Loading Boot Loader on LS1046ARDB through PCIe

1 Introduction

This document details the implementation of a use case where PCIe root complex (RC) provides boot images to the LS1046A configured as a PCIe endpoint (EP). The figure below shows the setup diagram for this use case.



Figure 1. Use case setup diagram

To provide access to CCSR registers and memory space, the endpoint has to configure BARs (inbound windows) using PBI commands. Similarly, the root complex needs to configure its outbound windows. With this, the root complex can configure endpoint's DDR controller registers and copy boot loader directly into endpoint's DDR.

Using PBI commands, OCRAM of endpoint is programmed with minimal firmware to configure CSU registers. The firmware waits until boot loader is loaded on endpoint. The root complex will set a flag indicating the completion of copy. On detection of the flag, the endpoint comes out of polling loop and jumps to DDR for loading U-Boot.

Contents

1	Introduction	1
2	Use case setup	2
3	Configuring LS1046ARDB as PCIe root complex	2
4	Configuring LS1046ARDB as PCIe endpoint	9
5	Appendix: Using C29XPCIE-RDB as PCIe root complex	22
6	Appendix: Hardware and software resources	24
7	Appendix: Related documentation	25
8	Revision history	



Use case setup

For this use case, two LS1046ARDBs are connected back to back using a PCIe extended cable. One LS1046ARDB is configured as a root complex and the other LS1046ARDB is configured as an endpoint.

This document describes:

- Steps to configure the LS1046ARDB as a PCIe root complex
- Steps to configure the LS1046ARDB as a PCIe endpoint
- · Procedure for transferring boot images from root complex to endpoint and booting endpoint

2 Use case setup

To implement the use case, two LS1046ARDBs are connected to each other using a PCIe extended cable. One of the two boards is configured as root complex and the other as endpoint. The figure below shows the use case setup.



Figure 2. Use case setup

RCW and PBI binary must be programmed on the endpoint as explained in Configuring LS1046ARDB as PCIe endpoint before connecting endpoint to root complex.

3 Configuring LS1046ARDB as PCIe root complex

This section describes how to configure an LS1046ARDB as a PCIe root complex and how to load boot loader on another LS1046ARDB configured as a PCIe endpoint.

The root complex needs to complete the following functions:

- Perform link training with endpoint
- · Program two outbound windows for memory transaction
- Initialize endpoint's DDR controller
- Copy RAMboot image to endpoint's DDR
- · Trigger boot process of endpoint

3.1 Board switch settings

The table below shows DIP switch settings of the LS1046ARDB where QSPI is the boot source.

DIP switch	Settings	Notes
SW3[1:8]	01000110	0 indicates OFF
SW4[1:8]	00111011	1 indicates ON
SW5[1:8]	00100010	

Table 1. DIP switch settings

3.2 RCW fields for PCIe configuration

The table below details the RCW fields to configure the LS1046ARDB as root complex.

 Table 2. POR parameters for PCIe controller (root complex)

Field name	Description	Value
RCW[HOST_AGT_PEX]	Selects between Root Complex (RC) and Endpoint (EP) modes	3'b000: All Host mode
RCW[SRDS_PRTCL_Sn]	Determines the link width	0x1133: SRDS_PRTCL_S1
		0x5506: SRDS_PRTCL_S2
RCW[SRDS_DIV_PEX_Sn]	Determines the link speed	2'b00: SRDS_DIV_PEX_S1
		2'b00: SRDS_DIV_PEX_S2

3.3 Program outbound windows

The table below describes the base addresses for LS1046A's PCIe controllers.

Table 3. PCIe controller base address

PCIe controller	Base address
PCIe controller 1	340_0000h
PCIe controller 2	350_0000h
PCIe controller 3	360_0000h

The root complex needs to access CCSRBAR register space and DDR SDRAM memory space of the endpoint. This can be done by programming root complex's outbound windows. In the current example, two outbound windows are used, which can be defined as follows:

- Outbound window 2: Represents a memory window of 512 MB from 0x48_1000_0000 to 0x48_2FFF_FFFF, to be translated to 0x00000000 for accessing endpoint's CCSRBAR and OCRAM memory space
- Outbound window 3: Represents a memory window of 512 MB from 0x48_4000_0000 to 0x48_5FFF_FFFF, to be translated to 0x80000000 for accessing endpoint's DDR SDRAM memory space

Program outbound window 2

Perform these steps to program outbound window 2:

- 1. Set up Index register:
 - a. Write 0x0000002 to address $\{0x700 + 0x200\}$ to set outbound window 2 as the current window.
- 2. Set up Region Base and Limit Address registers:
 - a. Write 0x10000000 to address {0x700 + 0x20C} to set the lower base address.
 - b. Write 0x0000048 to address $\{0x700 + 0x210\}$ to set the upper base address.
 - c. Write 0x2fffffff to address {0x700 + 0x214} to set the limit address.
- 3. Set up Target Address registers:
 - a. Write 0x00000000 to address {0x700 + 0x218} to set the lower target address.
 - b. Write 0x00000000 to address $\{0x700 + 0x21C\}$ to set the upper target address.
- 4. Configure the window through Region Control 1 register:
 - a. Write 0x00000000 to address {0x700 + 0x204} to define the type of the window to be memory space.
- 5. Enable the window:
 - a. Write 0x80000000 to address {0x700 + 0x208} to enable the window.



Figure 3. Outbound window 2

Program outbound window 3

Perform these steps to program outbound window 3:

- 1. Set up Index register:
 - a. Write 0x00000003 to address {0x700 + 0x200 } to set outbound window 3 as the current window.
- 2. Set up Region Base and Limit Address registers:
 - a. Write 0x40000000 to address {0x700 + 0x20C} to set the lower base address.
 - b. Write 0x00000048 to address $\{0x700 + 0x210\}$ to set the upper base address.
 - c. Write 0x5fffffff to address {0x700 + 0x214} to set the limit address.
- 3. Set up Target Address registers:
 - a. Write 0x80000000 to address {0x700 + 0x218} to set the lower target address.
 - b. Write 0x00000000 to address {0x700 + 0x21C} to set the upper target address.
- 4. Configure the window through Region Control 1 register:
 - a. Write 0x00000000 to address $\{0x700 + 0x204\}$ to define the type of the window to be memory space.
- 5. Enable the window:
 - a. Write 0x80000000 to address {0x700 + 0x208} to enable the window.



Figure 4. Outbound window 3

3.4 Initialize endpoint's DDR

The root complex initializes endpoint's DDR by configuring DDR controller registers in CCSR space. DDR configuration for the LS1046ARDB is provided in a target initialization file in CodeWarrior for ARMv8. Below is the path to the LS1046ARDB target initialization file within CodeWarrior installation directory:

<CWInstallDir>\Freescale\CW4NET_v2017.03\CW_ARMv8\Config\boards\LS1046A_RDB_init.py

QCVS tool can also be used to extract data for initializing DDR controller registers.

Perform below steps to initialize DDR of PCIe endpoint from PCIe root complex. All commands are run on the U-Boot prompt of the root complex (LS1046ARDB).

1. Program outbound window 2 to access CCSR space and OCRAM of PCIe endpoint:

=>	mw.l	3500900	2
=>	mw.l	3500904	0
=>	mw.l	350090c	10000000
=>	mw.l	3500910	00000048
=>	mw.l	3500914	2ffffff
=>	mw.l	3500918	00000000
=>	mw.l	350091c	0
=>	mw.l	3500908	80000000

Programmed registers are shown below.



Figure 5. Outbound window 2 register dump

2. Configure DDR controller registers of PCIe endpoint:

NOTE

Take endianess of DDR registers into consideration while configuring these registers.

=> mw.l 4811080000 ff010000 => mw.l 4811080080 22030480 => mw.l 48110800c0 00000000

=>	mw.l	4811080008	ff010000
=>	mw.l	4811080084	22030080
=>	mw.l	48110800c4	00000000
=>	mw.l	4811080010	00000000
=>	mw.l	4811080088	00000000
=>	mw.l	48110800c8	00000000
=>	mw.l	4811080018	00000000
=>	mw.l	481108008c	00000000
=>	mw.l	48110800cc	00000000
=>	mw.l	4811080104	180077d1
=>	mw.l	4811080100	00111002
=>	mw.l	4811080108	4390fcf2
=>	mw.l	481108010c	97015900
=>	mw.l	4811080b28	00000480
=>	mw.⊥	4811080b2c	c1000000
=>	mw.⊥	4811080110	00400465
=>	mw.⊥	4811080128	efbeadde
=>	mw.⊥	4811080114	11114000
=>	mw.⊥	4811080118	30060103
=>	mw.⊥	4811080200	30060100
=>	mw.⊥	4811080208	30060100
=>	(((W . 1	4811080210	30060100
=>	mr.r]	481108011C	00021000
=>	mu l	4011000204	00021000
	mw.1	4811080200	00021000
	mw 1	48110802214	00021000
= >	mw 1	4811080228	00040000
=>	mw.l	4811080230	00040000
=>	mw.l	4811080238	00040000
=>	mw.l	4811080224	00004008
=>	mw.l	481108022c	00004008
=>	mw.l	4811080234	00004008
=>	mw.l	481108023c	00004008
=>	mw.l	4811080124	0000fe1f
=>	mw.l	4811080130	00000002
=>	mw.l	4811080160	02000000
=>	mw.l	4811080164	00144005
=>	mw.l	4811080260	00000000
=>	mw.l	4811080168	00000000
=>	mw.l	481108016c	00006026
=>	mw.l	4811080250	00682205
=>	mw.l	4811080400	5475c532
=>	mw.l	4811080404	d40bbbd4
=>	mw.l	4811080408	54f5c22e
=>	mw.l	481108040c	01405dd9
=>	mw.l	4811080170	0507098a
=>	mw.l	4811080174	09±67586
=>	mw.l	4811080190	110d0c0a
=>	mw.⊥	4811080194	Ue151412
=>	mw.⊥	4811080194	00000080
=>	mw.l	4811080110	00400465

3. Program outbound window 3 to access DDR region of PCIe endpoint:

=> mw.l 3500900 3 => mw.l 3500904 0 => mw.l 350090c 40000000 => mw.l 3500910 00000048 => mw.l 3500914 5fffffff => mw.l 3500918 80000000 => mw.l 350091c 0 => mw.l 3500908 80000000

Programmed registers are shown below.

=> md 350	0900					
03500900:	0000003	00000000	80000000	40000000	@	Outbound window base address Register
03500910:	00000048	5fffffff	80000000	00000000	Н	Limit Address Pagister
03500920:	00000000	00000000	00000000	00000000		 Linin Address Register
03500930:	00000000	00000000	00000000	00000000		Outhound translation address Register
03500940:	00000000	00000000	00000000	00000000		Subbuild fullshullon address Register
03500950:	00000000	00000000	00000000	00000000		

Figure 6. Outbound window 3 register dump

4. Read endpoint's DDR from root complex.

=> md	484200	00000				
484200	0000:	deadbeef	deadbeef	deadbeef	deadbeef	
484200	0010:	deadbeef	deadbeef	deadbeef	deadbeef	
484200	0020:	deadbeef	deadbeef	deadbeef	deadbeef	
484200	0030:	deadbeef	deadbeef	deadbeef	deadbeef	
484200	0040:	deadbeef	deadbeef	deadbeef	deadbeef	
484200	0050:	deadbeef	deadbeef	deadbeef	deadbeef	

□ LS1046ARDB DDR at 0x82000000

Configuring LS1046ARDB as PCIe root complex

Figure 7. Endpoint's DDR data dump on root complex

 Use below commands to load RAMboot image on endpoint's DDR memory at 0x82000000. See the "Binary Files of RAM boot" section of AN12081 for details on RAMboot image. On the root complex, the RAMboot image is read from a USB stick.

```
=> usb start
=> fatload usb 0 4842000000 <file_name>
```

The figure below shows the output of the above commands.

```
=> usb start
starting USB...
USB0: Register 200017f NbrPorts 2
Starting the controller
USB XHCI 1.00
USB1: Register 200017f NbrPorts 2
Starting the controller
USB XHCI 1.00
USB2: Register 200017f NbrPorts 2
Starting the controller
USB XHCI 1.00
scanning bus 0 for devices... 2 USB Device(s) found
scanning bus 1 for devices... 1 USB Device(s) found
scanning bus 2 for devices... 1 USB Device(s) found
      scanning usb for storage devices... 1 Storage Device(s) found
=> fatload usb 0 4842000000 u-boot.bin reading u-boot.bin
709818 bytes read in 51 ms (13.3 MiB/s)
```

Figure 8. Loading RAMboot image on endpoint's DDR

6. Trigger the boot-up process of PCIe endpoint by writing a non-zero value at 0x10010000 location of OCRAM:

=> mw.l 4820010000 17. Open endpoint (LS1046ARDB) serial port terminal to see U-Boot log, as shown in the figure below.

```
U-Boot 2016.092.0+ga06b209 (Aug 21 2017 - 11:32:33 +0530)
SoC: LS1046AE Rev1.0 (0x87070010)
Clock Configuration:
       CPU0 (A72):1600 MHz CPU1 (A72):1600 MHz CPU2 (A72):1600 MHz
       CPU3(A72):1600 MHz
               600 MHz DDR:
                                     2100 MT/s FMAN: 700 MHz
       Bus:
Reset Configuration Word (RCW):
       00000000: 0c150010 0e000000 00000000 00000000
       00000010: 11335506 40000012 40025000 c1000000
       00000020: 00200000 0000000 00000000 00238800
       00000030: 20124000 00003101 00000096 00000001
Model: LS1046A RDB Board
Board: LS1046ARDB, boot from QSPI vBank 4
CPLD: V2.2
PCBA: V2.0
SERDES Reference Clocks:
SD1_CLK1 = 156.25MHZ, SD1_CLK2 = 100.00MHZ
I2C:
     ready
DRAM: Detected UDIMM 18ASF1G72AZ-2G3B1
8 GiB (DDR4, 64-bit, CL=15, ECC on)
       DDR Chip-Select Interleaving Mode: CS0+CS1
SEC0: RNG instantiated
FSL SDHC: 0
MMC: no card present
MMC: block number 0x2801 exceeds max(0x0)
MMC/SD read of PPA FIT header at offset 0x500000 failed
PSCI: PSCI does not exist.
Waking secondary cores to start from fff0c000
All (4) cores are up.
Using SERDES1 Protocol: 4403 (0x1133)
Using SERDES2 Protocol: 21766 (0x5506)
NAND: 512 MiB
MMC: MMC: no card present
*** Warning - MMC init failed, using default environment
EEPROM: NXID v1
In:
     serial
Out: serial
Err:
     serial
SATA link 0 timeout.
AHCI 0001.0301 32 slots 1 ports 6 Gbps 0x1 impl SATA mode
flags: 64bit ncg pm clo only pmp fbss pio slum part ccc apst
Found 0 device(s).
SCSI: Net:
MMC read: dev # 0, block # 2080, count 128 ...
MMC: no card present
MMC: block number 0x8a0 exceeds max(0x0)
Fmanl: Data at 00000000ffel3670 is not a firmware
PCIe0: pcie@3400000 Endpoint: no link
PCIel: pcie@3500000 Endpoint: x1 gen1
PCIe2: pcie@3600000 Endpoint: no link
No ethernet found.
Hit any key to stop autoboot: 0
=>
```

Figure 9. U-Boot log on endpoint terminal

This section describes how to configure the LS1046ARDB as PCIe endpoint using CodeWarrior.

4.1 Board switch settings

For board switch settings, see Board switch settings.

4.2 RCW fields for PCIe configuration

The table below details the RCW fields to configure the LS1046ARDB as endpoint.

Field name	Description	Value
RCW[HOST_AGT_PEX]	Selects between Root Complex (RC) and Endpoint (EP) modes	3'b001: All Agent mode
RCW[SRDS_PRTCL_Sn]	Determines the link width	0x1133: SRDS_PRTCL_S1
		0x5506: SRDS_PRTCL_S2
RCW[SRDS_DIV_PEX_Sn]	Determines the link speed	2'b00: SRDS_DIV_PEX_S1
		2'b00: SRDS_DIV_PEX_S2

Table 4. POR parameters for PCIe controller (endpoint)

4.3 Minimal firmware on endpoint's OCRAM

The endpoint's OCRAM needs to be programmed with minimal firmware, using PBI commands. Minimal firmware is required on endpoint's OCRAM to perform the following functions:

- Program CSU registers to allow secure world to make various peripherals' registers accessible to non-secure world software and bus masters
- Write 0x0 at 0x10010000 location of OCRAM
- Poll for non-zero value at 0x10010000 location of OCRAM
- Jump to 0x82000000 location of DDR if 0x10010000 location of OCRAM is programmed to a non-zero value

Below is the assembly code of OCRAM firmware:

```
/* Update CSU registers */
ldr x2, =0x1510000
ldr x0, =0xb8
ldr w1, =0xFF00FF00
loopa: str w1, [x2,x0]
sub x0,x0,#0x4
cbnz x0,loopa
/* Write 0x0 at 0x10010000 */
```

```
ldr x2, =0x10010000
ldr x0, =0x0
str x0, [x2]
/* Poll for non-zero value */
loopb: ldr x0, [x2]
cbz x0, loopb
/* Jump to 0x82000000 */
ldr x30, =0x8200000
br x30
```

4.3.1 Convert assembly file into binary file

Perform these steps in Ubuntu to convert the assembly file into binary file:

1. Make object (.o) file from assembly (.S) file:

```
aarch64-linux-gnu-gcc -c <assembly_filename.S>
```

2. Make binary (.bin) file from object (.o) file:

```
aarch64-linux-gnu-objcopy -0 binary <object_filename.o> <binary_filename.bin>
```

4.4 Program scratch register

SCFG_SCRATCHRW2 register is programmed with the OCRAM location where GPP jumps to after executing boot ROM code.

Program SCFG_SCRATCHRW2 register (offset 0x01570604) with 0x10000000 (address of OCRAM). Minimal firmware is programmed at OCRAM address 0x10000000.

4.5 Program inbound windows

After link training, the endpoint needs to make its CCSRBAR register space and DDR SDRAM memory space accessible from root complex. This can be done by programming endpoint's inbound windows. In the current example, two inbound windows are used, which can be defined as follows:

- Inbound window 0: Represents a memory window that matches to BAR0 (BAR Match mode), which maps to 0x00000000000000 in CCSRBAR and OCRAM memory space
- Inbound window 1: Represents a memory window that matches to BAR1 (BAR Match mode), which maps to 0x000000000000000 in DDR SDRAM memory space

Program inbound window 0

Perform these steps to program inbound window 0:

- 1. Set the BAR Mask (size) in BAR0_MASK register:
- a. Write 0x1FFF_FFFF to BAR0_MASK at offset 0x1010.
- 2. Set up Base Address Register (BAR0):
 - a. Write 0x80000000 to BAR0 at offset 0x10.

```
3. Set up Index register:
```

- a. Write 0x80000000 to Index Register at offset 0x900
- 4. Set up Region Control 2 register:
 - a. Write 0xC0000000 to Region Control 2 register at offset 0x908.

- 5. Set up Translation Address registers:
 - a. Write 0x80000000 to window 0's Lower Target Address register at offset 0x918.
 - b. Write 0x00000000 to window 0's Upper Target Address register at offset 0x91C.



Figure 10. Endpoint inbound window 0

Program inbound window 1

Perform these steps to program inbound window 1:

- Set the BAR Mask (size) in BAR1_MASK register:
 a. Write 0x1FFFFFF to BAR1_MASK at offset 0x1014.
- 2. Set up Base Address Register (BAR1):
 - a. Write 0x00000000 to BAR1 at offset 0x14.
- 3. Set up Index register:
 - a. Write 0x80000001 to Index register at offset 0x900.
- 4. Set up Region Control 2 register:
 - a. Write 0xC0000100 to Region Control 2 register at offset 0x908.
- 5. Set up Translation Address registers:
 - a. Write 0x00000000 to window 1's Lower Target Address register at offset 0x918.
 - b. Write 0x00000000 to window 1's Upper Target Address register at offset 0x91C.



Figure 11. Endpoint inbound window 1

4.6 PBL configuration using CodeWarrior

This section explains the steps to configure and generate RCW and PBI binary file using QCVS tool in CodeWarrior for ARMv8. Using CodeWarrior, you can perform the following functions:

• Modify RCW values

- Write 0x10000000 (the location of OCRAM) on SCFG_SCRATCHRW2 register (at 0x1570604) and 0x00000000 on SCFG_SCRATCHRW1 register (at 0x1570600)
- Program inbound window
- Program OCRAM with minimal firmware using PBI commands

Follow these steps in CodeWarrior to perform PBL configuration:

1. Start CodeWarrior Development Studio for QorIQ LS series - ARM V8 ISA.



Figure 12. CodeWarrior welcome window

2. Choose **File > New > QorIQ Configuration Project** to create a new QCVS project.

ø	C/C++ - CodeWarrior Develop	pment Studio for (QorlQ LS series - AR!	/ V8 ISA							_	
File	Edit Source Refactor N	avigate Search	Project Run Proc	essor Expert Window H	lelp							
	New	Alt+Shift+N	QorlQ Config	uration Project	Ctrl+Shift+Q	· 🗇 🔻					Quick Access	😰 🗟 C/C++
	Open File		CodeWarrior	riery Executable Importer						- 0	BE Outline 😫 🖲 Make Target 🗐	Task List " 🗖
	Close	Ctrl+W	Makefile Proj	ect with Existing Code								§₂ ⊽
	Close All	Ctrl+Shift+W	C++ Project	eet with Existing Code							An outline is not available.	
	Save	Ctrl+S	C Project									
	Save As	Chill Shift C	Project									
	Save All Revert	Ctri+Shift+S	Convert to a	C/C++ Autotools Project								
	Nevert		Convert to a	C/C++ Project (Adds C/C+	+ Nature)							
	Move	53	Source Folder									
8	Refresh	F2	Folder									
-	Convert Line Delimiters To		B Header File									
	Print	Ctrl+P	File from Terr	plate								
	Switch Workspace		G Class									
	Switch workspace		🗂 Task									
	Incature		📑 Other		Ctrl+N							
124 	Import											
	Descention	Alta Catao	-									
	Properties	Alt+Enter	-									
_	Exit											
												6 V P B
				e li	ks 😅 Console 📖 Properties							ę. L
				0 items	^		D- ut	t a contrar				
				Description		Kesource	Path	Location	Type			
1												
1												
0 ite	ems selected										http://cache.nxp.co.mpositeConte	entiar 🔤 🐨

Figure 13. Creating a QCVS project

3. Specify project name. Click Next.

New QorlQ Configuration Project			×
Create a QorlQ Configuration Project			
Choose the location for the new project			
Project name: Is1046ardb_rcw			
Use default location			
Location: C:/Users/nxf26874/OneDrive - NXP/Project/windows7/c	others/codew	Browse.	
Choose file system: default \vee			
? < Back Next >	Finish	Cance	I

Figure 14. Specifying project name

4. Select LS1046A as the processor to be used. Click Next.

🏴 New QorlQ Config	uration Project				\times
Devices					
Select the processor	you would like to u	ise			
Processor to be used:					
type filter text					
> LA series					
 LS series 					
LS1012A					
LS1023A					
LS1026A					
LS1043A					
LS1044A					
LS1046A					
LS1048A					
LS1084A					
LS1088A					
LS2040A					
LS2044A					
LS2048A					
LS2080A					
LS2081A					
LS2084A					
LS2085A					
LS2088A					
Choose a silicon revi	sion: 1.0 ~				
Creates a new config revison is selected.	uration project for	the LS1046A proces	sor. By default the l	atest silicor	n

Figure 15. Selecting processor 5. Select **PBL – Preboot Loader RCW Configuration** as the QCVS component. Click **Next**.

Figure 16. Selecting PBL as toolset

6. Select Create default configuration to start with the default PBL configration or select Import configuration from an existing PBL file if you want to import PBL configuration from an SDK image (for example, rcw_xxxx.bin). Click Finish.

(N				_	
New QorlQ Configur	ation Project				X
BL configuration					
Choose PBL configurat	ion				
Basic Configuration					
Create default control	onfiguration				
O Import configura	ation from an existing PB	L file			
Read from targe	t				
Create default configu					
Create default confidu	ration for PBL.				
Create default configu	ration for PBL.				ſ
create default configu	ration for PBL.				ĺ
Create default configu	ration for PBL.				
Create delauit configu	ration for PBL.				
Create default configu	ration for PBL.				
Create delauit configu	ration for PBL.				
Create delauit configu	ration for PBL.				
Create default configu	ration for PBL.				
Create default configu	ration for PBL.				
Create default configu	ration for PBL.				,
Create delauit configu	ration for PBL.				
Create default configu	ration for PBL.				
Create default configu	ration for PBL.				
Create default configu	ration for PBL.				
Create default configu	ration for PBL.				
	ration for PBL.	facet a	Einich	Cancel	

Figure 17. Selecting PBL configuration

7. On **Properties** page of **Component Inspector** view, check clock settings of the LS1046ARDB.

QCVS - CodeWarrior Development Studio for QorlQ LS series - ARM V8 ISA File Frit: Navinate Search Project Run Processor Evnett Window Help							-	٥	×
						Quick Access	🖻 🗟 C/C	15	ocvs
Project Explorer 8 Connections View		Component Inspect	or - SoC 😤 🕲 Componen	nts Library		Rasic	Advanced	N	- 0
2 Is1046ardh row		Descention)	component	is clotdy				-	_
		Properties		14-1	Densile				
		Name		Value	Details				
		Component name		SoC					
		Silicon revision		LS1046A_v1_0					
		Clock settings							
		System Clock (S	SYSCLK)						
		Clock frequer	ncy [MHz]	66.7	66.7 MHz				
		 Differential Sys 	tem Clock (DIFF_SYSCLK		100.1411				
		Clock frequer	icy [MHz]	100.0	100 MHz				
		Memory Clock	(DDRCLK)	100.0	100 100				
		Clock frequer	icy [MHZ]	100.0	100 MHz				
		SerDes 1 Reference	ence Clocks	100.0	100.1411				
		SD1_REF_CLK	1 [MHZ]	100.0	100 MHz				
🕾 Components - Is1046ardb_rcw 🕴	8 🗰 🍓 🔍 🗖	SD1_REF_CLK	2 [MHZ]	100.0	100 MHZ				
✓ ➢ Generator_Configurations									
> 🗁 OSs									
v 😓 Processors		L							
> 🛞 SoC:LS1046A_v1_0									
✓ ➢ Components		L							
BL:PBL									
Problems 8								§9	- 0
0 items									
Description A Recourse Path	Location	Type							
resource Patri	Location	type							

Figure 18. Clock settings of LS1046ARDB

8. Select the PBL component under Components folder in Components view to configure RCW fields.

The Case Andregate Sector Project for Project on Project on Project on PRL 11 Components library Back Advanced 11 Components library Project Explorer Components Inspector - PRL 11 Components library Back Advanced 11 Components library Project Explorer Project Explorer Project Explorer Value Details Project Explorer Project Explorer Project Explorer Value Details Project Explorer Project Explorer Project Explorer Project Explorer Value Details Project Explorer Project Explorer Project Explorer Project Explorer NOR Flash 16-bit Port size Project Explorer Project Explorer NOR Flash 16-bit Port size Project Explorer Project Explorer Project Explorer	CVS - CodeWarrior Development Studio for QorlQ LS series - ARM V8 ISA				- 6	×
Project Explore Component Inspector - PBL III Components Library Baic Advance Image: Components Library Processor Proble Ad. JTag speed (k Image: Configuration Pail Pail <td></td> <td></td> <td></td> <td></td> <td>Quick Access</td> <td>+ 🐻 QCVS</td>					Quick Access	+ 🐻 QCVS
Target connections Processor Proble AL, /Tag speed (L, IS1046A ortap 10230 Processor Proble AL, /Tag speed (L, IS1046A ortap 10230 PL Configuration Secretar PL Land Protocol Configuration <tr< td=""><td>Connections View 😫</td><td>😑 🗖 🗞 Component Inspector - PBL 🛛 🗞 Componen</td><td>nts Library</td><td></td><td>Basic Advanced</td><td></td></tr<>	Connections View 😫	😑 🗖 🗞 Component Inspector - PBL 🛛 🗞 Componen	nts Library		Basic Advanced	
Image: Source Value Details Processor Probe Ad. //lig speed (k. PBL PBL <td< td=""><td>Target connections</td><td>Properties Import Validation</td><td></td><td></td><td></td><td></td></td<>	Target connections	Properties Import Validation				
Image: Components - Is1046andb. Jr.w 32 Image: Components - Is1046andb. Jr.		Name	Value	Details		^
Problems II Image: Section Problems II Image: Section Problems II Image: Section Problems II Image: Section Problems II Image: Section Problems II Image: Section Problems II Image: Section Problems II Image: Section Problems II Image: Section Problems II Image: Section Problems II Image: Section Problems II Image: Section Problems III Image: Section Problems III Image: Section Problems III Image: Section Problems III Image: Section Problems III Image: Section Problems III Image: Section Problems III Image: Section Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems III Image: Problems IIII Image: Problems IIII Image: Problems IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	December 7 Decks 7 Decks Ad. (The second fit	Device	PBL	PBL		
Is to task wide to 2.30 Is to task wide to 2.30 <td>Processor Probe I Probe Ad Jiag speed (k</td> <td>RCW Source</td> <td>NOR Flach 16-bit Port size</td> <td></td> <td></td> <td></td>	Processor Probe I Probe Ad Jiag speed (k	RCW Source	NOR Flach 16-bit Port size			
SerDes PLL and Protocol Configuration SerDes PLL and Protocol Configuration Boot Configuration SerDes PLL and Protocol Configuration SerDes PLC and Protocol Configuration SerDes PL Configuration	LS1046A Cwtap 10230	> PLL Configuration	NOR HUSH TO DICPORT SIZE			
		SerDes PLL and Protocol Configuration				
Components Islow Components Islow So Cost So Cost So Cost<		> Misc. PLL-Related Configuration				
Components - Is1046ardb_row ISI Configuration Configu		> Boot Configuration				
 Components - Is1046ardb.rcw 32 Components - Is1046ardb.r		> Clocking Configuration				
Second Purpose Information > Generator_Configuration > Generator_Configuration > Socup A Pin Configuration > Group A Pin Configuration > Pi		> Memory and High-Speed I/O Configurat				
Components - Is1046ard p_row 12 Configuration Generator_Configuration		> General Purpose Information				
© Components - Is1046ardb r/cw III > © Generator, Configurations % LS1046a/t0_Cnf > © OSS > © Processors > © Components © Poblems III Description Problems III Description Percent Problems III Description Percent Percent Percent Problems III Description Percent Problems III Description Percent Percent Percent Problems III Description Percent Problems III Description Percent Percent Percent Problems III Description Percent Pe		 Pin Multiplexing Configuration 				
Secretary Configurations So Case of Configuration in the interval of the interval o	😢 Components - Is1046ardb_rcw 🕴 📄 🐨	Group A Pin Configuration				
SIC3Secific Configuration Processors Pollosi Pollosi Pollosi Pollosi Pollosity Configuration Expansion Pollosity Configuration Expansion Pollosity Configuration Expansion Pollosity Configuration Expansion Pollosity Configuration Expansion Pollosity Configuration Expansion Pollosity Configuration Expansion Pollosity Configuration Expansion Pollosity Configuration Expansion Pollosity Configuration Expansion Pollosity Configuration Expansion Pollosity Configuration Expansion 	Generator_Configurations	> Group B Pin Configuration				
> Poles S > Processors > © Components © PolePEL P	IS1046A_v1_0_Cnf	> SoC-Specific Configuration				
Polema SI Polema SI Port Mathematical Security of the secure of the s	> 🗁 OSs	> PLL and Clocking Configuration Expansion	c			
> ● Soci51046A,v1,0 ○ PBL DBL ● PBL/DBL ○ Iffet 0 Output Format Binary ● PbL/DBL ● Iffet 0 E Problems 32 ● Iffet 0 D Items ● Iffet 0 Description Resource Path Location Image: Path Image: Path Image: Path Image: Path Image: Path	v 💩 Processors	> PBI Data				_
Components Other Omega PBL/BL PBL/BL Other Image: Components Problems S2 0 0 0 Hems 0 0 Description Resource Path Location Type 0 0	> 🕲 SoC:LS1046A_v1_0	✓ PBL Data	0	4		
Poblems SI Ditems Ditems Description Resource Path Location Type	Components	Output Format	Binany			
Problems Problems Poblems Poblems Poblems Pescuree Path Location Type	BLPBL	output rollinat	billary			~
Problems 22 Otens Description Resource Path Location Type						
Description Resource Path Location Type	Problems Ditems				9	~
	Description Resource Path Loca	ation Type				
				A 1		

Figure 19. Configuring RCW fields

9. On **Import** page of **Component Inspector** view, click **Load from file** or **Read from target** to load RCW+PBL data from computer or from target, respectively. See *Layerscape Software Development Kit Documentation* for building the LS1046ARDB SDK images.

, QCVS - CodeWarrior Development Studio for QorlQ LS series - ARM V8 ISA File Edit Navigate Search Project Run Processor Expert Window Help			- • ×
1 → 2 (a) (b) (b) →			Quick Access 😰 🗟 C/C++ 🐻 QCVS
Project Explorer III Connections View 😫		🗞 *Component Inspector - PBL 🕴 🗞 Components Library	Basic Advanced 🕒 🌣 🗖 🗖
Target connections		Properties Import Validation	
B X	0 🗶	Get PBL data from file or from target - data and data format will be inserted into the input area below	ι.
Processor Probe T Probe Ad JTag speed (k		Load from file Read from target	
LS1046A cwtap 10230		Input data	
		Format: xxd Object Dump ~	
		Endianness: Little Endian \vee	
			^
Scomponents - Is1046ardb_rcw			
Generator_Configurations M LS1046A v1 0 Cnf			
> 😂 OSs			
Processors Processors		4	×
✓ Soc:LS1040A_V1_0 ✓ Soc:LS1040A_V1_0			· · ·
IB PBL:PBL		111 portue	
			- 0
Problems 😫			9 V - C
0 errors, 2 warnings, 0 others			
Description Assource Path	Location	Туре	
> & Warnings (2 items)			
			8

Figure 20. Accessing PBL data from file or from target

10. Click **Import** to update PBL configuration. In the current project, PBL data is loaded from the LSDK 18.03 image, rcw_1600.bin.

P QCVS - CodeWarrior Development Studio for QorlQ LS	series - ARM V8 IS	Ā								- 0	\times
File Edit Navigate Search Project Run Processor E	xpert Window H	Help									
	⇒ ¥								Quick Access	४ 4@ C/C++	CCVS 📷
Project Explorer Explorer Connections View 😂				*Component Inspective	ctor - PBL 🛛 🗞 Compone	ents Library			Basic A	dvanced 😬	~
Target connections				Properties Import	Validation						
Processor Probe T Probe Ad JTag speed (k LS1046A cwtap 10230			0 🕱	Get PBL data from file	e or from target - data and Read from target ect Dump	d data format will b	e inserted into the inp	put area below.			
 Components - Is1046ardb_rcw № Generator_Configurations IS1046A_v1_0_Cnf OSs SocLS1046A_v1_0 SocLS1046A_v1_0 PROENES 		8 8 8	▽ □ □	Format. 201000 Endiances: Utile En 000000010: ao 00000010: 00 00000020: 40 00000030: 00 00000060: 09 000000070: 09 000000070: 09 000000070: 09 000000000: 09 000000000: 09 000000000: 89 <	wisin > sign > 55 an 55 1 en 01 00 00 00 00 00 00 00 00 00 00 00 00 00 02 32 83 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 05 7 04 18 00 <td>00 0c 15 00 00 11 33 55 00 00 00 00 00 20 12 40 01 09 57 01 00 09 57 06 10 09 57 04 9e 09 57 04 9e 09 57 04 9e 09 50 08</td> <td>10 0e 00 00 00 06 40 00 00 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 440 10 00 00 00 440 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td> <td>.U.U. </td> <td></td> <td></td> <td>*</td>	00 0c 15 00 00 11 33 55 00 00 00 00 00 20 12 40 01 09 57 01 00 09 57 06 10 09 57 04 9e 09 57 04 9e 09 57 04 9e 09 50 08	10 0e 00 00 00 06 40 00 00 12 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 440 10 00 00 00 440 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.U.U. 			*
Problems 23										24	÷ ⊐ 🛛
Description	Resource	Path I	ocation	Type							
> & Warnings (2 items)	Resource	rau L	ocontroll	type							
								<u></u>			

Figure 21. Imported RCW file

11. Set **SRDS_PRTCL_S2** as 5506 to have PCIe controller 1/2/3 on SerDes2 lanes. Set **HOST_AGT_PEX** as 3'b001 to program all three PCIe controllers as Agent/Endpoint mode. You can configure other RCW fields as per your requirements. Click the link next to **PBI Data input** field to add/modify any PBI commands.

P QCVS - Code File Edit Navi	Warrior Development Stu igate Search Project F	idio for QorlQ LS seri Run Processor Expe	ies - ARM V8 ISA rt Window H	elp								- 0	3 ×	<
📑 🕶 🔛 🐚 🛛) 💁 🕶 😂 🛷 💌 🕗 🔻	- 9 - 9	*							Quick	k Access	🗟 C/C+	+ 👪 QC	٧S
Project Explo	orer 🖪 Connections View	12				S *Component Inspec	tor - PBL 🕴 🗞 Componen	ts Library			Basic Ad	vanced) v •	
Target conner	tions					Properties Import	alidation							
						Name		Value	Details					^
💽 🖻 🗙					0 🕱	RCW Source		Quad SPI (QSPI)	Warning: Please make sure					
Processor	Probe T., Probe Ad.,	JTag speed (k				> PLL Configurati	on						_	
LS1046A	cwtap	10230				SerDes PLL and	Protocol Configuration							
						SRDS_PRTCL_	S1 [128-143]	060001000100110011 - 1133						
						SRDS_PRTCL	S2 [144-159]	060101010100000110 - 5506						
						SRDS_PLL1_R	EF_CLK_SEL_S1 [160-161]	1 0						
						SRDS_PLL1_R	EF_CLK_SEL_S2 [162-163]	0 D						
						SRDS_PLL_PD	_S1 [168-169]	0b00 - PLL1 and PLL2 are n						
						SRDS_PLL_PD	_S2 [170-171]	0b00 - PLL1 and PLL2 are n						
						SRDS_DIV_PE	X_S1 [176-177]	0b00 - Can train up to a ma						
						SRDS_DIV_PE	X_S2 [178-179]	0b00 - Can train up to a ma						
				D 01 8	V D R	> Misc. PLL-Relat	ed Configuration							
 Components 	- Is104oardD_rcw 23					> Boot Configura	tion							
 Generator 	r_Configurations					> Clocking Config	juration							
1046 1046	6A_v1_0_Cnf					~ Memory and H	igh-Speed I/O Configurat							
> 🗁 OSs						HOST_AGT_P	EX [264-266]	0b001 - All agent mode						
 Processor 	S					> General Purpos	e Information							
> 😗 SoC:LS	1046A_v1_0					> Pin Multiplexin	g Configuration							
Compone	ents					> SoC-Specific Co	nfiguration							
B PRE-PB	L					> PLL and Clockin	g Configuration Expansic						_	
						✓ PBI Data								
						PBI Data input		(click here and press [] butt	Warning: Please make sure					
						~ PBL Data								~
🕄 Problems 🛙												9		
0 errors, 2 warni	ngs, 0 others													
Description	^	1	Resource	Path	Location	Туре								
> 💧 Warnings	(2 items)													

Figure 22. Selecting SerDes protocols and switching all PCIe controllers to Agent/ Endpoint mode

12. Program SCFG_SCRATCHRW2 register as 0x10000000 at 0x1570604.

📎 Component Inspector - PBL 🛛 🗞 Compone	ents Library	Basic Advanced 🏻 🖱 🗖
Properties Import Validation		
Name	PBI Data input	
SRDS_PLL1_REF_CLK_SEL_S2 [162-16	Select PBI command CCSR Write (4-byte)	
SRDS_PLL_PD_S1 [168-169]		
SRDS_PLL_PD_S2 [170-171]	Command parameters	
SRDS_DIV_PEX_S1 [176-177]	System address (0x) 570604	
SRDS_DIV_PEX_S2 [178-179]	Data (0x) 10000000	
> Misc. PLL-Related Configuration		
> Boot Configuration	🕂 Add Command 🛛 🗹 Modify Command	
> Clocking Configuration	Added PBI Commands:	🗶 🦄 🔶 🕂 🗗 🔜
 Memory and High-Speed I/O Configu 	CCSR Write to 0x57015c data=0x40100000	
HOST_AGT_PEX [264-266]	CCSR Write to 0x570600, data=0x00000000	
> General Purpose Information	CCSR Write to 0x570604, data=0x10000000	
> Pin Multiplexing Configuration	CCSR Write to 0x570178, data=0x0000e010	
SoC-Specific Configuration	CCSR Write to 0x180000, data=0x00000008	
EC1 [416-418]	CCSR Write to 0x570418, data=0x0000009e	
EC2 [419-421]	CCSR write to 0x57041c, data=0x0000009e	
LVDD_VSEL [422-423]	CCSR Write to 0x9/10420, data=0x80104e20	
I2C_IPGCLK_SEL [424-424]	CCSR Write to 0xeb08dc, data=0x00502880	
EM1 [425-425]	CCSR Write to 0x570158, data=0x00000300	~
EM2 [426-426]		
EMI2_DMODE [427-427]	Restore Apply	
EMI2_CMODE [428-428]	Like this property to add DPI commands to the DPI image. Commants (#) can be added in Pau in	anda
USB_DRVVBUS [429-429]	Version specific item: Settings supported only for devices using Reset Configuration Word (RCW)	ioue.
USB_PWRFAULT [430-430]	Warning: Please make sure "QSPI endianness change" PBI command is the last PBI command.	· · · · · · · · · · · · · · · · · · ·

Figure 23. Programming SCFG_SCRATCHRW2 register

13. Choose **CCSR Write** (4-byte) as the PBI command. Enter 570158 as the system address and 00001000 as the data to program ALTCBAR register with OCRAM base address. Click **Add Command**.

S Component Inspector - PBL 🛛 S Component	nts Library Basic Adv	/anced 🎦 🌣 🗆 🗖
Properties Import Validation		
Name ^ BOOT_HO [201-201]	PBI Data input Select PBI command CCSR Write (4-byte)	
SB_EN [202-202] IFC_MODE [203-211]	Command parameters	
Clocking Configuration Memory and High-Speed I/O Configu HOST AGT DEV (264-266)	System address (0x) 570158 Data (0x) 00001000	
General Purpose Information Pin Multiplexing Configuration	Add Command	
Group A Pin Configuration Group B Pin Configuration SoC-Specific Configuration PIL and Clocking Configuration Expansion	Added PBI Commands: ACS Write to 0x500918, data=0x00000000 ACS Write to 0x500908, data=0x000100c0 CCSR Write to 0x570158, data=0x00002000 ACS Write to 0x100524, data=0x00000000	
PBI Data PBI Data PBI Data input	ACS Write to 0x140544, data=0x00000000 CCSR Write to 0x570158, data=0x00001000	↓
~ PBL Data Offset	Restore Apply	
Output Format Additional Binary Data	Use this property to add PBI commands to the PBL image. Comments (#) can be added in Raw mode. Version specific item: Settings supported only for devices using Reset Configuration Word (RCW). Warning: Please make sure "QSPI endianness change" PBI command is the last PBI command.	~ ~
© 0x0 © do_AngelSVC() at syscalls.c:771 0x8	10021e88 🛽 🖻 main.S 🖾	
<pre># this directive allows the linker # which is the entry point .globl main #</pre>	r to see the "main" label	^
<pre># this directive allows the eclips .func main </pre>	se gdb to see a function called "main"	~

Figure 24. Programming ALTCBAR register

14. Choose **ACS Write from file** as the PBI command. Click **Load from file** and load binary file that needs to be programmed on OCRAM. Choose 64 as the byte count. Click **Add command**.

NOTE

The file size must be in multiples of 128 bytes.

💊 Component Inspector - PBL 🛛 🗞 Component	nts Library	Basic Advanced 🎦 🏹 🗖 🗖
Properties Import Validation		
Name ^	PBI Data input	
SRDS_PRTCL_S1 [128-143]	Select PBI command	ACS Write from file V
SRDS_PRTCL_S2 [144-159]	Command parameter	rs
SRDS_PLL1_REF_CLK_SEL_S1 [160-16	Contract data (D.)	000000
SRDS_PLL1_REF_CLK_SEL_S2 [162-16	System address (Ux)	000000
SRDS_PLL_PD_S1 [168-169]		0a0000141f2003d50000008200000000
SRDS_PLL_PD_S2 [170-171]		e04c09000000000e04c09000000000
SRDS_DIV_PEX_S1 [176-177]	Data (0x)	414238d53f3000f1a00000543f2000f1
SRDS_DIV_PEX_S2 [178-179]		600100543f1000f1a001005400c01ed5
Misc. PLL-Related Configuration		00113ed5000c40b200111ed55f111ed5 v
> Boot Configuration	Pute Count	64
> Clocking Configuration	Byte Count	64 ~
 Memory and High-Speed I/O Configu 		
HOST_AGT_PEX [264-266]		Add Command
> General Purpose Information	Added PBI Commands	: 🗙 🔆 🕁 🗾 🔂
> Pin Multiplexing Configuration	CCSR Write to 0x5701	58, data=0x00000300 ^
> SoC-Specific Configuration	ACS Write to 0x40089	0, data=0x01048000
> PLL and Clocking Configuration Expa	ACS Write to 0x50089	0, data=0x01048000
✓ PBI Data	ACS Write to 0x60089	∪, data=0x01048000 ~~~~
PBI Data input	Destaux Amelia	
✓ PBL Data	Restore Apply	
Offset	Use this property to ac	d PBI commands to the PBI image. Comments (#) can be added in Raw mode.
Output Format	Version specific item: S	Settings supported only for devices using Reset Configuration Word (RCW).
Additional Binary Data 🗸	Warning: Please make	sure "QSPI endianness change" PBI command is the last PBI command.

Figure 25. Adding PBI commands on PBI data from a binary file

Similarly, other PBI commands can be added or modified in QCVS tool. The figure below shows the additions and modifications made to PBI data.



Figure 26. Additions and modifications to PBI data

15. Click Generate Processor Expert Code to generate the PBL file.

Appendix: Using C29XPCIE-RDB as PCIe root complex

P QCVS - CodeWarrior Development Studi File Edit Navigate Search Project Ru	o for QorlQ LS ser n Processor Expe	ies - ARM V8 ISA rt Window He	elp					- o ×
😂 🕶 📾 🔞 💁 🕶 😂 🛷 💌 🖢 🗸	8 • • • • • •	•						Quick Access 🔡 🗟 C/C++ 📕 QCVS
Project Explorer 🖪 Connections View 😫 🗖 🗖 📢					S Component Inspect	or - PBL 🛛 🗞 Components Library		Basic Advanced 🕒 🏹 🗖
Target connections					Properties Import	alidation		
Processor Probe T_ Probe Ad_ JTag speed (k_ LS1046A cwtap 10230			Name SRDS_PRTCL_ SRDS_PLL1_R SRDS_PLL_PL SRDS_PLL_PD SRDS_PLL_PD SRDS_DIV_PE SRDS_DIV_PE SRDS_DIV_PE	Value	Details	^		
					Boot Configura	tion		
					> Clocking Config	uratior		
					~ Memory and H	gh-Spe		
Secomponents - Is1046ardb_rcw			E 🖬 🖸	~ ~ 8	HOST_AGT_P	EX [264 0b001 - All agent mode		
 Generator_Configurations 			—		> General Purpos	e Infor		
LS1046A_v1_0_Cnf					> Pin Multiplexin	g Confi		
> 🔄 OSs					> SoC-Specific Co	nfigura		
Processors	Generat	te Process	or Expert Code	•	> PLL and Clockin	g Conf		
> 199 SoctS1046AçV10 ≪ © Components 19 PBLPBL			V PBI Data	foliate have and server 1.2 houts	Warrian Diasas maka sus			
			PBI Data Input	(click here and press [] butt	. warning: Please make sure			
			Offset	0	1			
			Output Format	Binary				
			Additional Binary Data (click here and press [] butt			×		
								• •
Problems 😂								₽ ° ⊔ ⊟
0 errors, 2 warnings, 0 others		_			-			
Description		Resource	Path	Location	Туре			
> & Warnings (2 items)								
								8

Figure 27. Generating PBL file

16. Program QSPI flash with generated RCW+PBI binary file at 0x0 offset. Endpoint is ready to be plugged into the setup. Continue with instructions provided in Initialize endpoint's DDR.

5 Appendix: Using C29XPCIE-RDB as PCIe root complex

Alternatively, this use case setup can be created by using C29XPCIE-RDB as root complex and LS1046ARDB as endpoint. Following subsections describe configuration of C29XPCIE-RDB as PCIe root complex.

5.1 Board switch settings

The C29XPCIE-RDB can be used as a root complex by configuring it in Standalone Host mode. The DIP switch settings of the C29XPCIE-RDB for Standalone Host mode are shown in the table below.

DIP switch	Settings	Notes
SW4[1:8]	01011000	0 indicates ON
SW5[1:8]	11110000	1 indicates OFF
SW6[1:8]	00001111	
SW7[1:8]	10011111	
SW8[1:8]	00001011	

Table 5. DIP switch settings

5.2 Program outbound window

Programming outbound window involves mapping local address space to PCIe address space. This mapping can be performed by programming Local Access Window (LAW) registers of the C29x processor. For more details on LAW registers, see *C29x Crypto Coprocessor Family Reference Manual*.

U-Boot programs LAW2 registers for mapping to PCIe address space. Below are some details of LAW2 registers:

- Base address: 0x8000000
- Size: 512 MB
- Target ID: 0x2

Perform these steps to program an outbound window on the C29XPCIE-RDB:

- 1. Program PEX_PEXOWAR1 register as 0x8004401c:
 - PEX_PEXOWAR1[EN] = 1'b1
 - PEX_PEXOWAR1[RTT] = 4'b0100, memory read
 - PEX PEXOWAR1[WTT] = 4'b0100, memory write
 - PEX_PEXOWAR1[OWS] = 6'b011100, size 512 MB
- 2. Program PEX_PEXOWBAR1 register as 0x00080000
 - Window base address = 0x8000000
- 3. Program PEX_PEXOTAR1:
 - a. Set translation of outbound window as 0x00000000 (see figures below) using the following command:

=> mw.l e000ac20 0

This window can be used to access DDR controller registers and OCRAM space of endpoint. Follow instructions in Initialize endpoint's DDR to initialize the DDR controller of the endpoint.

=> md e000	Dac00						
e000ac00:	00000000	00000000	00000000	00000000			Outbound translation address register
e000ac10:	80044023	00000000	00000000	00000000	@#		
e000ac20:	00000000	00000000	00080000	00000000			Outbound window base address register
e000ac30:	8004401c	00000000	00000000	00000000	@		Outbound window attributes register
e000ac40:	00000000	00000000	000efc00	00000000		-	Outoound window autioutes register
e000ac50:	8008800f	00000000	00000000	00000000			

Figure 28. Setting translation of C29XPCIE-RDB outbound window as 0x00000000



Figure 29. C29XPCIE-RDB outbound window with translation address as 0x00000000

b. Change translation of outbound window to 0x80000000 (see figures below) using the following command:

=> mw.l e000ac20 00080000

Appendix: Hardware and software resources

Endpoint's DDR region can now be accessed through this window. Follow instructions in Initialize endpoint's DDR to load boot loader on endpoint's DDR.

=> md e00	0ac00					
e000ac00:	00000000	00000000	00000000	00000000		Outbound translation address register
e000ac10:	80044023	00000000	00000000	00000000	@#	
e000ac20:	00080000	00000000	00080000	00000000		Outbound window base address register
e000ac30:	8004401c	00000000	00000000	00000000	@	Outbound window attributes register
e000ac40:	00000000	00000000	000efc00	00000000		Outoound window attributes register
e000ac50:	8008800f	00000000	00000000	00000000		

Figure 30. Setting translation of C29XPCIE-RDB outbound window as 0x80000000



Figure 31. C29XPCIE-RDB outbound window with translation address as 0x80000000

NOTE

For programming an outbound window, bus master and memory space must be enabled in PCIe configuration space of C29XPCIE-RDB.

6 Appendix: Hardware and software resources

The table below shows the hardware and software tools you may require for use case setup.

 Table 6.
 Hardware and software tools

ΤοοΙ	How to access?						
Hardware tools							
QorIQ LS1046A reference design board	www.nxp.com						
C29x Crypto Coprocessor PCI Express Adapter Platform	www.nxp.com						
CodeWarrior TAP	www.nxp.com						
QorIQ LS Processor Probe Tips for CodeWarrior TAP	www.nxp.com						
Power Architecture processor (COP) Probe Tips for CodeWarrior TAP	www.nxp.com						
Software tools							
CodeWarrior Development Software for ARM v8 64-bit based QorIQ LS series Processors	www.nxp.com						
CodeWarrior Development Studio for Power Architecture Processors	www.nxp.com						
Layerscape Software Development Kit	www.nxp.com						
Linux SDK for QorlQ Processors	www.nxp.com						

7 Appendix: Related documentation

The table below lists documents that provide additional information related to the concept explained in this document.

Document	Description	How to access?
QorlQ LS1046A Reference Design Board Getting Started Guide (LS1046ARDBGSG)	Describes different components of the LS1046ARDB and explains how to set up and boot the board	www.nxp.com
QorlQ LS1046A Reference Design Board Reference Manual (LS1046ARDBRM)	Explains the LS1046ARDB interfaces and configuration	www.nxp.com
QorlQ LS1046A Reference Manual (LS1046ARM)	Provides a detailed description of the LS1046A multicore processor and its features, such as the memory map, serial interfaces, power supply, chip features, and clock information	www.nxp.com
QorlQ LS1046A Data Sheet (LS1046A)	Contains information on LS1046A pin assignments, electrical characteristics, hardware design considerations, package information, and ordering information	www.nxp.com
LS1046A Chip Errata (LS1046ACE)	Describes the latest fixes and workarounds for the chip. It is strongly recommended that this document is thoroughly researched prior to starting a design with the chip.	Contact your NXP sales representative
RAM Boot using CodeWarrior on LS1046ARDB Application Note (AN12081)	Explains how to deploy U-Boot directly to the DDR of the LS1046ARDB using CodeWarrior	www.nxp.com
C29x PCIe Card Quick Start Guide (C29XPCIEQS)	Explains C29x PCIe board settings and physical connections needed to boot the board	www.nxp.com
C29x PCIe Card User Guide (C29XPCIEUG)	Explains the C29x PCIe board interfaces and configuration	www.nxp.com
C29x Crypto Coprocessor Family Reference Manual (C29XRM)	Defines the functionality of the NXP C29x family	www.nxp.com
Layerscape Software Development Kit Documentation (LSDK-REV-yy-mm)	Describes LSDK, which is a complete Linux kit for NXP QorIQ Arm-based SoCs and the reference and evaluation boards available for them	www.nxp.com
CodeWarrior Development Studio for QorIQ LS series - ARM V8 ISA, Targeting Manual (CWARMv8TM)	Explains how to use the CodeWarrior Development Studio for QorIQ LS series - ARM V8 ISA product	www.nxp.com
CodeWarrior TAP Probe User Guide (CWTAPUG)	Provides details of CodeWarrior® TAP, which enables target system debugging via a standard debug port (usually JTAG) while connected to a developer's workstation via Ethernet or USB	www.nxp.com

Table 7. Related documentation

8 Revision history

The table below summarizes revisions to this document.

Table 8. Revision history

Revision	Date	Topic cross-reference	Change description		
Rev. 0	11/2018		Initial public release		

How to Reach Us:

Home Page: nxp.com

Web Support: nxp.com/support Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP 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 NXP 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. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, Freescale, the Freescale logo, CodeWarrior, Layerscape, and QorlQ are trademarks of NXP B.V. All other product or service names are the property of their respective owners. Arm and Cortex are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

© 2018 NXP B.V.

Document Number AN12246 Revision 0, 11/2018



