### Contents

<table>
<thead>
<tr>
<th>Section number</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Chapter 1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2.1</td>
<td>C293 PCIe card switch settings</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>Verify device detection</td>
<td>10</td>
</tr>
<tr>
<td>2.2.1</td>
<td>On X86 host</td>
<td>10</td>
</tr>
<tr>
<td>2.2.2</td>
<td>On P4080 host</td>
<td>11</td>
</tr>
<tr>
<td>2.3</td>
<td>X86 IOAT DMA</td>
<td>12</td>
</tr>
<tr>
<td>2.3.1</td>
<td>BIOS changes</td>
<td>12</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Kernel config changes</td>
<td>12</td>
</tr>
<tr>
<td>2.3.3</td>
<td>Verify IOAT DMA support</td>
<td>13</td>
</tr>
<tr>
<td>2.4</td>
<td>X86 disabling IOMMU hardware support</td>
<td>14</td>
</tr>
<tr>
<td>2.5</td>
<td>P4080 DMA support</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Chapter 3</td>
<td>Build procedure</td>
</tr>
<tr>
<td>3.1</td>
<td>Prerequisites</td>
<td>17</td>
</tr>
<tr>
<td>3.2</td>
<td>X86 host</td>
<td>17</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Compiling PowerPC toolchain</td>
<td>18</td>
</tr>
<tr>
<td>3.2.2</td>
<td>X86 PKC extended kernel compilation</td>
<td>18</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Boot X86 machine with new kernel</td>
<td>19</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Build All software components</td>
<td>19</td>
</tr>
<tr>
<td>3.2.4.1</td>
<td>Build firmware</td>
<td>19</td>
</tr>
<tr>
<td>3.2.4.2</td>
<td>Build driver</td>
<td>20</td>
</tr>
<tr>
<td>3.2.4.3</td>
<td>Build CLI app</td>
<td>20</td>
</tr>
<tr>
<td>3.2.5</td>
<td>Binaries location</td>
<td>20</td>
</tr>
<tr>
<td>3.2.6</td>
<td>Default install location</td>
<td>20</td>
</tr>
<tr>
<td>Section number</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.3</td>
<td>P4080 host</td>
<td>21</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Build P4080 linux kernel</td>
<td>21</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Build All software components</td>
<td>21</td>
</tr>
<tr>
<td>3.3.2.1</td>
<td>Build firmware</td>
<td>21</td>
</tr>
<tr>
<td>3.3.2.2</td>
<td>Build driver</td>
<td>22</td>
</tr>
<tr>
<td>3.3.2.3</td>
<td>Build CLI app</td>
<td>22</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Location of the binaries</td>
<td>22</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Build rootfs</td>
<td>22</td>
</tr>
<tr>
<td>3.3.5</td>
<td>Booting the P4080DS with the new kernel &amp; rootfs image</td>
<td>23</td>
</tr>
</tbody>
</table>

Chapter 4
C29x firmware build configuration

Chapter 5
Resource configuration

Chapter 6
Load driver

6.1 For X86 host

6.2 For P4080 host

Chapter 7
Driver load time configurations

Chapter 8
Pre-built binaries

Chapter 9
Running the driver with openssl

9.1 Cryptodev Framework (CDF)

9.1.1 Procedure to compile and insert CryptoDev Framework

9.2 Openssl

9.2.1 Procedure to compile openssl

Chapter 10
Driver test framework

10.1 Script path

10.2 Supported tests

C29x Crypto Offload User Guide, Rev. 0, 10/2013

Freescale Semiconductor, Inc.
<table>
<thead>
<tr>
<th>Section number</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.3</td>
<td>Usage</td>
<td>38</td>
</tr>
</tbody>
</table>

**Chapter 11**  
CLI

**Chapter 12**  
Directory structure
Chapter 1
Introduction

This document explains the procedure to build, configure, and use different software components for the Freescale C29x crypto coprocessor device.
Chapter 2
Hardware setup

This section explains the hardware setup and configuration used.

2.1 C293 PCIe card switch settings

This section explains the details applicable to the Freescale C293 PCIe development platform. These switch settings may not be valid for other version of cards. (For C291/C292, refer to the corresponding schematic or hardware spec).

- It is assumed that the card has been delivered with proper default DIP switch settings.
- SW7 (Switch 7) settings need to be changed as per the instructions below for the software to work properly.
- Refer to the C29x PCIe Card Quick Start Guide for the details about the switch configurations.
- SW7[1-5] are set towards ON label of the switch for PCIx4 setting and agent mode.
- SW7[1] must be ON for the driver to work. This puts the card in PKCAL mode. (Core is in boot hold off mode.)

The image below shows the SW7 setting on the board.
2.2 Verify device detection

2.2.1 On X86 host

- Run the `lspci` command on the host. A typical output shows the following device:
  
  ```
  05:00.0 Power PC: Freescale Semiconductor Inc Device 0800 (rev 10)
  ```

  **NOTE**
  The BDF 05:00.0 may change based on the root port under which this device is connected in your setup.

- Run `lspci -vvs <b:d:f>` to check and validate the PCIe details of the C2xx card. Following is an example output. Two BARs of 1M size must be detected, as shown in the highlighted text below:
  
  ```
  05:00.0 Power PC: Freescale Semiconductor Inc Device 0800 (rev 10) (prog-if 01)
  ```


  Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-

  Latency: 0, Cache Line Size: 64 bytes

  Interrupt: pin A routed to IRQ 90

  Region 0: Memory at bla00000 (32-bit, non-prefetchable) [size=1M]
2.2.2 On P4080 host

On P4080, device detection may be verified from u-boot logs.

The figure below shows the u-boot log and the highlighted portion indicates the device detection.
2.3 X86 IOAT DMA

This section explains the procedure to enable X86 IOAT DMA support.

2.3.1 BIOS changes

Browse through tabs in BIOS to enable IOAT DMA support.

NOTE

Different BIOS has different setting for IOAT DMA, adjust the setting configuration according to your machine. The following setting is for ASUS Z9PE-D8 WS motherboard and Intel Xeon CPU E5-2620.

Advanced->Chipset Configuration,
CPU IIO Bridge Configuration,
Intel(R) I/OAT [Enabled]
2.3.2 Kernel config changes

Following options need to be changed in kernel configuration menu, browse through the configuration menu as shown below.

Device Drivers -->
DMA Engine support -->
--- DMA Engine support
[ ] DMA Engine debugging
*** DMA Devices ***
< M> Intel MID DMA support for Peripheral DMA controllers
< M> Intel I/OAT DMA support
{ M} Timberdale FPGA DMA support
< M> Intel EG20T PCH / OKI Semi IOH (ML7213/ML7223/ML7831) DMA support
*** DMA Clients ***
[ ] Network: TCP receive copy offload
[ ] Async_tx: Offload support for the async_tx api
< > DMA Test client

The image below shows the config screen menu with the above mentioned changes.

![Figure 2-3. Config screen menu](image)

NOTE

These changes can be made when the updated kernel loads, it automatically shows this screen.

2.3.3 Verify IOAT DMA support

This section explains how to verify if the IOAT (I/O Acceleration Technology) DMA is enabled and supported.

1. `lsmod | grep ioat`, shows the following output:
X86 disabling IOMMU hardware support

root@car:~# lsmod | grep ioat
ioatdma 42148  24
 dca 14611  2 ioatdma,igb

2. cat /proc/interrupts

This displays some ioat MSIx/MSI interrupts, count and irq number. This may vary for different hosts.

83:          0          3   PCI-MSI-edge      ioat-msix
84:          3          0   PCI-MSI-edge      ioat-msix

3. lspci command on Linux also lists the IOATDMA (Intel Quickdata) PCI devices.

NOTE

It supports Xeon 5K series chipsets and above, check your bios to ensure whether or not you have IOAT.

2.4 X86 disabling IOMMU hardware support

To support virtualization, IOMMU Hardware Support needs to be disabled in Linux config. The image below shows the Linux config menu.
2.5 P4080 DMA support

To enable P4080DMA support, changes need to be done in kernel configuration menu, which is displayed during the kernel compilation process.

Browse through the config screen and make following changes:

[*] Device Drivers    --->
[*] DMA Engine support    --->
[ ] DMA Engine debugging
Procedure to compile kernel is explained in below sections.
Chapter 3
Build procedure

3.1 Prerequisites

- Contact your local Freescale Office for the C293 PCIe software.
- Extract the .tar file into a working folder and make sure that non-root user has R-W permission on the directory and all the files inside it.
- Download the QorIQ Linux SDK V1.4. Yocto Source ISO from www.freescale.com/sdk.
- SDK & driver compilation is verified on Ubuntu 12.04. Install the same version to avoid any package dependency related errors.
- wget & update-grub utilities should be present in the distribution. This is required for automated kernel requirement support inside the build.
- Machine on which the build is being issued should be connected to the Internet.
  - For X86, kernel version 3.8.13 is downloaded from kernel.org, PKC patch is applied and then the kernel is compiled. These steps are automated with one make command in the build.
  - For P4080, SDK compilation always downloads the required packages over the Internet.
- For Ubuntu and Debian hosts, reconfigure /bin/sh to point to bash.
  - To see if you are running dash, run:
    
    ```
    ls -al /bin/sh
    ```
  - To change your environment, run:
    
    ```
    sudo rm /bin/sh
    ```
    
    ```
    sudo ln -s /bin/bash /bin/sh
    ```
  - Now if you run, ls -al /bin/sh it points to bash.
3.2 X86 host

This section explains the build procedure for X86 machine.

3.2.1 Compiling PowerPC toolchain

Download SDK 1.4, install the ISO and build the toolchain.

1. After downloading, mount the SDK.
2. Run Install.
3. Agree to the EULA and give installation directory path.
4. From the SDK working directory, prepare the SDK with following commands:
   a. $ . /scripts/host-prepare.sh
   b. $ source ./fsl-setup-poky -m p4080ds
   c. $ bitbake fsl-image-minimal

   **NOTE**
   - SDK 1.4 should be installed on a Linux machine. Follow the SDK documentation to install SDK 1.4.
   - Make sure that this installed SDK path is specified in config.mk file.

3.2.2 X86 PKC extended kernel compilation

Standard kernel does not have PKC extensions, so a version of kernel needs to be downloaded, patched, and compiled.

This process is automated with the following single `make` command:

```
$ make x86_kernel
```

Following are some important points to note:

- Makefile downloads Linux kernel version 3.8.13 from kernel.org, patch it, and then compiles it. Make sure that `linux 3.8.13` kernel does not already exist in the machine.
- Kernel source code is downloaded using the command `wget`. Ensure that `wget` is installed in the build host system.
- `update-grub` utility is used to update the `grub.cfg` file with the new kernel. Make sure that `update-grub` utility is installed in the build host system.
- As part of kernel compilation process, a menu config screen is displayed. IOAT DMA settings can be changed if the host has that capability. IOAT config related details are explained in Kernel config changes.
3.2.3 Boot X86 machine with new kernel

- After the kernel compilation is complete, the `makefile` itself updates the `grub.cfg` file.
- Reboot the machine and select the new kernel from the grub menu.
- Subsequent build process must be done once the system has been booted with this kernel.

3.2.4 Build All software components

This section explains the procedure to build all the required software components with one command.

**NOTE**

Check if your board supports IOAT DMA or not.

- If it doesn’t, set `USE_HOST_DMA =n` in your `config.mk` file.
- If it does, set `USE_HOST_DMA =y` in your `config.mk` file.

$ make clean ARCH=x86: To clean the source

$ make ARCH=x86: To build firmware, driver, CLI app.

**NOTE**

Non root user should grant execute permission of the above commands and write permission for related directories.

Each component can be compiled individually, which is explained in below sub-sections.

3.2.4.1 Build firmware

Update `SDK_PATH` in `config.mk` file as:

```
SDK_PATH=<Absolute Path to SDK Build Directory>
```

$ make c29x_fw_clean: To clean the firmware

$ make c29x_fw: To compile the firmware
NOTE

• Firmware is same for X86 & P4080, firmware built once can be used for both the hosts.
• Non-root user should grant execute permission of the above commands and write permission for related directories.

3.2.4.2 Build driver

$ make c29x_driver_clean ARCH=x86 : To clean the host driver

$ make c29x_driver ARCH=x86 : To compile the host driver

NOTE
Any user id may be used (root or non-root) to build the driver.

If the host system is IOAT DMA capable and if driver has to use the DMA capability, then make sure in config.mk, set
USE_HOST_DMA=y.

3.2.4.3 Build CLI app

$ make c29x_cli_clean ARCH=x86 : To clean cli

$ make c29x_cli ARCH=x86 : To build cli

3.2.5 Binaries location

The location of all the compiled binaries is as follows:

<dir> :- Installation directory in which the source tar ball is extracted.
Driver :- <dir>/bin_images/x86_rc/fsl_crypto_offload_drv.ko
CLI :- <dir>/bin_images/x86_rc/c29x_cli
Firmware :- <dir>/bin_images/c29x_ep/u-boot-sd.bin

3.2.6 Default install location

make install copies all the following images from their above mentioned locations to /etc/crypto/. If it is not present, this directory is created by the makefile.

<dir>/crypto.cfg
<dir>/bin_images/x86_rc/c29x-cli
<dir>/bin_images/c29x_ep/u-boot-sd.bin
By default, `crypto.cfg` has the default location of the binaries. If any change has been made to the default location, `crypto.cfg` must be updated accordingly.

### 3.3 P4080 host

This section explains the build procedure for P4080 Host.

#### 3.3.1 Build P4080 linux kernel

P4080 linux kernel needs to be built only once, if a fresh SDK1.4 is used. This is because driver's standalone compilation needs pre-compiled kernel. Following is the command to compile kernel:

```
$ make p4080_kernel_clean : To clean p4080 kernel
$ make p4080_kernel : To compile p4080 kernel
```

A menu config screen is displayed in the build process. DMA related changes may be selected. See P4080 DMA support for details.

The compiled kernel image is in the standard SDK path: `.../build_p4080ds_release/tmp/deploy/images`. P4080DS should boot with this image.

**NOTE**

The kernel build command must be done from a non-root user id and the directory should have permission for this non-root user.

#### 3.3.2 Build All software components

```
$ make clean ARCH=ppc : To clean the source
$ make ARCH=ppc : To build firmware, driver, CLI app.
```

**NOTE**

The build command must grant execute permission of the above commands and write permission for related directories.

Each component may be compiled individually, as explained below.
3.3.2.1 Build firmware

$ make c29x_fw_clean : To clean the firmware

$ make c29x_fw : To compile the firmware

**NOTE**

- Firmware is same for X86 & P4080, firmware built once can be used for both the hosts.
- Non-root user should grant execute permission of the above commands and write permission for related directories.

3.3.2.2 Build driver

$ make c29x_driver_clean ARCH=ppc : To clean the host driver

$ make c29x_driver ARCH=ppc: To compile the host driver

**NOTE**

- Non-root user should grant execute permission of the above commands and write permission for related directories.
- If p4080DS host driver has to use the DMA capability, then make sure to give USE_HOST_DMA=y inside config.mk file.

3.3.2.3 Build CLI app

$ make c29x_cli_clean ARCH=ppc : To clean cli

$ make c29x_cli ARCH=ppc: To build cli

3.3.3 Location of the binaries

The locations of all the compiled binaries are as follows:

<dir> :- Installation directory in which the source tar ball is extracted.
Driver :: <dir>/bin_images/p4080_rc/fsl_crypto_offload_drv.ko
CLI :: <dir>/bin_images/p4080_rc/c29x_cli
Firmware :: <dir>/bin_images/c29x_ep/u-boot-sd.bin

These binaries may be copied using scp to the P4080DS host or can be built inside rootfs. For more information, see Build rootfs.
3.3.4 Build rootfs

This step is required when compiling openSSL package. Building rootfs also populates needed binaries and config files inside the rootfs. P4080DS may be booted up with this rootfs image.

$ make p4080_rootfs

The generated rootfs image can be flashed onto the P4080DS. Once the P4080DS is up, driver and firmware binaries can be found at:

/etc/crypto/u-boot-sd.bin
/etc/crypto/c29x_cli
/etc/crypto/crypto.cfg

/lib/modules/c2x0/fsl_crypto_offload_drv.ko
/lib/modules/3.8.13-rt9-QorIQ-SDK-V1.4/extra/cryptodev.ko

3.3.5 Booting the P4080DS with the new kernel & rootfs image

Refer to Freescale document for the procedure to flash the images and to boot the P4080DS.
Chapter 4
C29x firmware build configuration

This section explains the build time configuration parameters.

.../config.mk file is in the install directory. This file specifies different parameters with which software needs to be built. The following table explains the various parameters.

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDK_PATH</td>
<td>&lt;empty&gt;</td>
<td>SDK1.4 Path on the system. Required to build u-boot, firmware, driver, kernel, rootfs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>NOTE:</strong> This field must be provided.</td>
</tr>
<tr>
<td>P4080_EP</td>
<td>n</td>
<td>Defines the type of end point, for P4080_EP this variable should be set to 'y'. This field is for legacy support where in P4080DS can be used as EP.</td>
</tr>
<tr>
<td>C293_EP</td>
<td>y</td>
<td>Defines the type of end point, for C293_EP this variable should be set to 'y'.</td>
</tr>
<tr>
<td>DEBUG_PRINT</td>
<td>n</td>
<td>Defines the print level. If set to 'y', debug level prints are enabled.</td>
</tr>
<tr>
<td>INFO_PRINT</td>
<td>n</td>
<td>Defines the print level. If set to 'y', info level prints are enabled.</td>
</tr>
<tr>
<td>ERROR_PRINT</td>
<td>n</td>
<td>Defines the print level. If set to 'y', error level prints are enabled.</td>
</tr>
<tr>
<td>CONFIG_FSL_C2X0_HASH_OFFLOAD</td>
<td>n</td>
<td>To enable/disable hash offload support in driver.</td>
</tr>
<tr>
<td>CONFIG_FSL_C2X0_SYMMETRIC_OFFLOAD</td>
<td>n</td>
<td>To enable/disable symmetric algorithms offload support in driver.</td>
</tr>
<tr>
<td>RNG_OFFLOAD</td>
<td>n</td>
<td>To enable/disable RNG offload support in the driver.</td>
</tr>
<tr>
<td>USE_HOST_DMA</td>
<td>n</td>
<td>Specifies whether host DMA to be used by driver. Should be enabled/disabled based on the capability of host.</td>
</tr>
<tr>
<td>HIGH_PERF_MODE</td>
<td>y</td>
<td>To enable/disable high performance mode for driver and firmware. In high performance mode command ring support is disabled both from driver and firmware.</td>
</tr>
<tr>
<td>VIRTIO_C2X0</td>
<td>n</td>
<td>Specifies driver to work in Virtualization environment. All of the required support needs to be installed if this variable is made 'y'.</td>
</tr>
</tbody>
</table>

Table continues on the next page...
<table>
<thead>
<tr>
<th>Variable name</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENHANCE_KERNEL_TEST</td>
<td>n</td>
<td>Enhance PKC kernel test performance by disabling kernel test schedule and restricting enqueue/dequeue number of c29x_fw</td>
</tr>
</tbody>
</table>

The image below shows the default config.mk file.

```plaintext
# Device configurations

# SDK build path - Makefile has a target to build u-boot directly
SLK_PATH=

# Specifies type of FP
P4000_FP=n
C293_FP=y

# Controls the debug print level
DEBUG_PRINT=n
LIQUOR_PRINT=n
INFO_PRINT=n

# Enable HASH/SYMMETRIC offloading
CONFIG_FSL_C2X0_HASE_OFFLOAD=n
CONFIG_FSL_C2X0_SYMMETRIC_OFFLOAD=y

# Enable RNG offloading
RNG_OFFLOAD=n

# Specifies whether host DMA support to be enabled/disabled in the driver
USE_HOST_DMA=y

# Specifies whether driver/firmware is running high performance mode
HIGH_PERF_MODE=y

# Specify building host-driver to support virtualization
VIRTIO_C2X0=n

# Enhance pkc kernel test performance, disable kernel test schedule and restriction number of c29x_fw enqueue and dequeue crypto
ENHANCE_KERNEL_TEST=n
```

Figure 4-1. Default config.mk file
Chapter 5
Resource configuration

This section explains the configuration of different resources, which need to be created for driver-firmware to function properly.

These configuration options are specified in `/etc/crypto/crypto.cfg` file in the build. The build has a default version of the file, which is explained below.

This file contains the details about the firmware path and the ring pairs.

Ring details are as follows:

Ring: Ring is a circular queue that is used for job processing. There are two types of rings used in this driver. They are command ring and application ring. The command ring is used to send commands from the driver to the firmware. The command ring is disabled when `HIGH_PERF_MODE` flag is set in `config.mk`. The application rings are used to send crypto jobs from the driver to the firmware. Every ring contains the following details:

<table>
<thead>
<tr>
<th>depth</th>
<th>Ring size.</th>
</tr>
</thead>
<tbody>
<tr>
<td>affinity</td>
<td>Sec engine affinity.</td>
</tr>
<tr>
<td>priority</td>
<td>Relative priority of rings.</td>
</tr>
<tr>
<td>order</td>
<td>Whether the ring processing is ordered or not.</td>
</tr>
</tbody>
</table>

Ring details are application specific. Based on the requirements, user may change ring details before loading the driver module. User can create a maximum of 6 rings. If the ring configuration needs to be changed, the driver must be unloaded and reloaded for the new configuration to take effect.

The default configuration file specifies the configuration for a single device - two ring pairs (one command ring and one application ring) to be created for that device (command ring with a size of 16 entries and the application ring with 1024 entries).

/* Following configuration is for one device.
 * If more than one device is connected to the host, then
 * Please provide following config information per device.
* Start and end of each section is clearly demarcated,
* Please make sure same is followed if the config file
  * is changed (or) extended.
  */

/* Per device information - Start*/
<device>

/* Specifies the path of firmware to be loaded in this device.
 * Device will boot with this firmware image. Please note that firmware name
 * is constant, it can't be change.
 */

firmware:/etc/crypto/u-boot-sd.bin

/* Specifies number of ring pairs to be created. */
 rings:2

/* Per ring information - Start */
<ring>

/* Depth of each ring */
 depth:16

/* SEC affinity of ring */
 affinity:0

/* Relative priority of ring */
priority:1

/* Whether the ring is ordered (or) un-ordered */
 order:0

<end>
/* Per ring information - End */

/* Same as above */

/* Per ring information - Start */
<ring>
 depth:1024
 affinity:0
 priority:1
 order:0
<end>
/* Per ring information - End */
</end>

/* Per device information - End */
Chapter 6
Load driver

6.1 For X86 host
- Compile the driver, procedure is explained in X86 host.
- The driver uses the following default settings:
  - Firmware binary at /etc/crypto/u-boot-sd.bin
  - Configuration file at /etc/crypto/crypto.cfg
- To load driver with default settings:
  - make install
  - make insmod
- The user may override the default settings at driver load time using the command below:
  - insmod <dir>/bin_images/x86_rc/fsl_crypto_offload_drv.ko dev_config_file=<path>

6.2 For P4080 host
- Compile the driver.
- Inside the P4080DS:
  - insmod /lib/modules/c2x0/fsl_crypto_offload_drv.ko dev_config_file=<path>

1. dev_config_file = Path of configuration file with which driver boots up. See Resource configuration for the format and details of this file.
2. Driver as part of its load process downloads the specified firmware to the device and boots up the device.
3. Driver then performs handshake with firmware and creates the resources specified.
4. After all the operations are completed, following print can be seen in kernel logs

    root@p4080ds:~# dmesg | tail
    [FSL-CRYPTO-OFFLOAD-DRV] DevId:1 DEVICE IS UP

5. Driver state can also be verified from sysfs entry

    root@p4080ds:~# cat /sys/fsl_crypto/fsl_crypto_1/state
    DRIVER READY
6. Driver is up now and can be used to send crypto operations.
**Chapter 7**
**Driver load time configurations**

Following module parameters can be provided at the load time for driver:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
<th>Description</th>
<th>Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>dev_config_file</td>
<td>NULL</td>
<td>Device configuration file. Specifies all the configurations for the device, see the above section for details.</td>
<td>Y</td>
</tr>
<tr>
<td>napi_poll_count</td>
<td>1</td>
<td>Specifies the higher limit of the count for which worker threads should be polling.</td>
<td>N</td>
</tr>
<tr>
<td>wt_cpu_mask</td>
<td>One thread per cpu</td>
<td>Bit mask of CPUs on which worker threads needs to be scheduled.</td>
<td>N</td>
</tr>
<tr>
<td>dma_channel_count</td>
<td>1</td>
<td>DMA channel count for driver to use</td>
<td>N</td>
</tr>
<tr>
<td>dma_channel_cpu_mask</td>
<td>One channel for all CPUs</td>
<td>Defines the distribution of channels to CPUs</td>
<td>N</td>
</tr>
</tbody>
</table>

**Example:**

```
insmod fsl_crypto_offload_drv.ko dev_config_file=/etc/crypto/crypto.cfg napi_poll_count=10
wt_cpu_mask=0x3 dma_channel_count=8 dma_channel_cpu_mask=0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff
```
Chapter 8
Pre-built binaries

Pre-built binaries which are verified are uploaded as part of the release.

X86 Host binaries  <dir>/bin_images/x86_rc/
C293 EP images    <dir>/bin_images/c293_ep/
Chapter 9
Running the driver with openssl

Driver provides the crypto-offload to userspace applications through KCAPI (Kernel Crypto API). OpenSSL is one such application. Driver registers its algorithm support to the kernel through KCAPI.

9.1 Cryptodev Framework (CDF)
- This is /dev/crypto interface for the application.
- This module needs to be compiled and inserted in the host.
- Standard CDF does not have PKC extensions; a version of CDF is extended for PKC and is maintained in the build.

9.1.1 Procedure to compile and insert Cryptodev Framework

X86 Host

$ cd <dir>/crypto_patches/x86_deps/cryptodev-linux-1.5_v-2
$ make
$ insmod cryptodev.ko

P4080 host

CryptoDev Framework is built as part of rootfs build. For more information, see Build rootfs.

After P4080DS is up with this rootfs, the Cryptodev module may loaded using the command:

insmod /lib/modules/3.8.13-rt9-QorIQ-SDK-V1.4/extra/cryptodev.ko
9.2 Openssl

Standard openssl does not have PKC offload support in cryptodev engine (eng_cryptodev.c). A version of openssl is extended for PKC support and is maintained in the build.

9.2.1 Procedure to compile openssl

X86 Host

$ cd <dir>/crypto_patches/x86_deps/openssl-1.0.1c_v-2
$ ./config -D HAVE_CRYPTODEV
$ make
$ make install

P4080 host

Openssl is built as part of rootfs build. For more information, see Build rootfs.

Now, openssl commands can be used for any crypto operation.
Chapter 10
Driver test framework

Build supports driver test framework. This framework interacts directly with the driver through sysfs.

10.1 Script path
<dir>/perf/c29x_driver_perf_profile.sh

10.2 Supported tests
- RSA_PUB_OP_1K
- RSA_PUB_OP_2K
- RSA_PUB_OP_4K
- RSA_PRV_OP_1K
- RSA_PRV_OP_2K
- RSA_PRV_OP_4K
- DSA_SIGN_TEST_1K
- DSA_SIGN_TEST_2K
- DSA_SIGN_TEST_4K
- DSA_VERIFY_TEST_1K
- DSA_VERIFY_TEST_2K
- DSA_VERIFY_TEST_4K
- DSA_SIGN_VERIFY_TEST :- This test does the sign and verify both for 1K key.
- DSA_KEYGEN_TEST :- This test generates 1K keys, sign an image, verifies.
- ECDSA_KEYGEN_TEST :- This test generates 1K keys, sign an image, verifies.
- DH_KEYGEN_TEST
- ECDH_TEST
- ECDSA_VERIFY_TEST
- ECDSA_SIGN_TEST
- ECP_SIGN_TEST_256
10.3 Usage

$ bash c29x_driver_perf_profile.sh <test_name> [option]

test_name: Supported test names are mentioned in Supported tests.

Mandatory arguments for the test are as follows:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-m</td>
<td>cpu mask</td>
<td>Test threads are created on the CPUs masked by the value in above option.</td>
</tr>
<tr>
<td>-t</td>
<td>thread per cpu</td>
<td>Number of threads per cpu.</td>
</tr>
<tr>
<td>-s</td>
<td>Test duration in seconds</td>
<td>Test can be stopped anytime by ctrl + c.</td>
</tr>
<tr>
<td>-r</td>
<td>Test enqueue-dequeue count</td>
<td>Test will enqueue this many number of requests and completes only after getting this many number of responses.</td>
</tr>
</tbody>
</table>

**NOTE**

If -s & -r both are specified, then -s is selected by the test framework.

Example:
• $ bash c29x_driver_perf_profile.sh RSA_PUB_OP_1K -m 0xf -t 1 -s 10
• $ bash c29x_driver_perf_profile.sh RSA_PUB_OP_1K -m 0xf -t 1 -r 100000
• $ bash c29x_driver_perf_profile.sh --help
• -- help shows all the necessary help to run this test framework.

******** Result ***********

Test Name                   : 
Host CPU Frequency          : 
# job finished successfully : 
Per job in us               : 
Total jobs in 1 sec         : 

NOTE
• For P4080, this script is not a part of p4080 rootfs.
• Need to scp this script after P4080DS boots up.
• Since ppc kernel does not support floating point operations, ops/sec calculation is offloaded to a user space application program mini_calc.
• Source file is present at : <dir>/perf/min_calc/mini_calc.c
• This application needs to be cross compiled for P4080.
• The compiled ELF binary should be placed in the same directory where script is present inside P4080DS.
• Steps to run the script are same as explained above.
Chapter 11
CLI

Command line interface is provided on top of the driver to send control, debug and stat commands to firmware. CLI talks to driver through IOCTL.

CLI support is disabled, if HIGH_PERF flag in config.mk is enabled. For more information, refer to C29x firmware build configuration.

Refer to <dir>/docs/cli_readme.txt for help on CLI commands.
Chapter 12
Directory structure

|-- Makefile: Makefile with all the targets to build all the required software components.
|-- apps
|   |-- cli: CLI implementation
|-- bin_images
|   |-- c29x_ep
|   |-- p4080_rc
|   |-- x86_rc
|-- c29x-driver: Folder contains all the folders and files required for host driver
|   |-- Makefile
|   |-- algs: Folder contains all the algorithm implementation
|       |-- algs.c: Generic implementation required for all the algs.
|       |-- algs.h
|       |-- compat.h
|       |-- desc_constr.c
|       |-- desc_constr.h
|       |-- dh.c
|       |-- dh.h
|       |-- dsa.c
|       |-- dsa.h
|       |-- error.h
|       |-- hash.c
|       |-- hash.h
|       |-- rng.c
|       |-- rng.h
|       |-- rng_init.c
|       |-- rsa.c
|       |-- rsa.h
|       |-- sg_sw_sec4.h
|       |-- symdesc.h
|       |-- symmetric.c
|   |-- crypto_dev
|       |-- algs_reg.c
|       |-- algs_reg.h
|   |-- dcl: Folder contains the descriptor construction functions
|       |-- desc.h
|       |-- desc_constr.h
|       |-- pkc_desc.h
|-- host_driver
|   |-- command.c
|   |-- command.h
|   |-- common.h
|   |-- crypto_ctx.h
|   |-- device.h
|   |-- dma.c
|   |-- dma.h
|   |-- fsl_c2x0_crypto_layer.c: Crypto driver implementation.
|   |-- fsl_c2x0_crypto_layer.h
|   |-- fsl_c2x0_driver.c: PCI driver implementation
|   |-- fsl_c2x0_driver.h
|   |-- fsl_c2x0_virtio.h
|   |-- ioct1.h
|   |-- memmgr.c: Memory management code for the device memory
C29x Crypto Offload User Guide, Rev. 0, 10/2013

Freescale Semiconductor, Inc.
Appendix A
POR Configuration

The C29x PCIe card has user selectable switches or registers, for evaluating different frequency and boot configuration for this device. The table below shows how POR configuration is done through switches.

Table A-1. POR configuration through switches

<table>
<thead>
<tr>
<th>Switch</th>
<th>POR configuration</th>
<th>Signal name</th>
<th>Default setting</th>
<th>Signal meaning</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW4[1]</td>
<td>cfg_sys_pll[0]</td>
<td>IFC_AD0</td>
<td>ON</td>
<td>Rate between SYSCLK input and CCB clock (platform clock)</td>
<td>SW4[1,2,3] ON ON ON (000): 4:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SW4[1,2,3] ON ON OFF (001): 5:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SW4[1,2,3] ON OFF ON (010): 6:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Others are reserved.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SYSCLK on this board is 66.67 MHz.</td>
</tr>
<tr>
<td>SW4[4]</td>
<td>cfg_sys_speed</td>
<td>READY</td>
<td>OFF</td>
<td>0: SYSCLK frequency is at or below 66 MHz.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: SYSCLK frequency is above 66 MHz.</td>
<td></td>
</tr>
<tr>
<td>SW4[5]</td>
<td>cfg_core_pll[0]</td>
<td>IFC_AD3</td>
<td>OFF</td>
<td>Ratio between the e500 core clock and e500 CCB clock</td>
<td>SW4[5,6,7] ON ON ON (000): reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SW4[5,6,7] OFF ON (011): 1.5:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SW4[5,6,7] OFF OFF ON (100): 2:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SW4[5,6,7] OFF OFF OFF (101): 2.5:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SW4[5,6,7] OFF OFF OFF OFF ON (110): 3:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SW4[5,6,7] OFF OFF OFF OFF OFF ON (111): 3.5:1</td>
</tr>
<tr>
<td>SW5[1]</td>
<td>SW_CFG_ROM_LOC0</td>
<td>OFF</td>
<td>Boot ROM location</td>
<td>ON(0): Core clock frequency is greater than or equal to 600 MHz and less than 1001 MHz.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OFF(1): Core clock frequency is greater than or equal to 1001 MHz and less than 1201 MHz.</td>
<td></td>
</tr>
</tbody>
</table>

Table continues on the next page...
### Table A-1. POR configuration through switches (continued)

<table>
<thead>
<tr>
<th>Switch</th>
<th>POR configuration</th>
<th>Signal name</th>
<th>Default setting</th>
<th>Signal meaning</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW5[2]</td>
<td></td>
<td>SW_CFG_ROM_</td>
<td>OFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOC1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW5[3]</td>
<td></td>
<td>SW_CFG_ROM_</td>
<td>OFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOC2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW5[4]</td>
<td></td>
<td>SW_CFG_ROM_</td>
<td>OFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOC3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW5[5]</td>
<td>VCORE_MGN</td>
<td></td>
<td>ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW5[6]</td>
<td>BOOT_FLASH_SEL</td>
<td></td>
<td>ON</td>
<td>CS0/1 select</td>
<td></td>
</tr>
<tr>
<td>SW5[7]</td>
<td>F_BANK_SEL1</td>
<td></td>
<td>ON</td>
<td>NOR boot section choose</td>
<td></td>
</tr>
<tr>
<td>SW5[8]</td>
<td>F_BANK_SEL2</td>
<td></td>
<td>ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW6[1]</td>
<td>cfg_ddr_pll[0]</td>
<td>SW_TSEC1_TXD</td>
<td>ON</td>
<td>Clock ratio between 100MHz OSC clock input and DDR complex clock</td>
<td>SW6[1,2,3] ON ON ON (000): 8:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW6[4]</td>
<td>cfg_ddr_speed[0]</td>
<td>1588_CLK_OUT</td>
<td>ON</td>
<td>DDR complex speed configuration input</td>
<td>cfg_ddr_speed[0]:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OUT</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Others are reserved.

ON(0): CPU will run at v1.0 core voltage
OFF(1): CPU will run at v1.05 core voltage

ON (0): CS0 is connected to NOR flash; CS1 is connected to NAND flash.
OFF (1): CS0 is connected to NAND flash; CS1 is connected to NOR flash.

Set which section works as a boot section.

SW6[1,2,3] ON ON ON (000): 8:1
SW6[1,2,3] ON ON OFF (001): 10:1
SW6[1,2,3] ON OFF ON (010): 12:1
SW6[1,2,3] ON OFF OFF (011): 13:1
SW6[1,2,3] OFF ON ON (100): 14:1
SW6[1,2,3] OFF ON OFF (101): 15:1
SW6[1,2,3] OFF OFF ON (110): 16:1
SW6[1,2,3] OFF OFF OFF (111): Reserved

ON (0): DDR data rate is less than 967 MHz.
<table>
<thead>
<tr>
<th>Switch</th>
<th>POR configuration</th>
<th>Signal name</th>
<th>Default setting</th>
<th>Signal meaning</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW6[6]</td>
<td>cfg_plat_speed</td>
<td>IFC_PAR1</td>
<td>OFF</td>
<td>Platform speed configuration input</td>
<td>ON (0): Platform clock frequency is greater than or equal to 267 MHz and less than 320 MHz. OFF (1): Platform clock frequency is greater than or equal to 320 MHz and less than 401 MHz.</td>
</tr>
<tr>
<td>SW6[7]</td>
<td>cfg_boot_seq[0]</td>
<td>IFC_A26</td>
<td>OFF</td>
<td>Boot sequencer configuration options</td>
<td>SW6[7,8] ON OFF (01): Normal I²C addressing mode is used. Boot sequencer is enabled and loads configuration information from a ROM on the I²C1 interface. A valid ROM must be present. SW6[7,8] OFF ON (10): Extended I²C addressing mode is used. Boot sequencer is enabled and loads configuration information from a ROM on the I²C1 interface. A valid ROM must be present. SW6[7,8] OFF OFF (11): Boot sequencer is disabled. No I²C ROM is accessed. This is the default setting.</td>
</tr>
<tr>
<td>SW7[1]</td>
<td>cfg_cpu_boot</td>
<td>DMA_DDONE0_ N</td>
<td>OFF</td>
<td>CPU boot configuration inputs</td>
<td>ON (0): CPU boot hold off mode. The e500 core is prevented from booting until configured by an external master. OFF (1): The e500 core is allowed to boot without waiting for configuration by an external master.</td>
</tr>
</tbody>
</table>

*Table continues on the next page...*
### Table A-1. POR configuration through switches (continued)

<table>
<thead>
<tr>
<th>Switch</th>
<th>POR configuration</th>
<th>Signal name</th>
<th>Default setting</th>
<th>Signal meaning</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW7[5]</td>
<td>cfg_host_agt</td>
<td>IFC_A23</td>
<td>ON</td>
<td>Host/agent reset configuration input</td>
<td>ON (0): Agent on PCI express interface&lt;br&gt;OFF (1): Host/RC on PCI express interface</td>
</tr>
<tr>
<td>SW7[6]</td>
<td>cfg_sb_dis</td>
<td>HRESET_REQ_N</td>
<td>OFF</td>
<td></td>
<td>ON(0): Secure boot enabled&lt;br&gt;OFF(1): Secure boot disabled (POR)</td>
</tr>
<tr>
<td>SW7[7]</td>
<td>cfg_svr[0]</td>
<td>SW_TSEC2_TXD 0</td>
<td>OFF</td>
<td></td>
<td>Used to check which processor is set, C291, C292, or C293.</td>
</tr>
<tr>
<td>SW8[1]</td>
<td>cfg_gpinput[0]</td>
<td>IFC_AVD</td>
<td>ON</td>
<td>General-purpose POR configuration vector to be placed in GPPORCR</td>
<td>Software can then use this value to inform the operating system about initial system configuration. Typical interpretations include circuit board type, board ID number, or a list of available peripherals.</td>
</tr>
<tr>
<td>SW8[5]</td>
<td>cfg_eng_use[0]</td>
<td>SW_EC_MDC</td>
<td>OFF</td>
<td>To be used in the future to control functionality.</td>
<td>SW8[5:6] OFF OFF (1:1): Default operation&lt;br&gt;Others are reserved.</td>
</tr>
<tr>
<td>SW8[6]</td>
<td>cfg_eng_use[1]</td>
<td>UART0_RTS_N</td>
<td>ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW8[7]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW8[8]</td>
<td></td>
<td>TEST_SEL_N</td>
<td>OFF</td>
<td>PKCAL/SKMM mode</td>
<td>ON (0): PKCAL mode&lt;br&gt;OFF (1): SKMM mode</td>
</tr>
</tbody>
</table>

The table below shows the POR configuration through registers.

### Table A-2. POR configuration through registers

<table>
<thead>
<tr>
<th>Register</th>
<th>POR configuration</th>
<th>Signal name</th>
<th>Default setting</th>
<th>Signal meaning</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>R247</td>
<td>cfg_dram_type</td>
<td>1588_ALARM_OUT</td>
<td>NC (1)</td>
<td>Different voltage level from DDR3L</td>
<td>0: DDR3L 1.35V, CKE low at reset&lt;br&gt;1: DDR3 1.5V, CKE low at reset</td>
</tr>
</tbody>
</table>
### Table A-2. POR configuration through registers (continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>POR configuration</th>
<th>Signal name</th>
<th>Default setting</th>
<th>Signal meaning</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>R248</td>
<td>cfg_ddr_pll_backup</td>
<td>TSEC1_TXD3</td>
<td>NC (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
<td>1: Enabled</td>
</tr>
<tr>
<td>R249</td>
<td>cfg_ddr_half_full_mode</td>
<td>TSEC2_TXD1</td>
<td>NC (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Half mode</td>
<td>1: Full mode</td>
</tr>
<tr>
<td>R250</td>
<td>cfg_ec1_prtc</td>
<td>TSEC2_TXD3</td>
<td>NC (1)</td>
<td>Ethernet interface mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: RMII</td>
<td>1: RGMII</td>
</tr>
<tr>
<td>R251</td>
<td>cfg_ec2_prtc</td>
<td>TSEC1_TXD2</td>
<td>NC (1)</td>
<td>Ethernet interface mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: RMII</td>
<td>1: RGMII</td>
</tr>
<tr>
<td>R265</td>
<td>cfg_ifc_pb[0]</td>
<td>IFC_AD9</td>
<td>NC (1)</td>
<td>Corresponding pages per block if NAND flash is used for booting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>000: Reserved</td>
<td></td>
</tr>
<tr>
<td>R266</td>
<td>cfg_ifc_pb[1]</td>
<td>IFC_AD10</td>
<td>4.7k (0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>010: 1k pages per block</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>000: 512 pages per block</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100: 256 pages per block</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>101: 128 pages per block</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>110: 64 pages per block</td>
<td></td>
</tr>
<tr>
<td>R267</td>
<td>cfg_ifc_pb[2]</td>
<td>IFC_AD11</td>
<td>NC (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>111: 32 pages per block</td>
<td></td>
</tr>
<tr>
<td>R273</td>
<td>cfg_ifc_ecc_mode[0]</td>
<td>IFC_A25</td>
<td>NC (1)</td>
<td>IFC ECC correction mode if NAND flash is used for booting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00: 4b correction per 520 Byte sector</td>
<td></td>
</tr>
<tr>
<td>R274</td>
<td>cfg_ifc_ecc_mode[1]</td>
<td>IFC_A27</td>
<td>NC (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01: 8b correction per 520 Byte sector</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10: 24b correction per 520 Byte sector</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11: 40b correction per 520 Byte sector</td>
<td></td>
</tr>
<tr>
<td>R275</td>
<td>cfg_ifc_ecc_dec_en</td>
<td>IFC_A21</td>
<td>4.7k (0)</td>
<td>Enable IFC ECC checking on boot if NAND flash is used for booting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: ECC decoding disabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: ECC decoding enabled</td>
<td></td>
</tr>
<tr>
<td>R276</td>
<td>cfg_ifc_flash_mode</td>
<td>IFC_A22</td>
<td>NC (1)</td>
<td>Type of NOR/ NAND flash used for booting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: For NOR, multiplexed NOR flash (AVD type); for NAND, bad block indicator is at page 0 and at last page of each block.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: For NOR, normal asynchronous NOR flash; for NAND, bad block indicator is at page 0 and page 1 of each block.</td>
<td></td>
</tr>
<tr>
<td>R277</td>
<td>cfg_ifc_adm_mode</td>
<td>IFC_AD15</td>
<td>NC (1)</td>
<td>Which address bits are multiplexed with IFC data if NOR flash is used for booting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Lower order address bits are multiplexed with data on IFC_AD[0:15]</td>
<td></td>
</tr>
</tbody>
</table>

*Table continues on the next page...*
<table>
<thead>
<tr>
<th>Register</th>
<th>POR configuration</th>
<th>Signal name</th>
<th>Default setting</th>
<th>Signal meaning</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>R278</td>
<td>cfg_ifc_te</td>
<td>IFC_TE</td>
<td>NC (1)</td>
<td>IFC transceiver enabled</td>
<td></td>
</tr>
<tr>
<td>R279</td>
<td>cfg_srds_refclk</td>
<td>IFC_AD12</td>
<td>NC (1)</td>
<td>Input SerDes reference clock</td>
<td></td>
</tr>
<tr>
<td>R493</td>
<td>cfg_srds_pll_timeout_en</td>
<td>ASLEEP</td>
<td>NC (1)</td>
<td>Enable SerDes PLL timeout</td>
<td></td>
</tr>
<tr>
<td>R280</td>
<td>cfg_por_bist</td>
<td>IFC_OE_N</td>
<td>NC (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R284</td>
<td>cfg_fuse_rd_en</td>
<td>IFC_PAR0</td>
<td>NC (1)</td>
<td>Enable security fuse read</td>
<td></td>
</tr>
<tr>
<td>R286</td>
<td>cfg_test_port_mux_sel</td>
<td>UART1_TXD</td>
<td>NC (1)</td>
<td>Test port MUX select</td>
<td></td>
</tr>
<tr>
<td>R287</td>
<td>cfg_test_port_dis</td>
<td>IFC_WP_N</td>
<td>NC (1)</td>
<td>Disable test port</td>
<td></td>
</tr>
<tr>
<td>R289</td>
<td>cfg_60x</td>
<td>TSEC2_TXD2</td>
<td>NC (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R290</td>
<td>cfg_pcc_drowsy_en</td>
<td>IFC_A20</td>
<td>4.7k (0)</td>
<td>Enable PPC drowsy</td>
<td></td>
</tr>
<tr>
<td>R425</td>
<td>cfg_sdram_drowsy_en</td>
<td>IRQ_OUT_N</td>
<td>4.7k (0)</td>
<td>Enable SDRAM drowsy</td>
<td></td>
</tr>
</tbody>
</table>

In the above tables, ON indicates 0 and OFF indicates 1.