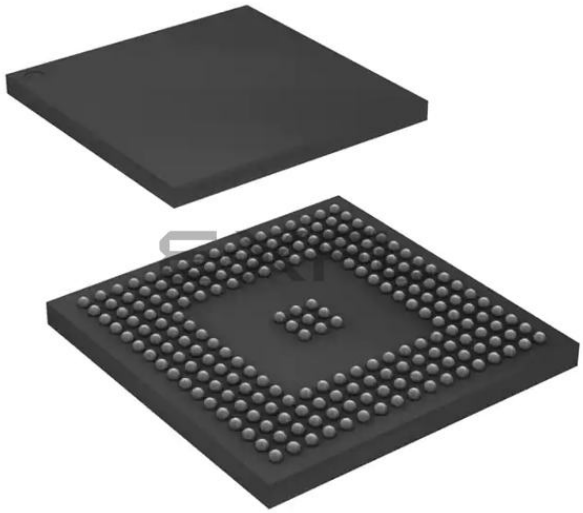


Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

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

Applications of Embedded - Microprocessors

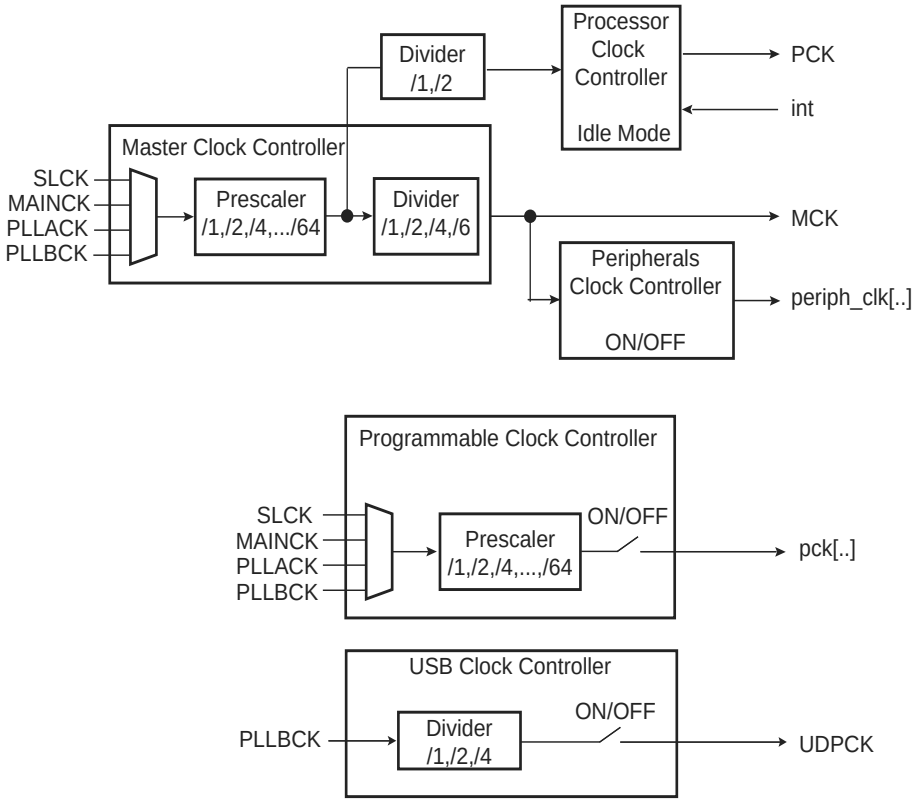
Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

Details	
Product Status	Active
Core Processor	ARM926EJ-S
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	400MHz
Co-Processors/DSP	-
RAM Controllers	SDRAM, SRAM
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	10/100Mbps
SATA	-
USB	USB 2.0 (2)
Voltage - I/O	1.8V, 3.3V
Operating Temperature	-40°C ~ 85°C (TA)
Security Features	-
Package / Case	217-LFBGA
Supplier Device Package	217-LFBGA (15x15)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at91sam9g20b-cu

8.5 Power Management Controller

- Provides:
  - the Processor Clock PCK
  - the Master Clock MCK, in particular to the Matrix and the memory interfaces.The MCK divider can be 1,2,4,6
  - the USB Device Clock UDPCK
  - independent peripheral clocks, typically at the frequency of MCK
  - 2 programmable clock outputs: PCK0, PCK1
- Five flexible operating modes:
  - Normal Mode, processor and peripherals running at a programmable frequency
  - Idle Mode, processor stopped waiting for an interrupt
  - Slow Clock Mode, processor and peripherals running at low frequency
  - Standby Mode, mix of Idle and Backup Mode, peripheral running at low frequency, processor stopped waiting for an interrupt
  - Backup Mode, Main Power Supplies off, VDDBU powered by a battery

Figure 8-3: SAM9G20 Power Management Controller Block Diagram



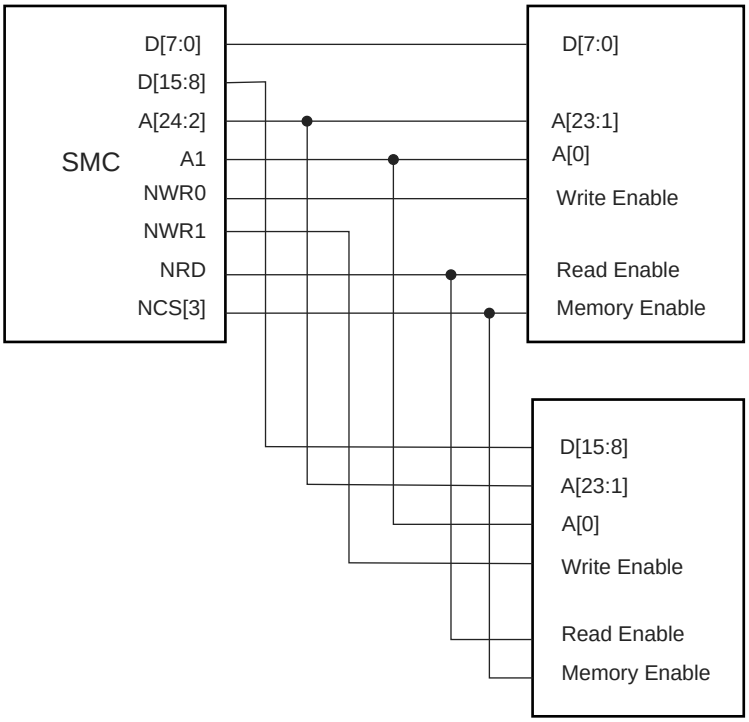
8.6 Periodic Interval Timer

- Includes a 20-bit Periodic Counter, with less than 1 µs accuracy
- Includes a 12-bit Interval Overlay Counter
- Real Time OS or Linux®/Windows CE® compliant tick generator

8.7 Watchdog Timer

- 16-bit key-protected only-once-Programmable Counter
- Windowed, prevents the processor being in a dead-lock on the watchdog access

Figure 20-6: Connection of 2 x 8-bit Devices on a 16-bit Bus: Byte Write Option

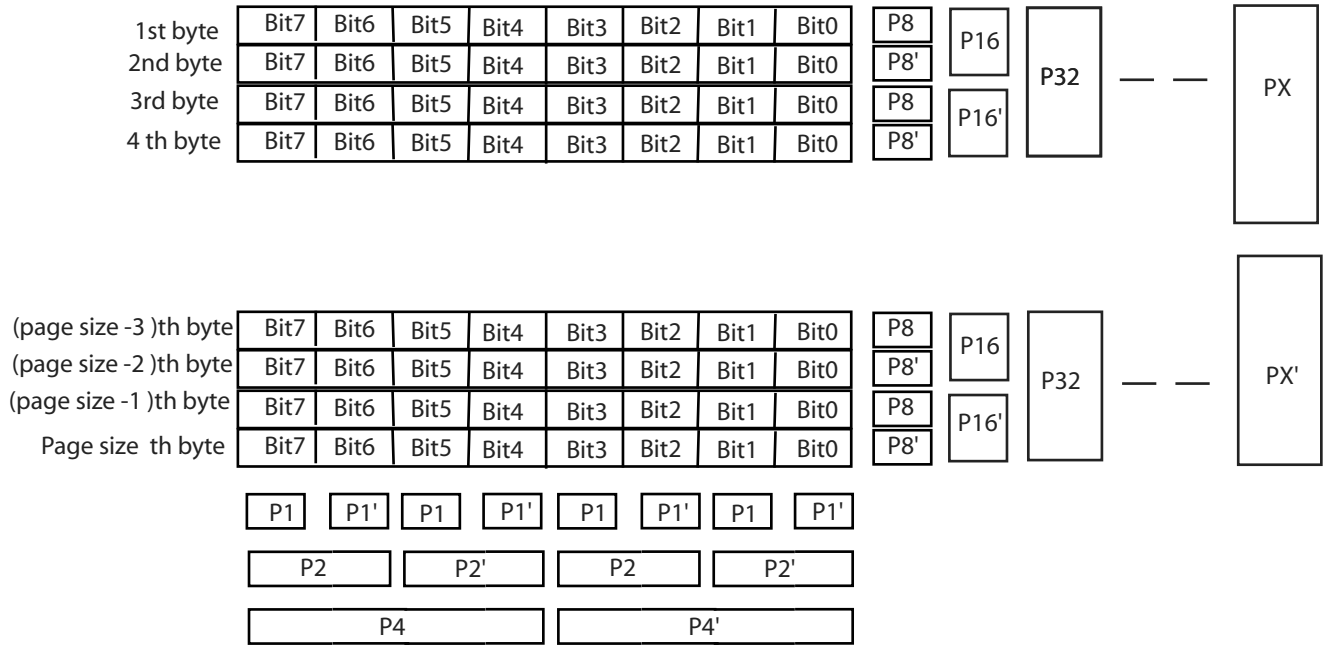


20.7.2.3 Signal Multiplexing

Depending on the BAT, only the write signals or the byte select signals are used. To save IOs at the external bus interface, control signals at the SMC interface are multiplexed. Table 20-3 shows signal multiplexing depending on the data bus width and the byte access type.

For 32-bit devices, bits A0 and A1 are unused. For 16-bit devices, bit A0 of address is unused. When Byte Select Option is selected, NWR1 to NWR3 are unused. When Byte Write option is selected, NBS0 to NBS3 are unused.

Figure 22-2: Parity Generation for 512/1024/2048/4096 8-bit Words



Page size = 512 Px = 2048
Page size = 1024 Px = 4096
Page size = 2048 Px = 8192
Page size = 4096 Px = 16384

P1=bit7(+)bit5(+)bit3(+)bit1(+)P1
P2=bit7(+)bit6(+)bit3(+)bit2(+)P2
P4=bit7(+)bit6(+)bit5(+)bit4(+)P4
P1'=bit6(+)bit4(+)bit2(+)bit0(+)P1'
P2'=bit5(+)bit4(+)bit1(+)bit0(+)P2'
P4'=bit7(+)bit6(+)bit5(+)bit4(+)P4'

To calculate P8' to PX' and P8 to PX, apply the algorithm that follows.

Page size = 2^n

```
for i =0 to n
begin
 for (j = 0 to page_size_byte)
 begin
  if(j[i] ==1)
   P[2^(i+3)]=bit7(+)bit6(+)bit5(+)bit4(+)bit3(+)
              bit2(+)bit1(+)bit0(+)P[2^(i+3)]
  else
   P[2^(i+3)]'=bit7(+)bit6(+)bit5(+)bit4(+)bit3(+)
               bit2(+)bit1(+)bit0(+)P[2^(i+3)]'
  end
end
```

22.5.2 ECC Parity Register 1

Name:ECC_PR1

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
NPARITY							
7	6	5	4	3	2	1	0
NPARITY							

NPARITY

Parity N

22.7.10 ECC Parity Register 9

Name:ECC_PR9

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY9						
15	14	13	12	11	10	9	8
NPARITY9				0	WORDADDR9		
7	6	5	4	3	2	1	0
WORDADDR9					BITADDR9		

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area

BITADDR9: corrupted bit address in the page between the 2304th and the 2559th bytes

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

WORDADDR9: corrupted word address in the page between the 2304th and the 2559th bytes

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless

NPARITY9

Parity N

23.4 Peripheral DMA Controller (PDC) User Interface

Table 23-1: Register Mapping

Offset	Register	Name[1]	Access	Reset
0x100	Receive Pointer Register	PERIPH_RPR	Read/Write	0
0x104	Receive Counter Register	PERIPH_RCR	Read/Write	0
0x108	Transmit Pointer Register	PERIPH_TPR	Read/Write	0
0x10C	Transmit Counter Register	PERIPH_TCR	Read/Write	0
0x110	Receive Next Pointer Register	PERIPH_RNPR	Read/Write	0
0x114	Receive Next Counter Register	PERIPH_RNCR	Read/Write	0
0x118	Transmit Next Pointer Register	PERIPH_TNPR	Read/Write	0
0x11C	Transmit Next Counter Register	PERIPH_TNCR	Read/Write	0
0x120	Transfer Control Register	PERIPH_PTCR	Write-only	–
0x124	Transfer Status Register	PERIPH_PTSR	Read-only	0

Note 1: PERIPH: Ten registers are mapped in the peripheral memory space at the same offset. These can be defined by the user according to the function and the peripheral desired (DBGU, USART, SSC, SPI, MCI, etc.)

23.4.5 Receive Next Pointer Register

Name:PERIPH_RNPR

Access: Read/Write

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

RXNPTR: Receive Next Pointer

RXNPTR contains next receive buffer address.

When a half duplex peripheral is connected to the PDC, RXNPTR = TXNPTR.

Once the PMC_PLLB register has been written, the user must wait for the LOCKB bit to be set in the PMC_SR. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to LOCKB has been enabled in the PMC_IER. All parameters in CKGR_PLLBR can be programmed in a single write operation. If at some stage one of the following parameters, MULB, DIVB is modified, LOCKB bit will go low to indicate that PLL B is not ready yet. When PLL B is locked, LOCKB will be set again. The user is constrained to wait for LOCKB bit to be set before using the PLL A output clock.

The USBDIV field is used to control the additional divider by 1, 2 or 4, which generates the USB clock(s).

Code Example:

```
write_register(CKGR_PLLBR,0x20030602)
```

PLL B input clock is main clock divided by 2. PLL B output clock is PLL B input clock multiplied by 4. Once CKGR_PLLBR has been written, LOCKB bit will be set after six slow clock cycles.

5. Selection of Master Clock and Processor Clock

The Master Clock and the Processor Clock are configurable via the PMC_MCKR.

The CSS field is used to select the clock source of the Master Clock and Processor Clock dividers. By default, the selected clock source is slow clock.

The PRES field is used to control the Master/Processor Clock prescaler. The user can choose between different values (1, 2, 4, 8, 16, 32, 64). Prescaler output is the selected clock source divided by PRES parameter. By default, PRES parameter is set to 1 which means that the input clock of the Master Clock and Processor Clock dividers is equal to slow clock.

The MDIV field is used to control the Master Clock divider. It is possible to choose between different values (0, 1, 2, 3). The Master Clock output is Master/Processor Clock Prescaler output divided by 1, 2, 4 or 6, depending on the value programmed in MDIV.

The PDIV field is used to control the Processor Clock divider. It is possible to choose between different values (0, 1). The Processor Clock output is Master/Processor Clock Prescaler output divided by 1 or 2, depending on the value programmed in PDIV.

By default, MDIV and PDIV are set to 0, which indicates that Processor Clock is equal to the Master Clock.

Once the PMC_MCKR has been written, the user must wait for the MCKRDY bit to be set in the PMC_SR. This can be done either by polling the status register or by waiting for the interrupt line to be raised if the associated interrupt to MCKRDY has been enabled in the PMC_IER.

The PMC_MCKR must not be programmed in a single write operation. The preferred programming sequence for the PMC_MCKR is as follows:

- If a new value for CSS field corresponds to PLL Clock,
  - Program the PRES field in the PMC_MCKR.
  - Wait for the MCKRDY bit to be set in the PMC_SR.
  - Program the CSS field in the PMC_MCKR.
  - Wait for the MCKRDY bit to be set in the PMC_SR.
- If a new value for CSS field corresponds to Main Clock or Slow Clock,
  - Program the CSS field in the PMC_MCKR.
  - Wait for the MCKRDY bit to be set in the PMC_SR.
  - Program the PRES field in the PMC_MCKR.
  - Wait for the MCKRDY bit to be set in the PMC_SR.

If at some stage one of the following parameters, CSS or PRES, is modified, the MCKRDY bit will go low to indicate that the Master Clock and the Processor Clock are not ready yet. The user must wait for MCKRDY bit to be set again before using the Master and Processor Clocks.

Note: IF PLLx clock was selected as the Master Clock and the user decides to modify it by writing in CKGR_PLLR (CKGR_PLLAR or CKGR_PLLBR), the MCKRDY flag will go low while PLL is unlocked. Once PLL is locked again, LOCK (LOCKA or LOCKB) goes high and MCKRDY is set. While PLLA is unlocked, the Master Clock selection is automatically changed to Slow Clock. While PLLB is unlocked, the Master Clock selection is automatically changed to Main Clock. For further information, see Section 25.8.2 “Clock Switching Waveforms”.

Code Example:

```
write_register(PMC_MCKR,0x00000001)
        wait (MCKRDY=1)
write_register(PMC_MCKR,0x00000011)
```

25.9.3 PMC System Clock Status Register

Name:PMC_SCSR

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
–	–	–	–	–	–	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	UHP	–	–	–	–	–	PCK

PCK: Processor Clock Status

0: The Processor clock is disabled.

1: The Processor clock is enabled.

UHP: USB Host Port Clock Status

0: The 12 and 48 MHz clock (UHPCK) of the USB Host Port is disabled.

1: The 12 and 48 MHz clock (UHPCK) of the USB Host Port is enabled.

UDP: USB Device Port Clock Status

0: The 48 MHz clock (UDPCK) of the USB Device Port is disabled.

1: The 48 MHz clock (UDPCK) of the USB Device Port is enabled.

PCKx: Programmable Clock x Output Status

0: The corresponding Programmable Clock output is disabled.

1: The corresponding Programmable Clock output is enabled.

29.6.3.1 Master Mode Block Diagram

Figure 29-5: Master Mode Block Diagram

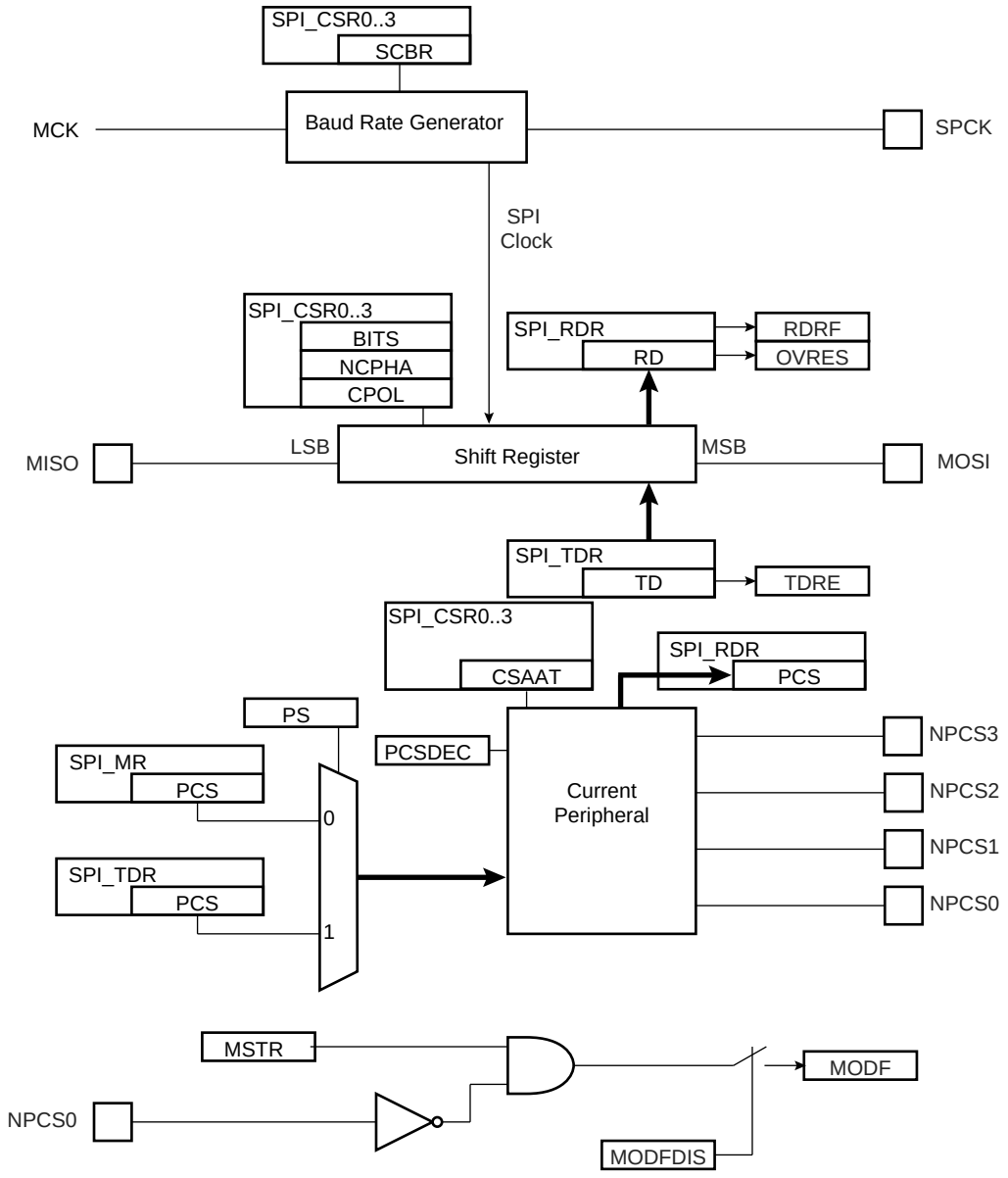


Figure 30-15: TWI Write Operation with Single Data Byte and Internal Address

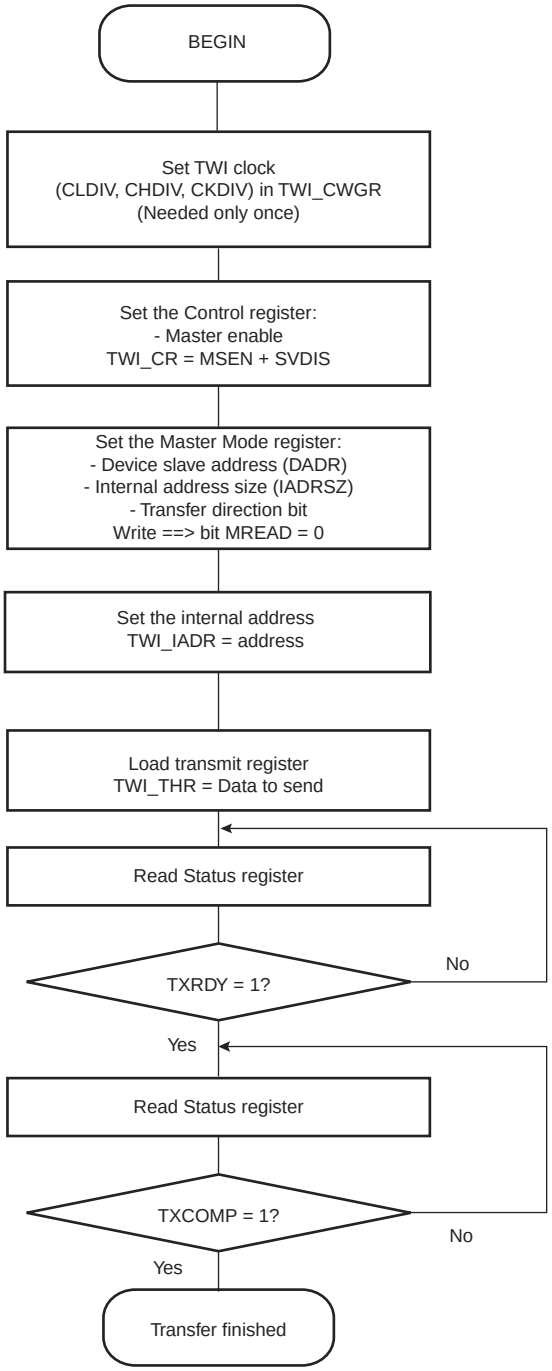


Figure 30-17: TWI Read Operation with Single Data Byte without Internal Address

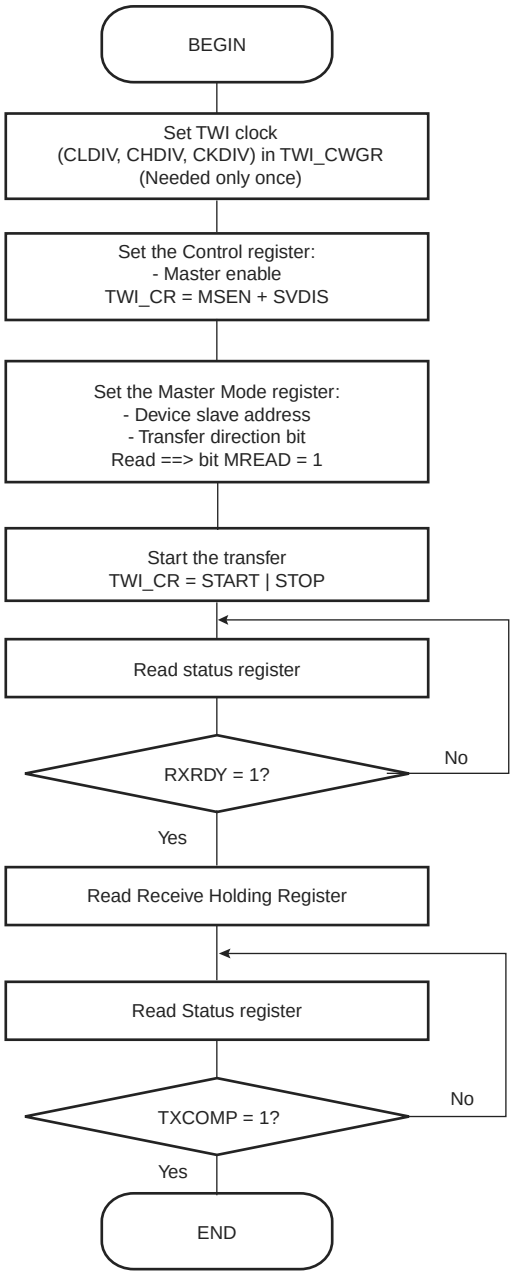


Table 31-2: Baud Rate Example (OVER = 0) (Continued)

Source Clock (MHz)	Expected Baud Rate (bit/s)	Calculation Result	CD	Actual Baud Rate (bit/s)	Error
25 000 000	38 400	40.69	40	38 109.76	0.76%
32 000 000	38 400	52.08	52	38 461.54	0.16%
32 768 000	38 400	53.33	53	38 641.51	0.63%
33 000 000	38 400	53.71	54	38 194.44	0.54%
40 000 000	38 400	65.10	65	38 461.54	0.16%
50 000 000	38 400	81.38	81	38 580.25	0.47%

The baud rate is calculated with the following formula:

$$BaudRate = MCK/CD \times 16$$

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

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate}\right)$$

31.6.1.3 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 in Figure 31-4.

Figure 31-4: Fractional Baud Rate Generator

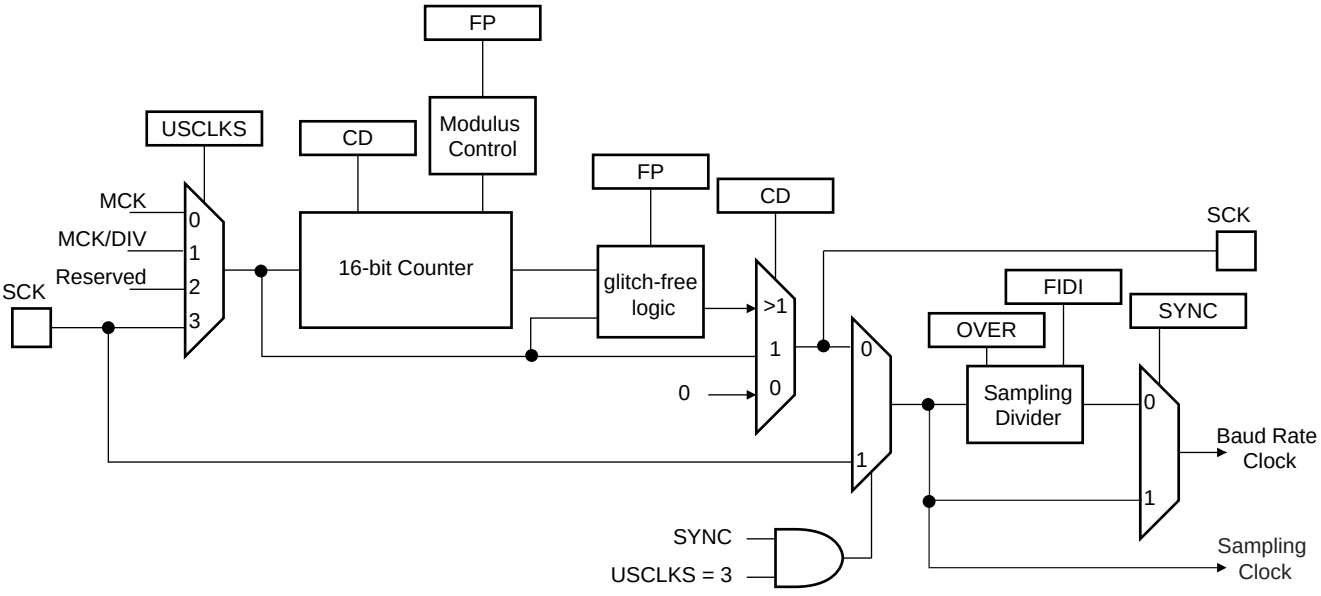
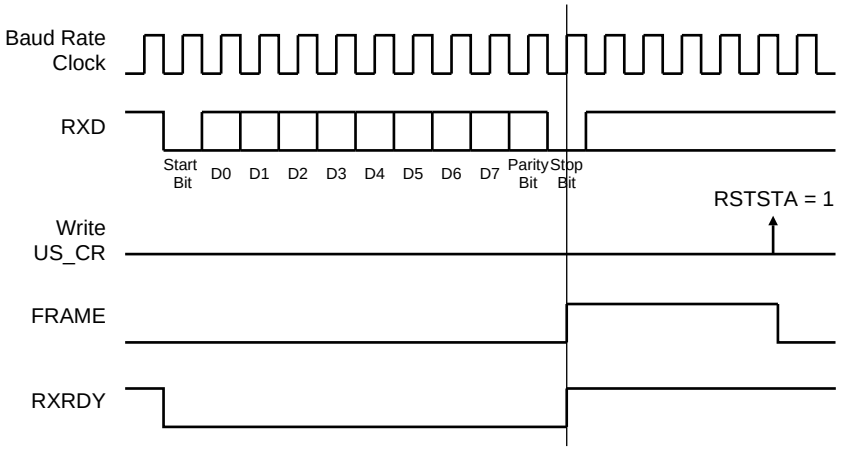


Figure 31-25: Framing Error Status



31.6.3.14 Transmit Break

The user can request the transmitter to generate a break condition on the TXD line. A break condition drives the TXD line low during at least one complete character. It appears the same as a 0x00 character sent with the parity and the stop bits at 0. However, the transmitter holds the TXD line at least during one character until the user requests the break condition to be removed.

A break is transmitted by writing the Control Register (US_CR) with the STTBRK bit at 1. This can be performed at any time, either while the transmitter is empty (no character in either the Shift Register or in US_THR) or when a character is being transmitted. If a break is requested while a character is being shifted out, the character is first completed before the TXD line is held low.

Once STTBRK command is requested further STTBRK commands are ignored until the end of the break is completed.

The break condition is removed by writing US_CR with the STPBRK bit at 1. If the STPBRK is requested before the end of the minimum break duration (one character, including start, data, parity and stop bits), the transmitter ensures that the break condition completes.

The transmitter considers the break as though it is a character, i.e. the STTBRK and STPBRK commands are taken into account only if the TXRDY bit in US_CSR is at 1 and the start of the break condition clears the TXRDY and TXEMPTY bits as if a character is processed.

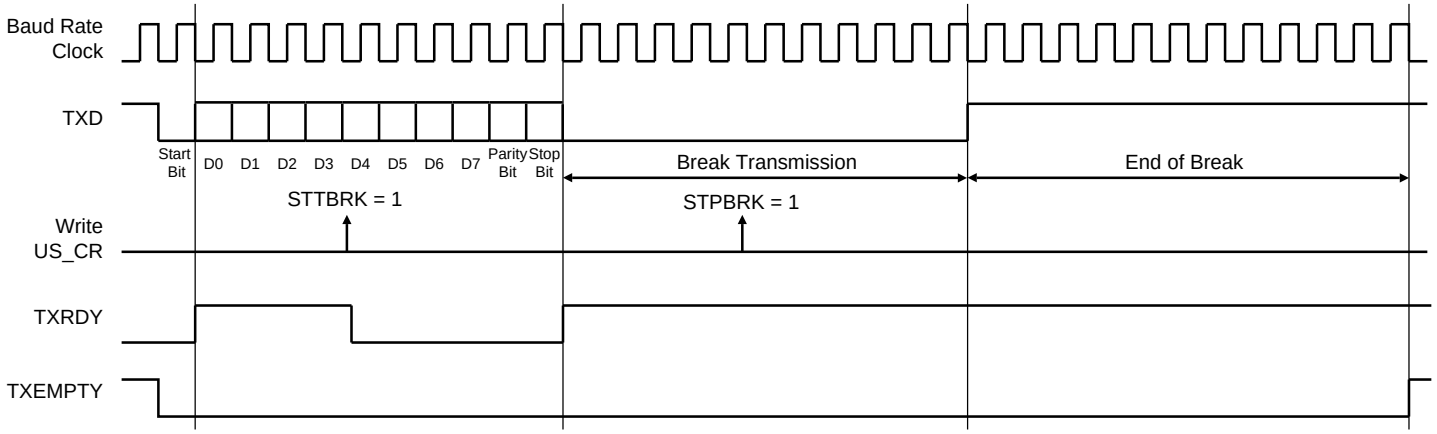
Writing US_CR with the both STTBRK and STPBRK bits at 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

After the break condition, the transmitter returns the TXD line to 1 for a minimum of 12 bit times. Thus, the transmitter ensures that the remote receiver detects correctly the end of break and the start of the next character. If the timeguard is programmed with a value higher than 12, the TXD line is held high for the timeguard period.

After holding the TXD line for this period, the transmitter resumes normal operations.

Figure 31-26 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

Figure 31-26: Break Transmission



32.8.4 SSC Receive Frame Mode Register

Name:SSC_RFMR

Access: Read/Write

31	30	29	28	27	26	25	24
FSLEN_EXT	FSLEN_EXT	FSLEN_EXT	FSLEN_EXT	–	–	–	FSEDGE
23	22	21	20	19	18	17	16
–	FSOS			FSLEN			
15	14	13	12	11	10	9	8
–	–	–	–	DATNB			
7	6	5	4	3	2	1	0
MSBF	–	LOOP	DATLEN				

DATLEN: Data Length

0: Forbidden value (1-bit data length not supported).
Any other value: The bit stream contains DATLEN + 1 data bits. Moreover, it defines the transfer size performed by the PDC2 assigned to the Receiver. If DATLEN is lower or equal to 7, data transfers are in bytes. If DATLEN is between 8 and 15 (included), half-words are transferred, and for any other value, 32-bit words are transferred.

LOOP: Loop Mode

0: Normal operating mode.

1: RD is driven by TD, RF is driven by TF and TK drives RK.

MSBF: Most Significant Bit First

0: The lowest significant bit of the data register is sampled first in the bit stream.

1: The most significant bit of the data register is sampled first in the bit stream.

DATNB: Data Number per Frame

This field defines the number of data words to be received after each transfer start, which is equal to (DATNB + 1).

FSLEN: Receive Frame Sync Length

This field defines the number of bits sampled and stored in the Receive Sync Data Register. When this mode is selected by the START field in the Receive Clock Mode Register, it also determines the length of the sampled data to be compared to the Compare 0 or Compare 1 register.

This field is used with FSLEN_EXT to determine the pulse length of the Receive Frame Sync signal.

Pulse length is equal to FSLEN + (FSLEN_EXT * 16) + 1 Receive Clock periods.

FSOS: Receive Frame Sync Output Selection

FSOS	Selected Receive Frame Sync Signal	RF Pin
0x0	None	Input-only
0x1	Negative Pulse	Output
0x2	Positive Pulse	Output
0x3	Driven Low during data transfer	Output
0x4	Driven High during data transfer	Output
0x5	Toggling at each start of data transfer	Output
0x6–0x7	Reserved	Undefined

33.5.10 Waveform Operating Mode

Waveform operating mode is entered by setting the WAVE parameter in TC_CMR (Channel Mode Register).

In Waveform Operating Mode the TC channel generates 1 or 2 PWM signals with the same frequency and independently programmable duty cycles, or generates different types of one-shot or repetitive pulses.

In this mode, TIOA is configured as an output and TIOB is defined as an output if it is not used as an external event (EEVT parameter in TC_CMR).

Figure 33-6 shows the configuration of the TC channel when programmed in Waveform Operating Mode.

33.5.11 Waveform Selection

Depending on the WAVSEL parameter in TC_CMR (Channel Mode Register), the behavior of TC_CV varies.

With any selection, RA, RB and RC can all be used as compare registers.

RA Compare is used to control the TIOA output, RB Compare is used to control the TIOB output (if correctly configured) and RC Compare is used to control TIOA and/or TIOB outputs.

35.5.3 Network Status Register

Name:EMAC_NSR

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
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	IDLE	MDIO	–

MDIO

Returns status of the mdio_in pin. Use the PHY maintenance register for reading managed frames rather than this bit.

IDLE

0: The PHY logic is running.

1: The PHY management logic is idle (i.e., has completed).

35.5.16 Specific Address 1 Bottom Register

Name:EMAC_SA1B

Access: Read/Write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

ADDR

Least significant bits of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

35.5.26.9 Late Collisions Register

Name:EMAC_LCOL

Access: Read/Write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
LCOL							

LCOL: Late Collisions

An 8-bit register counting the number of frames that experience a collision after the slot time (512 bits) has expired. A late collision is counted twice; i.e., both as a collision and a late collision.

36.6.10 UDP Endpoint Control and Status Register

Name:UDP_CSRx [x = 0..5]

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	RXBYTECNT		

23	22	21	20	19	18	17	16
RXBYTECNT							

15	14	13	12	11	10	9	8
EPEDS	–	–	–	DTGLE	EPTYPE		

7	6	5	4	3	2	1	0
DIR	RX_DATA_BK1	FORCESTALL	TXPKTRDY	STALLSENT ISOERROR	RXSETUP	RX_DATA_BK0	TXCOMP

WARNING: Due to synchronization between MCK and UDPCK, the software application must wait for the end of the write operation before executing another write by polling the bits which must be set/cleared.

```
//! Clear flags of UDP UDP_CSR register and waits for synchronization
#define Udp_ep_clr_flag(pInterface, endpoint, flags) { \
   pInterface->UDP_CSR[endpoint] &= ~(flags); \
   while ( (pInterface->UDP_CSR[endpoint] & (flags)) == (flags) ); \
   }
//! Set flags of UDP UDP_CSR register and waits for synchronization
#define Udp_ep_set_flag(pInterface, endpoint, flags) { \
   pInterface->UDP_CSR[endpoint] |= (flags); \
   while ( (pInterface->UDP_CSR[endpoint] & (flags)) != (flags) ); \
   }
```

Note: In a preemptive environment, set or clear the flag and wait for a time of 1 UDPCK clock cycle and 1peripheral clock cycle. However, RX_DATA_BLK0, TXPKTRDY, RX_DATA_BK1 require wait times of 3 UDPCK clock cycles and 3 peripheral clock cycles before accessing DPR.

TXCOMP: Generates an IN Packet with Data Previously Written in the DPR

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware):

0: Clear the flag, clear the interrupt.

1: No effect.

Read (Set by the USB peripheral):

0: Data IN transaction has not been acknowledged by the Host.

1: Data IN transaction is achieved, acknowledged by the Host.

After having issued a Data IN transaction setting TXPKTRDY, the device firmware waits for TXCOMP to be sure that the host has acknowledged the transaction.