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	47
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 11x10b; D/A 1x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	64-LQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atsam4n8ba-au

11.6.5.2 AND, ORR, EOR, BIC, and ORN

Logical AND, OR, Exclusive OR, Bit Clear, and OR NOT.

Syntax

```
op{S}{cond} {Rd,} Rn, Operand2
```

where:

op is one of:

AND logical AND.

ORR logical OR, or bit set.

EOR logical Exclusive OR.

BIC logical AND NOT, or bit clear.

ORN logical OR NOT.

S is an optional suffix. If S is specified, the condition code flags are updated on the result of the operation, see “Conditional Execution” .

cond is an optional condition code, see “Conditional Execution” .

Rd is the destination register.

Rn is the register holding the first operand.

Operand2 is a flexible second operand. See “Flexible Second Operand” for details of the options

Operation

The AND, EOR, and ORR instructions perform bitwise AND, Exclusive OR, and OR operations on the values in *Rn* and *Operand2*.

The BIC instruction performs an AND operation on the bits in *Rn* with the complements of the corresponding bits in the value of *Operand2*.

The ORN instruction performs an OR operation on the bits in *Rn* with the complements of the corresponding bits in the value of *Operand2*.

Restrictions

Do not use SP and do not use PC.

Condition Flags

If S is specified, these instructions:

- Update the N and Z flags according to the result
- Can update the C flag during the calculation of *Operand2*, see “Flexible Second Operand”
- Do not affect the V flag.

Examples

```
AND      R9, R2, #0xFF00
ORREQ    R2, R0, R5
ANDS     R9, R8, #0x19
EORS     R7, R11, #0x18181818
BIC      R0, R1, #0xab
ORN       R7, R11, R14, ROR #4
ORNS     R7, R11, R14, ASR #32
```

11.6.6.9 SMUAD and SMUSD

Signed Dual Multiply Add and Signed Dual Multiply Subtract

Syntax

op{X}{*cond*} *Rd*, *Rn*, *Rm*

where:

op is one of:

SMUAD Signed Dual Multiply Add.

SMUADX Signed Dual Multiply Add Reversed.

SMUSD Signed Dual Multiply Subtract.

SMUSDX Signed Dual Multiply Subtract Reversed.

If *X* is present, the multiplications are bottom × top and top × bottom.
If the *X* is omitted, the multiplications are bottom × bottom and top × top.

cond is an optional condition code, see “Conditional Execution” .

Rd is the destination register.

Rn, Rm are registers holding the first and second operands.

Operation

The SMUAD instruction interprets the values from the first and second operands as two signed halfwords in each operand. This instruction:

- Optionally rotates the halfwords of the second operand.
- Performs two signed 16 × 16-bit multiplications.
- Adds the two multiplication results together.
- Writes the result of the addition to the destination register.

The SMUSD instruction interprets the values from the first and second operands as two’s complement signed integers. This instruction:

- Optionally rotates the halfwords of the second operand.
- Performs two signed 16 × 16-bit multiplications.
- Subtracts the result of the top halfword multiplication from the result of the bottom halfword multiplication.
- Writes the result of the subtraction to the destination register.

Restrictions

In these instructions:

- Do not use SP and do not use PC.

Condition Flags

Sets the Q flag if the addition overflows. The multiplications cannot overflow.

Examples

```
SMUAD    R0, R4, R5  ; Multiplies bottom halfword of R4 with the bottom
                     ; halfword of R5, adds multiplication of top halfword
                     ; of R4 with top halfword of R5, writes to R0
SMUADX   R3, R7, R4  ; Multiplies bottom halfword of R7 with top halfword
                     ; of R4, adds multiplication of top halfword of R7
                     ; with bottom halfword of R4, writes to R3
SMUSD    R3, R6, R2  ; Multiplies bottom halfword of R4 with bottom halfword
                     ; of R6, subtracts multiplication of top halfword of R6
                     ; with top halfword of R3, writes to R3
SMUSDX   R4, R5, R3  ; Multiplies bottom halfword of R5 with top halfword of
```

11.6.6.10 SMUL and SMULW

Signed Multiply (halfwords) and Signed Multiply (word by halfword)

Syntax

```
op{XY}{cond} Rd,Rn, Rm
op{Y}{cond} Rd. Rn, Rm
```

For *SMULXY* only:

op is one of:

SMUL{*XY*} Signed Multiply (halfwords).

X and *Y* specify which halfword of the source registers *Rn* and *Rm* is used as the first and second multiply operand.
If *X* is B, then the bottom halfword, bits [15:0] of *Rn* is used.
If *X* is T, then the top halfword, bits [31:16] of *Rn* is used.If *Y* is B, then the bot tom halfword, bits [15:0], of *Rm* is used.
If *Y* is T, then the top halfword, bits [31:16], of *Rm* is used.

SMULW{Y} Signed Multiply (word by halfword).

Y specifies which halfword of the source register *Rm* is used as the second mul tiply operand.
If *Y* is B, then the bottom halfword (bits [15:0]) of *Rm* is used.
If *Y* is T, then the top halfword (bits [31:16]) of *Rm* is used.

cond is an optional condition code, see “Conditional Execution” .

Rd is the destination register.

Rn, Rm are registers holding the first and second operands.

Operation

The SMULBB, SMULTB, SMULBT and SMULTT instructions interprets the values from *Rn* and *Rm* as four signed 16-bit integers. These instructions:

- Multiplies the specified signed halfword, Top or Bottom, values from *Rn* and *Rm*.
- Writes the 32-bit result of the multiplication in *Rd.*

The SMULWT and SMULWB instructions interprets the values from *Rn* as a 32-bit signed integer and *Rm* as two halfword 16-bit signed integers. These instructions:

- Multiplies the first operand and the top, T suffix, or the bottom, B suffix, halfword of the second operand.
- Writes the signed most significant 32 bits of the 48-bit result in the destination register.

Restrictions

In these instructions:

- Do not use SP and do not use PC.
- *RdHi* and *RdLo* must be different registers.

Examples

```
SMULBT       R0, R4, R5  ; Multiplies the bottom halfword of R4 with the
                         ; top halfword of R5, multiplies results and
                         ; writes to R0
SMULBB       R0, R4, R5  ; Multiplies the bottom halfword of R4 with the
                         ; bottom halfword of R5, multiplies results and
                         ; writes to R0
SMULTT       R0, R4, R5  ; Multiplies the top halfword of R4 with the top
                         ; halfword of R5, multiplies results and writes
                         ; to R0
SMULTB       R0, R4, R5  ; Multiplies the top halfword of R4 with the
```

11.6.11.3 DMB

Data Memory Barrier.

Syntax

```
DMB{cond}
```

where:

cond is an optional condition code, see “Conditional Execution” .

Operation

DMB acts as a data memory barrier. It ensures that all explicit memory accesses that appear, in program order, before the DMB instruction are completed before any explicit memory accesses that appear, in program order, after the DMB instruction. DMB does not affect the ordering or execution of instructions that do not access memory.

Condition Flags

This instruction does not change the flags.

Examples

```
DMB  ; Data Memory Barrier
```

11.10.1.4 SysTick Calibration Value Register

Name: SYST_CALIB

Access: Read-write

Reset: 0x000000000

31	30	29	28	27	26	25	24
NOREF	SKEW	–					

23	22	21	20	19	18	17	16
TENMS							

15	14	13	12	11	10	9	8
TENMS							

7	6	5	4	3	2	1	0
TENMS							

The SysTick SYST_CSR register indicates the SysTick calibration properties.

- **NOREF: No Reference Clock**

It indicates whether the device provides a reference clock to the processor:

0: Reference clock provided.

1: No reference clock provided.

If your device does not provide a reference clock, the SYST_CSR.CLKSOURCE bit reads-as-one and ignores writes.

- **SKEW**

It indicates whether the TENMS value is exact:

0: TENMS value is exact.

1: TENMS value is inexact, or not given.

An inexact TENMS value can affect the suitability of SysTick as a software real time clock.

- **TENMS: Ten Milliseconds**

The reload value for 10 ms (100 Hz) timing is subject to system clock skew errors. If the value reads as zero, the calibration value is not known.

Read as 0x000030D4. The SysTick calibration value is fixed at 0x000030D4 (12500), which allows the generation of a time base of 1 ms with SysTick clock at 12.5 MHz (100/8 = 12.5 MHz).

11.11.2.1 MPU Type Register

Name: MPU_TYPE

Access: Read-write

Reset: 0x00000800

31	30	29	28	27	26	25	24
–							

23	22	21	20	19	18	17	16
IREGION							

15	14	13	12	11	10	9	8
DREGION							

7	6	5	4	3	2	1	0
–							SEPARATE

The MPU_TYPE register indicates whether the MPU is present, and if so, how many regions it supports.

- **IREGION: Instruction Region**

Indicates the number of supported MPU instruction regions.

Always contains 0x00. The MPU memory map is unified and is described by the DREGION field.

- **DREGION: Data Region**

Indicates the number of supported MPU data regions:

0x08 = Eight MPU regions.

- **SEPARATE: Separate Instruction**

Indicates support for unified or separate instruction and date memory maps:

0: Unified.

- **WKUPDBC: Wake-up Inputs Debouncer Period**

Value	Name	Description
0	IMMEDIATE	Immediate, no debouncing, detected active at least on one Slow Clock edge.
1	3_SCLK	WKUPx shall be in its active state for at least 3 SLCK periods
2	32_SCLK	WKUPx shall be in its active state for at least 32 SLCK periods
3	512_SCLK	WKUPx shall be in its active state for at least 512 SLCK periods
4	4096_SCLK	WKUPx shall be in its active state for at least 4,096 SLCK periods
5	32768_SCLK	WKUPx shall be in its active state for at least 32,768 SLCK periods
6	Reserved	Reserved
7	Reserved	Reserved

- **LPDBC: Low Power DeBounCer Period**

Value	Name	Description
0	DISABLE	Disable the low power debouncer.
1	2_RTCOUT0	WKUP0/1 in its active state for at least 2 RTCOUT0 periods
2	3_RTCOUT0	WKUP0/1 in its active state for at least 3 RTCOUT0 periods
3	4_RTCOUT0	WKUP0/1 in its active state for at least 4 RTCOUT0 periods
4	5_RTCOUT0	WKUP0/1 in its active state for at least 5 RTCOUT0 periods
5	6_RTCOUT0	WKUP0/1 in its active state for at least 6 RTCOUT0 periods
6	7_RTCOUT0	WKUP0/1 in its active state for at least 7 RTCOUT0 periods
7	8_RTCOUT0	WKUP0/1 in its active state for at least 8 RTCOUT0 periods

19.5.3 EEFC Flash Status Register

Name: EEFC_FSR

Address: 0x400E0A08

Access: Read-only

Offset: 0x08

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
–	–	–	–	FLERR	FLOCKE	FCMDE	FRDY

- **FRDY: Flash Ready Status**

0: The Enhanced Embedded Flash Controller (EEFC) is busy.

1: The Enhanced Embedded Flash Controller (EEFC) is ready to start a new command.

When it is set, this flags triggers an interrupt if the FRDY flag is set in the EEFC_FMR register.

This flag is automatically cleared when the Enhanced Embedded Flash Controller (EEFC) is busy.

- **FCMDE: Flash Command Error Status**

0: No invalid commands and no bad keywords were written in the Flash Mode Register EEFC_FMR.

1: An invalid command and/or a bad keyword was/were written in the Flash Mode Register EEFC_FMR.

This flag is automatically cleared when EEFC_FSR is read or EEFC_FCR is written.

- **FLOCKE: Flash Lock Error Status**

0: No programming/erase of at least one locked region has happened since the last read of EEFC_FSR.

1: Programming/erase of at least one locked region has happened since the last read of EEFC_FSR.

This flag is automatically cleared when EEFC_FSR is read or EEFC_FCR is written.

- **FLERR: Flash Error Status**

0: No Flash Memory error occurred at the end of programming (EraseVerify or WriteVerify test has passed).

1: A Flash Memory error occurred at the end of programming (EraseVerify or WriteVerify test has failed).

20. Fast Flash Programming Interface (FFPI)

20.1 Description

The Fast Flash Programming Interface (FFPI) provides parallel high-volume programming using a standard gang programmer. The parallel interface is fully handshaked and the device is considered to be a standard EEPROM. Additionally, the parallel protocol offers an optimized access to all the embedded Flash functionalities.

Although the Fast Flash Programming Mode is a dedicated mode for high volume programming, this mode is not designed for in-situ programming.

20.2 Embedded Characteristics

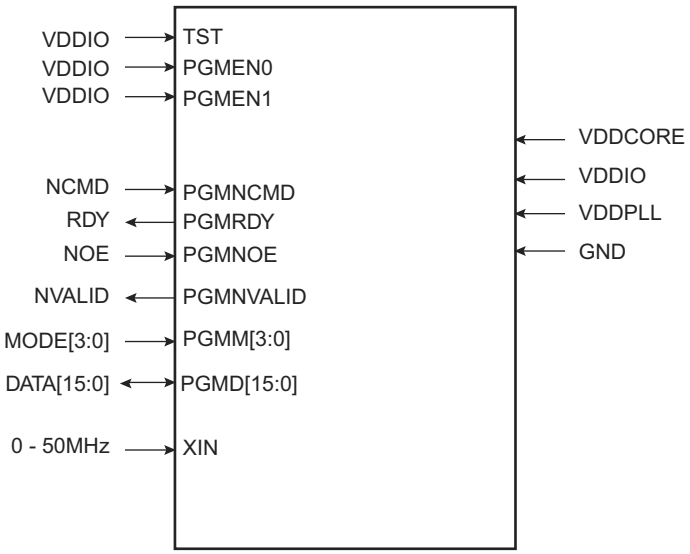
- Programming Mode for High-volume Flash Programming Using Gang Programmer
  - Offers Read and Write Access to the Flash Memory Plane
  - Enables Control of Lock Bits and General-purpose NVM Bits
  - Enables Security Bit Activation
  - Disabled Once Security Bit is Set
- Parallel Fast Flash Programming Interface
  - Provides an 16-bit Parallel Interface to Program the Embedded Flash
  - Full Handshake Protocol

20.3 Parallel Fast Flash Programming

20.3.1 Device Configuration

In Fast Flash Programming Mode, the device is in a specific test mode. Only a certain set of pins is significant. The rest of the PIOs are used as inputs with a pull-up. The crystal oscillator is in bypass mode. Other pins must be left unconnected.

Figure 20-1. SAM4NxB/C Parallel Programming Interface



23.5.5 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.

25.14 Clock Switching Details

25.14.1 Master Clock Switching Timings

Table 25-1 and give the worst case timings required for the Master Clock to switch from one selected clock to another one. This is in the event that the prescaler is de-activated. When the prescaler is activated, an additional time of 64 clock cycles of the newly selected clock has to be added.

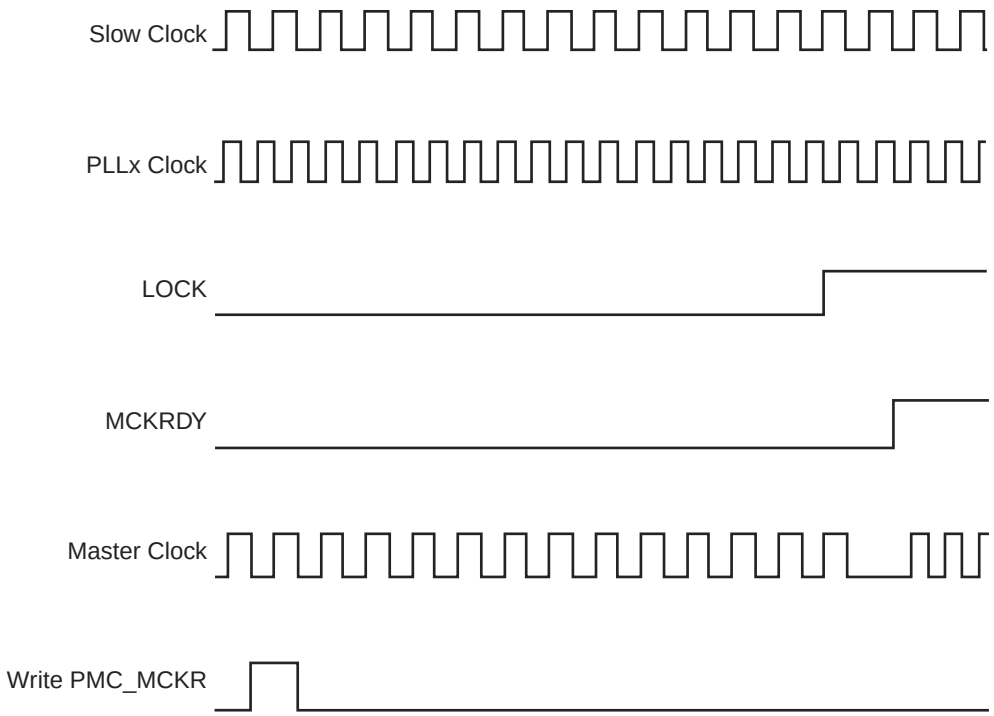
Table 25-1. Clock Switching Timings (Worst Case)

From To	Main Clock	SLCK	PLL Clock
Main Clock	–	4 x SLCK + 2.5 x Main Clock	3 x PLL Clock + 4 x SLCK + 1 x Main Clock
SLCK	0.5 x Main Clock + 4.5 x SLCK	–	3 x PLL Clock + 5 x SLCK
PLL Clock	0.5 x Main Clock + 4 x SLCK + PLLCOUNT x SLCK + 2.5 x PLLx Clock	2.5 x PLL Clock + 5 x SLCK + PLLCOUNT x SLCK	2.5 x PLL Clock + 4 x SLCK + PLLCOUNT x SLCK

Notes: 1. PLL designates the PLLA .
2. PLLCOUNT designates PLLACOUNT .

25.14.2 Clock Switching Waveforms

Figure 25-4. Switch Master Clock from Slow Clock to PLLx Clock



25.16.12 PMC Interrupt Enable Register

Name: PMC_IER

Address: 0x400E0460

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	XT32KERR	–	–	CFDEV	MOSCRCS	MOSCSELS
15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	–	LOCKA	MOSCXTS

- **MOSCXTS: Main Crystal Oscillator Status Interrupt Enable**
- **LOCKA: PLLA Lock Interrupt Enable**
- **MCKRDY: Master Clock Ready Interrupt Enable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Enable**
- **MOSCSELS: Main Oscillator Selection Status Interrupt Enable**
- **MOSCRCS: Main On-Chip RC Status Interrupt Enable**
- **CFDEV: Clock Failure Detector Event Interrupt Enable**
- **XT32KERR: Slow Crystal Oscillator Error Interrupt Enable**

Figure 28-7 shows Transmit Data Register Empty (TDRE), Receive Data Register (RDRF) and Transmission Register Empty (TXEMPTY) status flags behavior within the SPI_SR (Status Register) during an 8-bit data transfer in fixed mode and no Peripheral Data Controller involved.

Figure 28-7. Status Register Flags Behavior

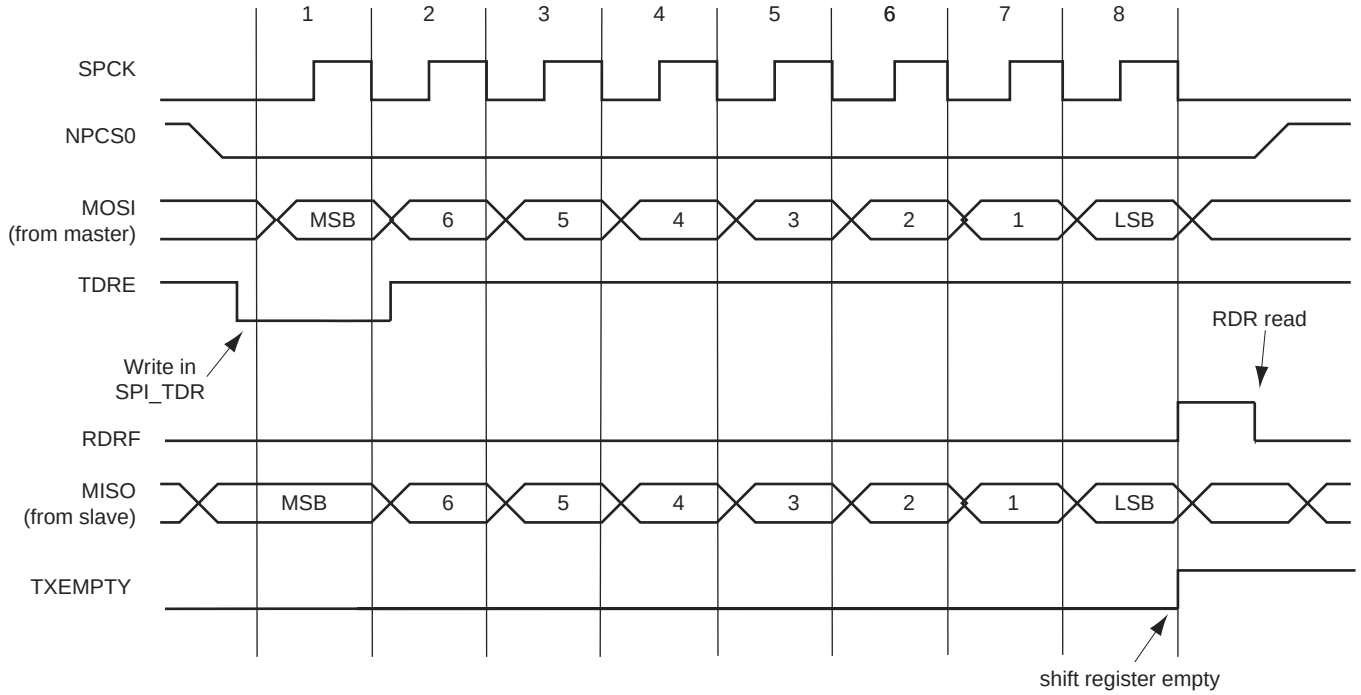
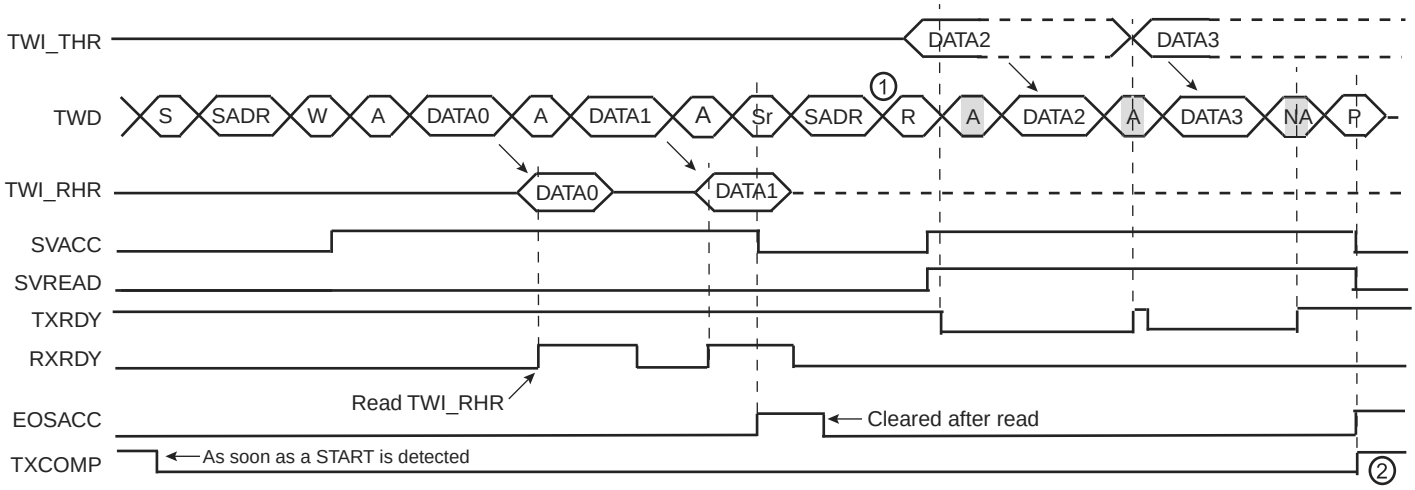


Figure 28-8 shows Transmission Register Empty (TXEMPTY), End of RX buffer (ENDRX), End of TX buffer (ENDTX), RX Buffer Full (RXBUFF) and TX Buffer Empty (TXBUFE) status flags behavior within the SPI_SR (Status Register) during an 8-bit data transfer in fixed mode with the Peripheral Data Controller involved. The PDC is programmed to transfer and receive three data. The next pointer and counter are not used. The RDRF and TDRE are not shown because these flags are managed by the PDC when using the PDC.

Figure 29-31. Repeated Start + Reversal from Write to Read Mode

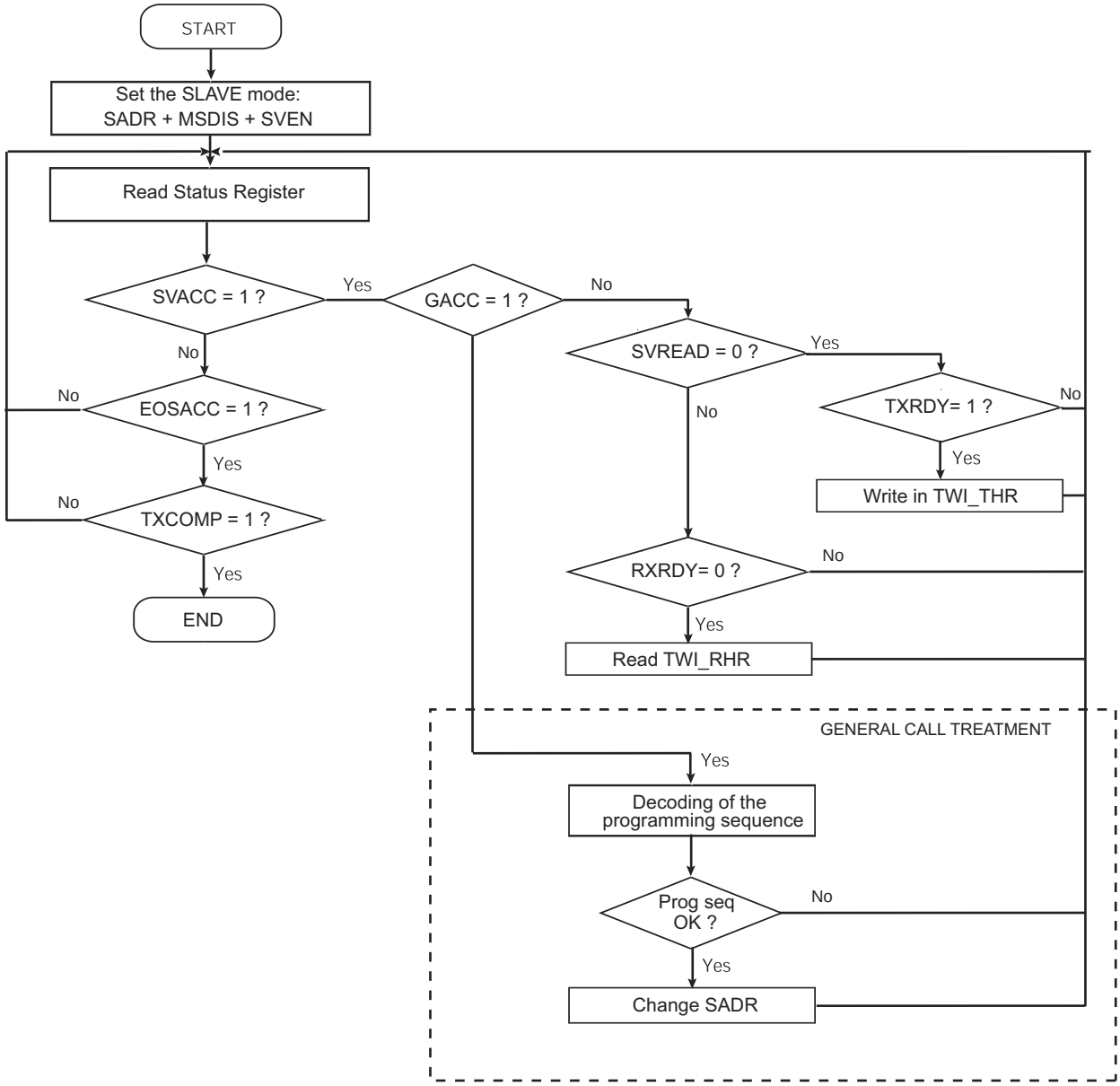


Notes:
1. In this case, if TWI_THR has not been written at the end of the read command, the clock is automatically stretched before the ACK.
2. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

29.10.6 Read Write Flowcharts

The flowchart shown in Figure 29-32 on page 561 gives an example of read and write operations in Slave mode. A polling or interrupt method can be used to check the status bits. The interrupt method requires that the interrupt enable register (TWI_IER) be configured first.

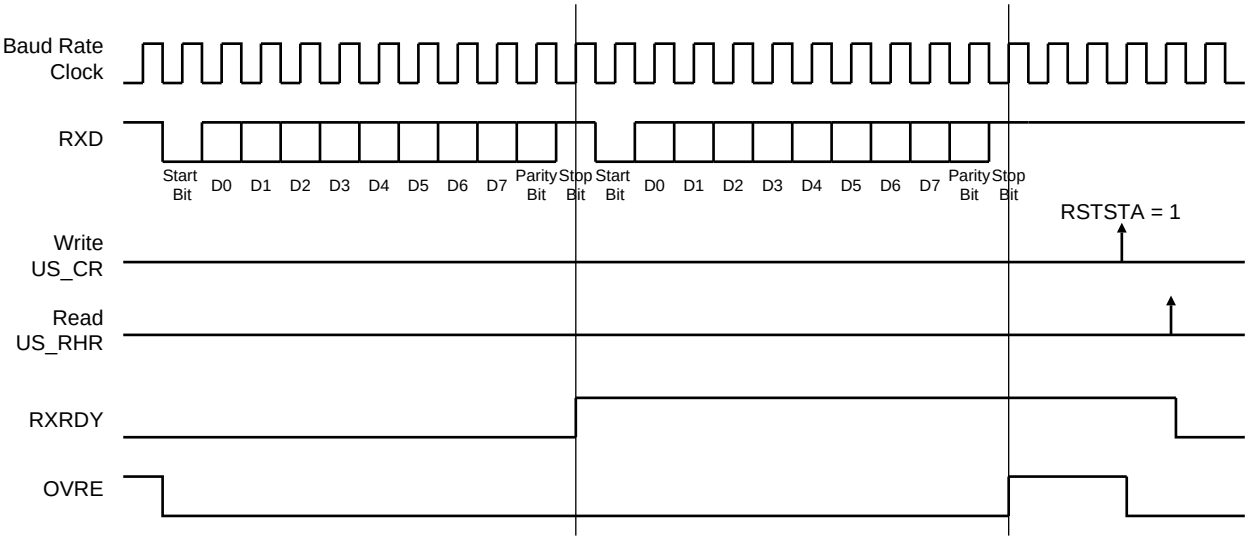
Figure 29-32. Read Write Flowchart in Slave Mode



31.7.3.4 Receiver Operations

When a character reception is completed, it is transferred to the Receive Holding Register (US_RHR) and the RXRDY bit in the Status Register (US_CSR) rises. If a character is completed while the RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into US_RHR and overwrites the previous one. The OVRE bit is cleared by writing the Control Register (US_CR) with the RSTSTA (Reset Status) bit to 1.

Figure 31-11. Receiver Status



31.7.3.5 Parity

The USART supports five parity modes selected by programming the PAR field in the Mode Register (US_MR). The PAR field also enables the Multidrop mode, see “Multidrop Mode” on page 610. Even and odd parity bit generation and error detection are supported.

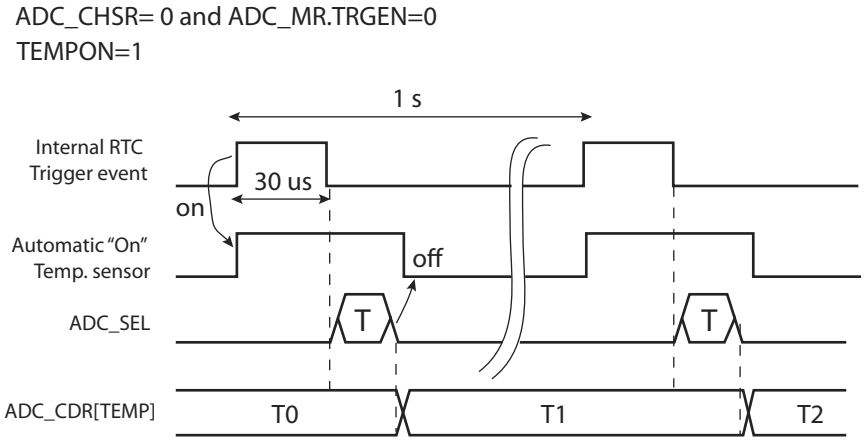
If even parity is selected, the parity generator of the transmitter drives the parity bit to 0 if a number of 1s in the character data bit is even, and to 1 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If odd parity is selected, the parity generator of the transmitter drives the parity bit to 1 if a number of 1s in the character data bit is even, and to 0 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If the mark parity is used, the parity generator of the transmitter drives the parity bit to 1 for all characters. The receiver parity checker reports an error if the parity bit is sampled to 0. If the space parity is used, the parity generator of the transmitter drives the parity bit to 0 for all characters. The receiver parity checker reports an error if the parity bit is sampled to 1. If parity is disabled, the transmitter does not generate any parity bit and the receiver does not report any parity error.

Table 31-9 shows an example of the parity bit for the character 0x41 (character ASCII “A”) depending on the configuration of the USART. Because there are two bits to 1, 1 bit is added when a parity is odd, or 0 is added when a parity is even.

Table 31-9. Parity Bit Examples

Character	Hexa	Binary	Parity Bit	Parity Mode
A	0x41	0100 0001	1	Odd
A	0x41	0100 0001	0	Even
A	0x41	0100 0001	1	Mark
A	0x41	0100 0001	0	Space
A	0x41	0100 0001	None	None

Figure 34-8. Temperature Conversion Only



Moreover it is possible to raise a flag only if there is a predefined change in the temperature measure. The user can define a range of temperature or a threshold in the ADC_TEMPCWR register, and the mode of comparison that can be programmed into the ADC_TEMPMR register by means of TEMPCMPMOD bitfield. These values will define the way the TEMPCHG flag will be raised in the ADC_ISR register.

This flag can be used to trigger an interrupt only if there is something to update or modify in the system, resulting from a temperature change and not being interrupted for each temperature measure end of conversion, especially if there is no change.

In any case, if TEMPON is set, the temperature can be read at anytime in ADC_CDR16 without any specific software intervention.

34.6.10 Enhanced Resolution Mode and Digital Averaging Function

The Enhanced Resolution Mode is enabled if LOWRES is cleared in the ADC Mode Register (ADC_MR), and the OSR bitfield is set to 1, 2 in ADC Extended Mode Register (ADC_EMR). The enhancement is based on a digital averaging function.

FREERUN must be set to 0 when digital averaging is used (OSR differs from 0 in ADC_EMR register).

There is no averaging on the last index channel if the measure is triggered by RTC event (see Section 34.6.9 on page 746).

In this mode the ADC Controller will trade conversion performance for accuracy by averaging multiple samples, thus providing a digital low-pass filter function.

If 1-bit enhancement resolution is selected (OSR=1 in the ADC_EMR register), the ADC effective sample rate is the maximum ADC sample rate divided by 4, therefore the oversampling ratio is 4.

When the 2-bit enhancement resolution is selected (OSR=2 in the ADC_EMR register), the ADC effective sample rate is the maximum ADC sample rate divided by 16 (oversampling ratio is 16).

The selected oversampling ratio applies to all enabled channels except for the temperature sensor channel when triggered by RTC event.

The average result is valid into the ADC_CDRn (n corresponding to the index of the channel) registers only if EOCn flag is set in the ADC_ISR register and OVREn flag is cleared in the ADC_OVER register. The average result for all channels is valid in the ADC_LCDR register only if DRDY is set and GOVRE is cleared in the ADC_ISR register.

36.3.4 Peripheral Power Consumption in Active Mode

Table 36-18. Power Consumption on VDDCORE (V_{DDIO} = 3.3V, V_{DDCORE} = 1.08V, T_A = 25°C)

Peripheral	Consumption (Typ)	Unit
PIO Controller A (PIOA)	2.68	µA/MHz
PIO Controller B (PIOB)	1.47	
PIO Controller C (PIOC)	2.94	
UART0	2.60	
UART1	3.09	
UART2	2.96	
UART3	0.93	
USART0	4.07	
USART1	4.03	
USART2	4.67	
PWM	2.21	
TWI0	3.28	
TWI1	2.84	
TWI2	2.86	
SPI	2.68	
Timer Counter 0 (TC0)	1.95	
TC1	1.15	
TC2	2.42	
TC3	1.93	
TC4	1.22	
TC5	1.23	
ADC	3.06	
DACC	1.74	

36.4 Oscillator Characteristics

36.4.1 32 kHz RC Oscillator Characteristics

Table 36-19. 32 kHz RC Oscillator Characteristics

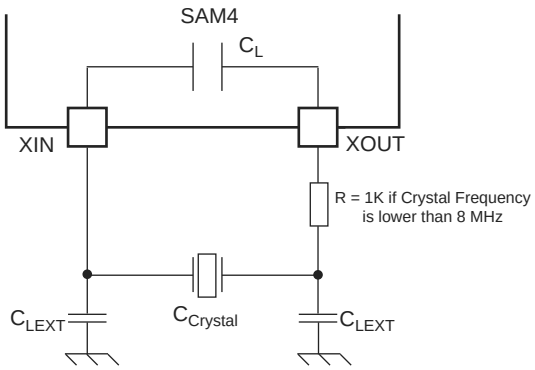
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OSC}	RC Oscillator Frequency		20	32	44	kHz
	Frequency Supply Dependency		-3		3	%/V
	Frequency Temperature Dependency	Over temperature range (-40 to 85 °C) versus T_A 25°C	-7		7	%
Duty	Duty Cycle		45	50	55	%
t_{START}	Start-up Time				100	µs
I_{DDON}	Current Consumption	After Startup Time Temp. range = -40 to 125 °C Typical consumption at 2.2V supply and T_A 25 °C		540	860	nA

36.4.5 3 to 20 MHz Crystal Oscillator Characteristics

Table 36-23. 3 to 20 MHz Crystal Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OSC}	Operating Frequency	Normal mode with crystal	3	16	20	MHz
$V_{rip(VDDPLL)}$	Supply Ripple Voltage (on VDDPLL)	RMS value, 10 kHz to 10 MHz			30	mV
	Duty Cycle		40	50	60	%
t_{START}	Startup Time	3 MHz, C_{SHUNT} = 3 pF 8 MHz, C_{SHUNT} = 7 pF 16 MHz, C_{SHUNT} = 7 pF with C_m = 8 fF 16 MHz, C_{SHUNT} = 7 pF with C_m = 1.6 fF 20 MHz, C_{SHUNT} = 7 pF			14.5 4 1.4 2.5 1	ms
I_{DDON}	Current Consumption (on VDDIO)	3 MHz 8 MHz 16 MHz 20 MHz		230 300 390 450	350 400 470 560	µA
P_{ON}	Drive Level	3 MHz 8 MHz 16 MHz, 20 MHz			15 30 50	µW
R_f	Internal Resistor	Between XIN and XOUT		0.5		MΩ
C_{LEXT}	Maximum External Capacitor on XIN and XOUT				17	pF
$C_{INTLOAD}$	Internal Load Capacitance	Integrated load capacitance ((XIN) (GND) and (XOUT)(GND) in series)	7.5	9	10.5	pF
C_{LOAD}	Internal Equivalent Load Capacitance	Integrated Load Capacitance (XIN and XOUT in series)	7.5	9	10.5	pF

Figure 36-11. 3 to 20 MHz Crystal Oscillator Schematics



$C_{LEXT} = 2 \times (C_{crystal} - C_L - C_{PCB})$

where:

C_{PCB} is the capacitance of the printed circuit board (PCB) track layout from the crystal to the SAM4 pin.

Table 37-8. 100-ball VFBGA Package Reference - Soldering Information (Substrate Level)

Ball Land	Diameter 0.27 mm
Soldering Mask Opening	275 µm

Table 37-9. Device and 100-ball VFBGA Package Maximum Weight

SAM4N	75	mg

Table 37-10. 100-ball VFBGA Package Characteristics

Moisture Sensitivity Level	3

Table 37-11. 100-ball VFBGA Package Reference

JEDEC Drawing Reference	MO-275-BBE-1
JESD97 Classification	e8

This package respects the recommendations of the NEMI User Group.