**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®-M7 |
| Core Size | 32-Bit Single-Core |
| Speed | 300MHz |
| Connectivity | CANbus, Ethernet, I²C, IrDA, LINbus, MMC/SD/SDIO, QSPI, SPI, SSC, UART/USART, USB |
| Peripherals | Brown-out Detect/Reset, DMA, I²S, POR, PWM, WDT |
| Number of I/O | 75 |
| Program Memory Size | 1MB (1M x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 384K x 8 |
| Voltage - Supply (Vcc/Vdd) | 3V ~ 3.6V |
| Data Converters | A/D 10x12b; D/A 2x12b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 105°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 100-LQFP |
| Supplier Device Package | 100-LQFP (14x14) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atsamv71n20b-aabt |

#### 26.4.3.2 Backup Reset

A backup reset occurs when the chip exits from Backup mode. While exiting Backup mode, the vddcore_nreset signal is asserted by the Supply Controller.

Field RSTC_SR.RSTTYP is updated to report a backup reset.

#### 26.4.3.3 Watchdog Reset

The watchdog reset is entered when a watchdog fault occurs. This reset lasts three SLCK cycles.

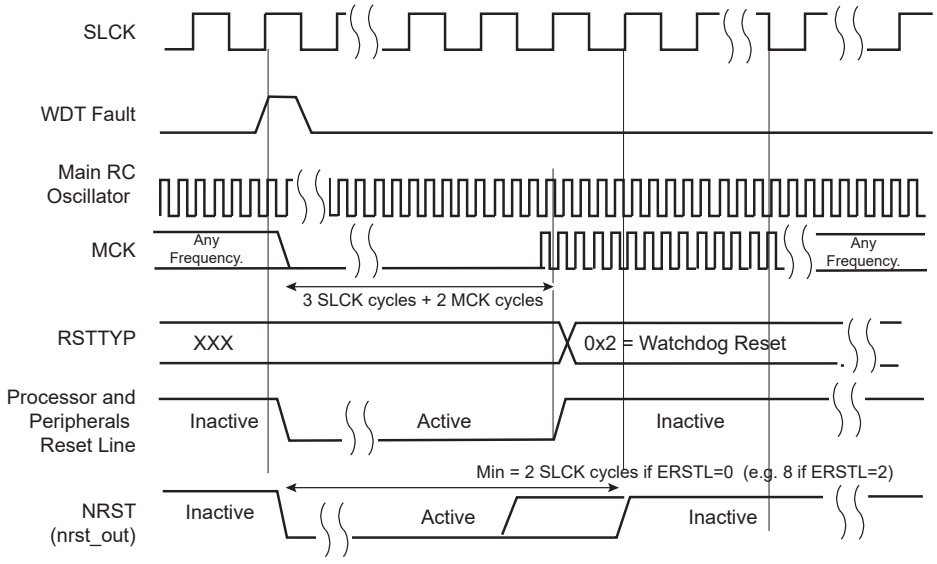
When in watchdog reset, the processor reset and the peripheral reset are asserted. The NRST line is also asserted, depending on the value of RSTC_MR.ERSTL. However, the resulting low level on NRST does not result in a user reset state.

The Watchdog Timer is reset by the proc_nreset signal. As the watchdog fault always causes a processor reset if WDT_MR.WDRSTEN is written to ‘1’, the Watchdog Timer is always reset after a watchdog reset, and the Watchdog is enabled by default and with a period set to a maximum.

When WDT_MR.WDRSTEN is written to ‘0’, the watchdog fault has no impact on the RSTC.

After a watchdog overflow occurs, the report on the RSTC_SR.RSTTYP may differ (either WDT_RST or USER_RST) depending on the external components driving the NRST pin. For example, if the NRST line is driven through a resistor and a capacitor (NRST pin debouncer), the reported value is USER_RST if the low to high transition is greater than one SLCK cycle.

**Figure 26-4. Watchdog Reset Timing Diagram**




#### 26.4.3.4 Software Reset

The RSTC offers commands to assert the different reset signals. These commands are performed by writing the Control register (RSTC_CR) with the following bits at ‘1’:

- RSTC_CR.PROCRST: Writing a ‘1’ to PROCRST resets the processor and all the embedded peripherals, including the memory system and, in particular, the Remap Command.
- RSTC_CR.EXTRST: Writing a ‘1’ to EXTRST asserts low the NRST pin during a time defined by the field RSTC_MR.ERSTL.

The software reset is entered if at least one of these bits is written to ‘1’ by the software. All these commands can be performed independently or simultaneously. The software reset lasts three SLCK cycles.

### 31.20.12 PMC USB Clock Register

**Name:** PMC_USB
**Offset:** 0x0038
**Reset:** 0x00000000
**Property:** Read/Write

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

| 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 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | USBDIV[3:0] | | | |
| Reset | | | | | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | USBS |
| Reset | | | | | | | | 0 |

**Bits 11:8 – USBDIV[3:0]** Divider for USB_48M
USB_48M is input clock divided by USBDIV+1.

**Bit 0 – USBS** USB Input Clock Selection

| Value | Description |
|---|---|
| 0 | USB_48M input is PLLA. |
| 1 | USB_48M input is UPLL. |

**Figure 35-32. Read/Write Cycles in Slow Clock Mode**

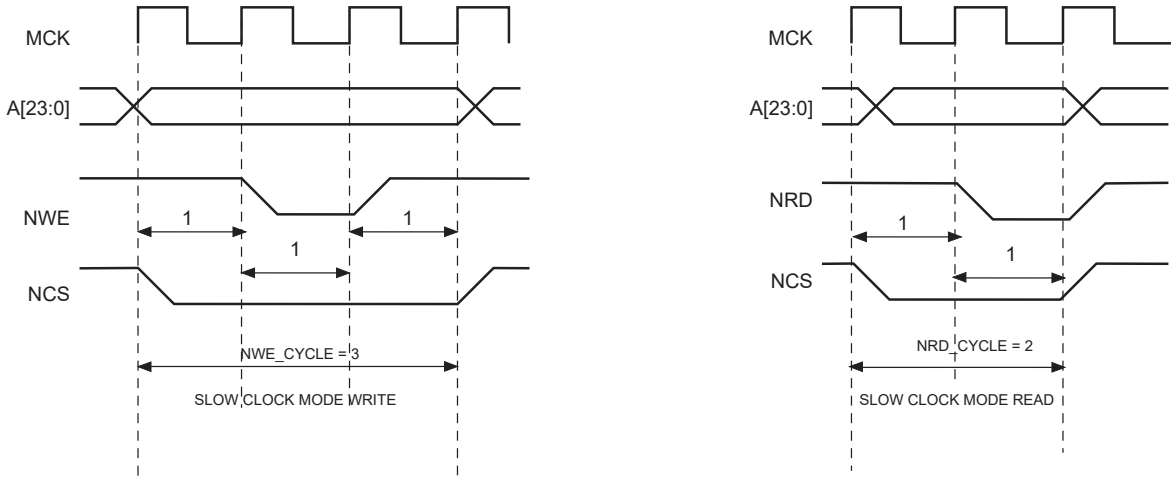



**Table 35-6. Read and Write Timing Parameters in Slow Clock Mode**

| Read Parameters | Duration (cycles) | Write Parameters | Duration (cycles) |
|---|---|---|---|
| NRD_SETUP | 1 | NWE_SETUP | 1 |
| NRD_PULSE | 1 | NWE_PULSE | 1 |
| NCS_RD_SETUP | 0 | NCS_WR_SETUP | 0 |
| NCS_RD_PULSE | 2 | NCS_WR_PULSE | 3 |
| NRD_CYCLE | 2 | NWE_CYCLE | 3 |

### 35.14.2 Switching from (to) Slow Clock Mode to (from) Normal Mode

When switching from Slow clock mode to Normal mode, the current Slow clock mode transfer is completed at a high clock rate, with the set of Slow clock mode parameters (see Figure 35-33). The external device may not be fast enough to support such timings.

Figure 35-34 illustrates the recommended procedure to switch from one mode to the other.

The destination address of received frames is compared against the data stored in the Specific Address registers once they have been activated. The addresses are deactivated at reset or when their corresponding Specific Address register Bottom is written. They are activated when Specific Address register Top is written. If a receive frame address matches an active address, the frame is written to the FIFO interface and on to DMA memory.

Frames may be filtered using the type ID field for matching. Four type ID registers exist in the register address space and each can be enabled for matching by writing a one to the MSB (bit 31) of the respective register. When a frame is received, the matching is implemented as an OR function of the various types of match.

The contents of each type ID register (when enabled) are compared against the length/type ID of the frame being received (e.g., bytes 13 and 14 in non-VLAN and non-SNAP encapsulated frames) and copied to memory if a match is found. The encoded type ID match bits (Word 0, Bit 22 and Bit 23) in the receive buffer descriptor status are set indicating which type ID register generated the match, if the receive checksum offload is disabled.

The reset state of the type ID registers is zero, hence each is initially disabled.

The following example illustrates the use of the address and type ID match registers for a MAC address of 21:43:65:87:A9:CB:

| | |
|---|---|
| Preamble | 55 |
| SFD | D5 |
| DA (Octet 0 - LSB) | 21 |
| DA (Octet 1) | 43 |
| DA (Octet 2) | 65 |
| DA (Octet 3) | 87 |
| DA (Octet 4) | A9 |
| DA (Octet 5 - MSB) | CB |
| SA (LSB) | 00 (see **Note**) |
| SA | 00(see **Note**) |
| SA | 00(see **Note**) |
| SA | 00(see **Note**) |
| SA | 00(see **Note**) |
| SA (MSB) | 00(see **Note**) |
| Type ID (MSB) | 43 |
| Type ID (LSB) | 21 |

**Note:** Contains the address of the transmitting device.

The previous sequence shows the beginning of an Ethernet frame. Byte order of transmission is from top to bottom, as shown. For a successful match to specific address 1, the following address matching registers must be set up:

| Value | Description |
|---|---|
| 0 | No channel register is loaded after the end of the channel transfer. |
| 1 | The channel controller loads the next descriptor after the end of the current transfer, i.e., when the USBHS_DEVDMASTATUS.CHANN_ENB bit is reset. |

**Bit 0 – CHANN_ENB** Channel Enable Command

| Value | Description |
|---|---|
| 0 | The DMA channel is disabled at end of transfer and no transfer occurs upon request. This bit is also cleared by hardware when the channel source bus is disabled at end of buffer.<br>If the LDNXT_DSC bit has been cleared by descriptor loading, the firmware must set the corresponding CHANN_ENB bit to start the described transfer, if needed.<br>If the LDNXT_DSC bit is cleared, the channel is frozen and the channel registers may then be read and/or written reliably as soon as both USBHS_DEVDMASTATUS.CHANN_ENB and CHANN_ACT flags read as 0.<br>If a channel request is currently serviced when this bit is cleared, the DMA FIFO buffer is drained until it is empty, then the USBHS_DEVDMASTATUS.CHANN_ENB bit is cleared.<br>If the LDNXT_DSC bit is set at or after this bit clearing, then the currently loaded descriptor is skipped (no data transfer occurs) and the next descriptor is immediately loaded. |
| 1 | The USBHS_DEVDMASTATUS.CHANN_ENB bit is set, thus enabling the DMA channel data transfer. Then, any pending request starts the transfer. This may be used to start or resume any requested transfer. |

### 39.6.58 Host Pipe x Disable Register (Interrupt Pipes)

**Name:** USBHS_HSTPIPIDRx (INTPIPES)
**Offset:** 0x0620 + x*0x04 [x=0..9]
**Reset:** 0
**Property:** Read/Write

This register view is relevant only if PTYPE = 0x3 in "Host Pipe x Configuration Register".

For additional information, see "Host Pipe x Mask Register (Interrupt Pipes)".

This register always reads as zero.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Clears the corresponding bit in USBHS_HSTPIPIMRx.

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

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | PFREEZEC | PDISHDMAC |
| Access | | | | | | | | |
| Reset | | | | | | | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | FIFOCONC | | NBUSYBKEC | | | | |
| Access | | | | | | | | |
| Reset | | 0 | | 0 | | | | |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SHORTPACKETIEC | RXSTALLDEC | OVERFIEC | NAKEDEC | PERREC | UNDERFIEC | TXOUTEC | RXINEC |
| Access | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 17 – PFREEZEC** Pipe Freeze Disable

**Bit 16 – PDISHDMAC** Pipe Interrupts Disable HDMA Request Disable

**Bit 14 – FIFOCONC** FIFO Control Disable

**Bit 12 – NBUSYBKEC** Number of Busy Banks Disable

**Bit 7 – SHORTPACKETIEC** Short Packet Interrupt Disable

**Bit 6 – RXSTALLDEC** Received STALLed Interrupt Disable

# 40. High-Speed Multimedia Card Interface (HSMCI)

## 40.1 Description

The High Speed Multimedia Card Interface (HSMCI) supports the MultiMedia Card (MMC) Specification V4.3, the SD Memory Card Specification V2.0, the SDIO V2.0 specification and CE-ATA V1.1.

The HSMCI includes a command register, response registers, data registers, timeout counters and error detection logic that automatically handle the transmission of commands and, when required, the reception of the associated responses and data with a limited processor overhead.

The HSMCI operates at a rate of up to Master Clock divided by 2 and supports the interfacing of 1 slot(s). Each slot may be used to interface with a High Speed MultiMedia Card bus (up to 30 Cards) or with an SD Memory Card. A bit field in the SD Card Register performs this selection.

The SD Memory Card communication is based on a 9-pin interface (clock, command, four data and three power lines) and the High Speed MultiMedia Card on a 7-pin interface (clock, command, one data, three power lines and one reserved for future use).

The SD Memory Card interface also supports High Speed MultiMedia Card operations. The main differences between SD and High Speed MultiMedia Cards are the initialization process and the bus topology.

HSMCI fully supports CE-ATA Revision 1.1, built on the MMC System Specification v4.0. The module includes dedicated hardware to issue the command completion signal and capture the host command completion signal disable.

## 40.2 Embedded Characteristics

- Compatible with MultiMedia Card Specification Version 4.3
- Compatible with SD Memory Card Specification Version 2.0
- Compatible with SDIO Specification Version 2.0
- Compatible with CE-ATA Specification 1.1
- Cards Clock Rate Up to Master Clock Divided by 2
- Boot Operation Mode Support
- High Speed Mode Support
- Embedded Power Management to Slow Down Clock Rate When Not Used
- Supports 1 Multiplexed Slot(s)
  - Each Slot for either a High Speed MultiMedia Card Bus (Up to 30 Cards) or an SD Memory Card
- Support for Stream, Block and Multi-block Data Read and Write
- – Minimizes Processor Intervention for Large Buffer Transfers
- Built in FIFO (from 16 to 256 bytes) with Large Memory Aperture Supporting Incremental Access
- Support for CE-ATA Completion Signal Disable Command
- Protection Against Unexpected Modification On-the-Fly of the Configuration Registers

### 40.14.15 HSMCI Interrupt Mask Register

**Name:** HSMCI_IMR
**Offset:** 0x4C
**Reset:** 0x0
**Property:** Read-only

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| | UNRE | OVRE | ACKRCVE | ACKRCV | XFRDONE | FIFOEMPTY | | BLKOVRE |
| Access | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | | 0 |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | CSTOE | DTOE | DCRCE | RTOE | RENDE | RCRCE | RDIRE | RINDE |
| Access | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | CSRCV | SDIOWAIT | | | | SDIOIRQA |
| Access | | | | | | | | |
| Reset | | | 0 | 0 | | | | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | NOTBUSY | DTIP | BLKE | TXRDY | RXRDY | CMDRDY |
| Access | | | | | | | | |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 31 – UNRE** Underrun Interrupt Mask

**Bit 30 – OVRE** Overrun Interrupt Mask

**Bit 29 – ACKRCVE** Boot Operation Acknowledge Error Interrupt Mask

**Bit 28 – ACKRCV** Boot Operation Acknowledge Received Interrupt Mask

**Bit 27 – XFRDONE** Transfer Done Interrupt Mask

**Bit 26 – FIFOEMPTY** FIFO Empty Interrupt Mask

**Bit 24 – BLKOVRE** DMA Block Overrun Error Interrupt Mask

**Bit 23 – CSTOE** Completion Signal Time-out Error Interrupt Mask

**Bit 22 – DTOE** Data Time-out Error Interrupt Mask

# 41. Serial Peripheral Interface (SPI)

## 41.1 Description

The Serial Peripheral Interface (SPI) circuit is a synchronous serial data link that provides communication with external devices in Master or Slave mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master”' which controls the data flow, while the other devices act as “slaves" which have data shifted into and out by the master. Different CPUs can take turn being masters (multiple master protocol, contrary to single master protocol where one CPU is always the master while all of the others are always slaves). One master can simultaneously shift data into multiple slaves. However, only one slave can drive its output to write data back to the master at any given time.

A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI)—This data line supplies the output data from the master shifted into the input(s) of the slave(s).
- Master In Slave Out (MISO)—This data line supplies the output data from a slave to the input of the master. There may be no more than one slave transmitting data during any particular transfer.
- Serial Clock (SPCK)—This control line is driven by the master and regulates the flow of the data bits. The master can transmit data at a variety of baud rates; there is one SPCK pulse for each bit that is transmitted.
- Slave Select (NSS)—This control line allows slaves to be turned on and off by hardware.

## 41.2 Embedded Characteristics

- Master or Slave Serial Peripheral Bus Interface
  - 8-bit to 16-bit programmable data length per chip select
  - Programmable phase and polarity per chip select
  - Programmable transfer delay between consecutive transfers and delay before SPI clock per chip select
  - Programmable delay between chip selects
  - Selectable mode fault detection
- Master Mode can Drive SPCK up to Peripheral Clock
- Master Mode Bit Rate can be Independent of the Processor/Peripheral Clock
- Slave Mode Operates on SPCK, Asynchronously with Core and Bus Clock
- Four Chip Selects with External Decoder Support Allow Communication with up to 15 Peripherals
- Communication with Serial External Devices Supported
  - Serial memories, such as DataFlash and 3-wire EEPROMs
  - Serial peripherals, such as ADCs, DACs, LCD controllers, CAN controllers and sensors
  - External coprocessors
- Connection to DMA Channel Capabilities, Optimizing Data Transfers

2. Configure the Master mode (DADR, CKDIV, MREAD = 1, etc.) or Slave mode.
3. Enable the DMA.
4. (Master Only) Write TWIHS_CR.START to start the transfer.
5. Wait for the DMA status flag indicating that the buffer transfer is complete.
6. Disable the DMA.
7. Wait for the RXRDY flag in the TWIHS_SR.
8. Set TWIHS_CR.STOP.
9. Read the penultimate character in TWIHS_RHR.
10. Wait for the RXRDY flag in the TWIHS_SR.
11. Read the last character in TWIHS_RHR.
12. (Only if peripheral clock must be disabled) Wait for the TXCOMP flag to be raised in TWIHS_SR.

#### 43.6.3.9 SMBus Mode

SMBus mode is enabled when a one is written to TWIHS_CR.SMBEN. SMBus mode operation is similar to I²C operation with the following exceptions:

- Only 7-bit addressing can be used.
- The SMBus standard describes a set of timeout values to ensure progress and throughput on the bus. These timeout values must be programmed into TWIHS_SMBTR.
- Transmissions can optionally include a CRC byte, called Packet Error Check (PEC).
- A set of addresses has been reserved for protocol handling, such as alert response address (ARA) and host header (HH) address. Address matching on these addresses can be enabled by configuring TWIHS_CR.

##### 43.6.3.9.1 Packet Error Checking

Each SMBus transfer can optionally end with a CRC byte, called the PEC byte. Writing a one to TWIHS_CR.PECEN enables automatic PEC handling in the current transfer. Transfers with and without PEC can be intermixed in the same system, since some slaves may not support PEC. The PEC LFSR is always updated on every bit transmitted or received, so that PEC handling on combined transfers is correct.

In Master Transmitter mode, the master calculates a PEC value and transmits it to the slave after all data bytes have been transmitted. Upon reception of this PEC byte, the slave compares it to the PEC value it has computed itself. If the values match, the data was received correctly, and the slave returns an ACK to the master. If the PEC values differ, data was corrupted, and the slave returns a NACK value. Some slaves may not be able to check the received PEC in time to return a NACK if an error occurred. In this case, the slave should always return an ACK after the PEC byte, and another method must be used to verify that the transmission was received correctly.

In Master Receiver mode, the slave calculates a PEC value and transmits it to the master after all data bytes have been transmitted. Upon reception of this PEC byte, the master compares it to the PEC value it has computed itself. If the values match, the data was received correctly. If the PEC values differ, data was corrupted, and TWIHS_SR.PECERR is set. In Master Receiver mode, the PEC byte is always followed by a NACK transmitted by the master, since it is the last byte in the transfer.

In combined transfers, the PECRQ bit should only be set in the last of the combined transfers.

Consider the following transfer:

S, ADR+W, COMMAND_BYTE, ACK, SR, ADR+R, DATA_BYTE, ACK, PEC_BYTE, NACK, P

See Read/Write Flowcharts for detailed flowcharts.

| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | STOP condition is sent just after completing the current byte transmission in Master Read mode.<br>• In single data byte master read, both START and STOP must be set.<br>• In multiple data bytes master read, the STOP must be set after the last data received but one.<br>• In Master Read mode, if a NACK bit is received, the STOP is automatically performed.<br>• In master data write operation, a STOP condition will be sent after the transmission of the current data is finished. |

**Bit 0 – START** Send a START Condition
This action is necessary when the TWIHS peripheral needs to read data from a slave. When configured in Master mode with a write operation, a frame is sent as soon as the user writes a character in the Transmit Holding Register (TWIHS_THR).

| Value | Description |
|---|---|
| 0 | No effect. |
| 1 | A frame beginning with a START bit is transmitted according to the features defined in the TWIHS Master Mode Register (TWIHS_MMR). |

### 45.8.10 I2SC Transmitter Holding Register

**Name:** I2SC_THR
**Offset:** 0x24
**Property:** Write-only

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| | THR[31:24] | | | | | | | |
| Access | W | W | W | W | W | W | W | W |
| Reset | – | – | – | – | – | – | – | – |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | THR[23:16] | | | | | | | |
| Access | W | W | W | W | W | W | W | W |
| Reset | – | – | – | – | – | – | – | – |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | THR[15:8] | | | | | | | |
| Access | W | W | W | W | W | W | W | W |
| Reset | – | – | – | – | – | – | – | – |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | THR[7:0] | | | | | | | |
| Access | W | W | W | W | W | W | W | W |
| Reset | – | – | – | – | – | – | – | – |

**Bits 31:0 – THR[31:0]** Transmitter Holding Register
Next data word to be transmitted after the current word if TXRDY is not set. If I2SC_MR.DATALENGTH specifies fewer than 32 bits, data is right-justified in the THR field.

| Value | Description |
|---|---|
| 0 | No LIN checksum error has been detected since the last RSTSTA. |
| 1 | A LIN checksum error has been detected since the last RSTSTA. |

**Bit 27 – LINIPE** LIN Identifier Parity Error (cleared by writing a one to bit US_CR.RSTSTA)

| Value | Description |
|---|---|
| 0 | No LIN identifier parity error has been detected since the last RSTSTA. |
| 1 | A LIN identifier parity error has been detected since the last RSTSTA. |

**Bit 26 – LINISFE** LIN Inconsistent Synch Field Error (cleared by writing a one to bit US_CR.RSTSTA)

| Value | Description |
|---|---|
| 0 | No LIN inconsistent synch field error has been detected since the last RSTSTA |
| 1 | The USART is configured as a slave node and a LIN Inconsistent synch field error has been detected since the last RSTSTA. |

**Bit 25 – LINBE** LIN Bit Error (cleared by writing a one to bit US_CR.RSTSTA)

| Value | Description |
|---|---|
| 0 | No bit error has been detected since the last RSTSTA. |
| 1 | A bit error has been detected since the last RSTSTA. |

**Bit 23 – LINBLS** LIN Bus Line Status

| Value | Description |
|---|---|
| 0 | LIN bus line is set to 0. |
| 1 | LIN bus line is set to 1. |

**Bit 15 – LINTC** LIN Transfer Completed (cleared by writing a one to bit US_CR.RSTSTA)

| Value | Description |
|---|---|
| 0 | The USART is idle or a LIN transfer is ongoing. |
| 1 | A LIN transfer has been completed since the last RSTSTA. |

**Bit 14 – LINID** LIN Identifier Sent or LIN Identifier Received (cleared by writing a one to bit US_CR.RSTSTA)
If USART operates in LIN Master mode (USART_MODE = 0xA):

If USART operates in LIN Slave mode (USART_MODE = 0xB):

| Value | Description |
|---|---|
| 0 | No LIN identifier has been sent since the last RSTSTA. |
| 1 | At least one LIN identifier has been sent since the last RSTSTA. |
| 0 | No LIN identifier has been received since the last RSTSTA. |
| 1 | At least one LIN identifier has been received since the last RSTSTA |

**Bit 13 – LINBK** LIN Break Sent or LIN Break Received (cleared by writing a one to bit US_CR.RSTSTA)
Applicable if USART operates in LIN master mode (USART_MODE = 0xA):

If USART operates in LIN Slave mode (USART_MODE = 0xB):

SFID2[10:9] decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence.

0: Store message in a Rx buffer

1: Debug Message A

2: Debug Message B

3: Debug Message C

SFID2[5:0] defines the index of the dedicated Rx Buffer element to which a matching message is stored.

#### 49.5.7.6 Extended Message ID Filter Element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address MCAN_XIDFC.FLESA plus two times the index of the filter element (0…63).

**Table 49-11. Extended Message ID Filter Element**

| | 31 | | | 24 | 23 | 16 | 15 | 8 | 7 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| F0 | EFEC [2:0] | | | EFID1[28:0] | | | | | | |
| F1 | EFT[1:0] | | – | EFID2[28:0] | | | | | | |

• F0 Bit 31:29 EFEC[2:0]: Extended Filter Element Configuration

All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = “100”, “101”, or “110”, a match sets the interrupt flag MCAN_IR.HPM and, if enabled, an interrupt is generated. In this case, register MCAN_HPMS is updated with the status of the priority match.

0: Disable filter element

1: Store in Rx FIFO 0 if filter matches

2: Store in Rx FIFO 1 if filter matches

3: Reject ID if filter matches

4: Set priority if filter matches

5: Set priority and store in FIFO 0 if filter matches

6: Set priority and store in FIFO 1 if filter matches

7: Store into Rx Buffer or as debug message, configuration of EFT[1:0] ignored

• F0 Bits 28:0 EFID1[28:0]: Extended Filter ID 1

First ID of extended ID filter element.

When filtering for Rx Buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only MCAN_XIDAM masking mechanism (see Extended Message ID Filtering) is used.

• F1 Bits 31:30 EFT[1:0]: Extended Filter Type

0: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID)

• when the transmission has been aborted due to lost arbitration.

• when an error occurred during frame transmission.

In DAR mode, all transmissions are automatically cancelled if they are not successful. The corresponding MCAN_TXBCF bit is set for all unsuccessful transmissions.

| Value | Description |
|---|---|
| 0 | No transmission request pending |
| 1 | Transmission request pending |

### 49.6.41 MCAN Transmit Buffer Transmission Occurred

**Name:** MCAN_TXBTO
**Offset:** 0xD8
**Reset:** 0x00000000
**Property:** Read-only

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| | TO31 | TO30 | TO29 | TO28 | TO27 | TO26 | TO25 | TO24 |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | TO23 | TO22 | TO21 | TO20 | TO19 | TO18 | TO17 | TO16 |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

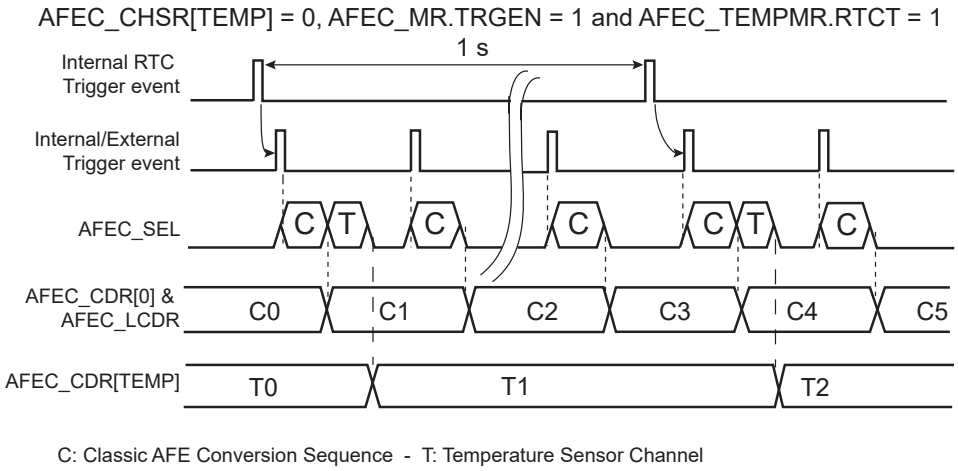
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | TO15 | TO14 | TO13 | TO12 | TO11 | TO10 | TO9 | TO8 |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TO7 | TO6 | TO5 | TO4 | TO3 | TO2 | TO1 | TO0 |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31 – TOx** Transmission Occurred for Buffer x
Each Transmit Buffer has its own Transmission Occurred bit. The bits are set when the corresponding MCAN_TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a ‘1’ to the corresponding bit of register MCAN_TXBAR.

| Value | Description |
|---|---|
| 0 | No transmission occurred. |
| 1 | Transmission occurred. |

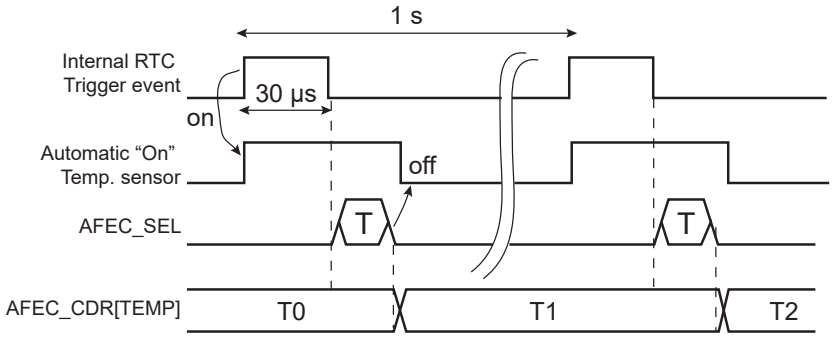
**Figure 52-9. Optimized Temperature Conversion Combined with Classical Conversions**




If RTCT is set and TRGEN is cleared, then all channels are disabled (AFEC_CHSR = 0) and only channel 11 is converted at a rate of one conversion per second.

This mode of operation, when combined with Sleep mode operation, provides a low-power mode for temperature measurement assuming there is no other AFE conversion to schedule at a higher sampling rate or no other channel to convert.

**Figure 52-10. Temperature Conversion Only**




Moreover, it is possible to raise a flag only if there is predefined change in the temperature measurement. The user can define a range of temperature or a threshold in AFEC_TEMPCWR and the mode of comparison in AFEC_TEMPMR. These values define the way the TEMPCHG flag will be raised in AFEC_ISR.

### 57.5.16 AES GCM H Word Register x

**Name:** AES_GCMHRx
**Offset:** 0x9C + x*0x04 [x=0..3]
**Reset:** 0x00000000
**Property:** R/W

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|---|
| | H[31:24] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | H[23:16] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

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

**Bits 31:0 – H[31:0]** GCM H Word x
The four 32-bit H Word registers contain the 128-bit GCM hash subkey *H*value.

Whenever a new key is written to the AES Key Register, two automatic actions are processed:

- GCM hash subkey *H* generation
- AES_GHASHRx Clear

If the application software requires a specific hash subkey, the automatically-generated H value can be overwritten in AES_GCMHRx. See Key Writing and Automatic Hash Subkey Calculation for details.

Generating a GCM hash subkey *H* by a write in AES_GCMHRx enables to:

- select the GCM hash subkey *H* for GHASH operations,
- select one operand to process a single GF128 multiply.

**Table 58-59. SMC Read Signals - NCS Controlled (READ_MODE = 0)**

| Symbol | VDDIO Supply | 1.8V Domain | 3.3V Domain | 1.8V Domain | 3.3V Domain | Unit |
|---|---|---|---|---|---|---|
| | Parameter | Min | | Max | | |
| NO HOLD Settings (NCS_RD_HOLD = 0) | | | | | | |
| $SMC_8$ | Data Setup before NCS High | 24.9 | 21.4 | – | – | ns |
| $SMC_9$ | Data Hold after NCS High | 0 | 0 | – | – | ns |
| HOLD Settings (NCS_RD_HOLD ≠ 0) | | | | | | |
| $SMC_{10}$ | Data Setup before NCS High | 13.4 | 11.7 | – | – | ns |
| $SMC_{11}$ | Data Hold after NCS High | 0 | 0 | – | – | ns |
| HOLD or NO HOLD Settings (NCS_RD_HOLD ≠ 0, NCS_RD_HOLD = 0) | | | | | | |
| $SMC_{12}$ | A0–A22 valid before NCS High | (NCS_RD_SETUP + NCS_RD_PULSE) × $t_{CPMCK}$ - 4.0 | (NCS_RD_SETUP + NCS_RD_PULSE) × $t_{CPMCK}$ - 3.9 | – | – | ns |
| $SMC_{13}$ | NRD low before NCS High | (NCS_RD_SETUP + NCS_RD_PULSE - NRD_SETUP) × $t_{CPMCK}$ - 2.8 | (NCS_RD_SETUP + NCS_RD_PULSE - NRD_SETUP) × $t_{CPMCK}$ - 4.2 | – | – | ns |
| $SMC_{14}$ | NCS Pulse Width | NCS_RD_PULSE length × $t_{CPMCK}$ - 0.9 | NCS_RD_PULSE length × $t_{CPMCK}$ - 0.2 | – | – | ns |

#### 58.13.1.9.2 Read Timings

**Table 58-60. SMC Read Signals - NRD Controlled (READ_MODE = 1)**

| Symbol | | VDDIO Supply | | Unit |
|---|---|---|---|---|
| | Parameter | Min | Max | |
| NO HOLD Settings (NRD_HOLD = 0) | | | | |
| $SMC_1$ | Data Setup before NRD High | 14.3 | – | ns |
| $SMC_2$ | Data Hold after NRD High | 0 | – | ns |
| HOLD Settings (NRD_HOLD ≠ 0) | | | | |
| $SMC_3$ | Data Setup before NRD High | 12.1 | – | ns |
| $SMC_4$ | Data Hold after NRD High | 0 | – | ns |
| HOLD or NO HOLD Settings (NRD_HOLD ≠ 0, NRD_HOLD = 0) | | | | |

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $t_S$ | Settling Time | Overdrive > 100 mV (ACC_ACR.ISEL = 0) | – | – | 1.5 | µs |
| | | Overdrive > 100 mV (ACC_ACR.ISEL = 1) | – | – | 0.15 | |

## 59.10 Temperature Sensor

The temperature sensor is connected to channel 11 of the AFE0.

The temperature sensor provides an output voltage ($V_{TEMP}$) that is proportional to absolute temperature (PTAT).

Improvement of the raw performance of the temperature sensor acquisition can be achieved by performing a single temperature point calibration to remove the initial inaccuracies ($V_{TEMP}$ and ADC offsets).

**Table 59-42. Temperature Sensor Characteristics**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{TEMP}$ | Output Voltage via AD11 | $T_A$ = 25°C | 0.64 | 0.72 | 0.8 | V |
| $dV_{TEMP}/dT$ | Temperature Sensitivity (Slope Voltage versus Temperature) | – | 2.06 | 2.33 | 2.60 | mV/°C |
| $t_S$ | VTEMP Settling Time | When $V_{TEMP}$ is sampled by the AFEC, the required track-and-hold time to ensure 1°C accurate settling | – | – | 1 | µs |
| – | Temperature Accuracy | After offset calibration over $T_A$ range [-40°C : +105°C] | -10 | – | 10 | °C |
| $t_{START}$ | Startup Time | – | – | – | 30 | µs |
| $I_{VDDIN}$ | Current Consumption | – | – | 130 | 270 | µA |

**Note:** AFE Gain Error and Offset error considered calibrated. This calibration at ambient temperature is not a feature of the product and is performed by the user's application.

## 59.11 12-bit DAC Characteristics

**Table 59-43. Analog Power Supply Characteristics**

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $I_{VDDIN}$ | Current Consumption | Sleep mode (Clock OFF) | – | 10 | – | µA |
| | | Normal mode with one output on,<br>DACC_ACR.IBCTLCHx =3 (see **Note 1**)<br>FS = 1 MSps, no $R_{LOAD}$, $V_{DDIN}$ = 3.3V | – | 200 | 800 | |