

Migrating Your Application to MPLAB® C30 v1.33

Updated 8/19/2005

Version 1.30 of the MPLAB C30 compiler (and associated tools) introduced several features that benefit application developers. One of the most significant additions is a new memory allocator, which allows more control over the placement of objects in dsPIC® memory. The new memory allocator was designed for backward compatibility with pre-existing projects. In most cases, projects will build successfully, although the arrangement of sections in memory may be different. Depending on how the pre-existing source code is written, some warnings or errors may be reported. This document describes how to deal with such cases. *No migration issues have been identified when porting an application from version 1.32 to 1.33.*

Changes to the C Syntax

The previous C syntax relied on reserved section names to determine the eventual memory placement. Beginning with v1.30, new variable attributes are used to specify dsPIC memory spaces such as *xmemory*, *ymemory*, and *eedata*. The new attribute syntax is more flexible and allows for combinations such as *near xmemory*. In most cases the new attributes may be combined with user defined section names (*auto_psv* is the notable exception.) The attribute syntax is fully described in Sections 2.3.1 and 2.3.2 of *MPLAB C30 C Compiler User's Guide* (DS51284d).

The MPLAB C30 compiler will accept the previous section syntax, but in some instances will interpret it differently. The memory placement convenience macros, *_YDATA*, *_YBSS*, and others are compatible with the new C syntax. The following table gives a terse comparison between the previous syntax and new syntax.

v1.20.xx section attribute syntax	v1.33 preferred syntax
Selecting X/Y memory regions	
section(".xbss")	space(xmemory) ^{1,2}
section(".xdata")	space(xmemory) ^{1,2}
section(".ybss")	space(ymemory) ^{1,2}
section(".ydata")	space(ymemory) ^{1,2}
Selecting FLASH memory regions	
section(".const,r")	space(auto_psv)
section(".eedata,r")	space(eedata)
Other...	
section(".pbss")	persistent

New MPLAB C30 attributes are highlighted in the following table:

New Attribute	Description
address(n)	Place data/function at a specific address
Psv	Place data in an unmanaged PSV section
reverse(n)	Align data to end at a particular boundary
Unordered	Allow the linker to place this variable anywhere in memory.

¹ By default, the compiler honors the *-msmall-data* or *-mlarge-data* option to determine nearness; a specific setting may be used by adding the *near* or *far* attribute.

² Assignment to the *.bss* or *.data* memory is determined automatically by the compiler depending upon whether or not initial values are provided.

Changes to the Assembly Syntax

The previous assembly syntax relied on reserved section names and single character flags to determine the eventual memory placement. Beginning with v1.30, new section attributes are used to specify dsPIC memory spaces and types such as *bss*, *data*, and *persistent*. The new attributes may be used in combination with user defined section names. The section attribute syntax is fully described in Section 6.3 of *MPLAB ASM30, LINK30, and Utilities User's Guide* (DS51317d).

The assembler will accept the previous section syntax with the exception of subsection numbers (which are rarely, if ever, used on the dsPIC.) The following table gives a terse comparison between the previous syntax and the new syntax.

v1.20.xx assembler section syntax	v1.33 preferred syntax
Selecting X/Y memory regions	
.section .xbss,"b"	.section .xbss,bss,xmemory
.section .xdata,"d"	.section .xdata,data,xmemory
.section .ybss,"b"	.section .ybss,bss,ymemory
.section .ydata,"d"	.section .ydata,data,ymemory
Selecting FLASH memory regions	
.section .const,"r"	.section .const,psv ³
.section .eedata,"r"	.section .eedata,eedata ⁴
.section sect,"x"	.section sect,code
Other...	
.section .bss,"b"	.section .bss,bss
.section .data,"d"	.section .data,data
.section .nbss,"b"	.section .nbss,bss,near
.section .ndata,"d"	.section .ndata,data,near
.section .pbss,"b"	.section .pbss,persistent

New section attributes are highlighted in the following table:

New Section Attribute	Description
address(n)	Place section at a specific address
align(n)	Align section to begin at a particular boundary
reverse(n)	Align section to end at a particular boundary
near	Allocate in the first 8K of memory

³ Sections with the *psv* attribute are allocated in program memory; the upper byte is not accessible.

⁴ Sections with the *eedata* attribute are allocated in data EEPROM memory.

Issues Related to ELF Object Format

Beginning with v1.30, MPLAB C30 supports ELF as an alternative to the COFF object file format. This option is intended to facilitate source level debugging of optimized code. The source level debugging format used with ELF object files is known as DWARF2.0. MPLAB IDE introduced support for ELF/DWARF in v7.01. IDE support is improved in MPLAB IDE v7.10.

By default, MPLAB IDE projects will be configured to use the COFF object format, as before. If the ELF/DWARF option is selected for an existing project, the project must be updated to use a v1.3x linker script. Old linker scripts are not compatible with ELF/DWARF. Linker scripts included with v1.3x may be used with either object format.

Changes to Library Names

If a project includes an application library such as `libdsp.a`, or a peripheral library such as `libp30F2010.a`, the project may fail to build because of changes to the library names. Beginning with v1.30, two versions of each library are included, one for each object format. For example, if the project includes `libdsp.a`, remove it and add `libdsp-coff.a`, or `libdsp-elf.a`, as appropriate.

If the project is built from a command line, and the library is specified with “-l” syntax (for example, `-ldsp`), the correct library will be selected automatically. For more information about the command line behavior, see section 1.2 of *dsPIC Language Tools Libraries* (DS51456B).

Changes to the Linker Scripts

Memory allocation in v1.33 proceeds in two steps. First a sequential allocation pass is performed, based on the mapping of input sections to output sections as defined in the linker script. This pass is equivalent to the memory allocation strategy of versions prior to v1.30. Next a best-fit allocator is invoked, to locate any sections which are not explicitly mapped in the linker script. Unmapped sections are located according to their attributes, such as `xmemory`, `ymemory`, etc. Details are provided in Section 10.5 of *MPLAB ASM30, LINK30, and Utilities User's Guide* (DS51317d).

The standard linker scripts included with MPLAB C30 have been revised to benefit from the new memory allocation strategy. All of the standard data memory sections have been removed, so they become unmapped and therefore eligible for best-fit allocation. To support this approach, the standard section names now have implied attributes. A list of reserved section names with implied attributes is provided in Section 6.3 of *MPLAB ASM30, LINK30, and Utilities User's Guide* (DS51317d).

Projects which specify a standard linker script should build correctly with v1.33. An issue with initialized variables, multiple source files, and old linker scripts that existed in v1.30 was corrected in v1.31.

Projects which include a custom linker script should build correctly with v1.33. To take advantage of the new memory allocator, comment out or remove statements in the custom linker script which define the following sections: `.xbss`, `.xdata`, `.pbss`, `.nbss`, `.ndata`, `.bss`, `.data`, `.ybss`, `.ydata`. Also remove the statements which calculate `__X_OVERFLOW` and `__NEAR_OVERFLOW`. As noted above, sections which do not appear in the linker script are eligible for best-fit memory allocation. To verify the effect of these changes, compare the memory usage report⁵ before and after editing the linker script.

Previously, the primary motivation for creating a custom linker script was to specify an absolute address for a user-defined section. Beginning with v1.30, this goal is more easily accomplished by using the `address()` attribute in C or assembly language, as shown:

```
int var __attribute__((address(0x900))); /* C variable definition */
.section sect,bss,address(0x900)      ; assembly section definition
```

Specific Error Messages

The following error messages may be reported when attempting to build pre-existing projects with v1.33. Errors must be corrected before a project can be successfully built.

Error: Attributes for section 'name' conflict with implied attributes

This assembler error indicates that a non-standard section flag has been applied to a reserved section name. Probably a "b" flag has been specified for a section that contains initial data values (for correct usage, the "d" should have been specified). Previous versions of the assembler may have accepted the non-standard flag; v1.33 will not. Although simply changing the section flag will correct this error, best practice would convert the source code to use new section attributes, as shown above.

Error: Could not allocate data memory

This linker error message is displayed when a section's attribute and alignment requirements can not be satisfied. If the project previously linked successfully, the problem may be related to variables allocated in Y memory. This issue is limited to devices with 8K of RAM, where only a portion of Y memory qualifies as `near`.

C30 allocates all variables in near memory space unless the `-mlarge-data` option has been selected. In previous versions, variables defined in Y memory were excluded from this rule. In v1.33, some projects may fail to link on devices with 8K of RAM because a solution for `ymemory,near` can not be found. The solution is to add the `far` attribute to some or all of the Y memory variables, or to specify the `-mlarge-data` option.

pic30-elf-ld: target coff-pic30 not found

This linker error message indicates that a pre-v1.30 linker script was included in a project that specifies the ELF object format. Old linker scripts are not compatible with ELF. The solution is to migrate the project to use a v1.3x linker script.

⁵ The new linker option `--report-mem` writes a memory usage report to the console.

Specific Warning Messages

The following warning messages may be reported when building pre-existing projects with version 1.33. Warnings do not prevent a project from building successfully, but should be reviewed and understood by the programmer.

Warning: Quoted section flags are deprecated⁶, use attributes instead

This assembler warning may appear whenever a section is explicitly named in assembly language or in the MPLAB C30 section attribute syntax. In previous versions, quoted section flags were used to specify section type, such as "x" for code and "d" for data. Beginning with version 1.30, new section attributes were introduced for this purpose. Although quoted section flags will still work, best practice would convert the statement to use the new section attributes, as shown above.

Warning: Implied attributes for section 'name' are deprecated

This assembler warning may appear whenever a section is explicitly named in assembly language or in the MPLAB C30 section attribute syntax. In previous versions reserved section names were used to specify memory spaces such as X memory or Y memory. Beginning with v1.30, new section attributes were introduced for this purpose. Although reserved section names will still work, best practice would convert the statement to use the new section attributes, as shown above.

⁶ Support for deprecated features may be removed in a future version.