9. MAXC2 MIDAS OPERATION

Midas is the loader/debugger used for the Maxc2 microprocessor. It has features for directly testing the Maxc2 hardware through the maintenance interface, for loading/dumping microprograms assembled by Micro, and for examining and modifying the storage in the microprocessor. It can control the microprocessor and memory power supplies, configure the memory, and enable/disable the various memory error correction-detection stuff. It also has a command for calling AltIO, the I/O program used by Tenex.

9.1. Starting Midas

To start Midas, simply say "Midas" to the executive or, more generally, "Midas com-file".

Midas command files have the extension ".Midas". Generally, there is one command file for each of the hardware diagnostics, with the same name as the diagnostic, e.g.:

dgbasic.mb	the diagnostic;
dgbasic.midas	the command file;
midas dgbasic	to the Executive executes the command file.

These command files load the diagnostic into the microprocessor and display various registers which are of interest when the microprogram is in use.

In addition, there are the following command files:

midas/i	initializes (should be executed whenever any Midas files move or change, or when a new Alto Operating System is installed).
midas/r	continues with the symbol table and display saved at the last call to AltIO.
midas	simply fires up Midas.
midas tenload	loads the Tenex microcode.
midas tengo	loads Tenex microcode and calls AltIO to start Tenex.
midas mexecgo	loads Tenex microcode and calls AltIO to start MicroExec.
midas init	loads new Tenex microcode and sets checksums.
midas debug	loads some special symbols for use with "Constructed-Test" (see below).

If you are already running Midas, and you want to switch microprograms, you can bug the "Run-program" menu item and then bug the name of the command file in the subsidiary menu which comes up. The "loader" menu item simply initializes a new Midas. "debug" is used to prepare for the "Constructed-test" stuff discussed later.

9.2. Midas Display

At the top of the Midas display are a number of register name-value menus. Below these are the name of the last microprogram loaded, the command comment line, the command menu, and the input text line. When you move the mouse around, the menu item selected (if any) turns black. Note that mouse actions execute when you *release* the mouse button, so you can move the mouse around with the button depressed without causing damage. If the mouse position does not select any screen item, nothing happens when the button is released.

Register areas are of different sizes. Smaller areas are already filled in with various microprocessor registers when you fire-up Midas. The unused ones at the right and bottom left of the display are appropriate for 36-bit stuff, but only the bottom right items are large enough for IM (instruction memory) stuff.

To display a new item, type its name, move the mouse to one of the register name areas, and push-and-release the left (or top) mouse button.

If the command line is empty, the selected register area will be cleared when the button is released.

When you push the right (bottom) mouse button over a name area in which an address is displayed, a subsidiary menu appears as follows:

A+1 A-1

"A+1" increments the address, displaying the next location. "A-1" decrements the address.

Releasing the middle button over an address item shows alternate forms of printout on the command comment line. If the input text line is non-empty, it will first display that item.

Releasing the left button over a value item, evaluates the input text and stores the value (or 0 if no text typed) in the selected register. The input text is limited to octal numbers with interspersed spaces permitted for readability.

Releasing the middle button over a value item shows an alternate display of the value on the command comment line. The alternate for IMA, NPC, and STK values is the nearest IM address tag less-equal to the value+offset (the value is also put on the input text line, so you can examine that location in the display area if you want to). For DM, DM1, and DM2 words, the alternate is three IM addresses with offsets and one flag. For IM the alternate is a symbolic printout of the microinstruction.

Releasing the right button over a value item appends the text of the value to the input text line.

9.3. Midas Command Menu

For the command menu, all mouse buttons are presently equivalent (this may change). For most common commands, equivalent input text sequences carry out the same action, as given below.

The general philosophy on mixing keyboard and mouse button control is that, when possible, a command involving some typing is carried out completely at the keyboard, whereas commands involving mouse buttons are carried out completely with the mouse.

Many of the commands are executed in overlays. When these get executed, the register display will turn black (the code for overlays resides where the display bit buffers would otherwise be). During loading and during execution of command files, the display is turned off to make the machine run faster.

Many of the commands put up a succession of subsidiary menus with assorted options as discussed below.

The long-running commands normally display an "Abort" menu item. When this is bugged or when control-C is typed, the action terminates.

Input	**Keyboard**	**Menu Item**	**Comments**
	;Q	Exit	Quit to Executive.
File		Read-Cmds	Execute command file (default extension ".Midas").
		Show-Cmds	Show equivalent command file text for selected menu items.
File		Write-Cmds	Write subsequent commands on file.
	;A	AltIO	Call AltIO with options (terminates command file).
		Run-Program	Run selected microprogram (restricted use in command files).
Files	;L	Load	Load .MB files.
Files		LoadSyms	Load only addresses from .MB files.
File	;D	Dump[2]	Dump compacted .MB file using the .MB file(s) of the previous load to control what’s dumped.
File	;C	Compare[2]	Compare microprocessor data to data specified in .MB file--compare file must have been created by Dump.
Addr	=		Print value of an address (illegal in command file).
IMaddr	;B	Break	Insert breakpoint. (Restarting from the breakpoint will succeed in all situations except when a memory write or read-modify-write has been given and MDR not yet loaded with the correct data or when the disk controller is active.)
[IMaddr]	;K	Kill-Break	Remove break at address (at IMA if nothing typed).
[IMaddr]	;G	Go[1]	Start at address (continue at IMA/NPC if nothing typed).

Input	Keyboard	Menu Item	Comments
[IMaddr]	;P	Go[1]	Same as ;G (mnemonic "Proceed", provided for compatibility with Maxc1 Midas).
[IMaddr]	:	Step	Single-step at address (at IMA/NPC if nothing typed). See caveats on breakpoints above.
[IMaddr]	;R	Repeat-Go[1]	Go at address, repeat endlessly after halts.
[IMaddr]	;S	Repeat-Step[1]	Repeatedly step at address.
		LMPEscan	Scan all local memories (IM, SM, DM, DM1, DM2) for parity errors.
		Power-On[1]	Turn on power (see below).
		Power-Off[1]	Turn off power (see below).
LDRaddrs		Constructed-Test[1]	(see below).
		Test[1]	Test register or memory (see below).
		Field-Loop[1]	For scoping (see below).
		Test-All[1]	Test everything (see below).

[1] Requires preceding "TimeOut" command in command file.

[2] Requires confirmation with <cr>, "Y", or "." (or by preceding "Confirm" command in command file).

9.4. Keyboard

There are a number of characters which are legal symbol constituents in microprograms but which will cause trouble for Midas when they appear in addresses.

Lower case typein is converted to upper case by Midas, so avoid lower case characters in microprogram identifiers. I recommend writing microprograms with the shift-lock key depressed. Avoid "=" in identifiers. "+" and "-" are ok so long as the following character (if any) is a letter.

Typing ahead is legal until the character you type would cause execution of a command. After that Midas will flush input and blink at you until the current command finishes.

At the end of a command the input text typed for that command is displayed on the input text line. This text remains valid and can be used as the argument for another mouse action. However, if you type any character (except control-A or backspace), the old input will be flushed before inserting the new character.

Keyboard editting characters are as follows:

control-A	delete last character
backspace	delete last character
control-Q	clear text line
del	clear text line

Other special keyboard characters are as follows:

control-C	aborts the current action
control-Z	aborts a command file
control-D	turns on the display
control-O	turns off the display

The interrupt characters above are ineffective during loading, dumping, or comparing, which typically take between 2 and 15 seconds. Indefinite duration commands, such as "Go", "Test", etc. always monitor the keyboard, so control-C can be used to terminate them.

Control-Z, control-D, and control-O are intended for use during command files. However, these characters do not take effect until the command file executes a command such as "Go" which monitors the keyboard. There is no way to abort a command file and give control back to Midas safely except during a "Go" or other long-running command. This is not expected to be a problem because commands are executed quickly.

After interrupting a "Go" with control-C or control-Z, proceeding with ";P" or ";G" will succeed except when the conditions discussed earlier for breakpoints apply.

Although command menu items "Step", "Go", "Break", "Kill-Break", "Repeat-Step", and "Repeat-Go" are provided, you will normally find it more convenient to execute these from the keyboard using the alternate command characters.

9.5. Command Files

Command files have default extension ".Midas". They are normally executed by typing "Midas com-file" to the executive or by bugging the name of a command file from the subsidiary menu put up by "Run-program". However, you can also execute a command file by typing a file name and bugging "Read-cmds" in the command menu.

To find out what text should be put in command files, you can bug the "Show-Cmds" item in the command menu. This will cause the command file text for each command to be displayed on the command comment line as the mouse selects it (you don’t have to execute the command to see the equivalent text). This text is complete except that the mouse button which executes the command isn’t shown unless you depress the mouse button. Usually the text "L " precedes the text given by "Show-Cmds", indicating that the left (or top) button was pressed and released to carry out the command. "M " precedes a middle-button command and "R " a right-button (or bottom-button) command. The name of the command is followed by a blank, then the command line text, and finally a carriage return. To terminate "Show-Cmds", bug "Conceal-Cmds" (this only appears in the command menu when "Show-Cmds" is in progress).

You can prepare a command file (default extension ".Midas") by typing a file name and bugging "Write-cmds". This causes text for subsequent commands to be put on the file. When you are done with this, bug "Stop-Write-Cmds" to close the file. ("Stop-Write-Cmds" only appears in the command menu when a command file is being written.) You will probably want to edit out your goofs with Bravo after the command file is written.

For command files associated with big programs, the symbol table is not entirely in core, so you may want to sort the command-file addresses in reverse-alphabetical order to minimize symbol-table swapping. Currently, Midas has barely enough symbol table space to hold all of the Tenex microcode's addresses, so none of the symbol blocks have to be swapped. However, this may change if more features are added to Midas or to the Tenex microcode.

Command files can be nested several levels (limited by the size of sysZone which must be big enough to accommodate OpenFile and buffers for the command files already open). However, there are the following *restrictions*:

(1) "AltIO" terminates command files (i.e., upon return to Midas from AltIO the command file will not be continued).

(2) "Run-Program" is illegal except in the top level command file. ("Run-Program" resets Midas, then calls "Read-Cmds". This reset operation smashes the symbol table, the display, and the stack back to their initial state. Hence, if you were to execute "Run-Program" from a subsidiary command file, that command file would be continued, but the higher level ones would not, and the sysZone buffers for the higher level command files would not be released.)

Extra <cr>s can be used in a command file for punctuation, and ";" can be used at the beginning of a line or after the last character of a command to begin a comment. The comment is terminated by <cr>.

Since Midas builds a table of file pointers during its initialization, when you edit an existing command file or .MB file, you should reinitialize Midas by typing "Midas/I". When you add new command files or .MB files you should update the "Midas.Programs" file appropriately and do "Midas/I". The form of "Midas.Programs" is discussed later.

There are a number of commands which can never occur when Midas is run interactively, but which are useful in command files. These are as follows:

Text Arg	**Menu Item**	**Comments**
Octal no.	SkipVEql	Skip the next command if the selected value menu item equals the value of the text arg.
Octal no.	SkipVGr	Skip the next command if the selected menu item's value is greater than the text arg.
Octal no.	SkipVLs	Skip the next command if the selected menu item's value is less than the text arg.
Octal no.	Skip	Skip N following commands, where N is the 16-bit value of the text arg.

Octal no.	BackSkip	Reset to byte 1 of the command file, then skip.
Octal no.	Return	Return out of current command file and skip N commands in the calling command file (if any), where N is the 16-bit value of the text arg.
	DisplayOn	Turn on the display, so that effects of subsequent commands can be observed. The display is initially off for a command file.
	DisplayOff	Turn off the display.
Octal no.	TimeOut	Argument is a 32-bit octal number of milliseconds after which to abort the immediately following command, if it has not finished by then. This is intended for use before "Go", which might hang indefinitely. If the timeout occurs Midas will skip the command after the "Go". TimeOut also turns on the display, which is necessary because the machinery which checks for timeouts is only active with the display on.
	Confirm	Supply confirmation for the command which follows (which should be one of the commands requiring confirmation).
File name	OpenOutput	Open an output file (default extension ".Report") on which text can be written.
	CloseOutput	Close the output file.
[text]	WriteMessage	Write the contents of the input text buffer on the output file. Note that if any text follows the WriteMessage, that text up to but not including the <cr> is what gets written. However, if <cr> immediately follows WriteMessage, then the contents of the input text buffer left by the previous command are what gets written. "~" is translated into <cr>.
text	Show-Error	Display the text arg on the command line, turn on the display if it was off, and query with "Abort" and "Continue" menu items.

The "right-button" value action allows the current values of any register to be appended to the input text buffer. This can be used in conjunction with WriteMessage to output various parts of the machine state to the output file.

9.6. Loading Programs

The "Run-program" command is a short method of executing the most common command files--those which load a microprogram and fix up the display for that program.

You can also load a program by typing a file name (default extension ".mb") and bugging "Load" in the command menu. However, this should rarely be necessary because all of the microprograms appear in the menu put up by "Run-program", which additionally reinitializes Midas.

It is a poor idea to do a "Load" when another microprogram has already been loaded because this will merge symbols from the new load into those already in the symbol table. In addition to being very slow, this may overrun the size of the symbol table on the disk (no provision is made for expanding the symbol table beyond its maximum size).

It is also a good idea to assemble microprograms as a single .MB file. Although Midas can load multiple .MB files (typed as a list separated by commas), this will cause additional fragmentation of the symbol table and thrashing while blocks are swapped in and out.

These recommendations follow because Midas takes advantage of alphabetical address ordering in .MB files to pack its symbol buffers nearly full. But when subsequent files are loaded the symbol buffers will be fragmented to about half-full, symbol buffer swapping will result, and the symbol searches will be longer.

9.7. Dumping Microprograms

After assembling a microprogram, it is a good idea to load and dump it from Midas. Dumping deletes all forward reference fixups left by Micro and compacts both data and addresses to use less disk space and load more quickly later. Also, undumped .MB files cannot be used with "Compare" because of forward references.

Note that only those memory words loaded by the original microprogram are dumped, so you cannot patch unused locations, dump the program, and expect the patches to survive.

9.8. Tenex Microcode

When a new release of the Tenex microcode is made, it must be loaded into the microprocessor, the checksums set, and then dumped onto SYS2. To carry out this initialization, type "Midas init" to the executive.

After dumping, SYS2;C can be used to compare the microstore against its correct contents whenever the checker program (INIT;G) fails. This is a good way of locating bad storage chips after a crash or verifying that the storage is OK before loading some other microprogram.

9.9. Power On-Off

The "Power-On" menu item allows the microprocessor, port, and memory to be powered on and tested in various ways. The first "Power-On" subsidiary menu shows the following:

Processor	Turn on processor and port power only.
Memory	Turn on memory and port power only and configure memory.
Both	Turn on port, processor, and memory cabinets’ power.
Test-Port-&-Processor	Repeatedly execute the maintenance interface operations for turning on port and processor power (for debugging power control hardware).
Test-Memory-Cabinet	Query for a memory cabinet number, then repeatedly execute the I/O instructions for powering on the memory cabinet.

After the processor is turned on, the local memories (IM, SM, DM, DM1, DM2) are zeroed to prevent parity errors from occurring.

When "Memory" or "Both" is bugged, another subsidiary menu will come up showing the various options for enabling/disabling error correction, core-zeroing, and FER on memory errors. For normal use by Tenex, let the default action take place for all of these options, and immediately bug the "do-it" menu item.

Power on takes about 15 seconds because the power supplies take a while to stabilize. After power-on the interrupt system is cleared and the flag register is zeroed.

The right-most four nine-bit bytes in CONFIG (appears on the Midas display) control the memory configuration as follows:

bits 0-8	represent quadrant J
bits 9-17	represent quadrant K
bits 18-26	represent quadrant L
bits 27-35	represent quadrant M

A "1" in the byte enables cabinet 0, "2" cabinet 1, "4" cabinet 2, ..., "200" cabinet 7. Hence, CONFIG might be set as follows:

3003003003	All four quadrants active in cabinets 0 and 1 (would result in 256K four-quadrant configuration)
17017017017	All four quadrants active in cabinets 0-3 (would result in 512K four-quadrant configuration)
3000003003	Quadrant K disabled, but cabinets 0 and 1 active (would result in 128K two-quadrant configuration avoiding quadrant K)

Memory power-on does a catastrophic memory reset. This may occasionally clobber memory words, but this complete reset has been required to unhang the memory on several occasions, due to problems which have not been fixed.

The first "Power-Off" subsidiary menu is analogous to the "Power-On" menu. Note that the port power will remain on unless you bug "both" because neither the memory nor the processor is operable without the port power being on.

Please be mindful of the following *terrible danger:* **The microprocessor, port, and memory power must be turned off before powering down the Alto.** When this rule was violated on two occasions, numerous Alto MI chips were clobbered, apparently due to power transients from the cable.

To make repairs to the microprocessor, it is sufficient to turn only the processor power off, provided that neither the PMEMI, KMEMI, nor PMAINT cards are disturbed. For all other maintenance, power down "Both". If you want to remove the PMAINT card, power off the Maxc processor and memory first, then the Alto disk, and finally, the Alto processor.

9.10. Testing Through the Maintenance Interface

"Test", "Constructed-Test", and "Test-All" allow the microprocessor to be tested from the Alto through the maintenance interface. Data patterns for the test are determined from the first subsidiary menu, as follows:

ZEROES	All-zeroes data
ONES	All-ones data
CYC1	36-bit word of all zeroes with a single-one bit cycled left one position each iteration
CYC0	36-bit cycled zero in word of ones
RANDOM	48-bit random numbers
SEQUENTIAL	0, 1, ..., sequential numbers
ALTZO	Alternating all-ones and all-zeroes patterns
ALT-SHOULD-BE	Alternating contents of SHOULD-BE with its ones-complement

Testing is controlled/described by five addresses on the display as follows:

LOW-HIGH	(For memory tests only) contains in bits 0-17 the lowest memory address tested and in bits 18-35 the highest address tested. Memory tests begin at the low address and seq-uentially test each address up to the high address, then loop.
LOOP-COUNT	The number of successful iterations of the test prior to failure or prior to aborting from the keyboard or with the mouse.
SHOULD-BE	What the data should have been.
DATA-WAS	What the data actually was.
BITS-CHECKED	Mask of bits checked (see below).

LOW-HIGH delimits the maximum address range. If the memory you select has fewer words than HIGH, then the maximum for that memory is used. This means that the last address tested for a memory test is as follows (assuming LOW-HIGH is left at its normal maximum value):

IM	LOOP-COUNT mod 10000 (octal)
SM	LOOP-COUNT mod 1000

DM	LOOP-COUNT mod 1000
DM1	LOOP-COUNT mod 1000
DM2	LOOP-COUNT mod 1000
RM	LOOP-COUNT mod 40
LM	LOOP-COUNT mod 40
STK	LOOP-COUNT mod 14

"Test-All" automatically loads BITS-CHECKED with a full-sized comparison mask prior to testing each item. It tests each register 250 times and makes 4 passes through each memory. It is a good idea to run "Test-All" whenever the hardware is in a suspicious state.

"Test", after showing the data-pattern menu, shows a menu of register and memory names, and executes a test of the one you select until the test fails or you halt the test from the keyboard. "Test" masks the current contents of BITS-CHECKED with the maximum-sized mask for the register or memory being tested. If you previously tested a small register, then you have to manually load BITS-CHECKED with a full-sized mask before testing a big register. If you don't want to check all the bits in a register, then clear the bits you don't want to check in BITS-CHECKED.

"Constructed-test" should only be used when the "debug" command file has been loaded. This requires a sophisticated understanding of the hardware and of the innards of Midas and is not recommended for novices. Midas makes use of a number of microinstructions while reading, writing, starting, and stopping the microprocessor. "Constructed-test" allows these microinstructions to be executed in non-standard sequences to beat on particular hardware problems through the maintenance interface. When using "Constructed-Test", a list of LDR addresses is typed (see DEBUG.MC for the instructions) separated by commas. If only one LDR address is typed, the maintenance interface's BR register is loaded once with the selected data pattern, then the LDR instruction is repeatedly executed for a scope loop. When two, three, etc., up to six LDR addresses are typed, a test loop occurs whereby BR is loaded with the next data pattern, the list of instructions is executed, and then BR is read back and compared against the data under control of BITS-CHECKED. The loop stops when (data-read-back xor data-sent-out) & BITS-CHECKED is non-zero.

"Field-Loop" exercises signal decoding for particular fields of the microinstruction for scope loops. A microinstruction is fabricated from a no-op microinstruction in which the field selected from the first subsidiary menu is replaced by various values. The second subsidiary menu allows the value in the selected field to be incremented, decremented, and shifted.