8-bit Data Types Integral data type

#include <Types.h>

char	signed byte	range: -128...127
SignedByte	signed byte	range: -128...127
Byte	unsigned byte	range: 0...255
VHSelect	unsigned char	range: 0..1; v = 0, h = 1
Pascal String	length prefix Byte, followed by up to 255 characters	
ASCIIZ String	unlimited number of characters followed by ASCII NUL (0x00)	

Notes: In all cases where an 8-bit parameter is passed to the system, your compiler must and will coerce it into a 16-bit value going in and back to an 8-bit value on the way out.

Signed data types cause automatic sign extension on conversion to larger data types. Also comparisons works differently. For instance:

```
char    theChar;
Byte    theByte;
short   theInt;

theChar = 0xFF;
theByte = 0xFF;

theInt = theChar;           /* theInt becomes -1;   ie, 0xFFFF  */
theInt = theByte;           /* theInt becomes 255;  ie, 0x00FF */

if ( theChar < 0 ) { .... }     /* TRUE */
if ( theByte < 0 ) { .... }     /* FALSE */
```

Strings: Character Arrays

In this Guide and in all Macintosh references, a "string" refers to a length-prefixed, Pascal-style string; i.e., and array of up to 256 bytes with the first byte indicating the length of the *rest of the array*. Other ways to refer to a Pascal string include Pstring, p-string and (rarely) Lstring.

The Mac data types Str255, StringPtr, and StringHandle all assume a length-prefixed array of 8-bit bytes. When passing the address of an empty string, you may pass the address of a byte of 0, or (usually) just pass 0 (NIL) as the address itself.

An ASCIIZ string is a common name for a standard C-language style array of characters. It is composed of any number of 8-bit bytes, terminated by a byte of 0. This may also be called a C-string.

The Mac system does not use ASCIIZ strings directly in any of its functions. However, for many "**String**xxx" function there is often a "**Text**xxx"

counterpart which works with unformatted text and requires a length parameter (obtainable via the C library call, strlen()).

Be aware that as an array of chars, each element will be treated as a signed entity. For instance:

```
if ( myString[35] < 0 ) { ...doIt... }
```

doIt will be executed whenever the character has a value higher than 127 (signed chars larger than 127 are negative and get promoted to negative integers).