Integrating QA·C into IAR Embedded Workbench

First Edition

by

Eur Ing Chris Hills BSc (Hons), C. Eng., MIET, MBCS, FRGS, FRSA

PhaedruS SystemS

The Art in Embedded Systems comes through Engineering discipline.

Contents

Integrating QA·C into IAR Embedded Workbench

The most effective way of doing static analysis is to do it frequently as the code is written from within the code development IDE. This app note will explain how to integrate QA·C into the IAR Embedded Workbench (EWB) to permit continuous static analysis as the source code is written. Additionally, QA·C can enforce local coding standards at the same time if configured to do so.

Background

In 1976, before the C language was even complete, Johnson (part of the C and UNIX team with Kernighan, Ritchie & Thompson) had created static analysis and the first lint (Johnson 1979) because programmers were, according to Dennis Ritchie, using "legal but dubious constructs" (Ritchie 1993). A compiler translates syntactically correct source even if it is semantic rubbish. Some simple examples of this include assigning a long to a char and losing 3 bytes of data; falling foul of integer promotion rules, which most people misunderstand; having a local variable inadvertently mask a global variable - the list goes on and on.

Due to the "trust the programmer" ethos around the C language, static source code analysis is required to find all the legal but very dangerous things that can inadvertently get into the binary. Therefore static analysis should be used frequently. Static analysis looks at the source logically, without compiling and running it, hence "static" analysis. Static analysis can find up to 80% of non-functional bugs very quickly. Trying to find these same bugs dynamically within a running system would require extremely long and complex testing.

QA·C from Programming Research has been around since 1985 and has proved itself over the decades.

QA·C has both command line and GUI options, making it useful as a stand alone code inspection tool, or it can be called by another tool such as Make, a VCS or, as in this case, an IDE.

IAR EWB to QA·C

This integration guide assumes that you have both the IAR compiler suite (Embedded Work Bench, or EWB) and the Programming Research's QA·C (QA·C) correctly installed. This example uses the IAR EWARM 6.5 compiler suite and QA·C V8.1. However it should work, with appropriate path changes, for the IAR EWB for ARM, MSP340, V850 and M32C.

Compiler Personality

The first step is to generate a Compiler Personality File. **NOTE:** The PRQA license server must be running in order to run the Automatic Compiler Personality Generator.

To generate the compiler personality file just run the Compiler Personality Generator `AutoCmpPersonGen.exe` . The default location for this is `C:\PRQA\CPG-3.0.0\bin`. However, this requires that the `PATH` to the compiler is in the Windows Environment Variables under `PATH`. This is NOT the default for the IAR installation. To check if the compiler path is set, type "`iccarm`" in the DOS window whilst it is in the `C:\PRQA\CPG-3.0.0\bin` directory. If the compiler runs, the path is set, in which case skip the next section and go to the section PATH Set: Compiler Personality Generator

PATH not set

If the compiler `PATH` is not set in the environment variable, the method is as follows:

Open a DOS or command window by, in Win7, entering "`cmd`" in the "`search programs and files`" window when you click the Windows start button.

In the DOS window we need to change directory to the CPG directory. Start by entering "`CD c:\ `" to get to the root directory and then enter "`cd PRQA`". Then "`dir /w`" to list the files and sub directories. Finally enter the CD command to move to the bin directory in the CPG directory, in this case "`cd CPG-3.0.0\bin`". Take care to use the correct version number.

Next you need to set the path to the compiler. This uses the command

```
SET PATH=%PATH%;"path-to-compiler".
```

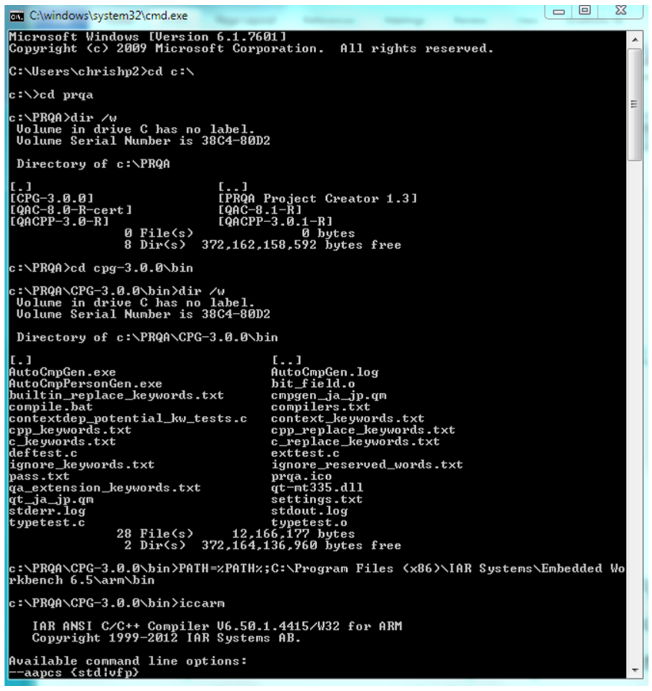
With my IAR installation this is

```
SET PATH=%PATH%; "C:\Program Files
    (x86)\IAR Systems\Embedded Workbench
    6.5\arm\bin"
```

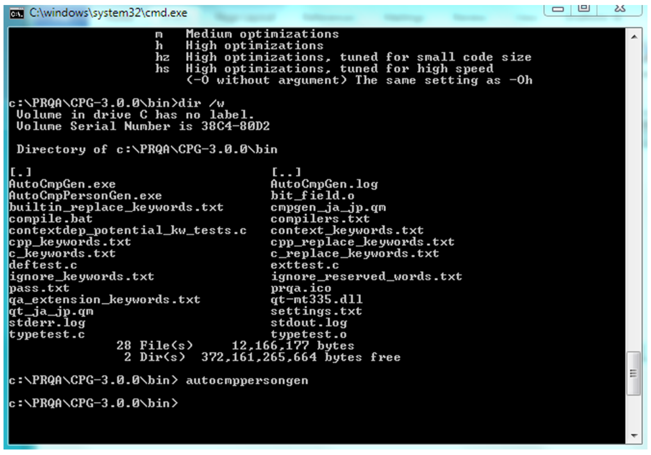
NOTE: The fastest way to get the full path information is to navigate to it in the Windows file explorer. Hit the down arrow at the end of the path window and copy it. The path can then be pasted into the DOS window using "`paste`" from the right click mouse menu. (Control-V will not work.)

To test the path, enter the command "`iccarm`" in the DOS window to invoke the compiler. You will then see the compiler information scroll past in the command window. Scroll back up to check that the compiler is the correct one. It will look something like this:

```
 C:\PRQA\CPG-3.0.0\bin>PATH=%PATH%;
C:\Program Files (x86)\IAR Systems\
Embedded Workbench 6.5\arm\bin

C:\PRQA\CPG-3.0.0\bin>iccarm
   IAR ANSI C/C++ Compiler
   V6.50.1.4415/W32 for ARM
    Copyright 1999-2012 IAR.

Available command line options:
--aapcs {std|vfp}
   Specify calling convention.
--aeabi
   Generate aeabi compliant code
--align_sp_on_irq
   Generate code to align SP on
   entry to __irq functions
```



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\chrishp2>cd c:\

c:\>cd prqa

c:\PRQA>dir /w
 Volume in drive C has no label.
 Volume Serial Number is 38C4-80D2

 Directory of c:\PRQA

[.]                         [..]
[CPG-3.0.0]                 [PRQA Project Creator 1.3]
[QAC-8.0-R-cert]            [QAC-8.1-R]
[QACPP-3.0-R]               [QACPP-3.0.1-R]
               0 File(s)              0 bytes
               8 Dir(s)  372,162,158,592 bytes free

c:\PRQA>cd cpg-3.0.0\bin

c:\PRQA\CPG-3.0.0\bin>dir /w
 Volume in drive C has no label.
 Volume Serial Number is 38C4-80D2

 Directory of c:\PRQA\CPG-3.0.0\bin

[.]                                 [..]
AutoCmpGen.exe                      AutoCmpGen.log
AutoCmpPersonGen.exe                bit_field.o
builtin_replace_keywords.txt        cmpgen_ja_jp.qm
compile.bat                         compilers.txt
contextdep_potential_kw_tests.c     context_keywords.txt
cpp_keywords.txt                    cpp_replace_keywords.txt
c_keywords.txt                      c_replace_keywords.txt
deftest.c                           exttest.c
ignore_keywords.txt                 ignore_reserved_words.txt
pass.txt                            prqa.ico
qa_extension_keywords.txt           qt-mt335.dll
qt_ja_jp.qm                         settings.txt
stderr.log                          stdout.log
typetest.c                          typetest.o
              28 File(s)     12,166,177 bytes
               2 Dir(s)  372,164,136,960 bytes free

c:\PRQA\CPG-3.0.0\bin>PATH=%PATH%;C:\Program Files (x86)\IAR Systems\Embedded Wo
rkbench 6.5\arm\bin

c:\PRQA\CPG-3.0.0\bin>iccarm

   IAR ANSI C/C++ Compiler V6.50.1.4415/W32 for ARM
   Copyright 1999-2012 IAR Systems AB.

Available command line options:
--aapcs {std|vfp}
```



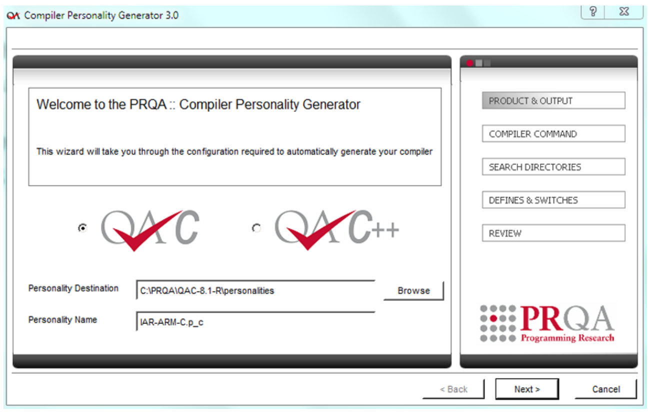
```
                  m   Medium optimizations
                  h   High optimizations
                  hz  High optimizations, tuned for small code size
                  hs  High optimizations, tuned for high speed
                      (-O without argument) The same setting as -Oh

c:\PRQA\CPG-3.0.0\bin>dir /w
 Volume in drive C has no label.
 Volume Serial Number is 38C4-80D2

 Directory of c:\PRQA\CPG-3.0.0\bin

[.]                                 [..]
AutoCmpGen.exe                      AutoCmpGen.log
AutoCmpPersonGen.exe                bit_field.o
builtin_replace_keywords.txt        cmpgen_ja_jp.qm
compile.bat                         compilers.txt
contextdep_potential_kw_tests.c     context_keywords.txt
cpp_keywords.txt                    cpp_replace_keywords.txt
c_keywords.txt                      c_replace_keywords.txt
deftest.c                           exttest.c
ignore_keywords.txt                 ignore_reserved_words.txt
pass.txt                            prqa.ico
qa_extension_keywords.txt           qt-mt335.dll
qt_ja_jp.qm                         settings.txt
stderr.log                          stdout.log
typetest.c                          typetest.o
              28 File(s)     12,166,177 bytes
               2 Dir(s)  372,161,265,664 bytes free

c:\PRQA\CPG-3.0.0\bin> autocmppersongen

c:\PRQA\CPG-3.0.0\bin>
```

PATH Set: Compiler Personality Generator

When the PATH does have the correct path to the compiler set, we can run the Compiler Personality Generator from within the DOS or command window. Enter the command "`AutoCmpPersonGen`"

The Compiler Personality Generator should start. The AutoCmpPersonGen is a windows application despite it being called from the command line in the DOS window.

Compiler Personality Generator: Screen 1


Compiler Personality Generator 3.0

Welcome to the PRQA :: Compiler Personality Generator

This wizard will take you through the configuration required to automatically generate your compiler

[✓ QA·C] QA·C++

Personality Destination: C:\PRQA\QAC-8.1-R\personalities

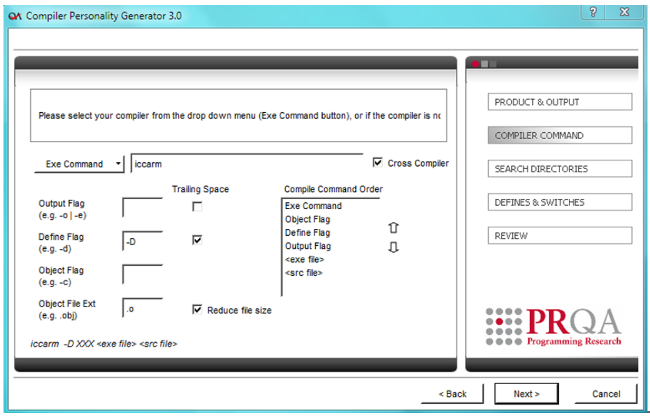
Personality Name: IAR-ARM-C.p_c

On the first page of the personality generator, ensure that QA·C is selected or QA·C++ if you are using QA·C++ *and programming in* C++. (Do not use QA·C++ for C source.)

Next you need to set the destination for the personality file. This will be the "`personality`" directory in QA·C. Finally set a name for the file in the box below the path. It should be "meaningful" as you potentially could have several personalities. In any case it needs to end in `.p_c` mine is `IAR-ARM-C.p_c`

Having completed the directory and personality name, click the "`Next`" button.

Compiler Personality Generator: Compiler Command


Compiler Personality Generator 3.0

Please select your compiler from the drop down menu (Exe Command button), or if the compiler is nc

Exe Command: iccarm

[✓] Cross Compiler

		Trailing Space
Output Flag (e.g. -o | -e)		☐
Define Flag (e.g. -d)	-D	✓
Object Flag (e.g. -c)		
Object File Ext (e.g. .obj)	.o	

[✓] Reduce file size

Compile Command Order:
Exe Command
Object Flag
Define Flag
Output Flag
<exe file>
<src file>

iccarm -D XXX <exe file> <src file>

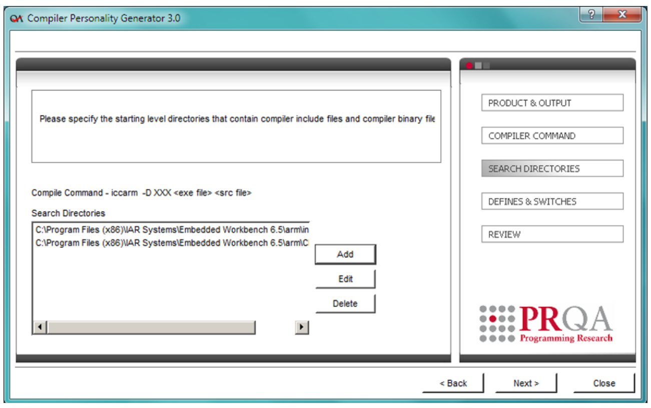
This will take you the Compiler Selection screen. Follow the screen instructions and select your compiler from the list under the EXE command button. In this case IAR and from the IAR sub menu select ARM.

The iccarm should appear in the EXE Command space and be in black text . If it is in red i.e. **`iccarm`**, the Compiler Personality Generator can't find the executable and you will need to re-check the paths.

Assuming the compiler name is in black, ensure that the "cross compiler" box is ticked and the rest of the settings are as shown. My notes say do NOT change any of the values unless you are really sure you know what you are doing (and even then it is best to check with PRQA first!).

Then click "`Next`" to get to the Search Directories screen.

Compiler Personality Generator: Search Directories


Compiler Personality Generator 3.0

Please specify the starting level directories that contain compiler include files and compiler binary file

Compile Command - iccarm -D XXX <exe file> <src file>

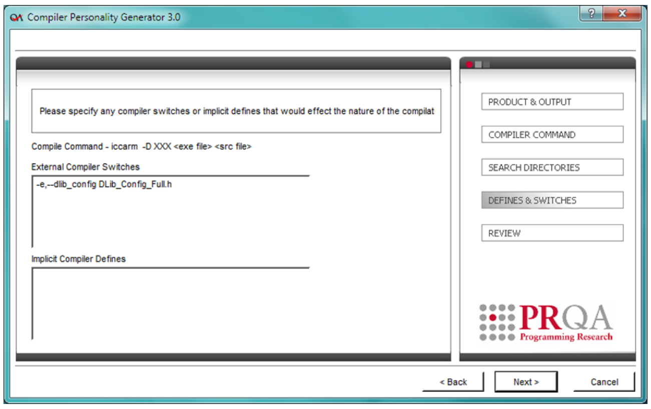
Search Directories
C:\Program Files (x86)\IAR Systems\Embedded Workbench 6.5\arm\in
C:\Program Files (x86)\IAR Systems\Embedded Workbench 6.5\arm\C

Now you need to add all the directories for the libraries. The tool will look at all subdirectories of the directory(s) you give it. So you should only need the base directory unless you are using additional libraries in other places.

```
C:\Program Files (x86)\IAR Systems\
   Embedded Workbench 6.5\arm\inc\
```

Now click the "`Next`" button to go to the Defines and Switches screen.

Compiler Personality Generator: Defines and Switches



Please specify any compiler switches or implicit defines that would effect the nature of the compilat

Compile Command - iccarm -D XXX <exe file> <src file>

External Compiler Switches

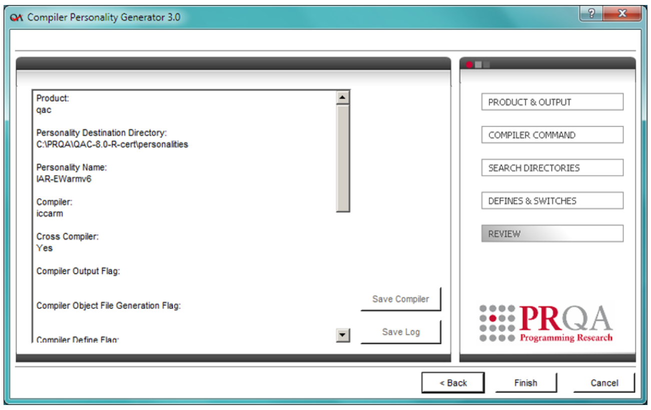
-e,--dlib_config DLib_Config_Full.h

Implicit Compiler Defines

Again my notes say (in block capitals this time) ***DO NOT CHANGE THESE UNLESS YOU REALLY KNOW WHAT YOU ARE DOING***. Even if you do think that you really know what you are doing, please call PRQA first. It is small "insignificant" changes here that can cause hours of pain later!

Press the "`Next`" button to go to the Review page.

Compiler Personality Generator: Review and Generate.



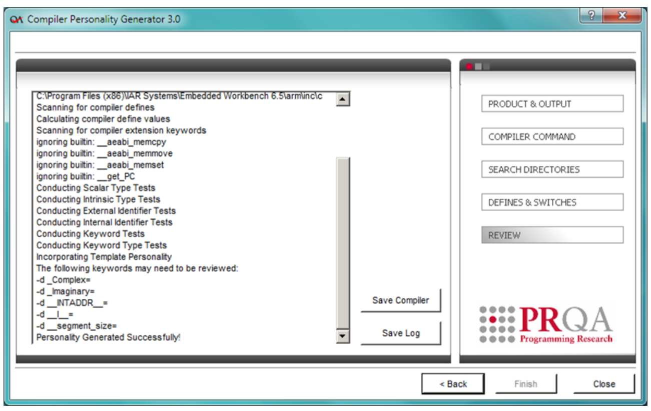
```
Product:
qac

Personality Destination Directory:
C:\PRQA\QAC-8.0-R-cert\personalities

Personality Name:
IAR-EWarmv6

Compiler:
iccarm

Cross Compiler:
Yes

Compiler Output Flag:

Compiler Object File Generation Flag:

Compiler Define Flag:
```

All being well you should see the options you selected in the previous screens for the IAR ARM compiler. Do recheck the settings and in particular the paths for the compiler and libraries. If anything is incorrect use the Back button to go back and adjust the settings.

Now you have finished, so click on the "`Finish`" button to let the tool start generating the Compiler Personality File. The tool will parse all the header files testing ALL the compiler keywords and tokens in ALL the directories and their sub directories in the search paths.

This may take some time probably several minutes, and my notes say: "go and get a cup of coffee".

The screenshot shows that a few keywords will be ignored (macro replaced with nothing), as CPG was not able to work out by running the compiler how to handle them. You will need to look at the IAR compiler manual and library sources to determine the solutions.



```
C:\Program Files (x86)\IAR Systems\Embedded Workbench 6.5\arm\inc\c
Scanning for compiler defines
Calculating compiler define values
Scanning for compiler extension keywords
ignoring builtin: __aeabi_memcpy
ignoring builtin: __aeabi_memmove
ignoring builtin: __aeabi_memset
ignoring builtin: __get_PC
Conducting Scalar Type Tests
Conducting Intrinsic Type Tests
Conducting External Identifier Tests
Conducting Internal Identifier Tests
Conducting Keyword Tests
Conducting Keyword Type Tests
Incorporating Template Personality
The following keywords may need to be reviewed:
-d _Complex=
-d _Imaginary=
-d __INTADDR__=
-d __I__=
-d __segment_size=
Personality Generated Successfully!
```

In this instance: the `_Complex` and `_Imaginary` keywords are handled acceptably as they are.

The `__segment_size` is an intrinsic function so should be removed from the compiler personality and added to the `force include .h` file (the argument of the `-fi` option in the compiler personality). QA·C has some `force include` files that contain definitions for specific compilers in `C:/PRQA/CPG-3.0.0/forceincludes`. See the QA·C Compiler Personality Generator User Guide for details on how to use them.

IMPORTANT: If you click the "`Back`" button at this point you will have to re-run the personality generation. This is because for the IAR compiler the file is saved on clicking the "`Close`" button to close the Compiler Personality Generator.

It you are producing a personality file for a compiler not on the list in the compiler selection screen you will need to explicitly save it using the "`Save Compiler`" button.

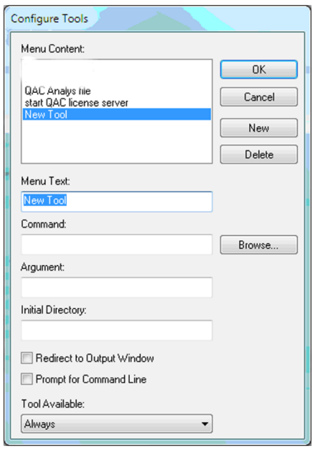
The log only needs to be saved if there is an error in generating the compiler personality file. **NOTE:** This does not include the case shown were there are some flagged keywords.

Close the Compiler Personality Generator. You will only need to run it again if you change compiler version or change the libraries. You should put a note or flag in your procedures that when you upgrade the compiler or libraries, other tools including the static analysers may need to update scripts and data files.

Now we have the compiler personality we can integrate QA·C into the IAR Embedded Workbench.

Integration of QA·C to IAR EWARM

Start the IAR EWARM compiler suite. Ensure that you have the correct ARM compiler set, since when there are multiple IAR compilers installed, the IAR EWB IDE can call any of them. The best way of doing this is to load a suitable project for the correct MCU.


Configure Tools

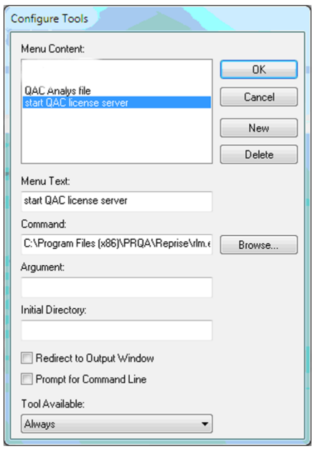
Menu Content:
QAC Analys file
start QAC license server
New Tool

Menu Text: New Tool
Command:
Argument:
Initial Directory:
[ ] Redirect to Output Window
[ ] Prompt for Command Line
Tool Available: Always

With the EWARM IDE running with a suitable project go to the "`TOOLS`" menu and select the `Configure Tools` option. This will get you the small Configure Tools dialogue shown. However, your dialogue will not have the two QAC entries shown.

Configure QA·C License Server

You are going to add an entry to start the QA·C license server. It is highly likely that unless the QA·C license server is automatically started at the PC boot time it will not be running when you want to do the static analysis.

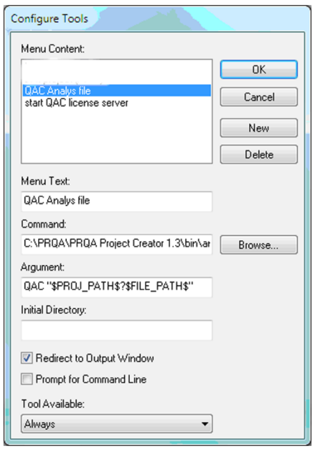

Configure Tools

Menu Content:
QAC Analys file
start QAC license server

Menu Text: start QAC license server
Command: C:\Program Files (x86)\PRQA\Reprise\rlm.e
Argument:
Initial Directory:
[ ] Redirect to Output Window
[ ] Prompt for Command Line
Tool Available: Always

Click on the "`New`" Button and change the menu text to "`Start QA·C License Server`". The Command box should contain the path to the license server. On a standard PC installation it will be `C:\Program Files (x86)\PRQA\Reprise\rml.exe` as shown in the picture. Click "`OK`" to save the dialogue.

Configure QA·C to Analyse C files


Configure Tools

Menu Content:
QAC Analys file
start QAC license server

Menu Text: QAC Analys file
Command: C:\PRQA\PRQA Project Creator 1.3\bin\ar
Argument: QAC "$PROJ_PATH$?$FILE_PATH$"
Initial Directory:
[✓] Redirect to Output Window
[ ] Prompt for Command Line
Tool Available: Always

The next step is to configure QA·C to analyse a single C file. On the IAR EWARM IDE select the "`Tools`" menu as above and click on "`Configure Tools`" and the "`New`" in the tools dialogue.

In the Menu text box enter "QA·C Analysis".

Then enter the command to run QA·C. On a standard QA·C installation, it will be `C:\PRQA\PRQA Project Creator 1.3\bin\analyse _ ide _ proj.bat`

Next you will need to enter the arguments which will be: `QA·C "$PROJ _ PATH$?$FILE _ PATH$"`

The syntax here is quite specific. `PROJ _ PATH` and `FILE _ PATH` are internal IAR EWB variables. They are not available outside the IAR environment.

Adusting Settings in the script files

The last thing to do is to adjust the script_setting.bat file. This file lives in the PR QA·C Project Creator installation `bin` directory. This file contains default vales for various settings and information on where the QA·C tool is installed. Some of the default settings can be over ridden by the `project _ settings.bat` file. The Project file will reside in the same directory as the IAR project file. The `analyse _ ide _ proj.bat` copies it there on its first run.

In the `script _ setting.bat` file you need to add the compiler personalities and any local coding standards or MISRA-C personalties.

Setting the compiler is done with C_IDE_COMPILER_PERSON

```
C_IDE_compiler_Person = "C:\PRQA….
    Personalities\IAR-arm-c.P_c
```

For the Message personality, e.g. a local coding standard it is `C _ MESSAGE _ PERSON` which requires a `.p _ s` file and for the Analyser personality `C _ ANALYSER _ PERSON which uses a -P _ a file.` For a MISRA-C personality these need to point by default as shown below:

```
C_MESSAGE_PERSON=%PRQA_INSTALL_
    DIR%\%QA·C_DIR%\m2cm\personalities\
    m2cm.p_s"
C_ANALYSER_PERSON="%PRQA_INSTALL_
    DIR%\%QA·C_DIR%\m2cm\personalities\
    m2cm.p_a"
```

NOTE:

For MISRA-C 1998 (C1) the files are:

`mcm.p _ s` and `mcm.p _ a`

For MISRA-C:2004 (C2) the files are:

`m2cm.p _ s` and `m2cm.p _ a`

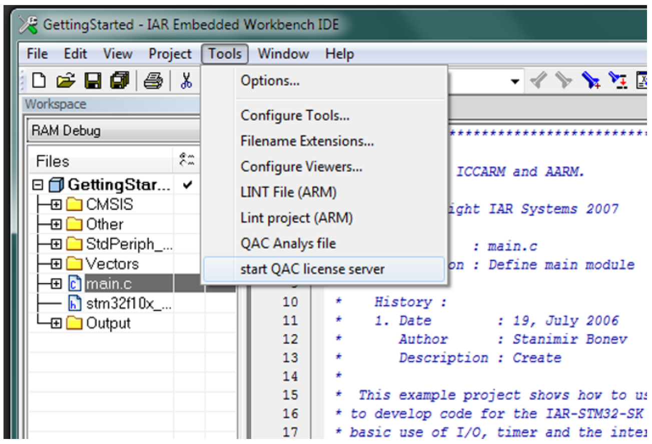
For MISRA C:2012 (C3) the files are:

`m3cm.p _ s` and `m3cm.p _ a`

Testing installation

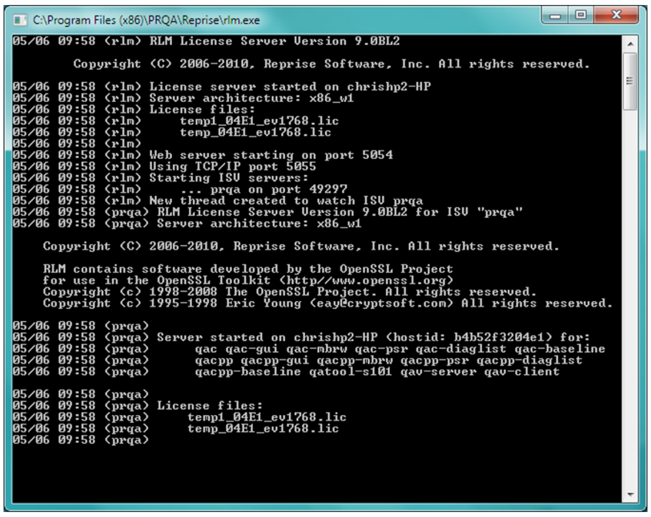
Start the configured IAR Embedded Workbench and load/write a simple "Hello World" type project using standard C with no IAR specific compiler extensions.

If it is not already running start the QA·C license server from the tools menu in the IAR Embedded Workbench.



```
************************
ICCARM and AARM.
ight IAR Systems 2007
   : main.c
on : Define main module
*   History :
*   1. Date         : 19, July 2006
*      Author       : Stanimir Bonev
*      Description  : Create
*
*  This example project shows how to us
* to develop code for the IAR-STM32-SK
* basic use of I/O, timer and the inte
```

Note: you cannot run more than one instance of the license server. The server will start a DOS box.

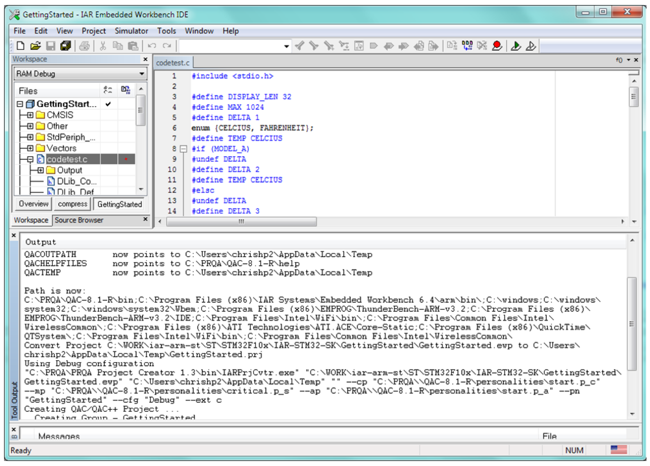


```
05/06 09:58 (rlm) RLM License Server Version 9.0BL2

        Copyright (C) 2006-2010, Reprise Software, Inc. All rights reserved.

05/06 09:58 (rlm) License server started on chrishp2-HP
05/06 09:58 (rlm) Server architecture: x86_w1
05/06 09:58 (rlm) License files:
05/06 09:58 (rlm)     temp1_04E1_ev1768.lic
05/06 09:58 (rlm)     temp_04E1_ev1768.lic
05/06 09:58 (rlm)
05/06 09:58 (rlm) Web server starting on port 5054
05/06 09:58 (rlm) Using TCP/IP port 5055
05/06 09:58 (rlm) Starting ISV servers:
05/06 09:58 (rlm)     ... prqa on port 49297
05/06 09:58 (rlm) New thread created to watch ISV prqa
05/06 09:58 (prqa) RLM License Server Version 9.0BL2 for ISV "prqa"
05/06 09:58 (prqa) Server architecture: x86_w1

    Copyright (C) 2006-2010, Reprise Software, Inc. All rights reserved.

    RLM contains software developed by the OpenSSL Project
    for use in the OpenSSL Toolkit (http//www.openssl.org)
    Copyright (c) 1998-2008 The OpenSSL Project. All rights reserved.
    Copyright (c) 1995-1998 Eric Young (eay@cryptsoft.com) All rights reserved.

05/06 09:58 (prqa)
05/06 09:58 (prqa) Server started on chrishp2-HP (hostid: b4b52f3204e1) for:
05/06 09:58 (prqa)      qac qac-gui qac-mbrw qac-psr qac-diaglist qac-baseline
05/06 09:58 (prqa)      qacpp qacpp-gui qacpp-mbrw qacpp-psr qacpp-diaglist
05/06 09:58 (prqa)      qacpp-baseline qatool-s101 qav-server qav-client

05/06 09:58 (prqa)
05/06 09:58 (prqa) License files:
05/06 09:58 (prqa)     temp1_04E1_ev1768.lic
05/06 09:58 (prqa)     temp_04E1_ev1768.lic
05/06 09:58 (prqa)
```

DO NOT CLOSE the DOS window or it will close the server and QA·C will not run. You can minimise the DOS window. Do check that the server is running correctly and has picked up the correct license files, which are displayed on the last two lines of the server start-up in the DOS window.

Set focus to the source code file in the editor. This is important and a bit awkward in EW-ARM as almost any action seems to remove the focus.

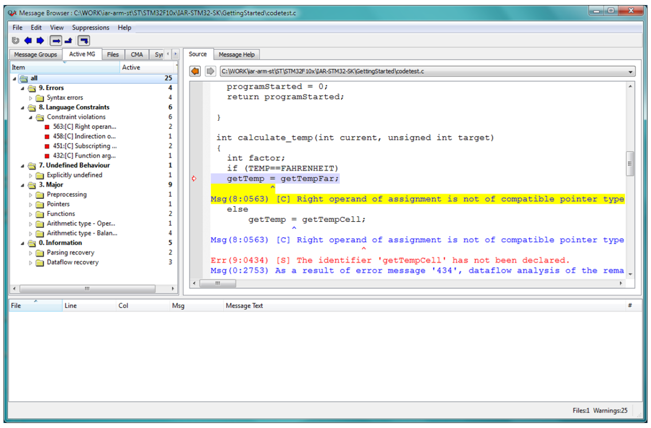
I find the best way of doing it is to click on the tab of the file you want to analysis and then the next action is to select the analysis in the tool menu. Doing anything else seems to remove the focus.



```
#include <stdio.h>

#define DISPLAY_LEN 32
#define MAX 1024
#define DELTA 1
enum {CELCIUS, FAHRENHEIT};
#define TEMP CELCIUS
#if (MODEL_A)
#undef DELTA
#define DELTA 2
#define TEMP CELCIUS
#elsc
#undef DELTA
#define DELTA 3
```

```
Output
QACOUTPATH       now points to C:\Users\chrishp2\AppData\Local\Temp
QACHELPFILES     now points to C:\PRQA\QAC-8.1-R\help
QACTEMP          now points to C:\Users\chrishp2\AppData\Local\Temp

Path is now:
C:\PRQA\QAC-8.1-R\bin;C:\Program Files (x86)\IAR Systems\Embedded Workbench 6.4\arm\bin\;C:\windows;C:\windows\
system32;C:\windows\system32\Wbem;C:\Program Files (x86)\EMPROG\ThunderBench-ARM-v3.2;C:\Program Files (x86)\
EMPROG\ThunderBench-ARM-v3.2\IDE;C:\Program Files\Intel\WiFi\bin\;C:\Program Files\Common Files\Intel\
WirelessCommon\;C:\Program Files (x86)\ATI Technologies\ATI.ACE\Core-Static;C:\Program Files (x86)\QuickTime\
QTSystem\;C:\Program Files\Intel\WiFi\bin\;C:\Program Files\Common Files\Intel\WirelessCommon\
Convert Project C:\WORK\iar-arm-st\ST\STM32F10x\IAR-STM32-SK\GettingStarted\GettingStarted.ewp to C:\Users\
chrishp2\AppData\Local\Temp\GettingStarted.prj
Using Debug configuration
"C:\PRQA\PRQA Project Creator 1.3\bin\IARPrjCvtr.exe" "C:\WORK\iar-arm-st\ST\STM32F10x\IAR-STM32-SK\GettingStarted\
GettingStarted.ewp" "C:\Users\chrishp2\AppData\Local\Temp" "" --cp "C:\PRQA\\QAC-8.1-R\personalities\start.p_c"
--mp "C:\PRQA\\QAC-8.1-R\personalities\critical.p_s" --ap "C:\PRQA\\QAC-8.1-R\personalities\start.p_a" --pn
"GettingStarted" --cfg "Debug" --ext c
Creating QAC/QAC++ Project ...
  Creating Group - GettingStarted
```

You should see, in the IAR EW-ARM Tool Output window, the responses from QA·C running, though not the analysis messages. Do check these messages, as it will show the start up of the QA·C and the files called. This window will also hold the clues as to any failures if QA·C does not start



```
  programStarted = 0;
  return programStarted;

}

int calculate_temp(int current, unsigned int target)
{
  int factor;
  if (TEMP==FAHRENHEIT)
  getTemp = getTempFar;
             ^
Msg(8:0563) [C] Right operand of assignment is not of compatible pointer type
  else
      getTemp = getTempCell;
              ^
Msg(8:0563) [C] Right operand of assignment is not of compatible pointer type
                             ^
Err(9:0434) [S] The identifier 'getTempCell' has not been declared.
Msg(0:2753) As a result of error message '434', dataflow analysis of the rema
```

All being well the QA·C front end will automatically start and open its own message browser. Look at the analysis in QA·C.

Test the system by adding a line to the project using an IAR specific extension.

For example `_ _arm _ _arm int j;`

Then re-run QA·C from the QA·C Analysis option in the Tools Menue and examine the output. There should not be an error for this line if the IAR –ARM personality file is correct and the settings.bat has been correctly configured.

References

IAR Embedded Workbench IDE User Guide

Johnson, S. C. (1979) Lint, a Program Checker. Unix Programmer's Manual, Seventh Edition 2B,

Lindgren, A. MISRA C—Some key rules to make embedded systems safer. IAR, IAR.

Ritchie, D. M. (1993). The Development of the C Language. ACM Second History of Programming Languages. Cambridge, Mass., AMC: 16.

QA·C CPG USER GUIDE

Integrating QA·C into IAR Embedded Workbench

First edition June 2014

Phaedrus Systems Library

The Phaedrus SystemsLibrary is a collection of useful technical documents on development. This includes project management, integrating tools like QA·C to IDE's, the use of debuggers, coding tricks and tips. The Library also includes the QuEST series.

Copies of this paper (and subsequent versions) with the associated files, will be available with other members of the Library, at:

http://library.phaedsys.com

PhaedruS SystemS

The Art in Embedded Systems comes through Engineering discipline.