Integrating QA·C into IAR Embedded Workbench

First Edition
by

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

The Art in Embedded Systems comes through Engineering discipline.
Contents

Background .................................................................................................................. 3

IAR EWB to QA·C ........................................................................................................ 3

Compiler Personality .................................................................................................. 4
  PATH not set .............................................................................................................. 4
  PATH Set: Compiler Personality Generator ............................................................. 5
  Compiler Personality Generator:
    Screen 1 ............................................................................................................ 5
  Compiler Personality Generator:
    Compiler Command ............................................................................................ 5
  Compiler Personality Generator:
    Search Directories ............................................................................................. 5
  Compiler Personality Generator:
    Defines and Switches ......................................................................................... 6
  Compiler Personality Generator:
    Review and Generate ......................................................................................... 6

Integration of QA·C to IAR EWARM ........................................................................ 7
  Configure QA·C License Server .............................................................................. 7
  Configure QA·C to Analyse C files ......................................................................... 7
  Adjusting Settings in the script files ...................................................................... 7

Testing installation .................................................................................................... 8

References .................................................................................................................... 9
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
```
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

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: Compiler Command
Compiler Personality Generator: Defines and Switches

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.

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.

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.

If 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.

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.

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

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.

Adjusting 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 values 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 personalities.

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-C2004 (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 QAC license server from the tools menu in the IAR Embedded Workbench.

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

DO NOT CLOSE the DOS window or it will close the server and QAC 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.

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

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

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

For example

```c
_ _ arm _ _ arm int j;
```

Then re-run QAC from the QAC Analysis option in the Tools Menu 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


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


QA·C CPG USER GUIDE
The Art in Embedded Systems comes through Engineering discipline.

**Integrating QA·C into IAR Embedded Workbench**

First edition June 2014  
© Copyright Chris A Hills 2014

The right of Chris A Hills to be identified as the author of this work has been asserted by him in accordance with the Copyright, Designs and Patents Act 1988

**Phaedrus Systems Library**

The Phaedrus Systems Library 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](http://library.phaedsys.com)