#104: MPW: Accessing Globals From Assembly Language

See also: MPW Reference Manual

Written by: Jim Friedlander January 12, 1987
Updated: March 1, 1988

This technical note demonstrates how to access MPW Pascal and MPW C globals from the MPW Assembler.

To allow access of MPW Pascal globals from the MPW Assembler, you need to identify the variables that you wish to access as external. To do this, use the `{$Z+}` compiler option. Using the `{$Z+}` option can substantially increase the size of the object file due to the additional symbol information (no additional code is generated and the symbol information is stripped by the linker). If you are concerned about object file size, you can “bracket” the variables you wish to access as external variables with `{$Z+}` and `{$Z-}`. Here’s a trivial example:

Pascal Source

```
PROGRAM MyPascal;
USES
   MemTypes,QuickDraw,OSIntf,ToolIntf;

VAR
   myWRect: Rect;
{$Z+} {make the following external}
   myInt: Integer;
{$Z-} {make the following local to this file (not lexically local)}
   err: Integer;

PROCEDURE MyAsm; EXTERNAL; {routine doubles the value of myInt}

BEGIN {PROGRAM}
   myInt:= 5;
   MyAsm; {call the routine, myInt will be 10 now}
   writeln('The value of myInt after calling myAsm is ', myInt:1);
END. {PROGRAM}
```

Assembly Source for Pascal

```
        CASE   OFF           ;treat upper and lower case identically
MyAsm   PROC   EXPORT        ;CASE OFF is the assembler's default
        IMPORT myInt:DATA    ;we need :DATA, the assembler assumes CODE
        ASL.W  #1,myInt      ;multiply by two
        RTS                  ;all done with this extensive routine, whew!
        END
```

The variable `myInt` is accessible from assembler. Neither `myWRect` nor `err` are accessible. If you try to access `myWRect`, for example, from assembler, you will get the following linker error:

```
     ### Link: Error   Undefined entry name:   MYWRECT.
```

C Source

In an MPW C program, one need only make sure that `MyAsm` is declared as an external function, that `myInt` is a global variable (capitalizations must match) and that the `CASE ON` directive is used in the Assembler:

```
     #include <types.h>
     #include <quickdraw.h>
     #include <fonts.h>
     #include <windows.h>
     #include <events.h>
     #include <textedit.h>
     #include <dialogs.h>
     #include <stdio.h>

     extern MyAsm();    /* assembly routine that doubles the value of myInt */
     short myInt;      /* we'll change the value of this variable from MyAsm */

     main()
     {
     WindowPtr MyWindow;
     Rect myWRect;

     myInt = 5;
     MyAsm();
     printf(" The value of myInt after calling myAsm is %d\n",myInt);
     } /*main*/
```

Assembly source for C

```
             CASE   ON             ;treat upper and lower case distinct
     MyAsm   PROC   EXPORT         ;this is how C treats upper and lower case
             IMPORT myInt:DATA     ;we need :DATA, the assembler assumes CODE
             ASL.W  #1,myInt       ;multiply by two
             RTS                   ;all done with this extensive routine, whew!
             END
```