1978 (?)

A Microprocessor Based CAI System with Graphic Capabilities*

by

Frank J. Mabry, Andrew W. Appel and Allan H. Levy, M.D.

I. Introduction

With the invention of the microprocessor chip, the economic restraints to personal computing have been substantially removed. Our laboratory has been concerned with developing microprocessor-based systems that are useful in education. We recognize that not all educational computing can be done well on a stand-alone basis; indeed, a user community actively participating in the interchange of information is an important component of the educational process, involving both feedback for the creation and evaluation of educational material. However, there are many aspects of the operation of a Computer Assisted Instruction (CAI) system in which such a system should function efficiently and adequately on a stand-alone basis, free of the costs of continuous communications.

With this in mind, we have developed a system which operates in a dual mode, on an independent basis, as well as connected to a communications network. We have constructed a hardware-software system which allows a standard PLATO IV (1) terminal to act as an independent computing system with the aid of a microprocessor equipped with digital cassette tape drives, and also to access remote systems, including PLATO, as well as standard teletype-like systems.

The PLATO system (2), developed by Bitzer and associates, has been a model for user flexibility and for the effective use of graphical techniques. Our system, in stand-alone mode, attempts to capture these attributes as far as possible.

II. Background

The Medical Computing Laboratory (MCL) of the College of Medicine at the University of Illinois has been involved with service delivery of computer based instruction to units within the College of Medicine, as well as outside the University via terminals connected to the PLATO system (3). Simultaneously, MCL research activity has been directed towards investigation of alternative forms of CAI delivery.

Significant progress has been made by Chen (4) in the development of a "local node" of computation with the potential for support of a 30-50 terminal CAI system, roughly equivalent in function to the larger PLATO system. Noting the ongoing development by Chen, the problems surrounding use of the existing medically oriented CAI systems, including single terminal systems, took our interest. We have attempted

*University of Illinois College of Medicine, School of Basic Medical Sciences at Urbana-Champaign

to develop a system with the capability of accessing teletype-like (ASCII*) communication systems and the PLATO system. The new system also was to be capable of stand-alone support of CAI activity and, in particular, should support the CAI language PILOT (Programmed Inquiry, Learning or Teaching) as specified by Starkweather (5).

The hardware configuration is detailed in Appendix A and shown in Figure 1 and includes the following:

- IMSAI microprocessor (based upon an INTEL 8080 microprocessor chip)
- 16K (K=1024) bytes of storage
- ASCII bi-directional communication ports (two)
- 1 millisecond interval timer
- Vectored interrupt driver
- Dual digital cassette tape drives (6)
- PLATO IV terminal and interface.

The software configuration used provides software to access CAI systems using teletype-like communications (e.g., Coursewriter on IBM systems or MUMPS on DEC systems). The software system also provides a BASIC** interpreter (referred to as NSBASIC [Not-So-BASIC]) which has extensions for allowing local programs to utilize the native capabilities of the PLATO IV terminal in a stand-alone mode. The capability remains for the PLATO IV terminal to access the CERL (Computer Based Education Research Laboratory) PLATO system which can be accomplished by a single switch setting on the system.

The general memory configuration and requirements of the system are given in Figure 2. A brief syntactic description of NSBASIC is given in Appendix B.

III. NSBASIC [Not-So-BASIC]

Beginning with an interpreter which provided the rudiments of the BASIC language, we have made extensions to support access to the native features of the PLATO IV terminal (particularly graphics), to allow "judging" of interactive responses, to provide program and data management primitives, and to facilitate communication under local program control with remote terminal systems.

The graphic extensions that have been added provide for line drawing (DRAW), absolute positioning (AT) and setting display mode (MODE). The DRAW statement allows the programmer to indicate the end points of one or more line segments. The coordinates of these end points

---

*ASCII - American Standard Code for Information Interchange.

**The BASIC Interpreter was written by Dr. Paul Tucker at CERL. It has been modified and extended in the course of the work described here.

may be any NSBASIC numeric expression. This allows the use of constants, expressions and/or variable reference. The AT statement both influences the beginning point of the next DRAW statement and where text is output in subsequent PRINT statements. The MODE command is used to indicate how output is to affect the present display. In mode "write" the dots of the display (512 x 512) are illuminated. In mode "erase" dots are turned off where lines or characters are displayed. Mode "rewrite" indicates that the 8 x 16 dot area associated with character output is to be erased before a new character is displayed.

Obviously CAI is more than graphics. Moreover, flexible student response judging is an important factor in providing effective student-computer interaction. In order to support sufficient capacity for interactive response processing new character string manipulation statements and functions were added to NSBASIC. The capabilities designed into these statements were influenced in roughly equal amounts by the capabilities found in PILOT and TUTOR*. As an example, the NSBASIC "MATCH" command allows a programmer to specify a list of words of varying length to be checked for in a string variable. The command indicates which word of the list (if any) was found and where it was found in the string. This facility allows a lesson author to specify lists of words that should or should not be found in a student's response and easily specify checking for their existence.

Many of the language primitives which were developed with response-judging in mind, have been useful in the development of both the PILOT translator and the graphic editor. For example, commands, such as MATCH, MOVE and FIND operating on strings can be used both to judge an interactive user response or to determine what source statement should be generated based on the character string value of a program statement which can be read as data.

Most users experienced in the PLATO system have come to expect "single-key-response," particularly when moving through multiple choice sequences. The facility for the programmer to direct massive display changes based upon a single keypress is widely accepted as a significant accomplishment of the system. Such sequences are successfully used at many points in student interaction to enhance the feeling of existence in a "friendly environment." NSBASIC provides several facilities to allow the programmer to support such user options. The "KEY" statement returns characters from the user one at a time as keys are pressed on the keyboard. An "ONFUNC" statement allows the programmer to establish a "flow-of-control" sequence to be invoked if any of the function keys on the PLATO IV terminal are pressed. In the event a function key is pressed, a function "FKEY" is available to allow the programmer to determine which specific function key was used.

---

*TUTOR is the programming language used for lesson writing on the PLATO system.

With the construction of the digital cassete drives and development of software for access to program and for data storage, program development support services were placed in the NSBASIC interpreter. Statements which support program storage, retrieval and linkage (DTSAVE, DTREST, DTLINK) were added. Each of these statements accepts two parameters. The first indicates a relative beginning block (each block is 256 bytes long) to access. The second parameter indicates which of the two drives to access. When a program "link" is requested the interpreter will read in the requested program and then begin executing at its first statement. The storage and retrieval facilities provide for control information to be recorded or utilized which specifies the program's length. The program storage and retrieval commands are generally used in "immediate" mode while the linkage facility may be incorporated to move between programs.

Data storage and retrieval is accomplished via the "DTREAD" and "DTWRITE" statements. The programmer specifies the information's location, length (in blocks of 256 bytes) relative block and drive number. The initial use of these statements was in the PILOT translator (described in section IV). Program source written and saved as though it were text, can be read as data translated in memory and written back as data for subsequent use via a DTLINK command.

This same approach was used to support the graphic editor (section V), a program merge utility, macro utility, stand alone 8080 assembler-loader and a source listing program.

IV. Microprocessor-supported PILOT

Following the addition of the string manipulation statements to NSBASIC, an NSBASIC program for translation from PILOT source to NSBASIC source was written. In essence this allows a user of the terminal to utilize the existing editing functions of the NSBASIC interpreter to enter the source of a PILOT program. The same constraints and capabilities as those found in Appendix B can be used in construction of a PILOT lesson. Following the entry of such a lesson, the standard editor supported function for program storage is utilized, and the use of the program linking function allows the initiation of the NSBASIC translator. The translator inputs the source of the previously written PILOT lesson treating it as data and translating it in a line-for-line process into NSBASIC. Following the successful translation of a PILOT program into NSBASIC source, the translation process is not required for subsequent uses of the lesson.

In addition to the core PILOT described by Starkweather in his PILOT-73 statement, an escape to NSBASIC was provided for in the translator. This feature, although not absolutely necessary, did allow immediate testing of the integration of PILOT source with graphic primitives and provided for early testing of concepts which seem appropriate to any implementation of graphics in a PILOT environment.

V. Graphic Source Generation

In the course of experimenting with the stand-alone processor connected to the PLATO IV terminal and following the development of

statements necessary for PILOT translation, a graphic editor was developed much like that found on the PLATO system (2). The microprocessor-based graphic editor provides for cursor movement with the directional keys. The cursor movement may be either in fine steps of one coordinate value per key stroke, or with the use of the shifted directional key 10 coordinate values per key stroke. Also, the cursor may be moved by use of the terminal touch panel. The graphic editor allows the user to identify specific points on the screen and to establish lines between these points. A user may also specify text to be displayed. Also available from graphic editor is the present numeric representation of all of the generated display. If, due to various display options, the display becomes obscured by other information displayed simultaneously, the graphic editor provides for regeneration of the graphic display after initial erasure of the screen. A help sequence is provided for the user explaining each of the options and how to actuate their use. Following the generation of a graphics display under the graphic editor, the user may request that a source be generated which accomplishes the display described to the graphic editor. This source may be used subsequently with an existing program. The integration of such source with existing programs is supported by a merge facility. Figure 3 portrays the steps in the graphic generation process. Figure 4 shows a display generated by the graphic editor which has included the user invoked "help" sequence and shows the subsequently produced NSBASIC source statements.

VI. Sized Writing and User Defined Special Characters

On the PLATO system, a programmer can specify that the display panel be used as a writing tablet for large or small letters. The actual characters written are specified by the standard "write" statement. This writing process is influenced by sizing and rotation commands.

We felt that this capability would be useful on the microprocessor based system. However, the sized writing function requires significant computation during the display process. In order to implement this capability on the microprocessor system with its appreciably lower computational power, a line writing editor was developed which executes the various "DRAW"s and "AT"s in advance, saving the graphic description for use with actual lessons. The description created by this process can be inserted into programs as source. The major functional difference between the preprocessed approach used on the microprocessor and that used on PLATO is the inability of the microprocessor system to display variable information. This is generally mitigated by the tendency on authors' parts to use sized writing for titling and descriptions which don't require an ability to display variable information.

The PLATO IV terminal has the capability for loading user defined special characters. These characters can be developed and edited on the PLATO system. The characters are formed within an 8 x 16 dot area. This facility has been used by lesson writers to enhance textual display (e.g., italics), in animation sequences and in display of special notation (e.g., chemical symbols or mathematical notation). Characters developed for such use are loaded into alternate character memory on the PLATO IV terminal by the central PLATO system.

In order to provide access to this graphic capability of the terminal a combination of microprocessor software (in NSBASIC) and software on the PLATO system was developed. The PLATO system software we developed provides a translation and transmission capability. This software accesses existing character set definitions and translates them into character sequences which are sent via the external output facility of the PLATO IV terminal to the ASCII input channel on the microprocessor. The microprocessor translates the character sequences received in this manner into the appropriate data to be transmitted for loading into the terminal when the specific user defined character set is desired. The data is saved on digital tape and can be loaded into the terminal by use of the NSBASIC "PRINT" statement.

VII. Programming Support

As with larger systems, the microprocessor based system has fostered development of programmer support services as programming has been underway. Support services have fallen into four general areas: aid for the programmer in developing software in NSBASIC, assembly language support for the production of 8080 machine language routines, assists in conversion of PLATO system based lessons to the microprocessor system and the previously mentioned PILOT language translation program.

Services for access to graphic capabilities have followed closely those found on PLATO (see sections V and VI). NSBASIC programs have been developed which support program listing, source statement merging and context editing. The program listing facility has gone through several modifications to allow listing on two different hardcopy terminals with differing control sequences and a local computing system which has a high speed printer. In the latter case, the listing program includes the "job control" statements necessary for the computing system as well as programmed delays to wait for responses from the computing system, thereby providing proper synchronization with activities on that system.

In the process of developing new machine language support routines, it became evident that our dependence on a remote system for machine language programming was not practical. To alleviate this dependence, an assembler-loader for the microprocessor was written in NSBASIC. The source of an assembly language routine is entered and edited in the same manner as standard NSBASIC programs, but is translated by an NSBASIC program and loaded immediately into the memory it is to execute in. The assembler optionally produces a printed listing of the assembly. Errors encountered during assembly are displayed to the user indicating line number and error type.

Both because of the large amount of lessons existing on the PLATO system but appropriate in the microprocessor environment and because both systems use the same display device, we developed software on the microprocessor for aiding the conversion of such lessons from PLATO to the microprocessor. Using a PLATO system lesson which prints lessons on ASCII devices, software was developed which causes the microprocessor to read in the text being converted via one of its ASCII interfaces. The TUTOR source is converted to NSBASIC on a statement by statement basis. "AT" and "DRAW" statements have the minor

syntactic differences between NSBASIC and TUTOR resolved. "WRITE" statements in TUTOR, which contain only constant information, are converted to NSBASIC "PRINT" statements. All other statements are prefixed as "REMark" statements, thereby allowing later conversion by the programmer.

VIII. Commentary

Development of Local System on a Microprocessor

Use of Dr. Tucker's BASIC interpreter has provided a "jumping off" point for local support software. The interpreter has not, throughout the course of our work, been felt to be an end but rather a base from which to extend as local needs became apparent. It has been this willingness to extend the statements supported by the interpreter which has allowed the rapid production of application software. It has allowed us to provide new statements to meet needs on several application and program support fronts simultaneously.

PILOT on the Microprocessor

The NSBASIC source program, which implemented the PILOT translation process, was written and implemented in a very short period of time due primarily to the interpretive nature of the language, and the fact that the string handling statements provided sufficient language power to rapidly translate the source. This same technique was utilized for developing the graphic editor and, to a lesser extent, the 8080 assembler-loader. The language primitives appear to have been the sufficient and necessary ones necessary for the completion of the translation tasks where the language "translated to" provides for a one-to-one mapping of the language "translated from." Had this not been the case, then in all likelihood the requirements in terms of primitives for translation would have been more extensive and required at a minimum table look up capabilities and the maintenance of such tables for variable and label reference.

Graphics in a Microprocessor Environment

An extension to the graphic editor, which seems worthwhile in light of the potentials for use of the microprocessor by an interactive user, is to allow the generation within the source of coordinate translation and scaling information. Following the specification of such information, the subsequent source generated can be utilized with multiple references to cause output in different areas of the screen by respecification of translation and scaling variables' values. Such a capability would not require internal modification of the actual source statements which would be more tedious than parameterization of translation and scaling information.

IX. Summary

A system has been developed which allows access to remote systems: PLATO and ASCII based communication systems. The system also has facilities for local production and use of PILOT lessons, for support of a generalized programming language (NSBASIC) and for

development of graphic sequences. The capabilities required for both a stand alone CAI system oriented towards a single user and for a system which provides access to remote CAI systems has been realized.

***

The authors wish to acknowledge the support of research assistants in Medical Computing Laboratory who aided the developments described here. Mr. Tom Szolyga developed both the PLATO IV - 8080 hardware interface and much of the translation software. Mr. A. B. Baskin has aided in clarifying portions of input/output control code and in describing the theory of the translation process to/from ASCII and the PLATO IV terminal.

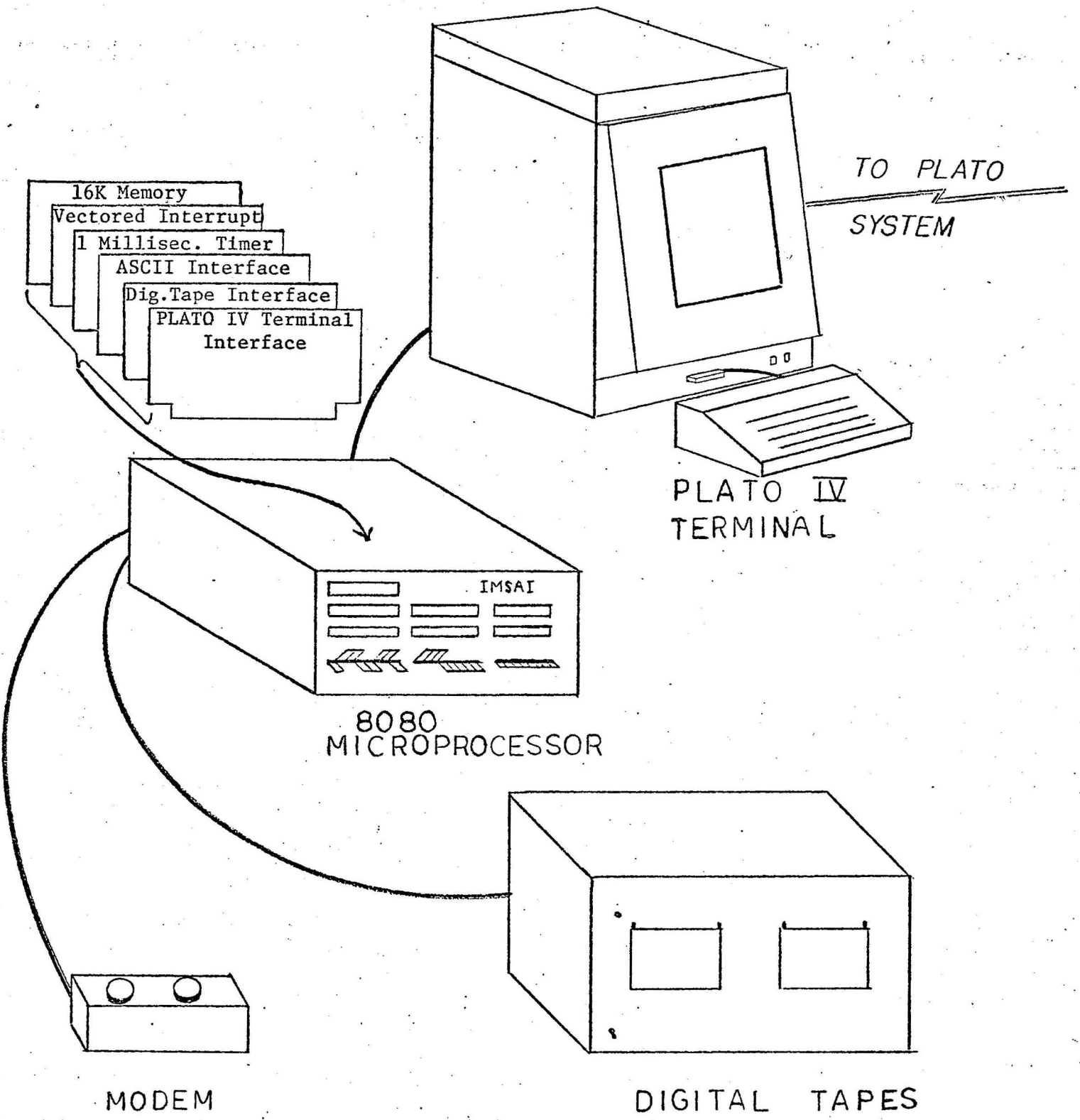


Figure 1

System Hardware Configuration

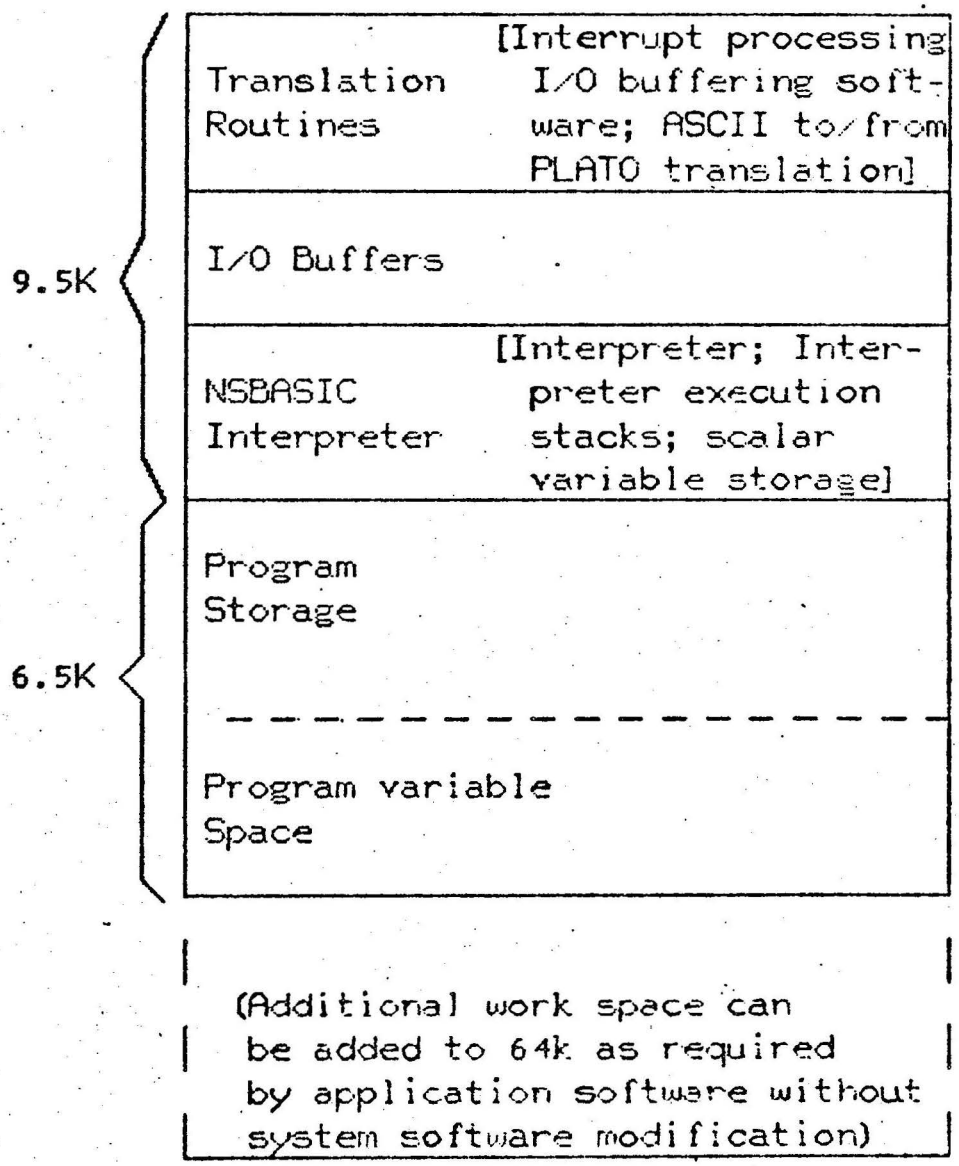


Figure 2.
Microprocessor Memory Configuration.

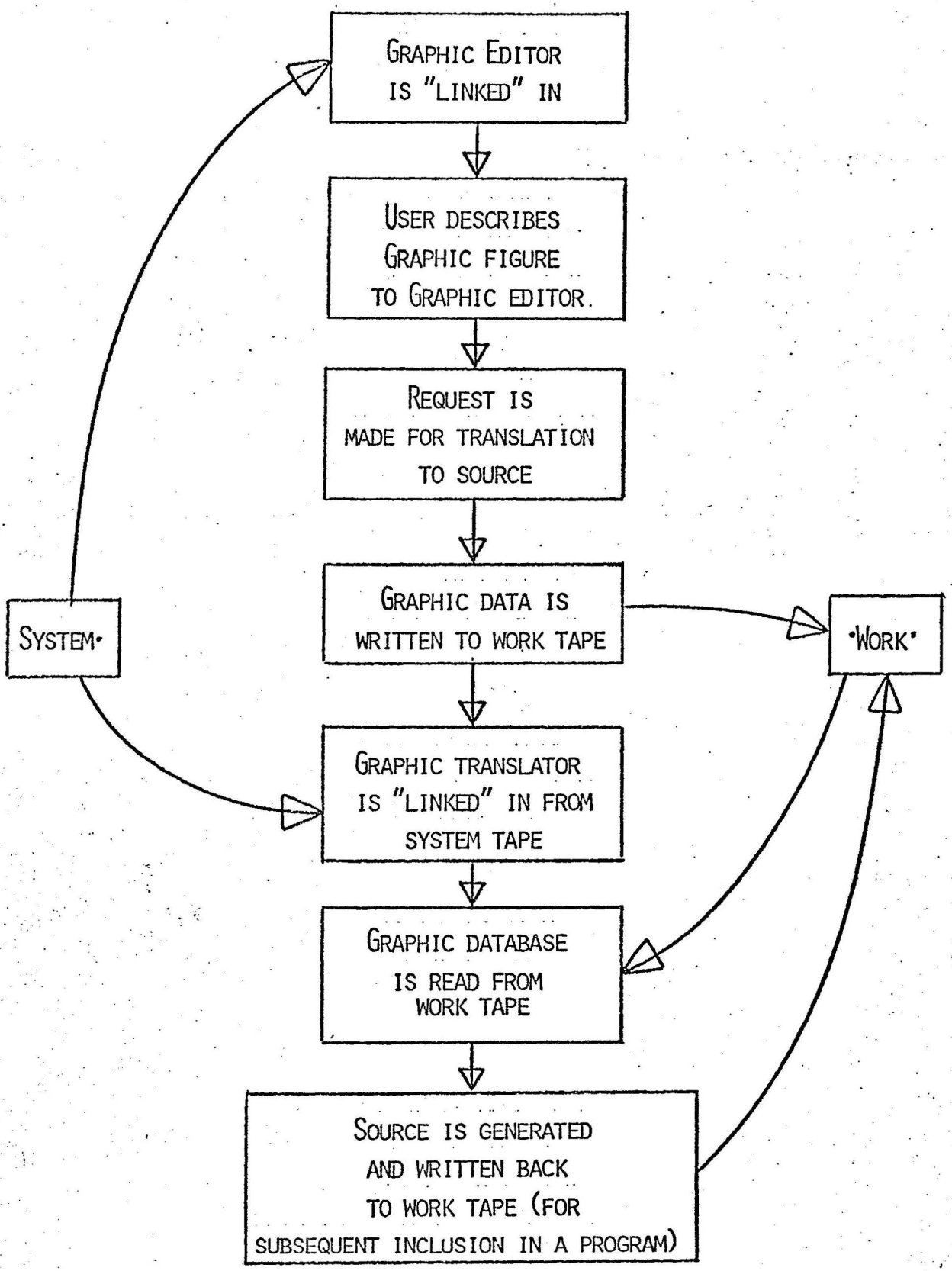
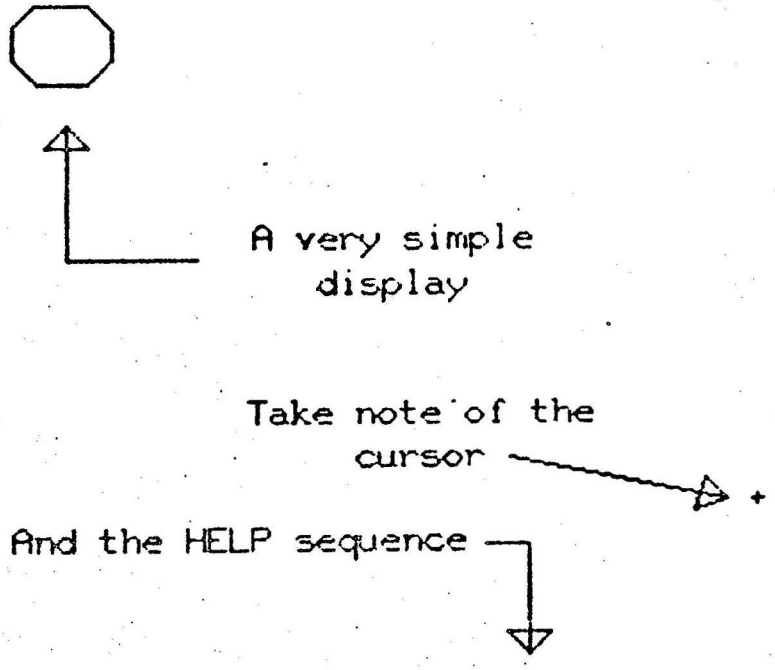


Figure 3

Steps in the Graphic Generation Process



DIRECTIONAL KEYS AS MARKED
'R'redisplay, 'I' for information, 'T' for text
'S'show present coords, 'L' line, 'P' point
'/' delete last graphic element; DATAI source generation option; Touch is available for rapid cursor movement

- a -

```
30 AT80,402:DRAW(88,410;108,410;117,401;117,390;108,381)
32 DRAW(89,381;80,390;80,401):AT100,366:DRAW(92,358)
34 DRAW(108,358;100,366;100,318;148,318):AT169,318
36 P."A very simple":P."    display":AT167,255
38 P."Take note of the":P."       cursor":AT351,223:P."+"
40 AT263,246:DRAW(343,230;335,238;331,226;343,230)
42 AT80,208:P."And the HELP sequence":AT254,214
44 DRAW(270,214;270,174;262,182;278,182;270,174)
```

- b -

Figure 4.
a - Graphic Editor Display
and "Help" Sequence.
b - The NSBASIC Source Generated
for 4-a by the Graphic Translator.

References

1. Stifle, Jack. The PLATO IV Terminal: Description of Operation. Urbana: University of Illinois Press (1974).

2. Sherwood, Bruce A. and Smith, Stanley G. "Educational Uses of the PLATO Computer System." Science (1976) Vol. 192, pp. 344-352.

3. Levy, Allan H. and Williams, Ben T. "Experiences with Health Sciences Education Using PLATO." Proceedings of the Society for Computer Medicine Conference, November, 1976.

4. Chen, Thomas T., Levy, Allan H. and Williams, Ben T. "A Depository Health-Computer Network." Medical Information (1973) Vol. I, No. 3, pp. 167-178.

5. Starkweather, John. PILOT Guide. USC 03.01.03 (1976) USSF Computer Center.

6. The Digital Group Cassette Storage System, CSS-RO, The Digital Group, Inc., Denver.