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	120MHz
Connectivity	EBI/EMI, I²C, IrDA, Memory Card, SPI, SSC, UART/USART, USB
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	128K x 8
Voltage - Supply (Vcc/Vdd)	1.62V ~ 3.6V
Data Converters	A/D 16x12b; 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/atsam4s8ca-an

12.4.1.9 Application Program Status Register

Name: APSR

Access: Read/Write

Reset: 0x000000000

31	30	29	28	27	26	25	24
N	Z	C	V	Q	–		

23	22	21	20	19	18	17	16
–				GE[3:0]			

15	14	13	12	11	10	9	8
–							

7	6	5	4	3	2	1	0
–							

The APSR contains the current state of the condition flags from previous instruction executions.

- **N: Negative Flag**

0: Operation result was positive, zero, greater than, or equal

1: Operation result was negative or less than.

- **Z: Zero Flag**

0: Operation result was not zero

1: Operation result was zero.

- **C: Carry or Borrow Flag**

Carry or borrow flag:

0: Add operation did not result in a carry bit or subtract operation resulted in a borrow bit

1: Add operation resulted in a carry bit or subtract operation did not result in a borrow bit.

- **V: Overflow Flag**

0: Operation did not result in an overflow

1: Operation resulted in an overflow.

- **Q: DSP Overflow and Saturation Flag**

Sticky saturation flag:

0: Indicates that saturation has not occurred since reset or since the bit was last cleared to zero

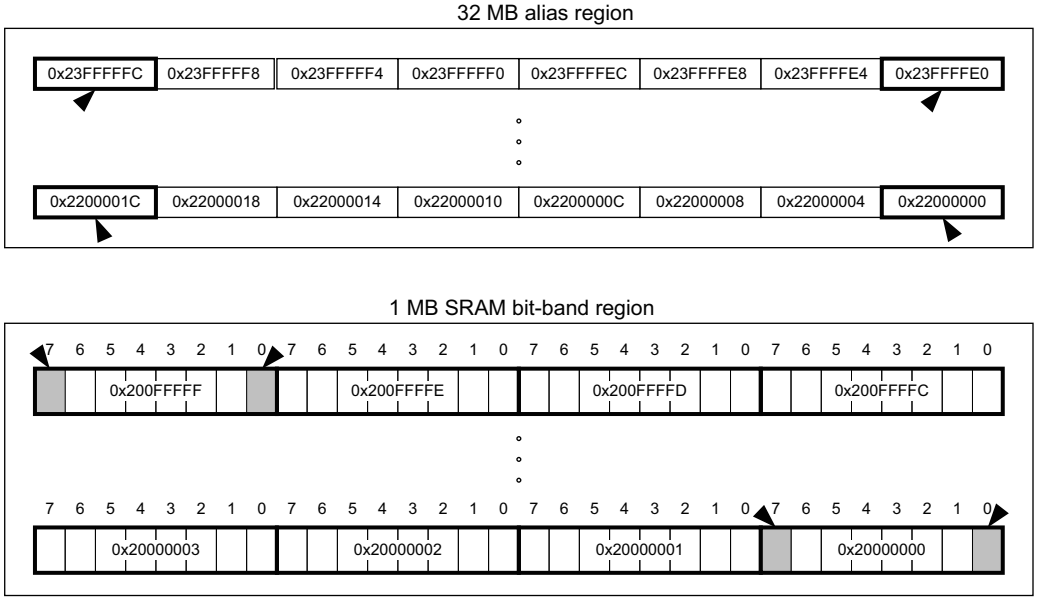
1: Indicates when an SSAT or USAT instruction results in saturation.

This bit is cleared to zero by software using an MRS instruction.

- **GE[19:16]: Greater Than or Equal Flags**

See “SEL” for more information.

Figure 12-4. Bit-band Mapping



Directly Accessing an Alias Region

Writing to a word in the alias region updates a single bit in the bit-band region.

Bit[0] of the value written to a word in the alias region determines the value written to the targeted bit in the bit-band region. Writing a value with bit[0] set to 1 writes a 1 to the bit-band bit, and writing a value with bit[0] set to 0 writes a 0 to the bit-band bit.

Bits[31:1] of the alias word have no effect on the bit-band bit. Writing 0x01 has the same effect as writing 0xFF. Writing 0x00 has the same effect as writing 0x0E.

Reading a word in the alias region:

- 0x00000000 indicates that the targeted bit in the bit-band region is set to 0
- 0x00000001 indicates that the targeted bit in the bit-band region is set to 1

Directly Accessing a Bit-band Region

"Behavior of Memory Accesses" describes the behavior of direct byte, halfword, or word accesses to the bit-band regions.

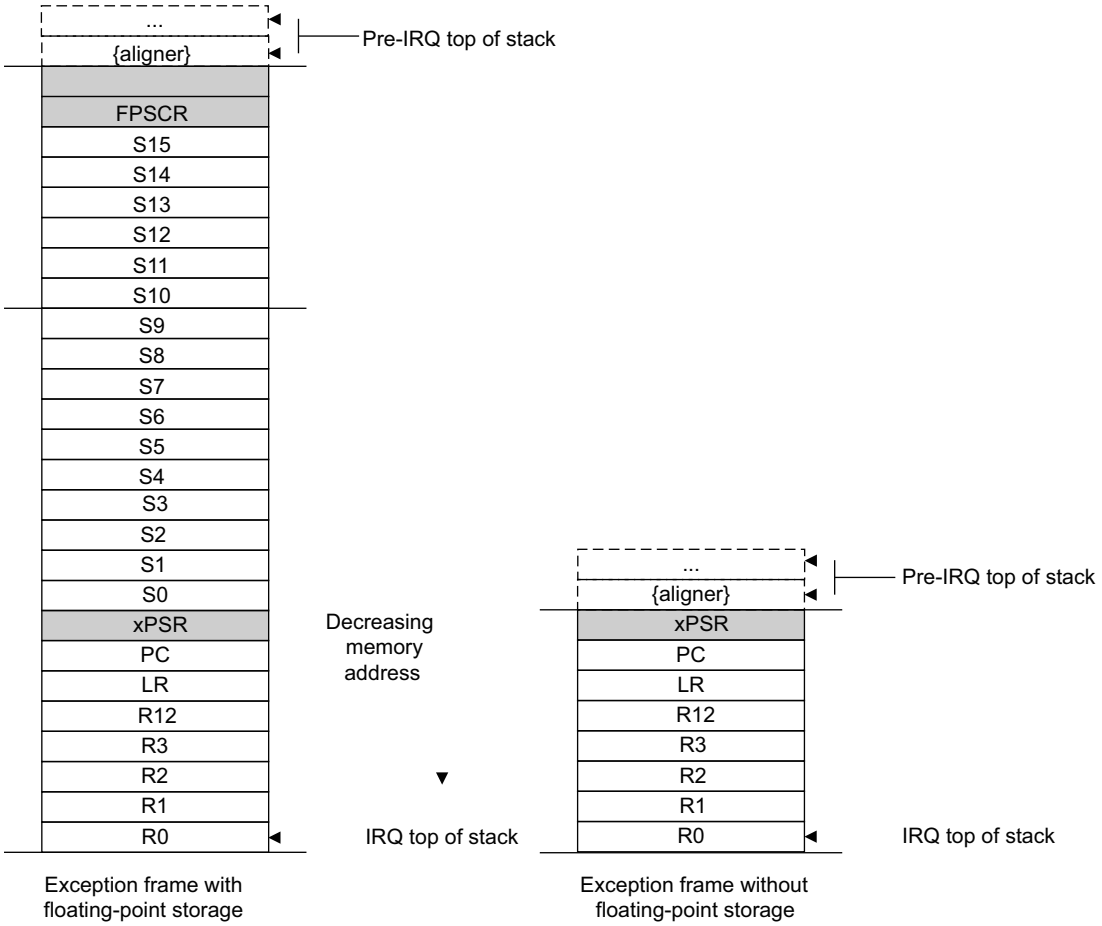
12.4.2.6 Memory Endianness

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0–3 hold the first stored word, and bytes 4–7 hold the second stored word. "Little-endian Format" describes how words of data are stored in memory.

Little-endian Format

In little-endian format, the processor stores the least significant byte of a word at the lowest-numbered byte, and the most significant byte at the highest-numbered byte. For example:

Figure 12-7. Exception Stack Frame



Immediately after stacking, the stack pointer indicates the lowest address in the stack frame. The alignment of the stack frame is controlled via the STKALIGN bit of the Configuration Control Register (CCR).

The stack frame includes the return address. This is the address of the next instruction in the interrupted program. This value is restored to the PC at exception return so that the interrupted program resumes.

In parallel to the stacking operation, the processor performs a vector fetch that reads the exception handler start address from the vector table. When stacking is complete, the processor starts executing the exception handler. At the same time, the processor writes an EXC_RETURN value to the LR. This indicates which stack pointer corresponds to the stack frame and what operation mode the processor was in before the entry occurred.

If no higher priority exception occurs during the exception entry, the processor starts executing the exception handler and automatically changes the status of the corresponding pending interrupt to active.

If another higher priority exception occurs during the exception entry, the processor starts executing the exception handler for this exception and does not change the pending status of the earlier exception. This is the late arrival case.

Exception Return

An Exception return occurs when the processor is in Handler mode and executes one of the following instructions to load the EXC_RETURN value into the PC:

- An LDM or POP instruction that loads the PC
- An LDR instruction with the PC as the destination.
- A BX instruction using any register.

Operation

The register access operation in MSR depends on the privilege level. Unprivileged software can only access the APSR. See “Application Program Status Register” . Privileged software can access all special registers.

In unprivileged software writes to unallocated or execution state bits in the PSR are ignored.

Note: When the user writes to BASEPRI_MAX, the instruction writes to BASEPRI only if either:
Rn is non-zero and the current BASEPRI value is 0
Rn is non-zero and less than the current BASEPRI value.

See “MRS” .

Restrictions

Rn must not be SP and must not be PC.

Condition Flags

This instruction updates the flags explicitly based on the value in *Rn*.

Examples

```
MSR  CONTROL, R1 ; Read R1 value and write it to the CONTROL register
```

12.6.11.8 NOP

No Operation.

Syntax

`NOP{cond}`

where:

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

Operation

NOP does nothing. NOP is not necessarily a time-consuming NOP. The processor might remove it from the pipeline before it reaches the execution stage.

Use NOP for padding, for example to place the following instruction on a 64-bit boundary.

Condition Flags

This instruction does not change the flags.

Examples

```
NOP  ; No operation
```

15.5.3 Real-time Timer Value Register

Name: RTT_VR

Address: 0x400E1438

Access: Read-only

31	30	29	28	27	26	25	24
CRTV							

23	22	21	20	19	18	17	16
CRTV							

15	14	13	12	11	10	9	8
CRTV							

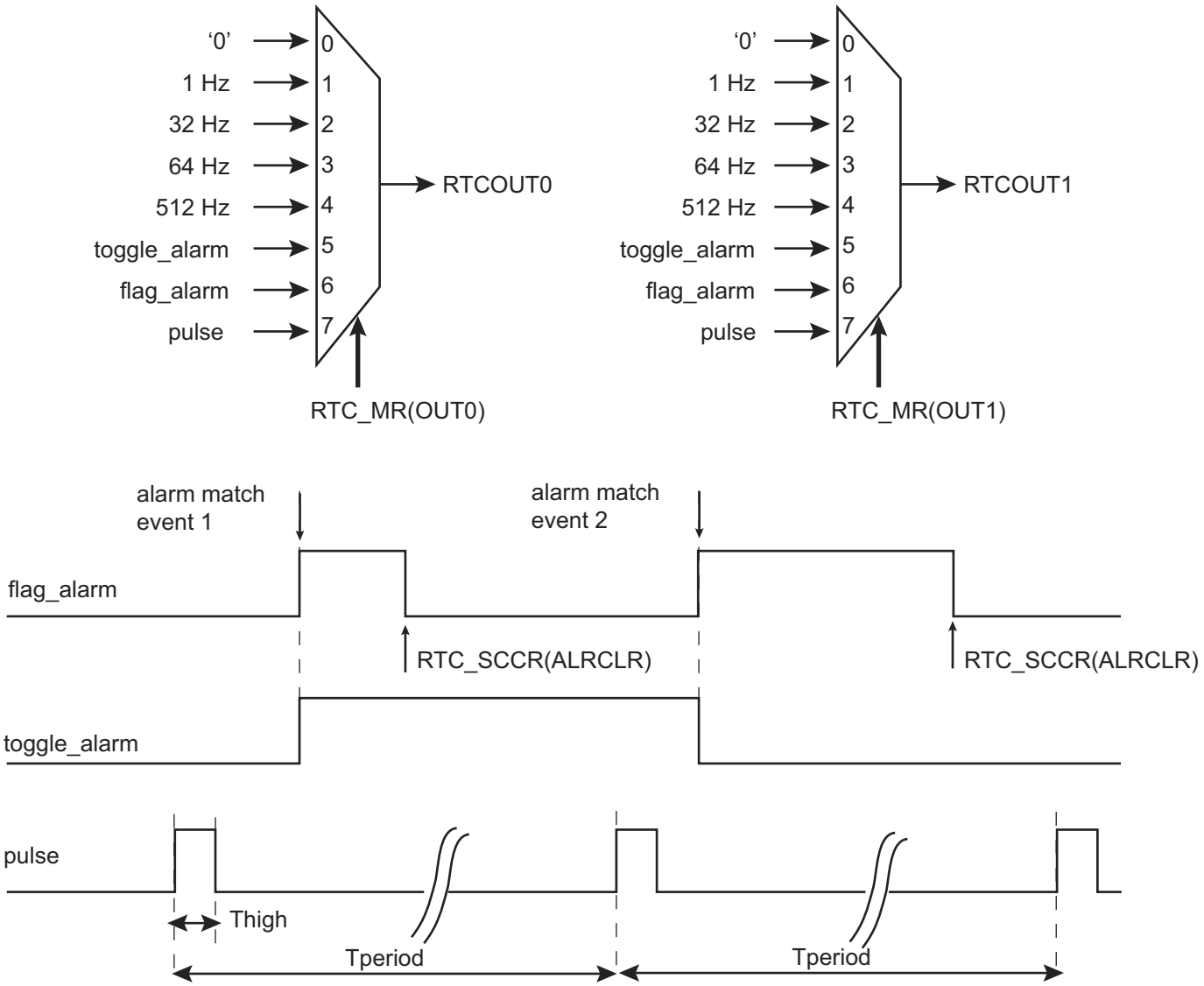
7	6	5	4	3	2	1	0
CRTV							

- **CRTV: Current Real-time Value**

Returns the current value of the Real-time Timer.

Note: As CRTV can be updated asynchronously, it must be read twice at the same value.

Figure 16-5. Waveform Generation



22. Cortex-M Cache Controller (CMCC)

22.1 Description

The Cortex-M Cache Controller (CMCC) is a 4-Way set associative unified cache controller. It integrates a controller, a tag directory, data memory, metadata memory and a configuration interface.

22.2 Embedded Characteristics

- Physically addressed and physically tagged
- L1 data cache set to 2 Kbytes
- L1 cache line size set to 16 Bytes
- L1 cache integrates 32-bit bus master interface
- Unified direct mapped cache architecture
- Unified 4-Way set associative cache architecture
- Write through cache operations, read allocate
- Round Robin victim selection policy
- Event Monitoring, with one programmable 32-bit counter
- Configuration registers accessible through Cortex-M Private Peripheral Bus (PPB)
- Cache interface includes cache maintenance operations registers

23.6.1 Transfer Address Register

Name: TR_ADDR

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: Transfer Address**

25.5 Arbitration

The Bus Matrix provides an arbitration technique that reduces latency when conflicting cases occur; for example, when two or more masters try to access the same slave at the same time. One arbiter per AHB slave is provided to arbitrate each slave differently.

The Bus Matrix provides the user with two arbitration types for each slave:

1. Round-robin arbitration (default)
2. Fixed priority arbitration

The field ARBT of MATRIX_SCFG is used to select the type of arbitration.

Each algorithm may be complemented by selecting a default master configuration for each slave.

In case of re-arbitration, specific conditions apply. See Section 25.5.1 “Arbitration Rules“.

25.5.1 Arbitration Rules

Each arbiter has the ability to arbitrate between requests of two or more masters. To avoid burst breaking and to provide the maximum throughput for slave interfaces, arbitration should take place during the following cycles:

1. Idle cycles: When a slave is not connected to any master or is connected to a master which is not currently accessing it.
2. Single cycles: When a slave is performing a single access.
3. End of burst cycles: When the current cycle is the last cycle of a burst transfer. For a defined burst length, predicted end of burst matches the size of the transfer but is managed differently for undefined burst length. See Section 25.5.1.1 “Undefined Length Burst Arbitration” on page 432“.
4. Slot cycle limit: When the slot cycle counter has reached the limit indicating that the current master access is too long and must be broken. See Section 25.5.1.2 “Slot Cycle Limit Arbitration” on page 432.

25.5.1.1 Undefined Length Burst Arbitration

In order to prevent slave handling during undefined length bursts (INCR), the Bus Matrix provides specific logic to re-arbitrate before the end of the INCR transfer.

A predicted end of burst is used for defined length burst transfer, which is selected between the following:

1. Infinite: No predicted end of burst is generated and therefore INCR burst transfer will never be broken.
2. Four-beat bursts: Predicted end of burst is generated at the end of each four beat boundary inside INCR transfer.
3. Eight-beat bursts: Predicted end of burst is generated at the end of each eight beat boundary inside INCR transfer.
4. Sixteen-beat bursts: Predicted end of burst is generated at the end of each sixteen beat boundary inside INCR transfer.

This selection can be done through the field ULBT of the Master Configuration Registers (MATRIX_MCFG).

25.5.1.2 Slot Cycle Limit Arbitration

The Bus Matrix contains specific logic to break accesses that are too long, such as very long bursts on a very slow slave (e.g. an external low-speed memory). At the beginning of the burst access, a counter is loaded with the value previously written in the SLOT_CYCLE field of the related MATRIX_SCFG and decreased at each clock cycle. When the counter reaches zero, the arbiter has the ability to re-arbitrate at the end of the current byte, half-word or word transfer.

25.5.2 Round-Robin Arbitration

Bus Matrix arbiters use the round-robin algorithm to dispatch the requests from different masters to the same slave. If two or more masters make a request at the same time, the master with the lowest number is serviced first. The others are then serviced in a round-robin manner.

27.6.2 Receive Counter Register

Name: PERIPH_RCR

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
RXCTR							
7	6	5	4	3	2	1	0
RXCTR							

- **RXCTR: Receive Counter Register**

RXCTR must be set to receive buffer size.

When a half-duplex peripheral is connected to the PDC, RXCTR = TXCTR.

0: Stops peripheral data transfer to the receiver.

1–65535: Starts peripheral data transfer if the corresponding channel is active.

29.17.7 PMC Clock Generator Main Oscillator Register

Name: CKGR_MOR

Address: 0x400E0420

Access: Read/Write

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

23	22	21	20	19	18	17	16
KEY							

15	14	13	12	11	10	9	8
MOSCXTST							

7	6	5	4	3	2	1	0
–	MOSCRCF			MOSCRCEN	WAITMODE	MOSCXTBY	MOSCXTEN

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

- **MOSCXTEN: Main Crystal Oscillator Enable**

A crystal must be connected between XIN and XOUT.

0: The main crystal oscillator is disabled.

1: The main crystal oscillator is enabled. MOSCXTBY must be cleared.

When MOSCXTEN is set, the MOSCXTS flag is set once the main crystal oscillator start-up time is achieved.

- **MOSCXTBY: Main Crystal Oscillator Bypass**

0: No effect.

1: The main crystal oscillator is bypassed. MOSCXTEN must be cleared. An external clock must be connected on XIN.

When MOSCXTBY is set, the MOSCXTS flag in PMC_SR is automatically set.

Clearing MOSCXTEN and MOSCXTBY bits resets the MOSCXTS flag.

Note: When the main crystal oscillator bypass is disabled (MOSCXTBY = 0), the MOSCXTS flag must be read at 0 in PMC_SR before enabling the main crystal oscillator (MOSCXTEN = 1).

- **WAITMODE: Wait Mode Command (Write-only)**

0: No effect.

1: Puts the device in Wait mode.

- **MOSCRCEN: Main On-Chip RC Oscillator Enable**

0: The main on-chip RC oscillator is disabled.

1: The main on-chip RC oscillator is enabled.

When MOSCRCEN is set, the MOSCRCS flag is set once the main on-chip RC oscillator start-up time is achieved.

31.6.24 PIO Peripheral ABCD Select Register 1

Name: PIO_ABCDSR1

Access: Read/Write

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 the PIO Write Protection Mode Register.

- **P0–P31: Peripheral Select**

If the same bit is set to 0 in PIO_ABCDSR2:

0: Assigns the I/O line to the Peripheral A function.

1: Assigns the I/O line to the Peripheral B function.

If the same bit is set to 1 in PIO_ABCDSR2:

0: Assigns the I/O line to the Peripheral C function.

1: Assigns the I/O line to the Peripheral D function.

31.6.54 PIO Parallel Capture Reception Holding Register

Name: PIO_PCRHR

Address: 0x400E0F64 (PIOA), 0x400E1164 (PIOB), 0x400E1364 (PIOC)

Access: Read-only

31	30	29	28	27	26	25	24
RDATA							

23	22	21	20	19	18	17	16
RDATA							

15	14	13	12	11	10	9	8
RDATA							

7	6	5	4	3	2	1	0
RDATA							

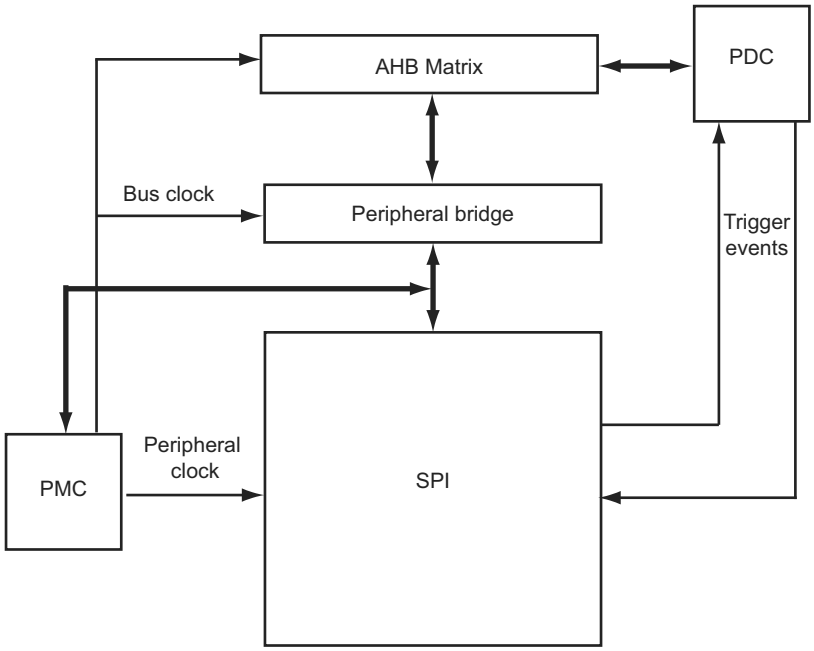
- **RDATA: Parallel Capture Mode Reception Data**

If DSIZE = 0 in PIO_PCMR, only the 8 LSBs of RDATA are useful.

If DSIZE = 1 in PIO_PCMR, only the 16 LSBs of RDATA are useful.

33.3 Block Diagram

Figure 33-1. Block Diagram



33.4 Application Block Diagram

Figure 33-2. Application Block Diagram: Single Master/Multiple Slave Implementation

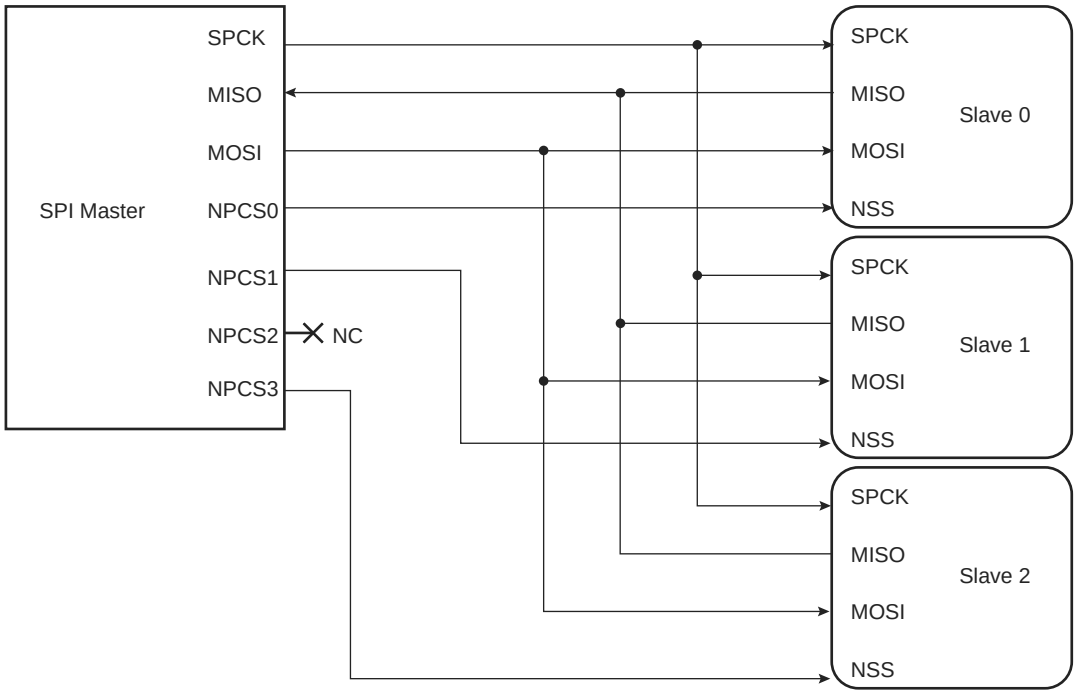


Figure 36-22. Timeguard Operations

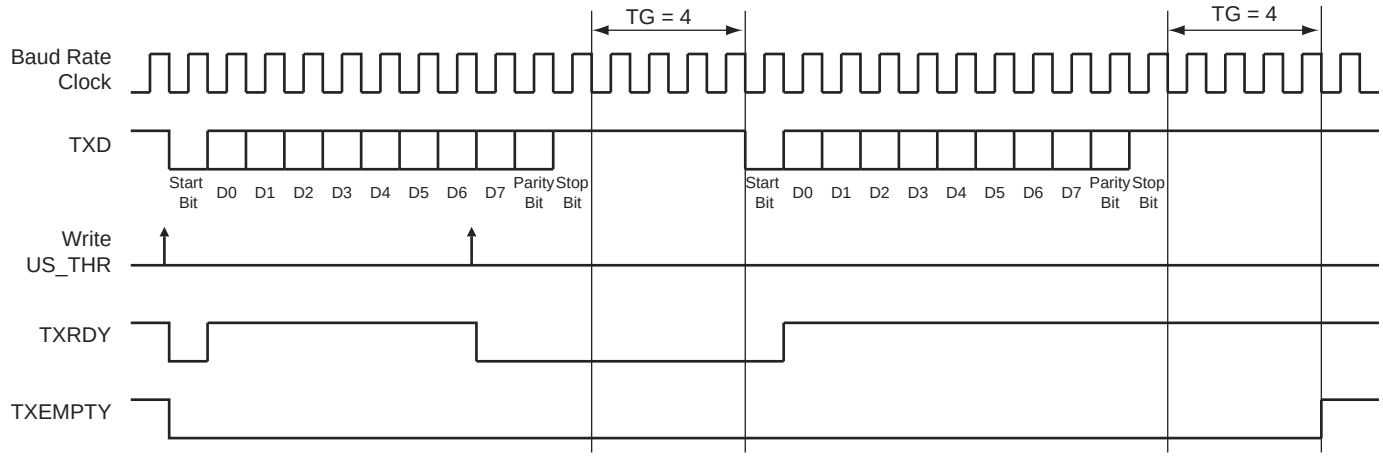


Table 36-9 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the baud rate.

Table 36-9. Maximum Timeguard Length Depending on Baud Rate

Baud Rate (bit/s)	Bit Time (µs)	Timeguard (ms)
1,200	833	212.50
9,600	104	26.56
14,400	69.4	17.71
19,200	52.1	13.28
28,800	34.7	8.85
38,400	26	6.63
56,000	17.9	4.55
57,600	17.4	4.43
115,200	8.7	2.21

36.6.3.11 Receiver Time-out

The Receiver Time-out provides support in handling variable-length frames. This feature detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the US_CSR rises and can generate an interrupt, thus indicating to the driver an end of frame.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of the Receiver Time-out register (US_RTOR). If the TO field is written to 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in the US_CSR remains at 0. Otherwise, the receiver loads a 16-bit counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in US_CSR rises. Then, the user can either:

- Stop the counter clock until a new character is received. This is performed by writing a 1 to the STTTO (Start Time-out) bit in the US_CR. In this case, the idle state on RXD before a new character is received will not provide a time-out. This prevents having to handle an interrupt before a character is received and allows waiting for the next idle state on RXD after a frame is received.
- Obtain an interrupt while no character is received. This is performed by writing a 1 to the RETTO (Reload and Start Time-out) bit in the US_CR. If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

37.7 Timer Counter (TC) User Interface

Table 37-6. Register Mapping

Offset[1]	Register	Name	Access	Reset
0x00 + channel * 0x40 + 0x00	Channel Control Register	TC_CCR	Write-only	–
0x00 + channel * 0x40 + 0x04	Channel Mode Register	TC_CMR	Read/Write	0
0x00 + channel * 0x40 + 0x08	Stepper Motor Mode Register	TC_SMMR	Read/Write	0
0x00 + channel * 0x40 + 0x0C	Reserved	–	–	–
0x00 + channel * 0x40 + 0x10	Counter Value	TC_CV	Read-only	0
0x00 + channel * 0x40 + 0x14	Register A	TC_RA	Read/Write[2]	0
0x00 + channel * 0x40 + 0x18	Register B	TC_RB	Read/Write[2]	0
0x00 + channel * 0x40 + 0x1C	Register C	TC_RC	Read/Write	0
0x00 + channel * 0x40 + 0x20	Status Register	TC_SR	Read-only	0
0x00 + channel * 0x40 + 0x24	Interrupt Enable Register	TC_IER	Write-only	–
0x00 + channel * 0x40 + 0x28	Interrupt Disable Register	TC_IDR	Write-only	–
0x00 + channel * 0x40 + 0x2C	Interrupt Mask Register	TC_IMR	Read-only	0
0xC0	Block Control Register	TC_BCR	Write-only	–
0xC4	Block Mode Register	TC_BMR	Read/Write	0
0xC8	QDEC Interrupt Enable Register	TC_QIER	Write-only	–
0xCC	QDEC Interrupt Disable Register	TC_QIDR	Write-only	–
0xD0	QDEC Interrupt Mask Register	TC_QIMR	Read-only	0
0xD4	QDEC Interrupt Status Register	TC_QISR	Read-only	0
0xD8	Fault Mode Register	TC_FMR	Read/Write	0
0xE4	Write Protection Mode Register	TC_WPMR	Read/Write	0
0xE8–0xFC	Reserved	–	–	–

Notes: 1. Channel index ranges from 0 to 2.
2. Read-only if TC_CMRx.WAVE = 0

- **ETRGS: External Trigger**

0: No effect.

1: Enables the External Trigger Interrupt.

37.7.15 TC QDEC Interrupt Enable Register

Name: TC_QIER

Address: 0x400100C8 (0), 0x400140C8 (1)

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
–	–	–	–	–	QERR	DIRCHG	IDX

- **IDX: Index**

0: No effect.

1: Enables the interrupt when a rising edge occurs on IDX input.

- **DIRCHG: Direction Change**

0: No effect.

1: Enables the interrupt when a change on rotation direction is detected.

- **QERR: Quadrature Error**

0: No effect.

1: Enables the interrupt when a quadrature error occurs on PHA, PHB.

38.14.18HSMCI Write Protection Status Register

Name: HSMCI_WPSR

Address: 0x400000E8

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
WPVSRC							
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 protection violation has occurred since the last read of the HSMCI_WPSR.

1: A write protection violation has occurred since the last read of the HSMCI_WPSR. 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**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.

Table 45-27. 48-lead LQFP Package Reference

JEDEC Drawing Reference	
JESD97 Classification	e3