



FTF 2016
TECHNOLOGY FORUM

DEVICE AND U-BOOT BRING-UP USING CODEWARRIOR-ARMv8

FTF-DES-N1834

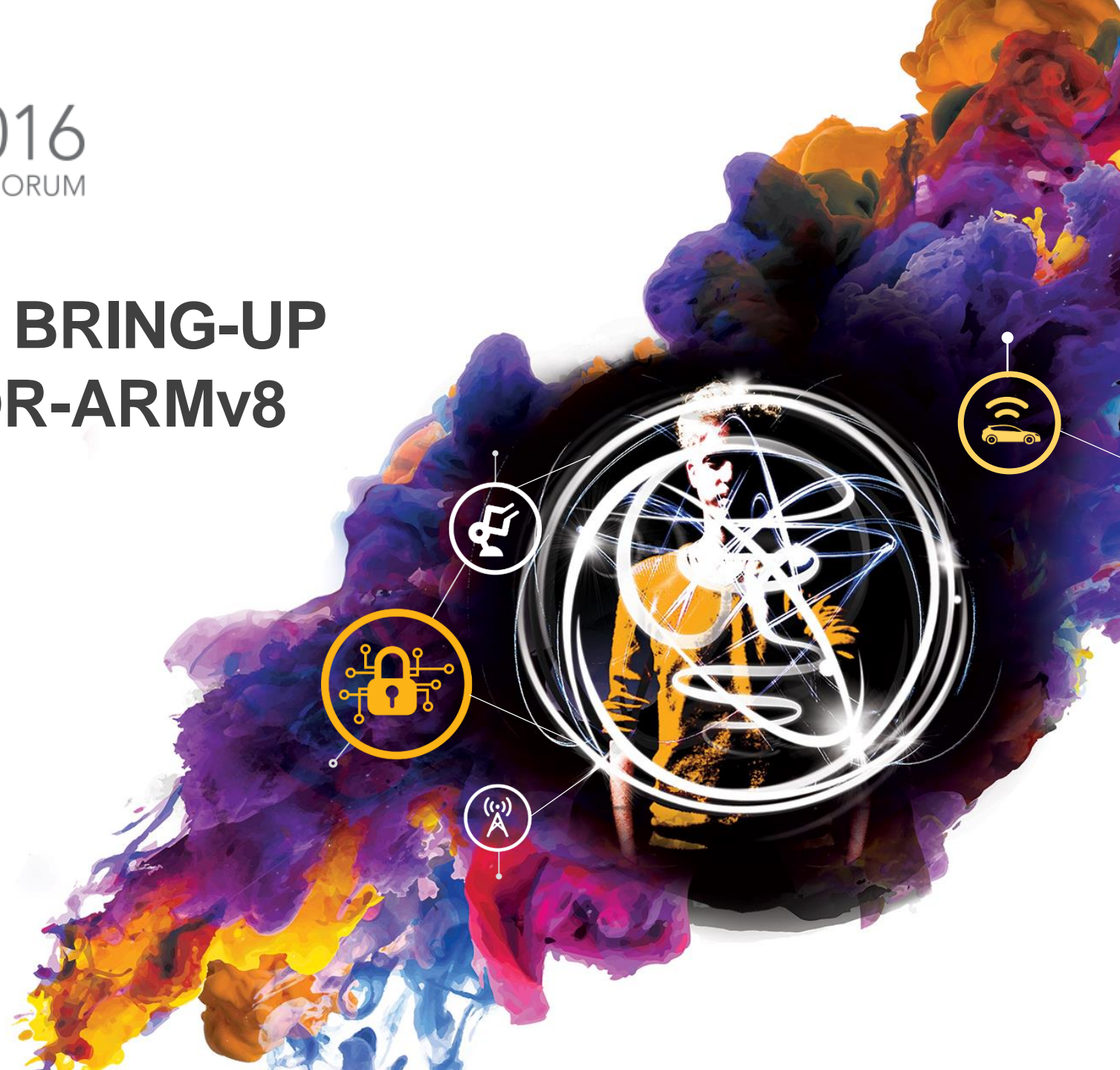
ROBERT MCGOWAN, CHIEF ARCHITECT

CATALIN UDMA

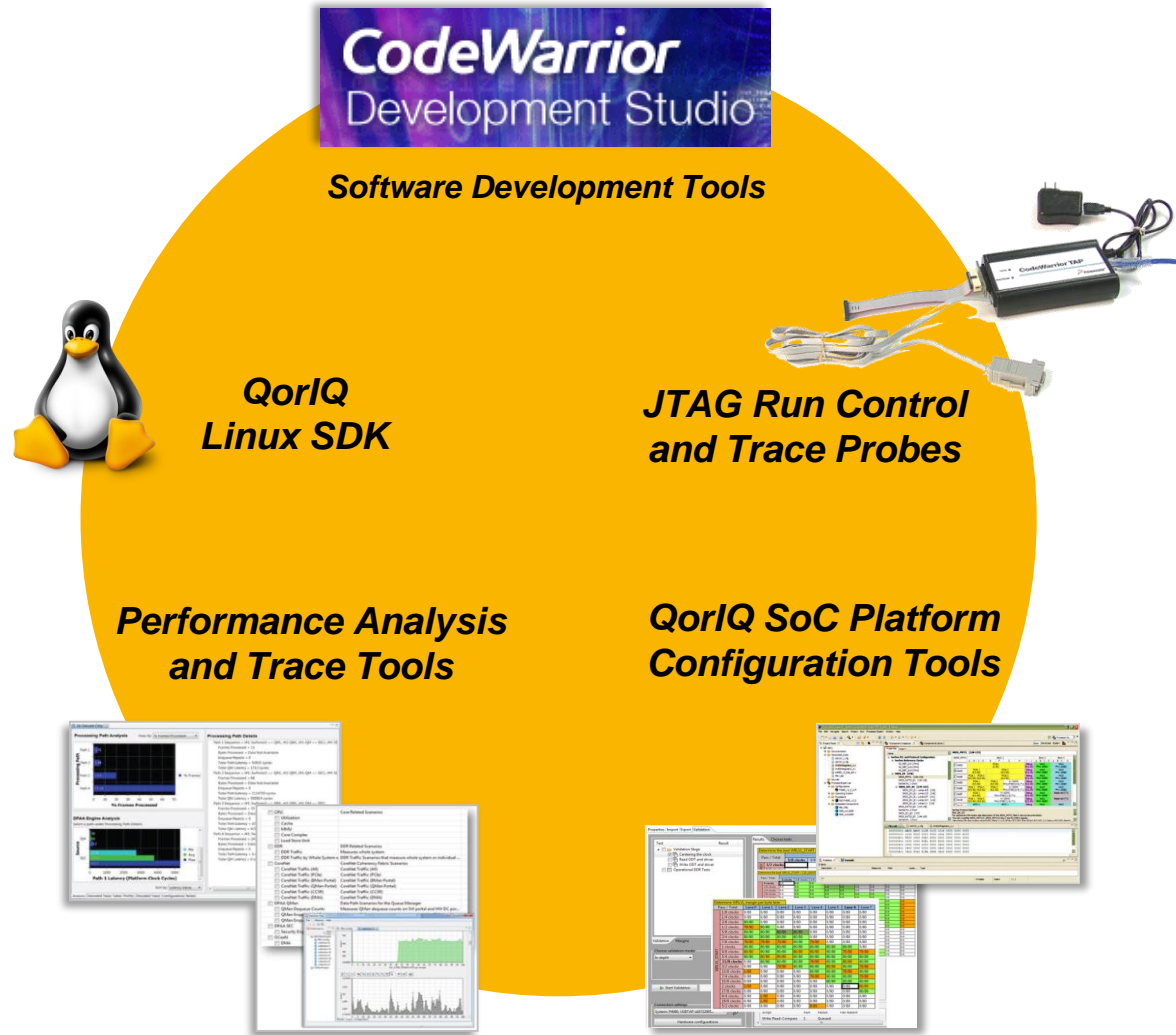
FTF-DES-N1834

MAY 16, 2016

PUBLIC USE



Software and Tools Enablement for QorIQ LS-Series



Training and Hands-on Goals

- The following material has been developed so you...
 - ...Become familiar with the LS2085A-RDB board
 - ...Perform LS2085A-RDB board bring-up using CodeWarrior-ARMv8
 - ...Perform U-Boot debug

AGENDA

- Lecture: Introduction/Overview
- Lecture: CodeWarrior
- Lecture: Board/Device Overview
- Activity: Create Bareboard Project
- Activity: Establish a CodeWarrior Connection
- Activity: Connection Diagnostic
- Activity: Flash Programmer
- Lecture: U-Boot Debug
- Activity: U-Boot Debug

INTRODUCTION/ OVERVIEW

QorIQ TOOLS

CodeWarrior for ARMv8 ISA
CW-TAP



CODEWARRIOR



CodeWarrior Family

QorIQ Tools

CodeWarrior for ARMv8

**CodeWarrior
for APP**

**CodeWarrior
for ARMv7**

**CodeWarrior
for Power
Architecture**

**CodeWarrior
for StarCore**

Configure

Build

Debug

**Trace and
Analysis**

CodeWarrior Development Studio

A Complete Development Environment Under Eclipse

Eclipse IDE

- Configuration Wizards
- Plug-in Architecture
- 3rd party community

Build Tools

- C/C++ Compiler

Initialization Tools

- SOC platform initialization and configuration

Run Control

- CW-TAP



Debugger

- Multicore aware
- Cross-triggering
 - Run/Stop of targets simultaneously
- Access to all on-chip resources
- Linux awareness

Software Analysis – Trace and Profile

- Leverages chip capabilities
 - Profiling Unit
 - In system trace buffering
- Trace/Code/Performance Viewer
- Offline trace visibility

CodeWarrior Aids Debug Through Multiple Phases

- SoC and board bring-up
 - Single-core and multicore (AMP) bare-metal debugger
 - Device introspection: core and SoC registers, memory
 - U-Boot

- Linux OS development
 - SMP aware kernel debug
 - Device driver development and debug
 - Aligned with QorIQ SDK & Linaro GNU toolchain

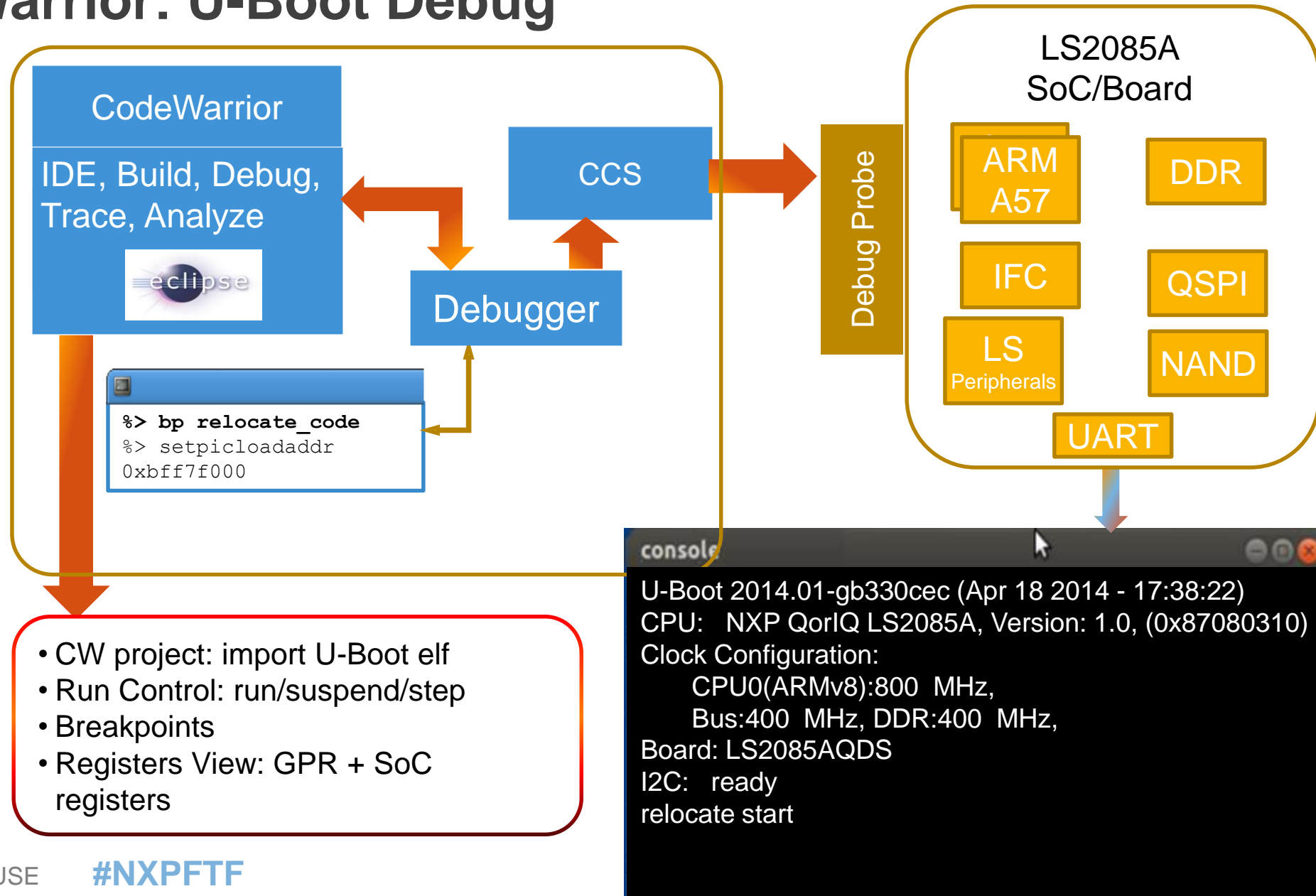
CodeWarrior Aids Debug Through Multiple Phases (2)

- Linux application development
 - GNU debugger compatible + extensions for Linux application debug
 - Linux target information: System Browser Linux kernel module development and debug
 - Aligned with QorIQ SDK and Linaro GNU toolchain
 - Target debug agent
- Performance Analysis
 - Core performance metrics and scenarios
 - SoC performance metrics and scenarios
 - Profiling from trace

CodeWarrior Aids Debug Through Multiple Phases (3)

- Non-intrusive debug through trace
 - Core and SoC trace sources: configuration, extraction, visibility
 - Post-mortem debugging: offline trace
 - Debug-print
 - Linux aware trace
 - Linux kernel trace
 - Code Coverage

CodeWarrior: U-Boot Debug



CodeWarrior JTAG Probe

- CodeWarrior TAP
 - Provides connection between CodeWarrior and target device
 - Target run control
 - Serial port pass through
 - Interface to host
 - Locally over USB
 - Remotely over Ethernet
 - Buy separately (below \$500)
- Reminder: no USB-TAP support

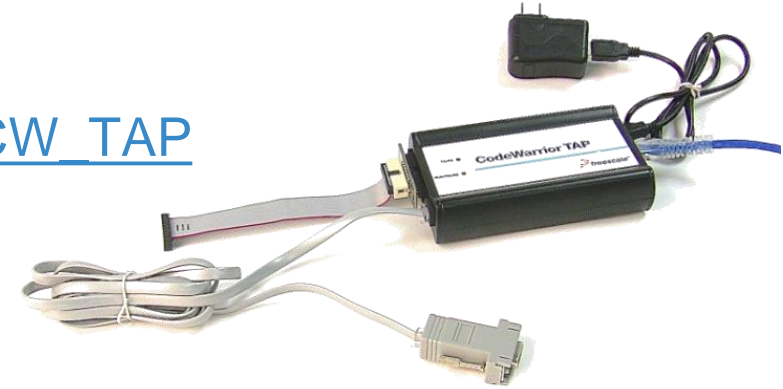


CodeWarrior TAP

- JTAG debugging and CodeWarrior run control requires:

Base Unit: (Part # [CWH-CTP-BASE-HE](#))

www.nxp.com/webapp/sps/site/prod_summary.jsp?code=CW_TAP



... and Probe Tip: (Part # [CWH-CTP-CTX10-YE](#))

www.nxp.com/webapp/sps/site/prod_summary.jsp?code=CWH-CTP-CTX10-YE

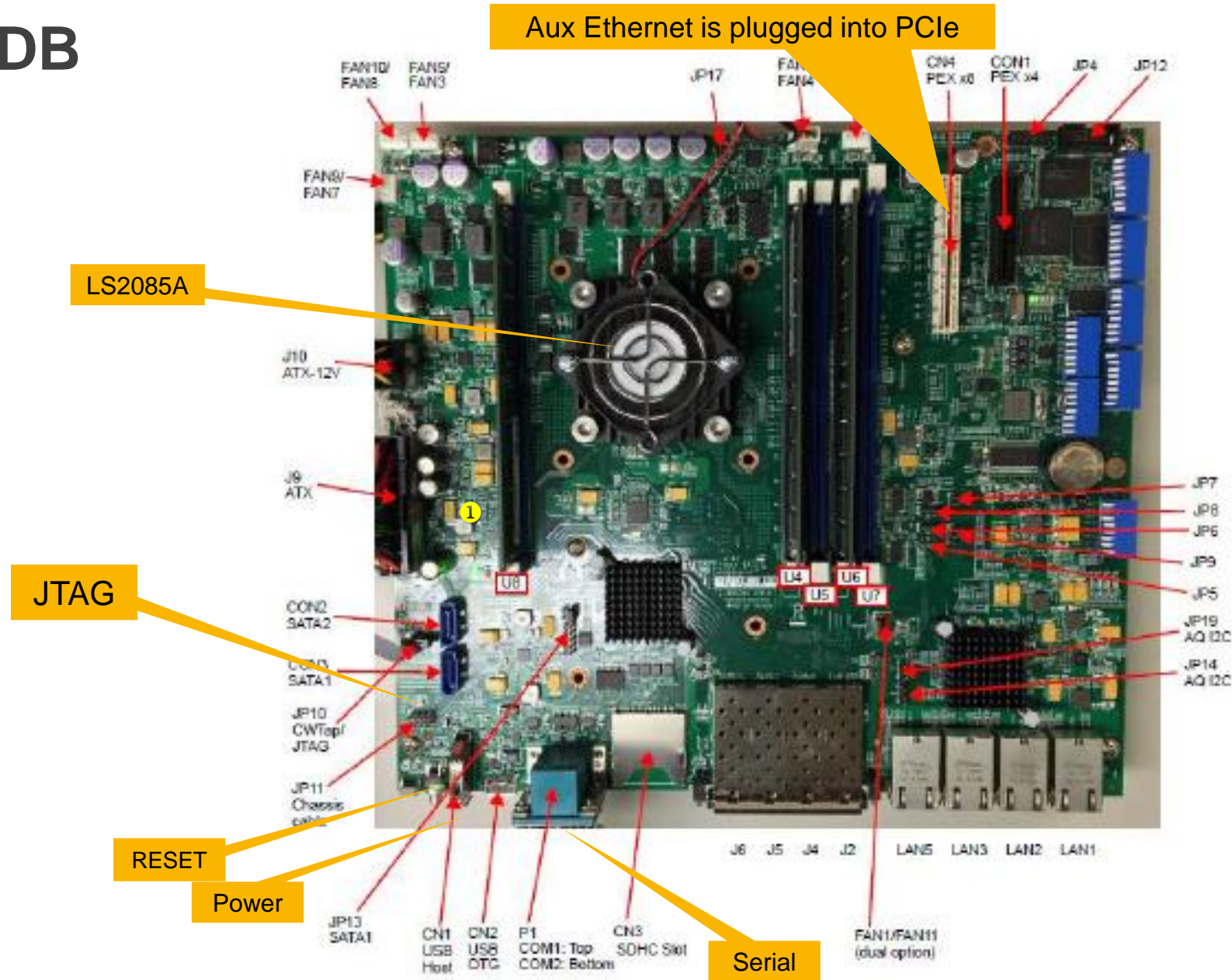


BOARD/DEVICE OVERVIEW

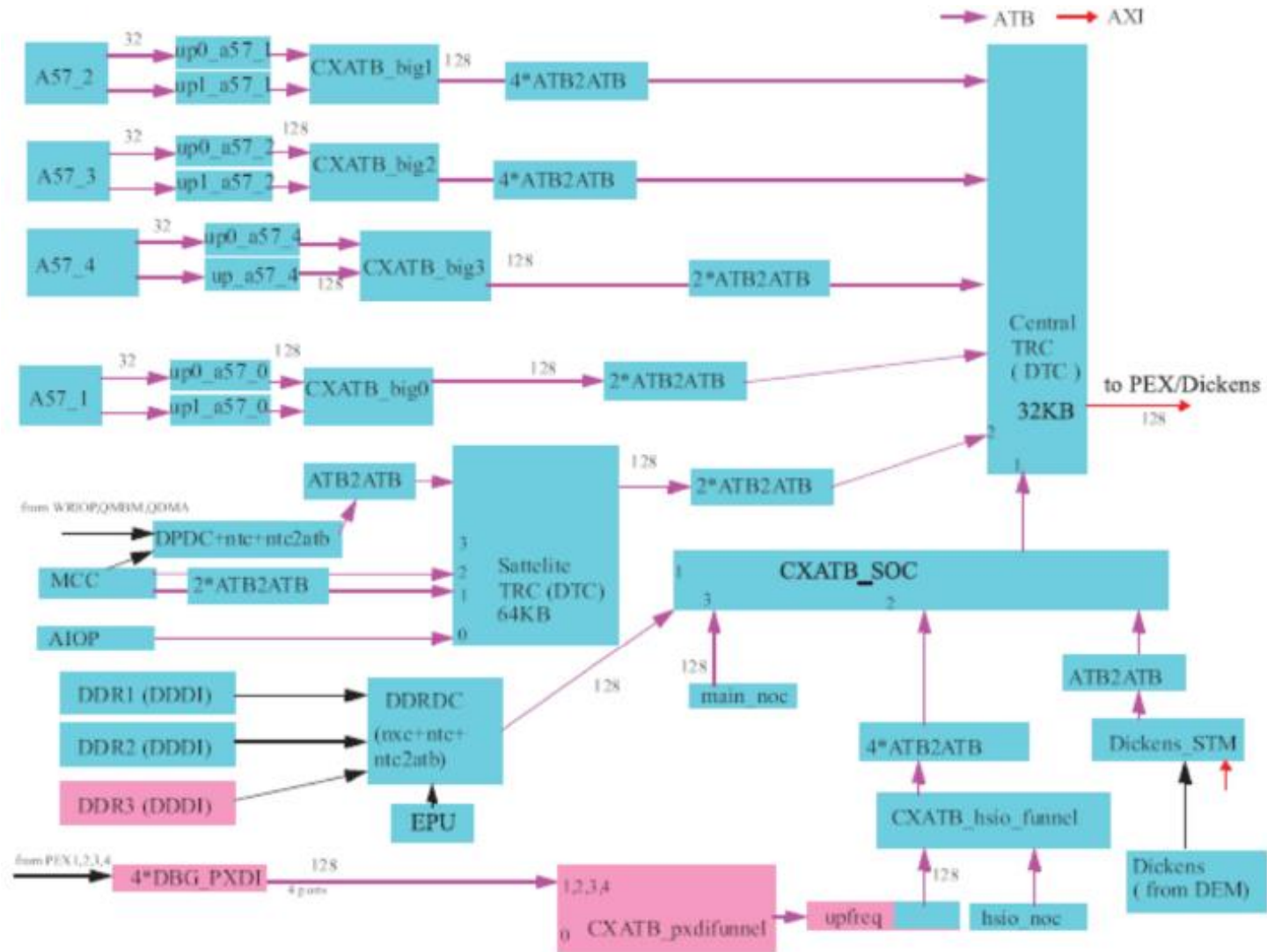
INTRODUCING THE LS2085A RDB



LS2085A RDB Top View



QorIQ LS2085A Debug Block Diagram



Debug Features

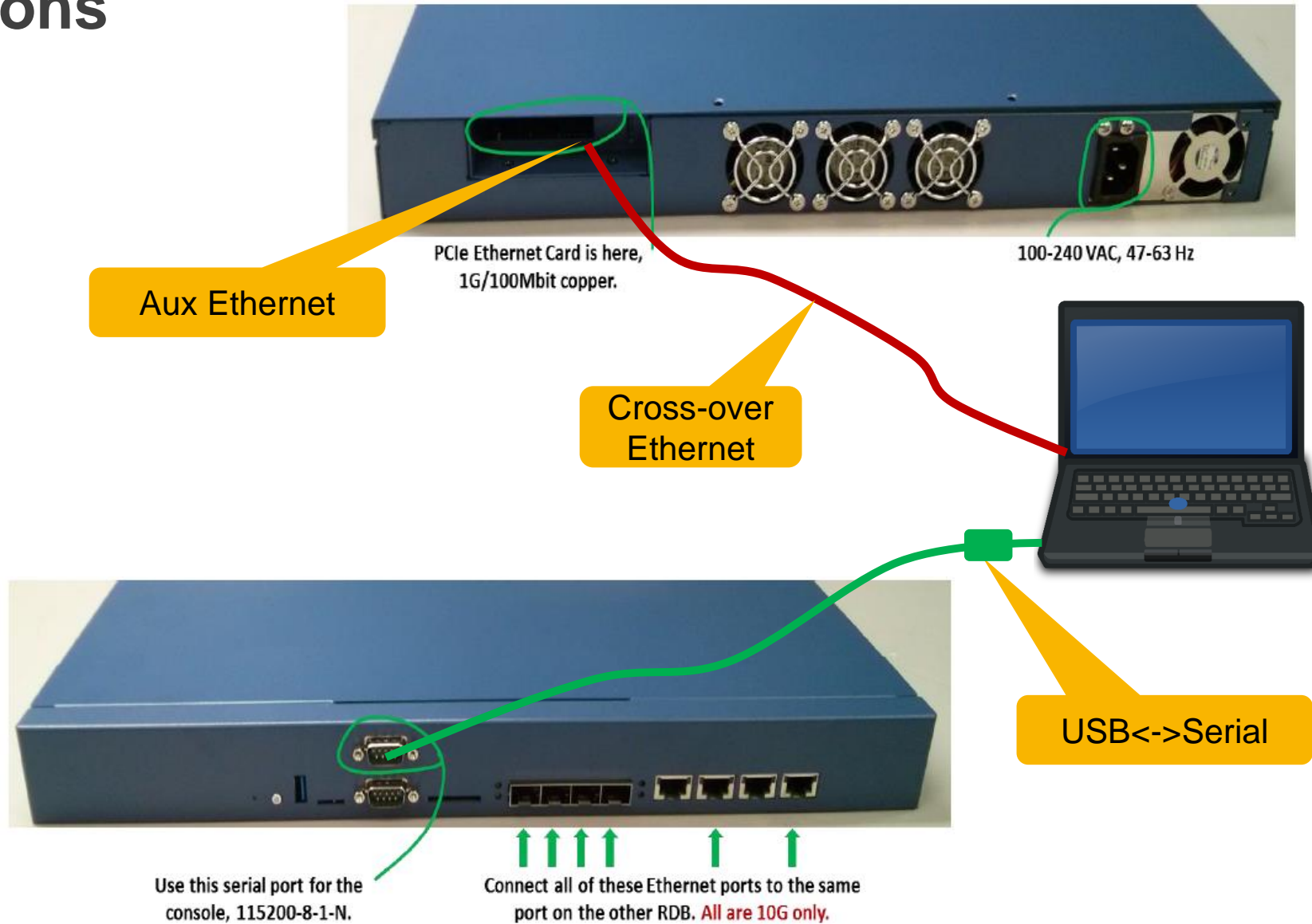
- Run-Control debug features in cores
 - Cross-triggering between cores
- Trace
 - Program trace (ETM)
 - System trace (STM)
 - Stored in internal memory or DDR
 - No external export via TPIU or Aurora
- EPU Performance Monitor

PREPARING THE ENVIRONMENT

What Has Been Done For You



Connections



Items That Have Been Setup for You

- Host OS
 - Best to use Linux on the host when developing Linux on the target
 - Multiple Linux OS supported
 - 64-bit Linux required
 - [Used Mint 17.1 for class](#)
- CodeWarrior for Networked Applications v2016.01
 - CodeWarrior for QorIQ ARMv8 ISA
- QorIQ Linux SDK for LS2085A RDB
 - Installed from ISOs – could also obtain from GIT
 - LS2085A-SDK-AARCH64-IMAGE-20160304-yocto
 - LS2085A-SDK-SOURCE-20160304-yocto
 - Did not use CACHE

Items That Have Been Setup for You

- Install on host
 - Yocto
 - Minicom / cutecom
 - 115200-8-N-1
 - Tftp server (not used in class)
 - telnet / putty (not used in class)
- Read RDB Quickstart Guide!
- Bitbake the SDK
- Install on target
 - Flash U-Boot

Class Information

- Linux Login
 - User: class
 - Password: codewarrior
- SDK is installed in ~/SDK
 - Need to use full path in tool: /home/class/SDK
- On desktop
 - Launcher to CodeWarrior – looks like rocket
 - shortcut to cutecom
 - Menu has link to terminal
 - Use for launch minicom
- No password on target Linux

RDB-LS2085A

SDK EAR6.0 Installed on LS2085A RDB



U-Boot Startup Messages

- Reset the RDB-LS2085A, interrupt the countdown
- Review the U-Boot output in the console window :

```
U-Boot 2015.10LS2085A-SDK+g3242b20 (Mar 21 2016 - 13:23:23 +0200)

SoC: LS2085E (0x87010010)
Clock Configuration:
  CPU0(A57):1800 MHz  CPU1(A57):1800 MHz  CPU2(A57):1800 MHz
  CPU3(A57):1800 MHz  CPU4(A57):1800 MHz  CPU5(A57):1800 MHz
  CPU6(A57):1800 MHz  CPU7(A57):1800 MHz
  Bus:      600 MHz  DDR:      1866.667 MT/s      DP-DDR:  1600 MT/s
Reset Configuration Word (RCW):
  00: 48303830 48480048 00000000 00000000
  10: 00000000 00200000 00200000 00000000
  20: 01012980 00002580 00000000 00000000
  30: 00000e0b 00000000 00000000 00000000
  40: 00000000 00000000 00000000 00000000
  50: 00000000 00000000 00000000 00000000
  60: 00000000 00000000 00027000 00000000
  70: 412a0000 00000000 00000000 00000000
Model: Freescale Layerscape 2085a RDB Board
Board: LS2085E-RDB, Board Arch: V1, Board version: D, boot from vBank: 4
```

U-Boot Startup Messages

```
DDR      15 GiB (DDR4, 64-bit, CL=13, ECC on)
         DDR Controller Interleaving Mode: 256B
         DDR Chip-Select Interleaving Mode: CS0+CS1
DP-DDR  4 GiB (DDR4, 32-bit, CL=11, ECC on)
         DDR Chip-Select Interleaving Mode: CS0+CS1
Waking secondary cores to start from fff0b000
All (8) cores are up.
Using SERDES1 Protocol: 42 (0x2a)
Using SERDES2 Protocol: 65 (0x41)
Flash: 128 MiB
NAND: 2048 MiB
MMC: FSL_SDHC: 0
AHCI 0001.0301 32 slots 1 ports 6 Gbps 0x1 impl SATA mode
flags: 64bit ncq pm clo only pmp fbss pio slum part ccc apst
Found 0 device(s).
SCSI: Net: crc32+
fsl-mc: Booting Management Complex ... SUCCESS
fsl-mc: Management Complex booted (version: 9.0.4, boot status: 0x1)
e1000: 68:05:ca:36:9c:7c
         DPMAC1@xgmii, DPMAC2@xgmii, DPMAC3@xgmii, DPMAC4@xgmii, DPMAC5@xgmii,
         DPMAC6@xgmii, DPMAC7@xgmii, DPMAC8@xgmii, e1000#0 [PRIME]

Hit any key to stop autoboot: 0
```

Linux

- Linux is automatically booting
- If U-Boot countdown has been interrupted, boot Linux with command “boot”
- When Linux booting is complete:
 - Login with user root and no password
 - Configure eth0 to 192.168.1.100

```
INIT: Entering runlevel: 5un-postinsts exists during rc.d purge
Configuring network interfaces... done.
Starting OpenBSD Secure Shell server: sshd
  generating ssh RSA key...
  generating ssh ECDSA key...
  generating ssh DSA key...
Poky (Yocto Project Reference Distro) 1.8.1 ls2085ar db /dev/ttyS1

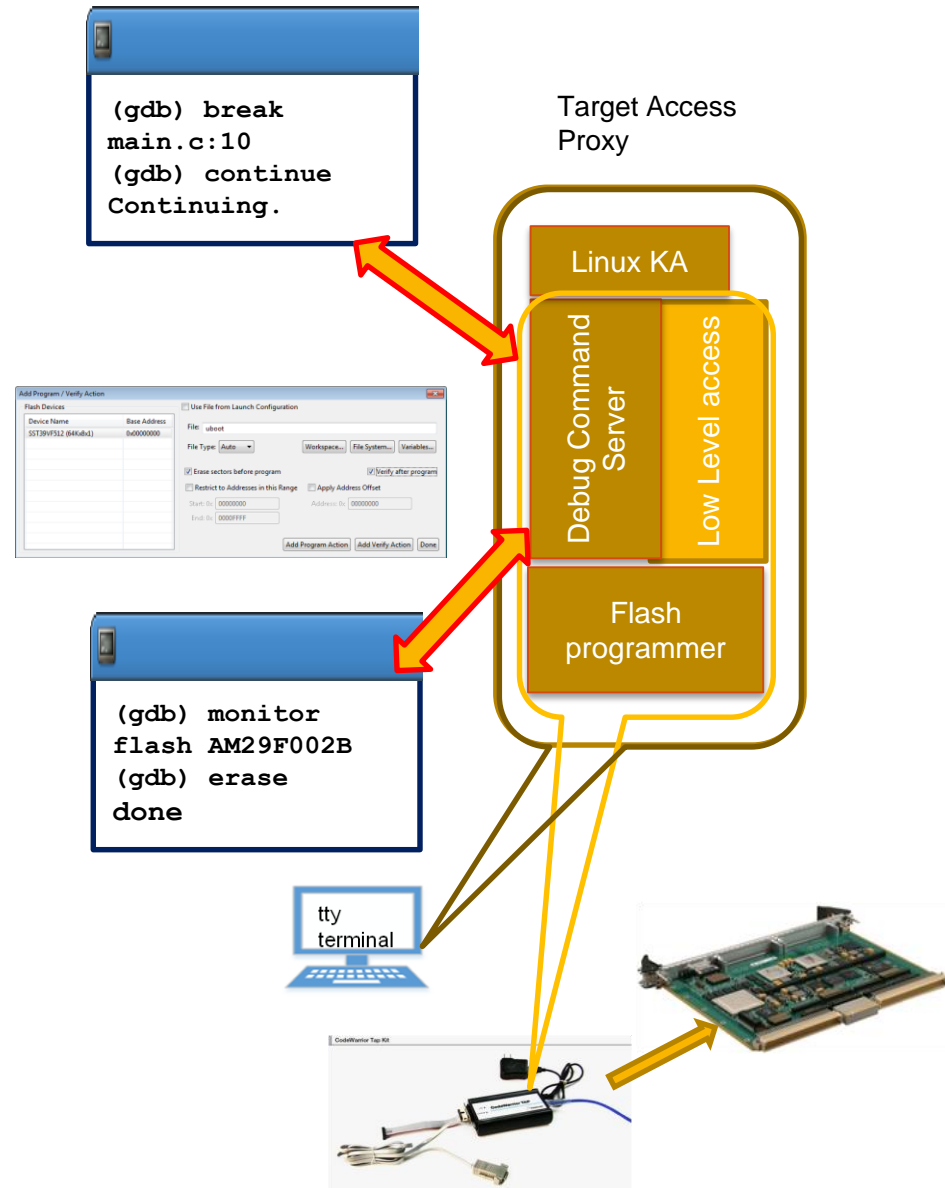
ls2085ar db login: root
root@ls2085ar db:~# ifconfig eth0 192.168.1.100
root@ls2085ar db:~#
```

BARE-METAL DEBUG



Bare-Metal Debug

- Target interface to real hardware / simulator
- Lightweight debugger engine accessible from both GUI and command line
- Compatible with the GNU debugger front-end
- Standard set of memory/register access commands + monitor extensions
- Simultaneous connectivity with multiple clients
- Single- and Multicore support



ACTIVITY

Create a CodeWarrior Bareboard Project

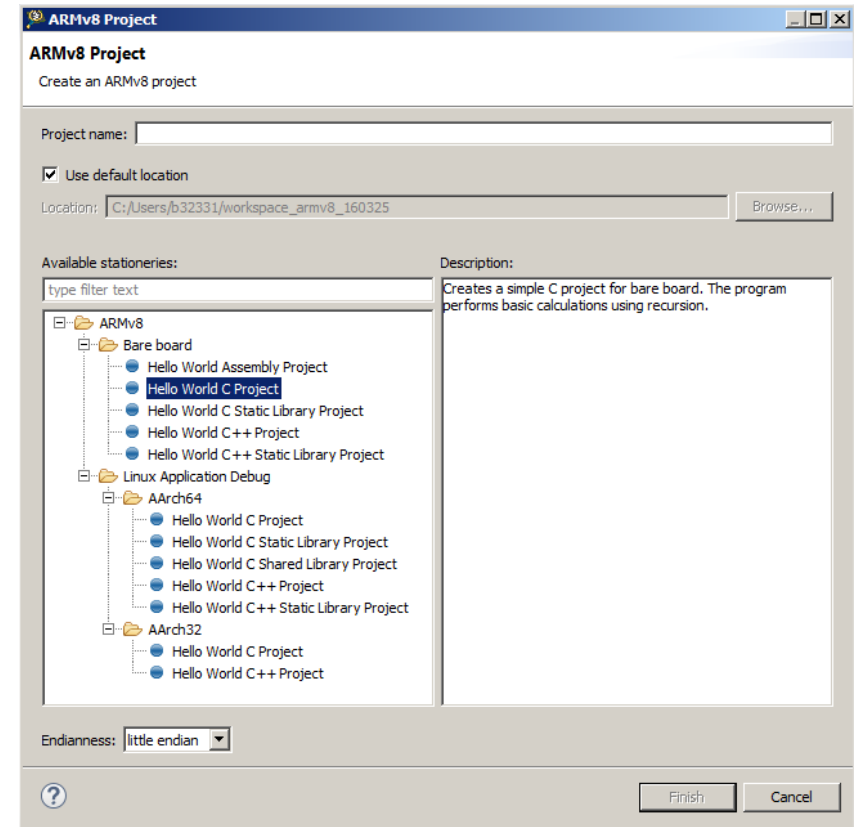
Activity – CodeWarrior Bareboard Debugging

Bareboard Debugging with CodeWarrior TAP Connection



Activity Summary:

1. Launch CodeWarrior-ARMv8 from the desktop
2. Go to Workbench
3. **File → New → ARMv8 Stationery**
4. Enter a **Project Name**
5. Choose:
 - Bare board project type (ASM, C, C++)
6. Press **Finish**



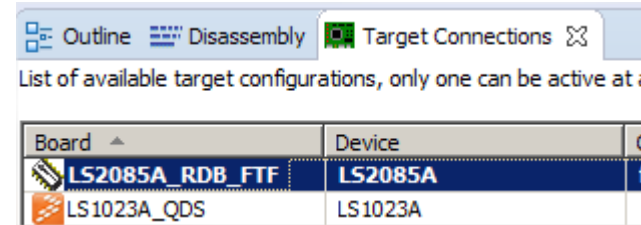
Create Bareboard Project – Activity (cont'd)

The project created can be used on any LS2 SoC and board supported:

- For any information about board hardware setup or project options please check the README section from Target Connection (double-click)

- You will find out about:

- **Boot** options switch configurations
- Default **RCW**
- **Memory Map** with accessible areas
- **SMP** debug configuration
- **Console I/O** or **UART I/O** selection
- Other useful info



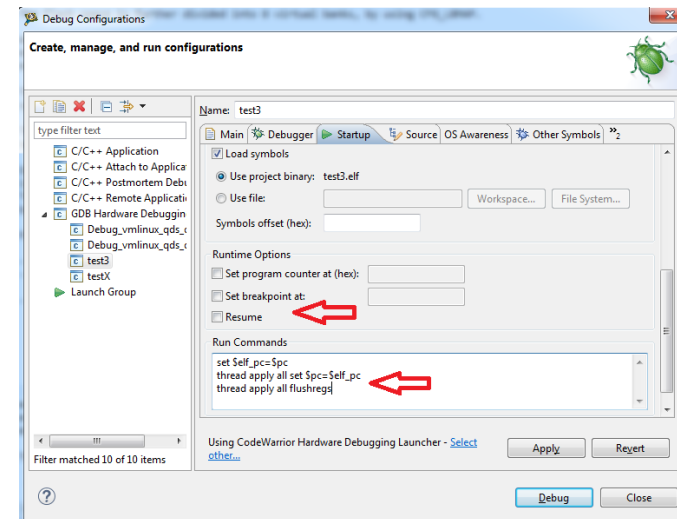
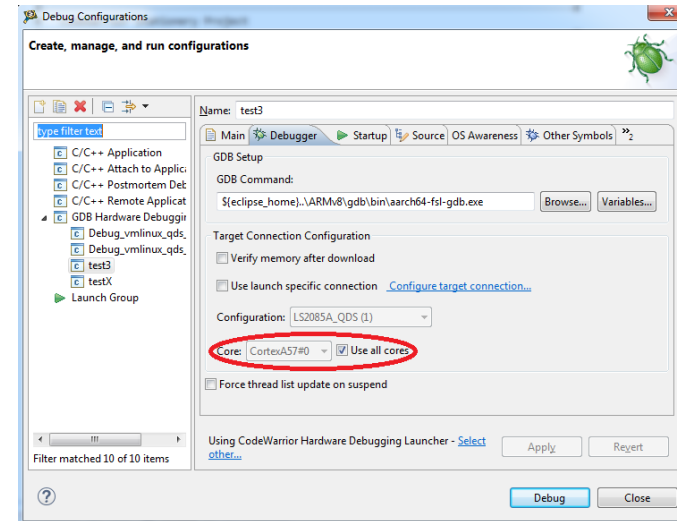
Code editor showing the README file for the LS2085A RDB project:

```
1 #-----
2 #  LS2085A RDB README
3 #-----
4
5 The LS2085A Reference Design Board (RDB) is a high-performance
6 development platform that supports the QorIQ(TM) LS2085A Layer
7
8 #-----
9 #  Switch settings for LS2085A RDB - CPLD v1.16
10 #-----
11
12 NOR_BOOT:
13 -----
14
15 PCB REV B:
16
17 SW4 : 0xFF = 11111111   SW1 : 0x12 = 00010010   SW10: 0xFF =
18                                     SW9  : 0x42 =
19 Where '1' = DOWN/ON
20
21 PCB REV C:
```


Create Bareboard Project – Activity (cont'd)

Activating SMP:

- Go to Debug Configuration
- Select *Debugger* tab
- Make sure “**Use all cores**” is checked
- Go to *Startup* tab
- Uncheck “**Set breakpoint at**” and “**Resume**” in the *Runtime Options* section
- Enter the following commands in the *Run Commands* section:
 - set \$Self_pc=\$SpC
 - thread apply all set \$SpC=\$Self_pc
 - thread apply all flushregs

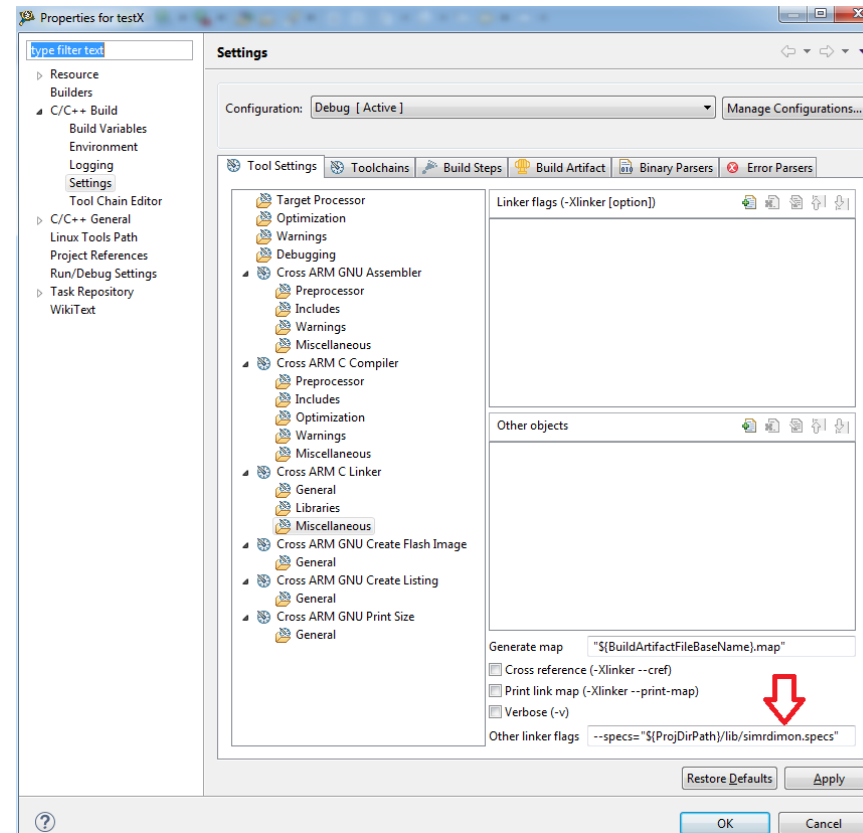


Create Bareboard Project – Activity (cont'd)

The default **I/O mode** is debugger console (in other words the *simrdimon* library is being used). You can change to **UART** I/O if needed by modifying the project build settings: navigate to

C/C++ Build -> Settings -> Cross ARM C (or C++) Linker -> Miscellaneous

- In *Other linker flags* section
- Replace *simrdimon.specs* with *uart.specs*
- Re-build the project



ACTIVITY

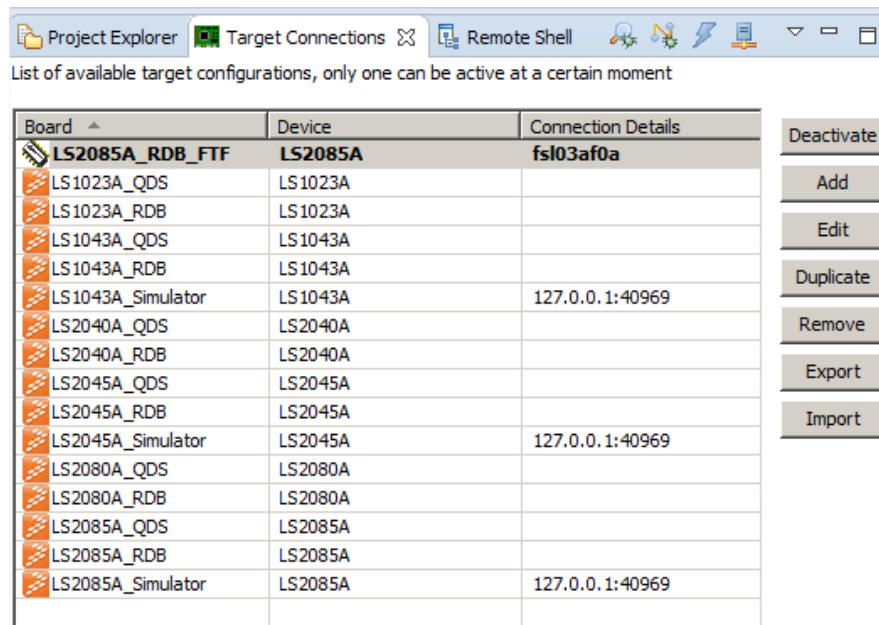
Establish a CodeWarrior Connection to the Target



Hardware Selection and Configuration Activity

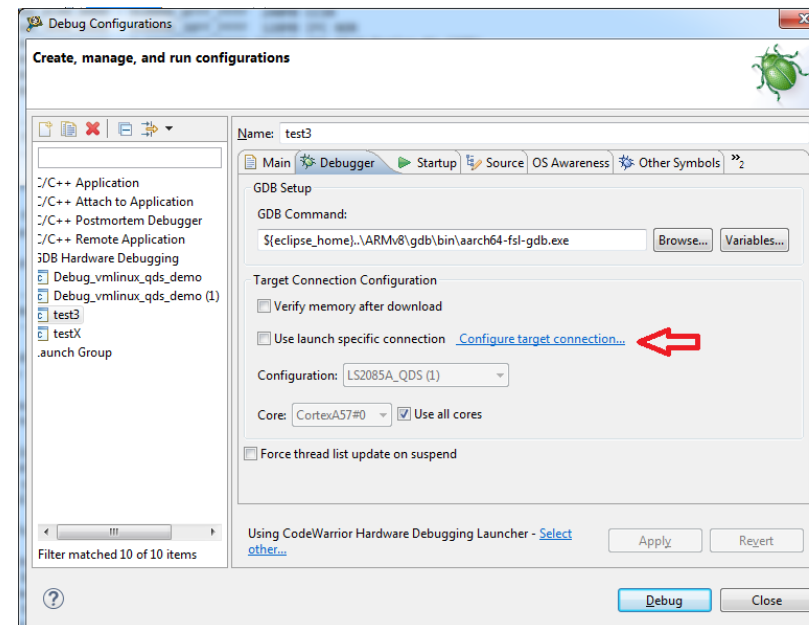
Target Connection Configurator (TCC)

- For connection settings, target SoC and board selection go to:
- *Left side in C/C++ perspective or Right side in the Debug perspective or*
- *Debug Configurations -> Debugger -> Configure target connection*



List of available target configurations, only one can be active at a certain moment

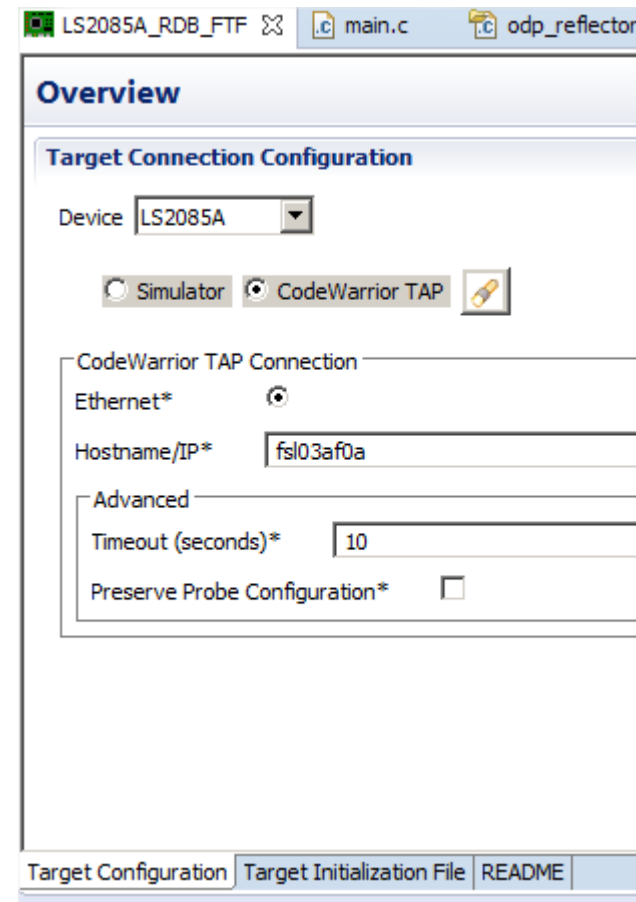
Board	Device	Connection Details
LS2085A_RDB_FTF	LS2085A	fsl03af0a
LS1023A_QDS	LS1023A	
LS1023A_RDB	LS1023A	
LS1043A_QDS	LS1043A	
LS1043A_RDB	LS1043A	
LS1043A_Simulator	LS1043A	127.0.0.1:40969
LS2040A_QDS	LS2040A	
LS2040A_RDB	LS2040A	
LS2045A_QDS	LS2045A	
LS2045A_RDB	LS2045A	
LS2045A_Simulator	LS2045A	127.0.0.1:40969
LS2080A_QDS	LS2080A	
LS2080A_RDB	LS2080A	
LS2085A_QDS	LS2085A	
LS2085A_RDB	LS2085A	
LS2085A_Simulator	LS2085A	127.0.0.1:40969



Hardware Selection and Configuration Activity

Target Connection Configurator (TCC) (cont'd)

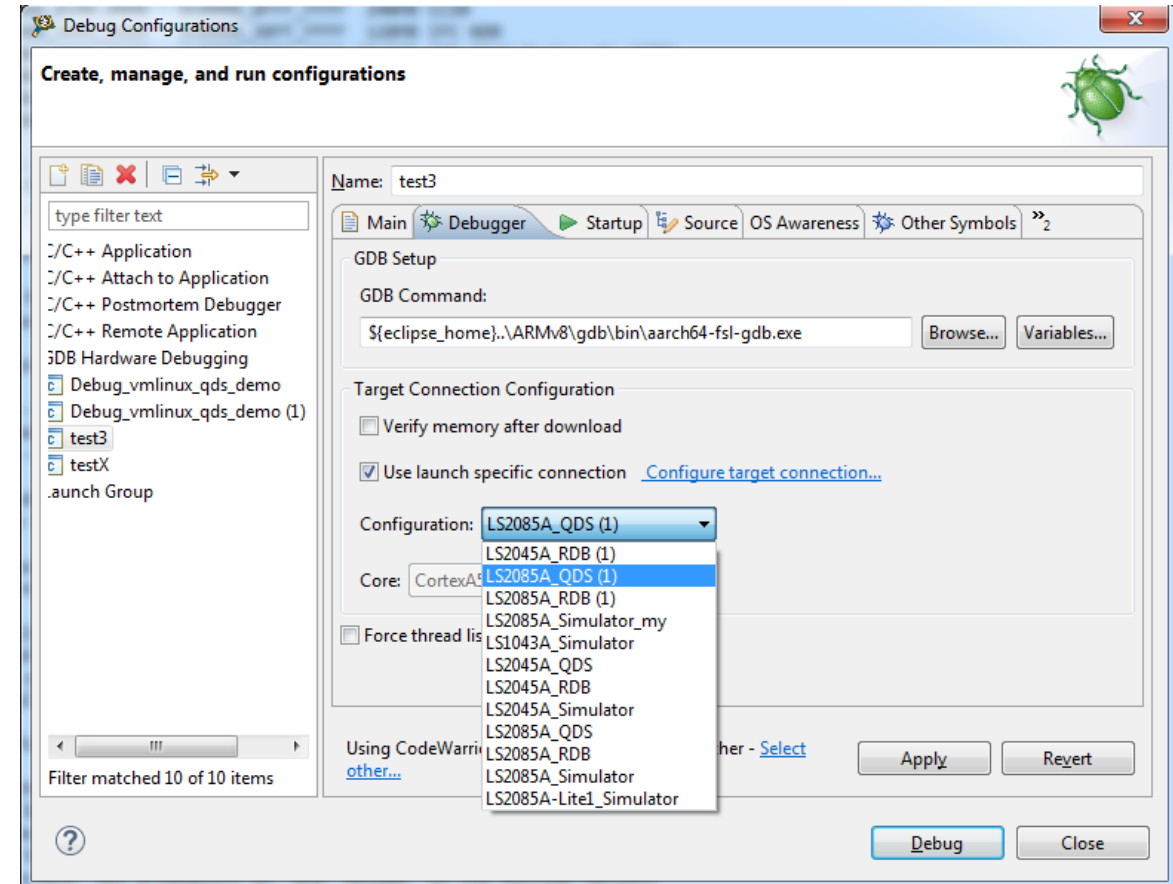
- To select the SoC & Board configuration simply double-click one of the pre-defined templates (📄) or select one and click *Duplicate* or *Edit* to create your own configuration (🔧)
- You can change:
 - **device** type
 - **connection** settings
 - **debug probe** type
- Or tweak the **initialization file**



Hardware Selection and Configuration Activity

Target Connection Configurator (TCC) (cont'd)

- Selected configuration in TCC view will be the default one for all projects (launches), except if *Use launch specific connect* is checked.
- In this case, any existing configuration can be selected:



ACTIVITY

Diagnose a CodeWarrior Connection to the Target



Diagnose a CodeWarrior Connection to the Target

Summary

- Run a Diagnose check on the existing connection
- Introduce different failures in the connection (software configuration, disconnect the CWTAP USB cable, power-off board)
- For each scenario run Diagnostic tests, view the results and the suggested corrective actions

Diagnose Connection Feature

Check if the new created TCC is reliable

- Select Diagnose Connection button from TCC

The screenshot shows the IDE interface with the Target Connections window open. The window displays a list of available target configurations, with the following data:

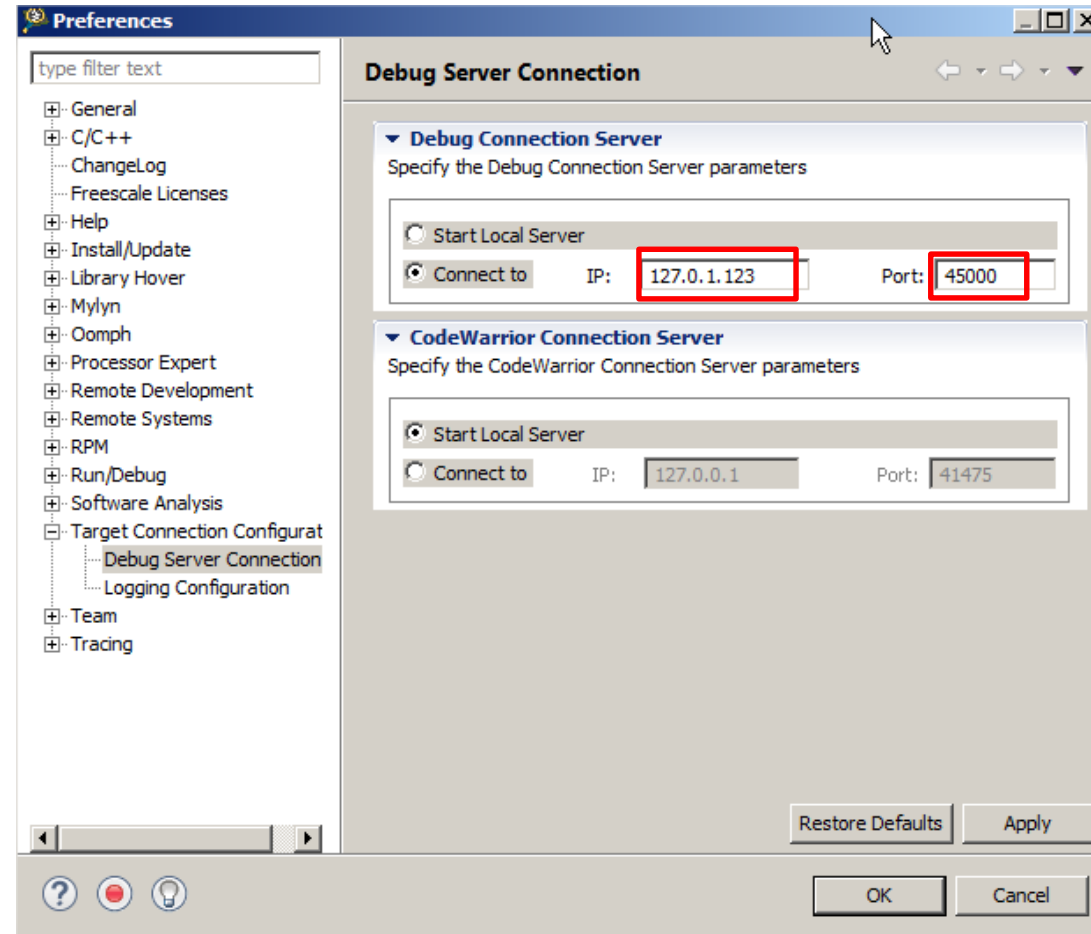
Board	Device	Connection Details	Probe	Buttons
LS2085A_RDB_FTF	LS2085A		CodeWarrior TAP	Deactivate
LS1023A_QDS	LS1023A		CodeWarrior TAP	Add

A tooltip for the 'Diagnose Connection' button is visible. Below the Target Connections window, the Connection Diagnostics window is open, showing a list of test steps and their status:

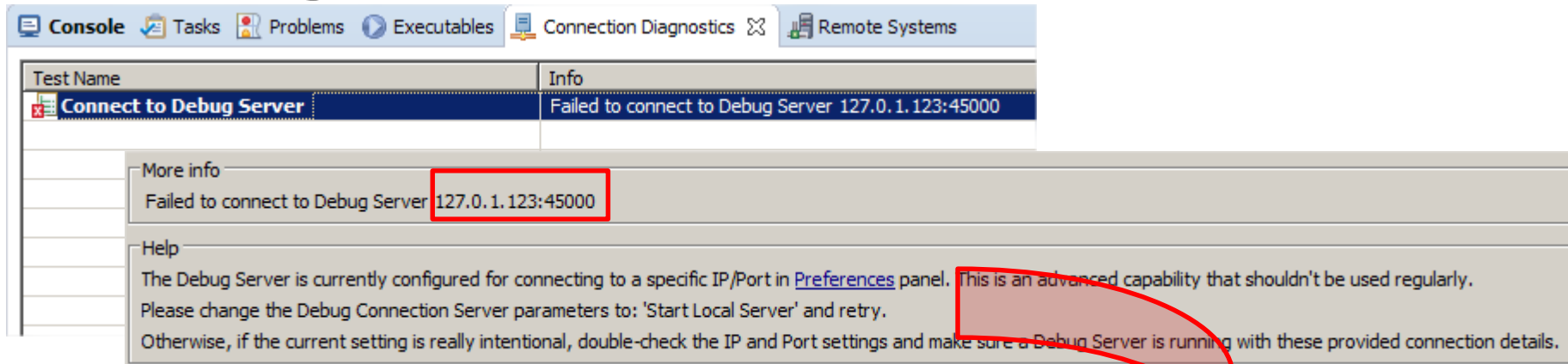
Test Name	Info
Connect to Debug Server	Version: 1.0.2.160317-d4961bc (built on Mar 17 2016 12:02:36)
Connect (telnet) to TAP probe	Skipped - only available for CodeWarrior TAP - Ethernet connections
Connect to CodeWarrior Connection Server	
Start built-in low-level JTAG tests	
Attach to CodeWarrior Connection Server	CCS Release Build 442p0
Connect to probe	
Power at probe tip	
IR Scan	IR length: 4
Bypass Scan	Bypass length: 35
TAP state moves	
Bit error stress patterns	Testing all zeros for 500 ms
Scan IDCODE	Detected IDCODE
End built-in low-level JTAG tests	
Connect to target	
Start post-config_chain user defined low-level JT	Skipped - no user-defined tests found
Run target initialization script	
Disconnect from target	

Connect to Debug Server (1)

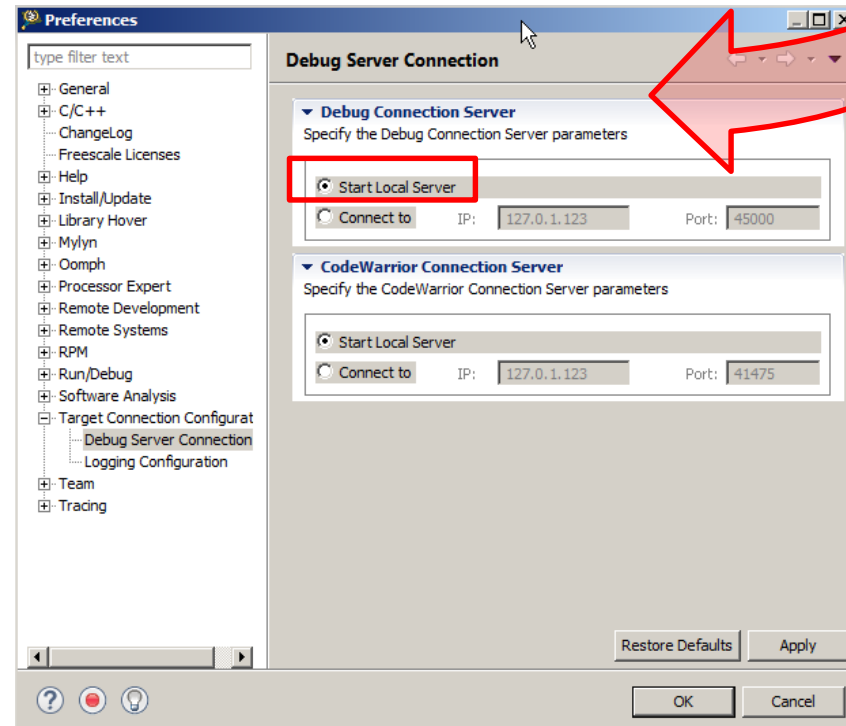
Introduce a failure – go to Window > Preferences > Target Connection Configuration > Debug Server Connection and change from Start Local Server to a custom IP/port as bellow



Connect to Debug Server (2)

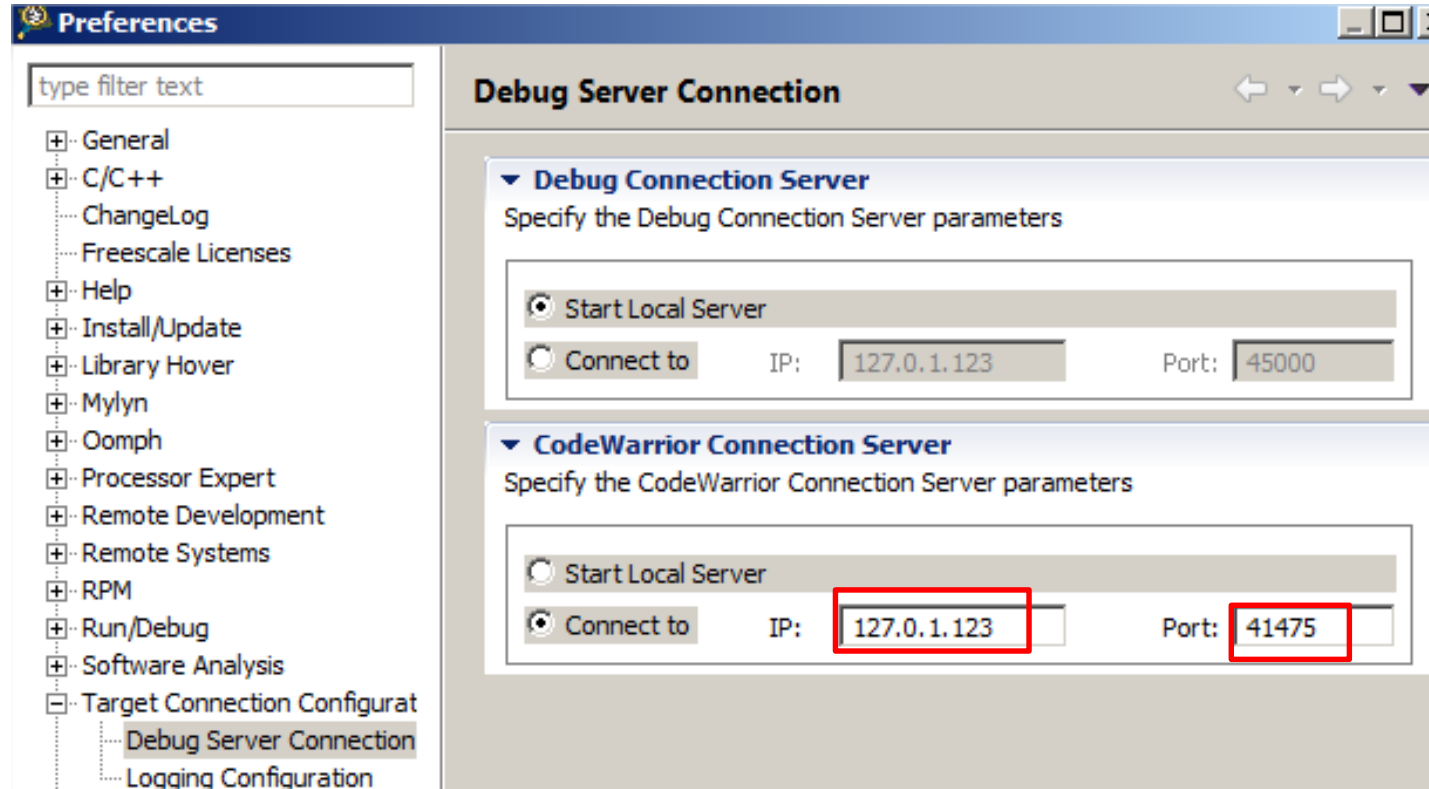


Check the Debug Server settings in the Preferences Panel (switch back to defaults)



Connect to CodeWarrior Connection Server (1)

Introduce a failure – go to Window > Preferences > Target Connection Configuration > CodeWarrior Connection Server and change from Start Local Server to a custom IP/port as below



Connect to CodeWarrior Connection Server (2)

The screenshot shows the 'Connection Diagnostics' panel with a table of test results:

Test Name	Info
Connect to Debug Server	Version: 1.0.2.160317-d4961bc (built on Mar 17 2016 12:02:36)
Connect (telnet) to TAP probe	Skipped - only available for CodeWarrior TAP - Ethernet connections
Connect to CodeWarrior Connection Server	Failed to connect to remote CCS running on 127.0.1.123:41475

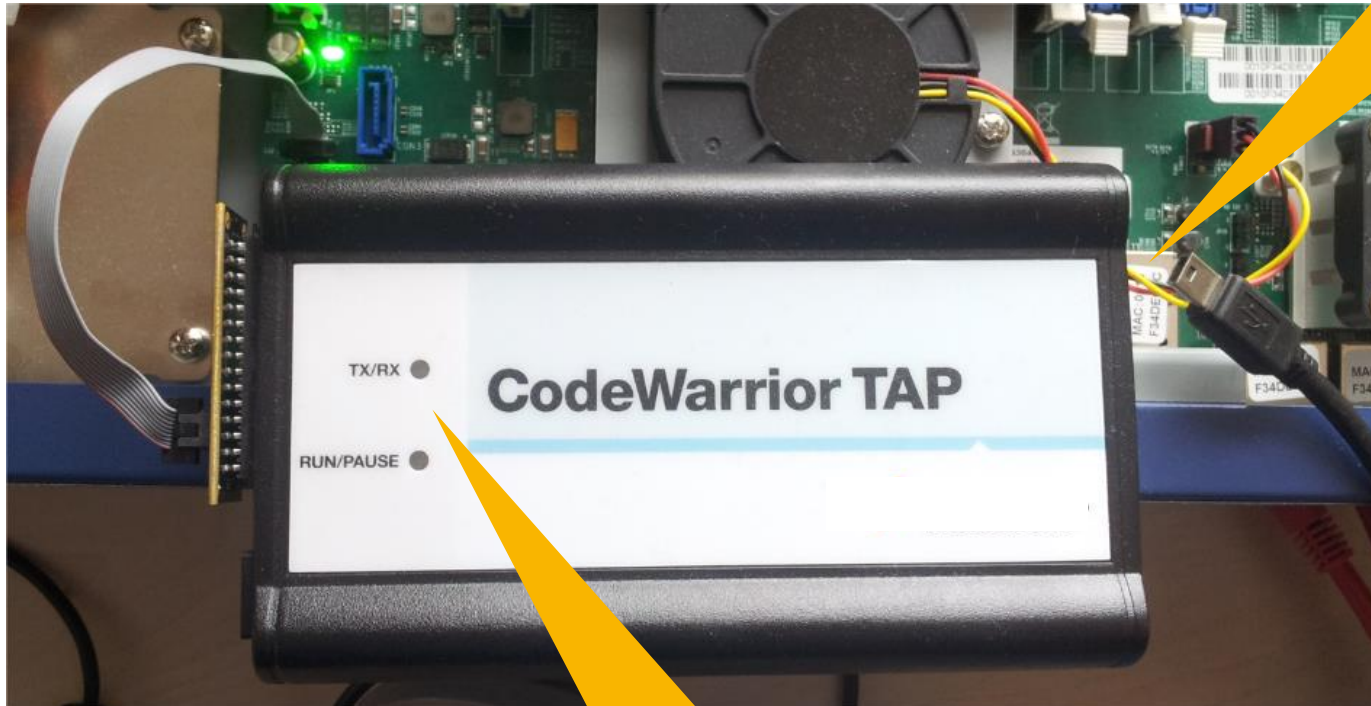
Below the table, the 'More info' section states: 'Failed to connect to remote CCS running on 127.0.1.123:41475'. The 'Help' section provides instructions: 'The CodeWarrior Connection Server is currently configured for connecting to a specific IP/Port in Preferences panel. This is an advanced capability that is less likely to be used regularly. Please change the CodeWarrior Connection Server parameters to: 'Start Local Server' and retry. Otherwise, if the current setting is really intentional, double-check the provided IP and Port settings and make sure a CodeWarrior Connection Server is running and available through network with these provided connection details.'

Check the Debug Server settings in Preferences Panel (switch back to defaults)

The screenshot shows the 'Preferences' dialog box with the 'Debug Server Connection' section selected. The 'CodeWarrior Connection Server' parameters are highlighted with a red box, showing the 'Start Local Server' radio button selected, with IP: 127.0.1.123 and Port: 41475. A large red arrow points from the 'Help' text in the previous screenshot to this section.

Connect to Probe (1)

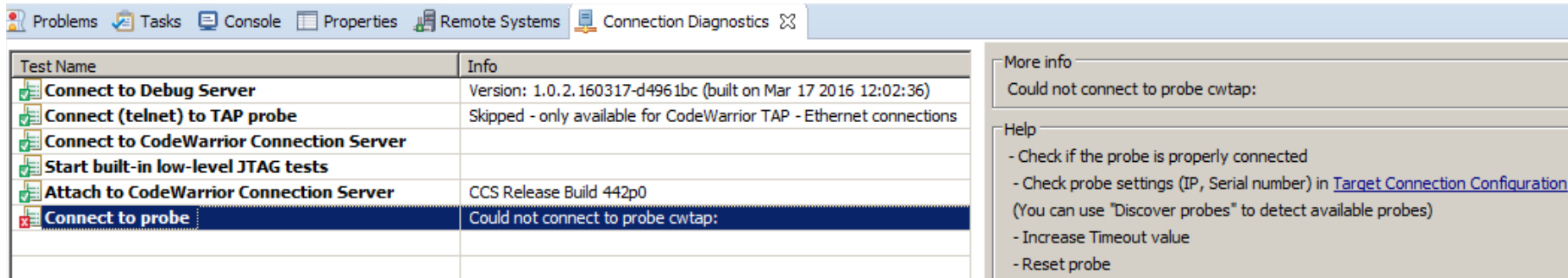
Introduce a failure – power down the cwtap probe



USB cable is disconnected

Both LEDs are off

Connect to Probe (2)

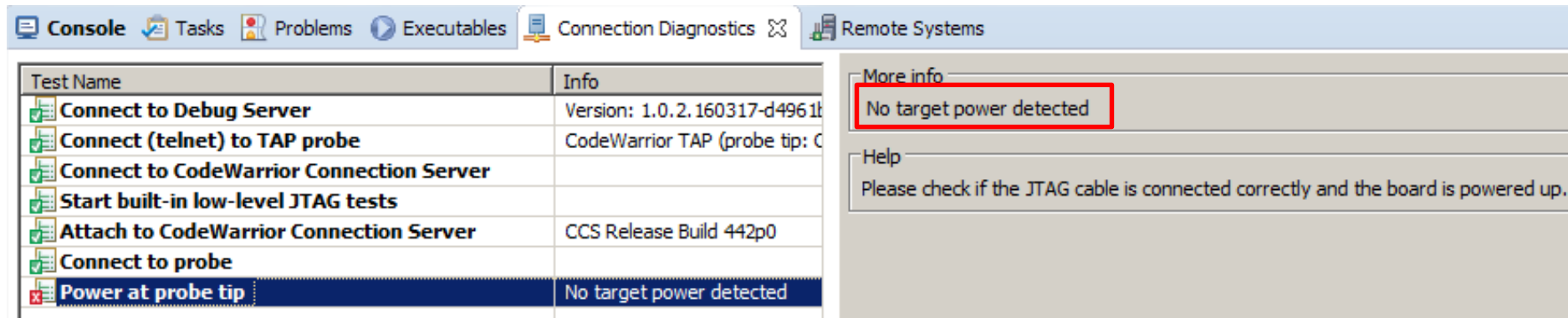


Connect back the USB cable to CWTAP probe to get rid of the failure.



Power at Probe Tip

Introduce a failure – power down the LS2085ARDB board using the PWR button



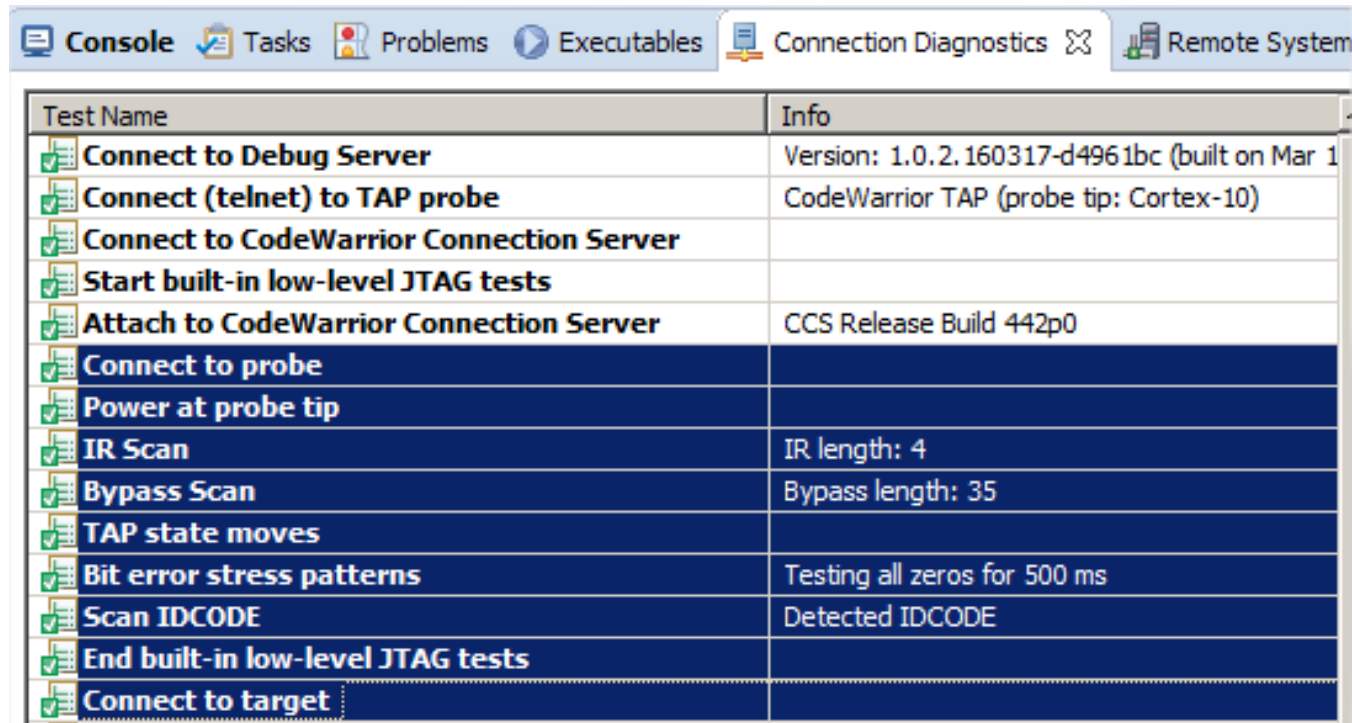
The screenshot shows the 'Connection Diagnostics' window in the CodeWarrior IDE. The window has a tabbed interface with 'Console', 'Tasks', 'Problems', 'Executables', 'Connection Diagnostics', and 'Remote Systems'. The 'Connection Diagnostics' tab is active, displaying a table of test results. The table has two columns: 'Test Name' and 'Info'. The 'Power at probe tip' test is highlighted in blue and shows a red error icon and the message 'No target power detected'. To the right of the table, there is a 'More info' section with a red box around the text 'No target power detected', and a 'Help' section with the text 'Please check if the JTAG cable is connected correctly and the board is powered up.'

Test Name	Info
Connect to Debug Server	Version: 1.0.2.160317-d496 1b
Connect (telnet) to TAP probe	CodeWarrior TAP (probe tip: C
Connect to CodeWarrior Connection Server	
Start built-in low-level JTAG tests	
Attach to CodeWarrior Connection Server	CCS Release Build 442p0
Connect to probe	
Power at probe tip	No target power detected

To get rid of the problem, just power up the board again

Run Low-level JTAG Tests to Check Probe JTAG Reliability

These tests are verifying if the JTAG is reliable using different low-level JTAG tests and also will “scan” the SoC after IDCODE to see if is ARM specific

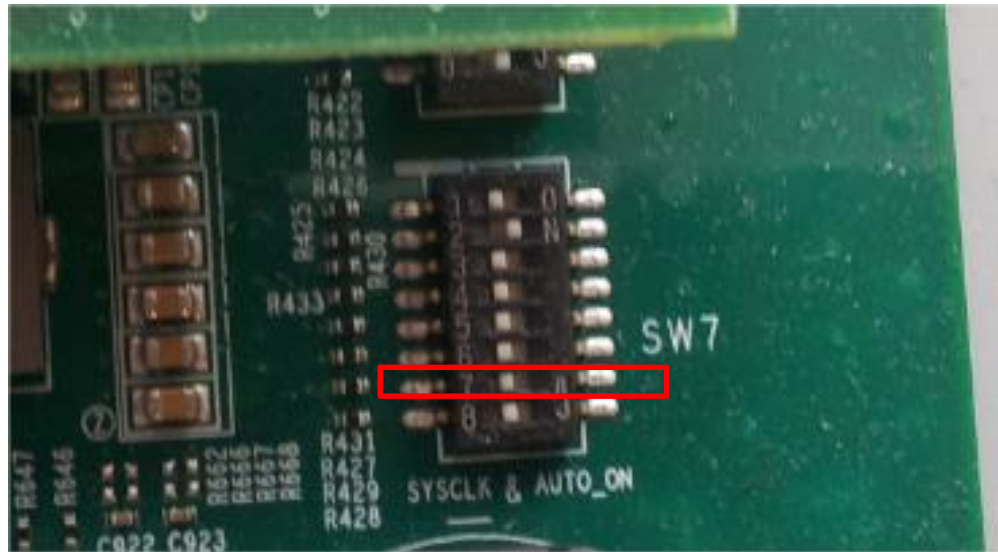


The screenshot shows the CodeWarrior IDE interface with the Console window open. The Console window displays a list of JTAG tests and their status. The tests are listed in a table with columns for Test Name and Info. The tests are: Connect to Debug Server, Connect (telnet) to TAP probe, Connect to CodeWarrior Connection Server, Start built-in low-level JTAG tests, Attach to CodeWarrior Connection Server, Connect to probe, Power at probe tip, IR Scan, Bypass Scan, TAP state moves, Bit error stress patterns, Scan IDCODE, End built-in low-level JTAG tests, and Connect to target. The tests are marked with a green checkmark in the first column, indicating they have been successfully executed. The Info column provides additional details for each test, such as the version of the debug server, the probe tip, the CCS release build, the IR length, the bypass length, the testing duration, and the detected IDCODE.

Test Name	Info
Connect to Debug Server	Version: 1.0.2.160317-d4961bc (built on Mar 1
Connect (telnet) to TAP probe	CodeWarrior TAP (probe tip: Cortex-10)
Connect to CodeWarrior Connection Server	
Start built-in low-level JTAG tests	
Attach to CodeWarrior Connection Server	CCS Release Build 442p0
Connect to probe	
Power at probe tip	
IR Scan	IR length: 4
Bypass Scan	Bypass length: 35
TAP state moves	
Bit error stress patterns	Testing all zeros for 500 ms
Scan IDCODE	Detected IDCODE
End built-in low-level JTAG tests	
Connect to target	

Scan IDCODE (1)

Introduce a failure – change the default ARM DAP (Debug Access Point) mode to NXP SAP (Service Access Point) setting SW7[7] = 0 (OFF)



Scan IDCODE (2)

Test Name	Info
Connect to Debug Server	Version: 1.0.2.160317-d4961bc (built on Mar 17
Connect (telnet) to TAP probe	Skipped - only available for CodeWarrior TAP - Et
Connect to CodeWarrior Connection Server	
Start built-in low-level JTAG tests	
Attach to CodeWarrior Connection Server	CCS Release Build 442p0
Connect to probe	
Power at probe tip	
IR Scan	IR length: 8
Bypass Scan	Bypass length: 1
TAP state moves	
Bit error stress patterns	Testing all zeros for 500 ms
Scan IDCODE	No DAP device found on the JTAG chain

More info

Detected IDCODE

TDO -----

|

* Device 0 IDCODE: 0A01E01D Device: FSL LS2085A rev 1.x

|

TDI -----

No DAP device found on the JTAG chain

```
LS2085A_RDB_FTF
43 0060 00 00 00 00 00 00 00 00
44 0070 00 70 02 00 00 00 00 03
45 0080 00 00 00 00 00 00 00 cb
46 0090 00 00 00 00 00 04 e0 30 00
47 00a0 80 00 00 00 00 00 8f 80 4e
48
49 #-----
50 #  JTAG configuration and setup
51 #-----
52
53 TBSCAN EN B needs to be configured
54 SW7[7] = 1 - TBSCAN EN B is high
55
56 #-----
57 #  GIC version
58 #-----
59
60 Generic Interrupt Controller Archit
61 Implementation: GIC-500
62
```

Target Configuration | Target Initialization File | README

To get rid of the problem change back to default ARM DAP (Debug Access Point) setting SW7[7] = 1 (ON)

ACTIVITY

Flashing U-Boot Image to the Target with CW Flash



FLASH PROGRAMMER



Flashing U-Boot Image to the Target with CW Flash

Summary

- Run Flash Programmer GUI
- Write U-Boot binary into NOR flash

Flashing u-boot image to the Target Activity (1)

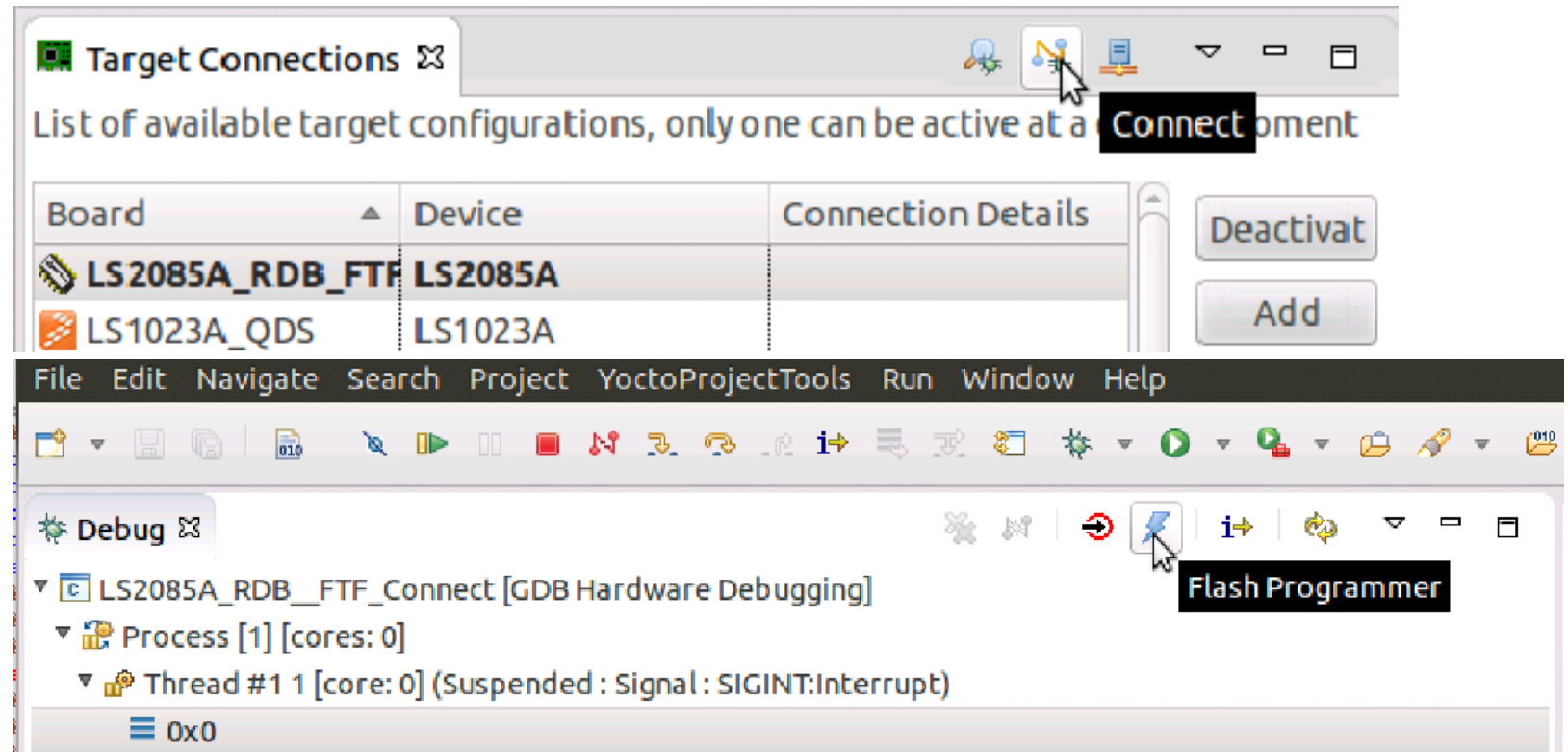
There are possible 3 ways to flash something in flash

1. *Flash Programmer GUI*
2. *GDB CLI*
3. *GDB Eclipse*

To use Flash Programmer GUI:

A. connect to target

B. Click Flash Programmer icon



Flashing U-Boot Image to the Target Activity (2)

The screenshot shows the CodeWarrior Flash Programmer window. The title bar reads "CodeWarrior Flash Programmer". Below the title bar, it says "Perform actions on the flash device".

Annotations and their corresponding elements in the interface:

- Drop-down list with all supported flashes:** Points to the "Devices:" dropdown menu showing "S29GL01GP (NOR)".
- Hover-information about current flash:** Points to a yellow box containing "Device Info" details: base address: 0x58000000, size: 0x8000000 (128.00MB), sectors: 1024, sector size: 0x20000 (128.00KB), geometry: 16x1, features: erase;erase_chip;dump;write;protect;unprotect;.
- Browse for U-Boot image from /home/class/SDK/LS2085A-SDK-20160304-yocto/build_ls2085ardb_release/tmp/deploy/images/ls2085ardb/u-boot-nor.bin:** Points to the "Browse" button next to the "File:" field.
- Check Erase and Unprotect:** Points to the "Unprotect" and "Erase" checkboxes in the "Action:" configuration area.
- Add action:** Points to the "Add Action" button.
- Execute it:** Points to the "Execute Sequence" button (a green play icon).
- Results in FP log:** Points to the log output at the bottom of the window.

The "Action:" configuration area includes: Action: Program, File: C:\uboot.bin, Offset: 0x100000, Size (bytes): 0x87f58, Unprotect (checked), Erase (checked), Protect (unchecked).

The "Action" table shows:

Action	Description
Program	0x87f58 bytes from C:\uboot.bin at 0x100000 with Unprotect Erase

The FP log output is:

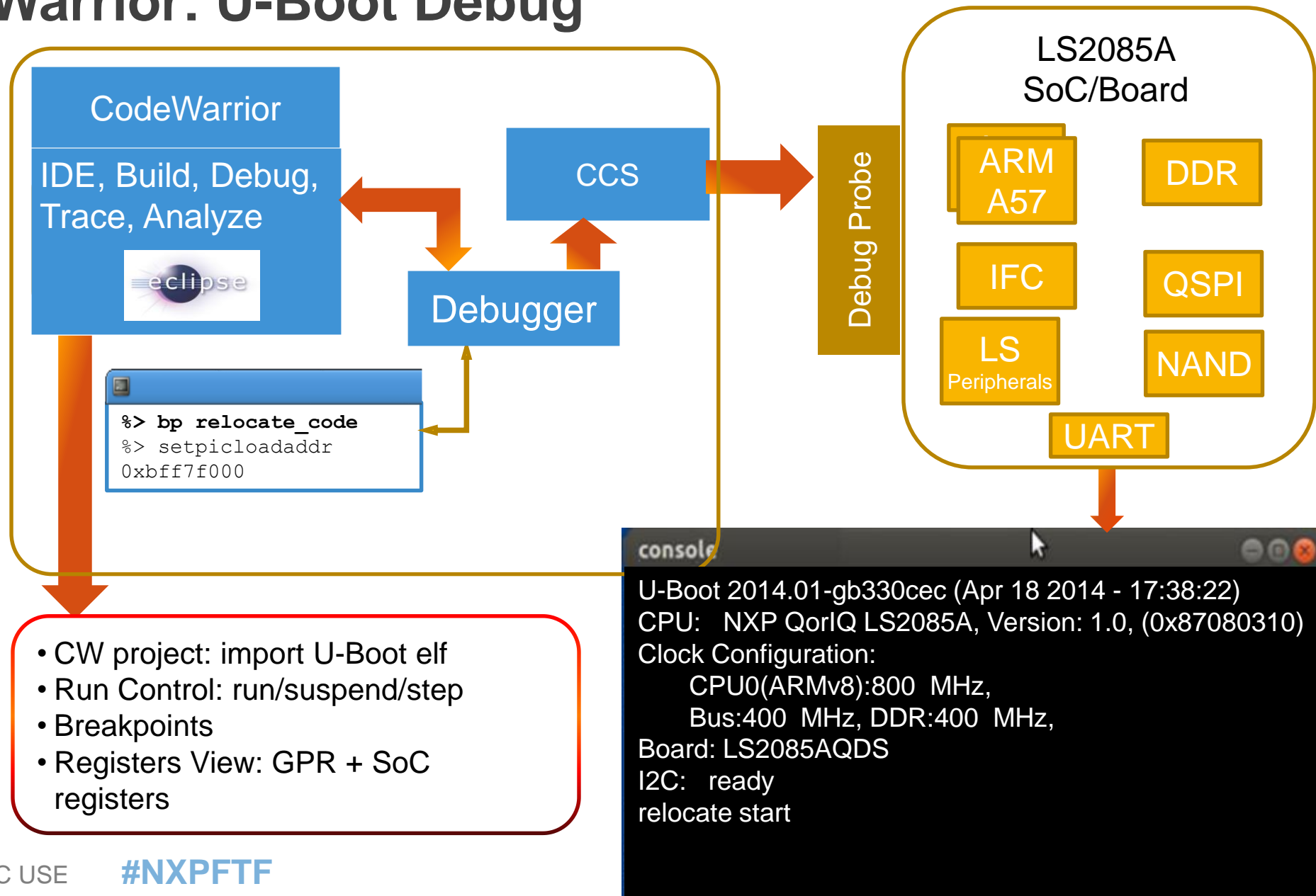
```
Unprotecting...
Unprotect 594.81KB in 0.14s
fi_erase 0x100000 0x94b3d
Erasing...
Erase 594.81KB in 2.52s
fi_write 0x100000 C:\u-boot-nor.bin -s 0x94b3d
Writing...
Write 594.81KB from C:\u-boot-nor.bin in 12.99s
```


Using Flash from Command Line

1. Edit `CW_ARMv8/ARMv8/gdb_extensions/flash/cwflash.py` with your board and connection settings
2. Start GDB console from `CW_ARMv8/ARMv8/gdb/bin/aarch64-fsl-gdb.bat`
 - `cd ../../gdb_extensions`
 - `source flash/cwflash.py`
3. Issue following command:
 - `fl_write --erase 0x100000 {u-boot_image_path}`
 - Wait a few seconds for the confirmation message
 - You are ready to debug U-Boot

U-BOOT DEBUG

CodeWarrior: U-Boot Debug



CodeWarrior: U-Boot Debug

- U-Boot bring-up and debugging
 - Import U-Boot ELF with symbol information
 - Debug from first U-Boot instruction (in flash)
 - Debug after U-Boot relocation in ram / relocate symbols
 - Debug to console prompt
 - Debug to kernel hand-off
- Registers View: GPR + SoC registers
- Debugging features:
 - Run control run/suspend/step
 - Breakpoints, in any ARMv8 EL mode
 - Disassembly, Memory view, Variable View, Expressions
- **Prerequisite**
 - U-Boot image (optionally with symbolic information, useful for source level debug)

ACTIVITY

U-Boot – Debug

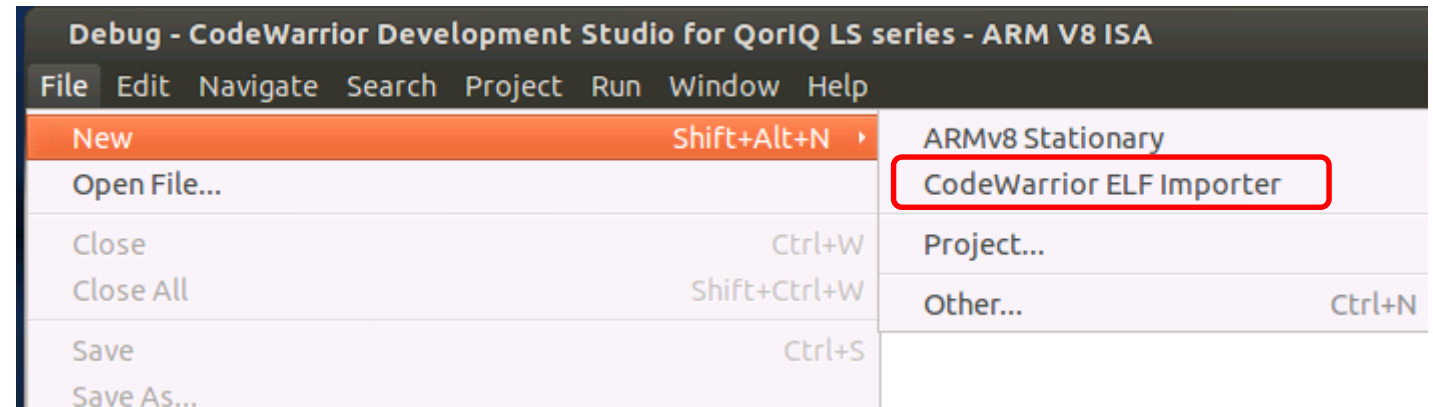


U-Boot Debug – Create Project – Activity

Select File > New >

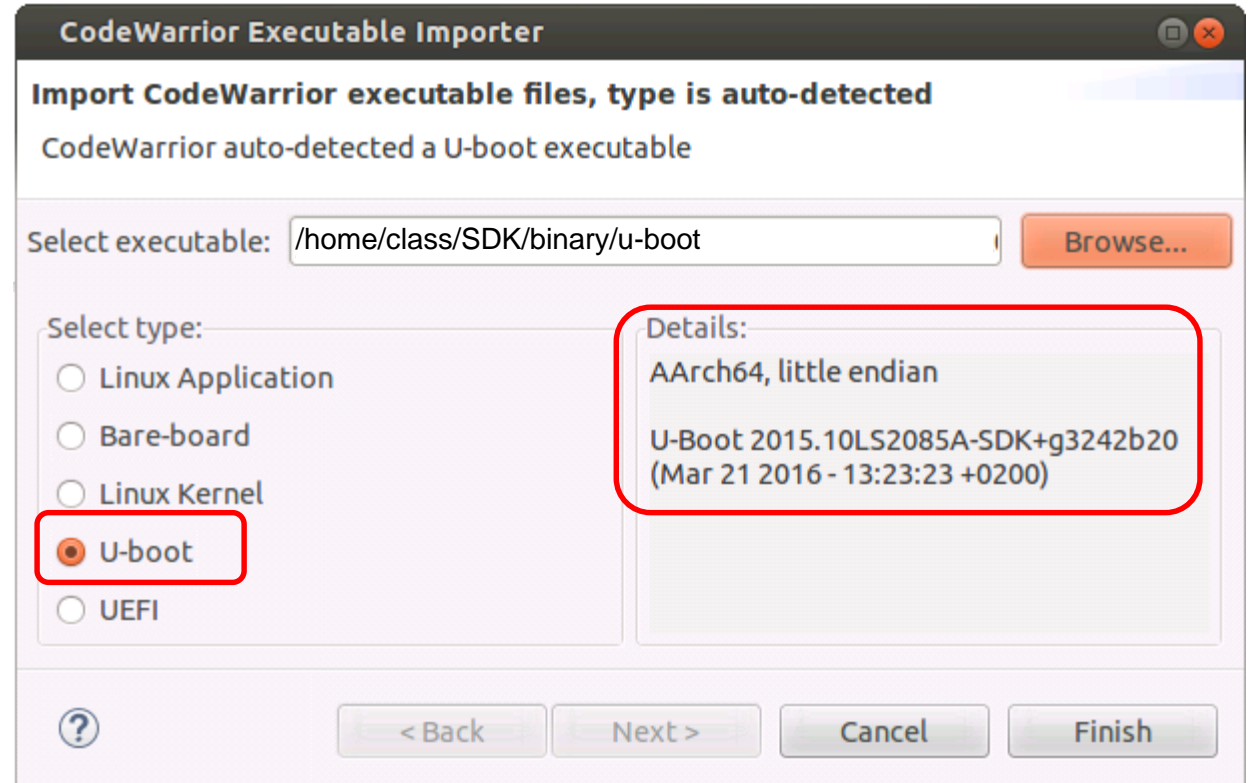
CodeWarrior ELF Importer

- Automatically detects the ELF type and applies the correct awareness settings



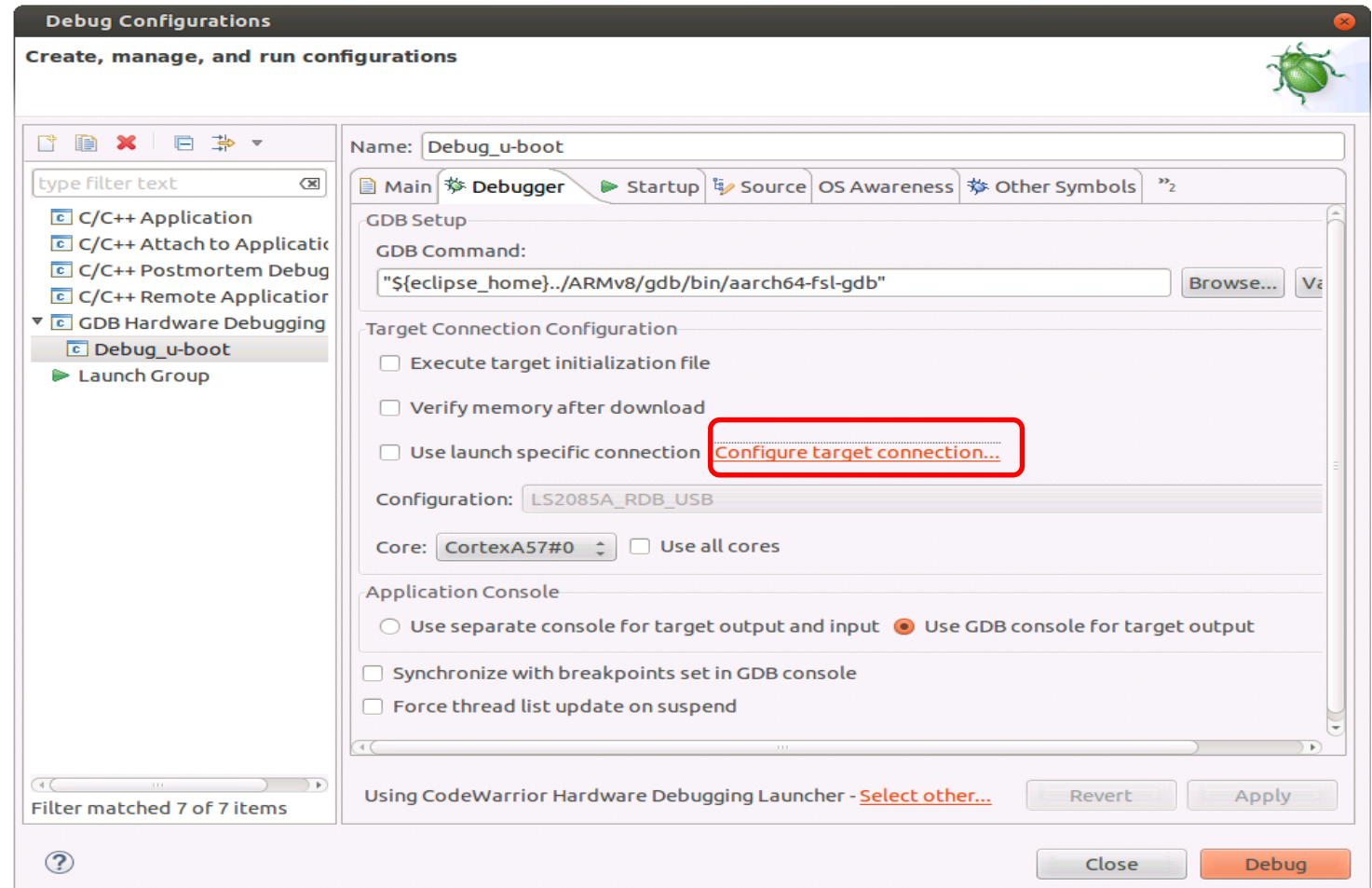
U-Boot Debug – Create Project – Activity

- Click Browse...
- Select your U-Boot file. The file must have debug information.
- CodeWarrior automatically:
 - Detect the ELF type as U-Boot
 - Show U-Boot summary information: U-Boot version, build time, configuration
 - Apply the right debugger settings for debugging U-Boot



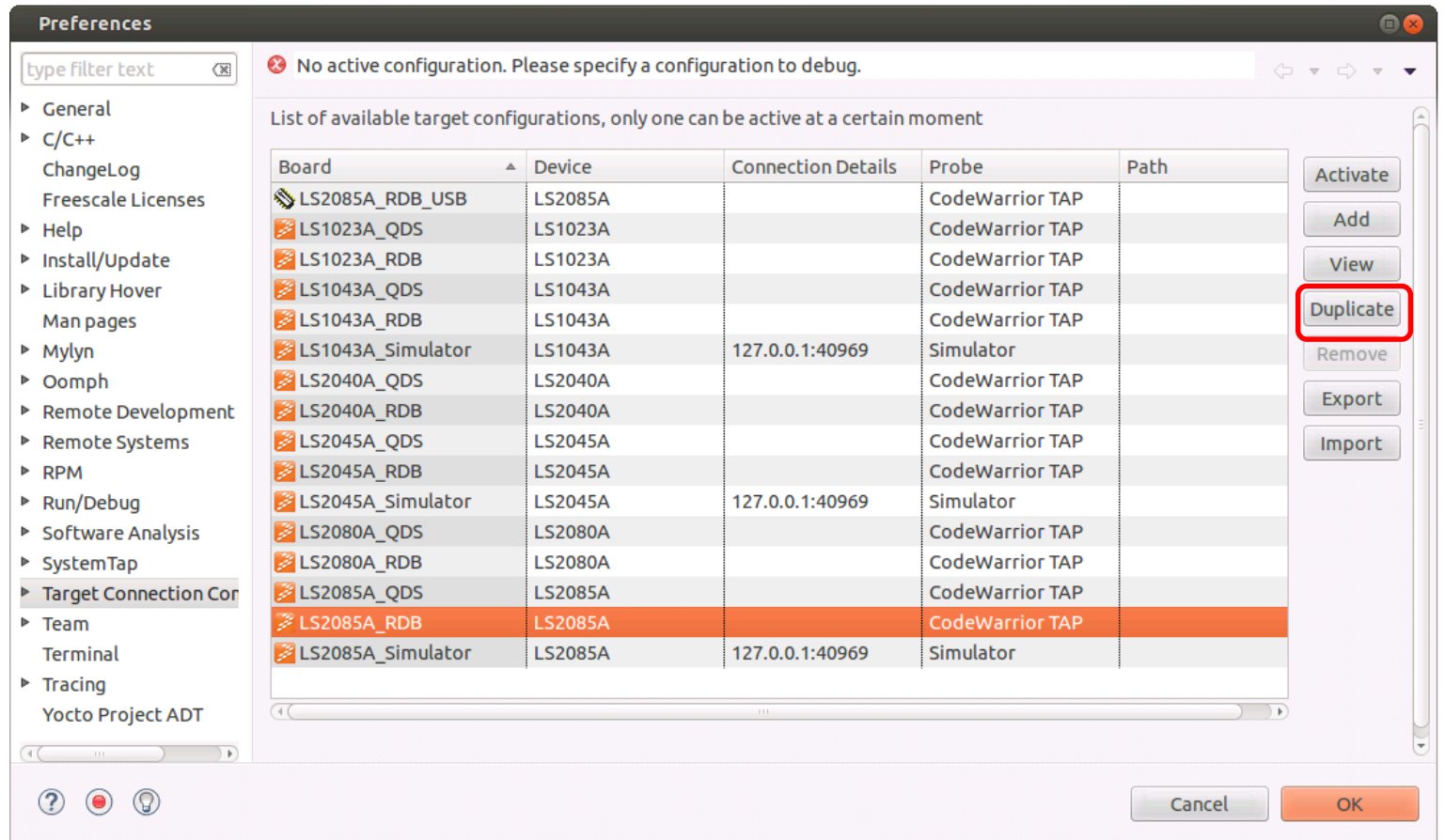
U-Boot Debug – Create Target Connection – Activity

If no target connection is set, configure one following the link “[Configure target connection...](#)”



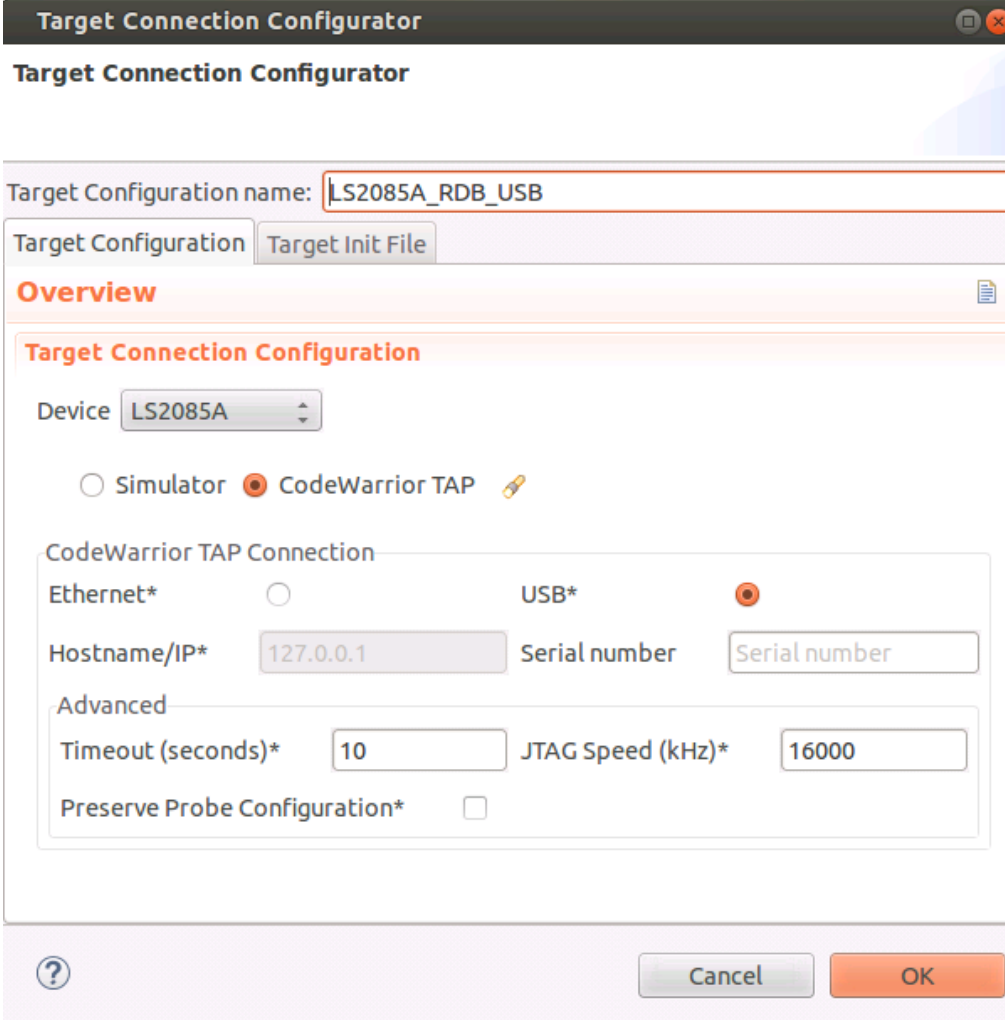
U-Boot Debug – Create Target Connection – Activity

- Select the desired target configuration template (e.g. LS2085A_RDB)
- Press “Duplicate” and set the new name
- Press Edit to change the template configuration



U-Boot Debug – Create Target Connection – Activity

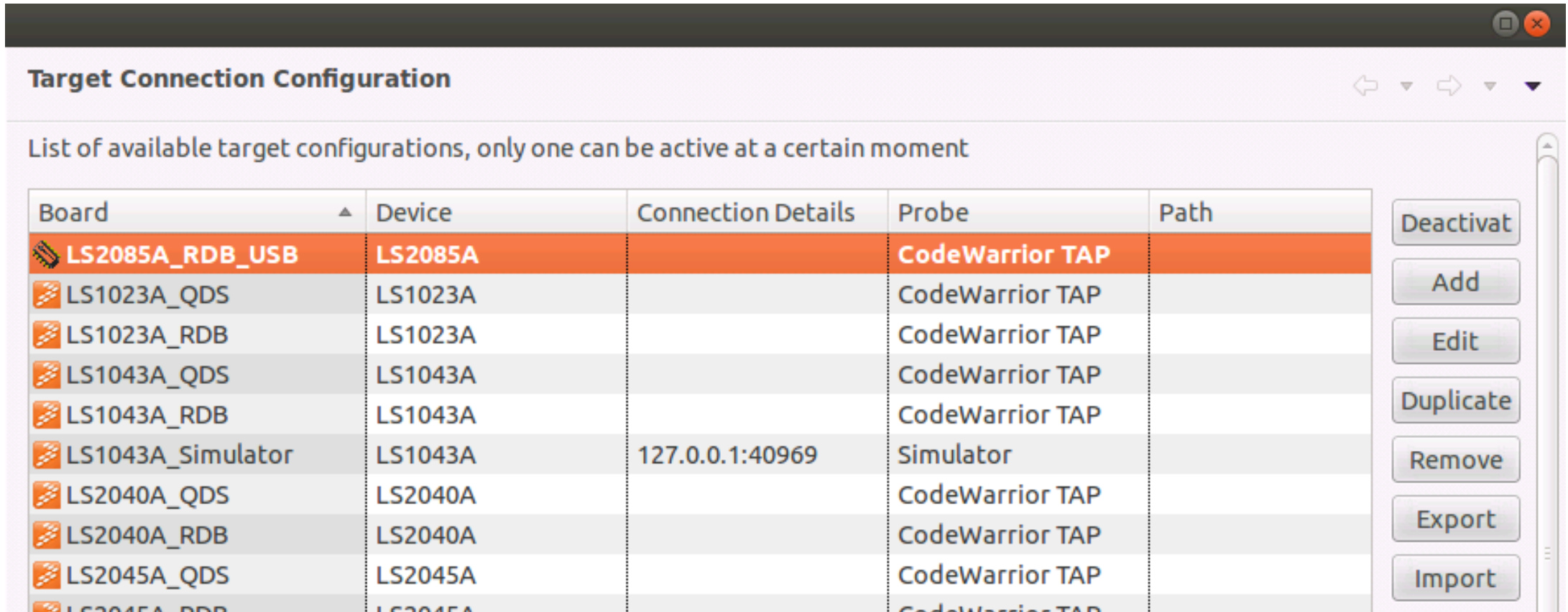
- Edit the name
(different than the
template name)
- Select USB
- Press OK



The screenshot shows the 'Target Connection Configurator' window. The title bar reads 'Target Connection Configurator'. Below the title bar, the window title is 'Target Connection Configurator'. The 'Target Configuration name' field contains 'LS2085A_RDB_USB'. There are two tabs: 'Target Configuration' (selected) and 'Target Init File'. The 'Overview' section is expanded, showing 'Target Connection Configuration'. The 'Device' dropdown is set to 'LS2085A'. Under 'CodeWarrior TAP Connection', the 'USB*' radio button is selected, while 'Ethernet*' is unselected. The 'Hostname/IP*' field contains '127.0.0.1' and the 'Serial number' field contains 'Serial number'. In the 'Advanced' section, the 'Timeout (seconds)*' field contains '10' and the 'JTAG Speed (kHz)*' field contains '16000'. The 'Preserve Probe Configuration*' checkbox is unchecked. At the bottom, there is a help icon (?), a 'Cancel' button, and an 'OK' button.











U-Boot Debug – Create Target Connection – Activity

The newly created target configuration is set active



Target Connection Configuration

List of available target configurations, only one can be active at a certain moment

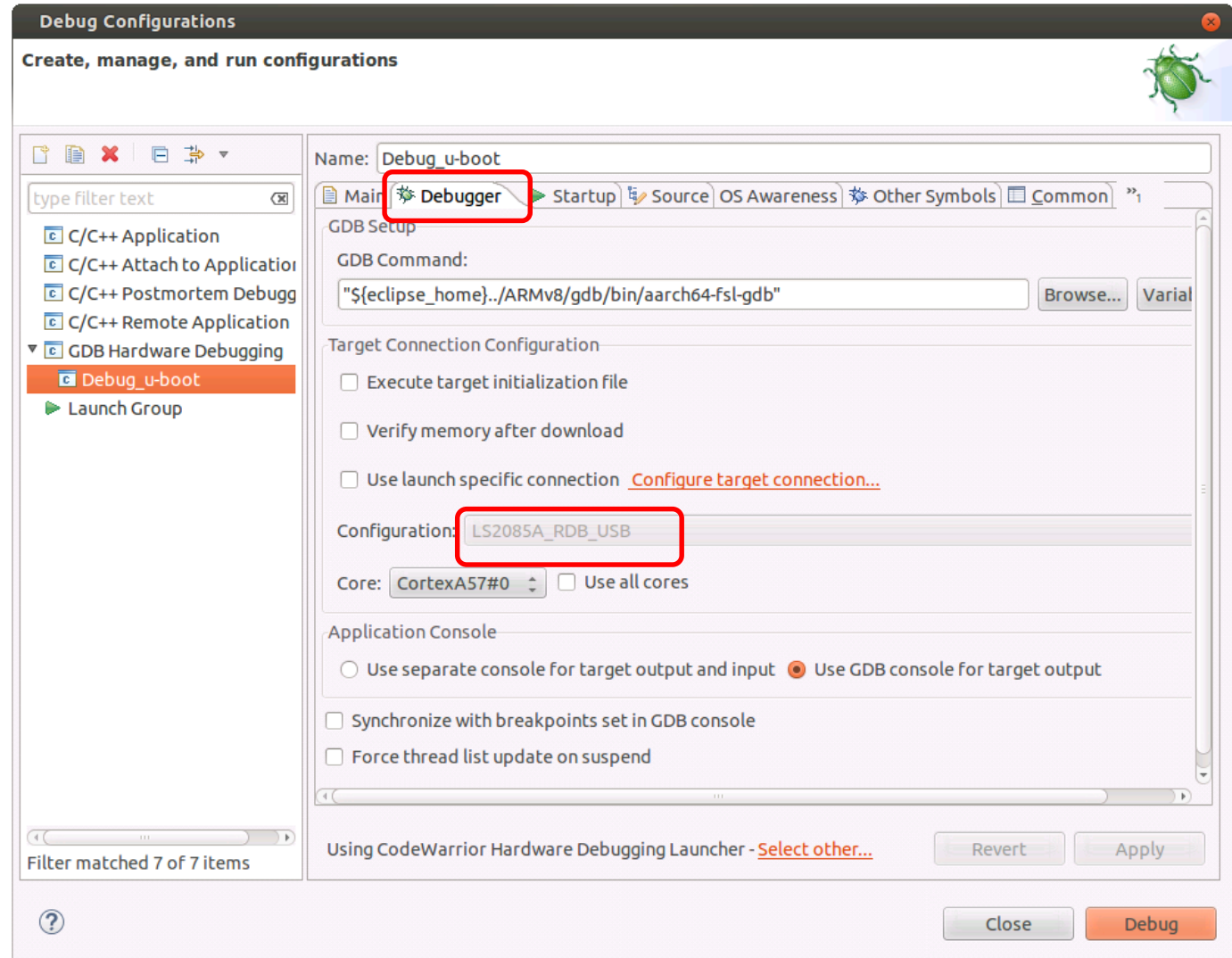
Board	Device	Connection Details	Probe	Path
 LS2085A_RDB_USB	LS2085A		CodeWarrior TAP	
 LS1023A_QDS	LS1023A		CodeWarrior TAP	
 LS1023A_RDB	LS1023A		CodeWarrior TAP	
 LS1043A_QDS	LS1043A		CodeWarrior TAP	
 LS1043A_RDB	LS1043A		CodeWarrior TAP	
 LS1043A_Simulator	LS1043A	127.0.0.1:40969	Simulator	
 LS2040A_QDS	LS2040A		CodeWarrior TAP	
 LS2040A_RDB	LS2040A		CodeWarrior TAP	
 LS2045A_QDS	LS2045A		CodeWarrior TAP	
 LS2045A_RDB	LS2045A		CodeWarrior TAP	

Deactivat
Add
Edit
Duplicate
Remove
Export
Import

U-Boot Debug – Other Project Settings – Activity

Debugger Tab

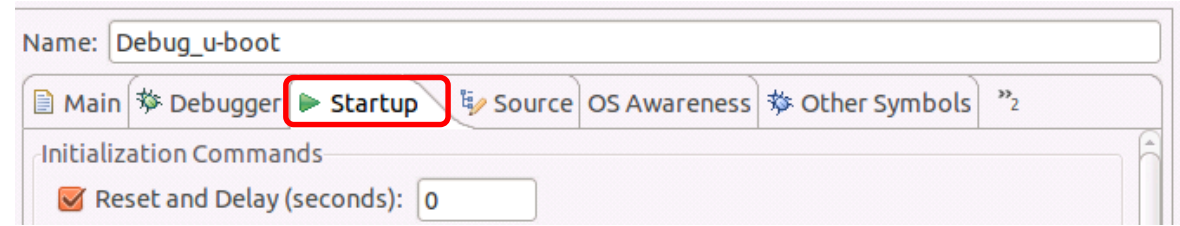
- Configuration automatically set to the default Target Connection



U-Boot Debug – Other Project Settings – Activity

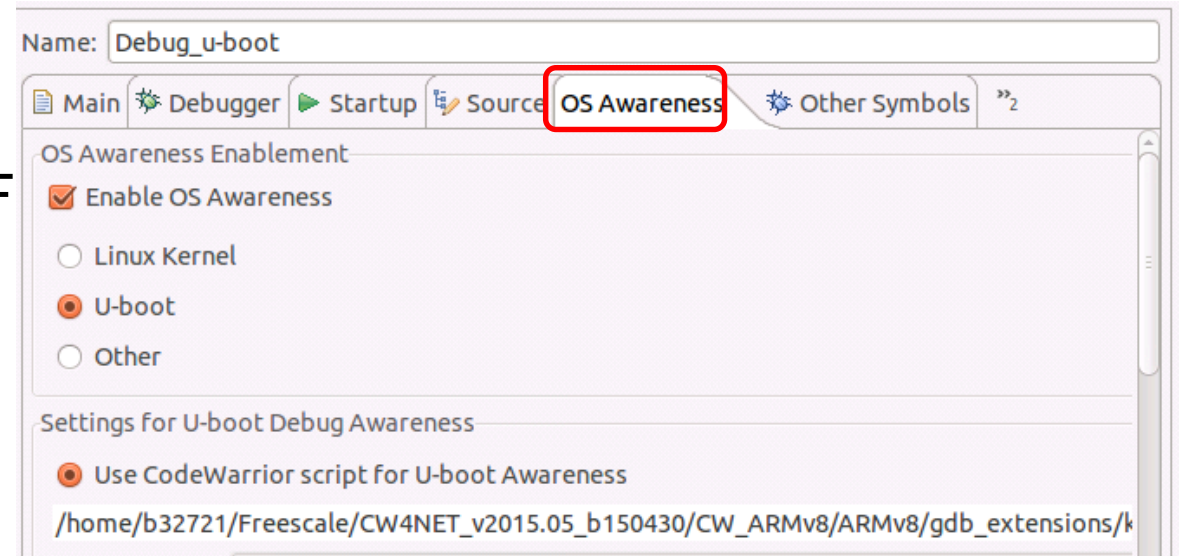
Startup Tab

- Uncheck Reset and Delay to attach to the current running U-Boot
- Check Reset and Delay = 0 to reset the target and debug U-Boot from the first instruction after reset



OS Awareness Tab

- U-Boot awareness settings have been automatically applied when CodeWarrior ELF Importer detects the U-Boot executable



U-Boot Debug – Debug Overview – Activity

- **U-Boot Awareness**

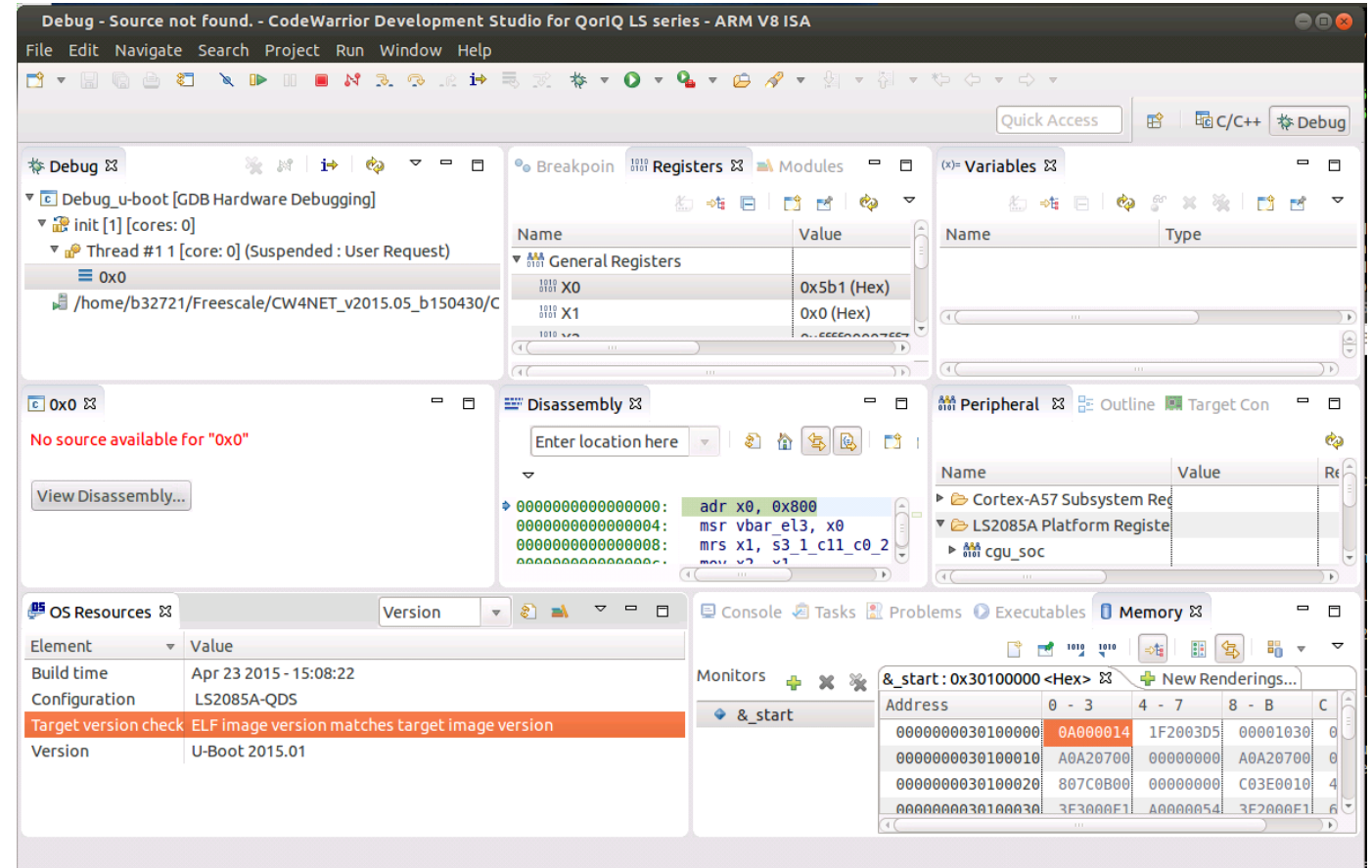
- A single U-Boot debug session while **the user is not aware about any stages or relocation offsets.**
- The debugger automatically detects each U-Boot stage and performs the corresponding action.
- The user can visualize meaningful U-Boot information about: U-Boot version, build time or memory information
- Target image vs. ELF image version check

- Demonstrate a full U-Boot debug session

- Debug from the first instruction after reset, to U-Boot entry point, running from Flash, after relocation to DDRAM and until U-Boot prompt is available
- All is done in a single debug session with no other changes

U-Boot Debug – Debug – Activity

- Click Launch Configuration > “Debug” button to start the debug session
- The target is reset and the debugger stops at the first instruction after reset
 - PC is 0x0 – BootRom
- No symbols or source file view available in U-Boot ELF – the U-Boot code has not been started yet



U-Boot Debug – Debug – Activity

- To demonstrate full U-Boot debug in a single session, set several breakpoints in some key points of booting process
- Set breakpoints from console or from file:
 - `_start`: U-Boot entry point
 - `relocate_code`: when running from Flash and relocation to DDRAM is started
 - `relocation_return`: first function executed from DDRAM

The screenshot displays a GDB debugging session for U-Boot. The interface is divided into several panels:

- Debug**: Shows the current thread as 'Thread #1 1 [core: 0] (Suspended: User Request)' at address '0x0' in the file '/home/b32721/Freescale/CW4NET_v2015.05_b1'.
- Breakpoints**: Lists three active hardware breakpoints:
 - Breakpoint 1 at address 0x00000000301033b8 (type: Hardware).
 - Breakpoint 2 at address 0x30100000 (file: arch/arm/cpu/armv8/start.S, line 23).
 - Breakpoint 3 at address 0x3010330c (file: arch/arm/lib/relocate_64.S, line 24).
 - Breakpoint 4 at address 0x301032c4 (file: arch/arm/lib/crt0_64.S, line 97).
- Source Code**: Shows the assembly code for the `relocate_code` function in `relocate_64.S`. The current instruction is `mov x29, sp` at line 24.
- Disassembly**: Shows the disassembly of the current instruction, including `adr x0, 0x800`, `msr vbar_el3, x0`, and `mrs x1, s3_1_c1`.
- Console**: Shows the GDB command `break start` and the resulting breakpoint list.

U-Boot Debug – Debug – Activity

- Entry point breakpoints hit
- Note: breakpoint set as SW breakpoint (default) to a Flash (read-only) address. U-Boot Awareness automatically changes the breakpoint type to HW for Flash address.

The screenshot displays the GDB IDE interface with several panels:

- Debug:** Shows the execution state: `Debug_u-boot [GDB Hardware Debugging]`, `init [1] [cores: 0]`, `Thread #1 1 [core: 0] (Suspended : Breakpoint)`, and `_start() at start.S:23 0x30100000`. The file path is `/home/b32721/Freescale/CW4NET_v2015.05_b1...`.
- Registers:** A table showing the state of general registers:

Name	Value	Description
X0	0x320 (Hex)	General Purpose
X1	0x4280000 (Hex)	General Purpose
X2	0x1e00000 (Hex)	General Purpose
- Disassembly:** Shows the assembly code at the breakpoint:

```
start:  
b 0x30100028 <reset>  
nop  
_TEXT_BASE:
```
- Console:** Displays the GDB startup message:

```
Debug_u-boot [GDB Hardware Debugging] /home/b32721/Freescale/CW4NET_v2015.05_b150430/CW_ARMv8/ARMv8/g  
<http://www.gnu.org/software/gdb/bugs/>.  
Find the GDB manual and other documentation resources online at:  
<http://www.gnu.org/software/gdb/documentation/>.  
For help, type "help".  
Type "apropos word" to search for commands related to "word".
```

A red box highlights the following note in the console output: `Note: automatically using hardware breakpoints for read-only addresses.`

U-Boot Debug – Debug – Activity

- Start relocation breakpoint hit
- U-Boot running from Flash (address 0x3010xxxx) prepare for running from DDRAM
- Booting messages on U-Boot console

```
Debug
└─ Debug_u-boot [GDB Hardware Debugging]
  └─ init [1] [cores: 0]
    └─ Thread #1 1 [core: 0] (Suspended : Breakpoint)
      └─ relocate_code() at relocate_64.S:24 0x301033
        /home/b32721/Freescale/CW4NET_v2015.05_b1:

relocate_64.S
start.S
22 ENTRY(relocate_code)
23     stp x29, x30, [sp, #-32]! /* crea
24     mov x29, sp
25     str x0, [sp, #16]
26     /*
27     * Copy u-boot from flash to RAM
28     */
```

```
U-Boot 2015.01 (Apr 23 2015 - 15:08:22) LS2085A-QDS

Clock Configuration:
CPU0(A57):1600 MHz CPU1(A57):1600 MHz CPU2(A57):1600 MHz
CPU3(A57):1600 MHz CPU4(A57):1600 MHz CPU5(A57):1600 MHz
CPU6(A57):1600 MHz CPU7(A57):1600 MHz
Bus: 600 MHz DDR: 1333.333 MHz DP-DDR: 1333.333 MHz

Reset Configuration Word (RCW):
00: 40282830 40400040 00000000 00000000
10: 00000000 00000000 00200000 00000000
20: 00c12980 00002580 00000000 00000000
30: 00000003 00000000 00000000 00000000
40: 00000000 00000000 00000000 00000000
50: 00000000 00000000 00000000 00000000
60: 00000000 00000000 00027000 00000000
70: 3f030007 00050000 00000000 00000000

Board: LS2085A-QDS, Board Arch: V1, Board version: B, boot from vBank: 0
FPGA: v5 (LS2085AQDS_2015_0218_1606), build 132 on Wed Feb 18 22:06:38 2015
SERDES1 Reference : Clock1 = 156.25MHz Clock2 = 100 separate SSCGMHz
SERDES2 Reference : Clock1 = 100 separate SSCGMHz Clock2 = 100 separate SSCGMHz
I2C: ready
DRAM: Initializing DDR...using SPD
Detected UDIMM 18ASF1G72AZ-2G1A1
Detected UDIMM 18ASF1G72AZ-2G1A1
DP-DDR: Detected UDIMM 18ASF1G72AZ-2G1A1
19.5 GiB
DDR 15.5 GiB (DDR4, 64-bit, CL=9, ECC on)
DDR Controller Interleaving Mode: 256B
DDR Chip-Select Interleaving Mode: CS0+CS1
DP-DDR 4 GiB (DDR4, 32-bit, CL=9, ECC on)
DDR Chip-Select Interleaving Mode: CS0+CS1
```

U-Boot Debug – Debug – Activity

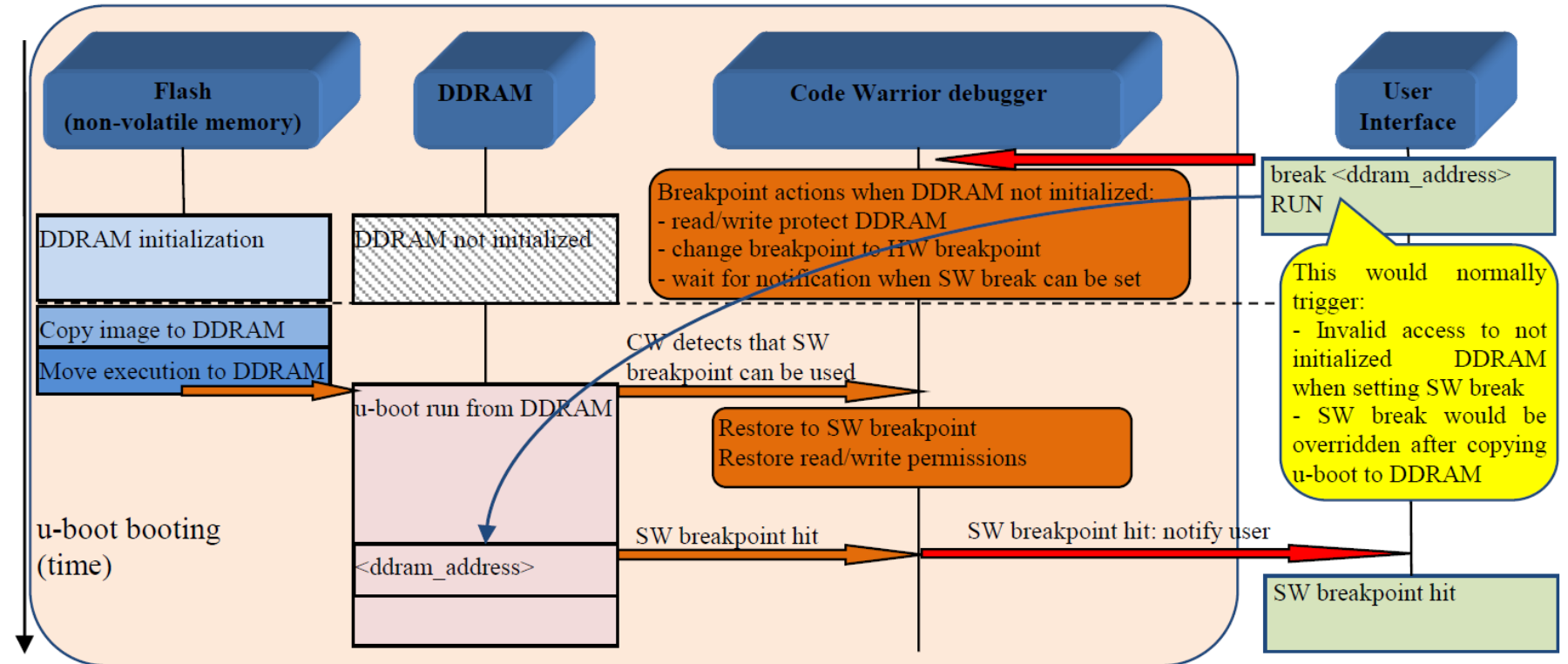
- U-Boot running from DDRAM (relocation_return breakpoint hit)
- DDRAM Address 0xFFF3xxxx
- U-Boot Awareness breakpoint hit (see console) – debugger automatically handles the symbols relocation for DDRAM

The screenshot displays a debugger interface with several panels:

- Debug**: Shows the execution state of 'Debug_u-boot [GDB Hardware Debugging]' with a thread suspended at `_main() at crt0_64.S:97 0xffff3b2c4`.
- Breakpoints**: A table showing general registers X0 and X1 with values `0xffffa7e80 (Hex)` and `0xffffa7e60 (Hex)`.
- Disassembly**: Shows assembly instructions starting with `bl 0xffff38138 <c runtime cpu` at address `00000000fff3b2c4`.
- Console**: Shows the execution flow, including a breakpoint hit at `_start () at arch/arm/cpu/armv8/start.S:23` and the message `U-boot Awareness: u-boot relocation breakpoint is hit`.

U-Boot Debug – Debug – Activity

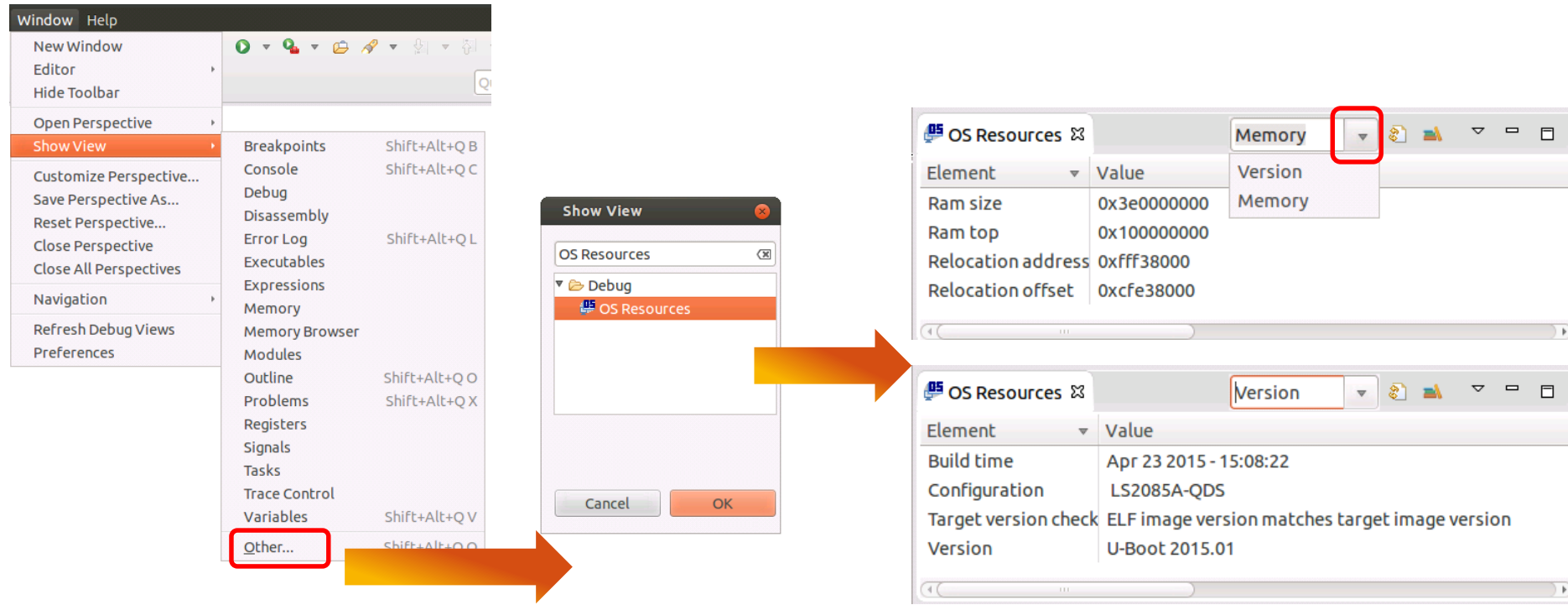
- U-Boot Awareness allows setting SW breakpoints to DDRAM before DDRAM initialization and before U-Boot relocation to DDRAM



U-Boot Debug – U-Boot information – Activity

OS Resources: Windows > Show View > Other > Debugger > OS Resources

- Visualize meaningful U-Boot information about: U-Boot version, build time or memory information
- Target image vs. ELF image version check



U-Boot Debug – Register / Peripheral – Activity

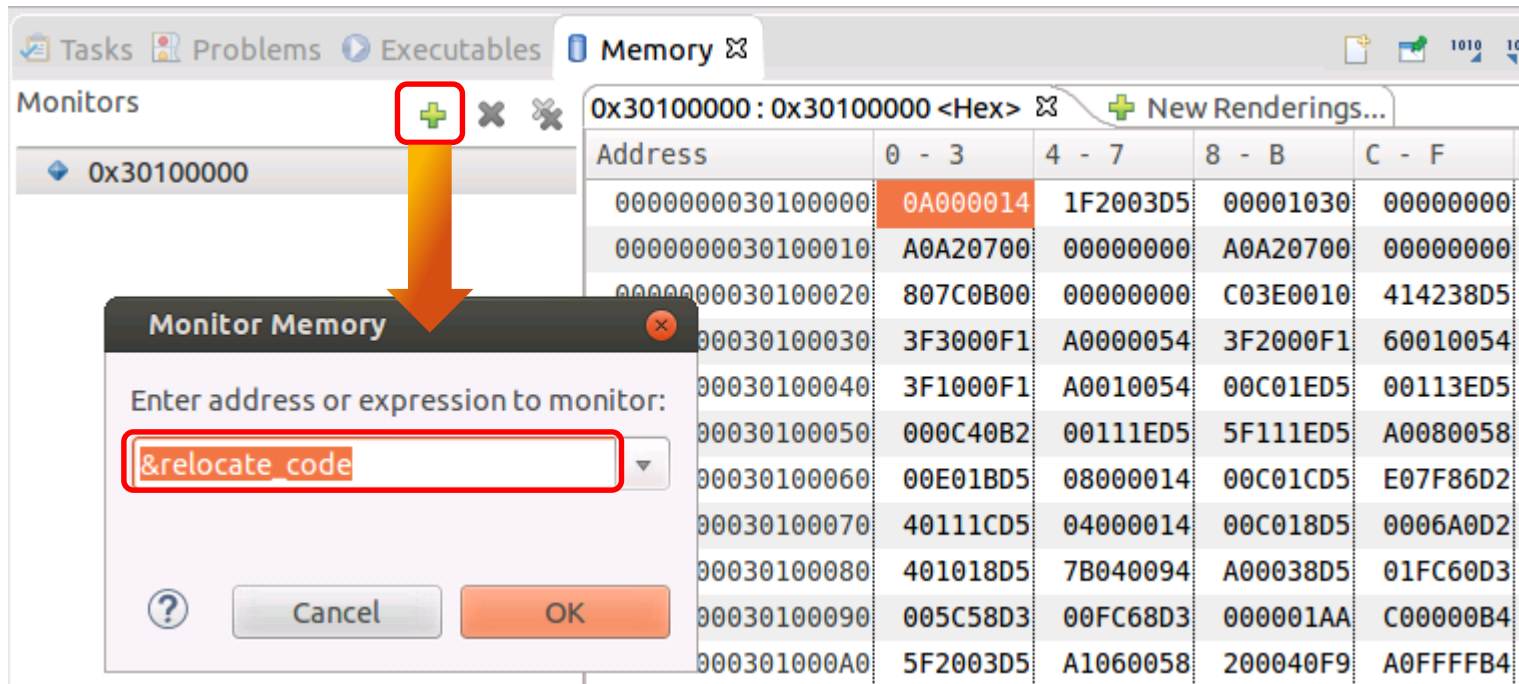
- Register view
- Peripheral: Windows > Show View > Other > Debugger > Peripherals

Name	Value
General Registers	
X0	0xffffa7e80 (Hex)
X1	0xffffa7e60 (Hex)
X2	0x40 (Hex)
X3	0x3f (Hex)
X4	0xffff9c100 (Hex)
X5	0xe7ff069f (Hex)
X6	0xffffffd0 (Hex)
X7	0x20 (Hex)
X8	0x10 (Hex)
X9	0xcfe38000 (Hex)
X10	0x3 (Hex)
X11	0x30164100 (Hex)
X12	0x80000002 (Hex)
X13	0x80000002 (Hex)
X14	0x80000002 (Hex)
X15	0x1800fad0 (Hex)
X16	0x1800f540 (Hex)
X17	0x1 (Hex)

Name	Value	Reset	Acces	Location	Description
Cortex-A57 Subsystem Registers					
Core					
PSTATE					
Special_Purpose					
ID					
System					
Exception					
Memory					
Thread					
Timer					
Reset					
Debug					
External_Debug_Interface					
PERF					
LS2085A Platform Registers					
cgu_soc				0x116000	
cop_dcfg				0x100000	
cop_rst				0x1200b0	
cop_tap				0x3	
dbg_cdtc_wrapper_dbg				0x4000	
dbg_gdi				0x78000	
dbg_sdtc_wrapper_dbg				0x84000	
DDR1				0x1080000	
DDR1_CS0_BNDS	0x000003ff	0x0	RW	0x1080000	Chip select 0 memory bounds
SA	0x0	0x0	RW	[31:16]	Starting address for chip select (bank) n. This value is compared against
EA	0x3ff	0x0	RW	[15:0]	Ending address for chip select (bank)n. This value is compared against
DDR1_CS1_BNDS	0x000003ff	0x0	RW	0x1080008	Chip select 1 memory bounds
DDR1_CS2_BNDS	0x00000000	0x0	RW	0x1080010	Chip select 2 memory bounds

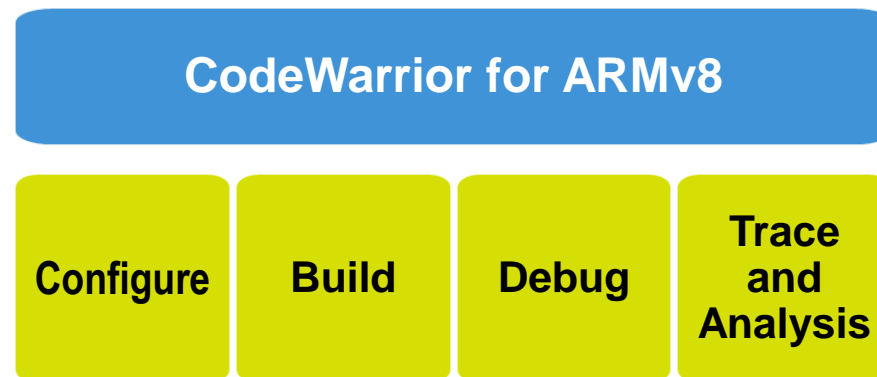
U-Boot Debug – Register / Peripheral – Activity

- Memory view: Window > Show View select “Memory”
- Add memory monitor
- Enter the address/expression



Summary

- This course has been a brief introduction into the LS2085 RDB board and the CodeWarrior tools available to debug the board
- Flash Programmer
- Connection Diagnostics
- U-Boot debug
- Digital Networking is introducing a new networking tools suite
 - CodeWarrior Development Studio for QorIQ LS Series – ARMv8 ISA
 - Tools covering Configuration, Build, Debug, and Analysis



Q & A



SECURE CONNECTIONS
FOR A SMARTER WORLD

ATTRIBUTION STATEMENT

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, CoolFlux, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE Classic, MIFARE DESFire, MIFARE Plus, MIFARE Flex, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TrenchMOS, UCODE, Freescale, the Freescale logo, AltiVec, C 5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and μ Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. 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. © 2015–2016 NXP B.V.

PUBLIC USE

Company Public information is information that has been declared public knowledge by the owner and can freely be given to anyone inside or outside NXP without any possible damage to NXP.

BACKUP



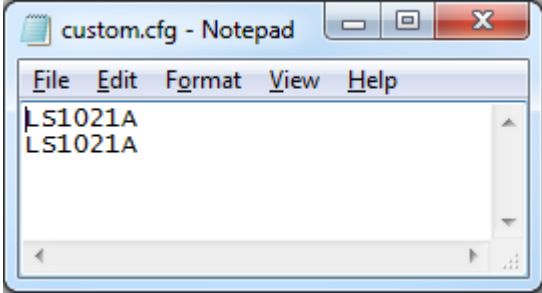
CUSTOM BOARD SETUP



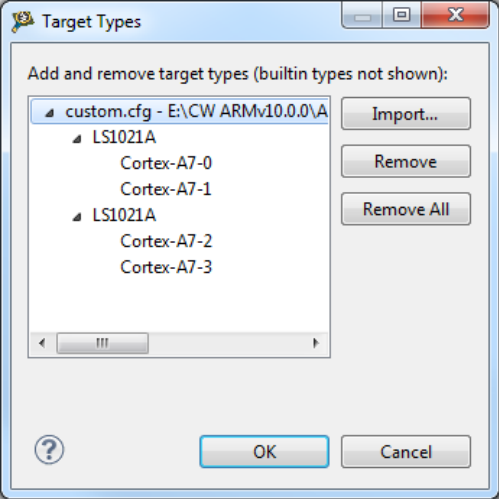
Stock NXP Boards vs. Customer Designs

- For stock NXP boards, CW comes with all the necessary initialization / configuration files
- For custom designs, first, using a JTAG configuration file, define the list of devices on the JTAG chain.
 - As example, for a board with two QorIQ LS2085A processors, JTAG configuration file is:

Import it.



```
LS1021A
LS1021A
```

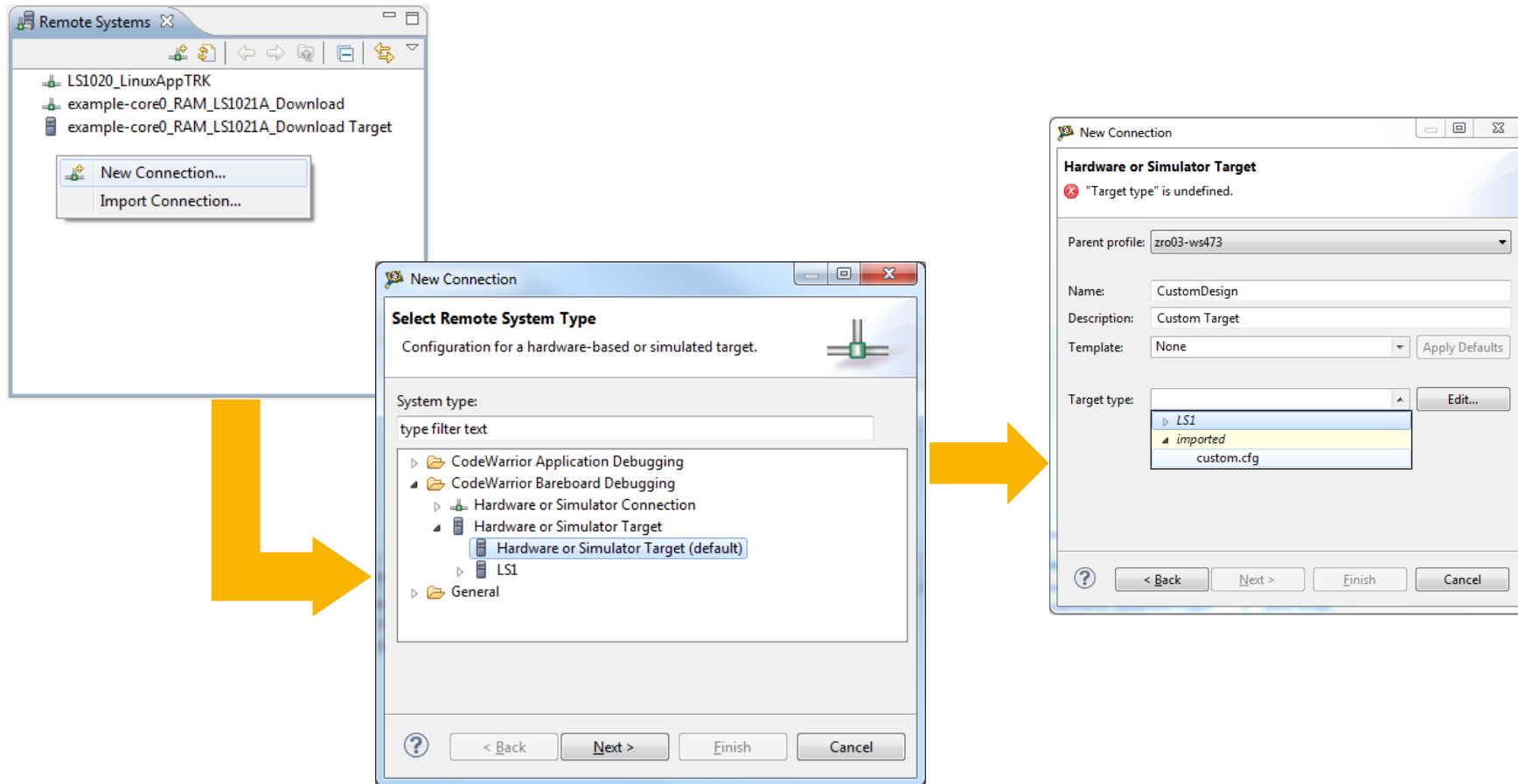


The Target Types dialog box shows the following structure:

- custom.cfg - E:\CW ARMv10.0.0\A
 - LS1021A
 - Cortex-A7-0
 - Cortex-A7-1
 - LS1021A
 - Cortex-A7-2
 - Cortex-A7-3

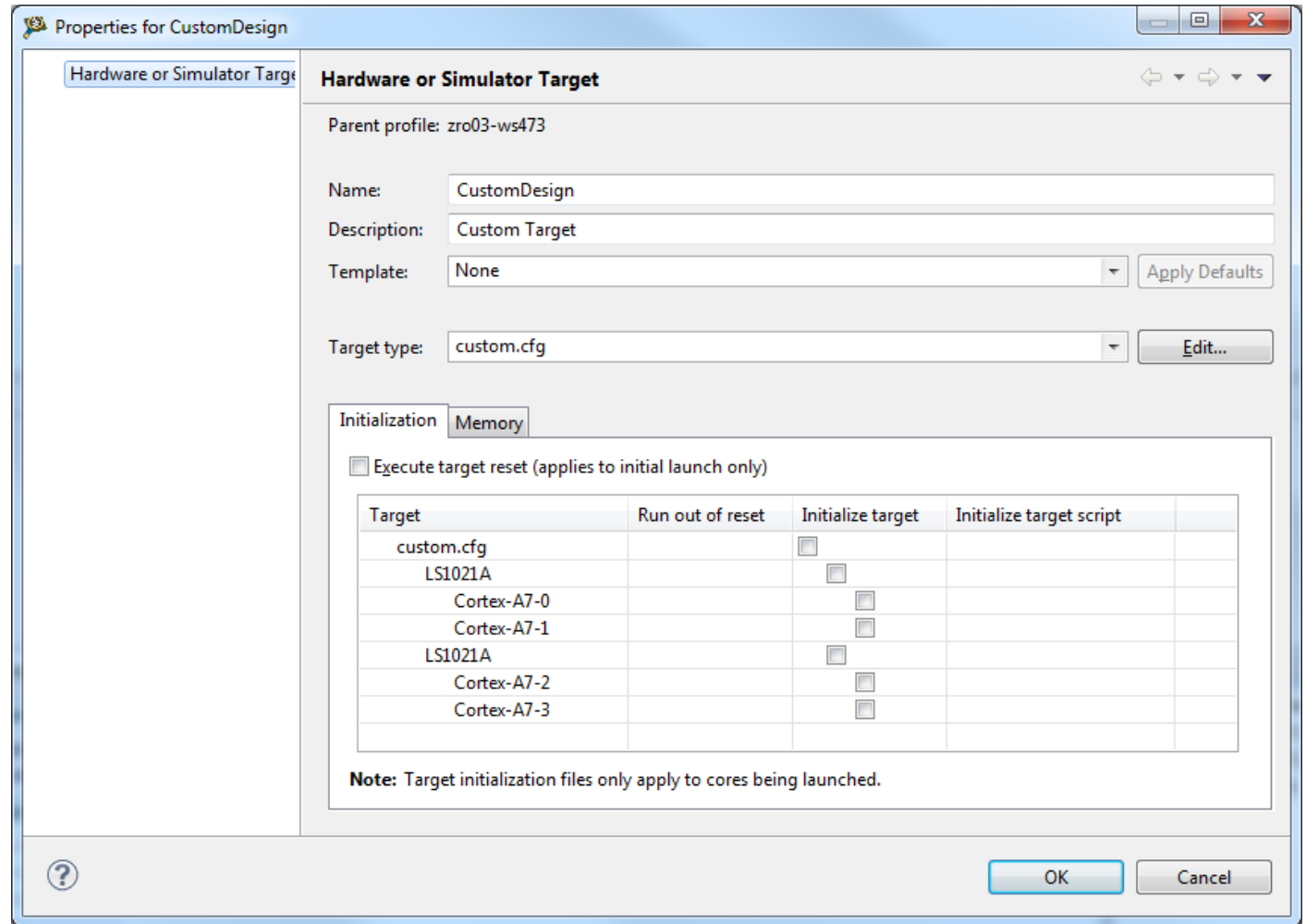
Creating a Custom Target System

- Create a new target system



Configure the Newly Created Target

- Check “Execute target reset” if you want to reset the SoC when first core connects.
- Select initialization files (for DDR, internal memories, flash controllers, etc.) and on what core(s) they should be executed.
- Select memory configuration files. Consider that the memory map may differ since, for example, some IP blocks may not be present in the new design.



New Connection and Debug

- Create a new connection and select for it the “Target” system that was just created.
- Apply the new connection to your project(s)

