APPLICATION NOTE

Converting Firmware Projects to Colde and IAR Embedded Workbench for ARM

Power Application Controller™

Marc Sousa
Senior Manager, Systems and Firmware

www.active-semi.com

TABLE OF CONTENTS

OVERVIEW

Users of the PAC52XX family of controllers have different choices when it comes to development environments. The PAC52XX family of controllers support two main development environments:

- CooCox CoIde
- IAR Embedded Workbench for ARM

Users will choose one of these toolsets based on several factors: performance, cost, support and existing skill sets and knowledge.

CooCox CoIde

CooCox's CoIde is an Integrated Development Environment (IDE) that supports ARM Cortex MCU based controllers, such as the PAC52XX family. It is a free IDE, based on the Eclipse platform. It supports all the major IDE features and functions such as code editor, compilation tools (which are GCC), debugger and peripheral register viewer.

CooCox also has a SWD emulator used for programming and debugging called CoLinkEx, and CooCox makes the design available for users for this emulator as well. Active-Semi makes a version of this emulator available for customers which offers high-voltage protection of your PC for high-voltage motor and power applications.

As of version 1.7.7, CoIde has integrated support for Active-Semi's PAC5210, PAC5220 and PAC5220 and this is the recommended version to download.

The CooCox CoIde is available for download on the web at the following URL:

http://www.coocox.org/CooCox_CoIDE.htm

IAR Systems Embedded Workbench for ARM

IAR System's Embedded Workbench for ARM supports all available ARM cores, from all vendors. Although there is no specific support for the PAC52XX family, the toolset may be customized to support Active-Semi devices through configuration files from Active-Semi.

IAR Embedded Workbench is a commercial product that must be licensed through IAR Systems. Customers with existing licenses may add support for Active-Semi products under their existing license, as long as they are currently using IAR Embedded Workbench for ARM.

Like CooCox, IAR Embedded Workbench for ARM has a code editor and project manager, compiler and debugger. IAR Embedded Workbench for ARM supports a wide variety of emulators for debugging that are also available for purchase.

More information on IAR Embedded Workbench for ARM can be found at the following URL:

http://www.iar.com/Products/IAR-Embedded-Workbench/ARM/

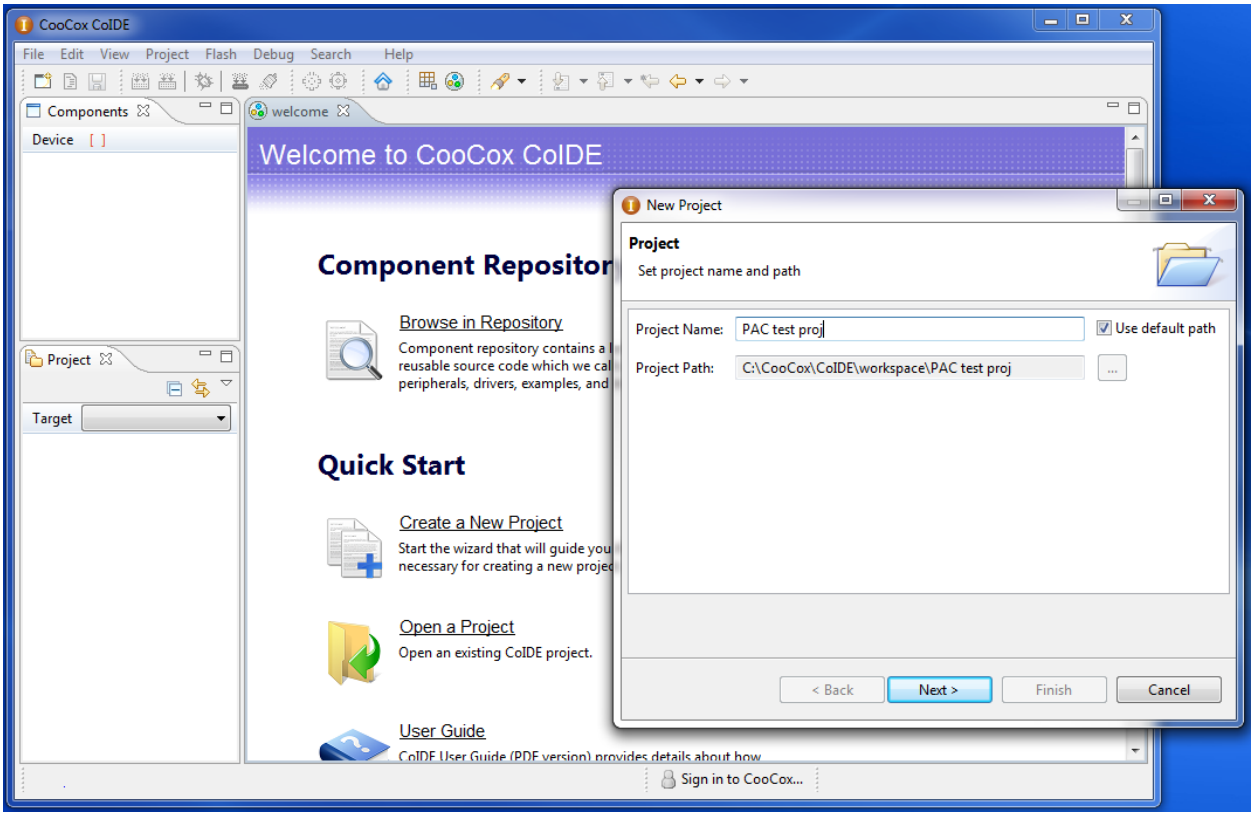
The sections below will illustrate how to convert from other firmware projects to CooCox CoIde and IAR Systems Embedded Workbench for ARM.

CONVERTING TO COOCOX COIDE

Creating a Project for the PAC52XX with CoIde

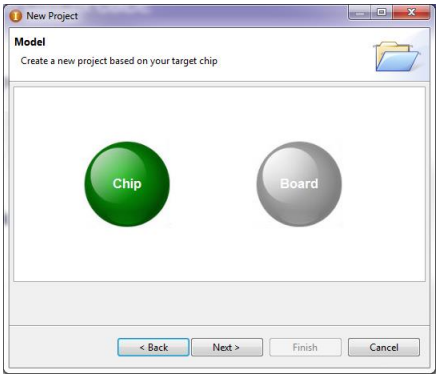
Users should begin by downloading version 1.7.7, or later, and installing on their system. For instructions on installation, refer to the online documentation from http://www.coocox.com. [1]

When you launch CoIde, on the main screen select “Create a New Project” under “Quick Start” as shown below. You will then see the “New Project” Form.



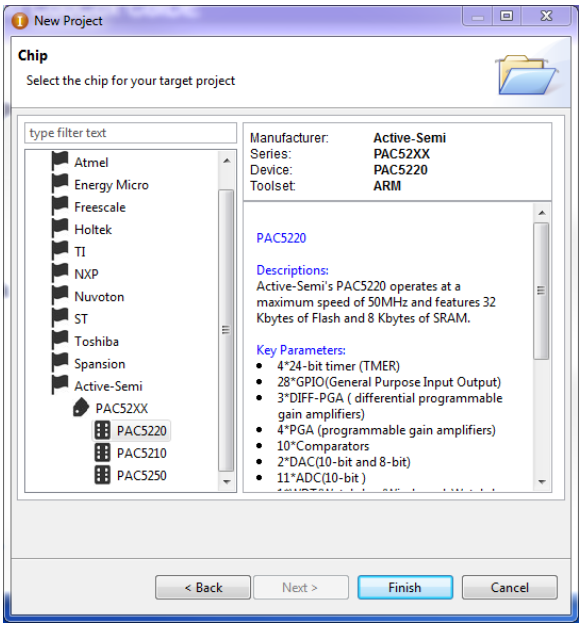
Next, type in the name of the project and select your Workspace (or use the default workspace) and press the “Next >” button. You will then see the following form.

[1] Be sure to follow the step to select the tool chain path, or building programs will not work.



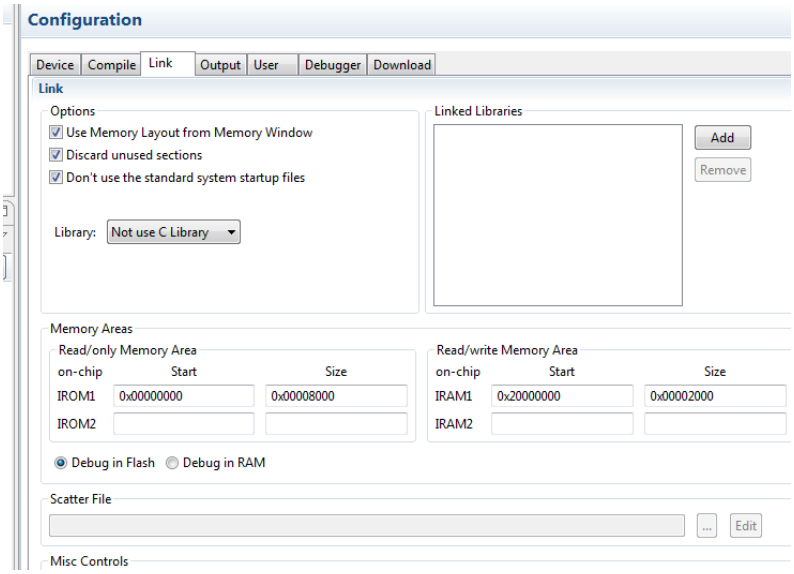
Select “Chip” and press the “Next >” button.

You will now see a form where you can select the chip vendor and device. At the bottom of this form, select “Active-Semi”, then “PAC52XX” and then the device you are using (for example, the PAC5220).

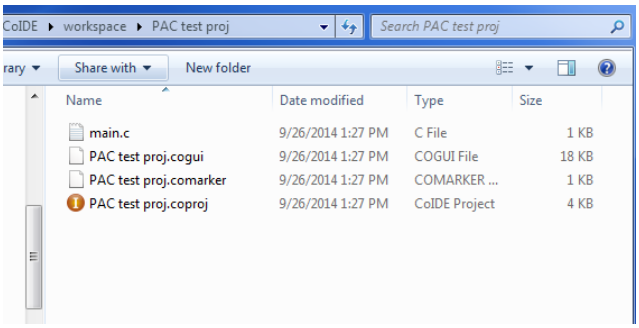


After selecting the device, click the “Finish” button and the project will be created.

You can examine the configuration options by right-clicking on the project and selecting “Configuration”. You will then see the “Configuration” tab and you can examine the project parameters. Below shows the “Link” tab which shows linker parameters for the PAC52XX.



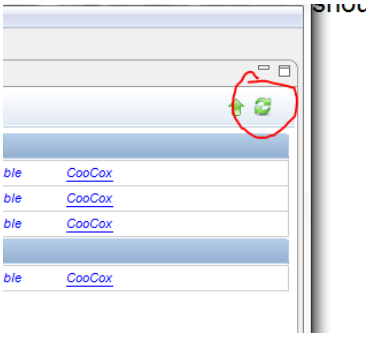
The project will be created with just one file (main.c) and will now build. The project directory should look like shown below.



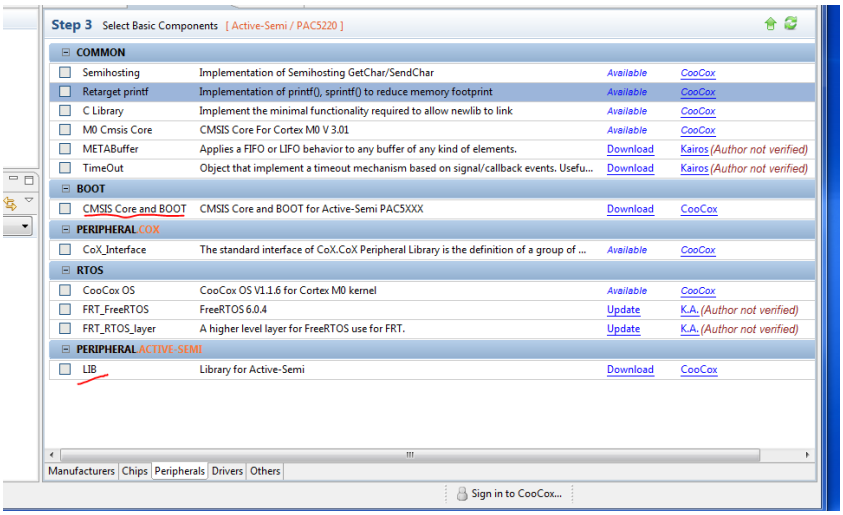
Press the “F7” key to build this simple project to verify that the default project was correctly created.

Configuring CoIde for the PAC52XX SDK

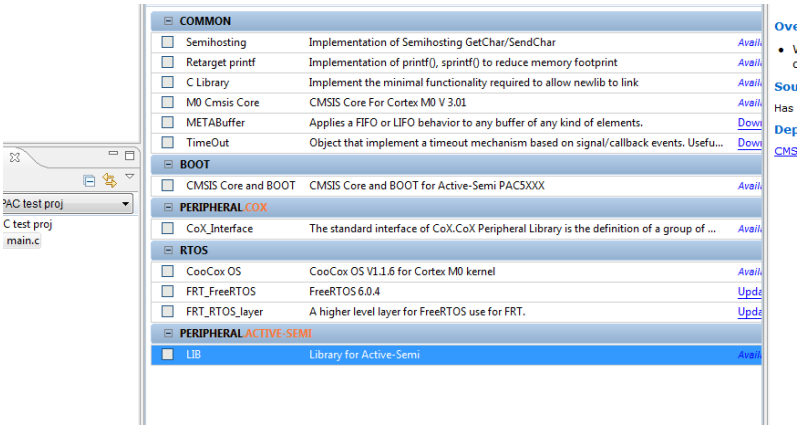
Next, CoIde will need to download and configure support for the PAC52XX SDK. To start this process, select the “Repository” view and click the “Refresh Basic Component List in the upper right hand corner.



CoIde will now download the components needed for Active-Semi from CooCox. When complete, the repository will now show the “CMSIS Core and BOOT” and “LIB” components for Active-Semi as shown below.



The user should then check both of these boxes, in order to build programs for Active-Semi’s PAC52XX family. When you check each of these boxes, CoIde will ask if you want to download the component. Click “OK”, and then after these downloads, this form should show these components as “installed”, as shown below.



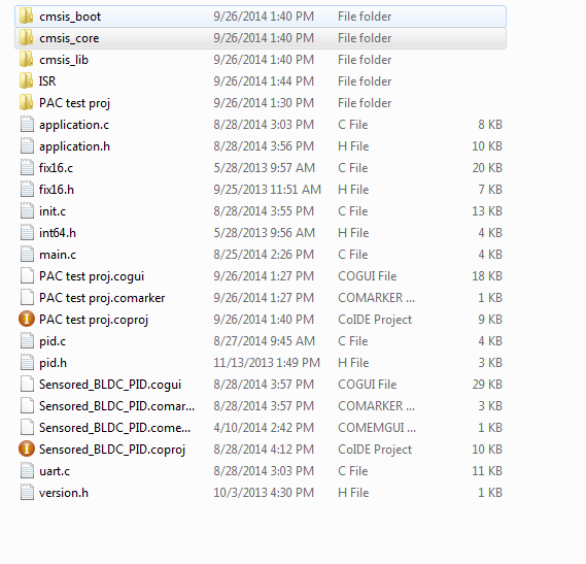
This is the CMSIS boot and PAC52XX SDK support needed to build programs for the PAC52XX family. To be sure that these are properly installed, build the project by pressing the “F7” key.

Now, the entire toolset configuration is all set for porting the firmware from another IDE.

Copying Application Files to CoIde

The user should now copy all the application firmware to the project directory. In this example, I have copied over a project into the project directory from a BLDC motor controller. The steps I have followed are below.

Next, copy application firmware files and directories. Do NOT copy any CMSIS, SDK or other project files from the old installation or old IDE. The project file should look similar to this.



Now all the files are in the project directory, needed for the build.

The user must now add the files to the project, as follows.

- For files, user must right-click on the project name (in the Project frame), and select “Add Files”, then select the files they wish to add
- If the user wishes to add a group (like a directory) to the project, then they should right-click on the project name, and select “Add Group”. In the example below, the group “ISR” has been added by the user. The groups “cmsis_boot”, “cmsis_core” and “cmsis_lib” have been added by CoIde, when the Active-Semi SDK has been configured in the project through the repository
    - If the user has added a group, to add files to the group the user right-clicks the group, then selects “Add Files” to add files to this group.

Finally, the project may be built by pressing the “F7” key.

CONVERTING TO IAR SYSTEMS EMBEDDED WORKBENCH FOR ARM

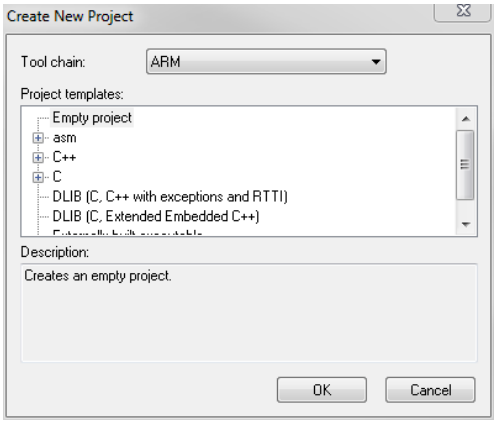
Creating a Project for the PAC52XX with IAR Embedded Workbench for ARM

Users should begin by downloading and installing the version of IAR Embedded Workbench for ARM on their system, according to the installation instructions.

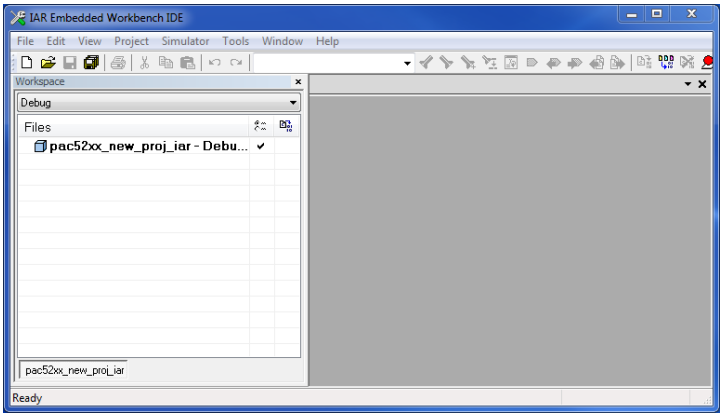
IAR Embedded Workbench does not have integrated support for the PAC52XX, so the user needs to perform some one-time configuration of the toolset before projects may be created. These instructions are available in the SDK for IAR on the Active-Semi website.

Once the PAC52XX support is installed into the IAR Embedded Workbench for ARM, and new project for the PAC52XX can be created.

The first step is to create a new project by selecting “Project” and then “New Project…” from the main menu. A dialog box will appear, and the user should select an empty project.



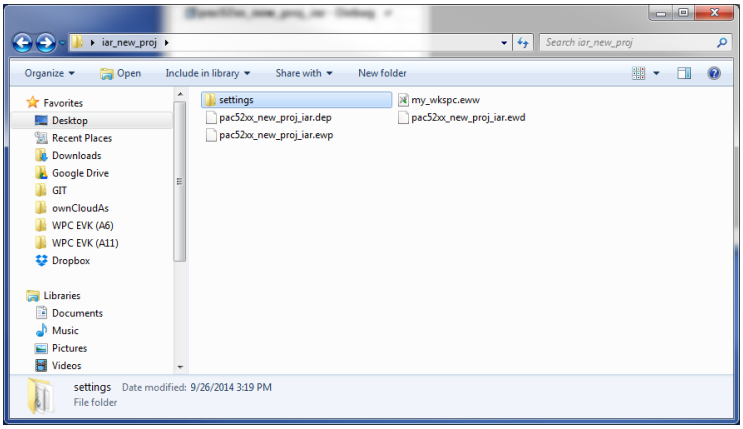
After this, the new project has been created in IAR Embedded Workbench for ARM as shown below.



For each new project, and IAR “workspace” must be created. The workspace is a file that references a set of different projects so that the user can have a set of projects to manage at the same time in the IDE.

If the user has not already selected a workspace, then they should save one by selecting “Save Workspace” from the “File” menu.

After the project and workspace have been saved, the project directory should contain the following files.

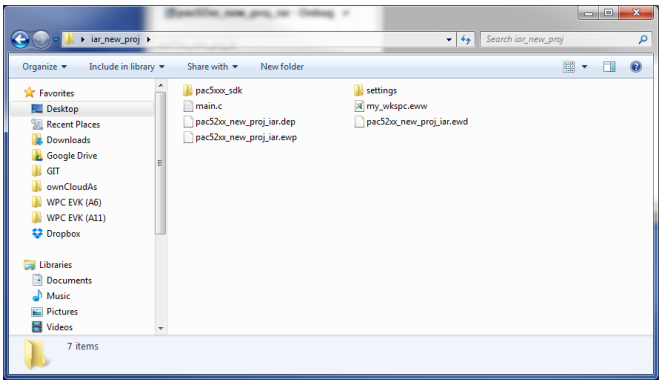


The project directory will now contain the IAR project files (*.dep, *.ewp, *.ewd) and optionally the workspace file (*.eww).

Copying Files to the IAR Project Directory

Once the project has been completed, then files may be added to the project directory.

Unlike CoIde, the SDK is not integrated into IAR Embedded Workbench, so the IAR SDK must be downloaded from the Active-Semi website, and then copied into the project directory. In the example below, the SDK files are put directly into the project directory, and a main.c file is created to call some of the SDK functions.



Once these files are copied into the project directory, the files must be added to the project.

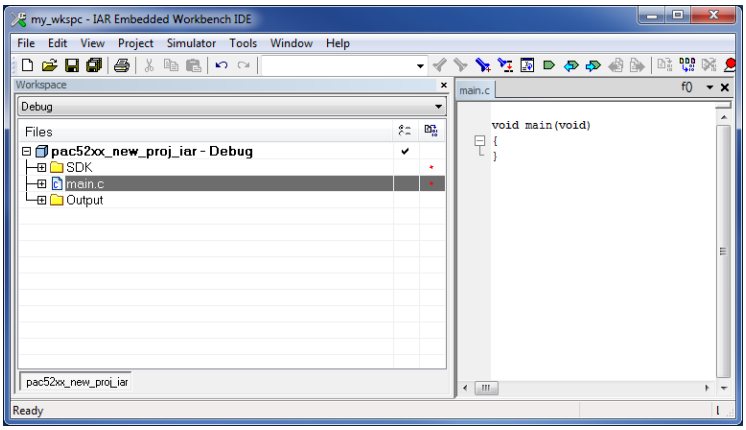
To add files, the user right-clicks on the project (in the Workspace tab) and selects “Add”, then “Add Files…”. The user may then select the files to add to the project.

To group files into directories in the project, the user adds a group by right clicking the project name, then selecting “Add”, then “Add Group….”. The group name is added and then users may add files to this group by right-clicking on the group and selecting “Add” and then “Add Files…”.

NOTE: Be sure to NOT copy any project files or any files that begin with "." to this directory, that come from the previous toolset.

For projects that use the PAC52XX SDK, all the SDK files must be added to the project.

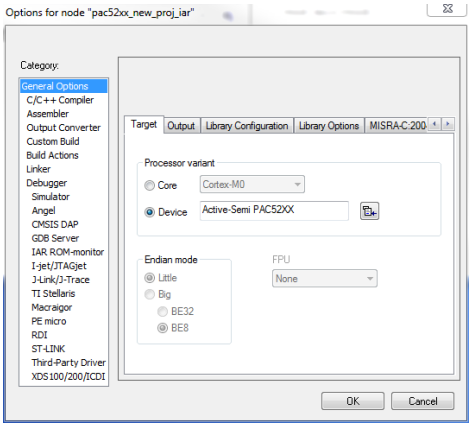
Once the files have all been added to the project, the IDE will look something like this.



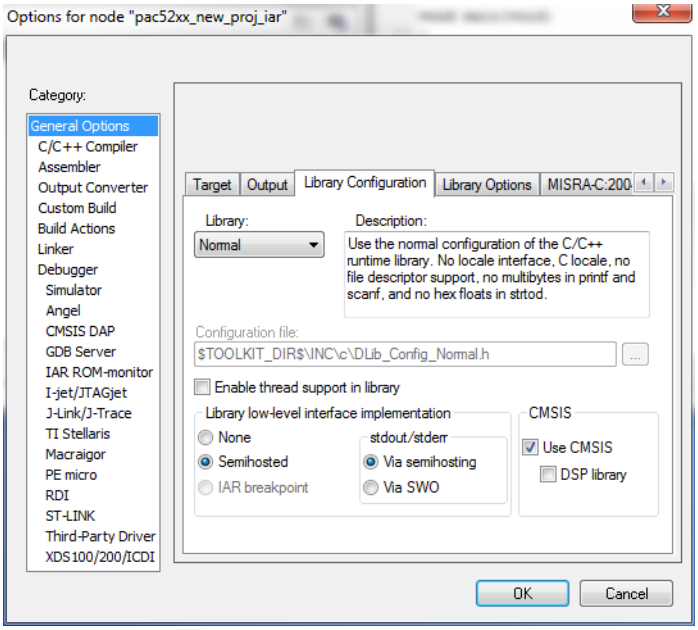
Configuring the IAR Project for PAC52XX

Once all files have been added to the IAR project, then the project must be configured for the PAC52XX device.

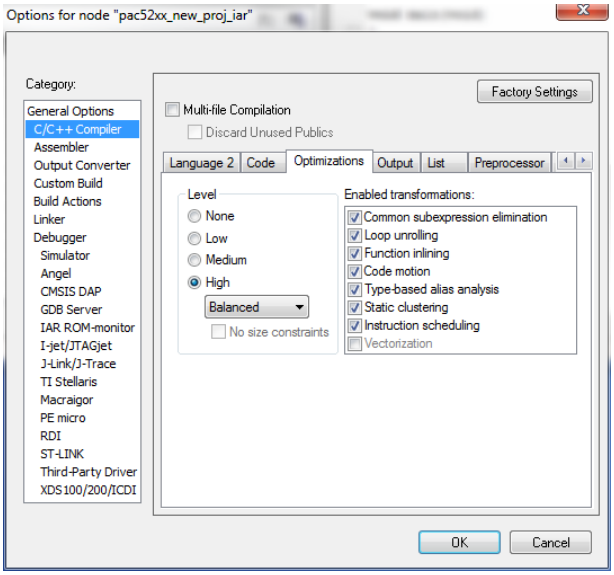
The user must first select the PAC52XX device to be used. To do this, the user must right-click the project in the Workspace tab on the left, and select "Options…". The option dialog box will come up and the user should select the "General Options" tab. The user should then select the Active-Semi PAC52XX device as shown below.



The user should then go the "Library Configuration" tab and select the "Use CMSIS" check-box. CMSIS is used for the PAC SDK and some of the other ARM peripheral header files for using the ARM Cortex peripherals.



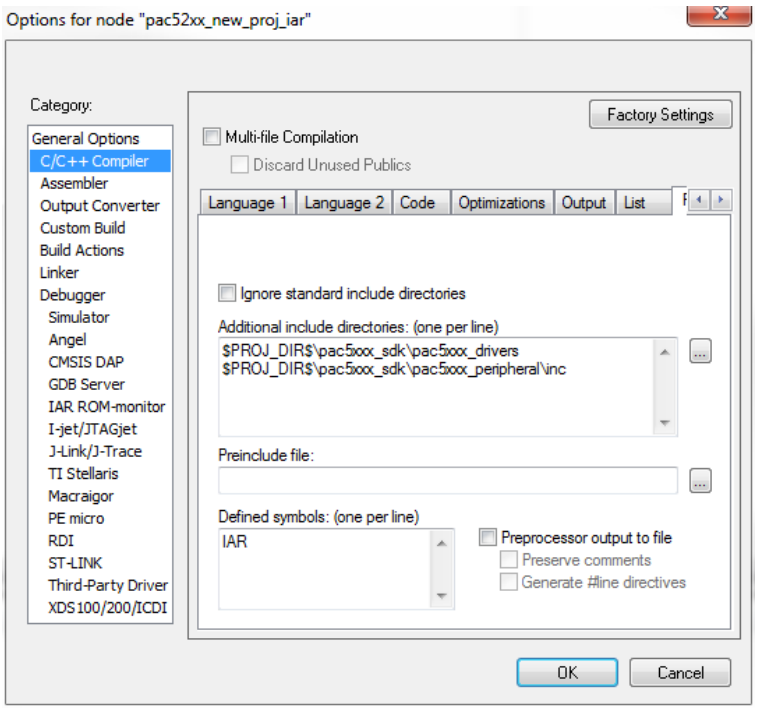
Next, the user should select the “C/C++ Compiler” category on the left and the “Optimizations” tab to select the level of compiler optimization the application needs. In most cases, the higher levels are used (High/Balanced) to provide adequate optimization.



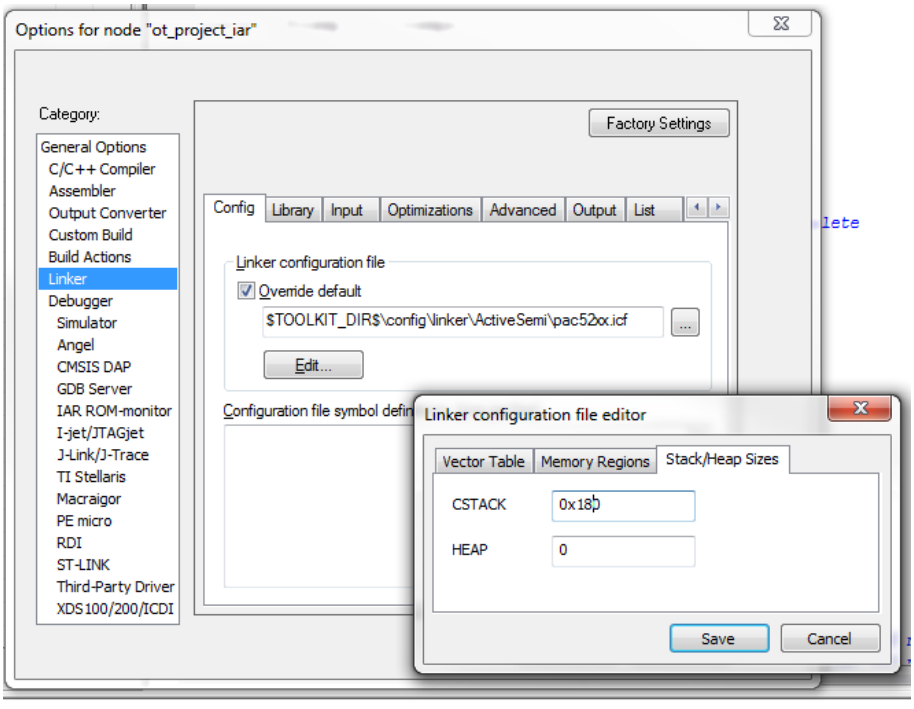
The user must now configure the include paths and symbols used by the application, by clicking on the “Preprocessor” tab on this form.

In the “Additional include directories: (one per line)” text box, the user should add the include paths for all header files needed to build the program. For example, the SDK include paths for the header files should be added.

The user must also define the symbol “IAR” on this form. When complete, this form should look as follows below.



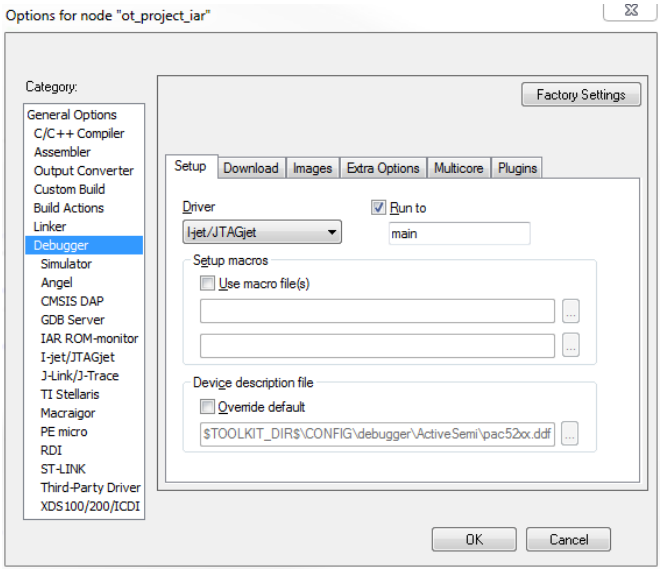
The IAR toolset tends to use more stack space than CooCox CoIde. To change the stack space for the project, the user should select the “Linker” list item then check the “Override default” check box, and enter the new stack size in the tool window as shown below.



In this example, the user has changed the stack size to 0x180.

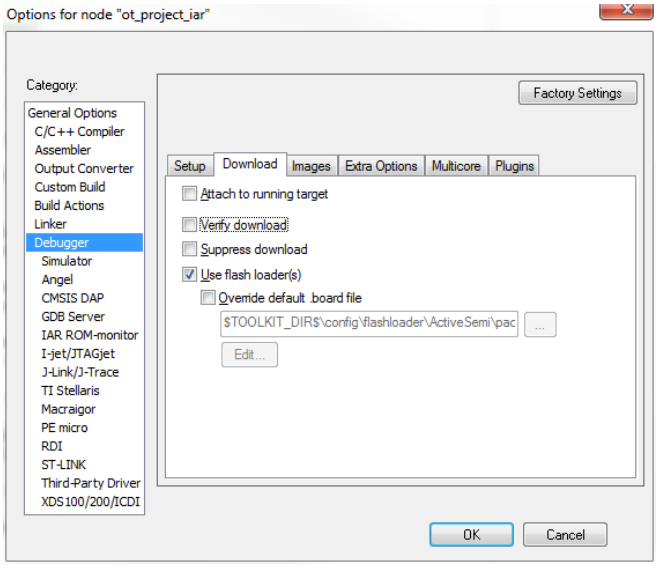
The project should now be fully configured so that it can be built. It may be built by pressing the “F7” button.

To configure the debugger for use with the I-Jet, the user should select the “Debugger” item from the list and select the “I-jet/JTAGjet” selection from the pull-down menu:



After making this selection, the user should select the “Download” tab to select the FLASH loader.

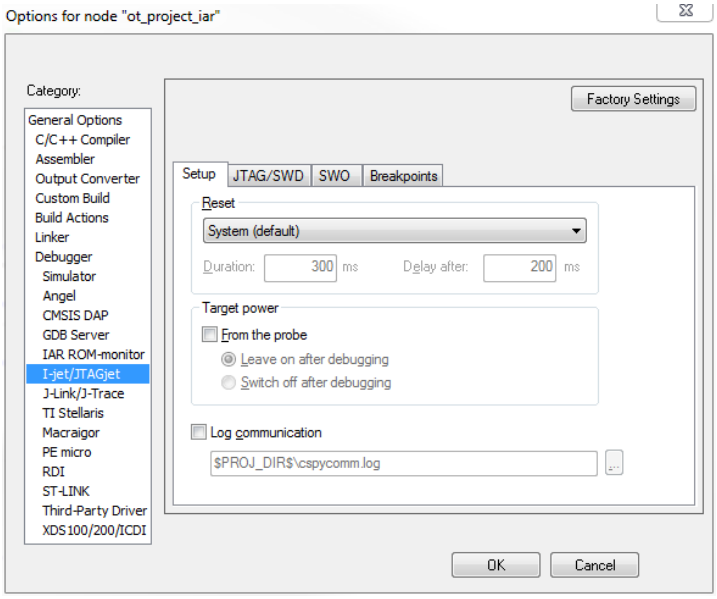
Next, the user should select the “Download” tab, to configure the adapter:



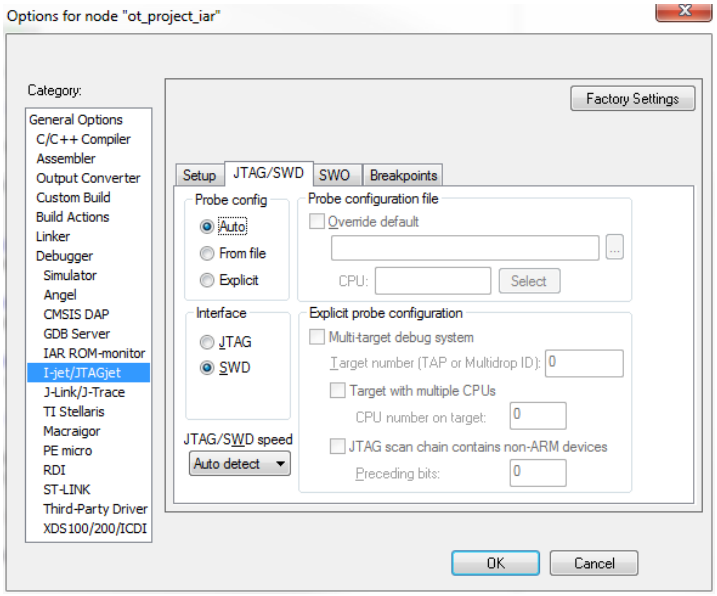
The tools will now use the FLASH loader for the PAC52XX to download the programs to the target.

Next, the user should instruct the debugger to turn off power to the target, and select SWD as the communications interface for the debugger.

To disable the adapter powering the target the user should select the “I-jet/JTAGjet” from the pull-down menu and uncheck the “From the probe” check-box from the target power group box:



Finally, the user should select SWD by changing to the “JTAG/SWD” tab and selecting the “SWD” radio button from the “Interface” group box:



COMPILING FUNCTIONS TO LINK IN RAM

To optimize performance in the PAC architecture, time-critical functions may execute in RAM, instead of FLASH memory. Different sets of tools have different ways to force the linker to mark a function to be linked into RAM (the .data section) instead of FLASH (the .code section).

If using the PAC52XX SDK, the user may mark functions to be complied in RAM as follows:

```
PAC5XXX_RAMLINK void my_function(void)
{
 /* Function */
}
```

The MACRO “PAC5XXX_RAMLINK” is defined in pac5xxx.h. This MACRO places a token before the function, which allows the compiler and linker to identify it to place into RAM.

If the user is not using this MACRO, each supported toolset has a different way to link functions into RAM.

To link a function into RAM using CooCox CoIde, the user needs to add the token “__attribute__((section(“.data”))) before the function, as follows:

```
__attribute__((section(".data"))) void my_function(void)
{
 /* Function */
}
```

This method also works for any GNU GCC compiler implementation.

To link a function into RAM using IAR Embedded Workbench for ARM, the user needs to add the keyword “__ramfunc” before the function, as follows:

```
__ramfunc__ void my_function(void)
{
 /* Function */
}
```

Using these techniques, the linker will place these functions into RAM, in the .data section according to the linker memory map.

ABOUT ACTIVE-SEMI

Active-Semi, Inc. headquartered in Dallas, TX is a leading innovative semiconductor company with proven power management, analog and mixed-signal products for end-applications that require power conversion (AC/DC, DC/DC, DC/AC, PFC, etc.), motor drivers and control and LED drivers and control along with ARM microcontroller for system development.

Active-Semi's latest family of Power Application Controller (PAC)™ ICs offer high-level of integration with 32-bit ARM Cortex M0, along with configurable power management peripherals, configurable analog front-end with high-precision, high-speed data converters, single-ended and differential PGAs, integrated low-voltage and high-voltage gate drives. PAC IC offers unprecedented flexibility and ease in the systems design of various end-applications such as Wireless Power Transmitters, Motor drives, UPS, Solar Inverters and LED lighting, etc. that require a microcontroller, power conversion, analog sensing, high-voltage gate drives, open-drain outputs, analog & digital general purpose IO, as well as support for wired and wireless communication. More information and samples can be obtained from http://www.active-semi.com or by emailing marketing@active-semi.com

Active-Semi shipped its 1 Billionth IC in 2012, and has over 120 in patents awarded and pending approval.