BOOST YOUR PRODUCTIVITY WITH CODEWARRIOR® DEVELOPMENT TOOLS FOR ARM®V8 ARCHITECTURE

MARLAN WINTER

QORIQ SW & SERVICES

AMF-DES-T2676 | AUGUST 2017

SECURE CONNECTIONS FOR A SMARTER WORLD

NXP and the NXP logo are trademarks of NXP B.V. All other product or service names are the property of their respective owners. © 2017 NXP B.V.

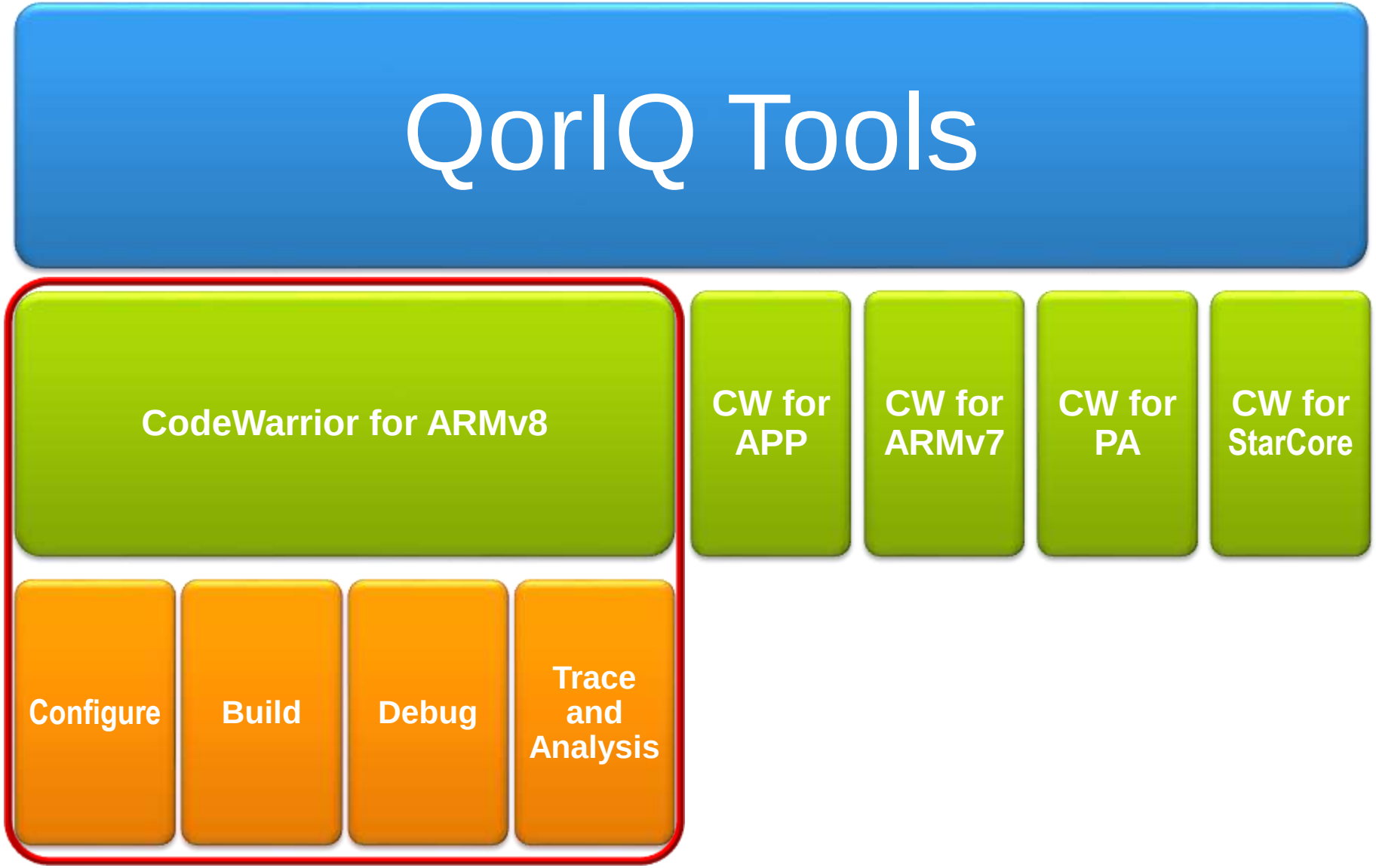
PUBLIC

AGENDA

- CodeWarrior Development Suite Overview
- Open source
- Configuration & validation Tools
  - Board bring-up QCVS
  - Flash programming
- Debug
  - Bare metal debug
  - U-boot debug
  - Linux app debug
  - Linux Kernel & module debug
- Analyze

Overview

CodeWarrior Family



CodeWarrior Development Studio

A Complete Development Environment Under Eclipse

- **Eclipse IDE**
  - Configuration Wizards
  - Plug-In Architecture
  - 3rd party community

- **Build Tools**
  - C/C++ Compiler
- **Configuration & 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
  - OS / Linux awareness
- **Software Analysis - Trace & Profile**
  - Leverages chip capabilities
    - Profiling Unit
    - In system trace buffering
  - Trace / Code / Performance Viewer
  - Offline trace visibility

CodeWarrior Usage Scenarios

- CodeWarrior Development Suite targets NXP’s network devices.
- SoC and board bring-up
  - Device and Linux configuration tools
  - Device configuration validation and optimization tools
  - Single core and multi-core bare-metal debugger
  - Eclipse GUI and command-line interfaces
  - SoC register details from the reference manual
  - Bare-metal utilities: Flash Programmer, Connection Diagnostics
  - Bundled with ARM EABI toolchain (Linaro) for bare-metal application development

CodeWarrior Usage Scenarios

- Linux development
  - GNU debugger compatible + extensions for Linux application debug
  - SMP aware kernel debugging
  - Linux kernel module development and debug
  - Aligned with NXP SDK: Linaro GNU toolchain, integrated target debug agent
- Non-intrusive debug through trace
  - Core and SoC trace sources: configuration, extraction, visibility
  - Post-mortem debugging: offline trace
  - Debug-print over STM
  - Linux aware trace
- Performance Analysis
  - Platform counter configuration and display
  - Predefined measurement scenarios

Open source

Components of CW Tool Suite

- **GDB:** is used in debugger
- **Debug Interface**: Connection from the host to the target for debug
  - Linux: UART or socket
  - QorIQ-LS products bare-metal: CW-TAP
- **Eclipse for C/C++**: the IDE and framework to edit, build and debug projects.
- **GNU ARM Eclipse plugins**: integration of the GNU gcc for ARM into Eclipse, provides panels and build tool integration including a wizard to create new projects.
- **GCC ARM Embedded**: compiler, linker, build tools and gdb, needed to compile and debug source code.
- **Cross Build Tools:** some tools like ‘echo’ or ‘rm’ are not present by default on Windows, this package closes that gap.
- CodeWarrior integrates with Yocto.

GDB – Gnu Debugger

- **GDB** is one of the most popular and used debuggers.
- GDB is free software provided by Free Software Foundation
  - http://www.gnu.org/software/gdb/

- GDB can be used to find what is going on `inside' another program while it executes – or what another program was doing at the moment it crashed
  - Current execution point context
  - Program’s source code
  - Registers
  - Stack frames
  - Program memory
  - Variables
  - Change program execution

QCvs

QCVS: Dissecting the Acronym

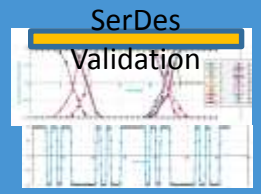
- Configuration of QorIQ processors is increasing in complexity
  - **Q**orIQ - All QorIQ SoCs products are supported
  - **C**onfiguration - Tools for *manually* defining a configuration for key SoC HW and SW features
  - **V**alidation - Tools for verifying and/or optimizing a configuration
  - **S**uite - All these tools under one app, in one framework (Eclipse + Processor Expert)

- One release; one distribution; one installation.
- Version 4.9 is the latest QCVS release.
- QCVS is available with CodeWarrior

QorIQ Configuration and Validation Tools

Eclipse-based tools for configuring, validating and fine-tuning QorIQ processors during board bring-up

SerDes Config and Validation



LS

Pre-boot Loader Configuration

DDR Configuration and Validation

Pin Muxing Configuration

QorIQ Configuration and Validation Suite

The configuration tools help you configure key HW and SW features in QorIQ designs

- **Pin muxing configuration tool**
  Configures available and used pins in SoC
- **Pre-boot loader / RCW configuration**
  Configures RCW and PBI
- **DDR Configuration & validation tool**
  Configures the DDR controllers, Shmoo controller properties to find optimal values and determine margins
- **SerDes configuration & validation tool**
  Configures lane protocols and speed. Run BIST and built-in Jitter Scope to evaluate and optimize SerDes configuration

Why QorIQ Configuration Suite?

- Configuration of QorIQ processors is increasing in complexity
  - Even more complexity is around the corner
  - We support many configuration settings
- Reference manuals are huge and intimidating to new customers
- Configuration problems during board bring-up are HARD and COSTLY
- Learning command line tools requires more training, etc.

- Solution/Strategy to solve these problems:
  - Extensible suite of tools with a common user interface
  - Support NPI and new processor revisions, aligned with DN roadmap

Pre-boot Loader

Configure RCW and PBI commands

Pre-boot Loader (RCW) Configuration

Processor Expert for QCS - Eclipse

File Edit Navigate Search Run Project Processor Expert Window Help

Project Explorer

- P2041RDB
  - Documentation
  - Generated_Code
    - DPAA1
    - InitDdrRegisters_1.c
    - p2041_v1_1ds_ddr.c
    - ddrCtrl_1.tcl
    - PBL.pbl
  - Imported_Files
  - Sources
  - ProcessorExpert.pe

*Component Inspector - PBL | Components Library

Properties | Import

Name	Value	Details
Reset Configuration Word (RCW)		
RCW Source	LBC FCM (NAND flash)	
PLL Configuration		
SerDes PLL and Protocol Configuration		
SerDes Reference Clocks		
SD_REF_CLK1 [MHz]	100.0	100 MHz
SD_REF_CLK2 [MHz]	125.0	125 MHz
SRDS_EN [178]	0b1 - SerDes enabled	
SRDS_PRTCL [128-133]	*0x0A - Bank 1: C-D: 2x SGMII (1.25...*	
SRDS_RIO_SPD [135]	0b0 - 2.5 or 5 Gbps/lane	
SRDS_RATIO_B1 [136-138]	0b011 - 40:1	
SerDes PLL 1 Clock	4.000 GHz	
SRDS_DIV_B1 [139-143]		
SRDS_RATIO_B2 [144-146]	0b011 - 40:1	
SerDes PLL 2 Clock	5.000 GHz	
SRDS_DIV_B2 [147]	0b0 - Divide by 1 off of Bank 2 PLL	
SRDS_LPD_B1 [152-161]		
SRDS_LPD_B2 [162-165]		
SRDS_LPD_B2 - Lane A [162]	0b1 - Lane powered down	This property has to be set to the s..
SRDS_LPD_B2 - Lane B [163]	0b0 - Lane not powered down	This property has to be set to the s..
SRDS_LPD_B2 - Lane C [164]	0b0 - Lane not powered down	This property has to be set to the s..
SRDS_LPD_B2 - Lane D [165]	0b0 - Lane not powered down	This property has to be set to the s..

Components - P2041RDB

- Generator_Configurations
  - P2041_v1_1_Cnf
- OSs
- Processors
  - SoC:P2041_v1_1
- Components
  - PBL:PBL
  - DT1:HWDeviceTree

Problems

1 error, 0 warnings, 0 others

Description

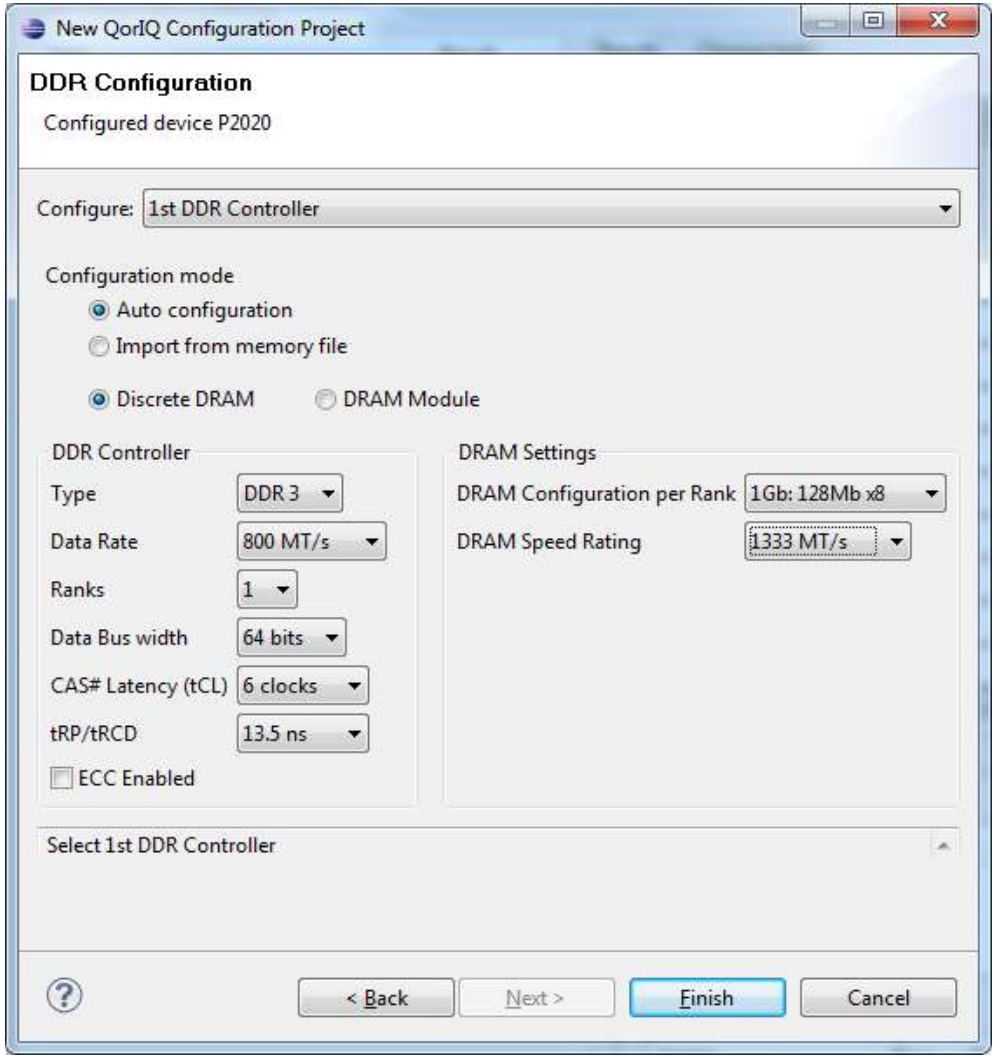
- Errors (1 item)
  - ERROR: Error in the component setting. More details provided by Component Inspector for this component

PBL Tool Key Features

- User friendly GUI for setting each RCW field and for entering PBI commands
- You can’t accidentally set field to a non-supported value
- Constraint checking
- Errata enforcement (no need to read to docs: RM and errata)
- Handles all packaging details (preamble, CRC, etc.)
- load PBL image from a working board and examine/tweak it.
- Numerous import and export formats supported (binary, SREC, XXD, and many more)
- No need to worry about endiannes

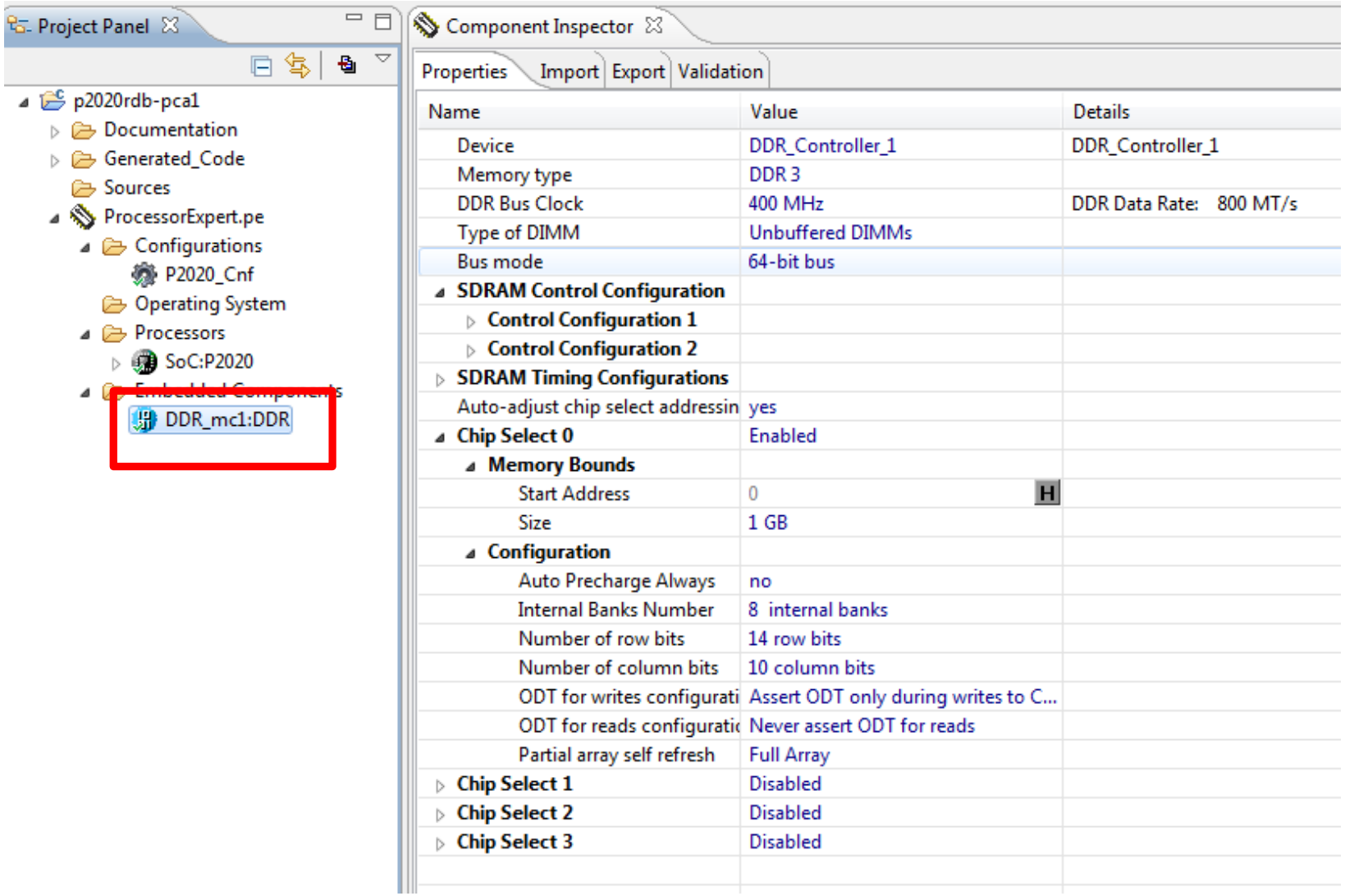
DDR Configuration

DDR Wizard simplifies configuration



- From memory data sheet:
  - Maximum speed rating
  - Capacity
  - Can read from SPD (validation, i.e., licensed feature)

View and edit DDR configuration



Review DDR registers values

Configuration Registers

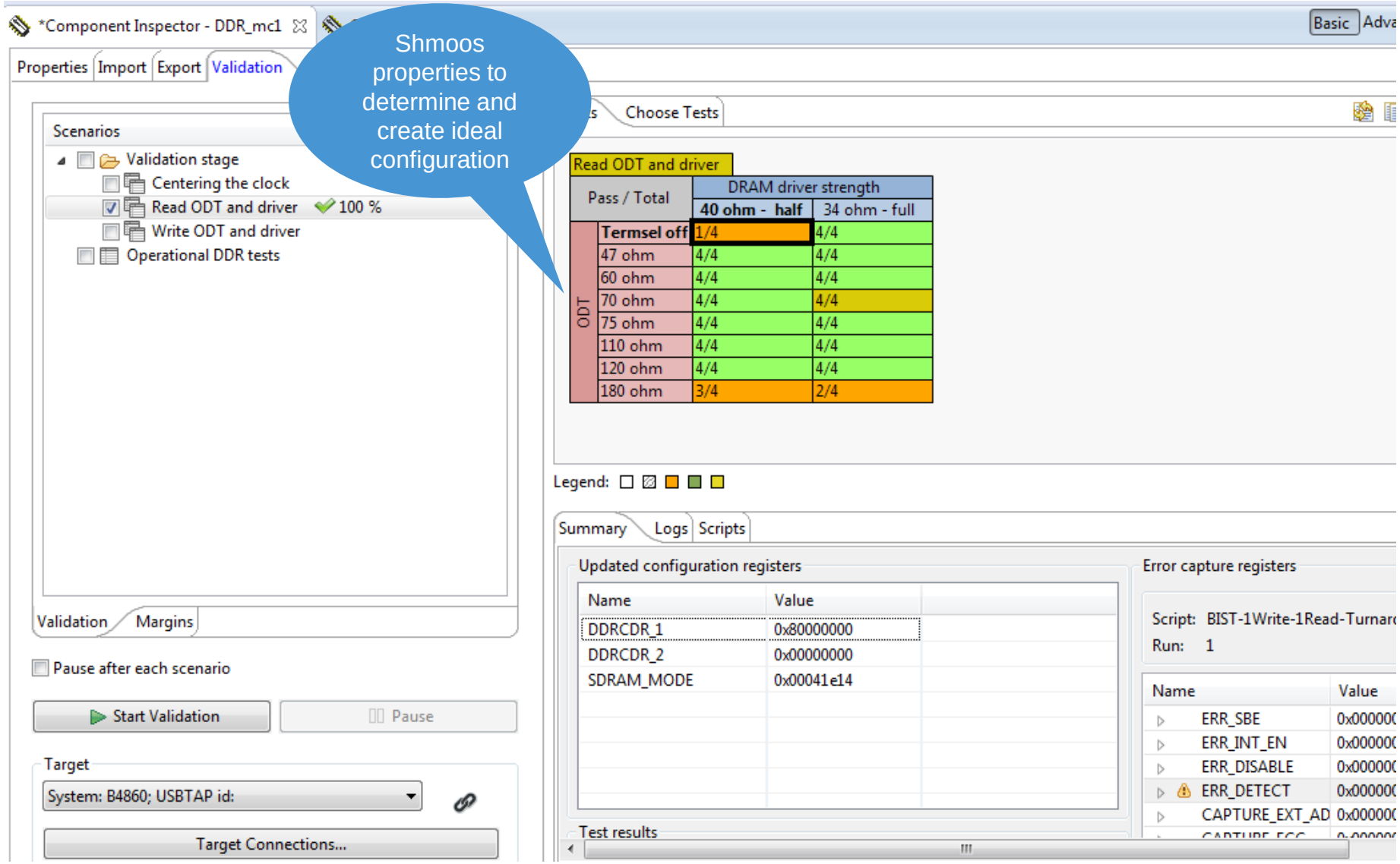
DDR_Controller_1

Reg. name	Init. value	After reset
Peripheral registers		
DDR1_CS0_BNDS	0000003F	00000000
DDR1_CS1_BNDS	00000000	00000000
DDR1_CS2_BNDS	00000000	00000000
DDR1_CS3_BNDS	00000000	00000000
DDR1_CS0_CONFIG	80014202	00000000
DDR1_CS1_CONFIG	00000000	00000000
DDR1_CS2_CONFIG	00000000	00000000
DDR1_CS3_CONFIG	00000000	00000000
DDR1_CS0_CONFIG_2	00000000	00000000
DDR1_CS1_CONFIG_2	00000000	00000000
DDR1_CS2_CONFIG_2	00000000	00000000
DDR1_CS3_CONFIG_2	00000000	00000000
DDR1_TIMING_CFG_3	00030000	00000000
DDR1_TIMING_CFG_0	00330104	00110105
DDR1_TIMING_CFG_1	6E6B8846	00000000
DDR1_TIMING_CFG_2	0FA8D0CC	00000000
DDR1_SDRAM_CFG	47000008	03000000
DDR1_SDRAM_CFG_2	24401050	00000000
DDR1_SDRAM_MODE	00061421	00000000

QCVS: DDR Validation Tool

- Run validation scenarios to automatically determine best value for key controller properties, given a specific DIMM or the board's discrete memory
  - write leveling start
  - clock adjust
  - read and write ODT
  - driver strength
- Run margins scenarios to determine confidence level
  - Given the ideal configuration, how much of a "working range" (margin) does each byte lane have?
    - Wide margins mean you can be confident DDR will work well under varying conditions—temperature and voltage.
    - Thin margins means low confidence. Board design may need fine-tuning
- Load setting from a working board then run validation and determine how much margin is available

DDR Validation Tool



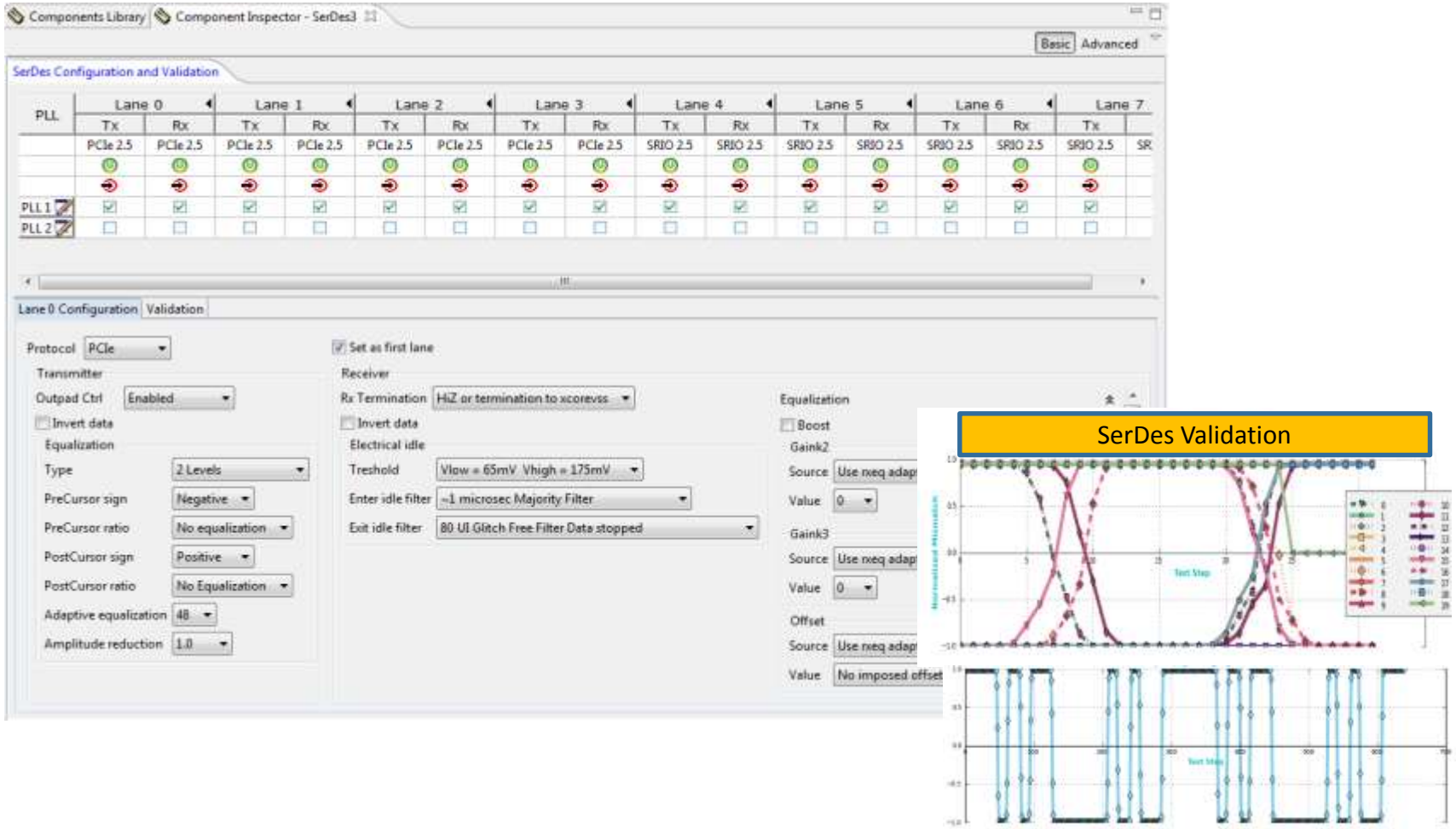
SerDes Configuration and Validation

SerDes Configuration and Validation

- Heavily muxed SerDes lanes make choosing protocols and speeds painful
  - PBL tool lets you graphically make these selections, but decision-making process is still very complex and difficult
  - Sophisticated new SerDes configuration UI will make this much, much easier
  - Load the existing SerDes setting from a board and then run validation test on it.

- With 10G and up, signal integrity becomes a serious concern. NXP SerDes (Lynx) have BIST and Jitter Scope features.
  - New SerDes Validation tool will give users a sophisticated GUI front end to the validation features built into the SerDes modules
  - Graphically see how clean the signal eye is
  - Manually fine tune electrical properties to get a cleaner eye
  - Many test patterns supported

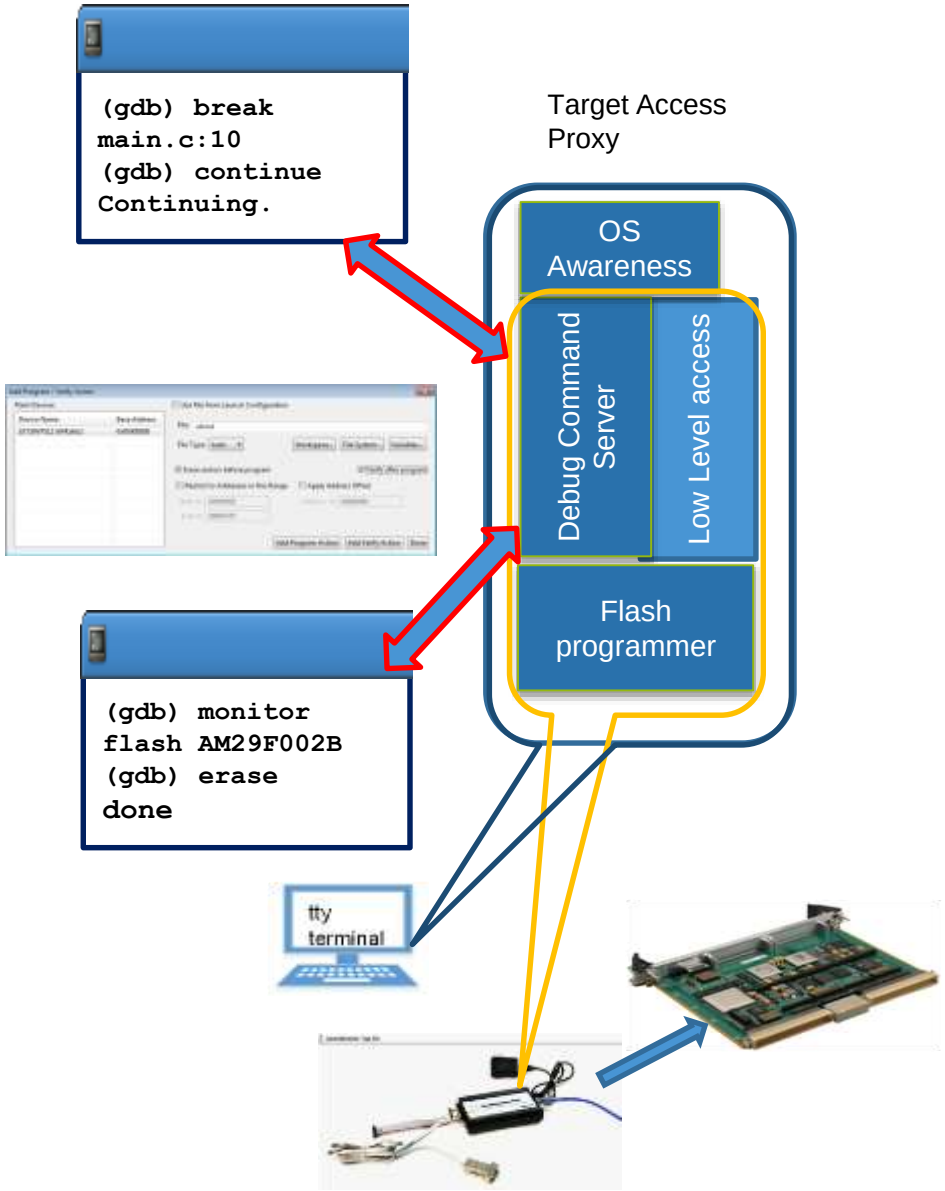
SerDes Configuration and Validation (1H 2015)



Bare-metal debug & flash programing

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



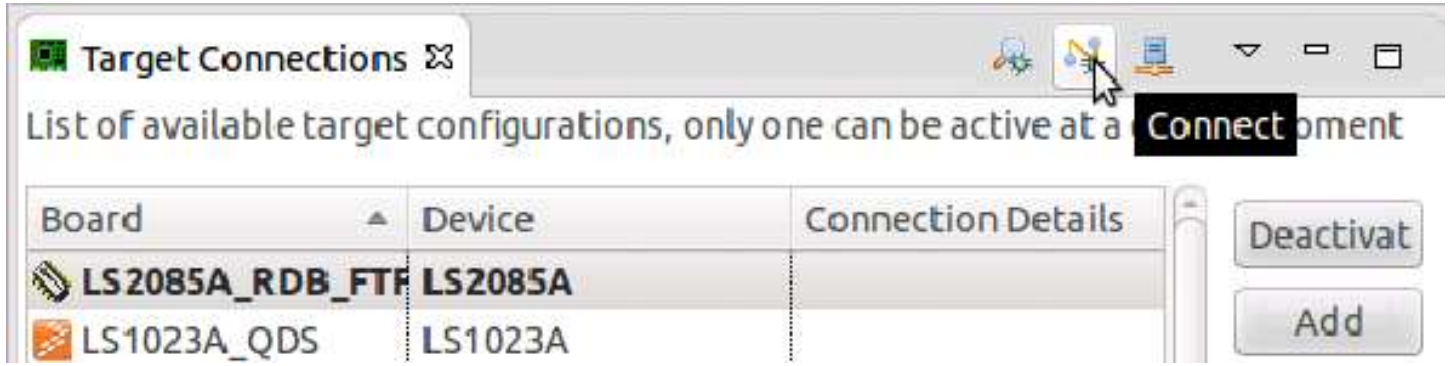
Flashing u-boot image to the Target (1)

There are possible 3 ways to flash something in flash

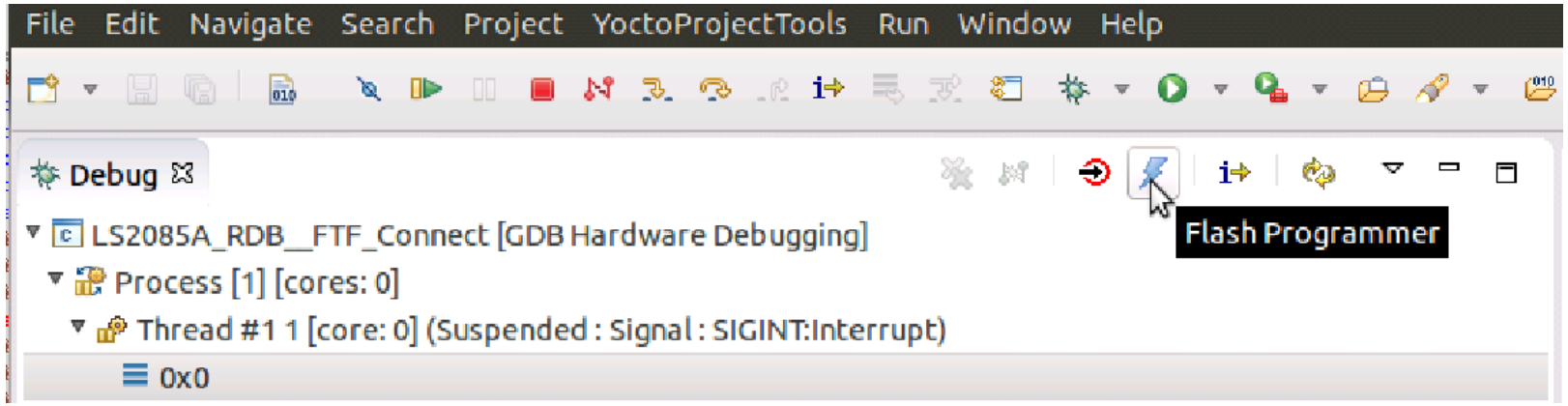
1. ***Flash Programmer GUI***
2. ***GDB CLI***
3. ***GDB Eclipse***

To use Flash Programmer GUI:

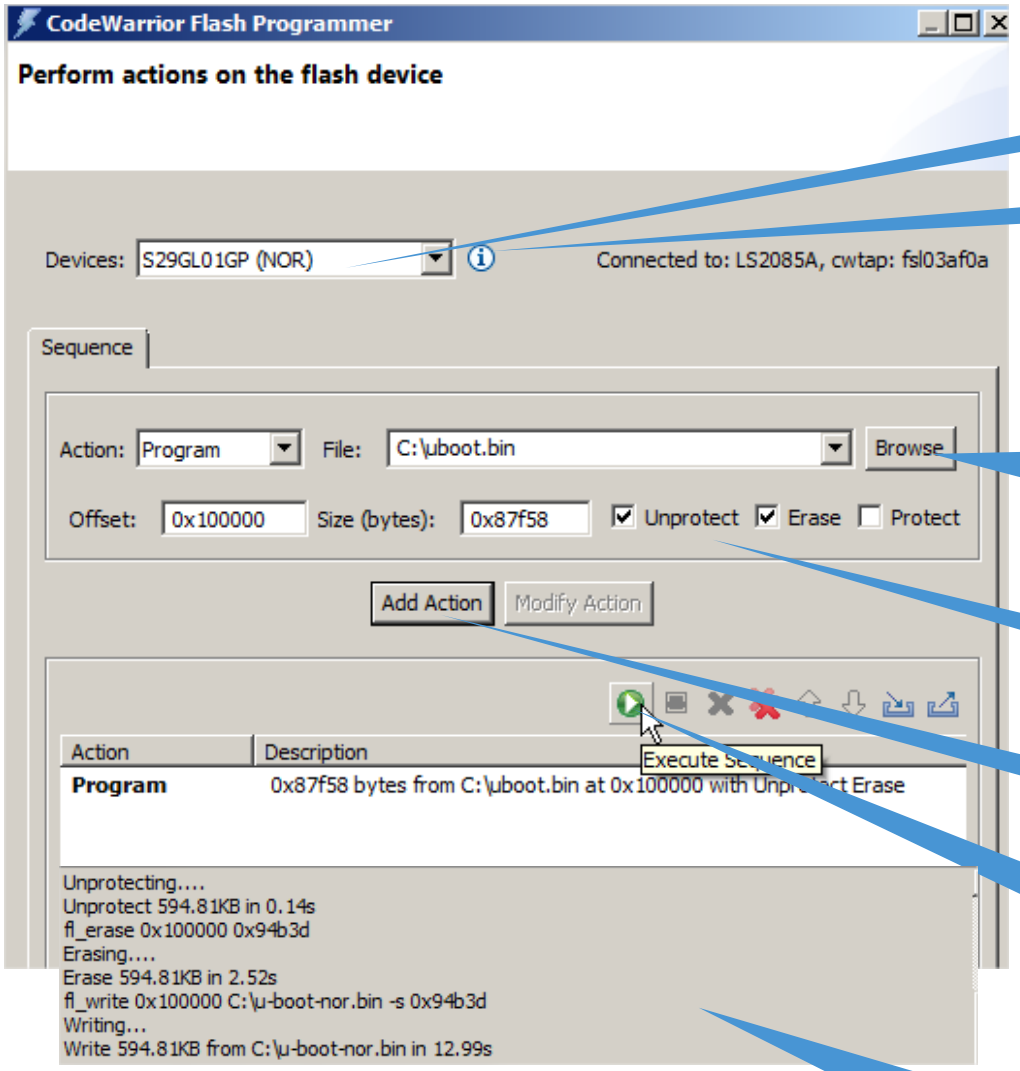
A. connect to target



B. Click Flash Programmer icon



Flashing U-Boot Image to the Target (2)



Drop-down list with all supported flashes

Hover-information about current flash

Device Info
base address: 0x580000000
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

Check Erase and Unprotect

Add action

Execute it

Results in FP log

Using Flash from Command Line

1. Edit *CW_ARMv8/ARMv8/gdb_extensions/flash/cwflash.py* with your board and connection settings

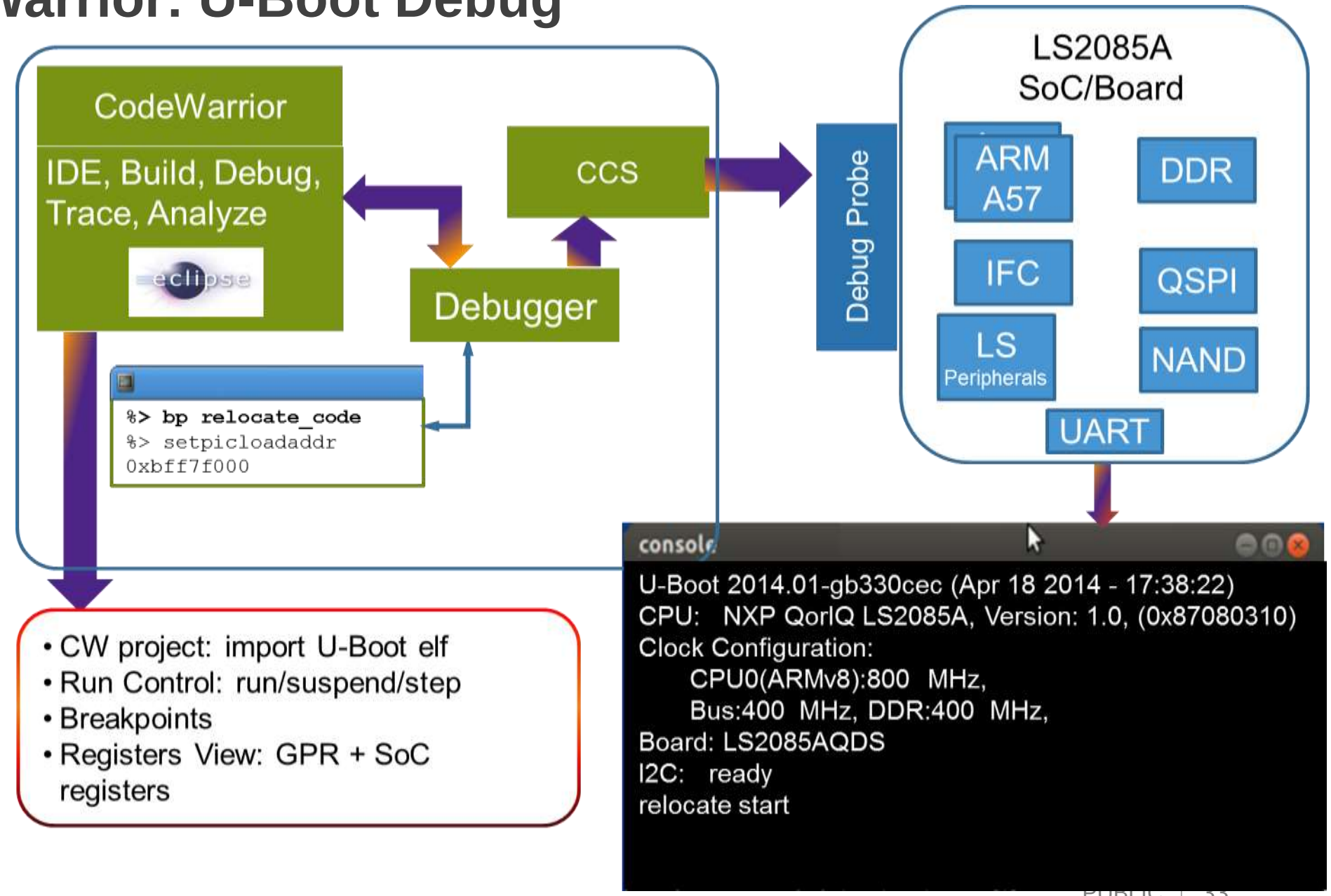
2. Start GDB console from *CW_ARMv8/ARMv8/gdb/bin/aarch64-fsl-gdb.bat*
   - *cd ../../gdb_extensions*
   - *source flash/cwflash.py*

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

Uboot debug

CodeWarrior: U-Boot Debug



- CW project: import U-Boot elf
- Run Control: run/suspend/step
- Breakpoints
- Registers View: GPR + SoC registers

U-Boot debug - features

- 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

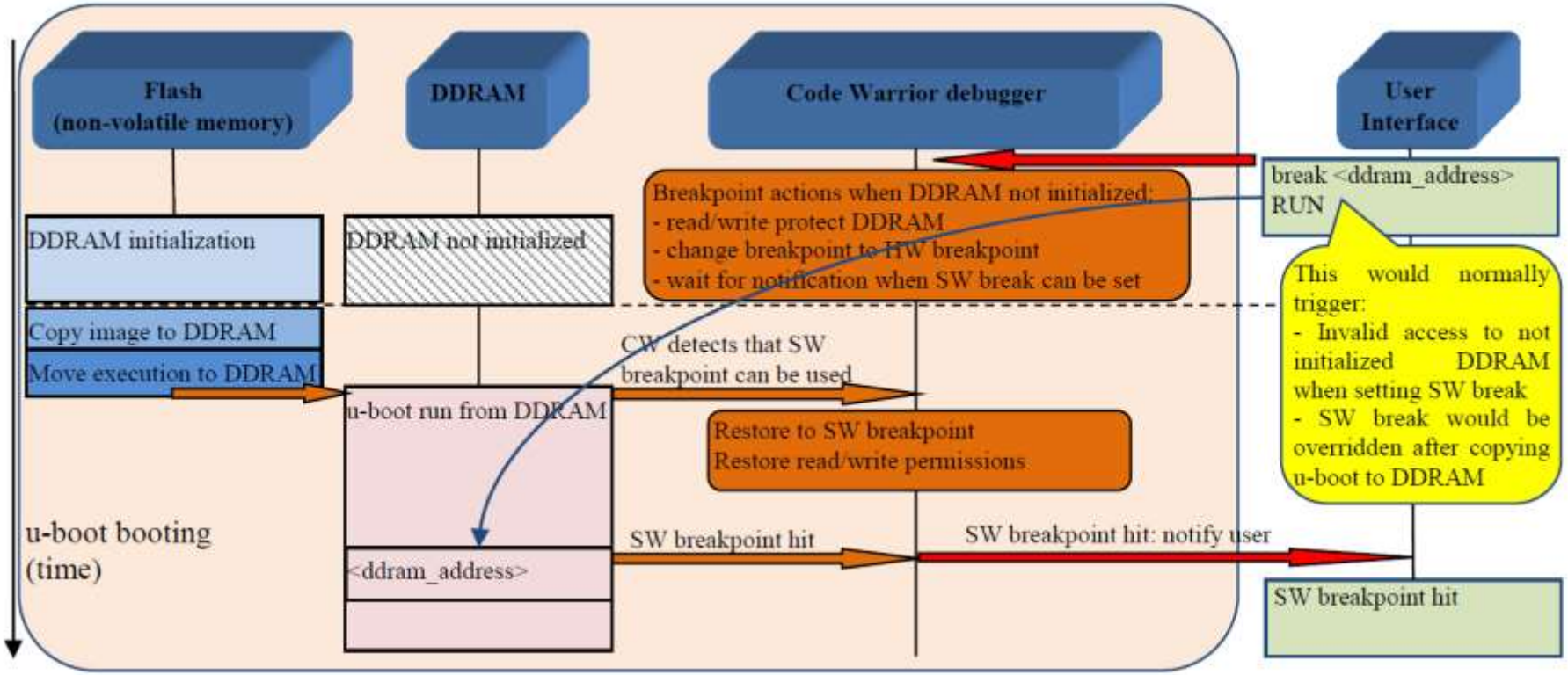
U-Boot Debug – Debug Overview

- **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 – All stages

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

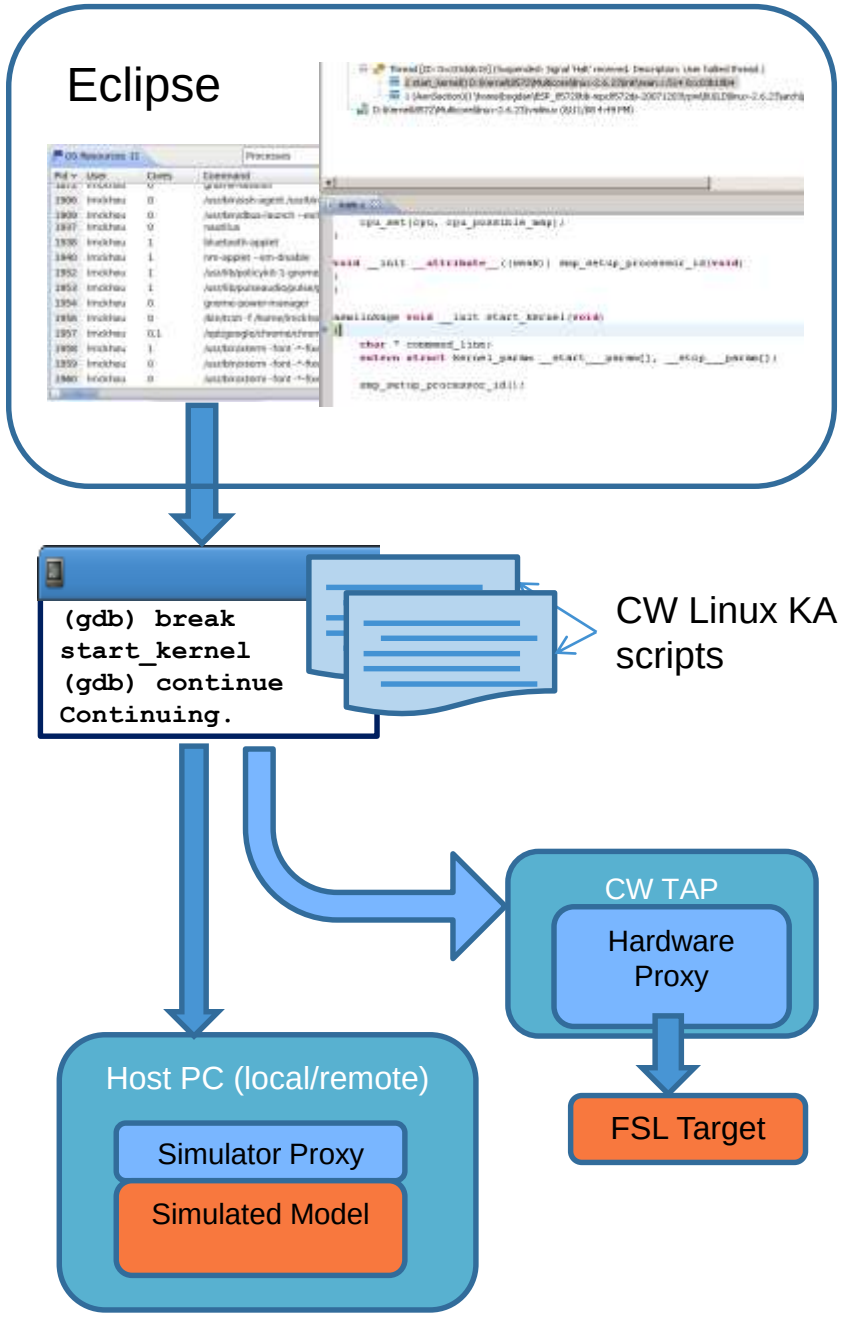


Linux kernel debug

Linux Kernel Awareness

- Linux Kernel Awareness features kernel threads information
  - Kernel modules list
  - Kernel threads list
  - MMU awareness
  - Kernel module debug, module insert/remove detection

- Available from Eclipse GUI and command line in debugger console

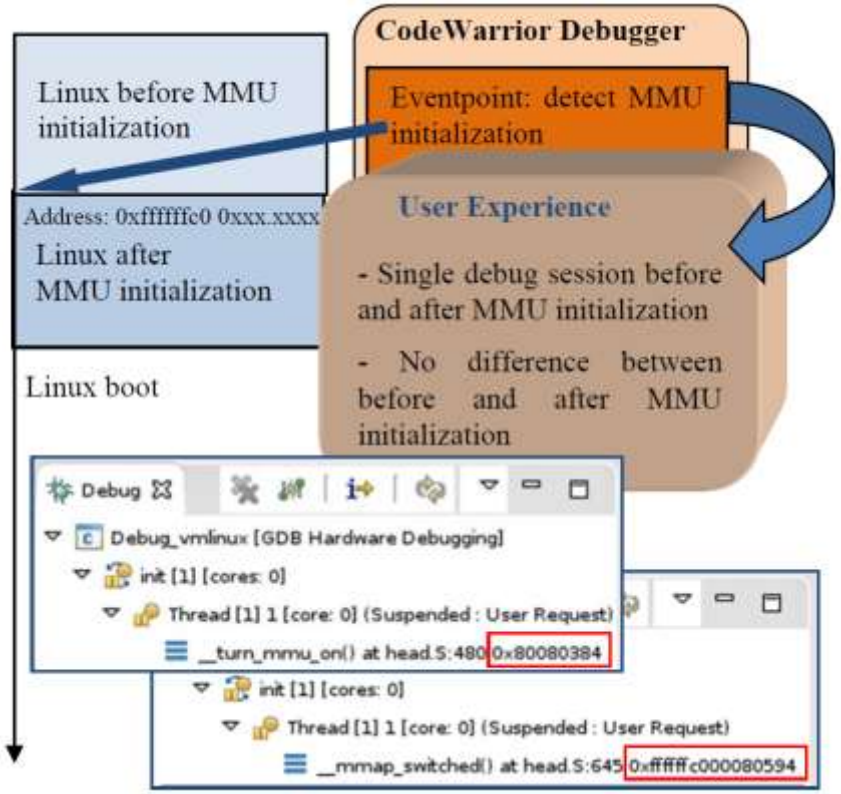


Linux kernel debug - features

- Linux kernel awareness
- Debug from Linux kernel entry point
- MMU enablement detection
- SMP debugging
- OS resources
- Registers View: GPR + SoC registers
- Debugging features:
  - Run control run/suspend/step
  - Breakpoints, in any ARMv8 EL mode
  - Disassembly, Memory view, Variable View, Expressions

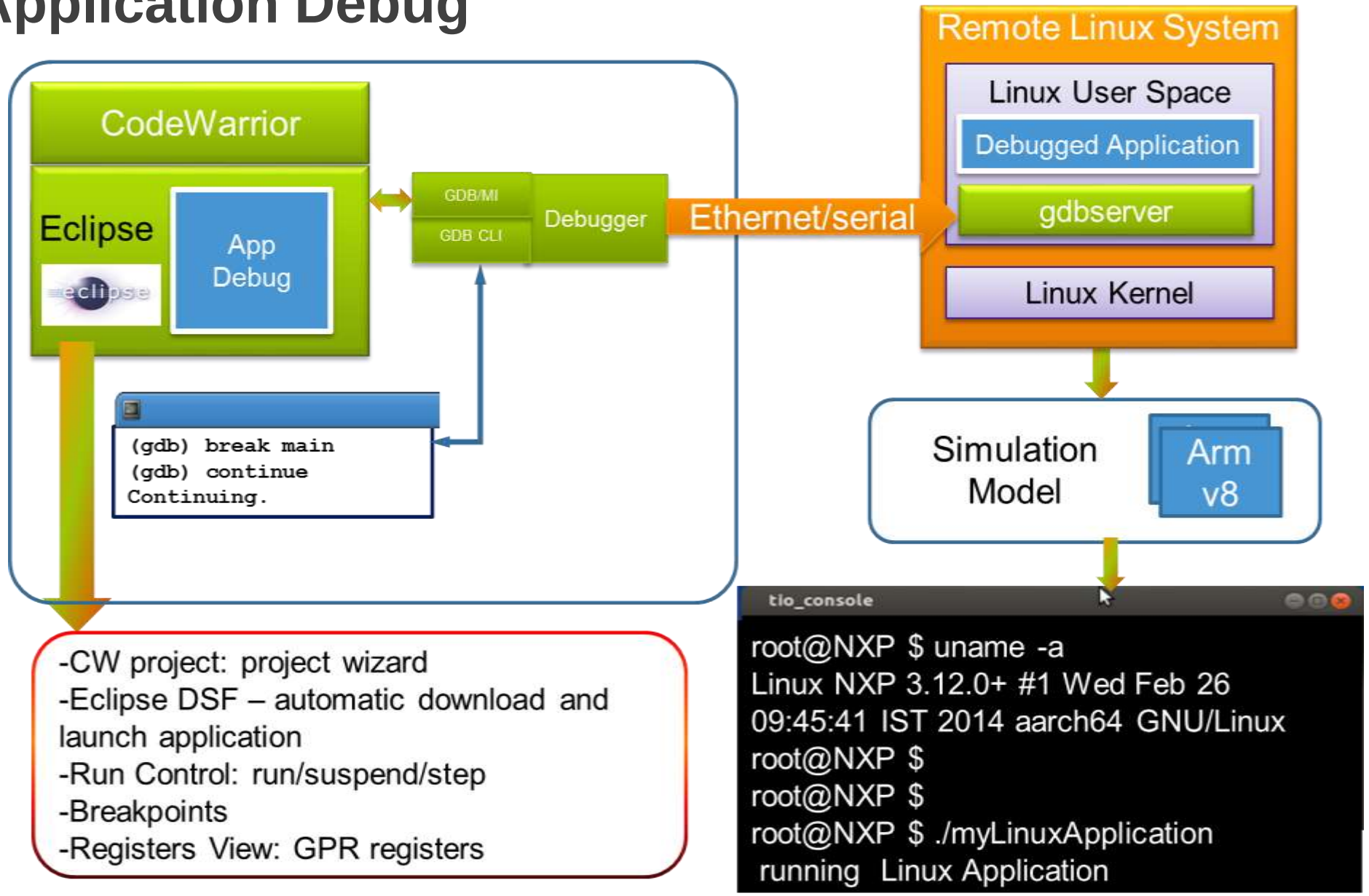
Linux Kernel Debug – MMU Awareness Capabilities

- CodeWarrior automatically
  - Detects the point where MMU initialization is done
  - Computes and applies the relocation offset
- The user is not aware of two debug configuration settings: before or after MMU initialization.
  - No difference between debugging before and after MMU initialization
  - No special action is required when moving before and after MMU initialization
  - No need for the user to know the current MMU initialization state and to manually apply the relocation offset



Linux app debug

Linux Application Debug



-CW project: project wizard
-Eclipse DSF – automatic download and launch application
-Run Control: run/suspend/step
-Breakpoints
-Registers View: GPR registers

CodeWarrior– Debugging ARM Target

Debug - simple_linux_app/src/main.c - CodeWarrior Development Studio for QorIQ LS series - ARM V8 ISA

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | C/C++ | Debug

Debug

- simple_linux_app [C/C++ Remote Application]
  - simple_linux_app.elf [1781] [cores: 6]
    - Thread #1 1781 [core: 6] (Suspended : Breakpoint)
      - main() at main.c:29 0x400558
  - Remote Shell
  - /home/b32721/Freescale/CW4NET_v2015.05_b150430/C

Regi

Name	Value
x0	0x1 (Hex)
x1	0xffffffffc78 (Hex)
x2	0xffffffffc88 (Hex)

OS Resources

Processes

- Processes
- Process groups
- Threads
- File descriptors
- Sockets
- Shared-memory regions
- Semaphores
- Message queues
- Kernel modules

Pid	User		
1	root		
2	root		
3	root		
4	root		
5	root		
6	root		
7	root		
8	root		
9	root	0	[rcu_bh]
10	root	0	[migration/0]
11	root	0	[watchdog/0]
12	root	1	[watchdog/1]
13	root	1	[migration/1]
14	root	1	[ksoftirqd/1]
16	root	1	[kworker/1:0H]
17	root	2	[watchdog/2]
18	root	2	[migration/2]
19	root	2	[ksoftirqd/2]
20	root	2	[kworker/2:0]
21	root	2	[kworker/2:0H]

main.c

```
26 int
27 main(void)
28 {
29   printf("Hello ARM World!" "\n");
30   return 0;
31 }
```

Disassembly

Enter location here

```
0000000000400558:   adrp x0, 0x400000
000000000040055c:   add x0, x0, #0x600
0000000000400560:   bl 0x4003e0 <puts@plt>
```

Console

simple_linux_app [C/C++ Remote Application] Remote Shell

```
root@ls2085aqds:~# chmod +x /home/root/simple_linux_app
Process /home/root/simple_linux_app.elf created; pid = 1
Listening on port 1234
Remote debugging from host 192.168.1.1
```

Remote Systems

- Local
- ScpConnection
  - Scp Files
  - Ssh Shells
  - Ssh Terminals

Two ways to run GDB

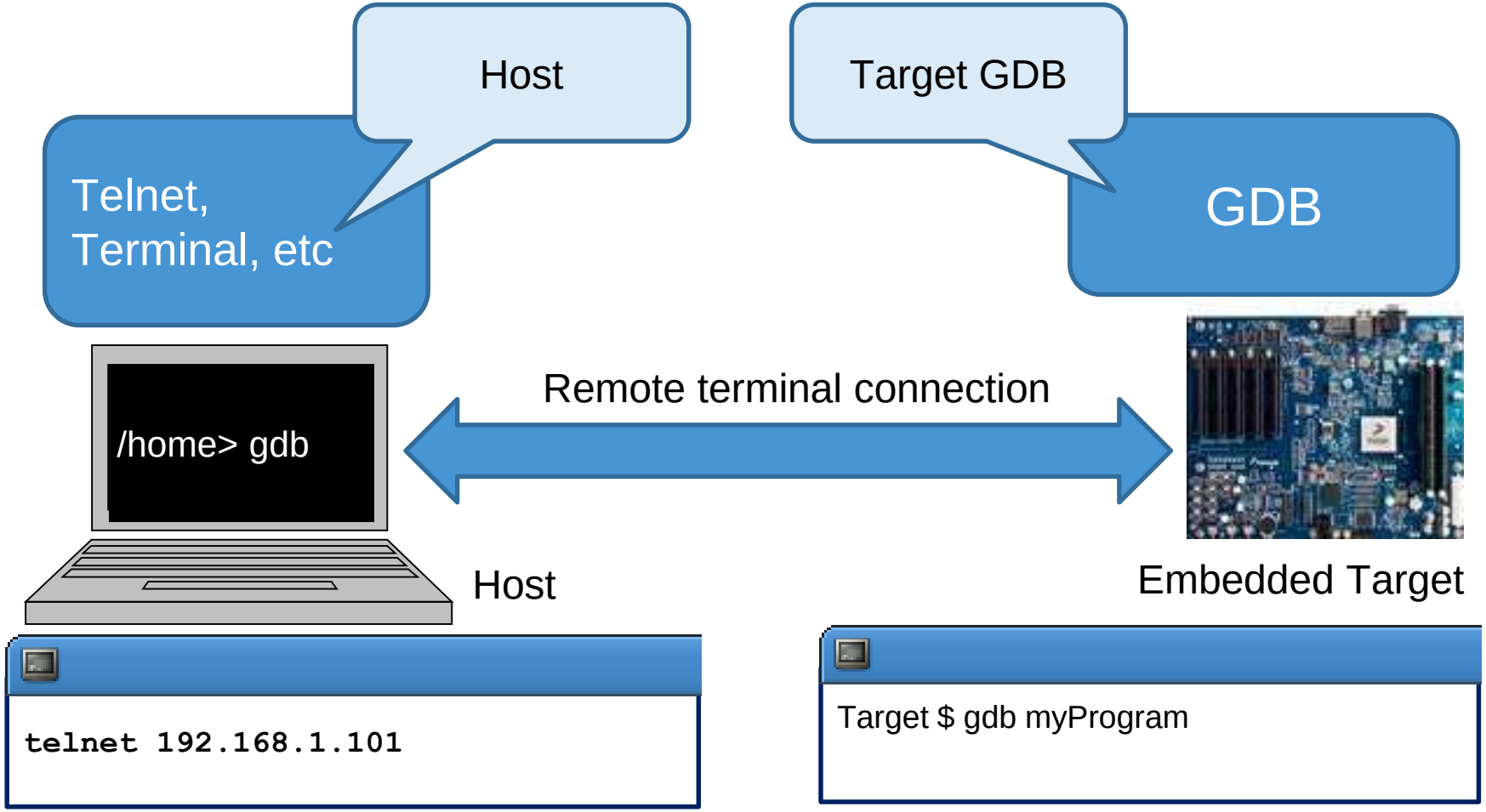
Target (self-hosted)

- GDB runs on the target (DUT)
  - E.g. Target OS: Linux
- Debugs an application running on the same system
- Interface with the target system using other applications
  - telnet into the target system to run GDB from the Linux command prompt

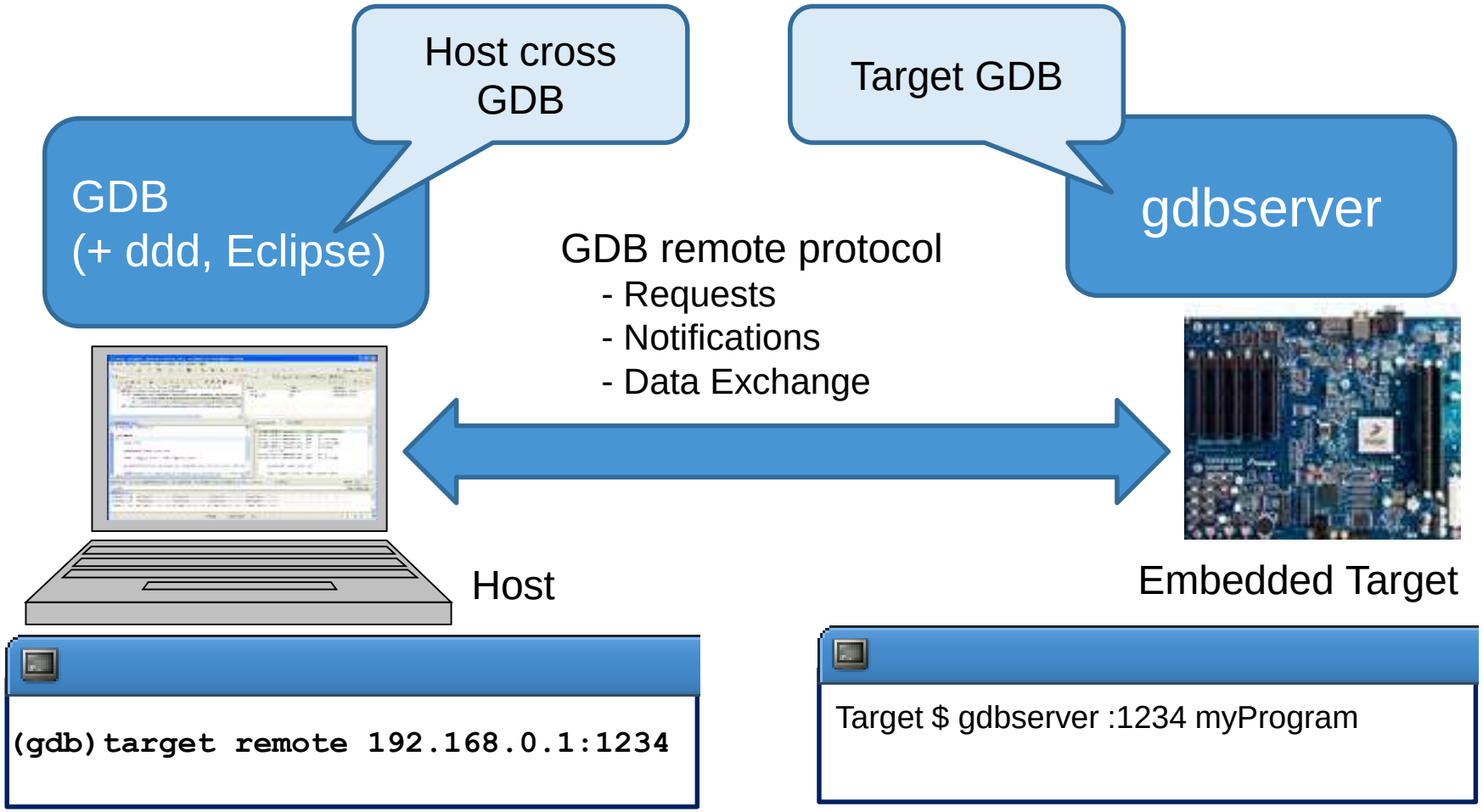
Native (Host)

- GDB runs on the development host
  - Host OS and Target OS are not necessarily the same
- Remotely debugs an application running on the target
  - Socket connection or UART connection over the OS's drivers and interface carries GDB commands and responses
  - Host GDB communicates with target GDB server

GDB Self-Hosted Target Debugging ARM Target



GDB Host Remote Debugging ARM Target



Linux application debug – features

- **gdbserver** Debug agent
  - User-space application
- Debug **scenarios** supported
  - **Download**, start & debug application from main
  - **Attach** to a running process
- Features
  - Read/write memory, registers, variables
  - **Threads creation/death detection**
  - Shared libraries awareness
  - Configurable signal policies
  - I/O redirection
- **OS Resources**
- CodeWarrior – GDB server interaction
  - Ethernet connection
  - Serial connection\

Linux application debug – Prerequisites

- **QorIQ LS** board
- **Linux** running on the target
- **Network connectivity** inside Linux
- **GDB server** debug agent on the target
- Ways of putting GDB server on the target
  - GDB server is included by default in the SDK image – no change required
  - Compile GDB Agent separately
    - **bitbake –c cleansstate gdb**
    - **bitbake gdb**
    - Use **SCP** to put GDBAgent on the target (we'll find the **ELF** in <YoctoInstallationPath>/fsl-qoriq-sdk/**build_ls2085ardb_release/tmp/work/aarch64-fsl-linux/gdb/7.7.1+fsl-r0/build/gdb/gdbserver/gdbserver**)

Trace and profile

Linux Trace

- Static probe points strategically located inside the kernel code
- Register/unregister with tracepoints via callback mechanism
- Can be used to profile, debug and understand kernel behavior

- Trace synchronization
  - Time correction
  - Multi-core
  - Dependency analysis, delay analyzer
  - Dependencies among processes

Linux Probe-less Trace

- Based on a software probe
  - Linux cross-compiled application
  - CW and SDK component

- Advantages
  - Speed
    - contains only what is needed
  - Speed
    - all services are hosted on target machine
  - Nonintrusive
    - no need to instrument the target application
  - Simple API
    - can be effortlessly integrated into any testing framework
  - Data-driven
    - the configurator and probe can be easily tuned up using xml files

Linux Probe-less Trace – Hardware setup

QorIQ LS board

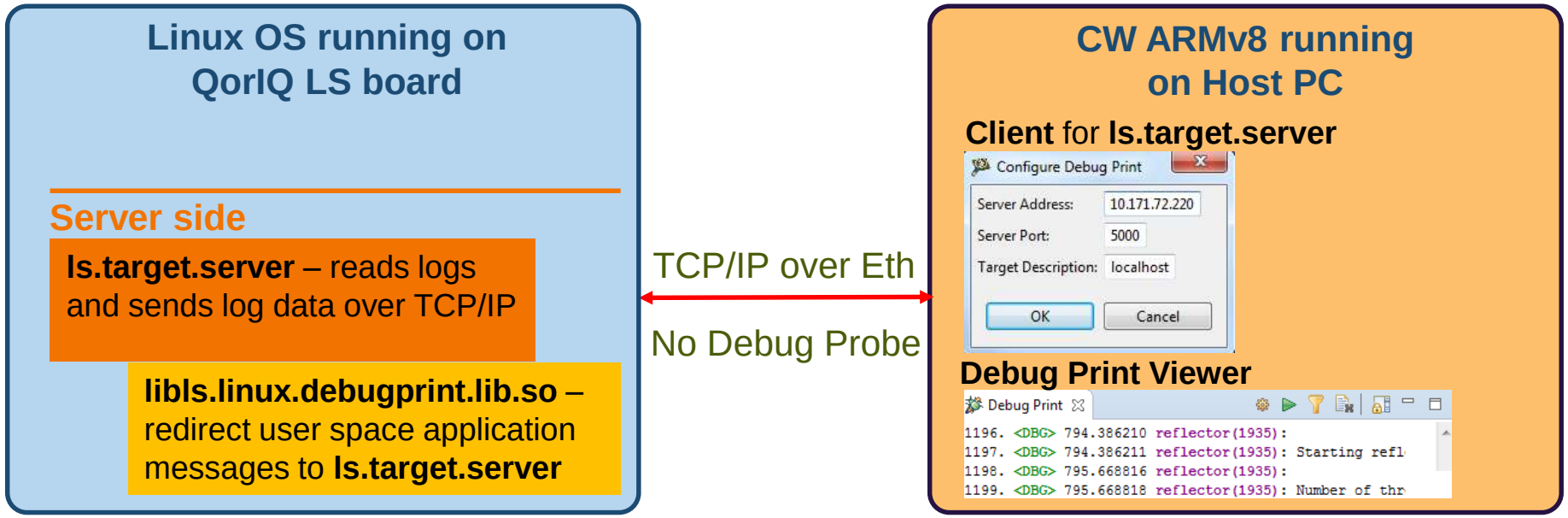
Linux standalone application included in ***CodeWarrior*** and **QorIQ SDK**

Ethernet cable + `linux.armv8.satrace`

Hardware Probe using JTAG
(E.g. CodeWarrior USB TAP)

Debug Print – Fundamentals

- Debug Print consists in:
  - **Server side**: running on target Linux OS for collecting Kernel Ring Buffer logs and application messages to standard output;
  - **Client side**: running under CW for getting data out of the server, display and various configurations



Debug Print Considerations

- Debug Print Client can show up messages from Kernel, Modules and User Applications in a easy straightforward fashion allowing filtering based on source/timestamps/keywords

- Attaching like use cases to a running application is not supported since the Debug Print redirect library must be loaded before application is getting started

CW-ARM: Performance Analysis / Scenarios Tool

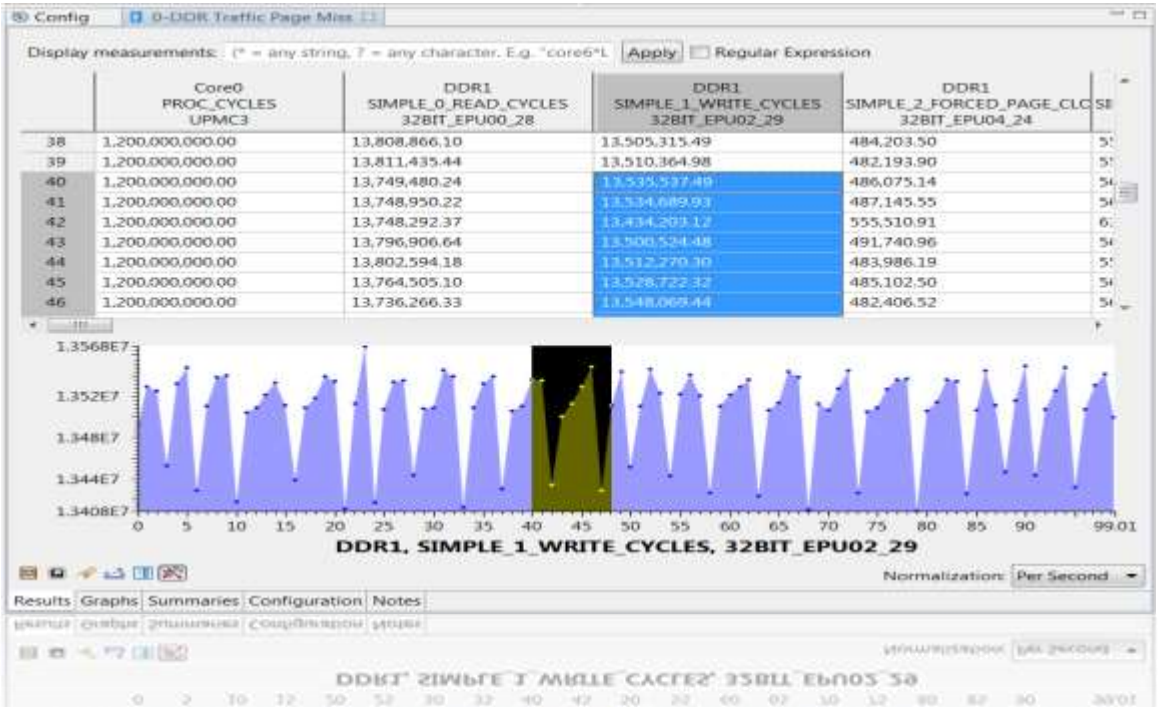
Optimized workflow for efficiently narrowing down performance issues anywhere on the system

Customer Benefits

- System Optimization for Cores and SoC
- Complexity Abstraction
- Delivers FSL expertise to users .
- Ease of Use
- Probe-less, field based usage.
- Streamlined to solve several performance issues

Key Features

- Stand alone or bundled with CW
- Performance Analysis including visualization
- Connection auto discovery
- “Canned” measurement scenarios
- 100+ scenarios covering Core and SoC blocks
- User defined measurement scenarios
- Compare pairs of runs
- Graphically visualize all measurements
- “Live” view of events and metrics
- Supports “bare metal” or Linux applications
- Python scripting support

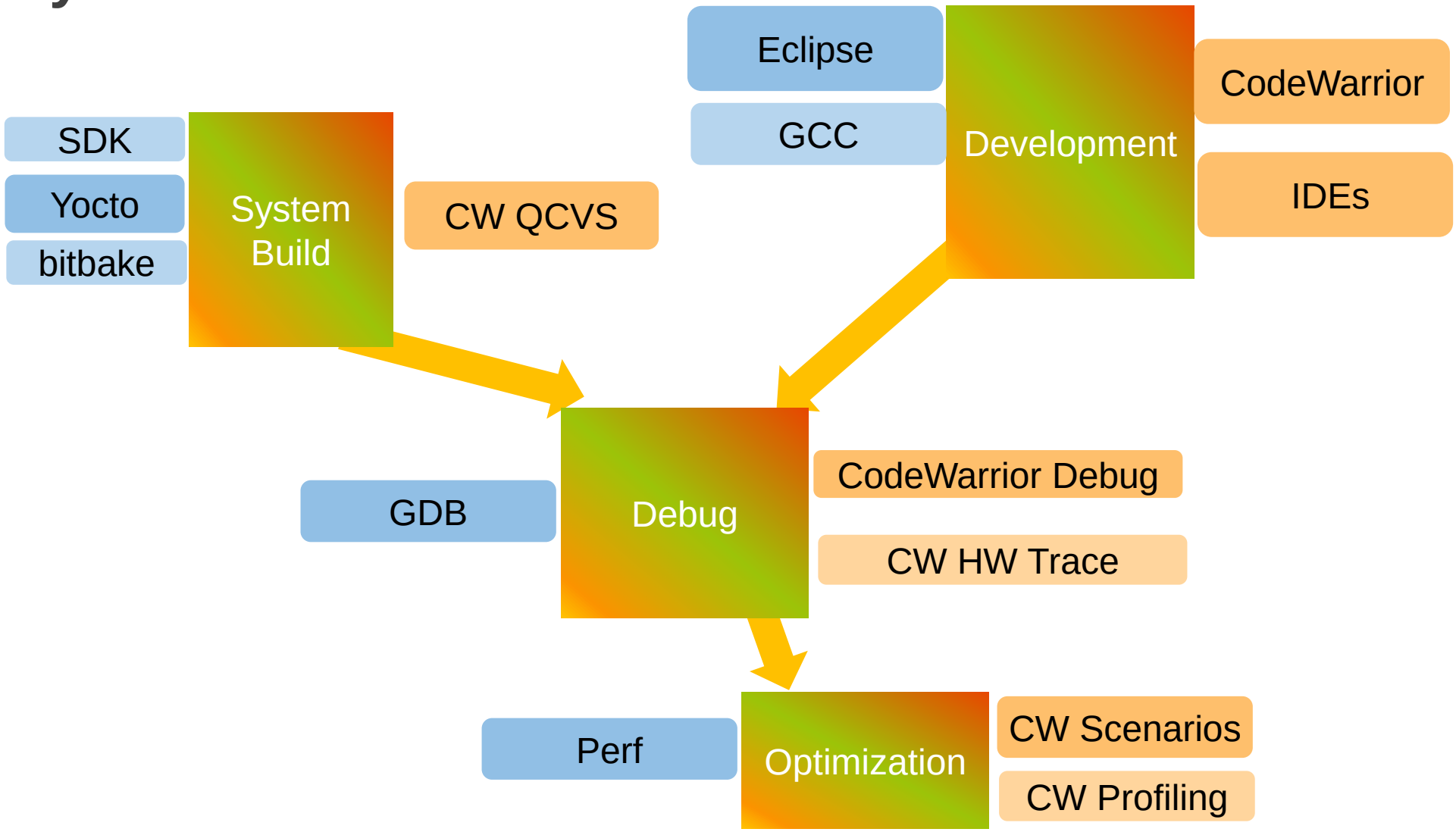


Devices supported

- P2040, P3041, P5020x P5040, P4080 (Revs 1,2)
- T1040,T2080
- T4240 (Rev1, 2)
- B4860 (Rev 2, 2.1)
- LS1020A, LS1021A, LS1022A
- LS1043A
- LS2080/40A
- LS2085/45A

Summary

Summary


Eclipse
GCC
Development
CodeWarrior
IDEs
SDK
Yocto
bitbake
System Build
CW QCVS
GDB
Debug
CodeWarrior Debug
CW HW Trace
Perf
Optimization
CW Scenarios
CW Profiling

SECURE CONNECTIONS
FOR A SMARTER WORLD

NXP and the NXP logo are trademarks of NXP B.V. All other product or service names are the property of their respective owners. © 2017 NXP B.V.