Customer Notification

EW78K

Embedded Workbench® for 78K

Operating Precautions

Y-IAR-EW78K-FULL-MOBILE

Y-IAR-EW78K-FULL

Renesas Electronics
www.renesas.com

Document No. R20UT0002ED0737
Date Published: May 2015

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.

2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.

4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.

5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.

6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.

7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti- crime systems; safety equipment; and medical equipment not specifically designed for life support.
   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.

8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.

9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.

10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority- owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

Table of Contents

A) Table of Operating Precautions for the IDE EW78K

No.	Outline	EW78K						
	Version	4.8a	5.2d	5.5.0	6.0.3	6.0.3.2	6.06.1	6.46.2
A2	An empty Workspace can not be saved	×	×	×	×	×	×	×
A10	Usage of Soft-Links in output path definition could cause the IDE to link two copies of the output files in the Workspace Windows	×	×	✓	✓	✓	✓	✓
A11	78K0R: Project settings for near-constant-location are not saved.	×	×	✓	✓	✓	✓	✓
A12	Heap size input value is limited to 64KB	×	×	✓	✓	✓	✓	✓
A13	Linker output file in format IEEE695 can not be generated	×	×	✓	✓	✓	✓	✓
A14	Empty Go to Function Window	×	×	×	×	×	×	✓
A15	Corrupted Default-File Filter	✓	×	✓	✓	✓	✓	✓
A16	IDE crashes if illegal Values defined for 78K0R Mirror Area	×	×	×	✓	✓	✓	✓
A17	MISRA C checker can not be enabled in EW78K Dialogue	✓	✓	✓	×	✓	✓	✓
A18	Actual Linker-MAP-File not automatically updated in Editor	-	-	-	×	×	×	×

B) Table of Operating Precautions for the Assembler A78K

No.	Outline	A78K					
	Version	4.60a	4.61a	4.62.1	4.70.1	4.71.1	4.80.1
B1	RSEG Directives can not be used in Macro Definitions	×	×	×	×	×	×
B2	Assembler File must contain at least one Directive	-	-	-	×	×	×

×: Applicable ✓: Not applicable - : Not checked

C) Table of Operating Precautions for C/C++ Compiler ICC78K

No.	Outline	ICC78K					
	Version	4.62.5	4.70.1	4.71.1	4.80.1	4.80.2	4.80.3
C5	No compiler message in case of a variable redefinition of the same data type but with the different object attribute	×	×	×	×	×	×
C66	Wrong Code generated for if condition resulting in single bit test	×	✓	✓	✓	✓	✓
C67	Default case is not executed if switch variable is larger than 0xFFFF	×	×	✓	✓	✓	✓
C68	Internal Compiler Error due to non terminated Jump Size Optimization	✓	×	✓	✓	✓	✓
C69	Internal Compiler Error at using intrinsic function '`__segment_begin`'	✓	×	✓	✓	✓	✓
C71	Internal Compiler Error using bit test and branch instruction	✓	×	✓	✓	✓	✓
C72	Wrong Code generated for storing variable to stack after Function Call	×	×	✓	✓	✓	✓
C73	Internal Compiler Error at Negation of Bitfield-Element	×	×	×	✓	✓	✓
C74	#pragma location Directive does not support Unions and Structures	×	×	×	×	×	×
C75	Wrong Code generated for Pointer Array Index	×	×	×	✓	✓	✓
C76	Internal Compiler Error while using __segment_size as memcpy Parameter	×	✓	✓	✓	✓	✓
C77	Bit Access generated although Keyword '__no_bit_access' was used	×	×	×	×	✓	✓
C78	Unclear Description of Parameter Passing for Structure Types in Compiler Manual	×	×	×	×	✓	✓
C79	Wrong Code generated causing an unreachable else Path	×	✓	✓	✓	✓	✓
C80	No Code generated for if Condition	×	×	×	×	×	✓
C81	MISRA C 2004 Rule 10.6 not triggered	×	×	×	×	×	×
C82	Wrong Code generated for Array Index	✓	×	×	×	×	✓

×: Applicable ✓: Not applicable - : Not checked

D) Table of Operating Precautions for the Linker XLINK

No.	Outline / Version	XLINK 5.00.1 5.00.2	5.10.8	5.2.6.1 9	5.3.1.26	5.4.1.30	5.6.0.36	6.0.3.49	6.2.2.68
D3	Breakpoint cannot be defined in Function (only XCOFF78K Format)	×	×	×	×	×	×	×	×
D29	Output file format UBROFF: Internal Linker Error 1	✓	✓	✓	✓	✓	✓	✓	✓
D30	Output file format UBROFF: Internal Linker Error 2	✓	✓	✓	✓	✓	✓	✓	✓
D31	Output file format ELF/DWARF: Error[e113]: Corrupt input file: "Illegal ELF-register."	✓	✓	×	✓	✓	✓	✓	✓
D32	ELF/DWARF Format: Wrong Return Type Entry	×	×	×	✓	✓	✓	✓	✓
D33	Definition of Segment Area Size ‘0’ causes Internal Linker Error	-	×	×	×	×	×	✓	✓
D34	Erroneously Error e16 ‘Segment too long’ is generated	-	-	-	-	✓	×	×	×

×: Applicable ✓: Not applicable - : Not checked

E) Table of Operating Precautions for C-SPY Debugger CS78K

No.	Outline / Version	CS78K 4.60a	4.60b	4.62.1	4.70.1	4.71.1	4.71.2	4.80.1	4.80.3
E34	If the same name is used for a data-object and for a data-type, this data-object can not be displayed in the Watch Window	✓	✓	✓	×	×	×	×	×
E43	C-SPY 78K0R Simulator Driver: Interrupt simulation only works correct at priority level three.	×	×	✓	✓	✓	✓	✓	✓
E44	C-SPY 78K0 MINICUBE2 Driver: Error message about old firmware version	✓	×	✓	✓	✓	✓	✓	✓
E45	C-SPY all Drivers: Update Time Watch Window	×	×	✓	✓	✓	✓	✓	✓
E46	C-SPY Simulator Driver: Incorrect Value shown in Live-Watch Window	×	×	✓	✓	✓	✓	✓	✓
E47	C-SPY 78K0 MINICUBE Driver: Incorrect System Clock Selection	×	×	✓	✓	✓	✓	✓	✓
E48	Incorrect Variable Address may be displayed in Event Window or Watch Window	×	×	✓	✓	✓	✓	✓	✓
E49	Stack Initialization in default cstartup-module triggers C-SPY Debugger stack observation	×	×	✓	✓	✓	✓	✓	✓
E50	Wrong display of array in C-SPY Watch Window	×	×	✓	✓	✓	✓	✓	✓
E51	C-SPY 78K Simulator Driver: Wrong macro access to 16bit data	×	×	✓	✓	✓	✓	✓	✓
E52	C-SPY 78K: Displayed floating point value in watch window may be wrong	×	×	✓	✓	✓	✓	✓	✓
E53	C-SPY 78K: Resetting a running application causes stack warning message	✓	✓	×	✓	✓	✓	✓	✓
E54	C-SPY 78K: Breakpoint can not be defined at some source lines	✓	✓	×	✓	✓	✓	✓	✓
E55	C-SPY 78K0R: Wrong Display of 16bit SFR in Memory Window	×	×	×	×	✓	✓	✓	✓
E56	C-SPY 78K0R IECUBE Driver: Inaccurate Time Measurement Result	×	×	×	×	✓	✓	✓	✓
E57	C-SPY all Drivers: Program Counter may be uninitialized	×	×	×	×	✓	✓	✓	✓

No.	Outline	CS78K							
	Version	4.60a	4.60b	4.62.1	4.70.1	4.71.1	4.71.2	4.80.1	4.80.3
E58	C-SPY 78K0R MINICUBE2 Driver: Broken Emulator Communication	×	×	×	×	✓	✓	✓	✓
E59	All C-SPY Drivers except Simulator Driver: System Macro __driverType not implemented	×	×	×	×	×	✓	✓	✓
E60	All C-SPY Drivers: Incorrect Flash Memory Upload in Run-Mode	×	×	×	×	×	×	✓	✓
E61	ORTI Plug in Error Message „Memory Exhausted”	-	-	×	×	×	×	✓	✓
E62	Constant Data Object located in Data Flash Area displayed incorrectly in Watch Window	-	-	×	×	×	×	✓	✓
E63	Reading Data-Flash-Memory causes an Error	-	-	×	×	×	×	×	✓

×: Applicable ✓: Not applicable - : Not checked

F) Table of Operating Precautions for the Assembler A78K0R

No.	Outline	A78K0R					
	Version	4.61a	4.62.1	4.70.1	4.71.1	4.80.1	4.80.2
F1	RSEG Directives can not be used in Macro Definitions	×	×	×	×	×	×
F11	Illegal indirect MOVW instruction is accepted and wrong Op-Code is generated	×	✓	✓	✓	✓	✓
F12	Illegal Op-Code generated if SFR symbol is defined after the usage	×	✓	✓	✓	✓	✓
F13	Directive DS64 is not implemented	×	×	×	×	✓	✓
F14	Wrong Code Generated for Bit Test Instructions	-	×	×	×	×	✓

×: Applicable ✓: Not applicable - : Not checked

G) Table of Operating Precautions for C/C++ Compiler ICC78K0R

No.	Outline	ICC78K0R					
	Version	4.70.1	4.71.1	4.71.2	4.80.1	4.80.2	4.80.3
G36	Internal Compiler Error due to non terminated Jump Size Optimization	×	✓	✓	✓	✓	✓
G37	Internal Compiler Error at using intrinsic function '__segment_begin'	×	✓	✓	✓	✓	✓
G38	Wrong Code generated for far Branch Inline-Assembler Instruction	×	✓	✓	✓	✓	✓
G39	Inline Assembler Range Error Message triggered by Mistake	×	✓	✓	✓	✓	✓
G41	Internal Compiler Error at far pointer access to I/O area	×	✓	✓	✓	✓	✓
G42	Internal Compiler Error at calling strcpy or memcpy in far data model	×	✓	✓	✓	✓	✓
G43	Wrong Pointer Access to Special-Function-Register in Data Model 'far'	×	✓	✓	✓	✓	✓
G44	Error Message Pe028 Triggered by Mistake	×	×	×	✓	✓	✓
G45	Internal Compiler Error: Casting SADDR Address into far Pointer	×	✓	✓	✓	✓	✓
G46	Error in Device Specific Header File	✓	×	✓	✓	✓	✓
G47	Internal Compiler Error: EctContextBase::GetValue	✓	×	✓	✓	✓	✓
G48	Wrong Offset Address Calculation	×	×	✓	✓	✓	✓
G49	Internal Compiler Error CoreUtil/General at using MISRA C and Option –header_context	×	×	×	✓	✓	✓
G50	Far Pointer defined instead of near Pointer	×	×	×	✓	✓	✓
G51	Internal Compiler Error at Negation of Bitfield-Element	×	×	×	✓	✓	✓
G52	Internal Compiler Error at Macro Expansion	×	×	×	✓	✓	✓
G53	Internal Compiler Error at Returning a negated right-shifted Value	×	×	×	✓	✓	✓
G54	Internal Compiler Error at Returning a Comparison Result	×	×	×	✓	✓	✓
G55	Wrong Code generated for Function Call directly after memcpy-Function call	×	×	×	✓	✓	✓
G56	Compilation process stalls	✓	×	×	✓	✓	✓
G57	Wrong Code could be generated for Near Pointer Indexing	×	×	×	✓	✓	✓
G58	#pragma location Directive does not support Unions and Structs	×	×	×	×	×	×
G59	Internal Compiler Error while using __segment_size as memcpy Parameter	×	×	×	✓	✓	✓
G61	Wrong Code generated for Bit Negation of 32bit Bitfield	×	×	×	✓	✓	✓
G62	CPU Cycle Information of CALLT Instruction missing in Compiler-List-File	✓	×	×	×	×	×

No.	Outline	ICC78K0R					
	Version	4.70.1	4.71.1	4.71.2	4.80.1	4.80.2	4.80.3
G63	Wong Code generated at far-Pointer Arithmetic	×	×	×	×	✓	✓
G64	Bit Access generated although Keyword '__no_bit_access' was used	×	×	×	×	✓	✓
G65	Wrong indirect post Increment of a Result of a post Increment	×	×	×	×	×	✓
G66	Unclear Description of Parameter Passing for Structure Types in Compiler Manual	×	×	×	×	✓	✓
G67	Internal Error in case of similar Function in 'switch' and 'if' Node	×	×	×	×	×	✓
G68	Unnecessary Padding Byte added to Arrays of Character	-	×	×	×	×	✓
G70	Wrong Code generated while Copying a 1-Bit Bitfield	×	×	×	×	✓	✓
G71	MISRA C 2004 Rule 10.6 not triggered	×	×	×	×	×	×
G72	Stack Content can be corrupted by ISR	×	×	×	×	×	✓
G73	Wrong Code generated for Array Index	×	×	×	×	×	✓

×: Applicable ✓: Not applicable - : Not checked

H) Description of Operating Precautions for the IDE EW78K

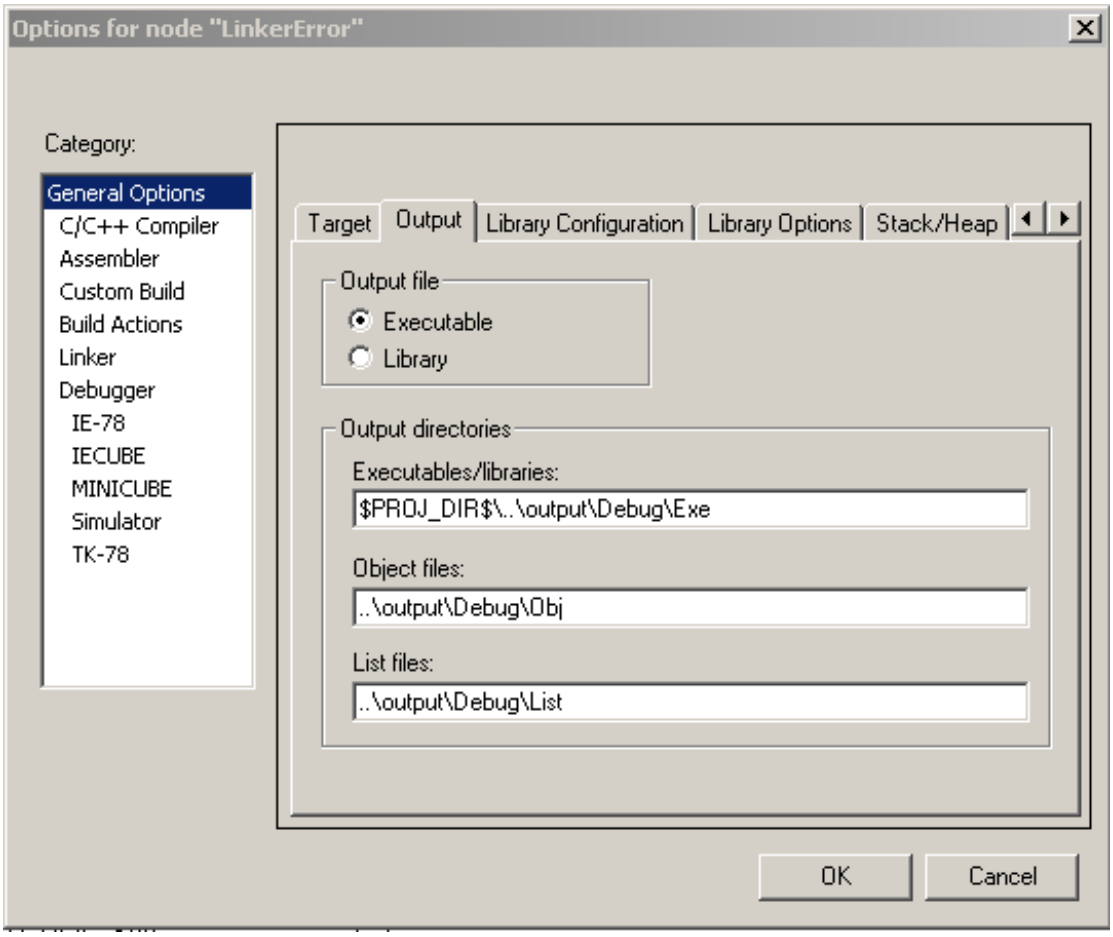
No. *A*2	**An empty workspace can not be saved**
	Details Although it is described in the user's manual an empty workspace can not be saved. *Workaround* Add at least one project to the workspace before saving. The project may be an empty project.

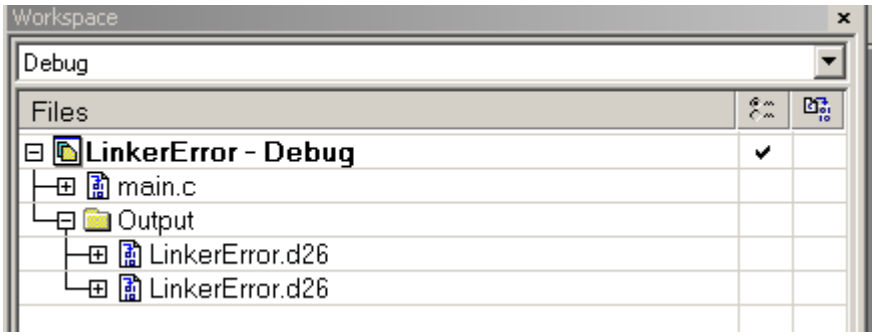
No. A10	**Usage of Soft-Links in output path definition could cause the IDE to link two copies of the output files in the Workspace Windows**

Details

If the IAR System soft-links (e.g. $PROJ_DIR$) are used to define the output file path, the Embedded Workbench may link two copies of the generated output file in the Workspace Window.

Example:





Workaround

Don't use soft-links in the output file path definition? The issue will be changed in next major update of EW78K.

No. A11	**78K0R: Project settings for near-constant-location are not saved**.

Details

The size of the near-constant-location-area is not saved between two Embedded Workbench sessions. Instead, the default values are loaded.

Workaround
If the default setting is modified, please set the new values manually.

No. A12	**Heap size input value is limited to 64KB**

Details

The maximum heap size that can be entered in the Embedded Workbench GUI is 64KB. In case of entering a larger value the following error message is generated:

Workaround
Please specify the heap-size directly in the used linker-control file instead of using the symbol '_HEAP_SIZE' defined in the Embedded Workbench GUI:

```
//-------------------------------------------------------------------
//      Heap segment
//-------------------------------------------------------------------
-Z(DATA)HEAP+0x12000=<start_address>-<end_address>
```

The problem will be fixed in the next EW78K platform update.

No. A13	**Linker output file in format IEEE695 can not be generated**

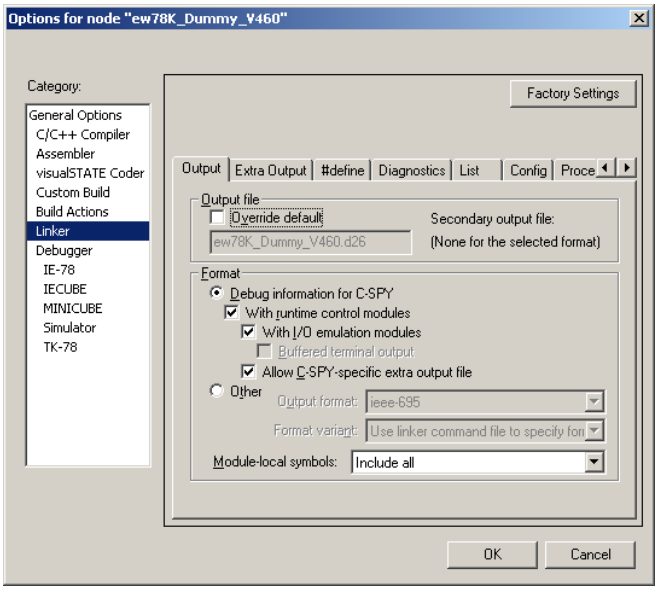
Details

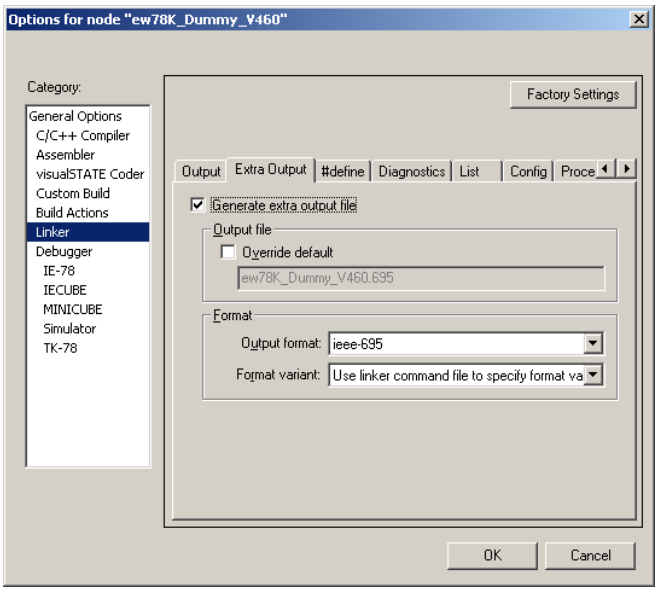
If a 78K0R target device and the linker output file format IEEE695 is selected, no output file is selected and the following error message is generated:

```
Fatal Error[e92]: Cannot use the 'ieee695' output format with this cpu
```

Workaround

Please select another output file format (e.g. C-SPY Debug Format), enable the generation of a second output file, and select the format IEEE695 for the second output file:



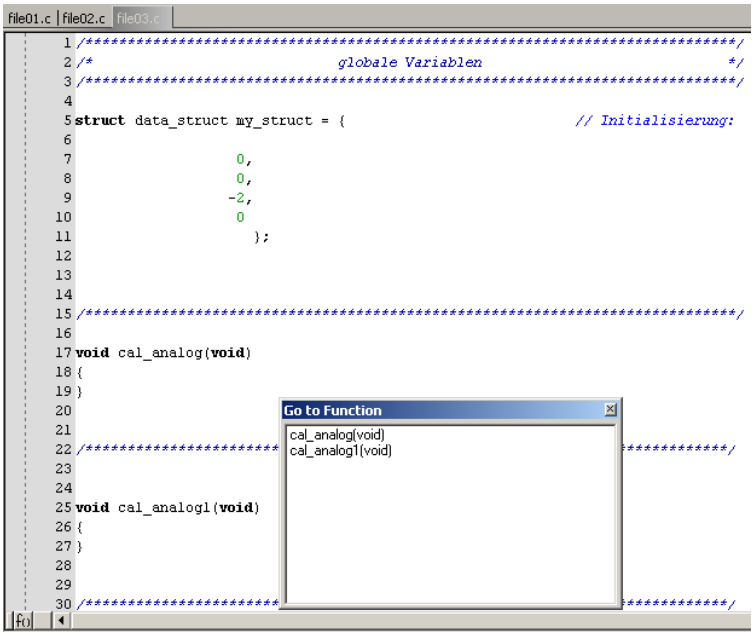


The problem will be fixed in the next EW78K platform update.

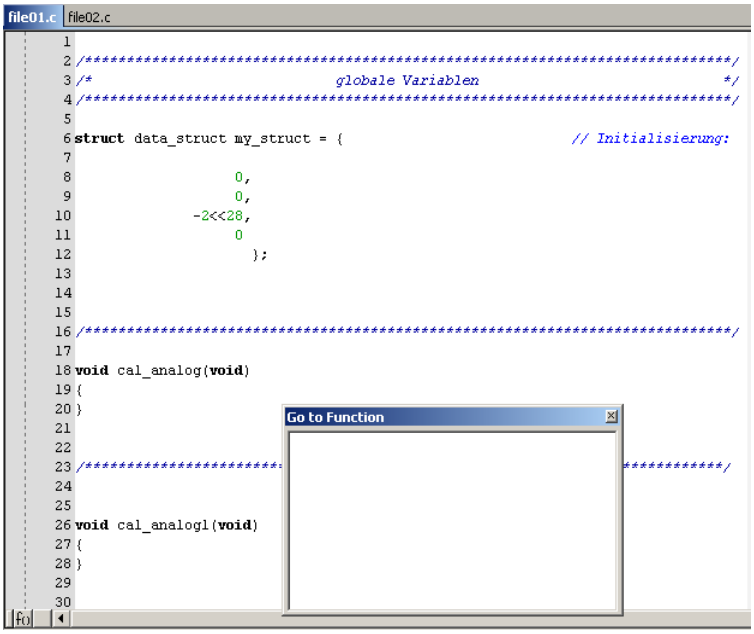
No. A14	**Empty Go to Function Window**

Details

Depending on some source code constructions (e.g. using shift operator to initialize a structure element) the Go to Function Window may be empty.
Correct Go to Function Window:



Empty Go to Function Window although there are several functions defined in the active source file:

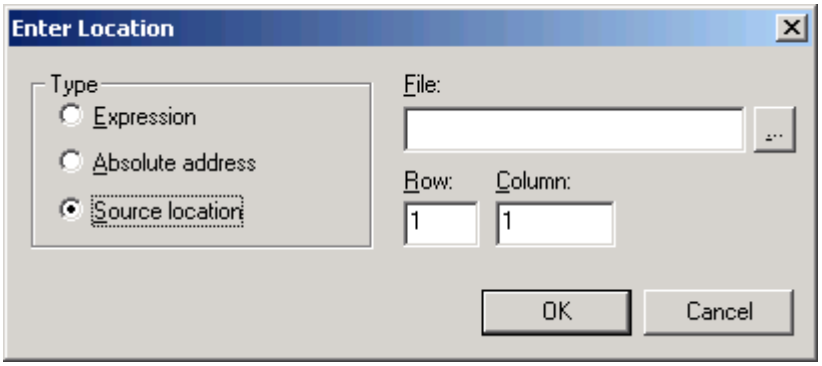


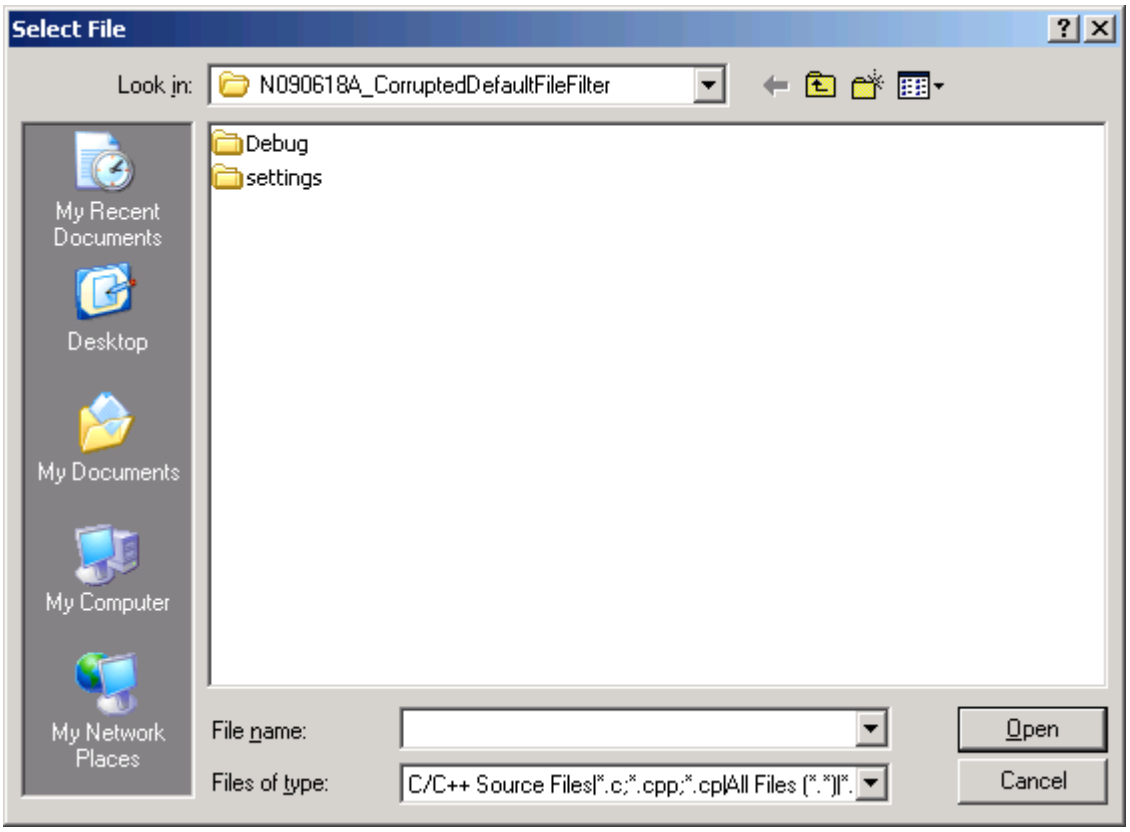
Workaround
None. The problem will be fixed in the next EW78K platform update.

No. A15	**Corrupted Default-File Filter**

Details

The default file filter of the C-SPY file selection dialogue after pressing the button '...' of the code breakpoint 'Enter Location Window' is corrupted and therefore no files are listed although there are source files in the selected folder:





Workaround

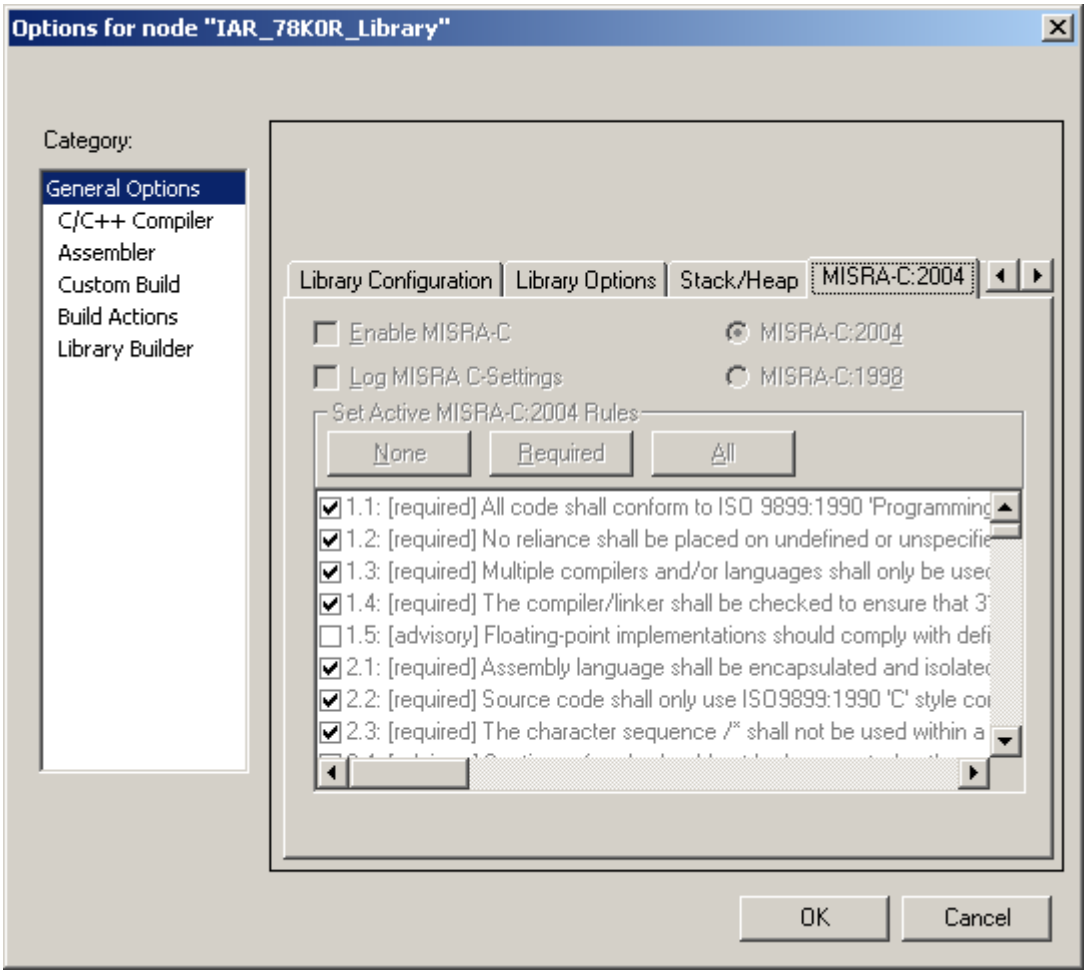
Enter '*.*' as file name to get a list of all available source files and select the file.

No. *A*16	**IDE crashes if illegal Values defined for 78K0R Mirror Area**
	Details The IDE crashes while starting the C-SPY Simulator, when illegal values for start address and size of near constant location in the project options were defined. Example: Start address: 0xFF000 Size: 4.25KB *Workaround* Use only valid values. By default a correct start address and the maximum size are selected. If the complete area isn't used, available space can be used by other segments without modifying the mirror area setting.

No. A17	**MISRA C checker can not be enabled in EW78K Dialogue**

Details

The MISRA C checker can not be enabled in the EW GUI:



Workaround

An update patch can be downloaded from the IAR Systems MyPages area to fix this problem. Further details are described here:

http://supp.iar.com/Updates/?product=EW78K&version=4.70&highlight=1003

As a workaround the compiler can be used by command line interface to use the MISRA C checker.

No. *A18*	**Actual Linker-MAP-File not automatically updated in Editor**
	IAR Reference: EW24451 *Details* Although the option ‘Scan for changed Files’ is enabled in EW tool options, a linker map file in HTML format is not automatically updated. *Workarounds* Use text format or update the file manually.

I) Description of Operating Precautions for the Assembler A78K

No. B1	**RSEG Directives can not be used in Macro Definitions**

Details

The assembler calculates a wrong relative jump-distance if the RSEG directive is used within a macro definition:

Example

```
mDummyMacro MACRO
        RSEG    CODE
        NOP
        ENDM
```

Workaround

Don't use the RSEG directive in macro definitions. The used code-segment must be defined in the code where the macro is expanded to.

No. B2	**Assembler File must contain at least one Directive**

Details

An assembler module without any assembler directive causes the following error message:

```
Error[As073]: Each file must contain at least one directive
```

Example

```
#if PLATFORM == RL78
   ; section without directive
#else
   ; section without directive
#endif
```

Workaround

Please use the END directive:

```
#if PLATFORM == RL78
   ; section code
   END
#else
   ; section code
   END
#endif
```

J) Description of Operating Precautions for the C/C++ Compiler ICC78K

No. C5	**No compiler message in case of a variable redefinition of the same data type but with the different object attribute**

Details

The compiler doesn't generate a message for the user if a variable is redefined with the same data type but with a different object attribute.

Example:

```
unsigned int i;
__no_init unsigned int i;
```

Workaround

Manual check by the user required.

No. C66	**Wrong Code generated for if condition resulting in single bit test**

For an if-condition resulting in a single bit test wrong code may be generated at optimization level medium and higher, if it was followed directly by a clear of the variable tested in both branches.

```
extern unsigned char  func1 (void);
extern unsigned char  func2 (unsigned char);

unsigned char var1;

void test(void)
{
  unsigned char local1     = func1();
  unsigned char mask = func2(local1);

  if(var1 &  0x40)  {
    var1 = 0;
    if     (mask & (0x04))  {
      var1 = 0x04;
    }
  }
  else  {
    var1 = 0;
    if     (mask & (0x01)) {
       var1 = 0x01;
    }
  }
}
```

Workarounds:

Reduce the optimization level for the affected function:

```
#pragma optimize=low
void test(void)
{
   …
}
```

No. C67	**Default case is not executed if switch variable is larger than 0xFFFF**

Due to a problem in the assembler switch routine, switch variable values larger than 0xFFFF does not execute the default case as expected.

```
volatile long lVal = 0x10000;

unsigned char test (void)
{
    unsigned char uchRet;

    switch(lVal)
    {
      case 0l:
      case 1l:
      case 2l:
      case 3l:
       uchRet = 1;
      break;
      case 4l:
      case 5l:
      default:
        uchRet = 0;
        break;
   }
   return uchRet;
}
```

Workarounds:

- the data type of the switch variable to short or smaller
- replace the switch command by if and else commands

The issue will be fixed in the next update (target May 2011)

No. C68	**Internal Compiler Error due to non terminated Jump Size Optimization**

At optimization level high some complex switch statements cause an internal compiler, because the jump size optimization will not terminate:

Workaround:

Reduce the optimization level to medium or low for the function including the switch statement by using #pragma optimization in front of the function definition.

The issue will be fixed in the next update (target May 2011)

No. C69	**Internal Compiler Error at using intrinsic function '__segment_begin'**

At optimization level medium or higher using the intrinsic function '__segment _begin' inside an if-statement or any kind of loop may cause an internal compiler error.

Example:

```
#include <intrinsics.h>
#pragma segment="MYSEG"

extern void func1(unsigned char*);

void test (void* ptr)
{
  if( ptr != ((void*)0) ) {
    func1(__segment_begin("MYSEG"));
  }
}
```

Workarounds:

1)
Reduce the optimization level to low for the function including the if statement by using #pragma optimization in front of the function definition:

```
#pragma optimize=low
void test (void* ptr)
{
  …
}
```

2)
Put the intrinsic function call in a function which is not inlined.

The issue will be fixed in the next update (target May 2011)

No. C71	**Internal Compiler Error using bit test and branch instruction**

Details

A bit test and branch instruction that jumped to the immediate next instruction could cause the compiler to generate an internal error, if optimization level low or none are used.

Example:

```
static __saddr unsigned char locvar;

void foo1(void)
{
    if (locvar & 0x02u)
    {
      …
    } else if (locvar & 0x04u)
    {
      …

    } else if (locvar & 0x08u)
    {
      …
    } else if (locvar & 0x10u)
    {
      …
    }
}
```

Workaround:

Use optimization level medium or higher.

No. C72	**Wrong Code generated for storing variable to stack after Function Call**

Details

Independent of the used memory model storing a variable to stack can generate a store to a wrong location, if the position is on top of stack and the value stored is a return value from a function with parameters.

Example:

char szBuffer[16];

```
void test(void)
{
  char* pszBuffer = szBuffer;
  pszBuffer += sprintf(pszBuffer, "%d ", 1);
}
```

Workaround:

Use a static pointer:

```
void workaround (void)
{
  static char* pszBuffer = szBuffer;
  pszBuffer += sprintf(pszBuffer, "%d ", 1);
}
```

No. C73	**Internal Compiler Error at Negation of Bitfield-Element**

Details

Independent of the selected optimization level an internal compiler error may occur if a negated bitfield element is used as return value:

`Internal Error: [CoreUtil/General]: Stack overflow`

Example

```
struct s{
  int m : 1;
};

int f1(struct s *p){
  return !p->m;
}
```

Workarounds

Use a temporary variable and select optimization level low:

```
int f1(struct s *p){
  unsigned int temp = !p->m;
  return temp;
}
```

No. C74	**#pragma location Directive does not support Unions and Structures**

Details

The #pragma location directive does not support unions and structs. An warning is generated to inform the user:

```
Warning[Pe609]: this kind of pragma may not be used here
```

Example

```
typedef struct
{
    unsigned char no0:1;
    unsigned char no1:1;
    unsigned char no2:1;
    unsigned char no3:1;
    unsigned char no4:1;
    unsigned char no5:1;
    unsigned char no6:1;
    unsigned char no7:1;
} __BITS8;

#pragma location = 0xFF22;
__sfr __no_init volatile union {
      unsigned char PM2;
      __BITS8       PM2_bit;};
```

Workaround

Use the @ operator instead of #pragma location to define an absolute address:

```
__sfr __no_init volatile union {
         unsigned char PM2;
          __BITS8       PM2_bit;
} @ 0xFF22;
```

No. C75	**Wrong Code generated for Pointer Array Index**

Details

In rare cases, the value of an index variable of a pointer array may be destroyed, if it is a local variable of type unsigned char and optimization level medium or higher is used.

Example

```
typedef struct
{
  unsigned char stringSize;  unsigned char string[1];
}STRING_01;

typedef struct
{
  unsigned char stringSize;  unsigned char string[7];
}STRING_07;

const STRING_07 string1 ={7,{0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08}};
const STRING_01 string2 ={1,{0x09}};
const STRING_01 string3 ={1,{0x0A}};
void *const array[3]={(void *)&string2,(void *)&string1,(void *)&string3};
unsigned char *ptr;
unsigned char dispbuffer[15];

void test (void)
{
   unsigned int     local1=1;
   DISP_STRING_01  *local2;
   unsigned char   *local3;
   unsigned char    local4;
   unsigned char    local5=0;

   if(local1 < 3u)  {
     local2 = (STRING_01 *)array[local1];
     local3 = &(local2->string[0]);
     local4 = local2->stringSize;
     ptr    = ((unsigned char *)(void *)&dispbuffer);
     if(local4 != 0) {
       do  {
         if (((*local3) > 1u) && ((*local3) <= 254u)) {
           local5 = 1;
         }
         else {
           local5 = 0;
         }
         local3 = &local3[local5];
         if(local4 >= local5) {
           local4 -= local5;
         }
         else {
           local5 = 0;
         }
       }while(local4 != 0);
     }
     else {
     }
  }
  else{
  }
}
```

Workarounds

Declare index variable (=local5) as volatile or reduce optimization level for this function to low

No. C76	**Internal Compiler Error while using __segment_size as memcpy Parameter**

Details

Using intrinsic function __segment_size as size parameter for memcpy function causes an internal compiler error:

Internal Error: [PaType – MemoryAttribute]: no memory attribute set

Example

```
#include <string.h>

#pragma segment="MY_SEGMENT_1" __near
#pragma segment="MY_SEGMENT_2" __near

void test(void)
{
    memcpy(__segment_begin("MY_SEGMENT_1"),
           __segment_begin("MY_SEGMENT_2"),
           __segment_size("MY_SEGMENT_2"));
}
```

Workaround

Use a temporary variable:

```
void workaround(void)
{
    size_t my_var;
    my_var= __segment_size("MY_SEGMENT_2");

    memcpy(__segment_begin("MY_SEGMENT_1"),
           __segment_begin("MY_SEGMENT_2"),
           my_var);
}
```

No. C77	**Bit Access generated although Keyword '__no_bit_access' was used**

Details

The compiler doesn't take care on the keyword __no_bit_access in pointer definitions.Although a pointer is correctly defined using the keyword '__no_bit_access', the compiler generates a bit access. For some I/O registers this causes an illegal I/O register access.

Example

```
volatile unsigned short __no_bit_access  v1;
volatile unsigned short __no_bit_access* ptr1 = &v1;

void test (void)
{
    *ptr1  = 0x0123U;
    *ptr1 |= 0x4000U;
}
```

Workaround

Use direct access instead of indirect pointer access

```
void workaround (void)
{
    v1  = 0x0123U;
    v1 |= 0x4000U;
}
```

No. C78	**Unclear Description of Parameter Passing for Structure Types in Compiler Manual**

IAR Reference: EW24225

Details

At page 108 of the RL78 C/C++ Compiler Reference Guide (2nd Edition) parameter passing to function is described. It is described that structure types parameters are passed via stack except the size is 1,2,4 and 4 bytes:

```
Structure types: struct, union, and classes, except structs and unions
of sizes 1, 2, and 4
```

This is correct, but additionally the structure type element must be word aligned. The alignment of the element is defined by the data type of the largest member.

Example

```
typedef struct  {
  unsigned char e1;
  unsigned char e2;
} s1_TYPE;
```

The above structure is passed via stack as only byte aligned elements are included.

Workaround

Include the structure type element in a union to force word alignment:

```
typedef union  {
  struct {
  unsigned char e1;
  unsigned char e2;
  };
  unsigned short dummy;
} s1_TYPE;
```

No. C79	**Wrong Code generated causing an unreachable else Path**

IAR Reference: EW24492

Details

Using high optimization level an optimization trying to determine whether a test had the same outcome for all values of the loop variable didn't handle expressions over- or under-flowing (going from UINT_MAX to zero, or vice versa) correctly.

The optimization incorrectly assumed ((loc1 - 4u) < 4u) would never be true for any value of loc1.

The problem can be triggered by tests inside loops, if
* the loop has constant lower and upper bounds,
* the expressions in the test consists of the loop variable and constants, and
* any expression in the test overflows or underflows when the lower or upper bounds are inserted in the test.

Example

```
void test(void)
{
  char loc1;
  char loc2 = 0u;
  for (loc1 = 0u; loc1 < 8u; loc1++ ) {
     if(loc1 < 4u) {
       if ((((loc1 < 4u)&&(0u < 2u)) ? foo1(0u,loc1) : 0u)) {
         loc2 |= (char)(0x01u << (loc1));
       }
     }
     else  {
       if (((((loc1-4u)<4u)&&(1u < 2u)) ? foo1(1u,(loc1 - 4u)) : 0u)) {
          loc2 |= (CHAR)(0x01u << (loc1));
       }
     }
   }
}
```

Workaround

Lower optimization level to medium or low.

No. C80	**No Code generated for if-Condition**

IAR Reference: EW24694

Details

A combination of cross-jump optimization and memory tracking may generate a faulty optimizations using high-speed optimization. As a result no code is generated for the if-condition.

Example

```
typedef  void  (*T_pFct)(void);
extern void  func1 (void)
static            unsigned char  volatile  a1[2];
static  __no_init unsigned char  volatile  a2[2];
static  __no_init unsigned char  volatile  a3[2];
static            unsigned char  const     a4[2]  = {0x11u, 0x22u};
static            T_pFct         const     tab[2] = {func1, (T_pFct)0};

unsigned char test(unsigned char const p1)
{
    unsigned char  loc1 = 0u;
    unsigned char  loc2 = 0u;
    unsigned char  loc3 = 0u;
    unsigned char  loc4 = 0u;
    if ( p1 < 2) {
            loc1 = a1[p1]; loc2 = a2[p1]; loc3 = a3[p1];
            if( (  loc1 == loc2  ) && (  loc2 == loc3  ) ) {
            }
            else {
                if (loc1 == loc2)   {
                    a3[ p1 ] = loc1 ;
                }
                else if (loc1 == loc3) {
                    a2[ p1 ] = loc1 ;
                }
                else if (loc2 == loc3) {
                    a3[ p1 ] = loc2 ;
                }
                else  {
                    a1[p1] = a4[p1]; a2[p1] = a4[p1];a3[p1] = a4[p1];
                    loc4 = 1u ;
                }
                loc1 = a1[ p1 ] ;
            }
        /* no code generated for following if condition at -Ohs */
        if( (  loc4 != 0u  ) && (  tab[p1] != (T_pFct)0  ) )  {
          ( tab[p1] )() ;
        }
    }
    return (loc1) ;
}
```

Workaround
Lower optimization level to medium or define local variable loc4 as volatile.

No. C81	**MISRA C 2004 Rule 10.6 not triggered**

IAR Reference: EW24733

Details

The compiler does not check MISRA-C 2004 rule 10.6 correctly. It bases the check on the usage of the constant instead of on the type of the constant.

Example:

```
#define UNSIGNED_CHAR_C  0x12
#define UNSIGNED_SHORT_C 0x1234
#define UNSIGNED_LONG_C  0x12345678

unsigned char  var1 = UNSIGNED_CHAR_C;  /* Error [Pm127]: */
unsigned short var2 = UNSIGNED_SHORT_C; /* no error MISRA C 2004 */
unsigned long  var3 = UNSIGNED_LONG_C;  /* no error MISRA C 2004 */
```

In above example error Pm127 should be triggered three times instead of only one.

Workaround
None; it will be fixed in next update.

No. C82	**Wrong Code generated for Array Index**

IAR Reference: EW25315

Details

Using an unsigned variable as index type can generate illegal indexes if the variable type is smaller than the pointer index type and optimization level ‘high’ is used.

Example:

```
const unsigned char id_tbl[2] = { 0x01, 0x02};
unsigned char id = 0x02;

int test( void )
{
  static unsigned char n;
  n = 2;
  while(n > 0)  {
    n--;
    if(id_tbl[n] == id) {
      break;
    }
  }
  return 0;
}
```

Workaround

Use a signed index variable: `static signed char n;`

K) Description of Operating Precautions for Linker (XLINK)

No. D3	**Breakpoint cannot be defined in function (only XCOFF78K Format)**

Details

In case of using a function with a name of 32 characters (or more) and using static local variables a debug problem occurs in the XCOFF78K format if the format modifier –ysp is set to truncate long symbol names. It is not possible to define a breakpoint within the function.

Workaround

Don't use the format modifier –ysp for the XCOFF78K format.
The format modifier –ysp was required by previous versions of the RENESAS debuggers. The format modifier is not necessary anymore if the following debugger versions are used:
ID78K0x-NS: V2.50 or later
ID78K0x-QB: V2.80 or later

No. D29	**Output file format UBROFF: Internal Linker Error 1**

Details

When generating output in the UBROF output format, an internal linker error may occur if statement information was generated for data declarations in assembler files:

```
              * * *   I N T E R N A L   E R R O R   * * *

In function: unknown
Diagnostic: unexpected exception
P0: 1    P1: 0
```

Workarounds

None. The problem is fixed in linker version V4.61t

No. D30	**Output file format UBROFF: Internal Linker Error 2**

Details

When generating output in the UBROF output format, an internal linker error may occur if a common segment is duplicated by linker option -K

```
              * * *   I N T E R N A L   E R R O R   * * *

In function: unknown
Diagnostic: unexpected exception
P0: 1    P1: 0
```

Workarounds

None. Please use linker version V5.00.1 or later.

No. D31	**Output file format ELF/DWARF: Error[e113]: Corrupt input file: "Illegal ELF-register."**

Details

The following sample causes a linker error e113 occurs in case of selecting the ELF/DWARF output file format:

```
Fatal Error[e113]: Corrupt input file: "Illegal ELF-register." in
module func_issue ( …)
```

Example:

```
unsigned char testvar;

void test_func(const unsigned char xxx)
{
    testvar = xxx;
}
```

Workarounds

None. Please use linker version V5.3.1.23 (available e/o February 2012) or later.

No. D32	**ELF/DWARF Format: Wrong Return Type Entry**

Details

When generating output in the ELF/ DWARF output format, XLINK output the type of the function instead of the return type of the function.

Workaround

Update XLINK to version V5.3.1.26 or later.

No. D33	**Definition of Segment Area Size ‘0’ causes Internal Linker Error**

Details

Definition of an area size of '0' in a packed segment definition (option –P) causes an internal linker error:

```
     IAR Universal Linker V5.4.1.30
    Copyright 1987-2012 IAR Systems AB.
Tool Internal Error:
Internal Error: In function:
Diagnostic: Value is too large to be represented as a unsigned 32-bit
quantity.
P0: 0    P1: 0
Internal Error: In function:
Diagnostic: Value is too large to be represented as a unsigned 32-bit
quantity.
P0: 0    P1: 0
Error while running Linker
```

Example

```
-P(CONST)MYCONST=1000:+0
```

Workaround

Please specify an area greater size than ‘0’

No. D34	**Erroneously Error e16 ‘Segment too long’ is generated**

IAR Reference EW24343

Details

When placing an empty segment (= size 0 bytes) in a placement range of 0 bytes using the notation START:+SIZE, erroneously error message e16 ‘Segment too long’ is generated even though the segment actually fits:

```
Error[e16]: Segment DFLIB_SHORT_RAM_RESERVED (size: 0 align: 0) is too
long for segment definition. At least 0 more bytes needed. The problem
occurred while processing the segment placement command
```

Workaround

Use a placement range greater than 0 bytes.

L) Description of Operating Precautions for Debugger (C-SPY)

No. E34	**If the same name is used for a data-object and for a data-type, this data-object can not be displayed in the Watch Window.**

Details

If the same name is used for a data-object and for a data-type, this data-object can not be displayed in the Watch Window. After adding the data-object to the Watch window, an error message is displayed instead of the value:

```
[syntax error, unexpected TYPE_NAME] column 1
```

Example

```
struct same_name {
    struct same_name * next;
    unsigned int dummy1;
    unsigned int dummy2;
};

struct same_name s1;
struct same_name *same_name;
```

Workaround

1) Use different names for data-objects and data-types
2) Enter the physical address of the data-object and the corresponding type-cast to the Watch Window instead of the symbolname.
   Example `(struct same_name*) 0xFB00`

The problem will be fixed in version V4.50a or later.

No. E43	**C-SPY 78K0R Simulator Driver: Interrupt simulation only works correct at priority level three.**

Details

If an interrupt level two to zero (highest) is defined, the interrupt simulation doesn't work correctly. Although the interrupt configuration (mask-flag and general interrupt enable flag) is correct, interrupts at any other level than three are disabled.

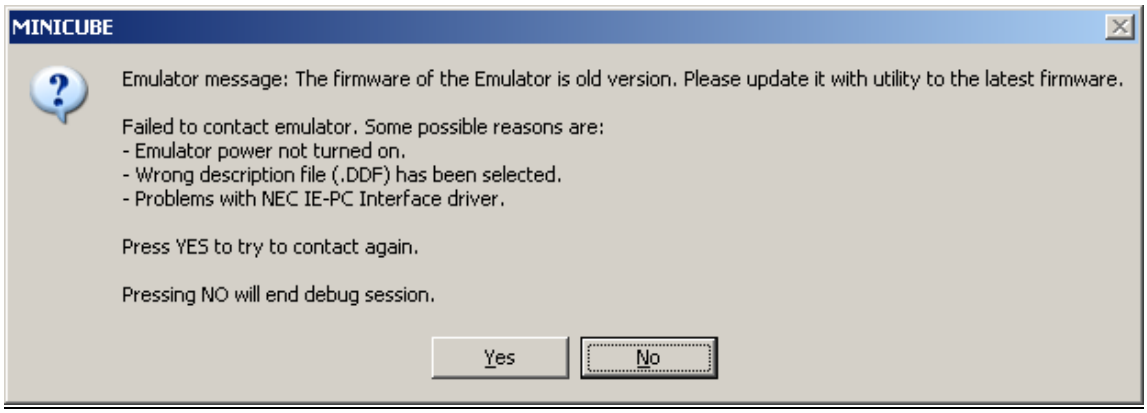
Workaround

Please use only priority level three (lowest) until the problem will be fixed in the next version.

No. E44	**C-SPY 78K0 MINICUBE2 Driver: Error message about old firmware version**

Details

After the installation of the update patch CS78KE_V460b the following error message will occur if the firmware-version of the MINICUBE2 is less than V4.06:



Workaround

The MINICUBE2 firmware V4.06 will be available b/o October 2008. Until then please contact the Renesas software tool support team (software_support-eu@lm.renesas.com) to receive further information fixing the problem.

No. E45	**C-SPY all Drivers: Update Time Watch Window**

Details

If a larger structure (size of several KB) shall be displayed in the C-SPY Watch Window, the update time can be up to five minutes if the OCD-emulator (e.g. MINICUBE2) is used and up to two minutes if the IECUBE emulator is used.

Workaround

None.

No. E46	**C-SPY Simulator Driver: Incorrect Value shown in Live-Watch Window**

Details

For certain source code when changing a element of a anonymous structure, an incorrect value is shown in the live watch window of the C-SPY simulator; when changing one of the bits, the whole base type value is changed.

```
#define TRUE   1
#define FALSE  0

volatile struct {
    UNSIGNED INT extP0_flag:1;
    UNSIGNED INT TM00_flag:1;
};

void test( void )
{
  extP0_flag = TRUE;
  extP0_flag = FALSE;

  TM00_flag = TRUE;
  TM00_flag = FALSE;
}
```

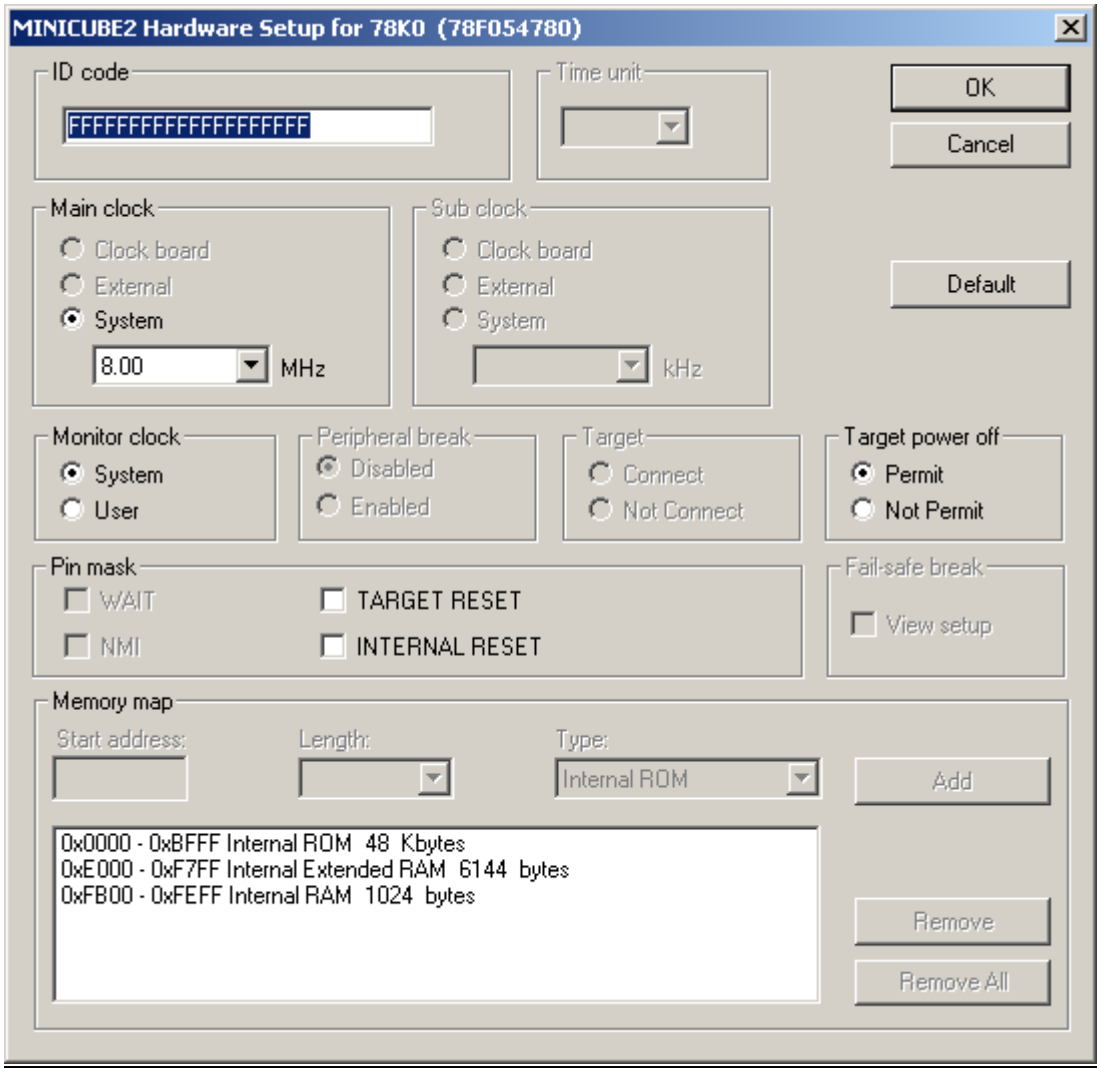
Workarounds

Use the Watch Window or use standard bitfields.

No. E47	**C-SPY 78K0 MINICUBE Driver: Incorrect System Clock Selection**

Details

If no oscillator is mounted on the target hardware and no external oscillator is mounted on the 78K0 MINICUBE2 clock board, three different system clocks (4 MHz, 8 MHz, or 16 MHz) can be provided by MINICUBE2. The selection is done in the C-SPY Hardware Setup Dialogue:



Independent of the selection, the provided system clock is always 4 MHz.

Workaround

Mount an external oscillator on the socket at the 78K0 MINICUBE2 clock board. If this is not acceptable, please contact the Renesas software tool support team (software_support-eu@lm.renesas.com) for further support.

No. E48	**Incorrect Variable Address may be displayed in Event Window or Watch Window**

Details

If a variable with the same name as one of the CPU registers (a, x, b, c, d, e, h, l) is used by an application, the symbol lookup cannot distinguish between variable and register name. The address of the symbol name found first is used, but it is undefined which symbol is found first and therefore a wrong address may be displayed.

Workaround

Please avoid using the variable names equal to the 78K register names until the problem is fixed.

No. E49	**Stack Initialization in default cstartup-module triggers C-SPY Debugger stack observation**

Details

A modified cstartup-module included in the compiler update patch V4.61a, triggers by fault the C-SPY stack-observation. In the modified cstartup-module the stack area is initialized to avoid faulty
IECUBE emulator fail safe breaks messages about a read access from uninitialized RAM.

Workaround

Please add the cstartup-module source code included in the EW78K (cstrtup.s26, subfolder 78K\src\lib\) to your application and change the fill-up value in line 135 from 0x00 to 0xCD.

```
;-------------------------------------------------------------------------------
;       CSTARTUP source for 78K
;
;       This module contains the code executed before the C/C++ "main"
;       function is called.
;       The code usually must be tailored to suit a specific hardware configuration.
;
;       Assembler options:
;
;       -D__STANDARD_MODEL__    To assemble for use with compiler standard
;                               code model.
;
;       -D__BANKED_MODEL__      To assemble for use with compiler banked
;                               code model.
;
;       -D__NEAR_MODEL__        To assemble for use with compiler near
;                               code model.
;
;       -D__FAR_MODEL__         To assemble for use with compiler far
;                               code model.
;
;       Linker options:
;
;       -D_CODEBANK_REG=0       To link for use with "standard" code model,
;                               no banked functions.
;
;       -D_CODEBANK_REG='addr'  To link for use with "banked" code model or
;                               "standard" code model with banked functions.
;                               'addr' = bank switch register address.
;
;-------------------------------------------------------------------------------
;       Copyright (c) 2003-2008 IAR Systems AB.
;       $Revision: 3577 $
;-------------------------------------------------------------------------------
…
   MOV A, #0xCD  ; line 135 change fill-up value from 0x00 to 0xCD
…
```

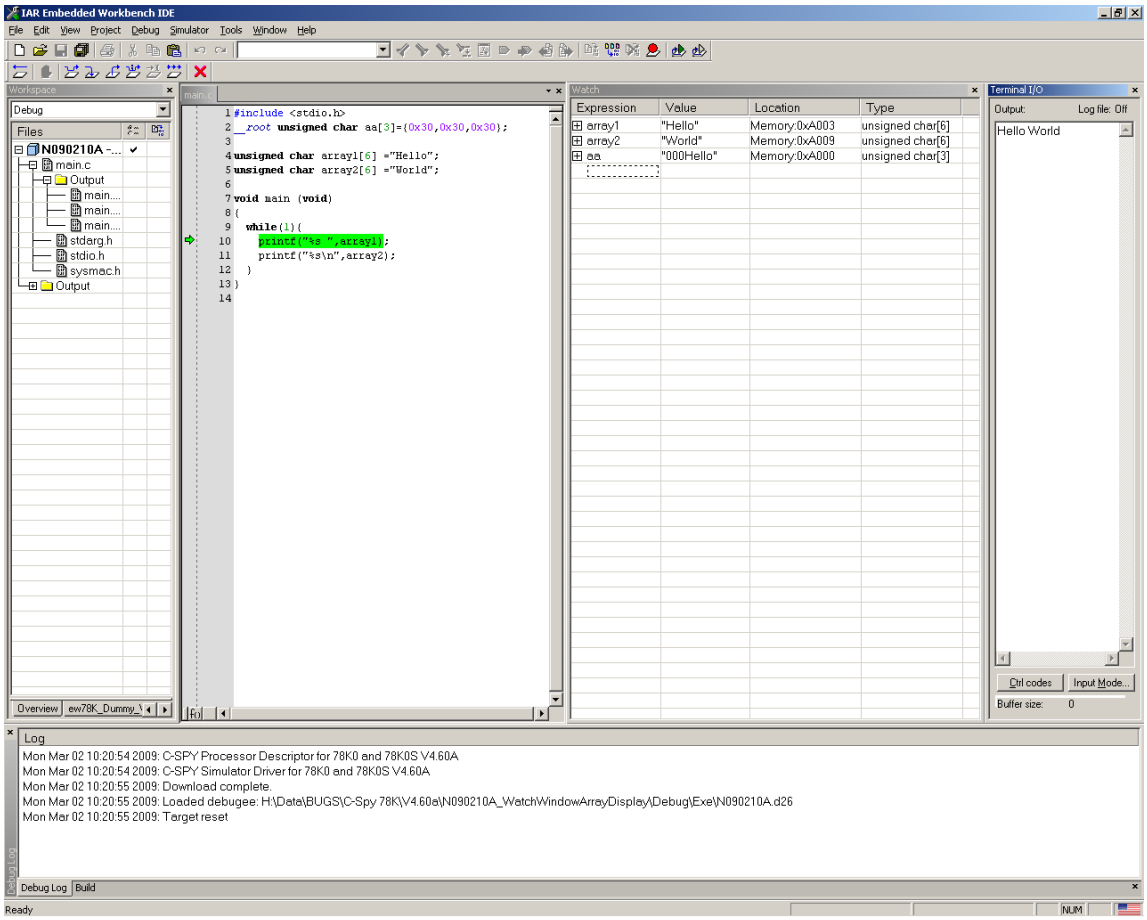
No. E50	**Wrong display of array in C-SPY Watch Window**

Details

If an array is displayed in the watch window, not only the correct content is displayed, but also the following addresses until the next string-end-character.

```
#include <stdio.h>
__root unsigned char aa[3]={0x30,0x30,0x30};

unsigned char array1[6] ="Hello";
unsigned char array2[6] ="World";
```



Workaround

None. The issue will be fixed in a future update.

No. E51	**C-SPY 78K Simulator Driver: Wrong macro access to 16bit data**

Details

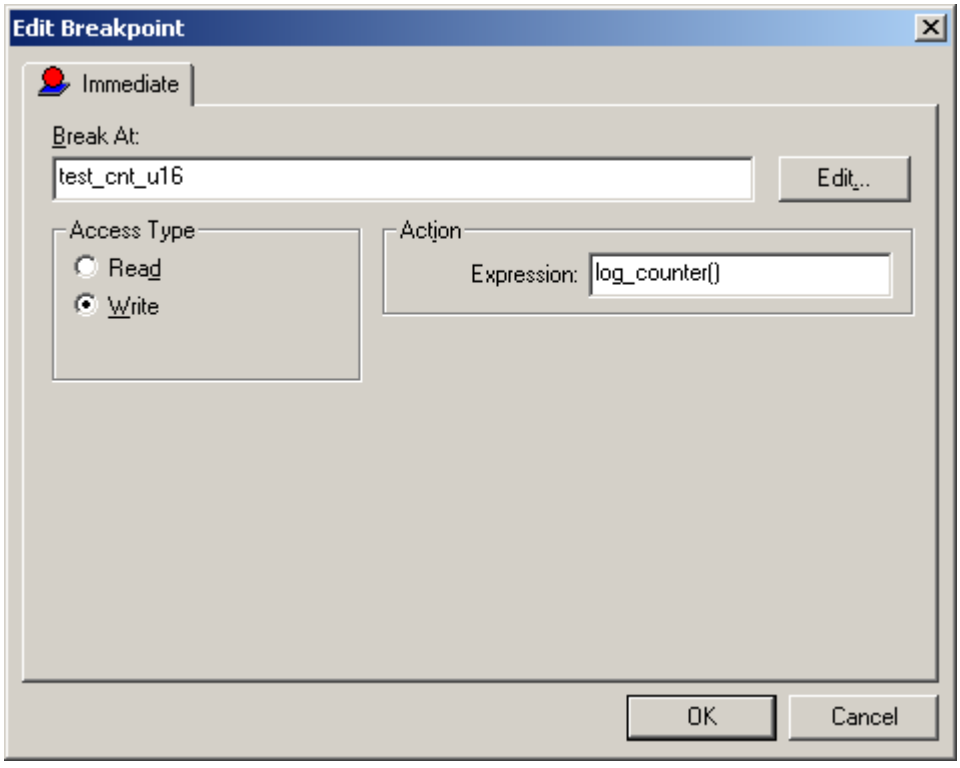
If a 16bit variable is accessed by a C-SPY macro triggered by an immediate breakpoint cause by an access to the same variable, the macro access may deliver a wrong result.

```
unsigned short  test_cnt_u16=0x1717;

void test (void)
{
    test_cnt_u16 ++;
}
```

C-SPY Macro:

```
log_counter()
{
  __message "Testcounter : ", test_cnt_u16:%d;
}
```



Workaround

Use a software breakpoint to trigger the C-SPY macro. The problem will be fixed in the next update.

No. E52	**C-SPY 78K: Displayed floating point value in watch window may be wrong**

Details

The displayed value of a floating point variable in the Watch Window may be incorrect.

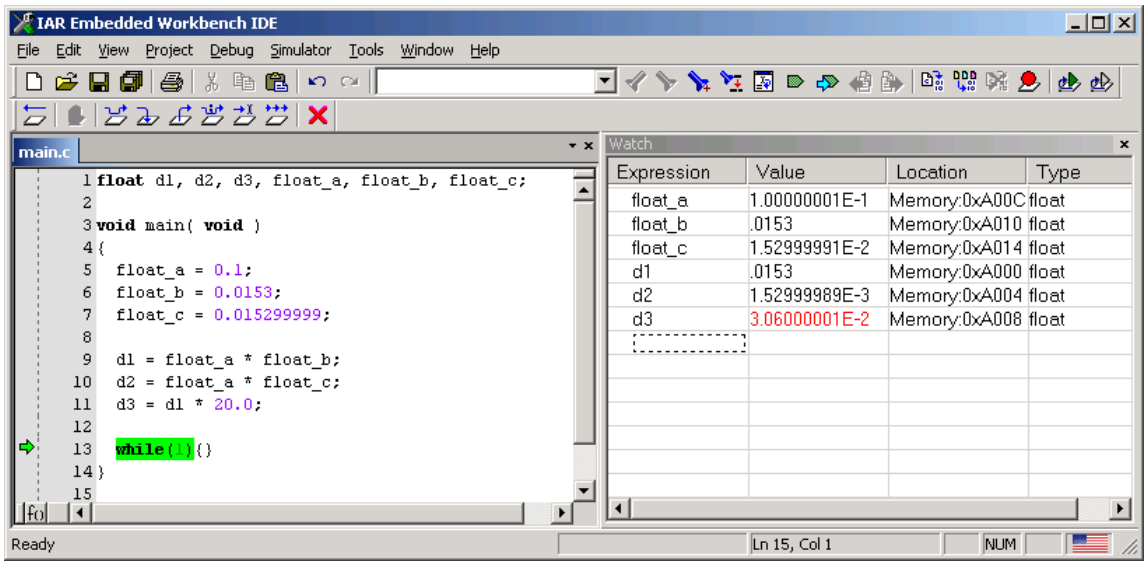
```
float d1, d2, d3, float_a, float_b, float_c;

void main( void )
{
  float_a = 0.1;
  float_b = 0.0153;
  float_c = 0.015299999;

  d1 = float_a * float_b;
  d2 = float_a * float_c;
  d3 = d1 * 20.0;

  while(1){}
}
```

The displayed value of ‘d1’ is wrong, but the application uses the correct value. This can be seen in the calculated value of d3.



Workaround

None. The problem will be fixed in the next update.

No. E53	**C-SPY 78K: Resetting a running application causes stack warning message**

Details

When reset an application while it is running a stack pointer out of range warning is generated.

```
The stack pointer for stack 'Stack' (currently Memory:0xFFC00) is
outside the stack range (Memory:<stack_start> to Memory: <stack_end>)
```

This error message is cause by a debugger problem and doesn't show an application problem.

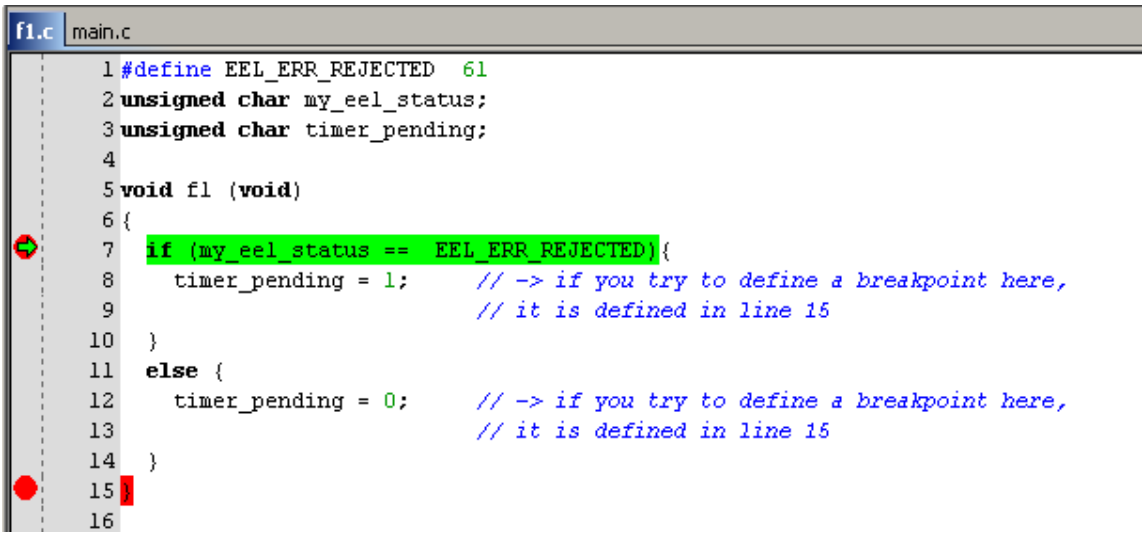
Workaround

Manually stop the application before resetting it.

No. E54	**C-SPY 78K: Breakpoint can not be defined at some source lines**

Details

Due to missing statement information in the debug information generated by the compiler, breakpoints could in rare cases not be set on specific C source lines.



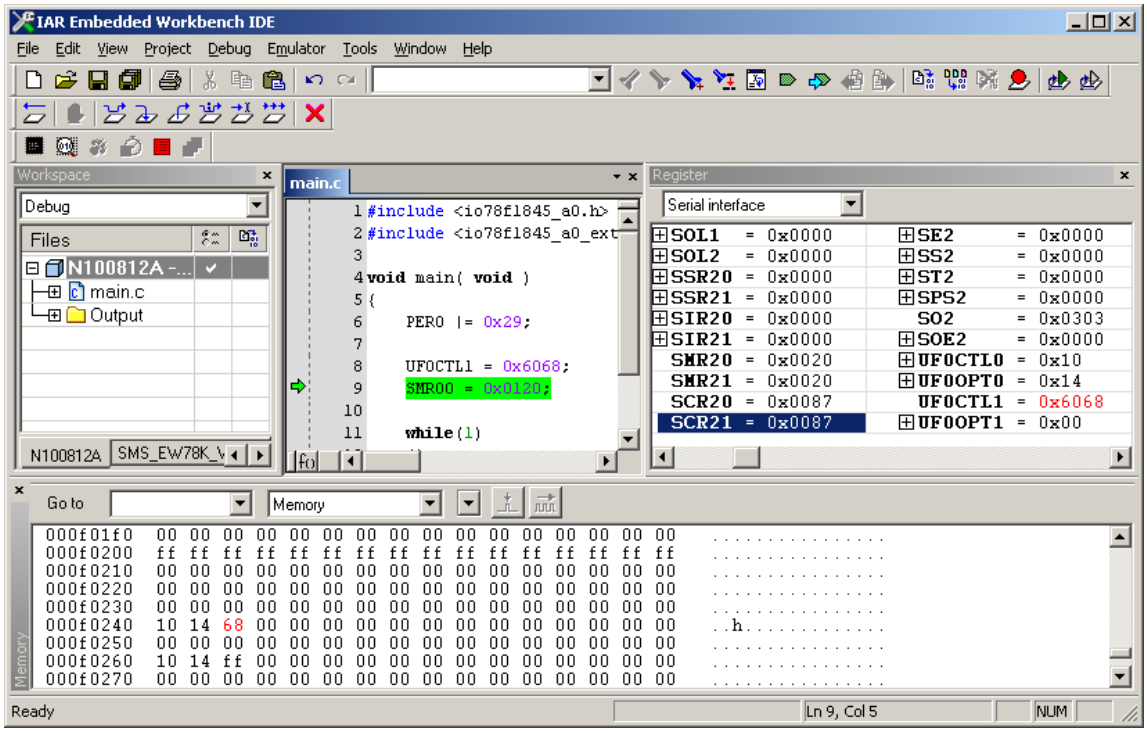
Workaround

Define the breakpoint in the assembler window. Please keep mind that this breakpoint may get invalid after a modification of the application.

No. E55	**C-SPY 78K0R: Wrong Display of 16bit SFR in Memory Window**

Details

The high byte of a 16bit special function register maybe displayed incorrectly in the Memory Window. The correct value is displayed in the Register Window



The issue occurs at the IECUBE driver as well as at the MINICUBE driver.

Workaround

Please use only the Register Window to check the correct content of special function registers.

No. E56	**C-SPY 78K0R IECUBE Driver: Inaccurate Time Measurement Result**

Details

The execution time measurement results displayed in the Register Window are inaccurate as they are calculated on a timer resolution of 17ns instead of the 16.6667ns.

Example
The execution time of 1000 nop-instructions at a CPU clock of 20MHz should be 50000 ns, but the displayed result is 50999 ns

Workaround

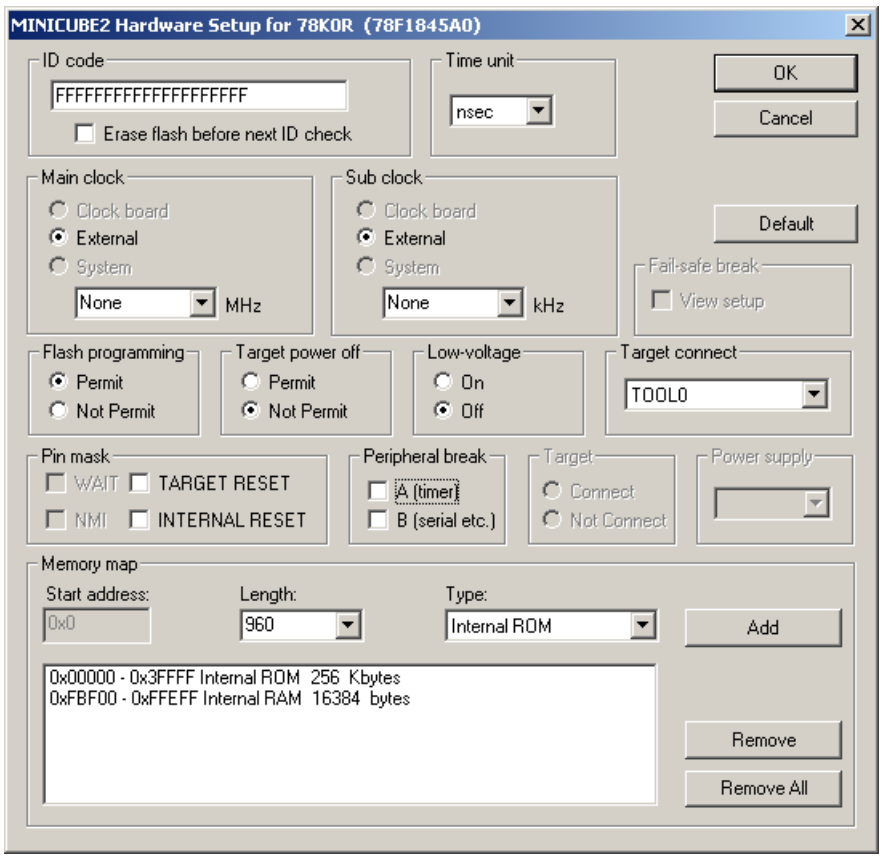
Please use a conditional timer or manually correct the displayed value by the factor (16.6667/17)

No. E57	**C-SPY all Drivers: Program Counter may be uninitialized**
	Details If additional images are downloaded to emulator, the program counter may be uninitialized (value 0xFFFF). *Workaround* Please use a manual RESET signal to initialize the Program counter.

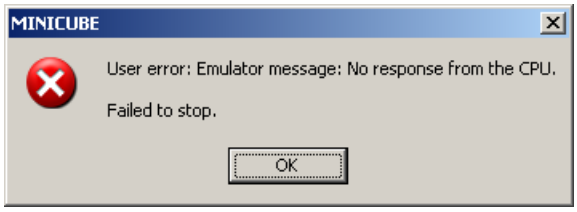
No. E58	**C-SPY 78K0R MINICUBE2 Driver: Broken Emulator Communication**

Details

After selecting the internal sub clock as CPU clock the communication between debugger and emulator will be broken at a manual stop, if single wire target connection (-> TOOL0) is used:



Example:
After a manual break while CPU is running at sub clock, the following error messages occur and the debugger session is closed:



Workaround

Please use the two wire target selection (TOOL0 + TOOL1).

No. E59	**All C-SPY Drivers except Simulator Driver: System Macro __driverType not implemented**

Details

Although described in the User's Manual, the system macro `__driverType` is not implemented. Using the macro causes the following error message:

```
Error: Unknown or ambiguous symbol. __driverType
```

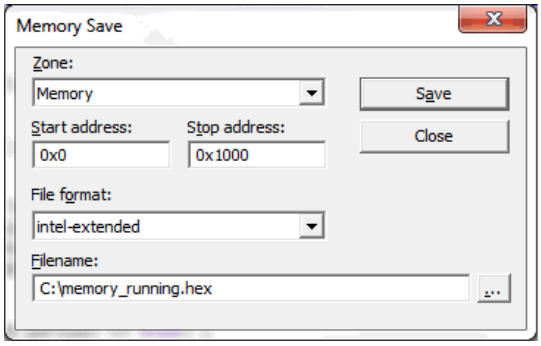
Workaround

None. The missing macro will be implemented in future update.

No. E60	**All C-SPY Drivers: Incorrect Flash Memory Upload in Run-Mode**

Details

When performing a memory upload (Debug --> Memory --> Save...) while the application is running the content of the output file is incorrect.



Workaround

Stop application before memory update.

No. E61	**ORTI Plug in Error Message „Memory Exhausted”**

Details

Due to a memory leak in the ORTI plug in a memory exhausted error message may be generated after selecting the ORT file:

```
Download complete.
Loaded debugee: C:\...\ORTI_Test_V471.d26
Target reset
ORTI Plug-in. ORTI Plug-in. File:
C:\...\TUTORIAL.ORT", memory exhausted.
ORTI Plug-in. ORTI Plug-in. Row: 65: ""GetEvent:
Called from invalid call context" = 0x4306", Col: 105
Disabled due to above error.
```

Workaround

Reduce size of table defined in ORT file.

No. E62	**Constant Data Object located in Data Flash Area displayed incorrectly in Watch Window**

Details

A constant data object located Data Flash area is displayed incorrectly in Watch Window. Instead of the correct value, at each break of the application a different and incorrect value is displayed.

Workaround

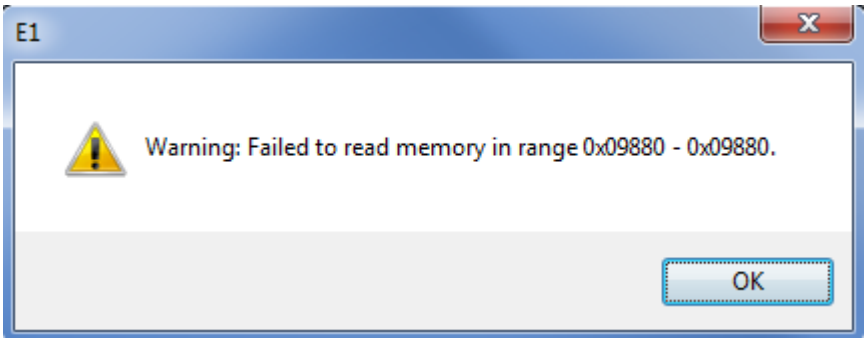
Use the Data Flash Window instead of Watch Window. Will be corrected in future update.

No. E63	**Reading Data-Flash-Memory causes an Error**

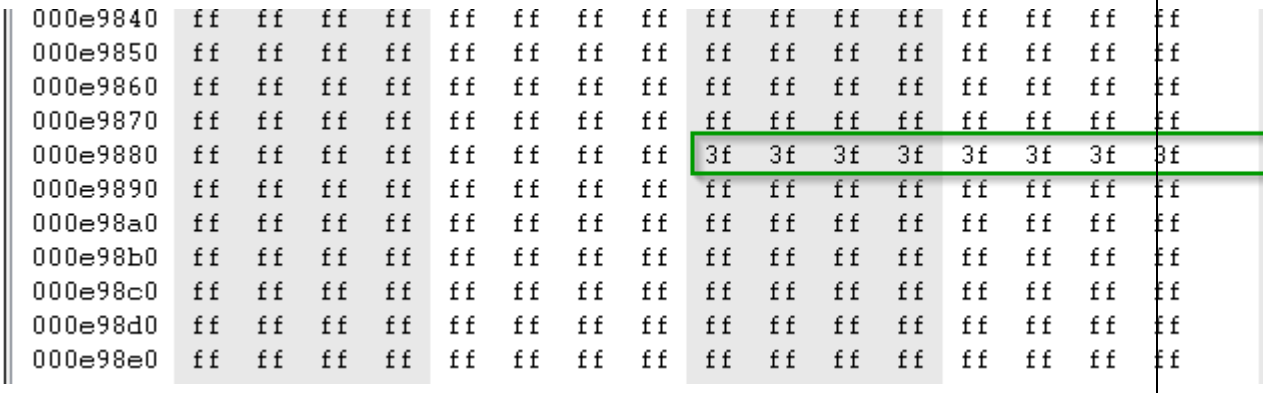
IAR Reference: EW25176

Details

Reading Data-Flash-Memory while application is running may cause errors.
The following error will be shown during the debug session start, if the Data-Flash-Window is opened with more than 12 address lines:



If such a warning occurs, the Data-Flash-Window shows wrong values:



Workaround

Turn of ‘Run to main’ feature or make the Data-Flash-Window so small that its data is read before C-SPY starts the application to reach main function.

M) Description of Operating Precautions for the Assembler A78K0R

No. F1	**RSEG Directives can not be used in Macro Definitions**

Details

The assembler calculates a wrong relative jump-distance if the RSEG directive is used within a macro definition:

Example

```
myDummyMacro MACRO
         RSEG    CODE
         NOP
         ENDM
```

Workaround

Don't use the RSEG directive in macro definitions. The used code-segment must be defined in the code where the macro is expanded to.

No. F11	**Illegal indirect MOVW instruction is accepted and wrong Op-Code is generated**

Details

For the illegal instruction MOVW AX,[BC] the opcode for MOVW, word[BC] is used but the offset address is not entered.

Example

```
        PUBLIC asm_func

        RSEG CODE:CODE
asm_func:

        MOVW AX,[BC]  ; -> illegal instruction, opcode for MOVW
                      ;AX,word[BC] generated, but no offset entered


        ret
```

Workaround

Please use correct instruction '`MOVW  AX,0x0000[BC]` '.

No. F12	**Illegal Op-Code generated if SFR symbol is defined after the usage**

Details

The assembler generates an illegal opcode, if a sfr-symbol is defined after the usage. Instead of a three byte instruction (2 byte opcode + 1byte for the low-byte SFR-address) a four byte instruction (2 byte opcode + 2byte address) is generated.

Example

```
        PUBLIC test
SFR1    DEFINE  0xFFFF0

        RSEG CODE
test:
        MOV1 SFR1.0,CY
        MOV1 SFR2.0,CY      ; illegal opcode generated
        RET

SFR2    DEFINE  0xFFFF1

        ret
```

Workaround

Please make sure that all SFR symbols are defined before using them.

No. F13	**Directive DS64 is not implemented**

Details

Although described until the 3rd edition of the 78K assembler manual, the directive DS64 is not implemented and usage causes a syntax error message:

```
Error[As001]: Invalid syntax <asm-source-file> <line-number>
```

Example

```
     PUBLIC v1

     RSEG NEAR_Z:DATA

V1:  DS64 1

      END
```

Workaround

Please use the DS or any other implemented DS<x> directive instead of DS64 (e.g. DS64 1 can be replaced by DS 8 or DS32 2). Documentation will be updated

No. F14	**Wrong Code Generated for Bit Test Instructions**

IAR Reference EW24018

Details

In case of using absolute segments (ASEG or ASEGN) the assembler generates wrong hex code for the bit test instructions like e.g. BZ and BNZ. A wrong branch address is calculated.

Example

```
    ASEGN C2:CODE,0x10
m1:
    MOV   a,#1
    CMP   a,#0
    BNZ   m1
    RET
```

List-File:

```
    000014 DF0E         BNZ   m1      <- wrong hex coode should be DFFA
```

Workaround

Use a relocatable instead of an absolute segment:

```
    RSEG RCODE:CODE

m1:
    MOV   a,#1
    CMP   a,#0
    BNZ   m1
    RET
```

List-File:

```
    000004 DFFA             BNZ   m1
```

N) Description of Operating Precautions for the C/C++ Compiler ICC78K0R

No. G36	**Internal Compiler Error due to non terminated Jump Size Optimization**

Details

At optimization level high or medium some complex switch statements cause an internal compiler, because the jump size optimization will not terminate:

Workaround:

Reduce the optimization level to low or the function including the switch statement by using #pragma optimization in front of the function definition.

The issue will be fixed in the next update (target May 2011)

No. G37	**Internal Compiler Error at using intrinsic function '__segment_begin'**

Details

At optimization level medium or higher using the intrinsic function '__segment _begin' inside an if-statement or any kind of loop may cause an internal compiler error.

Example:

```
#include <intrinsics.h>
#pragma segment="MYSEG"

extern void func1(unsigned char*);

void test (void* ptr)
{
  if( ptr != ((void*)0) ) {
    func1(__segment_begin("MYSEG"));
  }
}
```

Workarounds:

1)
Reduce the optimization level to low for the function including the if statement by using #pragma optimization in front of the function definition:

```
#pragma optimize=low
void test (void* ptr)
{
  …
}
```

2)
Put the intrinsic function call in a function which is not inlined.

The issue will be fixed in the next update (target May 2011)

No. G38	**Wrong code generated for far Branch Inline-Assembler Instruction**

Details

If the inline assembler instruction ‘br F:xxxxx’ is used inside an if statement and an earlier instruction had accessed a non-absolute near address, faulty code is generated. Instead of using the given high byte of the address (bit 16-23), 0x0F is used.

Example:

```
#include <intrinsics.h>
unsigned char Array[3];

extern void func2 (void);

void test (void)
{
  func2();
  if (Array [0]== 0xFF) {
    asm("br F:0x003010");
  }
}
```

The generated assembler code is

```
\   000009 EC10300F              br F:0x003010
```

Workarounds:

1)
Use an indirect branch via register AX

```
  if (Array [0]== 0xFF) {
    asm("movw AX,#0x3010");
    asm("br   AX        ");
  }
```

2)
If it is acceptable to use a call instead of a branch instruction, a C function pointer can be used:

```
  void (*ptr)(void);
  func2();
  if (Array [0]== 0xFF) {
    ptr= ( void (*)(void)) 0x3010;
    ptr();
  }
```

The issue will be fixed in the next update (target May 2011).

No. G39	**Inline Assembler Range Error Message triggered by Mistake**

Details

If the inline assembler instruction ‘br N:xxxx’ is used inside an if statement, range error message [As026] is triggered by mistake:

```
Error[As026]: Limit exceeded: Allowed range is 0 - 0xffff (0 - 65535),
value is 0xf3010 (995344)
```

Example:

```
#include <intrinsics.h>
unsigned char Array[3];

extern void func2 (void);

void test (void)
{
  func2();
  if (Array [0]== 0xFF) {
    asm("br N:0x3010");
  }
}
```

Workarounds:

1)
Use an indirect branch via register AX

```
  if (Array [0]== 0xFF) {
    asm("movw AX,#0x3010");
    asm("br   AX        ");
  }
```

2)
If it is acceptable to use a call instead of a branch instruction, a C function pointer can be used:

```
  void (*ptr)(void);
  func2();
  if (Array [0]== 0xFF) {
    ptr= ( void (*)(void)) 0x3010;
    ptr();
  }
```

The issue will be fixed in the next update (target May 2011).

No. G41	**Internal Compiler Error at far pointer access to I/O area**

Details

Using a far pointer access into the I/O register area cause an internal compiler error:

```
   Internal Error: [CoreUtil/General]: Size mismatch for "MOVW
   HL, ES:0xFFE0", inserted as 3 bytes, assembled as 4 bytes
```

Example:

```
#define GetIF0   (*((volatile unsigned short int __far *) (0xffe0u)))

void test (void)
{

  if (GetIF0 != 0xAAAAu) {
     …
  }
}
```

Workaround:

Avoid using pointer access to I/O area and use direct memory access.
The issue will be fixed in the next update (target May 2011).

No. G42	**Internal Compiler Error at calling strcpy or memcpy in far data model**

Details

Calling strcpy or memcpy in the far data model using optimization level ‘low’ may cause an internal compiler error:

```
   Internal error [assign_colors_C01]: coloring failed
```

Example:

```
#include "string.h"

typedef struct t_deviceData_tag
{
  unsigned char  cCompleteTypeNumber;
  unsigned char u8Data;
}t_deviceData;

typedef struct  t_meterCache_tag
{
  t_deviceData device[2];
} t_deviceCache;

t_deviceCache deviceCache;

void test(unsigned char u8Interface, unsigned  char u8Data)
{
  t_deviceCache *pDeviceCache = &deviceCache;
  memcpy((void*)&pDeviceCache->device[u8Interface].u8Data,
         (void*)u8Data, sizeof(u8Data) );
}
```

Workaround:
Use optimization level medium or higher.
The issue will be fixed in the next update (target May 2011).

No. G43	**Wrong Pointer Access to Special-Function-Register in Data Model 'far'**

Details

When using a pointer to a special-function-register in the far data model the ES register is set to 0x00 instead of 0x0F. During an IECUBE C-SPY debug session this causes a fail safe break:

```
Break reason: Illegal write to write protected area.
Unable to execute: driver error.
```

Example:

```
extern __sfr __no_init volatile unsigned char PM0 @ 0xFFF20;

typedef union tMyUnion_tag
{
    unsigned char              BYTE;
    struct
    {
        unsigned char          BIT0:1;
        unsigned char          BIT1:1;
        unsigned char          BIT2:1;
        unsigned char          BIT3:1;
        unsigned char          BIT4:1;
        unsigned char          BIT5:1;
        unsigned char          BIT6:1;
        unsigned char          BIT7:1;
    } bit_view;

} tMyUnion;

void test (void)
{
   ((volatile tMyUnion *)((&PM0)))->BYTE = (0x70));
}
```

Workaround:

Use a near pointer:

```
void test (void)
{
   ((volatile tMyUnion __near *)((&PM0)))->BYTE = (0x70));
}
```

The issue will be fixed in the next update (target June 2011).

No. G44	**Error Message Pe028 Triggered by Mistake**

Details

Using CLIB as runtime library, data model far and NULL pointer definition of header file “stddef.h” triggers error message Pe028 by mistake:

```
Error[Pe028] expression must have a constant value.
```

Example:

```
#include <stddef.h>

unsigned char __near * const __near DataPointerArray[1] ={ NULL};

typedef void (__near_func *NearFuncPtr)(void);

const NearFuncPtr __near FuncPointerArray[1] ={ NULL };
```

Workaround:

Use DLIB as runtime library.
The issue will be fixed in the next platform update (target October 2012).

No. G45	**Internal Compiler Error: Casting SADDR Address into far Pointer**

Details

When casting the address of a short address variable to a far pointer and using optimization level medium or high, the compiler could optimize the code in a way that caused an internal error when generating the assembler code.

Example

```
typedef struct MyStruct
{
  int mAlpha;
  int mBeta;
}tS;

__saddr struct MyStruct Gamma;

__root int myFunc(void)
{
  tS * pS;
  pS = &Gamma;

  if(pS->mAlpha != pS->mBeta)  {
    return 1;
  }
  else {
    return 0;
  }
}
```

Workaround

Use optimization level low.

No. G46	**Error in Device Specific Header File**

Details

If two 8bit-access register names are defined at the same I/O register address and also a 16bit-access symbol is defined, a wrong I/O register access is defined for the second 8bit register.

Affected devices: All RL78 devices including a serial array unit.

Example

Register Name	Access Size	Register Address
SDR00	16	0xFFF10
SIO00	8	0xFFF10
TXD0	8	0xFFF10

```
__saddr __no_init volatile union {
     unsigned short SDR00;
     struct {
          unsigned char SIO00;
          unsigned char TXD0;
     };
} @ 0xFFF10;
```

Due to the above definition the compiler generates code where register TXD0 is located at address 0xFFF11 instead of 0xFFF10.

Workaround

Please install the latest version of the header files using the corrected definitions.

Example

```
__saddr __no_init volatile union {
     unsigned short SDR00;
     union {
          unsigned char SIO00;
          unsigned char TXD0;
     };
} @ 0xFFF10;
```

The corrected header files are included in the service pack SP-EW78K-V4712 available at the IAR MyPages .

No. G47	**Internal Compiler Error: EctContextBase::GetValue**

Details

A far data access inside an interrupt function may cause an internal compiler error at optimization level medium or higher.

Example1

```
typedef struct
{
  int            rx_busy;
  unsigned char  rx_byte;
} uart_t;

__far uart_t uart;

__interrupt void isr_sr2(void)
{
    if(uart.rx_busy == 1)
    {
        uart.rx_byte = SDR21;
    }
}
```

Workarounds

1) Reduce the optimization level for the interrupt function:

```
#pragma optimize=low
 __interrupt void isr_sr2(void)
 {
    …
 }
```

2) use a near data access: `__near uart_t uart;`

No. G48	**Wrong Offset Address Calculation**

Details

A wrong offset address is calculated at optimization level medium or higher if the calculation touches the 64KB border 0xFFFF (near data access). Instead of an address inside the highest 64KB segment the corresponding address in the lowest 64KB segment is used, e.g. 0x00158 instead of 0xF0158.

Example

```
typedef unsigned char U08;
typedef union tMCMPC1_tag
{
    U08             u8_view;
    struct
    {
        U08          DIR0:1;
        U08          DIR1:1;
        U08          ADB0:1;
        U08          ADB1:1;
        U08          TEN:1;
        U08          ZPD:1;
        U08          TWIN:1;
        U08          AOUT:1;
    } bit_view;
    struct
    {
        U08          DIR0:1;
        U08          DIR1:1;
        U08          ADB0:1;
        U08          ADB1:1;
        U08          TEN:1;
        U08          ZPD:1;
        U08          TWIN:1;
        U08          AOUT:1;
    } bitgroup_view;
} tMCMPC1;

struct _tstMCMPCn_tag
{
   tMCMPC1 _xMCMPC1;
};

extern __near __no_init volatile tMCMPC1 _xxMCMPC1 @ 0xF016A;

#define _nAddrMCMPC1   (&_xxMCMPC1)
#define nAddrMCMPC1    (&_xxMCMPC1)
#define pMCMPC1        ((volatile tMCMPC1 __near *)(nAddrMCMPC1))

volatile tMCMPC1 __near * pTestPtr;

void test (void)
{
  pTestPtr = ((volatile tMCMPC1 __near *)(((volatile U08 __near *)(pMCMPC1))-(18)));
  *((volatile U08 __near *)(&(pTestPtr->u8_view)))=(U08)0x08U;
}
```

Workarounds

Reduce the optimization level for the function:

```
#pragma optimize=low
void test(void)
{
    …
}
```

No. G49	**Internal Compiler Error CoreUtil/General at using MISRA C and Option `–header_context`**

Details

Misra errors without a file-position cause an internal compiler error when the option --header_context is used

Example

```
__near_func void test (void);

__near_func void test()
{

}
```

Workarounds

None. Issue will be fixed in future update.

No. G50	**Far Pointer defined instead of near Pointer**

Details

Although defined correctly as pointer to near near object according to the description at page 205 of the compiler manual (C78K-4, April 2010) a pointer to far object is generated by the compiler, if the pointer is a member of a structure.

Example

```
typedef unsigned char tu8;

typedef struct {
 __far tu8       *       TestPtr1;  /* should not point at __far */
       tu8 __far *       TestPtr2;
       tu8       *       TestPtr3;
}TestPtrStruct;

__near TestPtrStruct TestPtrStruct1;
__far  TestPtrStruct TestPtrStruct2;
```

Workarounds

None. Issue will be fixed in future update.

No. G51	**Internal Compiler Error at Negation of Bitfield-Element**

Details

Independent of the selected optimization level an internal compiler error may occur if a negated bitfield element is used as return value:

`Internal Error: [CoreUtil/General]: Access violation`

Example

```
struct s{
  int m : 1;
};

int test(struct s *p)
{
  return !p->m;
}
```

Workarounds

Use a temporary variable and select optimization level low:

```
int test(struct s *p)
{
  unsigned int temp = !p->m;
  return temp;
}
```

No. G52	**Internal Compiler Error at Macro Expansion**

Details

Using an optimization level medium or higher an internal compiler error may occur at the following macro expansion:

`Internal Error: [CoreUtil/General]: Stack overflow`

Example

```
#define EXPAND(x) x x x x x x x x x x

int test(int b)
{
      int n = b+1;
      int m = b+2;

      EXPAND (EXPAND (n+=m; m-=n;))
      EXPAND (EXPAND (n+=m; m-=n;))
      return n+m;
}
```

Workarounds

Reduce optimization to level low for the function:

```
#pragma optimize=low
int test(int b)
{
  …
}
```

No. G53	**Internal Compiler Error at Returning a negated right-shifted Value**

Details

Independent of the used optimization level an internal compiler error may occur at returning a negated right-shifted value:

`Internal Error: [CoreUtil/General]: Access Violation`

Example

```
int test(unsigned int x)
{
  return -((int)(x >> 15));
}
```

Workaround

Reduce optimization level to low for the function and use a temporary variable:

```
#pragma optimize=low
int test(unsigned int x)
{
  int temp = x >> 15;
  return -(temp);
}
```

No. G54	**Internal Compiler Error at Returning a Comparison Result**

Details

If an optimization level medium or higher is used an internal compiler error may occur at returning a comparison result including a logical and operation:

`Internal Error: [CoreUtil/General]: Access Violation`

Example

```
int test(int x)
{
  return (x & 2) == 0;
}
```

Workaround

Reduce optimization level to low for the function:

```
#pragma optimize=low
int test(int x)
{
  …
}
```

No. G55	**Wrong Code generated for Function Call directly after memcpy-Function call**

Details

At optimization level high the compiler generates wrong code at parameter preparation for a function call directly afterwards a memcpy function call.
After the memcpy function call the local variable 'y' is not updated before calling memtest:

Example

```
#include<stdio.h>
#include<string.h>

int func1(int p1){
   …
}

int test(int *p1){
  int y=100;
  memcpy(&y, p1, 2);
  return func1(y);
}
```

Workarounds

1) Reduce optimization to level medium for the function:

```
#pragma optimize=medium
int test(int b)
{
  …
}
```

2) Add a nop-instruction by using inline assembler between the function calls:

```
int test(int *yy){
  int y=100;
  memcpy(&y, yy, 2);
  asm("nop");
  return func1(y);
}
```

2) Define the local variable as volatile:

```
int test(int *yy){
  volatile int y=100;
  memcpy(&y, yy, 2);
  return func1(y);
}
```

No. G56	**Compilation process stalls**

Details

Comparing two non-volatile char variables located in saddr memory space could cause the 78K0R compiler to hang on higher optimization levels.

Example

```
__saddr char a = 5;
__saddr char b = 7;

int test(void)
{
  if (a == b)  {
    return 1;
  }
  return 0;
}
```

Workarounds

1) Reduce optimization to level medium for the function:

```
#pragma optimize=medium
int test(void)
{
  …
}
```

2) Define variable as volatile

3) Define only one SADDR varibale

No. G57	**Wrong Code could be generated for Near Pointer Indexing**

Details

Near pointer indexing could generate a faulty use of the word[BC] address mode, if it is in the form of array[-var] or of array[<constant>-var].

Example

```
extern unsigned short len;
extern unsigned char  buffer[];
extern unsigned char  result;

void test (void)
{
    result = buffer[10 - len];
}
```

Workaround

Please use a temporary variable to calculate the index:

```
extern unsigned short len;
extern unsigned char  buffer[4];
extern unsigned char  result;

void workaround (void)
{
   unsigned char temp;
   temp   = 10-len;
   result = buffer[temp];
}
```

No. G58	**#pragma location Directive does not support Unions and Structs**

Details

The #pragma location directive does not support unions and structs. An warning is generated to inform the user:

```
Warning[Pe609]: this kind of pragma may not be used here
```

Example

```
typedef struct
{
    unsigned char no0:1;
    unsigned char no1:1;
    unsigned char no2:1;
    unsigned char no3:1;
    unsigned char no4:1;
    unsigned char no5:1;
    unsigned char no6:1;
    unsigned char no7:1;
} __BITS8;

#pragma location = 0xFFF22;
__sfr __no_init volatile union {
      unsigned char PM2;
      __BITS8       PM2_bit;};
```

Workaround

Use the @ operator instead of #pragma location to define an absolute address:

```
__sfr __no_init volatile union {
         unsigned char PM2;
          __BITS8       PM2_bit;
} @ 0xFFF22;
```

No. G59	**Internal Compiler Error while using __segment_size as memcpy Parameter**

Details

Using intrinsic function __segment_size as size parameter for memcpy function causes an internal compiler error:

Internal Error: [CoreUtil/General]: Access Violation

Example

```
#include <string.h>

#pragma segment="MY_SEGMENT_1" __near
#pragma segment="MY_SEGMENT_2" __near

void test(void)
{
    memcpy(__segment_begin("MY_SEGMENT_1"),
           __segment_begin("MY_SEGMENT_2"),
           __segment_size("MY_SEGMENT_2"));
}
```

Workaround

Use a temporary variable:

```
void workaround(void)
{
    size_t my_var;
    my_var= __segment_size("MY_SEGMENT_2");

    memcpy(__segment_begin("MY_SEGMENT_1"),
           __segment_begin("MY_SEGMENT_2"),
           my_var);
}
```

No. G61	**Wrong Code generated for Bit Negation of 32bit Bitfield**

Details

If a 32bit bitfield is used wrong code is generated to negate a single bit. Instead of negating the port-bit the result is always 1 due to the instruction sequence.

Example

```
#include <io78f1845_a0.h>

struct {
  unsigned long bit0:1;
} s1;

void error (void)
{
  s1.bit0 = !P12_bit.no4;
}
```

Workarounds

Use one or two 16bit bitfields:

```
struct {
  unsigned int bit0:1;
} s2;

void workaround (void)
{
  s2.bit0 = !P12_bit.no4;
}
```

No. G62	**CPU Cycle Information of CALLT Instruction missing in Compiler-List-File**

Details

The CPU cycle information of CALLT instructions in missing the compiler list file:

```
     16              f1();
   \                     ??main_0:
   \   000000 61..           CALLT     [__T_f1]
     17              f2();
   \   000002 FD....         CALL      f2                  ;; 3 cycles.
```

CPU cycle information was added to list file since compiler version V4.71.x.

Example

```
__callt void f1 (void)
{
}

void f2 (void)
{
}

void main (void)
{
  while(1) {
    f1();
    f2();
  }
}
```

Workaround

None. Listed as improvement proposal for future update

No. G63	**Wong Code generated at far-Pointer Arithmetic**

Details

Constant folding of far pointers might generate faulty addresses if the addition causes an overflow into bit 12.

Example

```
#define DFLASH_START_PTR ((unsigned char __far *)(0xE9800uL))

volatile unsigned long result;

void test(void)
{
    result = (unsigned long)(DFLASH_START_PTR + 2047uL);  // correct
    result = (unsigned long)(DFLASH_START_PTR + 2048uL);  // incorrect
}
```

Workaround

Use 32bit arithmetic:

```
    result = (unsigned long)(DFLASH_START_PTR) + 2048uL;
```

No. G64	**Bit Access generated although Keyword '__no_bit_access' was used**

Details

The compiler doesn't take care on the keyword __no_bit_access in pointer definitions.Although a pointer is correctly defined using the keyword '__no_bit_access', the compiler generates a bit access. For some I/O registers this causes an illegal I/O register access.

Example

```
volatile unsigned short __no_bit_access  v1;
volatile unsigned short __no_bit_access* ptr1 = &v1;

void test (void)
{
    *ptr1  = 0x0123U;
    *ptr1 |= 0x4000U;
}
```

Workaround

Use direct access instead of indirect pointer access

```
void workaround (void)
{
    v1  = 0x0123U;
    v1 |= 0x4000U;
}
```

No. G65	**Wrong indirect post Increment of a Result of a post Increment**

Details

Independent of the selected optimization level the compiler generates wrong code for the indirect post increment of a result of a post increment. This issue only occurs in the DLIB runtime is used.

Example

```
#include <stdio.h>
#include <assert.h>

char   c[2] = {'a','b'};
char *pc[2] = {&c[0],&c[1]};
char **ppc  = &pc[0];

int test(void)
{
    char cc_ret;
    cc_ret = *(*ppc++)++;
    assert(pc[0]==pc[1]);
    return (int)cc_ret;
}
```

Workaround

Use separate statements for post increment:

```
int workaround (void)
{
   …
   cc_ret = *(*ppc);    /* problem */
   (*ppc)++;
   ppc++;
   …
}
```

No. G66	**Unclear Description of Parameter Passing for Structure Types in Compiler Manual**

IAR Reference: EW24225

Details

At page 108 of the RL78 C/C++ Compiler Reference Guide (2nd Edition) parameter passing to function is described. It is described that structure types parameters are passed via stack except the size is 1,2,4 and 4 bytes:

```
Structure types: struct, union, and classes, except structs and unions
of sizes 1, 2, and 4
```

This is correct, but additionally the structure type element must be word aligned. The alignment of the element is defined by the data type of the largest member.

Example

```
typedef struct  {
  unsigned char e1;
  unsigned char e2;
} s1_TYPE;
```

The above structure is passed via stack as only byte aligned elements are included.

Workaround

Include the structure type element in a union to force word alignment:

```
typedef union  {
  struct {
  unsigned char e1;
  unsigned char e2;
  };
  unsigned short dummy;
} s1_TYPE;
```

No. G67	**Internal Error in case of similar Function in ‘switch’ and ‘if’ Node**

IAR Reference: EW24227

Details

An internal error is generated in some cases, when several similar function calls exist in many "switch" and "if" nodes and optimization levels ‘high size’ and ‘high balanced’ are used.

Example

Due to complexity the sample is not listed here. It is available on request at Renesas Software-Tool-Support Team.

Workaround

Choose optimization level ‘high speed’ or medium

No. G68	**Unnecessary Padding Byte added to Arrays of Character**

IAR Reference: EW24453

Details

Alignment of arrays is set to two by the 78K0R compiler even if they are placed at an absolute location

Example

__root const char array[3] @0x08000= {0x01,0x02,0x03};

Compiler list file:

```
\                                 In  segment NEAR_CONST, align 2, root
   3          __root const char arr[3] = {0x01,0x02,0x03};
\                        arr:
\   000000 01020300              DB 1, 2, 3, 0
```

For the 78K0R compiler, string literals and arrays always have an alignment of two, unless placed at an absolute address. Note: This will cause padding for odd-sized objects.

Workaround

None.

No. G70	**Wrong Code generated while Copying a 1-Bit Bitfield**

IAR Reference: EW24645

Details

Assigning a value from one 1-bit bitfield to another 1-bit bitfield can fail if the byte offset of the bitfield in the struct is not zero and an optimization level medium or higher is used.

Example

```
typedef struct
{
 unsigned long u32var1;
 unsigned char u1var6_1:1;
 unsigned char u1var6_2:1;
 unsigned char u1var6_3:1;
 unsigned char u1var6_4:5;
}s1_T;

void test(s1_T * in, s1_T * out)
{
  out->u1var6_1 = in->u1var6_1;
  out->u1var6_2 = in->u1var6_2;
  out->u1var6_3 = in->u1var6_3;
  out->u1var6_4 = in->u1var6_4;
}
```

Workaround
Lower optimization level to medium or low.

No. G71	**MISRA C 2004 Rule 10.6 not triggered**

IAR Reference: EW24733

Details

The compiler does not check MISRA-C 2004 rule 10.6 correctly. It bases the check on the usage of the constant instead of on the type of the constant.

Example:

```
#define UNSIGNED_CHAR_C  0x12
#define UNSIGNED_SHORT_C 0x1234
#define UNSIGNED_LONG_C  0x12345678

unsigned char  var1 = UNSIGNED_CHAR_C;  /* Error [Pm127]: */
unsigned short var2 = UNSIGNED_SHORT_C; /* no error MISRA C 2004 */
unsigned long  var3 = UNSIGNED_LONG_C;  /* no error MISRA C 2004 */
```

In above example error Pm127 should be triggered three times instead of only one.

Workaround
None; it will be fixed in next update.

No. G72	**Stack Content can be corrupted by ISR**

IAR Reference: EW24895

Details

Due scheduling error in the optimizer, the stack content can be corrupted if stack is used for temporary storage in a function and an interrupt occurs also using temporary storage

Example:

In below sample the address of data located on stack is stored in register HL to access it indirectly. Due to the error the stack pointer is modified to free the stack size before the last access to the data is finished. If now an interrupt using stack area occurs between modification of stack pointer and data processing, the data is corrupted:

```
\   00003D 16                   MOVW      HL, AX             ;; 1 cycle
\   00003E 710103               MOV1      S:0xFFF03.0, CY    ;; 2 cycles
\   000041 A7                   INCW      HL                 ;; 1 cycle
\   000042 1002                 ADDW      SP, #0x2           ;; 1 cycle
```

If an interrupt using stack memory occurs here, data used in the next indirect memory access are corrupted:

```
\   000044 71B4                 MOV1      CY, [HL].3         ;; 1 cycle
\   000046 710103               MOV1      S:0xFFF03.0, CY    ;; 2 cycles
```

The correct code should be:

```
\   000040 16                   MOVW      HL, AX             ;; 1 cycle
\   000041 A7                   INCW      HL                 ;; 1 cycle
\   000042 71B4                 MOV1      CY, [HL].3         ;; 1 cycle
\   000044 710103               MOV1      S:0xFFF03.0, CY    ;; 2 cycles
\   000047 1002                 ADDW      SP, #0x2           ;; 1 cycle
```

Workaround
Avoid optimization level high balanced and high speed.

No. G73	**Wrong Code generated for Array Index**

IAR Reference: EW25315

Details

Using an unsigned variable as index type can generate illegal indexes if the variable type is smaller than the pointer index type and optimization level ‘high’ is used.

Example:

```
const unsigned char id_tbl[2] = { 0x01, 0x02};
unsigned char id = 0x02;

int test( void )
{
  static unsigned char n;
  n = 2;
  while(n > 0)  {
    n--;
    if(id_tbl[n] == id) {
      break;
    }
  }
  return 0;
}
```

Workaround

Use a signed index variable: `static signed char n;`

O) Valid Specification

Item	Date published	Document No.	Document Title
1	May 2012	UIDEEW-4	78K IAR Embedded Workbench® IDE Project Management and Building Guide
2	May 2010	C78K-4	78K IAR C/C++ Compiler Reference Guide
3	May 2010	A78K-3	78K IAR Assembler Reference Guide
4	May 2009	M78K-3	78K IAR Embedded Workbench Migration Guide
5	October 2012	UCS78K-1	78K C-SPY Debugging Guide
6	June 2012	XLINK-540	IAR Linker and Library Tools Reference Guide
7	January 2011	EWMISRAC1998-4	IAR Embedded Workbench MISRA C 1998 Reference Guide
8	December 2009	EWMISRAC2004-2	IAR Embedded Workbench MISRA C 2004 Reference Guide

P) Revision

Edition	Date published	Document No.	Comment
1	05-07-2004	CESCN0004V10	First release.
2	26-10-2004	CESCN0004V11	Items A1, A2, C2, C3, D1 added
3	06-12-2004	CESCN0004V12	Items A3, A4, A5, B4, C4 added, EW78K version V4.20a
4	17-01-2005	CESCN0004V13	Items C5, D2, E1 added
5	11-02-2005	CESCN0004V14	Items C6, C7, C8 added
6	07-03-2005	CESCN0004V15	Items C9, C10 added
7	08-04-2005	CESCN0004V16	Items C11, D3, D4, D5, D6 added
8	20-04-2005	CESCN0004V17	Item C12 added
9	10-05-2005	CESCN0004V18	Item C13 added
10	27-05-2005	CESCN0004V19	Items C14, E2 added
11	01-06-2005	CESCN0004V20	Items C15, C16 added
12	22-07-2005	CESCN0004V21	Items C17, B2, D7, E3 added, EW78K version V4.30a
13	18-08-2005	CESCN0004V22	Items C18, C19, D8, D9, D10, E4 added
14	02-09-2005	CESCN0004V23	Items C20, C21, C22 added
15	13-09-2005	CESCN0004V24	Patch Update for Compiler V4.30c and Debugger V4.30b
16	13-10-2005	CESCN0004V25	Items D11, E5, E6, E7 added
17	26-10-2005	CESCN0004V26	Items E8, E9 added
18	14-11-2005	CESCN0004V27	Items E10, E11, E12,E13 added, Patch Update for C-SPY Debugger V4.30d

Edition	Date published	Document No.	Comment
19	01-12-2005	CESCN0004V28	Items E14, E15, E16 added
20	15-12-2005	CESCN0004V29	Patch Update for C-SPY Debugger V4.30e
21	13-01-2006	CESCN0004V30	Item E17 added
22	26-01-2006	CESCN0004V31	Items C23, C24 added
23	02-03-2006	CESCN0004V32	Items C25, E18 added
24	13-03-2006	CESCN0004V33	Items C26, E19, E20 added
25	15-03-2006	CESCN0004V34	Correction of table (C)
26	03-04-2006	CESCN0004V35	Items C27, E21,E22 added
27	13-04-2006	CESCN0004V36	Items A6, E23 added
28	09-06-2006	CESCN0004V37	Item C25 updated, items B3, C28, C29 added
29	11-07-2006	CESCN0004V38	Item C30 added, EW78K version V4.40a
30	20-07-2006	CESCN0004V39	Items A7, C31, C32, G1, G2 added
31	04-08-2006	CESCN0004V40	Items A8, A9, B4, B5, F3,F4 added
32	01-09-2006	CESCN0004V41	Items B4, A9, F3 updated, items C33, C34, D12, D13 added
33	07-09-2006	CESCN0004V42	Items D12, D13 updated
34	06-10-2006	U18447EE1V0IF00	Items C35, C36, D14, E24, G3, G4 added Items D12, D13 updated Items C1, C2, C3, C7, C8, D2 removed Patch Update for compiler ICC78K and ICC78K0R version V4.40b and for linker XLINK version 4.60c new NEC Electronics world-wide document number
35	23-10-2006	U18447EE2V0IF00	Items D15, E25, E26, G5 added
36	03-11-2006	U18447EE3V0IF00	Items C37, E27, E28, E29, G6 added
37	17-11-2006	U18447EE3V1IF00	Items D16, E30 added
38	23-11-2006	U18447EE3V2IF00	Items E31, E32 added, patch update for C-SPY V4.40c
39	15-12-2006	U18447EE3V3IF00	Items C38 , G7 , E33 added
40	02-02-2007	U18447EE3V4IF00	Items E34, E35 , F5, F6, added
41	27-02-2007	U18447EE3V5IF00	Items C39 , C40 , G8 , G9 added
42	09-03-2007	U18447EE3V6IF00	Item E36 added
43	14-05-2007	U18447EE3V7IF00	EW78K version V4.50a Items C4, C6, C9, C10, C11, C12, C13, C14, C15, C16, C17, E1 removed Items C41, D17, D18, G10 added
44	18-06-2007	U18447EE3V8IF00	Items C42 , C43, G11, F7 added, update of disclaimer, update of valid specification table
45	22-06-2007	U18447EE3V9IF00	Items G12, E37 added Items D1, D4, D5, D6 removed Linker update V4.60i

Edition	Date published	Document No.	Comment
46	09-07-2007	U18447EE4V0IF00	Compiler update V4.50b, C-SPY update TK78K V4.50b, Item E38 added
47	01-08-2007	U18447EE4V1IF00	Items E39 , G13 added
48	27-08-2007	U18447EE4V2IF00	Items C44, G14 added
49	28-09-2007	U18447EE4V3IF00	Items E40, G15 added
50	26-10-2007	U18447EE4V4IF00	Compiler update V4.50c Item E40 updated, Items A10, C45, G16 added
51	05-11-2007	U18447EE4V5IF00	Item C46 added
52	22-11-2007	U18447EE4V6IF00	Item E41 added
53	06-12-2007	U18447EE4V7IF00	Items C47 , G17 added
54	15-01-2008	U18447EE4V8IF00	Items C48 , G18 added
55	28-01-2008	U18447EE4V9IF00	Item C49 added
56	11-02-2008	U18447EE5V0IF00	Items C50 , G19 added
57	07-03-2008	U18447EE5V1IF00	Items C51 , E42, G20 added
58	17-04-2008	U18447EE5V2IF00	Items C52, G21, F8 added
59	05-05-2008	U18447EE5V3IF00	Items C53, D20 added
60	21-05-2008	U18447EE5V4IF00	Items C18-C28, C30, D7, E3,E4, E7, E10-E12 removed Embedded Workbench update EW78K V4.60a Item D20 corrected
61	12-06-2008	U18447EE5V5IF00	Item D21, F9 added
62	09-07-2008	U18447EE5V6IF00	Items C54, G22 added, items E8, E13, E15, E16 removed C-SPY Update V4.60b (support of new 78K0R/Ix3 series)
63	17-07-2008	U18447EE5V7IF00	Items E43, E44, F10 added
64	22-08-2008	U18447EE5V8IF00	Item A11 added, linker update V4.61h
65	15-09-2008	U18447EE5V9IF00	Items C55, C56, C57, E45, G23 added
66	21-10-2008	U18447EE6V0IF00	Items C58, E46, E47 added
67	15-12-2008	U18447EE6V1IF00	Assembler and compiler update V4.61a, Item C58 corrected, Items G1, G2, G3,G4 removed Item A12, A13, G24 added
68	19-01-2009	U18447EE6V2IF00	Items D22, ,E48, G25 added
69	28-01-2009	U18447EE6V3IF00	Items C59, E49 ,G26 added
70	13-02-2009	U18447EE6V4IF00	Items F11, C60 added
71	02-03-2009	U18447EE6V5IF00	Items A14, E50, F12 added
72	09-03-2009	U18447EE6V6IF00	Items D23, D24 added, linker update V4.61l Items D8, D9, D10, D11, D20 removed
73	04-05-2009	U18447EE6V7IF00	Items C61, E51, G27 added
74	08-05-2009	U18447EE6V8IF00	Item G28 added
75	20-05-2009	U18447EE6V9IF00	Item G29 added

Edition	Date published	Document No.	Comment
76	02-07-2009	U18447EE6VAIF00	Update EW78K V4.62, Items A15, E52 added, Items A1, A3, B2, B4, C31…C36, C40, C41, E2, E5, E6, E9, E14, E17… E23, F3, G5, G8…G10 removed
77	07-07-2009	U18447EE6VBIF00	Item C62 added, compiler update V4.50e added
78	27-08-2009	U18447EE6VCIF00	Item E53 added, correction item C62: V4.60a affected
79	15-09-2009	U18447EE6VDIF00	Item D25 added, items D12, D13, D15 removed, linker update V4.61p
80	11-11-2009	U18447EE6VEIF00	Items C63, G30 added
81	13-11-2009	U18447EE6VFIF00	Item D26, D27 added, items D16, D17 removed, linker update V4.62r
82	23-11-2009	U18447EE6VGIF00	Items C63, G30 updated Items C64, G31 added Items C42, C43, C44, C47, G11, and G12 removed
83	26-11-2009	U18447EE6VHIF00	Items C65, G32 added, chapter ‘Valid Specification ‘updated
84	13-01-2010	U18447EE6VIIF00	Item A16, D28 added, linker update V4.61s added
85	02-02-2010	U18447EE6VJIF00	Items D29 and G33 added, item D18 removed Linker update V4.61t added Correction of item C56; compiler version V4.50c, v4.50e are not effected.
86	09-03-2010	U18447EE6VKIF00	Item G13, G14, G15 deleted Compiler Update patch V4.62.5 added
87	28-04-2010	R20UT0002ED0700	New company name, new document number Items C66, G34, G35 added
88	18-05-2010	R20UT0002ED0701	Linker Update 5.00.1 Item D30 added, item D14 removed
89	25-06-2010	R20UT0002ED0702	EW78K Update V4.70.1, Specification Update, Items C29, C37,C38, C39, C45, C46, C48, C49, C50, C51, C52, C53, E24, G6, G7, G16, G17, G18, G19, G20, G21 removed Item E54 added
90	09-08-2010	R20UT0002ED0703	Item A17 added Update of support email addresses
91	01-09-2010	R20UT0002ED0704	Items C67, C68, E55, E56, and G36 added
92	20-10-2010	R20UT0002ED0705	Items C69, C70, G37, G38, G39, and G40 added
93	15-11-2010	R20UT0002ED0706	Item C71 added, update items C70, G38, G40
94	12-01-2011	R20UT0002ED0707	Item G41 added
95	22-02-2011	R20UT0002ED0708	Item G42 added
96	11-04-2011	R20UT0002ED0709	Item G43, G44 added
97	16-05-2011	R20UT0002ED0710	Item C72, E57 added
98	05-07-2011	R20UT0002ED0711	EW78K update V4.71.1, Items A4, A5, A9, B3, B5, C54–C60, C62, D19, D21, E31-E32, F2-F6, G22-24, and G32 removed, items E58 and G45 added

Edition	Date published	Document No.	Comment
99	20-07-2011	R20UT0002ED0712	Items E59, G46, and G47 added
100	16-08-2011	R20UT0002ED0713	EW78K update V4.71.2 Item E28, E29, E33, E36, G25-G29 removed, Items G48 and F13 added, G46 updated Link to current document version changed.
101	13-09-2011	R20UT0002ED0714	Item F13 updated, items G49 and G50 added
102	13-10-2011	R20UT0002ED0715	Items C73, E60, G51, G52, G53 , G54 and G55 added; items E34 and G44 updated.
103	23-02-2012	R20UT0002ED0716	Items B2, D31, G56 and E61 added; items D22, D23, D24, D25 removed
104	27-02-2012	R20UT0002ED0717	Item G57 added
105	03-04-2012	R20UT0002ED0718	Item D32 added New Renesas Order Codes since 01.04.2012
106	20-04-2012	R20UT0002ED0719	Items C74, C75 and G58 added, G44 updated
107	05-07-2012	R20UT0002ED0720	Items C76 and G59 added, item C72 updated (issue may also occur in standard memory model
108	01-08-2012	R20UT0002ED0721	Items C77, C78, and E62 added
109	06-08-2012	R20UT0002ED0722	Incorrect issue numbers used: Items G60 (instead of C77) and G61 (instead of C78) added
110	31-10-2012	R20UT0002ED0723	EW78K Update V4.80.1 Items A6,A7,A8, C61, E38 and G40 removed Description of item G60 corrected
111	11-03-2013	R20UT0002ED0724	Item G62 and G63 added
112	03-04-2013	R20UT0002ED0725	XLINK update V5.6.0.36, item D33 added, item G46 updated, items D26 and D27 removed, previous Renesas order codes removed
113	15-05-2013	R20UT0002ED0726	Items C77, G64 added
114	11-06-2013	R20UT0002ED0727	Item G65 added
115	16-07-2013	R20UT0002ED0728	Item F14 added
116	18-10-2013	R20UT0002ED0729	Items C78 , G66 and G67 added
117	03-02-2014	R20UT0002ED0730	Items G68 and D34 added
118	14-02-2014	R20UT0002ED0731	Items A18, C79 and G69 added
119	12-05-2014	R20UT0002ED0732	Update SP-EW78K V4.80.2 Item G70 added, items C63, C65, C70, D28, F7-F10, G30, G31, and G33 removed
120	21-05-2014	R20UT0002ED0733	Item C80 added
121	07-08-2014	R20UT0002ED0734	Items C81, G71 and G72 added
122	23-02-2015	R20UT0002ED0735	Item E63 added
123	07-04-2015	R20UT0002ED0736	Items C82 and G73 added
124	26-05-2015	R20UT0002ED0737	Update SP-EW78K V4.80.3 Update item G68 Items C64, E25-E27, E30, E35, E37, E39, E40-E42, G34, G35, G60, and G69 removed.

Before using this material, please visit our website to confirm using the most current document available:
Current version of this document

In case of any technical question related to the Embedded Workbench for 78K, please feel free to contact the Renesas Software-Tool-Support Team

Renesas Electronics Europe

R20UT0002ED0737
May 2015