Freescale Semiconductor
Application Note

Document Number: AN4837
Rev. 0, 11/2013

Porting Legacy DSC Applications to Freescale's 32-bit DSC Family

Run DSP Test Sample Projects on an MC56F84789

by *John L. Winters*
Micro controller MCU Applications

The DSP test programs (FIR, IIR, auto-correlation), included with Freescale's CodeWarrior 8.3 for DSC, run as delivered in Processor Expert sample projects supplied for the DSP56858. This application note will step the user through conversion of these projects to an Eclipse-based CodeWarrior 10.5 application for the MC56F84789 featured on the TWR-56F8400. This enables the rich collection of software available for the DSP5685x to be used on the latest DSCs from Freescale, as well as the porting of customer projects to such devices. The algorithms presented in this document are not dependent on peripheral differences between various DSC devices. This technique is generally applicable to all such use cases. This note's intended audience consists of those with interest in the DSC, regardless of Integrated Development Environment (IDE) experience.

Contents

1. Overview of project conversion processes 1
2. Generation of the DSP56858 function test sample code with CW8.3 3
3. Porting from DSP56858 to MC56F8367 with CW8.3 4
4. Importing to CW10.5 and porting to MC56F84789 . 11
5. Testing and Validation 24
6. Conclusion 24

1 Overview of project conversion processes

1.1 Materials needed

For this procedure, Freescale's CodeWarrior 8.3 for DSC (CW 8.3) is needed. The cost of this special edition integrated development environment is free to those who register at www.freescale.com.

Steps outlined below with this IDE do not call for actual hardware. The DP56858 is supported by the simulator included with CW 8.3. No debug pods are needed, and neither are boards.

Other requirements include:

- A PC upon which CW 8.3 may be installed. Operating systems supported include Windows XP and Windows 7.
- CodeWarrior for MCU 10.5 (CW 10.5) (also available from www.freescale.com)
- In addition, the product TWR-56F8400 is needed to run the completed project. The TWR-56F8400 module is a small board that may be run standalone without the Tower System, or in the Tower System. It will host the finished, ported application.

1.2 Porting method selection

There are two basic ways of porting a project from CW 8.3 to CW 10.5.

A new project can be created for the target device, MC56F84789 (supplied with the TWR-56F8400), and the files dragged into the project from the old project. This does not take care of project settings however, nor does it satisfy the new file naming conventions for the file housing the main program. For this and other reasons, this approach was not selected for this application note. It remains a viable approach, but will not be discussed herein.

The second way of porting, the one selected for this note, involves a new feature of CW10.5 that enables it to directly import projects for CW8.3. This greatly simplifies the process and removes many potential sources of error. It remains to sort out the complications that arise from porting not only from an old to a new CW version, but also porting from one *device* to another. The porting will be from the DSP56858, a RAM-based 120 MHz processor, to a flash-based 100 MHz processor, the MC56F84789. Dynamic memory allocation differences between the two devices will also be dealt with. The DSP56858 has an external memory bus, whereas the MC56F84789 does not.

1.3 Overview of steps

The porting process begins with the generation of one of the included DSP function test programs using CW8.3 for the DSP56858. It is run using the simulator to establish the baseline project. See Section 2, "Generation of the DSP56858 function test sample code with CW8.3 for details.

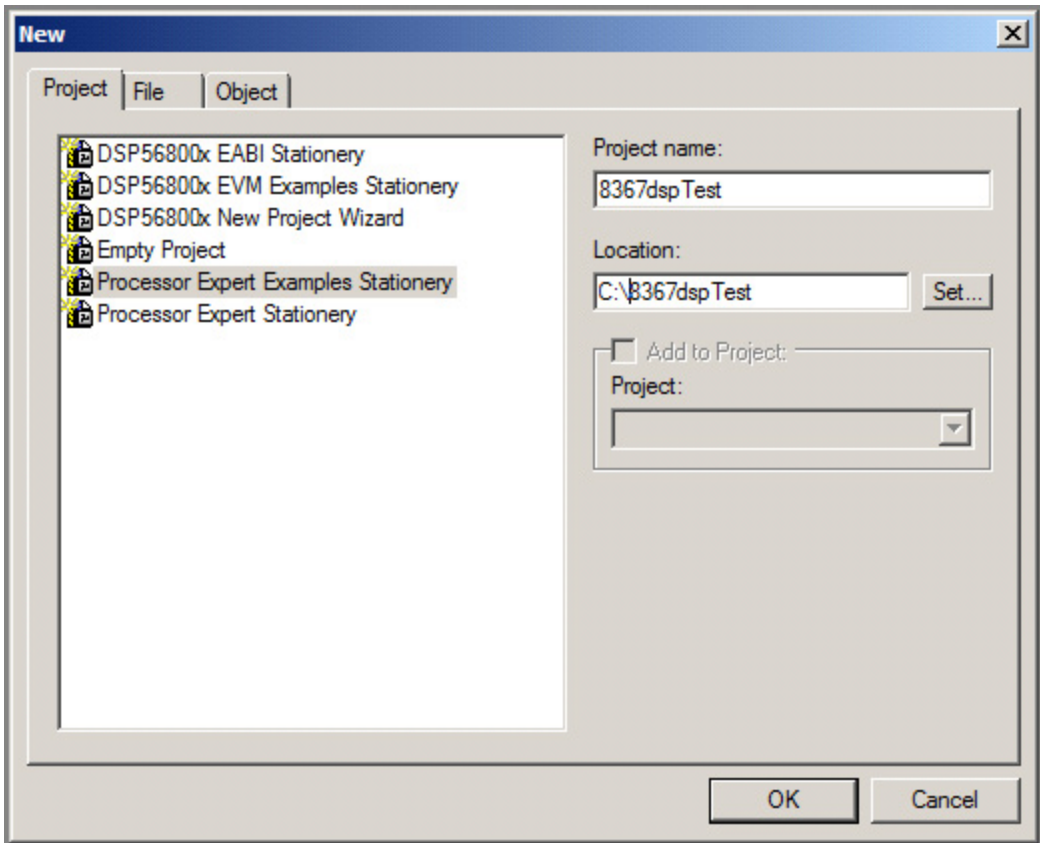
Next, the project is ported to MC56F8367 using CW8.3. This is necessary because the CW10.5 does not support the DSP56858, and because the test program we are interested in is not supplied ready to run on MC56F8367. This is also run on the simulator for verification. See Section 3, "Porting from DSP56858 to MC56F8367 with CW8.3 for details.

The remaining steps each take place using CW10.5. They involve importing the MC56F8367 project and testing it in the simulator, then porting it from the MC56F8367 to the MC56F84789 and running it on hardware (the simulator is not yet supported for the MC56F846789 in CW10.5). See Section 4, “Importing to CW10.5 and porting to MC56F84789 for details.

2 Generation of the DSP56858 function test sample code with CW8.3

1. Obtain Code Warrior for DSC 8.3 at www.freescale.com.
2. Install and run CodeWarrior for DSC, version 8.3.
3. Select a project for the DSP56858 from the processor expert examples stationery TestApplications, such as Signal, DSP_FUNC_DFR for the 56858. This is done from the File tab of the IDE.
4. From the File menu, select New (shorthand File/New...).
5. Select the Project tab of the popped up window, then highlight Processor Expert Examples Stationery.
6. Give the instantiated project a Project name, such as 8367dspTest and click OK.

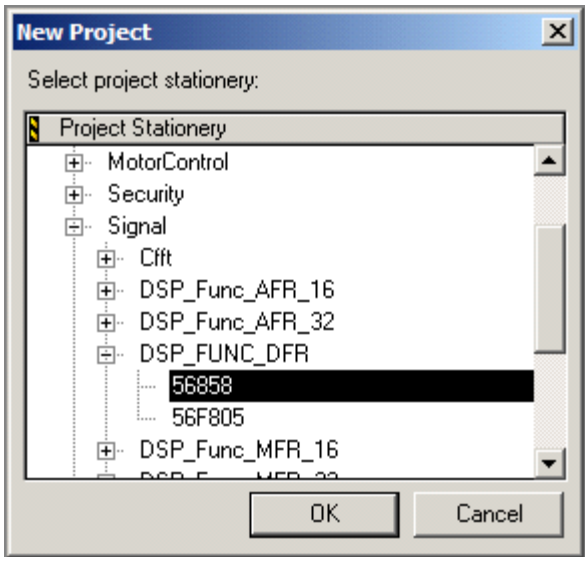
Figure 1. CodeWarrior for DSC



7. Highlight TestApplications in the resultant pop-up.

8. Use the + to expand TestApplications, Signal, and DSP_FUNC_DFR. This function provides testing of visible assembly language implementations of standard DSP algorithms such as FIR, IIR, and auto-correlation. This introduces the assembly language programmer to best practice programming methods for the DSC, as the code is not hidden in libraries.
9. Select 56858 since the 56858 uses the 56800E processor, whereas the 56F805 uses the V1. The V3 is used for the MC56F84xxx and MC56F82xxx devices, but is able to run 56800E code as well.
10. Click OK and wait while the processor expert creates the test harness, including the system under test.

Figure 2. CodeWarrior New Project



3 Porting from DSP56858 to MC56F8367 with CW8.3

If you wish to skip the steps of Section 3, a completed project for the MC56F8367 in .zip file format associated with this application note is available and may be used as the input to section four. The file name is *8367dspTest.zip*.

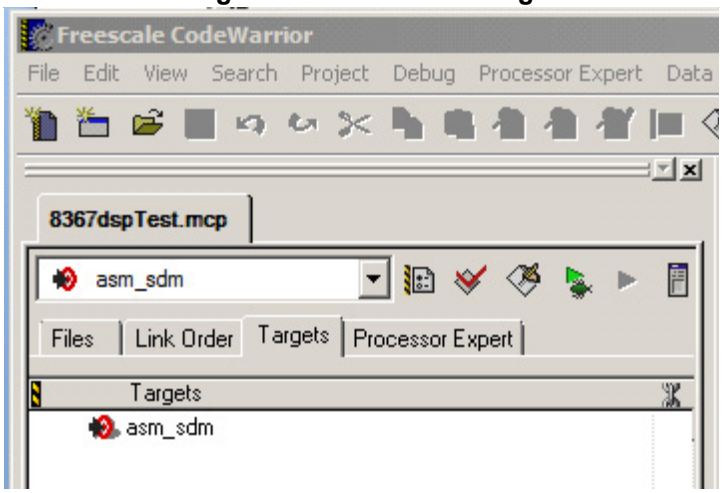
3.1 Simplify the project to be converted

What we do to simplify now means less work porting later. By removing unnecessary options, the work is streamlined for this pedagogic case, as follows.

1. In the project window created on the left of the screen, select the Targets tab. We are only interested in the small memory model with assembly language (most efficient) generated code.
2. Select the asm_sdm target by clicking the target symbol next to asm_sdm (right or left click). The arrow should now point to it.
3. Delete all targets except for the asm_sdm configuration by right clicking the text next to the target symbol and selecting Delete.

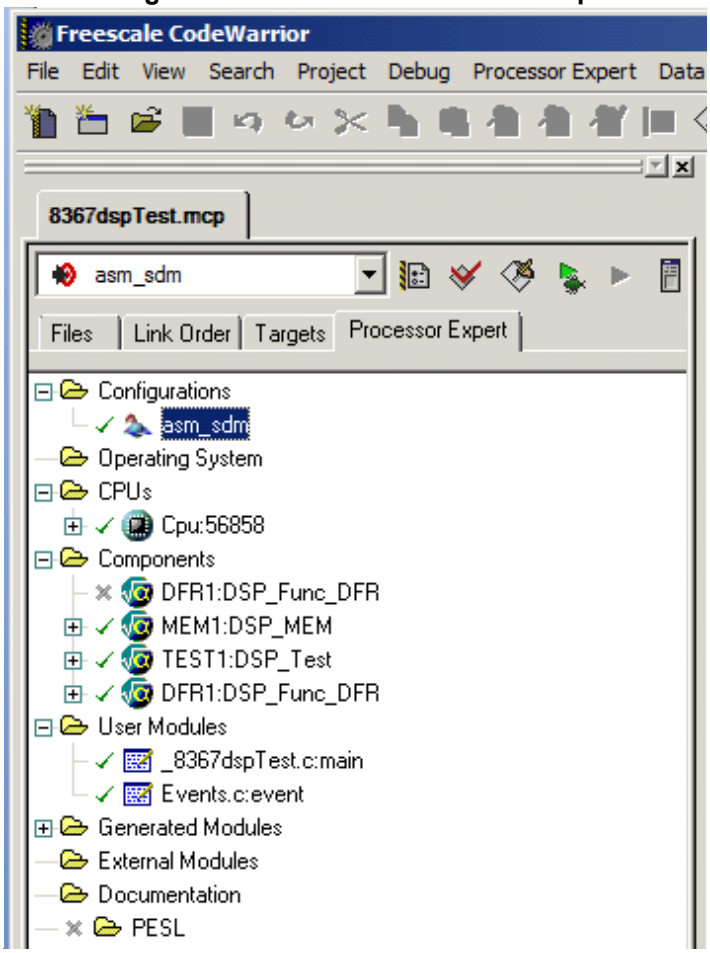
4. Remove all targets except for the asm_sdm target, again to simplify.

Figure 3. CodeWarrior Targets



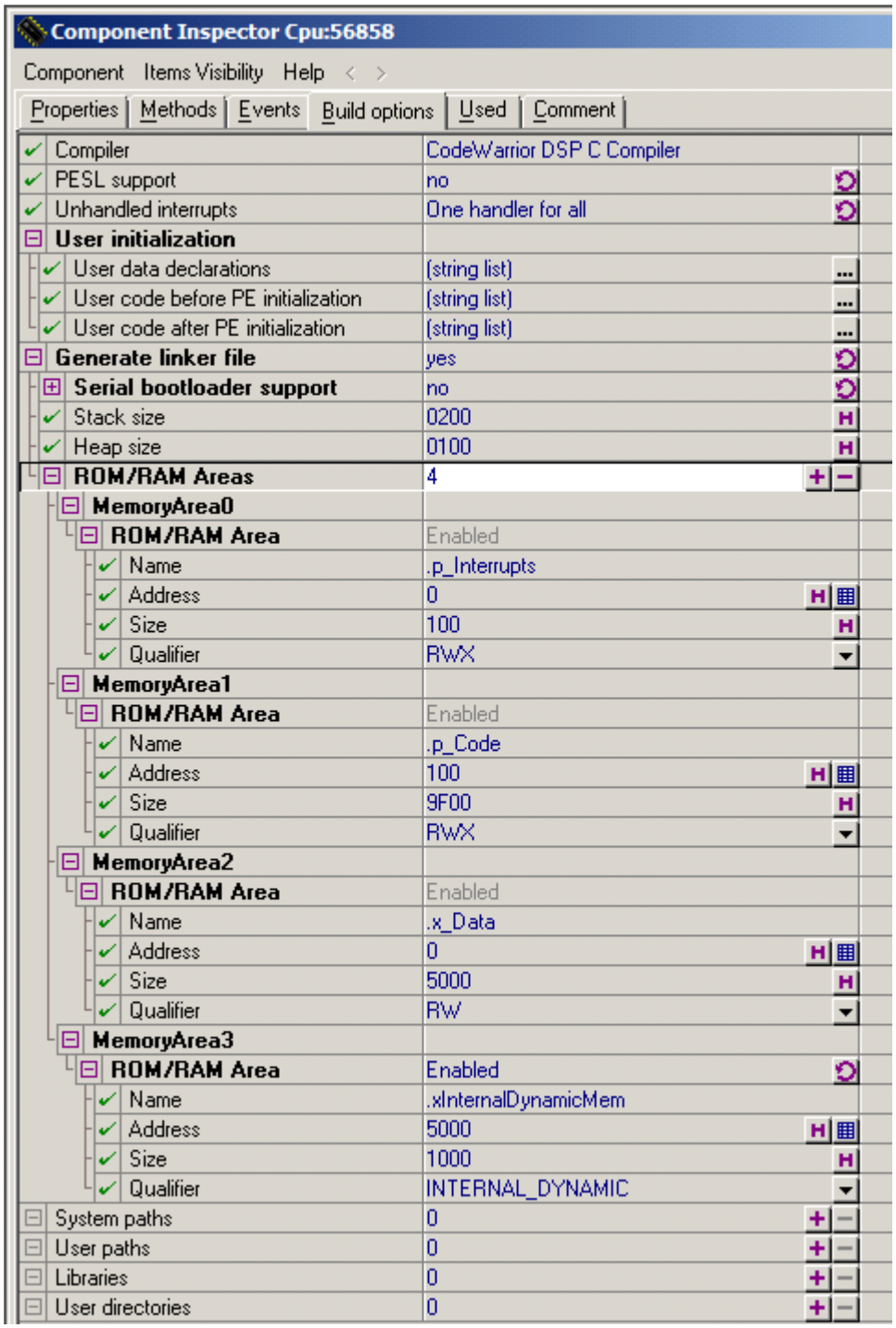
5. Remove all unused configurations by selecting the Processor Expert tab of the project window and right clicking. Select Delete Configuration for all configurations that don't have a green checkmark.

Figure 4. CodeWarrior Processor Expert



6. Using Processor Expert, modify the Cpu:56858 build options tab by reducing the number of ROM/RAM areas to four. To do this, double click the CPU:56858 component.
7. Select the Build Options tab within Component Inspector CPU:56858. Click the minus sign on the ROM/RAM Areas line, removing the external memory area which is unnecessary (the devices being ported to do not have external memory).

Figure 5. CodeWarrior Component Inspector CPU:56858 Build Options

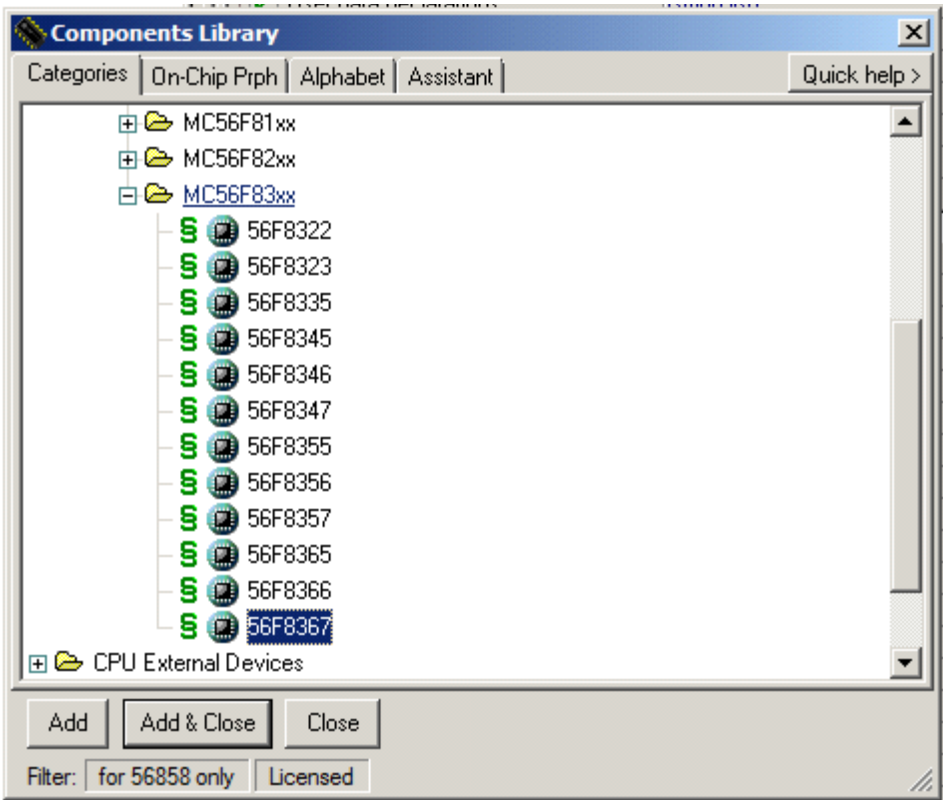


At this point the project *may* be tested by using the F5 key which will build and run the project in the simulator. The console log will have the results of the test. However, it is *not* required to test this project in order to complete the following steps, which include testing further along in the process.

3.2 Retarget for the MC56F8367

1. Using the Processor Expert tab, add the 8367 CPU and make it active by right-clicking in the white space below the components in the project window and selecting Add Component(s).
2. In the resultant pop-up, select Categories. Continue expanding items under CPU until you reach more than one choice. Then expand MC56F83XX and select MC56F8367.
3. Hit Add & Close.

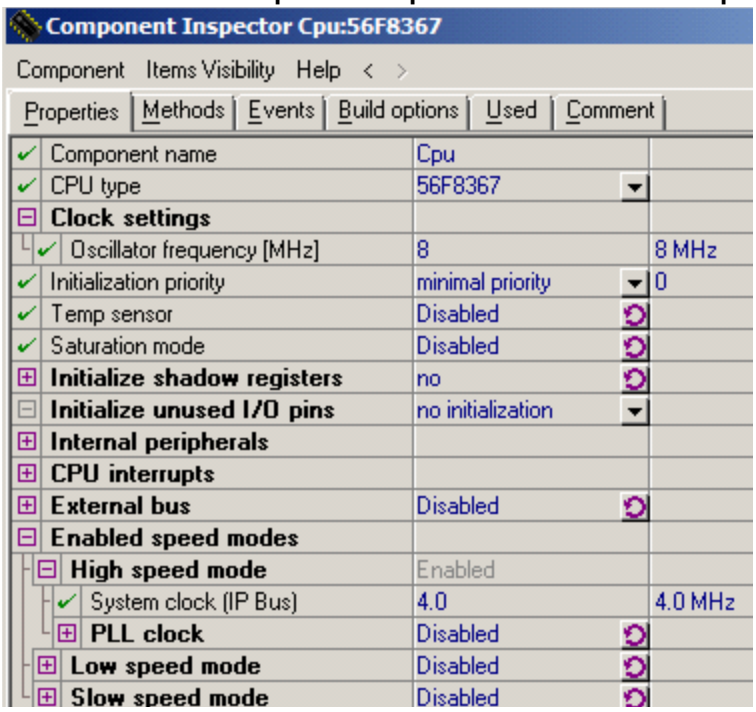
Figure 6. CodeWarrior Components Library Categories



4. To make the new CPU the active CPU, right click it and Select CPU as Target.
5. Configure the Properties of the Active CPU for PLL clock disabled by double clicking the MC56F8367 component and selecting the Properties tab.

6. Turn off PLL clock so it reads "disabled" then change the system clock from 60 to 4 MHz. This is necessary because we inted to run this case on the simulator, rather than on hardware, and we don't want the code to dwell forever waiting for the PLL to lock (See Section 1.1, "Materials needed). When the PLL is disabled, the DSP56858 cannot run at 60 Mhz.

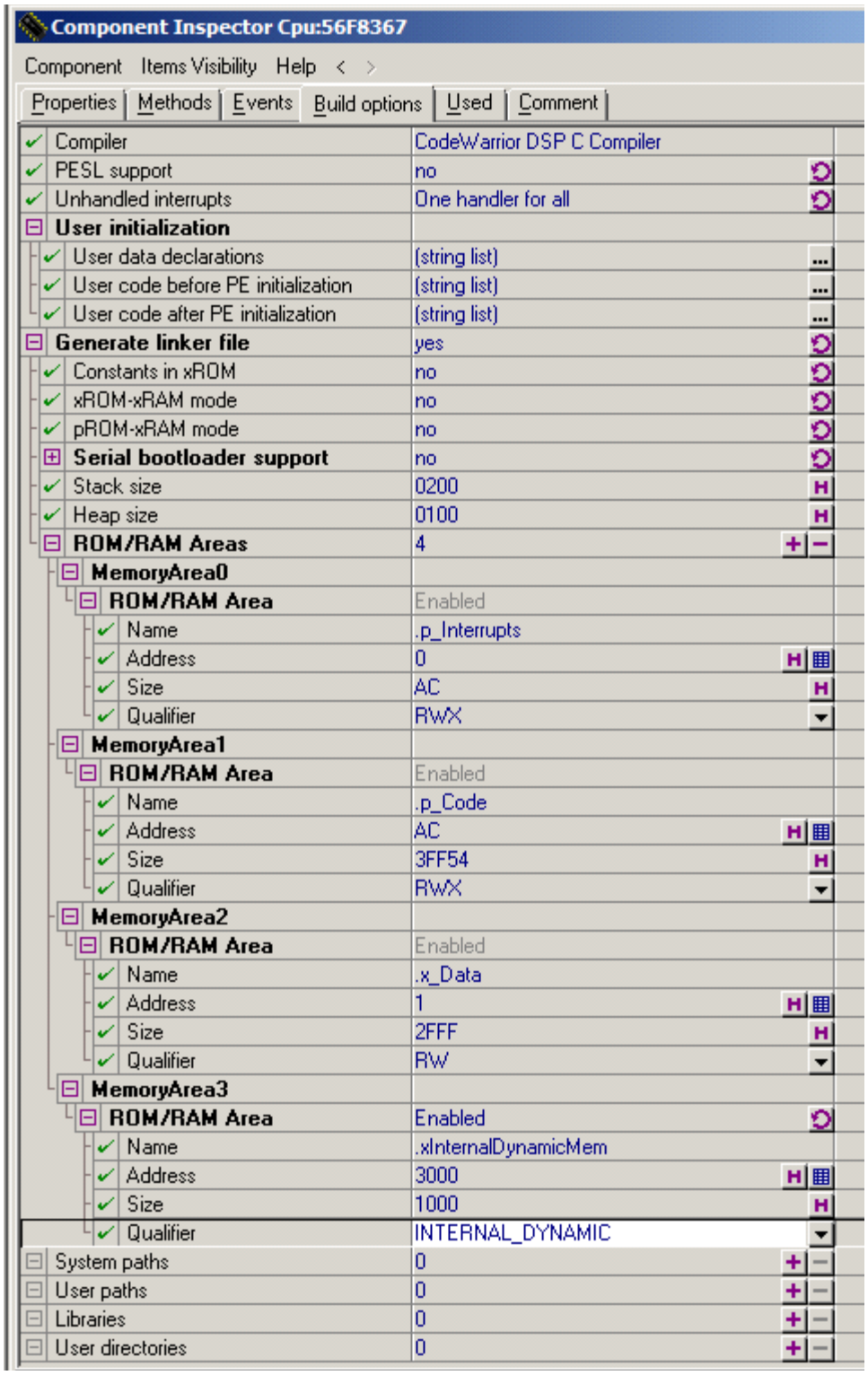
Figure 7. CodeWarrior Component Inspector CPU:56F8367 Properties



7. Configure the MC56F8367 CPU with build settings similar to the MC56858 CPU, otherwise memory segments will be missing and the execution will not be correct when internal memory is allocated using the MEM component. This means adding one memory area (the fourth) to the MCF8367 CPU. (MemorayArea3 detailed below.) To do this, double click the MC56F8367 CPU component and select the Build options tab.
8. Make room for a new memory segment by changing the size of the MemoryArea2 to 2FFF. The new segment will be used for internal dynamic memory allocations. This method of memory handling was used in the original SDK supplied with DSP56858, so these routines will be ported over as well. That is why the memory segment used for the memory pool must be considered, and allocated here.
9. Add a new memory area with the + sign on RAM/RAM areas which will be called MemoryArea3 and configure it as (shown are hexidecimal radix numbers only):

ROM/RAM Area Enabled; *Name*.xInternalDynamicMem; *Address* 3000; *Size* 1000; *Qualifier* INTERNAL_DYNAMIC.

Figure 8. CodeWarrior Component Inspector CPU:56F8367 Build Options



10. In the CPU component for the 8367 device, use the Target Settings Panels to change the M56800E Target Settings from the 858 settings to the 8367 settings. When you mouse over the button in the project window for this it will read "asm_sdm Settings...", the name of the target, and settings.
11. Click that button and scroll down to the M56800E Target Settings, selecting it. Choose the last processor on the list, MC56F8367.
12. Chose the correct configuration file for the processor: C:\Program Files\Freescale\CodeWarrior for DSC56800E v8.3\M56800E Support\initialization\56836x_flash.cfg. The above path will vary depending on where you installed CW8.3. This means the Processor is set to MC56F8367 and the initialization file is set to C:\Program Files\Freescale\CodeWarrior for DSC56800E v8.3\M56800E Support\initialization\56836x_flash.cfg, or a similar path.

Because MC56F8367 is supported in CW10.5 (but not DSP56858), and since the example code we wish to use is available only for the DSP56858, this step is required. DSP56858 projects should not be imported directly with CW10.5 since the resulting project may *appear* normal, but will have serious issues beyond the scope of this note.

3.3 Configure the debug settings for MC56F8367

1. Configure the Remote Debugging settings of the target to use the simulator.
2. Be sure that the new CPU is configured in its properties for saturation mode, or the tests will fail. The saturation feature of the core is used in this DSP example code to maximize performance, so the algorithm depends on it to get the correct results in the shortest time possible. The library takes advantage of the limiting invoked by the saturation mode to reduce cycle count.
3. Remove the 56858 CPU from the project now that the new CPU, MC56F8367, has been added. This will help us avoid erroneous attempts to import a DSP56858 project in to CW10.5 later on.

3.4 Build and Run the MC56F8367 project

1. Build and run the project, observing the output in the console window. It will be necessary to hit run twice, because by default it will stop at the main program after it begins simulating. It should pass the tests.

No hardware is required because the simulator is used. This will also be true once the project is imported into CW10.5, since CW10.5 has a simulator for the MC56F8367 device.

2. Close the console window, the debugger, and the window proclaiming the error count.

3.5 Clean up the MC56F8367 project

1. Remove the object code for all targets using the project menu and close the project as well as the IDE. When closing the IDE, an error may be triggered because the CW8.3 IDE was not designed for Windows 7. This will not affect results.

At this point we have a working project that tests DSP algorithms on the MC56F8367 ready to import into the Eclipse-based CodeWarrior for MCU, version 10.5. Remember where this project is located as it serves as the input to Section 4.

4 Importing to CW10.5 and porting to MC56F84789

Because help is available for CW10.5 through the Freescale Technical Information Center, the level of detail in terms of illustrations is limited in this section. While CodeWarrior Classic is supported, more ready support exists for CW10.5.

Again, if you skipped the steps of Section 3, a completed project for the MC56F8367 in .zip format is available and may be used as the input to this section. The file name is *8367dspTest.zip*.

A completed project for the MC56F84789 is also available for those who wish to skip Section 4 as well, in order to simply explore the code on the MC56F84789.

4.1 Import and run the MC56F8367 simulator project

The following steps will assist you in importing the classic CodeWarrior for DSC 8.3 project into the default workarea. You will eventually be directed to browse to the location of the classic CW project and then you will open the .mcp file from the importation dialog boxes.

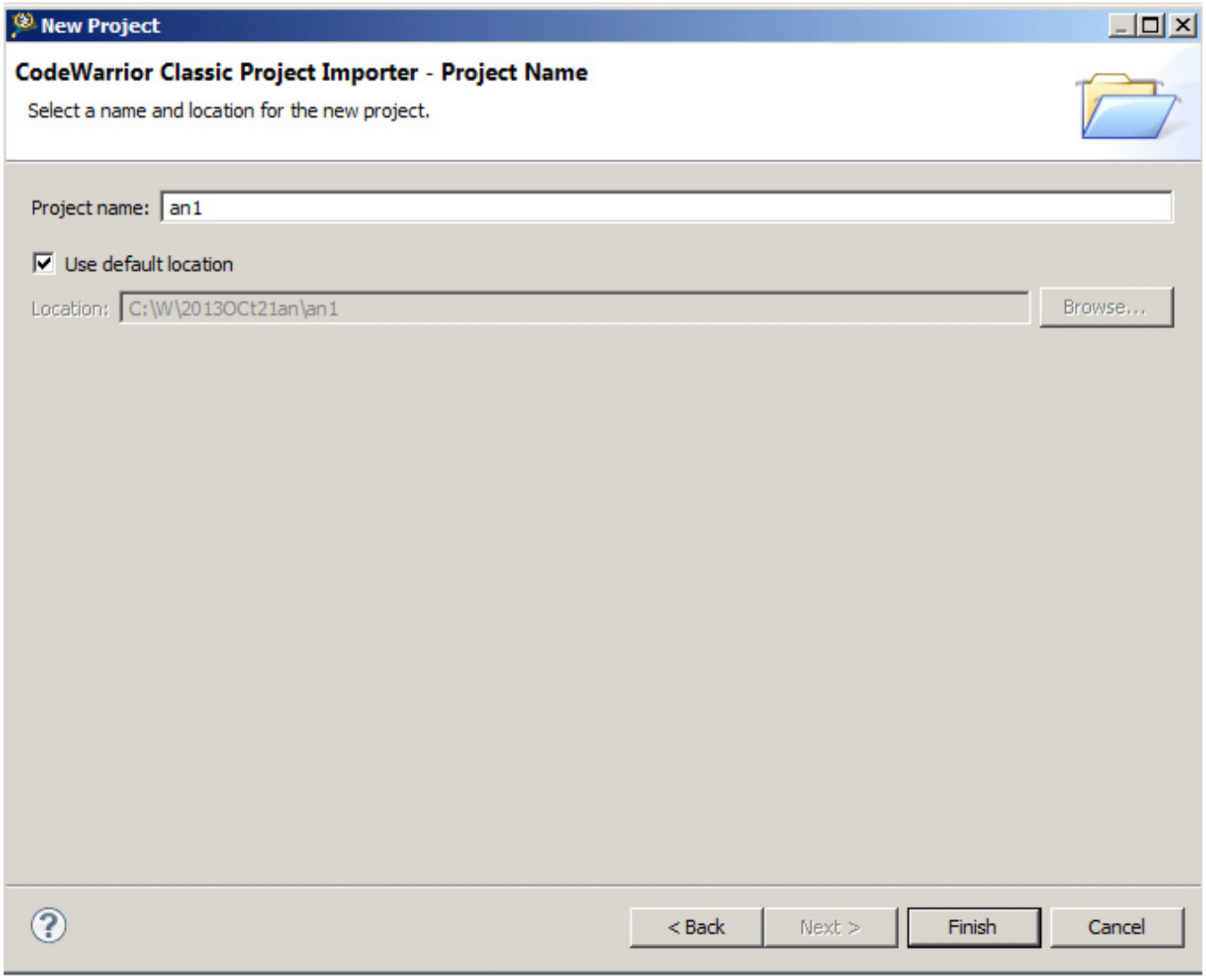
NOTE

It may be necessary to avoid the Browse button in some cases. If browsing results in an error that persists after a reboot of the computer, type the full path to the project's .mcp file in the box. This anomaly can occur when the file at which the browser attempts to start does not exist. If it does, type a valid path into the text input box.

1. Starting with a fresh workarea, run the Eclipse IDE, CodeWarrior for MCU version 10.5.
2. Go the workbench. The selection to import this Classic CW project is, starting with the File menu, File/import/Codewarrior/Codewarrior Classic Project.

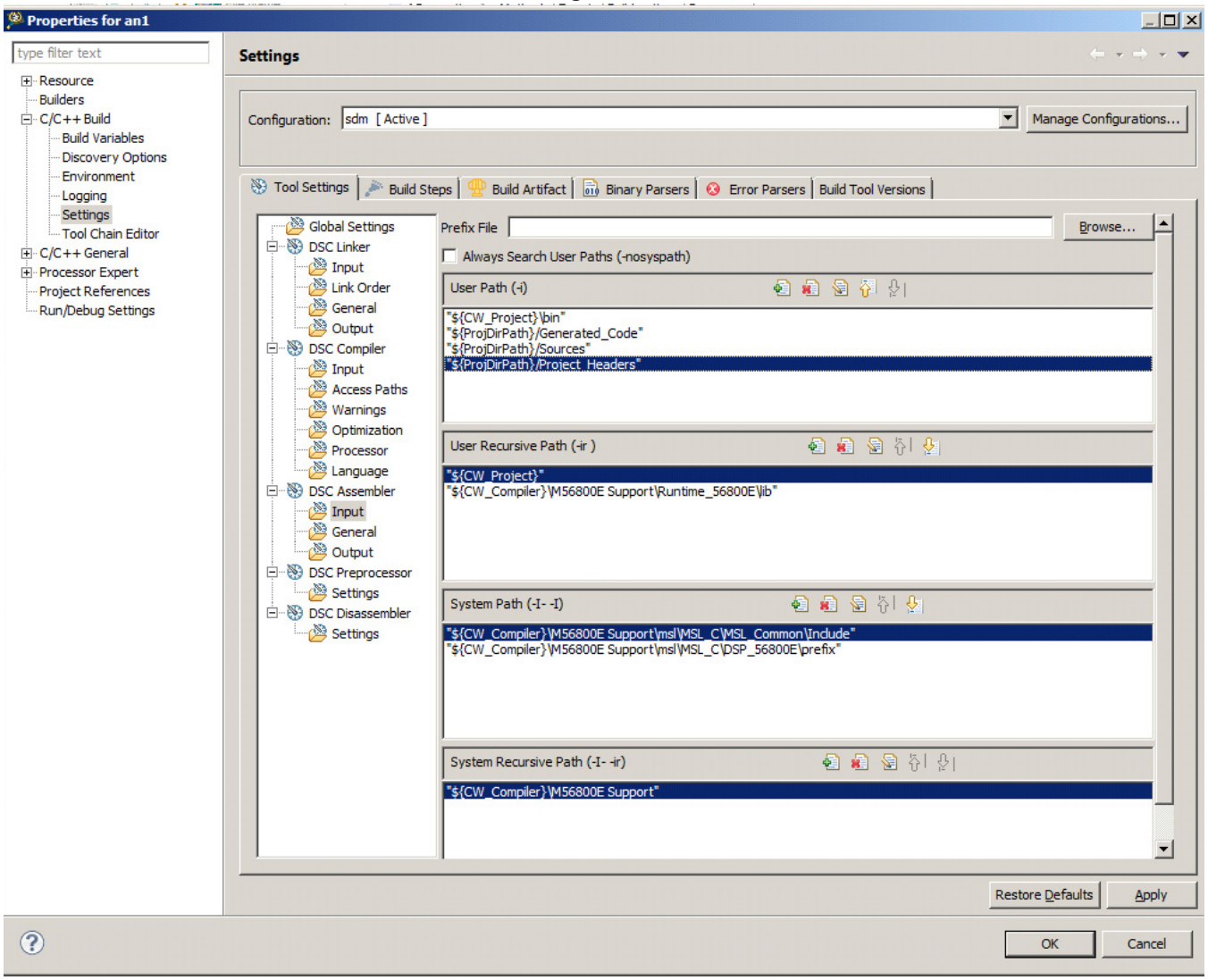
3. Take all the defaults until you come to the screen that shows the default location not being used. Select the default location by checking the box (as shown below) so the project is collected into your new temporary workarea. You can export it to a more permanent location later if desired. Your project name may differ.

Figure 9. CodeWarrior Classic Project Importer



4. Build the project. It will build with two warnings. To fix one warning, remove the Project_Headers path that is not found from the project properties under the asm, input.

Figure 10.



5. To fix the other warning, regenerate the Processor Expert code by expanding the project and looking for Processor Expert. Open it and right click, selecting the regeneration of the code.

Note that processor expert code should be generated by the version of Processor Expert with which Processor Expert is to be used. If code is frozen or extracted from Processor Expert, each case needs consideration based on the circumstances.

These warnings are artifacts of the conversion process. Similar artifacts may appear when other projects are ported with this method, but should not delay the porting process significanlty.

6. Clean and build the project (with no errors, no warnings.)

7. Debug the sdm target (run with the debugger as the program loading agent) using the simulator. It will pass.

The console output should be as follows:

```
testdfr16 - Started
testdfr16 - ***** Test of methods in assembly code *****
testdfr16 - Test FIR...
testdfr16 - Test FIR with modulo addressing in internal memory
testdfr16 - Test FIR with modulo addressing in external memory
testdfr16 - Test FIR with linear addressing in external memory
testdfr16 - Test FIRDEC by factor of 2
testdfr16 - Test FIRDEC with odd number samples
testdfr16 - Test FIRInt by factor of 2
testdfr16 - Test FIRInt by factor of 3 with modulo addressing in internal memory
testdfr16 - Test FIRInt by factor of 3 with modulo addressing in external memory
testdfr16 - Test FIRInt by factor of 3 with linear addressing in external memory
testdfr16 - Test FIRInt by factor of 4
testdfr16 - Testing Cross Correlation...
testdfr16 - Testing code for option CORR_RAW
testdfr16 - Testing code for option CORR_BIAS
testdfr16 - Testing code for option CORR_UNBIAS
testdfr16 - Testing AUTO CORRELATION...
testdfr16 - Testing code for option CORR_RAW
testdfr16 - Testing code for option CORR_BIAS
testdfr16 - Testing code for option CORR_UNBIAS
testdfr16 - Test IIR...
testdfr16 - Test IIR with modulo addressing in internal memory
testdfr16 - Test IIR with modulo addressing in external memory
testdfr16 - Test IIR with linear addressing in external memory
testdfr16 - Passed
testdfr16 - Ended
```

It will output to the console all the tests that passed.

NOTE

When external memory is not available, as in this case, internal memory is automatically allocated by the memory manager.

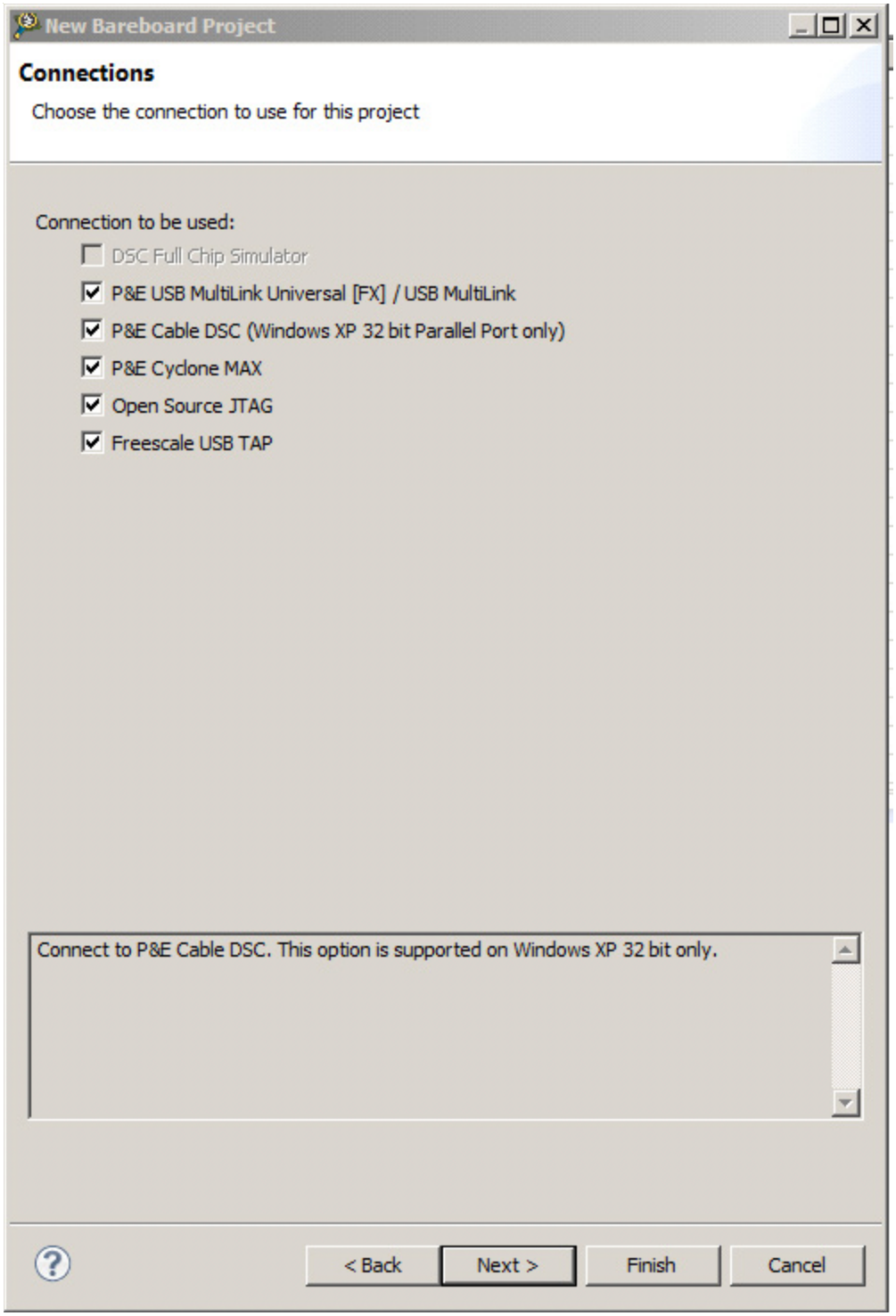
4.2 Generate debug configurations for the future target

This leaves us with a project that runs only in the simulator and has no other debugger options. To remedy this situation, there needs to exist a bare board project (from that template) for the desired target in the same workarea that has all the needed debug configurations available. That is the motivation for the next steps to create such a project, a project that will not be modified after creation, but only exists in order for the imported project to reference the debug configurations.

1. In the same work area create a new default project bareboard project for the MC56F84789 (or other desired processor of similar or newer vintage). This is done with: File/New/Bareboard Project.
2. Give the Project a name such as x, since it will not be further developed, but only exists for reference.
3. Next, select and expand the 56800/E(DSC) Devices list that pops up:
   56800/E(DSC) / MC56F847xx/MC56F84789, Application.

4. Next, select all the possible connections to make sure they don't need to be added later. Depending on your version of CW, the simulator may be deprecated. (It is deprecated on CW10.5.) It is only deprecated for the MC56F84xxx and MC56F82xxx devices. Since this is the MC56F84789, it is deprecated as shown below in this screen shot from CW10.5:

Figure 11. New Bareboard Project Connections



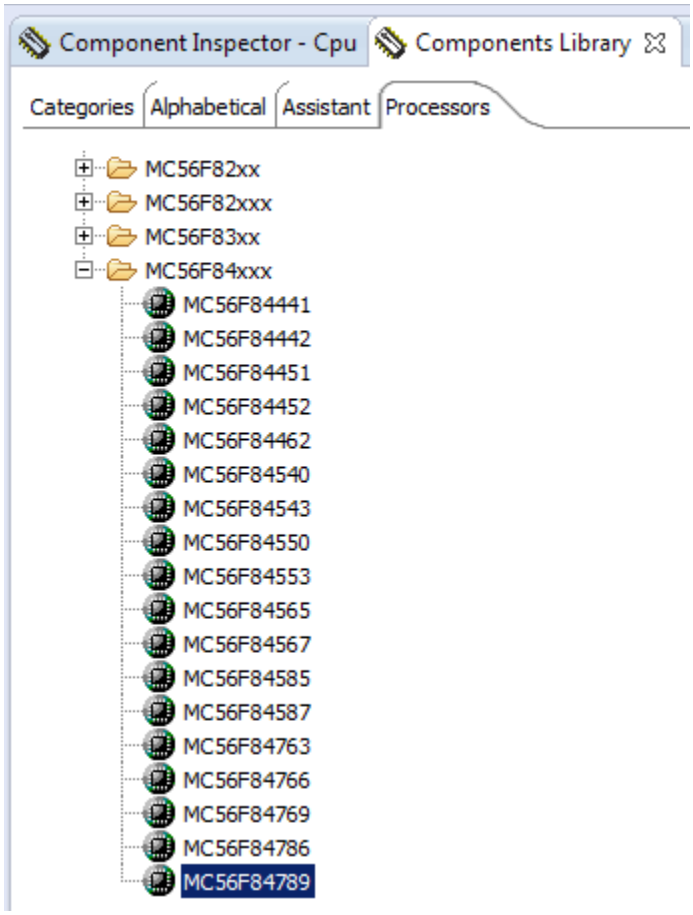
5. Select all the possible debug options you could potentially need.
6. Next, on the Language and Build Tools Options panel that pops up, select Mixed C and ASM to get all the header files.
7. Next, under Rapid Application Development select Processor Expert, since the project we are working with is a PEx project. Finish.

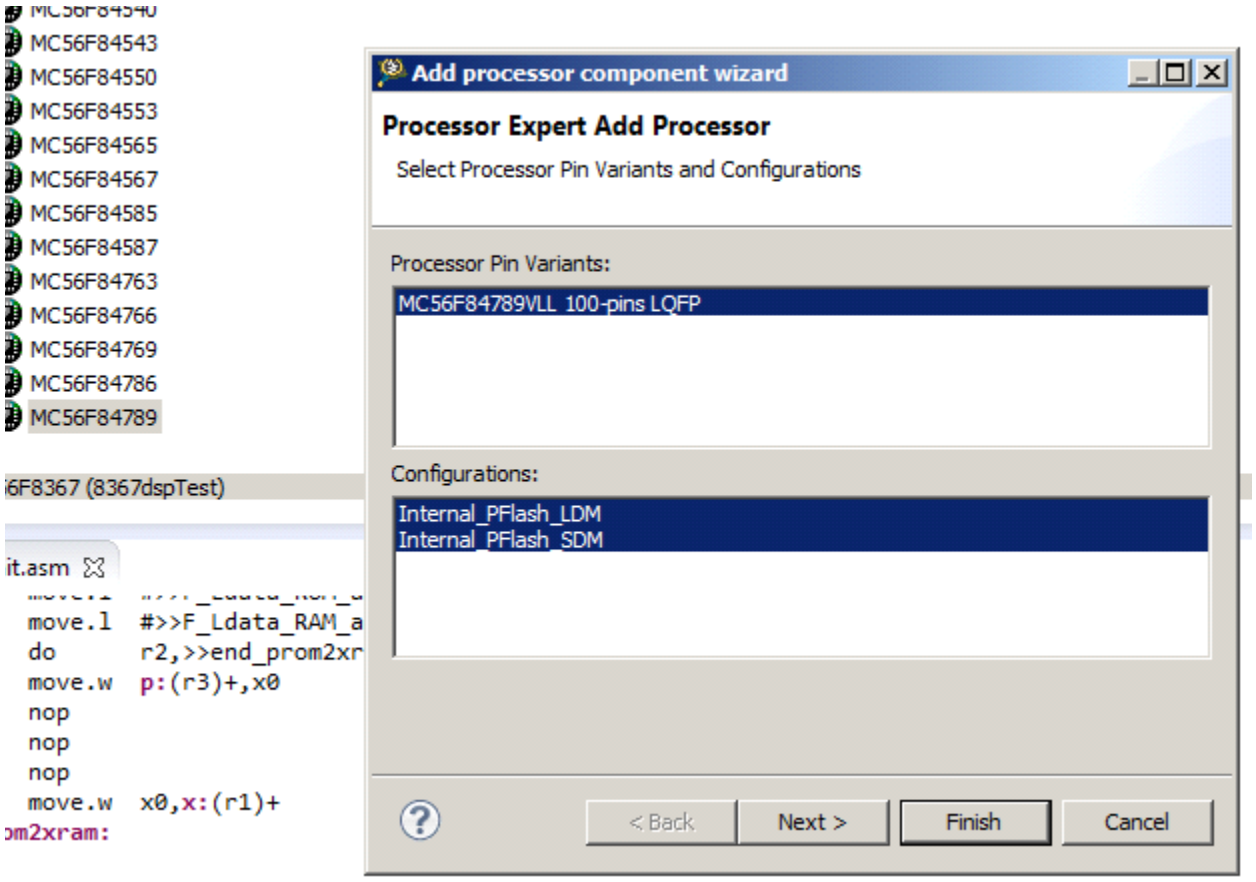
At this point, the bare board reference project containing the correct debug configurations needed for the imported project exists and may be referenced. There is no need to build or run this reference bare board project. It was only created for its debug configurations which are only readily available when they exist on some project in the workarea (unlike CodeWarrior classic where it makes no difference). See the Quick Start Guide for the TWR-56F8400 board for details of creating such a new project if necessary.

4.3 Port from MC56F8367 to MC56F84789

1. In the imported project (8367dspTest in the Eclipse workspace if you followed the naming suggestion), add the MC56F84789 CPU (or the target you wish to use) to the project in a manner similar as in the Classic CW case, with similar considerations. The Processor Expert CPU list is accessed as shown below.

Figure 12. Processor Expert CPU list access

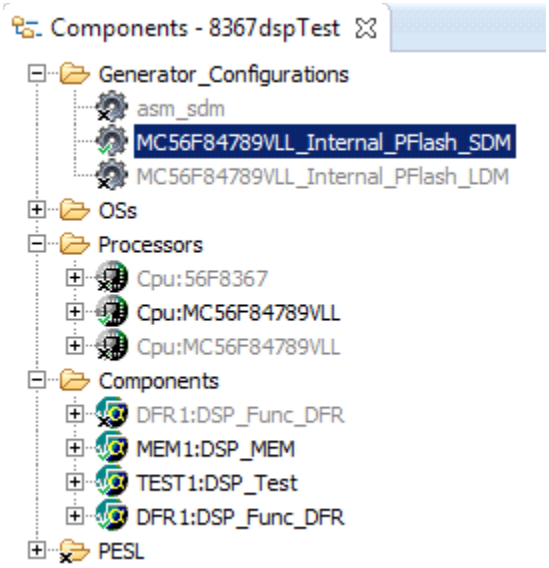




2. Right click on the ProcessorExpert.pe item in the project tree and open it. Then reset the perspective (if necessary) and then under Processor Expert tab select Show Views.
3. Look to the upper right of the project tree and find the Components Library. Select the Processors sub-tab.
4. Expand 56800 (which includes all DSC).
5. Expand MC56F84xxx, and select the appropriate processor, (e.g. MC56F84789) and the defaults that follow in the pop-ups.

Note that there are now three Generator_Configurations. The small model is most economic when it is possible to use it. We want to use MC56F84789VLL_Internal_PFlash_SDM. To select it, double click it.

Figure 13.



Delete unused Generator_Configurations and Processors, having added already the MC56F84789 by right-click/delete.

4.3 Create the needed Debug Configuration

4.3.1 Debug Configurations

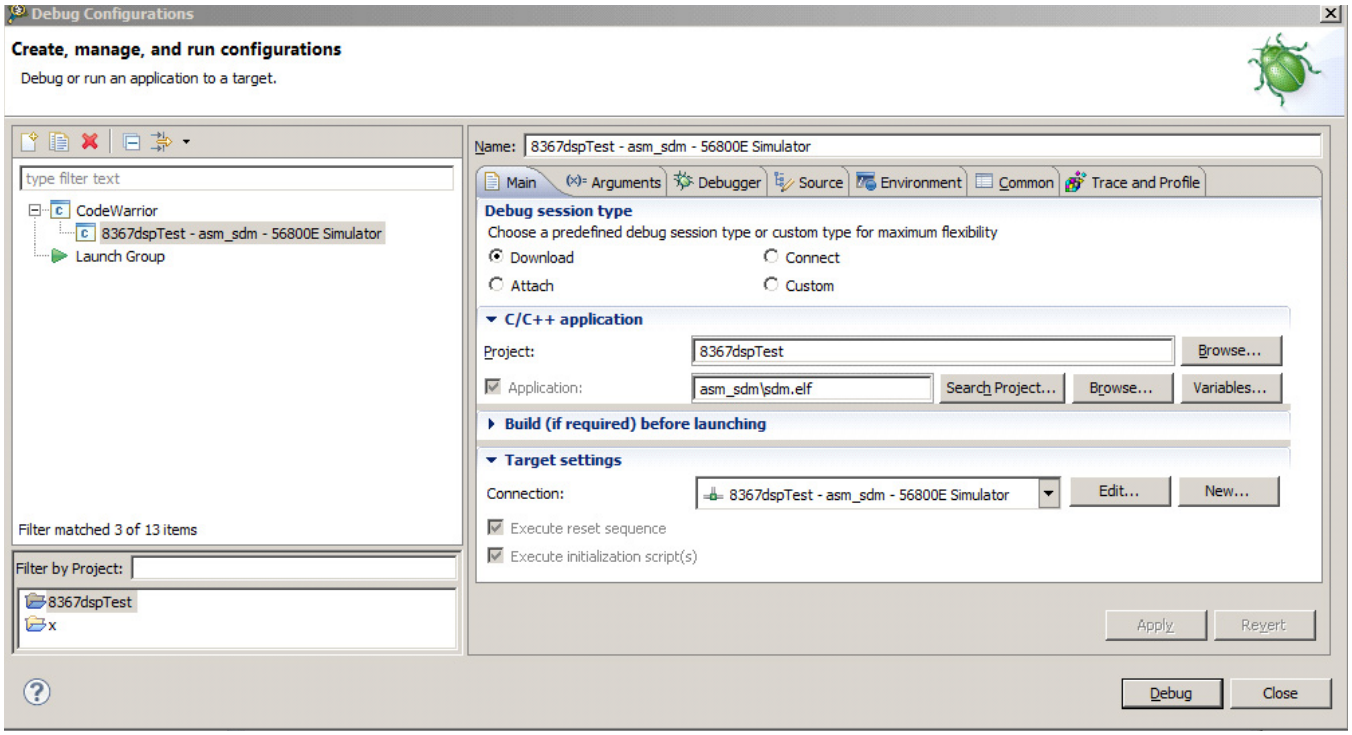
CodeWarrior 10.5 needs debug configurations set up in a project in order for the debugger to connect to the target in the desired manner. The purpose of creating such a configuration is to allow the debugger to communicate to your TWR-56F8400 board containing the MC56F84789 target device.

Rather than make debugger configurations from scratch, it is easier to clone an existing debug configuration in the project and then edit it, taking advantage of the other debug configurations in other projects in the workarea.

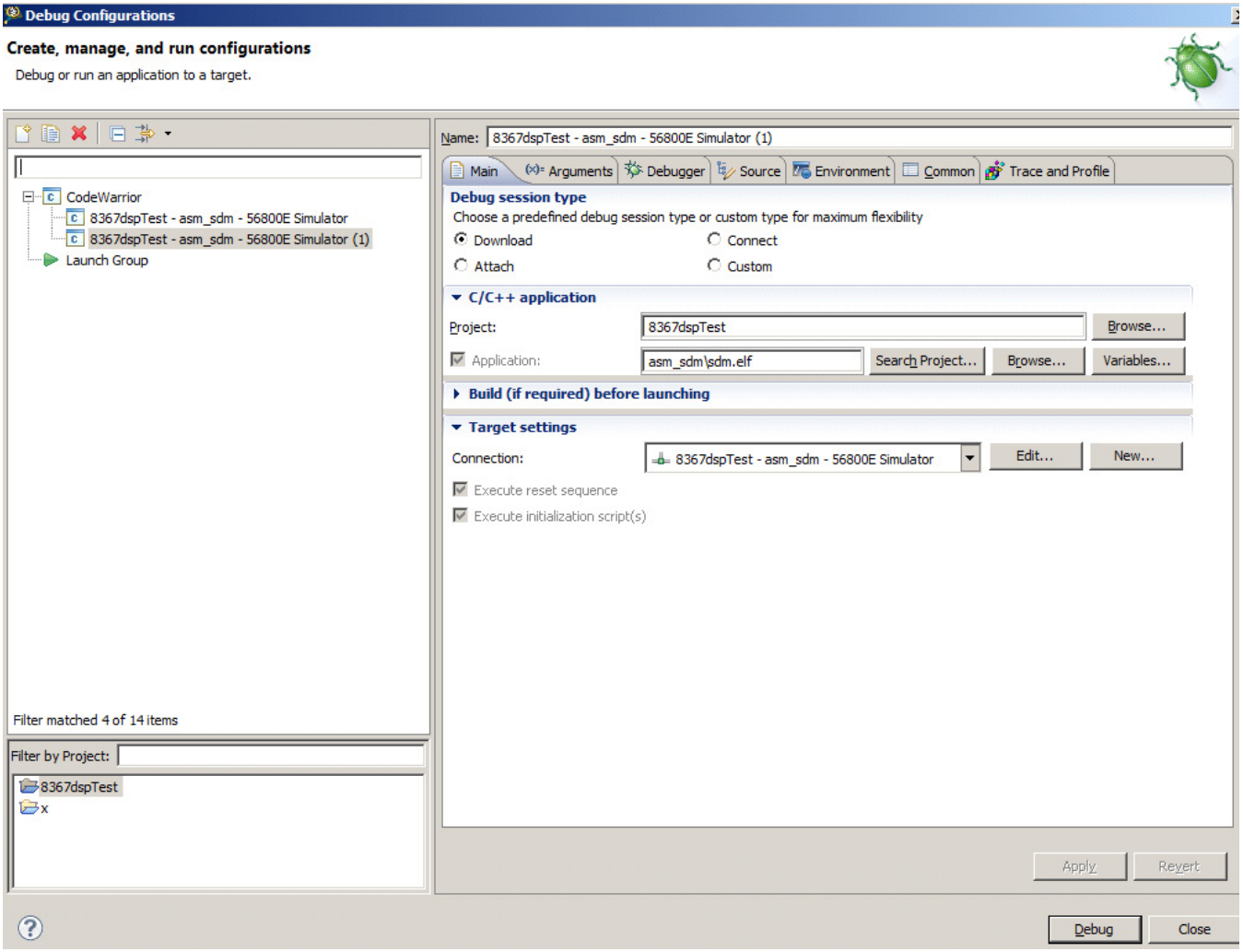
Following that, the debug configuration will be used to test the application on the hardware.

4.3.2 Clone and edit the existing Debug Configuration

1. Edit the Debug Configurations, cloning the one that exists and make the new clone into one that uses the hardware pod that you wish to use to run on hardware. To do this, right-click on the project, 8367dspTest, and select Debug As/Debug Configurations. You will see this:



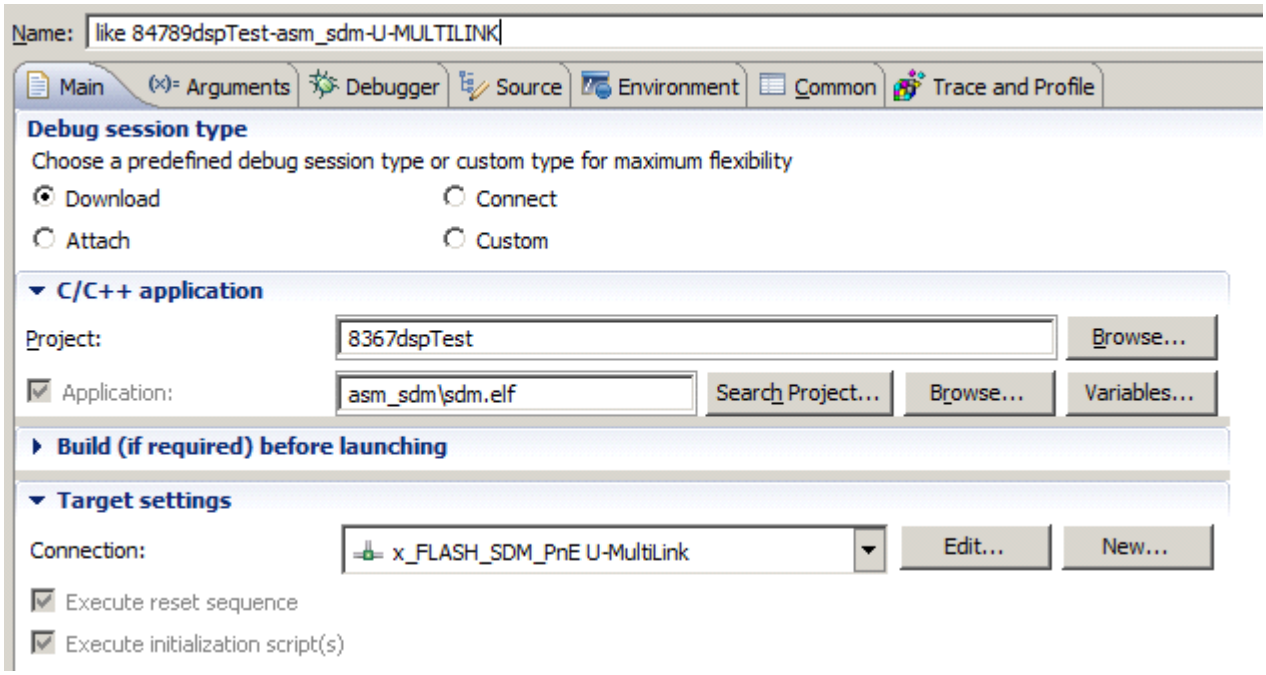
The icon just to the left of the red x is the clone icon. Use it and you will see this:



2. Update the name field to something like 84789dspTest-asm_sdm-U-MULTILINK.
3. Session type is Download, which is correct.
4. Project is correct, as well as the .elf file selected.
5. Target settings show the use of the simulator. This needs to be updated with the pull down arrow on the combination box.

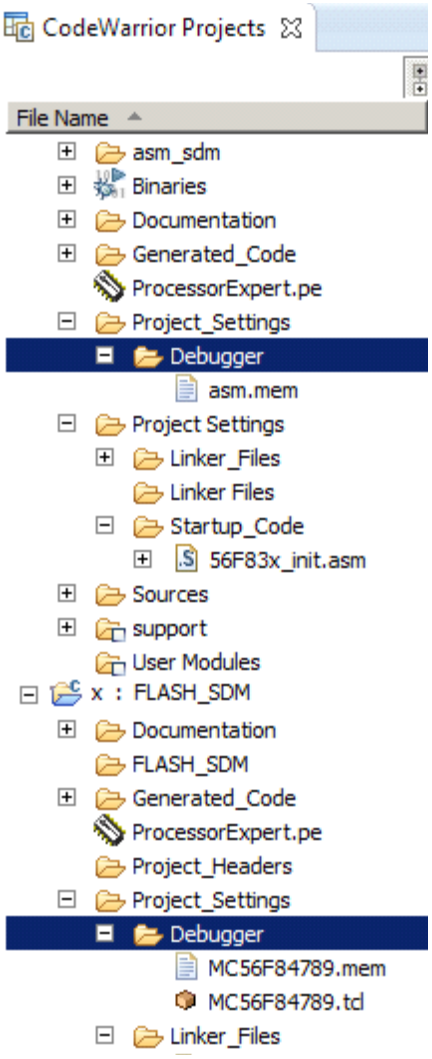
6. Pulling it down, and switching to the correct choice, we then see this (note that a new name has been typed in):

Figure 14.



You can also edit the target settings to check them using the Edit button. However, on this version of Code Warrior, 10.5, they will be correct.

7. Copy the debugger related files, present on the project created for its debug configurations, but not on the main project we are working on. Do this by first deleting asm.mem in the debugger folder. Then copy the individual files MC56F84789.mem and MC56F84789.tcl to the folder from which asm.mem was just deleted.
8. Now, with hardware, there is no need to go back to simulation. Clean and build the application.



9. The new debug configuration can now be saved and used to run the project with the debugger.

4.4.3 Running the project

1. Test it on hardware, the MC56F84789EVME as supplied soldered to the TWR-MC56F8400.
2. Be sure the U-MULTILINK is connected between the target board and the computer.

See the Quick Start Guide for the MC56F84789 for further instructions.

4.4.4 Project download availalbe

Again, for those who wish to skip the steps in Section four, *a completed project for the MC56F84789 in zip file format* associated with this application note is available for those that just want to explore the code on the MC56F84789.

5 Testing and validation

The Processor Expert example project considered in this application note consists of a test harness that runs on the DSP56858. Once it is ported, running the test harness self-verified the port to the new target.

While the project is converted and runs, is it independent of the old project.

A careful inspection of the CW10.5 project created does reveal a link back to the old CW8.3 project. This link is not needed with the method used in this application note. Because we are looking for a complete conversion, it would be desirable to prevent the projects from relying on each other. This link will not cause serious issues, but to dissolve it, take the following steps.

1. On the left of the properties window of the CW10.5 project, expand C/C++ Build and select Build Variables, also on the left. You will see a Name called CW_Project which points back to the old project.
2. In the value column for this Name, CW_Project, use the string ${ProjDirPath} instead of the string pointing back to the old CW8.3 project. Apply and click OK.
3. On the left of the properties window of the CW10.5 project, expand C/C++ Build and select Settings, also on the left. In the Settings portion of the window (right portion) the Tool Settings tab should already be selected. If not, select it and click the Input icon below the DSC assembler icon. Delete the line containing ${CW_Project}\bin by selecting the line, using the delete key and then clicking apply. Click OK.
4. On the left of the properties window of the CW10.5 project, expand C/C++ Build and select Settings, also on the left. In the Settings portion of the window (right portion) the Tool Settings tab should already be selected. If not, select it. Click the Access Paths icon below the DSC Compiler icon. Delete the line containing ${CW_Project}\bin by selecting the line, using the delete key and clicking apply. Click OK.

Now the project can be cleaned and built again with the old CW project also removed from the computer with no warnings, and no undesired connection back to the old project.

6 Conclusion

Using the method revealed in this application note, DSC projects that consist of core/memory routines (not using peripherals) may be ported in a direct manner from older DSC products, starting with the first 56800E DSC, the DSP56858 onwards. This includes the entire line of 56800E core-based DSC products.

The large collection of example code available to users of the DSP56858 is now readily portable to all DSC users. This code includes source code generation and included test harnesses.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN4837
Rev. 0
11/2013