/arrior Development Studio
for eTPU v10.x
Quick Start

SYSTEM REQUIREMENTS

Hardware	Intel® Pentium® 4 processor, 2 GHz or faster, Intel® Xeon™, Intel® Core™, AMD Athlon™ 64, AMD Opteron™, or later 1 GB RAM CD-ROM drive for CD installation Microsoft Mouse compliant pointing device Internet connectivity for web downloads and update access
Operating System	Microsoft® Windows 7 (32/64-bit) Home Premium, Professional, Ultimate or Windows® XP Professional (SP3) (32/64-bit) Home, Professional
Software	Java 1.6 update 11 or newer
Disk Space	500 MB, additional space required during installation

This document explains how to install the CodeWarrior Development Studio for eTPU v10.x software and how to create, build, and debug a demo ETPU project.

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

Section A: Installing CodeWarrior Development Studio for eTPU v10.x Software

1. Install the software.
   a. Launch the Installer — the welcome page is displayed.
   b. Follow the wizard instructions to install the software to the desired location.

n software installation is finished, wizard displays the installation complete page.

d. In the installation complete page, you can clear the **Launch Application** checkbox to start the application later.

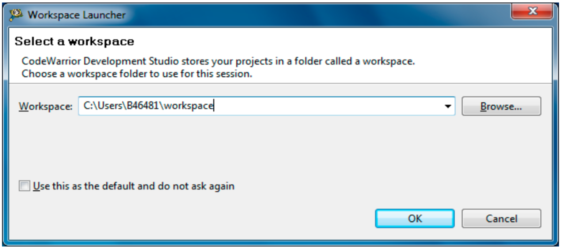
e. Click **Finish** to exit the installer.

Section B: Creating Project

1. Launch the CodeWarrior Development Studio for eTPU v10.x

a. For Windows, select **Start > Programs > Freescale CodeWarrior > CW for eTPU v10.x > CodeWarrior IDE** — the **Workspace Launcher** dialog box appears.

Workspace Launcher Dialog Box



Workspace Launcher

Select a workspace

CodeWarrior Development Studio stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

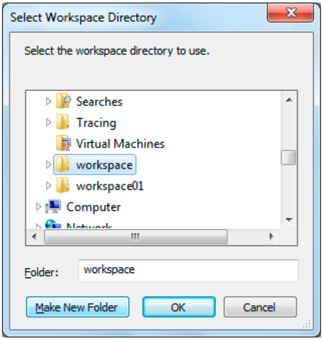
Workspace: C:\Users\B46481\workspace

□ Use this as the default and do not ask again

b. Click **Browse** to change the location of your project's Workspace.

Select Workspace Directory dialog box appears.

Select Workspace Directory Dialog Box



Select Workspace Directory

Select the workspace directory to use.

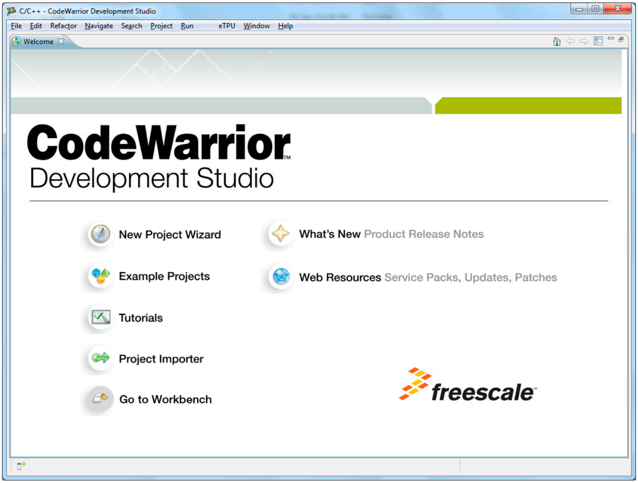
Folder: workspace

c. Select the required folder or click **Make New Folder** to create a new folder for storing your projects.

OK — the **Select Workspace Directory** dialog box closes.

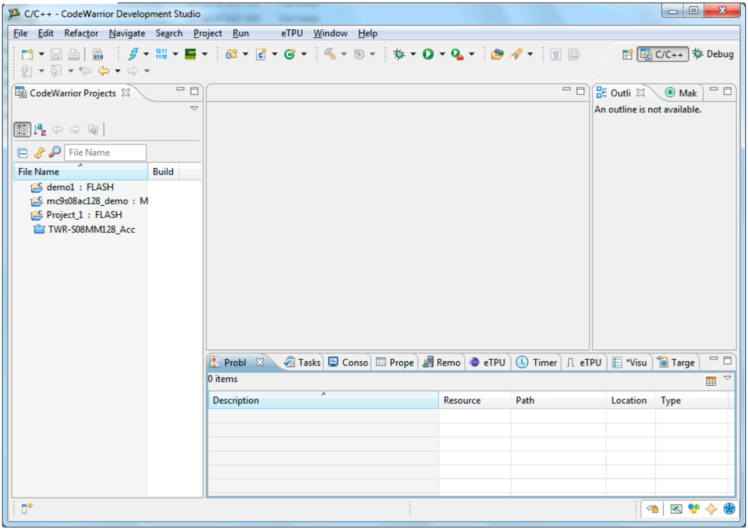
e. Click **OK** — the application launches and the **Welcome page** appears. Click **Help > Welcome** to view the welcome page at any time.

Welcome Page


C/C++ - CodeWarrior Development Studio
File Edit Refactor Navigate Search Project Run eTPU Window Help
Welcome
CodeWarrior Development Studio
New Project Wizard
What's New Product Release Notes
Example Projects
Web Resources Service Packs, Updates, Patches
Tutorials
Project Importer
Go to Workbench
freescale

f. In the **Welcome** page, click **Go to Workbench** — the Workbench window appears.

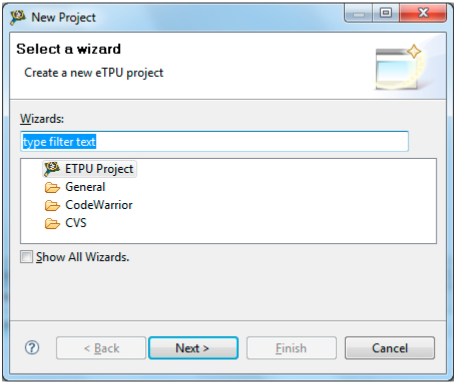
Workbench Window


C/C++ - CodeWarrior Development Studio
File Edit Refactor Navigate Search Project Run eTPU Window Help
C/C++ Debug
CodeWarrior Projects
File Name
File Name Build
demo1 : FLASH
mc9s08ac128_demo : M
Project_1 : FLASH
TWR-S08MM128_Acc
Outli Mak
An outline is not available.
Probl Tasks Conso Prope Remo eTPU Timer eTPU *Visu Targe
0 items
Description Resource Path Location Type

2. Create a new project

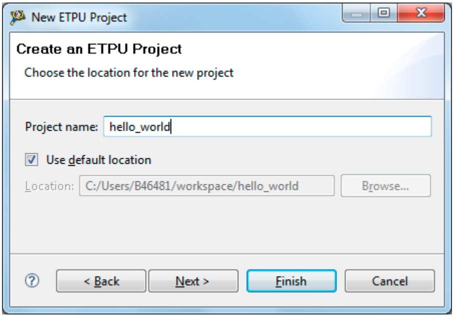
   a. From the IDE menu bar, select **File > New > Project** — the **New Project** wizard starts; the **Select a wizard** page appears.

Select a wizard Page



   b. Select **ETPU Project** and click **Next** — the **New ETPU Project** wizard appears and displays **Create an ETPU Project** page.

Create an ETPU Project Page



New ETPU Project

Create an ETPU Project

Choose the location for the new project

Project name: hello_world

☑ Use default location

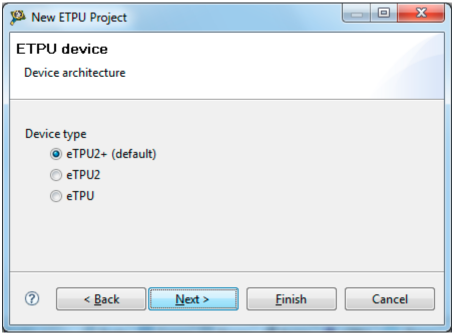
Location: C:/Users/B46481/workspace/hello_world

c. Enter `hello_world` in the **Project name** field.

NOTE The **Location** field shows the default project location. If you do not want to use the default location, clear the **Use default location** checkbox. In the **Location** text box, enter the full path of the directory in which you want to create your project including the project name.
Alternatively, click **Browse** and select the desired location from the **Browse For Folder** dialog box and click **OK**. Ensure that you append the path with the name of the project to create a new location for your project.

d. Click **Next** — the **ETPU device** page appears.

ETPU Device Page



New ETPU Project

ETPU device

Device architecture

Device type

● eTPU2+ (default)

○ eTPU2

○ eTPU

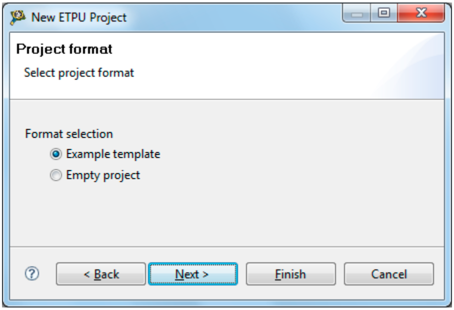
e. Select **eTPU2+** — the wizard configures the project settings for eTPU2+ architecture.

…e target architecture can be changed anytime. To change the target architecture, right-click the project and select **Properties**.
- Expand **C/C++ Build** and select **Settings**.
- In the **Tool Settings** tab, expand **ETPU Compiler** and select **Compiler Settings**.
- Change the architecture from the **Architecture** drop-down list.
- Clean and Recompile the project.

To use the simulator in ETPU mode, you have to change the Target Processor also. You can change the target processor setting from **Run > Debug configurations > CodeWarrior Download > hello_world_ETPU2P configuration_ETPU2P > Debugger**.

f. Click **Next** — **Project format** page appears.

Project format Page



New ETPU Project

Project format

Select project format

Format selection

● Example template

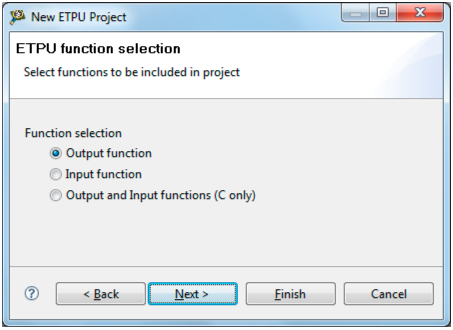
○ Empty project

g. Select the project fromat you want, default is **Example template**.

NOTE The empty project contains the minimum code, requirements to build, debug simulate an etpu/etpu2/etpu2+ project.

h. Click **Next** - **ETPU function selection** page appears.

ETPU function selection Page



New ETPU Project

ETPU function selection

Select functions to be included in project

Function selection

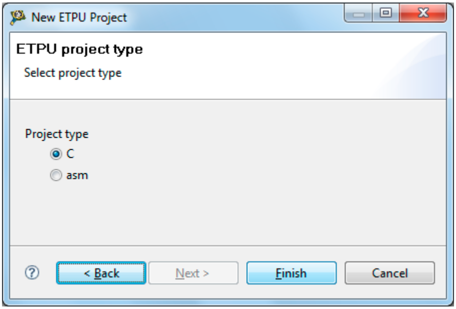
● Output function
○ Input function
○ Output and Input functions (C only)

i. Select the function you want, default is **Output function**.

NOTE The **Output and input functions (C only)** is a more complex project including both functions, and is available only in C language.

j. Click **Next** — **ETPU project type** page appears.

ETPU project type Page



New ETPU Project

ETPU project type

Select project type

Project type

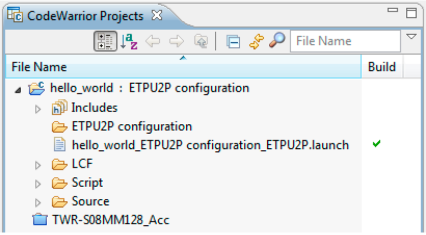
● C
○ asm

k. Select the project type you want, default is **C** project type.

NOTE The assembler project is a simplified version of the C project type.

l. Click **Finish** — IDE creates and builds the project; you can view the newly created project in the **CodeWarrior Projects** view in the IDE.

CodeWarrior Projects View



3. Expand the project to view all the project files.

4. Open **Example Projects** page from the **Welcome** screen for instructions about how to import *set1* project.

Section C: Debugging Project

1. Enter debug mode

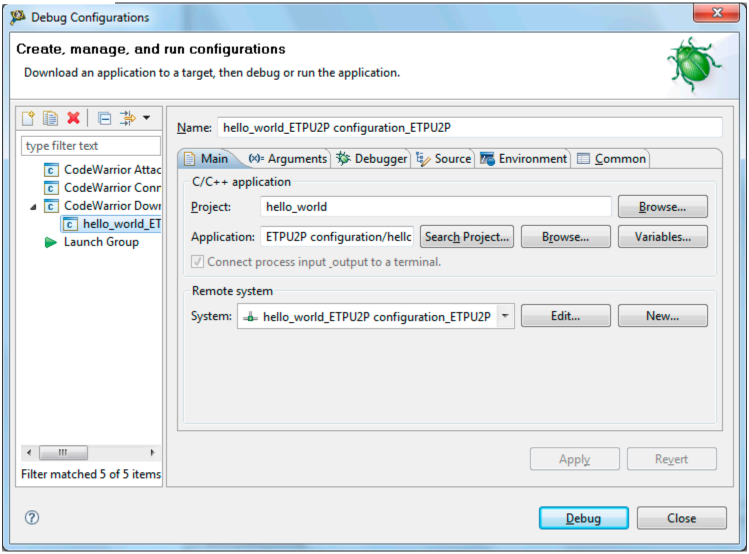
NOTE To debug, select a project from the **CodeWarrior Projects** view and click **Debug** . Alternatively, you can perform the following steps.

a. Select the desired project in the **CodeWarrior Projects** view.

b. Select **Project > Build Project** to build the project. Alternatively, right-click on the project and select **Build Project** from the context menu.

c. From the IDE menu bar, select **Run > Debug Configurations** — the **Debug Configurations** dialog box appears.

Debug Configurations



Debug Configurations

Create, manage, and run configurations

Download an application to a target, then debug or run the application.

type filter text

- CodeWarrior Attac
- CodeWarrior Conr
- CodeWarrior Dowr
  - hello_world_ET
- Launch Group

Filter matched 5 of 5 items

Name: hello_world_ETPU2P configuration_ETPU2P

Main | Arguments | Debugger | Source | Environment | Common

C/C++ application

Project: hello_world

Application: ETPU2P configuration/hellc

☑ Connect process input _output to a terminal.

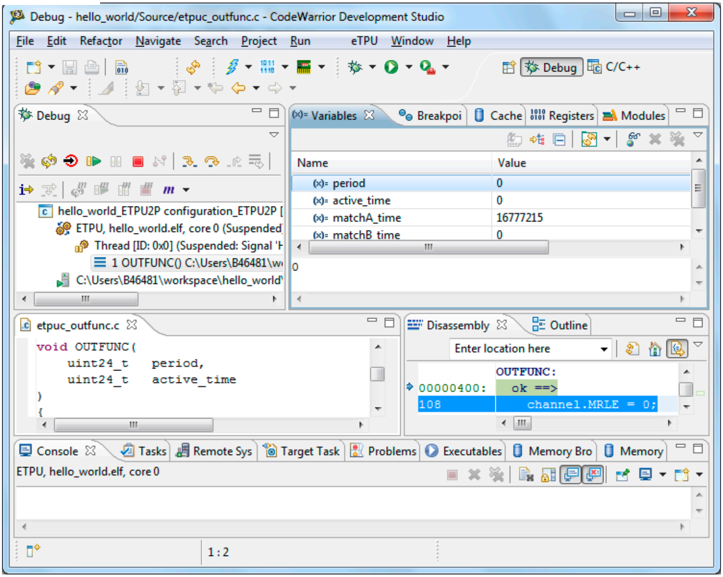
Remote system

System: hello_world_ETPU2P configuration_ETPU2P

d. Expand the **CodeWarrior Download** tree control in the left pane and select **hello_world_ETPU2 configuration_ETPU2P** — A set of tabbed configuration panels appear in the right pane of the dialog box.

e. Click **Debug** — program downloads to the simulator and the **Confirm Perspective Switch** dialog box appears and the execution halts.

Debug Perspective


Debug - hello_world/Source/etpuc_outfunc.c - CodeWarrior Development Studio
File Edit Refactor Navigate Search Project Run eTPU Window Help
Debug
C/C++
Debug
hello_world_ETPU2P configuration_ETPU2P [
ETPU, hello_world.elf, core 0 (Suspended
Thread [ID: 0x0] (Suspended: Signal 'H
1 OUTFUNC() C:\Users\B46481\w
C:\Users\B46481\workspace\hello_world\
Variables
Breakpoi
Cache
Registers
Modules
etpuc_outfunc.c
Disassembly
Outline
Enter location here
Console
Tasks
Remote Sys
Target Task
Problems
Executables
Memory Bro
Memory
ETPU, hello_world.elf, core 0
1:2

Name	Value
period	0
active_time	0
matchA_time	16777215
matchB_time	0

```
void OUTFUNC(
      uint24_t    period,
      uint24_t    active_time
)
{
```

```
           OUTFUNC:
00000400:    ok  ==>
108            channel.MRLE = 0;
```

NOTE When the debug session starts:

A new icon appears in the system tray -`ccssim2.exe`, this program is used to simulate the eTPU processor. Right-click on it and you can see the **ccssim2 menu**: **Show console**, **Hide console**, **About ccssim2** and **Quit**.

NOTE To debug more than one project simultaneously, you will need to configure the CCS server port number and the visualization bind port manually such that each project will have its own distinct ports. To configure the CCS server port, edit the Remote system configuration in Main tab of the Debug Configurations dialog box. To configure vizualization bind port, edit the ETPU simulator options in the Debugger tab of the Debug Configurations dialog box.

NOTE eTPU processing is event-driven, in the sense that eTPU microcode only runs to service a request from an event. For more information, refer to the chapter *Functions and Threads* of the *eTPU Reference Manual*.

The projects generated by the new project wizard contains an init.tcl script that is responsible for the initialization of the simulator. This initialization script is customized such that the initialization thread of the configured functions with corresponding parameters also start when the project debug initiates.

The initialization TCL script is automatically executed when entering debug and the result of the execution can be observed in **Debugger Shell** view. The TCL script allows among other things the change of parameters at the run-time. For more information, refer to the step Configure Visualization plug-in.

Modifying the script or C code might result in an uninitialized simulator on which debug operations will not work. When changing the init.tcl files or files included by the simulator ensure that no errors are shown in the **Debugger Shell** view.

NOTE To debug more than one project simultaneously, the CCS server port number from remote system configuration. You can configure the remote system from the **Main** tab of the **Debug Configuration** dialog box. The visualization bind port from simulator options (Debugger panel from Launch configuration) need to be manually changed so that each project will have its own distinct ports.

Also when two projects are in debug simultaneously the script command from debugger shell are executed on default connection. This can be observed and changed with the help of switchtarget command

f. Click on the thread in the **Debug** view — the program counter icon ➡ on the marker bar points to the next statement to be executed.

g. In the **Debug** view, click **Step Over** ↷ — the debugger executes the current statement and halts at the next statement.

kpoint and execute program to breakpoint.

a. In the editor area, double-click on the marker bar next to the statement — the breakpoint indicator (blue dot) appears next to the statement.

b. From the **Debug** view, click Resume ▶ — the debugger executes all statements up to but not including the breakpoint statement.

3. Control the execution of the program

a. From the **Debug** view, click **Step Over** ↷ — the debugger executes the breakpoint statement and halts at the next statement.

b. From the **Debug** view, click **Resume** ▶ — the program execution resumes.

c. From the **Debug** view, click **Terminate** ■ - the debug session ends.

4. Visualize the eTPU signals

a. Select **eTPU** > **New Signal Viewer** — the **eTPU Signal** view appears.

b. Click the A⌐ button and move the mouse over the signals to use the signal color indication fetaure.
All the signals that have events in the corresponding selection will appear along with their names.

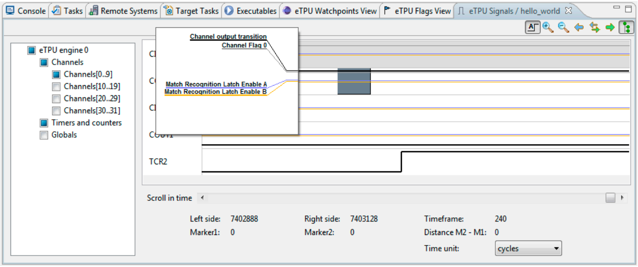
NOTE The visualization plug-in can also be used in debug mode. The signal view does not display spike signals (MRLA, MRLB). They are shown in **Execution Trace View**.

Scenarios with multiple projects per workspace were not tested and might result in unpredictable behavior.

The visualization plug-in stores data in a circular buffer. When the buffer is full, the oldest events are overwritten. By default, the buffer size is 10000. See **Logging options** below for options.

Please note that for this release TCR1 is by default disabled. You can enable it from **Logging options**.

eTPU Signal View



Left side: 7402888 Right side: 7403128 Timeframe: 240

Marker1: 0 Marker2: 0 Distance M2 - M1: 0

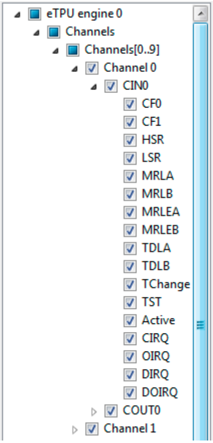
Time unit: cycles

c. Customize view to your needs:

- Select a signal in signal tree to add the signal at the bottom of the view and deselect a signal to remove the signal from the view.

- Expand the tree to view and select the overlapped signals of the channels.

Overlapped signals enabled by default



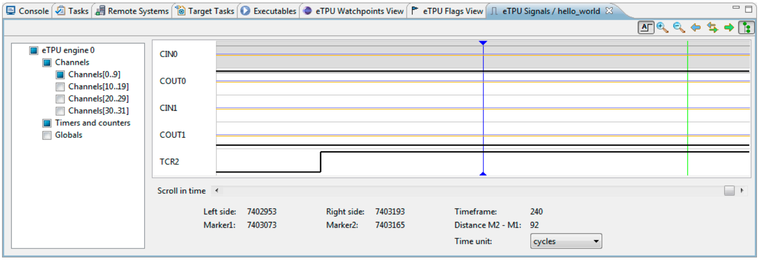
- eTPU engine 0
  - Channels
    - Channels[0..9]
      - ☑ Channel 0
        - ☑ CIN0
          - ☑ CF0
          - ☑ CF1
          - ☑ HSR
          - ☑ LSR
          - ☑ MRLA
          - ☑ MRLB
          - ☑ MRLEA
          - ☑ MRLEB
          - ☑ TDLA
          - ☑ TDLB
          - ☑ TChange
          - ☑ TST
          - ☑ Active
          - ☑ CIRQ
          - ☑ OIRQ
          - ☑ DIRQ
          - ☑ DOIRQ
        - ☑ COUT0
      - ☑ Channel 1

- After selecting the signals, you can hide the signal tree using **Show Signal Tree** button from the toolbar menu .

d. Zoom in and zoom out: .

e. There are two time markers, *blue* and *green*, which can be used for measurement purpose. Click on signal charts to set a time marker at that corresponding timestamp, by default the closest marker near the mouse ved to the mouse position. The time markers can also be dragged to another timestamp using the mouse.

Time Markers



- ■ eTPU engine 0
  - ■ Channels
    - ■ Channels[0..9]
    - □ Channels[10..19]
    - □ Channels[20..29]
    - □ Channels[30..31]
  - ■ Timers and counters
  - □ Globals

Left side: 7402953 Right side: 7403193 Timeframe: 240
Marker1: 7403073 Marker2: 7403165 Distance M2 - M1: 92
Time unit: cycles

f. Go to left marker using ⇦ button or to right marker using ⇨ button. Double-click the marker value to edit the timestamps of the markers.

g. Zoom Markers ⇆ , make the area between markers to fill the view.

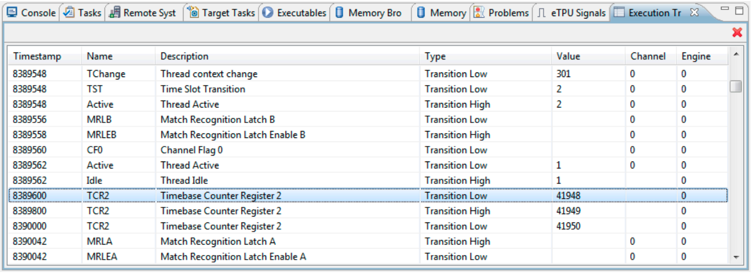
h. Use up and down arrow keys to change the active waveform selection. The background of the active waveform changes to grey. In the above image, CIN0 is the active waveform.

i. Use left and right arrow keys to snap the selected marker to the nearest transition in the active waveform. In the above image, the blue marker is selected. The markers selection can be changed using *Ctrl+Left arrow* to select left marker, and *Ctrl+Right arrow* for right marker.

5. Visualize the execution trace

a. Select **eTPU > Execution Trace** — the **Execution Trace** view appears.

Execution Trace View



Timestamp	Name	Description	Type	Value	Channel	Engine
8389548	TChange	Thread context change	Transition Low	301	0	0
8389548	TST	Time Slot Transition	Transition Low	2	0	0
8389548	Active	Thread Active	Transition High	2	0	0
8389556	MRLB	Match Recognition Latch B	Transition Low		0	0
8389558	MRLEB	Match Recognition Latch Enable B	Transition High		0	0
8389560	CF0	Channel Flag 0	Transition Low		0	0
8389562	Active	Thread Active	Transition Low	1	0	0
8389562	Idle	Thread Idle	Transition High	1		0
8389600	TCR2	Timebase Counter Register 2	Transition Low	41948		0
8389800	TCR2	Timebase Counter Register 2	Transition High	41949		0
8390000	TCR2	Timebase Counter Register 2	Transition Low	41950		0
8390042	MRLA	Match Recognition Latch A	Transition High		0	0
8390042	MRLEA	Match Recognition Latch Enable A	Transition Low		0	0

b. Select an event in execution trace view to put a *red* marker in signal view at the corresponding timestamp.

Event selection can also be changed using Up/Down arrow keys or Page Up/Page Down keys.

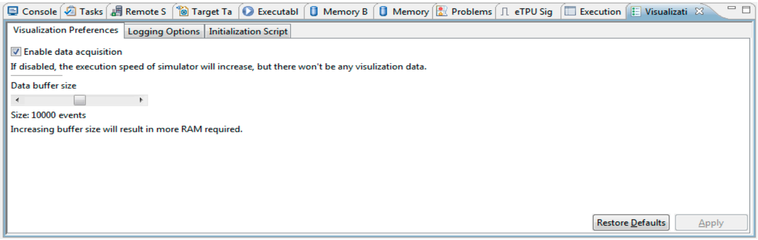
NOTE To be able to view both execution trace and signal view you can move a view by dragging the title into another area, or you can right-click on view's title and select **Detached**.

The **Execution Trace** View uses 255 to mark an event that is not related to a channel.

6. Configure Visualization plug-in

   a. Select **eTPU > Options** — the **Visualization Properties** view appears.

Visualization Properties View



☑ Enable data acquisition

If disabled, the execution speed of simulator will increase, but there won't be any visulization data.

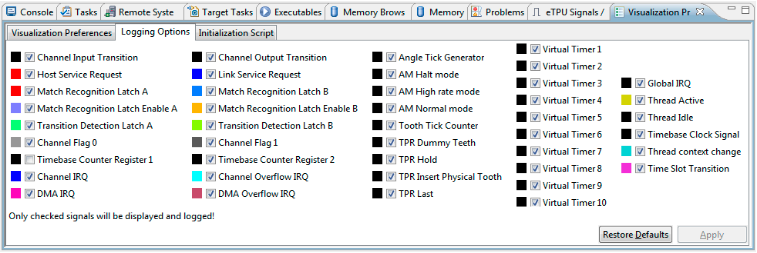
Data buffer size

Size: 10000 events

Increasing buffer size will result in more RAM required.

   b. In **Logging Options** panel you can select the desired signals to be logged and visualized.

Logging Options View



⬛ ☑ Channel Input Transition
🟥 ☑ Host Service Request
🟥 ☑ Match Recognition Latch A
🟪 ☑ Match Recognition Latch Enable A
🟩 ☑ Transition Detection Latch A
▨ ☑ Channel Flag 0
⬛ □ Timebase Counter Register 1
🟦 ☑ Channel IRQ
🟪 ☑ DMA IRQ

⬛ ☑ Channel Output Transition
🟦 ☑ Link Service Request
🟦 ☑ Match Recognition Latch B
🟧 ☑ Match Recognition Latch Enable B
🟩 ☑ Transition Detection Latch B
▨ ☑ Channel Flag 1
⬛ ☑ Timebase Counter Register 2
🟦 ☑ Channel Overflow IRQ
🟥 ☑ DMA Overflow IRQ

⬛ ☑ Angle Tick Generator
⬛ ☑ AM Halt mode
⬛ ☑ AM High rate mode
⬛ ☑ AM Normal mode
⬛ ☑ Tooth Tick Counter
⬛ ☑ TPR Dummy Teeth
⬛ ☑ TPR Hold
⬛ ☑ TPR Insert Physical Tooth
⬛ ☑ TPR Last

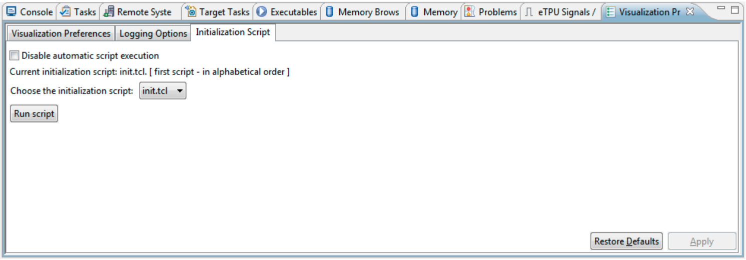
⬛ ☑ Virtual Timer 1
⬛ ☑ Virtual Timer 2
⬛ ☑ Virtual Timer 3
⬛ ☑ Virtual Timer 4
⬛ ☑ Virtual Timer 5
⬛ ☑ Virtual Timer 6
⬛ ☑ Virtual Timer 7
⬛ ☑ Virtual Timer 8
⬛ ☑ Virtual Timer 9
⬛ ☑ Virtual Timer 10

⬛ ☑ Global IRQ
🟨 ☑ Thread Active
⬛ ☑ Thread Idle
⬛ ☑ Timebase Clock Signal
🟦 ☑ Thread context change
🟪 ☑ Time Slot Transition

Only checked signals will be displayed and logged!

   c. In the **Initialization Script** panel you can disable the automatic script execution, and can change the initialization script that is executed when entering debug. By default the first (alphabetical) `.tcl` file found in the `Script` folder of the project is executed.

Initialization Script View



□ Disable automatic script execution

Current initialization script: init.tcl. [first script - in alphabetical order]

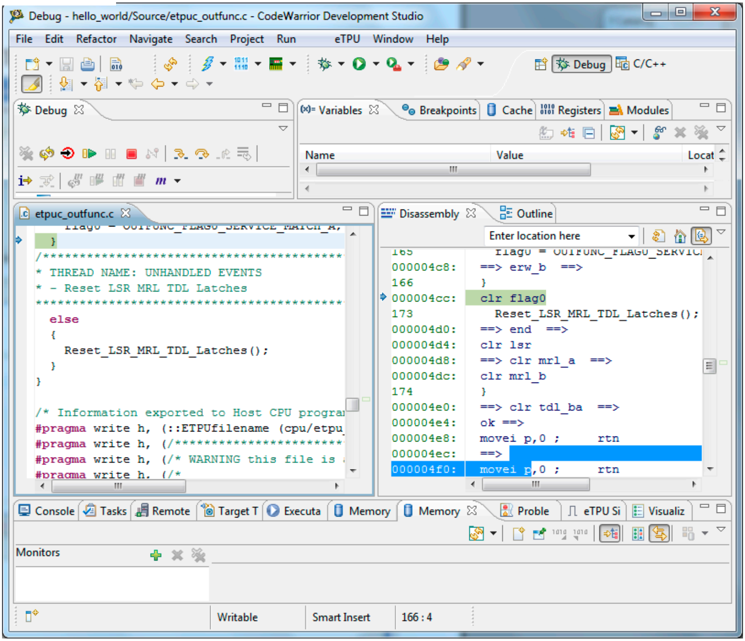
Choose the initialization script: init.tcl

NOTE Click **Apply** to save the changes.

Disabling events in **Logging Options** view will not delete the events already logged.

7. Virtual timers
   a. Virtual timers allow the measurement of the execution time in the system cycles.
   b. Right-click on the left editor vertical ruler to configure the start and stop points.

Virtual Timers Window



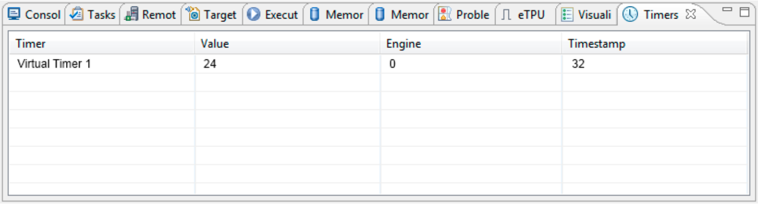
```
   flag0 = OUTFUNC_FLAG0_SERVICE_MATCH_A;
 }
/*********************************************
* THREAD NAME: UNHANDLED EVENTS
* - Reset LSR MRL TDL Latches
*********************************************
  else
  {
    Reset_LSR_MRL_TDL_Latches();
  }
}

/* Information exported to Host CPU prograi
#pragma write h, (::ETPUfilename (cpu/etpu_
#pragma write h, (/***********************
#pragma write h, (/* WARNING this file is :
#pragma write h, (/*
```

```
165             flag0 = OUTFUNC_FLAG0_SERVIC
000004c8:     ==> erw_b  ==>
166           }
000004cc:     clr flag0
173             Reset_LSR_MRL_TDL_Latches();
000004d0:     ==> end  ==>
000004d4:     clr lsr
000004d8:     ==> clr mrl_a  ==>
000004dc:     clr mrl_b
174           }
000004e0:     ==> clr tdl_ba  ==>
000004e4:     ok ==>
000004e8:     movei p,0 ;      rtn
000004ec:     ==>
000004f0:     movei p,0 ;      rtn
```

c. Open the Timers View from **eTPU > eTPU Analysis > Timers**.

Timers View



Timer	Value	Engine	Timestamp
Virtual Timer 1	24	0	32

d. Run the code, the value filed from the **Timers** view will be updated with the most recent value.

e. You can reconfigure the start and stop points from right click menu.

f. Alternatively, the virtual timer’s data can be observed in the execution trace and in the signal view in the **Timers and Counters** group.

... this version only one timer can be used from eclipse. To use all available virtual timers, read the simulator manual.

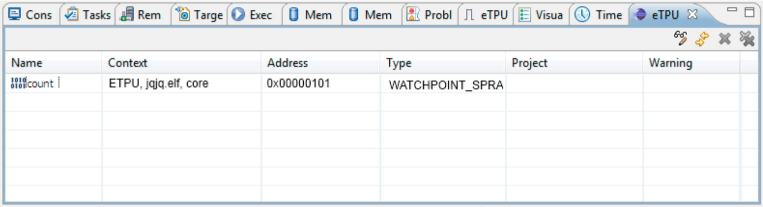
8. Watchpoints

   a. Watchpoints allow execution break when the watchpoint condition is met.

   b. Follow the steps to set a watchpoint on a variable:

      - Select **eTPU > eTPU Analysis > Watchpoints** from the IDE menu bar to display the Watchpoints View.

Watchpoints View



Name	Context	Address	Type	Project	Warning
count	ETPU, jqjq.elf, core	0x00000101	WATCHPOINT_SPRA		

      - Select **New watchpoint**.

      - Write the variable name and select the access type than click **OK**.

      - The execution breaks when the variable is written or read, depending on the selected access type.

      - The variable that causes the break is highlighted yellow in the watchpoints view.

   c. To delete a single watchpoint, select the desired watchpoint in the Watchpoints View and click the Delete selected watchpoint button ✖.

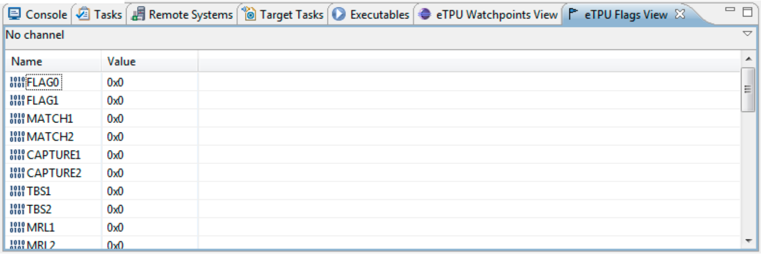
   d. To delete all watchpoints, click the Delete All button ✖.

9. Flags View

   a. Flags view window allows the visualization of all channels flags during execution.

   b. Select **eTPU > eTPU Analysis > Flags Viewer** from the IDE menu bar to display the Flags View.

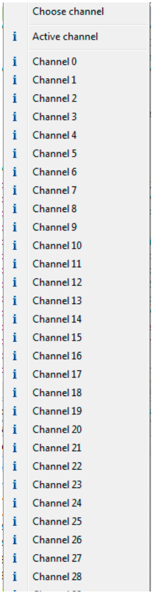
Flags View



Name	Value
FLAG0	0x0
FLAG1	0x0
MATCH1	0x0
MATCH2	0x0
CAPTURE1	0x0
CAPTURE2	0x0
TBS1	0x0
TBS2	0x0
MRL1	0x0
MRL2	0x0

c. Select the channel to watch or the **Active Channel** from the drop-down menu from the right corner of the view.

Select Channel

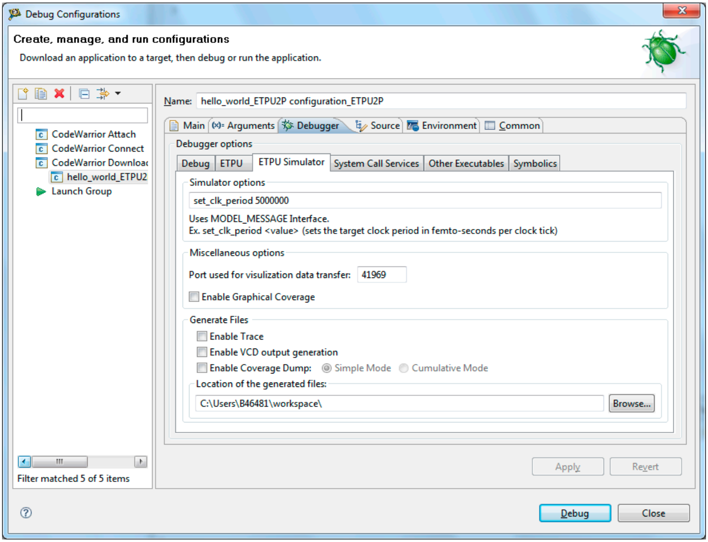


d. A list of updated values of all the flags is available in the view.

r Options

a. Open **Run > Debug Configurations > CodeWarrior Download > hello_world_ETPU2P configuration_ETPU2P > Debugger > ETPU Simulator**.

ETPU Simulator options



Debug Configurations

Create, manage, and run configurations

Download an application to a target, then debug or run the application.

CodeWarrior Attach
CodeWarrior Connect
CodeWarrior Downloac
hello_world_ETPU2
Launch Group

Filter matched 5 of 5 items

Name: hello_world_ETPU2P configuration_ETPU2P

Main | Arguments | Debugger | Source | Environment | Common

Debugger options

Debug | ETPU | ETPU Simulator | System Call Services | Other Executables | Symbolics

Simulator options

set_clk_period 5000000

Uses MODEL_MESSAGE Interface.
Ex. set_clk_period <value> (sets the target clock period in femto-seconds per clock tick)

Miscellaneous options

Port used for visulization data transfer: 41969

□ Enable Graphical Coverage

Generate Files

□ Enable Trace
□ Enable VCD output generation
□ Enable Coverage Dump: ● Simple Mode ○ Cumulative Mode

Location of the generated files:

C:\Users\B46481\workspace\ | Browse...

Apply | Revert

Debug | Close

b. From this page, you can configure trace, coverage dump and VCD output.These features are described in the simulator manual.

11. Graphical Coverage

a. Can be enabled from **Run > Debug Configurations > CodeWarrior Download > hello_world_ETPU2P configuration_ETPU2P > Debugger > ETPU Simulator**.

b. If enabled, the trace coverage information is displayed in the left editor vertical ruler.

NOTE This feature is subject to changes.

12. Save and Load History.

a. Select **eTPU > Save History** to save the content of the data buffer.

b. Select **eTPU > Load History** to load the saved data in buffer.

n application

a. Select **File > Exit** — the application window closes.

Congratulations!

You have created, built, and debugged an ETPU project using CodeWarrior Development Studio for eTPU v10.x!

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 Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor 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 Semiconductor 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 Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

How to Contact Us

Corporate Headquarters	Freescale Semiconductor, Inc. 6501 William Cannon Drive West Austin, Texas 78735 U.S.A.
World Wide Web	`http://www.freescale.com/codewarrior`
Technical Support	`http://www.freescale.com/support`

Revised: 18 September 2013