



CodeWarrior™ Development Studio

for StarCore™ and SDMA Architectures

Quick Start for Windows® Operating Systems and Embedded Cross Trigger

This Quick Start explains how to set up a sample project to use the Embedded Cross Trigger (ECT) configurator. The configurator sets up trigger signals among the StarCore, SmartDMA™ (SDMA), and ARM® cores. First, you learn to use the ECT with two cores. Then, you learn to use the ECT with three cores:

Section A: Using the ECT with Two Cores

In this section, you create and save an ECT matrix. You use this matrix to define a trigger that stops the ARM core when the StarCore part enters debug mode.

1. Open sample StarCore project

- Select **Start > Programs > Freescale CodeWarrior > CodeWarrior for StarCore and SDMA V1.0 > CodeWarrior IDE** from Windows task bar— IDE starts; CodeWarrior main window appears
- From CodeWarrior menu bar, select **File > Open** — **Open** dialog box appears
- Use dialog box to find and open ECT example directory at this location, where *CodeWarrior* is path to your CodeWarrior installation:

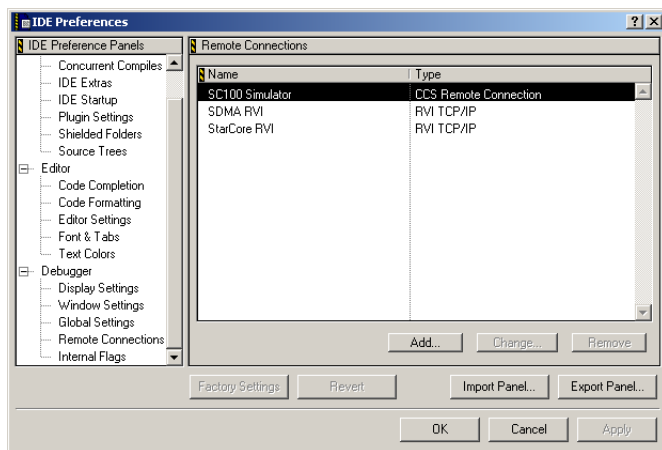
```
CodeWarrior\ (CodeWarrior_Examples) \ECT_Examples\
StarCore
```

- Select **StarCore.mcp** file
- Click **Open** button — system opens project; project window appears, docked at left side of main window

NOTE As an example, this Quick Start shows how to create a remote connection for MXC-05 hardware. If you have different hardware, use these steps as a guide to create an appropriate remote connection.

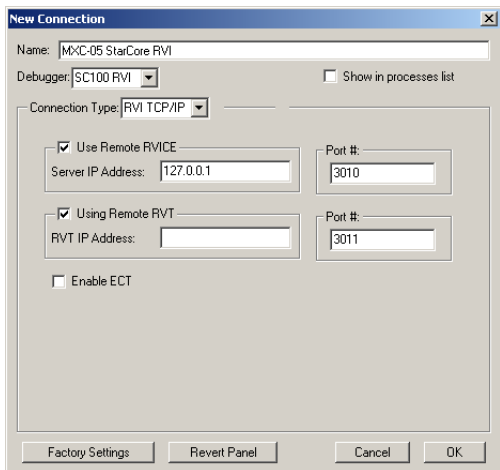
- Select **Edit > Preferences** from CodeWarrior menu bar — **IDE Preferences** window appears
- Select **Remote Connections** from **IDE Preference Panels** list on left side of window — panel appears in window

IDE Preferences Window — Remote Connections



Add button — **New Connection** dialog box appears

New Connection Dialog Box

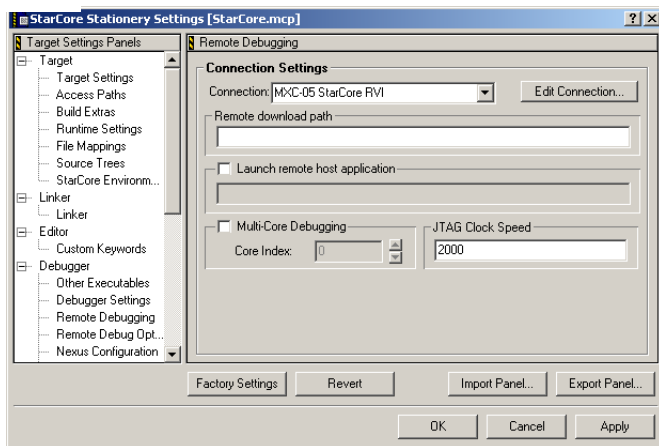


- d. In **Name** text box, enter **MXC-05 StarCore RVI**
- e. Use **Debugger** list box to specify **SC100 RVI**
- f. In **Server IP Address** text box, enter IP address of RVIC server
- g. Check **Enable ECT** checkbox
- h. Click **OK** button — IDE saves new remote-connection configuration; name of new remote connection appears in panel list
- i. Click **OK** button — IDE Preferences window closes

3. Configure remote debugging for StarCore project

- a. Select **Edit > StarCore Stationary Settings** from CodeWarrior menu bar — Target settings window appears
- b. Select **Remote Debugging** from **Target Settings Panels** list on left side of window — panel appears in window

Target Settings Window — Remote Debugging

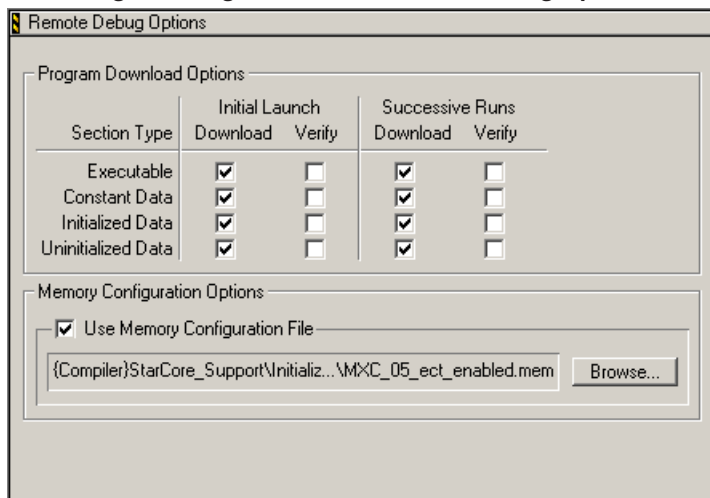


- c. Use **Connection** list box to specify remote connection that you created in step 2.

4. Configure remote-debugging options for project

- a. Select **Remote Debug Options** from **Target Settings Panels** list on left side of window — panel appears in window

Target Settings Window — Remote Debug Options





Click **Use Memory Configuration File** checkbox

- c. Click **Browse** button — **Choose the Debugger Memory Configuration File** dialog box appears
- d. Use dialog box to navigate to

`CodeWarrior\StarCore_Support\Initialization_Files\MemoryConfigFiles`

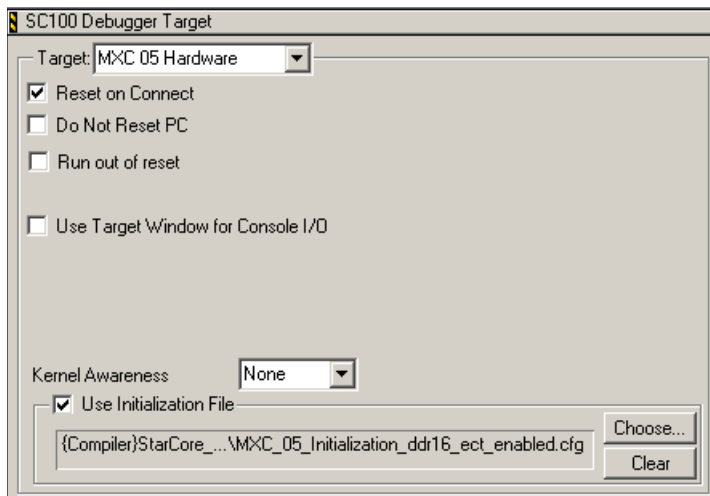
where *CodeWarrior* is path to your CodeWarrior installation

- e. Use dialog box to select appropriate memory-configuration file for your hardware:
 - `i.300-30_Memory_Example.mem`
 - `MXC_05_ect_enabled.mem`
 - `MXC275-30_Memory_Example_ect_enabled.mem`
- f. Use **Path Type** list box to specify **Compiler Relative**
- g. Click **Open** button — path to selected memory-configuration file now appears to left of **Browse** button

5. Configure SC100 debugger target for project

- a. Select **SC100 Debugger Target** from **Target Settings Panels** list on left side of window — panel appears in window

Target Settings Window — SC100 Debugger Target





Target list box to specify appropriate target for your hardware

- c. Check **Use Initialization File** checkbox
- d. Click **Choose** button — **Choose the StarCore Register Init File** dialog box appears
- e. Use dialog box to navigate to

`CodeWarrior\StarCore_Support\Initialization_Files\RegisterConfigFiles`

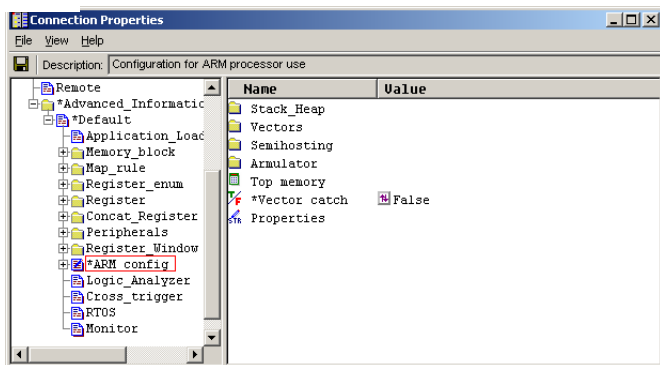
where *CodeWarrior* is path to your CodeWarrior installation

- f. Use dialog box to select appropriate register-initialization file for your hardware:
 - `i.300-30\i.300-30_Initialization_ect_enabled.cfg`
 - `MXC_05\MXC_05_Initialization_dds16_ect_enabled.cfg`
 - `MXC275-30\MXC275-30_Initialization_ect_enabled.cfg`
- g. Use **Path Type** list box to specify **Compiler Relative**
- h. Click **Open** button — path to selected initialization file now appears to left of **Choose** button
- i. Click **OK** button — IDE saves changes; Target Settings window closes

6. Configure RealView® Debugger

- a. Select **Start > Programs > ARM > RealView Developer Suite vx.x > RealView Debugger vx.x** from Windows task bar (where **x.x** represents version number) – RealView Debugger Code window appears
- b. Select **Target > Connection Properties** from RealView Debugger main menu bar – **Connection Properties** window appears

Connection Properties Window

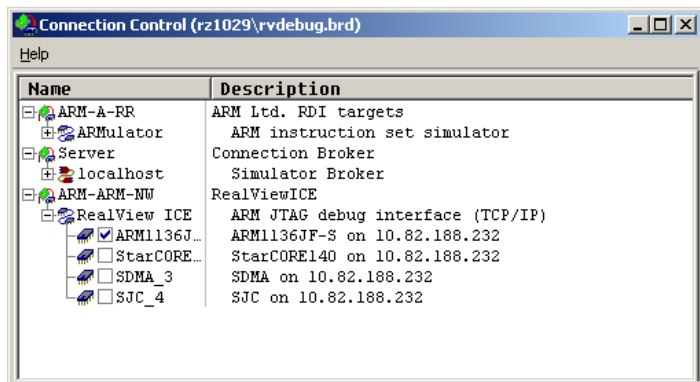


- On left side of Connection Properties window, select **CONNECTION=RealView ICE > Advanced Information > Default > ARM_config**
- On right side of Connection Properties window, click arrow button in **Value** column—debugger sets **Vector catch** to False
- Select **File > Close**—Connection Properties window closes

7. Connect to target hardware

- From RealView Debugger main menu bar, select **Target > Connect to Target**—debugger connects to target hardware; **Connection Control** window appears

Connection Control Window



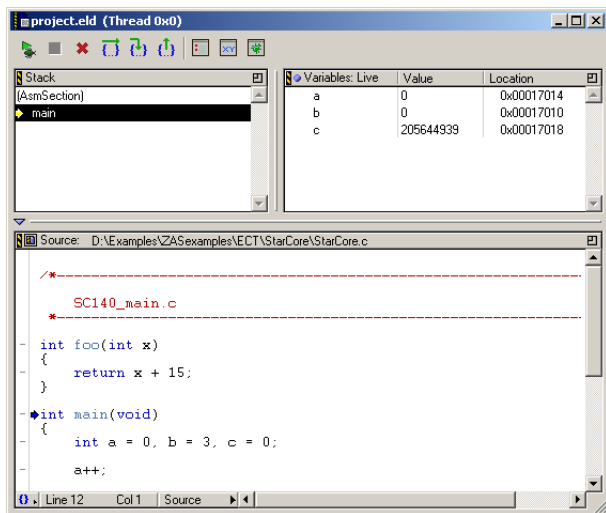
Right side of Connection Control window, select **ARM-ARM-NW**
> realView ICE—list of remote connections appears

- c. Check checkbox next to appropriate target connection

8. Build and debug StarCore project

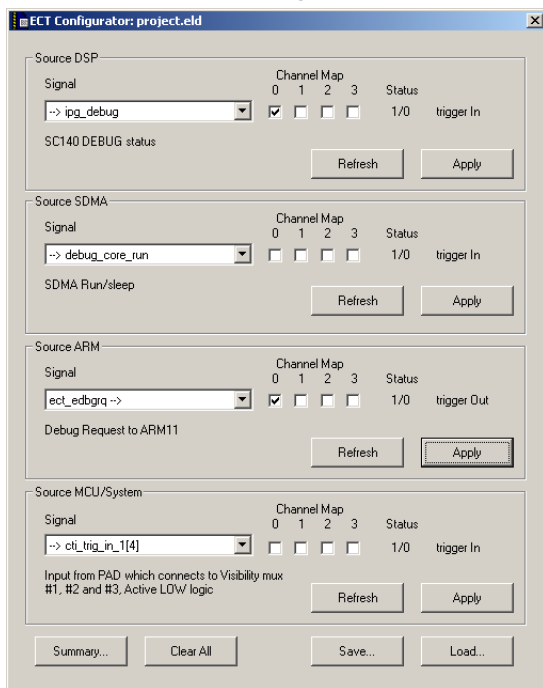
- a. Select **Project > Make** from CodeWarrior menu bar — IDE builds (compiles, assembles, and links) source code; IDE generates executable program
- b. Select **Project > Debug** — debugging session starts; thread window appears

Thread Window



- a. With thread window open, select **Debug > ECT > Configure** — ECT Configurator appears

ECT Configurator



ECT Configurator: project.eld

Source DSP

Signal: --> ipg_debug

Channel Map	0	1	2	3	Status
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1/0 trigger In

SC140 DEBUG status

Refresh Apply

Source SDMA

Signal: --> debug_core_run

Channel Map	0	1	2	3	Status
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1/0 trigger In

SDMA Run/sleep

Refresh Apply

Source ARM

Signal: ect_edbgrq -->

Channel Map	0	1	2	3	Status
	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1/0 trigger Out

Debug Request to ARM11

Refresh Apply

Source MCU/System

Signal: --> ctl_trig_in_1[4]

Channel Map	0	1	2	3	Status
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1/0 trigger In

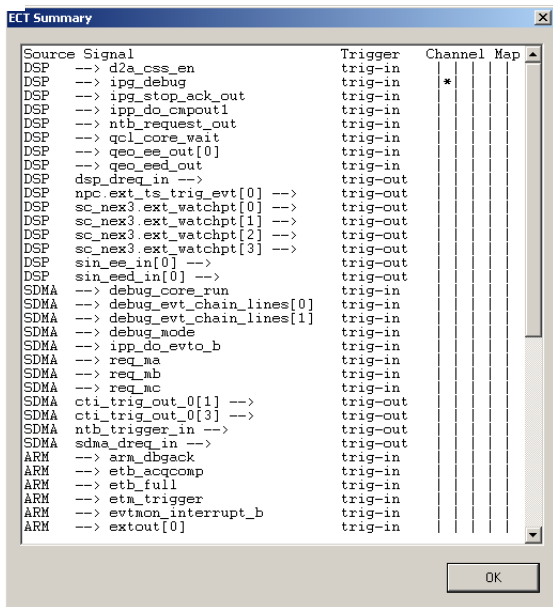
Input from PAD which connects to Visibility mux #1, #2 and #3, Active LOW logic

Refresh Apply

Summary... Clear All Save... Load...

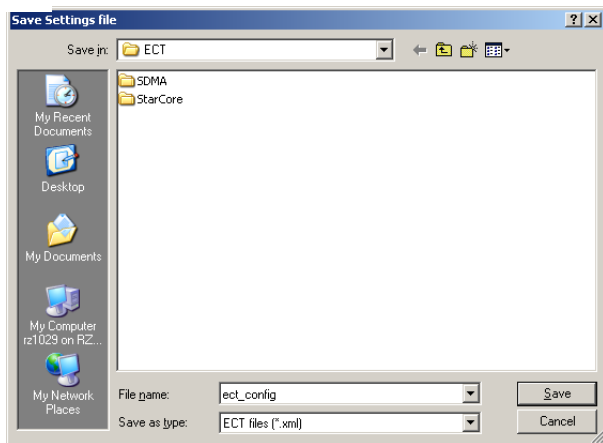
- b. In **Source DSP** group, use **Signal** list box to specify -->**ipg_debug**, check Channel Map **0** checkbox, and click **Apply** button
- c. In **Source ARM** group, use **Signal** list box to specify **ect_edbgrq-->**, check Channel Map **0** checkbox, and click **Apply** button
- d. Click **Summary** button to view summary of ECT signals that you mapped to channels

ECT Summary



- e. Click **OK** button — **ECT Summary** window closes
- f. In ECT Configurator, click **Save** button — **Save Settings** file dialog box appears

Save Settings File Dialog Box



- g. Use dialog box to navigate to

CodeWarrior \ (CodeWarrior_Examples) \ ECT_Examples

where *CodeWarrior* is path to your CodeWarrior installation

- h. In **File name** text box, enter `ect_config`
 i. Use **Save as type** list box to specify **ECT files (*.xml)**
 j. Click **Save** button — IDE saves current ECT configuration

NOTE In Section B, you will load this saved ECT configuration.

- k. Close ECT Configurator window

10. Run StarCore project and ARM core

- From CodeWarrior IDE menu bar, select **Project > Run** — IDE runs StarCore project; StarCore part runs
- From RealView Debugger menu bar, select **Debug > Run** — ARM core runs
- From CodeWarrior IDE menu bar, select **Debug > Break** — IDE halts execution of StarCore project and goes into debug mode; notice that ARM core also stops



leWarrior and RealView debugging session

- a. From CodeWarrior menu bar, select **Debug > Kill** — debug session ends; thread window closes
- b. Close CodeWarrior IDE's console windows
- a. From RealView debugger menu bar, select **Target > Disconnect** — RealView debugger disconnects from hardware

Section B: Using the ECT with Three Cores

In this section, you load the ECT matrix that you saved from Section A and then modify it. You define a new channel mapping that stops both the ARM core and the SDMA core when the StarCore part enters debug mode.

NOTE You must complete Section A before you continue with this section.

1. Open sample SDMA project

- a. From CodeWarrior menu bar, select **File > Open** — **Open** dialog box appears
- b. Use dialog box to find and open SDMA example directory at this location, where *CodeWarrior* is path to your CodeWarrior installation:

```
CodeWarrior\ (CodeWarrior_Examples) \ECT_Examples\  
SDMA
```

- c. Select **SDMA.mcp** file
- d. Click **Open** button — system opens project; project window appears, docked at left side of main window

NOTE As an example, this Quick Start shows how to create a remote connection for MXC-05 hardware. If you have different hardware, use these steps as a guide to create an appropriate remote connection.

- a. Select **Edit > Preferences** from CodeWarrior menu bar — **IDE Preferences** window appears
- b. Select **Remote Connections** from **IDE Preference Panels** list on left side of window — panel appears in window
- c. Click **Add** button — **New Connection** dialog box appears
- d. In **Name** text box, enter **MXC-05 SDMA RVI**
- e. Use **Debugger** list box to specify **SDMA RVI**
- f. In **Server IP Address** text box, enter IP address of RVICE server
- g. Check **Enable ECT** checkbox
- h. Click **OK** button — IDE saves new remote-connection configuration; name of new remote connection appears in panel list
- i. Click **OK** button — IDE Preferences window closes

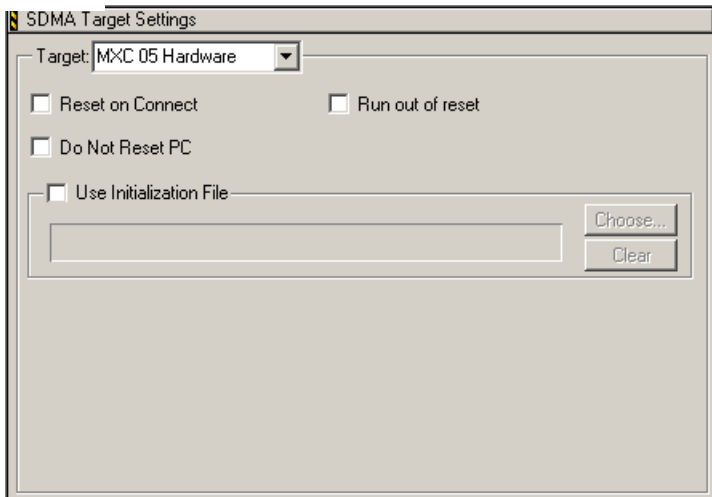
3. Configure remote debugging for SDMA project

- a. Select **Edit > SDMA Stationery Settings** from CodeWarrior menu bar — Target settings window appears
- b. Select **Remote Debugging** from **Target Settings Panels** list on left side of window — panel appears in window
- c. Use **Connection** list box to specify remote connection that you created in step 2.
- d. Set **JTAG Clock Speed** to 2000

4. Configure SDMA target settings

- a. Select **SDMA Target Settings** from Target Settings Panels list on left side of window — panel appears in window

NOTE Make sure to select **SDMA Target Settings**, not **SDMA Target**.



- b. Use **Target** list box to specify appropriate target for your hardware
- c. Click **OK** button — IDE saves changes; Target settings window closes

5. Connect to target hardware

- a. From RealView Debugger main menu bar, select **Target > Connect to Target** — debugger connects to target hardware; **Connection Control** window appears
- b. On left side of Connection Control window, select **ARM-ARM-NW > RealView ICE**—list of remote connections appears
- c. Check checkbox next to appropriate target connection

6. Build and debug StarCore project

- a. Select **Window > x StarCore.mcp**, where **x** is a number — StarCore project window comes forward
- b. Select **Project > Make** — IDE builds (compiles/assembles and links) executable program
- c. Select **Project > Debug** — debug session starts; thread window appears



id debug SDMA project

- Select **Window > x SDMA.mcp**, where **x** is a number — SDMA project window comes forward
- Select **Project > Make** — IDE builds (compiles/assembles and links) executable program
- Select **Project > Debug** — debug session starts; thread window appears

8. Wake up SDMA core

NOTE This step ensures that the SDMA core is not in sleep mode. This way, the SDMA core will respond to ECT events.

- From RealView Debugger main menu bar, select **Tools > Include Commands from File — Select File to Include Commands from** dialog box appears
- Use dialog box to navigate to

`CodeWarrior\ (CodeWarrior_Examples)\ECT_Examples`

where *CodeWarrior* is path to your CodeWarrior installation

- Select `wakeupSdma.inc` file
- Click **Open** button — SDMA wakeup script runs
- From CodeWarrior menu bar, select **Window > x sdma.eld**, where **x** is a number — SDMA thread window comes forward; SDMA core is no longer in sleep mode

9. Configure ECT

- From CodeWarrior menu bar, select **Window > x project.eld**, where **x** is a number — StarCore thread window comes forward
- Select **Debug > ECT > Configure** — ECT Configurator appears
- Click **Clear All** button — ECT Configurator deletes previous channel mappings
- Click **Load** button — **Load Settings file** dialog box appears



dialog box to navigate to

`CodeWarrior\ (CodeWarrior_Examples)\ECT_Examples`

where *CodeWarrior* is path to your CodeWarrior installation

- f. Select `ect_config.xml` file
- g. Click **Open** button — ECT Configurator loads saved settings
- h. Click **Summary** button of ECT Configurator — **ECT Summary** window appears; note that settings match those saved in step 9 of Section A.
- i. Click **OK** button — ECT Summary window closes
- j. In **Source SDMA** group, use **Signal** list box to specify **sdma_dreq_in-->**, check Channel Map **0** checkbox, and click **Apply** button

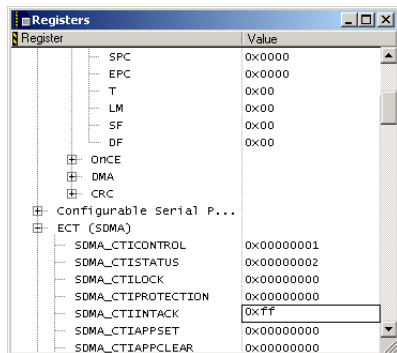
NOTE In this step, you created one additional channel mapping for the ECT matrix. This additional mapping persists for the current debugging session only. To include the additional mapping in future debugging sessions, re-save the `ect_config.xml` file.

- k. Close ECT Configurator window

10. Run StarCore, SDMA, and ARM cores

- a. Select **Window > x project.eld**, where **x** is a number — StarCore thread window comes forward
- b. Select **Project > Run** — IDE runs StarCore project; StarCore part runs
- c. Select **Window > x sdma.eld**, where **x** is a number — SDMA thread window comes forward
- d. Select **View > Registers** — Registers window appears

Registers Window



Register	Value
SPC	0x0000
EPC	0x0000
T	0x00
LM	0x00
SF	0x00
DF	0x00
OnCE	
DMA	
CRC	
Configurable Serial P...	
ECT (SDMA)	
SDMA_CTICONTROL	0x00000001
SDMA_CTISTATUS	0x00000002
SDMA_CTILOCK	0x00000000
SDMA_CTIProtection	0x00000000
SDMA_CTIINTACK	0xff
SDMA_CTIAPPSET	0x00000000
SDMA_CTIAPPCLEAR	0x00000000

- e. Navigate to **SDAM RVI > ECT (SDMA) > SDMA_CTIINTACK**
- f. Double-click **Value** corresponding to **SDMA_CTIINTACK**
- g. Enter 0xff — SDMA_CTIINTACK register is manually set; SDMA interrupt is acknowledged

NOTE The visible value of **SDMA_CTIINTACK** does not change, but the register is still set, acknowledging the SDMA interrupt.

- h. Close Registers window
- i. Select **Window > x sdma.eld**, where **x** is a number — SDMA thread window comes forward
- j. Select **Project > Run** — IDE runs SDMA project; SDMA core runs
- k. From RealView debugger menu bar, select **Debug > Run** — ARM core runs
- l. From CodeWarrior IDE, select **Window > x project.eld**, where **x** is a number — StarCore thread window comes forward
- m. From CodeWarrior menu bar, select **Debug > Break** — IDE halts execution of StarCore project and goes into debug mode; notice that both SDMA core and ARM core also stop



CodeWarrior debugging session

- a. From CodeWarrior menu bar, select **Debug > Kill** — debug session ends; thread window closes
- b. Close CodeWarrior IDE's console and project windows
- c. Select **File > Exit** — CodeWarrior IDE closes

12. End RealView debugging session

- a. From RealView debugger menu bar, select **Target > Disconnect** — RealView debugger disconnects from hardware
- b. Select **File > Exit** — **Exit** dialog box appears
- c. Click **Yes** button — RealView debugger closes

Congratulations!

You just used CodeWarrior software to run a simple StarCore and SDMA program that uses the ECT.



d the Freescale logo are trademarks of Freescale Semiconductor, Inc.
a trademark or registered trademark of Freescale Semiconductor, Inc. in the
United States and/or other countries. All other product or service names are the property of
their respective owners.

Copyright © 2005–2006 by Freescale Semiconductor, Inc. All rights reserved.

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. 7700 West Parmer Lane Austin, TX 78729 U.S.A.
World Wide Web	http://www.freescale.com/codewarrior
Technical Support	http://www.freescale.com/support