly Program

All programming for the PDP-4/340 has been done in the MIDAS assembly language, which was originally written for the TX-0 at M.I.T.[9] Our version of MIDAS runs on a local PDP-1 computer, but assembles programs for the PDP-4/340 target machine as well as for the PDP-1. Another version of MIDAS has recently been written to run on the 1604A and to assemble for the PDP-4/340 target system. The power of MIDAS and the flexibility of its relocating linking loader have been an important contribution to the success and usefulness of the PDP-4/340.

MIDAS is a symbolic assembly program which permits flexible use of macros and conditional assembly. Using this capability, a series of macros has been defined which generate display instructions in the format required by the 340. One macro, for example, automatically converts Δx and Δy specifications for vectors from the one's complement representation used by the assembler into the sign and magnitude notation used by the 340 display. It packs the two components into the appropriate places in a PDP-4 word along with additional bits to indicate bright or dark and escape from vector mode or not. Another macro, dtext, generates a series of subroutine calls to display textual information. The user wishing to assemble a specific message writes, for example:

dtext this message will be displayed.

MIDAS generates relocatable binary code which is linked at load time by the MIDAS relocating linking loader. The relocation scheme permits references to external symbols plus or minus any absolute constant; no transfer vector is used. The MIDAS linking loader loads user-supplied routines and can also load library subroutines from paper tape or DECtape. The subroutines may be computational subroutines for the PDP-4 or display subroutines for the 340; for example, a multiply routine or members of a character set.

An Executive Routine

The hardware flexibility of the PDP-4/340 system places a considerable burden on the programmer. In order to ease this burden on those who wish to make use of parts of the equipment in standard ways, an EXECUTIVE program is provided in the program library. The EXECUTIVE handles PDP-4 interrupts, pushbuttons, lights, relays, knobs, and the display. For example, by merely setting an appropriate bit the user can cause EXECUTIVE to treat any pushbutton and light combination as a complementing flip-flop

[9] MIDAS was first written by R. Saunders. It was translated for the PDP-1 by D. Gross, A. Kotok, W. Mann, and R. Saunders. Unfortunately, no generally available documentation exists.

rather than a momentary pushbutton. Similarly, EXECUTIVE will convert the raw typewriter data or control button actions into character codes delivered to the user as each pushbutton is depressed, or released, or both. Specific interrupts are passed on to the user program, or cleared and ignored, as directed by the user. Knob and trackball counts are accumulated and can be restricted to any specified range.

It should be emphasized that the EXECUTIVE permits the user to handle anything he wishes in his own way; the EXECUTIVE handles everything else. Furthermore, the user can modify EXECUTIVE's behavior during execution. EXECUTIVE is not a monitor in the normal sense; rather, it is a relocatable subroutine which resides in memory only when loaded as part of the user's package of programs.

Operating Experience

Matters of Taste

People have used the console facilities in many different ways. Some people like the light pen; others prefer to use a trackball-controlled cursor. Some people assign a unique meaning to every button; others wish to use a smaller set of buttons and serial sequence coding. Those who are accustomed to using typewriter keyboards often ignore the other buttons. Other people use the typewriter keyboard for alphanumeric communications, the bank of 18 buttons for "control," and the bank of 36 buttons for binary data insertion. A few tolerant individuals seem to adapt to almost any control philosophy, as long as they get response from the system. Some prefer feedback to be visual, on the CRT or via lights. Others prefer audible responses.

The design philosophy has allowed users to follow their whims in the use of the console equipment.

Problems—Guidance for Future Work

Many of the programming difficulties encountered are related to the short (18-bit) word length of the PDP-4/340. In this "bit-poor" design, the 340 display uses modes in order to get the maximum data packed into each word. When the display is in vector mode, for instance, 16 of the bits in each word are interpreted as Δx and Δy information. When in increment mode, the same 16 bits specify plotting of four successive adjacent points. When in control mode, the same 16 bits have an entirely different meaning. One cannot tell by merely examining a single word what effect that word will have on the display. Programming must therefore be done in word pairs. Each word is a member of two overlapping pairs, because it will be interpreted correctly only if the previous word has set the correct mode. The present word must, in turn, set the correct mode for the next word.

Once debugged, the display programs operate very well with the mode switching design. The programmer must learn ways to keep his modes consistent, especially during dynamic modification of display files. Most mode errors show up very clearly in the resulting display. However, mode errors involving the deposit-save register instruction (DDS), are especially troublesome since it modifies the contents of memory. In particular, the point mode instruction which sets the y-coordinate is interpreted in control mode as a DDS instruction. It changes the contents of memory at a location whose address is equal to the intended y-coordinate value. If the intended y-coordinate is changed as part of a moving figure, different places in memory are changed. This, together with the secondary effects of executing changed instructions, makes such an error difficult to trace.

A better design, from the point of view of the programmer, would be "bit-rich" so that each instruction could be complete within itself. Each instruction in such a design would contain bits to specify whether it generates a vector, increments, points, or a control function. These bits would serve the function of an operation code in an ordinary computer. That is, instructions should have a fixed operational meaning independent of the route by which they are reached. With such fixed meanings, one could change individual display instructions without concern about the modes of adjacent instructions.

The long persistence phosphor used in the 340 reduces flicker problems, but it correspondingly limits dynamic changes in display. The wait required for newly changed digits to become legible is sometimes annoying. Repetition rates as low as 10 to 15 c/s have been used. Because this equipment has been solely for exploratory purposes for short durations (a few minutes to an hour) by any one user, those processes in which there is appreciable flicker have not caused any physiological problems. There is no hardware-imposed lower limit on the refresh rate of the display. As more and more demands are made on the complexity and volume of the display and on the processing in the control computer, the system gradually degrades. The practical limit is determined by the tolerance of the individual user.

There is yet no other *single* technique with the apparent flexibility of a light pen for communicating with a computer process. It can be used to define position in a plotting space, select items, insert parameters, and make process decisions. On the other hand, the hardware and software sophistication necessary to make the light pen operate to our satisfaction has not yet been achieved in this system. Users tend to use the other control transducers (buttons, etc.) because they are easier to program. We have found the RAND Tablet to be superior for entering positional information.

The rate of interaction which is practical to achieve with the host processing system is very low-grade time-stealing. Access for service is possible only between jobs. Because the job mix is heavily weighted below three minutes running time, processes requiring occasional bursts of interaction between the display and host systems are practical. Constant interaction is very inefficient for the host because it is waiting for the man most of the time, but it is impossible for the display system to obtain instant access in any other manner. The balance of power needed between a display control computer and the host system is heavily influenced by the frequency with which it is possible to interrupt the host machine, and the overhead processing costs involved.

Conclusions

An extremely flexible and general display system has been implemented; it is one which the user can mold to his purposes via programming.

The most powerful organizational feature of the equipment is the sharing of memory between a conventional small computer and a special picture computer for display generation. Each can be made to affect the operation of the other through interrupt and memory store operations. There is no need for a high-speed data channel between separate memories for picture storage and computation, since any part of the memory is immediately available to either control section. The presence of the small computer removes the load of display control and console monitoring from the host processing system, reserving its power for things the small computer cannot do.

The power of the system is enhanced by endowing the display with some of the characteristics of a computer. This makes it possible to reflect the structure of a picture in a subroutine structure. Correspondences between sections of display code and actual objects make it easy to consider each section of display code as an analog of the object drawn. The code is more than an enumeration of the lines of the picture. It is a prescription for assembling the picture from subparts, each of which is represented by repetitive display subroutine calls. A small change in one display subroutine may then show up in many places in the picture.

To achieve the flexibility of use appropriate to a research installation such as this one, certain design tradeoffs were made. Display generation speed was traded for stability and precision. Character generation speed was traded for character set generality. The apparent conveniences of wired-in functions were traded for programmability. We feel that our design choices have been justified by the wide variety of uses made of this display. In fact, the success in using this system has recently led to the construction of additional copies of it, using a PDP-7 in place of the PDP-4.

Authors' Note

The concept of display subroutining as used in this equipment was originally described by I. E. Sutherland and W. R. Sutherland in a M.I.T. Lincoln Laboratory (Lexington, Mass.) internal memorandum dated January 9, 1964.

There have been two recent publications concerning the design of general purpose display systems with essentially the same objectives as the SCHOOLER system described here.[10,11] The work reported by Ninke on GRAPHIC I employs similar hardware, but it uses a separate buffer memory to store the display programs and for communication with the host processing system. In use, much heavier emphasis has been placed on the light pen in the GRAPHIC I system. Rippy and Humphries described MAGIC, a system with very different organization based on serial logic and with less processing power local to the console system.

These three programmable systems represent a departure from the wired logic display console designs of three or four years ago.

[10] W. K. Ninke, "GRAPHIC 1—A remote graphical display console system," *AFIPS Conf. Proc.* (*FJCC*), vol. 27, pt. 1, Washington, D.C.: Spartan Books, 1965, pp. 839–846.

[11] D. E. Rippy and D. E. Humphries, "MAGIC—A machine for automated graphics interface to a computer," *AFIPS Conf. Proc.* (*FJCC*), vol. 27, pt. 1, Washington, D.C.: Spartan Books, 1965, pp. 819–830.