MIDAS: A Microprocessor Display and Animation System

by

Robert F. Gurwitz Read T. Fleming

and

Andries van Dam

November 19, 1979

Department of Computer Science
Brown University
Box 1910
Providence, Rhode Island 02912

TABLE OF CONTENTS

ABSTRACT

An interactive graphics program has been developed to simulate and animate the operation of a typical microcomputer system. MIDAS, A Microprocessor Interpreter Display and Animation System, allows the user full control over the simulation and the display, as well as several auxiliary functions that enhance its capabilities as an instructional tool. The illustration of the activity of the computer, based on the Intel 8080 microprocessor, takes the form of an animated block diagram of the CPU and its peripherals. It shows the operation of the system at various levels of detail, down to the level of the devices' internal registers, buffers, control lines, and busses.

This paper describes the design, implementation, and use of MIDAS. It discusses its effectiveness as a tool for teaching the complex, asynchronous interaction between devices of a computer system, known as "handshaking." It also discusses a strategy for developing a generalized tool for simulating/animating arbitrary computer systems.

CR Categories: 1.5, 8.1, 8.2

Keywords: computer aided instruction, computer animation, computer science education, interactive graphics, microprocessors, simulation

1 Introduction

MIDAS, a Microprocessor Interpreter Display and Animation System, provides its users with a real time simulation of a microcomputer system, an animated display of the simulation, and facilities for controlling their presentation. It is intended for instructional use, to acquaint students with the operation of a typical computer system based on the most popular production microprocessor, the Intel 8080. It allows viewing of this operation on the register and bus transfer level. MIDAS is comprised of three parts. First, there is a discrete simulation of the microprocessor system, including the CPU, memory, a console, DMA controller, floppy disk controller, interrupt control unit, bus control unit, and a keyboard interface. Second, animation of the simulated system's block diagram is controlled by a display processor which is driven by the simulator. Finally, a supervisor allows interactive control over the operation of the simulated computer, as well as such aspects of the display as rate of presentation, view of the diagram, and level of detail of the display. There are utility functions provided which are accessed through a simple command language and interpreted by the supervisor.

MIDAS is an interactive system that attempts to let its users follow the complex activity of a working computer at their own pace, and at a level of detail that makes the operations

understandable. In order to do this, it provides special facilities that are geared toward allowing a high degree of user control in as uncomplicated a manner as possible. This paper describes these facilities in detail. The motivation and design goals behind the MIDAS project will be discussed, along with the features the system offers, its implementation and the degree to which it fulfills the stated objectives. It also examines MIDAS in a more general context, as an example of a system which employs techniques of presentation and interaction that are fundamentally different from those used in classical instructional tools.

2 Background

The original motivation for the design and implementation of MIDAS came from a desire to illustrate the complex interactions that exist at the level of individual hardware control lines during the operation of a computer system. This "handshaking" is a dynamic process that is difficult to teach using conventional methods. Typically, these methods have included the study of manufacturer's specifications and operating manuals that employ static block diagrams, waveform timing diagrams, and narrative descriptions to convey the precise operating characteristics of hardware. Since many of these processes run asynchronously, it is difficult to describe them adequately with standard illustrative techniques. From our experience, both in trying to teach these concepts in computer architecture courses, and in attempting to learn about communications protocols, interrupt handling, and I/O in various systems ourselves, it became apparent that real-time graphics would help the user to visualize these ideas. It was felt that by combining a low-level simulation of a hypothetical computer with an animated block diagram of the system in operation, we could develop dynamic methods of illustrating handshaking and asynchronous control. Our experience with real time, interactive, vector graphics [BUGS76] suggested that placing this simulation and animation under direct user control was the best way tc proceed [STRA74].

Computer animation for various purposes is not a new concept. The work of Knowlton [KNOW65], Baecker [BAEC75], and Hopgood [HOPG74] describe applications of computer animation and computer production of animated films, and demonstrate the feasiblity of such systems. For example, a method of illustrating LISP programs has been developed at the University of British Columbia [DION75] and used for teaching with some success. However, to our knowledge, no other current system exists that animates the low-level operation of computers and peripherals in real time, with a high degree of user control.

MIDAS is an experimental implementation of this type of presentation. It was decided that this system be designed for use in the introductory computer architecture course at Brown. It was also felt that the simulation of a microprocessor based system would be the most appropriate subject for this treatment. Reasons for this included the fact that microprocessor systems are small and hence suitable for an initial implementation under time and size constraints, and their increasing proliferation in a variety of applications would make such an implementation useful. The Intel 8080 [INTE75] was chosen as the CPU since it was one of the first, and most well known of the microprocessors in current use. It also has a wide enough range of features in the areas of interrupt handling and I/O to be useful for illustrating handshaking. The system to be simulated around the CPU was designed to contain devices typically found in microprocessor

based computers. It will be described in more detail in section 3.1.

Several design constraints were defined. The system had to operate slowly enough to be comprehensible. For teaching purposes, the operator would have to be able to stop it at any time and be able to have some "instant replay" facility. The contents of devices' registers, as well as the status of all control lines and the progress of bus transfers would have to be depicted. Because a block diagram at such a low level could become cluttered, steps had to be taken to allow the activity taking place on the display to be followed by a student without confusion. The animation had to be devised so as to attract the viewer's attention and highlight important events as they take place. Finally, two capabilities were deemed essential to the goal of making the display "scrutable": the ability for the user to smoothly "pan" over the block diagram, and similarly to be able to smoothly "zoom in" on individual devices while having lower level details of the picture come into view. These logical zooming and panning capabilities would allow the user to view the action at any of several levels of detail, while concentrating on individual areas of interest in the diagram. The notion of representing a picture on multiple levels of detail has its antecedents in classical parts "explosion" diagrams, and more recently, in text manipulation systems. These allow authors to structure text in such a way as to allow access to multiple

representations, hiding detail as desired [NELS72]. This technique has also been used in applications where data bases are explored graphically, on a number of different levels [BOLT78].

3 Overview

3.1 THE SIMULATION

The simulation part of MIDAS consists of an incremental (finite state) simulation of a hypothetical system built around the Intel 8080 microprocessor. The other devices in the system have been designed to simulate the activity of "typical" currently available hardware, and thus correspond to no specific commercial devices. In addition to the CPU and 1K random access memory, these include a status decoder, console, keyboard interface, direct memory access (DMA) controller, floppy disk controller, priority interrupt control unit (PICU), and bus control unit (BCU). They allow depiction of DMA and peripheral interface activity, as well as CPU-memory operations. The devices are tied together by an 8-bit bi-directional data bus, and a 16-bit address bus. The configuration (showing devices and interconnecting busses, but not control lines) is illustrated in figure 1.

The CPU's entire 78 instruction repertoire is implemented. All programmer accessible registers and flags, as well as some that are not directly accessible under program control, are maintained by the simulation and displayed, as are the 8080's control lines. The 8080 is doubly clocked, with two clock pulses defining a clock state. A varying number of

clock states are required to form any of ten 8080 machine cycles. These cycles perform basic operations such as instruction fetch, and memory, I/O, and stack reads and writes. One or more of these cycles make up the execution of an instruction, requiring from 4 to 18 clock states to complete. The simulation and display processes of MIDAS are closely tied to these clock states. MIDAS simulates the microcomputer system at the state level; that is, one clock state is the smallest time unit that the simulation deals with. This results in occasional compromises in depicting certain activies (e.g., "pulses" last an entire state), but an effort has been made to retain a high degree of accuracy in representing the relative timing of all operations.

A status decoder in the system derives I/O and memory read and write signals from a status byte sent out on the data bus at the end of clock state T1 of every machine cycle. This byte is unique for each of the ten different machine cycle types, and encodes the information needed for these control signals. The console simulates a programmer's panel, similar to those found on most minicomputers. It contains a switch register and several addressing functions which are tied to function keys available to the user. The console can perform DMA operations (read and write memory with the CPU locked out). A keyboard interface allows data to be typed into the system from the user's alphanumeric keyboard. Thus, interrupt

driven character communication, such as teletype handling, can be simulated. A DMA controller and floppy disk controller allow disk data to be read and written directly from memory. The simulated disk has 32 tracks each with eight 128 byte sectors, and can accept commands to seek and format tracks, and read, write, and verify sectors. Its controller communicates with the DMA controller via its own 8 bit bi-directional data bus (independent of the system data bus). A PICU accepts interrupts from the keyboard interface, DMA controller, and the console. It has a mask register which allows device interrupts to be held pending. Vector and control registers within the PICU allow different interrupt priorities and interrupt handling sequences to be programmed. Finally, a BCU arbitrates bus access requests by devices other than the CPU which can perform DMA operations. These requests are accepted from the DMA controller and the console on a simple priority basis.

3.2 THE DISPLAY PROCESSOR

Tied to the simulation is a display which animates the operation of the system. The display consists of a block diagram of the simulated computer. Each device in the system is represented by a box containing the device's registers and flags. Single bit control lines, represented as thin single

lines, tie devices together. Busses, shown by double lines of different widths, indicate 8 bit data and 16 bit address paths between devices. Data in the system is represented by hexadecimal digits contained in registers or flowing across the busses. The display is organized so that elements are shown in various levels of detail. The degree of detail is controlled by the user with a dial. As the user turns the dial, the diagram "zooms in" smoothly and continuously, until a new level of detail pops into view. At the top level the boxes, busses and some control lines are shown. As the diagram zooms in, line identifiers appear, followed by high level registers and flags, and finally internal registers. In addition to smooth zooming in or out, the user can also "pan" over the block diagram with the use of a joystick.

These effects are shown in figure 2, where the sequence of three photographs shows an area of the block diagram displayed at three successive levels of logical "zoom". Figure 2a shows the overall block diagram at the highest level. In figure 2b, we have zoomed in on the CPU and its neighboring devices. At this level, the line identifiers have appeared. Finally, figure 2c shows a detailed view of the same area "zoomed in". Here, the registers and flags of the various devices can be seen. Note that the SYNC line from the BCU to CPU are active (dark) in the figures, while all other control

lines are not. Also, note the presence of data on the busses in the sequence.

In order to show the activity of the simulated computer in operation, a repertoire of several animation effects is used. Bus transfers are depicted by the movement of characters representing the hexadecimal equivalent of the data moving across the bus paths. Because the display is tied to the CPU's clock states, the movement of data is arranged in such a way that bus transfers are completed by the end of the state they start in. To preserve the accuracy of the simulation, all transfer paths terminate in registers that represent the availability of data on the bus. These registers may correspond in some cases to latches which hold the data until it can be gated to its destination register in the following clock state. The contents of all registers are updated once each clock state to insure that they display the correct data by the end of that state. Changes in an individual register's contents are highlighted by blinking the appropriate data in the display. Activity on control lines is shown by brightening line intensity to indicate a high or active line. All lines, registers, and devices in the diagram are clearly marked by identifying legends.

3.3 THE SUPERVISOR

When the system is activated, the simulation and display processor are invoked by the supervisor on a state by state basis, with the simulation updating status information maintained for each device and the display processor then illustrating the changes with the appropriate animation effects. Several status displays alongside the animated block diagram indicate the current clock state, machine cycle type, and the mnemonic of the 8080 instruction being executed. There are three operating modes. In normal mode, the simulation operates continuously at a speed controlled by the user. The system can also be run one clock state at a time, in single step mode, with the user hitting a function key for each state. The third, or fast run mode, allows the simulator to run at higher speed for a number of clock states specified by the user. In this mode, animation effects are abbreviated, with the displayed registers and lines updated, but no bus transfers shown. During these three modes, a set of user functions are available through a function keyboard, for controlling the display directly. Commands to invoke utility functions can be entered from an alphanumeric keyboard when the simulator is stopped, a state known as command mode.

The auxiliary functions are available to the user through a simple command language. Commands are available to enter one

of the operating modes described above. In addition, while the simulator is stopped, the user may issue commands to display or alter the contents of simulated memory, alter the contents of any device's registers or the state of any control line, clear system memory, or reset the status of the entire system (registers, control lines) to start-up values. The contents of simulated memory may be saved in or loaded from a specified disk file. Similarly, the state of the entire simulated system, including the contents of all registers and control lines, may be saved in or recalled from a disk file. This feature allows events to be saved and "replayed" at any time.

As mentioned earlier, several commands are available which may be issued while the simulator is running. These commands are tied to function keys. They include commands to stop the simulation and enter command mode; to "freeze" the display during a simulated clock state; and to lock the joystick, which controls panning, over any position of the display to concentrate on a specific device or area. A set of predefined views of the diagram may be tied to function keys so that they may be recalled at the touch of a button. The simulator is automatically "backed up" for one state, allowing "instant replay" of the previous state at any time. This is especially useful in single step mode. Finally, a group of keys is dedicated to controlling functions of the simulated console.

4 Implementation

MIDAS has been implemented on the Brown University Graphics System (BUGS). BUGS is a stand alone system currently consisting of two Digital Scientific Meta4 16 bit user microprogrammable minicomputers, a Vector General CRT display, and a high speed graphics processor, the SIMALE, designed and built at Brown [WEBB73]. The minicomputers are used for general purpose computation and for graphics processing. One has the Vector General display and its interaction devices (joystick, analog dials, function keys, light pen, data tablet, and alphanumeric keyboard) as peripherals. The SIMALE was designed as a high speed processor used for real time graphics transformations, windowing and clipping, image correlation, and interpretation of graphics primitives (lines, text, etc.). It employs four parallel ALUs and has an effective instruction cycle of 38 nsec. BUGS offers vector manipulation facilities for a large variety of applications, particularly those requiring sophisticated real time interaction, arbitrary 3-D transformations, and analog inputs for display control (e.g., MIDAS-like smooth windowing and zooming on a large drawing) [BUGS76]. A diagram of the configuration of BUGS is shown in figure 3.

The implementation of MIDAS is comprised of three parts: the simulator, the display processor, and the supervisor. The simulator maintains a status record which holds all information

on each device of the simulated system. It is invoked once per simulated clock state to update this status record. Communication between the simulator and the display processor is done entirely through the status record and a set of pseudo-display orders. These orders specify the animation effect to be performed and the subject of the operation. The animation operations include: move for illustrating bus transfers, flash and change for changing the contents of a displayed register or flag, and brite and dim which are used for changing the status of control lines. The objects of these orders are the different modifiable entities in the display: the busses, registers, flags, and control lines. In each simulated state, the simulator constructs lists of these orders which are interpreted by the display processor to update the display. The supervisor invokes the simulator and the display processor for each clock state. Modification of the display resulting from auxiliary function commands (like resetting the system, altering of system registers, reading in a new status record, or "instant replay") cause the supervisor to update the status record and invoke the display processor with its own display orders. The relation of the components of the implementation is shown in figure 4.

5 Discussion and Conclusions

As a result of our initial experience with MIDAS as a teaching tool, several conclusions about the techniques used have been made. We have found that MIDAS reduces the time required for teaching concepts of handshaking and asynchronous control in general, and the architecture of the Intel 8080 specifically, compared with conventional methods (study of manufacturer's literature, classroom lectures, etc.). A significant reduction in teaching time over conventional classroom lectures has been observed, when MIDAS is used with properly prepared students. This preparation should include an introduction to the organization of the 8080, relevant terminology, and the functions of the peripherals in the system. The main reason for the success of MIDAS as a teaching tool seems to be the use of user controlled animation. The system attracts students' attention, and the control facilities available for instant replay, single step running, freezing the display, etc., are a great help with instruction. The use of descriptive messages, such as cycle type descriptions and instruction mnemonics also help in understanding the action. The ability to hide and smoothly reveal different levels of detail at his own pace, greatly helps the student visualize the operation of the system in stages, making it easier to introduce new concepts.

Several factors were found to be important in developing this type of dynamic instruction system. The display must be well organized to reduce clutter and confusion. Animation effects must be carefully chosen to highlight relevant areas of the display. The speed of such an animation must be slow enough to avoid confusion of the viewer. Interactive speed control is a great help in determining the speed appropriate to each user. Adequate preparation of students with introductory lectures and study greatly enhances MIDAS' effectiveness. The system as it now stands cannot be used to teach these concepts "cold," without prior preparation of the student. It is definitely not a replacement for classroom teaching. Rather, it should be viewed as an adjunct, providing concrete reinforcement of ideas taught in the classroom. In our experience, the system was operated by a teaching assistant for a group of four or five students gathered around the display. The system is also effective when students are given subsequent "hands-on" experience.

Being an initial implementation, there are several improvements that can be suggested. We have found that the system is not fully self-disclosing. Better methods, such as the use of color for highlighting, would improve its ability to draw the user's attention to the right places in the display. At present, there is too much activity taking place in the display for a student to follow it totally unassisted. The use of raster-scan graphics in future implementations of MIDAS or similar systems

might facilitate color, but would restrict the dynamics presently available with high performance vector graphics. The effects of such restrictions on the quality of the presentation and possible ways of compensating for them, remain areas for future study.

Another question that needs to be answered is how MIDAS compares with more traditional teaching methods. The reduction in teaching time over classroom lectures mentioned earlier provides only one basis for comparison. Some reduction in time would also probably be observed if MIDAS is used in conjunction with self study programs. In contrast with other educational aids, MIDAS has several advantages. Hardware development systems that include real hardware systems and various programming and debugging aids, and are specifically geared toward education, are available through manufacturers. These systems give students the kind of hands on experience no simulator can duplicate. However, aids such as these cannot illustrate the internal data transfers and sequence of operations as MIDAS does. More importantly, MIDAS allows the user to exercise a high level of control over what he sees, and the speed at which things proceed. As a result, he is able to master concepts more easily by being in control of the process which presents them. Ideally, a system which combines the feeling of working with actual hardware (with the attendant real signal level and timing considerations), with a tool having MIDAS' illustrative power and user controls, would have the most to offer in terms of education.

Other conventional teaching tools, such as "canned" lessons on film or videotape, sometimes employ animated diagrams to illustrate important concepts. More often, they are just variants of classroom instruction transferred to new media. In any case, the ability for the student to control the presentation is not available. Another advantage of MIDAS over these methods, is the fact that presentations can be easily tailored to the students' needs. While it is true that the implications and capabilities of MIDAS has not been fully explored, the potential utility and advantages of it seems clear.

MIDAS is a system designed around the Intel 8080 microprocessor. However, it could just as easily have been based on another processor. One possibility for overcoming the fact that MIDAS is "hardwired" for the presentation of a single computer system, would be to combine its capabilities with a means for generating new simulations and animations of other systems. These could be produced by the modification of a hardware specification language to include animation instructions. For example, ISP (Instruction Set Processor) is one such language: a register transfer language for specifying the instruction set architecture of digital processors [BARB77]. Simulators for descriptions written in this language currently exist. By adding animation constructs to such a language, and using a simulator for it to drive a MIDAS-like display, an extensible system could be developed.

The extensibility of its techniques to digital electronics systems other than microcomputers, or tasks which are not purely instructional are also feasible. Some of these useful applications might include testing of proposed system designs, communications protocols, and other development tasks where low level design and debugging are necessary. For example, a byproduct of the MIDAS simulation is the ability to do machine language debugging in a straightforward manner on programs which do asynchronous input and output. Another possible use is in the maintenance and repair of digital electronics systems. It is not hard to envision MIDAS, or systems like it simulating and animating other computer systems for use by students, engineers, or customers needing to learn about specific machine architectures or general operating concepts. The applicability of a MIDAS-like presentation to areas which are outside the world of digital electronics, but also have multiple asynchronous activities, can also be imagined. The same advantages it has over conventional methods of instruction might also be found in simulating analog electronic circuits, mechanical systems, or even more abstract systems (such as the flow of data in a computer program, or in an office).

In conclusion, we believe MIDAS represents a significant advance over conventional teaching methods. While the techniques it employs need further development and problems of cost effective implementation remain, the need for advanced methods of

technical education, both in schools and in industry, seems to make such development worthwhile. Ultimately, extension of these techniques to a more general purpose tool for simulating and animating would be desirable. A system allowing interactive animation programming, like Xerox PARC's SMALLTALK [LEAR76], and MIDAS-like simulation and user control would be an example of this. Future work needs to be done to exploit the full potential of these methods, and make MIDAS-like educational aids a practical reality.

6 References

[BAEC75] Baecker, R.M., "Two Systems Which Produce Animated Representations of the Execution of Computer Programs," SIGCSE Bulletin 7:1, 1975. pp. 158-67.

[BARB77] Barbacci, M., G. Barnes, R. Cattell, and D. Sieworiek, "The ISPS Computer Description Language," Department of Computer Science Technical Report, Carnegie-Mellon University, August, 1977.

[BOLT78] Bolt, R.A., Spatial Data-Management, Architecture Machine Group, MIT, 1978.

[BUGS76] Brown University Graphics System, The Brown University Graphics System Overview, Computer Science Program, Brown University, 1976.

[DION75] Dionne, M.S. and A.K. Mackworth, ANTICS -- A System for Animating LISP Programs, Dept. of Computer Science, University of British Columbia, 1975.

[HOPG74] Hopgood, F.R.A., "Computer Animation Used as a Tool in Teaching Computer Science," Proc. 1974 IFIP, North Holland Publishing, Amsterdam, 1974. pp. 889-92.

[INTE75] Intel Corp., Intel 8080 Microcomputer System's Guide, Intel Corp., Santa Clara CA, 1975.

[KNOW65] Knowlton, K.C. "Computer Produced Movies," Science 150, 1965.

[LEAR76] Learning Research Group, Personal Dynamic Media, Xerox PARC, March, 1976.

[NELS72] Nelson, T., Computer Lib, Chicago, 1972.

[STRA74] Strauss, C.M. and T. Banchoff, "Computer-Encouraged Serendipity in Pure Mathematics", IEEE Proceedings, Special Issue on Computer Graphics. Spring, 1974.

[WEBB73] Webber, H.H., "SIMALE" SIGMICRO Newsletter, 3:4, January, 1973. pp. 25-44.

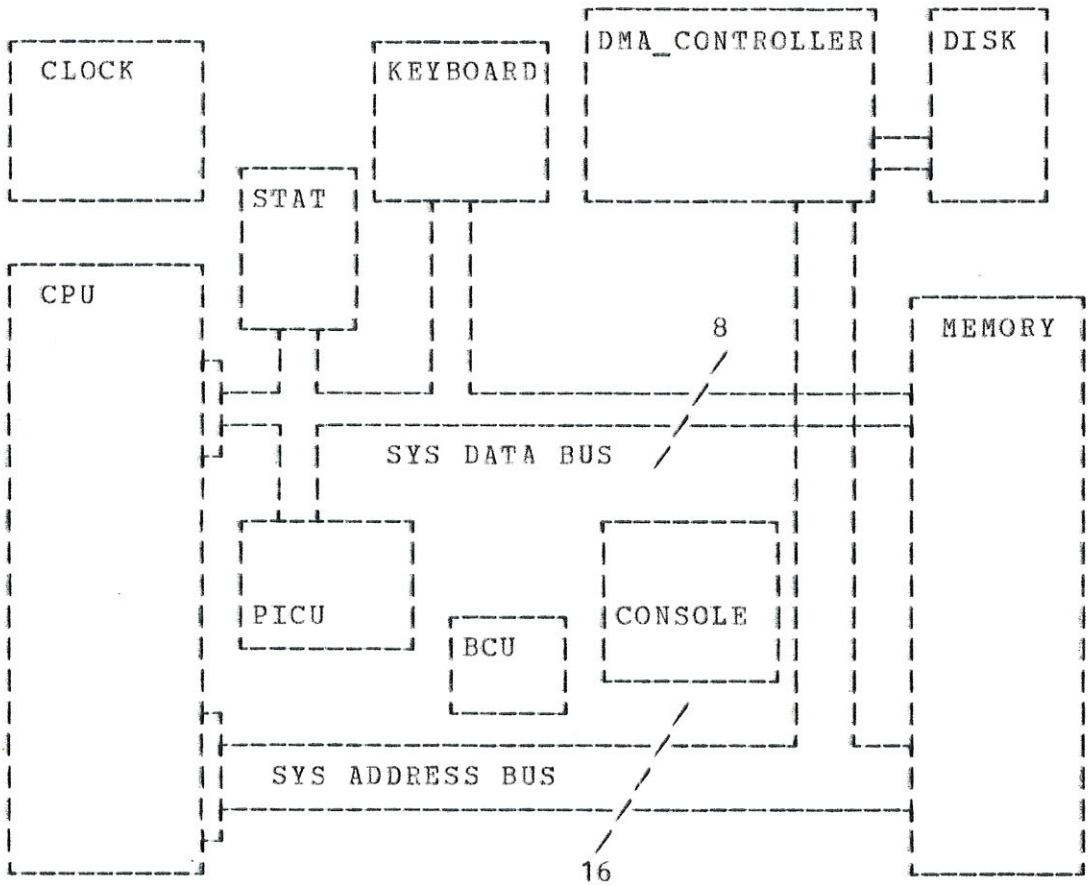

FIG. 1: MIDAS BLOCK DIAGRAM

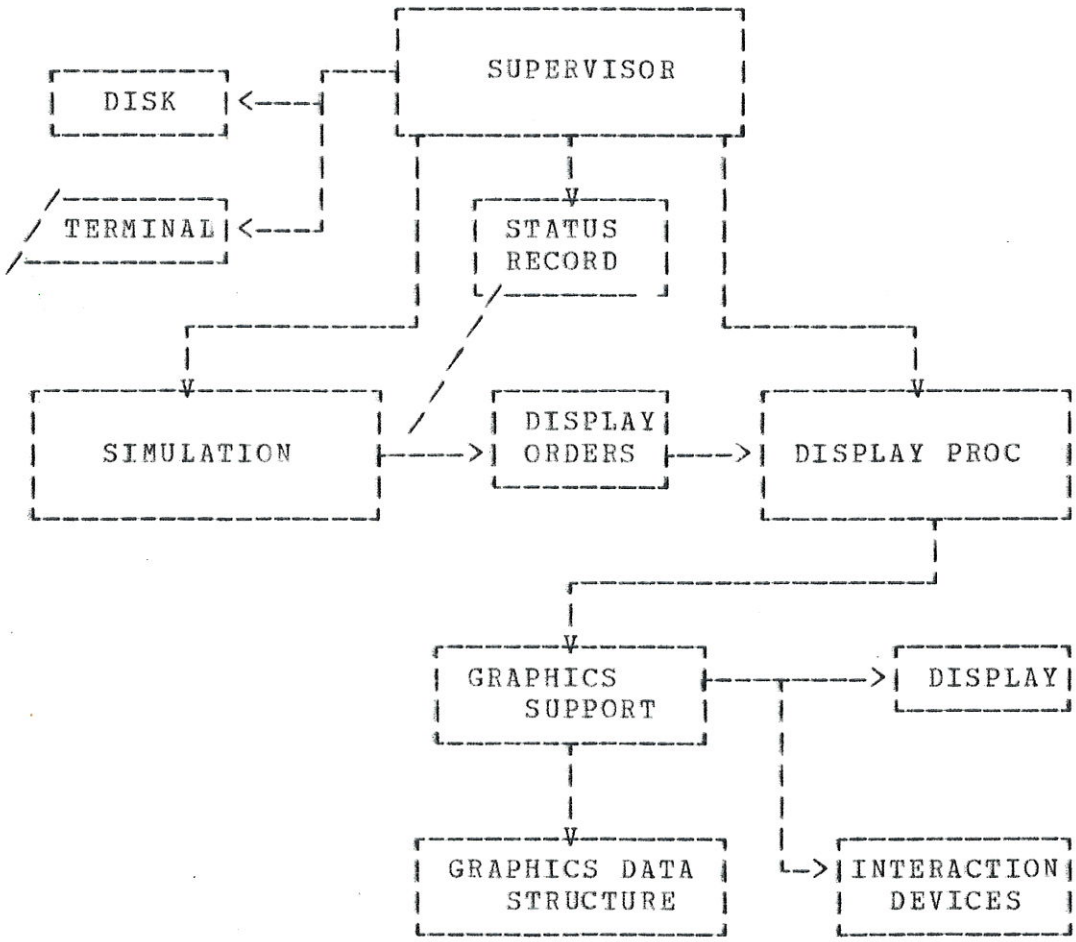

FIG. 4: MIDAS IMPLEMENTATION

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

EDITORIAL DEPARTMENT--11TH FLOOR

345 EAST 47th STREET, NEW YORK, NEW YORK 10017

DIRECT NUMBER
(212) 644-7585

November 5, 1980

Mr. Robert Gurwitz
Bolt, Beranek & Newman
50 Moulton St.
Cambridge, MA 02238

Re: Title MIDAS: A Microprocessor Display and Animation System

IEEE XXXXXX/J. Education - Feb.'81

Dear

Your paper is scheduled for publication. To complete your manuscript kindly supply the material indicated:

____ 1) Abstract (125-200 words)

XXX 2) Biography (Birth date and place, education, employments, special fields of research, and membership in other professional societies.) On Read T. Fleming

XXX 3) Author(s) Photograph (Not to exceed 9-1/2 cm (3-3/4 in) across the widest part of the head.) Of Read T. Fleming

XXX 4) Figures (Original or good reproducible copies.) Fig. 3. The Brown University Graphics System.

____ 5) Additional Material

Please send immediately Read T. Fleming's biography and photograph; and an original Fig. 3. The Brown University Graphics System.

When submitting any requested material please identify by author, title, journal, and month.

To maintain publication schedules, your immediate attention will be appreciated. Kindly return this form with your reply.

Very truly yours,
C. Elenowitz
Production Manager