IAR Embedded Workbench®

IAR Embedded Workbench® for AVR32 Migration Guide

Migrating from version 3.x to version 4.x

Mv4x_AVR32-1

EDITION NOTICE

First edition: May 2011

Part number: Mv4x_AVR32-1

This guide applies to version 4.1x of IAR Embedded Workbench® for Atmel® Corporation's AVR32 microprocessor family.

Internal reference: M1, Too6.1, ISUD.

Migrating from version 3.x to 4.x

This chapter gives hints for porting your application code and projects to the new version 4.x from the old version 3.x.

Migration considerations

To migrate your old project, consider these topics:

- IAR Embedded Workbench IDE
- C language changes
- Runtime library changes.

Note that not all items in the list might be relevant for your project. Consider carefully which actions that are needed in your case.

Code written for version 3.x might generate warnings or errors in version 4.x.

IAR Embedded Workbench IDE

When you upgrade to the new version of the IAR Embedded Workbench IDE, you must consider the issues described in this section.

INSTALLATION DIRECTORY

When you install your new version of the IAR Embedded Workbench IDE, you cannot install it in the same installation directory as your old version.

The old default installation path is:

`c:\Program Files\IAR Systems\Embedded Workbench 5.`*`n`*`\`

The new default installation path is:

`c:\Program Files\IAR Systems\Embedded Workbench 6.`*`n`*`\`

Note the difference in version number of IAR Embedded Workbench.

PROJECT SETTINGS IN THE OPTIONS DIALOG BOX

The **Options** dialog box—available from the **Project** menu—has changed. This table lists the most important changes:

Category>Page	Changes
C/C++ Compiler>Language	See *C language changes*, page 4.
Linker	The tabs are listed in a new order.
Linker>Processing	The name of this page is now Checksum.

Table 1: Overview of changes in the Project options dialog box

PROJECT FILES

Even though some of the pages in the **Options** dialog box have changed, your old project file can be used in your new version of the IAR Embedded Workbench IDE.

C language changes

In version 4.x, the compiler by default conforms to the C99 standard (ISO/IEC 9899:1999 including technical corrigendum No.3), hereafter referred to as *Standard C* in this guide. Optionally, you can make the compiler conform to the C89 standard instead (ISO 9899:1990 including all technical corrigenda and addenda). In C89 mode, you cannot use any C99 language features or any C99 library symbols.

In version 3.x, the compiler by default conforms to the C89 standard. Optionally, you can use some C99 features.

To migrate to version 4.x, you must consider:

- Options for language support
- Options for language conformance
- Obsolete C89 features in your source code.

OPTIONS FOR LANGUAGE SUPPORT

This table lists the differences in the options for enabling language support:

Language features	In version 4.x	In version 3.x
C89*	`--c89`†	Supported by default.
C99* (Standard C)	Supported by default.	Some features available when `-e` is used.

Table 2: Enabling language features

*** C89 and C99, respectively, but with some minor exceptions. For more information, see the compiler documentation.**

† `--c89` **disables C99 library symbols and C99 language features.**

Note: C99 mode does not allow variable length arrays (VLA) by default; use the command line option `--vla` to enable such support. Embedded C++ has not changed.

OPTIONS FOR LANGUAGE CONFORMANCE

The options for C/C++ language conformance differ between the two versions; this table lists these differences:

In version 4.x Option in IDE vs on the command line	In version 3.x Option in IDE vs on the command line	Description
Standard with IAR extensions `-e`	**Allow IAR extensions** `-e`	Accepts IAR extensions and IAR relaxations to Standard C.
Standard Supported by default on the command line	**Relaxed ISO/ANSI** Supported by default on the command line	Accepts IAR relaxations to Standard C.
Strict `--strict`	**Strict ISO/ANSI** `--strict_ansi`	Strict adherence to the standard.

Table 3: Options for language conformance

OBSOLETE C89 FEATURES IN YOUR SOURCE CODE

There are some C89 features that are accepted by the compiler in version 3.x, but which are not accepted by the compiler in version 4.x when you compile in C99 mode. Warnings or errors will be generated. To omit these diagnostic messages, you must either compile the source code in C89 mode or rewrite your source code.

These C89 features are not accepted by the compiler in version 4.x when compiling in C99 mode:

- Implicit `int` variables

```
static k; /* k was implicit int. */
```

- Implicit `int` parameters

```
myFunction(i,j)
{
  /* i and j were implicit int. */
}
```

- Implicit `int` returns

```
myFunction()
{
  /* Returned implicit int 0. */
}
```

- Implicit returns

```
myFunction()
{
  /* Returned 0. */
}
```

- Implicit returns

```
myFunction()
{
  return; /* Returned 0. */
}
```

Runtime library changes

In version 4.x, the compiler and assembler automatically search for system header files in a predestined directory (relative to the compiler/assembler executable file). In version 3.x, you must specify the include file search paths explicitly.

In version 4.x, these compiler options are available for this:

`--dlib`	Uses DLIB system header files.
`--dlib_config` *`token`*	Uses DLIB system header files. The option also lets you specify the runtime library configuration to use. In version 3.x, the option lets you specify a runtime library configuration file, but in version 4.x the option also accepts tokens.
`--no_system_include`	Disables the automatic search for system header files. You must specify the include file search path explicitly, just as in version 3.x. This option is useful if you have well-established script files for building your application project and you do not want to apply to the new include file system immediately.
`--system_include_dir`	Specifies the include directory explicitly, where the compiler can find the system header files.

These corresponding assembler options are available:

`--no_system_include`	Disables the automatic search for system header files. You must specify the include file search path explicitly, just as in version 3.x. This option is useful if you have well-established script files for building your application project and you do not want to apply to the new include file system immediately.
`--system_include_dir`	Specifies the `inc` directory explicitly, where the assembler can find the system header files.

For detailed reference information about these options, see the *IAR C/C++ Development Guide for AVR32* and the *AVR32 IAR Assembler Reference Guide*.