Reprinted from *AFIPS Conference Proceedings*, National Computer Conference, 1974, pp. 257-265. (AUGMENT,20184,).

Line processor—A device for amplification of display terminal capabilities for text manipulation

by DONALD I. ANDREWS

Stanford Research Institute
Menlo Park, California

INTRODUCTION

The Line Processor is a microcomputer-based device that was developed at Stanford Research Institute's Augmentation Research Center (ARC). It was designed to make it possible to use inexpensive alphanumeric video display terminals with ARC's sophisticated interactive information manipulation system, NLS.[1]

NLS is a software system that runs in a timesharing environment and is accessed via enhanced display terminals we call workstations, in many cases through a computer network.

The Line Processor concept involves placing a processor in the transmission line between the main computer and the display terminal. This enables us to expand the capabilities of the alphanumeric display terminal so that it will meet the requirements of an NLS workstation terminal. The contributions of the Line Processor concept for the most part address the needs of the software system NLS and are as follows:

The microcomputer implements a "virtual two dimensional alphanumeric terminal." Primarily, the virtual terminal allows device independent manipulation of displayed text. The microprocessor "maps" the virtual terminal commands from the main computer into actual commands for the particular display attached to it.

The "virtual terminal" was designed to mate with the interactive text manipulation techniques developed at ARC especially for NLS. Understanding these techniques is most useful in understanding the Line Processor operation.[2]

Attached directly to the Line Processor is a pointing device that allows the user to point to any character on the screen. This is achieved without any modifications to the standard alphanumeric video display terminal.

The Line Processor also implements a "scrolling window" feature. This makes the terminal usable as both a two-dimensional applications display terminal and a teletypewriter terminal simultaneously, without scrambling of text. The "scrolling window" concept is extremely important in an application such as ours where the display terminal is used for both purposes simultaneously.

Communication to and from the main computer consists of 7-bit ASCII characters over bit-serial transmission lines or computer networks.

In developing the Line Processor, we have established what we feel is the minimum set of display terminal capabilities that enable two-dimensional interactive text manipulation. These requirements are discussed in detail.

Before explaining the Line Processor characteristics and operation, we will briefly describe the NLS software system and the NLS workstation environment.

NLS AND THE WORKSTATION ENVIRONMENT

NLS is a highly interactive system that joins many capabilities. It supports structured text files, very flexible editing, techniques for viewing and studying, arbitrary word processing and document production, and aids to applications and system programming. (For a general description of ARC's goals and efforts see Reference 1 which sites related papers.)

The system can be operated from a range of terminal types: from typewriter terminals to high speed graphics displays. Our main effort has been to develop a carefully human engineered display oriented system. The display version of the system introduces the user to the world of two-dimensional computer interaction, which is much more natural than the one-dimensional mode forced upon typewriter device users.[2]

The NLS program runs as a subsystem of a TENEX timesharing system on ARC's Digital Equipment Corporation PDP-10. The PDP-10 is connected to the ARPA network and many of our users gain access via that network.[3,4]

NLS users employ two special input devices continuously: a mouse and a five-finger keyset. We use the term "workstation" to mean a display with standard keyboard and these devices, arranged on a special table in a convenient manner for effective working. (See Figure 1.)

The two unusual workstation input devices, the mouse and keyset, are commercially available separately, but are offered as standard options on only a few products. (Mice and keysets can be purchased from Imlac, Cybernex and Computer Displays.)

The five-finger keyset has five long keys as shown in Figure 2. The user rests his fingers lightly on the keys and strikes chords to input characters. The typical user learns enough binary codes in a couple of hours to do useful work.

The user can point to characters on the screen by rolling

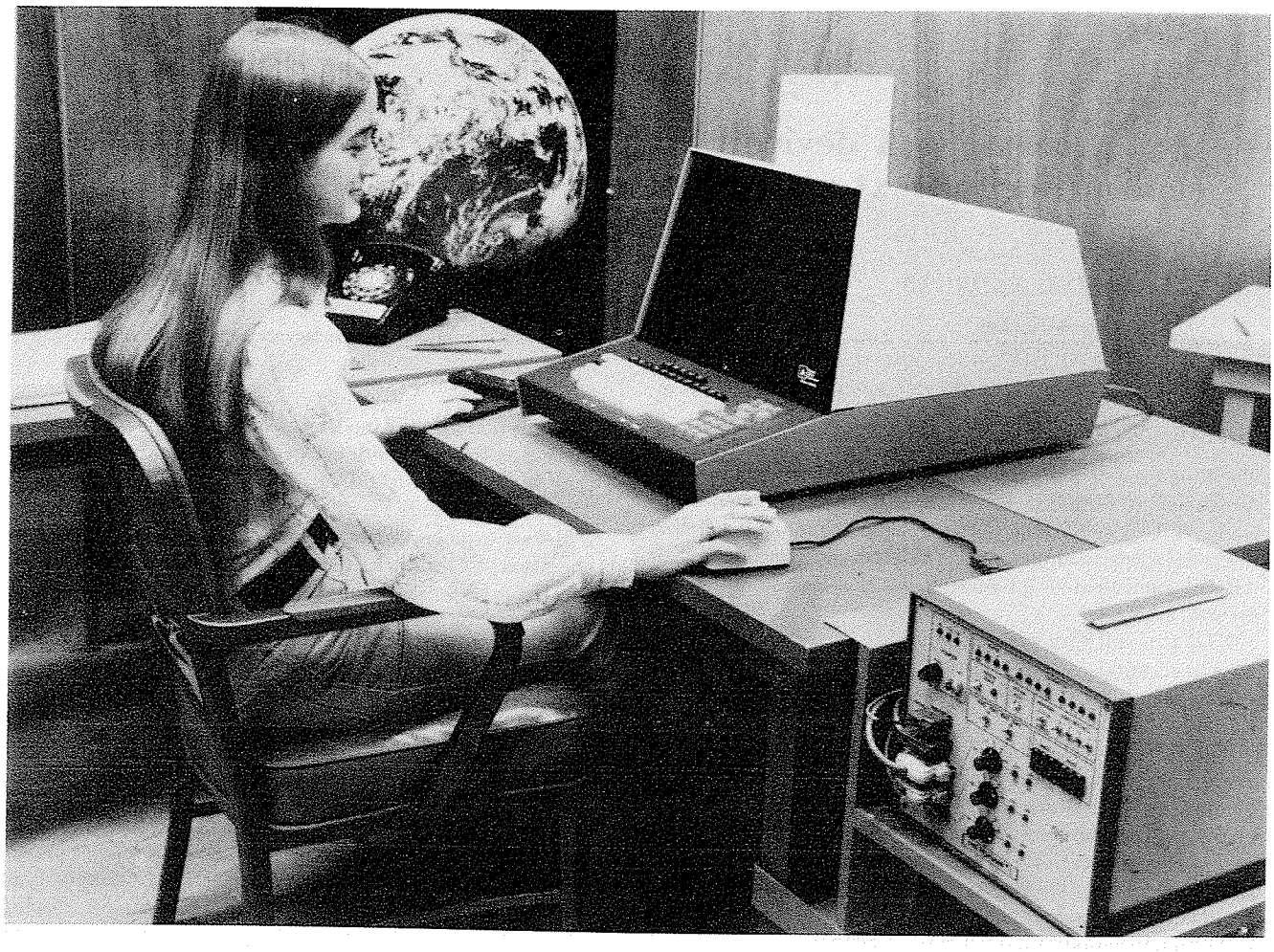

Figure 1—NLS workstation with line processor

the mouse on any flat surface. Potentiometers are connected to the orthogonal wheels on the mouse, and they provide analogue signals that are translated into digital X and Y position coordinates by a two channel A to D convertor. (See Figure 3.)

The user can operate three buttons on the mouse to confirm commands, abort commands, provide a case shift for the keyset and perform other programmable functions. The workstation has no other special function keys or buttons.

With the right hand on the mouse, and the left hand on the keyset, the user can edit, change his view and perform other control operations rapidly and easily. He does not remove his hands from the mouse and keyset, except to type large amounts of text. The mouse and keyset are further described in References 5 and 6.

The final element of the workstation environment is the user. It is intended that the user does the majority of his day-to-day work via NLS and the workstation. The user becomes increasingly proficient at using the system and the human engineering aspects of the workstation become very important.

GOALS AND PROBLEMS

For several years, we have been wanting to make NLS available to remote users on inexpensive display terminals. The primary problem has been lack of commercial availability of adequate displays. At ARC, we have a custom-built display system that meets our needs, but is not available to others.[7]

To be an adequate NLS workstation, a display terminal must meet several general requirements such as screen size and character set. These will be discussed later.

In addition, we require that the terminals have a mouse or comparable pointing device.

Also we prefer to use terminals that are commercially available and maintained nationwide.

Remote operation over high speed phone lines and computer networks such as the ARPA Network is absolutely necessary.

Further, they should be available in single quantities, and operate as a time-sharing system typewriter terminal as well as an NLS workstation.

Very few displays on the market meet our requirements. At this time, the only commercially available product with mouse and keyset that meets our requirements is a minicomputer-based display terminal, the IMLAC PDS-1. It makes a satisfactory NLS workstation, but appropriately configured it costs about $18,000. We would like to see workstations available for about $5,000.

Many low-cost, alphanumeric video display terminals on the market today are attractive and almost meet our needs. Unfortunately, many are designed as replacements for typewriter terminals, IBM 2265's, or other existing terminals. From our point of view, the designers of these terminals took an unnecessarily limited view of their possible applications.

Even those alphanumeric video terminals that meet our general requirements are not suitable for NLS use since they lack provisions for adequate graphical pointing devices—and it is generally very expensive or impossible to interface devices such as a mouse to existing products.

In addition, we would like our users to have a choice of terminals to use as workstations. But there is no industry standard for the terminal function codes, such as delete line. What is worse, there is no general standard as to how some of the functions are carried out on the screen. Hence we have to write and maintain a software driver for each different type of terminal we support. We already support a number of different hardcopy terminals, and we would prefer not to aggravate the problem if possible. In our development, we rewrite, modify and extend large parts of our software system continually and increasing the volume and complexity of the software naturally increases our development and maintenance expense.

SOLUTION

To solve the problems mentioned above, we built a special purpose I/O device based on a four-bit microcomputer, which we call a Line Processor.

The combination of a Line Processor and an adequate video display terminal results in an effective NLS workstation.

The Line Processor connects to the display terminal's RS232 bit-serial interface. No modifications are required of the display terminal. (See Figure 4.)

The mouse and keyset connect directly to the Line Processor. All special purpose hardware for these connections is neatly localized within the Line Processor.

Another standard RS232 full-duplex interface connects the Line Processor and the main computer. Communications over these lines consists of 7-bit ASCII characters. ASCII control characters are avoided to the extent that transmission over the ARPA Network involves a simple TTY-type connection.

The microcomputer implements the "virtual NLS terminal." It maps the terminal-independent Display Manipulation Protocol into the specific terminal functions for the particular terminal being used. All composite terminals (display terminal plus Line Processor) are logically the same at the point of main computer connection, and as a result, only one software driver is required.

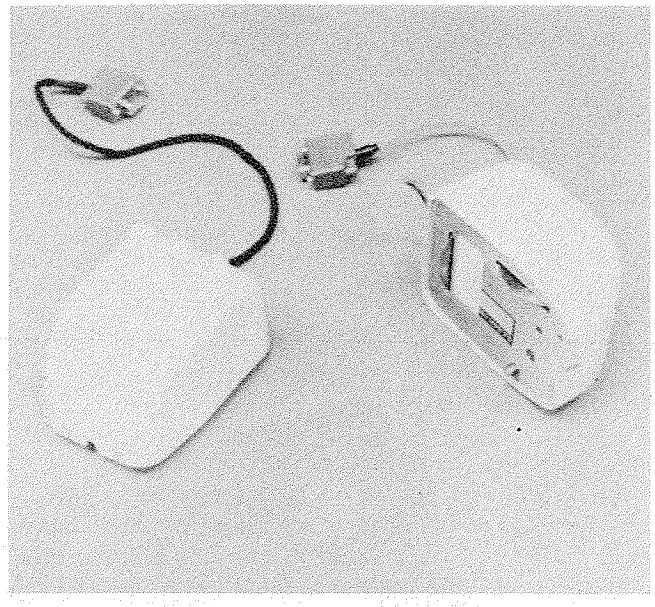

Figure 2—Mouse

The microcomputer can be programmed to work with any alphanumeric display that meets certain requirements described below.

The microcomputer program does not require any software maintenance expense, since it is treated as a hardware device when the development is completed. On the other hand, the microcomputer program could be changed without altering the Line Processor hardware, if that proved to be useful.

REJECTED SOLUTIONS

We considered several alternative solutions to the problems mentioned above. Here are the other alternatives that we investigated:

1. Have a special workstation terminal built by a reputable manufacturer. In 1972, we sent out a request to several manufacturers for a proposal to modify their standard terminal to meet our requirements. There was virtually no interest in serving our needs, and all indications were that special workstation terminals would be very expensive. The manufacturers either were not interested in modifying their standard products or would not address themselves to what they viewed as a relatively small market.

   We expect that in the future video terminals will have the necessary I/O capabilities and be microcomputer controlled so that adaptation to applications such as ours would be relatively easy.
2. Modify an existing alphanumeric terminal ourselves. There are several disadvantages to this approach. Having the modifications made on a production basis

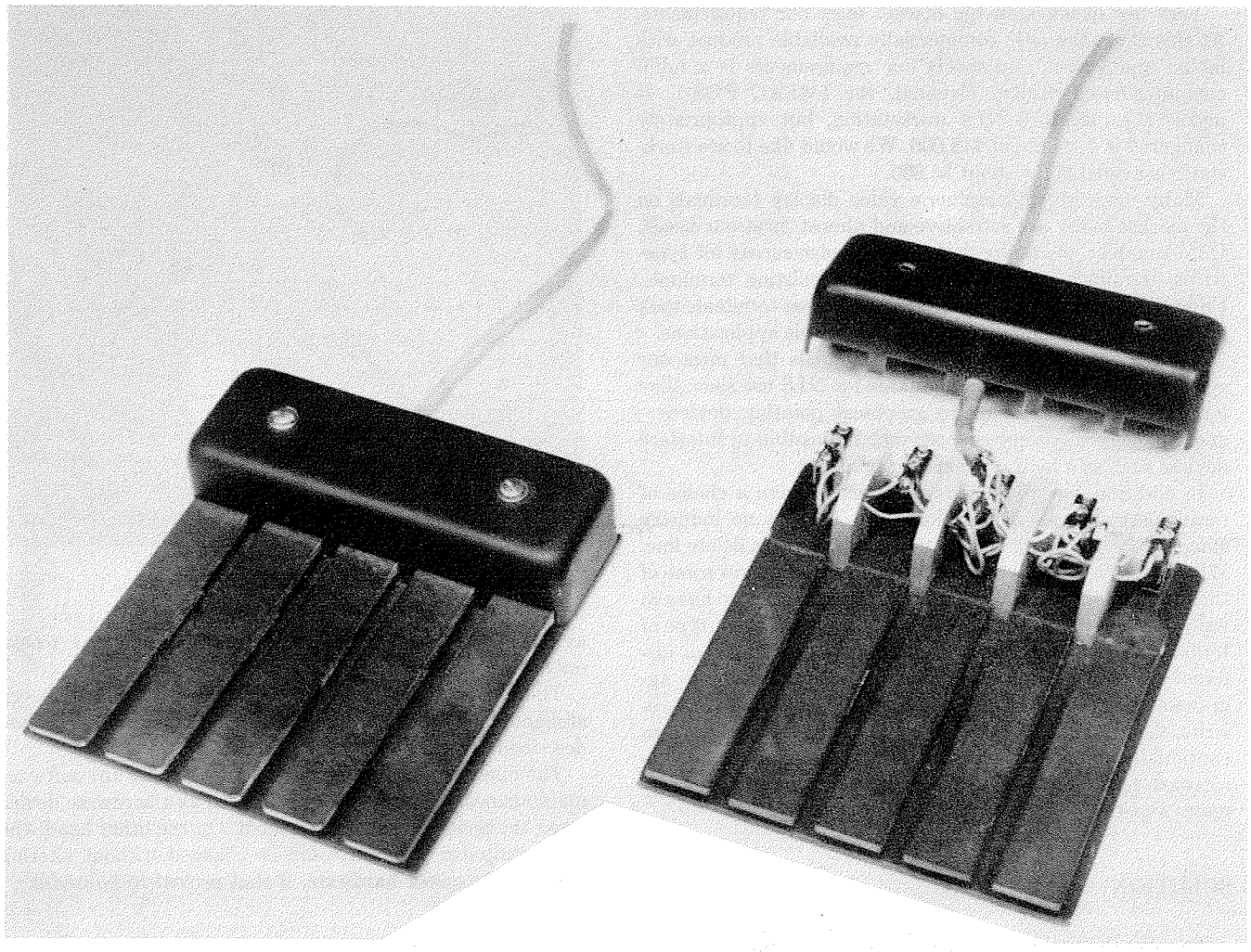

Figure 3—Keyset

and convincing the manufacturer to maintain modified terminals are big obstacles. Also, once the terminal is obsolete or no longer manufactured for any reason, we are faced with our original problems again.

3. Use of a programmable terminal that offers the necessary I/O capabilities. This is exactly what we did by programming the IMLAC PDS-1, but we feel that it results in a workstation that is too expensive. Other programmable terminals that met our general requirements exist in the same general price range as the PDS-1; one of our primary objectives was to reduce the price per terminal. But the price of suitable programmable terminals may drop enough in the next few years that this will be an attractive solution.
4. Build a very simple hardware device to interface the mouse and keyset to a communications line. The Line Processor performs several functions that need not be performed at the terminal site. For example, mouse tracking, TTY-simulation and display manipulation protocol implementation could be performed in the main computer. The device proposed here would simply transmit keyset chords and mouse position changes to the main computer.

There are two serious problems with this solution. Tracking the mouse by the main computer, when connected by a computer network, would be unfeasible because of network delay times. Also, the terminal would not have the "scrolling window" feature.

This is a reasonable idea but, compared to the Line Processor, the device would increase significantly the compute load on the main computer, and it would increase the amount of transmission over the terminal-computer connection. Furthermore, it would require more software in the main computer. In a large and fixed system where terminals were not connected by networks, this may be an advisable approach, but we felt that it was unsuitable for our configuration. Our software system is continually evolving, our main

computer is overloaded and we have many of the terminals connected via computer networks.

In considering each of these alternatives, we were concerned with the cost of the development, but primarily we were concerned with the quality of the resulting NLS workstation and the end cost to the consumer. Providing a means of access to NLS for remote users is one of many R&D goals at ARC. Hence the cost of development of the Line Processor was borne by ARC and its sponsors, and did not have to be passed on to the consumer.

LINE PROCESSOR FUNCTIONS

There are several important functions that the microcomputer performs to make the combination of display and Line Processor an effective workstation.

1. Protocol Implementation—There are several protocols involved. They are identified in Figure 4.

   The Display Manipulation protocol is exactly the same for every Line Processor workstation and is sent by the applications program to the Line Processor to change the display image. It does not affect the display terminal directly, but is translated by the microcomputer into the Terminal Function protocol.

   The Display Manipulation protocol is designed to work with any alphanumeric terminal with cursor control and line editing functions such as delete line and insert line.

   The Line Processor talks to the display terminal in the Terminal Function protocol. This is defined by the terminal manufacturer and usually consists of ASCII control codes, or sequences of control codes, interspersed with ASCII text to be written on the display screen.

   The Line Processor workstation serves as both a timesharing system typewriter terminal and a two-dimensional applications system display output terminal. Hence, there are potentially two streams of output going from the main computer to the Line Processor on the same communication line: the Display Manipulation protocol, and any teletypewriter terminal output that the timesharing system or applications programs send. The teletypewriter output would be generated if the user were using the terminal as a typewriter terminal, or if the user received an error message or some type of system-wide message. These two streams of output are separated by the Line processor, and TTY-type output is displayed in a TTY-simulation area. This means that the teletypewriter output is not scrambled in with the display output, but it is scrolled—teletypewriter fashion—in a small portion of the screen. The applications program has control over the size and location of the TTY-simulation area. (The TTY-simulation and window concepts are described in reference 2.)

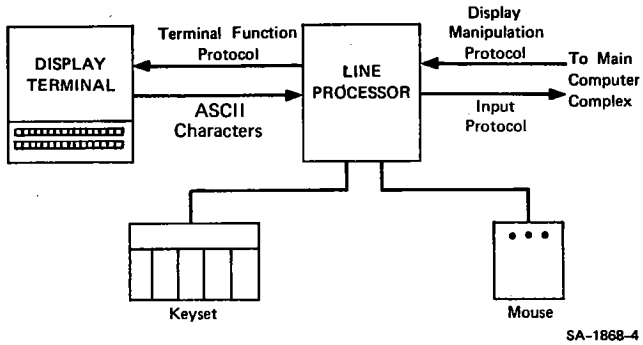

Figure 4—Display terminal, line processor and main computer connections

2. Keyset and Mouse Button Transmission (Input Protocol)

   Information from the input devices (keyboard, mouse and keyset) are incorporated into an Input protocol by the Line Processor and sent to the main computer. The protocol is quite simple. Control characters and mouse button changes are sent as short strings of characters that include the mouse tracking spot location (line and character position). Characters from the display's keyboard are sent in unescorted fashion, that is, each single ASCII character is sent.

   Input from the display terminal's keyboard simply comes into the Line Processor from the full-duplex display connection. This consists of ASCII characters that the user has typed on the keyboard. Local editing from the keyboard and batch transmission features within the terminal, if present, are not used for NLS applications.

   There are three buttons on the mouse. A change of button positions sends in one character to the main computer, which indicates the new three-button status. The applications program knows the state of the mouse buttons and can interpret any subsequent input accordingly. This in effect allows eight different interpretations of keyset and keyboard input.[2]

   The five-finger keyset input is interpreted as "chords". The normal condition is all-keys-up, and the end of a stroke is indicated by all-keys-up. Any keys that were depressed during the stroke are "OR'ed" together to comprise the chord, which eliminates the problem of striking keys in unison. Thus, a keyset input stroke is a non-zero, five-bit integer. It is converted to an input character in the Line Processor, and sent on to the main computer in unescorted fashion.

   The five-finger keyset and mouse buttons are gated directly into the microcomputer. These signals are not "clean" and may be "bouncing" for a few milliseconds after a button is pushed. The switches are sampled at a rate at which the bouncing can be detected, and the final reading is not taken until the switches are stable.

3. Mouse Tracking

The Line Processor reads the mouse position from the A to D convertor and "tracks" its movements on the screen with the standard display cursor implemented within the video terminal. This is done by periodically reading the mouse position and, if it has changed, sending a position-cursor command (a part of the Terminal Function protocol) to the display.

The tracking allows the user to point to any character on the screen at any time, yet the main computer is only informed of the mouse position when the user strikes an appropriate key or button.

The mouse moves in a smooth continuous manner, but the tracking spot (cursor) moves from one character position to the next in discrete jumps. The tracking will appear jumpy rather than smooth if the baud rate between the Line Processor and the display is not high enough. Exactly what the baud rate has to be depends on how many characters must be sent for the position cursor function. We have found that 2400 baud is a minimum baud rate with the standard four-character sequence for cursor positioning found on most alphanumeric terminals.

4. Display Control

The Line Processor maintains control of the display because it is the only device sending characters to the display terminal. The applications program in the main computer manipulates the display only by way of the Display Manipulation protocol.

The mouse tracking and the TTY-simulation feature mean that the microcomputer must have a fair amount of software logic to keep track of the display terminal's cursor position and know where to move it when necessary.

TERMINAL REQUIREMENTS

A display terminal must meet several requirements to form an effective workstation with a Line Processor. These requirements are due to either system or user considerations:

1. System Considerations

It must be possible to perform several kinds of display functions by way of the terminal's RS232 connection. That is, the Terminal Function protocol must be adequate and operable under computer control. We will discuss the actual functions that are required.

One of the key functions is positioning the cursor. For the mouse tracking to be satisfactory, it must be a very quick process (within one millisecond), and the cursor must not be displayed in any extraneous positions on the screen while being moved to the destination. The cursor must be addressed by the character and line number of the screen position.

The appearance of the cursor must be suitable for tracking. A blinking underline cursor is not very satisfactory since it is not always visible. Some displays implement the cursor by reversing the video in the entire dot matrix of the character in question. The resulting reverse video rectangle that moves around the screen is satisfactory for tracking.

It is necessary to have a high speed connection between the Line Processor and the display, as mentioned before. We feel that 2400 baud is satisfactory; we prefer 9600 baud because it makes a more effective workstation.

The bulk of the display manipulations are done via delete line and insert line functions. These manipulations usually involve positioning the cursor and then issuing the command.

Nearly all terminals have a clear screen function, and we require it.

We expect carriage return and line feed to work on the display just as they work on a standard teletypewriter device. That is, carriage return moves the cursor to the left margin and line feed moves it to the next line without changing the character position on the line. Although a single "next line" code is often useful, we do not feel it is reasonable for a manufacturer to omit either of these two fundamental functions.

When a user selects a character on the screen by pointing to it with the mouse tracking spot and pushing a button, the applications program usually responds by "marking" that character for confirmation. The "marking" feature is very important to the interaction process and it has given us a bit of trouble in transforming an alphanumeric display into an effective workstation. We will describe the problem in some detail.

The "marking" is done by altering the appearance of the character without obliterating it. If the user selected the wrong character he will abort (or "back out of") that selection and select another. The NLS program responds by removing the first mark and putting up another. Hence, it must be possible to "mark" and "unmark" characters on the screen without altering the text in any other way and without rewriting the character. On displays where single characters can be made to blink, reverse in video, or change in some other way, marking will be implemented using that feature.

But very few displays have the capability of "marking" a character without using up a character position on each side of the character, which is unsatisfactory for our purposes. We have had success by showing the "mark" by "flashing" the cursor at the marked character position at a rate of about three times a second; the cursor is returned to the tracking spot between flashes. This is satisfactory if the display-Line Processor connection is of high enough baud

rate (4800 or 9600 baud) and the cursor positioning within the display is fast and does not result in extraneous flashes on the screen.

Clearly, the former type of "marking" is preferred because the latter scheme results in a flashing tracking spot and becomes confusing when there are several "marks" on the screen. Probably the most desirable type of "marking" feature would be to make the character blink at about 3 cps.

Occasionally, it is desirable to write a string on the display with a special appearance, to get the user's attention. We call this "standout mode." Most displays have some kind of standout feature such as blinking, underline, high intensity or reverse video—we require some kind of standout mode. This differs from marking: in marking an existing character on the screen is altered without rewriting, and in standout mode new text is written on the screen.

Of course, all these functions take some finite time for the display to carry out. We would like to see all functions performed as fast as possible. In particular, we expect that all functions except delete line and clear screen will be performed in one millisecond or less. We expect delete line and clear screen to be done in about 5 to 7 milliseconds. We set an upper bound of 120 milliseconds for any function execution time.

2. User Considerations

We consider 24 lines by 64 characters a minimum adequate screen size; however, 27 lines by 80 characters is much more useful. The most desirable would be a full text page of 66 lines by 80 characters. No matter what the screen size, we expect the terminal to have enough memory capacity to nearly fill the screen with text.

The terminal must display the full ASCII character set, including upper and lower case letters.

The keyboard should be a standard typewriter style keyboard with the full ASCII character set, including control characters. Some form of key rollover feature and a comfortable feel are very important.

The display output should be readable, easy to look at and flicker free. Our applications are geared toward comfortable day-long use in an office. The terminal should be absolutely quiet and a pleasant thing to work with in all respects.

HARDWARE DESCRIPTION

The organization of the Line Processor is outlined in Figure 5. The Line Processor is discussed from a hardware standpoint in Reference 8.

The heart of the device is an Intel 4004 CPU, which is a four-bit parallel microcomputer in a single MOS integrated circuit chip. The program resides in up to six Intel 1702A programmable ROM (PROM) chips. Each PROM contains 256 8-bit bytes. These PROM chips are mounted in sockets and can be removed, erased and rewritten in a few minutes.

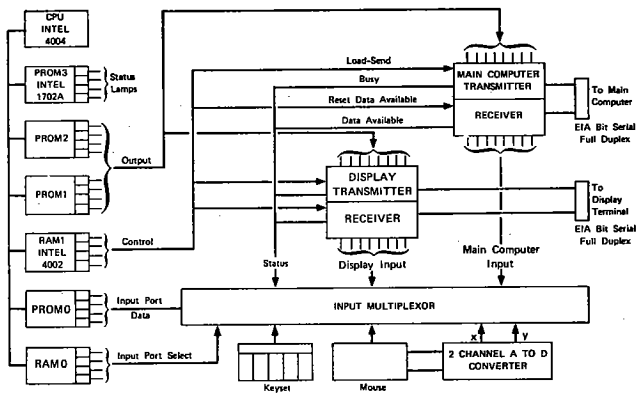

Figure 5—Microcomputer and I/O configuration

The read/write memory consists of 320 four-bit bytes packaged in four Intel 4002 RAM chips. These Intel chips are part of Intel's MCS-4 Micro Computer Set, and are easily connected. Both the RAM and PROM chips have I/O ports that are addressed like memory, but they are accessed with different CPU instructions.

The input devices are multiplexed into one input port. A four-bit address on another port selects the particular four bits of input to be read. The output devices (two serial transmitters) are loaded via two ports and controlled via another port.

The general configuration is such that Intel 4001 ROM chips can be substituted for the PROM chips, if the production quantity warrants the initial cost of cutting the ROM masks.

The total cost of all parts is about $1200 per unit in single quantities. We intend to find a firm that will manufacture Line Processors and provide nation-wide maintenance. A firm is currently making them for us on a limited basis for $1800 each.

CONCLUSIONS

The Line Processor approach can effectively upgrade a peripheral device and amplify its capabilities with no actual modifications to the device itself. At the same time, it can optimize the interface between the device and the main computer complex, with respect to both hardware and software.

In our application of NLS workstation development, the concept has been beneficial. The alphanumeric terminal needs no modification, and the main computer does not need to know the true nature of the workstation. In other applications, new peripherals could conceivably be interfaced to existing software without modifications to either.

A Line Processor coupled with a satisfactory alphanumeric video display terminal results in an NLS workstation that is as effective as a high speed general CRT workstation, except for one respect: the inability to arbitrarily move text on the screen without rewriting it. But, intelligent display manipulation algorithms reduce this problem to the point that users hardly realize that it exists.[2]

Other applications for the Line Processor concept are appearing. A microcomputer-based device is being developed at the University of California, Santa Barbara, to make a PLATO IV terminal appear to be a general purpose ASCII terminal for ARPA Network Use. At ARC, we are considering using a simplified Line Processor to make a line printer with a difficult interface appear to be a serial ASCII device, to avoid purchasing an expensive controller.

A prototype Line Processor coupled to a Delta Data 5200 display terminal has been in operation at ARC and over the ARPA Network since September 1973. The microprogram is 1024 8-bit bytes long and operates with the main computer connection set from 300 to 2400 baud. More recent versions operate at 4800 baud with either a Delta 5200 terminal or a Hazeltine H2000 terminal.

We are currently expanding the microprogram to provide more services to the user. For example, we are providing a hard copy output connection to the Line Processor to allow the user to obtain a printout at the same time he uses the workstation for unrelated matters.

We expect the microcomputer equipment to become cheaper and faster. These developments will allow the Line Processor and similar devices to have more capabilities. It appears that such devices, if properly designed, could tend to reduce software problems and expenses.

A growing trend, brought about by computer networks and interconnections of various kinds, is to divide workloads and define appropriate interfaces. With this method, the computing takes place over several processors, hopefully each best suited to its workload. We have accomplished that with the Line Processor. We are in the infancy of a "distributed computing" era in which microcomputer devices such as the Line Processor will clearly have a growing role.

ACKNOWLEDGMENTS

The development of the Line Processor has been a team effort by several members of ARC. The hardware design and prototype construction were done by Martin Hardy and Rodney Bondurant. Charles Irby and Kenneth Victor programmed NLS to use Line Processor workstations and modified our TENEX for the Input protocol. The design of the display manipulation protocol and the micro-programming were done by the author. Dr. Richard Watson oversaw the effort and was a primary source of useful criticisms during the writing of this paper. We all owe many thanks to Delta Data Systems for extended use of a Delta 5200 video display demonstrator during development of the Line Processor.

The work reported here is currently supported primarily by the Advanced Research Projects Agency of the Department of Defense, and also by the Rome Air Development Center of the Air Force and by the Office of Naval Research.

REFERENCES

1. Engelbart, D. C., R. W. Watson and J. C. Norton, "The Augmented Knowledge Workshop," *AFIPS Proceedings*, National Computer Conference, June 1973, (SRI Catalog Item 14724).
2. Irby, C. H., "Display Techniques for Interactive Text Manipulation," *Proceedings of the National Computer Conference*, May 1974, (SRI-ARC Catalog Item 20183).
3. Roberts, L. G. and B. D. Wessler, *The ARPA Network*, Advanced Research Projects Agency, Information Processing Techniques, Washington D. C., May 1971, (SRI-ARC Catalog Item 7750).
4. Bobrow, D. G., et al., "TENEX, A Paged Time Sharing System for the PDP-10," presented at *ACM Symposium on Operating Systems Principles*, 18-20 October, 1971, Bolt Beranek and Newman Inc., 15 August 1971, (SRI-ARC Catalog Item 7736).
5. English, W. K., D. C. Engelbart and M. A. Berman, "Display-Selection Techniques for Text Manipulation," *IEEE Transactions on Human Factors in Electronics*, Vol. HFE-8, Number 1, pp. 5-15 March 1967, (SRI-ARC Catalog Item 9694).
6. Engelbart, D. C., "Design Considerations for Knowledge Workshop Terminals," *AFIPS Proceedings*, National Computer Conference, June 1973, (SRI-ARC Catalog Item 14851).
7. Engelbart, D. C. and W. K. English, "A Research Center for Augmenting Human Intellect," *AFIPS Conference Proceedings*, Vol. 33, 1968, (SRI-ARC Catalog Item 3954)
8. Hardy, M. E., "Micro Processor Technology in the Design of Terminal Systems," *Proceedings of the IEEE COMPCON*, 1974, (SRI-ARC Catalog Item 20185).

APPENDIX: LINE PROCESSOR PROTOCOL AND OPERATION

The Display Manipulation protocol calls for the transmission of unescorted characters and short command strings. Command strings begin with an escape character: we use 34 octal. Characters within the command strings are seven bit ASCII printable characters. Sending printable characters, where possible, rather than control characters makes debugging and troubleshooting much less painful.

The Line Processor operates in one of two modes. One mode simulates a teletypewriter. The other is the normal display mode that allows NLS display manipulation.

The mode is specified by mode set commands from the main computer. The Line Processor responds to the "enter display mode" command (also called the interrogate command) by sending a string in protocol format that informs the main computer of the display screen size, length of time it takes to delete a line and the baud rate; the last two parameters are important for timing considerations.

To make the displays function correctly, it is necessary to send the proper number of "padding" or null characters while the display performs involved functions, like delete line and clear screen.

The microcomputer is programmed to send the padding characters, but it has limited buffer space and must receive pads from the main computer as well. It would be fine to have the main computer refrain from sending anything in most applications, but our users will frequently be connected by way of a computer network. In such an environment, the only way to ensure the proper timing is to send the appropriate number of padding characters.

The number of padding characters that need to be sent depends on the length of time it takes the display to perform the function and the baud rate going into the Line Processor. So, we have installed a baud rate switch on the Line Processor that can be read by the microcomputer.

Hence, when responding to the "enter display mode" command, the microcomputer includes the baud rate setting along with the other terminal parameters. From the baud rate and the delete time parameter, the applications program can compute the number of pads to be sent.

Display Manipulation Protocol Primitives

POSITION CURSOR (X, Y)
This command positions the cursor to the designated spot and stops the mouse tracking process. Any subsequent unescorted characters are written on the display starting at the cursor position.

RESUME TRACKING
This command is used after positioning the cursor and writing a string, to start the mouse tracking process again. Note: A string is written by sending: POSITION CURSOR; (the string); RESUME TRACKING.

DELETE LINE
The line at which the cursor has been positioned, is deleted.

INSERT LINE
A new line is inserted after the line on which the cursor was positioned.

CLEAR SCREEN
The entire screen is cleared.

BEGIN STANDOUT MODE
All text written on the screen subsequent to this command will be in "standout mode."

END STANDOUT MODE
This returns the text writing mode to normal.

RESET
This command resets the line processor to normal mode and clears the screen.

WRITE A STRING OF BLANKS (N = number of blanks)
It is useful to be able to clear a short area of the screen with this command.

PUSH BUG SELECTION (X, Y)
The coordinates for the bug selection mark are pushed on a stack, and the indicated character is marked.

POP BUG SELECTION
The top entry on the bug selection stack is removed and the mark at the corresponding screen position is removed.

SPECIFY TTY-SIMULATION WINDOW (Y1, Y2)
Y1 and Y2 specify the top and bottom line for the scrolling window. Any subsequent TTY-type output will be scrolled in this window.

ENTER DISPLAY MODE (INTERROGATE)

ENTER NORMAL MODE