

VERSION INFO: \$Id: api.txt,v 1.1 2000/10/11 22:33:07 cheiny Exp \$

Class: DEBUGGER

(debugger 'clock)  
Clock the mmp associated with this debugger.

(debugger 'processor)  
Return the mmp associated with this debugger.

(debugger 'mpe-debugger n)  
Return the nth mpe debugger associated with this debugger.

Class: MPE-DEBUGGER

(mpe-debugger 'debugger)  
Return the mmp debugger associated with this mpe debugger.

(mpe-debugger 'unit-number)  
Return the unit number of the mpe associated with this debugger.

(mpe-debugger 'processor)  
Return the mpe associated with this debugger.

(mpe-debugger 'find-symbol pname)  
Find the symbol with the specified name. The symbol name must be a string.  
Symbols names are case sensitive.

(mpe-debugger 'find-symbol-or-register pname)  
Find the symbol or register with the specified name. The symbol name must be  
a string. Symbol names are case sensitive.

(mpe-debugger 'runtime-eval expr)  
Evaluate an expression in the context of the debugger. This includes bindings for  
the symbols \*default-mpe\* and &p which are bound to the mpe associated with the  
debugger and \*default-mpe-debugger\* and &d which are bound to the debugger itself.  
Also, the tilde method of accessing nuon symbols is available within the expression  
just like in before and after methods.

(mpe-debugger 'disassemble start count &optional port)  
Disassemble instructions starting at the specified address. Instructions are  
disassembled and printed to the specified port until count instructions have been  
printed.

(disassemble start count &optional port)  
Disassemble instructions starting at the specified address.

(mpe-debugger 'clock-mmp)  
Clock the mmp associated with this debugger.

(mpe-debugger 'run)  
Start the mpe running.

(mpe-debugger 'running?)  
Returns #t if the mpe is running and #f otherwise.

(mpe-debugger 'step)  
Single step the mpe.

(mpe-debugger 'step-over)  
Single step the mpe stepping over subroutines.

(mpe-debugger 'wait-for-halt)

Wait for the mpe to halt after run or single step.

`*detect-conflicts*` is bound to `#t` to cause instruction conflicts to be detected. To disable this feature set this to `#f`.

`(select-processor i &optional debugger)`  
Selectes the specified processor in the specified debugger. The debugger defaults to `*default-debugger*`.

`*use-dependencies*` is bound to `#t` to indicate that files should only be reassembled if one of the source files they depend on have changed since the last assembly. This is done by reading a dependency list from the object (".mpo") file. To force reassembly on every load or restart set this to `#f`.

`(load-debug-file filename &key debugger)`  
Load nuon debug file with `*default-mpe-debugger*` bound to the specified debugger. The debugger defaults to `*default-mpe-debugger*`.

`(load-source-file filename &key initialize? debugger)`  
Load nuon source code into the mpe associated with the debugger. The debugger defaults to `*default-mpe-debugger*`.

`(load-object-file filename &key initialize? debugger)`  
Load nuon object code into the mpe associated with the debugger. The debugger defaults to `*default-mpe-debugger*`.

`(load-coff-file filename &key initialize? debugger)`  
Load coff format binary code into the mpe associated with the debugger. The debugger defaults to `*default-mpe-debugger*`.

`(load-srecord-file filename &key debugger)`  
Load a motorola s-record file into the mmp associated with the debugger. The debugger defaults to `*default-debugger*`.

`(load-binary-file addr filename &key debugger)`  
Load a binary file at the specified address mpe associated with the debugger. The debugger defaults to `*default-debugger*`.

#### HANDY FUNCTIONS

`(set-source-path! path &optional debugger)`  
Set the source path for the specified mpe debugger. The debugger defaults to `*default-mpe-debugger*`.

`(run &optional debugger)`  
Start the mpe associated with the specified mpe debugger running. Instructions are executed when the mpe is clocked. The debugger defaults to `*default-mpe-debugger*`.

`(stop &optional debugger)`  
Stop the mpe associated with the specified mpe debugger. The debugger defaults to `*default-mpe-debugger*`.

`(restart &optional debugger)`  
Restarts the last program loaded into the mpe associated with the specified debugger. The debugger defaults to `*default-mpe-debugger*`.

`(dump &optional debugger)`  
Dump the registers of the mpe associated with the specified debugger. The debugger defaults to `*default-mpe-debugger*`.

`(write-image name &optional x-size y-size &key base mode mpe)`

Write an image from display memory to a .pcx file. The x-size and y-size parameters default to the display height and width. The base defaults to the start of external ram and the mode defaults to \*display-mode\*. The mpe defaults to \*default-mpe\*.

(write-raw-image name &optional x-size y-size &key base mode mpe)  
Write an image from display memory to a .pcx file. The x-size and y-size parameters default to the display height and width. The base defaults to the start of external ram and the mode defaults to \*display-mode\*. The mpe defaults to \*default-mpe\*. This function differs from write-image in that no color space conversion is performed; the Y component of colors is written into the green channel of the output image, Cr into the red, and Cb into the blue.

(runtime-eval expr &optional debugger)  
Evaluate the specified expression in the debugger context. This includes bindings for the symbols \*default-mpe-debugger\* and \*default-mpe\*. The debugger defaults to \*default-mpe-debugger\*.

(elapsed-ticks &optional processor)  
Return the number of elapsed ticks for the specified processor. The processor defaults to \*default-mmp\*.

(bus-info &optional processor)  
Return bus usage information for the specified processor. The processor defaults to \*default-mmp\*.

(find-symbol pname &optional debugger)  
Find the value of the named symbol. The processor defaults to \*default-mpe-debugger\*.

(make-assembler-command src-file bin-file err-file flags)  
The debugger calls this function to build a command line to invoke the assembler. It returns the command line string.

For example:

```
(make-assembler-command "foo.a" "foo.mpo" "foo.err" "-alpha,broken")
```