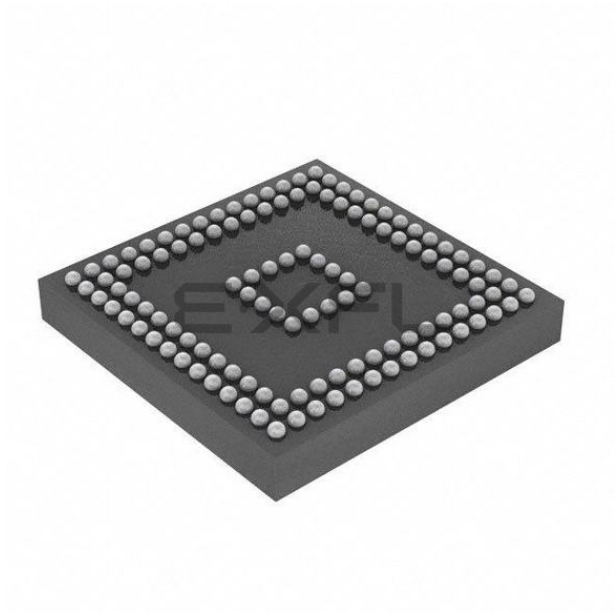


**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Active |
| Core Processor | ARM® Cortex®-M4F |
| Core Size | 32-Bit Single-Core |
| Speed | 120MHz |
| Connectivity | CANbus, EBI/EMI, Ethernet, I²C, IrDA, LINbus, MMC/SD, QSPI, SPI, UART/USART, USB |
| Peripherals | Brown-out Detect/Reset, DMA, I²S, POR, PWM, WDT |
| Number of I/O | 99 |
| Program Memory Size | 512KB (512K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 192K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.71V ~ 3.63V |
| Data Converters | A/D 32x12b; D/A 2x12b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 120-TFBGA |
| Supplier Device Package | 120-TFBGA (8x8) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atsame54p19a-ctut |

**Related Links**

9.4 NVM User Page Mapping

### 21.12.14 Tamper Control

**Name:** TAMPCTRL
**Offset:** 0x60
**Reset:** 0x00000000
**Property:** PAC Write-Protection, Enable-Protected

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| | | | | DEBNC4 | DEBNC3 | DEBNC2 | DEBNC1 | DEBNC0 |
| Access | | | | | | | | |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | | | | TAMLVL4 | TAMLVL3 | TAMLVL2 | TAMLVL1 | TAMLVL0 |
| Access | | | | | | | | |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | IN4ACT[1:0] | |
| Access | | | | | | | | |
| Reset | | | | | | | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IN3ACT[1:0] | | IN2ACT[1:0] | | IN1ACT[1:0] | | IN0ACT[1:0] | |
| Access | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 24, 25, 26, 27, 28 – DEBNC** Debounce Enable of Tamper Input INn

| Value | Description |
|---|---|
| 0 | Debouncing is disabled for Tamper input INn |
| 1 | Debouncing is enabled for Tamper input INn |

**Bits 16, 17, 18, 19, 20 – TAMLVL** Tamper Level Select of Tamper Input INn

| Value | Description |
|---|---|
| 0 | A falling edge condition will be detected on Tamper input INn. |
| 1 | A rising edge condition will be detected on Tamper input INn. |

**Bits 0:1, 2:3, 4:5, 6:7, 8:9 – INACT** Tamper Channel n Action
These bits determine the action taken by Tamper Channel n.

| Value | Name | Description |
|---|---|---|
| 0x0 | OFF | Off (Disabled) |
| 0x1 | WAKE | Wake and set Tamper flag |
| 0x2 | CAPTURE | Capture timestamp and set Tamper flag |
| 0x3 | ACTL | Compare RTC signal routed between INn and OUT pins . When a mismatch occurs, capture timestamp and set Tamper flag |

### 22.8.21 Channel Interrupt Enable Set

**Name:** CHINTENSET
**Offset:** 0x4D + n*0x10 [n=0..31]
**Reset:** 0x00
**Property:** PAC Write-Protection

This register allows the user to enable an interrupt without doing a read-modify-write operation. Changes in this register will also be reflected in the Channel Interrupt Enable Clear (CHINTENCLR) register.
This register affects the DMA channel that is selected in the Channel ID register (CHID.ID).

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | SUSP | TCMPL | TERR |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bit 2 – SUSP** Channel Suspend Interrupt Enable
Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Channel Suspend Interrupt Enable bit, which enables the Channel Suspend interrupt.

| Value | Description |
|---|---|
| 0 | The Channel Suspend interrupt is disabled. |
| 1 | The Channel Suspend interrupt is enabled. |

**Bit 1 – TCMPL** Channel Transfer Complete Interrupt Enable
Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Channel Transfer Complete Interrupt Enable bit, which enables the Channel Transfer Complete interrupt.

| Value | Description |
|---|---|
| 0 | The Channel Transfer Complete interrupt is disabled. |
| 1 | The Channel Transfer Complete interrupt is enabled. |

**Bit 0 – TERR** Channel Transfer Error Interrupt Enable
Writing a '0' to this bit has no effect.

Writing a '1' to this bit will set the Channel Transfer Error Interrupt Enable bit, which enables the Channel Transfer Error interrupt.

| Value | Description |
|---|---|
| 0 | The Channel Transfer Error interrupt is disabled. |
| 1 | The Channel Transfer Error interrupt is enabled. |

### 22.8.23 Channel Status

**Name:** CHSTATUS
**Offset:** 0x4F + n*0x10 [n=0..31]
**Reset:** 0x00
**Property:** -

This register affects the DMA channel that is selected in the Channel ID register (CHID.ID).

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CRCERR | FERR | BUSY | PEND |
| Access | | | | | R/W | R | R | R |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bit 3 – CRCERR** Channel CRC Error
This bit is set when the CRC monitor detects data corruption. This bit is cleared bu writing '1' to it, or by clearing the CRC Error bit in the INTPEND register (INTPEND.CRCERR).

**Bit 2 – FERR** Channel Fetch Error
This bit is cleared when a software resume command is executed.

This bit is set when an invalid descriptor is fetched.

**Bit 1 – BUSY** Channel Busy
This bit is cleared when the channel trigger action is completed, when a bus error is detected or when the channel is disabled.

This bit is set when the DMA channel starts a DMA transfer.

**Bit 0 – PEND** Channel Pending
This bit is cleared when the channel trigger action is started, when a bus error is detected or when the channel is disabled. For details on trigger action settings, refer to CHCTRLB.TRIGACT.

This bit is set when a transfer is pending on the DMA channel, as soon as the transfer request is received.

For IP, TCP or UDP checksum offload to be useful, the operating system containing the protocol stack must be aware that this offload is available so that it can make use of the fact that the hardware can either generate or verify the checksum.

#### 24.6.6.1 Receiver Checksum Offload

When receive checksum offloading is enabled in the GMAC Network Configuration Register (NCFGR.RXCOEN), the IPv4 header checksum is checked as per RFC 791, where the packet meets the following criteria:

- If present, the VLAN header must be four octets long and the CFI bit must not be set.
- Encapsulation must be RFC 894 Ethernet Type Encoding or RFC 1042 SNAP Encoding.
- IPv4 packet
- IP header is of a valid length

The GMAC also checks the TCP checksum as per RFC 793, or the UDP checksum as per RFC 768, if the following criteria are met:

- IPv4 or IPv6 packet
- Good IP header checksum (if IPv4)
- No IP fragmentation
- TCP or UDP packet

When an IP, TCP or UDP frame is received, the receive buffer descriptor gives an indication if the GMAC was able to verify the checksums. There is also an indication if the frame had SNAP encapsulation. These indication bits will replace the type ID match indication bits when the receive checksum offload is enabled. For details of these indication bits refer to “Receive Buffer Descriptor Entry”.

If any of the checksums are verified as incorrect by the GMAC, the packet is discarded and the appropriate statistics counter incremented.

#### 24.6.6.2 Transmitter Checksum Offload

The transmitter checksum offload is only available if the full store and forward mode is enabled. This is because the complete frame to be transmitted must be read into the packet buffer memory before the checksum can be calculated and written back into the headers at the beginning of the frame.

Transmitter checksum offload is enabled by setting bit [11] in the DMA Configuration register. When enabled, it will monitor the frame as it is written into the transmitter packet buffer memory to automatically detect the protocol of the frame. Protocol support is identical to the receiver checksum offload.

For transmit checksum generation and substitution to occur, the protocol of the frame must be recognized and the frame must be provided without the FCS field, by making sure that bit [16] of the transmit descriptor word 1 is clear. If the frame data already had the FCS field, this would be corrupted by the substitution of the new checksum fields.

If these conditions are met, the transmit checksum offload engine will calculate the IP, TCP and UDP checksums as appropriate. Once the full packet is completely written into packet buffer memory, the checksums will be valid and the relevant DPRAM locations will be updated for the new checksum fields as per standard IP/TCP and UDP packet structures.

If the transmitter checksum engine is prevented from generating the relevant checksums, bits [22:20] of the transmitter DMA writeback status will be updated to identify the reason for the error. Note that the frame will still be transmitted but without the checksum substitution, as typically the reason that the substitution did not occur was that the protocol was not recognized.

- If bit 17 of the Network Control register is written with a one, then the priority enable vector of the priority-based pause frame will be set equal to the value stored in the Transmit PFC Pause register [7:0]. For each entry equal to zero in the Transmit PFC Pause register [15:8], the pause quantum field of the pause frame associated with that entry will be taken from the transmit pause quantum register. For each entry equal to one in the Transmit PFC Pause register [15:8], the pause quantum associated with that entry will be zero.
- The Transmit Pause Quantum register resets to a value of 0xFFFF giving maximum pause quantum as default.

After transmission, a pause frame transmitted interrupt will be generated (bit 14 of the Interrupt Status register) and the only statistics register that will be incremented will be the Pause Frames Transmitted register.

PFC Pause frames can also be transmitted by the MAC using normal frame transmission methods.

### 24.6.18 Energy Efficient Ethernet Support

**Features**

- Energy Efficient Ethernet according to IEEE 802.3az
- A system's transmit path can enter a low power mode if there is nothing to transmit.
- A PHY can detect whether its link partner's transmit path is in low power mode, and configure its own receive path to enter low power mode.
- Link remains up during lower power mode and no frames are dropped.
- Asymmetric, one direction can be in low power mode while the other is transmitting normally.
- LPI (Low Power Idle) signaling is used to control entry and exit to and from low power modes.
  **Note:** LPI signaling can only take place if both sides have indicated support for it through auto-negotiation.

**Operation**

- Low power control is done at the MII (reconciliation sublayer).
- As an architectural convenience in writing the 802.3az it is assumed that transmission is deferred by asserting carrier sense - in practice it will not be done this way. This system will know when it has nothing to transmit and only enter low power mode when it is not transmitting.
- LPI should not be requested unless the link has been up for at least one second.
- LPI is signaled on the MII transmit path by asserting 0x01 on txd with tx_en low and tx_er high.
- A PHY on seeing LPI requested on the MII will send the sleep signal before going quiet. After going quiet it will periodically emit refresh signals.
- The sleep, quiet and refresh periods are defined in 802.3az, Table 78-2.
- LPI mode ends by transmitting normal idle for the wake time. There is a default time for this but it can be adjusted in software using the Link Layer Discovery Protocol (LLDP) described in 802.3az, Clause 79.
- LPI is indicated at the receive side when sleep and refresh signaling has been detected.

### 24.6.19 802.1Qav Support - Credit-based Shaping

A credit-based shaping algorithm is available on the two highest priority queues and is defined in the standard 802.1Qav: Forwarding and Queuing Enhancements for Time-Sensitive Streams. This allows traffic on these queues to be limited and to allow other queues to transmit.

Traffic shaping is enabled via the CBS (Credit Based Shaping) Control register. This enables a counter which stores the amount of transmit 'credit', measured in bytes that a particular queue has. A queue may

### 24.9.17 GMAC TX Partial Store and Forward Register

**Name:** TPSF
**Offset:** 0x040
**Reset:** 0x00000FFF
**Property:** -

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| | ENTXP | | | | | | | |
| Access | R/W | | | | | | | |
| Reset | 0 | | | | | | | |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | |
| Reset | | | | | | | | |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | TPB1ADR[11:8] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 1 | 1 | 1 | 1 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TPB1ADR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bit 31 – ENTXP** Enable TX Partial Store and Forward Operation

**Bits 11:0 – TPB1ADR[11:0]** Transmit Partial Store and Forward Address Watermark value.

Writing a zero to this bit has no effect.

Writing a '1' to this bit clears the DPLL1 Lock Fall interrupt flag.

**Bit 24 – DPLL1LCKR** DPLL1 Lock Rise
This flag is cleared by writing a '1' to it.

This flag is set on a zero-to-one transition of the DPLL1 Lock Rise bit in the Status register (STATUS.DPLL1LCKR) and will generate an interrupt request if INTENSET.DPLL1LCKR is '1'.

Writing a zero to this bit has no effect.

Writing a '1' to this bit clears the DPLL1 Lock Rise interrupt flag.

**Bit 19 – DPLL0LDRTO** DPLL0 Loop Divider Ratio Update Complete
This flag is cleared by writing a '1' to it.

This flag is set on a zero-to-one transition of the DPLL0 Loop Divider Ratio Update Complete bit in the Status register (STATUS.DPLL0LDRTO) and will generate an interrupt request if INTENSET.DPLL0LDRTO is '1'.

Writing a zero to this bit has no effect.

Writing a '1' to this bit clears the DPLL0 Loop Divider Ratio Update Complete interrupt flag.

**Bit 18 – DPLL0LTO** DPLL0 Lock Timeout
This flag is cleared by writing a '1' to it.

This flag is set on a zero-to-one transition of the DPLL0 Lock Timeout bit in the Status register (STATUS.DPLL0LTO) and will generate an interrupt request if INTENSET.DPLL0LTO is '1'.

Writing a zero to this bit has no effect.

Writing a '1' to this bit clears the DPLL0 Lock Timeout interrupt flag.

**Bit 17 – DPLL0LCKF** DPLL0 Lock Fall
This flag is cleared by writing a '1' to it.

This flag is set on a zero-to-one transition of the DPLL0 Lock Fall bit in the Status register (STATUS.DPLL0LCKF) and will generate an interrupt request if INTENSET.DPLL0LCKF is '1'.

Writing a zero to this bit has no effect.

Writing a '1' to this bit clears the DPLL0 Lock Fall interrupt flag.

**Bit 16 – DPLL0LCKR** DPLL0 Lock Rise
This flag is cleared by writing a '1' to it.

This flag is set on a zero-to-one transition of the DPLL0 Lock Rise bit in the Status register (STATUS.DPLL0LCKR) and will generate an interrupt request if INTENSET.DPLL0LCKR is '1'.

Writing a zero to this bit has no effect.

Writing a '1' to this bit clears the DPLL0 Lock Rise interrupt flag.

**Bit 12 – DFLLRCS** DFLL Reference Clock Stopped
This flag is cleared by writing a '1' to it.

| USER*m* | User Multiplexer | Description | Path Type(1) |
|---|---|---|---|
| m = 55 | ADC0 START | ADC0 start conversion | A, S, R |
| m = 56 | ADC0 SYNC | Flush ADC0 | A, S, R |
| m = 57 | ADC1 START | ADC1 start conversion | A, S, R |
| m = 58 | ADC1 SYNC | Flush ADC1 | A, S, R |
| m = 59..60 | AC_SOC 0..1 | AC SOC x | A |
| m = 61..62 | DAC_START0..1 | DAC0..1 start conversion | A |
| m = 63..66 | CCL_LUTIN 0..3 | CCL input | A |
| others | Reserved | - | - |

1) A = Asynchronous path, S = Synchronous path, R = Resynchronized path

| Value | Description |
|---|---|
| 0x00 | No channel selected |
| 0x01 | Channel 0 selected |
| 0x02 | Channel 1 selected |
| 0x03 | Channel 2 selected |
| 0x04 | Channel 3 selected |
| 0x05 | Channel 4 selected |
| 0x06 | Channel 5 selected |
| 0x07 | Channel 6 selected |
| 0x08 | Channel 7 selected |
| 0x09 | Channel 8 selected |
| 0x0A | Channel 9 selected |
| 0x0B | Channel 10 selected |
| 0x0C | Channel 11 selected |
| 0x0D | Channel 12 selected |
| 0x0E | Channel 13 selected |
| 0x0F | Channel 14 selected |
| 0x10 | Channel 15 selected |
| 0x11 | Channel 16 selected |
| 0x12 | Channel 17 selected |
| 0x13 | Channel 18 selected |
| 0x14 | Channel 19 selected |
| 0x15 | Channel 20 selected |
| 0x16 | Channel 21 selected |
| 0x17 | Channel 22 selected |
| 0x18 | Channel 23 selected |
| 0x19 | Channel 24 selected |
| 0x1A | Channel 25 selected |
| 0x1B | Channel 26 selected |
| 0x1C | Channel 27 selected |
| 0x1D | Channel 28 selected |
| 0x1E | Channel 29 selected |

# 32. PORT - I/O Pin Controller

## 32.1 Overview

The IO Pin Controller (PORT) controls the I/O pins of the device. The I/O pins are organized in a series of groups, collectively referred to as a PORT group. Each PORT group can have up to 32 pins that can be configured and controlled individually or as a group. The number of PORT groups on a device may depend on the package/number of pins. Each pin may either be used for general-purpose I/O under direct application control or be assigned to an embedded device peripheral. When used for general-purpose I/O, each pin can be configured as input or output, with highly configurable driver and pull settings.

All I/O pins have true read-modify-write functionality when used for general-purpose I/O; the direction or the output value of one or more pins may be changed (set, reset or toggled) explicitly without unintentionally changing the state of any other pins in the same port group by a single, atomic 8-, 16- or 32-bit write.

The PORT is connected to the high-speed bus matrix through an AHB/APB bridge.

## 32.2 Features

- Selectable input and output configuration for each individual pin
- Software-controlled multiplexing of peripheral functions on I/O pins
- Flexible pin configuration through a dedicated Pin Configuration register
- Configurable output driver and pull settings:
  - Totem-pole (push-pull)
  - Pull configuration
  - Driver strength
- Configurable input buffer and pull settings:
  - Internal pull-up or pull-down
  - Input sampling criteria
  - Input buffer can be disabled if not needed for lower power consumption
- Input event:
  - Up to four input event pins for each PORT group
  - SET/CLEAR/TOGGLE event actions for each event input on output value of a pin
  - Can be output to pin

- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

#### 39.6.6.7 Transmit Cancellation

The CAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer the CPU has to write a '1' to the corresponding bit position (=number of Tx Buffer) of register TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register TXBCF to '1'.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

**Note:** In case a pending transmission is canceled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

#### 39.6.6.8 Tx Event Handling

To support Tx event handling the CAN has implemented a Tx Event FIFO. After the CAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in 39.9.4 Tx Event FIFO Element.

When a Tx Event FIFO full condition is signaled by IR.TEFF, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag IR.TEFL is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by TXEFC.EFWM, interrupt flag IR.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index TXEFS.EFGI has to be added to the Tx Event FIFO start address TXEFC.EFSA.

### 39.6.7 FIFO Acknowledge Handling

The Get Indexes of Rx FIFO 0, Rx FIFO 1 and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (refer to 39.8.29 RXF0A, 39.8.33 RXF1A and 39.8.47 TXEFA). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.

When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO's Get Index.

Due to the fact that the CPU has free access to the CAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when

### 39.8.10 Timestamp Counter Value

**Name:** TSCV
**Offset:** 0x24
**Reset:** 0x00000000
**Property:** Read-only

**Note:**

1. A write access to TSCV while in internal mode clears the Timestamp Counter value. A write access to TSCV while in external mode has no impact.
2. A “wrap around” is a change of the Timestamp Counter value from non-zero to zero not caused by the write access to TSCV.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | |
| Reset | | | | | | | | |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | |
| Reset | | | | | | | | |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | TSC[14:8] | | | | | | |
| Access | | R | R | R | R | R | R | R |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TSC[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 14:0 – TSC[14:0]** Timestamp Counter
The internal Timestamp Counter value is captured on start of frame (both Rx and Tx). When TSCC.TSS = 0x1, the Timestamp Counter is incremented in multiples of CAN bit times [1...16] depending on the configuration of TSCC.TCP. A wrap around sets interrupt flag IR.TSW.

**Bit 14 – PXE** Protocol Exception Event
This field is cleared on read access.

| Value | Description |
|---|---|
| 0 | No protocol exception event occurred since last read access. |
| 1 | Protocol exception event occurred. |

**Bit 13 – RFDF** Received a CAN FD Message
This field is cleared on read access.

| Value | Description |
|---|---|
| 0 | Since this bit was reset by the CPU, no CAN FD message has been received. |
| 1 | Message in CAN FD format with FDF flag set has been received. This bit is set independent of acceptance filtering. |

**Bit 12 – RBRS** BRS flag of last received CAN FD Message
This field is cleared on read access.

| Value | Description |
|---|---|
| 0 | Last received CAN FD message did not have its BRS flag set. |
| 1 | Last received CAN FD message had its BRS flag set. This bit is set together with RFDF, independent of acceptance filtering. |

**Bit 11 – RESI** ESI flag of last received CAN FD Message
This field is cleared on read access.

| Value | Description |
|---|---|
| 0 | Last received CAN FD message did not have its ESI flag set. |
| 1 | Last received CAN FD message had its ESI flag set. |

**Bits 10:8 – DLEC[2:0]** Data Last Error Code
Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error.

**Bit 7 – BO** Bus_Off Status

| Value | Description |
|---|---|
| 0 | The CAN is not Bus_Off. |
| 1 | The CAN is in Bus_Off state. |

**Bit 6 – EW** Error Warning Status

| Value | Description |
|---|---|
| 0 | Both error counters are below the Error_Warning limit of 96. |
| 1 | At least one of the error counter has reached the Error_Warning limit of 96. |

**Bit 5 – EP** Error Passive

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | The Command Index Check is disabled. |
| 1 | ENABLED | The Command Index Check is enabled. |

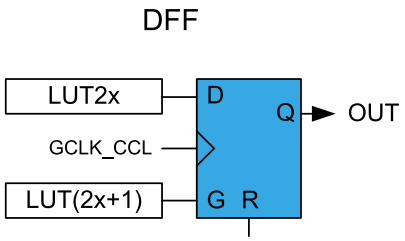
**Bit 3 – CMDCCEN** Command CRC Check Enable
If this bit is set to 1, the peripheral checks the CRC field in the response. If an error is detected, it is reported as a Command CRC Error (CMDCRC) in EISTR. If this bit is set to 0, the CRC field is not checked. The position of the CRC field is determined according to the length of the response.

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | The Command CRC Check is disabled. |
| 1 | ENABLED | The Command CRC Check is enabled. |

**Bits 1:0 – RESPTYP[1:0]** Response Type
This field is set according to the response type expected for the command index (CMDIDX).

| Value | Name | Description |
|---|---|---|
| 0 | NORESP | No Response |
| 1 | RL136 | Response Length 136 |
| 2 | RL48 | Response Length 48 |
| 3 | RL48BUSY | Response Length 48 with Busy |

**Figure 41-14. D Flip Flop**




When the even LUT is disabled (LUTCTRL0.ENABLE=0 / LUTCTRL2.ENABLE=0), the flip-flop is asynchronously cleared. The reset command (R) is kept enabled for one APB clock cycle. In all other cases, the flip-flop output (OUT) is refreshed on rising edge of the GCLK_CCL, as shown in Table 41-2.

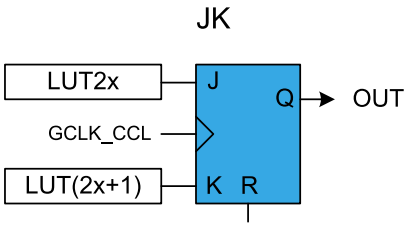
**Table 41-2. DFF Characteristics**

| R | G | D | OUT |
|---|---|---|---|
| 1 | X | X | Clear |
| 0 | 1 | 1 | Set |
| | | 0 | Clear |
| | 0 | X | Hold state (no change) |

**JK Flip-Flop (JK)**

When this configuration is selected, the J-input is driven by the even LUT output (LUT0 and LUT2), and the K-input is driven by the odd LUT output (LUT1 and LUT3), as shown in Figure 41-15.

**Figure 41-15. JK Flip Flop**




When the even LUT is disabled (LUTCTRL0.ENABLE=0 / LUTCTRL2.ENABLE=0), the flip-flop is asynchronously cleared. The reset command (R) is kept enabled for one APB clock cycle. In all other cases, the flip-flop output (OUT) is refreshed on rising edge of the GCLK_CCL, as shown in Table 41-3.

**Table 41-3. JK Characteristics**

| R | J | K | OUT |
|---|---|---|---|
| 1 | X | X | Clear |
| 0 | 0 | 0 | Hold state (no change) |
| 0 | 0 | 1 | Clear |
| 0 | 1 | 0 | Set |
| 0 | 1 | 1 | Toggle |

There is no difference on the final result when using any of the options for this service. The choice has to be made according to the available resources (RAM, Time) and also taking into account the expected security level.

For this service, two exclusive Calculus Modes are possible. The following table describes the Calculus Mode Options.

**Table 43-51. ExpMod Service Calculus Mode Option**

| Option | Explanation |
|---|---|
| PUKCL_EXPMOD_FASTRSA | Performs a Fast computation |
| PUKCL_EXPMOD_REGULARRSA | Performs a Regular computation, slower than the Fast version, but using Regular calculus methods |

For this service, four window sizes are possible. The window size in bits is those of the windowing method used for the exponent.

The choice of the window size is a balance between the size of the parameters and the computation time:

- Increasing the window size increases the precomputation workspace.
- Increasing the window size reduces the computation time (may not be relevant for very small exponents).

The following table details the size of the precomputation workspace, depending on the chosen window size option.

**Table 43-52. ExpMode Service Window Size Options and Precomputation Space Size**

| Option specified | Size of the PrecompBase Workspace (bytes) | Content of the Workspace |
|---|---|---|
| PUKCL_EXPMOD_WINDOWSIZE_1 | 3*(u2ModLength + 4) + 8 | x |
| PUKCL_EXPMOD_WINDOWSIZE_2 | 4*(u2ModLength + 4) + 8 | $x\ x^3$ |
| PUKCL_EXPMOD_WINDOWSIZE_3 | 6*(u2ModLength + 4) + 8 | $x\ x^3\ x^5\ x^7$ |
| PUKCL_EXPMOD_WINDOWSIZE_4 | 10*(u2ModLength + 4) + 8 | $x\ x^3\ x^5\ x^7\ x^9\ x^{11}\ x^{13}\ x^{15}$ |

The exponent can be located in RAM or in the data space. If one part of the exponent is in Crypto RAM this must be mandatory signaled by using the option PUKCL_EXPMOD_EXPINPUKCCRAM.

The following table describes this option.

**Table 43-53. ExpMod Service Exponent in Crypto RAM Option**

| Option | Purpose |
|---|---|
| PUKCL_EXPMOD_EXPINPUKCCRAM | The exponent can be read from any data space of memory, including Flash, RAM or even Crypto RAM. When at least one word the exponent is in Crypto RAM, this option has to be set. |

#### 43.3.5.2.6 Code Example

```
PUKCL_PARAM PUKCLParam;
PPUKCL_PARAM pvPUKCLParam = &PUKCLParam;


PUKCL(u2Option) =...;
```

#### 43.3.6.11.6 Constraints

No overlapping between either input and output are allowed. The following conditions must be avoided to ensure that the service works correctly:

- nu1ModBase, nu1CnsBase, nu1PointABase, nu1PrivateKey, nu1ScalarNumber, nu1OrderPointBase,nu1ABase, nu1Workspace or nu1HashBase are not aligned on 32-bit boundaries
- {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength + 8}, {nu1PointABase, 3*u2ModLength+ 12},{nu1PrivateKey, u2ScalarLength + 4},{nu1ScalarNumber, u2ScalarLength + 4},{nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} or {nu1HashBase, u2ScalarLength + 4} are not in Crypto RAM
- u2ModLength is either: < 12, > 0xffc or not a 32-bit length
- All overlapping between {nu1ModBase, u2ModLength + 4}, {nu1CnsBase, u2ModLength +8}, {nu1PointABase, 3*u2ModLength + 12}, {nu1PrivateKey, u2ScalarLength + 4}, {nu1ScalarNumber, u2ScalarLength + 4}, {nu1OrderPointBase, u2ScalarLength + 4}, {nu1ABase, u2ModLength + 4}, {nu1Workspace, <WorkspaceLength>} and {nu1HashBase, u2ScalarLength + 4}

#### 43.3.6.11.7 Status Returned Values

**Table 43-86. ZpEcDsaGenerateFast Service Return Codes**

| Returned Status | Importance | Meaning |
|---|---|---|
| PUKCL_OK | – | The computation passed without problem. The signature is the good one. |
| PUKCL_WRONG_SELECTNUMBER | Warning | The given value for nu1ScalarNumber is not good to perform this signature generation. |

### 43.3.6.12 Verifying an ECDSA Signature (Compliant with FIPS186-2)

#### 43.3.6.12.1 Purpose

This service is used to verify an ECDSA signature following the FIPS 186-2. It performs the second step of the Signature Verification.

A hash value (HashVal) must be provided as input, it has to be previously computed from the message to be signed using a secure hash algorithm.

As second significant input, the Signature is provided to be checked. This service checks the signature and fills the status accordingly.

#### 43.3.6.12.2 How to Use the Service

#### 43.3.6.12.3 Description

The operation performed is:

*Verify* = *EcDsaVerifySignature*($Pt_A$, *HashVal*, *Signature*, *CurveParameters*, *PublicKey*)

The points used for this operation are represented in different coordinate systems. In this computation, the following parameters need to be provided:

- A the input point is filled with the affine values (X,Y) and Z = 1 (pointed by{nu1PointABase, 3*u2ModLength + 12})
- Cns the working space for the Fast Modular Constant not initialized (pointed by {nu1CnsBase,u2ScalarLength + 8})
- P the modulus filled (pointed by {nu1ModBase,u2ModLength + 4})

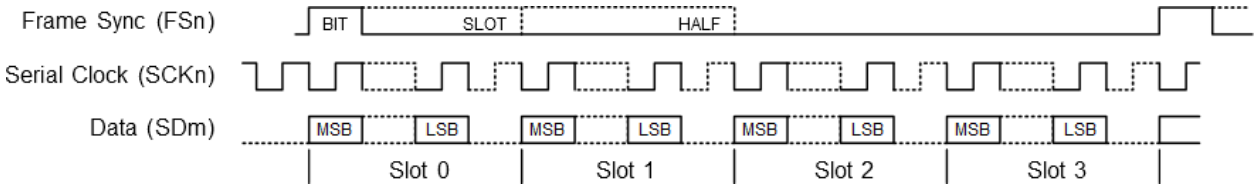
**Bits 4:0 – MUXPOS[4:0]** Positive MUX Input Selection
These bits define the MUX selection for the positive ADC input. If the internal bandgap voltage or temperature sensor input channel is selected, then the Sampling Time Length bit group in the Sampling Control register must be written with a corresponding value.

| Value | Name | Description |
|---|---|---|
| 0x00 | AIN0 | ADC AIN0 pin |
| 0x01 | AIN1 | ADC AIN1 pin |
| 0x02 | AIN2 | ADC AIN2 pin |
| 0x03 | AIN3 | ADC AIN3 pin |
| 0x04 | AIN4 | ADC AIN4 pin |
| 0x05 | AIN5 | ADC AIN5 pin |
| 0x06 | AIN6 | ADC AIN6 pin |
| 0x07 | AIN7 | ADC AIN7 pin |
| 0x08 | AIN8 | ADC AIN8 pin |
| 0x09 | AIN9 | ADC AIN9 pin |
| 0x0A | AIN10 | ADC AIN10 pin |
| 0x0B | AIN11 | ADC AIN11 pin |
| 0x0C | AIN12 | ADC AIN12 pin |
| 0x0D | AIN13 | ADC AIN13 pin |
| 0x0E | AIN14 | ADC AIN14 pin |
| 0x0F | AIN15 | ADC AIN15 pin |
| 0x10 | AIN16 | ADC AIN16 pin |
| 0x11 | AIN17 | ADC AIN17 pin |
| 0x12 | AIN18 | ADC AIN18 pin |
| 0x13 | AIN19 | ADC AIN19 pin |
| 0x14 | AIN20 | ADC AIN20 pin |
| 0x15 | AIN21 | ADC AIN21 pin |
| 0x16 | AIN22 | ADC AIN22 pin |
| 0x17 | AIN23 | ADC AIN23 pin |
| 0x18 | SCALEDCOREVCC | 1/4 Scaled Core Supply |
| 0x19 | SCALEDVBAT | 1/4 Scaled VBAT Supply |
| 0x1A | SCALEDIOVCC | 1/4 Scaled I/O Supply |
| 0x1B | BANDGAP | Bandgap Voltage |
| 0x1C | PTAT | Temperature Sensor |
| 0x1D | CTAT | Temperature Sensor |
| 0x1E | DAC | DAC Output |

| CTRLA.RESOLUTION | Bits [n:0] |
|---|---|
| 0x0 - NONE | - |
| 0x1 - DITH4 | 3:0 |
| 0x2 - DITH5 | 4:0 |
| 0x3 - DITH6 | 5:0 (depicted) |

By configuring RXCTRL and/or TXCTRL, data bits can be left-adjusted or right-adjusted in the slot. It can also configure the data transmission/reception with either the MSB or the LSB transmitted/received first and starting the transmission/reception either at the transition of the FSn pin or one clock period after.

**Figure 51-6. TDM Format Reception and Transmission Sequence**




Data bits are sent on the falling edge of the Serial Clock and sampled on the rising edge of the Serial Clock. The FSn pin provides a frame synchronization signal, at the beginning of slot 0. The delay between the frame start and the first data bit is defined by writing the CLKCTRLn.BITDELAY field.

The Frame Sync pulse can be either one SCKn period (BIT), one slot (SLOT), or one half frame (HALF). This selection is done by writing the CLKCTRLn.FSWIDTH field.

The number of slots is selected by writing the CLKCTRLn.NBSLOTS field.

The number of bits in each slot is selected by writing the CLKCTRLn.SLOTSIZE field.

The length of transmitted words can be chosen among 8, 16, 18, 20, 24, and 32 bits by writing the DATASIZE field in the Serializer Control register (RXCTRL and/or TXCTRL).

If the slot allows more data bits than the number of bits specified in the RXCTRL. and/or TXCTRL.DATASIZE bit field, additional bits are appended to the transmitted or received data word as specified in the RXCTRL. and/or TXCTRL.EXTEND bit field. If the slot allows less data bits than programmed, the extra bits are not transmitted, or received data word is extended based on the EXTEND field value.

### 51.6.6 PDM Reception

In Pulse Density Modulation (PDM) reception mode, continuous 1-bit data samples are available on the SDI line on each SCKn rising edge, e.g. by a MEMS microphone with PDM interface. When using two channel PDM microphones, the second one (right channel) is configured to output data on each SCKn falling edge.

For one PDM microphone, the $I^2S$ controller should be configured in normal Receive mode with one slot and 16- or 32-bit data size, so that 16 or 32 samples of the microphone are stored into each data word.

For two PDM microphones, the $I^2S$ controller should be configured in PDM2 mode with one slot and 32-bit data size. The Rx Serializer will store 16 samples of each microphone in one half of the data word, with left microphone bits in lower half and right microphone bits in upper half, like in compact stereo format.

Based on oversampling frequency requirement from PDM microphone, the SCKn frequency must be configured in the $I^2S$ controller.

> A microphone that requires a sampling frequency of *fs* = 48 kHz and an oversampling frequency of *fo*=64 × *fs* would require an SCKn frequency of 3.072 MHz.

After selecting a proper frequency for GCLK_I2S_n and according Master Clock Division Factor in the Clock Unit n Control register (CLKCTRLn.MCKDIV), SCKn must be selected as per required frequency.

In PDM mode, only the clock and data line (SCKn and SDIn) pins are used.