

Aurora Trace for CodeWarrior Power Architecture

1. Introduction

This document describes how to prepare the Aurora interface to extract trace from the Aurora port (for processors that supports Aurora) using a Gigabit tap with the Aurora daughter board.

This Gigabit TAP with Trace connects to the Freescale device, using an Aurora SerDes communications channel. The card is selectable between 1 or 2 lanes and 2.5GHz and 3.125GHz, based upon the hardware board. The card has 4 GBytes of RAM to store large volumes of Nexus formatted trace data.

The Gigabit TAP with Trace sends collected trace data to the host computer workstation running CodeWarrior, using the Gigabit TAP's Gigabit Ethernet connection.

This document explains the following:

- Trace configuration
- Enabling Aurora
- Executing Aurora training script
- RCW settings
- Probe settings

Contents

1. Introduction.....	1
2. Preliminary Background	2
3. Configuration	2
4. Preparing Aurora interface for collecting Aurora Nexus trace	6
5. Collecting Data.....	12
6. Viewing Trace Data.....	12

2. Preliminary Background

Tracing is a technique that obtains diagnostic and performance information about a program’s execution. This information is used for debugging and additionally, depending on the type and details of the information contained in the trace log, to diagnose common problems with the software. The trace log includes information about the trace source, type of event, description of the event, and time stamp value.

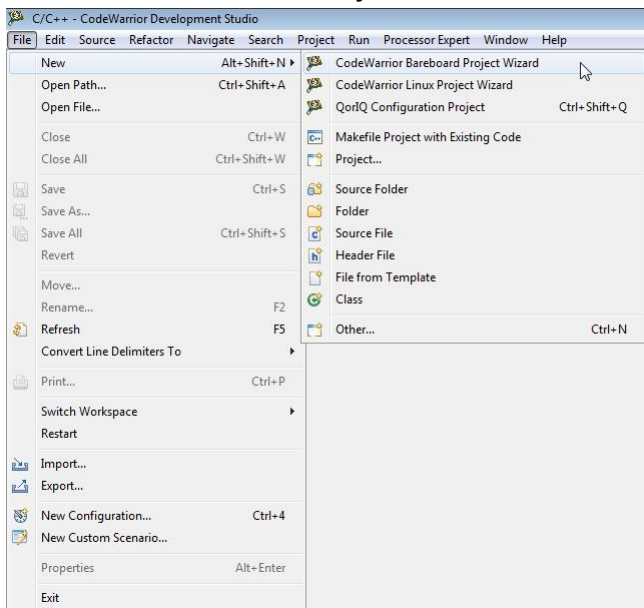
3. Configuration

To collect trace, you need a project to define your application, a launch configuration to set up a debugger connection to the target, and a trace configuration to define the type of trace data you would like to collect.

3.1. Creating a CodeWarrior project

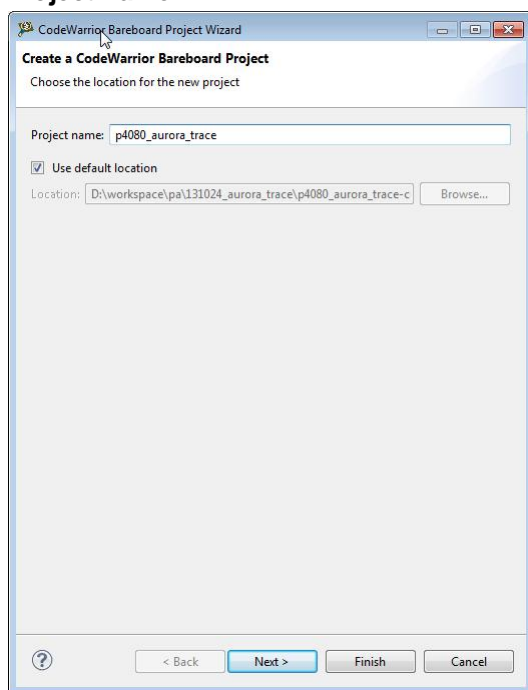
1. Open CodeWarrior IDE.
2. Choose **File > New**, to create a new bareboard project.

Figure 1. CodeWarrior Bareboard Project Wizard



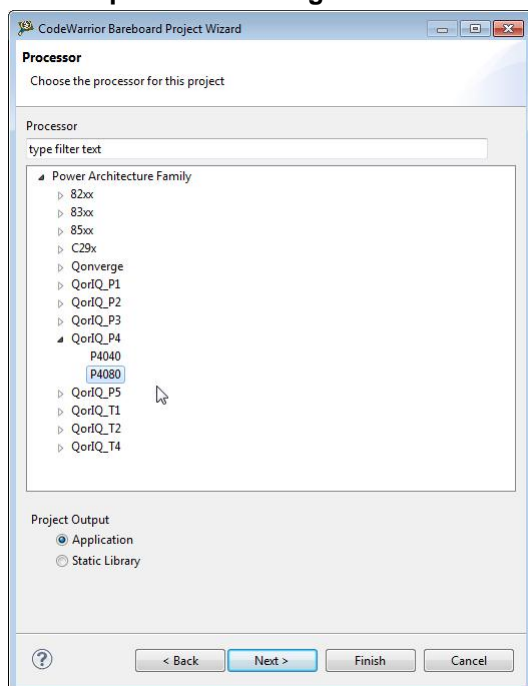
3. Specify **Project Name** and select **Next**.

Figure 2. Project name



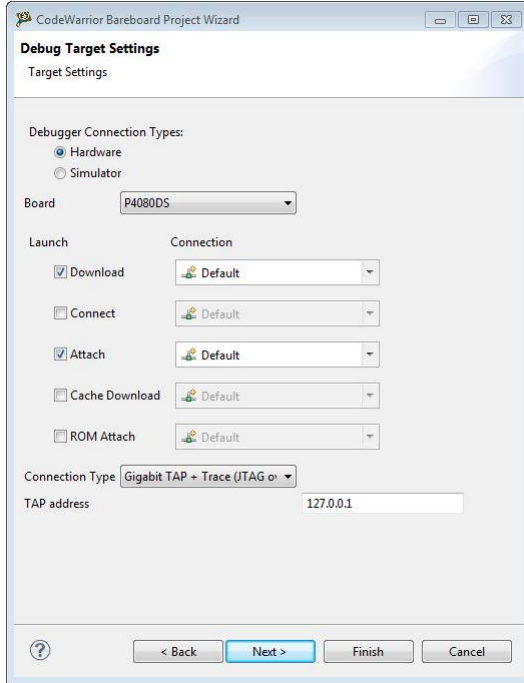
4. Choose **Processor type** and select **Next**.

Figure 3. Choose processor dialog



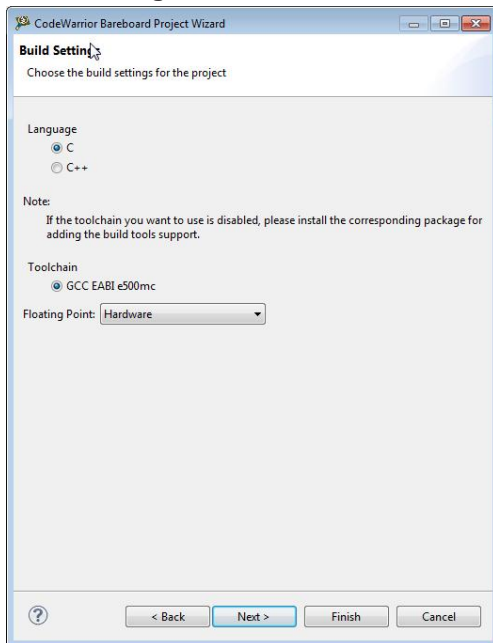
5. Choose **Debugger Connection Type**, **Board**, **Launch Connection**, **Connection Type** - Gigabit TAP + Trace (JTAG over Aurora cable), specify **TAP address** and select **Next**, as shown in [Figure 4](#).

Figure 4. Debug Target Settings



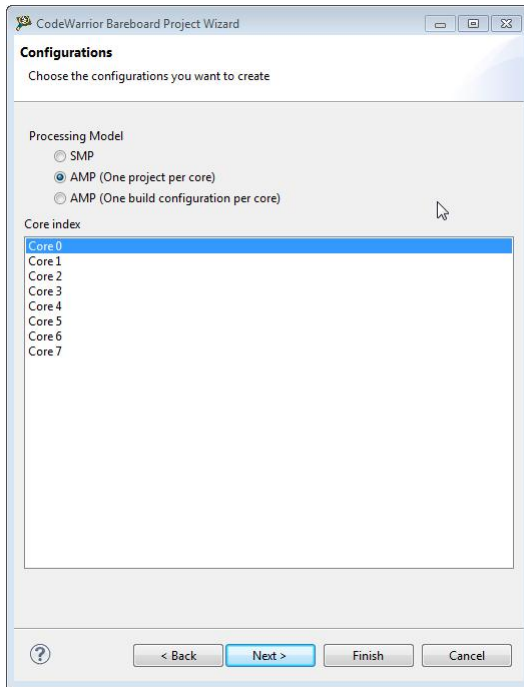
6. From the **Build Settings** dialog, update options as per your requirements and select **Next**.

Figure 5. Build Settings



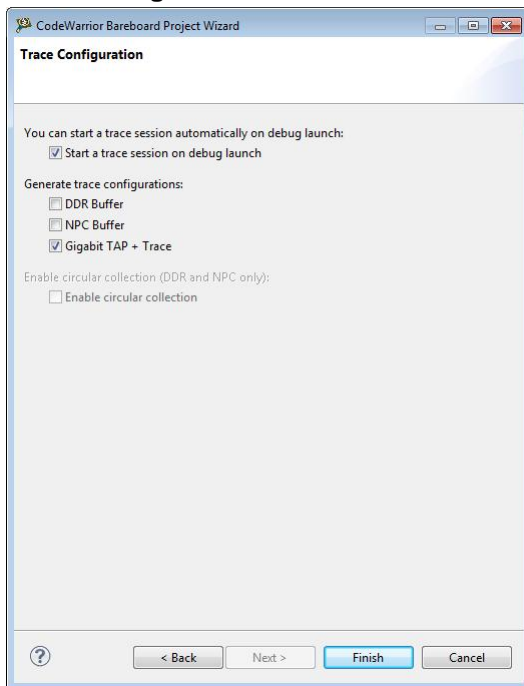
7. From the **Configurations** dialog, select **Next**.

Figure 6. Configuration dialog



8. From the **Trace Configuration** dialog, select **Start a trace session on debug launch** and select **Gigabit TAP + Trace**, then select **Finish** to end the wizard.

Figure 7. Trace Configuration



9. After this, build the project, **Project > Build All**.

3.2. Configuring trace

Define the trace configuration before debugging the application. For more information about configuring trace, see, *CodeWarrior Development Studio for Power Architecture® Processors Version 10.x Tracing and Analysis Tools User Guide*, available at <CWInstallDir>\PA\Help\PDF folder.

4. Preparing Aurora interface for collecting Aurora Nexus trace

For collecting Nexus trace from the target, using Aurora, prepare Aurora interface to be used by the probe. A TCL script or the Aurora training script is used to train Aurora on the target.

4.1. Enabling Aurora

You can enable Aurora on the following boards using RCW:

- B4860QDS
- T4240/T4160QDS
- P2040/41RDB
- P3041DS
- P4080DS
- P5010/20DS
- P5021/40DS
- P5010/20RDB
- P5021/40RDB

NOTE The T4240RDB, T4160RDB, and P4080 PCIe board do not support Aurora. The B4080 board or any other target that does not have PA cores is not supported for collecting trace over Aurora, since there is no core to execute U-Boot.

The RCW and switch configuration for these boards is as follows:

- On all the supported boards, the Aurora debug lanes must be configured to run at 2.5 Gbps.
- For B4860QDS board, the Aurora lanes need to be properly routed to Aurora connector on the board. This is done automatically from U-Boot, when a RCW with Aurora enabled is detected. Ensure that RCW on board is having Aurora enabled (SERDES port #1), and the speed is set to 2.5 Gbps.
- For the T4240QDS board, SW4[7:8]=00 (SW_SD4=00, 100MHz reference clock), SW9[2:3]=00 (SW_SD4_MX_CTL=00, SD4 Lane 4-7 muxed to SATA/Aurora), SW7.8=1, and ensure the RCW on board is having Aurora enabled (SERDES port #4), and the speed is set to 2.5 Gbps.
- For P4080DS board, SW8.8=1

- For P2040/41DS boards, SW2.8=1 to select Aurora and SW[5:6]=00 to select 100MHz reference clock for SerDes bank 2, and SRDS_PRTCL must have lane A set for debug, which can be done using one of the following configuration: SRDS_PRTCL_14 or SRDS_PRTCL_1C.
- For P5010/20DS boards, SW5[1:2]=00 to select a 100MHz SerDes reference clock, and the SRDS_DIV_B1 for the I-J lanes (RCW bit 143) must be ON in the used RCW to enable the clock divider.
- For P5040/21DS boards, SW2.5=0 (LANE_8_SEL) and SW11.6=1 (LANE_9_SEL), and SRDS_PRTCL must have the Debug I-J lanes set for debug.
- For P5010/20/21/40RDB boards, SW8.8=1 to connect the JTAG port to Aurora connector, SRDS_PRTCL must have the Debug I-J lanes set for debug, and the SRDS_DIV_B1 for the I-J lanes (RCW bit 143) must be ON in the used RCW to enable the clock divider.

When Aurora is enabled, the Aurora interface needs to be trained so that it can run properly. For more information about Aurora training, see [Executing Aurora training script](#).

4.2. Execute Aurora training script

The Aurora training script should be executed before configuring or executing trace through the Aurora port. You can execute the training script to train Aurora on all targets that supports Aurora.

- [QorIQ processors from P series](#)
- [QorIQ processors T4240/T4160QDS target](#)
- [QorIQ processors B4860QDS target](#)

4.2.1. QorIQ processors from P series

For the following QorIQ Processor P Series targets, the TCL script that is used to train Aurora on the target is available in the CodeWarrior installation folder at the location specified below:

```
PA/PA_Support/Initialization_Files/Aurora/train_aurora.tcl
```

- P2040/41RDB
- P3041DS
- P4080DS
- P5010/20DS
- P5021/40DS
- P5010/20RDB
- P5021/40RDB

The routine inside this file that executes the required training is called `train_aurora`. There are different ways to execute the Aurora training script on the QorIQ Processor P Series targets depending on the following scenarios.

4.2.1.1. Execute Aurora training script using CodeWarrior

If you are downloading and debugging the application using CodeWarrior, execute the TCL script using the following steps:

1. Copy `train_aurora.tcl` script to your project in the CFG folder.
2. Add the following line to the start of your initialization file:

```
source "path/train_aurora.tcl"
```

3. Add a call to `train_aurora_dbg` procedure in the `init_platform` procedure:

```
train_aurora_dbg
```

4. After debugging the target, the initialization target script is executed, and Aurora is initialized and ready for trace collection.

If the Aurora probe fails training on a clean P2040/41RDB configuration, apply the following workaround to get the probe trained:

1. Start debugging using the `jtag_chain` file and select `train_aurora.tcl` as init file in the connection settings.
2. After the board stops at the start of the main file, open **Debugger Shell** and run the following command:

```
protocol ccs::reset_to_user
```

3. Reset the probe from telnet and wait for `ccs` connection to re-establish.
4. Terminate the current session from CodeWarrior and debug again.

4.2.1.2. Execute Aurora training script from debugger shell

You can execute the training script from the CodeWarrior **Debugger Shell**. Enter the following command in the Debugger Shell to source the training script:

1. In the shell, enter:

```
source "path/train_aurora.tcl"
```

2. In the shell, execute the training script:

```
train_aurora_dbg
```


3. Look for a confirmation message that Aurora interface was successfully trained. If you do not receive the message, execute the training script again.

4.2.1.3. Execute Aurora training script outside CodeWarrior

You can execute the training script from `ccs` when processor is not already debugged by CodeWarrior. Open the `ccs` console, source the `train_aurora.tcl` file, and then execute the training by running the script, `train_aurora`.

1. Open `ccs` console.
2. In the console, enter:

```
% config cc gtap:jtag:1:ip_addr
% config cc gtap:aurora:x2x2:ip_addr
% ccs::config_chain core_name
% source "path/train_aurora.tcl"
% train_aurora
```

3. Look for a confirmation message that the Aurora interface was successfully trained. If it was not successfully trained, execute the training script again.

NOTE Use the custom JTAG file to override RCW and enable the Aurora interface for trace collection.

4.2.2. QorIQ processors T4240/T4160QDS target

For the T4240QDS target, the TCL script that is used to train Aurora on the target is available at the CodeWarrior installation folder:

```
PA/PA_Support/Initialization_Files/Aurora/train_aurora_t4.tcl
```

The routine inside this file which executes the required training is called `train_aurora`. There are different ways to execute the Aurora training script on the T4240QDS target, which depends on the following scenarios.

4.2.2.1. Executing Aurora training using CodeWarrior

If you are downloading and debugging the application using CodeWarrior, execute the TCL script using the following steps:

1. Copy `train_aurora_t4.tcl` script to your project in the CFG folder.
2. Add the following line to execute the initialization file:

```
source "path/train_aurora_t4.tcl"
```

3. Add a call to `train_aurora_dbg` procedure in the `init_platform` procedure:

```
train_aurora_dbg
```

4. After debugging the target, the initialization target script is executed, and Aurora is initialized and is ready for trace collection.

4.2.2.2. Executing Aurora training script from debugger shell

You can execute the training script from the CodeWarrior Debugger Shell. Enter the following command in the Debugger Shell to source the training script:

1. In the shell, enter:

```
source "path/train_aurora_t4.tcl"
```

2. In the shell, execute the training script:

```
train_aurora_dbg
```

3. Look for a confirmation message that Aurora interface was successfully trained. If you do not receive the message, execute the training script again.

4.2.2.3. Executing Aurora training script outside CodeWarrior

You can execute the training script from `ccs` when processor is not already debugged by CodeWarrior. You need to open `ccs` console, source the `train_aurora_t4.tcl` file, and then execute the training by calling `train_aurora`.

1. Open `ccs` console.
2. In the console, enter:

```
% config cc gtap:jtag:1:ip_addr
% config cc gtap:aurora:x2x2:ip_addr
% ccs::config_chain core_name
% source "path/train_aurora_t4.tcl"
% train_aurora
```

3. Look for a confirmation message that Aurora interface was successfully trained. If it was not successfully trained, execute the training script again.

NOTE Use the custom JTAG file to override the RCW and enable Aurora interface for trace collection.

4.2.3. QorIQ processors B4860QDS target

For the B4860QDS target, the TCL script that is used to train Aurora on the target is available in the CodeWarrior installation folder:

```
PA/PA_Support/Initialization_Files/Aurora/train_aurora_b4.tcl
```

The routine inside this file which executes the required training is called `train_aurora`. There are different ways to execute the Aurora training script on the B4860QDS target, which depends on the following scenarios.

4.2.3.1. Settings for debugger launch – General approach

Execute U-Boot, a proper U-Boot detects that Aurora is enabled in RCW and initializes cross-points for routing Aurora signals to Aurora connector.

4.2.3.2. Executing Aurora training script from Debugger Shell

You can execute the training script from the CodeWarrior **Debugger Shell**. Enter the following command in the **Debugger Shell** to source the training script:

1. In the shell, enter:

```
source "path/train_aurora_b4.tcl"
```

2. In the shell, execute the training script:

```
train_aurora_dbg
```

3. Look for a confirmation message that Aurora interface was successfully trained. If it was not successfully trained, execute the training script again.

4.2.3.3. Executing Aurora training script outside CodeWarrior

You can run the training script from `ccs` when processor is not already debugged by CodeWarrior. Open a `ccs` console, source the `train_aurora_b4.tcl` file, and then execute the training by running the script `train_aurora`.

1. Open `ccs` console.
2. In the console, enter:



```
% config cc gtap:jtag:1:ip_addr  
% config cc gtap:aurora:x2x2:ip_addr  
% ccs::config_chain core_name  
% source "path/train_aurora_b4.tcl"
```

```
% train_aurora
```

3. Look for a confirmation message that Aurora interface was successfully trained. If it was not successfully trained, execute the training script again.

5. Collecting Data

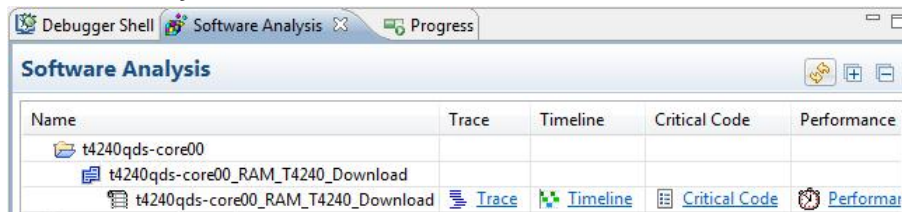
Once the project is created, a launch configuration and trace configuration is defined, and the Aurora interface is trained, you can start collecting data trace.

1. From the **Debug** view, select **Resume** , to start the data collection.
2. Let the application execute for few seconds, then select **Suspend** , to start collecting the trace data.

6. Viewing Trace Data

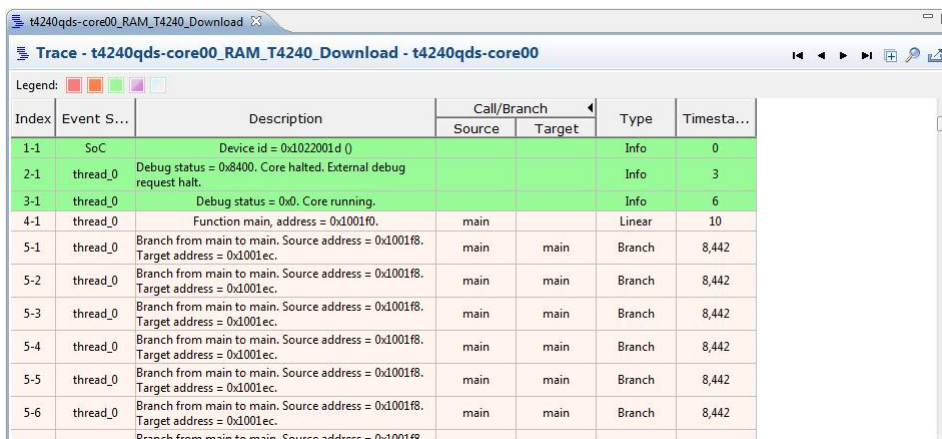
The **Software Analysis** view appears automatically if you have chosen to display the trace results automatically. To open the **Software Analysis** view, choose **Window > Show View > Software Analysis**.

Figure 8. Software Analysis view



From the **Software Analysis** view, expand the project and select the **Trace** link, and you can see the trace data in the **Trace** viewer.

Figure 9. Trace viewer



For more information about Trace viewer fields, see *CodeWarrior Development Studio for Power Architecture® Processors Version 10.x Tracing and Analysis Tools User Guide*, available at <CWInstallDir>\PA\Help\PDF folder.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or 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: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo, CodeWarrior, QorIQ, StarCore are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. QorIQ Qonverge 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.

© Freescale Semiconductor, Inc. 2014.