

1

Accessing Macintosh Traps from Object Logo

=====

Copyright ©1987 Coral Software Corp.

This is an Object Logo specification which describes how Macintosh traps are accessed. Users should be aware that Macintosh traps perform at best minimal error checking and will generally crash the computer if given improper arguments. See the file "Traps Examples" in the "Examples" folder on the Object Logo disk for examples of how the primitives presented here are used, and *Inside Macintosh* for detailed information on the traps themselves.

Two primitives are supplied for calling Macintosh traps: `.RegTrap` (for register-based traps) and `.StackTrap` (for stack-based traps). Object Logo performs a certain amount of coercion of arguments for these primitives, as explained below, but very little error-checking of the values and parameters passed to them.

```
.RegTrap TrapNum (global command or operation)
(.RegTrap TrapNum {"Errchk"} {Reg1 Type1 Value1} ... {RegN TypeN ValueN}
  {ReturnReg Return Type})
```

calls the Macintosh register-based trap specified by trap number *TrapNum* with arguments specified by the *Reg*, *Type*, and *Value* inputs. If *ReturnReg* and *Return Type* are specified, it outputs the value returned by the trap; otherwise, it doesn't output. If the input immediately following *TrapNum* is the word `Errchk`, `.RegTrap` checks the low word of register D0 when the trap returns and signals an error if it is non-zero. Each *{Reg Type Value}* triplet specifies a register for passing a value, the type of value, and the value itself. The *{ReturnReg Return Type}* pair specifies which register to return a value from (if any) and what type of value it is. See **Argument Coercion for Traps**, below.

TrapNum is the actual machine-language instruction that is executed, so it should be the decimal equivalent of an "A-trap". The value of *TrapNum* should reflect the desired setting of bits for the Macintosh Trap Dispatcher (for example, a trap number with bit 9 set indicates that register A0 should be preserved through the trap call).

```
? to _newhandle :size
> output (.regtrap 41250 "d0 "long :size "a0 "handle)
> end
_NEWHANDLE defined.
? to _disposhandle :h
> (.regtrap 40995 "a0 "handle :h)
> end
_DISPOSHANDLE defined.
? make "hdl _newhandle 30
? show :hdl
{A HANDLE}
? _disposhandle :hdl
```

2

`.StackTrap TrapNum` (global command or operation)
`(.StackTrap TrapNum {"Errchk"} {Type1 Value1} ... {TypeN ValueN} {ReturnType})`

calls the Macintosh stack-based trap specified by trap number *TrapNum* with arguments specified by the *Type* and *Value* inputs. If *ReturnType* is specified, it outputs the value returned by the trap; otherwise, it doesn't output. If the input immediately following *TrapNum* is the word `Errchk`, `.StackTrap` checks the top word of the processor stack when the trap returns and signals an error if it is non-zero. Each {*Type Value*} pair specifies a type of value and the value itself. The *ReturnType* input specifies what type of value is to be returned, if any. See **Argument Coercion for Traps**, below.

TrapNum is the actual instruction that is executed, so it should be the decimal equivalent of an "A-trap". The value of *TrapNum* should reflect the desired setting of bits for the Macintosh Trap Dispatcher (for example, a trap number with bit 9 set indicates that register A0 should be preserved through the trap call).

```
? to _sysbeep :dur
> (.stacktrap 43464 "word :dur)
> end
_SYSBEEP defined.
? _sysbeep 20
? _sysbeep 17/2
? to _tickcount
> output (.stacktrap 43381 "long)
> end
_TICKCOUNT defined.
? show _tickcount
2214344
? to _frontwindow
> output (.stacktrap 43300 "pointer)
> end
_FRONTWINDOW defined.
? to _hidewindow :w
> (.stacktrap 43286 "errchk "pointer :w)
> end
_HIDEWINDOW defined.
? to _showwindow :w
> (.stacktrap 43285 "errchk "pointer :w)
> end
_SHOWWINDOW defined.
? (make "w _frontwindow) (_hidewindow :w) (_sysbeep 60) (_showwindow :w)
```

3

Argument Coercion for Traps

As mentioned above, the user must specify the types of values being passed to and returned from a Macintosh trap. These types should match those specified for the trap in *Inside Macintosh*. The `.RegTrap` and `.StackTrap` primitives allow the following types to be specified:

Input value types:

WORD	The value is coerced (if possible) to a 16-bit integer.
LONG	The value is coerced (if possible) to a 29-bit integer. (Use two consecutive WORD values to pass a full 32-bit integer.)
HANDLE	If the value is a Mactype, the handle containing the data is used. (This works for the Cursor, Picture, Region, and Polygon Mactypes). If the value is a Macintosh handle, it is used without coercion.
POINTER	If the value is a word, a Pascal-style pointer to a copy of the word's characters is used. If the value is a MacType, the handle containing the data is locked and dereferenced, and the resulting pointer is used. (This works for the Cursor, Picture, Region, and Polygon Mactypes). If the value is a Macintosh handle, it is dereferenced and locked, and the resulting pointer is used. If the value is a Macintosh pointer, it is used without coercion.

Output value types:

WORD	A 16-bit positive integer is returned.
LONG	A 29-bit positive integer is returned.
STRING	A Logo word is returned.
POINTER	The value returned by the trap is returned without coercion.
(other)	If the trap returned a handle, an Object Logo Mactype of the specified type is created using the value returned by the trap and returned. Otherwise, the value returned by the trap is returned without coercion.

The value type designators have been chosen to conform more to *Inside Macintosh* conventions than to Object Logo. Specifically, note that `WORD` refers to a 16-bit numerical value, not an Object Logo word.

There is no way to get a full 32-bit value returned from a stack trap. Logo words being passed to traps are truncated to 255 characters.