**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®-M4 |
| Core Size | 32-Bit Single-Core |
| Speed | 100MHz |
| Connectivity | I²C, IrDA, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, DMA, POR, PWM, WDT |
| Number of I/O | 79 |
| Program Memory Size | 512KB (512K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 64K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.62V ~ 3.6V |
| Data Converters | A/D 17x10b; D/A 1x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°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/atsam4n8ca-aur |

**Table 4-4. SAM4N8/16 100-ball VFBGA Pinout**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A1 | ADVREFP | C6 | PC9 | F1 | VDDOUT | H6 | PA12/PGMD0 |
| A2 | VDDPLL | C7 | TMS/SWDIO/PB6 | F2 | PA18/PGMD6/AD1 | H7 | PA9/PGMM1 |
| A3 | PB9/PGMCK/XIN | C8 | PA1/PGMEN1 | F3 | PA17/PGMD5/AD0 | H8 | VDDCORE |
| A4 | PB8/XOUT | C9 | PA0/PGMEN0 | F4 | GND | H9 | PA6/PGMNOE |
| A5 | JTAGSEL | C10 | PC16 | F5 | GND | H10 | PA5/PGMRDY |
| A6 | PB11 | D1 | PB1/AD5 | F6 | PC26 | J1 | PA20/AD3 |
| A7 | PB10 | D2 | PC30 | F7 | PA4/PGMNCMD | J2 | PC12/AD12 |
| A8 | PC20 | D3 | PC31 | F8 | PA28 | J3 | PA16/PGMD4 |
| A9 | PC19 | D4 | PC22 | F9 | TST | J4 | PC6 |
| A10 | TDO/TRACESWO/ PB5 | D5 | PC5 | F10 | PC8 | J5 | PA24 |
| B1 | GND | D6 | PA29 | G1 | PC15/AD11 | J6 | PA25 |
| B2 | PC25 | D7 | PA30/AD14 | G2 | PA19/PGMD7/AD2 | J7 | PA11/PGMM3 |
| B3 | PB14 | D8 | GND | G3 | PA21/PGMD9/AD8 | J8 | VDDCORE |
| B4 | PB13/DAC0 | D9 | PC14 | G4 | PA15/PGMD3 | J9 | VDDCORE |
| B5 | PC23 | D10 | PC11 | G5 | PC3 | J10 | TDI/PB4 |
| B6 | PC21 | E1 | VDDIN | G6 | PA10/PGMM2 | K1 | PA23 |
| B7 | TCK/SWCLK/PB7 | E2 | PB3/AD7 | G7 | PC1 | K2 | PC0 |
| B8 | PA31 | E3 | PB2/AD6 | G8 | PC28 | K3 | PC7 |
| B9 | PC18 | E4 | GND | G9 | NRST | K4 | PA13/PGMD1 |
| B10 | PC17 | E5 | GND | G10 | PA27 | K5 | PA26 |
| C1 | PB0/AD4 | E6 | GND | H1 | PC13/AD10 | K6 | PC2 |
| C2 | PC29/AD13 | E7 | VDDIO | H2 | PA22/AD9 | K7 | VDDIO |
| C3 | PC24 | E8 | PC10 | H3 | PC27 | K8 | VDDIO |
| C4 | ERASE/PB12 | E9 | PA2/PGMEN2 | H4 | PA14/PGMD2 | K9 | PA8/XOUT32/ PGMM0 |
| C5 | VDDCORE | E10 | PA3 | H5 | PC4 | K10 | PA7/XIN32/ PGMNVALID |

### 10.2.2 PIO Controller B Multiplexing

**Table 10-3. Multiplexing on PIO Controller B (PIOB)**

| I/O Line | Peripheral A | Peripheral B | Peripheral C | Extra Function | System Function | Comments |
|---|---|---|---|---|---|---|
| PB0 | PWM0 | TWD2 | | AD4[1] | | |
| PB1 | PWM1 | TWCK2 | | AD5[1] | | |
| PB2 | URXD1 | NPCS2 | | AD6/WKUP12[2] | | |
| PB3 | UTXD1 | PCK2 | | AD7[1] | | |
| PB4 | TWD1 | PWM2 | | | TDI[3] | |
| PB5 | TWCK1 | | | WKUP13[4] | TDO/TRACESWO[3] | |
| PB6 | | | | | TMS/SWDIO[3] | |
| PB7 | | | | | TCK/SWCLK[3] | |
| PB8 | | | | | XOUT[3] | |
| PB9 | | | | | XIN[3] | |
| PB10 | | URXD3 | | | | |
| PB11 | | UTXD3 | | | | |
| PB12 | | | | | ERASE[3] | |
| PB13 | | PCK0 | | DAC0[5] | | 64/100 pins versions |
| PB14 | NPCS1 | PWM3 | | | | 64/100 pins versions |

Notes:
1. To select this extra function, refer to Section 34.5.3 “Analog Inputs”.
2. Analog input has priority over WKUPx pin.
3. Refer to Section 6.2 “System I/O Lines”.
4. WKUPx can be used if PIO controller defines the I/O line as "input".
5. DAC0 is enabled when DACC_MR.DACEN is set. See Section 35.7.2 “DACC Mode Register”.

```
                        ; bottom halfword of R5, multiplies results and
                        ; and writes to R0
SMULWT      R4, R5, R3  ; Multiplies R5 with the top halfword of R3,
                        ; extracts top 32 bits and writes to R4
SMULWB      R4, R5, R3  ; Multiplies R5 with the bottom halfword of R3,
                        ; extracts top 32 bits and writes to R4.
```

#### 11.6.11.1 BKPT

Breakpoint.

Syntax

```
BKPT #imm
```

where:

imm is an expression evaluating to an integer in the range 0-255 (8-bit value).

Operation

The BKPT instruction causes the processor to enter Debug state. Debug tools can use this to investigate system state when the instruction at a particular address is reached.

*imm* is ignored by the processor. If required, a debugger can use it to store additional information about the breakpoint.

The BKPT instruction can be placed inside an IT block, but it executes unconditionally, unaffected by the condition specified by the IT instruction.

Condition Flags

This instruction does not change the flags.

Examples

```
BKPT 0xAB   ; Breakpoint with immediate value set to 0xAB (debugger can
            ; extract the immediate value by locating it using the PC)
```

Note: ARM does not recommend the use of the BKPT instruction with an immediate value set to 0xAB for any purpose other than Semi-hosting.

#### 11.6.11.2 CPS

Change Processor State.

Syntax

```
CPSeffect iflags
```

where:

effect is one of:

- IE Clears the special purpose register.
- ID Sets the special purpose register.

iflags is a sequence of one or more flags:

- i Set or clear PRIMASK.
- f Set or clear FAULTMASK.

Operation

CPS changes the PRIMASK and FAULTMASK special register values. See “Exception Mask Registers” for more information about these registers.

Restrictions

The restrictions are:

- Use CPS only from privileged software, it has no effect if used in unprivileged software
- CPS cannot be conditional and so must not be used inside an IT block.

Condition Flags

This instruction does not change the condition flags.

Examples

```
CPSID i  ; Disable interrupts and configurable fault handlers (set PRIMASK)
CPSID f  ; Disable interrupts and all fault handlers (set FAULTMASK)
CPSIE i  ; Enable interrupts and configurable fault handlers (clear PRIMASK)
CPSIE f  ; Enable interrupts and fault handlers (clear FAULTMASK)
```

### 13.5.2 Reset Controller Status Register

**Name:** RSTC_SR

**Address:** 0x400E1404

**Access:** Read-only

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

- **URSTS: User Reset Status**

0 = No high-to-low edge on NRST happened since the last read of RSTC_SR.

1 = At least one high-to-low transition of NRST has been detected since the last read of RSTC_SR.

- **RSTTYP: Reset Type**

| Value | Name | Description |
|---|---|---|
| 0 | General Reset | First power-up Reset |
| 1 | Backup Reset | Return from Backup Mode |
| 2 | Watchdog Reset | Watchdog fault occurred |
| 3 | Software Reset | Processor reset required by the software |
| 4 | User Reset | NRST pin detected low |

Reports the cause of the last processor reset. Reading this RSTC_SR does not reset this field.

- **NRSTL: NRST Pin Level**

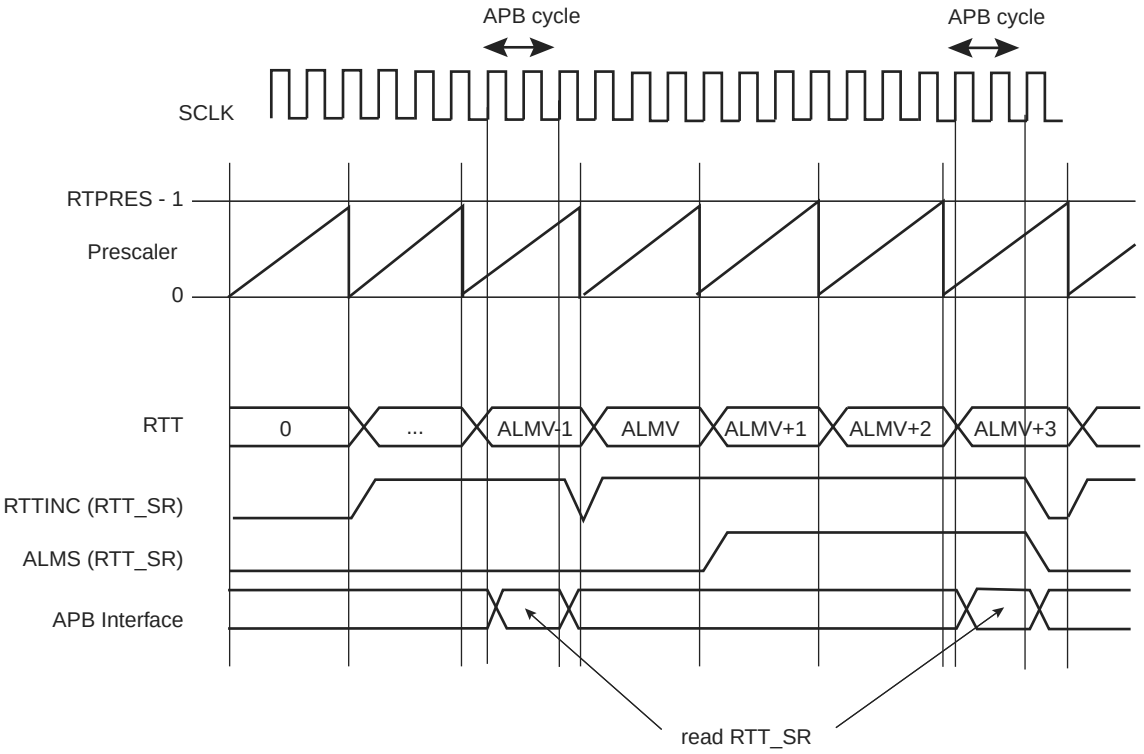
Registers the NRST Pin Level at Master Clock (MCK).

- **SRCMP: Software Reset Command in Progress**

0 = No software command is being performed by the reset controller. The reset controller is ready for a software command.

1 = A software reset command is being performed by the reset controller. The reset controller is busy.

**Figure 14-2. RTT Counting**



### 15.6.3 RTC Time Register

**Name:** RTC_TIMR

**Address:** 0x400E1468

**Access:** Read-write

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

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

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

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | SEC | | | | | | |

- **SEC: Current Second**

The range that can be set is 0 - 59 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **MIN: Current Minute**

The range that can be set is 0 - 59 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **HOUR: Current Hour**

The range that can be set is 1 - 12 (BCD) in 12-hour mode or 0 - 23 (BCD) in 24-hour mode.

- **AMPM: Ante Meridiem Post Meridiem Indicator**

This bit is the AM/PM indicator in 12-hour mode.

0 = AM.

1 = PM.

All non-significant bits read zero.

### 25.16.3 PMC System Clock Status Register

**Name:** PMC_SCSR

**Address:** 0x400E0408

**Access:** Read-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | PCK2 | PCK1 | PCK0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | – | – | – |

- **PCKx: Programmable Clock x Output Status**

0 = The corresponding Programmable Clock output is disabled.

1 = The corresponding Programmable Clock output is enabled.

### 25.16.8 PMC Clock Generator Main Clock Frequency Register

**Name:** CKGR_MCFR

**Address:** 0x400E0424

**Access:** Read-Write

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

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

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

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

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

- **MAINF: Main Clock Frequency**

Gives the number of Main Clock cycles within 16 Slow Clock periods.

- **MAINFRDY: Main Clock Ready**

0 = MAINF value is not valid or the Main Oscillator is disabled or a measure has just been started by means of RCMEAS.

1 = The Main Oscillator has been enabled previously and MAINF value is available.

Note: To ensure that a correct value is read on the MAINF bitfield, the MAINFRDY flag must be read at 1 then another read access must be performed on the register to get a stable value on the MAINF bitfield.

- **RCMEAS: RC Oscillator Frequency Measure (write-only)**

0 = No effect.

1 = Restarts measuring of the main RC frequency. MAINF will carry the new frequency as soon as a low to high transition occurs on the MAINFRDY flag.

The measure is performed on the main frequency (i.e. not limited to RC oscillator only), but if the main clock frequency source is the fast crystal oscillator, the restart of measuring is not needed because of the well known stability of crystal oscillators.

### 27.7.8 PIO Input Filter Disable Register

**Name:** PIO_IFDR

**Address:** 0x400E0E24 (PIOA), 0x400E1024 (PIOB), 0x400E1224 (PIOC)

**Access:** Write-only

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| P31 | P30 | P29 | P28 | P27 | P26 | P25 | P24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| P7 | P6 | P5 | P4 | P3 | P2 | P1 | P0 |

This register can only be written if the WPEN bit is cleared in “PIO Write Protect Mode Register” .

- **P0-P31: Input Filter Disable**

0: No effect.

1: Disables the input glitch filter on the I/O line.

### 28.7.3 Master Mode Operations

When configured in Master Mode, the SPI operates on the clock generated by the internal programmable baud rate generator. It fully controls the data transfers to and from the slave(s) connected to the SPI bus. The SPI drives the chip select line to the slave and the serial clock signal (SPCK).

The SPI features two holding registers, the Transmit Data Register and the Receive Data Register, and a single Shift Register. The holding registers maintain the data flow at a constant rate.

After enabling the SPI, a data transfer begins when the processor writes to the SPI_TDR (Transmit Data Register). The written data is immediately transferred in the Shift Register and transfer on the SPI bus starts. While the data in the Shift Register is shifted on the MOSI line, the MISO line is sampled and shifted in the Shift Register. Receiving data cannot occur without transmitting data. If receiving mode is not needed, for example when communicating with a slave receiver only (such as an LCD), the receive status flags in the status register can be discarded.

Before writing the TDR, the PCS field in the SPI_MR register must be set in order to select a slave.

After enabling the SPI, a data transfer begins when the processor writes to the SPI_TDR (Transmit Data Register). The written data is immediately transferred in the Shift Register and transfer on the SPI bus starts. While the data in the Shift Register is shifted on the MOSI line, the MISO line is sampled and shifted in the Shift Register. Transmission cannot occur without reception.

Before writing the TDR, the PCS field must be set in order to select a slave.

If new data is written in SPI_TDR during the transfer, it stays in it until the current transfer is completed. Then, the received data is transferred from the Shift Register to SPI_RDR, the data in SPI_TDR is loaded in the Shift Register and a new transfer starts.

The transfer of a data written in SPI_TDR in the Shift Register is indicated by the TDRE bit (Transmit Data Register Empty) in the Status Register (SPI_SR). When new data is written in SPI_TDR, this bit is cleared. The TDRE bit is used to trigger the TransmitPDC channel.

The end of transfer is indicated by the TXEMPTY flag in the SPI_SR register. If a transfer delay (DLYBCT) is greater than 0 for the last transfer, TXEMPTY is set after the completion of said delay. The master clock (MCK) can be switched off at this time.

The transfer of received data from the Shift Register in SPI_RDR is indicated by the RDRF bit (Receive Data Register Full) in the Status Register (SPI_SR). When the received data is read, the RDRF bit is cleared.

If the SPI_RDR (Receive Data Register) has not been read before new data is received, the Overrun Error bit (OVRES) in SPI_SR is set. As long as this flag is set, data is loaded in SPI_RDR. The user has to read the status register to clear the OVRES bit.

Figure 28-5, shows a block diagram of the SPI when operating in Master Mode. Figure 28-6 on page 510 shows a flow chart describing how transfers are handled.

### 28.8.11 SPI Write Protection Status Register

**Name:** SPI_WPSR

**Address:** 0x400080E8

**Access:** Read-only

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

- **WPVS: Write Protection Violation Status**

0 = No Write Protect Violation has occurred since the last read of the SPI_WPSR register.

1 = A Write Protect Violation has occurred since the last read of the SPI_WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

This Field indicates the APB Offset of the register concerned by the violation (SPI_MR or SPI_CSRx)

### 29.11.2 TWI Master Mode Register

**Name:** TWI_MMR

**Address:** 0x40018004 (0), 0x4001C004 (1), 0x40040004 (2)

**Access:** Read-write

**Reset:** 0x00000000

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

- **IADRSZ: Internal Device Address Size**

| Value | Name | Description |
|---|---|---|
| 0 | NONE | No internal device address |
| 1 | 1_BYTE | One-byte internal device address |
| 2 | 2_BYTE | Two-byte internal device address |
| 3 | 3_BYTE | Three-byte internal device address |

- **MREAD: Master Read Direction**

0 = Master write direction.

1 = Master read direction.

- **DADR: Device Address**

The device address is used to access slave devices in read or write mode. Those bits are only used in Master mode.

### 29.11.4 TWI Internal Address Register

**Name:** TWI_IADR

**Address:** 0x4001800C (0), 0x4001C00C (1), 0x4004000C (2)

**Access:** Read-write

**Reset:** 0x00000000

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

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

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

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

- **IADR: Internal Address**

0, 1, 2 or 3 bytes depending on IADRSZ.

### 29.11.9 TWI Interrupt Mask Register

**Name:** TWI_IMR

**Address:** 0x4001802C (0), 0x4001C02C (1), 0x4004002C (2)

**Access:** Read-only

**Reset:** 0x00000000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TXBUFE | RXBUFF | ENDTX | ENDRX | EOSACC | SCL_WS | ARBLST | NACK |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | OVRE | GACC | SVACC | – | TXRDY | RXRDY | TXCOMP |

- **TXCOMP: Transmission Completed Interrupt Mask**
- **RXRDY: Receive Holding Register Ready Interrupt Mask**
- **TXRDY: Transmit Holding Register Ready Interrupt Mask**
- **SVACC: Slave Access Interrupt Mask**
- **GACC: General Call Access Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **NACK: Not Acknowledge Interrupt Mask**
- **ARBLST: Arbitration Lost Interrupt Mask**
- **SCL_WS: Clock Wait State Interrupt Mask**
- **EOSACC: End Of Slave Access Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **ENDTX: End of Transmit Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**
- **TXBUFE: Transmit Buffer Empty Interrupt Mask**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

# 31. Universal Synchronous Asynchronous Receiver Transmitter (USART)

## 31.1 Description

The Universal Synchronous Asynchronous Receiver Transceiver (USART) provides one full duplex universal synchronous asynchronous serial link. Data frame format is widely programmable (data length, parity, number of stop bits) to support a maximum of standards. The receiver implements parity error, framing error and overrun error detection. The receiver time-out enables handling variable-length frames and the transmitter timeguard facilitates communications with slow remote devices. Multidrop communications are also supported through address bit handling in reception and transmission.

The USART features three test modes: remote loopback, local loopback and automatic echo.

The USART supports specific operating modes providing interfaces on RS485 and SPI buses, with ISO7816 T = 0 or T = 1 smart card slots and infrared transceivers. The hardware handshaking feature enables an out-of-band flow control by automatic management of the pins RTS and CTS.

The USART supports the connection to the Peripheral DMA Controller, which enables data transfers to the transmitter and from the receiver. The PDC provides chained buffer management without any intervention of the processor.

## 31.2 Embedded Characteristics

- Programmable Baud Rate Generator
- 5- to 9-bit Full-duplex Synchronous or Asynchronous Serial Communications
  - 1, 1.5 or 2 Stop Bits in Asynchronous Mode or 1 or 2 Stop Bits in Synchronous Mode
  - Parity Generation and Error Detection
  - Framing Error Detection, Overrun Error Detection
  - MSB- or LSB-first
  - Optional Break Generation and Detection
  - By 8 or by 16 Over-sampling Receiver Frequency
  - Optional Hardware Handshaking RTS-CTS
  - Receiver Time-out and Transmitter Timeguard
  - Optional Multidrop Mode with Address Generation and Detection
- RS485 with Driver Control Signal
- ISO7816, T = 0 or T = 1 Protocols for Interfacing with Smart Cards
  - NACK Handling, Error Counter with Repetition and Iteration Limit
- IrDA Modulation and Demodulation
  - Communication at up to 115.2 Kbps
- SPI Mode
  - Master or Slave
  - Serial Clock Programmable Phase and Polarity
  - SPI Serial Clock (SCK) Frequency up to Internal Clock Frequency MCK/6
- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo
- Supports Connection of:
  - Two Peripheral DMA Controller Channels (PDC)
- Offers Buffer Transfer without Processor Intervention

## 31.6 Product Dependencies

### 31.6.1 I/O Lines

The pins used for interfacing the USART may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the desired USART pins to their peripheral function. If I/O lines of the USART are not used by the application, they can be used for other purposes by the PIO Controller.

To prevent the TXD line from falling when the USART is disabled, the use of an internal pull up is mandatory. If the hardware handshaking feature is used, the internal pull up on TXD must also be enabled.

**Table 31-3. I/O Lines**

| Instance | Signal | I/O Line | Peripheral |
|---|---|---|---|
| USART0 | CTS0 | PA8 | A |
| USART0 | RTS0 | PA7 | A |
| USART0 | RXD0 | PA5 | A |
| USART0 | SCK0 | PA2 | B |
| USART0 | TXD0 | PA6 | A |
| USART1 | CTS1 | PA25 | A |
| USART1 | RTS1 | PA24 | A |
| USART1 | RXD1 | PA21 | A |
| USART1 | SCK1 | PA23 | A |
| USART1 | TXD1 | PA22 | A |
| USART2 | CTS2 | PC17 | A |
| USART2 | RTS2 | PC16 | A |
| USART2 | RXD2 | PC9 | A |
| USART2 | SCK2 | PC14 | A |
| USART2 | TXD2 | PC10 | A |

### 31.6.2 Power Management

The USART is not continuously clocked. The programmer must first enable the USART Clock in the Power Management Controller (PMC) before using the USART. However, if the application does not require USART operations, the USART clock can be stopped when not needed and be restarted later. In this case, the USART will resume its operations where it left off.

Configuring the USART does not require the USART clock to be enabled.

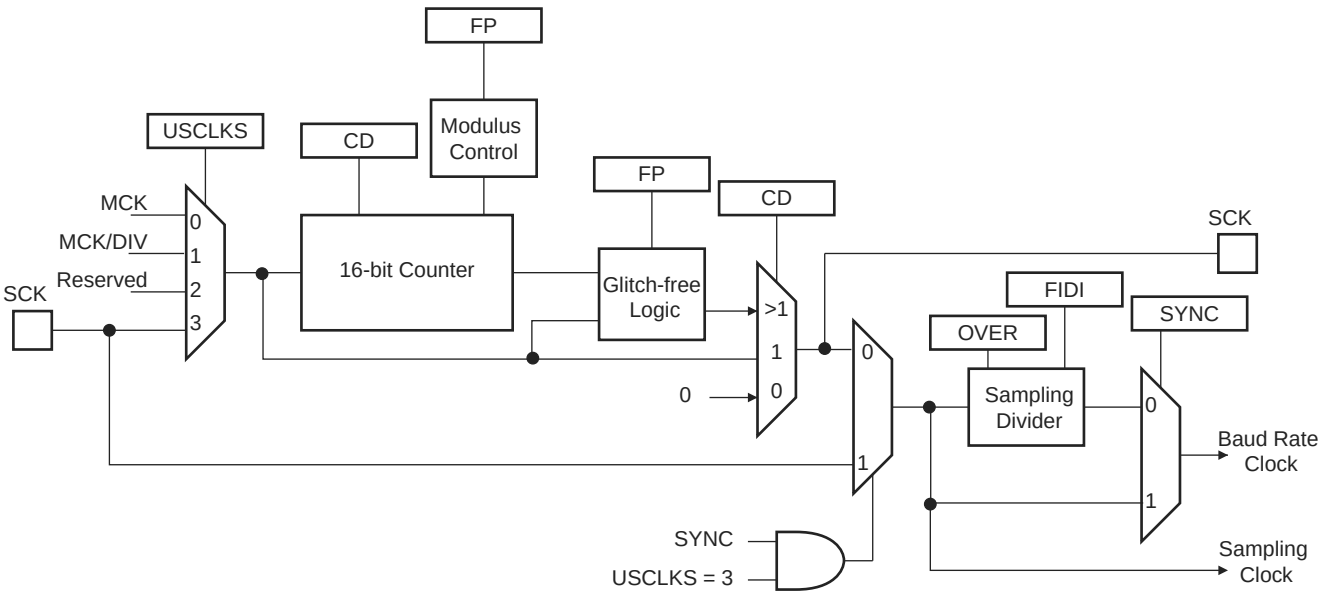
### 31.7.1.2 Fractional Baud Rate in Asynchronous Mode

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 the Baud Rate Generator Register (US_BRGR). If FP is not 0, the fractional part is activated. The resolution is one eighth of the clock divider. This feature is only available when using USART normal mode. The fractional Baud Rate is calculated using the following formula:

$$Baudrate = \frac{SelectedClock}{\left(8(2-Over)\left(CD + \frac{FP}{8}\right)\right)}$$

The modified architecture is presented below:

**Figure 31-4. Fractional Baud Rate Generator**




### 31.7.1.3 Baud Rate in Synchronous Mode or SPI Mode

If the USART is programmed to operate in synchronous mode, the selected clock is simply divided by the field CD in US_BRGR.

$$BaudRate = \frac{SelectedClock}{CD}$$

In synchronous mode, if the external clock is selected (USCLKS = 3), the clock is provided directly by the signal on the USART SCK pin. No division is active. The value written in US_BRGR has no effect. The external clock frequency must be at least 3 times lower than the system clock. In synchronous mode master (USCLKS = 0 or 1, CLK0 set to 1), the receive part limits the SCK maximum frequency to MCK/3 in USART mode, or MCK/6 in SPI mode.

When either the external clock SCK or the internal clock divided (MCK/DIV) is selected, the value programmed in CD must be even if the user has to ensure a 50:50 mark/space ratio on the SCK pin. If the internal clock MCK is selected, the Baud Rate Generator ensures a 50:50 duty cycle on the SCK pin, even if the value programmed in CD is odd.

### 35.7.5 DACC Interrupt Disable Register

**Name:** DACC_IDR

**Address:** 0x4003C010

**Access:** Write-only

| 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 |
| – | – | – | – | – | TXBUFE | ENDTX | TXRDY |

- **TXRDY: Transmission Ready Interrupt Disable**

Disables ready for transmission interrupt.

- **ENDTX: End of PDC Interrupt Disable**
- **TXBUFE: Buffer Empty Interrupt Disable**