CodeWarrior Development Studio for Microcontrollers V11.x

Profiling and Analysis Quick Start

This Quick Start explains how to collect trace data after creating, building, and running a project on a target in the CodeWarrior for Microcontrollers version 11.x debugger. The document also explains how to view data collected in various viewers on the target hardware. This Quick Start applies to all targets that support tracing, such as HCS08, ColdFire V1-V4, Kinetis, S12z, and DSC.

NOTE In the procedures that follow, advanced users can use numbered steps. Novices may use the more detailed instructions provided by substeps.

Section A: Setting Up Device

Before collecting trace and critical data on the target hardware, make sure that the device on which you want to work is connected to the target board. For example, if you are working on the ColdFire V1 target with the MCF51QE128 device, you need to connect and set up MCF51QE128 device to the board. You can refer relevant documentation to understand how to set up the device for a particular target. You can find the required documentation at the Freescale website:
http://www.freescale.com/webapp/sps/site/homepage.jsp?code=PCMCR01&tid=FSH

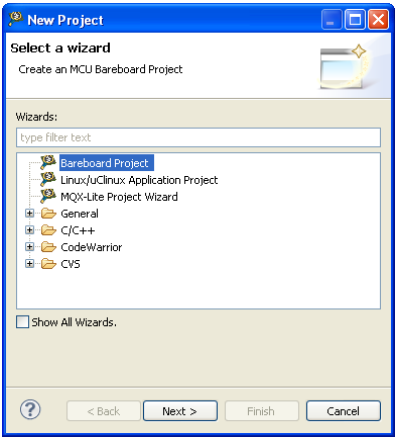
Section B: Collecting Data

1. Launch the CodeWarrior IDE
   a. Select **Start** > **Programs** > **Freescale CodeWarrior** > **CW for MCU v11.x** > **CodeWarrior** — the **Workspace Launcher** dialog box appears.
   b. Click **Browse** to specify the location where you want to store your project.
   c. Click **OK** — CodeWarrior launches.

2. Create a new project

   a. From the CodeWarrior IDE menu bar, select **File > New > Project** — the **New Project** dialog box appears.

New Project Dialog Box



   b. Select **Bareboard Project** and click **Next** — the **Create an MCU Bareboard Project** page appears.

NOTE You can also open the **Create an MCU Bareboard Project** page directly by selecting **File > New > Bareboard Project**.

   c. In the **Project name** field, type the name of your project.

   d. Click **Next** — the **Devices** page appears.

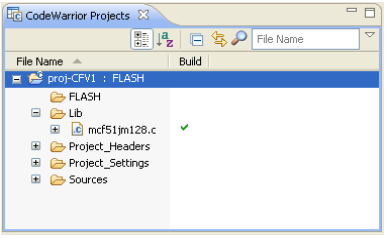
   e. Select the required device for your project from the list of available families.

   f. Click **Next** — the **Connections** page appears.

   g. Select the available connection, and follow the steps of the wizard.

   h. Click **Finish** — the project is created and appears in the **CodeWarrior Projects** view.

CodeWarrior Projects View



3. Build project
   a. Select the project in the **CodeWarrior Projects** view.
   b. Select **Project > Build Project** to build the project.
4. Configure launch configuration
   a. Right-click on the project in the **CodeWarrior Projects** view and select **Debug As > Debug Configurations** from the context menu — the **Debug Configurations** dialog box appears.
   b. Expand **CodeWarrior Download** in the tree structure on the left, and select the launch configuration corresponding to the project you are using. For example, select *<project_name>_FLASH_PnE U - MultiLink*.
   c. Click the **Trace and Profile** tab.
   d. Check the **Enable Trace and Profile** checkbox to enable the project for tracing.
   e. Select the required trace mode and trigger/tracepoints options for trace collection. The most frequent trace modes that you may use are:

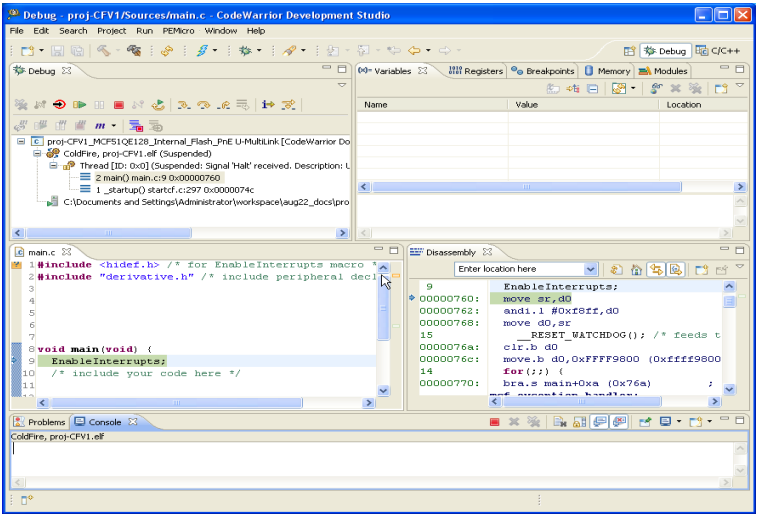
      – Automatic — Collects only the last buffer of trace. The buffer is circular; trace is collected without interruption, and when the data reaches the buffer end, the data from the beginning of the buffer gets overwritten.

      – Continuous — Collects trace data continuously as soon as the first start trigger/tracepoint is hit. The trace buffer is read, processed, and emptied periodically every time it fills.
   f. Select the **Continuous** option from the **Select Trace Mode** group.
   g. Click **Apply** to save the settings.

5. Debug project

   a. Click **Debug** — the **Debug** perspective appears and the execution halts at the first statement of `main()`.

Debug Perspective



6. Collect data

   a. In the **Debug** view, click **Resume** — the execution begins and data measurement starts.

   b. Let the application run for several seconds.

   c. In the **Debug** view, click **Suspend** — the execution stops.

Section C: Viewing Data

NOTE This section shows screens of trace data collected on the ColdFire V1 target as an example. All targets display the collected trace data in the same trace viewers.

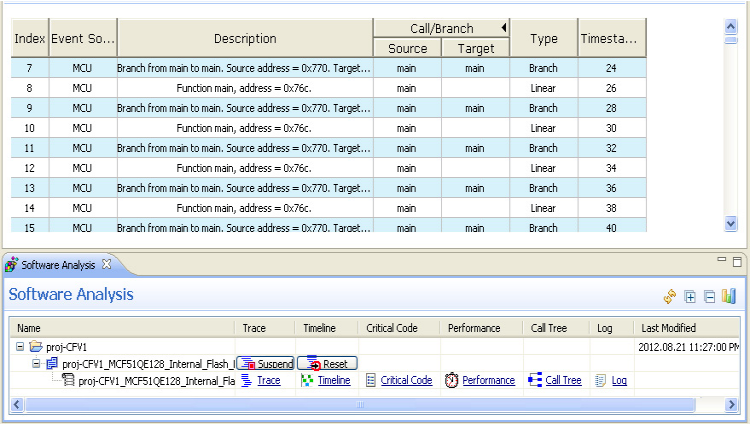
1. View trace data

   a. In the **Software Analysis** view that opens automatically after data collection, expand the data source — the **Trace, Timeline**, **Critical Code, Performance**, and **Call Tree** hyperlinks appear.

NOTE You can open the **Software Analysis** view manually by selecting **Window > Show View > Software Analysis** in the CodeWarrior IDE menu bar or by using the **Fast View** menu on the left side of the status bar.

   b. Click the **Trace** hyperlink — the **Trace Data** viewer appears displaying the trace data.

Trace Data Viewer and Software Analysis View



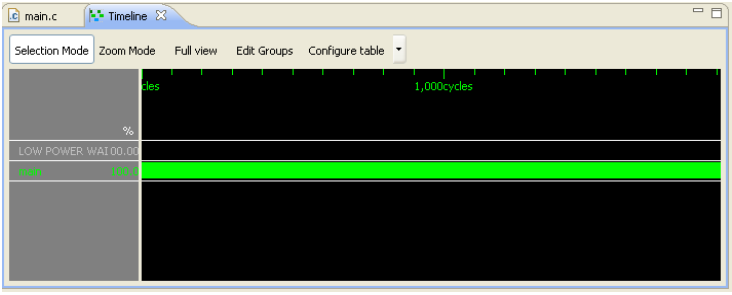
Index	Event So...	Description	Call/Branch		Type	Timesta...
			Source	Target		
7	MCU	Branch from main to main. Source address = 0x770. Target...	main	main	Branch	24
8	MCU	Function main, address = 0x76c.	main		Linear	26
9	MCU	Branch from main to main. Source address = 0x770. Target...	main	main	Branch	28
10	MCU	Function main, address = 0x76c.	main		Linear	30
11	MCU	Branch from main to main. Source address = 0x770. Target...	main	main	Branch	32
12	MCU	Function main, address = 0x76c.	main		Linear	34
13	MCU	Branch from main to main. Source address = 0x770. Target...	main	main	Branch	36
14	MCU	Function main, address = 0x76c.	main		Linear	38
15	MCU	Branch from main to main. Source address = 0x770. Target...	main	main	Branch	40

NOTE All targets except HCS08 generate trace data on the default stationery project. You can change the source code of `main.c` to generate desired trace data.

2. View timeline data

   a. Click the **Timeline** hyperlink — the **Timeline** viewer appears.

TimeLine Viewer

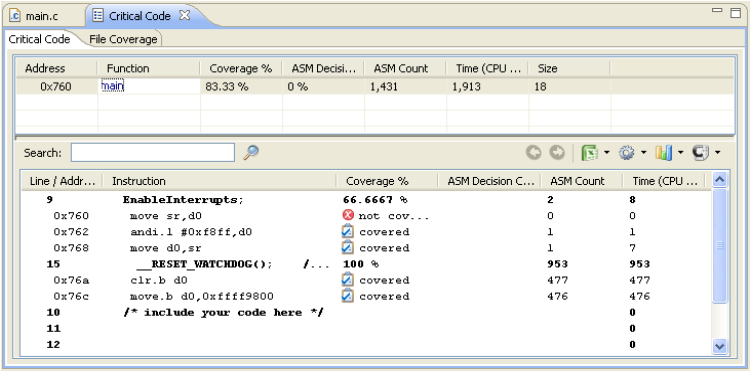


3. View critical code data

   a. Click the **Critical Code** hyperlink — the **Critical Code** viewer appears.

   b. Click a function in the **Function** column. For example, click the `main()` function, and view its statistics at the bottom of the **Critical Code** viewer.

Critical Code Viewer



4. View performance data

   a. Click the **Performance** hyperlink — the **Performance** viewer appears.

   b. Click a function in the **Function Name** column. For example, click the `main()` function, and view its statistics at the bottom of the **Performance** viewer.

Performance Viewer

main.c | Performance

Function Name	Num Calls	Inclusive	Min In...	Max Incl...	Avg Incl...	Percen...	Exclusive	Min Ex...	Max Excl...	Avg Excl...	Percen...
main	1	1,913	1,913	1,913	1,913	100	1,913	1,913	1,913	1,913	100

NOTE The **Performance** viewer also displays call pair relationships for a selected function in the bottom view. In the default stationery project, there is no call to the `main()` function from another function; also the `main()` function is not calling a function, so the data is empty in the *Call Site Performance* table and *Caller* and *Callee* graphs.

5. View call tree data

   a. Click the **Call Tree** hyperlink — the **Call Tree** viewer appears.

Call Tree

main.c | Call Tree

Function Name	Num Calls	% Total calls of parent	% Total times it was called	Inclusive Time (Cycles; 50.0 MHz)
<START>				
main	1	100.00	100.00	1,913

NOTE The ColdFire V1 target does not generate **Critical Code**, **Performance**, and **Call Tree** data in the **Automatic (One-buffer)** trace mode.

6. In the **Debug** view, click **Terminate** — the debug session ends.

7. Select **File > Exit** — the CodeWarrior IDE window closes.

NOTE For more information on various viewers of trace, refer *Profiling and Analysis Tools User Guide*.

Congratulations!

You have created, built, and debugged a Microcontrollers project using CodeWarrior and collected trace data successfully!

How to Contact Us

World Wide Web	http://www.nxp.com/codewarrior
Technical Support	http://www.nxp.com/support

14 July 2017