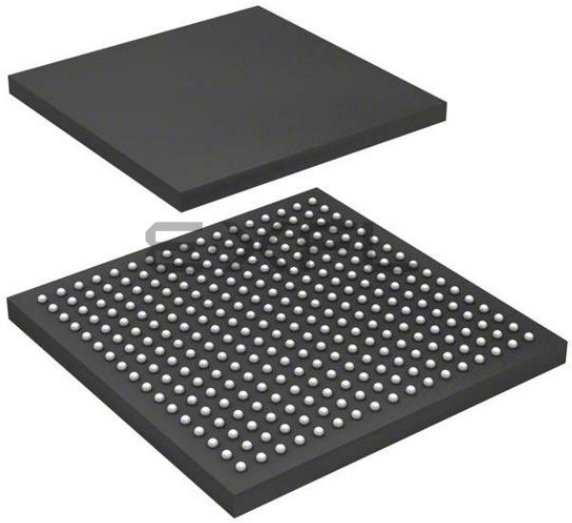


**Welcome to E-XFL.COM**

## Understanding Embedded - Microprocessors

Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

## Applications of Embedded - Microprocessors

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

| Details | |
|---|---|
| Product Status | Active |
| Core Processor | ARM® Cortex®-A5 |
| Number of Cores/Bus Width | 1 Core, 32-Bit |
| Speed | 500MHz |
| Co-Processors/DSP | Multimedia; NEON™ MPE |
| RAM Controllers | LPDDR1, LPDDR2, LPDDR3, DDR2, DDR3, DDR3L, QSPI |
| Graphics Acceleration | Yes |
| Display & Interface Controllers | Keyboard, LCD, Touchscreen |
| Ethernet | 10/100Mbps (1) |
| SATA | - |
| USB | USB 2.0 + HSIC |
| Voltage - I/O | 3.3V |
| Operating Temperature | -40°C ~ 105°C (TA) |
| Security Features | ARM TZ, Boot Security, Cryptography, RTIC, Secure Fusebox, Secure JTAG, Secure Memory, Secure RTC |
| Package / Case | 289-LFBGA |
| Supplier Device Package | 289-LFBGA (14x14) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atsama5d27c-cnr |

### 14.5.19 L2CC Clean Physical Address Line Register

**Name:** L2CC_CPALR

**Address:** 0x00A007B0

**Access:** Read/Write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| TAG | | | | | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| TAG | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| TAG | | IDX | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IDX | | | – | – | – | – | C |

**C: Cache Synchronization Status**

0: No background operation is in progress. When written, must be zero.

1: A background operation is in progress.

**IDX: Index number**

**TAG: Tag number**

### 18.13.15 Security Peripheral Select x Registers

**Name:** MATRIX_SPSELRx [x=1..3]

**Address:** 0xF00182C0 (0), 0xFC03C2C0 (1)

**Access:** Read/Write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| NSECP31 | NSECP30 | NSECP29 | NSECP28 | NSECP27 | NSECP26 | NSECP25 | NSECP24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| NSECP23 | NSECP22 | NSECP21 | NSECP20 | NSECP19 | NSECP18 | NSECP17 | NSECP16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| NSECP15 | NSECP14 | NSECP13 | NSECP12 | NSECP11 | NSECP10 | NSECP9 | NSECP8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| NSECP7 | NSECP6 | NSECP5 | NSECP4 | NSECP3 | NSECP2 | NSECP1 | NSECP0 |

This register can only be written if the WPEN bit is cleared in the Write Protection Mode Register.

Each MATRIX_SPSELR can configure the access security type for up to 32 peripherals:

– MATRIX_SPSELR1 configures the access security type for peripheral identifiers 0–31 (bits NSECP0–NSECP31).

– MATRIX_SPSELR2 configures the access security type for peripheral identifiers 32–63 (bits NSECP0–NSECP31).

– MATRIX_SPSELR3 configures the access security type for peripheral identifiers 64–95 (bits NSECP0–NSECP31).

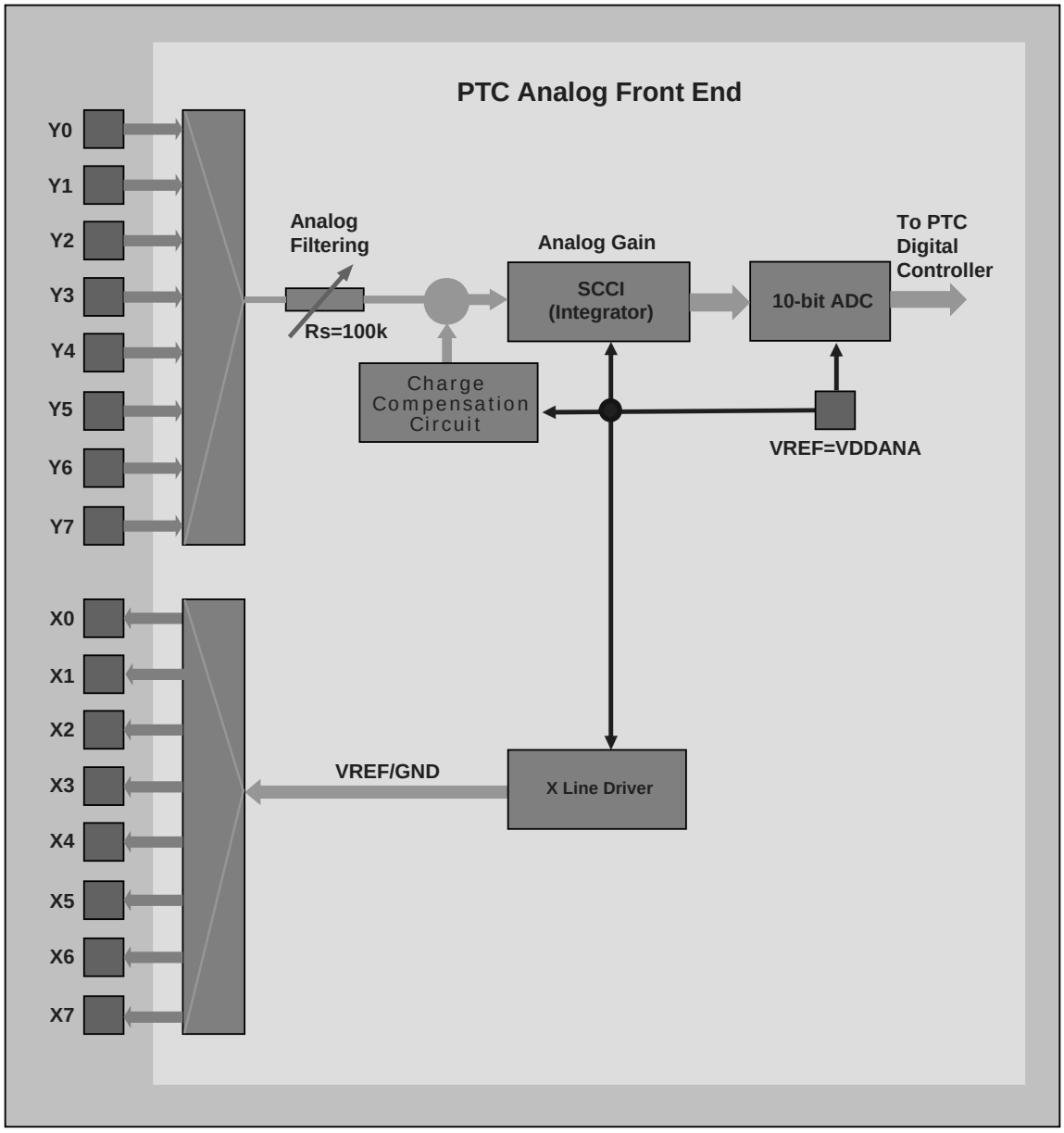
**Note:** The actual number of peripherals implemented is device-specific; see Table 18-9 ”Peripheral Identifiers” for details.

**NSECPy: Non-secured Peripheral**

0: The selected peripheral address space is configured as “Secured” access (value of ‘0’ has no effect if the peripheral security type is “Peripheral Always Non-secured”).

1: The selected peripheral address space is configured as “Non-secured” access (value of ‘1’ has no effect if the peripheral security type is “Peripheral Always Secured”).

**Figure 29-5: PTC Analog Front End**




### 29.6.5 Operations in Mutual Capacitance

A mutual capacitance sensor is formed between two I/O lines, a PTC_X electrode for transmitting, and a PTC_Y electrode for receiving. The mutual capacitance between the PTC_X and PTC_Y electrodes is calibrated and measured by the PTC. It is not necessary to connect all X and Y lines; when unused, they can be left unconnected.

**KEY: Password**

| Value | Name | Description |
|---|---|---|
| 0x37 | PASSWD | Writing any other value in this field aborts the write operation. |

**MOSCSEL: Main Clock Oscillator Selection**

0: The 12 MHz oscillator is selected.

1: The 8 to 24 MHz crystal oscillator is selected.

**CFDEN: Clock Failure Detector Enable**

0: The clock failure detector is disabled.

1: The clock failure detector is enabled.

**Table 40-13: Example of Pdelay_Resp Frame in 1588 Version 2 (UDP/IPv6) Format**

| Frame Segment | Value |
|---|---|
| Preamble/SFD | 55555555555555D5 |
| DA (Octets 0–5) | — |
| SA (Octets 6–11) | — |
| Type (Octets 12–13) | 86dd |
| IP stuff (Octets 14–19) | — |
| UDP (Octet 20) | 11 |
| IP stuff (Octets 21–37) | — |
| IP DA (Octets 38–53) | FF0200000000006B |
| Source IP port (Octets 54–55) | — |
| Dest IP port (Octets 56–57) | 013F |
| Other stuff (Octets 58–61) | — |
| Message type (Octet 62) | 03 |
| Other stuff (Octets 63–93) | — |
| Version PTP (Octet 94) | 02 |

For the multicast address 011B19000000 sync and delay request frames are recognized depending on the message type field, 00 for sync and 01 for delay request.

**Table 40-14: Example of Sync Frame in 1588 Version 2 (Ethernet Multicast) Format**

| Frame Segment | Value |
|---|---|
| Preamble/SFD | 55555555555555D5 |
| DA (Octets 0–5) | 011B19000000 |
| SA (Octets 6–11) | — |
| Type (Octets 12–13) | 88F7 |
| Message type (Octet 14) | 00 |
| Version PTP (Octet 15) | 02 |

## 46.5 Product Dependencies

### 46.5.1 I/O Lines

Both TWD and TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor. When the bus is free, both lines are high. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

TWD and TWCK pins may be multiplexed with PIO lines. To enable the TWIHS, the user must program the PIO Controller to dedicate TWD and TWCK as peripheral lines. When High-speed Slave mode is enabled, the analog pad filter must be enabled.

The user must not program TWD and TWCK as open-drain. This is already done by the hardware.

**Table 46-4: I/O Lines**

| Instance | Signal | I/O Line | Peripheral |
|---|---|---|---|
| TWIHS0 | TWCK0 | PC0 | D |
| TWIHS0 | TWCK0 | PC28 | E |
| TWIHS0 | TWCK0 | PD22 | B |
| TWIHS0 | TWCK0 | PD30 | E |
| TWIHS0 | TWD0 | PB31 | D |
| TWIHS0 | TWD0 | PC27 | E |
| TWIHS0 | TWD0 | PD21 | B |
| TWIHS0 | TWD0 | PD29 | E |
| TWIHS1 | TWCK1 | PC7 | C |
| TWIHS1 | TWCK1 | PD5 | A |
| TWIHS1 | TWCK1 | PD20 | B |
| TWIHS1 | TWD1 | PC6 | C |
| TWIHS1 | TWD1 | PD4 | A |
| TWIHS1 | TWD1 | PD19 | B |

### 46.5.2 Power Management

Enable the peripheral clock.

The TWIHS may be clocked through the Power Management Controller (PMC), thus the user must first configure the PMC to enable the TWIHS clock.

### 46.5.3 Interrupt Sources

The TWIHS has an interrupt line connected to the Interrupt Controller. In order to handle interrupts, the Interrupt Controller must be programmed before configuring the TWIHS.

**Table 46-5: Peripheral IDs**

| Instance | ID |
|---|---|
| TWIHS0 | 29 |
| TWIHS1 | 30 |

This gives a maximum baud rate of peripheral clock divided by 8, assuming that peripheral clock is the highest possible clock and that the OVER bit is set.

- Baud Rate Calculation Example

Table 47-5 shows calculations of CD to obtain a baud rate at 38,400 bit/s for different source clock frequencies. This table also shows the actual resulting baud rate and the error.

**Table 47-5: Baud Rate Example (OVER = 0)**

| Source Clock (MHz) | Expected Baud Rate (bit/s) | Calculation Result | CD | Actual Baud Rate (bit/s) | Error |
|---|---|---|---|---|---|
| 3,686,400 | 38,400 | 6.00 | 6 | 38,400.00 | 0.00% |
| 4,915,200 | 38,400 | 8.00 | 8 | 38,400.00 | 0.00% |
| 5,000,000 | 38,400 | 8.14 | 8 | 39,062.50 | 1.70% |
| 7,372,800 | 38,400 | 12.00 | 12 | 38,400.00 | 0.00% |
| 8,000,000 | 38,400 | 13.02 | 13 | 38,461.54 | 0.16% |
| 12,000,000 | 38,400 | 19.53 | 20 | 37,500.00 | 2.40% |
| 12,288,000 | 38,400 | 20.00 | 20 | 38,400.00 | 0.00% |
| 14,318,180 | 38,400 | 23.30 | 23 | 38,908.10 | 1.31% |
| 14,745,600 | 38,400 | 24.00 | 24 | 38,400.00 | 0.00% |
| 18,432,000 | 38,400 | 30.00 | 30 | 38,400.00 | 0.00% |
| 24,000,000 | 38,400 | 39.06 | 39 | 38,461.54 | 0.16% |
| 24,576,000 | 38,400 | 40.00 | 40 | 38,400.00 | 0.00% |
| 25,000,000 | 38,400 | 40.69 | 40 | 38,109.76 | 0.76% |
| 32,000,000 | 38,400 | 52.08 | 52 | 38,461.54 | 0.16% |
| 32,768,000 | 38,400 | 53.33 | 53 | 38,641.51 | 0.63% |
| 33,000,000 | 38,400 | 53.71 | 54 | 38,194.44 | 0.54% |
| 40,000,000 | 38,400 | 65.10 | 65 | 38,461.54 | 0.16% |
| 50,000,000 | 38,400 | 81.38 | 81 | 38,580.25 | 0.47% |

The baud rate is calculated with the following formula:

Baud rate = MCK / CD $\times$ 16

The baud rate error is calculated with the following formula. It is not recommended to work with an error higher than 5%.

$$\text{Error} = 1 - \left(\frac{\text{Expected Baud Rate}}{\text{Actual Baud Rate}}\right)$$

#### 47.7.1.2 Fractional Baud Rate in Asynchronous Mode

The baud rate generator previously defined is subject to the following limitation: the output frequency changes by only integer multiples of the reference frequency. An approach to this problem is to integrate a fractional N clock generator that has a high resolution. The generator architecture is modified to obtain baud rate changes by a fraction of the reference source clock. This fractional part is programmed with the FP field in FLEX_US_BRGR. If FP is not 0, the fractional part is activated. The resolution is one eighth of the clock divider. The fractional baud rate is calculated using the following formula:

$$\text{Baud rate} = \frac{\text{Selected Clock}}{\left(8(2 - \text{OVER})\left(\text{CD} + \frac{\text{FP}}{8}\right)\right)}$$

The modified architecture is presented in Figure 47-3.

At any time, the software can perform a reset on the receiver or the transmitter of the USART by setting the corresponding bit, RSTRX and RSTTX respectively, in FLEX_US_CR. The software resets clear the status flag and reset internal state machines but the user interface configuration registers hold the value configured prior to software reset. Regardless of what the receiver or the transmitter is performing, the communication is immediately stopped.

The user can also independently disable the receiver or the transmitter by setting RXDIS and TXDIS respectively in FLEX_US_CR. If the receiver is disabled during a character reception, the USART waits until the end of reception of the current character, then the reception is stopped. If the transmitter is disabled while it is operating, the USART waits the end of transmission of both the current character and character being stored in the USART Transmit Holding Register (FLEX_US_THR). If a timeguard is programmed, it is handled normally.

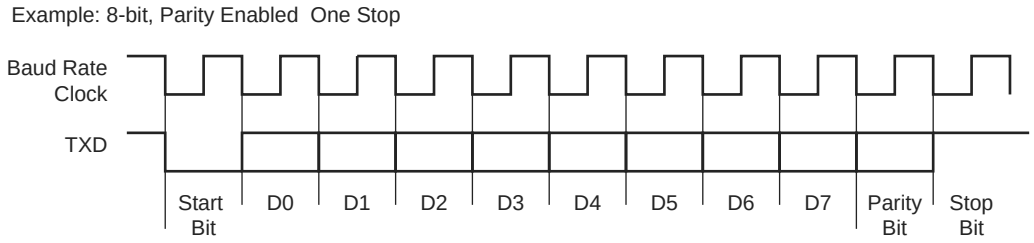
### 47.7.3 Synchronous and Asynchronous Modes

#### 47.7.3.1 Transmitter Operations

The transmitter performs the same in both Synchronous and Asynchronous operating modes (SYNC = 0 or SYNC = 1). One start bit, up to 9 data bits, 1 optional parity bit and up to 2 stop bits are successively shifted out on the TXD pin at each falling edge of the programmed serial clock.

The number of data bits is selected by the CHRL field and the MODE9 bit in FLEX_US_MR. Nine bits are selected by setting the MODE9 bit regardless of the CHRL field. The parity bit is set according to the PAR field in FLEX_US_MR. The even, odd, space, marked or none parity bit can be configured. The MSBF bit in FLEX_US_MR configures which data bit is sent first. If written to 1, the most significant bit is sent first. If written to 0, the less significant bit is sent first. The number of stop bits is selected by the NBSTOP field in FLEX_US_MR. The 1.5 stop bit is supported in Asynchronous mode only.

**Figure 47-5: Character Transmit**




The characters are sent by writing in FLEX_US_THR. The transmitter reports two status bits in the USART Channel Status Register (FLEX_US_CSR): TXRDY (Transmitter Ready), which indicates that FLEX_US_THR is empty and TXEMPTY, which indicates that all the characters written in FLEX_US_THR have been processed. When the current character processing is completed, the last character written in FLEX_US_THR is transferred into the shift register of the transmitter and FLEX_US_THR is emptied, thus TXRDY rises.

Both TXRDY and TXEMPTY bits are low when the transmitter is disabled. Writing a character in FLEX_US_THR while TXRDY is low has no effect and the written character is lost.

**Figure 47-6: Transmitter Status**

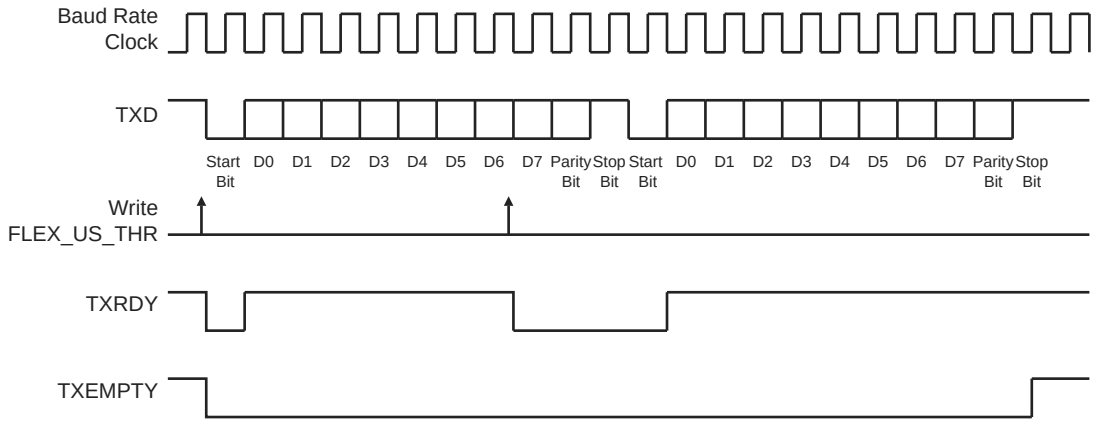
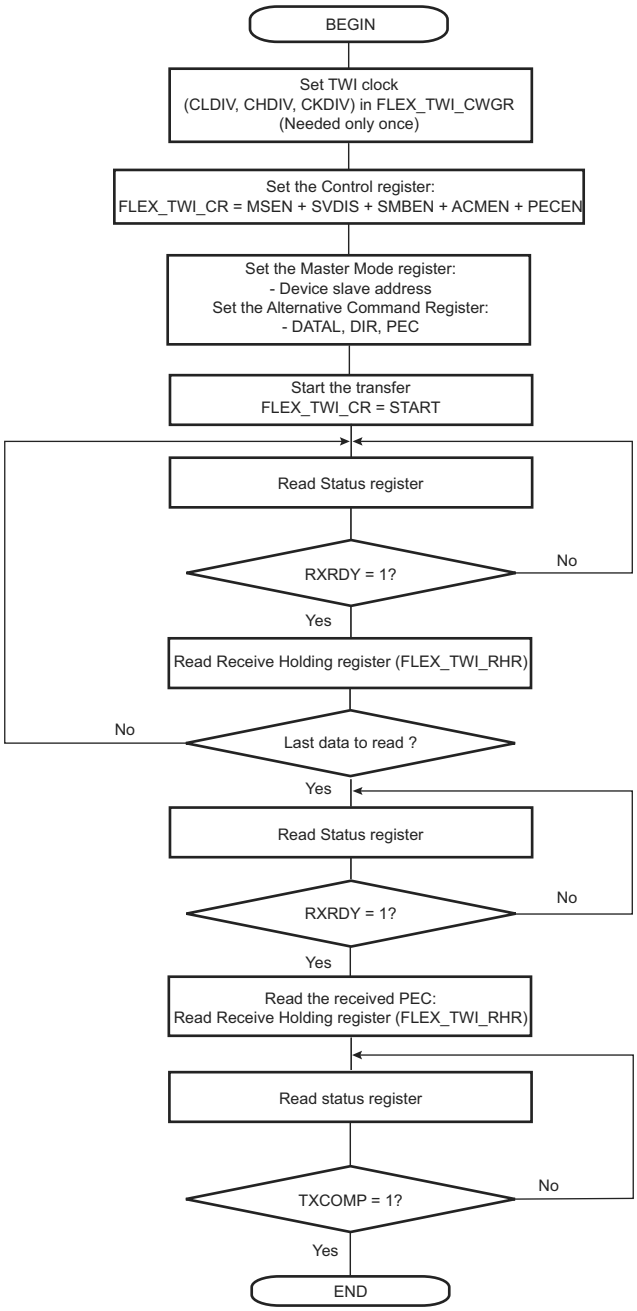


**Figure 47-106: TWI Read Operation with Multiple Data Bytes with Alternative Command Mode with PEC**



### 47.10.47 SPI Receive Data Register (FIFO Multiple Data, 16-bit)

**Name:** FLEX_SPI_RDR (FIFO_MULTI_DATA_16)

**Address:** 0xF8034408 (0), 0xF8038408 (1), 0xFC010408 (2), 0xFC014408 (3), 0xFC018408 (4)

**Access:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RD1 | | | | | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| RD1 | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RD0 | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RD0 | | | | | | | |

**Note:** If FIFO is enabled (FLEX_SPI_CR.FIFOEN) and FLEX_SPI_FMR.RXRDYM > 0, see Section 47.8.7.7 “SPI Multiple Data Mode” for details.

**RDx: Receive Data**

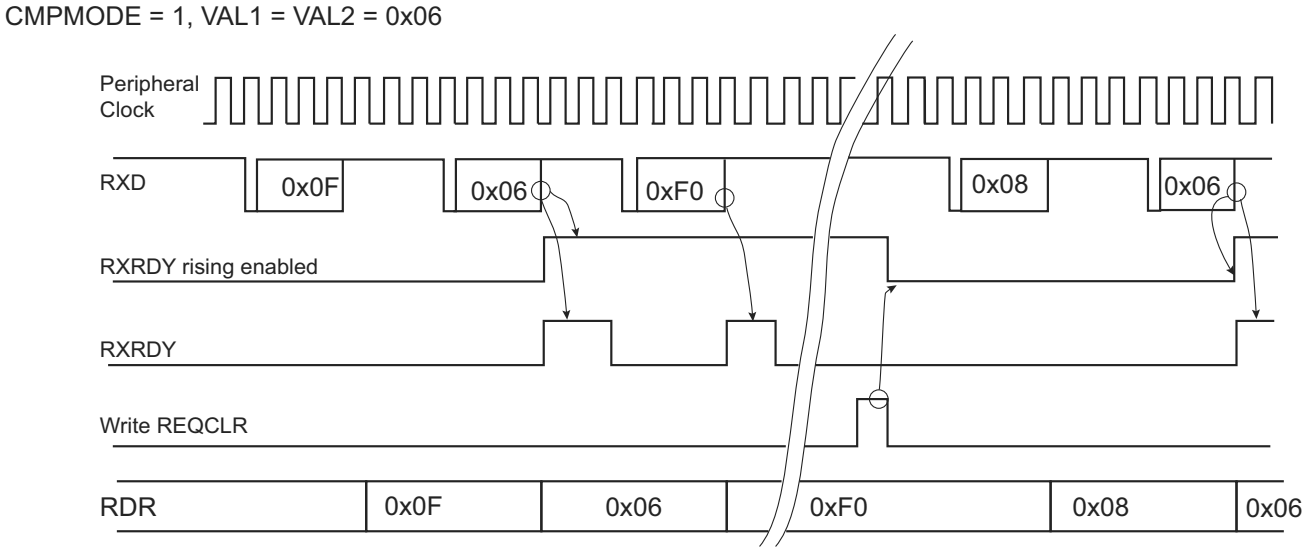
First unread data in the Receive FIFO. Data received by the SPI Interface is stored in this register in a right-justified format. Unused bits are read as zero.

VAL1.

- If VAL1 is strictly lower than VAL2, then any value between VAL1 and VAL2 sets the CMP flag.
- If VAL1 is strictly higher than VAL2, then the flag CMP is set to 1 if either received character equals VAL1 or VAL2.

By programming the CMPMODE bit to 1, the comparison function result triggers the start of the loading of UART_RHR (see Figure 48-12). The trigger condition occurs as soon as the received character value matches the condition defined by the programming of VAL1, VAL2 and CMPPAR in UART_CMPR. The comparison trigger event can be restarted by writing a one to the REQCLR bit in UART_CR.

**Figure 48-12: Receive Holding Register Management**




### 48.5.6 Asynchronous and Partial Wakeup (SleepWalking)

Asynchronous and partial wakeup (SleepWalking) is a means of data preprocessing that qualifies an incoming event, thus allowing the UART to decide whether or not to wake up the system. SleepWalking is used primarily when the system is in Wait mode (refer to Section 33. “Power Management Controller (PMC)”) but can also be enabled when the system is fully running.

No access must be performed in the UART between the enable of asynchronous partial wakeup and the wakeup performed by the UART.

If the system is in Wait mode and asynchronous and partial wakeup is enabled, the maximum baud rate that can be achieved equals 19200.

If the system is running or in Sleep mode, the maximum baud rate that can be achieved equals 115200 or higher. This limit is bounded by the peripheral clock frequency divided by 16.

The UART_RHR must be read before enabling asynchronous and partial wakeup.

When SleepWalking is enabled for the UART (refer to Section 33. “Power Management Controller (PMC)”), the PMC decodes a clock request from the UART. The request is generated as soon as there is a falling edge on the RXD line as this may indicate the beginning of a start bit. If the system is in Wait mode (processor and peripheral clocks switched off), the PMC restarts the fast RC oscillator and provides the clock only to the UART.

As soon as the clock is provided by the PMC, the UART processes the received frame and compares the received character with VAL1 and VAL2 in UART_CMPR (Section 48.6.10 “UART Comparison Register”).

The UART instructs the PMC to disable the clock if the received character value does not meet the conditions defined by VAL1 and VAL2 fields in UART_CMPR (see Figure 48-14).

If the received character value meets the conditions, the UART instructs the PMC to exit the full system from Wait mode (see Figure 48-13).

The VAL1 and VAL2 fields can be programmed to provide different comparison methods and thus matching conditions.

- If VAL1 equals VAL2, then the comparison is performed on a single value and the wakeup is triggered if the received character equals VAL1.
- If VAL1 is strictly lower than VAL2, then any value between VAL1 and VAL2 wakes up the system.
- If VAL1 is strictly higher than VAL2, then the wakeup is triggered if the received character equals VAL1 or VAL2.
- If VAL1 = 0 and VAL2 = 255, the wakeup is triggered as soon as a character is received.

**Table 49-2: I/O Lines (Continued)**

| | | | |
|---|---|---|---|
| SPI1 | SPI1_NPCS0 | PD28 | A |
| SPI1 | SPI1_NPCS1 | PA26 | D |
| SPI1 | SPI1_NPCS1 | PC5 | D |
| SPI1 | SPI1_NPCS1 | PD29 | A |
| SPI1 | SPI1_NPCS2 | PA27 | D |
| SPI1 | SPI1_NPCS2 | PC6 | D |
| SPI1 | SPI1_NPCS2 | PD30 | A |
| SPI1 | SPI1_NPCS3 | PA28 | D |
| SPI1 | SPI1_NPCS3 | PC7 | D |
| SPI1 | SPI1_SPCK | PA22 | D |
| SPI1 | SPI1_SPCK | PC1 | D |
| SPI1 | SPI1_SPCK | PD25 | A |

### 49.6.2 Power Management

The SPI can be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the SPI clock.

### 49.6.3 Interrupt

The SPI interface has an interrupt line connected to the interrupt controller. Handling the SPI interrupt requires programming the interrupt controller before configuring the SPI.

**Table 49-3: Peripheral IDs**

| Instance | ID |
|---|---|
| SPI0 | 33 |
| SPI1 | 34 |

### 49.6.4 Direct Memory Access Controller (DMAC)

The SPI interface can be used in conjunction with the DMAC in order to reduce processor overhead. For a full description of the DMAC, refer to Section 38. “DMA Controller (XDMAC)”.

#### 49.7.7.6 Single Data Mode

In Single Data mode, only one data is written every time SPI_TDR is accessed, and only one data is read every time SPI_RDR is accessed.

When SPI_FMR.TXRDYM = 0, the Transmit FIFO operates in Single Data mode.

When SPI_FMR.RXRDYM = 0, the Receive FIFO operates in Single Data mode.

If Master mode is used (SPI_MR.MSTR=1), the Receive FIFO must operate in Single Data mode.

If Variable Peripheral Select mode is used (SPI_MR.PS=1), the Transmit FIFO must operate in Single Data mode.

See Section 49.8.6 ”SPI Transmit Data Register” and Section 49.8.3 ”SPI Receive Data Register”.

- DMAC

When FIFOs operate in Single Data mode, the DMAC transfer type must be configured either in bytes, halfwords or words depending on SPI_MR.PS bit value and SPI_CSRx.BITS field value.

The same conditions for transfer type apply when FIFOs are disabled.

#### 49.7.7.7 Multiple Data Mode

Multiple Data mode minimizes the number of accesses by concatenating the data to send/read in one access.

When SPI_FMR.TXRDYM > 0, the Transmit FIFO operates in Multiple Data mode.

When SPI_FMR.RXRDYM > 0, the Receive FIFO operates in Multiple Data mode.

Multiple data can be read from the Receive FIFO only in Slave mode (SPI_MR.MSTR=0).

The Transmit FIFO can be loaded with multiple data in the same access by configuring TXRDYM>0 and when SPI_MR.PS=0.

In Multiple Data mode, up to two data can be written in one SPI_TDR write access. It is also possible to read up to four data in one SPI_RDR access if SPI_CSRx.BITS is configured to ‘0’ (8-bit data size) and up to two data if SPI_CSRx.BITS is configured to a value other than ‘0’ (more than 8-bit data size).

The number of data to write/read is defined by the size of the register access. If the access is a byte-size register access, only one data is written/read. If the access is a halfword size register access, then up to two data are read and only one data is written. Lastly, if the access is a word-size register access, then up to four data are read and up to two data are written.

Written/read data are always right-aligned, as described in Section 49.8.4 “SPI Receive Data Register (FIFO Multiple Data, 8-bit)”, Section 49.8.5 “SPI Receive Data Register (FIFO Multiple Data, 16-bit)” and Section 49.8.7 “SPI Transmit Data Register (FIFO Multiple Data, 8- to 16-bit)”.

As an example, if the Transmit FIFO is empty and there are six data to send, either of the following write accesses may be performed:

- six SPI_TDR-byte write accesses
- three SPI_TDR-halfword write accesses

With a Receive FIFO containing six data, any of the following read accesses may be performed:

- six SPI_RDR-byte read accesses
- three SPI_RDR-halfword read accesses
- one SPI_RDR-word read access and one SPI_RDR-halfword read access
- TDRE and RDRF Configuration

In Multiple Data mode, it is possible to write one or more data in the same SPI_TDR/SPI_RDR access. The TDRE flag indicates if one or more data can be written in the FIFO depending on the configuration of SPI_FMR.TXRDYM/RXRDYM.

As an example, if two data are written each time in SPI_TDR, it is useful to configure the TXRDYM field to the value ‘1’ so that the TDRE flag is at ‘1’ only when at least two data can be written in the Transmit FIFO.

Similarly, if four data are read each time in SPI_RDR, it is useful to configure the RXRDYM field to the value ‘2’ so that the RDRF flag is at ‘1’ only when at least four unread data are in the Receive FIFO.

- DMAC

It is mandatory to configure DMAC channel size (byte, halfword or word) according to FLEX_SPI_FMR.TXRDYM/RXRDYM configuration. See Section 49.7.7.7 ”Multiple Data Mode” for constraints.

#### 49.7.7.8 FIFO Pointer Error

A FIFO overflow is reported in SPI_SR.

If the Transmit FIFO is full and a write access is performed on SPI_TDR, it generates a Transmit FIFO pointer error and sets SPI_SR.TXFPTEF.

### 52.6.30 ISC Gamma Correction Control Register

**Name:** ISC_GAM_CTRL

**Address:** 0xF0008094

**Access:** Read/Write

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | | | RENABLE | GENABLE | BENABLE | ENABLE |

**ENABLE: Gamma Correction Enable**

0: Gamma correction is disabled.

1: Gamma correction is enabled.

**BENABLE: Gamma Correction Enable for B Channel**

0: 12 bits to 10 bits compression is performed skipping two bits.

1: Piecewise interpolation is used to perform 12 bits to 10 bits compression for the blue channel.

**GENABLE: Gamma Correction Enable for G Channel**

0: 12 bits to 10 bits compression is performed skipping two bits.

1: Piecewise interpolation is used to perform 12 bits to 10 bits compression for the green channel.

**RENABLE: Gamma Correction Enable for R Channel**

0: 12 bits to 10 bits compression is performed skipping two bits.

1: Piecewise interpolation is used to perform 12 bits to 10 bits compression for the red channel.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index MCAN_TXFQS.TFQPI. An Add Request increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (MCAN_TXFQS.TFQF = '1') is signalled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a '1' to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFO's Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via MCAN_TXBAR. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (see Table 53-8). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index MCAN_TXFQS.TFQPI (0…31) × Element Size to the Tx Buffer Start Address MCAN_TXBC.TBSA.

#### 53.5.5.4 Tx Queue

Tx Queue operation is configured by programming MCAN_TXBC.TFQM to '1'. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx Buffer referenced by the Put Index MCAN_TXFQS.TFQPI. An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (MCAN_TXFQS.TFQF = '1'), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

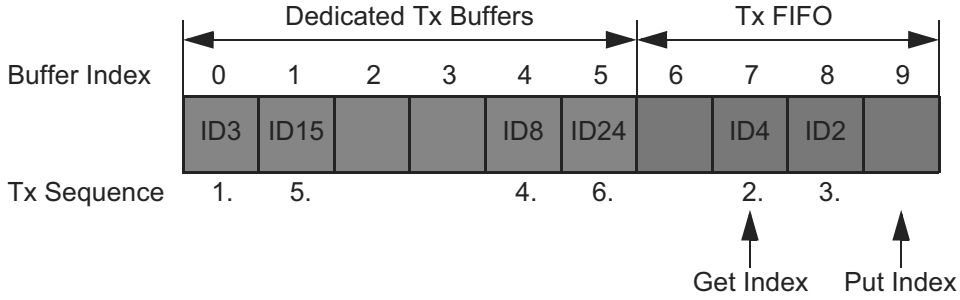
The application may use register MCAN_TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (see Table 53-8). Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index MCAN_TXFQS.TFQPI (0…31) × Element Size to the Tx Buffer Start Address MCAN_TXBC.TBSA.

#### 53.5.5.5 Mixed Dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of dedicated Tx Buffers and a Tx FIFO. The number of dedicated Tx Buffers is configured by MCAN_TXBC.NDTB. The number of Tx Buffers assigned to the Tx FIFO is configured by MCAN_TXBC.TFQS. In case MCAN_TXBC.TFQS is programmed to zero, only dedicated Tx Buffers are used.

**Figure 53-10: Example of Mixed Configuration Dedicated Tx Buffers / Tx FIFO**




Tx prioritization:

- Scan dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by MCAN_TXFS.TFGI)
- Buffer with lowest Message ID gets highest priority and is transmitted next

#### 53.5.5.6 Mixed Dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of dedicated Tx Buffers and a Tx Queue. The number of dedicated Tx Buffers is configured by MCAN_TXBC.NDTB. The number of Tx Queue Buffers is configured by MCAN_TXBC.TFQS. In case MCAN_TXBC.TFQS is programmed to zero, only dedicated Tx Buffers are used.

### 54.7.5 TC Register AB

**Name:** TC_RABx [x=0..2]

**Address:** 0xF800C00C (0)[0], 0xF800C04C (0)[1], 0xF800C08C (0)[2], 0xF801000C (1)[0], 0xF801004C (1)[1], 0xF801008C (1)[2]

**Access:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RAB | | | | | | | |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| RAB | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RAB | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RAB | | | | | | | |

**RAB: Register A or Register B**

RAB contains the next unread capture Register A or Register B value in real time. It is usually read by the DMA after a request due to a valid load edge on TIOAx.

When DMA is used, the RAB register address must be configured as source address of the transfer.

#### 57.4.5.2 Fuse Programming

All the fuses can be written by software.

The sequence of instructions to program fuses is the following:

1. Write the key code 0xFB in the Key Register (SFC_KR).
2. Write the word to program in the corresponding Data Register (SFC_DRx).
   For example, if fuses 0 to 31 must be programmed, Data Register 0 (SFC_DR0) must be written. If fuses 32 to 61 must be programmed, Data Register 1 (SFC_DR1) must be written. Only the data bits set to level ‘1’ are programmed.
3. Wait for flag PGMC to rise in the Status Register (SFC_SR) by polling or interrupt.
4. Check the value of flag PGMF: if it is set to 1, it means that the programming procedure failed. After programming, the fuses are read back in the corresponding SFC_DRx.

#### 57.4.5.3 Fuse Masking

It is possible to mask a fuse array. Once the fuse masking is enabled, the data registers from SFC_DR20 to SFC_DR23 are read at a value of ‘0’, regardless of the fuse state (the registers that are masked depend on the SFC hardware customizing).

To activate fuse masking, the MSK bit of the SFC Mode Register (SFC_MR) must be written to level ‘1’. The MSK bit is set-only. Only a hardware reset can disable fuse masking.

The MSK bit has no effect on the programming of masked fuses.

### 57.4.6 Fuse Functions

The “Fuse Box Controller” section defines the fuse bits that can be used as general purpose bits when standard boot is used.

If secure boot is used, refer to the device “Secure Boot Strategy” application note included in the Secure Package.

### 58.6.11 ICM User Initial Hash Value Register

**Name:** ICM_UIHVALx [x=0..7]

**Address:** 0xF8040038

**Access:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| VAL | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VAL | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| VAL | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VAL | | | | | | | |

**VAL: Initial Hash Value**

When IMC_CFG.UIHASH is set, the Initial Hash Value is user-programmable.

To meet the desired standard, use the following example values.

For ICM_UIHVAL0 field:

| Example | Comment |
|---|---|
| 0x67452301 | SHA1 algorithm |
| 0xC1059ED8 | SHA224 algorithm |
| 0x6A09E667 | SHA256 algorithm |

For ICM_UIHVAL1 field:

| Example | Comment |
|---|---|
| 0xEFCDAB89 | SHA1 algorithm |
| 0x367CD507 | SHA224 algorithm |
| 0xBB67AE85 | SHA256 algorithm |

For ICM_UIHVAL2 field:

| Example | Comment |
|---|---|
| 0x98BADCFE | SHA1 algorithm |
| 0x3070DD17 | SHA224 algorithm |
| 0x3C6EF372 | SHA256 algorithm |

For ICM_UIHVAL3 field:

| Example | Comment |
|---|---|
| 0x10325476 | SHA1 algorithm |
| 0xF70E5939 | SHA224 algorithm |
| 0xA54FF53A | SHA256 algorithm |

**Table 72-4: SAMA5D2 Datasheet Rev. 11267D Revision History (Continued)**

| Issue Date | Changes |
|---|---|
| 12-May-16 | Section 61. “Analog-to-Digital Converter (ADC)” (cont’d)<br>Updated Section 61.2 “Embedded Characteristics”<br>Updated Figure 61-1 “Analog-to-Digital Converter Block Diagram”<br>Revised Section 61.5 “Product Dependencies”<br>Revised Section 61.6.1 “Analog-to-Digital Conversion”<br>Updated Section 61.6.3 “ADC Reference Voltage” and Section 61.6.4 “Conversion Resolution”<br>Updated Section 61.6.7 “Conversion Triggers”<br>Section 61.6.9 “Comparison Window”: in fourth paragraph, instance of “ADC_SR” corrected to “ADC_ISR”<br>Section 61.6.10 “Differential and Single-ended Input Modes”: changed title (was “Differential Inputs”) and revised content<br>Updated Section 61.6.11 “ADC Timings”, Section 61.6.12 “Last Channel Specific Measurement Trigger”, Section 61.6.13 “Enhanced Resolution Mode and Digital Averaging Function” and Section 61.6.14 “Automatic Error Correction”<br>Instances of GND renamed to GNDANA in Figure 61-15 “Touchscreen Switches Implementation”, Figure 61-18 “Touchscreen Switches Implementation” and Figure 61-20 “Touchscreen Pen Detect”<br>Updated Section 61.6.16 “Asynchronous and Partial Wakeup (SleepWalking)”<br>Section 61.6.17.1 “Classic ADC Channels Only (Touchscreen Disabled)”: changed title (was “Classical ADC Channels Only”)<br>Section 61.6.19 “Register Write Protection”: updated list of protectable registers<br>Table 61-8 “Register Mapping”:<br>- defined 0x48 as reserved<br>- added row 0x4C / Channel Offset Register / ADC_COR<br>- added offset 0x7C for ADC_CDR11<br>- defined offset range 0x80–0x90 as reserved<br>- added row 0x94 / Analog Control Register / ADC_ACR<br>- defined offset range 0xC4–0xD0 as reserved<br>- added row 0xD4 / Correction Values Register / ADC_CVR<br>- added row 0xD8 / Channel Error Correction Register / ADC_CECR<br>- added row 0xDC / Touchscreen Correction Values Register / ADC_TSCVR<br>- defined offset 0xE0 as reserved<br>Section 61.7.2 “ADC Mode Register”: updated TRACKIM field description<br>Added LCCHG (Last Channel Change) bit in Section 61.7.9 “ADC Interrupt Enable Register”, Section 61.7.10 “ADC Interrupt Disable Register”, Section 61.7.11 “ADC Interrupt Mask Register” and Section 61.7.12 “ADC Interrupt Status Register”<br>Section 61.7.13 “ADC Last Channel Trigger Mode Register”: updated CMPMOD field description<br>Section 61.7.16 “ADC Extended Mode Register”: updated CMPMODE field description; added descriptions for fields OSR ASTE<br>Section 61.7.18 “ADC Channel Offset Register”: added address; removed bits OFF11:OFF0 from bitmap; modified DIFFx field description<br>Section 61.7.20 “ADC Analog Control Register”: added address; added IBCTL field<br>Section 61.7.25 “ADC Trigger Register”: added sentence about write protection<br>Removed Section “Correction Select Register”<br>Added sentence about write protection in Section 61.7.26 “ADC Correction Values Register” and Section 61.7.27 “ADC Channel Error Correction Register”<br>Added Section 61.7.28 “ADC Touchscreen Correction Values Register” |

**Table 72-5: SAMA5D2 Datasheet Rev. 11267C Revision History (Continued)**

| Issue Date | Changes |
|---|---|
| 8-Jan-16 | Section 41. “Inter-IC Sound Controller (I2SC)” |
| | Section 41.6.3 “Master, Controller and Slave Modes”: removed text fragment: ‘in order to avoid unwanted glitches on the I2SWS and I2SCK pins.’ |
| | Section 41.8.2 “Inter-IC Sound Controller Mode Register”: removed text fragment: ‘in order to avoid unexpected behavior on the I2SWS, I2SCK and I2SDO outputs.’ and added note (2) below IMCKDIV field description. |
| | Section 44. “Flexible Serial Communication Controller (FLEXCOM)” |
| | Restored all references to ISO7816 specification |
| | Updated Figure 44-3 “Fractional Baud Rate Generator” |
| | Added Figure 44-27 “RTS line software control when FLEX_US_MR.USART_MODE = 2” |
| | Section 44.10.6 “USART Mode Register”: updated USART_MODE field description (SPI_MASTER item) |
| | Section 44.10.44 “SPI Mode Register”: added LBHPC bit |
| | Section 55. “Universal Asynchronous Receiver Transmitter (UART)” |
| | Section 55.6.9 “UART Baud Rate Generator Register”: in CD field description, corrected equation after “If BRSRCCK = 1” |
| | Section 59. “Quad SPI Interface (QSPI)” |
| | Section 59.7.5 “QSPI Status Register”: updated RDRF, TDRE, TXEMPTY, and OVRES field descriptions |
| | Section 48. “Secure Digital Multimedia Card Controller (SDMMC)” |
| | Section 48.12.41 “SDMMC Preset Value Register”: updated CLKGSEL field description |
| | Section 49. “Image Sensor Controller (ISC)” |
| | Section 49.1 “Description”: removed "serial csi-2 based CMOS/CCD sensor" (not supported). |
| | Section 50. “Controller Area Network (MCAN)” |
| | Changed MCAN interrupt line names to MCAN_INT0 and MCAN_INT1 thoughout the section |
| | Section 50.6.7 “MCAN CC Control Register”: added bit NISO |
| | Section 51. “Timer Counter (TC)” |
| | Reformatted and renamed Table 51-2 “Channel Signal Description” |
| | Section 51.6.3 “Clock Selection”: updated notes (1) and (2) |
| | Section 52. “Pulse Density Modulation Interface Controller (PDMIC)” |
| | Replaced all instances of “PCK” with “GCLK” |
| | Section 52.2 “Embedded Characteristics”: removed ‘Multiplexed PDM Input Support’ characteristic |
| | Updated Section 52.5.2 “Power Management” and Section 52.6.2.1 “Description” |
| | Section 52.6.2.6 “Gain and Offset Compensation”: updated *dgain* bullet |
| | Section 52.7.3 “PDMIC Converted Data Register”: updated DATA field description |
| | Section 52.7.8 “PDMIC DSP Configuration Register 0”: updated OSR field description |
| | Section 61. “Security Module” |
| | Section 61.5.5 “SECUMOD Status Clear Register”: removed MCKM field description |
| | Section 61.5.18 “SECUMOD Wake Up Register”: removed TPML field description |
| | Section 77. “Analog-to-Digital Converter (ADC)” |
| | Section 77.7.2 “ADC Mode Register”: updated TRACKTIM and TRANSFER field descriptions. |