Macintosh Technical Notes

#166: MPW C Functions Using Strings or Points as Arguments

See also: MPW C Manual, Appendix H:
"Functions Using Strings or Points as Parameters"

Written by: Fred A. Huxham November 2, 1987
Updated: March 1, 1988

MPW 2.0 includes new C interfaces to ROM routines which no longer do string and point conversions. These new interfaces are described here.

In MPW prior to 2.0, the C interfaces to Macintosh OS and Toolbox routines that had strings or points as arguments required following these rules:

1. Strings must be passed as C strings (null terminated).
2. Points must be passed by address.

With this method, all these functions would end up calling glue code to:

1. Convert the C strings to Pascal strings.
2. Dereference the address of the Point before pushing it onto the stack.

Because of this your applications ended up being slightly larger (due to the glue code needed) and slightly slower (due to the time it takes to execute the glue code).

MPW 2.0 C interfaces include a new set of ALL CAPS routines that have strings or points as arguments. These routines require following these rules:

1. Strings must be passed as Pascal strings (preceded by a length byte).
2. Points must be passed by value.

Calling these new routines results in smaller and faster code. In other words, these new interfaces are your friend.

Some Examples

First, some Point examples:

The routine `PtInRect`, (old style, mixed case), requires a point passed by address. The new interface:

```
pascal Boolean PTINRECT(pt,r)
    Point pt;
    Rect *r;
    extern 0xA8AD;
```

requires that the `Point` argument be passed by value.

And now, some string examples:

The routine `StringWidth`, (old style, mixed case), required a C string as an argument. The new interface:

```
pascal short STRINGWIDTH(s)
    Str255 *s;
    extern 0xA88C;
```

requires that the argument be passed as a Pascal string.

Pascal Strings

Another new feature of MPW 2.0 C is the creation of Pascal strings. You can now create a Pascal string by using the "`\p`" option. The following example demonstrates this new feature:

```
cString1 = "This is a C string"
pString2 = "\pThis is a Pascal string"
```

The first line will create a C, null terminated, string, while the second line will create a Pascal, preceded by a length byte, string.