Boolean data type

```
#include <Types.h>

typedef enum {    /* ie, enumerated 16-bit integer */
  false = 0,
  true = 1,
  FALSE = 0,
  TRUE = 1
} Boolean;
```

Notes: As an enumerated C-language data type, a Boolean is simply a 16-bit short containing either 0 or 1. In a statement such as:

```
if ( foo ) { doit... }
```

'doit' will be executed in all cases except when 'foo' is exactly zero. Thus, you can use a short instead of a Boolean.

The Pascal storage of a 'Boolean' data type is somewhat different - and the difference can be important. Pascal, and thus the Macintosh toolbox, considers a BOOLEAN to be exactly 1 bit long; for instance, a PACKED array of 16 BOOLEANs will fit into a 16-bit word (as in **GetKeys**). When a BOOLEAN toolbox function returns 'true', the actual value is 256 (bit 0 of the high byte is set).

Your compiler's glue library is nice enough to fetch that byte off the stack and extend it into a 16-bit integer. When calling the toolbox, the compiler also takes care of storing non-zero values into the high byte of any BOOLEAN parameter. But there are still two places where this might cause a problem:

- When you create a '**callback**' routine, such as a file filter (**SFGetFile**) or an event pre-processing filter (**ModalDialog**), you are expected to return a BOOLEAN to the system. In that case, you must NOT return an integer 1. As long as you correctly declare your routine as a "pascal Boolean", you will never have a problem.
- When accessing global variables listed as BOOLEAN, be sure to check the first byte at the global address. 0 means false, else means true. In some cases, a value of 0xFF is used to indicate true.