FTF FREESCALE TECHNOLOGY FORUM 2014

Introduction to Pre-Boot Loader Supported by QorIQ Processors

FTF-NET-F0152

Zhongcai Zhou | Application Engineer

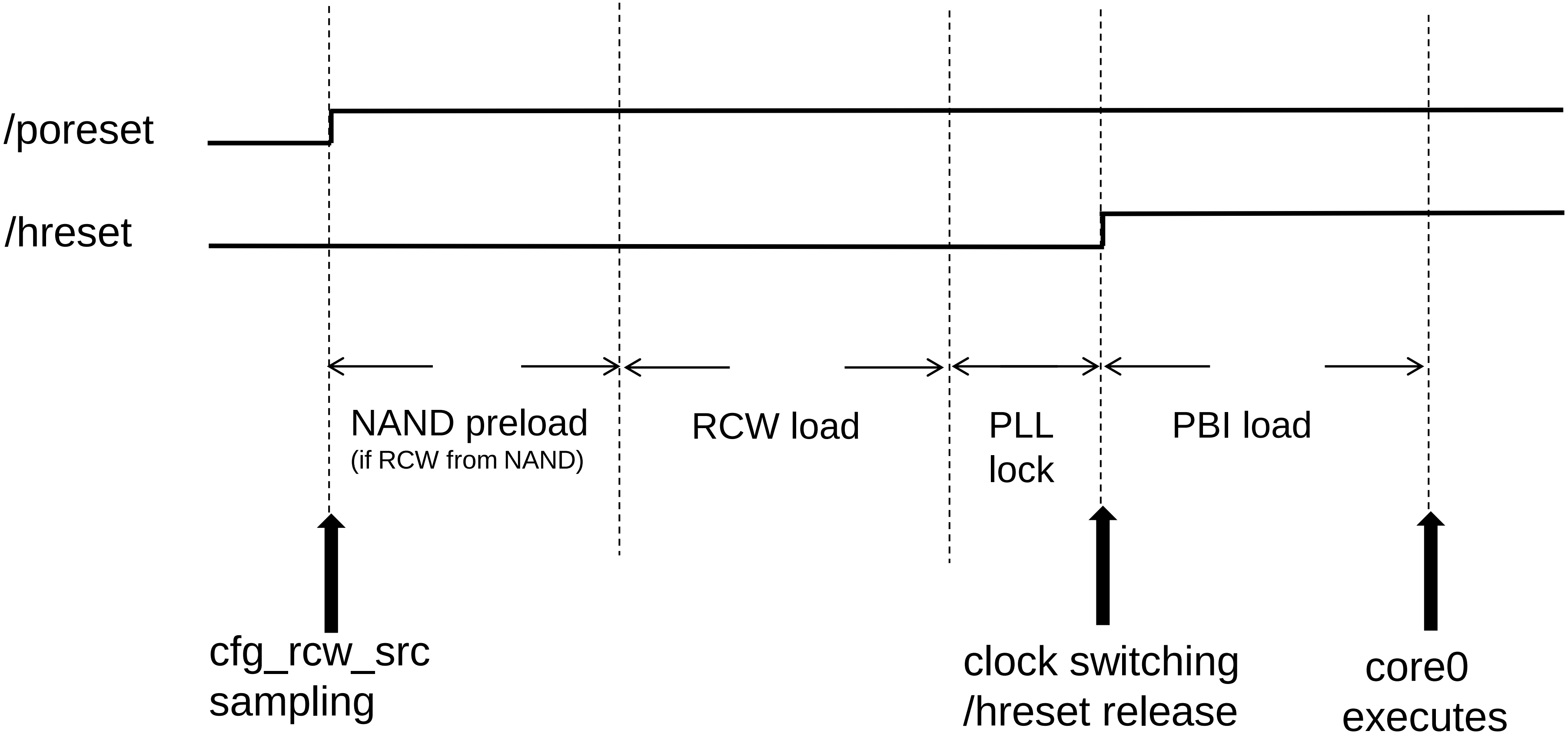
APR.2014

External Use

Introduction

- What does Pre-Boot Loader (PBL) do?
  - Device configuration
  - Initialization before the core fetches instructions
- History
  - Before PBL, how did we do these?
  - MPC8548
    - Device configuration
      - Pins strapping
    - Initialization
      - I2C Boot sequencer
- Improvements
  - Greatly reduce the number of pin strapping
  - Expand the sources of boot sequencer from I2C only to I2C, eSPI, eSDHC, NAND flash, NOR flash

PBL in the Power Up Sequence



Description of PBL Functionality

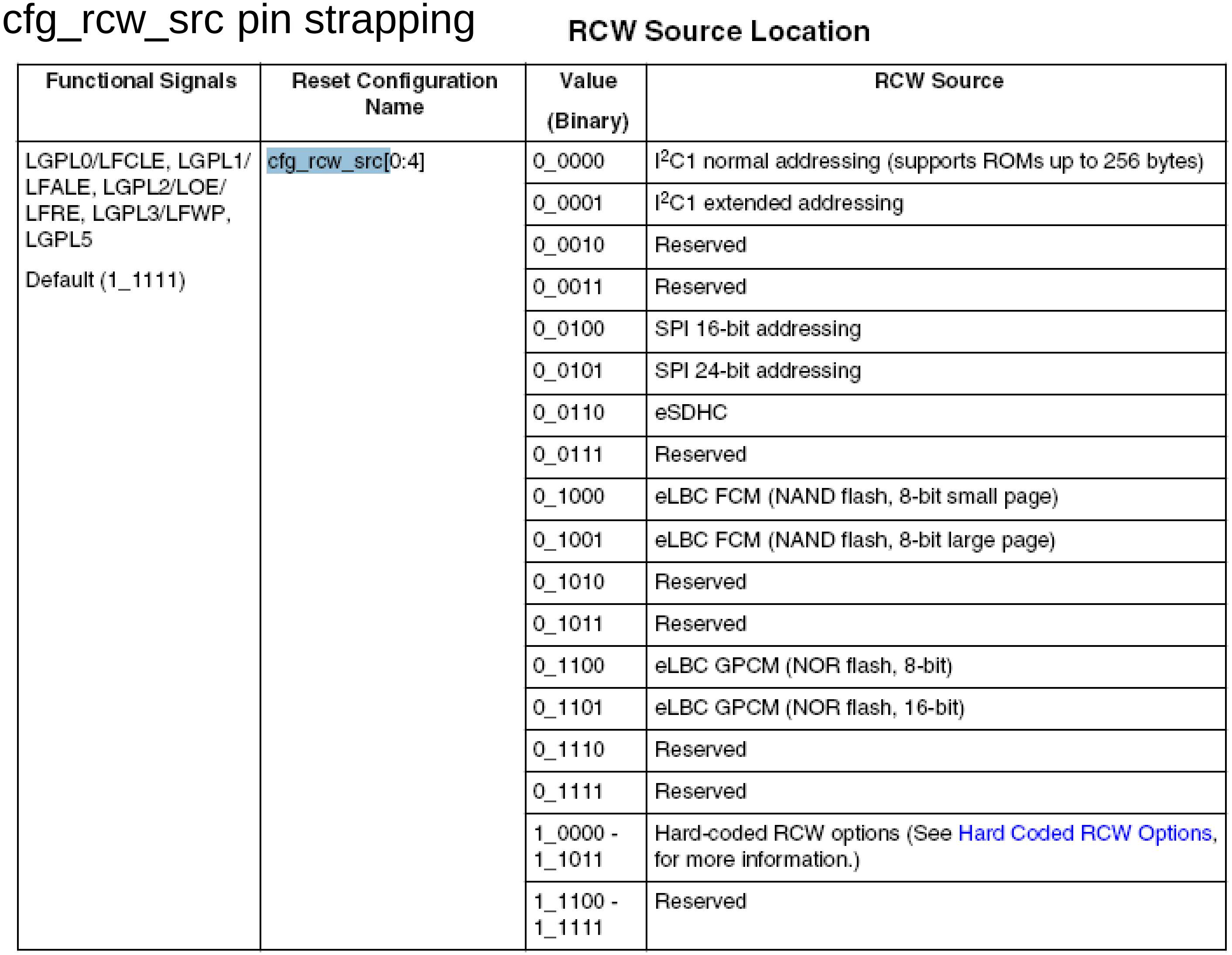
- PBL RCW phase
  - PBL runs with SYSCLK
  - Device samples cfg_rcw_src at the rising edge of poreset_b
    - Determine the source of the RCW
  - PBL fetches the RCW from the source, configure the device
    - It checks the format of RCW
    - It is written to DCFG_RCWSR0-15
  - PLLs start to work and lock according to the ratios specified in RCW
  - Error reports to DCFG_RSTRQPBLSR

Description of PBL Functionality (continued)

- PBL PBI phase
  - PBL switches to platform clock
  - PBL checks RCW[PBI_SRC]
    - If PBI is disabled, then PBL is done
    - If PBI is enabled, proceed to fetch PBI data from the source
  - PBL finishes the PBI, release the core0 to fetch instruction if RCW[BOOT_HO] is 0
    - The boot code location is specified in RCW[BOOT_LOC] for Power Architecture based device
  - Error reports to DCFG_RSTRQPBLSR
  - For both RCW and PBI phase, if there is any error, the boot stops and /RESET_REQ is asserted

CFG_RCW_SRC for Device with eLBC

cfg_rcw_src pin strapping

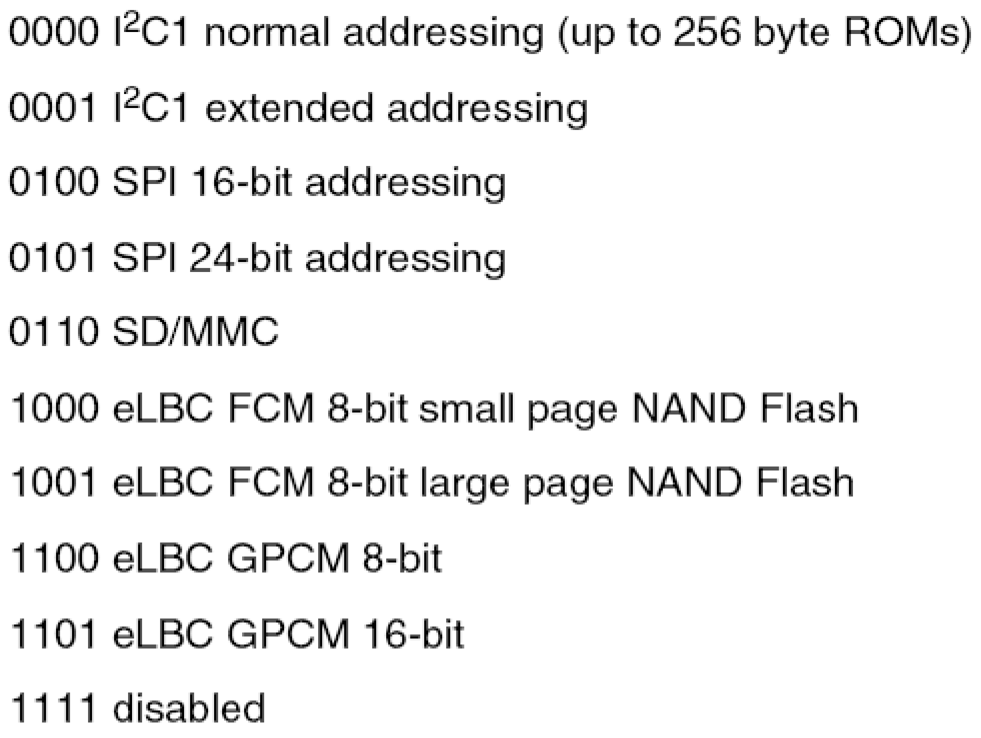


RCW Source Location

Functional Signals	Reset Configuration Name	Value (Binary)	RCW Source
LGPL0/LFCLE, LGPL1/ LFALE, LGPL2/LOE/ LFRE, LGPL3/LFWP, LGPL5 Default (1_1111)	cfg_rcw_src[0:4]	0_0000	I^{2}C1 normal addressing (supports ROMs up to 256 bytes)
		0_0001	I^{2}C1 extended addressing
		0_0010	Reserved
		0_0011	Reserved
		0_0100	SPI 16-bit addressing
		0_0101	SPI 24-bit addressing
		0_0110	eSDHC
		0_0111	Reserved
		0_1000	eLBC FCM (NAND flash, 8-bit small page)
		0_1001	eLBC FCM (NAND flash, 8-bit large page)
		0_1010	Reserved
		0_1011	Reserved
		0_1100	eLBC GPCM (NOR flash, 8-bit)
		0_1101	eLBC GPCM (NOR flash, 16-bit)
		0_1110	Reserved
		0_1111	Reserved
		1_0000 - 1_1011	Hard-coded RCW options (See Hard Coded RCW Options, for more information.)
		1_1100 - 1_1111	Reserved

RCW for Device with eLBC

- 512-bits(64-bytes)
  - Bits related to PBL or booting
  - RCW[192:195] -- PBI_SRC



0000	I^2C1 normal addressing (up to 256 byte ROMs)
0001	I^2C1 extended addressing
0100	SPI 16-bit addressing
0101	SPI 24-bit addressing
0110	SD/MMC
1000	eLBC FCM 8-bit small page NAND Flash
1001	eLBC FCM 8-bit large page NAND Flash
1100	eLBC GPCM 8-bit
1101	eLBC GPCM 16-bit
1111	disabled

RCW for Device with eLBC (continued)

- RCW[196:200] -- BOOT_LOC

0_0000	PCIe1
0_0001	PCIe2
0_0010	PCIe3
0_1000	sRIO1
0_1001	sRIO2
1_0000	Memory complex 1
1_0001	Memory complex 2
1_0100	Interleaved memory complexes
1_1000	eLBC FCM 8-bit small page NAND Flash
1_1001	eLBC FCM 8-bit large page NAND Flash
1_1100	eLBC GPCM 8-tb
1_1101	eLBC GPCM 16-bit

Devices with IFC

- CFG_RCW_SRC[0:8] (based on T4240)

  It needs more bits to specify the IFC NOR/NAND options

  NOR: port size/address shift/AVD

  NAND: port size/page/block/BBI/ECC

  This is due to the difference between IFC and eLBC

- If RCW/PBI is not from IFC, but BOOT_LOC is from IFC, then RCW[IFC_MODE] determines the IFC configuration

Restriction on CFG_RCW_SRC, PBI_SRC, BOOT_LOC

- RCW and pre-boot initialization data must be loaded from the same non-volatile memory device

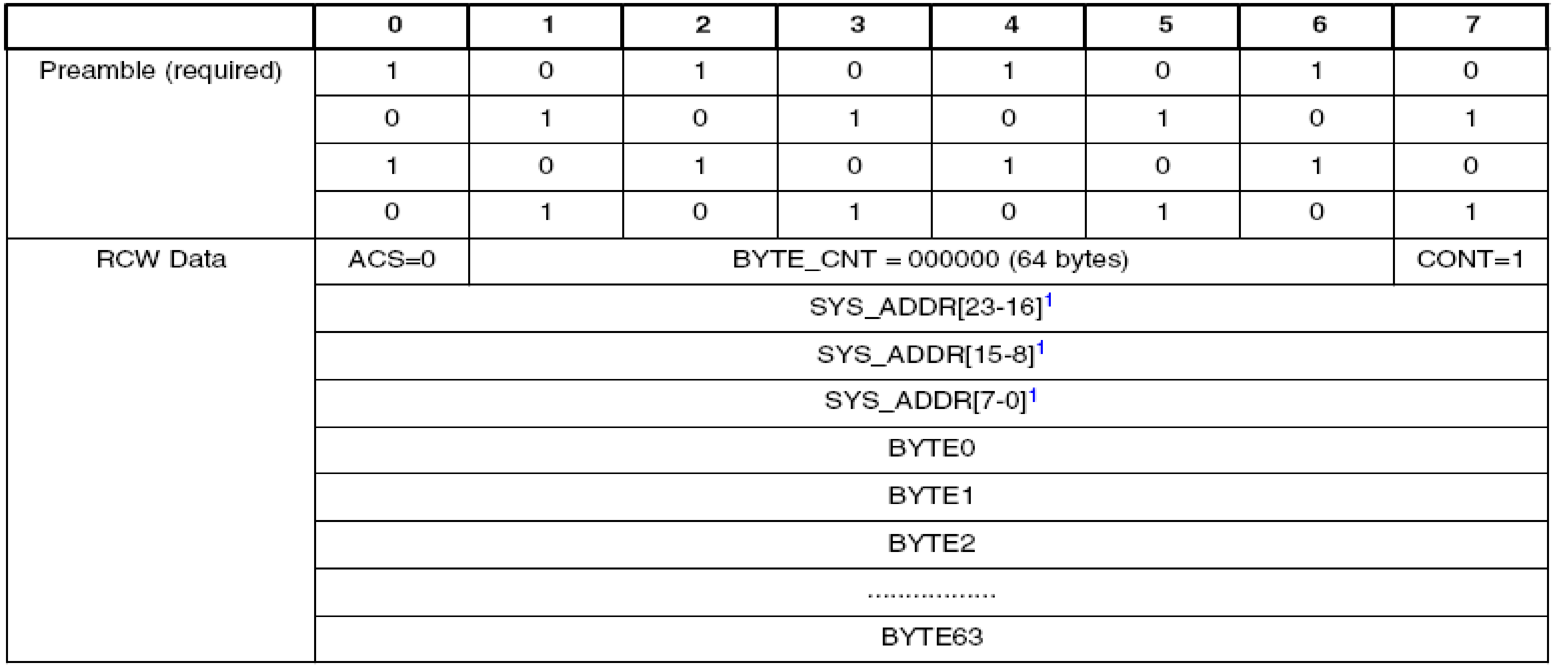
  In the design, PBI_SRC is ignored. PBI is always loaded from CFG_RCW_SRC

- At most, only one given IFC option can be used for CFG_RCW_SRC, PBI_SRC, and BOOT_LOC

CFG_RCW_SRC	PBI_SRC	BOOT_LOC	
I2C	I2C	NAND or NOR	OK
eSPI	eSPI	NAND or NOR	OK
eSDHC	eSDHC	NAND or NOR	OK
NAND	NAND	NOR	No
I2C	eSPI	NAND or NOR	eSPI is ignored No error

PBL Data Format: RCW Format

Required Format of Data Structure Consumed by PBL



	0	1	2	3	4	5	6	7
Preamble (required)	1	0	1	0	1	0	1	0
	0	1	0	1	0	1	0	1
	1	0	1	0	1	0	1	0
	0	1	0	1	0	1	0	1
RCW Data	ACS=0	BYTE_CNT = 000000 (64 bytes)						CONT=1
	SYS_ADDR[23-16][1]							
	SYS_ADDR[15-8][1]							
	SYS_ADDR[7-0][1]							
	BYTE0							
	BYTE1							
	BYTE2							
	....							
	BYTE63							

A
5
A
5

R
C
W

PBL checks the correctness of data structure

- If not detecting preamble 0xA5A5, it reports error code 0x70
- If ACS=1, it reports 0x71 (ACS is logic 1 during RCW)
- If BYTE_CNT!=000000, it reports 0x72(Byte count is not 64 for RCW)
- If SYS_ADDR!=0x0e0100,it reports 0x74(Addr is in protected space)
  - SYS_ADDR for RCW points to DCFG_RCWSR0

For new Layerscape devices, DCFG_RCWSR0 address is different from Power based device

PBL Data Format: PBI Format Address/Data Pair, Write (Exception: PBL Command)

	0	1	2	3	4	5	6	7
First Pre-Boot Initialization Command (optional)	ACS	BYTE_CNT						CONT=1
	SYS_ADDR[23-16]							
	SYS_ADDR[15-8]							
	SYS_ADDR[7-0]							
	BYTE0							
	BYTE1							
	BYTE2							
	....							
	BYTE N-1 (up to 63)							
Second Pre-Boot Initialization Command (optional)	ACS	BYTE_CNT						CONT=1
	SYS_ADDR[23-16]							
	SYS_ADDR[15-8]							
	SYS_ADDR[7-0]							
	BYTE0							
	BYTE1							
	BYTE2							
	....							
	BYTE N-1 (up to 63)							
....								
....								

End of PBL Data Structure

	0	1	2	3	4	5	6	7
End Command (required, special CRC Check command with CONT=0)	ACS=0	BYTE_CNT = 00100 (4 bytes)						CONT=0
	SYS_ADDR[23:16] = 0x13[2]							
	SYS_ADDR[15:8] = 0x80[2]							
	SYS_ADDR[7:0] = 0x40[2]							
	CRC0							
	CRC1							
	CRC2							
	CRC3							

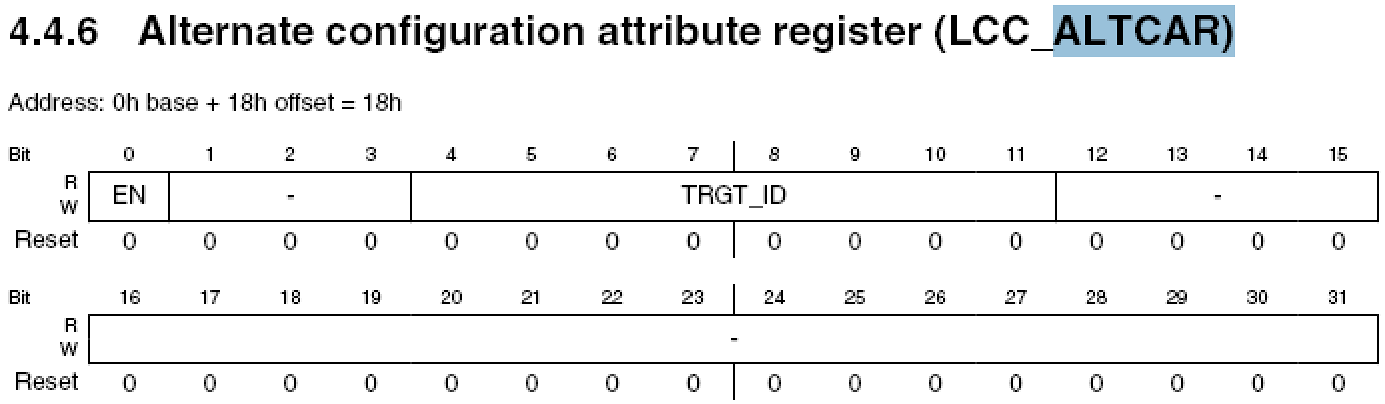
- CONT=0, end command
- PBL reports 0x79(Invalid End command error) if it detects the followings:
  - BYTE_CNT != 4
  - ACS==1
  - For Power Architecture based device, SYS_ADDR!=0x138040

ACS - Alternate Configuration Space

- If ACS=0, it access CCSR address space (16 Mbytes)

- If ACS=1, it access Alternate Configuration Space (16 Mbytes)
  - ALTCBARH, ALTCBARL : Defines base address
  - ALTCAR registers: Target
- By changing the alternate configuration space base address

PBL can access the whole physical address space



4.4.6 Alternate configuration attribute register (LCC_ALTCAR)

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EN	-			TRGT_ID								-			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	-															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CRC Check Command

- For Power Architecture based device, the last 8 bytes must be:
  0x08_138040 + 4-byte CRC
  0x08 means: ACS=0, BYTE_CNT=4, CONT=0
  0x138040 is “CRC check” command

- PBL command
  - SYS_ADDR equals to PBL CCSR address space, PBL treats this command entry as a PBL internal command.

    0x138000 is the PBL address space for Power Architecture based devices

For new Layerscape device, the CCSR address for PBL is different.

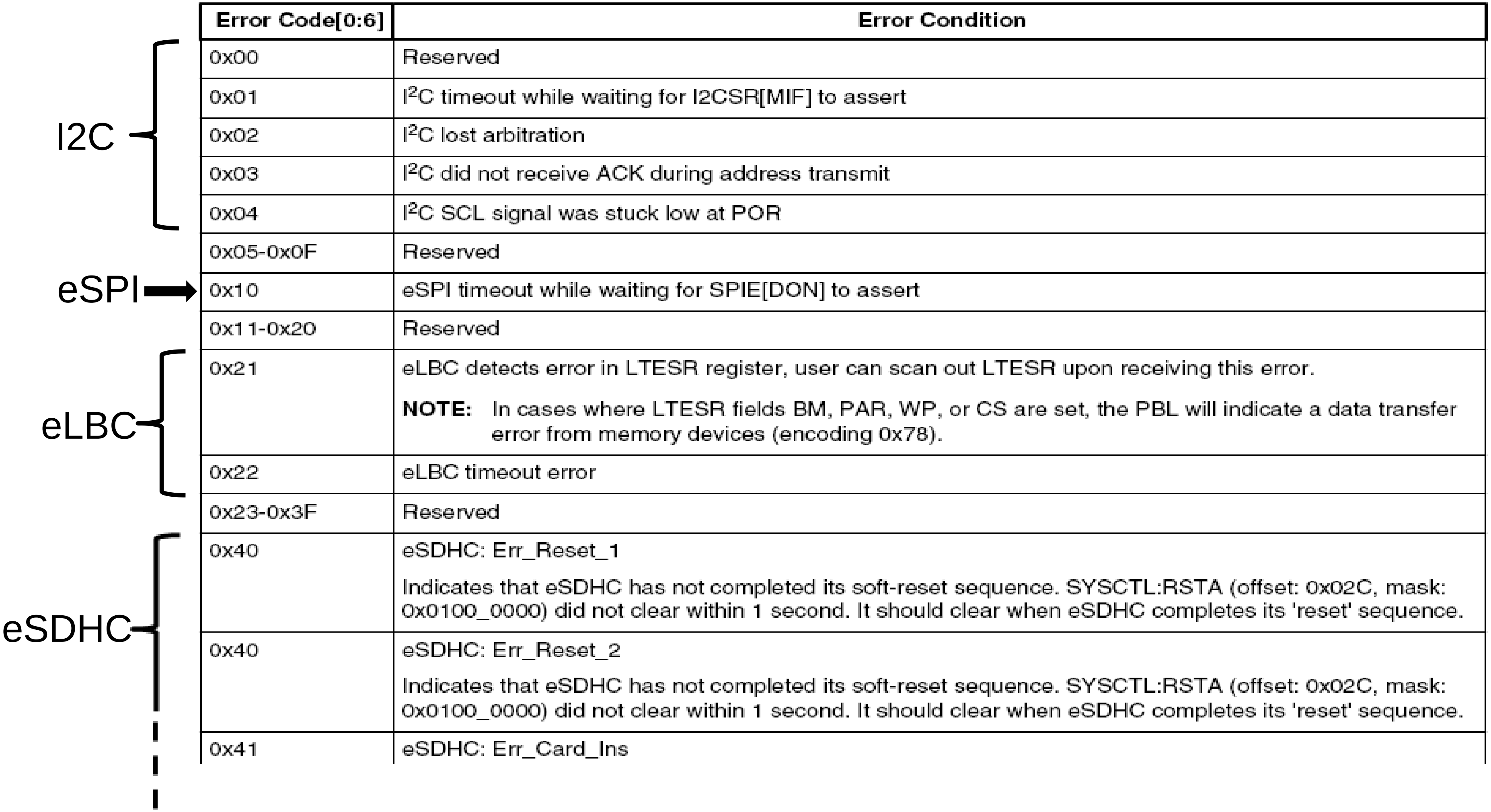
All PBL Commands

Command Name	SYS_ADDR (Power Architecture)	Parameter	Description
Flush	0x13_8000		Chases the previous write with a read from the same address.
CRC check	0x13_8040	Next 4 bytes are CRC value	Checks CRC for data blocks
Jump	0x13_8080	Target data address	PBL reads next data from the jump address
Wait	0x13_80C0	Number of PBL clock(platform clock/2)	Wait a number of cycles before reading next data

PBL Error Code

- PBL checks the correctness of PBL data format. It reports to DCFG_RSTRQPBLSR for any error
  - Preamble/Header errors
  - End command errors, including CRC error
  - Interface errors from I2C, eSPI, eSDHC, eLBC/IFC
- The register can be read out from JTAG to help debug if the device does not boot.

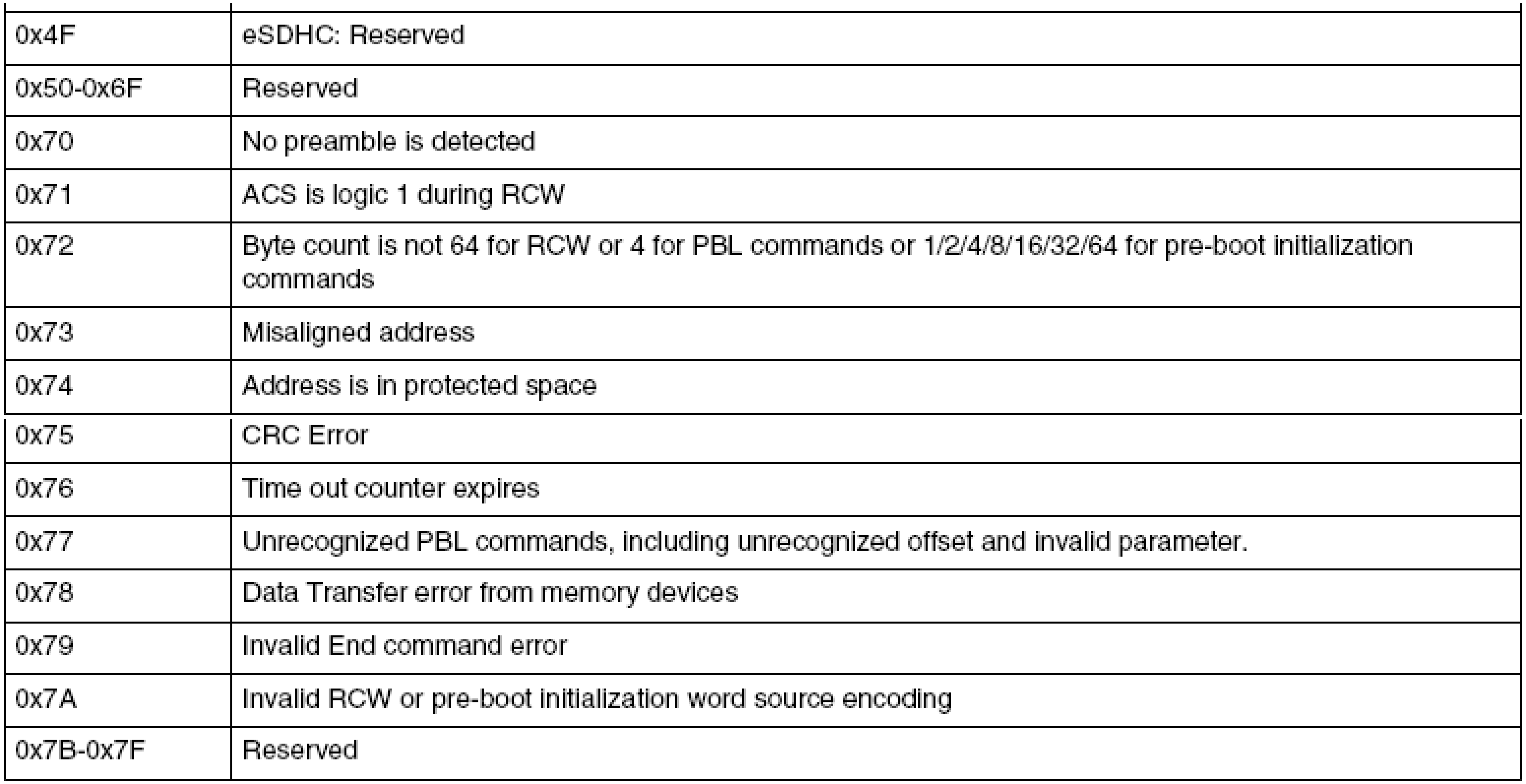
PBL Error Code (continued)


I2C
eSPI
eLBC
eSDHC

Error Code[0:6]	Error Condition
0x00	Reserved
0x01	I^2C timeout while waiting for I2CSR[MIF] to assert
0x02	I^2C lost arbitration
0x03	I^2C did not receive ACK during address transmit
0x04	I^2C SCL signal was stuck low at POR
0x05-0x0F	Reserved
0x10	eSPI timeout while waiting for SPIE[DON] to assert
0x11-0x20	Reserved
0x21	eLBC detects error in LTESR register, user can scan out LTESR upon receiving this error. **NOTE:** In cases where LTESR fields BM, PAR, WP, or CS are set, the PBL will indicate a data transfer error from memory devices (encoding 0x78).
0x22	eLBC timeout error
0x23-0x3F	Reserved
0x40	eSDHC: Err_Reset_1 Indicates that eSDHC has not completed its soft-reset sequence. SYSCTL:RSTA (offset: 0x02C, mask: 0x0100_0000) did not clear within 1 second. It should clear when eSDHC completes its 'reset' sequence.
0x40	eSDHC: Err_Reset_2 Indicates that eSDHC has not completed its soft-reset sequence. SYSCTL:RSTA (offset: 0x02C, mask: 0x0100_0000) did not clear within 1 second. It should clear when eSDHC completes its 'reset' sequence.
0x41	eSDHC: Err_Card_Ins

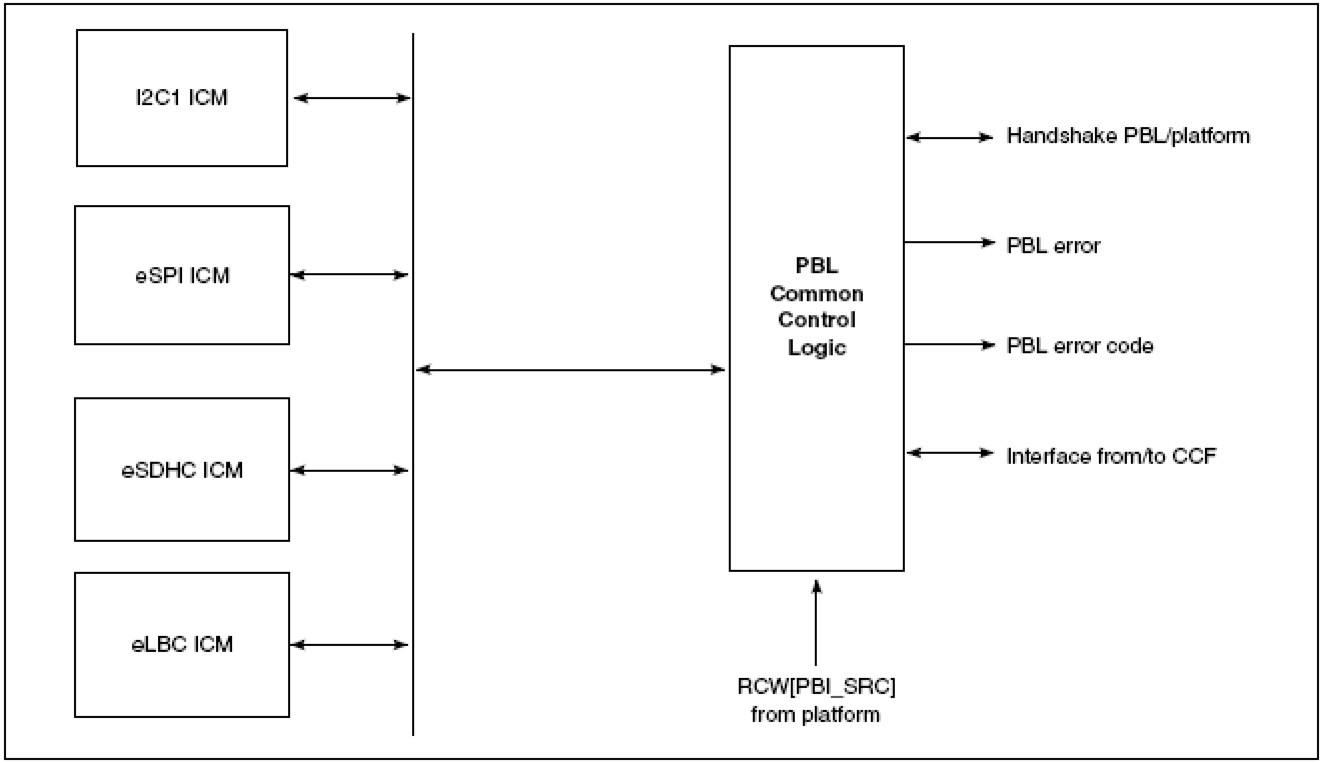
Many, many eSDHC error entries

PBL Error Code (continued)



eSDHC	0x4F	eSDHC: Reserved
	0x50-0x6F	Reserved
End Cmd	0x70	No preamble is detected
	0x71	ACS is logic 1 during RCW
	0x72	Byte count is not 64 for RCW or 4 for PBL commands or 1/2/4/8/16/32/64 for pre-boot initialization commands
	0x73	Misaligned address
	0x74	Address is in protected space
	0x75	CRC Error
	0x76	Time out counter expires
	0x77	Unrecognized PBL commands, including unrecognized offset and invalid parameter.
	0x78	Data Transfer error from memory devices
	0x79	Invalid End command error
	0x7A	Invalid RCW or pre-boot initialization word source encoding
	0x7B-0x7F	Reserved

PBL Block Diagram



Pre-Boot Loader (PBL) Block Diagram

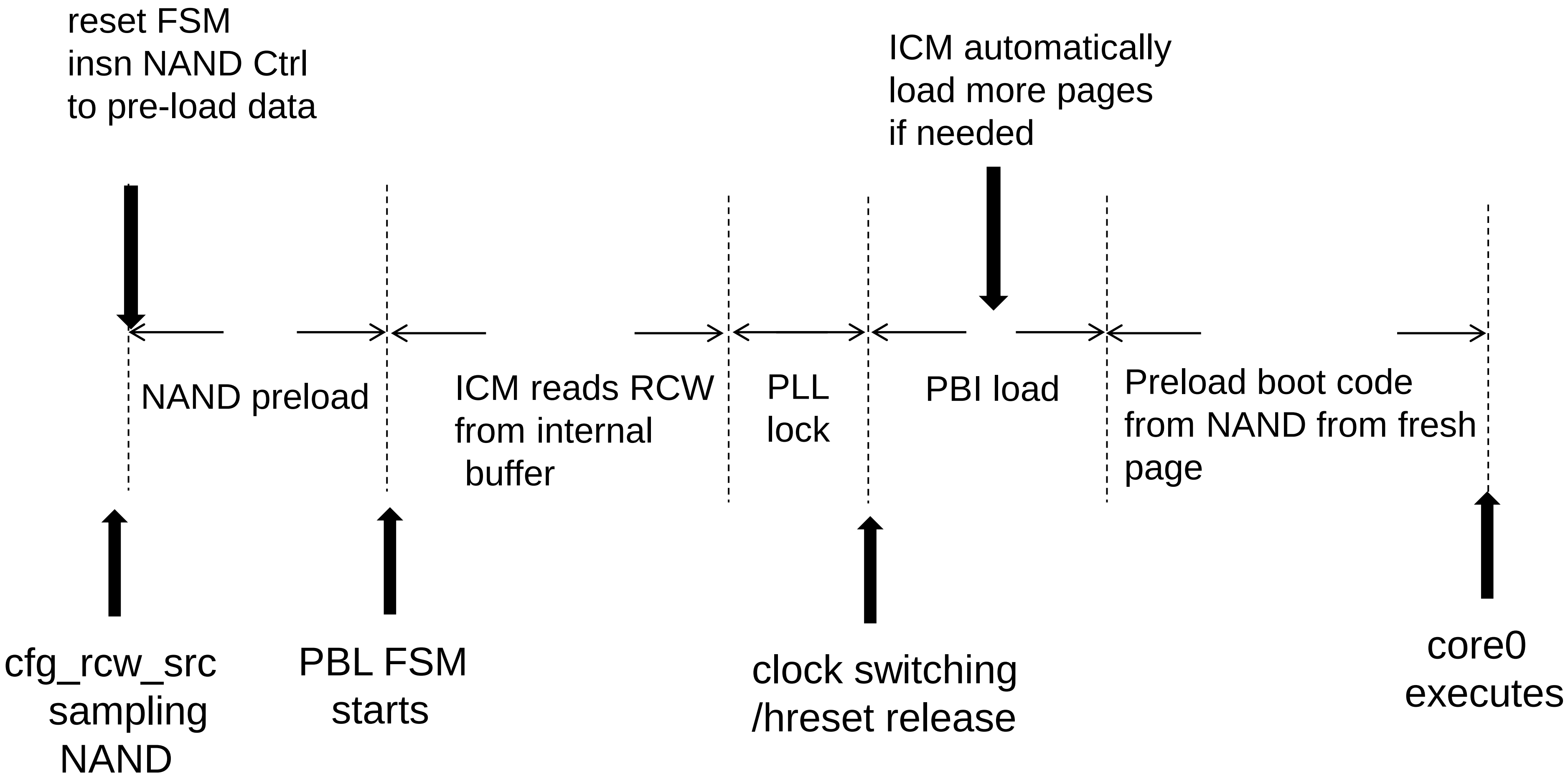
ICM: Interface Control Module

What Does ICM (Interface Control Module) Do?

- Each interface has its own ICM (except eLBC GPCM or IFC NOR)
- It has a state machine to go through the register initialization
- Initiate the read process and polling the status until read data is back
- Passing the data to PBL CCL
- If there is any error, it reports the error back to PBL CCL
- All the register value will be reset back to their POR value when PBL has completed

ICM for NAND

- ICM works with power on reset machine and NAND controller to pre-fetch data from NAND device



reset FSM
insn NAND Ctrl
to pre-load data

ICM automatically
load more pages
if needed

NAND preload

ICM reads RCW
from internal
buffer

PLL
lock

PBI load

Preload boot code
from NAND from fresh
page

cfg_rcw_src
sampling
NAND

PBL FSM
starts

clock switching
/hreset release

core0
executes

I2C

- PBL works with I2C1
- PBL supports I2C devices
  - Normal 16-bit Addressing
  - Extended 24-bit Addressing
- When standard I2C EEPROMs are used, it is assumed that each EEPROM holds 256 bytes of data. After 256 bytes have been received, the PBL will increment the slave address and access the next EEPROM. A maximum of eight standard I2C EEPROMs may be used with the PBL
- Extended I2C EEPROMs have variable sizes. However, the I2C controller increments the slave address after 64 Kbytes have been received

eSPI

- SPI_CS0 is used
- Only SPI Mode 0 is used
  eSPI memory device must also support mode 0
- A value of 0x03 is used for the read command. The SPI slave device must decode 0x03 as a read command

How Is the Speed of I2C, eSPI and eSDHC determined during RCW and PBI?

- Target speed:

  I2C: 100K
  eSPI: 2Mhz
  eSDHC:

  Identification process is 400Khz
  Normal phase is 25Mhz for SD, 20Mhz for MMC

- This is based on design assumption of

  SYSCLK 200MHz for RCW phase, CCB 2GHz for PBI phase
  The real speed is scaled according to the real SYSCLK/CCB clock

  For example, SYSCLK 100Mhz, CCB=800Mhz
  I2C: RCW=100Mhz/200Mhz *100KHz=50KHz
  PBI =800Mhz/2000Mhz *100KHz=40Khz

PBL Application Examples

- ✓ **Errata Workaround**
- ✓ **Boot from eSPI, eSDHC and NAND**

Errata Workaround: Example

A-006559: Configuration is required for proper e500mc operation

Affects: GEN

Description: In order to ensure data integrity between the core and the platform (the core running asynchronously with the CoreNet platform), special internal registers must be set. These internal registers are part of the debug configuration register address space (DCSR)

Impact: Data and instruction corruption is possible if the workaround is not followed

Workaround: Set the internal register bits [16:19] for each core before bringing the cores out of reset. The internal register of Core0 is at offset 0x2_1008; the internal register of Core1 is at offset 0x2_1028; the internal register of Core2 is at offset 0x2_1048; and the internal register of Core3 is at offset 0x2_1068. This configuration may be done using PBI code

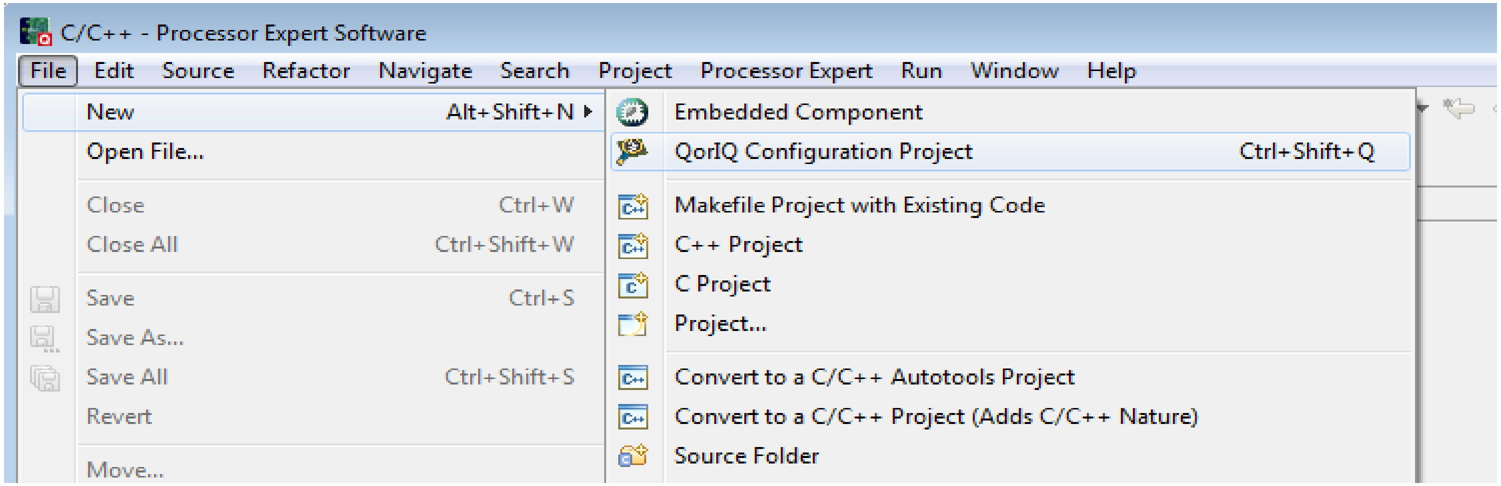
An example for PBI initialization:

1. Write a value of 0x0000_0000 to ALTCBARH register at offset 0x10
2. Write a value of 0x0000_0000 to ALTCBARL register at offset 0x14
3. Write a value of 0x81d0_0000 to ALTCAR register at offset 0x18,
   this step set the target ID to DCSR
4. Write a value of 0x0000_f000 to the internal register at offset 0x2_1008,
   set configuration for core0
5. Write a value of 0x0000_f000 to the internal register at offset 0x2_1028,
   set configuration for core1
6. Write a value of 0x0000_f000 to the internal register at offset 0x2_1048,
   set configuration for core2
7. Write a value of 0x0000_f000 to the internal register at offset 0x2_1068,
   set configuration for core3

Fix plan: No plans to fix

QCS Tool (V3.0.4) – Steps to Implement Example 2

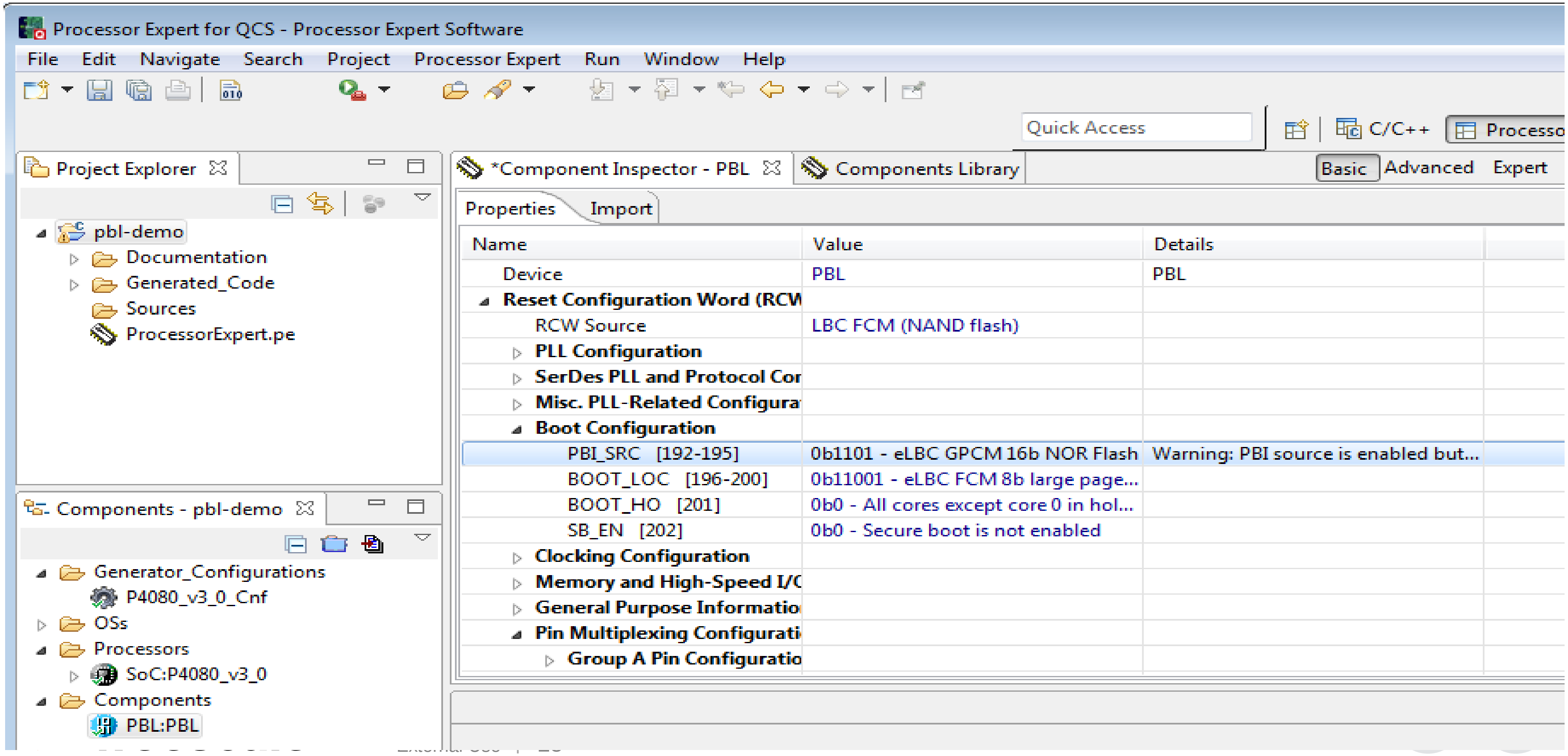
- File -> New -> QorIQ Configuration Project
- Fill the “project name”, click Next
- Choose Processor and Silicon Revision, click Next
- Select “PBL – Preboot Loader RCW configuration”
  Click Next
- Choose “Import configuration from an existing PBL file” *(if starting from scratch, choose “Use RCW Hard-coded configuration” as update field as needed”*
  Use “Browse” to select the file
  Choose “File Format”, click Finish.



QCS Tool – Steps to Implement Example 2 (continued)

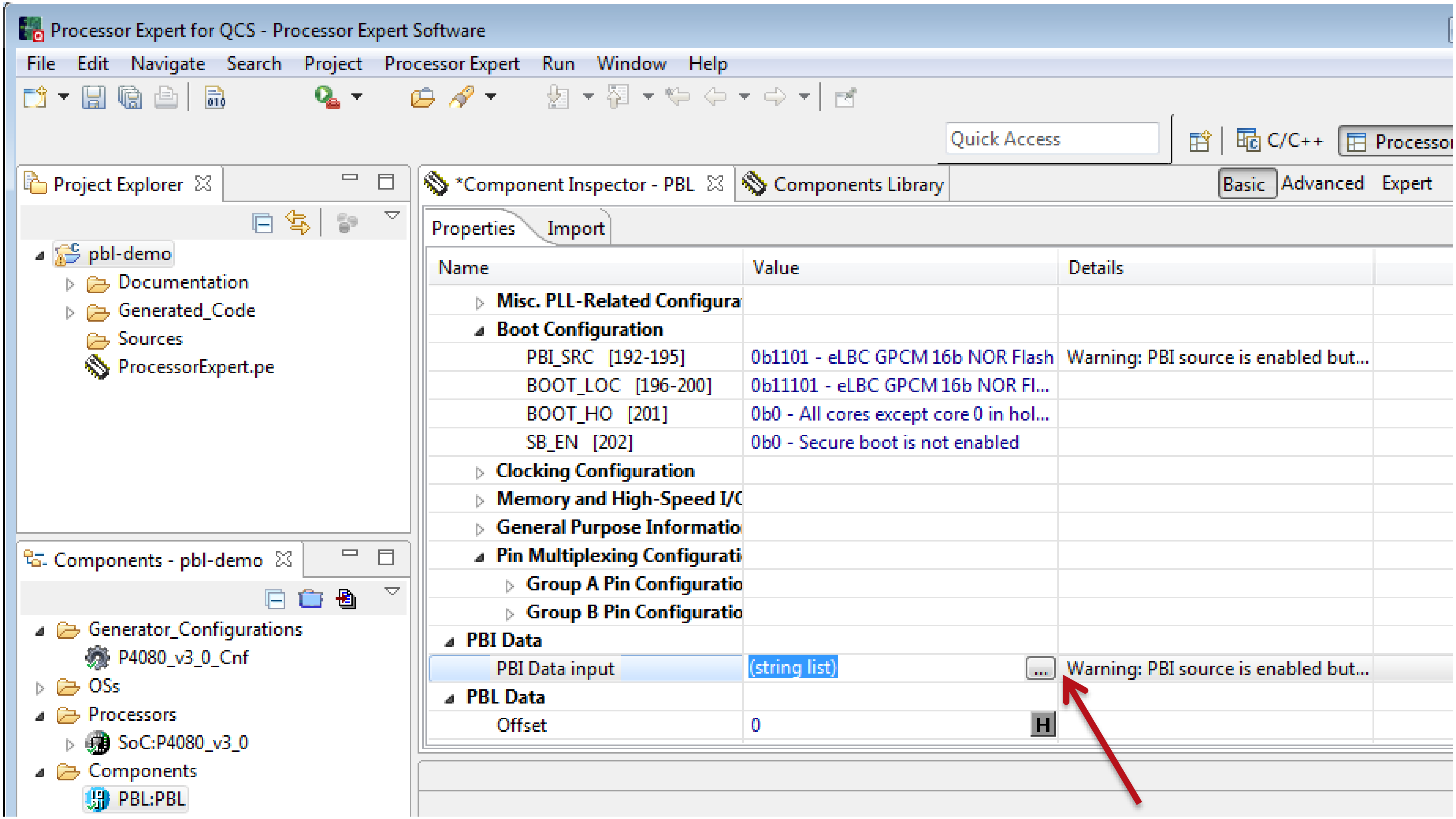
- Project Explorer->*project name* (pbl-demo)
- Components -> PBL:PBL
- On the right side, Choose the appropriate items to modify

(1) Boot Configuration-> PBI_SRC: 0b1101 – eLBC GPCM 16b NOR flash



Name	Value	Details
Device	PBL	PBL
Reset Configuration Word (RCW		
RCW Source	LBC FCM (NAND flash)	
PLL Configuration		
SerDes PLL and Protocol Cor		
Misc. PLL-Related Configura		
Boot Configuration		
PBI_SRC [192-195]	0b1101 - eLBC GPCM 16b NOR Flash	Warning: PBI source is enabled but...
BOOT_LOC [196-200]	0b11001 - eLBC FCM 8b large page...	
BOOT_HO [201]	0b0 - All cores except core 0 in hol...	
SB_EN [202]	0b0 - Secure boot is not enabled	
Clocking Configuration		
Memory and High-Speed I/C		
General Purpose Informatio		
Pin Multiplexing Configurati		
Group A Pin Configuratio		

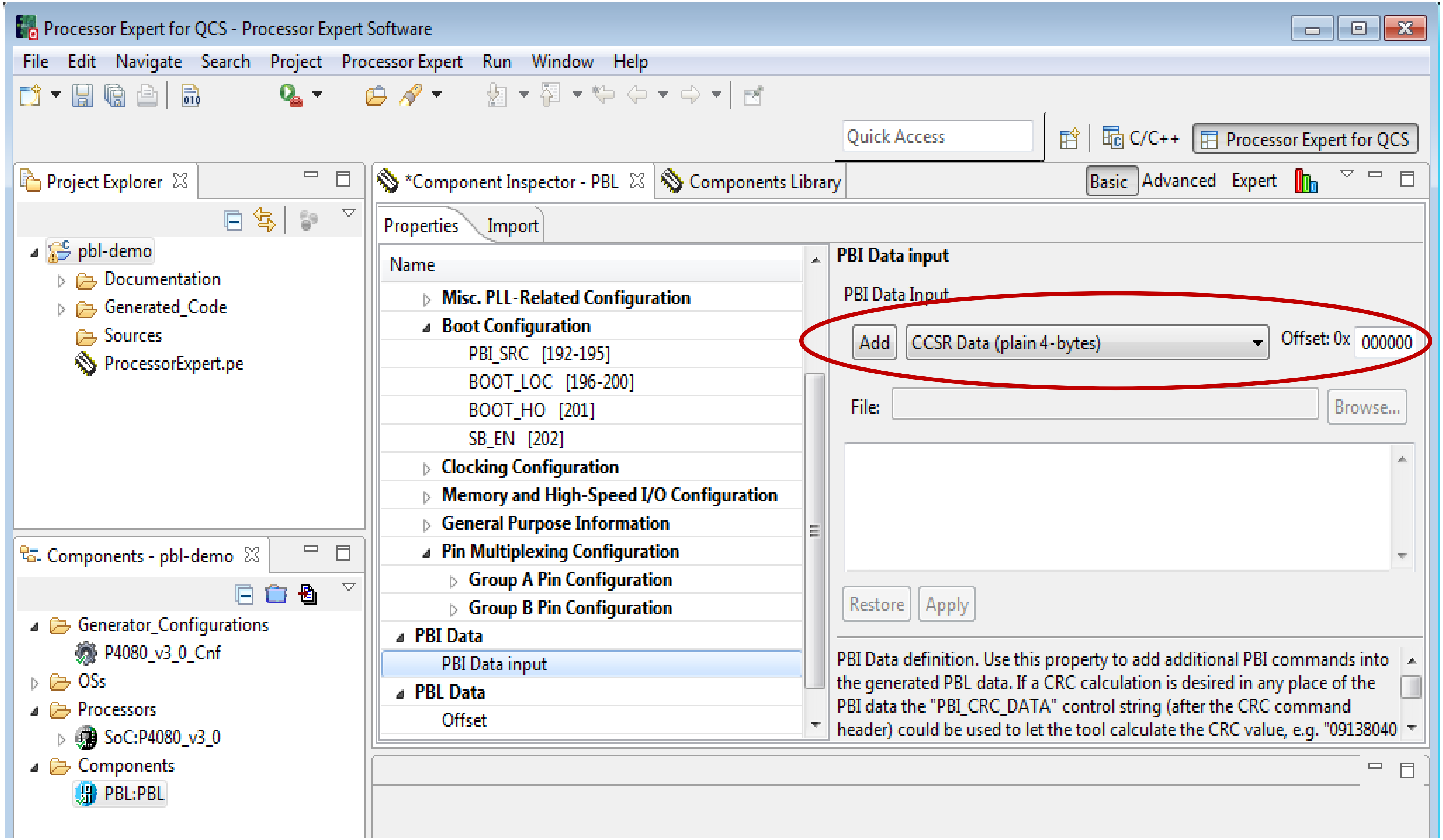
QCS Tool – Steps to Implement Example 2 (continued)



Name	Value	Details
Misc. PLL-Related Configura		
Boot Configuration		
PBI_SRC [192-195]	0b1101 - eLBC GPCM 16b NOR Flash	Warning: PBI source is enabled but...
BOOT_LOC [196-200]	0b11101 - eLBC GPCM 16b NOR Fl...	
BOOT_HO [201]	0b0 - All cores except core 0 in hol...	
SB_EN [202]	0b0 - Secure boot is not enabled	
Clocking Configuration		
Memory and High-Speed I/(		
General Purpose Informatio		
Pin Multiplexing Configurati		
Group A Pin Configuratio		
Group B Pin Configuratio		
PBI Data		
PBI Data input	(string list) ...	Warning: PBI source is enabled but...
PBL Data		
Offset	0 H	

Click here

QCS Tool – Steps to Implement Example 2 (continued)



Processor Expert for QCS - Processor Expert Software

Project Explorer:
- pbl-demo
  - Documentation
  - Generated_Code
  - Sources
  - ProcessorExpert.pe

Component Inspector - PBL:

Name
- Misc. PLL-Related Configuration
- Boot Configuration
  - PBI_SRC [192-195]
  - BOOT_LOC [196-200]
  - BOOT_HO [201]
  - SB_EN [202]
- Clocking Configuration
- Memory and High-Speed I/O Configuration
- General Purpose Information
- Pin Multiplexing Configuration
  - Group A Pin Configuration
  - Group B Pin Configuration
- PBI Data
  - PBI Data input
- PBL Data
  - Offset

PBI Data input

PBI Data Input

Add | CCSR Data (plain 4-bytes) | Offset: 0x 000000

File: | Browse...

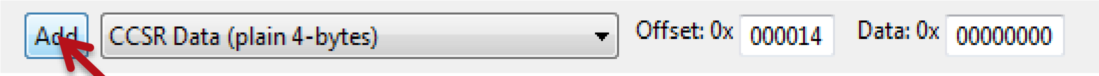
Restore | Apply

PBI Data definition. Use this property to add additional PBI commands into the generated PBL data. If a CRC calculation is desired in any place of the PBI data the "PBI_CRC_DATA" control string (after the CRC command header) could be used to let the tool calculate the CRC value, e.g. "09138040

Components - pbl-demo:
- Generator_Configurations
  - P4080_v3_0_Cnf
- OSs
- Processors
  - SoC:P4080_v3_0
- Components
  - PBL:PBL

Add PBI Data

(1) Offset 0x000014, Data 0x00000000, click add



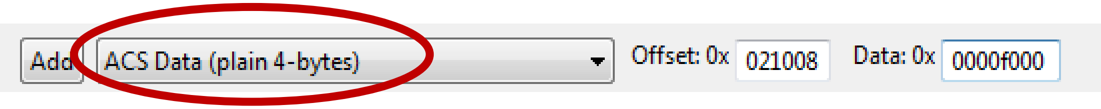
Add | CCSR Data (plain 4-bytes) | Offset: 0x 000014 | Data: 0x 00000000

Click

(2) offset 0x000018, Data 0x81d00000, click add

(3) Choose ACS Data(4 Byte)

offset 0x021008, Data 0x0000f000



Add | ACS Data (plain 4-bytes) | Offset: 0x 021008 | Data: 0x 0000f000

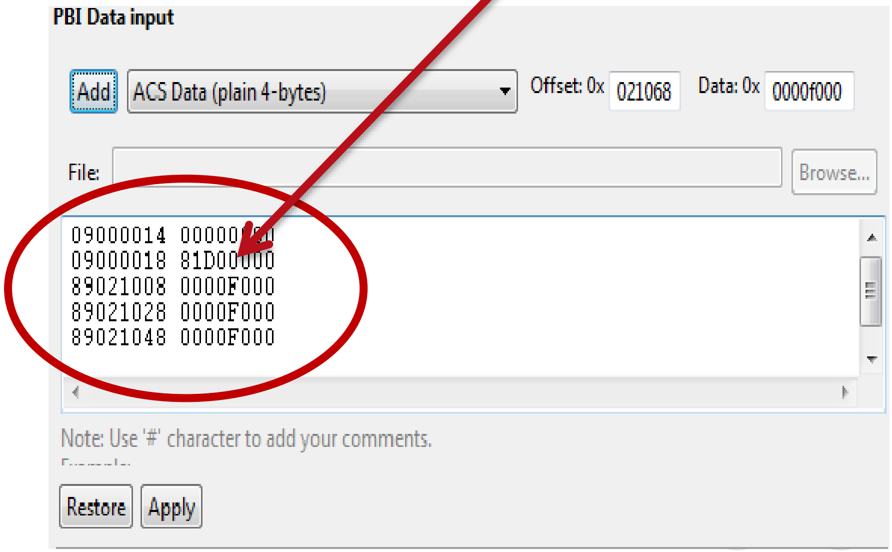
(4) Repeat (3) with offset 0x021028

(5) Repeat (3) with offset 0x021048

(6) Repeat (3) with offset 0x021068

(7) Click Apply to save the data

PBI data



PBI Data input

Add | ACS Data (plain 4-bytes) | Offset: 0x 021068 | Data: 0x 0000f000

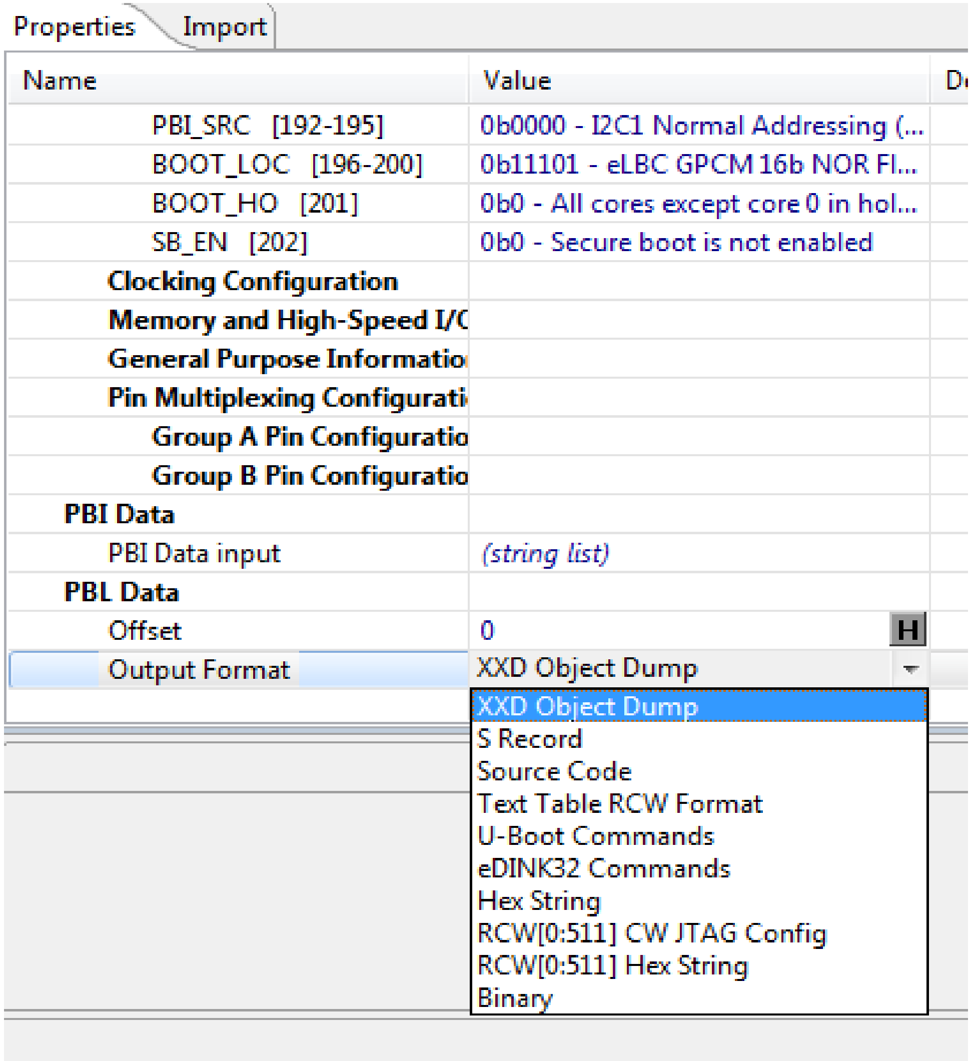
File: | Browse...

```
09000014 00000000
09000018 81D00000
89021008 0000F000
89021028 0000F000
89021048 0000F000
```

Note: Use '#' character to add your comments.

Restore | Apply

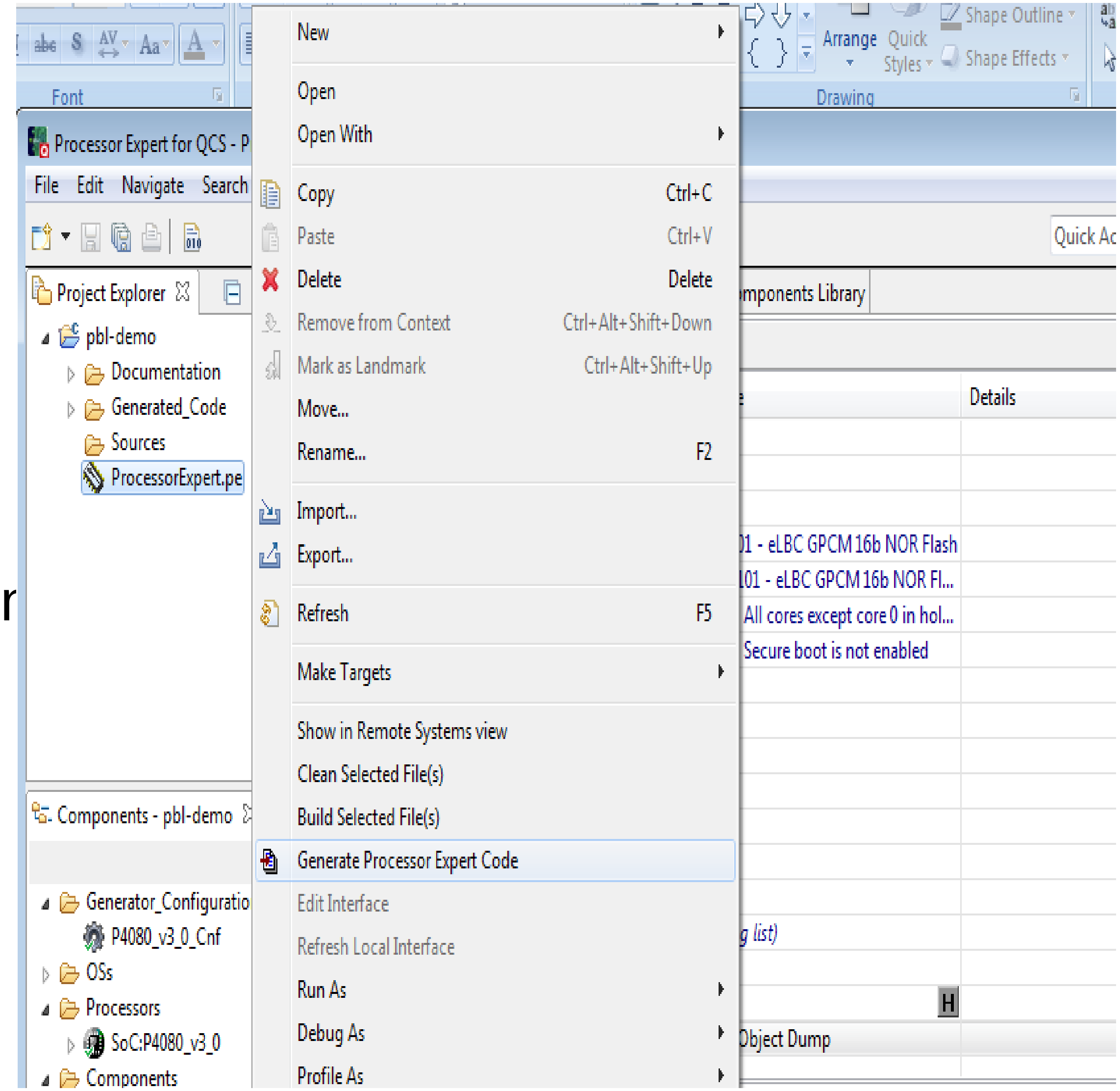
Choose the Output Format



Name	Value	D
PBI_SRC [192-195]	0b0000 - I2C1 Normal Addressing (...	
BOOT_LOC [196-200]	0b11101 - eLBC GPCM 16b NOR Fl...	
BOOT_HO [201]	0b0 - All cores except core 0 in hol...	
SB_EN [202]	0b0 - Secure boot is not enabled	
Clocking Configuration		
Memory and High-Speed I/C		
General Purpose Informatio		
Pin Multiplexing Configurati		
Group A Pin Configuratio		
Group B Pin Configuratio		
PBI Data		
PBI Data input	*(string list)*	
PBL Data		
Offset	0	
Output Format	XXD Object Dump	

Generate Output

1. Choose ProcessExpert.pe
2. Right click, choose “Generate Processor Expert Code”
3. Output workspace/project/Ger
   rated_code/PBL.pbl



PBL Application Boot from eSPI, eSDHC and NAND

- PBL-based devices
  - BOOT_LOC: no options for boot from eSPI or eSDHC
- Solution
  - Use PBL to write the image. The whole image is written with PBI

Steps to Produce PBL Image for U-boot for eSPI

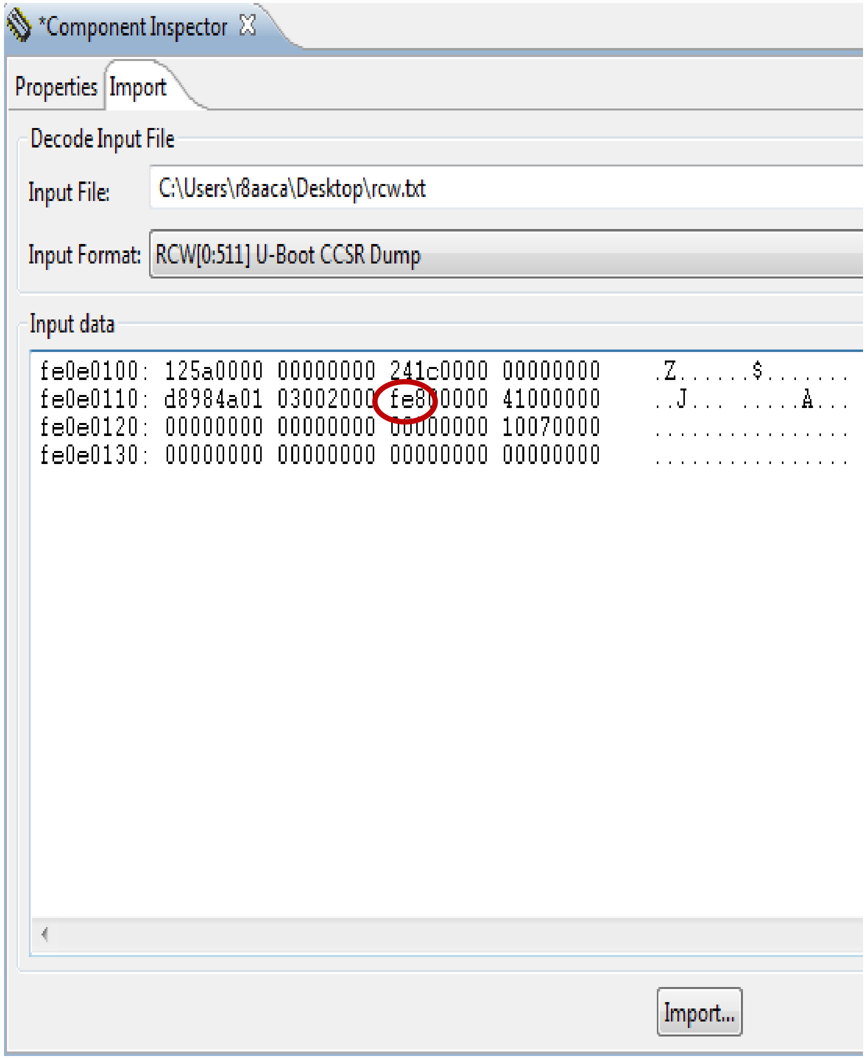
1. After creating a QorIQ Configuration Project, the project appears in the Project Panel view
2. Find “Import" tab in window "Component Inspector", select "Input Format" value to be "RCW[0:511] U-Boot CCSR Dump“
   - Copy/paste RCW data from u-boot to Input data

   or

   - Use Browse if RCW is saved in a file
3. Change "fe8" in the second row to "580“

   This enables PBI from eSPI and BOOT_LOC is

   Memory Complex 1
4. Click “Import”



Input File: C:\Users\r8aaca\Desktop\rcw.txt

Input Format: RCW[0:511] U-Boot CCSR Dump

```
fe0e0100: 125a0000 00000000 241c0000 00000000   .Z......$.......
fe0e0110: d8984a01 03002000 fe800000 41000000   ..J... .....A...
fe0e0120: 00000000 00000000 00000000 10070000   ................
fe0e0130: 00000000 00000000 00000000 00000000   ................
```

Steps to Produce PBL Image for U-boot for eSPI (continued)

Linux

1. make ARCH=powerpc CROSS_COMPILE=powerpc-linux-gnu- P3041DS_SPIFLASH
2. xxd u-boot.bin > u-boot.xxd

QCS Tool

1. Click button in "Value" column of row "PBI Data input"
   "PBI Data input" view will appears
2. Paste commands in the right into text field
   Configure CPC as 1M SRAM, u-boot image will be written by PBI to CPC
3. Add u-boot image as PBI command
   - Select "ACS File (XXD Object Dump)“
   - Change Offset to "f80000“
   - Click “Browse" to select the file "u-boot.xxd" produced above
   - Click "Add",
     Content of the "u-boot.xxd" will be pasted
4. Paste "09138000 00000000" and "091380c0 00000000" at the end.
   flush
   wait
5. Click "Apply".

PBI Data

```
09010000 00200400
09138000 00000000
091380c0 00000100
09010100 00000000
09010104 fff0000b
09010f00 08000000
09010000 80000000
09000d00 00000000
09000d04 fff00000
09000d08 81000013
09000010 00000000
09000014 ff000000
09000018 81000000
09110000 80000403
09110020 2d170008
09110024 00100008
09110028 00100008
0911002c 00100008
09138000 00000000
091380c0 00000000
```

Steps to Produce PBL Image for U-boot for eSPI (continued)

- Find and click "Generate Processor Expert Code" in menu "Project", after it finished, click "Generated_Code" in "Project Panel" window and "PBL1.pbl" appears, find the file "PBL1.pbl" in workspace of this project

- Linux

```
xxd -r PBL1.pbl > u-boot.pbl
```

- Burn u-boot.pbl to eSPI to boot

```
=> loadb 1000000 (or tftp 100000 u-boot.pbl )
=> sf probe 0
=> sf erase 0 100000
=>sf write 1000000 0 $filesize
```

Introducing The QorIQ LS2 Family

Breakthrough, software-defined approach to advance the world's new virtualized networks

New, high-performance architecture built with ease-of-use in mind

Groundbreaking, flexible architecture that abstracts hardware complexity and enables customers to focus their resources on innovation at the application level

Optimized for software-defined networking applications

Balanced integration of CPU performance with network I/O and C-programmable datapath acceleration that is right-sized (power/performance/cost) to deliver advanced SoC technology for the SDN era

Extending the industry's broadest portfolio of 64-bit multicore SoCs

Built on the ARM® Cortex®-A57 architecture with integrated L2 switch enabling interconnect and peripherals to provide a complete system-on-chip solution

QorIQ LS2 Family

Key Features

SDN/NFV Switching

Data Center

Wireless Access

Unprecedented performance and ease of use for smarter, more capable networks

High performance cores with leading interconnect and memory bandwidth

- 8x ARM Cortex-A57 cores, 2.0GHz, 4MB L2 cache, w Neon SIMD
- 1MB L3 platform cache w/ECC
- 2x 64b DDR4 up to 2.4GT/s

A high performance datapath designed with software developers in mind

- New datapath hardware and abstracted acceleration that is called via standard Linux objects
- 40 Gbps Packet processing performance with 20Gbps acceleration (crypto, Pattern Match/RegEx, Data Compression)
- Management complex provides all init/setup/teardown tasks

Leading network I/O integration

- 8x1/10GbE + 8x1G, MACSec on up to 4x 1/10GbE
- Integrated L2 switching capability for cost savings
- 4 PCIe Gen3 controllers, 1 with SR-IOV support
- 2 x SATA 3.0, 2 x USB 3.0 with PHY

See the LS2 Family First in the Tech Lab!

4 new demos built on QorIQ LS2 processors:

- Performance Analysis Made Easy
- Leave the Packet Processing To Us
- Combining Ease of Use with Performance
- Tools for Every Step of Your Design

freescale™

www.Freescale.com