

Configuring the CodeWarrior Tools to Debug a BSC9131RDB Board

by *Freescale Semiconductor, Inc.*
Austin, TX

This application note describes how to configure the Freescale CodeWarrior tools to support developing and debugging of embedded software on the BSC9131RDB board.

Because the QorIQ Qonverge BSC9131 processor contains two disparate processor cores, writing software for the part requires two different sets of CodeWarrior tools: one to manage the DSP core, and another to manage the system core. Care must be taken to ensure that the two sets of tools do not interact with each other. Both the proper configuration sequences and the potential pitfalls are described here.

This application note assumes the use of CodeWarrior for StarCore DSPs v10.2.10 or later, and CodeWarrior for Power Architecture® v10.1.2 or later.

Contents

1	BSC9131 overview	2
2	Debugger configuration strategies	4
3	BSC9131RDB board setup	6
4	Two-TAP connection scheme	7
5	Revision history.....	33
Appendix A	How to Disable the SC3850 Caches	34
Appendix B	DSP Debugging with a BSP Present.....	38
Appendix C	Ethernet TAP Run Controller Options	39

1 BSC9131 overview

The Freescale QorIQ Qonverge BSC9131 contains two processor cores, each of whose microarchitecture is optimized for a specific purpose:

- StarCore SC3850 DSP core implements high-throughput signal processing functions
- Power Architecture e500 core implements high-volume network functions

These cores manage a number of powerful peripherals, all of which are interconnected through a low-latency switching fabric (Figure 1). However, for debugging setup purposes, only the cores are considered in this application note.

NOTE

Both cores have equal access to the processor memory and peripherals.

More information on the BSC9131 can be found in the *BSC9131 QorIQ Qonverge Multicore Baseband Processor Reference Manual*.

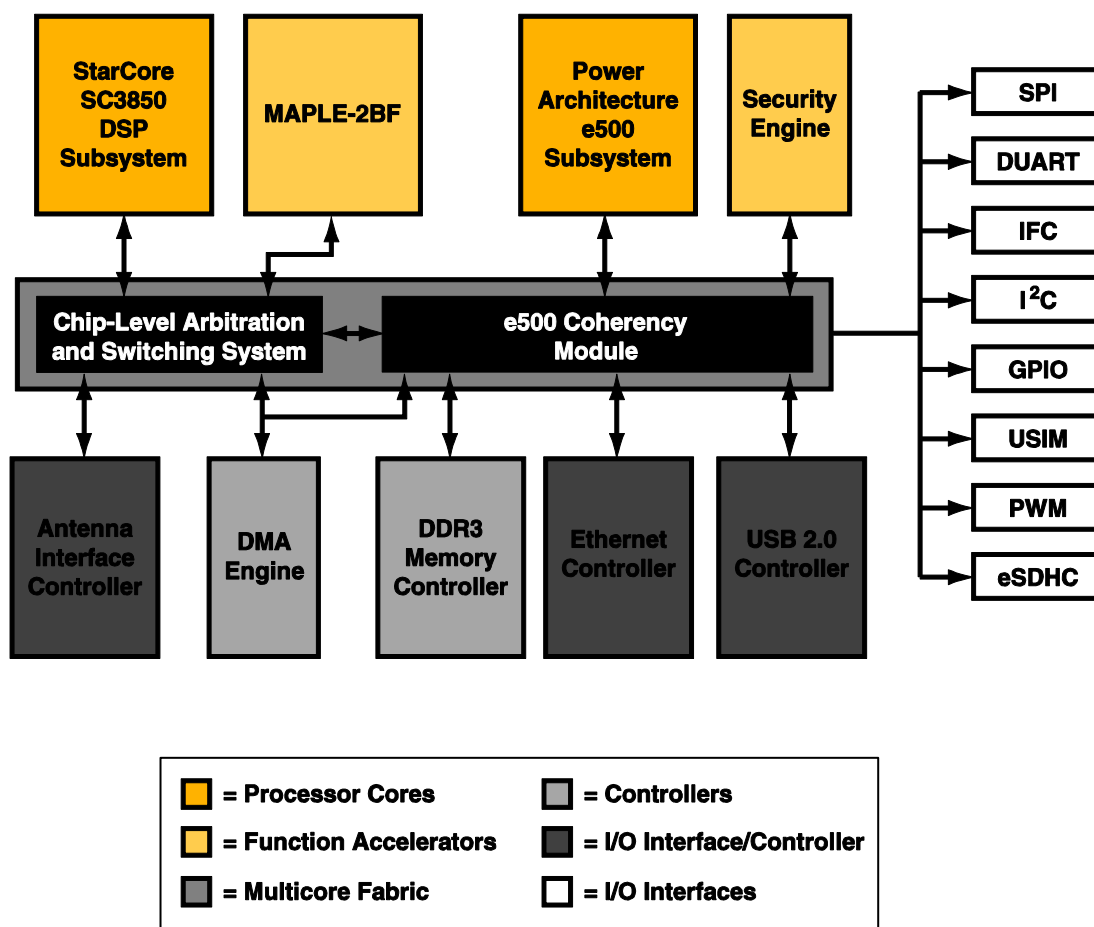


Figure 1. Key components of the BSC9131

1.1 Hardware considerations

Each core has its own unique debug port. The SC3850 uses an OCE port while the e500 core uses a COP port. Under certain circumstances, this can require the use of two different TAP run controllers to debug code on the BSC9131RDB.

NOTE

The use of a second TAP run controller is optional. The BSC9131RDB board supports the debugging of both processor cores through a single TAP run controller connected to the COP debug header connector. The settings on switch block SW6 determine whether one or two TAP run controllers are used when debugging code on the board.

1.2 Software considerations

The two different processor cores use different instruction sets. The SC3850 DSP core has instructions that optimize DSP algorithms, such as a multiply-accumulate instruction; the e500 core has instructions that optimize network processing. Because both cores use different instruction sets, each requires a different CodeWarrior debugger. Specifically, CodeWarrior for StarCore DSPs v10.2.x or later is required to write and debug DSP code for the SC3850 core; and CodeWarrior for Power Architecture v10.1.x or later is used to write and debug networking code for the e500 core.

The CodeWarrior tools communicate with the TAP run controllers through a CodeWarrior Connection Server (CCS) software module. The CCS enables the tools to communicate transparently to different types of run controllers that might also use different connection interfaces. For example, the CCS can interact with a run controller through a USB cable or via an Ethernet network connection. More importantly, the CCS can coordinate the commands issued by multiple CodeWarrior debuggers and direct them to the appropriate run controller. The CCS, therefore, enables CodeWarrior for StarCore DSPs and CodeWarrior for Power Architecture to command and control their respective cores within the BSC9131.

When using two CodeWarrior debuggers to debug code on the BSC9131RDB, it is critical that one debugger “owns” control of the board. The CodeWarrior debugger that owns the board is responsible for initializing and resetting the board when its session starts. The other CodeWarrior debugger establishes communications with the board through the CCS, but must not perform any initialization or reset operations, lest it wipe out a debug session in progress. Stated another way, the first (or primary) CodeWarrior debugger that starts a session on the board must be the owner. It is responsible for launching the CCS module and performing the board initialization and reset. The second (or secondary) CodeWarrior debugger piggybacks off of the active CCS module to communicate with the board. It must not do any initialization or reset actions when it establishes its debug session. To ensure that the secondary debugger session does not perform these operations, its configuration settings must be adjusted.

Either the CodeWarrior for StarCore DSPs or the CodeWarrior for Power Architecture IDE can be the primary debugger.

2 Debugger configuration strategies

The BSC9131RDB supports two run controller schemes as follows:

- One TAP manages the board
- Two separate TAPs manage the board

The two connection schemes are shown conceptually in [Figure 2](#).

NOTE

The figure shows that TAP run controllers use the USB interface to connect to the workstation. However, with the appropriate TAPs an Ethernet network interface can be used to support remote debugging sessions with the board.

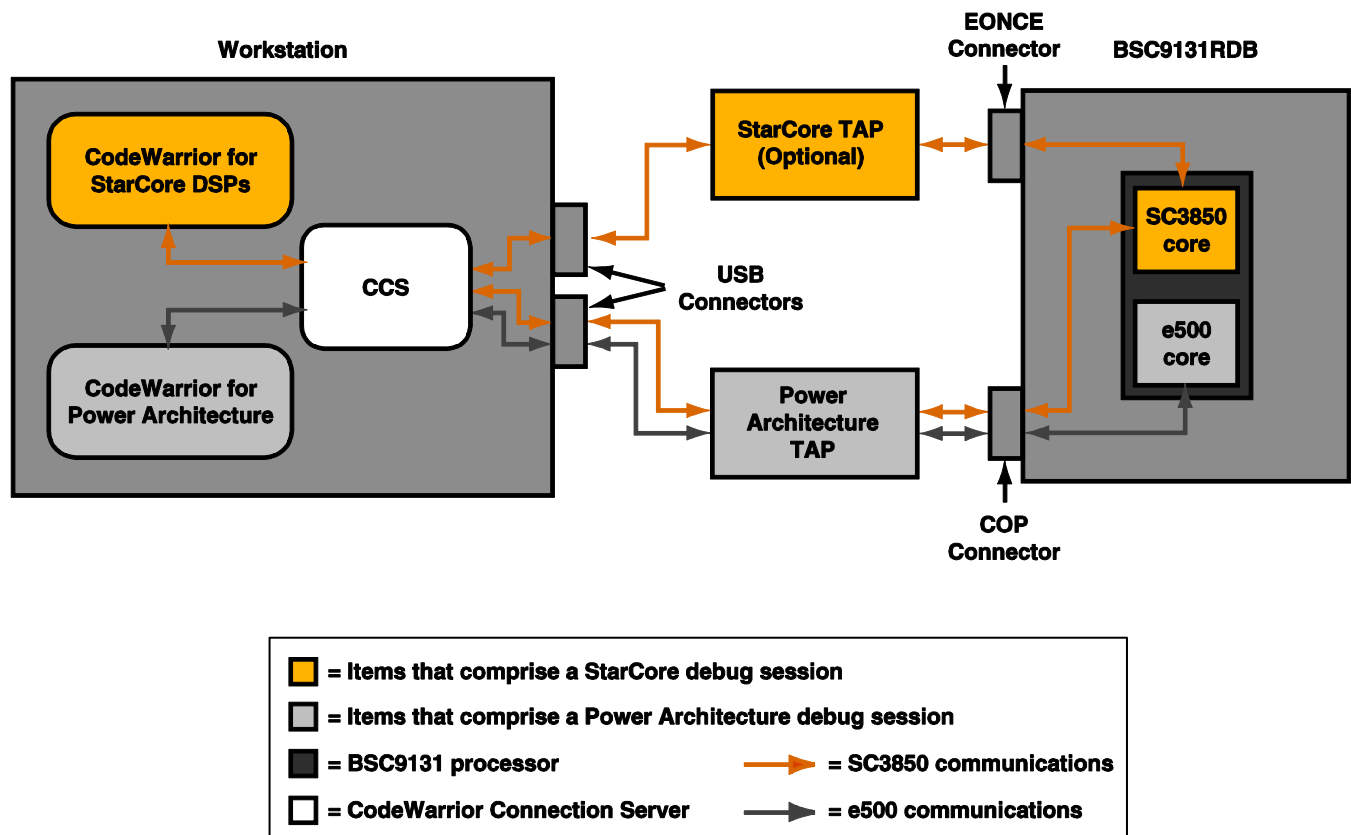


Figure 2. Run controller schemes supported by the BSC9131RDB

The type of run controller scheme selected requires that certain configuration choices be made when setting up a CodeWarrior project. These choices provide default settings that support a specific run controller scheme.

NOTE

The choice of TAP scheme is made in the **Debug Configurations** window. However, for the CodeWarrior for StarCore DSPs tools, the choice of TAP scheme can also be in the New Project wizard. This document shows the choice being made in the wizard.

In addition to the run controller scheme, during the initial debug process it is preferable to have debug configurations that disable the caches in each core. Once the program algorithms and control logic are debugged and tested, debug configurations that enable the caches can be used for performance tuning the code.

All of these choices (number of run controllers, debug with or without caches) determine which CodeWarrior build and debug configuration to use. For a CodeWarrior project named BSC9131, [Table 1](#) summarizes the debug configuration choices available for a given connection scheme and cache use.

Table 1. Debug configuration choices for debugging with a specific TAP scheme

Core	Cache	Single TAP	Two TAPs
SC3850	On	BSC9131_Debug_PSC9131_HW_RDB_Core 00	BSC9131_Debug_PAC9131 ^{SC} _HW_RDB_Core 00
	Off	Custom-built BSC9131_Debug_PSC9131_HW_RDB_Core 00	Custom-built BSC9131_Debug_PAC9131 ^{SC} _HW_RDB_Core 00
e500	On	BSC9131-core0_Cache_PSC9131_Download	BSC9131-core0_Cache_PSC9131 ^{PA} _Download
	Off	BSC9131-core0_RAM_PSC9131_Download	BSC9131-core0_RAM_PSC9131 ^{PA} _Download

Notice that when a two-TAP scheme is used, the debug configuration names have a core-specific suffix as part of the configuration name. This suffix, marked in red in the Two TAPs column above, is either SC for the SC3850 core, or PA for the e500 core. The suffix indicates that the initialization sequence of the debug configuration scans the board JTAG chain only for its specific processor core and disregards the other one.

Finally, to debug SC3850 code with the caches off, adjustments must be made to the StarCore project's build configuration settings and linker command file to disable that core's L1 and L2 caches. Information on how to do this can be found in [Appendix A](#).

NOTE

Currently, the CodeWarrior tools still use the part name PSC9131, rather than BSC9131, in the wizard and for generating the debug configuration names. This behavior will be changed in future tool releases.

3 BSC9131RDB board setup

Figure 3 shows a section of the BSC9131RDB board. The TAP connection headers and switch block SW6, which determines the TAP scheme, are identified.

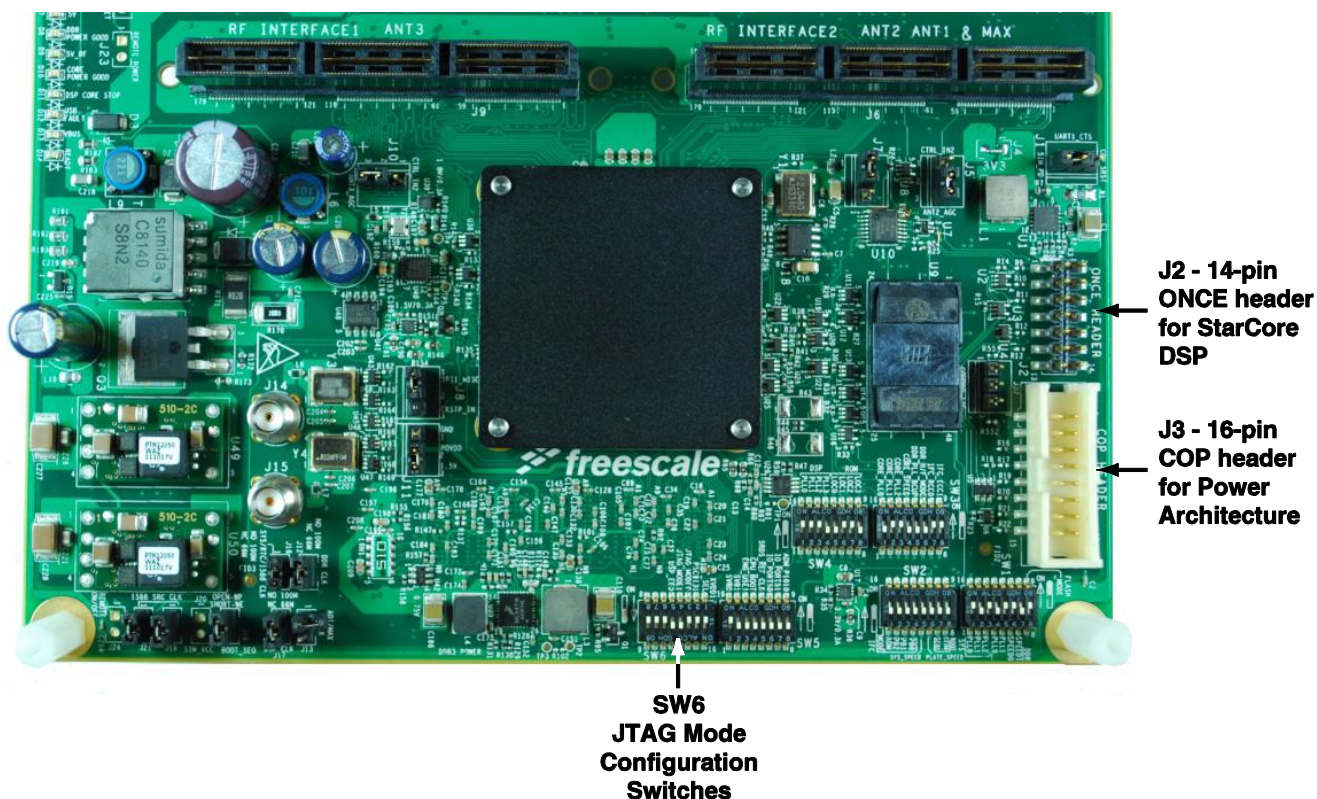


Figure 3. Key items on the BSC9131RDB board for debug setup

3.1 TAP connections

Note that the StarCore OnCE header (used to interface to the OCE port of the SC3850 core) differs from that of the Power Architecture COP header. Therefore, a two-TAP scheme requires two different TAP run controllers. A one-TAP scheme uses a TAP run controller connected to the COP header. [Table 2](#) contains the information on the required USB TAP run controllers and their part numbers. For information on using Gigabit TAP run controllers over an Ethernet connection, see [Appendix C](#).

Table 2. USB TAPs used for BSC9131RDB debugging

Board header type	TAP name	Part number
14-pin ONCE	USB TAP for StarCore	CWH-UTP-ONCE-HE
16-pin COP	USB TAP for JTAG/COP Power Architecture	CWH-UTP-PPCC-HE

3.2 Switch settings

DIP switches three (SW6:3) and four (SW6:4) on switch block SW6 configure the board JTAG mode to support different run controller schemes. Table 3 summarizes the purpose of these switch positions and how they determine which CodeWarrior debug configuration to use. Note that switch position 0 equals ON, and switch position 1 equals OFF.

Table 3. Effect of SW6:3 and SW6:4 on board TAP scheme and debug configuration choice

SW6:3	SW6:4	TAPs used	JTAG topology	CodeWarrior debug configuration ¹
ON	ON	1	Both e500 and SC3850 cores accessed through COP header	BSC9131_Debug_PSC9131_HW_RDB_Core 00 BSC9131-core0_RAM_PSC9131_Download
ON	OFF	1	SC3850 only accessed through COP header	BSC9131_Debug_PAC9131 ^{SC} _HW_RDB_Core 00
OFF	ON	1	Boundary scan	N/A
OFF	OFF	2	e500 accessed through COP header, SC3850 accessed through ONCE header	BSC9131_Debug_PAC9131 ^{SC} _HW_RDB_Core 00 BSC9131-core0_RAM_PSC9131 ^{PA} _Download

¹ Assumes that both CodeWarrior projects use the name BSC9131, and that debugging is done with the core caches disabled.

With both switches set to ON, the board supports the simultaneous debug of SC3850 and e500 code through a USB TAP for the JTAG/COP Power Architecture run controller attached to the COP header. This is the simplest form of debugging for both cores. The joint connection is managed for the debuggers by the CCS; thus, the USB serial numbers are not required as they are in the two-TAP setup scheme described here. For SW6:3 set to ON and SW6:4 set to OFF, SC3850 code can be debugged through a USB TAP for JTAG/COP Power Architecture. With both switches set to OFF, the board supports simultaneous debug of SC3850 and e500 code, with each core managed by its own TAP.

4 Two-TAP connection scheme

This section describes how to set up the BSC9131RDB board to use a two-TAP connection scheme for debugging. It also explains the choices required to generate the proper debug configurations for both CodeWarrior debug sessions. In this example, the CodeWarrior for StarCore DSPs debugger owns the board. Details on board setup can be found in the *BSC9131RDB Hardware Getting Started Guide*.

4.1 TAP setup

Connect a USB TAP for StarCore to the ONCE header (J2) on the board. Connect a USB TAP for JTAP/COP Power Architecture to the COP header (J3) on the board. Use USB cables to connect the TAPs to the workstation that runs the CodeWarrior tools.

Obtain the USB serial number for each TAP. These numbers are required to set up the CodeWarrior debug configurations properly for the two-TAP scheme. If the USB serial numbers are not visible on the devices, use the CCS console command `findcc utaps` to fetch them. With TAPs connected and the board powered, proceed as follows:

1. Start the CCS by going to {CodeWarrior Installation}\SC\ccs\bin or {CodeWarrior Installation}\PA\ccs\bin and launching `ccs.exe`.

The CCS icon appears in the Windows status bar (Figure 4).



Figure 4. CCS icon in the Windows status bar

2. Right-click on the CCS icon to display a menu.
3. Choose **Show console**.

A CodeWarrior Connection Server console window appears, as shown in Figure 5.

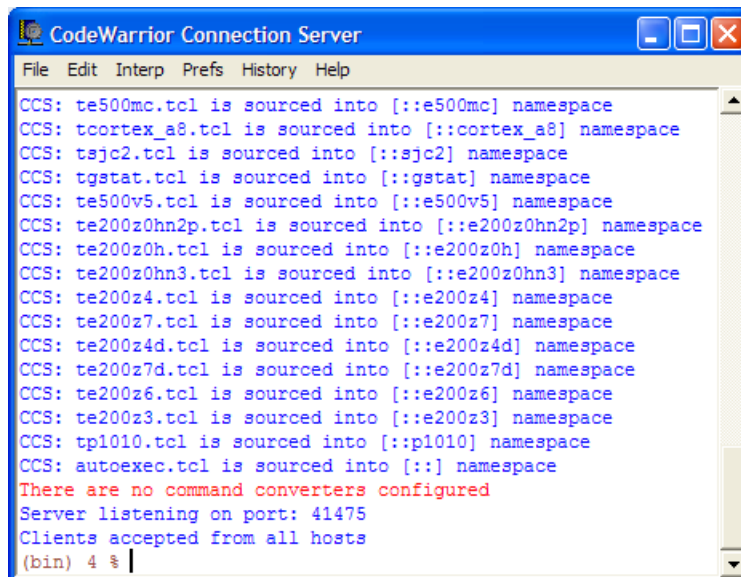


Figure 5. CCS console window

4. Enter `findcc utaps` into the console window.

The console displays the TAPs and their 8-digit serial numbers. Save these numbers for the debug configuration setup.

4.2 Switch positions

Place both switches SW6:3 and SW6:4 into the OFF (1) position.

4.3 CodeWarrior for StarCore DSPs project setup

This section describes the setup of the CodeWarrior project to debug the SC3850 core on the BSC9131. The choice of TAP scheme is made in the CodeWarrior New Project wizard. Choose the correct project template to avoid connectivity errors and to avoid having to update the connection information in debug configuration settings. See [section 2](#) for more information.

NOTE

The BSC9131RDB often has Board Support Package (BSP) software installed in its firmware. This software starts when the board is powered up and can affect debugging code on the SC3850 core. For information on how to eliminate the interaction of the BSP software with the StarCore debugger, see [Appendix B](#).

4.3.1 Select the proper project for the build

1. In CodeWarrior for StarCore DSPs, choose **File > New StarCore Project**.

The **Create a StarCore Project** page appears (Figure 6).

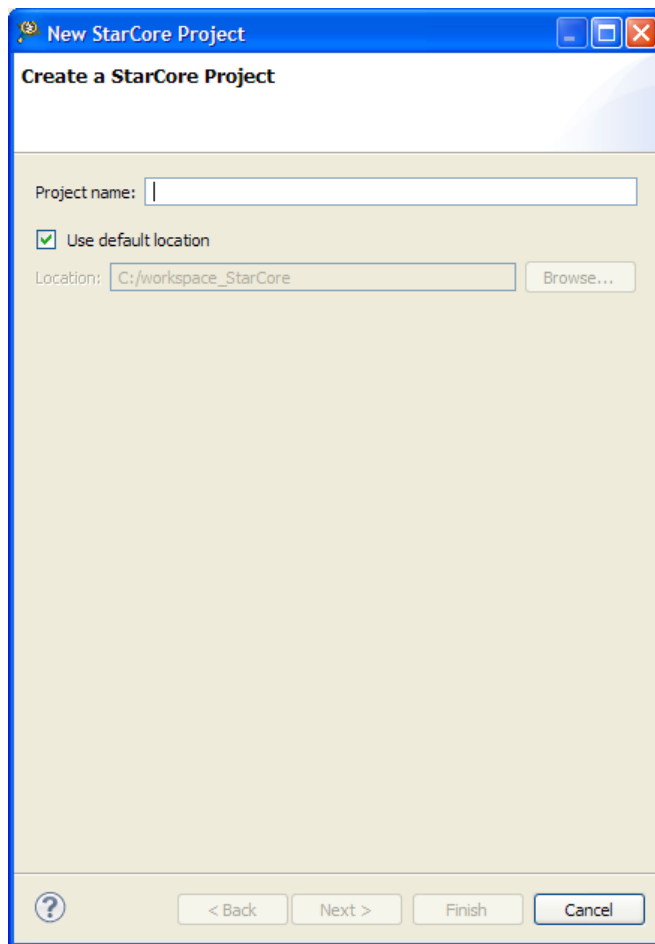


Figure 6. Create a StarCore Project page

2. Enter BSC9131 for the **Project name** option. Click **Next**.

The **Devices** page appears (Figure 7).

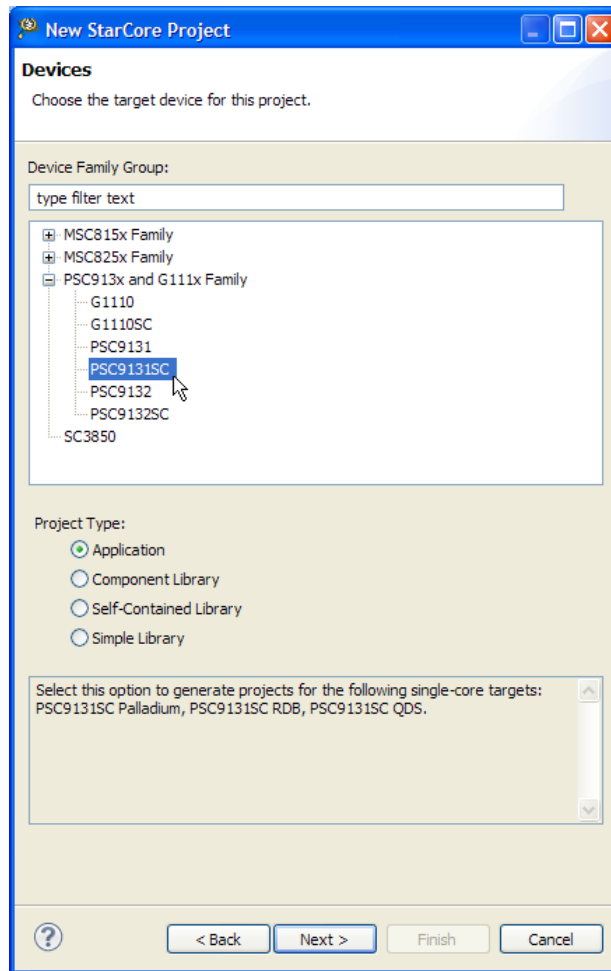


Figure 7. Devices page

3. In the **Device Family** group, choose PSC9131SC from the **PSC913x and G111x Family** list. Click **Next**.

NOTE

For a two-TAP scheme, PSC9131SC is the required choice. See section 2 for details.

NOTE

The choice of part in the wizard currently uses the old name of PSC9131, instead of BSC9131.

The **Build Settings** page appears (Figure 8).

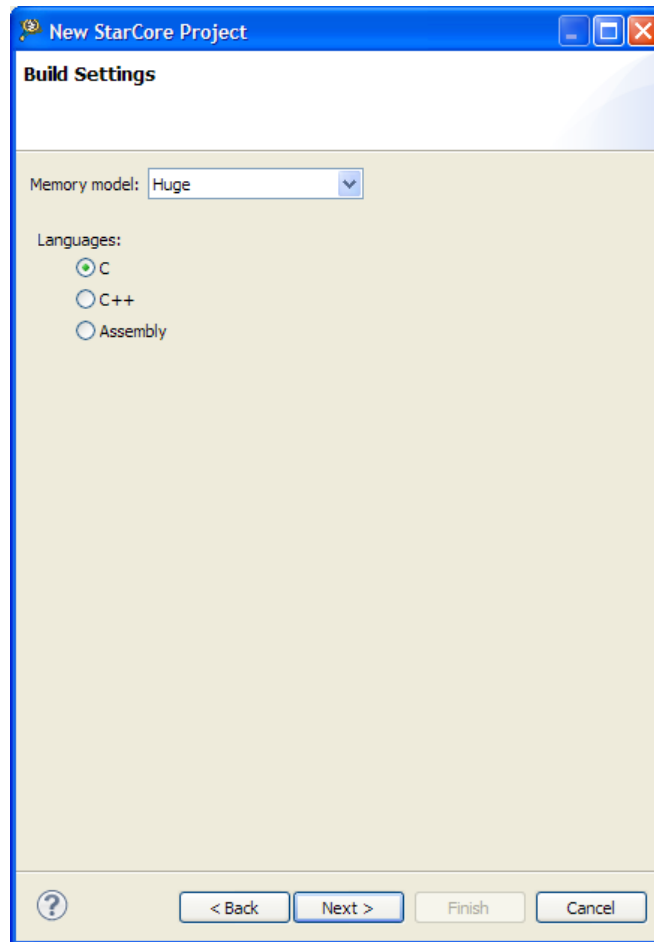


Figure 8. The Build Settings Page

4. If necessary, choose the programming language. Leave the **Memory Model** option on Huge. Click **Next**.

The **SmartDSP OS** page appears (Figure 9).

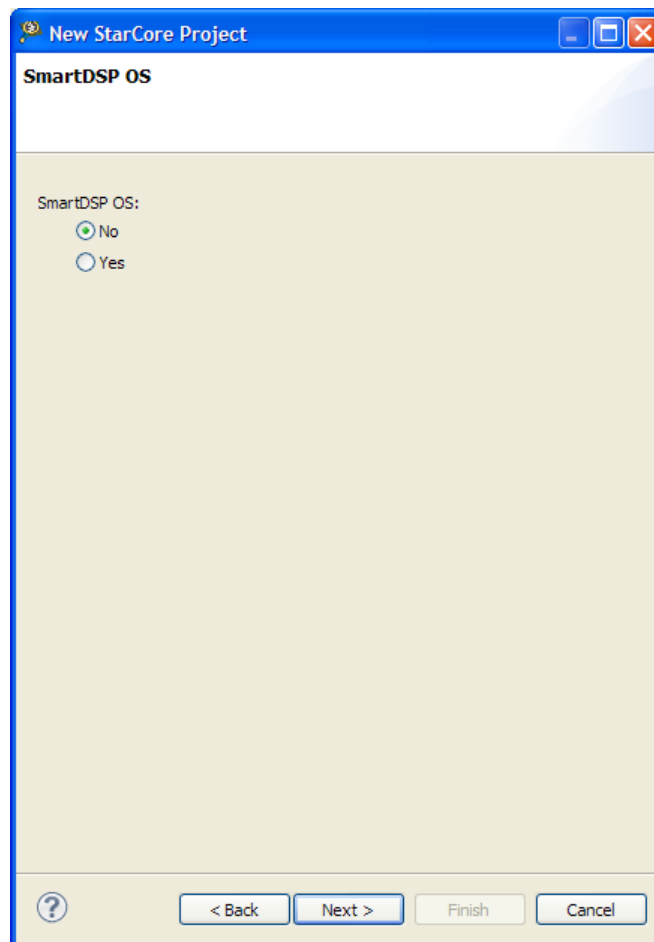


Figure 9. SmartDSP OS page

5. For this example, use the default option of **No** for the **SmartDSP OS** option. Click **Next**.

The **Launch Configurations** page appears (Figure 10).

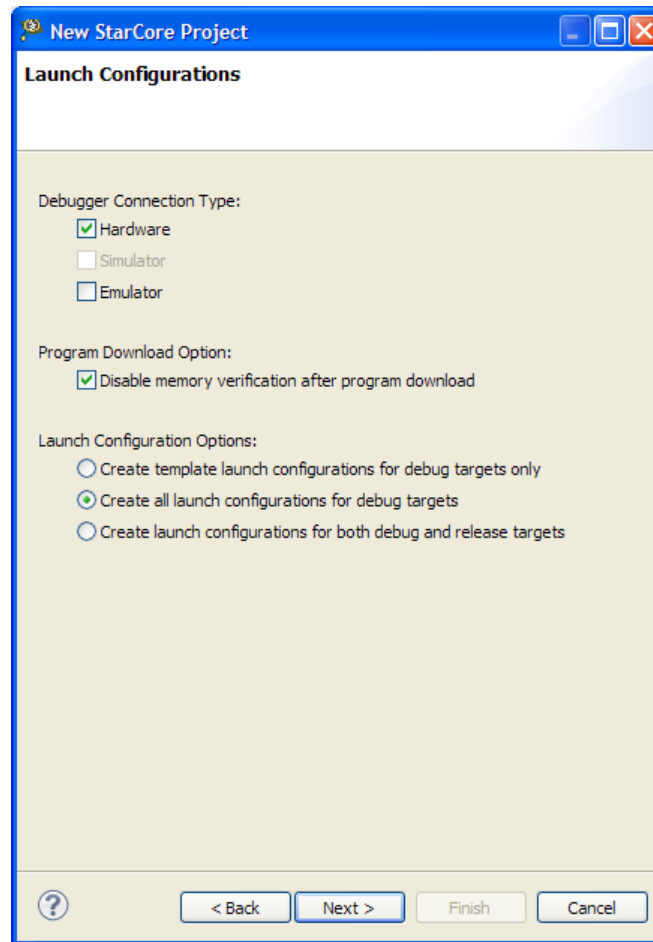


Figure 10. Launch Configurations page

6. For this example, the default options are suitable, so click **Next**.

The **Hardware** page appears (Figure 11).

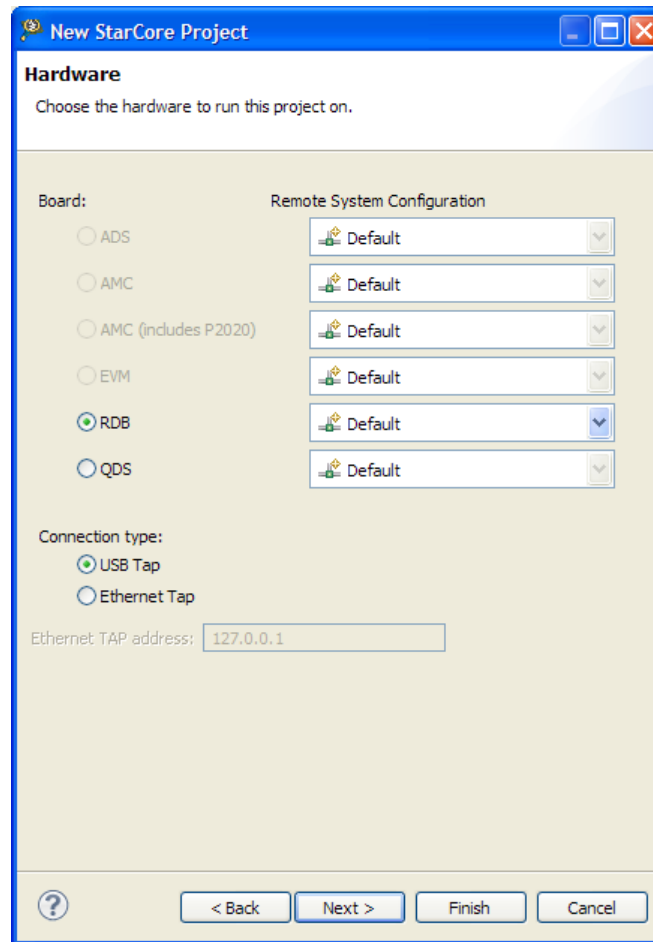


Figure 11. Hardware page

7. Confirm that the **Board** option has **RDB** selected, and that **Connection Type** option specifies **USB Tap**. Click **Next**.

The **Software Analysis Trace and Profile** page appears (Figure 12).

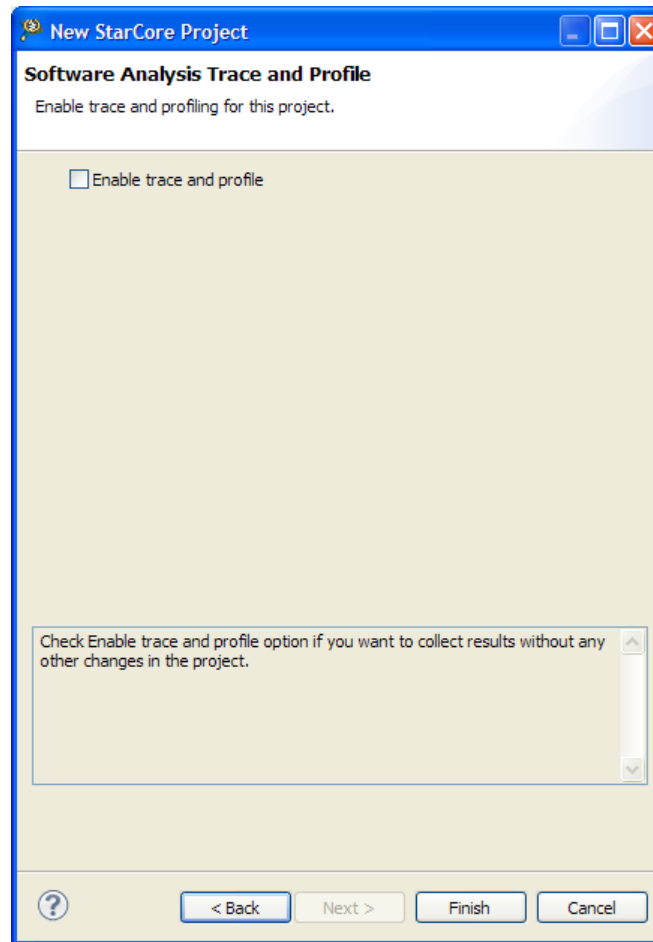


Figure 12. Software Analysis Trace and Profile page

8. Leave the option **Enable trace and profile** unchecked. Click **Finish**.

The wizard creates the StarCore project BSC9131 : Debug_PSC9131SC_HW, which appears in the **CodeWarrior Projects** view (Figure 13).

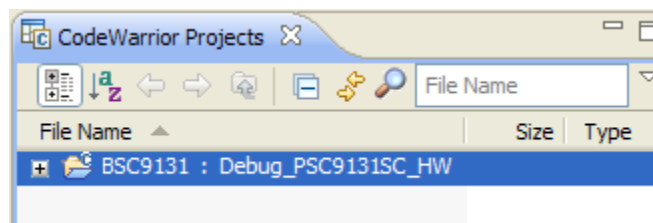


Figure 13. SC3850-specific project displayed by the CodeWarrior Projects view

NOTE

Confirm that the project has the string “SC” as part of the name. If it does not, the debugger will not configure the board JTAG chain properly for a two-TAP scheme. This can be corrected in the debugger settings, by using the **System Type** menu in [Figure 16](#) to select the correct configuration.

9. Build the BSC9131 : Debug_PSC9131SC_HW project. The last few lines of build status information in the **Console** view should confirm the successful generation of a BSC9131.eld file.

NOTE

The default build settings for a StarCore project enable the caches on the SC3850 core. The linker arguments and a linker command file must be modified to disable the caches. See [Appendix A](#).

4.3.2 Set up the debug configuration

Now that the project builds code successfully, the debug configuration must be modified slightly to support the two TAPs. Have the 8-digit USB serial number available for the USB TAP for StarCore during this stage, as it is required for one of the configuration settings.

NOTE

To obtain the USB serial number for the TAP, see section [4.1](#).

1. Choose **Run > Debug Configurations**.

The Debug Configurations window appears (Figure 14).

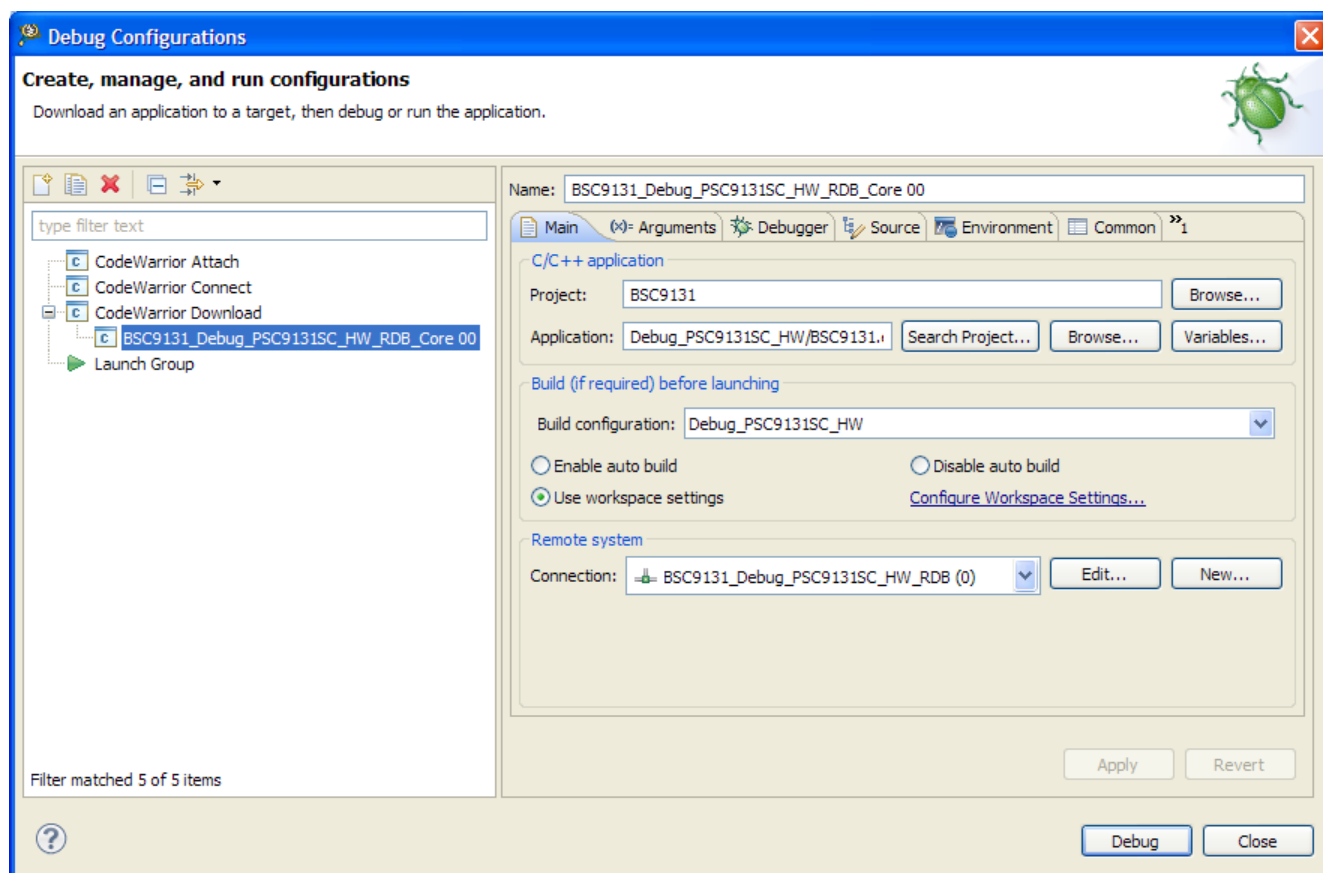


Figure 14. Debug configurations window

2. Expand the **CodeWarrior Download** list and choose the debug configuration BSC9131_Debug_PSC9131SC_HW_RDB_Core 00.

Details as to the debug configuration settings appear at the right.

3. Select the **Main** tab if it is not active. In the **Remote System** group, for the **Connection** option, click **Edit**.

The **Properties for BSC9131_Debug_PSC9131SC_HW_RDB (0)** window appears (Figure 15).

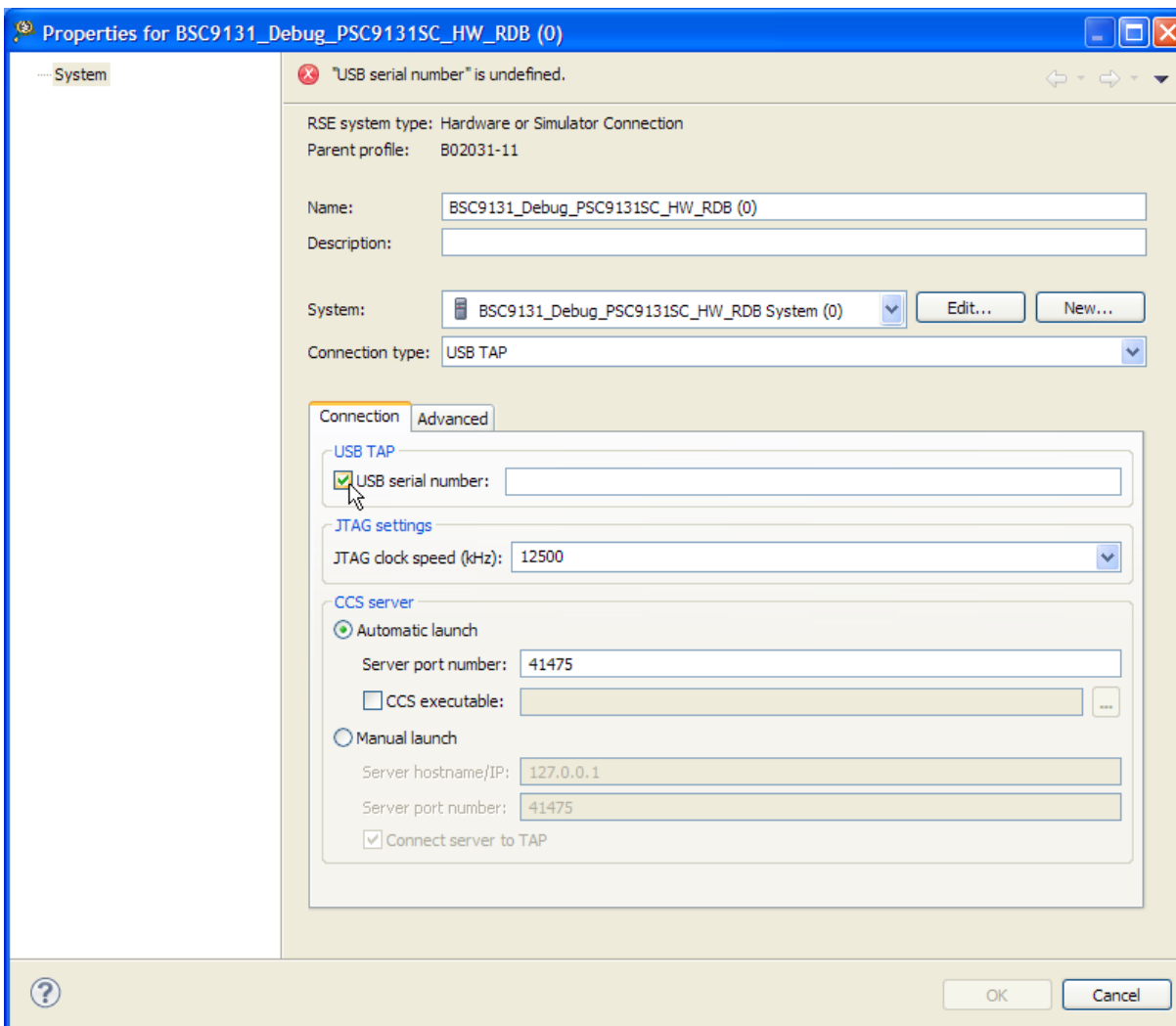


Figure 15. Properties window for the BSC9131 debug configuration

4. Select the **Connection** tab. Under the **USB TAP** group, check the **USB serial number** option, as shown in the figure.

The option box becomes active.

5. Enter the 8-digit USB serial number for the USB TAP for StarCore run controller into the option box.

- Now confirm that the CodeWarrior for StarCore DSPs debugger, as the planned owner of the board, handles the reset responsibilities. Go to the **System** option on this window and click **Edit**.

The **Properties for BSC9131_Debug_PSC9131SC_HW_RDB System (0)** window appears (Figure 16).

Under the **Initialization** tab, the **Execute reset** and **Initialize target** options should be checked.

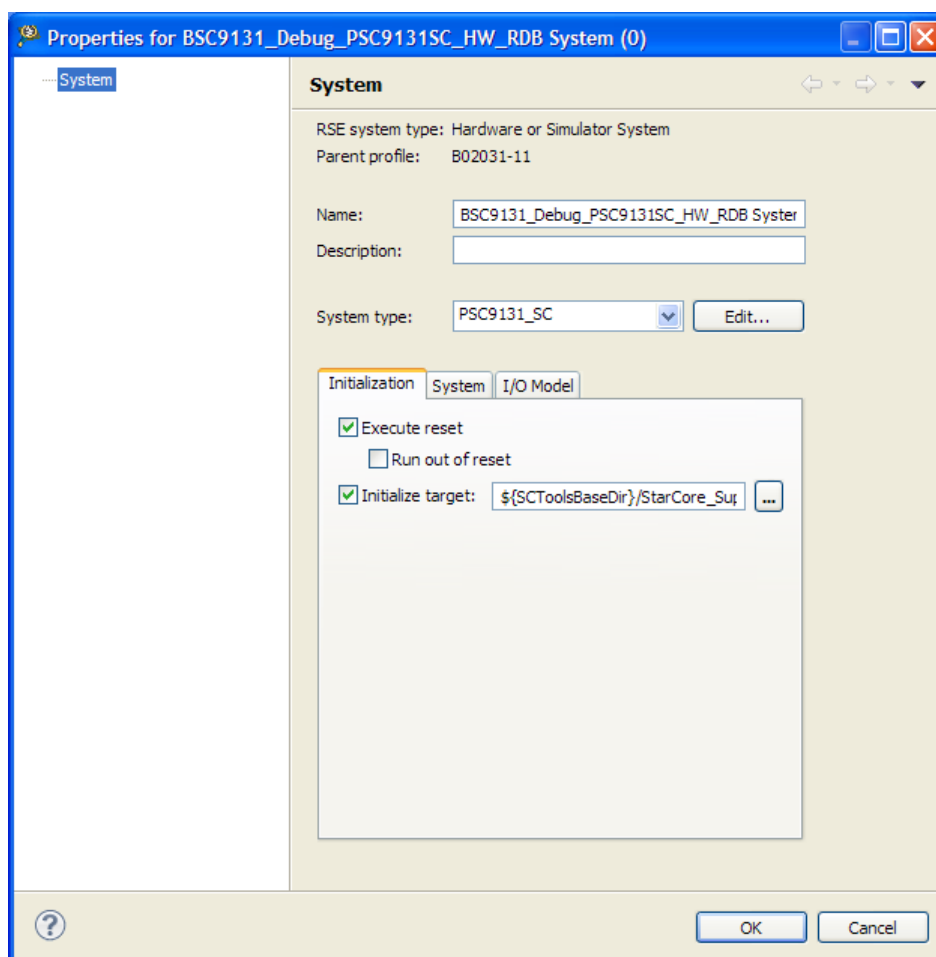


Figure 16. System properties window for the BSC9131 debug configuration

- Click **OK** to dismiss the **Properties for BSC9131_Debug_PSC9131SC_HW_RDB System (0)** window.
- Click **OK** to dismiss the **Properties for BSC9131_Debug_PSC9131SC_HW_RDB (0)** window.
- In the **Debug Configurations** window, click **Apply** to save the changes to the debug configuration.

This completes the setup of the StarCore project and its debug configuration for the SC3850 core.

4.4 CodeWarrior for Power Architecture project setup

This section describes the how to create a CodeWarrior project to debug the e500 core on the BSC9131.

4.4.1 Set up the project

1. In CodeWarrior for Power Architecture, choose **File > New Power Architecture Project**.

A **Create a Power Architecture Project** page appears (Figure 17).

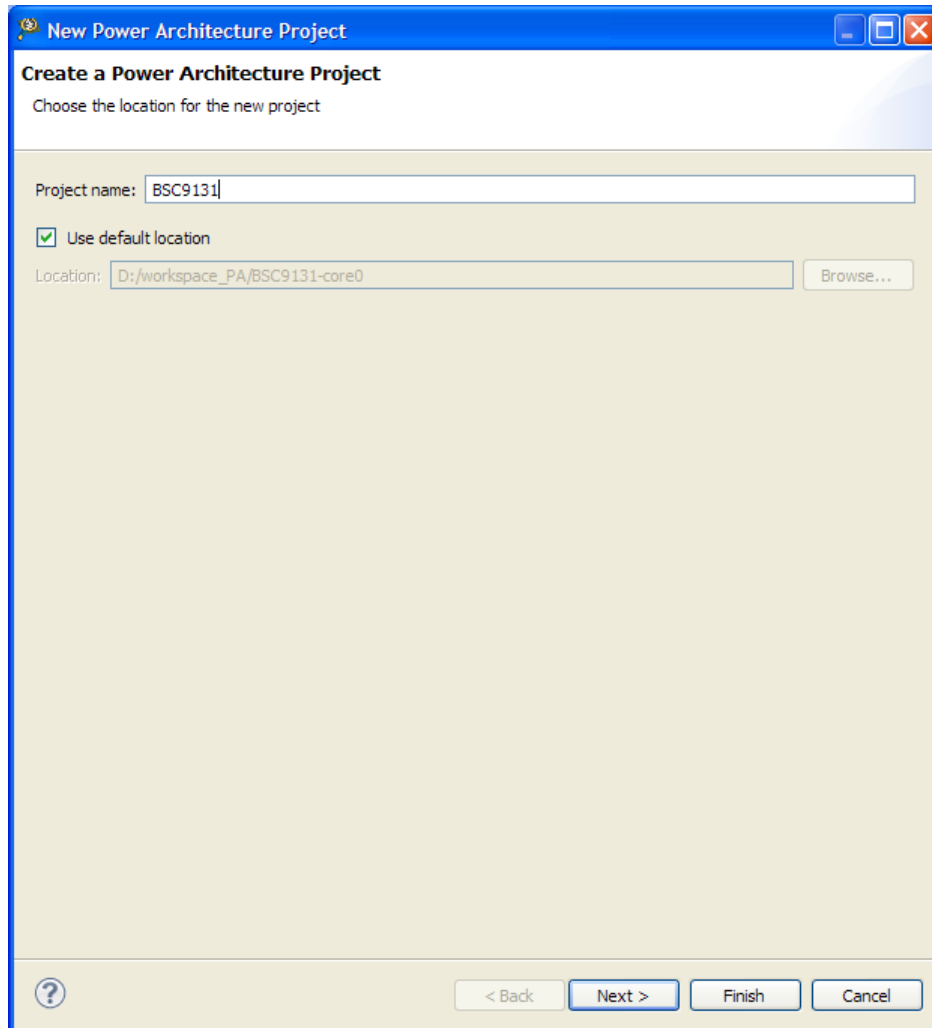


Figure 17. Create a New Power Architecture Project page

2. In the **Project name** option, enter BSC9131. Click **Next**.

The **Processor** page appears (Figure 18).

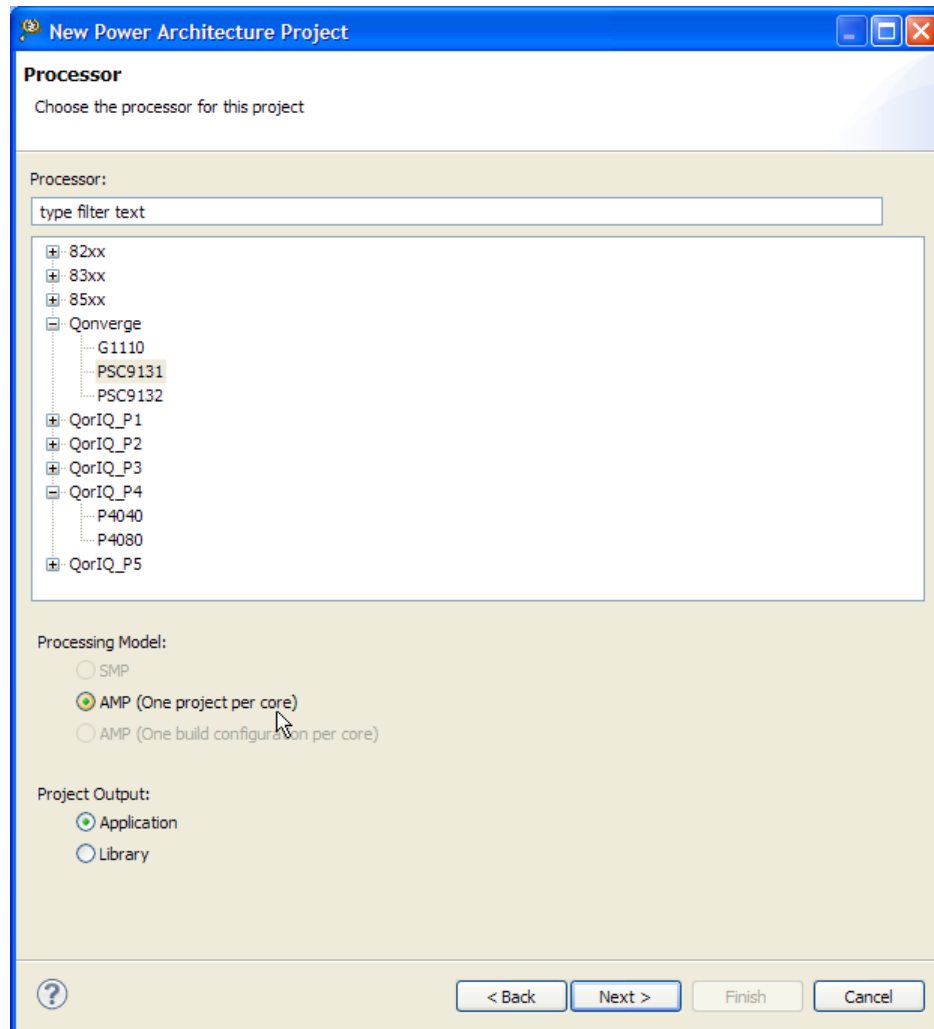


Figure 18. The Processor page.

3. Under the **Processor** option, expand the Qonverge list and choose BSC9131. For the **Processing Model** option, choose **AMP (One project per core)**. Click **Next**.

The **Build Settings** page appears (Figure 19).

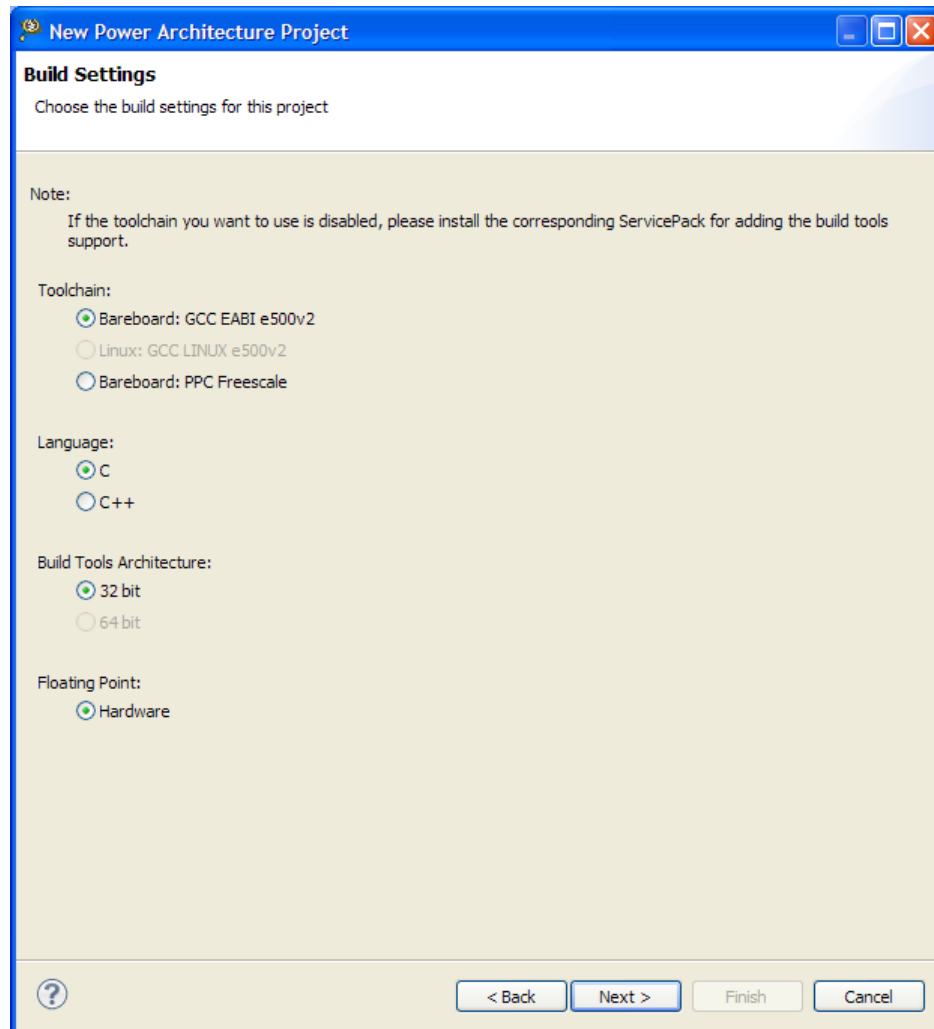


Figure 19. Build Settings page

4. Use the defaults of Bareboard: GCC EABI e500v2 for the **Toolchain** option, C for the **Language** option, 32 bit for the **Build Tools Architecture** option and Hardware for the **Floating Point** option. Click **Next**.

The **Launch Configurations** page appears (Figure 20).

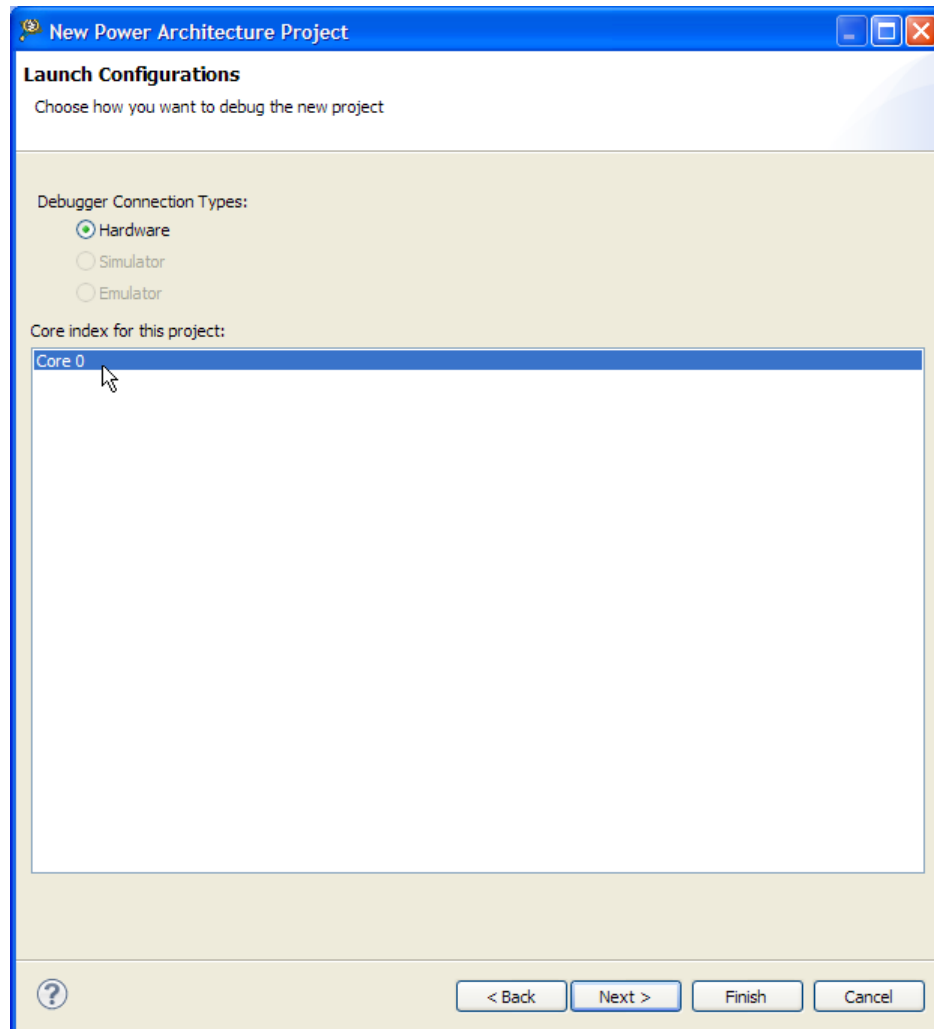


Figure 20. Launch Configurations page

5. For the **Debugger Connection Type** option, choose **Hardware**. For the **Core index for this project** option choose **Core 0**, which highlights it, as shown in the figure. Click **Next**.

The **Hardware** page appears (Figure 21).

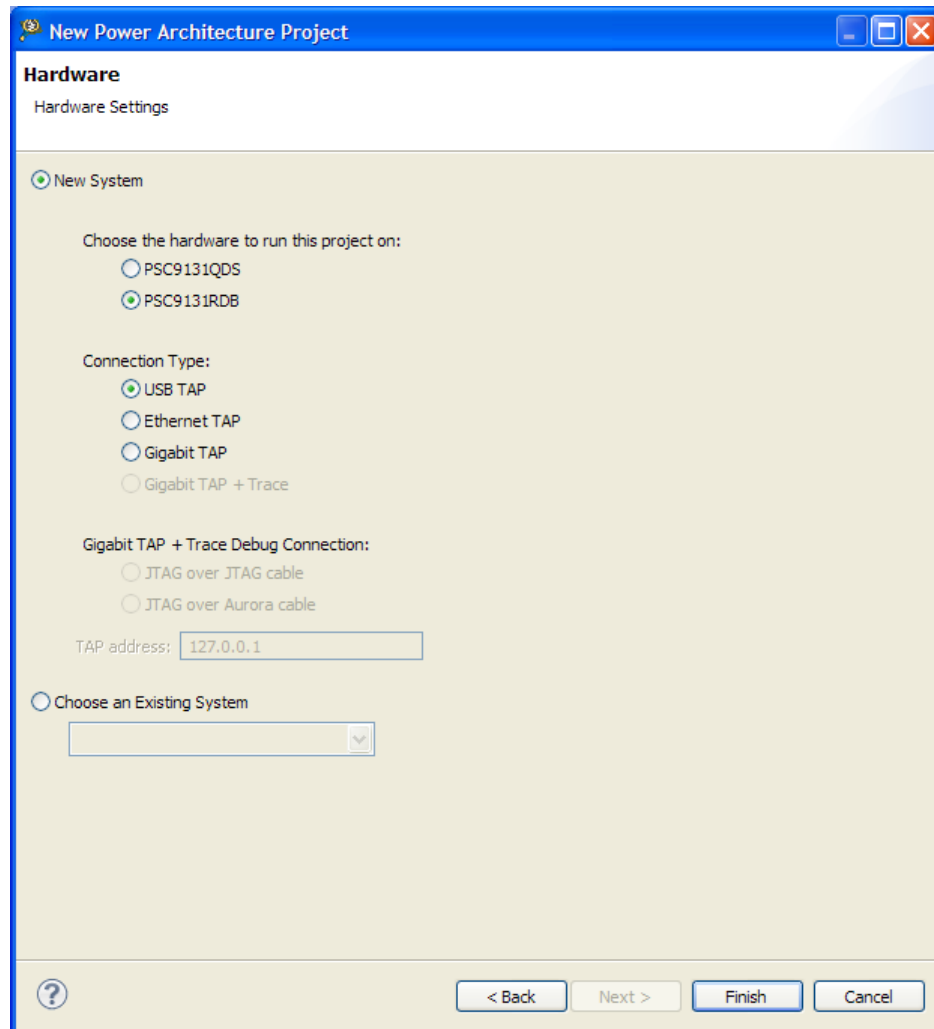


Figure 21. Hardware page

6. For the **Choose the hardware to run this project on** option, select **PSC9131RDB**. For the **Connection Type** option, pick **USB TAP**. Click **Finish**.

The wizard generates the project with the name `BSC9131-core0 : RAM`, and it appears in the **CodeWarrior Projects** view (Figure 22).

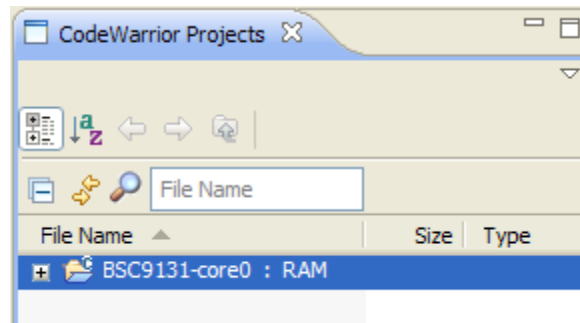


Figure 22. CodeWarrior BSC9131 Power Architecture project

7. Build the `BSC9131-core0 : RAM` project. In the **Console** view, use the status messages to verify that the file `BSC9131-core0.elf` was generated.

4.4.2 Set up the debug configuration

Now that the project builds successfully, the debug configuration must be set up. This process accomplishes the following:

- The choice of debug configuration determines the TAP scheme and whether the e500 core caches are enabled during debugging.
- Since this debug session is the secondary one, the default action of resetting the core must be disabled.

See [Table 1](#) for further information on the debug configuration choices.

Have the 8-digit USB serial number available for the USB TAP for JTAG/COP Power Architecture during this stage, as it is required for one of the configuration settings.

NOTE

To obtain the USB serial number for the TAP, see section [4.1](#).

1. Select **Run > Debug Configurations** to display the settings window.

The **Debug Configurations** window appears (Figure 23).

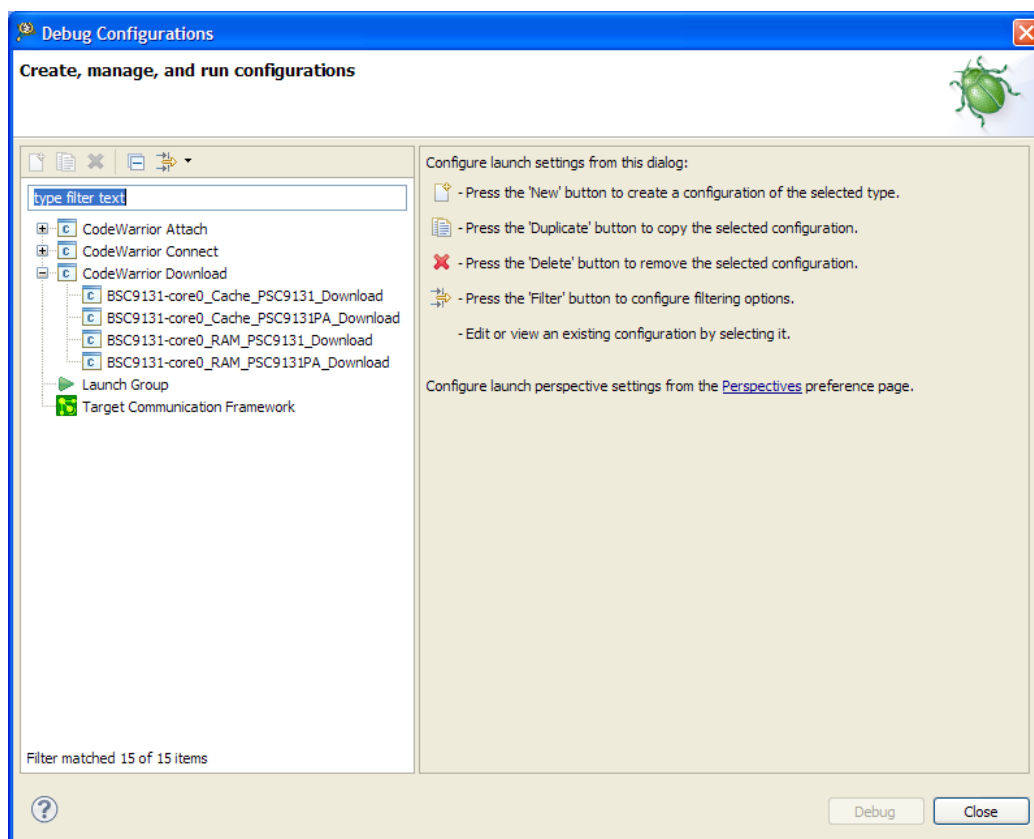


Figure 23. Debug Configurations window

2. Expand the `CodeWarrior Download` list to display four debug configurations for the BSC9131 project.

The choice of debug configuration determines the TAP scheme and whether the core caches are enabled. Strings in the configuration name indicate if the caches are enabled; in general, see the following:

- The string `_Cache` specifies that the core caches are enabled.
- The string `_RAM` specifies that the core caches are disabled.
- The appearance of `PA` in the debug configuration name indicates that it supports a two-TAP scheme.

3. For this example, a two-TAP scheme is used with the caches disabled, so choose the BSC9131-core0_RAM_PSC9131PA_Download configuration. Note that the name also uses PSC9131PA, which specifies the two-TAP scheme. Details of the settings for this particular configuration appear at the right in the **Debug Configurations** window (Figure 24).

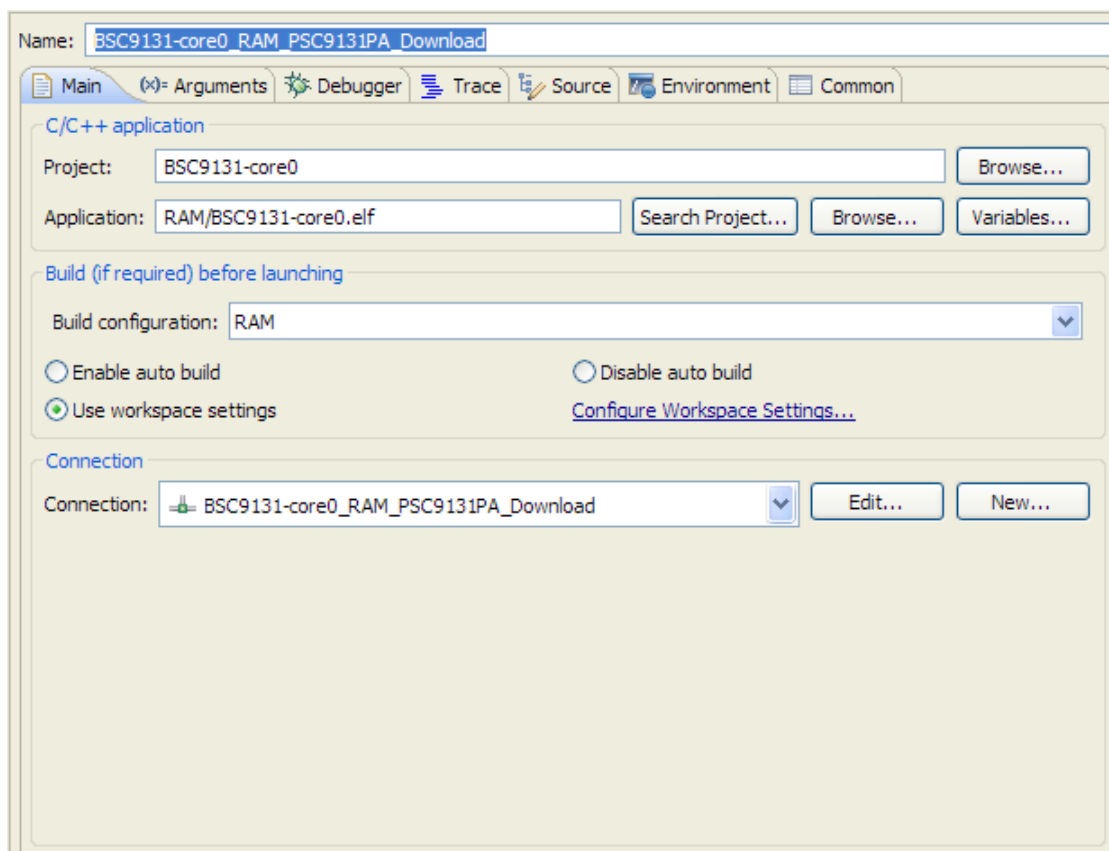


Figure 24. Settings for the BSC9131 Power Architecture project

Confirm that the **Main** tab is selected.

4. Under the **Remote system** group, for the **Connection** option, click **Edit**.

A **Properties for BSC9131-core0_RAM_PSC9131PA_Download** window appears (Figure 25).

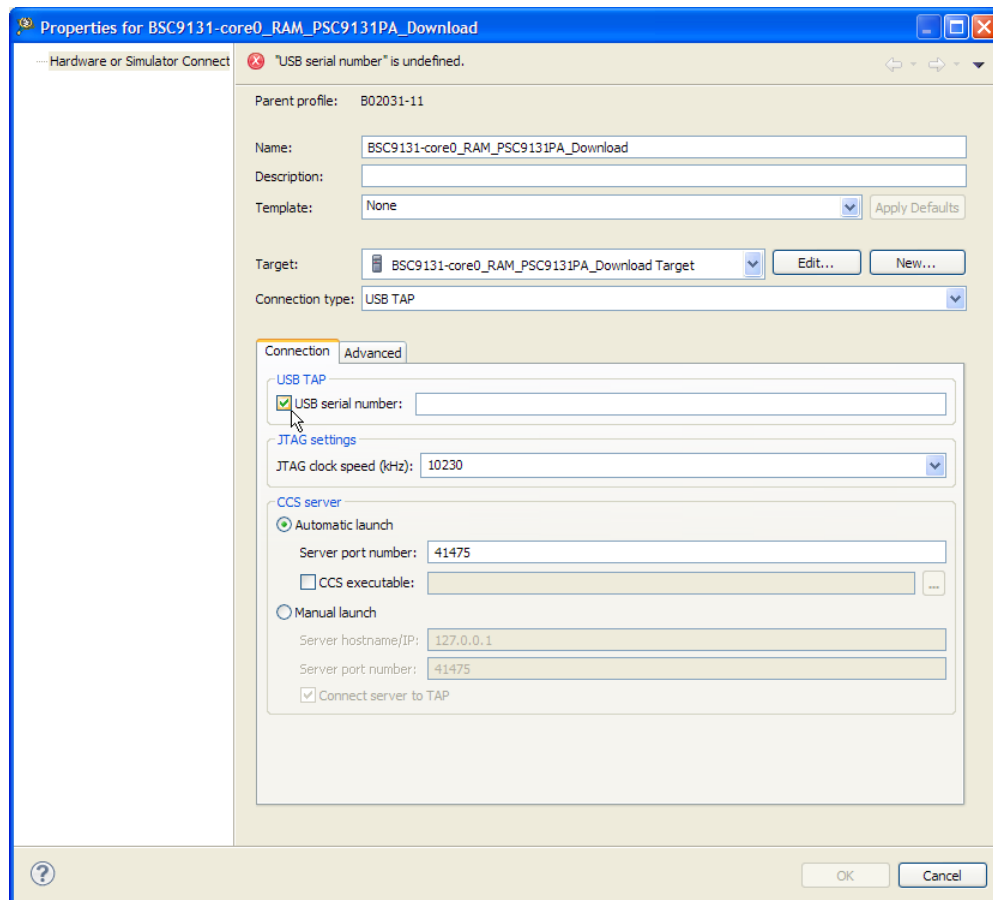


Figure 25. Connection properties settings for the Power Architecture project

5. Click on the **Connection** tab to select it. Under the **USB TAP** group, check the **USB serial number** option, as shown in the figure.

The option box becomes active.

6. Enter the 8-digit USB serial number for the USB TAP for JTAG/COP TAP for Power Architecture into the option box.

7. For the **Target** option, click **Edit**.

The **Properties for BSC9131-core0_RAM_PSC9131PA_Download Target** window appears (Figure 26).

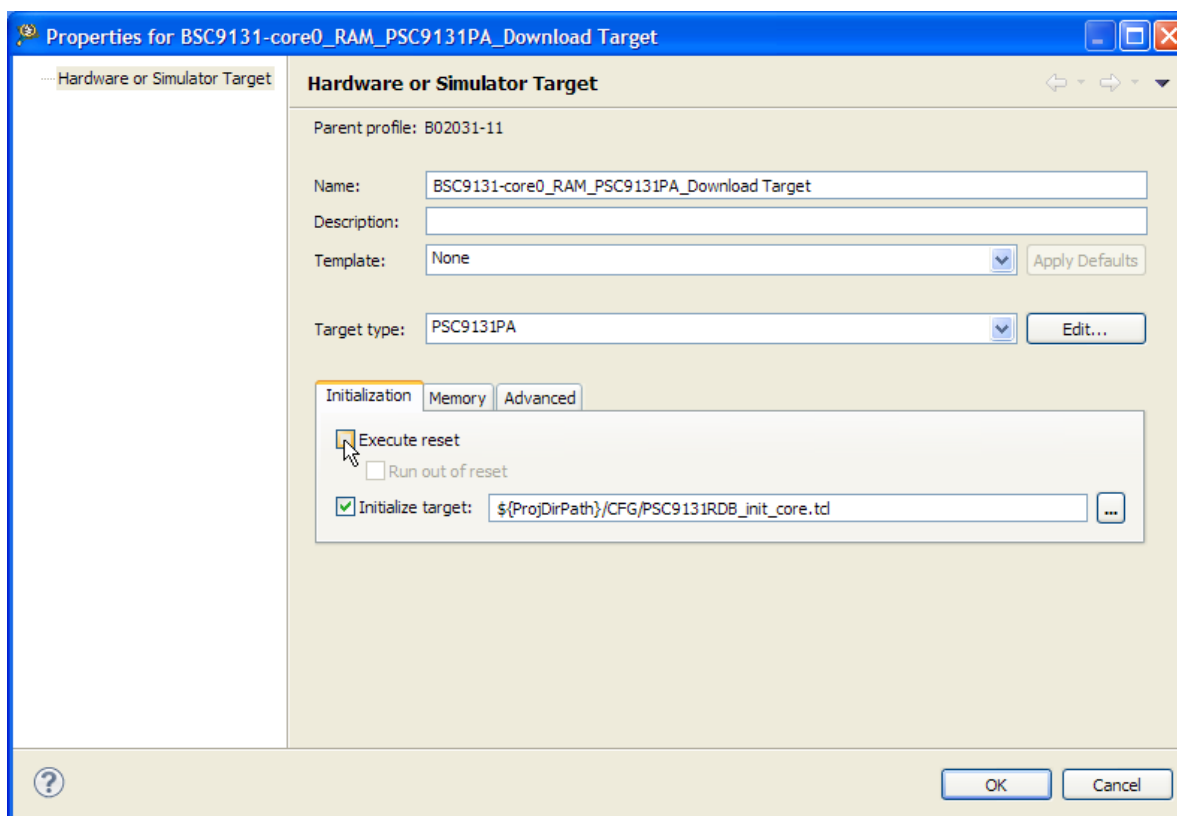


Figure 26. Target Properties window for the project

8. Because this debug session is the secondary one, under the **Initialization** tab uncheck the **Execute reset** option as shown.
9. Click **OK** to dismiss the **Properties for BSC9131-core0_RAM_PSC9131PA_Download Target** window.
10. Click **OK** to dismiss the **Properties for BSC9131-core0_RAM_PSC9131PA_Download** window.
11. In the **Debug Configurations** window, click **Apply** to save the changes.

This completes the setup of the Power Architecture project and its debug configuration for the e500 core.

4.5 Running the debugger sessions

Now that the debug configuration for both the StarCore and Power Architecture projects is properly set up, it can be used to start debugging sessions on the board. Recall the following:

- The CodeWarrior for StarCore debugger is the primary session and must launch first to own (reset) the board.
- The CodeWarrior for Power Architecture debugger is the secondary session and must be launched after the primary session starts.

If this sequence is not followed, the debuggers will fail to connect to the board, or will act erratically. To start the primary session, perform the following:

1. In the CodeWarrior for StarCore DSPs, choose **Run > Debug Configurations**.

The **Debug Configurations** window appears, if it is not already present on the screen.

2. Choose the `BSC9131_Debug_PSC9131SC_HW_RDB_Core 00` configuration.

The debug configuration settings appear to the right in this window.

3. Click **Debug**.

The **Debug** perspective appears. The debugger resets the BSC9131 processor, downloads the application code onto the SC3850 core and prepares it to launch. When ready, the application thread appears in the **Debug** view (Figure 27).

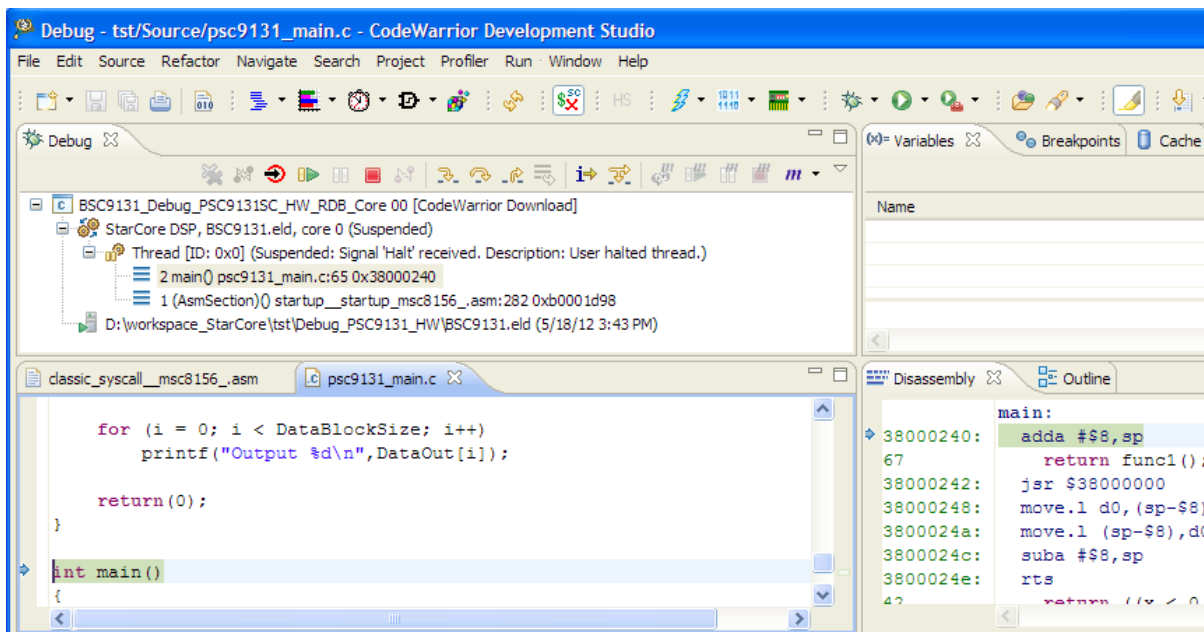


Figure 27. Debug view for the StarCore debugger session

This has established the primary debug session. Now start the secondary debug session, as follows:

1. In the CodeWarrior for Power Architecture, choose **Run > Debug Configurations**.

The **Debug Configurations** window appears, if it is not already present on the screen.

2. Choose the BSC9131-core0_RAM_PSC9131PA_Download configuration.

The debug configuration settings appear to the right in this window.

3. Click **Debug**.

The **Debug** perspective appears. The debugger downloads the application code onto the e500 core and prepares it to launch. When ready, the application thread appears in the **Debug** view (Figure 28).

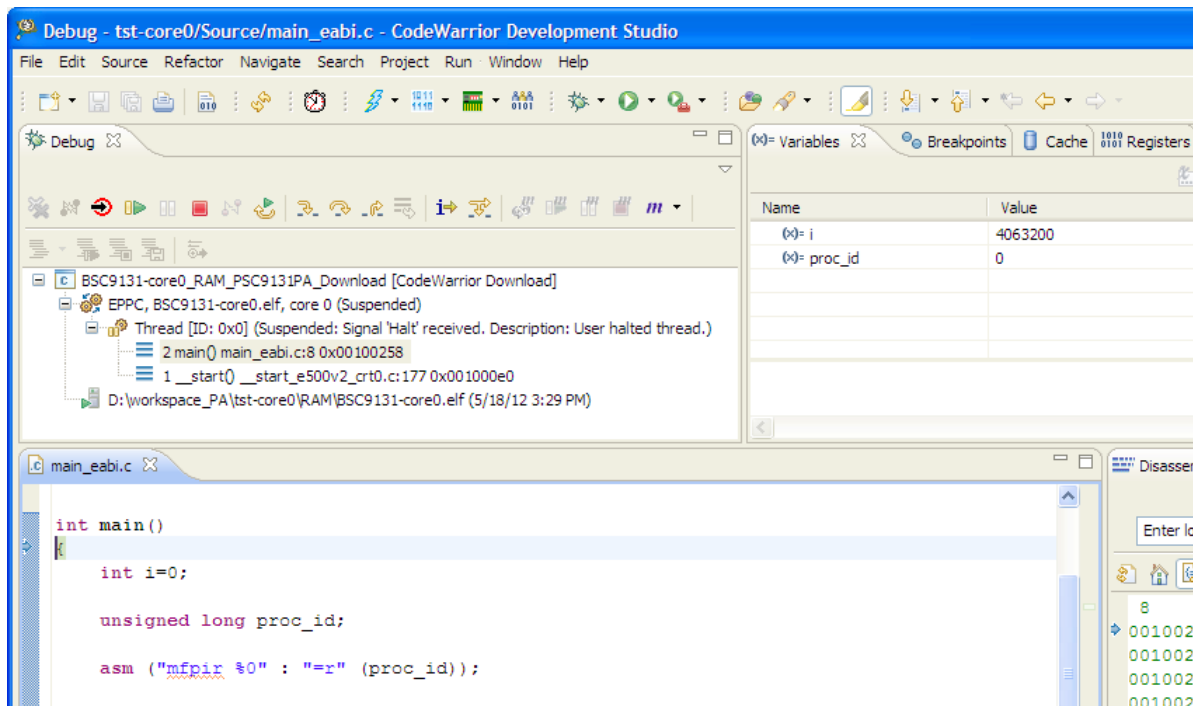


Figure 28. Debug view for the Power Architecture debugger session

The two debugger sessions can be run independently or in tandem, depending upon the nature of the application design.

5 Revision history

Table 4 provides a revision history for this application note.

Table 4. Revision history

Rev. number	Date	Substantive change
0	07/2012	Initial public release

Appendix A

How to Disable the SC3850 Caches

The default build settings for a project generated by the New StarCore Project wizard enable the caches on the SC3850 core. However, for initial code development, these caches should be disabled until the application algorithms and control logic are debugged and tested.

To disable the caches, an argument in the linker command line of the build configuration and a statement in the linker command file `mmu_attr.l3k` must be modified. Specifically, if SmartDSP OS is not being used, for the symbol `ENABLE_CACHE`:

- `ENABLE_CACHE` should be set to `-1` as an argument in the command line for the StarCore linker
- `ENABLE_CACHE` should be set to `-1` in the linker command file `mmu_attr_cpp.l3k`

These modifications are explained in the following sections.

A.1 Modify the linker argument

1. In the C/C++ perspective, select the `BSC9131 : Debug_PSC9131SC_HW` project in the **CodeWarrior Projects** view,
2. Choose **Project > Properties**.

A **Properties for BSC9131** window appears.

3. Expand the **C/C++ Build** list and choose **Settings**.

The **Settings** window appears.

4. Under the **Tool Settings** tab, select **Linker Settings** to display the build arguments that are presented to the linker.

5. Modify the argument `-D_ENABLE_CACHE = 1`, changing it to `-D_ENABLE_CACHE=-1`, as shown in Figure 29.

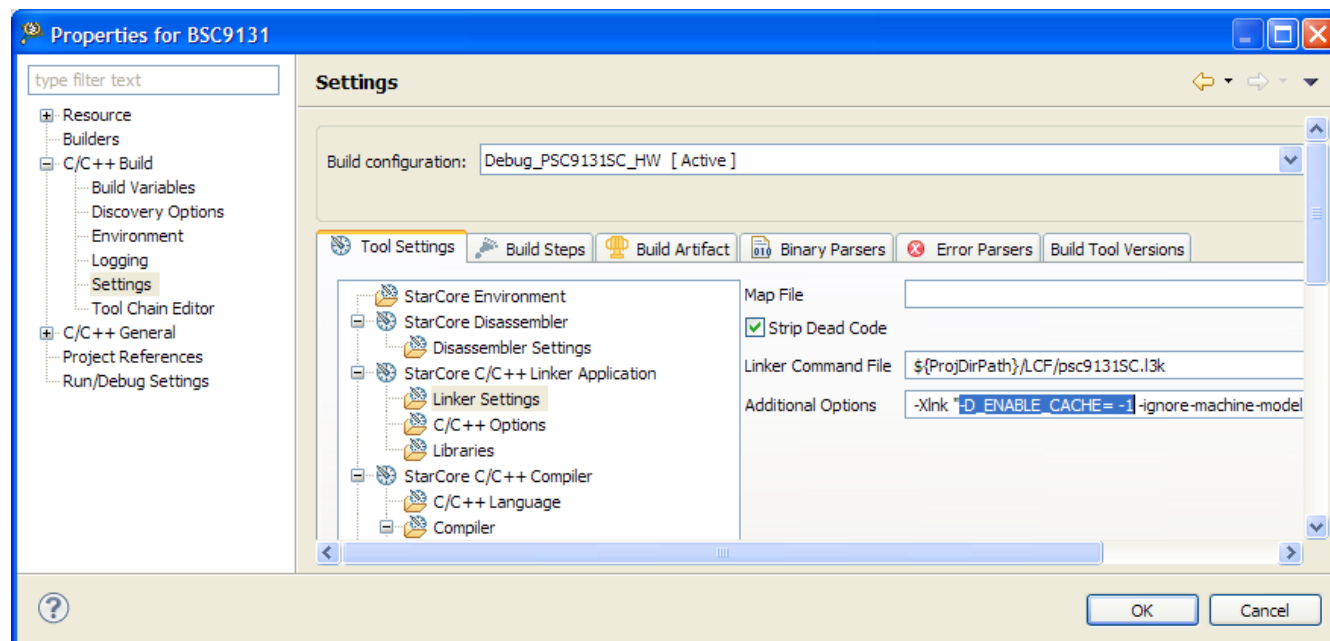


Figure 29. Modifying the linker command line to disable caches

6. Click **OK** to save the changes and dismiss the **Properties for BSC9131** window.

A.2 Modify the linker command file

1. In the **CodeWarrior Project** view, open the LCF folder and double-click on the file `mmu_attr.l3k`.

The contents of the file appear in the **Editor** view (Figure 30).

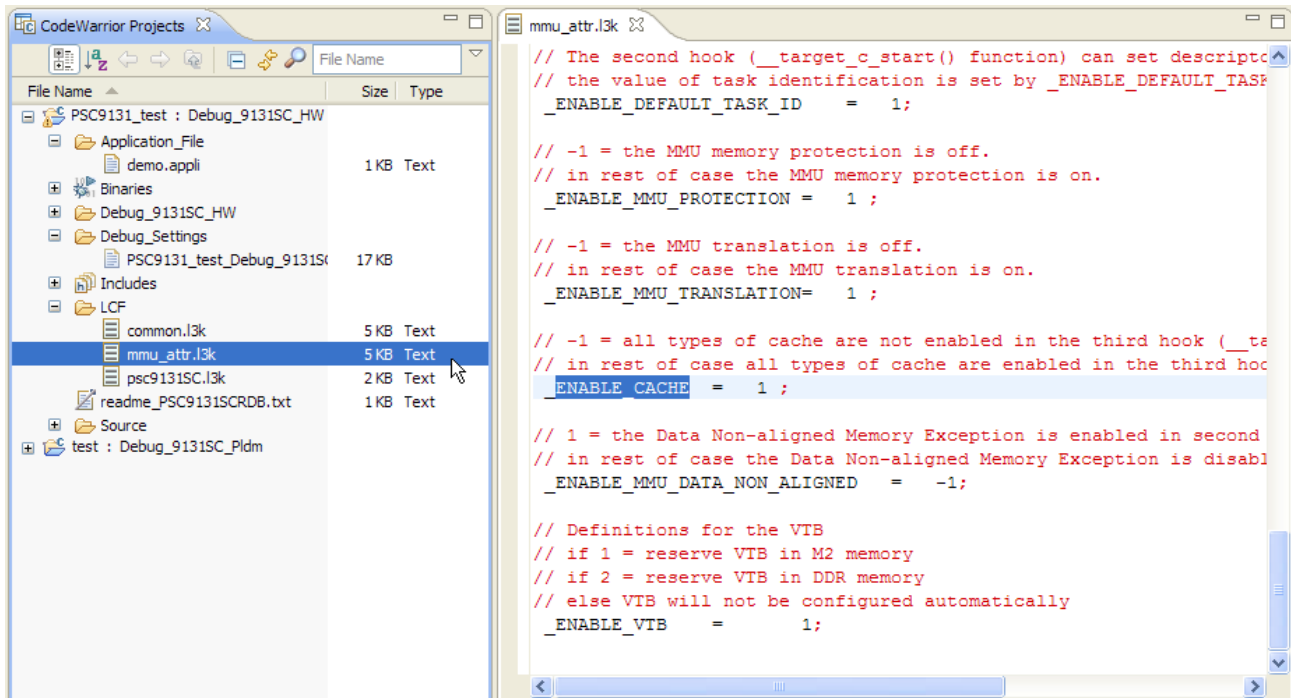


Figure 30. Where to edit the linker command file to disable caches

2. Change the line `_ENABLE_CACHE = 1` to `_ENABLE_CACHE = -1`. Save the changes.

A.3 Disabling the caches in SmartDSP OS

To disable the caches for the SC3850 core when running SmartDSP OS, do not modify the linker command line arguments or the linker command file. This is because SmartDSP OS manages the cache policy for this core. Changing the SmartDSP OS cache policy requires editing a specific header file. Proceed as follows:

1. Expand the `Source` folder for the project in the **CodeWarrior Project** view.
2. Open the file `os_config.h` with the editor by clicking on it.

Modify the following lines:

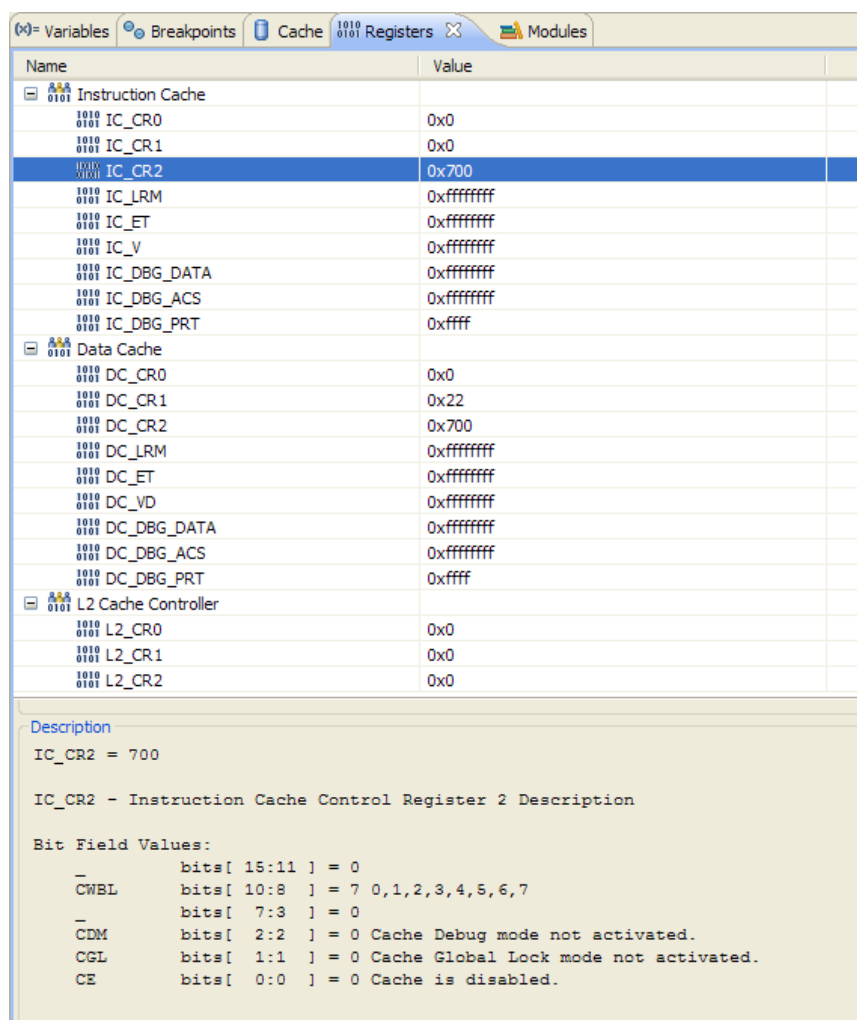
```
#define DCACHE_ENABLE    ON
#define ICACHE_ENABLE    ON
#define L2CACHE_ENABLE   ON
```

Change the values ON to OFF.

3. Save the changes and build the project.

A.4 Verify that the caches are disabled

After the project is built, to confirm that these changes took effect, examine the cache enable (CE) bit in the SC3850 cache control registers. These bits are located in the IC_CR2, CD_CR2, and L2_CR2 registers that control the instruction cache, data cache, and L2 cache, respectively. Open the **Registers** view and then expand the Instruction Cache, Data Cache, and L2 Cache Controller register groups. The CE bit (bit 0) in each of these registers should be clear (zero), as shown in [Figure 31](#).



Name	Value
Instruction Cache	
IC_CR0	0x0
IC_CR1	0x0
IC_CR2	0x700
IC_LRM	0xffffffff
IC_ET	0xffffffff
IC_V	0xffffffff
IC_DBG_DATA	0xffffffff
IC_DBG_ACS	0xffffffff
IC_DBG_PRT	0xffff
Data Cache	
DC_CR0	0x0
DC_CR1	0x22
DC_CR2	0x700
DC_LRM	0xffffffff
DC_ET	0xffffffff
DC_VD	0xffffffff
DC_DBG_DATA	0xffffffff
DC_DBG_ACS	0xffffffff
DC_DBG_PRT	0xffff
L2 Cache Controller	
L2_CR0	0x0
L2_CR1	0x0
L2_CR2	0x0

Description	
IC_CR2	= 700
IC_CR2 - Instruction Cache Control Register 2 Description	
Bit Field Values:	
bits[15:11]	= 0
CWBL	bits[10:8] = 7 0,1,2,3,4,5,6,7
bits[7:3]	= 0
CDM	bits[2:2] = 0 Cache Debug mode not activated.
CGL	bits[1:1] = 0 Cache Global Lock mode not activated.
CE	bits[0:0] = 0 Cache is disabled.

Figure 31. Checking that the SC3850 core caches are disabled

Appendix B

DSP Debugging with a BSP Present

The BSC9131RDB boards come equipped with U-Boot, from WUSDK 1.0 or later, preinstalled in the firmware. When the board is powered up, U-Boot automatically loads and executes on the e500 core. While U-Boot executes, the user must disable cross-triggering in order to reliably debug the DSP core. This means that, to debug on the DSP SC3850 core, a command must be issued on the U-Boot console that allows debug command and control of the DSP core. This procedure is described in the following sequence:

1. Verify that DIP switches three (SW6:3) and four (SW6:4) on switch block SW6 are set to OFF to select the two-TAP scheme. See section 3.2 for more information on the switch positions. This prevents the StarCore debugger's use of the JTAG chain from interfering with the operation of the Power Architecture e500 core.
2. Connect the USB TAP for StarCore DSPs to the ONCE header (J2). Use a USB cable to connect the TAP to the workstation that runs the CodeWarrior for StarCore DSPs tools.
3. Connect an RS-232 serial cable between the host workstation and the BSC9131RDB board UART connector (J25).
4. Run a terminal console program on the workstation. The settings should be as follows:
 - Baud rate: 115200 bps
 - Flow control (for both hardware and software): OFF
5. Apply power to the board.

A U-Boot start-up log should appear on the terminal program. When U-Boot completes its bootstrap process, it presents a command prompt.

6. At the U-Boot command prompt, type: `run debug_halt_off`.
7. The StarCore project must use a debug configuration for a two-TAP project. (That is, the debug configuration name should have a SC3850 core-specific suffix, "SC".) Again, this avoids having the StarCore debugger interfere with the operation of the Power Architecture e500 core. See section 2 for more information on selecting the proper debug configuration.
8. For the StarCore project, in the **Properties for BSC9131_Debug_PSC9131SC_HW_RDB System (0)** window (Figure 16) and under the **Initialization** tab, ensure that the **Execute reset** option is *unchecked*.
9. Build and debug the StarCore project with the CodeWarrior for StarCore DSPs IDE.

More information on the BSP and its programs can be found in the *BSC9131x BSP User Guide*.

Appendix C

Ethernet TAP Run Controller Options

For those using the network to share the BSC9131 RDB board, the Freescale Gigabit TAP can be used. The Gigabit TAP can support both the two-TAP scheme and the one-TAP scheme, because it consists of a base networking module and interchangeable probe tips. Probe tips for both Power Architecture and StarCore board headers are available.

[Table 5](#) summarizes the part configurations for debugging the BSC9131 using the Gigabit TAP.

Table 5. Gigabit TAP information for BSC9131 debugging

Board header type	Part name	Part number
n/a	Gigabit TAP	CWH-GTP-BASE-HE
14-pin ONCE	Gigabit TAP Probe TIP for StarCore	CWH-GTP-STC-YE
16-pin COP	Gigabit TAP Probe TIP for JTAG Power Architecture	CWH-GTP-JTAG-YE

How to Reach Us:

Home Page:

freescale.com

Web Support:

freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: <http://www.reg.net/v2/webservices/Freescale/Docs/TermsandConditions.htm>.

Freescale, the Freescale logo, CodeWarrior and StarCore are trademarks of Freescale Semiconductor, Inc. Reg. U.S. Pat. & Tm. Off. QorIQ Converge is a trademark of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2011-2012 Freescale Semiconductor, Inc.

