Migration guide



Migrating from Keil µVision® for 8051 to IAR Embedded Workbench® for 8051

Use this guide as a guideline when converting project files from the μ Vision IDE and source code written for Keil toolchains for 8051 to IAR Embedded Workbench for 8051.

| | Product | Version number |
|----------------|---------------------------------|-----------------|
| Migrating from | Keil µVision IDE (C51 compiler) | V4.x, V5.x |
| Migrating to | IAR Embedded Workbench for 8051 | V9.30 and newer |

Migration overview

Migration of an existing project from Keil μ Vision requires that you collect information about your current project and then apply this information to the new IAR Embedded Workbench project. In addition, you need to make some changes in the actual source code. The information in this guide simplifies this process.

Note: If you are new to using IAR Embedded Workbench, we suggest that you first look at the user guides and tutorials which you can find in the IAR Information Center.

Project conversion

To migrate existing Keil μ Vision applications to IAR Embedded Workbench there is a tool called **IAR Project Converter**. This is a GUI application included with IAR Embedded Workbench, available via the **Tools** menu.

The **IAR Project Converter** tool converts μ Vision project files into IAR Embedded Workbench project files without changing the original file. Information about source files, include paths, defined symbols, and build configuration is transferred. As an option, also source code substitutions are performed and you can add your own substitution rules including support for regular expressions. There are a number of pre-defined substitution rules.

Procedure

- 1. Start IAR Embedded Workbench.
- 2. Start IAR Project Converter available in the Tools menu.
- Navigate to the µVision project to convert by clicking the browse button.
- 4. Click the **Execute** button and a new IAR Embedded Workbench project file will be created.
- Add the new project to a workspace by choosing Add Existing Project in the Project menu.
- 6. Set the relevant project options by choosing Options in the Project menu.
 Hint: Open the original project in µVision, walk through the options and set the corresponding options in IAR Embedded Workbench as suggested in the section *Important tool settings* below.

| text directory of source project C: \Kel_v5\C51\Examples\BLINKY Project file conversion | AR Pr | oject Converter - 4.0.2 | | | | 23 |
|---|---|---|---|------|--------|---|
| C:\Kel_v5\C51\Examples\BLINKY Project file conversion ✓ Enable ?? Project type Keil uVision for 8051 File extension(s) *.uvproj *.uvproj Source code substitution ? I (\hr +1^>\code(\s+1*1\V)) \$1_code\$2 2 (\hr +1^>\code(\s+1*1\V)) \$1_code\$2 2 (\hr +1^>\code(\s+1*1\V)) \$1_code\$2 3 (\hr +1^>\code(\s+1*1\V)) \$1_code\$2 4 (\hr +1^>\code(\s+1*1\V)) \$1_coda\$2 5 (\hr +1^>\codata(\s+1*1\V)) \$1_coda\$2 6 (\hr +1^>\codata(\s+1*1\V)) \$1_coda\$2 7 ((\n^*(\s*1\+1)^\r)) \$1_coda\$2 6 (\hr +1^>\codata(\s+1*1\V)) \$1_coda\$2 7 ((\n^*(\s*1\+1)^\code(\s+1*1\V)) \$1_coda\$2 6 (\hr +1^>)codata(\s+1*1\V)) \$1_coda\$2 7 ((\n^*(\s*1\+1)^\coda(\s+1*1\V)) \$1_coda\$2 7 ((\n^*(\s*1\+1)^\coda(\s*1*1\V)) \$1_coda\$2 8 ~((\n^*(\s*1\+1)^\coda(\s*1*1\V)) \$1_coda\$2 8 ~((\n | ot dire | ctory of source project | | | | |
| Project file conversion Project type Kell uVision for 8051 File extension(s) *.uvproj Source code substitution Enable @ Tag changed code with comments Substitution rules (the rules will be applied as ordered in the list) Nr From 1 (h+1^)code(s+1 *)) \$1_code\$2 2 (h+1^)data(s+1 *)) \$1_cidata\$2 3 (h+1^)idata(s+1 *)) \$1_idata\$2 4 (h+1^)idata(s+1 *)) \$1_idata\$2 5 (h+1^)pdata(s+1 *)) \$1_odata\$2 6 (h+1^)pdata(s+1 *)) \$1_odata\$2 7 ((/n*(h+1))fr(s+1 #pragma error "Use o 8 ~((h+1^)pdata(s+1 *)) \$1_odata\$2 7 ((/n*((1a-2A-2O-9 y+#pragma error "Use o) *.c;*.cpp;*.h;*.hpp;*.asm;*.s Execute Close | C:\Ke | il v5\C51\Examples\BLINKY | | | | |
| Project file conversion ✓ Enable Project type Keil uVision for 8051 File extension(s) *.uvproj Source code substitution ✓ Enable ✓ Tag changed code with comments Substitution rules (the rules will be applied as ordered in the list) ✓ Enable ✓ Enable ✓ Tag changed code with comments Substitution rules (the rules will be applied as ordered in the list) | | | | | | |
| Enable | roject | file conversion | | | | |
| Project type Keil uVision for 8051 File extension(s) *.uvproj Source code substitution Prise mather Project type Source code substitution Prise Project code substitution Substitution rules (the rules will be applied as ordered in the list) Nr From To Add Copy Edit Q (h+1^/)cdata((s+1)*[10) \$1cdata\$2 S (h+1^/)pdata((s+1)*[10) \$1pdata\$2 S (h+1^/)cdat((s+1)*[10) \$1pdata\$2 S (h+1^/)cdat(s+1)*[10] \$1pdata\$2 Perform substitutions on file(s) with extension(s) *.c;*.cpp;*.h;*.hp | | bla 🙆 | | | | |
| Keil uVision for 8051 File extension(s) *.uvproj Source code substitution Image: | Droje | ct type | | | | |
| Keil dvision for aussi ✓ File extension(s) *.uvproj *.uvproj Source code substitution ✓ Enable ✓ Tag changed code with comments Substitution rules (the rules will be applied as ordered in the list) ✓ Add I (h+1^)code(\s+1*1\)) \$1_code\$2 ✓ 2 (h+1^)data(\s+1*1\)) \$1_idata\$2 ✓ 3 (h+1^)idata(\s+1*1\)) \$1_idata\$2 ✓ 4 (h+1^)idata(\s+1*1\)) \$1_idata\$2 ✓ 5 (h+1^)idata(\s+1*1\)) \$1_pdata\$2 ✓ 6 (h+1^)ifa(\s+1*1\)) \$1_pdata\$2 ✓ 7 ((/c.*\h+1)^ifa(\s+1*1\)) \$1_pdata\$2 ✓ 8 ~(((h^*((a-zA-20-9) 1) mpragma error "Use o) Export Export Perform substitutions on file(s) with extension(s) *.c;*.cpp;*.h;*.hpp;*.asm;*.s ✓ ✓ *.c;*.cpp;*.h;*.hpp;*.asm;*.s ✓ Execute Close | Kaila | utivities for 0051 | | | | |
| File extension(s) *.uvproj Source code substitution | Kelli | UVISION FOR 8051 | | ~ | | |
| *.uvproj Source code substitution Image: Substitution rules (the rules will be applied as ordered in the list) Nr From 1 (\h+\ ^)code(\s+\ *\)) 2 (\h+\ ^)data(\s+\ *\)) 3 (\h+\ ^)data(\s+\ *\)) 3 (\h+\ ^)data(\s+\ *\)) 5 (\h+\ ^)data(\s+\ *\)) 5 (\h+\ ^)data(\s+\ *\)) 6 (\h+\ ^)data(\s+\ *\)) 7 (('\c^*(h+\))/far(\s+\) 8 ~(((\h^*((Ia-zA-20-9) I)n#pragma error "Use o) Perform substitutions on file(s) with extension(s) *.c;*.c;#.cpp;*.h;*.hpp;*.asm;*.s #.c;*.cpp;*.h;*.hpp;*.asm;*.s ✓ | File e | xtension(s) | | | | |
| Source code substitution Source code substitution Tag changed code with comments Substitution rules (the rules will be applied as ordered in the list) Nr From (h+1^)code(\s+1*1)) (h+2^-)data(\s+1)(*1)) (h+2^-)data(\s+1)(*1 | *.u\ | /proj | | | | |
| Source code substitution Tag changed code with comments Substitution rules (the rules will be applied as ordered in the list) Nr From 1 (\h+ ^\code(\s+ * \)) \$1_code\$2 2 (\h+ ^\code(\s+ * \)) \$1_data\$2 3 (\h+ ^\code(\s+ * \)) \$1_data\$2 3 (\h+ ^\code(\s+ * \)) \$1_adata\$2 4 (\h+ ^\code(\s+ * \)) \$1_bdata\$2 5 (\h+ ^\code(\s+ * \)) \$1_bdata\$2 6 (\h+ ^\code(\s+ * \)) \$1_bdata\$2 6 (\h+ ^\code(\s+ * \)) \$1_pdata\$2 7 ((\code(\s+ *)) \$1_pdata\$2 7 ((\code(\s+ *)) \$1_mpragma error "Use o, 8 \code(\h+ ^\code(\s+ *\) #pragma error "The Perform substitutions on file(s) with extension(s) *.c;*.c;pp;*.h;*.hpp;*.asm;*.s Execute Close | | | | | | |
| 1 (\h+\+)\code(\s+*\)) \$1_code\$2 2 (\h+\-)\data(\s+*\)) \$1_data\$2 3 (\h+\-)\data(\s+*\)) \$1_data\$2 4 (\h+\-)\data(\s+*\)) \$1_x\data\$2 5 (\h+\-)\data(\s+*\)) \$1_bdata\$2 6 (\h+\-)\data(\s+*\)) \$1_bdata\$2 7 ((\cap\cap\cap\cap\cap\cap\cap\cap\cap\cap | Ena | ible 🔞 🗹 T | ag changed code with comm | ents | | |
| 2 (h+ ^)data(\s+1* \)) \$1data\$2 3 (\h+ ^)idata(\s+1* \)) \$1idata\$2 4 (\h+ ^)idata(\s+1* \)) \$1idata\$2 5 (\h+ ^)idata(\s+1* \)) \$1idata\$2 6 (\h+ ^)idata(\s+1* \)) \$1bdata\$2 7 (((^.*(h+)))^far(\s+1) #pragma error "Use o Export Perform substitutions on file(s) with extension(s) *.c;*.cp;*.h;*.hpp;*.asm;*.s ✓ Execute Close | Ena Subst | ible @ T titution rules (the rules will b From | ag changed code with comm e applied as ordered in the l To | ist) | | Add |
| 3 (h+1^)/data(\$+1(*1()) \$1data\$2 4 (h+1^)/data(\$+1(*1()) \$1data\$2 5 (h+1^)/data(\$+1(*1()) \$1bdata\$2 6 (h+1^)/adata(\$+1(*1()) \$1bdata\$2 7 ((/h*((1a-zA-zO-9) '\n#pragma error "Use o Export Perform substitutions on file(s) with extension(s) *.c;*.cp;*.h;*.hpp;*.asm;*.s Execute Close | Ena Subst | ble @ ✓ T titution rules (the rules will b From (\h+ ^)code(\s+ * \)) | ag changed code with comm e applied as ordered in the l To \$1code\$2 | ist) | | Add Copy |
| * - (I+*)/Jodata(\$+I(* 0)) \$1Jodat\$2 Remove 5 (I++/)/Jodata(\$+ * 0)) \$1Jodat\$2 Import 6 (I++/)/Jodata(\$+ * 0) \$1Jodat\$2 Import 7 ((/(*(i++))/Jar(\$+) #pragma error "Use o Export 8 ^((I)*((Ia-zA-20-9] '\n#pragma error "The ') Export Perform substitutions on file(s) with extension(s) *.c;*.cpp;*.h;*.hpp;*.asm;*.s Close | Ena Subst | ble O T T T btution rules (the rules will b From (\h+ ^)code(\s+ * \)) (\h+ ^)(b) (\h+ ^)cdata(\s+ * \)) (\h+ ^)(b) | ag changed code with comm e applied as ordered in the I To \$1_code\$2 \$1_data\$2 tidata\$2 | ist) | | Add Copy |
| 6 (\h+ ^`)pdata(\$+ `\)) \$1_pdata\$2 Import 7 ((\n'+ ^`)pdata(\$+) #pragma error "Use o Export 8 ^((\h+ (\int_{Ca-ZA-20-9}) \n#pragma error "The \n") Export Perform substitutions on file(s) with extension(s) *.c;*.cpp;*.h;*.hpp;*.asm;*.s Execute Close | Ena Subst | ble I T titution rules (the rules will b From (\h+ ^)code(\s+ * \)) (\h+ ^)data(\s+ * \) (\h+ ^)idata(\s+ * \) (\h+ ^)idata(\s+ * \) (\h+ ^)idata(\s+ * \) | ag changed code with comm e applied as ordered in the I To \$1code\$2 \$1data\$2 \$1idata\$2 \$1idata\$2 \$1idata\$2 | ist) | • | Add Copy Edi <u>t</u> |
| 7 ((^(.*\h+)))far(\s+ #pragma error "Use o 8 ^((\\h+'([fa-zA-Z0-9] \n#pragma error "The \ny") Perform substitutions on file(s) with extension(s) *.c;*.cpp;*.h;*.hpp;*.asm;*.s *.c;*.cpp;*.h;*.hpp;*.asm;*.s \ny" | Ena Subst | ble | ag changed code with comm e applied as ordered in the I To \$1_code\$2 \$1_data\$2 \$1_idata\$2 \$1_idata\$2 \$1_idata\$2 \$1_bidata\$2 | ist) | * | Add Copy Edi <u>t</u> Remove |
| 8 ^((!\n*((fa-zA-Z0-9 1 \n#pragma error "The Y Export Perform substitutions on file(s) with extension(s) *.c;*.cpp;*.h;*.hpp;*.asm;*.s Execute Close | Ena Subst Nr 1 2 3 4 5 6 | ble @ ∑ T titution rules (the rules will b From (\h+ ^)code(\s+ * \)) (\h+ ^)data(\s+ * \)) (\h+ ^)data(\s+ * \)) (\h+ ^)bdata(\s+ * \)) (\h+ ^)bdata(\s+ * \)) | ag changed code with comm e applied as ordered in the I To \$1code\$2 \$1data\$2 \$1idata\$2 \$1tdata\$2 \$1bdata\$2 \$1bdata\$2 | ist) | * | Add Copy Edi <u>t</u> Remove |
| Perform substitutions on file(s) with extension(s) *.c;*.cpp;*.h;*.hpp;*.asm;*.s Execute | Ena Subst Nr 1 2 3 4 5 6 7 | $ \begin{array}{c} \hline core abdotation \\ \hline core abdotation $ | ag changed code with comm e applied as ordered in the I To \$1code\$2 \$1data\$2 \$1idata\$2 \$1xidata\$2 \$1bdata\$2 \$1pdata\$2 #pragma error "Use o | ist) | * * | Add Copy Edi <u>t</u> Remove Import |
| *.c;*.cpp;*.h;*.hpp;*.asm;*.s | Ena Subst Nr 1 2 3 4 5 6 7 8 | ble Image: Transform tibution rules (the rules will b From (\h+ ^)code(\s+ * \)) (\h+ ^)data(\s+ * \)) (\h+ ^)pdata(\s+ * \)) (\h+ ^)pdata(\s+ * \)) (\h+(\h)pdata(\s+ * \)) (\h+(\h)pdata(\s+ * \)) (\h+(\h)pdata(\s+1)*]) | ag changed code with comm e applied as ordered in the I To \$1code\$2 \$1data\$2 \$1idata\$2 \$1tdata\$2 \$1tdata\$2 \$1bdata\$2 \$1pdata\$2 #pragma error "Use o \n#pragma error "The | ist) | • | Add Copy Edi <u>t</u> Remove Import Export |
| <u>Execute</u> Close | ✓ Ena Subst Nr 1 2 3 4 5 6 7 8 Perfo | ble @ | ag changed code with comm e applied as ordered in the I To \$1code\$2 \$1data\$2 \$1data\$2 \$1tdata\$2 \$1tdata\$2 \$1bdata\$2 \$1pdata\$2 #pragma error "Use o \n#pragma error "The ith extension(s) | ist) | • | Add Copy Edi <u>t</u> Remove Import Export |
| <u>E</u> xecute Close | Ena Subst Nr 1 2 3 4 5 6 7 8 Perfc *.c; | ble @ ∑ T titution rules (the rules will b From (\h+ ^)code(\s+ * \)) (\h+ ^)data(\s+ * \)) (\h+ ^)data(\s+ * \)) (\h+ ^)data(\s+ * \)) (\h+ ^)data(\s+ * \)) (\h+ ^)pdata(\s+ * \)) (\h+ ^)pdata(\s+ * \)) (\h+ ^)pdata(\s+ * \)) (\h+ ^)pdata(\s+ * \)) (\h+ ^)pfar(\s+ ^((\h*(([a-2A-Z0-9] | ag changed code with comm e applied as ordered in the I To \$1code\$2 \$1data\$2 \$1data\$2 \$1tdata\$2 \$1tdata\$2 \$1bdata\$2 \$1pdata\$2 #pragma error "Use o \n#pragma error "The th extension(s) | ist) | * | Add Copy Edi <u>t</u> Remove Import Export |
| <u>E</u> xecute Close | Ena Subst Nr 1 2 3 4 5 6 7 8 Perfc *.c; | ble Image: T titution rules (the rules will b From (h+1^>code(\s+ * \)) (h+1^>)data(\s+ * \)) (h+1^>)codata(\s+ * \) (h+1^>)codata(\s+ * \s+ | ag changed code with comm e applied as ordered in the I To \$1_code\$2 \$1_data\$2 \$1_idata\$2 \$1_idata\$2 \$1_bdata\$2 \$1_bdata\$2 \$1_bdata\$2 \$1_pdata\$2 #pragma error "De o yn#pragma error "The ith extension(s) | ist) | • | Add Copy Edi <u>t</u> Remove Import Export |
| | Ena Subst Nr 1 2 3 4 5 6 7 8 Perfc *.c; | ble | ag changed code with comm e applied as ordered in the I To \$1_code\$2 \$1_data\$2 \$1_idata\$2 \$1_bdata\$2 \$1_bdata\$2 \$1_bdata\$2 \$1_pdata\$2 \$1_pdata\$2 #pragma error "De o yn#pragma error "The ith extension(s) | ist) | • | Add Copy Edi <u>t</u> Remove Import Export |

Basic code differences

The following sections show some of the basic differences between code written for the Keil toolchain and IAR Embedded Workbench that you should handle before building your converted project.

Initialization code

In IAR Embedded Workbench, initialization code is primarily located in the file cstartup.s51.

This file contains system startup code executed after reset, but before the main() function is called. Data/segment initialization, stack pointer initialization and other things are performed here. This code is part of the runtime library but can be overridden by including a copy of this assembler file in your project. You find the file in the folder 8051\src\lib in the IAR Embedded Workbench installation.

The function int __low_level_init (void) is called from cstartup.s51. Its purpose is to perform any hardware initialization required before segment initialization and calling main (). You may include your own version of this function in your project by adding a copy of the file low_level_init.c, located in the folder 8051\src\lib, and edit it according to you needs.

Special Function Registers

Each device supported by IAR Embedded Workbench has its own header file that contains variable definitions to access the SFRs of the device, both for C/C++ and assembler. The naming convention for these header files is iodevice-name.h and they are located in 8051\inc.

Example: ioEFM8SB20F64G.h

The names of SFR variables might sometimes differ from the names used in the Keil toolchain. SFR bit access differs because IAR Embedded Workbench does not support the sbit keyword which is commonly used in the Keil toolchain for the purpose of defining variables to access specific SFR bits. Instead, a C struct where each member represents one or more bits is used for this purpose. This struct is named *SFR-name_bit*, where *SFR-name* is the name of the corresponding SFR variable represented as a byte.

Sometimes several SFR bits, which are semantically related within the same SFR, are represented by separate sbit variables in the Keil toolchain while they can be combined into a single bit-field in IAR Embedded Workbench. This can make it necessary to rewrite source code which manipulates such bits or define your own variables to access the bits separately.

The <u>sfr</u> memory type attribute keyword is used in IAR Embedded Workbench to define/declare SFR variables, for example as follows:

__sfr __no_init volatile unsigned char TLO @ 0x8A;

To define a variable which represents an SFR which consists of two bytes there is no special keyword like sfr16 which is used in the Keil toolchain. The same keyword is used but the variable has the type unsigned short instead of unsigned char:

sfr no init volatile unsigned short ADCO @ 0xBD;

The same restrictions as in the Keil toolchain apply when using this type of variable, the low byte should immediately precede the high byte address wise and the low byte is the address of the variable. There is one difference though, when writing to the variable Keil writes the most significant byte first and the least significant byte last while IAR Embedded Workbench writes them in the opposite order. Generally, the writing order of SFRs can affect the hardware behavior so care must be taken when utilizing multi-byte SFRs.

Interrupt Service Routines

The interrupt keyword is used in the Keil toolchain when defining an interrupt function (ISR). In IAR Embedded Workbench, the extended keyword ___interrupt is used for the same purpose. The interrupt keyword has a numeric parameter which the Keil compiler automatically translates into an interrupt vector number. IAR Embedded Workbench requires the application developer to specify the vector number directly using the vector pragma directive. For convenience, the device-specific header file containing the SFR access variables also contain defined names for the interrupt vectors. Here is an example comparing the syntax used in the two toolchains:

Migrating from Keil µVision for 8051 to IAR Embedded Workbench for 8051

| Keil toolchain | IAR Embedded Workbench |
|--|-----------------------------------|
| | <pre>#pragma vector=0x0B</pre> |
| <pre>void timer0_isr(void) interrupt 1 {</pre> | interrupt void timer0_isr(void) { |
| ••• | |
| } | } |

It is also possible to specify a register bank which will be used by an interrupt function. In the Keil toolchain, this can be done for any function, but in IAR Embedded Workbench it is only applicable to interrupt functions. The using keyword is used in the Keil toolchain whereas the register_bank pragma directive is used in IAR Embedded Workbench. Here is an example:

| Keil toolchain | IAR Embedded Workbench |
|--|------------------------------------|
| | <pre>#pragma register_bank=2</pre> |
| | #pragma vector=0x0B |
| <pre>void timer0_isr(void) interrupt 1 using 2</pre> | interrupt void timer0_isr(void) |
| { | { |
| ••• | ••• |
| } | } |

Inline assembler

Inline assembler is handled somewhat differently in the Keil toolchain than in IAR Embedded Workbench. The Keil toolchain provides mechanisms to insert assembler code only if configured to generate a separate assembler source file (.src) which is then assembled to object code. IAR Embedded Workbench integrates the assembler code directly into the object code when compiling, but of course it is possible to generate a separate assembler source file (.s51) if desired.

The mechanism used in the Keil toolchain are the pragma directives ASM and ENDASM (or the alternative keyword __asm). There are no corresponding pragmas in IAR Embedded Workbench, the asm keyword (or its alias __asm) is used instead. Note that __asm in the Keil toolchain does not have the same syntax or semantics as __asm in IAR Embedded Workbench. Here is an example of two equivalent code sequences:

| Keil toolchain | IAR Embedded Workbench |
|----------------|------------------------|
| #pragma ASM | |
| MOV A, #33H | asm("MOV A, #0x33\n" |
| MOV R1, #22H | "MOV R1, #0x22\n" |
| ADD A, R1 | "ADD A, R1\n" |
| JMP \$ | "JMP \$"); |
| #pragma ENDASM | |

Managing memory

The *memory models* used in the Keil toolchain – which can be applied to functions, constants and variables – correspond to the concepts of *code model* and *data model* in IAR Embedded Workbench. The code and data models are set per project and specify the default storage model for functions and data respectively. The code model can be overridden for individual functions using *function memory attribute* extended keywords in the source code. The data model can be overridden for individual data objects or pointers by specifying *data memory attribute* extended keywords in the source code. The data memory attributes correspond to *memory type specifiers* in the Keil toolchain.

Data with an integer type represented by more than one byte is stored as big endian in the Keil toolchain. In IAR Embedded Workbench such data is stored as little endian. This fact might require source code to be rewritten to work as expected.

Many library functions declared in string.h (such as memcpy, memcmp, strcat etc.) cannot handle parameters which are located in different memory types. This might cause unexpected results.

Bit variables

Declaring a single-bit variable (which will be located in the bit-addressable area of the internal RAM) is done in the Keil toolchain using the type *bit*. In IAR Embedded Workbench, this is accomplished by declaring a variable which has the type *bool* in combination with the __*bit* data memory attribute. A declaration using the __*bit* attribute has limitations; it must be located outside the function scope, and the object attribute __*no_init* must also be used which means that such a variable cannot be initialized at the same time it is declared. If, in code written for the Keil toolchain, a variable declared with the *bit* keyword resides within a function, a function argument list, or if a function returns a value of type *bit*, a supported type such as *bool* should be used in IAR Embedded Workbench.

Variables previously declared using the type *bit* have another type in IAR Embedded Workbench. Therefore they are no longer bit-oriented and references to such variables should be reviewed carefully because bit-oriented operators such as \sim , $^$, & and | will most probably not have the same effect after the change of type.

Building your project

After successfully converting the Keil μ Vision project and considered the basic code differences described above, you will still most likely need to fine-tune parts of the source code so that it follows the IAR Embedded Workbench syntax.

- 1. Select your device under Project>Options>General Options.
- 2. Choose Project>Make.
- 3. To find the different errors/warnings, press F4 (Next Error/Tag).
 - This will bring you to the location in the source code that generated this error/warning.
- 4. For each error/warning, modify the source code to match the IAR Embedded Workbench syntax. Note: You might have to consult the <u>IAR C/C++ Compiler User Guide</u> for this step.
- 5. After correcting one or more errors/warnings, repeat the procedure.

Note: It is always a good idea to start by correcting the first couple of errors/warnings in different source files because errors and warnings later in the source code might just be effects of faulty syntax at the beginning of the source code.

Important tool settings

This is an overview of the most important tool settings. Note that many settings do not have a one-to-one mapping. An example is the memory segment configuration which is available in Keil μ Vision through the Options dialog box. In IAR Embedded Workbench segment configuration is all done in the linker configuration file which can be pointed out in the **Options** dialog box.



Migrating from Keil $\mu Vision$ for 8051 to IAR Embedded Workbench for 8051

| Device Torgetti Output I Listing User C51 A51 BL51 Locate BL51 Misc Debug Utities Sticon Laboratories. Inc. 2005 IF015 Xal (MHz) 20.0 Wency Model: Sinal: vanables in DATA Code Rom Size: Large: 64K program Operating system: None Off-chip Code memory Statt: Statt: Size: Eprom Fam Eprom Ram Eprom Ram Eprom Ram Ram Ram Ram Ram Banks: Saxt: Dot Defaults OK Cancel Defaults Heip | Options for Target Simulator Device Target Output Listing User C51 A51 BL51 Locate BL51 Misc Debug Utilities Silicon Laboratories, Inc. C8051F015 Yall (MH2): 20.0 Memory Model: Small: variables in DATA Code Rom Size: Large: 64K program Off-chip Code memory Statt: Size: Eprom End: Statt: Statt: Statt: Statt: Statt: Statt: Statt: Eprom Ram Ram Statt: Statt: | Options for node "project1" Calegony: Careeral Options Static Analysis C/C++ Compler Assembler Custom Build Build Actions Linker Debugger Terget Stack/Heap Data Pointer Code Bank Output Library Cor 4 > Device information Device: C8051F015 Cpde model Near Use extended stack Dgta model Caling convention Number of virtual registers: By Alm memory RoM mapped as data CODE memory Use MDU DK |
|--|---|---|
|--|---|---|

Migrating from Keil $\mu Vision$ for 8051 to IAR Embedded Workbench for 8051

| Keil µVision | IAR Embedded Workbench | | |
|--|---|--|--|
| Output type | | | |
| V Options for Target 'Simulator' | Options for node "project1" | | |
| Options for Target 'Simulator' Device Target Oxtput Listing User C51 A51 BL51 Locate BL51 Mac Debug Utities Select Folder for Qbjects Name of Executable: BLINKY If Qebug Information If Brogge Information If Create HEX File HEX Format: HEX-80 If Create HEX File MEX Format: If Executable: If Create Batch File OK Cancel Defaults Help | Options for node "project1" Category: General Options Static Analysis C/C++ Compiler Asembler Custom Buid Buid Actions Linker Debugger Third Party Driver F52 System Navig Infineon Segger J-Link Nordic Semiconduc ROM-Montor Analog Devices Sibals Simulator | | |
| | OK Cancel | | |
| Output format | | | |
| Options for Target 'Simulator' Device Target Oxtput Listing User C51 A51 BL51 Locate BL51 Misc Debug Utilites Select Folder for Objects Name of Executable: BLINKY © Create Executable: \vBLINKY Ø Debug Information © Create HEX File HEX Format. HEX-80 C Create Library: \vBLINKY/LIB OK Cancel OK Cancel | Options for node "project1" Calegony: General Options Static Analysis C/C++ Compiler Assembler Custom Build Build Actions Infree Debugger Trict Debugger Three Seger J-Link ROM-Montor Analog Devices Simulator With // Oe mation modules Format variagt: None Module-local symbols: Include all Module-local symbols: Include all | | |
| Compiler options | | | |
| Options for Target 'Simulator' Device Target Output Listing User Ugdefine: Ugdefine: Ugdefine: Level: 8: Reuse Common Entry Code Implasis: Favor speed Implasis: | Options for node "project1" Category: General Options Static Analysis C/C++ Compler Assembler Custom Build Build Actions Linker Debugger Third-Party Driver FS2 System Navig Infineon Segger 3-Link Nordic Semiconduc ROM-Monitor Analog Devices Stabs Simulator | | |
| | | | |

Migrating from Keil µVision for 8051 to IAR Embedded Workbench for 8051

| Keil µVision | IAR Embedded Workbench | | |
|--|---|--|--|
| Defined symbols and include directories | | | |
| Options for Target 'Simulator' | Options for node "project1" | | |
| Device Target Output Listing User CST A51 BL51 Locate BL51 Misc Debug Utilities Preprocessor Symbols | Category: Factory Settings General Options Discard Unused Publics State Analysis Discard Unused Publics Assembler Cutoput Assembler Discard Unused Publics Debugger Third-Party Driver Third-Party Driver Additional include directories: (one per line) Segary 3-Jink Preinclude file: Nordic Semiconduct Defined symbols: (one per line) Silabs Simulator | | |
| Linker options Options for Target 'Simulator' Device Target Output Listing User C51 A51 BL51 Locate Device Target Output Listing User C51 A51 BL51 Locate Device Target Output Listing User C51 A51 BL51 Locate Device Target Output Listing User C51 A51 BL51 Locate Device Target Output Listing User C51 A51 BL51 Locate Device Target Output Listing User C51 A51 BL51 Locate Code Range: | Options for node "project1" X Category: Factory Settings State Analysis C/C++ Compler C/C++ Compler Corfig Output, Extra Output, List Log #define Diagnostics (*) Debugger Unker Unker configuration file Debugger Unker configuration file Unker configuration file Debugger Trick-Party Driver Stool KUT_DIRS'config'devices \Silcon Labs\Ink51ew_C805 RotM-Monitor Analog Devices StoOLKUT_DIRS LiB\ Image Stool Kontor StoOLKUT_DIRS LiB\ Image Image Baw binary image StoOLKUT_DIRS LiB\ Image Image Baw binary image Symbol: Segnent: Align: Image Image Image | | |

Note: We recommend that you verify all settings to make sure they match your project requirements.

IAR Systems, IAR Embedded Workbench, IAR Connect, C-SPY, C-RUN, C-STAT, IAR Visual State, visualSTATE, IAR KickStart Kit, I-jet, I-jet Trace, I-scope, IAR Academy, IAR, and the logotype of IAR Systems are trademarks or registered trademarks owned by IAR Systems AB. All other product names are trademarks of their respective owners.

All information is subject to change without notice. IAR Systems assumes no responsibility for errors and shall not be liable for any damage or expenses.

© 2018 IAR Systems AB. Part number: EW8051_MigratingFromKeil-5. Fifth edition: Mars 2018