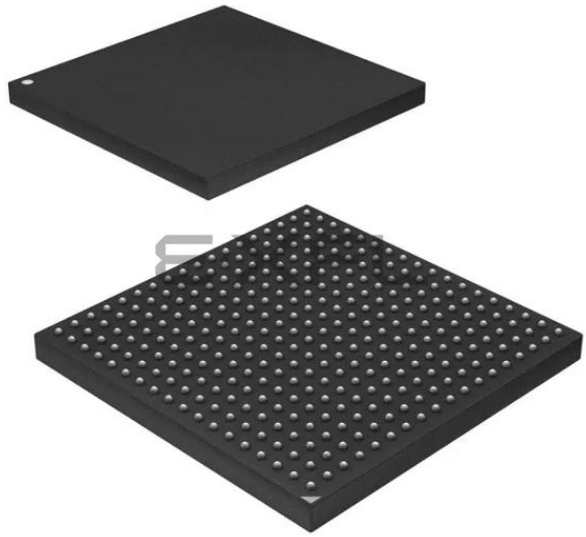


**Welcome to E-XFL.COM**

## 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 | ARM® Cortex®-A5 |
| Number of Cores/Bus Width | 1 Core, 32-Bit |
| Speed | 536MHz |
| Co-Processors/DSP | - |
| RAM Controllers | LPDDR, LPDDR2, DDR2 |
| Graphics Acceleration | No |
| Display & Interface Controllers | LCD, Touchscreen |
| Ethernet | 10/100/1000Mbps (1) |
| SATA | - |
| USB | USB 2.0 (3) |
| Voltage - I/O | 1.2V, 1.8V, 3.3V |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Security Features | AES, SHA, TDES, TRNG |
| Package / Case | 324-LFBGA |
| Supplier Device Package | 324-LFBGA (15x15) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/atsama5d33a-cu |

### 10.4.4 CP 15 Register Access

CP15 registers can only be accessed in privileged mode by:

- MCR (Move to Coprocessor from ARM Register) instruction is used to write an ARM register to CP15.
- MRC (Move to ARM Register from Coprocessor) instruction is used to read the value of CP15 to an ARM register.

Other instructions such as CDP, LDC, STC can cause an undefined instruction exception.

The assembler code for these instructions is:

```
MCR/MRC{cond} p15, opcode_1, Rd, CRn, CRm, opcode_2.
```

The MCR/MRC instructions bit pattern is shown below:

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| cond | | | | 1 | 1 | 1 | 0 |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| opcode_1 | | | L | CRn | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| Rd | | | | 1 | 1 | 1 | 1 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| opcode_2 | | | 1 | CRm | | | |

- **CRm[3:0]: Specified Coprocessor Action**

Determines specific coprocessor action. Its value is dependent on the CP15 register used. For details, refer to CP15 specific register behavior.

- **opcode_2[7:5]**

Determines specific coprocessor operation code. By default, set to 0.

- **Rd[15:12]: ARM Register**

Defines the ARM register whose value is transferred to the coprocessor. If R15 is chosen, the result is unpredictable.

- **CRn[19:16]: Coprocessor Register**

Determines the destination coprocessor register.

- **L: Instruction Bit**

0: MCR instruction

1: MRC instruction

- **opcode_1[23:20]: Coprocessor Code**

Defines the coprocessor specific code. Value is c15 for CP15.

- **cond [31:28]: Condition**

### 20.7.2 Shutdown Mode Register

**Name:** SHDW_MR

**Address:** 0xFFFFFE14

**Access:** Read/Write

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

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | RTCWKEN | – |

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

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CPTWK0 | | | | – | – | WKMODE0 | |

- **WKMODE0: Wake-up Mode 0**

| Value | Name | Description |
|---|---|---|
| 0 | NO_DETECTION | No detection is performed on the wake-up input |
| 1 | RISING_EDGE | Low to high transition triggers the detection process |
| 2 | FALLING_EDGE | High to low level transition triggers the detection process |
| 3 | ANY_EDGE | Any edge on the wake-up input triggers the detection process |

- **CPTWK0: Debounce Counter on Wake-up 0**

Defines the minimum duration of the WKUP1 pin after the occurrence of the selected triggering edge (WKMODE0).

The SHDN pin is released if the WKUP0 holds the selected level for (CPTWK * 16 + 1) consecutive Slow Clock cycles after the occurrence of the selected triggering edge on WKUP0.

- **RTCWKEN: Real-time Clock Wake-up Enable**

0 = The RTC Alarm signal has no effect on the Shutdown Controller.

1 = The RTC Alarm signal forces the de-assertion of the SHDN pin.

### 25.5.2 Fuse Mode Register

**Name:** FUSE_MR

**Address:** 0xFFFFE404

**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 |
| – | – | – | – | – | – | – | MSK |

- **MSK: Mask Fuse Status Registers**

0: No effect.

1: Masks the first 8 FUSE_SRx registers.

### 27.7.40 PIO Level Select Register

**Name:** PIO_LSR

**Address:** 0xFFFFF2C4 (PIOA), 0xFFFFF4C4 (PIOB), 0xFFFFF6C4 (PIOC), 0xFFFFF8C4 (PIOD), 0xFFFFFAC4 (PIOE)

**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 |

- **P0-P31: Level Interrupt Selection**

0: No effect.

1: The interrupt source is a level-detection event.

- **ENRDM: Enable Read Measure**

Reset value is 0.

0 (OFF): DQS/DDR_DATA phase error correction is disabled.

1 (ON): DQS/DDR_DATA phase error correction is enabled.

- **NB: Number of Banks**

Reset value is 4 banks. If LC_LPDDR1 is set to 1, NB is not relevant.

| Value | Name | Description |
|---|---|---|
| 0 | 4_BANKS | 4 banks memory devices |
| 1 | 8_BANKS | 8 banks. Only possible when using the DDR2-SDRAM and low-power DDR2-SDRAM devices. |

- **NDQS: Not DQS**

Reset value is 1; not DQS is disabled.

0: (ENABLED) Not DQS is enabled.

1: (DISABLED) Not DQS is disabled.

This field is found only in the DDR2-SDRAM devices.

- **DECOD: Type of Decoding**

Reset value is 0.

| Value | Name | Description |
|---|---|---|
| 0 | SEQUENTIAL | Method for address mapping where banks alternate at each last DDR-SDRAM page of the current bank. |
| 1 | INTERLEAVED | Method for address mapping where banks alternate at each SDRAM end page of the current bank. |

- **UNAL: Support Unaligned Access**

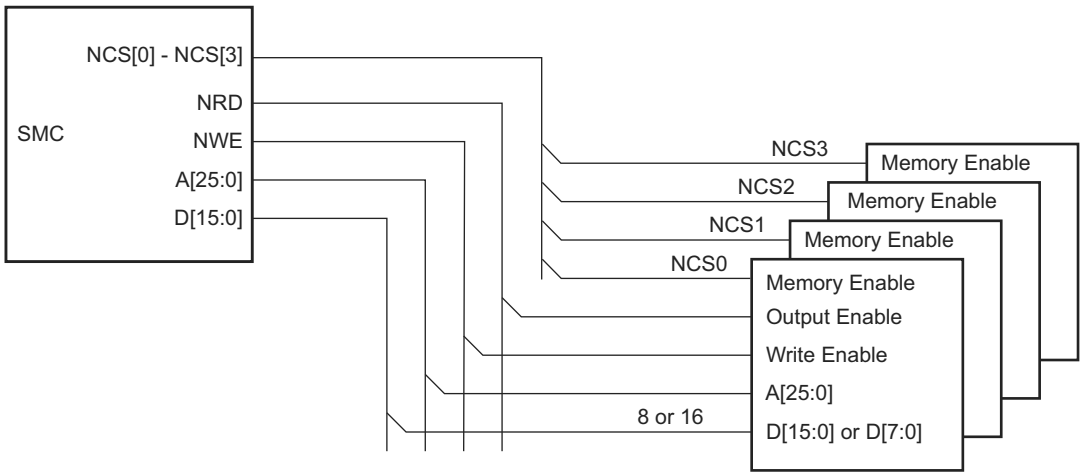
Reset value is 0; unaligned access is not supported.

0 (UNSUPPORTED): Unaligned access is not supported.

1 (SUPPORTED): Unaligned access is supported.

This mode is enabled with masters which have an AXI interface.

**Figure 30-3. Memory Connections for External Devices**




## 30.9 Connection to External Devices

### 30.9.1 Data Bus Width

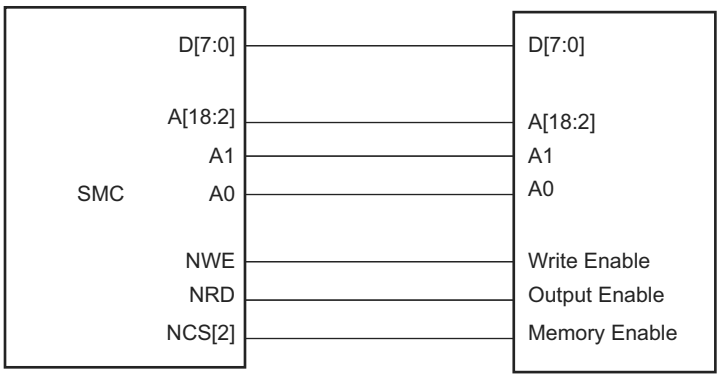
A data bus width of 8 or 16 bits can be selected for each chip select. This option is controlled by the field DBW in the SMC Mode Register (HSMC_MODE) for the corresponding chip select.

Figure 30-4 shows how to connect a 512K x 8-bit memory on NCS2. Figure 30-5 shows how to connect a 512K x 16-bit memory on NCS2.

### 30.9.2 Byte Write or Byte Select Access

Each chip select with a 16-bit data bus can operate with one of two different types of write access: Byte write or Byte select access. This is controlled by the BAT field of the HSMC_MODE register for the corresponding chip select.

**Figure 30-4. Memory Connection for an 8-bit Data Bus**



### 32.7.13 LCD Controller Interrupt Mask Register

**Name:** LCDC_LCDIMR

**Address:** 0xF0030034

**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 |
| – | – | PPIM | HCRIM | HEOIM | OVR2IM | OVR1IM | BASEIM |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | FIFOERRIM | – | DISPIM | DISIM | SOFIM |

- **SOFIM: Start of Frame Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DISIM: LCD Disable Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **DISPIM: Power UP/Down Sequence Terminated Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **FIFOERRIM: Output FIFO Error Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **BASEIM: Base Layer Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **OVR1IM: Overlay 1 Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **OVR2IM: Overlay 2 Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

- **HEOIM: High End Overlay Interrupt Mask Register**

0: Interrupt source is disabled.

1: Interrupt source is enabled.

### 32.7.57 Overlay 2 Layer Channel Status Register

**Name:** LCDC_OVRCHSR2

**Address:** 0xF0030248

**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 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | – | – | – | A2QSR | UPDATESR | CHSR |

- **CHSR: Channel Status Register**

When set to one this field disables the layer at the end of the current frame.

- **UPDATESR: Update Overlay Attributes In Progress**

When set to one this bit indicates that the overlay attributes will update on the next Frame.

- **A2QSR: Add To Queue Pending Register**

When set to one this bit indicates that the head pointer is still pending.

- **VDSCR: DMA Descriptor Loaded for V component**

When set to one this flag indicates that a descriptor has been loaded successfully. This flag is reset after a read operation.

- **VADD: Head Descriptor Loaded for V component**

When set to one this flag indicates that the descriptor pointed to by the head register has been loaded successfully. This flag is reset after a read operation.

- **VDONE: End of List Detected for V component**

When set to one this flag indicates that an End of List condition has occurred. This flag is reset after a read operation.

- **VOVR: Overflow Detected for V component**

When set to one this flag indicates that an overflow occurred. This flag is reset after a read operation.

### 37.6.15 Hash Register Top

**Name:** EMAC_HRT

**Address:** 0xF802C094

**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:**

Bits 63:32 of the hash address register. See “Hash Addressing” on page 1082.

### 39.7.3.7 Peripheral Chip Select Decoding

The user can program the SPI to operate with up to 15 peripherals by decoding the four Chip Select lines, NPCS0 to NPCS3 with 1 of up to 16 decoder/demultiplexer. This can be enabled by writing the PCSDEC bit at 1 in the Mode Register (SPI_MR).

When operating without decoding, the SPI makes sure that in any case only one chip select line is activated, i.e., one NPCS line driven low at a time. If two bits are defined low in a PCS field, only the lowest numbered chip select is driven low.

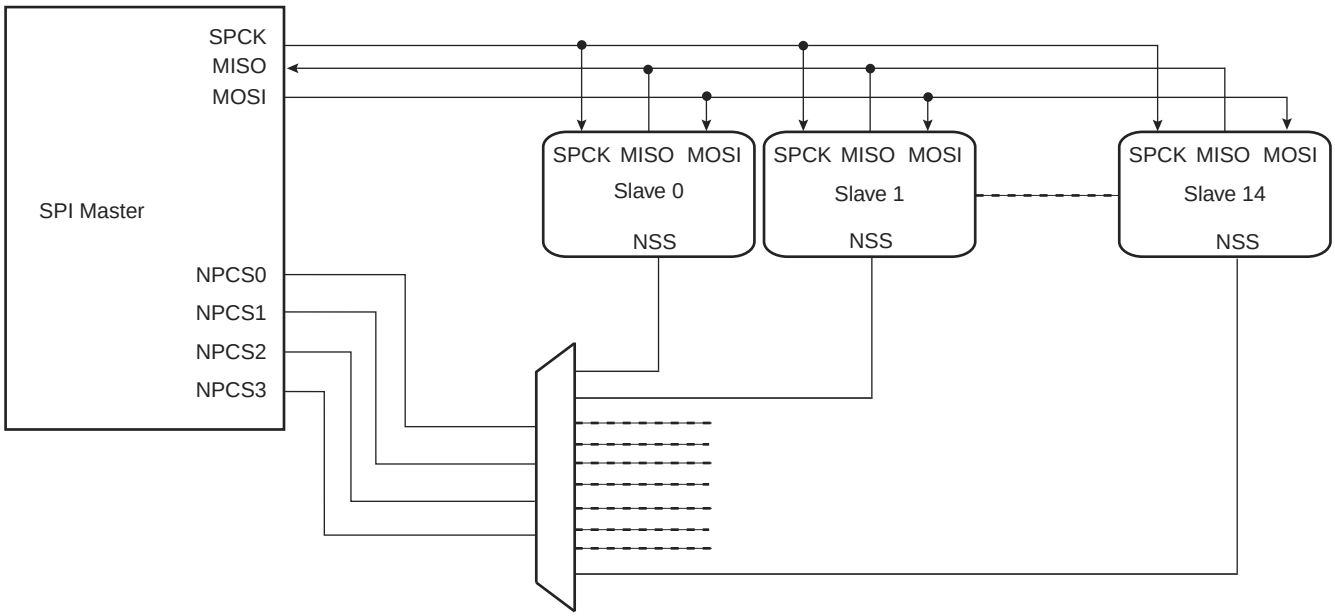
When operating with decoding, the SPI directly outputs the value defined by the PCS field on NPCS lines of either the Mode Register or the Transmit Data Register (depending on PS).

As the SPI sets a default value of 0xF on the chip select lines (i.e. all chip select lines at 1) when not processing any transfer, only 15 peripherals can be decoded.

The SPI has only four Chip Select Registers, not 15. As a result, when decoding is activated, each chip select defines the characteristics of up to four peripherals. As an example, SPI_CRS0 defines the characteristics of the externally decoded peripherals 0 to 3, corresponding to the PCS values 0x0 to 0x3. Thus, the user has to make sure to connect compatible peripherals on the decoded chip select lines 0 to 3, 4 to 7, 8 to 11 and 12 to 14. Figure 39-9 below shows such an implementation.

If the CSAAT bit is used, with or without the DMAC, the Mode Fault detection for NPCS0 line must be disabled. This is not needed for all other chip select lines since Mode Fault Detection is only on NPCS0.

**Figure 39-9. Chip Select Decoding Application Block Diagram: Single Master/Multiple Slave Implementation**




### 39.7.3.8 Peripheral Deselection without DMA

During a transfer of more than one data on a Chip Select without the DMA, the SPI_TDR is loaded by the processor, the flag TDRE rises as soon as the content of the SPI_TDR is transferred into the internal shift register. When this flag is detected high, the SPI_TDR can be reloaded. If this reload by the processor occurs before the end of the current transfer and if the next transfer is performed on the same chip select as the current transfer, the Chip Select is not de-asserted between the two transfers. But depending on the application software handling the SPI status register flags (by interrupt or polling method) or servicing other interrupts or other tasks, the processor may not reload the SPI_TDR in time to keep the chip select active (low). A null Delay Between Consecutive Transfer (DLYBCT) value in the SPI_CSR register, will give even less time for the processor to reload the SPI_TDR. With some SPI slave peripherals, requiring the chip select line to remain active (low) during a full set of transfers might lead to communication errors.

TXRDY used in Slave mode:

0: As soon as data is written in the TWI_THR, until this data has been transmitted and acknowledged (ACK or NACK).

1: It indicates that the TWI_THR is empty and that data has been transmitted and acknowledged.

If TXRDY is high and if a NACK has been detected, the transmission will be stopped. Thus when TRDY = NACK = 1, the programmer must not fill TWI_THR to avoid losing it.

*TXRDY behavior in Slave mode* can be seen in Figure 40-26 on page 1255, Figure 40-29 on page 1257, Figure 40-31 on page 1258 and Figure 40-32 on page 1259.

- **SVREAD: Slave Read (automatically set / reset)**

This bit is only used in Slave mode. When SVACC is low (no Slave access has been detected) SVREAD is irrelevant.

0: Indicates that a write access is performed by a Master.

1: Indicates that a read access is performed by a Master.

*SVREAD behavior* can be seen in Figure 40-26 on page 1255, Figure 40-27 on page 1256, Figure 40-31 on page 1258 and Figure 40-32 on page 1259.

- **SVACC: Slave Access (automatically set / reset)**

This bit is only used in Slave mode.

0: TWI is not addressed. SVACC is automatically cleared after a NACK or a STOP condition is detected.

1: Indicates that the address decoding sequence has matched (A Master has sent SADR). SVACC remains high until a NACK or a STOP condition is detected.

*SVACC behavior* can be seen in Figure 40-26 on page 1255, Figure 40-27 on page 1256, Figure 40-31 on page 1258 and Figure 40-32 on page 1259.

- **GACC: General Call Access (clear on read)**

This bit is only used in Slave mode.

0: No General Call has been detected.

1: A General Call has been detected. After the detection of General Call, if need be, the programmer may acknowledge this access and decode the following bytes and respond according to the value of the bytes.

*GACC behavior* can be seen in Figure 40-28 on page 1256.

- **OVRE: Overrun Error (clear on read)**

This bit is only used in Master mode.

0: TWI_RHR has not been loaded while RXRDY was set

1: TWI_RHR has been loaded while RXRDY was set. Reset by read in TWI_SR when TXCOMP is set.

- **NACK: Not Acknowledged (clear on read)**

NACK used in Master mode:

0: Each data byte has been correctly received by the far-end side TWI slave component.

1: A data byte or an address byte has not been acknowledged by the slave component. Set at the same time as TXCOMP.

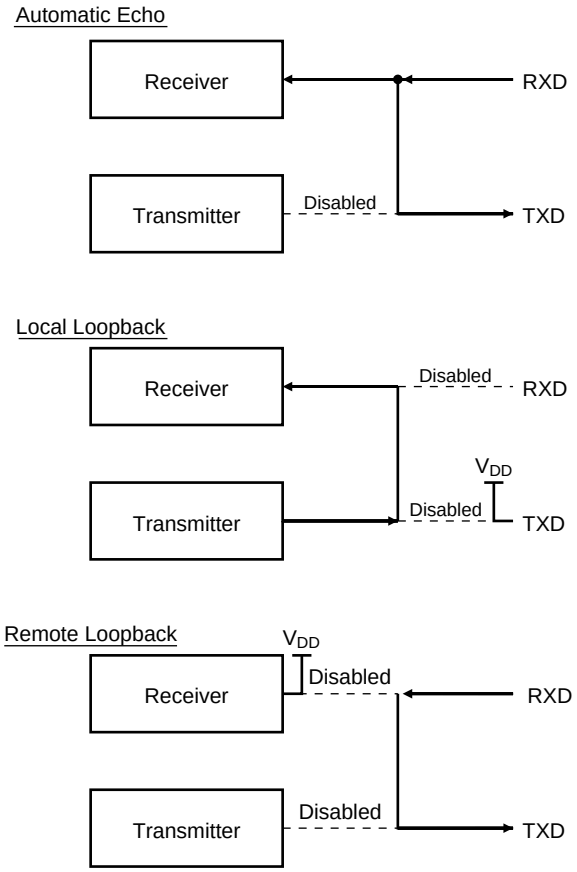
NACK used in Slave Read mode:

0: Each data byte has been correctly received by the Master.

1: In read mode, a data byte has not been acknowledged by the Master. When NACK is set the programmer must not fill TWI_THR even if TXRDY is set, because it means that the Master will stop the data transfer or re initiate it.

Note that in Slave Write mode all data are acknowledged by the TWI.

**Figure 42-12.Test Modes**




### 42.5.6 Debug Communication Channel Support

The Debug Unit handles the signals COMMRX and COMMTX that come from the Debug Communication Channel of the ARM Processor and are driven by the In-circuit Emulator.

The Debug Communication Channel contains two registers that are accessible through the ICE Breaker on the JTAG side and through the coprocessor 0 on the ARM Processor side.

As a reminder, the following instructions are used to read and write the Debug Communication Channel:

```
MRC                    p14, 0, Rd, c1, c0, 0
```

Returns the debug communication data read register into Rd

```
MCR                    p14, 0, Rd, c1, c0, 0
```

Writes the value in Rd to the debug communication data write register.

The bits COMMRX and COMMTX, which indicate, respectively, that the read register has been written by the debugger but not yet read by the processor, and that the write register has been written by the processor and not yet read by the debugger, are wired on the two highest bits of the status register DBGU_SR. These bits can generate an interrupt. This feature permits handling under interrupt a debug link between a debug monitor running on the target system and a debugger.

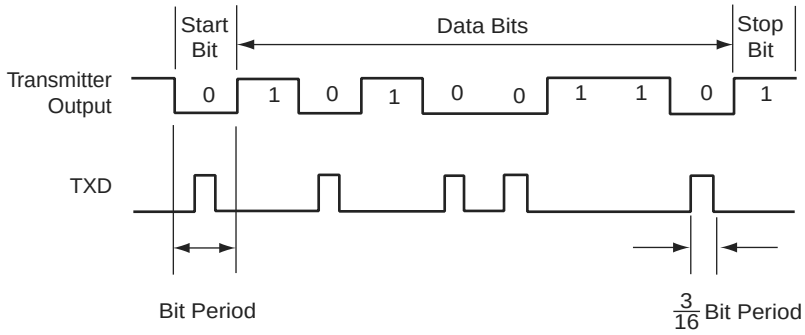
### 42.5.7 Chip Identifier

The Debug Unit features two chip identifier registers, DBGU_CIDR (Chip ID Register) and DBGU_EXID (Extension ID). Both registers contain a hard-wired value that is read-only. The first register contains the following fields:

- EXT - shows the use of the extension identifier register

Figure 44-33 shows an example of character transmission.

**Figure 44-33.IrDA Modulation**




### 44.7.5.2 IrDA Baud Rate

Table 44-13 gives some examples of CD values, baud rate error and pulse duration. Note that the requirement on the maximum acceptable error of ±1.87% must be met.

**Table 44-13. IrDA Baud Rate Error**

| Peripheral Clock | Baud Rate (Bit/s) | CD | Baud Rate Error | Pulse Time (µs) |
|---|---|---|---|---|
| 3,686,400 | 115,200 | 2 | 0.00% | 1.63 |
| 20,000,000 | 115,200 | 11 | 1.38% | 1.63 |
| 32,768,000 | 115,200 | 18 | 1.25% | 1.63 |
| 40,000,000 | 115,200 | 22 | 1.38% | 1.63 |
| 3,686,400 | 57,600 | 4 | 0.00% | 3.26 |
| 20,000,000 | 57,600 | 22 | 1.38% | 3.26 |
| 32,768,000 | 57,600 | 36 | 1.25% | 3.26 |
| 40,000,000 | 57,600 | 43 | 0.93% | 3.26 |
| 3,686,400 | 38,400 | 6 | 0.00% | 4.88 |
| 20,000,000 | 38,400 | 33 | 1.38% | 4.88 |
| 32,768,000 | 38,400 | 53 | 0.63% | 4.88 |
| 40,000,000 | 38,400 | 65 | 0.16% | 4.88 |
| 3,686,400 | 19,200 | 12 | 0.00% | 9.77 |
| 20,000,000 | 19,200 | 65 | 0.16% | 9.77 |
| 32,768,000 | 19,200 | 107 | 0.31% | 9.77 |
| 40,000,000 | 19,200 | 130 | 0.16% | 9.77 |
| 3,686,400 | 9,600 | 24 | 0.00% | 19.53 |
| 20,000,000 | 9,600 | 130 | 0.16% | 19.53 |
| 32,768,000 | 9,600 | 213 | 0.16% | 19.53 |
| 40,000,000 | 9,600 | 260 | 0.16% | 19.53 |
| 3,686,400 | 2,400 | 96 | 0.00% | 78.13 |
| 20,000,000 | 2,400 | 521 | 0.03% | 78.13 |
| 32,768,000 | 2,400 | 853 | 0.04% | 78.13 |

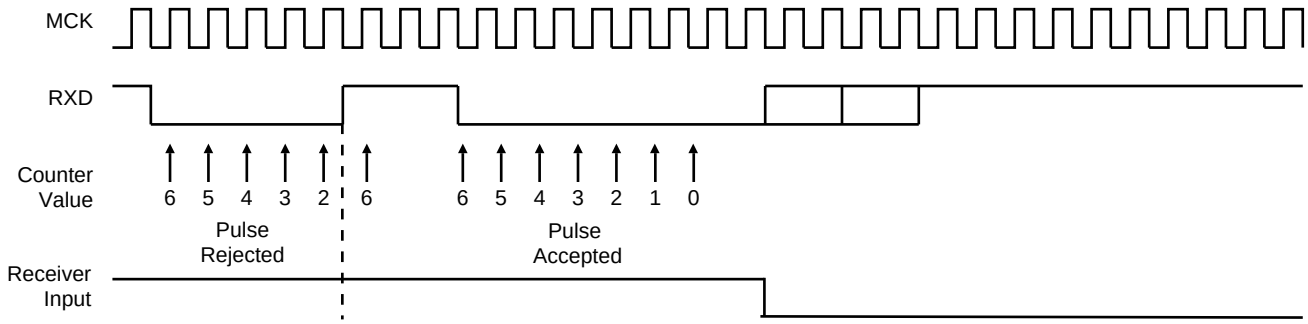
### 44.7.5.3 IrDA Demodulator

The demodulator is based on the IrDA Receive filter comprised of an 8-bit down counter which is loaded with the value programmed in US_IF. When a falling edge is detected on the RXD pin, the Filter Counter starts counting down at the

Master Clock (MCK) speed. If a rising edge is detected on the RXD pin, the counter stops and is reloaded with US_IF. If no rising edge is detected when the counter reaches 0, the input of the receiver is driven low during one bit time.

Figure 44-34 illustrates the operations of the IrDA demodulator.

**Figure 44-34.IrDA Demodulator Operations**




The programmed value in the US_IF register must always meet the following criteria:

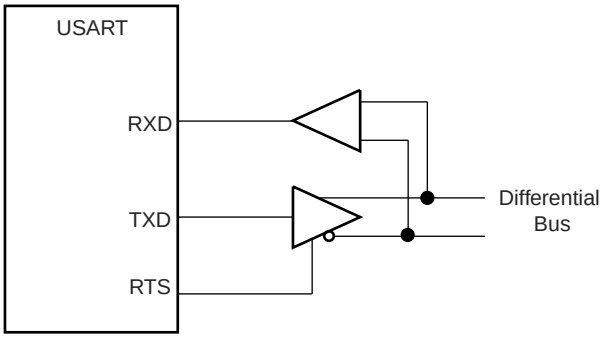
$t_{MCK}$ * (IRDA_FILTER + 3) < 1.41 µs

As the IrDA mode uses the same logic as the ISO7816, note that the FI_DI_RATIO field in US_FIDI must be set to a value higher than 0 in order to assure IrDA communications operate correctly.

### 44.7.6 RS485 Mode

The USART features the RS485 mode to enable line driver control. While operating in RS485 mode, the USART behaves as though in asynchronous or synchronous mode and configuration of all the parameters is possible. The difference is that the RTS pin is driven high when the transmitter is operating. The behavior of the RTS pin is controlled by the TXEMPTY bit. A typical connection of the USART to a RS485 bus is shown in Figure 44-35.

**Figure 44-35.Typical Connection to a RS485 Bus**




The USART is set in RS485 mode by writing the value 0x1 to the USART_MODE field in the Mode Register (US_MR).

The RTS pin is at a level inverse to the TXEMPTY bit. Significantly, the RTS pin remains high when a timeguard is programmed so that the line can remain driven after the last character completion. Figure 44-36 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

#### 44.7.7.6 Character Reception

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 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 a one to the RSTSTA (Reset Status) bit the US_CR.

To ensure correct behavior of the receiver in SPI Slave Mode, the master device sending the frame must ensure a minimum delay of 1 Tbit between each character transmission. The receiver does not require a falling edge of the slave select line (NSS) to initiate a character reception but only a low level. However, this low level must be present on the slave select line (NSS) at least 1 Tbit before the first serial clock cycle corresponding to the MSB bit.

#### 44.7.7.7 Receiver Timeout

Because the receiver baud rate clock is active only during data transfers in SPI Mode, a receiver timeout is impossible in this mode, whatever the Time-out value is (field TO) in the Time-out Register (US_RTOR).

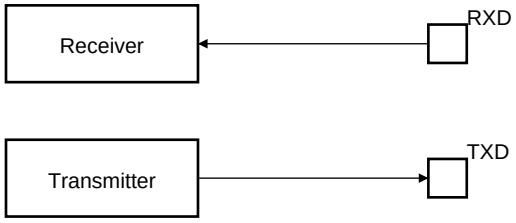
### 44.7.8 Test Modes

The USART can be programmed to operate in three different test modes. The internal loopback capability allows on-board diagnostics. In the loopback mode the USART interface pins are disconnected or not and reconfigured for loopback internally or externally.

#### 44.7.8.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

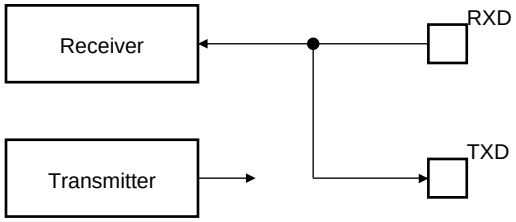
**Figure 44-39.Normal Mode Configuration**




#### 44.7.8.2 Automatic Echo Mode

Automatic echo mode allows bit-by-bit retransmission. When a bit is received on the RXD pin, it is sent to the TXD pin, as shown in Figure 44-40. Programming the transmitter has no effect on the TXD pin. The RXD pin is still connected to the receiver input, thus the receiver remains active.

**Figure 44-40.Automatic Echo Mode Configuration**




#### 44.7.8.3 Local Loopback Mode

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 44-41. The TXD and RXD pins are not used. The RXD pin has no effect on the receiver and the TXD pin is continuously driven high, as in idle state.

### 44.8.15 USART Baud Rate Generator Register

**Name:** US_BRGR

**Address:** 0xF001C020 (0), 0xF0020020 (1), 0xF8020020 (2), 0xF8024020 (3)

**Access:** Read-write

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

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | FP | | |

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

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

This register can only be written if the WPEN bit is cleared in “USART Write Protect Mode Register” on page 1431.

- **CD: Clock Divider**

| | **USART_MODE ≠ ISO7816** | | | |
|---|---|---|---|---|
| | **SYNC = 0** | | **SYNC = 1 or USART_MODE = SPI (Master or Slave)** | **USART_MODE = ISO7816** |
| **CD** | **OVER = 0** | **OVER = 1** | | |
| 0 | Baud Rate Clock Disabled | | | |
| 1 to 65535 | Baud Rate = Selected Clock/(16*CD) | Baud Rate = Selected Clock/(8*CD) | Baud Rate = Selected Clock/CD | Baud Rate = Selected Clock/(FI_DI_RATIO*CD) |

- **FP: Fractional Part**

0: Fractional divider is disabled.

1–7: Baud rate resolution, defined by FP x 1/8.

### 45.9.11 CAN Abort Command Register

**Name:** CAN_ACR

**Address:** 0xF000C028 (0), 0xF8010028 (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 |
| MB7 | MB6 | MB5 | MB4 | MB3 | MB2 | MB1 | MB0 |

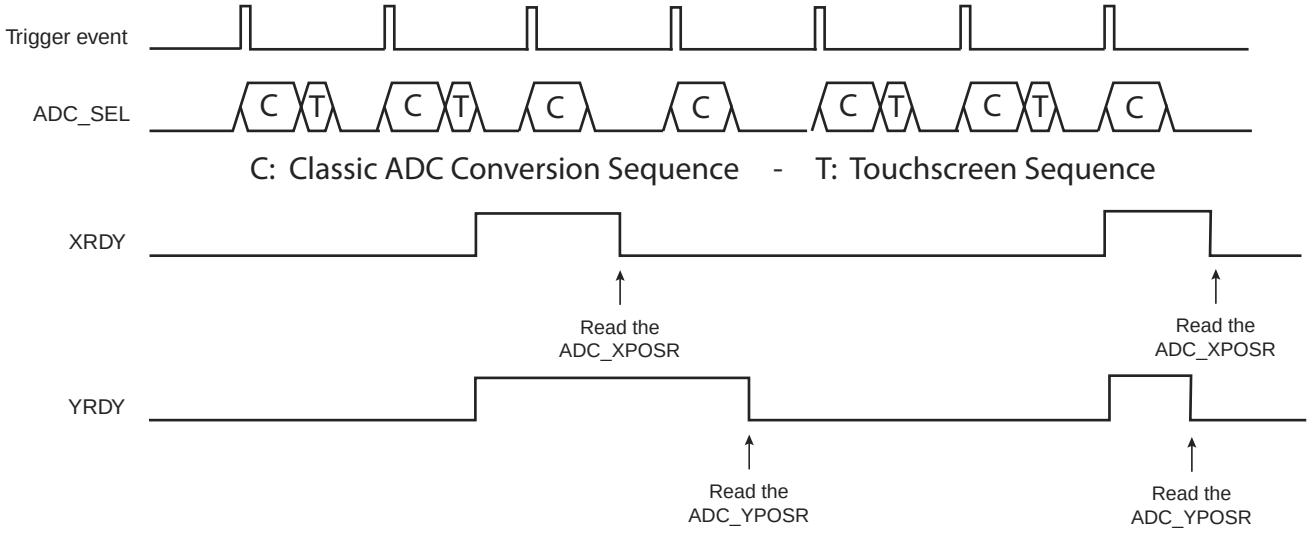
This register initializes several abort requests at the same time.

- **MBx: Abort Request for Mailbox x**

| Mailbox Object Type | Description |
|---|---|
| Receive | No action |
| Receive with overwrite | No action |
| Transmit | Cancels transfer request if the message has not been transmitted to the CAN transceiver. |
| Consumer | Cancels the current transfer before the remote frame has been sent. |
| Producer | Cancels the current transfer. The next remote frame is not serviced. |

It is possible to set the MACR field (in the CAN_MCRx) for each mailbox.

**Figure 49-13.Insertion of Touchscreen sequences (TSFREQ = 2; TSAV = 1)**




### 49.7.8 Measured Values, Registers and Flags

As soon as the controller finishes the Touchscreen sequence, XRDY, YRDY and PRDY are set and can generate an interrupt. These flags can be read in the “ADC Interrupt Status Register”. They are reset independently by reading in ADC_XPOSR, ADC_YPOSR and ADC_PRESSR. for classic ADC conversions.

The “ADC Touchscreen X Position Register” presents XPOS ($V_X$ - $V_{Xmin}$) on its LSB and XSCALE ($V_{XMAX}$ - $V_{Xmin}$) aligned on the 16th bit.

The “ADC Touchscreen Y Position Register” presents YPOS ($V_Y$ - $V_{Ymin}$) on its LSB and YSCALE ($V_{YMAX}$ - $V_{Ymin}$) aligned on the 16th bit.

To improve the quality of the measure, the user must calculate: XPOS/XSCALE and YPOS/YSCALE.

$V_{XMAX}$, $V_{Xmin}$, $V_{YMAX}$, and $V_{Ymin}$ are measured at the first start up of the controller. These values can change during use, so it can be necessary to refresh them. Refresh can be done by writing ‘1’ in the CALIB field of the control register (ADC_CR).

The “ADC Touchscreen Pressure Register” presents Z1 on its LSB and Z2 aligned on the 16th bit. See Section 49.7.4 to know how use them.

### 49.7.9 Pen Detect Method

When there is no contact, it is not necessary to perform a conversion. However, it is important to detect a contact by keeping the power consumption as low as possible.

The implementation polarizes one panel by closing the switch on ($X_P$/$U_L$) and ties the horizontal panel by an embedded resistor connected to $Y_M$ / Sense. This resistor is enabled by a fifth switch. Since there is no contact, no current is flowing and there is no related power consumption. As soon as a contact occurs, a current is flowing in the Touchscreen and a Schmitt trigger detects the voltage in the resistor.

The Touchscreen Interrupt configuration is entered by programming the PENDET bit in the “ADC Touchscreen Mode Register”. If this bit is written at 1, the controller samples the pen contact state when it is not converting and waiting for a trigger.

To complete the circuit, a programmable debouncer is placed at the output of the Schmitt trigger. This debouncer is programmable up to $2^{15}$ ADC clock periods. The debouncer length can be selected by programming the field PENDBC in “ADC Touchscreen Mode Register”.

Due to the analog switch’s structure, the debouncer circuitry is only active when no conversion (Touchscreen or classic ADC channels) is in progress. Thus, if the time between the end of a conversion sequence and the arrival of the next trigger event is lower than the debouncing time configured on PENDBC, the debouncer will not detect any contact.

**Figure 54-11.SPI Slave Mode 0 and 3**

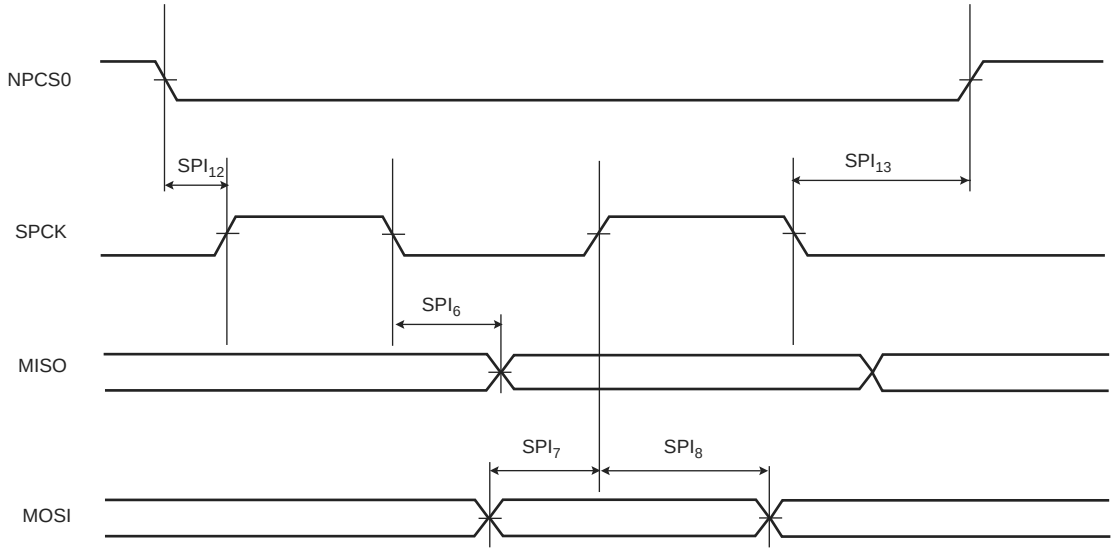



**Figure 54-12.SPI Slave Mode 1 and 2**

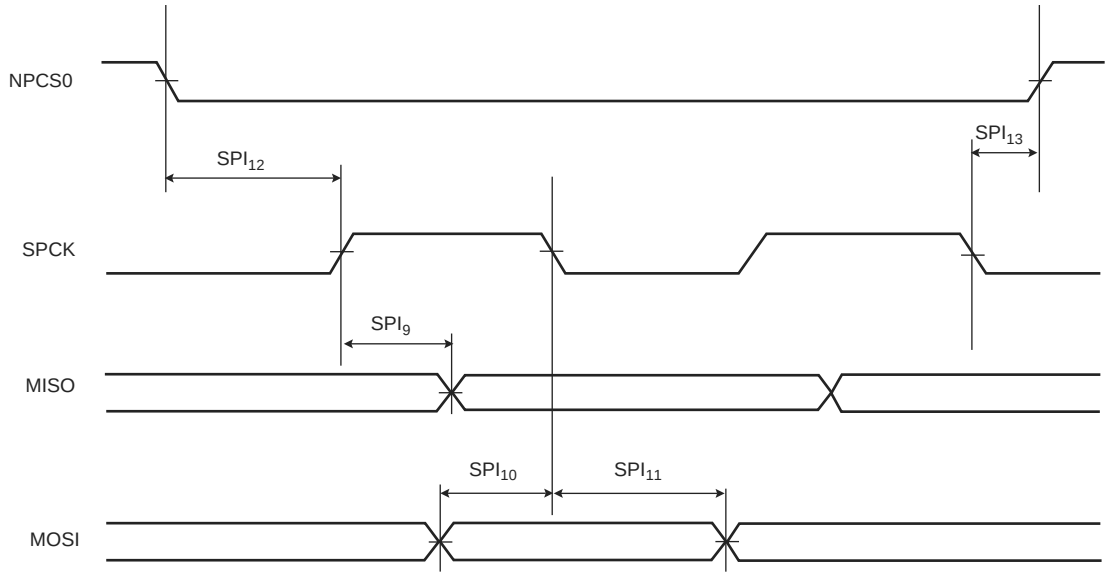



**Figure 54-13.SPI Slave Mode - NPCS Timings**