```
┌────┼────┐
| B U G S |
└────┼────┘
  MIDAS
```

MIDAS User's Guide

by

Robert F. Gurwitz and Read T. Fleming

Revised: October 21, 1977
Printed: November 14, 1977

Program in Computer Science

Box F

Brown University

Providence, RI 02912

TABLE OF CONTENTS

1 INTRODUCTION

The purpose of this manual is to describe the use of MIDAS, a Microprocessor Interpreter and Display and Animation System, which provides the user with a real time simulation of a microcomputer system, and graphical display of that simulation. The system is intended for instructional use, to accquaint the user with the workings of a typical computer system based around a production microprocessor, the Intel 8080. It consists of a discrete simulation of such a system, and a display program which gives the user a real time animation of the system's operation. MIDAS provides facilities for the user to control both the simulation and the features of the animation interactively. What follows is a user's guide to the operation of MIDAS. The user is expected to have some familiarity with the instruction cycle of the Intel 8080 and its instruction set. A set of operation codes is provided in Appendix B.

In addition to the simulation and display parts of the system, MIDAS also has a supervisor which performs utility operations important in the use of the system. These functions include initialization of the system, a command processor for a small command language which is the user's interface with the supervisor, and programs which enable the user to save, restore, and modify simulated memory, registers, flags and lines. It also allows real time operation of the simulation, which facilitates operations which would be infeasable in normal operation. The supervisor also performs disk I/O, needed for saving and restoring system status at any time.

MIDAS is implemented on the Brown University Graphics System (BUGS). BUGS is a stand alone system consisting of two Digital Scientific Meta4 microprogrammable minicomputers, a Vector General display, the SIMALE (a high speed parallel processor, designed and built at Brown, that is used for real time graphics transformations, windowing, perspective division, correlation, and other operations), and miscellaneous peripherals. The system offers vector manipulation facilities for a large variety of applications, particularly those requiring sophisticated real-time interaction, arbitrary three dimensional transformations, structured display files, and analog inputs for display control (e.g. smooth windowing and zooming on a large drawing, etc.). For further information on BUGS, see [3]. The user is assumed to be familiar with the basics of operation of the system.

2 SYSTEM OVERVIEW

2.1 MOTIVATION

MIDAS is an instructional system intended to be used in an introductory computer architecture course studying the 8080, or by those who wish to familiarize themselves with the operation of the 8080 specifically, or microcomputer systems in general. It is particularly well suited for illustrating those asynchronous processes that take place during the operation of a computer system and its peripherals. It demonstrates the sometimes complex interaction between devices at the level of individual control lines, known as "handshaking." It has been observed that these complex, dynamic processes are better taught using the dynamic animation methods employed by MIDAS, than with conventional teaching techniques (chalk-talks at the blackboard, study of manuals with block diagrams and waveforms). Just as one picture is worth a thousand words, we are finding that one dynamic picture is worth a thousand static pictures.

Because of the great detail of activity taking place in the animation, it is suggested that MIDAS be used in a laboratory environment operated by an instructor experienced in its use, and only after suitable classroom preparation. That is, students should not expect to learn about the 8080 and handshaking by seeing MIDAS "cold," with no prior preparation. It may also be useful for the student to operate MIDAS on his own after he has learned about the basic instruction cycle of the 8080, and is familiar with the operation of the peripherals.

2.2 THE DISPLAY

The system's basic display consists of a block diagram of the simulation (see figure 1, section 3.2). MIDAS' display facilities allow the user to smoothly zoom in on, or pan over any part of the diagram interactively. As one zooms in on a portion of the schematic, finer levels of detail come into view. This allows the user to view the system on multiple levels, from a macro overview of the entire operation, to a detailed view of the internal registers and flags of any device. A facility exists for saving different views of the schematic and recalling them at the touch of a function key.

During normal operation of the simulation, the basic diagram is animated to show bus transfers (actual data is shown moving across the busses), register and flag contents and modification, and control line status. The speed of the animation can be controlled by the user with a dial. Thus, the user can follow the sequence of events at his/her own pace. In addition, full zooming and panning capabilities can be used during this animation. Facilities are also available to freeze the animation at any time, single step through it, and replay states of the simulation. The command language allows the user to access other utility functions described in following sections.

2.3 THE SIMULATION

The simulation part of MIDAS consists of a discrete (finite state) simulation of a hypothetical system built around the Intel 8080 Microprocessor. This simulation updates a data base of system status shared with the display processor, consisting of simulated memory and a "status record" in which information necessary for the simulation (register contents, flag and line status, and control information) is stored. There is status information for each of the devices in MIDAS. In addition to the CPU and memory, these include a Console, Clock, Keyboard Interface, Bus Control Unit (BCU), Status Decoder, Floppy Disk Controller, Direct Memory Access Controller (DMAC), and Priority Interrupt Control Unit (PICU). For further description of these devices see section 5. They allow depiction of DMA and peripheral interface activity, as well as CPU-Memory operations. While the simulated CPU is based on an actual production device, all other devices have been designed to simulate the activity of "typical" currently available hardware and correspond to no specific commercial devices.

The status information maintained by the system for each device is such that their operation can be simulated and displayed on a highly detailed level. This level extends to the state of all external registers and control lines of each device, but does not show the internal operation of the device. (For example, the internal bus transfers of the CPU are not simulated.) However, an attempt has been made to retain a high degree of accuracy in representing the relative timing of all operations.

3 FACILITIES

This section describes the various facilities available to the user in MIDAS. It concentrates on the modes of operation, the basic display that MIDAS creates, the various input devices, and the animation effects used to illustrate the simulation.

3.1 MODES OF OPERATION

There are five operating modes which MIDAS can be in during execution. These are Command Mode, Real Time Mode, Normal Mode, Single Step Mode, and Debug Mode. Each of these modes allows for utilization of different parts of the MIDAS system. In Command Mode, where the system is placed on invocation or any time the simulation is stopped, commands may be entered through the alphanumeric keyboard. These commands can invoke the utility functions of the system or restart the system in any of the other four modes. On entry into Command Mode, the simulation is stopped after completion of the current clock state, and the display remains static at the point where Command Mode was entered. The commands accepted by the system are described in section 4.2.

In Normal Mode, the simulation and display programs operate in such a way that the simulation is called once every clock state and the display program then executes for a period of time determined by the user via DIAL1. Thus, the speed at which the system operates is at a level where the user can follow the display at his own pace. Normal Mode is entered from Command Mode.

Single Step Mode operates just like Normal Mode, except that Command Mode is returned to automatically at the end of simulation of a single clock state.

Real Time Mode is also entered from Command Mode. In this mode, the simulation proceeds at a rate closer to that of the operation of the simulated system in real time. Certain of the animation effects are disabled. This mode is meant for letting the simulation operate in a "useful" way (such as reading or writing a block of data from disk, or executing any program in simulated memory in an approximation of real time).

Finally, a system debugger mode is available for the system programmer, which allows access to internal simulator variables. This Debug Mode is a state of the system which makes use of a separate display. It allows the user to examine timing waveforms corresponding to all system control lines, in addition to maintaining the same ability to examine and change register contents as are available in the other modes.

The modes described above are available to the user through the command language which may only be used in Command Mode. This necessitates a return to Command Mode before switching to any other mode (this does not apply to Single Step mode which may be entered at any time by hitting FK1). Command Mode can be reached from any mode (except Debug) by hitting FK0.

3.2 THE BASIC DISPLAY

A standard schematic view of the system is the basis of the display. The layout of the schematic is shown in figure 1 below. Note that only the devices and system busses are shown. Registers, flags and lines are omitted for clarity. Contained within the schematic are several elements. For each device, a box appears on the display along with legends identifying the device and the lines eminating from it. In addition, the registers and flags of the device are displayed with an identifying legend and a representation of the data they contain. This representation is in the form of hexadecimal digits for registers, and an on/off indicator for flags. Tying the devices together are three types of data paths: single bit control lines, brightened when high; an 8 bit data bus, the two hexadecimal digits of data travelling across the bus when active; and a 16 bit address bus in which four hex digits of data travel when active.

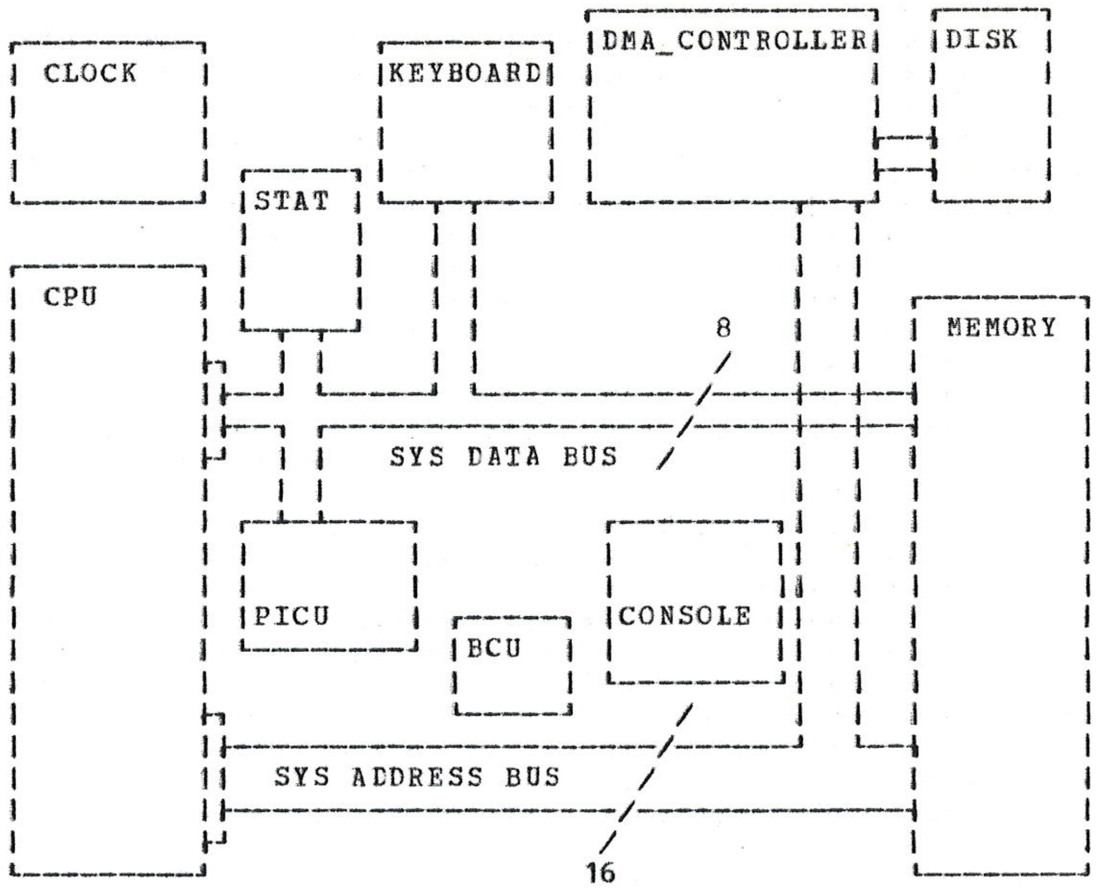

FIG. 1: MIDAS BASIC DISPLAY FRAME

This basic display frame is contained within a viewport. Outside this viewport is an area for status and command prompts generated by the system. The user may choose to "zoom" in on any part of the basic display frame, or "pan" over it. This is accomplished via the joystick which positions a window on a portion of the basic display frame. Zooming is accomplished with DIAL2. As the user moves in, more complex levels of detail appear (such as register data, flags, etc.). This frame is static. Within it, several techniques are employed in animating the operation of the system. Zoom and pan functions can be done in all modes except Debug.

The operation of the system is traced by means of several status displays appearing outside the system schematic viewport. These include a mode indicator, telling which of the modes the system is currently in, and a simulation status display, showing the clock state and machine cycle type that the simulated CPU is currently executing, as well as the mnemonic for the current instruction operation code in the instruction register. These prompts correspond to Intel's description of the operating state of the 8080 CPU, and the user is directed to the _Intel 8080 Microcomputer System User's Guide_ [1], for a full description of these states. In the lower part of the screen, a current file display is shown, telling the user the name of the last memory or status record file he called up. There is also a command prompt/input buffer which appears only in Command Mode. This buffer displays one line of input from the Vector General keyboard and gives the user feedback messages from commands he has typed in. In modes other than Command Mode, a one character buffer is displayed showing the input character for the keyboard interface device of the simulation.

3.3 INPUT TO THE SYSTEM

The system takes input from the various input devices attached to BUGS. Commands are entered from the alphanumeric keyboard. Line editing is done to this input, with the back cursor control (<--) used for character deletion, line feed (LF) used for line deletion, and carriage return (CR) used to complete entry of a line. All alphabetic input is folded to upper case.

The alpha keyboard is used as input to the keyboard interface device in the operating modes. One character at a time may be input to the system through the keyboard, with the corresponding ASCII code appearing in the keyboard interface data buffer in the simulation.

The joystick is used to control X,Y "panning" of the display, while DIAL2 controls "zoom", as described in section 3.2. The joystick position can be locked using FK7. To unlock it, FK7 is hit again. DIAL1 controls the speed of the display (clockwise is slower).

A "standard" view of the basic display frame is available to the user. This view includes the major area of activity in the display (namely the CPU, status decoder and busses). Hitting FK6 sets the window over the standard view area and freezes the window position. It is unfrozen by hitting FK7, as above. In addition, the user may set up his own views of the diagram, and assign them to any of keys 8-15.

The function keys (FKs) are other input devices. Keys 0-7, the top row, are for controlling system functions, while keys 24-30 in the bottom row are for initiating simulated console functions on the simulated system. Key functions are listed below.

KEY	FUNCTION
0	Enter COMMAND MODE

1	Enters SINGLE STEP MODE
2	Freezes display in NORMAL and SS MODES
3	Repeats last simulator state
6	Locks display at standard view
7	Locks joystick X,Y view position
8-15	User defined views
24	Console SINGLE STEP key
25	Console RUN/STOP key
26	Console INTERRUPT key
27	Console LOAD MEMORY from SWITCH REGISTER
28	Console LOAD ADDRESS REGISTER
29	Console INCREMENT ADDRESS REGISTER
30	Console DECREMENT ADDRESS REGISTER

3.4 ANIMATION EFFECTS

In Normal and Single Step Mode the following animation effects are used to illustrate system operation. Control lines, single lines terminated by arrows, are indicated as being low when dim and high when bright. Busses (data paths), wide paths delineated by parallel lines, are 8 bits wide (system data bus and DMA/disk data bus) or 16 bits wide (system address bus). Data travelling over them is indicated by hexadecimal digits moving along the paths. Register contents are displayed in hexadecimal digits. When register contents change, the digits of the appropriate register flash. Single bit flags are indicated as reset by an empty box, or set with a shaded box.

The display is updated dynamically to reflect the simulation in each clock state. Events during each state may not take place with accurate relative timing. However, the simulation is accurate on a clock state basis. See section 5 for more information on the simulation.

One convention observed in the display, is the method used for illustrating bus transfers. All devices that terminate busses have registers that correspond to bus buffers. These registers do not correspond to actual registers found in a particular device, in some cases. They are used for illustrative purposes. In a typical bus transfer, data will move along the bus and be placed in one of these registers. Then at some later time, a control line will be set, indicating gating of the data into some other register. It is at this point when the data is considered to be in the destination device. In addition, when data moves along a bus toward a destination, it moves only to the device to which it is directed, rather than to all devices on the bus. This too is done for illustrative purposes.

4 USING THE SYSTEM

This section describes the basics of using MIDAS. It includes sections on invccation of MIDAS, the command formats, and system responses and error messages.

4.1 INVOKING MIDAS

After making certain that all of BUGS is powered-up and GMS is available, the user invokes the system by typing MIDAS at the terminal. After a brief initialization period, a display, consisting of MIDAS status prompts and the system diagram, should appear on the VG screen. If the message **ERROR IN CALL TO MIDINIT appears at the Teleray terminal, the user should re-IPL BUGS and retry. If the display still does not appear at the Vector General, the appropriate maintenance personnel should be contacted.

After initialization is completed the user should make sure that DIAL2, the zoom control, is turned to its leftmost position. The system is now in Command Mode and the user should see the highest level of detail in the block diagram of the system on the VG screen. The user may now pan over or zoom in on the display as described in section 3.3.

To enter any of the other modes (except Debug) from Command Mode, the GO command is used. If no operands are supplied, Normal Mode is entered and the simulation and animation begin to run. The status display is updated to reflect the current clock state and machine cycle the system is in. The display may now be frozen at any time by hitting FK2. It is unfrozen by hitting FK2 again. The simulation status is always automatically saved for one state. By hitting FK3, this saved copy is recalled, and the last state may be redisplayed. Command Mode is re-entered by hitting FK0. The command buffer display reappears and the simulation is suspended at the completion of the current state.

The GC command can be issued with the <u>S</u>ingle operand to enter Single Step Mode. Hitting FK1 has the same effect as this command. By issuing the GO command with the <u>R</u>ealtime operand and a decimal number, Real Time Mode is entered for the specified number of states. FK0 returns the user to command mode.

4.2 COMMAND LANGUAGE

MIDAS provides the user with several utility functions accessible through the command language. Contents of the one kilobyte simulated memory can be displayed (DISPLAY command) or altered (STORE). The contents of registers, state of flags, and control lines may be altered (SET). Memory pages may be saved on, or loaded from disk (FILE, LOAD). Simulation status records (containing the entire state of the system, except memory) may also be saved on or recalled from disk (SAVE, RECALL). System status can be cleared (RESET) or the entire system, including memory, can be cleared (CLEAR). Debug Mode may be entered (DEBUG). In addition, some GMS commands may be executed from within the system (GMS). Views of the basic display may be saved and associated with function keys (VIEW). Finally, operation of the system can be terminated (END).

In the following sections, the format of the commands is described. Operands appear in angle brackets. Abbreviations for commands are underlined and optional operands are set in square brackets.

4.2.1 CLEAR

```
CLEAR
```

The system is cleared. Registers, flags and lines of all devices are reset to startup values. Memory is set to all zeroes.

4.2.2 DISPLAY

```
DISPLAY <hex addr>
```

The locations of the current memory page are displayed beginning with the specified address. The address must be an even hexadecimal value in the range X'000' to X'400'.

The contents of the specified location and the following 11 bytes are displayed in six groups of two bytes.

4.2.3 END

```
END
```

This terminates MIDAS and returns the user to GMS. GMS is re-IPLed.

4.2.4 FILE

```
FILE  [<filename>]
```

The current memory page is written to disk. If no filename is given, it is written to the default file named '..TEMP.. MEMM'. Otherwise it is put into a file with the specified filename and filetype MEMM. If a file with that name already exists, it is replaced. Otherwise, one is created.

4.2.5 GMS

```
GMS <gms command> [<command args>]
```

The specified command is passed to GMS. Up to four arguments may be specified with the command. The user is cautioned that any command that may be loaded will be executed (including those wihich may be disasterous for MIDAS). If no GMS command is specified, the command is ignored. Bad GMS commands result in an error message displayed in the prompt area. Commands which give typed output, have that output displayed at the Teleray terminal. For information on GMS commands, see the GMS User's Guide [2].

4.2.6 GO

```
┌──────────────────────────────────────────────────────────┐
|                                                          |
|         ┌                           ┐                    |
|         |REALTIME <decimal value>   |                    |
|    GO   |                           |                    |
|         |SINGLESTEP                 |                    |
|         └                           ┘                    |
└──────────────────────────────────────────────────────────┘
```

The simulation is started in the specified mode (see section 3.1). If no operand is given, normal mode is entered. If real time is specified, a decimal number between 1 and 32,767 must be specified that indicates the number of clock states to execute.

4.2.7 LOAD

```
┌──────────────────────────────────────────────────────────┐
|                                                          |
|    LOAD  [<filename>]                                    |
|                                                          |
└──────────────────────────────────────────────────────────┘
```

A memory page (1024 bytes) is read into simulation memory from disk. If no filename is given, the default file, '..TEMP.. MEMM' is loaded if it exists. Otherwise the file with the specified filename, and filetype MEMM is loaded.

4.2.8 RECALL

```
┌──────────────────────────────────────────────────────────┐
|                                                          |
|    RECALL [<filename>]                                   |
|                                                          |
└──────────────────────────────────────────────────────────┘
```

The current status record is replaced with the contents of the specified MIDA file on disk. If no filename is given, the default file '..TEMP.. MIDA' is assumed. The display is updated to reflect the new system status.

4.2.9 RESET

```
RESET [START [<go operands>]]
```

The system status is reset as with the CLEAR command (see 4.2.1), but memory is not cleared. If the start operand is given, the system is then restarted in the specified mode. Sub-operands may be specified as in the GO command (see section 4.2.6).

4.2.10 SAVE

```
SAVE [<filename>]
```

The current state of the system is saved on disk. If no filename is specified, the default file '..TEMP.. MIDA' will be used. Otherwise it is stored in the file with the specified filename and filetype MIDA. If one already exists, it will be replaced. Otherwise one will be created.

4.2.11 SET

```
                   <hex value>

SET <idname>       SET

                   RESET
```

The register, line, or flag with the specified idname is altered. For flags and lines, SET and RESET are used. For registers, a hex value padded on the left with zeroes to the appropriate length is used to replace the old contents. The display is updated to reflect the new value or status. See Appendix A for a list of valid idnames.

4.2.12 STORE

```
STORE <hex addr> <hex value> [<hex value>...]
```

The locations of the current memory page are altered and displayed, beginning with the specified address. The address must be an even hexadecimal value in the range X'000' to X'400'. The contents of the specified location are displayed in six groups of two bytes, as in DISPLAY (see 4.2.2). Any number of groups of up to two bytes of hexadecimal data may be specified. Single hex digits are padded on the left with a zero to form a byte value.

4.2.13 VIEW

```
VIEW <n> [OFF]
```

The current auto-view key setting is changed. If OFF is not specified, the current window position is saved and associated with key n+8 (valid values of n are from 0 to 7, corresponding to fkeys 8 to 15). The corresponding key lights, and when pressed freezes the window over the saved view position. If OFF is specified, the specified key is disabled, its light is extinguished, and the previous view is no longer associated with that key. It is recommended that the freeze key (key 7) be used to hold the desired window position during execution of the view command.

4.2.14 DEBUG

```
┌──────────────────────────────────────────────┐
│                                              │
│    DEBUG                                     │
│                                              │
└──────────────────────────────────────────────┘
```

Debug Mode is entered. The basic display frame is replaced by the debugger display. FK31 returns the user to Command Mode, regenerating the basic display schematic. It is not recommended that this command be issued by general users.

4.3 MESSAGES

The following messages may arise during execution of commands. They appear in the command buffer of the display.

ENTER COMMAND:
Normal command mode prompt

INVALID COMMAND OPERAND; RE-ENTER:
An invalid operand was entered with a command. It was not executed. See the appropriate section on command formats in this manual.

INVALID COMMAND; RE-ENTER:
An invalid command was entered at the keyboard.

INVALID ADDRESS OR DATA; INCOMPLETE:
An address or data for STORE command was invalid. Any valid data preceeding the bad character was stored correctly. Verify with the DISPLAY command.

FILE NOT FOUND:
The file specified in the CURRENT FILE display was not found on disk for RECALL or LOAD commands. If no filename appears, '..TEMP..' did not exist. Use GMS LISTF to list available MEMM or MIDA files on disk.

ERROR IN DISK WRITE:
An error occurred while writing the file specified in the CURRENT FILE display. This probably means there is no more contiguous disk space available. From GMS, ERASE unwanted files, re-IPL, and use P_A_C_K. Otherwise, if space is available, contact the appropriate maintenance personnel.

BAD COMMAND PASSED TO GMS:
The command specified in the GMS command was invalid, or caused an error.

The following messages are serious errors. They should not occur during normal operation. All of these messages occur at the Teleray terminal and should be reported as soon as possible.

**ERROR IN CALL TO MIDINIT
Issued at initialization time (start or clear) if the basic display frame cannot be read in. Try re-IPLing. If this does not work contact appropriate maintenance personnel.

**DISPLAY ERROR
Issued for an error occuring during display operations. Usually non-fatal.

**SEGLOAD FAILED FOR XXX, CODE YYY
Routine XXX of MIDAS could not be loaded by GMS during execution. Re-IPL, and contact appropriate maintenance personnel.

W: PROGRAM INTERRUPT-(XXXX) YYY-AT ZZZ
Program interrupt. RE-IPL and contact appropriate maintenance personnel.

5 THE SIMULATED SYSTEM

Following are brief descriptions of each of the devices in the system. Each includes an operational description and a list of what is displayed for that device.

5.1 THE CPU

As mentioned above, the CPU of the system is a simulation of the Intel 8080 Microprocessor [1]. This is a processor that employs a word length of 8 bits and the capability for 16 bit addressing. A stack pointer is maintained for program subroutines. The CPU has a file of 6 user accessable registers that may be used in pairs for addressing operations. In addition, there is a Program Counter, Accumulator, and a Temporary Register that is not directly accessable to the user, but which is used to hold intermediate results in some operations. There is also a complement of five comparison indicator flags (for sign, arithmetic carry, parity, zero and auxiliary carry). The processor executes 78 instructions, all of which are simulated. All the above registers and flags are displayed, including a data register and address register (these terminate the address and data busses). The 10 control lines are also shown, as are the data and address busses.

5.2 CLOCK GENERATOR

The clock generator provides two clock pulses needed by the CPU. These are displayed with the two clock input lines to the CPU.

5.3 STATUS DECODER

The Status Decoder receives the "status byte" output by the CPU during T1 of its machine cycle. This byte determines what type of cycle the CPU is performing (e.g., memory read, I/O write, etc.), and the decoder uses it and the CPU DBIN and WR control lines to derive control signals which interface the CPU to other devices, such as memory. A status register displays the status byte sent out by the CPU.

5.4 MEMORY

MIDAS memory consists of a 1024 (1K) byte memory area maintained by the simulation. The memory is random access and assumed to have a cycle time faster than the CPU or any peripheral so that a "memory ready" control line is not necessary. An address register and data register are displayed, along with read and write control lines.

5.5 BUS CONTROL UNIT

The BCU controls direct memory access (DMA) operations by devices external to the CPU. These currently include the console and the disk DMA interface. Devices signal requests to the BCU which then determines, through a priority network, which one gets service. The HOLD and HLDA (Hold Acknowledge) lines of the CPU are used to stop it during the DMA transfer, and an "enable" signal is sent to the device to be serviced.

5.6 PRIORITY INTERRUPT CONTROL UNIT

This device takes as input an interrupt request line from each device that can interrupt the CPU. When one or more of the requests is active, the PICU raises the interrupt line to the CPU. When the CPU acknowleges the interrupt, the PICU provides it with the RESTART instruction which is part of the 8080 interrupt cycle. A mask register allows device requests to be masked off (held pending), and a vector register provides the number of the highest priority interrupting device. By setting a control register, the PICU can be programmed to handle the interrupt sequence in different ways.

5.7 THE CONSOLE

A console is provided that is able to do DMA. Simulated on the console are data and address registers, a switch register, and an Interrupt Flag (set/reset by the user via a function key). The console enables loading or displaying of any memory location, or any of the console registers. Console functions are activated through function keys (see section 3.3).

5.8 DISK CONTROLLER AND DMA CONTROLLER

These two devices act as control for a floppy disk peripheral. The simulated disk has 32 tracks, each with eight 128 byte sectors, and resides in a GMS file on the real BUGS disk. The disk controller accepts commands to seek and format tracks, and to read, write, and verify sectors. At the end of a disk operation it sends a status byte to the DMA controller indicating the success or failure of the operation. Latency and access times are simulated by fixed delays of several machine cycles. A sector/track address register is maintained by the controller, as well as data and command registers.

Interfacing the Disk to the rest of the system is a DMA controller. This device has data and address registers, along with read and write lines to memory, so that it can perform data transfers between the memory and itself. It communicates with the disk over an 8-bit bidirectional data bus and eight control lines. The DMA controller also has a special command/status register and a count register to keep track of block data transfers. All of the controller's registers are accessable to the 8080 as I/O ports.

Typically, an 8080 program will set up the count and address registers with initial values and issue a command byte to start a disk operation. The DMA controller then handles requests by the disk to fetch or store data. Upon completion, it saves the status byte sent by the disk and raises the interrupt request line to the interrupt control unit.

5.9 KEYBOARD INTERFACE

The Keyboard Interface is a device which is used to illustrate interrupt-driven I/O. Each time a key is struck when MIDAS is in Normal or Real Time mode, the character is sent to the keyboard interface and an interrupt request is signalled to the interrupt control unit. An I/O interrupt routine can then read the character via an 8080 input instruction. The interface has a character register where incoming data is displayed.

6 APPENDIX A: SYSTEM IDNAMES

This section contains a list of idnames to be used in the SET command utility. The names are arranged by device. Usually the names correspond to the legends in the basic display. However, in some cases they don't. Certain lines are overlaid, such as the I/O lines from the Status Decoder. In these cases, the line destination is specified (i.e., IORDMA and IORCON refer to different sections of the same displayed line). This is done to simplify the picture, rather than show each of the lines seperately. For each name, the type is also given. Reg refers to an 8-bit register unless it is an address register, in which case it is 16-bits. Reg pair refers to two 8-bit registers concatenated together. Flag and line refer to one bit flags and lines. These require the SET and RESET operands.

NAME	REFERENCE
CPU	
CPUBC	B,C reg pair
CPUB	B reg
CPUC	C reg
CPUDE	D,E reg pair
CPUD	D reg
CPUE	E reg
CPUHL	H,L reg pair
CPUH	H reg
CPUL	L reg
CPUWZ	W,Z reg pair
CPUW	W reg
CPUZ	Z reg
CPUPC	Program Counter reg pair
CPUPCH	PC H reg
CPUPCL	PC L reg
CPUSP	Stack Pointer reg pair
CPUSPH	SP H reg
CPUSPL	SP L reg
CPUAR	Address reg
CPUDB	Data bus buffer reg
CPUIR	Instruction reg
CPUA	Accumulator reg
CPUTMP	Temporary reg
CPUACT	ALU latch reg
CPUALU	ALU output reg
CPUSGN	Sign flag
CPUCAR	Carry flag
CPUZER	Zero flag
CPUPAR	Parity flag
CPUAXC	Auxiliary carry flag
INTE	Interrupt enable flag
HLDA	Hold acknowledge line

SYNC	Sync line
DBIN	Data bus input line
WR	Write line
WAIT	Wait flag

<u>Status Decoder</u>

STRSR	Status byte reg
STRMR	Memory Read line
STRMW	Memory write line
IORKEY	I/O Read to Keyboard Interface line
IORDMA	I/O Read to DMA Controller line
IORCON	I/O Read to Console line
IORICU	I/O Read to Interrupt Control Unit line
IOWICU	I/O Write to Interrupt Control Unit line
IOWDMA	I/O Write to DMA Controller line
IOWCON	I/O Write to Console line
STRINA	Interrupt Acknowledge line

<u>Memory</u>

MEMAR	Memory Address reg
MEMDB	Memory Data Bus reg

<u>Priority Interrupt Control Unit</u>

ICUDB	Data Bus reg
ICUMSK	Mask reg
ICUVCT	Vector reg
ICUCTL	Control reg
INT	Interrupt line

<u>Bus Control Unit</u>

BECPU	Bus Enable CPU line
BECON	Bus Enable Console line
BEDMA	Bus Enable DMA Controller line
HOLD	Hold line

<u>Console</u>

CONAR	Console Address reg
CONSW	Switch reg
CONDB	Data Bus reg
CONOUT	Output reg
CONMR	Memory Read line
CONMW	Memory Write line
CONHLD	Hold Request line
CONINT	Interrupt Request line

<u>DMA Controller</u>

DMAAR	Address reg
DMADB	Data Bus reg
DMACT	Count reg
DMAMR	Memory Read line

DMAMW	Memory write line
DMAHLD	Hold Request line
DMAINT	Interrupt Request line
DMASR	Status Byte reg
DMADR	Data To-From Device reg
DMACS	Command Sent line
DMASD	Stop Data line
DMAIBF	Input Buffer Full line
DMAODS	Output Data Strobe line

<u>Floppy Disk Controller</u>

DSKTRK	Track-Sector reg
DSKCMD	Command Byte reg
DSKDR	Data To-From DMA Controller reg
DSKCR	Command Received line
DSKSTS	Status Strobe line
DSKIDS	Input Data Strobe line
DSKOBE	Output Buffer Empty line

<u>Keyboard Interface</u>

KEYDB	Data Bus Buffer reg
KEYINT	Interrupt line

7 APPENDIX E: INTEL 8080 INSTRUCTION SET

8 REFERENCES

[1] Intel Corp., Intel 8080 Microcomputer System User's Guide, Santa Clara, Ca.: Intel Corp., 1975.

[2] Burns, R.W., Graphics Monitor System User's Guide, Brown University Graphics System, 1975.

[3] Brown University Software Engineering Group, The Brown University Graphics System Overview, Computer Science Program, Brown University, 1976.