Customer Notification

EW78K-xxx-EE

Embedded Workbench® for 78K Integrated Development Environment

Operating Precautions

EW78K-FULL-EE
EW78K-KS16-EE
EW78K-KS4-EE

Document No. U18447EE6VBIF00
Date Published: July 2009

Table of Contents

(A) Table of Operating Precautions for the IDE EW78K

No.	Outline	EW78K					
	Version	4.3a	4.4b	4.6b	4.8a	5.2d	5.5.0
A2	An empty Workspace can not be saved	✗	✗	✗	✗	✗	✗
A4	Supported Path Length is limited	✗	✓	✓	✓	✓	✓
A5	Source Files can not be added directly to a user defined Group	✗	✓	✓	✓	✓	✓
A6	Project Files using output file paths containing illegal drive letters can not be opened	✗	✗	✗	✓	✓	✓
A7	Wrong Definition of RAM segments in XCL-file templates for 78K0S/Kx1+ devices	✓	✓	✗	✓	✓	✓
A8	Code Banking Information must be modified by the User	-	-	✗	✓	✓	✓
A9	Corrupt Default-Values for Near constant location Definition	-	-	✓	✓	✓	✓
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 is not be generated	✓	✓	✗	✗	✗	✓
A14	Empty Go to Function Window	-	-	✗	✗	✗	✗
A15	Corrupted Default-File Filter	✓	✓	✓	✓	✗	✓

✗:	Applicable
✓:	Not applicable
- :	Not checked

(B) Table of Operating Precautions for the Assembler A78K

No.	Outline	A78K					
	Version	4.30a	4.40a	4.50a	4.60a	4.61a	4.62a
B1	RSEG Directives can not be used in Macro Definitions	✗	✗	✗	✗	✗	✗
B3	It is not possible to use an assembler DEFINE to an external symbol	✗	✗	✓	✓	✓	✓
B5	EVEN Directive doesn't align Data to even Address.	✗	✗	✓	✓	✓	✓

✗:	Applicable
✓:	Not applicable
- :	Not checked

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

No.	Outline	ICC78K					
	Version	4.50b	4.50c	V4.50e	4.60a	V4.61a	V4.62a
C5	No compiler message in case of a variable redefinition of the same datatype but with the different object attribute	✗	✗	✗	✗	✗	✗
C29	No message about MISRA Rule 1 violation	✗	✗	✗	✓	✓	✓
C37	Warning [Pe177] generated by fault	✗	✗	✗	✓	✓	✓
C38	Fatal error in case of using experimental option –mfc	✗	✗	✗	✓	✓	✓
C39	Fatal error in case of using C++-style definition of a local variable	✗	✗	✗	✓	✓	✓
C42	Internal compiler error for bit-test of array element (78K0S core only)	✓	✓	✓	✓	✓	✓
C43	Internal compiler error in case of using a default segment name for a user-defined segment	✓	✓	✓	✓	✓	✓
C44	Register-bank selection of interrupt function may be ignored	✗	✓	✓	✓	✓	✓
C45	Internal Compiler Error may occur if calculation result is zero	✗	✗	✗	✓	✓	✓
C46	Internal Compiler Error may occur if instruction DBNZ is used	✓	✗	✗	✓	✓	✓
C47	Internal Compiler Error occurs if bit complement and bit-and operation are combined in one command	✗	✓	✓	✓	✓	✓
C48	Wrong code generated for access to multi-dimensional array	✗	✗	✗	✓	✓	✓
C49	Compilation process can not be completed	✓	✗	✗	✓	✓	✓
C50	Spurious linker warning about conflicting data types	✗	✗	✗	✓	✓	✓
C51	Extended EC++: Instantiating a template class may cause an internal error	✗	✗	✗	✓	✓	✓
C52	Wrong code may be generated if the intrinsic function '__get_interrupt_state' is used	✗	✗	✗	✓	✓	✓
C53	Internal Compiler Error occurs if second instruction parameter is a SFR-address	✓	✗	✗	✓	✓	✓
C54	Fatal Error (Uncontrolled termination) occurs if option –Ohs is used	✓	✓	✓	✗	✓	✓
C55	MISRA C 2004 Rule 17.4 triggered by mistake	✓	✓	✓	✗	✓	✓
C56	Banked Memory Model: Stack corrupted by wrongly generated code	✗	✗	✗	✗	✓	✓
C57	Banked Memory Model: Function Parameter not set	✓	✓	✓	✗	✓	✓
C58	Wrong parameter passing to library function for signed 32bit comparison	✓	✓	✓	✗	✓	✓
C59	Internal Compiler error in functions using an endless loop	✗	✗	✗	✗	✓	✓
C60	Wrong code generated for masking a bit of 16bit-high byte	✓	✗	✗	✓	✓	✓

No.	Outline	ICC78K					
	Version	4.50b	4.50c	V4.50e	4.60a	V4.61a	V4.62a
C61	**Missing Warning about change of sign due to integer conversion**	✗	✗	✗	✗	✗	✓
C62	**Usage of uninitialized carry-flag**	✗	✗	✓	✓	✓	✓

✗:	Applicable
✓:	Not applicable
- :	Not checked

(D) Table of Operating Precautions for the Linker XLINK

No.	Outline	XLINK								
	Version	4.60a	4.60c	4.60f	4.60g	4.60i	4.61c	4.61h	4.61l	4.61n
D3	**Breakpoint cannot be defined in Function (only XCOFF78K Format)**	✗	✗	✗	✗	✗	✗	✗	✗	✗
D12	**Memory Bank Area is not filled up**	✗	✓	✓	✓	✓	✓	✓	✓	✓
D13	**Corrupted IRQ table for 78K0R devices in case of using the XCOFF78K output format**	✗	✓	✓	✓	✓	✓	✓	✓	✓
D14	**Due to a wrong symbol definition XLINK error message [e149] is generated**	Please have a look at item C36 or G4								
D15	**Unused odd address isn't filled up**	✗	✓	✓	✓	✓	✓	✓	✓	✓
D16	**Range Error occurred by mistake**	✗	✗	✓	✓	✓	✓	✓	✓	✓
D17	**Error in 78K0R xcl-file (lnk78f11xx_xx.xcl) templates**	✗	✗	✓	✓	✓	✓	✓	✓	✓
D18	**Missing information for far pointer in XCOFF78K output format**	✗	✗	✗	✗	✓	✓	✓	✓	✓
D19	**Spurious linker warning about type conflict**	Please have a look at item C50 or G19								
D21	**Output file format UBROFF5: Error [e62] is generated erroneously if multiple modules are defined in one assembler source file**	✓	✓	✓	✓	✓	✗	✓	✓	✓
D22	**Output file format IEEE695: Missing enum datatype debug information**	✗	✗	✗	✗	✗	✗	✗	✓	✓
D23	**Output file format XCOFF78K: Usage of untyped segments may cause a corrupted file**	✓	✓	✓	✓	✓	✗	✗	✓	✓
D24	**Output file format RAW-BINARY: XLINK may hang up**	✓	✓	✓	✓	✓	✓	✗	✓	✓

✗:	Applicable
✓:	Not applicable
- :	Not checked

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

No.	Outline	CS78K							
	Version	4.40a	4.40b	4.40c	4.50a	4.50b	4.60a	4.60b	4.62a
E24	C-SPY Driver for 'IECUBE': Real-time Memory Window Update interrupts application	✗	✓	✓	✓	✓	✓	✓	✓
E25	Starting C-SPY by command line: Wrong Simulator started	✗	✗	✗	✗	✗	✓	✓	✓
E26	Starting C-SPY by command line: C-SPY driver for 'IECUBE' crashes in case of using 78K0R emulator	✗	✗	✗	✗	✗	✓	✓	✓
E27	Event-Breakpoint is deleted incompletely	✗	✗	✗	✗	✗	✓	✓	✓
E28	C-SPY Driver for 'MINICUBE': Wrong display of main clock source of QB-78K0MINI-EE	✗	✗	✗	✓	✓	✓	✓	✓
E29	C-SPY Driver for 'IE-78K': C-SPY fatal error in case of illegal SFR access	✗	✗	✗	✓	✓	✓	✓	✓
E30	C-SPY Driver for 78K0R 'IECUBE' or 'MINICUBE': Fatal error after selecting 'SFR' in disassembly window	✗	✗	✗	✗	✗	✓	✓	✓
E31	C-SPY Driver for 78K0R 'IECUBE' or 'MINICUBE': Memory read access by macro is blocked	✗	✗	✓	✓	✓	✓	✓	✓
E32	Code Coverage information is incomplete in case of using banked memory systems	✗	✗	✓	✓	✓	✓	✓	✓
E33	C-SPY Driver for 78K0R 'MINICUBE': The input field for the main clock source allows only selecting a value from a predefined list.	✗	✗	✗	✓	✓	✓	✓	✓
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	✗	✗	✗	✗	✗	✓	✓	✓
E35	Argument variables can not be used to define a code breakpoint by source location	✗	✗	✗	✗	✗	✓	✓	✓
E36	C-SPY Driver for 78K0R Simulator: Instruction 'mov memory_location[C],A' simulated incorrectly	✗	✗	✗	✓	✓	✓	✓	✓
E37	C-SPY Driver for 78K0R IECUBE: OP-Fetch before execution can not be defined	✗	✗	✗	✗	✗	✓	✓	✓
E38	C-SPY Driver or TK78K: Download to memory banks failed	✗	✗	✗	✗	✓	✓	✓	✓
E39	C-SPY Driver for 78K0R: High Byte of Program Counter (bit16-bit23) is set to 0x00	✗	✗	✗	✗	✗	✓	✓	✓

No.	Outline	CS78K							
	Version	4.40a	4.40b	4.40c	4.50a	4.50b	4.60a	4.60b	4.62a
E40	**C-SPY Driver for 78K0R: A file in Intel-Hex- or Motorola-S-Record format can not be downloaded**	✗	✗	✗	✗	✗	✓	✓	✓
E41	**C-SPY Simulator Driver: Wrong mask-flag is used to control an interrupt**	✗	✗	✗	✗	✗	✓	✓	✓
E42	**C-SPY 78K0 IECUBE Driver: Full trace break doesn't work**	✗	✗	✗	✗	✗	✓	✓	✓
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**	✓	✓	✓	✓	✓	✗	✗	✓

✗: Applicable ✓: Not applicable **-** : Not checked

(F) Table of Operating Precautions for the Assembler A78K0R

No.	Outline	A78K0R					
	Version	4.40a	4.50a	4.60a	4.61a	4.62a	
F1	**RSEG Directives can not be used in Macro Definitions**	✗	✗	✗	✗	✗	
F2	**It is not possible to use an assembler DEFINE to an external symbol**	✗	✓	✓	✓	✓	
F4	**EVEN Directive doesn’t align Data to even Address.**	✗	✓	✓	✓	✓	
F5	**Automatic Replacement of DBNZ Instruction causes Linker Error Message**	✗	✓	✓	✓	✓	
F6	**Invalid Register in XCH Instruction causes the generation of wrong Op-Code**	✗	✓	✓	✓	✓	
F7	**Invalid XCH instruction doesn’t cause a syntax error**	✗	✗	✓	✓	✓	
F8	**Wrong Op-Code generated for *MOV <register>, SFR-address* instruction**	✗	✗	✓	✓	✓	
F9	**Illegal MOV instruction is accepted and wrong Op-Code is generated.**	✗	✗	✗	✓	✓	
F10	**Invalid operand of branch instruction causes fatal assembler error**	✗	✗	✗	✓	✓	
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**	✗	✗	✗	✗	✓	

✗:	Applicable
✓:	Not applicable
- :	Not checked

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

No.	Outline	ICC78K0R					
	Version	4.50a	4.50b	4.50c	4.60a	V4.61a	V4.62a
G6	Warning [Pe177] generated by fault	✗	✗	✗	✓	✓	✓
G7	Fatal error in case of using experimental option –mfc	✗	✗	✗	✓	✓	✓
G11	Internal compiler error occurs if a default segment name is used for a user-defined segment	✗	✓	✓	✓	✓	✓
G12	Wrong access to far and byte-aligned structure	✗	✓	✓	✓	✓	✓
G13	Wrong code generated for indirect memory access	✗	✗	✓	✓	✓	✓
G14	Register-bank selection of interrupt function may be ignored	✗	✗	✓	✓	✓	✓
G15	Wrong access to local variable located on stack	✗	✗	✓	✓	✓	✓
G16	Internal Compiler Error may occur if calculation result is zero	✗	✗	✗	✓	✓	✓
G17	Internal Compiler Error occurs if bit complement and bit-and operation are combined in one command	✗	✗	✗	✓	✓	✓
G18	Wrong code generated for access to multi-dimensional array	✗	✗	✗	✓	✓	✓
G19	Spurious linker warning about conflicting data types	✗	✗	✗	✓	✓	✓
G20	Extended EC++: Instantiating a template class may cause an internal error	✗	✗	✗	✓	✓	✓
G21	Internal Compiler Error occurs if numeric constant is used as function pointer	✗	✗	✗	✓	✓	✓
G22	Fatal Error (Uncontrolled termination) occurs if option –Ohs is used	✓	✓	✓	✗	✓	✓
G23	MISRA C 2004 Rule 17.4 triggered by mistake	✓	✓	✓	✗	✓	✓
G24	DLIB Floating Point Function overwrites SADDR area	✗	✗	✗	✗	✓	✓
G25	Misaligned structure access	✓	✓	✓	✓	✗	✓
G26	Wrong parameter passing of far pointer	✗	✗	✗	✗	✗	✓
G27	Missing Warning about change of sign due to integer conversion	✗	✗	✗	✗	✗	✓
G28	Delayed insertion of DI instruction	✗	✗	✗	✗	✗	✓
G29	Misaligned 16bit-access	✗	✗	✗	✗	✗	✓

✗: Applicable ✓: Not applicable - : Not checked

(H) Description of Operating Precautions for the IDE EW78K

No. A2	**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. A4	**Supported Path Length is limited**
	Details The supported path length to project files by the Embedded Workbench is limited to about 100 characters. If this limit is exceeded the project file cannot be opened anymore and you received the following error message: Cannot find the file 'C:\...\...\<projectname.ewp>' (or one of its components). Make sure the path and filename are correct and that all required libraries are available. _Workaround_ Reduce the path length.

No. A5	**Source Files can not be added directly to a user defined Group**
	Details Source files can not be added directly to a user defined group, because the in the ‘Add Files…’ dialogue there is no field to specify a group. _Workaround_ Select the group in the Workspace-Window before open the ‘Add Files…’ dialogue. The files will be added automatically to the selected group.

No. A6	**Project Files using output file paths containing illegal drive letters can not be opened**

Details

Specifying absolute output directories with paths containing illegal drive letters caused IAR Embedded Workbench to exit without any further message.

Workaround

Before it is possible to open a project-file (*.ewp) using path with invalid drive letters, the project file has to be corrected manually with an editor of your choice:

```
<name>ExePath</name>
  <state>Debug\Exe</state>
</option>
<option>
  <name>ObjPath</name>
  <state>Debug\Obj</state>
</option>
<option>
  <name>ListPath</name>
  <state>Debug\List</state>
</option>
```

No. A7	**Wrong Definition of RAM segments in XCL-file templates for 78K0S/Kx1+ devices**

Details

In the XCL-file templates for the 78K0S/Kx1+ devices included in version V4.40a of the Embedded Workbench for 78K, the address-definition for the RAM segments is wrong.
If you use this templates, the following a linker error will occur:

```
Fatal Error[e140]: The range declaration used in
-Z(DATA)NEAR_I,NEAR_Z,NEAR_N,HEAP+_HEAP_SIZE=FE80-FE1F is illegal since 0xfe80 > 0xfe1f.

Fatal! Execution terminated...
```

The XCL-file templates for the following devices are effected:

```
Device         XCL-file template
µPD78F9200    lnk78f9200.xcl
µPD78F9201    lnk78f9201.xcl
µPD78F9202    lnk78f9202.xcl
µPD78F9210    lnk78f9210.xcl
µPD78F9211    lnk78f9211.xcl
µPD78F9212    lnk78f9212.xcl
µPD78F9221    lnk78f9221.xcl
```

Workaround

Please correct the above listed XCL-file templates as follows:

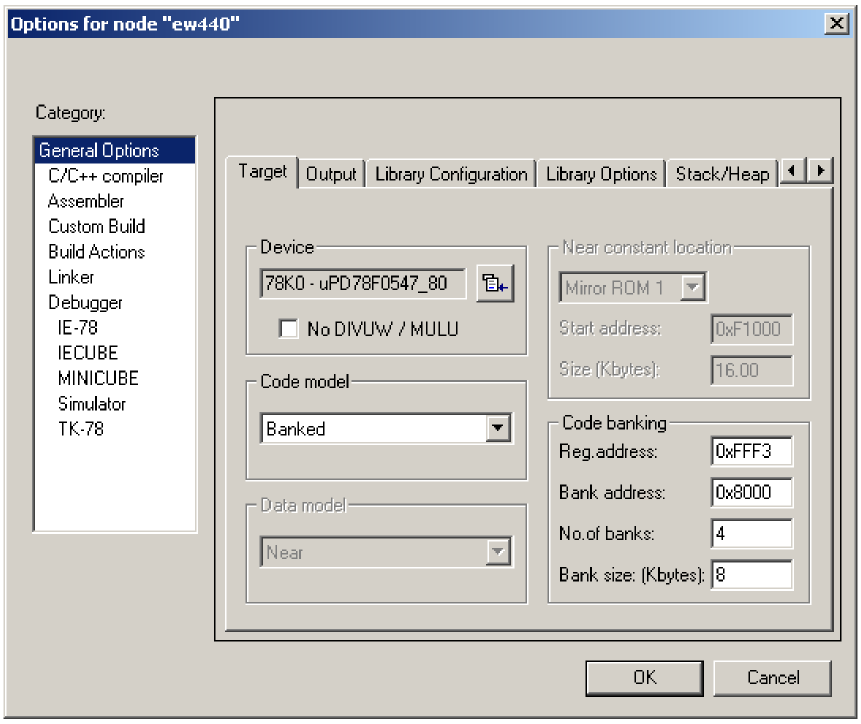
```
//--------------------------------------------------------------------
-
//      Near data and heap segments.
//--------------------------------------------------------------------
-
-Z(DATA)NEAR_I,NEAR_Z,NEAR_N,HEAP+_HEAP_SIZE=FE80-FEFF

//--------------------------------------------------------------------
-
//      Stack segment.
//--------------------------------------------------------------------
-
-Z(DATA)CSTACK+_CSTACK_SIZE=FE80-FEFF
```

For the above listed devices the internal RAM area is smaller than the SADDR-area and therefore it is recommended to define all global variables as SADDR-variables to get short and fast code.

No. A8	Code Banking Information must be modified by the User

Details

If the memory model ‘Banked’ or ‘Standard allowing banking’ is selected, the Code Banking information must be modified by the user according to the selected device, although a specific device is selected from the device list. The Embedded Workbench uses default values for the Code Banking definition independent of the selected device.
Example:



Options for node "ew440"

Category: General Options

Target

Device: 78K0 - uPD78F0547_80
☐ No DIVUW / MULU

Code model: Banked

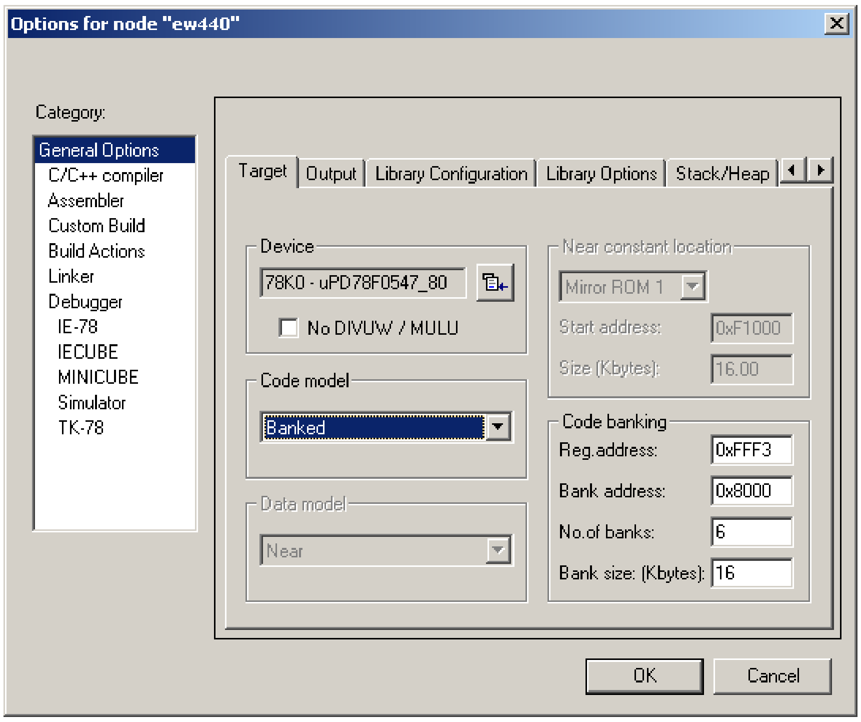
Data model: Near

Near constant location: Mirror ROM 1
Start address: 0xF1000
Size (Kbytes): 16.00

Code banking
Reg.address: 0xFFF3
Bank address: 0x8000
No.of banks: 4
Bank size: (Kbytes): 8

Workaround

Please enter the correct values according to the user’s manual of the selected device:



Options for node "ew440"

Category: General Options

Target

Device: 78K0 - uPD78F0547_80
☐ No DIVUW / MULU

Code model: Banked

Data model: Near

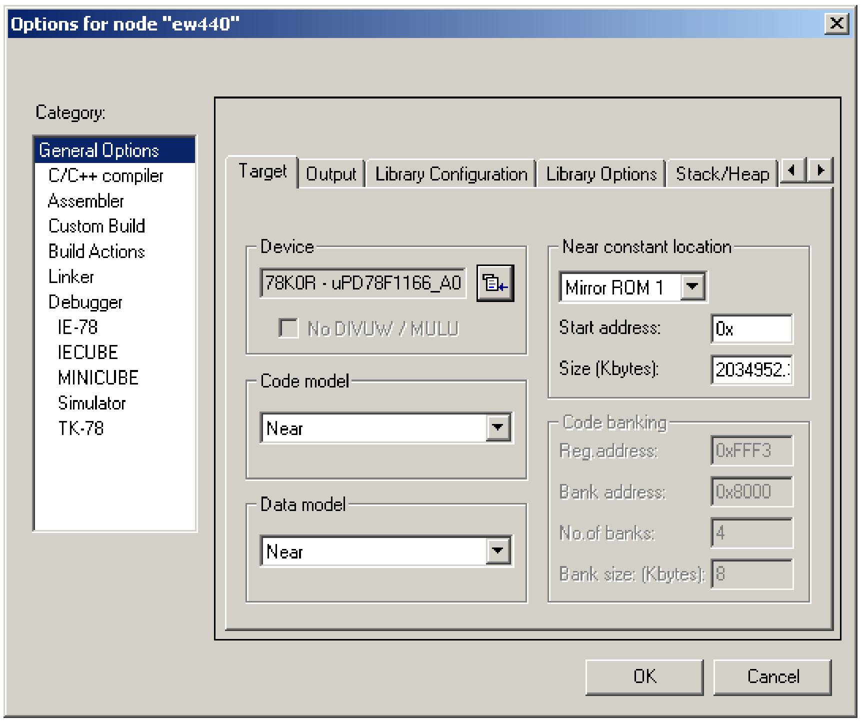
Near constant location: Mirror ROM 1
Start address: 0xF1000
Size (Kbytes): 16.00

Code banking
Reg.address: 0xFFF3
Bank address: 0x8000
No.of banks: 6
Bank size: (Kbytes): 16

No. A9	**Corrupt Default-Values for Near constant location Definition**

Details

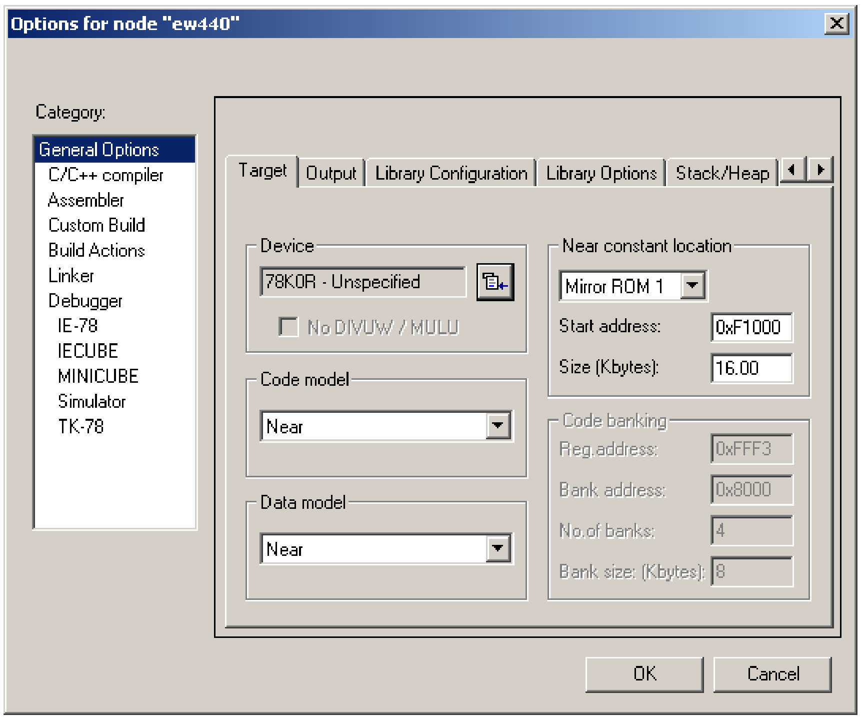
If a specific 78K0R is selected the, default values for the Near constant location are corrupt. Example:



Device: 78K0R - uPD78F1166_A0
No DIVUW / MULU
Code model: Near
Data model: Near
Near constant location: Mirror ROM 1
Start address: 0x
Size (Kbytes): 2034952.:
Code banking
Reg.address: 0xFFF3
Bank address: 0x8000
No.of banks: 4
Bank size: (Kbytes): 8

Workaround

Please select the unspecified '78K0R' device:



Device: 78K0R - Unspecified
No DIVUW / MULU
Code model: Near
Data model: Near
Near constant location: Mirror ROM 1
Start address: 0xF1000
Size (Kbytes): 16.00
Code banking
Reg.address: 0xFFF3
Bank address: 0x8000
No.of banks: 4
Bank size: (Kbytes): 8

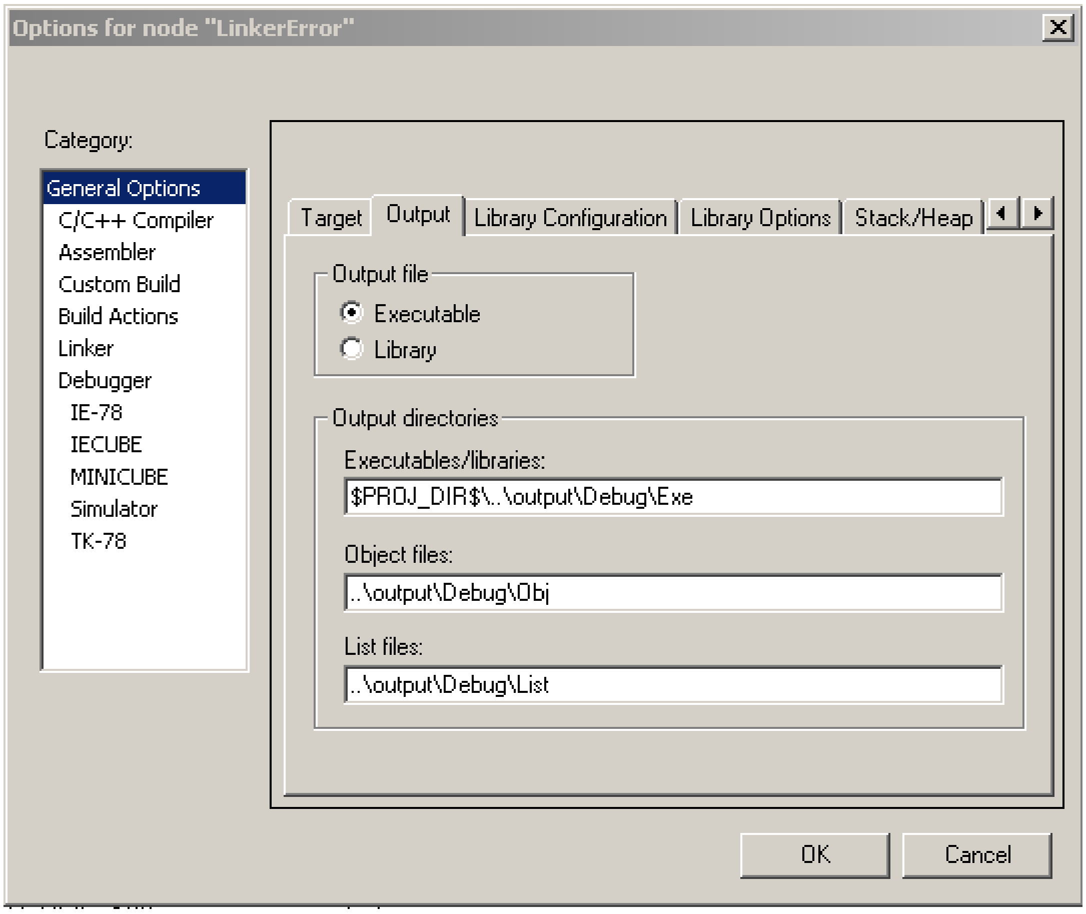
Additionally the further device configuration must be done manually:
XCL-file selection in menu `Project->Options->Linker->Config`
DDF- file selection in menu `Project->Options->Debugger`

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:



Options for node "LinkerError"

Category:
- General Options
- C/C++ Compiler
- Assembler
- Custom Build
- Build Actions
- Linker
- Debugger
  - IE-78
  - IECUBE
  - MINICUBE
  - Simulator
  - TK-78

Target | Output | Library Configuration | Library Options | Stack/Heap

Output file: [✓ Executable] Library

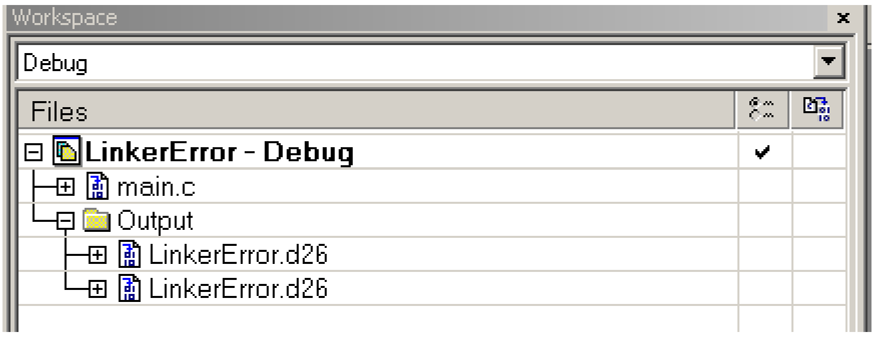
Output directories

Executables/libraries: $PROJ_DIR$\..\output\Debug\Exe

Object files: ..\output\Debug\Obj

List files: ..\output\Debug\List

OK | Cancel



Workspace

Debug

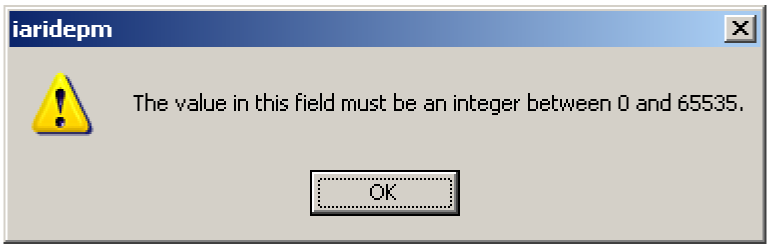
```
Files
⊟ LinkerError - Debug        ✓
  ├⊞ main.c
  └⊟ Output
     ├⊞ LinkerError.d26
     └⊞ LinkerError.d26
```

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:
	
	
	iaridepm
	The value in this field must be an integer between 0 and 65535.
	
	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 is not be generated**

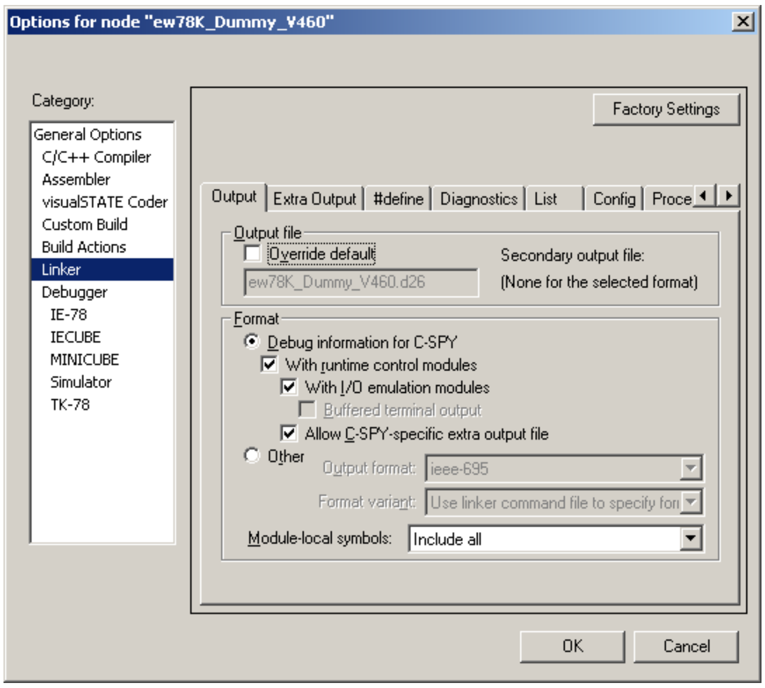
Details

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:



Options for node "ew78K_Dummy_V460"

Category: Linker

Output file: Override default [ ] — ew78K_Dummy_V460.d26

Secondary output file: (None for the selected format)

Format: Debug information for C-SPY [✓]

With runtime control modules [✓]

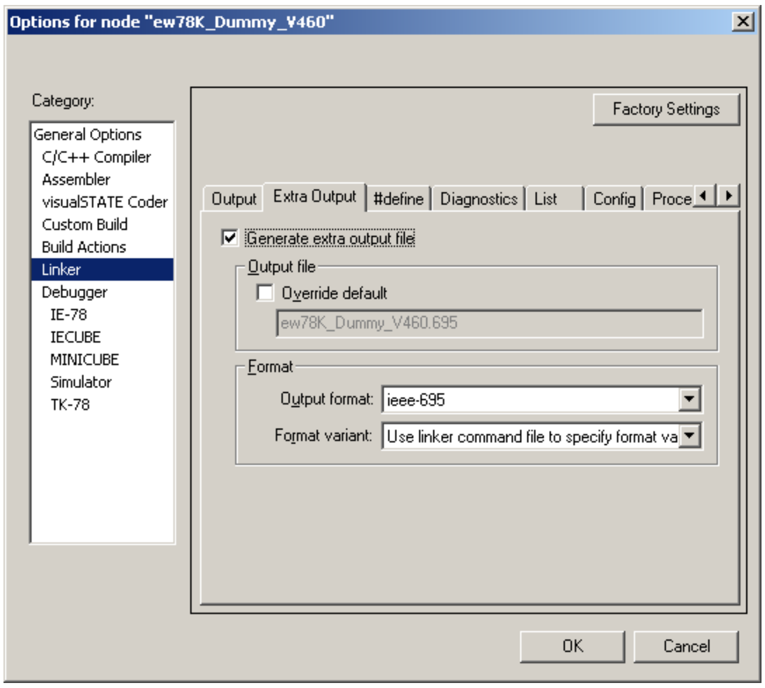
With I/O emulation modules [✓]

Buffered terminal output [ ]

Allow C-SPY-specific extra output file [✓]

Other [ ] — Output format: ieee-695; Format variant: Use linker command file to specify for

Module-local symbols: Include all



Options for node "ew78K_Dummy_V460"

Category: Linker — Extra Output

Generate extra output file [✓]

Output file: Override default [ ] — ew78K_Dummy_V460.695

Format: Output format: ieee-695

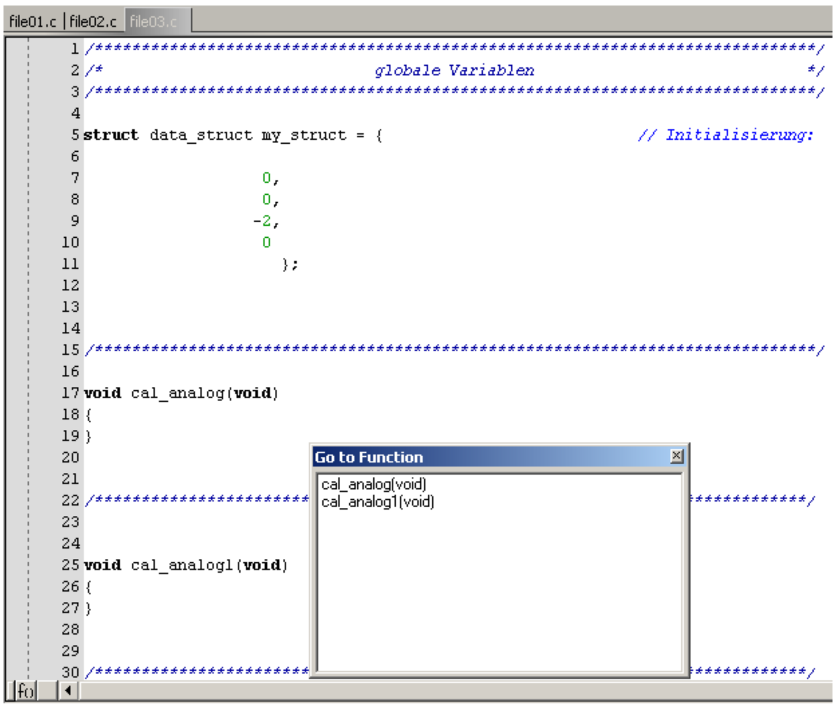
Format variant: Use linker command file to specify format va

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

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

Details

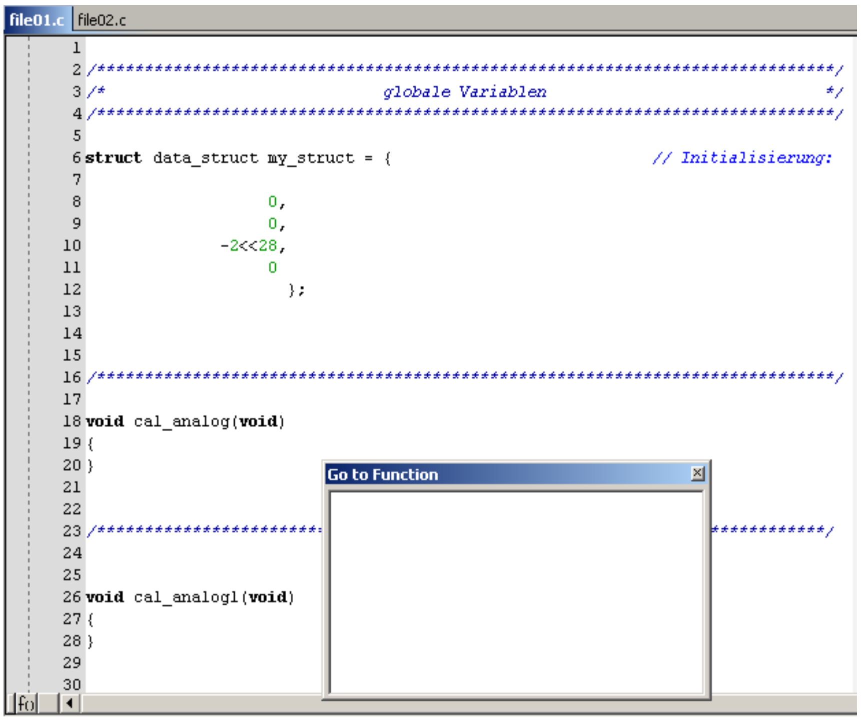
Depending on some source code constructions (e.g. using shift operator to initialize a structure element) the Go to Function Window may be empty.
Correct Go to Function Window:



```
/*************************************************************************/
/*                         globale Variablen                             */
/*************************************************************************/

struct data_struct my_struct = {                    // Initialisierung:

                     0,
                     0,
                    -2,
                     0
                       };



/*************************************************************************/

void cal_analog(void)
{
}


/***********************                               *************/


void cal_analog1(void)
{
}


/***********************                               *************/
```

```
cal_analog(void)
cal_analog1(void)
```

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



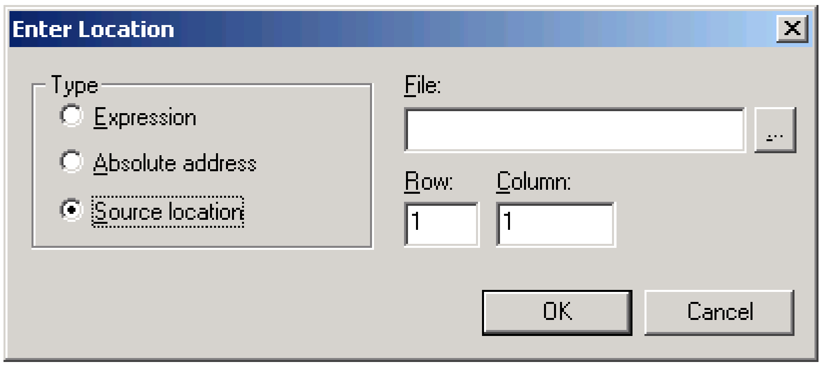
```

/*************************************************************************/
/*                         globale Variablen                             */
/*************************************************************************/

struct data_struct my_struct = {                    // Initialisierung:

                     0,
                     0,
                -2<<28,
                     0
                       };



/*************************************************************************/

void cal_analog(void)
{
}


/************************                              *************/


void cal_analog1(void)
{
}


```

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

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

Details

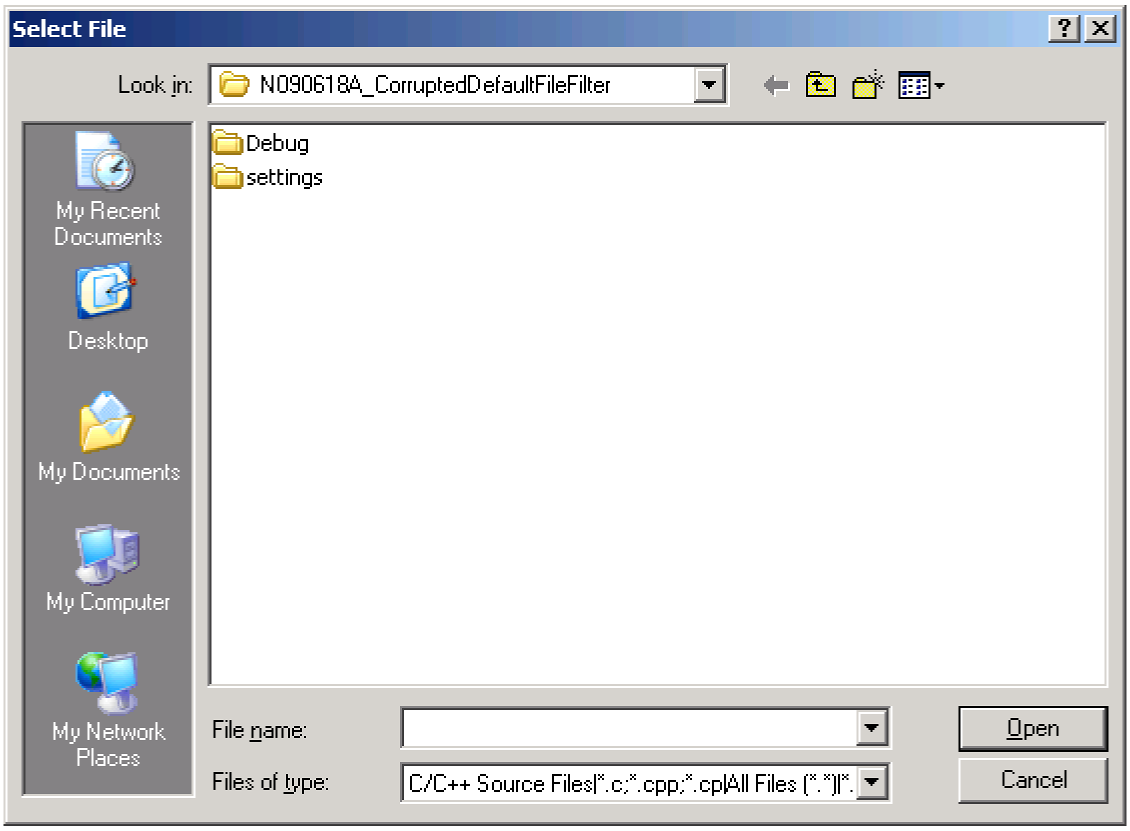
The default file filter of the C-Spy file selection dialogue after pressing the button '...'
of the code breakpoint 'Enter Location Window' is corrupted and therefore no files are listed although there are source files in the selected folder:


Enter Location

Type: Expression / Absolute address / [✓ Source location]

File:

Row: 1 Column: 1


Select File

Look in: N090618A_CorruptedDefaultFileFilter

Debug
settings

File name:

Files of type: C/C++ Source Files|*.c;*.cpp;*.cp|All Files (*.*)|*.

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

(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. B3 **It is not possible to use an assembler DEFINE to an external symbol**

Details

In case of using an assembler DEFINE to an external symbol, the linker will generate the following error:

```
Fatal Error[e20]: Corrupt file.  External index out of range in module
MODUL2 ( C:\....\test.r26 )
```

Example

```
    EXTERN S2

SYM DEFINE S2
```

Workaround

None. The assembler version V4.41a or later will generate an error for such cases.

No. B5 **EVEN Directive doesn't align Data to even Address.**

Details

The EVEN directive aligns to an even address relative to the start module-startaddress of the segment instead of an absolute even address. In case of an odd module-startaddress also all symbols aligned with an even-directive are located at an odd address. In this case a linker error message will be generated for each access to the misaligned variable:

```
     IAR Universal Linker V4.60A/386
     Copyright 1987-2006 IAR Systems. All rights reserved.

Error[e18]: Range error,   Even value expected
 File: H:\Data\...\even.asm, Line: 17
 Source:         MOVW  S:integer1, AX
 Where $ = test_even + 0x1  [0xA8]   in module "even",
       offset 0x1 in segment part 1, segment CODE
 What: (integer2 + 2) & 1 [0x1]
       Allowed range: 0x0 - 0x0   Operand: integer2 [0xfe23]
       in module even, Offset 0x2 in segment part 0, segment SADDR_Z
```

Example

```
    RSEG SADDR_Z

CharVar1:   DS 1
    ALIGNRAM 1
IntVar1:    DS 2
```

Workaround

Please align the segment-start address to an even address:

```
    RSEG SADDR_Z(1)

CharVar1:   DS 1
    ALIGNRAM 1
IntVar1:    DS 2
```

(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. C29	**No message about MISRA Rule 1 violation**

	Details
	
	MISRA C rule 1 concerns the ISO 9899 C conformance without extension meaning strict ANSI C.
	When a project is compiled with IAR extensions and the first MISRA C rule enabled no errors or warnings come up to indicate this contradiction.
	
	Example:
	
	`volatile __saddr int i;`
	
	`__callt void test(void)`
	`{`
	`  i++;`
	`}`
	
	Workaround:
	
	None.

No. C37 Warning [Pe177] generated by fault

Details

If a variable defined as '__root' is additionally defined as 'static', the compiler will generate the warning message [Pe177] by fault:

```
Warning[Pe177]: variable "test1" was declared but never referenced
```

The keyword '__root' informs the linker that the variable should be located even it is not referenced. This implies already that a variable might not be used in the module and that this declaration is done on purpose.

Example:

```
static __root const char test1= 0x01;
```

Workaround:

The problem will be fixed in the next major update. So far please use one of the following workarounds:

1) Don't define a variable as '`__root`' and '`static`'
2) Disable warning [Pe177] for such definitions:

```
    #pragma diag_suppress=Pe177
    static __root const char test1 = 0x01;
    #pragma diag_default=Pe177
```

No. C38 Fatal error in case of using experimental option –mfc

Details

If two static functions of the same name are exist in modules that are compiled simultaneously by using the currently experimental option –mfc, a fatal error occurs:

```
Internal Error: [CoreUtil/General]: OgModuleLables – label already
defined. Fatal error detected, aborting.
```

Example:

```
source file f1.c:
static unsigned char func1 (unsigned char p1)
{
    // code doesn't matter
    return (1);
}

source file f1.c:
static unsigned int func1 (unsigned int p1)
{
    // code doesn't matter
    return (1);
}
```

Workaround:

The problem will be fixed in the platform release, when the option –mfc will be officially introduced (V4.4xx, schedule is December 2007)

No. C39	**Fatal error in case of using C++-style definition of a local variable**

Details

If IAR Systems compiler extensions are enabled, it is allowed to define local variables directly before using them. But if such local variable is defined as static a fatal error is generated:

```
Internal Error: [symbol_lookup_M31]: symbol not found for mode 1
(backend generating) (P0: 0, P1: 0)
```

Example:

```
void test (void)
{
    for( static int i = 0; i<10; i++);
}
```

Workaround:

Define local variables according to the ANSI C standard at the beginning of a function.

```
void workaround (void)
{
    static int i;
    for( i = 0; i<10; i++);
}
```

No. C42	Internal compiler error for bit-test of array element (78K0S core only)

Details

In case of using a 78K0S device (µPD78F9xxxx, µPD789xxxx) pointer or array expressions of objects located in the short address area that result in a bit test instruction cause the following internal error for 78K0S.

```
Internal error [AsmLine – OgAsm]: Error [43] Illegal effective address
Fatal error detected aborting
```

Example:

```
        unsigned char v1,v2;
__saddr unsigned char buffer[5];

void test(void)
{
  if(buffer[v1]&0x80) {
      v2=1;
  }
}
```

Workaround:

Locate the array in the standard RAM, i.e. remove the key word __saddr:

```
        unsigned char v1,v2;
        unsigned char buffer[5];

void test(void)
{
  if(buffer[v1]&0x80) {
      v2=1;
  }
}
```

No. C43	Internal compiler error in case of using a default segment name for a user-defined segment.

Details

In case of using a default segment name of the compiler for user-defined segment of constant data, an internal compiler error occurs after the warning about using a default segment name.

```
Internal error [Front end]: Invalid C99 IL expression kind
Fatal error detected aborting.
```

Example:

```
#pragma location = "CODE"
__root const unsigned char counter=23;

void test(unsigned char *p1)
{
   *p1=*((volatile const unsigned char *)&counter);
}
```

Workaround:

Do not use the compiler default segment names for user-defined segments

No. C44	**Register-bank selection of interrupt function may be ignored**

Details

In case of using an optimization level higher than ‘low’ the compiler may ignore the register-bank selection of the user (#pragma bank) for some interrupt functions.

Example:

```
#include <io78f0893.h>
extern void  f2 (unsigned char );

typedef enum {
    GPT_1,
    GPT_2
}ENUM1;
typedef enum {
    GPT_3,
    GPT_5
} ENUM2;
typedef struct {
    ENUM1         s1;
    unsigned char s2;
    void*         s3;
}STRUCT1_T;
typedef struct{
        ENUM2           s4;
        unsigned short  s5;
} STRUCT2_T;


#pragma bank = 2
__interrupt void isr( void )
{
    unsigned short u16PR0sav, u16PR1sav;
    u16PR0sav = PR0 ;
    u16PR1sav = PR1 ;
    __enable_interrupt();

    if (ptr1[((unsigned char) 0)].s1 == GPT_1)   {
        array[((unsigned char) 0)].s4 = GPT_5;
    }
    f2( ((unsigned char) 0) );

    __disable_interrupt();
    PR0 = u16PR0sav;
    PR1 = u16PR1sav;

}
```

Workaround:

Please reduce the optimization level for the interrupt function, if the instruction ‘SEL RB2’ isn’t generated for your interrupt function:

```
#pragma optimize = s 3
#pragma bank = 2
__interrupt void isr( void )
{
 …
}
```

No. C45	**Internal Compiler Error may occur if calculation result is zero**

Details

Code examples where a calculation result is zero may cause an internal compiler error.

Example:

```
signed int i,k;
int test(void)
{
  k=90-(9-i)*10;
}
```

Workaround:

Try to rewrite the arithmetic expression to avoid a zero result:

```
int test (void)
{
  k=90-(90-10*i);
}
```

No. C46	**Internal Compiler Error may occur if instruction DBNZ is used**

Details

If the instruction DBNZ is used an internal compiler error may occur:

```
Internal Error: [CoreUtil/General]: Size mismatch for "DBNZ
S:v1, ??test_0", inserted as 2 bytes, assembled as 3 bytes.
```

Example:

```
__saddr unsigned char v1;

void test (void)
{
  if (!--v1){
    …
  }
}
```

Workaround:

Lower the optimization to level medium to avoid the usage of the instruction DBNZ

```
#pragma optimization = z 6
void test (void)
{
  …
}
```

No. C47	**Internal Compiler Error occurs if bit complement and bit-and operation are combined in one command**

Details

If the C command to complement a special functions register bit is combined with a bit and command to mask a single bit and an assignment to an integer variable, an internal error occurs:

```
Internal Error: [CoreUtil/General]: Illegal state
```

Example:

```
#include <io78F0547_80.h>

unsigned int IntVar;

void test(void)
{
  IntVar = ~P0_bit.no0 & 0x01;
}
```

Workaround:

Please split up the operations in separate lines of code.

```
unsigned int   IntVar;

void test(void)
{
  IntVar = ~P0_bit.no0;
  IntVar = IntVar & 0x01;

}
```

No. C48	**Wrong code generated for access to multi-dimensional array**

Details

In a case of using optimization type speed level high, the compiler may generate wrong code for the access of multi-dimensional arrays.

Example:

```
static void test (void)
{
    unsigned short x, y;

    for (y = 0; y < 8; y++){
        for (x = 0; x < 128; x++) {
            buffer[y][x] = 0x00;
        }
    }
}

void dummy( void )
{
   test();
}
```

Workaround:

Please reduce the optimization level to medium or use while instead of for loops.

No. C49 Compilation process can not be completed

Details

In a case of using optimization level high and allow the usage of the worksegment, the compilation process can not be completed for certain code examples. No error message is generated; the compilation process must be terminated manually.

Workaround:

Please reduce the optimization level to medium or don’t allow worksegment usage.

No. C50 Spurious linker warning about type conflict

Details

The compiler could in some cases (e.g. high level of nested typedef types) emit data type incorrect debug information for typedef types. When linking with XLINK, this could result in a spurious type conflict warning:

```
Warning[w6]: Type conflict for external/entry "<object-name>", in
module file2 against external/entry in module file1; different types
```

The generated code is correct.

Workaround:

Please reduce the level of nested typedef types.

No. C51 Extended EC++: Instantiating a template class may cause an internal error

Details

Instantiating a template class like vector on a function type may result in an internal error

```
Internal Error: [Visit types]: Error type
```

Example:

```
enum eState { state1,state2};

template <class T, T init> class CEnum
 {
 public:
   CEnum()                           {m_Value = init; }
   operator unsigned char () const   {return (unsigned char)m_Value; }
   void operator +=(unsigned char arg){m_Value = (T)(m_Value + arg) }
 private:
      T m_Value;
 };

static      __saddr __no_init CEnum<enum eState, state1> state;

void test(void)
{
                  state += state2;
}
```

Workaround:

None.

No. C52	**Wrong code may be generated if the intrinsic function '__get_interrupt_state' is used**

Details

If the intrinsic functions '__get_interrupt_state' and '__disable_interrupt' are used in the same function, the compiler may store the program status word (PSW) to a register before interrupts are disabled instead when the function' __get_interrupt_state' is called. The register content instead of the actual PSW content is used for further actions.

Example:

```
#include <intrinsics.h>

void test(void)
{
  __disable_interrupt();
  {
     istate_t is = __get_interrupt_state();
     __enable_interrupt();
     __set_interrupt_state(is);
  }
}
```

Workaround:

No direct workaround, but an assembler function can be used as replacement

No. C53	**Internal Compiler Error occurs if second instruction parameter is a SFR-address**

Details

For the instructions ADD, ADDC, SUB, SUBC, AND, OR, XOR and CMP, when the first parameter is register A and the second parameter is an SFR address above the SADDR memory area, the SFR address is treated as a SADDR address. This causes an internal compiler error:

```
Internal Error: [CoreUtil/General]:
Size mismatch for "…", inserted as 2 bytes, assembled as 3 bytes.
```

Example:

```
#include <io78f0515_48.h>

unsigned char test(unsigned int p1)
{
   unsigned char retVal = 0 ;

   while( (0 == IF1H ) &&  ( 0 != p1-- ) )   {
       p1--;
   }
   return (retVal);
}
```

Workaround:

None. The problem will be fixed in V4.60a

No. C54	**Fatal Error (Uncontrolled termination) occurs if option –Ohs is used**

Details

If the following sample is compiled by using option –Ohs a fatal error occurs:

```
Fatal Error[c0000005hìø_°a´_”°°_„ìø_ˆ°°_"]: Uncontrolled termination
```

In case of using WindowsXP the user is asked to inform Microsoft about this issue.

Example:

```
typedef struct
{
   unsigned char MyByte;
}T_MYSTRUCT;

extern void func1(unsigned char *, unsigned short , unsigned short);

void func2(T_MYSTRUCT *p1, unsigned char p2)
{
   unsigned short local;

   local = (0x0040) + (p2 * 10);
   func1((unsigned char*)p1,local,10);
}


unsigned char test( void )
{
   unsigned char local1=201;
   unsigned char local2=0;
   T_MYSTRUCT local3;
   T_MYSTRUCT *plocal3 = &local3;

   do {
      func2(plocal3, local1);
      if ( plocal3->MyByte != 0x00)  {
          local2 ++;
      }
      local1++;
   } while ( local1 < 204 );
   return (local2);
}
```

Workaround:

Use either option –Ohm instead of option –Ohs or disable ‘code inlining’ by option –no_inline if option –Ohs is used

No. C55 — MISRA C 2004 Rule 17.4 triggered by mistake

Details

MISRA C rule 17.4 is triggered by mistake for arrays included in structures:

```
Error [Pm152]: array indexing shall only be applied to objects defined
as an array type (MISRA C 2004 rule 17.4)
```

Example:

```
typedef unsigned char uint8;

void test(void);

void test(void)
{
  struct {
    uint8 u8Array[4];
  } tStruct;

  tStruct.u8Array[0] = 5u;
  tStruct.u8Array[1] = tStruct.u8Array[0];
}
```

Workaround:

Disable rule 17.4 by using the #pragma diag_suppress directive for source lines accessing an array included in a structure:

```
typedef unsigned char uint8;

void test(void);

void test(void)
{
  struct {
    uint8 u8Array[4];
  } tStruct;

  #pragma diag_suppress = Pm152
  tStruct.u8Array[0] = 5u;
  tStruct.u8Array[1] = tStruct.u8Array[0];
  #pragma diag_default = Pm152
}
```

No. C56 — Banked Memory Model: Stack corrupted by wrongly generated code

Details

If banked memory model and an optimization level larger than low is used, the compiler may generate wrong that corrupts the stack if a comparison of 32bit value with a constant is made. The example to demonstrate the occurrence is too complex to be listed in this document.

Workaround:

Reduce the optimization level for the function where the problem occurs by using the directive #pragma optimize=low.

```
#pragma optimization = low
void foo1 (void)
{
  …
}
```

No. C57	**Banked Memory Model: Function Parameter not set**

Details

If an optimization level larger than medium is used, the compiler generates wrong code by not passing the constant function parameter of the banked function ‘func2’ the following sample:

```
extern unsigned char global_1, buffer_1[8], buffer_2[8];
extern              void func1 (void);
extern              void func2 (unsigned char);
extern              void func3 (void);
extern __non_banked void func4 (unsigned char);

void test(void)
{
      if (buffer_1[1]=='Y') {
            if (buffer_1[2]=='S') {
                  switch (buffer_1[3]) {
                        case 0x11: {
                              func3();
                              buffer_1[2] = 'T';
                              break;
                        }
                        case 0x12:{
                              func2(1);
                              buffer_1[2] = 'T';
                              break;
                        }
                        default :
                        {
                              buffer_2[2]='S';
                              func4(2);
                              break;
                        }
                  }
            }
            else {
                  if (buffer_1[2]=='T') {
                  }
                  else {
                        buffer_1[2]--;
                  }
            }
      }
}
```

Workarounds:

1) Reduce the optimization level of the function where the problem occurs by using the directive #pragma optimize=medium.

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

2) Define function ‘func2’ as ‘non-banked’ function:

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

No. C58	**Wrong parameter passing to library function for signed 32bit comparison**

Details

If an optimization level low or higher is used, the compiler generates wrong code for the following sample. The parameter passing to the library function for the signed 32bit comparison is incorrect; the first parameter must be passed in register AX, BC.

```
#define HIBYTE(w) ((unsigned char)(((w) >> 8) & 0x00FF))
#define LOBYTE(w) ((unsigned char)( (w)       & 0x00FF))
#define BUILD_WORD(h, l)            (((unsigned short)(h) << 8) + (l))

unsigned short test(signed long i)
{
  unsigned char local1, local2 = 0;
  i *= 10;

  if(i < 0){
        local1 = 0x80;
        i = -i;
  }
  else {
         local1 = 0;
  }
  while(i > 0x7FF){
        i /= 2;
        local2++;
  }
  if(local1) {
      i = ~(unsigned short)i + 1 & 0x7FF;
      }
  return BUILD_WORD(LOBYTE(i), HIBYTE(i) | local2 << 3 | local1);
}
```

Workaround:

1) Reduce the optimization level of the function where the problem occurs by using the directive #pragma optimize=none.

```
#pragma optimization = none
unsigned short test (signed long i)
{
  …
}
```

No. C59	Internal Compiler error in functions using an endless loop

Details

Functions containing an if - statement using different amounts of stack and immediately followed by a ‘while(1) ; ‘ construction might generate an internal error.

```
#include <stdio.h>

      unsigned short s1, s2;

extern unsigned short f1 ( unsigned short, unsigned short);

void main(void)
{
   while (1)
   {
     if(s1 != s2) {
       printf("dummy text: 0x%hx vs 0x%hx \n",s1,s2);
     }
     while(1);
  }
}
```

Workaround:

Replace the endless loop while(1) by a loop using a variable:

```
#include <stdio.h>

      unsigned short s1, s2;

extern unsigned short f1 ( unsigned short, unsigned short);

const unsigned char s3=0;

void main(void)
{

   while (1)
   {
     if(s1 != s2) {
       printf("dummy text: 0x%hx vs 0x%hx \n",s1,s2);
     }
     while(s3==0);
   }

}
```

The problem will be fixed in the next compiler update.

No. C60	**Wrong code generated for masking a bit of 16bit-high byte**

Details

In dependent of the used optimization level the compiler generates wrong code for comparing an unsigned 16bit value with a constant bit pattern with either one bit of the upper byte (high byte) set or cleared.

```
#define MASK   0x0200

typedef struct {
    unsigned short element1;
} struct1;

void test( struct1 * parameter1 )
{
  if ((parameter1->element1 & MASK) != 0) {
      …
  }
}
```

Workarounds:

1) Casting the unsigned 16bit value to signed value:

```
void test( struct1 * parameter1 )
{
  if ((((signed short)parameter1->element1) & MASK) != 0) {
      …
  }
}
```

2) Upgrade to a new compiler version V4.60a or later

No. C61	**Missing Warning about change of sign due to integer conversion**

If the sign of a constant given in hexadecimal or octal format is changed due an integer conversion, the compiler doesn't generate a warning (Pe068).

```
short test (void)
{
  return (0x8000);
}
```

Workaround:

Use the decimal format:

```
short workaround (void)
{
  return (32768);
}
```

Form the next compiler version onwards a remark will be generated if the sign of a constant given in hexadecimal or octal format is changed due to an integer conversion. As result the behavior will be the same for constants given in decimal and hexadecimal format.

No. C62	**Usage of uninitialized carry-flag**

If a speed-optimization level medium or higher is used, the compiler may use the carry flag before using initialize it. This problem is demonstrated in following sample in the following sample:

```
unsigned int v1;
unsigned int v2;
unsigned int v3;

volatile unsigned int r1;

void test(void)
{
          unsigned char v4;
            signed int  v5;
  static  unsigned int  v6;

  v4 =  (v1/4 / 256)+1;
  v2 =  (v1/4);             /* error: CY isn't cleared before usage*/
  v6 =  (v1/4) ;
  v5 =  v3 / 16 / v4;
  r1 =  v6 + v2 + v5 + v4;
}
```

Workaround:

Reduce the optimization to low.
In the special sample above a local temp variable can be used to avoid the problem:

```
void test(void)
{
          unsigned char v4;
            signed int  v5;
  static  unsigned int  v6;
          unsigned int  temp = v1/4;

  v4 =  (temp / 256)+1;
  v2 =  temp;
  v6 =  temp;
  v5 =  v3 / 16 / v4;
  r1 =  v6 + v2 + v5 + v4;
}
```

To fix the problem, please download compiler patch V4.50e available at the IAR Systems MyPages area (www.iar.com -> Menu MyPages or http://supp.iar.com/MyPages/).
Alternatively please feel free to contact the NEC software tool support team (software_support@eu.necel.com).

(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 NEC 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. D12	Memory Bank Area is not filled up.

Details

Although the options –H and –h are used correctly, an area at the end of a memory bank may not be filled up. This is documented in the linker map-file:

```
                ****************************************
                *                                      *
                *              CHECKSUMS               *
                *                                      *
                ****************************************

Symbol      Checksum  Memory  Start      End       Initial value
------      --------  ------  -----      ---       -------------
__checksum    0x7f2f   CODE       0000 - 7FFD           0x0
                       CODE       8000 - BFFC
                       CODE   00018000 - 0001BFFD
                       CODE   00028000 - 0002BFFF
                       CODE   00038000 - 0003BFFF
```

The correct CKECHSUMS section should be as follows:

```
                ****************************************
                *                                      *
                *              CHECKSUMS               *
                *                                      *
                ****************************************

Symbol      Checksum  Memory  Start      End       Initial value
------      --------  ------  -----      ---       -------------
__checksum    0x7f2f   CODE       0000 - 7FFD           0x0
                       CODE       8000 - BFFF
                       CODE   00018000 - 0001BFFF
                       CODE   00028000 - 0002BFFF
                       CODE   00038000 - 0003BFFF
```

Workarounds

Either use the previous linker to version V4.59q until V4.60b or later is available or fill-up the areas manually by adding the following lines of source code:

```
__root const unsigned char fill1[3] @ 0x0BFFD = {0xFF,0xFF,0xFF};
__root const unsigned char fill2[2] @ 0x1BFFE = {0xFF,0xFF};
```

No. D13	Corrupted IRQ table for 78K0R devices in case of using the XCOFF78K output format
	Details In case of using the XCOFF78K format and a device of the 78K0R-series, the IRQ table contains wrong entries. The addresses of the ISRs are fixed to 0x0000. *Workarounds* Upgrade the linker to version V4.60c or later.

No. D15	Unused odd address isn't filled up
	Details In some cases an unused odd address directly after a constant located by absolute memory allocation is not filled-up, although the options -H and -h are used. Example: `__root const unsigned char const1    @ 0x1080 = 0x01;` The following address 0x01081 is not filled up, although -h00000-BFFF is used *Workarounds* Upgrade the linker to version V4.60c or later.

No. D16 **Range Error occurred by mistake**

Details

In same cases the definition of a near constant causes a range error by mistake.

```
Error[e18]: Range error, Limit exceeded

  Where $ = main + 0x1  [0x8C6]
            in module "main" (.\main_z3.r26),
            offset 0x1 in segment part 4, segment CODE
  What: (array + 1) [0x1001]
  Allowed range: 0xF0000 - 0xFFFFF
  Operand: array [0x1001]
           in module main (.\main_z3.r26),
           Offset 0x1 in segment part 3, segment NEAR_CONST
```

Example:

```
const __near unsigned char array[10]={0,1,2,3,4,5,6,7,8,9};
```

Workarounds

Use the option –Rw to reduce the message level to warning. Now an output file is generated and it is still possible to be noticed about other range problems.
The problem will be fixed in the next major update V4.50a.

No. D17	Error in 78K0R xcl-file (lnk78f11xx_xx.xcl) templates

Details

If a module includes object definitions of a size of more than 64KB, error will occur:

```
Error[e16]: Segment XCODE (size: 0x16905 align: 0) is too long for
segment definition. At least 0x6905 more bytes needed. The problem
occurred while  processing the segment placement command "-
Z(CODE)XCODE=[000D8-3FFFF]/10000", where at the moment of placement
the available memory  ranges were "CODE:ef3-ceff,CODE:cf2a-
ffff,CODE:10000-1ffff,CODE:20000-2ffff,CODE:30000-3ffff"
```

The problem is caused by an incorrect definition of the far code segment XCODE in the 78K0R xcl-file templates.

```
//-------------------------------------------------------------------
//      Far functions code segment.
//-------------------------------------------------------------------
-Z(CODE)XCODE=[000D8-3FFFF]/10000
```

Workarounds

Please correct the definition by using the option –P (=packed segments):

```
//-------------------------------------------------------------------
//      Far functions code segment.
//-------------------------------------------------------------------
-P(CODE)XCODE=[000D8-3FFFF]/10000
```

Corrected XCL-file templates will be included in the EW78K V4.50a and later.

No. D18	Missing information for far pointer in XCOFF78K output format

Details

Some information for far pointer is missing in the debug format XCOFF78K and therefore the highest byte (bit16 –bit23) of the pointer address is displayed incorrect in the watch window of the NEC debugger ID78K0R-QB. The generated code is correct.

Example:

```
__root const unsigned char test[5] ={0,1,2,3,4};

void test (void)
{

  const unsigned char __far *LocalFarPointer;

  LocalFarPointer  = &test[0];
  …
}
```

Workarounds

The problem will be fixed in XLINK V4.60i and later.

No. D21 **Output file format UBROFF5: Error [e62] is generated erroneously if multiple modules are defined in one assembler source file**

Details

The linker error message [e62] is generated erroneously if multiple modules are defined in one assembler source file and the output file format UBROFF5 is selected:

```
Error[e62]: File name "C:\...\test.s26" used for multiple files
```

Example:

```
    MODULE m1
; some assembler code
    ENDMOD

    MODULE m1
; some assembler code
    ENDMOD

    END
```

Linker output format selection: -FUBROFF5

Workarounds

Define only one module per assembler source file or use the current version of the UBROFF format. The problem will be fixed in a future linker version.

No. D22 **Output file format IEEE695: Missing enum datatype debug information**

Details

An IEE695-format output doesn’t include debug information of enum-datatypes.

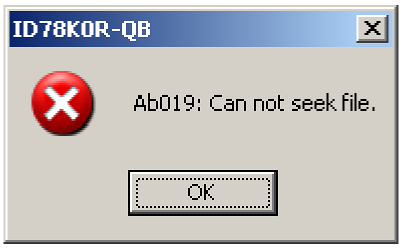
Example:

```
enum colour {rot, blau, gruen, gelb, braun};
enum colour my_enum;

void test (void)
{
    if (my_enum <= braun) {
        my_enum++;
    }
}
```

Linker output format selection: `-FIEEE695`

Workarounds

None. The problem will be fixed in a future linker version.

No. D23	**Output file format XCOFF78K: Usage of untyped segments may cause a corrupted file**
	Details If untyped segments (e.g `RSEG MYDATA(1)`) are used in an assembler file, the generated NEC debug file (format xcoff78) may be corrupted. The NEC debugger will generate the following error message if such file shall be downloaded:  Ab019: Can not seek file. *Workarounds* Please use only typed segment: RSEG MYDATA(1):DATA

No. D24	**Output file format RAW-BINARY: XLINK may hang up**
	Details If an output file in RAW-BINARY format shall be generated the linker may hang and the process must be manually killed. *Workarounds* None. The problem will be fixed in a future version

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

No. E24	C-SPY Driver for 'IECUBE': Realtime Memory Window Update interrupts application
	Details To update the content of the Realtime Memory Window a running application is interrupted by the C-SPY debugger. This procedure is started when the Realtime Memory is Window is opened once and is continued even if the Real-time Memory is closed again. *Workaround* The issue will be fixed in version V4.40b or later. To disable the procedure in version V4.30x and V4.40a use the following procedure: - close the Real-time Memory Window - close the Embedded Workbench - delete the subfolder 'settings' of the project folder In the next debug-session the application isn't interrupted anymore until the Real-time Memory Window isn't opened.

No. E25	Starting C-SPY by command line: Wrong Simulator started
	Details In case of calling C-SPY from the command line, independent of the option setting in command line or the project options always the simulator for 78K0 is started. *Workaround* Please use a 'Debug-project' to start a debug session for an externally build application. The issue will be fixed in future update (V4.50a or later).

No. E26	Starting C-SPY by command line: C-SPY driver for 'IECUBE' crashes in case of using 78K0R emulator
	Details In case of calling C-SPY from the command line with the driver for IECUBE and a 78K0R emulator, the debugger crashes. This is independent from the debug project. *Workaround* Please use a 'Debug-project' to start a debug session for an externally build application. The issue will be fixed in a future update (V4.50a or later).

No. E27	**Event-Breakpoint is deleted incompletely**
	Details If an event-breakpoint is deleted in the Breakpoint Window while the Event Window is open, the breakpoint is only removed form the Breakpoint Window, but not deleted. Although the breakpoint isn't listed anymore in the Breakpoint Window, it is still active. The corresponding event can not be deleted, because it is still in use. *Workaround* Please close the Event Window before deleted a breakpoint in the Breakpoint Window. If an event-breakpoint had been deleted while the Event Window was open, the breakpoint can not be deleted anymore by the C-SPY debugger. To remove the breakpoint please close the Embedded Workbench and delete the file `'<project_name>.dni'` in the subfolder `'setting'` of your project folder. This file contains only settings of the last debug session is automatically created again after starting a new debug-session.

No. E28	**C-SPY Driver for 'MINICUBE': Wrong display of main clock source of QB-78K0MINI-EE**
	Details In the Hardware Setup Window always the System Clock is displayed as main clock source. If an oscillator is mounted at the internal socket CLK1, clock board should be displayed as main clock source. This is only a display problem, if an oscillator is mounted at the internal socket this clock is used as main clock. *Workaround* The issue is fixed in version V4.50a and later.

No. E29	**C-SPY Driver for 'IE-78K': C-SPY fatal error in case of illegal SFR access**
	Details In case of using the C-SPY debugger and the IE-78K-driver any illegal SFR access causes a fatal C-SPY error. *Workaround* The issue is fixed in version V4.50a and later.

No. E30	C-SPY Driver for 78K0R ‘IECUBE’ or ‘MINICUBE’: Fatal error after selecting 'SFR' in disassembly window
	Details If in the disassemble window the memory area 'SFR' is selected, the debugger generates a `FATAL ERROR: unknown exception in driver (#M1)` and the debug session is closed. *Workaround* Don’t select the memory area ‘SFR’ in the disassemble window, because code execution is not possible in this area. The issue is fixed in version V4.50a and later.

No. E31	C-SPY Driver for 78K0R ‘IECUBE’ or ‘MINICUBE’: Memory read access by macro is blocked
	Details Memory read is blocked when executing a macro from a breakpoint during execution. *Workaround* Please update to version V4.40c or later.

No. E32	Code Coverage information is incomplete in case of using banked memory systems.
	Details If an application uses banked memory, the Code Coverage information is incomplete. *Workaround* Please update to version V4.40c or later.

No. E33	C-SPY Driver for 78K0R ‘MINICUBE’: The input field for the main clock source allows only selecting a value from a predefined list.
	Details The input field for the main clock source allows only selecting a value from a predefined list. Therefore it is not possible to enter the correct frequency, if an external clock of a frequency not listed is used. The selection of a different frequency causes a C-SPY fatal error after switching to the external clock. *Workaround* The problem will be fixed in version V4.50a or later. In case of any urgent request, please contact NEC Electronics Tool Support Team (‘software_support@eu.necel.com’).

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. E35	**Argument variables can not be used to define a code breakpoint by source location.**

Details

Argument Variables can not be used to define a code breakpoint by source location,
e.g. `$PROJ_DIR$\source\main.c, row 26`.

In case of using argument variables, the error message 'The file does not exist' is displayed although path and filename are correct.

Workaround

This issue is listed as an improvement proposal for future versions.

No. E36	**C-SPY Driver for 78K0R Simulator: Instruction 'mov memory_location[C],A' simulated incorrectly**

Details

The instruction 'mov memory_location[C],A' is not simulated correctly.
A wrong value is written to memory.

Extract from compiler list file, the incorrectly simulated instruction is marked **red**.

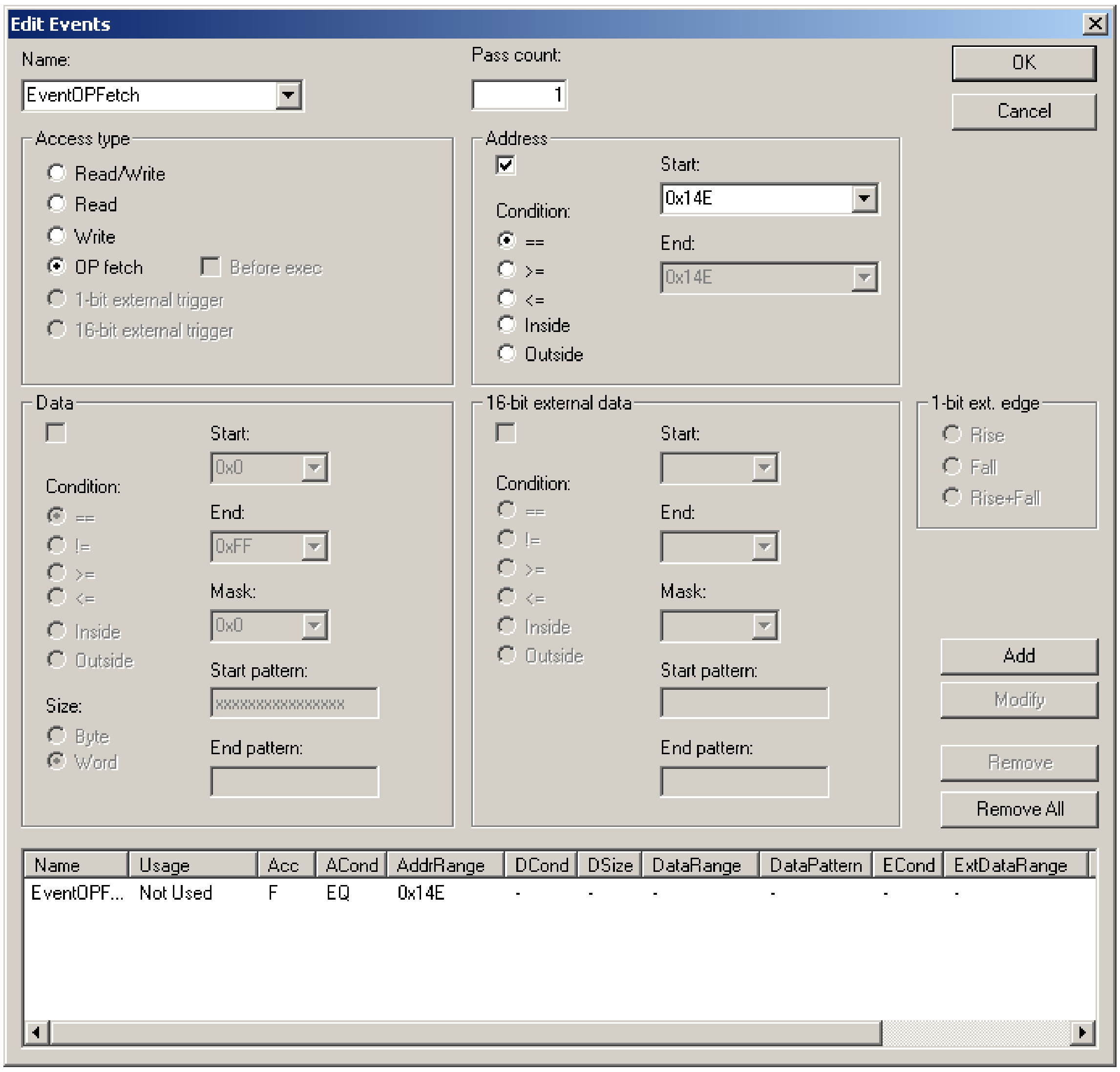
```
      8             for( i=0 ; i < SIZE; i++ ) {
   \   000001 F0             CLRB       X
   \   000002 EF07           BR         S:??main_0
      9               array[ i ] = 0;
   \                      ??main_1:
   \   000004 60             MOV        A, X
   \   000005 72             MOV        C, A
   \   000006 F1             CLRB       A
   \   000007 28....         MOV        (array & 0xFFFF)[C], A
     10             }
   \   00000A 80             INC        X
   \                      ??main_0:
```

Workaround

None. The problem will be fixed in version V4.50a.

No. E37	C-SPY Driver for 78K0R IECUBE: OP-Fetch before execution can not be defined

Details
The checkbox 'Before Exec' in Event Definition Dialogue is always disabled. The user can not define an OP-fetch before execution event.



Edit Events

Name: EventOPFetch
Pass count: 1

Access type: ( ) Read/Write ( ) Read ( ) Write [✓ OP fetch] ( ) 1-bit external trigger ( ) 16-bit external trigger
☐ Before exec

Address: ☑
Start: 0x14E
End: 0x14E
Condition: [✓ ==] ( ) >= ( ) <= ( ) Inside ( ) Outside

Data: ☐
Start: 0x0
End: 0xFF
Mask: 0x0
Start pattern: xxxxxxxxxxxxxxxx
End pattern:
Condition: [✓ ==] ( ) != ( ) >= ( ) <= ( ) Inside ( ) Outside
Size: ( ) Byte [✓ Word]

16-bit external data: ☐
Start:
End:
Mask:
Start pattern:
End pattern:
Condition: ( ) == ( ) != ( ) >= ( ) <= ( ) Inside ( ) Outside

1-bit ext. edge: ( ) Rise ( ) Fall ( ) Rise+Fall

Name	Usage	Acc	ACond	AddrRange	DCond	DSize	DataRange	DataPattern	ECond	ExtDataRange
EventOPF...	Not Used	F	EQ	0x14E	-	-	-	-	-	-

Workaround
None. Please define an OP-fetch event at the previous instruction.

No. E38	C-SPY Driver for TK-78K: Download to memory banks failed

Details
In case of using a 78K0-device of more than 60KB internal ROM or FLASH memory, the download to memory bank fails.

Workaround
None. The problem is fixed in C-SPY Version V4.50b.

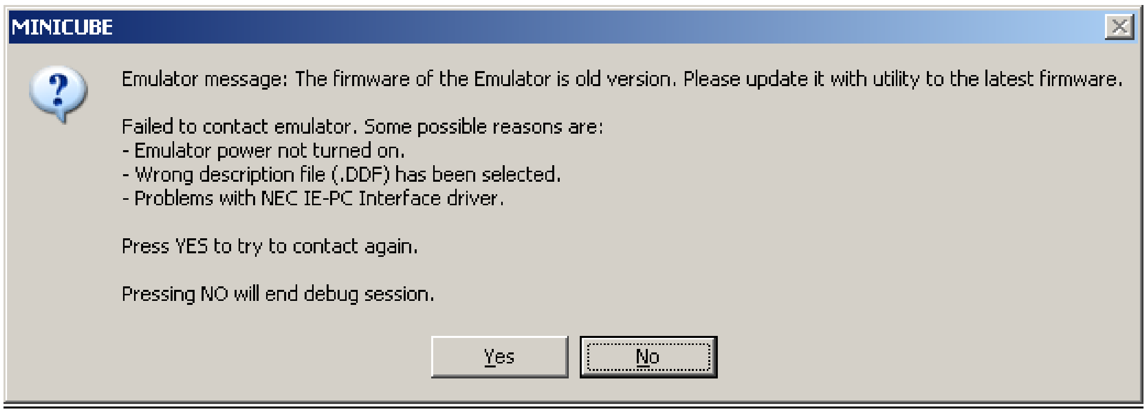
No. E39	C-SPY Driver for 78K0R: High Byte of Program Counter (bit16-bit23) is set to 0x00
	Details The set-next-statement-command sets the high byte of the Program Counter (bit16-bit23) always to zero. If the set-next-statement-command is used inside a far function, this will corrupt the debug session and a manual reset is necessary. Also a manual change of the Program Counter in the Register Window sets the high byte always to zero. The issue concerns the 78K0R C-SPY driver for IECUBE, MINICUBE, and TK-Interface. *Workaround* None. The problem will be fixed in future C-SPY Version.

No. E40	C-SPY Driver for 78K0R: A file in Intel-Hex- or Motorola-S-Record format can not be downloaded
	Details If an Intel-Hex or a Motorola-S-Record file is downloaded instead of debug file the C-SPY debuggers doesn't finish the download and stays at an endless loop. The complete Embedded Workbench has to be closed afterwards *Workaround* None. Please use any NEC programmer (QB-MINI2, PG-FP4) to program intel-hex files. The problem will be fixed in future C-SPY Version.

No. E41	C-SPY Simulator Driver: Wrong mask-flag is used to control an interrupt
	Details By mistake a wrong mask flag may be used to control an interrupt. Example: For the microcontroller µPD78F0547 the mask flag TMMKH0 (MK0H_bit.no4) is used to control the interrupt INTTM0 instead the correct mask flag TMMK000 (MK0H_bit.no6). *Workaround* None. The problem will be fixed in the next C-SPY Version. In case of an urgent request please contact the NEC software tools support team (software_support@eu.necel.com) and list used microcontroller.

No. E42	C-SPY 78K0 IECUBE Driver: Full trace break doesn't work
	Details In case of using a trace size of less than 128KB, a defined full trace break doesn't stop the application. *Workaround* Please use the max. trace size of 128KB.

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:  MINICUBE Emulator message: The firmware of the Emulator is old version. Please update it with utility to the latest firmware. Failed to contact emulator. Some possible reasons are: - Emulator power not turned on. - Wrong description file (.DDF) has been selected. - Problems with NEC IE-PC Interface driver. Press YES to try to contact again. Pressing NO will end debug session. *Workaround* The MINICUBE2 firmware V4.06 will be available b/o October 2008. Until then please contact the NEC software tool support team (software_support@eu.necel.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;
}
```

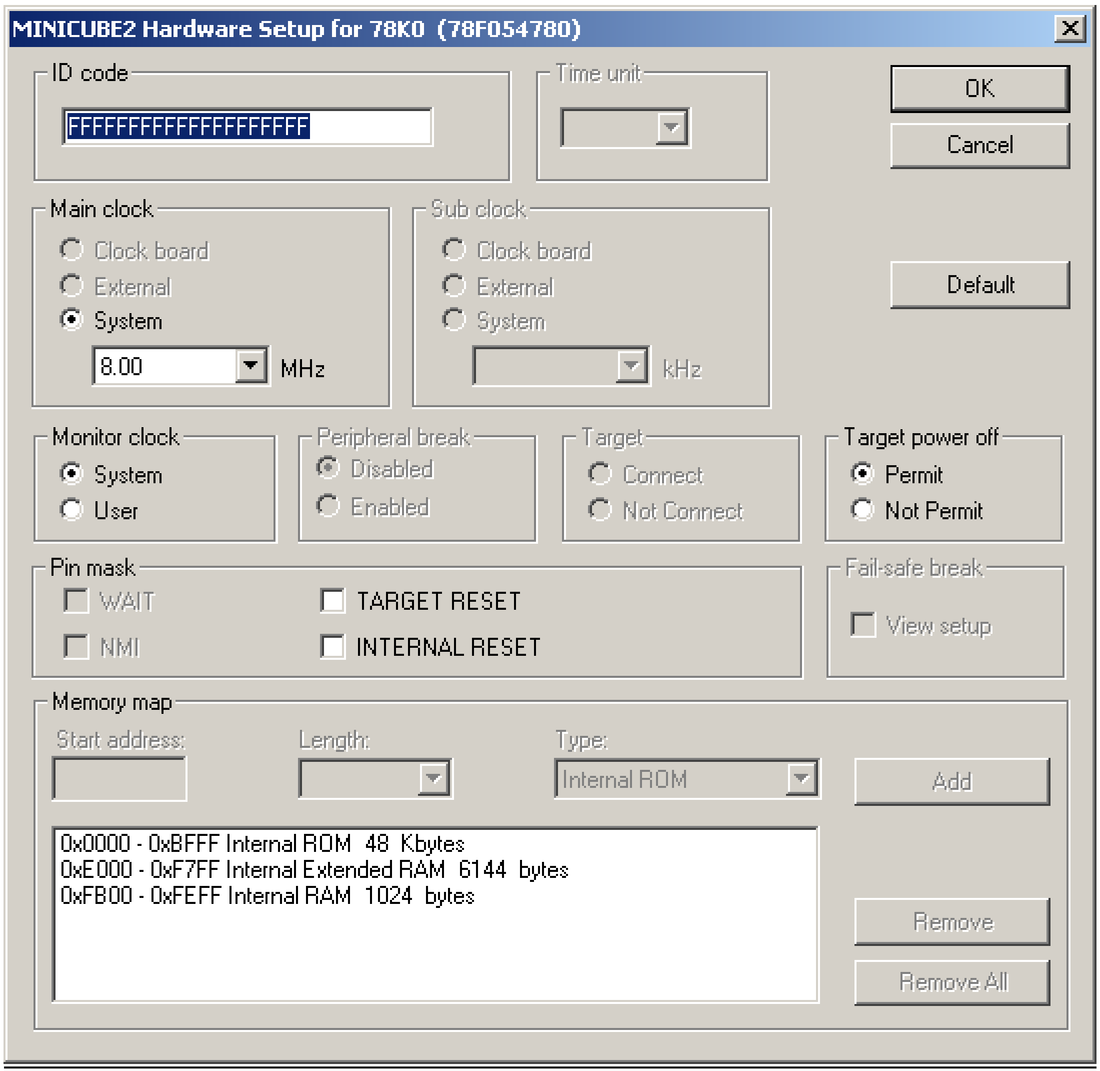
Workarounds

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:



MINICUBE2 Hardware Setup for 78K0 (78F054780)

ID code: FFFFFFFFFFFFFFFFFFFF

Main clock: Clock board / External / [✓ System] 8.00 MHz

Monitor clock: [✓ System] / User

Peripheral break: [✓ Disabled] / Enabled

Target: Connect / Not Connect

Target power off: [✓ Permit] / Not Permit

Pin mask: WAIT, NMI, TARGET RESET, INTERNAL RESET

Memory map:

```
0x0000 - 0xBFFF Internal ROM  48  Kbytes
0xE000 - 0xF7FF Internal Extended RAM  6144  bytes
0xFB00 - 0xFEFF Internal RAM  1024  bytes
```

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 NEC NEC software tool support team (software_support@eu.necel.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.

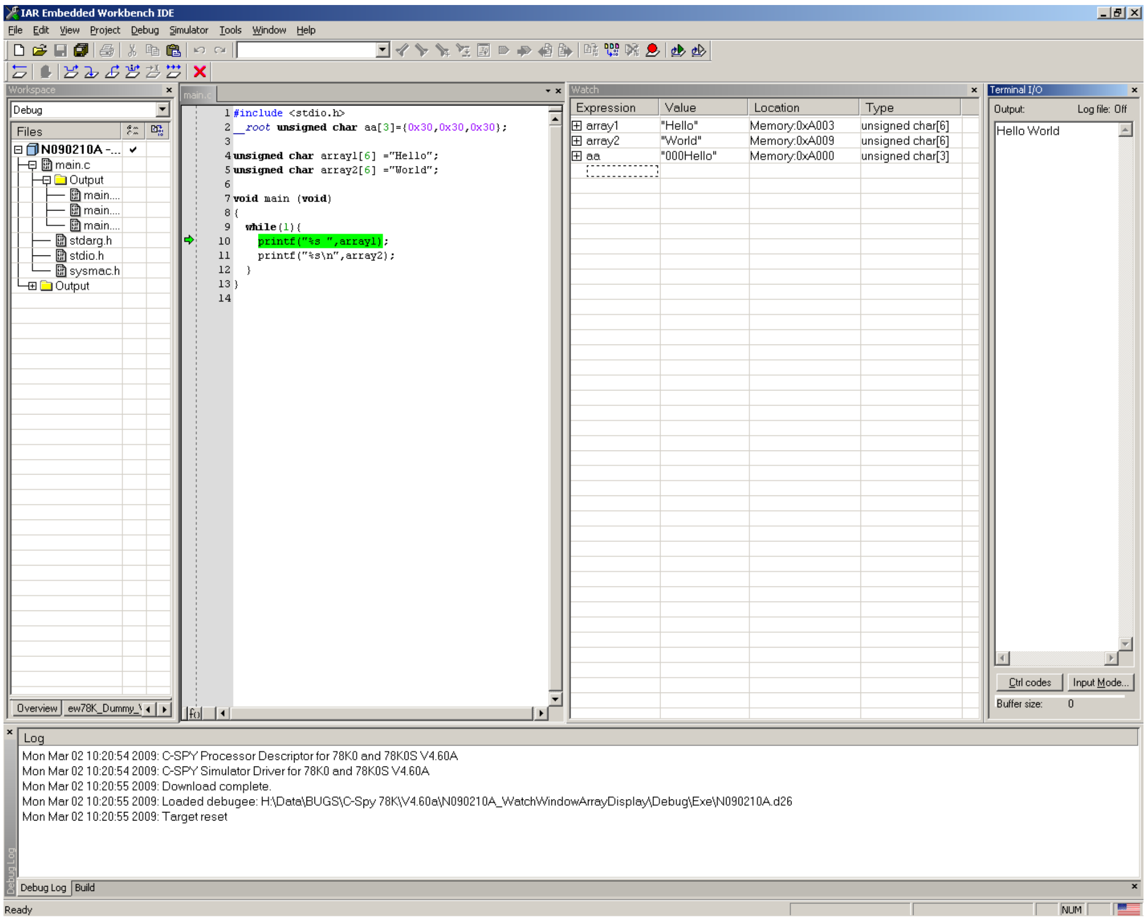
```
;------------------------------------------------------------------------------
;       CSTARTUP source for 78K
;
;       This module contains the code executed before the C/C++ "main"
;       function is called.
;       The code usually must be tailored to suit a specific hardware configuration.
;
;       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";
```



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

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

void main (void)
{
  while(1){
    printf("%s ",array1);
    printf("%s\n",array2);
  }
}
```

Expression	Value	Location	Type
⊞ array1	"Hello"	Memory:0xA003	unsigned char[6]
⊞ array2	"World"	Memory:0xA009	unsigned char[6]
⊞ aa	"000Hello"	Memory:0xA000	unsigned char[3]

```
Hello World
```

```
Log
Mon Mar 02 10:20:54 2009: C-SPY Processor Descriptor for 78K0 and 78K0S V4.60A
Mon Mar 02 10:20:54 2009: C-SPY Simulator Driver for 78K0 and 78K0S V4.60A
Mon Mar 02 10:20:55 2009: Download complete.
Mon Mar 02 10:20:55 2009: Loaded debugee: H:\Data\BUGS\C-Spy 78K\V4.60a\N090210A_WatchWindowArrayDisplay\Debug\Exe\N090210A.d26
Mon Mar 02 10:20:55 2009: Target reset
```

Workaround

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

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

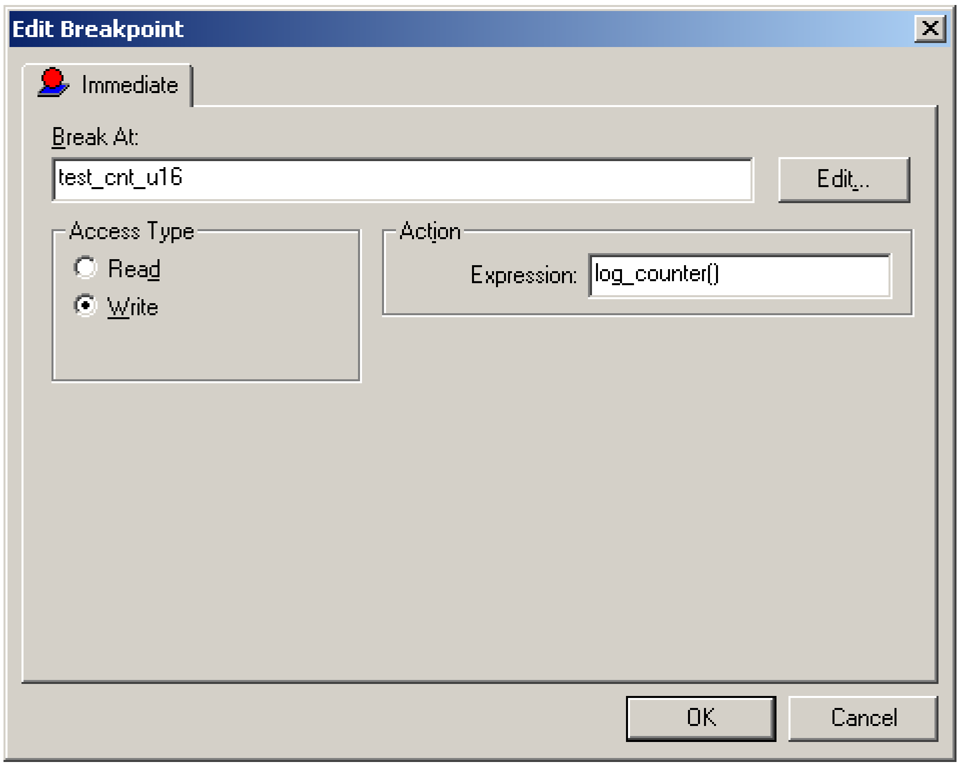
Details

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;
}
```



Break At: test_cnt_u16

Access Type: Read [✓ Write]

Action: Expression: log_counter()

Workaround

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

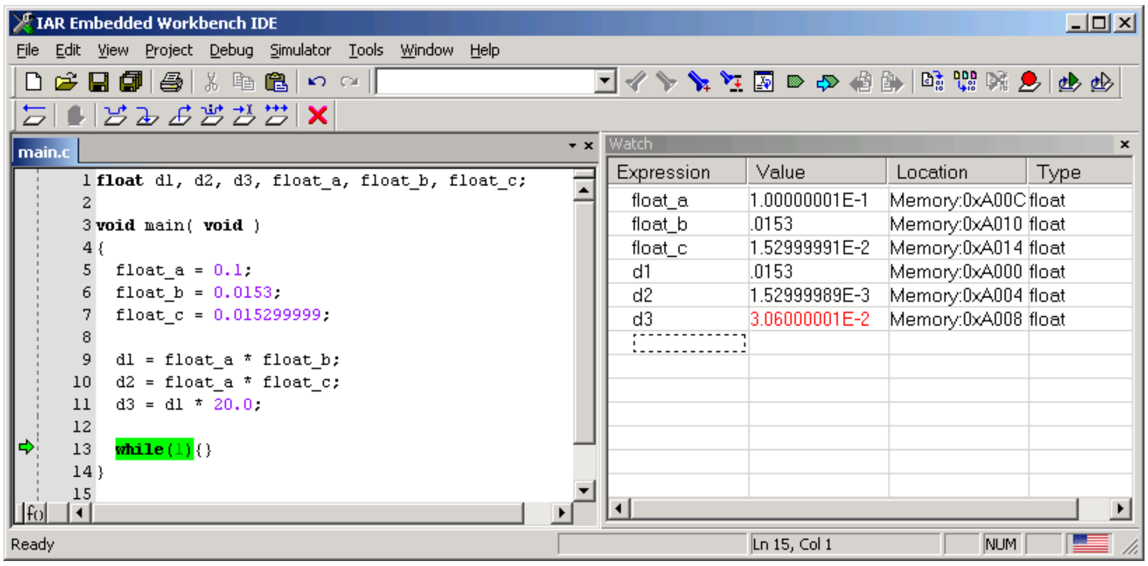
No. E52	C-SPY 78K: Displayed floating point value in watch window may be wrong

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.


IAR Embedded Workbench IDE
File Edit View Project Debug Simulator Tools Window Help
main.c
Watch
Ready
Ln 15, Col 1
NUM

```
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){}
}
```

Expression	Value	Location	Type
float_a	1.00000001E-1	Memory:0xA00C	float
float_b	.0153	Memory:0xA010	float
float_c	1.52999991E-2	Memory:0xA014	float
d1	.0153	Memory:0xA000	float
d2	1.52999989E-3	Memory:0xA004	float
d3	3.06000001E-2	Memory:0xA008	float

Workaround

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

(M) Description of Operating Precautions for the Assembler A78K0R

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

	<u>*Details*</u>
	
	The assembler calculates a wrong relative jump-distance if the RSEG directive is used within a macro definition:
	
	<u>*Example*</u>
	
	`myDummyMacro MACRO`
	`        RSEG    CODE`
	`        NOP`
	`        ENDM`
	
	<u>*Workaround*</u>
	
	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. F2	**It is not possible to use an assembler DEFINE to an external symbol**

	<u>*Details*</u>
	
	In case of using an assembler DEFINE to an external symbol, the linker will generate the following error:
	
	`Fatal Error[e20]: Corrupt file.  External index out of range in module MODUL2 ( C:\....\test.r26 )`
	
	<u>*Example*</u>
	
	`    EXTERN S2`
	
	`SYM DEFINE S2`
	
	<u>*Workaround*</u>
	
	None. The assembler version V4.41a or later will generate an error for such cases.

No. F4	**EVEN Directive doesn't align Data to even Address.**

Details

The EVEN directive aligns to an even address relative to the start module-startaddress of the segment instead of an absolute even address. In case of an odd module-startaddress also all symbols aligned with an even-directive are located at an odd address. In this case a linker error message will be generated for each access to the misaligned variable:

```
     IAR Universal Linker V4.60A/386
     Copyright 1987-2006 IAR Systems. All rights reserved.

Error[e18]: Range error,   Even value expected
 File: H:\Data\...\even.asm, Line: 17
 Source:         MOVW  S:integer1, AX
 Where $ = test_even + 0x1  [0xA8]   in module "even",
       offset 0x1 in segment part 1, segment CODE
 What: (integer2 + 2) & 1 [0x1]
       Allowed range: 0x0 - 0x0   Operand: integer2 [0xfe23]
       in module even, Offset 0x2 in segment part 0, segment SADDR_Z
```

Example

```
    RSEG SADDR_Z

CharVar1:   DS 1
    EVEN
IntVar1:    DS 2
```

Workaround

Please align the segment-start address to an even address:

```
    RSEG SADDR_Z(1)

CharVar1:   DS 1
    EVEN
IntVar1:    DS 2
```

No. F5 **Automatic Replacement of DBNZ Instruction causes Linker Error Message**

Details

In case of using the 78K0 DBNZ instruction with the 78k0R assembler, the assembler outputs a warning that this instruction is not available and will be replaced by DEC and BNC instruction. But the automatic replacement causes the following linker error message:

```
Error[e18]: Range error, Limit exceeded
```

Example

```
asm_func:
       push BC
       DBNZ  C,m1           ; is replaced by DEC C and BNZ m1
       DBNZ  B,m1           ; is replaced by DEC B and BNZ m1
m1:

       pop BC
       ret
```

Workaround

Please use directly the correct instructions:

```
asm_func:
       push BC
       DEC C
       BNZ m1
       DEC B
       BNZ m1
m1:

       pop BC
       ret
```

No. F6 **Invalid Register in XCH Instruction causes the generation of wrong Op-Code**

Details

In case of using the XCH instruction with two registers, register A (R1) must be the first parameter. If register A (R1) is used as second parameter a wrong op-code is generated instead of displaying error message [Ab0006] illegal register:

Example

```
asm_func:
       push AX
       xch  X,A
       pop  AX
       ret
```

Workaround

Please use only the correct XCH instruction with register A (R1) as first parameter:

```
asm_func:
       push AX
       xch  A,X
       pop  AX
       ret
```

No. F7	**Invalid XCH instruction doesn't cause a syntax error**

Details

The XCH instruction with two registers requires that the first parameter is the register A (R1).
If by mistake register X (R0) is used as first parameter, the assembler doesn't generate an error message, but inserts instead the op-code of the instruction xch A, <register>.

Example

```
asm_func:
       …
       xch  X,D
       …
       ret
```

Workaround

Please use only the correct XCH instruction with register A (R1) as first parameter:

```
asm_func:
       …
       mov  A,D
       xch  A,X
       mov  D,A
       …
       ret
```

No. F8	**Wrong Op-Code generated for *MOV <register>, SFR-address* instruction**

Details

In case of using an SFR symbol name or absolute SFR address for a MOV <register> instruction where register was unequal A, the assembler generated wrong opcodes.
Independent of the used register, the opcode for MOV A, instruction is generated

Example

```
asm_func:
       …
       mov   X, PM0        ; wrong opcodes generated
       mov   B, 0xFFF20    ; wrong opcodes generated
       …
       ret
```

Workaround

Please use the 16bit sfr-address until the problem will be fixed:

```
asm_func:
       …
       mov   X, 0xFF20
       mov   B, 0xFF20
       …
       ret
```

No. F9	**Illegal MOV instruction is accepted and wrong Op-Code is generated**

Details

The assembler accepts illegal mov instructions from another register than register A (mov <register1>, <register2>) and generates always the op code for the correct instruction mov <register1>,A.

Example

```
asm_func:
       …
       mov  B,C    ; illegal instruction, opcode for mov B,A generated
       mov  D,H    ; illegal instruction, opcode for mov D,A generated
       …
       ret
```

Workaround

Please use correct instructions ‘mov <reg1>, A’ only.

No. F10	**Invalid operand of branch instruction causes fatal assembler error**

Details

The usage of an invalid operand for the unconditional branch instruction causes a fatal error of the assembler and an abnormal termination.
Additionally the user is asked to send a problem report to Microsoft (only version v4.60a)

Example

```
asm_func:
       …
       br CS:0xDF00 ; invalid operand
       …
       ret
```

Workaround

Please use only correct operands described in manual, e.g.

```
       mov  CS, #0x0F
       movw AX, 0xDF00
       br   AX

       br  F:0xFDF00
       br  N:0xDF00
       br  0x0010
```

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.

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

No. G6	**Warning [Pe177] generated by fault**

	Details
	
	If a variable defined as '__root' is additionally defined as 'static', the compiler will generate the warning message [Pe177] by fault:
	
	`Warning[Pe177]: variable "test1" was declared but never referenced`
	
	The keyword '__root' informs the linker that the variable should be located even it is not referenced. This implies already that a variable might not be used in the module and that this declaration is done on purpose.
	
	Example:
	
	`static __root __far const char test1= 0x01;`
	
	Workaround:
	
	The problem will be fixed in the next major update. So far please use one of the following workarounds:
	1) Don't define a variable as '`__root`' and '`static`'
	2) Disable warning [Pe177] for such definitions:
	
	`#pragma diag_suppress=Pe177`
	`static __root __far const char test1 = 0x01;`
	`#pragma diag_default=Pe177`

No. G7	**Fatal error in case of using experimental option –mfc**

Details

If two static functions of the same name are exist in modules that are compiled simultaneously by using the currently experimental option –mfc, a fatal error occurs:

```
Internal Error: [CoreUtil/General]: OgModuleLables – label already
defined. Fatal error detected, aborting.
```

Example:

```
source file f1.c:
static unsigned char func1 (unsigned char p1)
{
    // code doesn't matter
    return (1);
}

source file f1.c:
static unsigned int func1 (unsigned int p1)
{
    // code doesn't matter
    return (1);
}
```

Workaround:

The problem will be fixed in the platform release, when the option –mfc will be officially introduced (V4.4xx, schedule is December 2007)

No. G11	**Internal compiler error occurs if a default segment name is used for a user-defined segment.**

Details

In case of using a default segment name of the compiler for user-defined segment of constant data, an internal compiler error occurs after the warning about using a default segment name .

```
Internal error [Front end]: Invalid C99 IL expression kind
Fatal error detected aborting.
```

Example:

```
#pragma location = "CODE"
__root const unsigned char counter=23;

void test(unsigned char *p1)
{
   *p1=*((volatile const unsigned char *)&counter);
}
```

Workaround:

Do not use the compiler default segment names for user-defined segments

No. G12	**Wrong access to far and byte-aligned structure**

Details

In case of using the data model near and accessing a 16bit value in a far and byte-aligned structure, the compiler splits the word load in two char loads. During the split it reverts to using the default pointer, in this case near. This causes a read from the wrong location.

Example:

```
#define FAR_ADDRESS       0x030000

#pragma pack(1)
typedef struct { unsigned char     element1;
                 unsigned short    element2;
                 unsigned char     element3;
               } MyStruct;
#pragma pack()

unsigned short Test1;

void test(void)
{
    Test1 = ((MyStruct __far *) FAR_ADDRESS)->element2;
}
```

Workaround:

Please use the far data model or update to version V4.50b

No. G13	**Wrong code generated for indirect memory access**

Details

In case of using the data model near wrong code may be generated for an indirect memory access. The instruction 'mov A,[HL+C]' is used instead of 'mov A,[HL+B]'.

Workaround:

Please use the far data model.

No. G14	**Register-bank selection of interrupt function may be ignored**

Details

In case of using an optimization level higher than 'low' the compiler may ignore the register-bank selection of the user (#pragma bank) for some interrupt functions.

Example:

```
#include <io78f1188_e4.h>
extern void  f2 (unsigned char );

typedef enum {
    GPT_1,
    GPT_2
}ENUM1;
typedef enum {
    GPT_3,
    GPT_5
} ENUM2;
typedef struct {
    ENUM1         s1;
    unsigned char s2;
    void*         s3;
}STRUCT1_T;

typedef struct{
        ENUM2           s4;
        unsigned short  s5;
} STRUCT2_T;


#pragma bank = 2
__interrupt void isr( void )
{
    unsigned short u16PR0sav, u16PR1sav;
    u16PR0sav = PR00 ;
    u16PR1sav = PR01 ;
    __enable_interrupt();

    if (ptr1[((unsigned char) 0)].s1 == GPT_1)   {
        array[((unsigned char) 0)].s4 = GPT_5;
    }
    f2( ((unsigned char) 0) );

    __disable_interrupt();
    PR00 = u16PR0sav;
    PR01 = u16PR1sav;
}
```

Workaround:

Please reduce the optimization level for the interrupt function, if the instruction 'SEL RB2' isn't generated for your interrupt function:

```
#pragma optimize = s 3
#pragma bank = 2
__interrupt void isr( void )
{
 …
}
```

No. G15	**Wrong access to local variable located on stack**

Details

In case of using multiple nested if statements (level > 4), multiple accesses to local variables located on the stack, and an optimization level greater than low, wrong code may generated for stack-access at lower if statement levels.

Workaround:

Please reduce the optimization level to low for the function showing the problem:

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

No. G16	**Internal Compiler Error may occur if calculation result is zero**

Details

Code examples where a calculation result is zero may cause an internal compiler error.

Example:

```
signed int i,k;
int test(void)
{
  k=90-(9-i)*10;
}
```

Workaround:

Try to rewrite the arithmetic expression to avoid a zero result:

```
int test (void)
{
  k=90-(90-10*i);
}
```

No. G17	**Internal Compiler Error occurs if bit complement and bit-and operation are combined in one command**

Details

If the C command to complement a special functions register bit is combined with a bit and command to mask a single bit and an assignment to an integer variable, an internal error occurs:

```
Internal Error: [CoreUtil/General]: Illegal state
```

Example:

```
#include <io78F1166_A0.h>

unsigned int IntVar;

void test(void)
{
  IntVar = ~P0_bit.no0 & 0x01;
}
```

Workaround:

Please split up the operations in separate lines of code.

```
unsigned int   IntVar;

void test(void)
{
  IntVar = ~P0_bit.no0;
  IntVar = IntVar & 0x01;

}
```

No. G18	**Wrong code generated for access to multi-dimensional array**

Details

In a case of using optimization type speed level high, the compiler may generate wrong code for the access of multi-dimensional arrays.

Example:

```
static void test (void)
{
    unsigned short x, y;

    for (y = 0; y < 8; y++){
        for (x = 0; x < 128; x++) {
            buffer[y][x] = 0x00;
        }
    }
}


void dummy( void )
{
   test();
}
```

Workaround:

Please reduce the optimization level to medium or use while instead of for loops.

No. G19	**Spurious linker warning about type conflict**

Details

The compiler could in some cases (e.g. high level of nested typedef types) emit data type incorrect debug information for typedef types. When linking with XLINK, this could result in a spurious type conflict warning:

```
Warning[w6]: Type conflict for external/entry "<object-name>", in
module file2 against external/entry in module file1; different types
```

The generated code is correct.

Workaround:

Please reduce the level of nested typedef types.

No. G20	**Extended EC++: Instantiating a template class may cause an internal error**

Details

Instantiating a template class like vector on a function type may result in an internal error

```
Internal Error: [Visit types]: Error type
```

Example:

```
enum eState { state1,state2};

template <class T, T init> class CEnum
 {
 public:
   CEnum()                              {m_Value = init; }
   operator unsigned char () const      {return (unsigned char)m_Value; }
   void operator +=(unsigned char arg){m_Value = (T)(m_Value + arg) }
 private:
      T m_Value;
 };

static      __saddr __no_init CEnum<enum eState, state1> state;

void test(void)
{
                  state += state2;
}
```

Workaround:

None.

No. G21	**Internal Compiler Error occurs if numeric constant is used as function pointer**

Details

If an numeric constant is used as function pointer, an internal compiler error occurs:

```
Internal Error: [Call]: Diagnostics: Illegal Operand
```

Example:

```
void test(void)
{
    (*(void(*)())0x1000)();
}
```

Workaround:

Use a variable instead of the numeric constant.

```
unsigned int address = 0x1000;

void test(void)
{
  (*(void(*)())address)();
}
```

No. G22 Fatal Error (Uncontrolled termination) occurs if option –Ohs is used

Details

If the following sample is compiled by using option –Ohs a fatal error occurs:

```
Fatal Error[c0000005hìø_°a´_”°°_„ìø_ˆ°°_"]: Uncontrolled termination
```

In case of using Version V4.60a and WindowsXP the user is asked to inform Microsoft about this issue.

Example:

```
typedef struct
{
   unsigned char MyByte;
}T_MYSTRUCT;

extern void func1(unsigned char *, unsigned short , unsigned short);

void func2(T_MYSTRUCT *p1, unsigned char p2)
{
   unsigned short local;

   local = (0x0040) + (p2 * 10);
   func1((unsigned char*)p1,local,10);
}


unsigned char test( void )
{
   unsigned char local1=201;
   unsigned char local2=0;
   T_MYSTRUCT local3;
   T_MYSTRUCT *plocal3 = &local3;

   do {
      func2(plocal3, local1);
      if ( plocal3->MyByte != 0x00)  {
          local2 ++;
      }
      local1++;
   } while ( local1 < 204 );
   return (local2);
}
```

Workaround:

Use either option –Ohm instead of option –Ohs or disable ‘code inlining’ by option –no_inline if option –Ohs is used

No. G23	**MISRA C 2004 Rule 17.4 triggered by mistake**

Details

MISRA C rule 17.4 is triggered by mistake for arrays included in structures:

```
Error [Pm152]: array indexing shall only be applied to objects defined
as an array type (MISRA C 2004 rule 17.4)
```

Example:

```
typedef unsigned char uint8;

void test(void);

void test(void)
{
  struct {
    uint8 u8Array[4];
  } tStruct;

  tStruct.u8Array[0] = 5u;
  tStruct.u8Array[1] = tStruct.u8Array[0];
}
```

Workaround:

Disable rule 17.4 by using the #pragma diag_suppress directive for source lines accessing an array included in a structure:
T

```
typedef unsigned char uint8;

void test(void);

void test(void)
{
  struct {
    uint8 u8Array[4];
  } tStruct;

  #pragma diag_suppress = Pm152
  tStruct.u8Array[0] = 5u;
  tStruct.u8Array[1] = tStruct.u8Array[0];
  #pragma diag_default = Pm152
}
```

No. G24	**DLIB Floating Point Function overwrites SADDR area**

Details

Some DLIB floating point functions use the SADDR area 0xFFE20 … 0xFFE27 without reserving it and therefore may override application data.

Workarounds:

- reserve the area by a dummy variable unused by the application
  `__no_init __root char dummy[8] @0xFFE20;`
- exclude the area in the segment definition in the XCL-file:
  `-Z(DATA)SADDR_I,SADDR_Z,SADDR_N=FFE28-FFEDF`

No. G25	**Misaligned structure access**

Details

In the following sample the compiler generates a misaligned access to a byte-aligned structure. The compiler uses a 16bit-instruction to write the return value of the function although the structure maybe located at an odd address.

Example:

```
typedef struct
{
unsigned char Value;
unsigned char Invers;
} TWO_CHAR;


extern TWO_CHAR func1 ( unsigned char Value );

volatile TWO_CHAR result;

void test (void)
{
    result = func1 (0xaa);  // illegal word access
}
```

Workarounds:

Increase the alignment of the structure manually:

```
#pragma data_alignment=2
volatile TWO_CHAR result;
```

No. G26 **Wrong parameter passing of far pointer**

Details

In the following sample the compiler generates wrong code during parameter passing of a far pointer, if an optimization level medium or higher is selected. Instead of the correct segment address (= high byte of 20 bit value), a fixed segment address 0xFxxxx is used.

Example:

```
typedef struct
{
   const unsigned char __far* StartAdr;
   const unsigned char __far* EndAdr;
} AREA1;


extern const unsigned char array[2][2048];
extern unsigned char func1(const unsigned char __far*, const unsigned
char __far*);

const AREA1 s2[2] = {
   { &array[0][0], &array[0][2047] },
   { &array[1][0], &array[1][2047] }
};

unsigned char result;

void test (void)
{
  while(1) {
    result = func1 ( s2[0].StartAdr, s2[1].EndAdr);
  }
}
```

Workarounds:

Use the optimization level low for the interested function:

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

No. G27	**Missing Warning about change of sign due to integer conversion**

	If the sign of a constant given in hexadecimal or octal format is changed due to an integer conversion, the compiler doesn't generate a warning (Pe068).
	
	`short test (void)`
	`{`
	`  return (0x8000);`
	`}`
	
	Workaround:
	
	Use the decimal format:
	
	`short workaround (void)`
	`{`
	`  return (32768);`
	`}`
	
	Form the next compiler version onwards a remark will be generated if the sign of a constant given in hexadecimal or octal format is changed due to an integer conversion. As result the behavior will be the same for constants given in decimal and hexadecimal format.

No. G28 **Delayed insertion of DI instruction**

If the optimization level ‘high speed’ is used, the insertion of a DI instruction may be delayed, so that the instructions of a following ‘if’-condition are execution before the interrupt was disabled.

```
#include <intrinsics.h>

typedef struct
{
  unsigned char e0;
  unsigned char e1;
} T_s1;

extern void func1 (volatile T_s1*) ;
extern void func2 (volatile T_s1*) ;

volatile unsigned char  var1;
volatile T_s1          *var2;
volatile T_s1          *var3;

void test(void)
{
   if (var1 >= 3) {
      __disable_interrupt();
      if (var2)          {
         func1(var2);
      }
      if (var3)     {
         func2(var3);
      }
      __enable_interrupt();
   }
}
```

Workarounds:

1) use the inline assembler instead of the intrinsic functions:

```
void test(void)
{
   if (var1 >= 3) {
      asm("DI");
      if (var2)          {
         func1(var2);
      }
      if (var3)     {
         func2(var3);
      }
      __enable_interrupt();
   }
}
```

2) use a different optimization setting (e.g. medium or high size)

No. G29	**Misaligned 16bit-access**

	Although the compiler option '–`disable_data_alignment`' is set, the compiler uses a word instruction to increment a 16bit variable.
	
	`unsigned short count;      // located at odd address`
	
	`void test(void)`
	`{`
	`     count++;`
	`}`
	
	Workarounds:
	
	1) don't use the compiler option '–`disable_data_alignment`' (recommended)
	2) use the `#pragma data_alignment` directive to force that the 16bit value is located at an even address:
	
	`#pragma data_alignment=2`
	` unsigned short count;   // force location at even address, even if`
	`                         // option --disable_data_alignment is set`
	`#pragma data_alignment=1`

(O) Valid Specification

Item	Date published	Document No.	Document Title
1	March 2008	UEW-7	78K IAR Embedded Workbench® IDE User Guide
2	May 2006	C78K-2	78K IAR C/C++ Compiler Reference Guide
3	May 2006	A78K-2	78K IAR Assembler Reference Guide
4	May 2005	M78K-2	78K IAR Embedded Workbench Migration Guide
5	February 2008	CS78KHW-3	78K C-SPY Hardware Debugger Systems Guide
6	December 2007	XLINK-461A	IAR Linker and Library Tools Reference Guide
7	February 2008	EWMISRAC1998-3	IAR Embedded Workbench MISRA C 1998 Reference Guide
8	March 2008	EWMISRAC2004-1	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
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

Edition	Date published	Document No.	Comment
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
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

Edition	Date published	Document No.	Comment
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
76	02-07-2009	U18447EE6VAIF00	Update EW78K V4.62a, 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	07-07-09	U18447EE6VBIF00	Item C62 added, Compiler patch icc78K V4.50e added