**Features**

**32-bit ARM Cortex-M0 Core**

* Nested Vectored Interrupt Controller(NVIC): 1 Wake up and 1 peripheral interrupt
* 16-bit or 32-bit System timer (Sys Tick): System timer for OS task management
* Creation and Management of Cipher Key

**On-chip Memory**

* EEPROM
  + 512 Kbits
  + Configuration/Key/User data storage
  + 15 User Zones of 2 Kbits Each
  + Retention 10 years
  + Erase/Write Endurance: 100K
* SRAM
  + On chip 64 Kbytes

**Serial Interface**

* UART
  + Full duplex double buffer
  + Parity can be enabled or disabled
  + Built-in dedicated baud rate generator
  + Various error detection functions (parity error, framing errors, and overrun errors)
  + External X-tal for UART
* SPI0
  + Slave, Mode 0
  + Up to 40 MHz SCK
  + Symmetric cipher core control
* SPI1
  + Master/Slave
  + Master: Up to 10 MHz SCK
  + Slave: Up to 1.5 MHz SCK
  + Mode 0, 1, 2, 3
* GPIO
  + 8 GPIOs

**Clock, Reset and Voltage**

* Clock
  + Built in OSC.
  + Main Clock: 50/200 MHz
* Reset
  + Built in power on reset
  + Software reset
* 1.5V, 3.3V Supply Voltage

**Debug**

* Serial Wire Debug Port(SW-DP)

**Low Power Consumption Mode**

* The GPIO is sufficient to power up and down
* PMU clock gating of Cortex-M0

**Asymmetric cipher function**

* ECC-P256, RSA-2048
* ECDSA
* ECDH

**Symmetric cipher function**

* AES-128/256
* Modes of Operation: Confidentiality (ECB, CBC, CFB, OFB, CTR)
* Creation and Management of Cipher Key

**Crypto Device function**

* User ID, User Serial (Manufacture ID), MIDR, RVC

**HASH Algorithm**

* SHA-256 for authentication

**PUF**

* Generate random using two different phase counters

**Application**

* Print cartridge, GPS, Navigation
* Mobile Device, IPC, CCTV, DVD
* Set-Top Boxes (STBs), Etc.

**Standards**

* ECC, RSA FIPS 186-3, 186-4
* AES-128/256 FIPS 140-2
* SHA-256 FIPS 180
* TRNG NIST SP 800-90B compatibility

**Benefits**

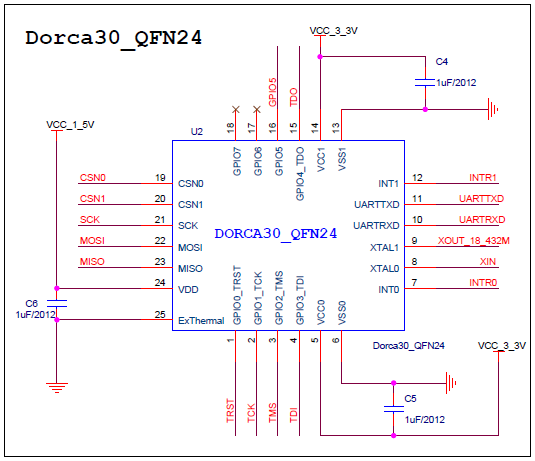
* Read/Write, Encrypted, or Read-only User Zone Options
* Ease use of Crypto Device to replace of existing EEPROM devices
* Authenticate Consumer products, Components, and Network equipment
* Protect Sensitive Firmware
* Securely Store Sensitive Data
* Manage Warranty Claims
* Securely Store Identity Data



Block Diagram

**Schematic Diagram**

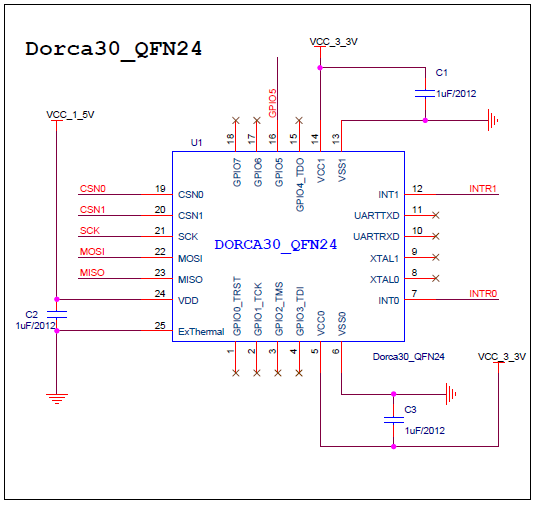
* MCU Application



**DORCA-3 QFN24**

**DORCA-3 QFN24**

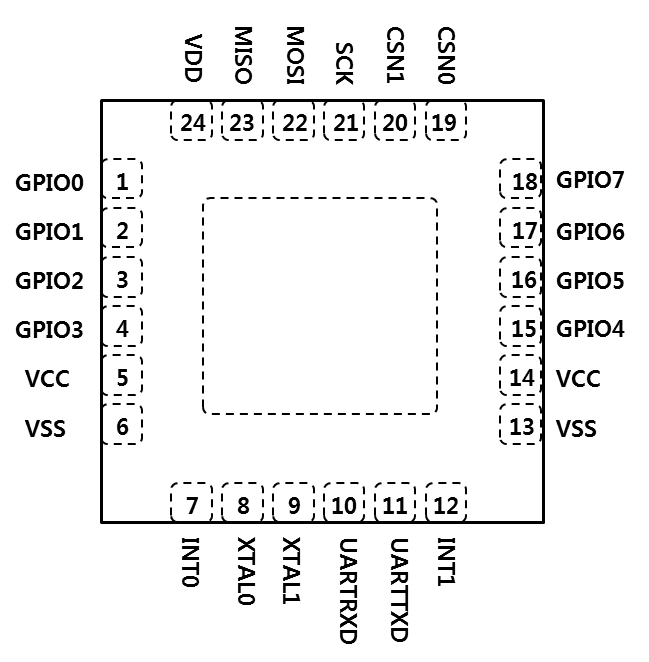
* Stand Alone Application



**DORCA-3 QFN24**

**DORCA-3 QFN24**

**Package**

* 24-LEAD (4mm X 4mm X 0.75mm) PLASTIC QFN

Package Top View

Contents

[Contents 3](#_Toc503795580)

[Figure 4](#_Toc503795581)

[1 Introduction 5](#_Toc503795582)

[1.1 Applications 5](#_Toc503795583)

[1.2 Device Features 5](#_Toc503795584)

[1.3 Crypto Operation 5](#_Toc503795585)

[2 Device configurations 6](#_Toc503795586)

[2.1 Symmetric Cipher Parts 6](#_Toc503795587)

[2.2 Asymmetric Cipher Parts 9](#_Toc503795588)

[3 Device Functions 11](#_Toc503795589)

[4 E-MCU to DORCA-3 Interface 13](#_Toc503795590)

[4.1 SPI0 Interface 13](#_Toc503795591)

[4.2 SPI1 Interface 14](#_Toc503795592)

[5 Address Map 15](#_Toc503795593)

[5.1 CORTEX-M0 AMBA Bus Address Map 15](#_Toc503795594)

[5.2 Symmetric Cipher parts Address Map 16](#_Toc503795595)

[6 Registers 18](#_Toc503795596)

[6.1 SPI0 registers 18](#_Toc503795597)

[6.2 CORTEX-M0 registers 43](#_Toc503795598)

[7 EEPROM Configuration 48](#_Toc503795599)

[8 Revision History 49](#_Toc503795600)

Figure

[Figure 2‑1 Main Control state machine diagram 7](#_Toc503795576)

[Figure 4‑1 SPI0 Normal Mode Write in Address Mode 13](#_Toc503795577)

[Figure 4‑2 SPI0 Normal Mode Read in Address Mode 13](#_Toc503795578)

[Figure 4‑3 Motorola SPI frame format with SPO=0 and SPH=1 14](#_Toc503795579)

# Introduction

The following sections introduce the features and functions of the DORCA-3 crypto device.

## Applications

DORCA-3 is designed to apply high security rules to the product. These security rules can be used to protect the data, to protect the functionality of the product, and to prevent replication.

* Product authentication

DORCA-3 has the function of preventing reproduction or illegal modification of products.

* Exchanging Security Keys

DORCA-3 has Public-Key Cryptosystems. User can use this function to exchange keys safely.

* Storing Security Data

You can store secret keys used for ciphering. Can save configuration, calibration or other secret data.

## Device Features

The DORCA-3 has an Electrically Erasable Programmable Read-Only Memory (EEPROM). The EEPROM can be used for key storage, miscellaneous write/read data, read-only, secret data, consumption logging, and security configuration. DORCA-3 has 32-bit ARM Cortex-M0 Core. This core is in charge of public key operation. DORCA-3 has 64 Kbytes SRAM, it is used for M0 code execution region and user code region. DORCA-3 has SPI0, SPI1, UART and GPIO interfaces. SPI0 can have a slave mode. With SPI0 user can control symmetric cipher core. SPI1 can have both a slave and a master mode. With SPI1 in a slave mode, user can control asymmetric cipher core. DORCA-3 has a power saving mode. In sleep mode internal oscillator is disabled. DORCA-3 has a symmetric cipher function which is ECC-P256, ECDSA and ECDH. DORCA-3 has a symmetric cipher function which is AES-128/256. AES supports ECB, CBC, CFB, OFB, CTR operating modes. DORCA-3 has SHA hash function and PUF function. DORCA-3 can save a User ID, a User Serial (Manufacture ID) and MIDR counter values.

## Crypto Operation

DORCA-3 save control information to the EEPROM. These control information is a configuration data. The configuration data is protected by password. DORCA-3 can encrypt or decrypt an input data with AES. And the result is read by an external MCU. DORCA-3 encrypt user data and save to a EEPROM. External MCU(E-MCU) can read saved data. When E-MCU request the saved data, the DORCA-3 returns encrypted data. DORCA-3 has authentication function using SHA.

# Device configurations

DORCA-3 is composed of two parts. One is asymmetric cipher part and the other is symmetric cipher part. The asymmetric cipher part is composed of a CORTEX-M0 and an asymmetric cipher hardware. The symmetric cipher part is a Security Processor Unit(SPU). The symmetric cipher part is composed of a symmetric cipher hardware and a main control hardware. The asymmetric cipher part take charge of ECC-P256, RSA-2048, ECDSA and ECDH. The symmetric part take charge of AES-128/256, SHA-256 and main control function. The main control function consists of state machine hardware. The following sections explain operation of each functions. An external MCU controls DORCA-3. The DORCA-3 has two interfaces to the external MCU. One is SPI0 for the symmetric cipher part. The other is SPI1 for the asymmetric cipher part. The external MCU can control DORCA-3 main control hardware through SPI0 interface. The external MCU can control a CORTEX-M0 and asymmetric cipher hardware through SPI1.

## Symmetric Cipher Parts

The Symmetric Cipher part consist of a Symmetric Cipher Hardware and a Main Control Hardware. First the Main Control Hardware parts are as follows.

### Main Control Hardware

The Main Control Hardware can have 14 main states. Each main state has independent operation. Most of operations are processed in one main state, but some operations are processed in several main states. When the DORCA-3 wakes up, it processes the initial procedures automatically, then goes to ST0\_STANDBY state. Usually, if DORCA-3 finishes a certain function, it always goes to ST0\_STANDBY state. A hardware logic sends main state to ST0\_STANDBY state when the DORCA-3 finishes a function, or E-MCU must control to send main state to ST0\_STANDBY state in some functions. If DORCA-3 finishes state abnormally, it may can’t process another function normally.



Figure 2‑1 Main Control state machine diagram

Figure 2‑1 shows every state which the main state can have. When power is on, the DORCA-3 begins an initial procedure. The initial procedure starts from ST0\_IDLE state, and stops to ST0\_STANDBY state. The initial procedure starts automatically when the power is up. User can skip ST0\_CHK\_RSFLAG state. See the following sections to control skip function of ST0\_CHK\_RSFLAG state.

#### ST0\_CHK\_RSFLAG state control

This state starts if a power on reset is entered on the DORCA-3 or if you control a software reset in register block. E-MCU can control the software reset register through SPI0 interface. The software reset register name and address are RG\_SOFT\_RESET[0]:RG\_SWRESET (0x1\_0600).

In this state the HW restores backup data to a corresponding EEPROM page. In this procedure the HW uses a restore control information. The restore control information is already stored in the EEPROM. This restore control information was generated when the backup process was made.

If you want to skip this state, you can control writing “1” to a certain EEPROM bit. This EEPROM bit field name and address is EE\_MEM\_BKUP\_RSFLAG[1]:EE\_MEM\_BKUP\_CTRL[0]:EE\_MEM\_BKUP\_NOTUSE (0x0\_EF01). EE\_MEM\_BKUP\_RSFLAG is a EEPROM page name. EE\_MEM\_BKUP\_RSFLAG page address is 0x0\_EF00. EE\_MEM\_BKUP\_CTRL is a byte field name. EE\_MEM\_BKUP\_NOTUSE is a bit field name. If EE\_MEM\_BKUP\_NOTUSE bit value is ‘1’, then the main state control skips the ST0\_CHK\_RSFLAG state.

#### ST0\_PON\_READ state control

The hardware prepares initial values to process a normal operation. The DORCA-3 main control hardware executes this state automatically.

#### ST0\_CPEEP2SRAM state control

The hardware load CORTEX-M0 image. The DORCA-3 main control hardware executes this state automatically.

#### ST0\_CM0 state control

The hardware cannot set this state, the E-MCU can set this state through SPI0. The E-MCU controls RG\_ST0\_OPMODE(0x1\_0604) register to set ST0\_CM0 state. If the E-MCU want to communicate with CORTEX-M0, E-MCU sets ST0\_CM0 state first. After then, E-MCU sends control information to CORTEX-M0. If CORTEX-M0 receive control information, it controls asymmetric cipher hardware. Some of CORTEX-M0 control may affect the symcipher hardware blocks. A detail explanation is given later of this document.

#### ST0\_STDSPI state control

The hardware cannot set this state, the E-MCU can set this state through SPI0. The E-MCU controls RG\_ST0\_OPMODE(0x1\_0604) register to set ST0\_STDSPI state. The E-MCU control registers in this state.

But some of registers is controlled in the other state. DORCA-3 runs SHA-256 function in ST0\_STDSPI state.

#### ST0\_EE\_CFG state control

The hardware cannot set this state, the E-MCU can set this state through SPI0. The E-MCU controls RG\_ST0\_OPMODE(0x1\_0604) register to set ST0\_EE\_CFG state. DORCA-3 has a EEPROM which contains configuration information. These configuration information is used for DORCA-3 normal operation. A E-MCU can write or read a value on the EEPROM in this state.

#### ST0\_RANDOM state control

The hardware cannot set this state, the E-MCU can set this state through SPI0. The E-MCU controls RG\_ST0\_OPMODE(0x1\_0604) register to set ST0\_EE\_RANDOM state. DORCA-3 can generate a random value in this state.

#### ST0\_SYMCIP state control

The hardware cannot set this state, the E-MCU can set this state through SPI0. The E-MCU controls RG\_ST0\_OPMODE(0x1\_0604) register to set ST0\_SYMCIP state. DORCA-3 operates symmetric cipher functions in this state.

#### ST0\_OKA state control

The hardware cannot set this state, the E-MCU can set this state through SPI0. The E-MCU controls RG\_ST0\_OPMODE(0x1\_0604) register to set ST0\_OKA state. OKA stands for Opaque Key Assignment. OKA is a secure data exchange algorithm which include secure key exchange between two users. OKA algorithm is the intellectual property right of NEOWINE.

#### ST0\_MIDR state control

The hardware cannot set this state, the E-MCU can set this state through SPI0. The E-MCU controls RG\_ST0\_OPMODE(0x1\_0604) register to set ST0\_MIDR state. MIDR stands for Monotonic Increase Decrease Register. The MIDR counter is a counter with an increment or decreasing output in one direction. Increase or decrease can be specified by the user. Users can then select the counter change direction to increase or decrease and hold it. E-MCU can only change the MIDR counter value in one direction when the change direction is fixed.

#### ST0\_PERM\_GET state control

The hardware cannot set this state, the E-MCU can set this state through SPI0. The E-MCU controls RG\_ST0\_OPMODE(0x1\_0604) register to set ST0\_PERM state. The user uses this state to acquire five permissions. The user needs the permission to use some functions. In this case, the user acquires the permission from this state. When the user completes the action of the required function, the user returns the permission.

#### ST0\_EE\_OW\_CTRL state control

The hardware cannot set this state, the E-MCU can set this state through SPI0. The E-MCU controls RG\_ST0\_OPMODE(0x1\_0604) register to set ST0\_OW\_CTRL state. This state is used for the user clear EEPROM for security reasons, or the contents excluded from the Root Serial 0 (EE\_RS\_ x0).

Second the Symmetric Cipher Hardware configurations are as follows.

### Symmetric Cipher Hardware

Symmetric cipher hardware handles the encryption and decryption using AES and the authentication using SHA. It also generates random. It manages writing and reading of EEPROM. It manages a key generation, key storage and key change.

## Asymmetric Cipher Parts

Asymmetric cipher parts consist of the CORTEX-M0 and the Asymmetric Cipher Hardware. The CORTEX-M0 controls the asymmetric Cipher Hardware. The Asymmetric Cipher Hardware is responsible for performing ECC, ECDH and ECDSA algorithms.

### CORTEX-M0 Hardware

#### Cortex-M0 Core

The Cortex-M0 processor is an entry-level 32-bit ARM Cortex processor designed for a broad range of embedded applications. It offers significant benefits to developers, including:

• simple, easy-to-use programmers model

• highly efficient ultra-low power operation

• excellent code density

• deterministic, high-performance interrupt handling

• upward compatibility with the rest of the Cortex-M processor family.

#### Cortex-M0 Core Peripherals

The Cortex-M0 core peripherals are:

**NVIC**

An embedded interrupt controller that supports low latency interrupt processing.

**System Control Block**

The System Control Block (SCB) is the programmers model interface to the processor. It provides system implementation information and system control, including configuration, control, and reporting of system exceptions.

**Optional system timer**

The optional system timer, SysTick, is a 24-bit count-down timer. If implemented, use this as a Real Time Operating System (RTOS) tick timer or as a simple counter.

**Public Key Engine (Asymmetric Cipher)**

The PK Crypto Engine is a very flexible solution based on a scalable array of dual-field processing elements that can be used to execute all operations & algorithms required for PK Crypto-systems:

•Elliptic Curve Cryptography (ECC)

•Diffie-Hellman (D-H & ECD-H) Key Exchange

•Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)

• Primality Test (Rabin-Miller) & Key Generation

• Any other crypto algorithm can be supported on request

**Embedded EEPROM and SRAM**

Dorca-3 has 64 KB of EEPROM and SRAM. The EEPROM stores the cortex-M0 code and the rom code of the asymmetric cipher. The SRAM is shared by the Cortex-M0 and Asymmetric Cipher.

**Registercm0**

Registercm0 is basically used by Cortex M0 to control the EEPROM. Generates the control signals needed to write to or read from the EEPROM. It is also used when selecting IO or SPI, and also used when Cortex M0 generates a random value.

**Timers**

The Dual Input Timers module, Timers is an AMBA slave module and connects to the APB. The Dual-Timer module consists of two programmable 32/16-bit down counters that can generate interrupts on reaching zero. A Timer module can be programmed for a 32-bit or 16-bit counter size and one of three timer modes using the Control Register. The operation of each Timer module is identical. It has one of three timer modes:

• free-running

• periodic

• one-shot

**UART**

The UART is an AMBA slave module that connects to the Advanced Peripheral Bus (APB).

The UART provides:

• Compliance to the AMBA Specification (Rev 2.0) onwards for easy integration into SoC implementation.

• Separate 16x8 transmit and 16x12 receive First-In, First-Out memory buffers(FIFOs) to reduce CPU interrupts.

• Programmable FIFO disabling for 1-byte depth.

• Programmable baud rate generator. This enables division of the reference clock by (1x16) to (65535 x16) and generates an internal x16 clock. The divisor can be a fractional number enabling you to use any clock with a frequency >3.6864MHz as the reference clock.

• Standard asynchronous communication bits (start, stop and parity). These are added prior to transmission and removed on reception.

**GPIOs (General-Purpose Input/Output)**

The GPIO is a general purpose I/O device. It has the following properties:

• three registers : Data, Direction, Interrupt Registers

• 32 input or output lines with programmable direction

• word and halfword read and write access

• address-masked byte write to facilitate quick bit set and clear operations

• address-masked byte read to facilitate quick bit test operations

• maskable interrupt generation based on input value change.

**SPI**

It has SPI0 and SPI1, SPI0 only operates as slave, and it is used by EMCU to control Symmetric Cipher. SPI1 is used by Cortex-M0 to control Symmetric Cipher. It can operate as master or slave.

### Asymmetric Cipher Hardware

This Public Key Engine(Asymmetric Cipher) has following features.

•1 multiplier architecture

• ECC/ECDH/ECDSA operations up to 512 bits

• DSA up to 2048 bits

• Supports prime field GF(p) and binary field GF(2m) fields .

# Device Functions

### Key Load Function

User generate keys and store them in EEPROM. This process is key loading function. The keys can be saved in four places. The key length is 128 or 256 bits length. The four key names stored in EEPROM are EE\_AES\_KEY\_x0, EE\_AES\_KEY\_x1, EE\_AES\_KEY\_x2, and EE\_AES\_KEY\_x3. There is an order when making keys. The order must be created first by EE\_AES\_KEY\_x0 then EE\_AES\_KEY\_x1, EE\_AES\_KEY\_x2, EE\_AES\_KEY\_x3. For the method to create EE\_AES\_KEY\_ x0, E-MCU determines the key plain text (key) and encrypts AES encryption using the EE\_SEED\_KEY, then pass the cipher text to DORCA-3. DORCA-3 decrypt the cipher text with EE\_SEED\_KEY and save the plain text (key) to the EE\_AES\_KEY\_x0 EEPROM storage location. Create the following keys in the same way. DORCA-3 use EE\_AES\_KEY\_x0 as AES key to create EE\_AES\_KEY\_x1. DORCA-3 use EE\_AES\_KEY\_x1 as AES key to create EE\_AES\_KEY\_x2. DORCA-3 use EE\_AES\_KEY\_x2 as AES key to create EE\_AES\_KEY\_x3.

### AES Encryption(Decryption) Function

AESEncrypt control takes the plaintext from 16 bytes and encryptions and outputs ciphertext. The key used for AES encryption are notified to DORCA-3 by E-MCU using the RG\_EE\_KEY\_AES\_xN register. AESDecrypt control takes the ciphertext input from 16 bytes and decodes and outputs a plaintext. The keys used for AES decryption are notified to DORCA-30 by E-MCU using the RG\_EE\_KEY\_AES\_xN register.

### AES Encryption Write(Read) Function

With AESEncwrite control, 16 bytes of data can be written to the EE\_USER\_ZONE\_Mx area of the EEPROM. AESEncwrite procedures are as follows. E-MCU encrypts the 16 bytes plaintext into ciphertext using the EE\_KEY\_AES\_xN key and writes it to DORCA-30. For DORCA-3, receive the 16 bytes data and decrypt it using the EE\_KEY\_AES\_xN key and store it in the appropriate EEPROM. The AESEncREad procedure is as follows. E-MCU sends control that DORCA-3 reads user data 16 bytes from EEPROM. For DORCA-3, read the 16 bytes plain text from EEPROM and cipher it using EE\_KEY\_AES\_xN key. E-MCU reads the cipher text after waiting DORCA-3 finish decryption. AES Encryption Write (or Read) function encrypt or decrypt 16 bytes at one time. E-MCU can process 4 encryptions (or decryptions) continuously. But E-MCU should not control EEPROM address cross over the page boundary of EEPROM.

### OKA2 Function

OKA (Opaque Key Assignment) is a key exchange method that safely exchange keys between two DORCA-3. The key exchange between two DORCA-3 is called 1:1 mode. After key exchange, two DORCA-3 can exchange data using AES cipher. In 1:1 mode safe data exchange is possible between two DORCA-3. In 1:0 mode safe data exchange is possible between E-MCU and DORCA-3.

### EEPROM Erase Function

This function enables you to read or clear EEPROM specific information. User use this function for special purpose. Users should review sufficiently before using this feature to determine its intended use. This function is usually not used.

### PUF (Physical Uncronable Function)

This chapter describes how to create and use the Root Serial corresponding to the unique number of DORCA-3. The Root Serial is a unique number for each device. And this value is a fixed value and cannot be changed. When

#### Root Serial(RS) Creation Function

The Root Serial consists of four 256 bits values. For DORCA-3, use hardware random generator and root serial generation circuitry inside DORCA-3 to make the root serial. To make the Root Serial1, DORCA-3 use the Root Serial0. User can input a Root Serial2 an Root Serial3.

#### Root Serial(RS) SHA Read Function

User cannot read the Root Serial0 and the Root Serial2 directly. User can read the SHA result of them. The SHA input is generated by the Root Serial, the random value, and the user input data.

#### Root Serial(RS) Read Function

User can read the Root Serial1 and the Root Serial3 directly. But user has to get EE\_UID\_PW password permission to read them.

### Random Generation Function

Random Generator can generate random values in three ways. The first is to generate a random value through SPI0 when the user wants a random value. The second one can be created when a random value is desired in Cortex-M0. Finally, Symmetric cipher can generate and take random values when they are needed.

### MIDR Function

The MIDR (Monotonic Increasing or Decrease Register) is a function that can be written to and read from the register only in the direction of increasing or decreasing. The direction of increase or decrease can be set as desired by the user.

### SHA Authentication Function

This function allows SHA authentication between EMCU and DORCA-3 to be performed. The result of SHA performed in EMCU is compared with the result of SHA performed in DORCA-3 to derive SHA authentication result.

### ECDH (Elliptic Curve Diffie Hellman) Function

Elliptic Curve Diffie-Hellman key exchange is one way to generate key values on an elliptic curve and exchange encryption keys so that they can share a shared key with other keys on an unencrypted network. DORCA-3 supports ECC P-256, P384, P-521 curves etc and supports up to 512 bits.

### ECDSA(Elliptic Curve Digital Signature Algorithm) Function

ECDSA implements electronic signatures on elliptic curves and works on ECC P-256, P-384, and P-521 curves etc. ECDSA operations can be executed in both fields GF(p)-prime field or GF(2m)-binary field. ECDSA signatures can be generated and verified.

# E-MCU to DORCA-3 Interface

DORCA-3 has SPI0, SPI1, UART and GPIO interfaces. In generally DORCA-3 is used as a security function chip not as a MCU. When the DORCA-3 is used as a security function chip, SPI0 and SPI1 is slave mode. When the DORCA-3 is used as a MCU, SPI1 is master mode.

## SPI0 Interface

SPI0 has write / read protocol as shown in Figure 4-1 and 4-2 below. SPI0 is primarily used by external MCUs to control symmetric cipher.

### SPI0 Protocol Timing Diagram

#### SPI0 Normal Mode Write

Figure 4‑1 SPI0 Normal Mode Write in Address Mode



#### SPI0 Normal Mode Read

Figure 4‑2 SPI0 Normal Mode Read in Address Mode



## SPI1 Interface

SPI1 is used by EMCU to control PKE(Asymmetric Cipher) through Cortex-M0. It basically supports Motorola SPI frame type. The main feature of the Motorola SPI format is that the inactive state and phase of the SCK signal are programmable through the SPO and SPH bits within the SPI1 control register.

**SPO, clock polarity**

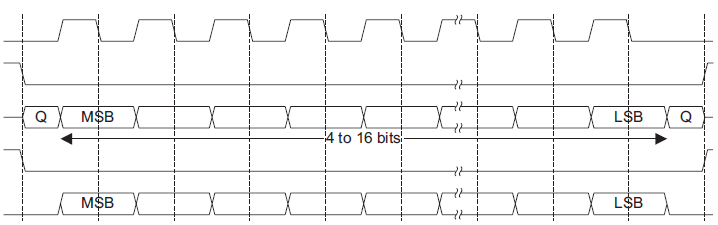
When the SPO clock polarity control bit is LOW, it produces a steady state low value on the SCK pin. If the SPO clock polarity control bit is HIGH, a steady state high value is placed on the SCK pin when data is not being transferred.

**SPH, clock phase**

The SPH control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the SPH phase control bit is LOW, data is captured on the first clock edge transition. If the SPH clock phase control bit is HIGH, data is captured on the second clock edge transition.

For Frame format used between EMCU and Cortex-M0, set SPO to 0 and SPH to 1. The transfer signal sequence for Motorola SPI format with SPO=0, SPH=1 is shown in Figure 4-3, which covers both single and continuous transfers.

Figure 4‑3 Motorola SPI frame format with SPO=0 and SPH=1



MISO

MOSI

SCK

CSN0

# Address Map

DORCA-3 has CORTEX-M0. The CORTEX-M0 has AMBA bus. And DORCA-3 has address map for the Symmetric Cipher parts and the Asymmetric Cipher parts.

## CORTEX-M0 AMBA Bus Address Map

Table 5‑1 CORTEX-M0 AMBA Bus Address Map



## Symmetric Cipher parts Address Map

Symmetric Cipher parts include EEPROM and the symmetric cipher core. E-MCU can access EEPROM and registers with SPI0 interface.

Table 5‑2 EEPROM and register Address Map(SPI0)

| **ADDR(HEX)** | **M0 ACCESS** | **CIP CORE ACCESS** | **Type** | **NAME/RANGE** | | | | **BYTE SIZE(DEC)** | | **DESCRIPTION** | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Group1** | **Group2** | **Group3** |  | |  | |
| 0x0E800 |  |  | EEPROM | EE\_CM0/ EE\_SYMCIP | EE\_KEY\_ASYMCIP\_xN | EE\_KEY\_ASYMCIP\_x0 . . EE\_KEY\_ASYMCIP\_x3 | 256 | | 1. KEY zone 2. Asymmetric Key storage area (64Byte \* 4) | |
| … |  |  |
| E8FF |  |  |
| E900 |  |  | EE\_SYMCI | EE\_KEY\_AES\_xN | EE\_KEY\_AES\_x0 . . EE\_KEY\_AES\_x3 | 256 | | 1. KEY zone 2. Symmetric Key storage area (64Byte \* 4) | |
| … |  |  |
| E9FF |  |  |
| EA00 |  |  | EE\_RS\_xN | EE\_RS\_x0 . . EE\_RS\_x3 | 256 | | 1. KEY zone 2. ROOT SERIAL storage area (64Byte \* 4) | |
| … |  |  |
| EAFF |  |  |
| EB00 |  |  | EE\_CONFIG | EE\_CONFIG\_NW 등 | 1536 | | 1. Configuration zone 2. This zone contains all information to control the CM0 zone, KEY zone, Configuration zone, and User zone. | |
| … |  |  |
| F0FF |  |  |
| F100 |  |  | EE\_USER\_ZONE\_M | EE\_USER\_ZONE\_M01 . . EE\_USER\_ZONE\_M15 | 3840 | | 1. 15 user zones 2. Store user data. | |
| … |  |  |
|  |  |  |
|  |  |  |
| 0x0FFFF |  |  |
| 0x10000 |  |  | IUM |  |  | 64 | | IUM(RESERVED) | |
| … |  |  |
| 1003F |  |  |
| … |  |  | REGISTER | RESERVED |  |  |  | |  | |
| 0x10100 |  |  | RG\_EEBUF |  |  | 64 | | Used for EEPROM write operations and BIST test application. Not used for EEPROM read operation. | |
| … |  |  |
| 1013F |  |  |
| 0x10140 |  |  | RESERVED |  |  |  | |  | |
| … |  |  | RESERVED |  |  |  | |  | |
| 0x10200 |  |  | RESERVED |  |  |  | |  | |
| … |  |  |
| 1023F |  |  |
| … |  |  | RESERVED |  |  |  | |  | |
| 0x10300 |  |  | RG\_EEBUF | RG\_ENCINBUF | RG\_ENCINBUF0 | 16 | | First ENC input 128 bits buffer share with RG\_CMDBUF[15:0]. | |
| … |  |  |
| 1030F |  |  |
| 0x10310 |  |  | RG\_ENCINBUF1 | 16 | | Second ENC input 128 bits buffer share with RG\_CMDBUF[31:16]. | |
| … |  |  |
| 1031F |  |  |
| 0x10320 |  |  | RG\_ENCOUTBUF | RG\_ENCOUTBUF0 | 16 | | First ENC output 128 bits buffer share with RG\_CMDBUF[47:32]. | |
| … |  |  |
| 1032F |  |  |
| 0x10330 |  |  | RG\_ENCOUTBUF1 | 16 | | Second ENC output 128 bits buffer share with RG\_CMDBUF[63:48]. | |
| … |  |  |
| 1033F |  |  |
| 0x10400 |  |  | RG\_DECINBUF | RG\_DECINBUF0 | 16 | | First DEC input 128 bits buffer share with RG\_CMDBUF[15:0]. | |
| … |  |  |
| 1040F |  |  |
| 0x10410 |  |  | RG\_DECINBUF1 | 16 | | Second DEC input 128 bits buffer share with RG\_CMDBUF[31:16]. | |
| … |  |  |
| 1041F |  |  |
| 0x10420 |  |  | RG\_DECOUTBUF | RG\_DECOUTBUF0 | 16 | | First DEC output 128 bits buffer share with RG\_CMDBUF[47:32]. | |
| … |  |  |
| 1042F |  |  |
| 0x10430 |  |  | RG\_DECOUTBUF1 | 16 | | Second DEC output 128 bits buffer share with RG\_CMDBUF[63:48]. | |
| … |  |  |
| 1043F |  |  |
| … |  |  | RESERVED |  |  |  | |  | |
| 10500 |  |  | RESERVED |  |  | 32 | | RESERVED | |
| … |  |  |
| 1051F |  |  |
| … |  |  | RESERVED |  |  |  | |  | |
| 0x10600 |  |  | RG\_SYMCIP |  |  | 512 | | DORCA-3 control registers. | |
| … |  |  |
| 107FF |  |  |

# Registers

## SPI0 registers

### RG\_EEBUF buffer address map

RG\_EEBUF is used for encryption, decryption and EEPROM write.

SPI0 address width is 17 digits.

| **ADDR(HEX)** | **WR** | **BIT** | **NAME/RANGE** | **DESCRIPTION** | **RESET  VALUE** | |
| --- | --- | --- | --- | --- | --- | --- |
| RG\_EEBUF100 Use this register to write and read the data in DORCA-3 internal RG\_EEBUF buffer. And it is used to write data to EEPROM. The RG\_EEBUF buffer size is 512 bits(64 Bytes). | | | | | |  | |
| 0x10100 | WR | [7:0] | MCU : RG\_EEBUF[0] | RG\_EEBUF[0] | 0x00 | |
| … | WR | [7:0] |  |  | 0x00 | |
| 1010F | WR | [7:0] | MCU : RG\_EEBUF[15] | RG\_EEBUF[15] | 0x00 | |
| 0x10110 | WR | [7:0] | MCU : RG\_EEBUF[16] | RG\_EEBUF[16] | 0x00 | |
| … | WR | [7:0] |  |  | 0x00 | |
| 1011F | WR | [7:0] | MCU : RG\_EEBUF[31] | RG\_EEBUF[31] | 0x00 | |
| 0x10120 | WR | [7:0] | MCU : RG\_EEBUF[32] | RG\_EEBUF[32] | 0x00 | |
| … | WR | [7:0] |  |  | 0x00 | |
| 1012F | WR | [7:0] | MCU : RG\_EEBUF[47] | RG\_EEBUF[47] | 0x00 | |
| 0x10130 | WR | [7:0] | MCU : RG\_EEBUF[48] | RG\_EEBUF[48] | 0x00 | |
| … | WR | [7:0] |  |  | 0x00 | |
| 1013F | WR | [7:0] | MCU : RG\_EEBUF[63] | RG\_EEBUF[63] | 0x00 | |
| 10140 |  |  | RESERVED |  |  | |
| 102FF |  |  | RESERVED |  |  | |
| RG\_EEBUF300 Use this registers as the input and output buffer when performing AES encryption. | | | | | |  | |
| 0x10300 | WR | [7:0] | RG\_EEBUF[0] MCU : RG\_ENCINBUF0[0] | Encoder0 input buffer or Key0 input buffer | 0x00 | |
| … | [7:0] | … | Encoder0 input buffer or Key0 input buffer | 0x00 | |
| 1030F | [7:0] | RG\_EEBUF[15] MCU : RG\_ENCINBUF0[15] | Encoder0 input buffer or Key0 input buffer | 0x00 | |
| 0x10310 | WR | [7:0] | RG\_EEBUF[16] MCU : RG\_ENCINBUF1[0] | Encoder1 input buffer or Key1 input buffer | 0x00 | |
| … | [7:0] | … | Encoder1 input buffer or Key1 input buffer | 0x00 | |
| 1031F | [7:0] | RG\_EEBUF[31] MCU : RG\_ENCINBUF1[15] | Encoder1 input buffer or Key1 input buffer | 0x00 | |
| 0x10320 | WR | [7:0] | RG\_EEBUF[32] MCU : RG\_ENCOUTBUF0[0] | Encoder0 output buffer | 0x00 | |
| … | [7:0] | … | Encoder0 output buffer | 0x00 | |
| 1032F | [7:0] | RG\_EEBUF[47] MCU : RG\_ENCOUTBUF0[15] | Encoder0 output buffer | 0x00 | |
| 0x10330 | WR | [7:0] | RG\_EEBUF[48] MCU : RG\_ENCOUTBUF1[0] | Encoder1 output buffer | 0x00 | |
| … | [7:0] | … | Encoder1 output buffer | 0x00 | |
| 1033F | [7:0] | RG\_EEBUF[63] MCU : RG\_ENCOUTBUF1[15] | Encoder1 output buffer | 0x00 | |
| RESERVED |  |  | RESERVED |  |  | |
| RG\_EEBUF400 Use this registers as the input and output buffer when performing AES decryption. | | | | | |  | |
| 0x10400 | WR | [7:0] | RG\_EEBUF[0] MCU : RG\_DECINBUF0[0] | Decoder0 input buffer or Key0 input buffer | 0x00 | |
| … | [7:0] | … | Decoder0 input buffer or Key0 input buffer | 0x00 | |
| 1040F | [7:0] | RG\_EEBUF[15] MCU : RG\_DECINBUF0[15] | Decoder0 input buffer or Key0 input buffer | 0x00 | |
| 0x10410 | WR | [7:0] | RG\_EEBUF[16] MCU : RG\_DECINBUF1[0] | Decoder1 input buffer or Key1 input buffer | 0x00 | |
| … | [7:0] | … | Decoder1 input buffer or Key1 input buffer | 0x00 | |
| 1041F | [7:0] | RG\_EEBUF[31] MCU : RG\_DECINBUF1[15] | Decoder1 input buffer or Key1 input buffer | 0x00 | |
| 0x10420 | WR | [7:0] | RG\_EEBUF[32] MCU : RG\_DECOUTBUF0[0] | Decoder0 output buffer | 0x00 | |
| … | [7:0] | … | Decoder0 output buffer | 0x00 | |
| 1042F | [7:0] | RG\_EEBUF[47] MCU : RG\_DECOUTBUF0[15] | Decoder0 output buffer | 0x00 | |
| 0x10430 | WR | [7:0] | RG\_EEBUF[48] MCU : RG\_DECOUTBUF1[0] | Decoder1 output buffer | 0x00 | |
| … | [7:0] | … | Decoder1 output buffer | 0x00 | |
| 1043F | [7:0] | RG\_EEBUF[63] MCU : RG\_DECOUTBUF1[15] | Decoder1 output buffer | 0x00 | |

### Register address map

| **ADDR(HEX)** | **WR** | **BIT** | **NAME/RANGE** | **DESCRIPTION** | **RESET  VALUE** |
| --- | --- | --- | --- | --- | --- |
| RG\_SOFT\_RESET | | | | |  |
| 0x10600 |  | [7:2] | RESERVED |  |  |
| WR | [1] | RG\_SWRESET\_EE | Reset EEPROM. Test register. To reset EEPROM, write 1 then Write 0 to this register bit. 0 : Normal 1 : RESET | 0x0 |
| WR | [0] | RG\_SWRESET | Reset symcipher hardware parts. To reset symchpher hardware parts, write 1 then write 0 to this register bit. 0 : Normal 1 : RESET | 0x0 |
| 0x10601 | A | - | RG\_ACCESS | Use this register to control DORCA-3. This register is access register. That is, it is not a register that writes and reads values. | - |
| 0x10602 | A | - | RG\_ACCESS2 | Use this register to control DORCA-3. This register is access register. That is, it is not a register that writes and reads values. Used to control ST0\_EEP\_OW\_CTRL function. | - |
| 0x10603 |  |  | RESERVED |  |  |
| RG\_ST0\_OPMODE Use this register to designate the DORCA-3 main control state. When you finish the control action on each function state, go to ST0\_STANDBY State and wait for the next control. The register values corresponding to the state for each function are shown below. ST0\_STANDBY is in standby mode and in standby mode. The other state is the state that DORCA-3 performs specific actions. When DORCA-3 ends a particular operation, E-MCU sets this register to ST0\_STANDBY state. | | | | |  |
| 0x10604 |  | [7:4] | RESERVED |  |  |
| WR | [3:0] | RG\_ST0\_OPMODE | To enter the desired main state, write the following corresponding values in this register : To end control and change to standby state, write a value of " 0x1 " to this reigister and write access to the RG\_ACCESS register. 4'h1 : Set main state(ST0) to ST0\_STANDBY state. 4'h5 : Set main state(ST0) to ST0\_CM0 state. 4'h6 : Set main state(ST0) to ST0\_STDSPI state. 4'h7 : Set main state(ST0) to ST0\_EE\_CFG state. 4'h8 : Set main state(ST0) to ST0\_RANDOM state. 4'h9 : Set main state(ST0) to ST0\_SYMCIP state. 4'hA : Set main state(ST0) to ST0\_OKA state. 4'hB : Set main state(ST0) to ST0\_MIDR state. 4'hC : Set main state(ST0) to ST0\_PERM\_GET state.  4'hF : Set main state(ST0) to ST0\_EEP\_OW\_CTRL state. The correct order of control for this register is as follows. (Correct use examples.) PWR\_ON(or SW RESETB)-> ST0\_STANDBY -> ST0\_STDSPI-> ST0\_STANDBY -> ST0\_CM0 -> ST0\_STANDBY -> ST0\_EE\_CFG -> ST0\_STANDBY (Incorrect use example.) PWR\_ON(or SW RESETB) -> ST0\_STANDBY -> ST0\_STDSPI-> ST0\_CM0 -> ST0\_EE\_CFG -> ST0\_STANDBY | 0x0 |
| RG\_ST1\_CM0\_OPMODE, RESERVED | | | | |  |
| 0x10605 |  |  | RESERVED |  |  |
| RG\_ST1\_STDSPI\_OPMODE User(E-MCU) can control ST1\_STDSPI state with this register. | | | | |  |
| 0x10606 |  | [7:3] | RESERVED |  |  |
|  | [2:0] | RG\_ST1\_STDSPI\_OPMODE | 3'h4 : Set ST1\_STDSPI state to ST1\_STDSPI\_SHA state. Other values : Not defined as a specific action. If user wants to ends ST1\_STDSPI\_SHA state and writes ' 1 ' to this register. | 0x0 |
| RG\_ST1\_EE\_CFG\_OPMODE, RESERVED | | | | |  |
| 0x10607 |  |  | RESERVED |  |  |
| RG\_ST1\_RND\_OPMODE User(E-MCU) can control ST1\_RND state with this register. | | | | |  |
|  |  | [7:3] | RESERVED |  |  |
| 0x10608 |  | [2:0] | RG\_ST1\_RND\_OPMODE | 3'h2 : Set ST1\_RND state to ST1\_RND\_GEN\_SPI0 state.  - E-MCU write '0x2' to this register to create a random value through SPI0 interface. At this state the E-MCU controls random generation function. 3'h4 : Set ST1\_RND state to ST1\_RND\_GEN\_SYMCIP state.  - E-MCU write '0x2' to this register, to make the symcipher creates a random value. At this state the hardware (symcipher) controls random generation function. | 0x0 |
| RG\_ST1\_SYMCIP\_OPMODE User(E-MCU) can control ST1\_SYMCIP state with this register E-MCU sets this register as ST1\_SYMCIP\_STANDBY state at the end of a specific operation. | | | | |  |
| 0x10609 |  | [7:4] | RESERVED |  |  |
|  | [3:0] | RG\_ST1\_SYMCIP\_OPMODE | 4'h1: Set ST1\_SYMCIP state to ST1\_SYMCIP\_STANDBY state. 4'h2: Set ST1\_SYMCIP state to ST1\_SYMCIP\_AESEncrypt state. 4'h3: Set ST1\_SYMCIP state to ST1\_SYMCIP\_AESDecrypt state. 4'h4: Set ST1\_SYMCIP state to ST1\_SYMCIP\_AESEncWrite state. 4'h5: Set ST1\_SYMCIP state to ST1\_SYMCIP\_AESEncRead state. 4'h6: Set ST1\_SYMCIP state to ST1\_SYMCIP\_AESKeyLoad state. 4'h7: Set ST1\_SYMCIP state to ST1\_SYMCIP\_RSCreate state. 4'h8: Set ST1\_SYMCIP state to ST1\_SYMCIP\_RSSHARead state. 4'h9: Set ST1\_SYMCIP state to ST1\_SYMCIP\_RSDirRead state. 4'hA: Set ST1\_SYMCIP state to ST1\_SYMCIP\_SHAAuth state. 4'hB: Set ST1\_SYMCIP state to ST1\_SYMCIP\_AESLock state. 4'hC Reserved 4'hD Reserved 4'hE: Set ST1\_SYMCIP state to ST1\_SYMCIP\_STOP0 state. 4'hF: Set ST1\_SYMCIP state to ST1\_SYMCIP\_STOP1 state. | 0x0 |
| RG\_ST1\_OKA\_OPMODE User(E-MCU) can control ST1\_OKA state with this register E-MCU sets this register as ST1\_OKA\_STANDBY state at the end of a specific operation. | | | | |  |
| 1060A |  | [7:3] | RESERVED |  |  |
|  | [2:0] | RG\_ST1\_OKA\_OPMODE | 3'h1 : Set ST1\_OKA state to ST1\_OKA\_STANDBY state. 3'h2 : Set ST1\_OKA state to ST1\_OKA\_OKA2\_KEY\_GEN state. 3'h3 : Set ST1\_OKA state to ST1\_OKA\_OKA2\_ED state. | 0x0 |
| RG\_ST1\_MIDR\_OPMODE This register controls MIDR counter backup procedure. | | | | |  |
| 1060B |  | [7:1] | RESERVED |  |  |
|  | [0] | RG\_ST1\_MIDR\_EEP\_RD\_START | 0 : CONFIG PAGE Read, Backup, RSFLAG SET finish 1 : CONFIG PAGE Read, Backup, RSFLAG SET start | 0x0 |
| RG\_ST1\_PERM\_GET\_OPMODE, RESERVED | | | | |  |
| 1060C |  |  | RESERVED |  |  |
|  | | | | |  |
| 1060D |  |  | RESERVED |  |  |
|  | | | | |  |
| 1060E |  |  | RESERVED |  |  |
| RG\_ST1\_EEP\_OW\_CTRL\_OPMODE User(E-MCU) can control ST1\_EEP\_OW\_CTRL state with this register E-MCU sets this register as ST1\_EEP\_OW\_CTRL\_STANDBY state at the end of a specific operation. | | | | |  |
| 1060F |  | [7:3] | RESERVED |  |  |
| WR | [2:0] | RG\_EEP\_OW\_CTRL\_OPMODE | 3'h0 : RESERVED 3'h1 : Set ST1\_EEP\_OW\_CTRL state to ST1\_EEP\_OW\_CTRL\_STANDBY state. 3'h2 : Set ST1\_EEP\_OW\_CTRL state to ST1\_EEP\_OW\_CTRL\_DETOUR state. 3'h3 : Set ST1\_EEP\_OW\_CTRL state to ST1\_EEP\_OW\_CTRL\_DESTROY0 state. 3'h4 : Set ST1\_EEP\_OW\_CTRL state to ST1\_EEP\_OW\_CTRL\_DESTROY1 state. 3'h5 ~ 3'h7 : RESERVED | 0x0 |
| 0x10610 |  |  | RESERVED |  |  |
| …. |  |  |  |  |  |
| 0x10618 |  |  | RESERVED |  |  |
| RG\_ST2\_SYMCIP\_OPMODE User(E-MCU) can control ST2\_SYMCIP state with this register. When the E-MCU write registry values, hardware performs control actions corresponding to the values. When control operation is completed, E-MCU sets the register to ST2\_AES\_STANDBY state. E-MCU sets this register as ST2\_AES\_STANDBY state at the end of a specific operation. | | | | |  |
| 0x10619 |  | [7:4] | RESERVED |  |  |
| WR | [3:0] | RG\_ST2\_SYMCIP\_OPMODE\_AES | 3'h1 : Set ST2\_SYMCIP\_OPMODE\_AES state to ST2\_SYMCIP\_OPMODE\_AES\_STANDBY state. 3'h2 : Set ST2\_SYMCIP\_OPMODE\_AES state to ST2\_SYMCIP\_OPMODE\_AES\_INITTIC state. 3'h3 : Set ST2\_SYMCIP\_OPMODE\_AES state to ST2\_SYMCIP\_OPMODE\_AES\_KEYTIC state. 3'h4 : Set ST2\_SYMCIP\_OPMODE\_AES state to ST2\_SYMCIP\_OPMODE\_AES\_RUNREADY state.  ... 3'h8 : Set ST2\_SYMCIP\_OPMODE\_AES state to ST2\_SYMCIP\_OPMODE\_RSCREATE state. 3'h9 : Set ST2\_SYMCIP\_OPMODE\_AES state to ST2\_SYMCIP\_OPMODE\_AES\_KEYLOAD state. 3'hA ~3'hD : This state is used by a hardware control part. 3'hE : Set ST2\_SYMCIP\_OPMODE\_AES state to ST2\_SYMCIP\_OPMODE\_AES\_DEC\_WR state.   At this state a input cipher text is decrypted and save to the EEPROM. 3'hF : Set ST2\_SYMCIP\_OPMODE\_AES state to ST2\_SYMCIP\_OPMODE\_AES\_ENC\_RD state.   At this state a plain text from EEPROM is encrypted and E-MCU reads the encrypted data. | 0x0 |
| RG\_EE\_USER\_ZONE\_SEL The EEPROM has 15 user zones. One user zone consists of 4 pages.  One page consists of 4 sub pages. One page size is 64 Bytes(512-bit). One sub page consists of 16 Bytes(128-bit). The user can write or read in sub-page or page units. | | | | |  |
| 1061A |  | [7:6] | RG\_EE\_UZ\_SUBFRAMENUM | 2'h0 : [127:0] of selected page 2'h1 : [255:128] of selected page 2'h2 : [383:256] of selected page 2'h3 : [511:384] of selected page | 0x0 |
|  | [5:4] | RG\_EE\_UZ\_SUBPAGENUM | 2'h0 : subpage 0 (0x00 ~ 0x3F) 2'h1 : subpage 1 (0x40 ~ 0x7F) 2'h2 : subpage 2 (0x80 ~ 0xBF) 2'h3 : subpage 3 (0xC0 ~ 0xFF) | 0x0 |
| WR | [3:0] | RG\_EE\_UZ\_PAGENUM | 4'h0 : RESERVED 4'h1 : EE\_USER\_ZONE\_M01 (0xF100 ~ 0xF1FF) … 4'hF : EE\_USER\_ZONE\_M15 (0xFF00 ~ 0xFFFF) | 0x0 |
| 0x1061B |  |  | RESERVED |  |  |
| RG\_EE\_CFG\_RD\_RG\_EEBUF\_ST This is a access register. If the user write any data at this register, One page EEPROM data is read and save to RG\_EEBUF. Before wirte a data to the EEPROM, The user controls this register first. | | | | |  |
| 0x1061C | A | - | RG\_EE\_CFG\_RD\_RG\_EEBUF\_ST | Access control is writing '0x0' to this register. | - |
| RG\_ST3\_SYMCIP\_RSCREATE\_OPMODE This register is used to create the root serial(RSCreate operation). User(E-MCU) can control ST3\_SYMCIP\_RSCREATE\_OPMODE state with this register. When the E-MCU write registry values to this register and access RG\_ACCESS register, hardware performs control actions corresponding to the values.  When control operation is completed, E-MCU sets the register to ST3\_SYMCIP\_RSCREATE\_STANDBY state and access RG\_ACCESS register. E-MCU sets this register as ST3\_SYMCIP\_RSCREATE\_STANDBY state at the end of a specific operation. | | | | |  |
| 0x1061D |  | [7:3] | RESERVED |  |  |
| WR | [2:0] | RG\_ST3\_SYMCIP\_RSCREATE\_OPMODE | 3'h1 : Set ST3\_SYMCIP\_RSCREATE\_OPMODE state to ST3\_SYMCIP\_RSCREATE\_STANDBY state. 3'h2 : Set ST3\_SYMCIP\_RSCREATE\_OPMODE state to ST3\_SYMCIP\_RSCREATE\_ENC1 state. 3'h3 : Set ST3\_SYMCIP\_RSCREATE\_OPMODE state to ST3\_SYMCIP\_RSCREATE\_ENC2 state. 3'h4 : Set ST3\_SYMCIP\_RSCREATE\_OPMODE state to ST3\_SYMCIP\_RSCREAETE\_WR\_EEP state.  3'h7 : Set ST3\_SYMCIP\_RSCREATE\_OPMODE state to ST3\_SYMCIP\_RSCREATE\_WR\_EEBUF state. | 0x0 |
| 1061E |  |  | RESERVED |  |  |
| RG\_ST3\_SYMCIP\_KEYLOAD\_OPMODE This register is used to create the root serial(AESKeyLoad operation). User(E-MCU) can control RG\_ST3\_SYMCIP\_KEYLOAD\_OPMODE state with this register. When the E-MCU write registry values to this register and access RG\_ACCESS register, hardware performs control actions corresponding to the values.  When control operation is completed, E-MCU sets the register to ST3\_SYMCIP\_KEYLOAD\_STANDBY state and access RG\_ACCESS register. E-MCU sets this register as ST3\_SYMCIP\_KEYLOAD\_STANDBY state at the end of a specific operation. | | | | |  |
| 1061F |  | [7:3] | RESERVED |  |  |
| WR | [2:0] | RG\_ST3\_SYMCIP\_KEYLOAD\_OPMODE | 3'h1 : Set ST3\_SYMCIP\_KEYLOAD\_OPMODE state to ST3\_SYMCIP\_KEYLOAD\_STANDBY state. 3'h2 : ST3\_SYMCIP\_KEYLOAD\_DEC1 state. 3'h3 : ST3\_SYMCIP\_KEYLOAD\_DEC2 state. 3'h4 : ST3\_SYMCIP\_KEYLOAD\_WR\_EEP state. | 0x0 |
| RG\_EE\_KEY\_AES\_CTRL Provides the location of EEPROM storage for the keys used for AESEncrypt, AESDecrypt, AESEncRead and AESEncwrite operations. It also tells the location of the EEPROM to store the keys that were created when performing AESKeyLoad operations. Keys from AESKeyLoad operations are used for AESEncrypt, AESDecrypt, AESEncRead, and AESEncwrite operations. | | | | |  |
| 0x10620 |  | [7:2] | RESERVED |  |  |
| WR | [1:0] | RG\_EE\_KEY\_AES\_xN | 2'h0 : EE\_KEY\_AES\_x0 2'h1 : EE\_KEY\_AES\_x1 2'h2 : EE\_KEY\_AES\_x2 2'h3 : EE\_KEY\_AES\_x3 | 0x0 |
| RG\_UZID | | | | |  |
| 0x10621 |  |  | RESERVED |  |  |
| RG\_KL\_CTRL Use this register for AESKeyLoad operations. | | | | |  |
| 0x10622 |  | [7:5] | RESERVED |  |  |
| WR | [4] | RG\_KL\_KeySaveSel | This register selects between the key made with AES decryptor and the value entered with AES text input. 0 : Select a value made with AES decryption. 1 : Select a value that enters the AES text input. | 0x0 |
| WR | [3:2] | RG\_KL\_TextSel | The register that selects the text message into the AES decryptor input. 2'h0 : Select ciphertext that E-MCU enters as AES text input. 2'h1 : Use the key value EE\_KEY\_ASYMCIP\_x0 as the AES text value that you created as an ECDH result. 2'h2 : Use the full key value made with OKA as the AES text entry. | 0x0 |
| WR | [1:0] | RG\_KL\_KeySel | A registry that selects the key message that enters the AES decryptor input. 2'h0 : Use the EE\_key\_SEEDs stored in the EEPROM with the AES key input. 2'h1 : Use the EE\_key\_AES\_x0 stored in the EEPROM with the AES key input. 2'h2 : Use the EE\_key\_AES\_x1 stored in the EEPROM with the AES key input. 2'h3 : Use the EE\_key\_AES\_x2 stored in the EEPROM with the AES key input. | 0x0 |
| RG\_RSCREATE\_CTRL Use this register for RSCreate, RSSHARead, and RSDirRead operations. | | | | |  |
| 0x10623 |  | [7] | RESERVED |  |  |
|  | [6] | DirReadAES\_KEY\_x3 | This register is used to read EE\_AES\_KEY\_x3. You can read EE\_AES\_KEY\_x3 with UID\_PWM permission. If the value of RG\_EE\_RS\_xN is " 1 " (RS\_x1), or " 3 " (RS\_x3), then EE\_AES\_KEY\_x3 is not readable. That is, if the RG\_EE\_RS\_x1 value is " 1 " and DirReadAES\_key\_x3 (AES\_KEY\_x3) value is " 1 ", the RS\_x1 value can be read. | 0x0 |
| WR | [5:4] | RG\_EE\_RS\_xN | Used for RSCreate, RSSHARead, and RSDirRead operations. In RSCreate mode, you specify the keys to generate. In RSSHARead, RSDirRead mode, specify the key to be read. | 0x0 |
|  | [3] | RESERVED |  |  |
| WR | [2] | RG\_RSC\_KeySaveSel | Used for RSCreate, RSSHARead operations. | 0x0 |
| WR | [1] | RG\_RSC\_GEN\_RND1 | Used for RSCreate operations. When creating RND1 (using the AE256 key as [255:128]), This register must be set to ' 1 ' before the RND\_GEN command and clear to ' 0 ' after creation. | 0x0 |
| WR | [0] | RG\_RSC\_GEN\_RND0 | Used for RSCreate operations. When creating RND0 (using the AE256 key as [127:0]), This register must be set to ' 1 ' before the RND\_GEN command and clear to ' 0 ' after creation. | 0x0 |
| RG\_SHAAUTH\_CTRL Used for authentication(SHAAuth) operations. Authentication can be made in two directions.  For the first method, DORCA-3 performs authentication. If E-MCU gives the certification message to DORCA-3, DORCA-3 performs the authentication using the authentication message. For the second method, DORCA-3 performs authentication. If DORCA-3 gives the certification message to E-MCU, E-MCU performs the authentication using the authentication message. Both methods are necessary for full certification. | | | | |  |
| 0x10624 |  | [7:2] | RESERVED |  |  |
| WR | [1] | rST2\_SYMCIP\_SHAAuth\_STAY\_DP | 0 : None 1 : Write " 1 " to complete the SHA-authentication operation. | 0x0 |
| WR | [0] | RG\_SHAAuthQuest\_SYMCIP\_EMCU | 0 : E-MCU asks the question. E-MCU creates authentication messages(AuthMsgMCU[255:0] and AuthText[127:0]) and send it to DORCA-3.  1 : DORCA-3 asks the questions. DORCA-3 creates authentication messages(AuthMsgDevice[255:0] and AuthRND[127:0]) and send it to E-MCU. | 0x0 |
| 0x10625 |  |  | RESERVED |  |  |
| RG\_PERM\_GET\_CTRL This register writes the information that DORCA-3 require to control ST0\_PERM\_GET state. | | | | |  |
| 0x10626 |  | [7:3] | RESERVED |  |  |
| WR | [2:0] | RG\_EE\_PW\_ADDR | The register that tells the EEPROM where the password is stored.  3'h5 : EE\_SUPER\_PW 3'h4 : EE\_DETOUR\_PW 3'h3 : EE\_DESTROY0\_PW 3'h2 : EE\_DESTROY1\_PW 3'h1 : EE\_EEPROM\_PW 3'h0 : EE\_UID\_PW | 0x1 |
| RG\_PERM\_GET\_CTRL1 This register tells each password permission acquisition state. | | | | |  |
| 0x10627 |  | [7:6] | RESERVED |  |  |
| R | [5] | RG\_PERM\_SUPER\_PASS | 0 : Failed to acquire SUPER\_PASS password permission(authorization). 1 : Succeeded to acquire SUPER\_PASS password permission(authorization). |  |
| R | [4] | RG\_PERM\_DETOUR\_PASS | 0 : Failed to acquire DETOUR\_PASS password permission(authorization). 1 : Succeeded to acquire DETOUR\_PASS password permission(authorization). |  |
| R | [3] | RG\_PERM\_DESTROY0\_PASS | 0 : Failed to acquire DESTROY0\_PASS password permission(authorization). 1 : Succeeded to acquire DESTROY0\_PASS password permission(authorization). |  |
| R | [2] | RG\_PERM\_DESTROY1\_PASS | 0 : Failed to acquire DESTROY1\_PASS password permission(authorization). 1 : Succeeded to acquire DESTROY1\_PASS password permission(authorization). |  |
| R | [1] | RG\_PERM\_EEPROM\_PASS | 0 : Failed to acquire EEPROM\_PASS password permission(authorization). 1 : Succeeded to acquire EEPROM\_PASS password permission(authorization). |  |
| R | [0] | RG\_PERM\_UID\_PASS | 0 : Failed to acquire UID\_PASS password permission(authorization). 1 : Succeeded to acquire UID\_PASS password permission(authorization). |  |
| RG\_PERM\_RELEASE If E-MCU writes ' 0 ' to this register in ST0\_PERM\_GET state, the DORCA-3 returns all acquired password permissions. | | | | |  |
| 0x10628 | A | - | RG\_PERM\_RELEASE |  |  |
| RG\_PERM\_GET\_EE\_RD\_PRE\_SP When E-MCU writes ' 0 ' to this register in ST0\_PERM\_GET state, the DORCA-3 starts the process of obtaining the password permission. In other words, DORCA-3 reads and backs up the corresponding EEPROM configuration area and waits for E-MCU to write the PW\_CT. | | | | |  |
| 0x10629 | A | - | RG\_PERM\_GET\_EE\_RD\_PRE\_SP |  |  |
| 0x1062A |  |  | RESERVED |  |  |
| …. |  |  | RESERVED |  |  |
| 0x10634 |  |  | RESERVED |  |  |
| RG\_AES\_CTRL This register controls the AES and ARIA operations. The operations include encryption and decryption. | | | | |  |
| 0x10635 |  | [7] | RESERVED |  | - |
| WR | [6:4] | RG\_AES\_OPMODE | Register for the selection of five modes of operation. 3'h0 : ECB 3'h1 : CBC 3'h2 : OFB 3'h3 : CTR 3'h4 : CFB | 0x0 |
| WR | [3] | RG\_AES\_2\_1\_FRAME | This register selects frame length. This register selects one or two frame encryption(decryption) processing in ST1\_SYMCIP\_AESEncrypt state of ST0\_SYMCIP state. Once set in two frame mode, the symcipher performs encryption or decryption after E-MCU writes 2 frames. The first frame performance result is saved to RG\_EEBUF[383:256] and the second frame performance result is saved to RG\_EEBUF [511:384].  1 : Two frame mode 0 : One frame mode | 0x0 |
| WR | [2] | RG\_BYPASS | 1(BYPASS), 0(Normal) In BYPASS mode, the LSB bit value is changed for each byte of the input text.  Example) INPUT TEXT : 0xC7 ….. 5D OUTPUT TEXT: 0xC6 ….. 5C | 0x0 |
| WR | [1] | RG\_128\_256 | 1(128), 0(256) | 0x0 |
| WR | [0] | RG\_AES\_ARIA | 1(AES), 0(ARIA) | 0x0 |
| 0x10636 |  |  | RESERVED |  |  |
| 0x10637 |  |  | RESERVED |  |  |
| RG\_SHA\_CTRL :  1. Check register contorl oder at TV0610001 1.1. RG\_ST0\_OPMODE -> RG\_ST1\_STDSPI\_OPMODE -> RG\_SHA\_CTRL 1.2. RG\_SHA\_CTRL -> RG\_ST0\_OPMODE -> RG\_ST1\_STDSPI\_OPMODE | | | | |  |
| 0x10638 |  | [7:2] | RESERVED |  |  |
| WR | [1] | RG\_SHA\_ONLY\_FRM\_SEL | 1 : SHA only multi frame. 0 : SHA only single frame. | 0x0 |
| WR | [0] | RG\_SHA\_MF\_STOP | 1 : SHA multi frame stop. 0 : normal | 0x0 |
| 0x10639 |  |  | RESERVED |  |  |
| 1063A |  |  | RESERVED |  |  |
| 1063B |  |  | RESERVED |  |  |
| RG\_OKA\_CTRL This register controls AES in ST0\_OKA state. | | | | |  |
| 1063C |  | [7:2] | RESERVED |  |  |
| WR | [1] | RG\_OKA\_10\_11N | The following function can be controlled only when the value of EE\_CONFIG\_NW:EE\_CONFIG\_NW\_CTRL0:EE\_OKA\_10\_11N is '0'. 1 : OKA 1:0 communication 0 : OKA 1:1 or 1:N communication (default) | 0x0 |
| WR | [0] | RG\_OKA\_2\_1\_FRAME | 1. Precautions for running OKA in two frame mode.  (1) DORCA-3 conduct a key initialization in the wait time for the first frame input.  (2) Therefor, in two frame mode, E-MCU should encrypt(or decrypt) even number of frames.   (3) If E-MCU finishes encryption(or decryption) in odd number frames, then following enryption(or decryption), E-MCU must begin with Key generation operation.  1 : 2 frame mode 0 : 1 frame mode | 0x0 |
| RG\_AES\_TVALUE7 This register sets AES twist value. | | | | |  |
| 1063D | WR | [7:0] | RG\_AES\_TVALUE7 | This register only works in AES. 0x00 : Standard AES Mode 0xXX : Twist AES Mode | 0x0 |
| RG\_AES\_TVALUE8 This register sets AES twist value. | | | | |  |
| 1063E |  | [7:4] | RESERVED |  |  |
| WR | [3:0] | RG\_AES\_TVALUE8 | This register only works in AES. 0x0 : Standard AES Mode 0xX : Twist AES Mode | 0x0 |
| 1063F |  |  | RESERVED |  |  |
| …. |  |  | RESERVED |  |  |
| 1064F |  |  | RESERVED |  |  |
| RG\_SLEEP\_TIMER[12:0] RESET VALUE : 0x1FFF Set this register value to 0 after power on. | | | | |  |
| 0x10650 | WR | [4:0] | RG\_SLEEP\_TIMER\_MSB |  | 0x1F |
| 0x10651 | WR | [7:0] | RG\_SLEEP\_TIMER\_LSB |  | 0xFF |
| 0x10652 |  |  | RESERVED |  |  |
| …. |  |  | RESERVED |  |  |
| 0x1065F |  |  | RESERVED |  |  |
| …. |  |  | RESERVED |  |  |
| …. |  |  | RESERVED |  |  |
| 106AF |  |  | RESERVED |  |  |
|  | | | | |  |
| 106B5 | WR | [7:2] | RESERVED |  |  |
| [2:0] | RESERVED |  |  |
| 106B6 |  |  | RESERVED |  |  |
| …. |  |  |  |  |  |
|  | | | | |  |
| 106C0 |  |  | RESERVED |  |  |
| …. |  |  |  |  |  |
| 106DF |  |  | RESERVED |  |  |
| …. |  |  |  |  |  |
| 106EF |  |  | RESERVED |  |  |
| 106F0 |  | [7:1] | RESERVED |  |  |
| 106F1 |  |  | RESERVED |  |  |
| RG\_RNDGEN\_USER | | | | |  |
| 10700 |  | [7:1] | RESERVED |  |  |
| WR | [0] | RG\_RNDGEN\_USER | 1 : RNDGEN user mode  User can enter random values that the user specifies. 0 : RNDGEN normal mode  The internal random generator produces a random. | 0x0 |
| RG\_RNDGEN\_EEBUF\_CLR This register can clear generated random value in RNDGEN user mode. To clear a random value, write '1', then '0' to this register. | | | | |  |
| 10701 |  | [7:1] | RESERVED |  |  |
| WR | [0] | RG\_RNDGEN\_EEBUF\_CLR | 1 : RG EEBUF Clear 0 : | 0x0 |
| RG\_MCUAuthResult | | | | |  |
| 10720 |  | [7:1] | RESERVED |  |  |
| R | [0] | RG\_MCUAuthResult | 1 : Auth Pass 0 : Auth Fail | 0x0 |

## CORTEX-M0 registers

### SSP(SPI1) Features

• Compliance to the AMBA Specification (Rev 2.0) for easy integration into SoC implementation.

• Master or slave operation.

• Programmable clock bit rate and prescale.

• Separate transmit and receive first-in, first-out memory buffers, 16 bits wide, 8 locations deep.

• Programmable choice of interface operation, SPI, Microwire, or TI synchronous serial.

• Programmable data frame size from 4 to 16 bits.

• Independent masking of transmit FIFO, receive FIFO, and receive overrun interrupts.

• Internal loopback test mode available.

### SSP(SPI1) Operation

Following reset, the PrimeCell SSP logic is disabled and must be configured when in this state.

Control registers SSPCR0 and SSPCR1 need to be programmed to configure the peripheral as a master or slave operating under one of the following protocols:

• Motorola SPI

• Texas Instruments SSI

• National Semiconductor.

The bit rate, derived from the external SSPCLK, requires the programming of the clock prescale register SSPCPSR.

You can either prime the transmit FIFO, by writing up to eight 16-bit values when the PrimeCell SSP is disabled, or allow the transmit FIFO service request to interrupt the CPU. Once enabled, transmission or reception of data begins on the transmit (SSPTXD) and receive (SSPRXD) pins.

### SSP(SPI1) registers

SSP Base Address : 0x4000\_2200

SSP Register Address : SSP Base Address + Offset

Table 6‑1 Summary of SSP Registers

| ADDR(HEX)  Offset | Type | Width | NAME | DESCRIPTION | RESET |
| --- | --- | --- | --- | --- | --- |
| 0x00 | WR | 16 | SSPCR0 | Control register 0. | 0x0 |
| 0x04 | WR | 4 | SSPCR1 | Control register 1. | 0x0 |
| 0x08 | WR | 16 | SSPDR | Receive FIFO(read) and transmit FIFO data register(write). | 0x--- |
| 0x0C | R | 5 | SSPSR | Status register. | 0x03 |
| 0x10 | WR | 8 | SSPCPSR | Clock prescale register. | 0x0 |
| 0x14 | WR | 4 | SSPIMSC | Interrupt mask set and clear register. | 0x0 |
| 0x18 | R | 4 | SSPRIS | Raw interrupt status register. | 0x8 |
| 0x1C | R | 4 | SSPMIS | Masked interrupt status register. | 0x0 |
| 0x20 | W | 4 | SSPICR | Interrupt clear register. | 0x0 |
| 0x24 | WR | 2 | SSPDMACR | DMA control register. | 0x0 |

Table 6‑2 SSP Registers Details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ADDR(HEX)**  **Offset** | **WR** | **BIT** | **NAME/RANGE** | **DESCRIPTION** | **RESET  VALUE** |
| Control register 0 (SSPCR0)  SSPCR0 is control register 0 and contains five bit fields that control various functions within the SSP. | | | | | |
| 0x0 | WR | [15:8] | SCR | Serial clock rate. The value SCR is used to generate the transmit and receive bit rate of the SSP. The bit rate is:  where CPSDVSR is an even value from 2-254, programmed  through the SSPCPSR register and SCR is a value from 0-255. | 0x0 |
| WR | [7] | SPH | **SSPCLKOUT** phase, applicable to Motorola SPI frame format only. | 0x0 |
| WR | [6] | SPO | **SSPCLKOUT** polarity, applicable to Motorola SPI frame  format only. | 0x0 |
| WR | [5:4] | FRF | Frame format  00 Motorola SPI frame format.  01 TI synchronous serial frame format.  10 National Microwire frame format.  11 Reserved, undefined operation. | 0x0 |
| WR | [3:0] | DSS | Data Size Select:  0000 Reserved, undefined operation.  0001 Reserved, undefined operation.  0010 Reserved, undefined operation.  0011 4-bit data.  0100 5-bit data.  0101 6-bit data.  0110 7-bit data.  0111 8-bit data.  1000 9-bit data.  1001 10-bit data.  1010 11-bit data.  1011 12-bit data.  1100 13-bit data.  1101 14-bit data.  1110 15-bit data.  1111 16-bit data. | 0x0 |
| Control register 1 (SSPCR1)  SSPCR1 is the control register 1 and contains four different bit fields, that control various functions within the SSP. | | | | | |
| 0x4 | - | [15:4] | - | Reserved, read unpredictable, should be written as 0. | - |
| WR | [3] | SOD | Slave-mode output disable. This bit is relevant only in the slave mode, MS=1. In multiple-slave systems, it is possible for an SSP master to broadcast a message to all slaves in the system while ensuring that only one slave drives data onto its serial output line. In such systems the **RXD** lines from multiple slaves could be tied together.  To operate in such systems, the SOD bit can be set if the SSP slave is not supposed to drive the **SSPTXD** line:  0 SSP can drive the **SSPTXD** output in slave mode.  1 SSP must not drive the **SSPTXD** output in slave mode. | 0x0 |
| WR | [2] | MS | Master or slave mode select.  This bit can be modified only when the SSP is disabled, SSE=0:  0 Device configured as master, default.  1 Device configured as slave. | 0x0 |
| WR | [1] | SSE | Synchronous serial port enable:  0 SSP operation disabled.  1 SSP operation enabled. | 0x0 |
| WR | [0] | LBM | Loop back mode:  0 Normal serial port operation enabled.  1 Output of transmit serial shifter is connected to input of receive serial shifter internally. | 0x0 |
| Data register (SSPDR)  SSPDR is the data register and is 16-bits wide. When SSPDR is read, the entry in the receive FIFO, pointed to by the current FIFO read pointer, is accessed. As data values are removed by the SSP receive logic from the incoming data frame, they are placed into the entry in the receive FIFO, pointed to by the current FIFO write pointer. When SSPDR is written to, the entry in the transmit FIFO, pointed to by the write pointer, is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. It is loaded into the transmit serial shifter, then serially shifted out onto the **SSPTXD** pin at the programmed bit rate. When a data size of less than 16 bits is selected, the user must right-justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right-justified in the receive buffer. | | | | | |
| 0x8 | WR | [15:0] | DATA | Transmit/Receive FIFO:  **Read** Receive FIFO.  **Write** Transmit FIFO.  You must right-justify data when the SSP is programmed for a data size that is less than 16bits. Unused bits at the top are ignored by transmit logic. The receive logic automatically right-justifies. | - |
| Status register (SSPSR)  SSPSR is a RO status register that contains bits that indicate the FIFO fill status and the SSP busy status. | | | | | |
| 0x10 | - | [15:5] | - | Reserved, read unpredictable, should be written as 0. | - |
| R | [4] | BSY | SSP busy flag (read only):  0 SSP is idle.  1 SSP is currently transmitting and/or receiving a frame or the transmit FIFO is not empty | 0x0 |
| R | [3] | RFF | Receive FIFO full (read only):  0 Receive FIFO is not full.  1 Receive FIFO is full. | 0x0 |
| R | [2] | RNE | Receive FIFO not empty, (read only):  0 Receive FIFO is empty.  1 Receive FIFO is not empty. | 0x0 |
| R | [1] | TNF | Transmit FIFO not full, (read only):  0 Transmit FIFO is full.  1 Transmit FIFO is not full. | 0x1 |
| R | [0] | TFE | Transmit FIFO empty, (read only):  0 Transmit FIFO is not empty.  1 Transmit FIFO is empty. | 0x1 |
| Clock prescale register (SSPCPSR)  SSPCPSR is the clock prescale register and specifies the division factor by which the input **SSPCLK** must be internally divided before further use. The value programmed into this register must be an even number between  2-254. The least significant bit of the programmed number is hard-coded to zero. If an odd number is written to this register, data read back from this register has the least significant bit as zero. | | | | | |
| 0x14 | - | [15:8] | - | Reserved, read unpredictable, must be written as 0. | - |
| WR | [7:0] | CPSDVSR | Clock prescale divisor. Must be an even number from 2-254, depending on the frequency of **SSPCLK**. The least significant bit always returns zero on reads. | 0x0 |
| Interrupt mask set or clear register (SSPIMSC)  The SSPIMSC register is the interrupt mask set or clear register. It is a RW register. On a read this register gives the current value of the mask on the relevant interrupt. A write of 1 to the particular bit sets the mask, enabling the  interrupt to be read. A write of 0 clears the corresponding mask. All the bits are cleared to 0 when reset. | | | | | |
| 0x18 |  | [15:4] | - | Reserved, read as zero, do not modify. | - |
| WR | [3] | TXIM | Transmit FIFO interrupt mask:  0 Transmit FIFO half empty or less condition interrupt is masked.  1 Transmit FIFO half empty or less condition interrupt is not masked. | 0x0 |
| WR | [2] | RXIM | Receive FIFO interrupt mask:  0 Receive FIFO half full or less condition interrupt is masked.  1 Receive FIFO half full or less condition interrupt is not masked. | 0x0 |
| WR | [1] | RTIM | Receive timeout interrupt mask:  0 Receive FIFO not empty and no read prior to timeout period interrupt is masked.  1 Receive FIFO not empty and no read prior to timeout period interrupt is not masked. | 0x0 |
| WR | [0] | RORIM | Receive overrun interrupt mask:  0 Receive FIFO written to while full condition interrupt is masked.  1 Receive FIFO written to while full condition interrupt is not masked. | 0x0 |
| Raw interrupt status register (SSPRIS)  The SSPRIS register is the raw interrupt status register. It is a RO register. On a read this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect. | | | | | |
| 0x1C | - | [15:4] | - | Reserved, read as zero, do not modify | - |
| R | [3] | TXRIS | Gives the raw interrupt state, prior to masking, of the SSPTXINTR interrupt | 0x1 |
| R | [2] | RXRIS | Gives the raw interrupt state, prior to masking, of the SSPRXINTR interrupt | 0x0 |
| R | [1] | RTRIS | Gives the raw interrupt state, prior to masking, of the SSPRTINTR interrupt | 0x0 |
| R | [0] | RORRIS | Gives the raw interrupt state, prior to masking, of the SSPRORINTR interrupt | 0x0 |
| Masked interrupt status register (SSPMIS)  The SSPMIS register is the masked interrupt status register. It is a RO register. On a read this register gives the current masked status value of the corresponding interrupt. A write has no effect. | | | | | |
| 0x20 | - | [15:4] | - | Reserved, read as zero, do not modify | - |
| R | [3] | TXMIS | Gives the transmit FIFO masked interrupt state, after masking, of the SSPTXINTR interrupt | 0x0 |
| R | [2] | RXMIS | Gives the receive FIFO masked interrupt state, after masking, of the SSPRXINTR interrupt | 0x0 |
| R | [1] | RTMIS | Gives the receive timeout masked interrupt state, after masking, of the SSPRTINTR interrupt | 0x0 |
| R | [0] | RORMIS | Gives the receive over run masked interrupt status, after masking, of the SSPRORINTR interrupt | 0x0 |
| Interrupt clear register (SSPICR)  The SSPICR register is the interrupt clear register and is write-only. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect. | | | | | |
| 0x24 | - | [15:2] | - | Reserved, read as zero, do not modify | - |
| W | [1] | RTIC | Clears the SSPRTINTR interrupt | 0x0 |
| W | [0] | RORIC | Clears the SSPRORINTR interrupt | 0x0 |
| DMA control register (SSPDMACR)  The SSPDMACR register is the DMA control register. It is a RW register. All the bits are cleared to 0 on reset. | | | | | |
| 0x28 | - | [15:2] | - | Reserved, read as zero, do not modify | - |
| WR | [1] | TXDMAE | Transmit DMA Enable. If this bit is set to 1, DMA for the transmit FIFO is enabled | 0x0 |
| WR | [0] | RXDMAE | Receive DMA Enable. If this bit is set to 1, DMA for the receive FIFO is enabled. | 0x0 |

# EEPROM Configuration

# Revision History

| Version | Revision | Date | Comments | Editors |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 003 |  | ~ | Editing. | HCLEE |
| 002 |  | 2018.01.16 | Release to Dream Security wo SPI0 register. | HCLEE |
| 001 | - | 2017.12.07 | Document creation. | HCLEE |