**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 | Obsolete |
| Core Processor | ST7 |
| Core Size | 8-Bit |
| Speed | 8MHz |
| Connectivity | LINbusSCI, SPI |
| Peripherals | LVD, POR, PWM, WDT |
| Number of I/O | 32 |
| Program Memory Size | 60KB (60K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 2K x 8 |
| Voltage - Supply (Vcc/Vdd) | 3.8V ~ 5.5V |
| Data Converters | A/D 11x10b |
| Oscillator Type | External |
| Operating Temperature | -40°C ~ 125°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 44-LQFP |
| Supplier Device Package | - |
| Purchase URL | https://www.e-xfl.com/product-detail/stmicroelectronics/st72f361j9tce |

**RESET SEQUENCE MANAGER** (Cont'd)

The $\overline{\text{RESET}}$ pin is an asynchronous signal which plays a major role in EMS performance. In a noisy environment, it is recommended to follow the guidelines mentioned in the electrical characteristics section.

### 6.3.3 External Power-On RESET

If the LVD is disabled by option byte, to start up the microcontroller correctly, the user must ensure by means of an external reset circuit that the reset signal is held low until $V_{DD}$ is over the minimum level specified for the selected $f_{OSC}$ frequency.

A proper reset signal for a slow rising $V_{DD}$ supply can generally be provided by an external RC network connected to the $\overline{\text{RESET}}$ pin.

### 6.3.4 Internal Low Voltage Detector (LVD) RESET

Two different RESET sequences caused by the internal LVD circuitry can be distinguished:

- Power-On RESET
- Voltage Drop RESET

The device $\overline{\text{RESET}}$ pin acts as an output that is pulled low when $V_{DD} < V_{IT+}$ (rising edge) or $V_{DD} < V_{IT-}$ (falling edge) as shown in Figure 3.

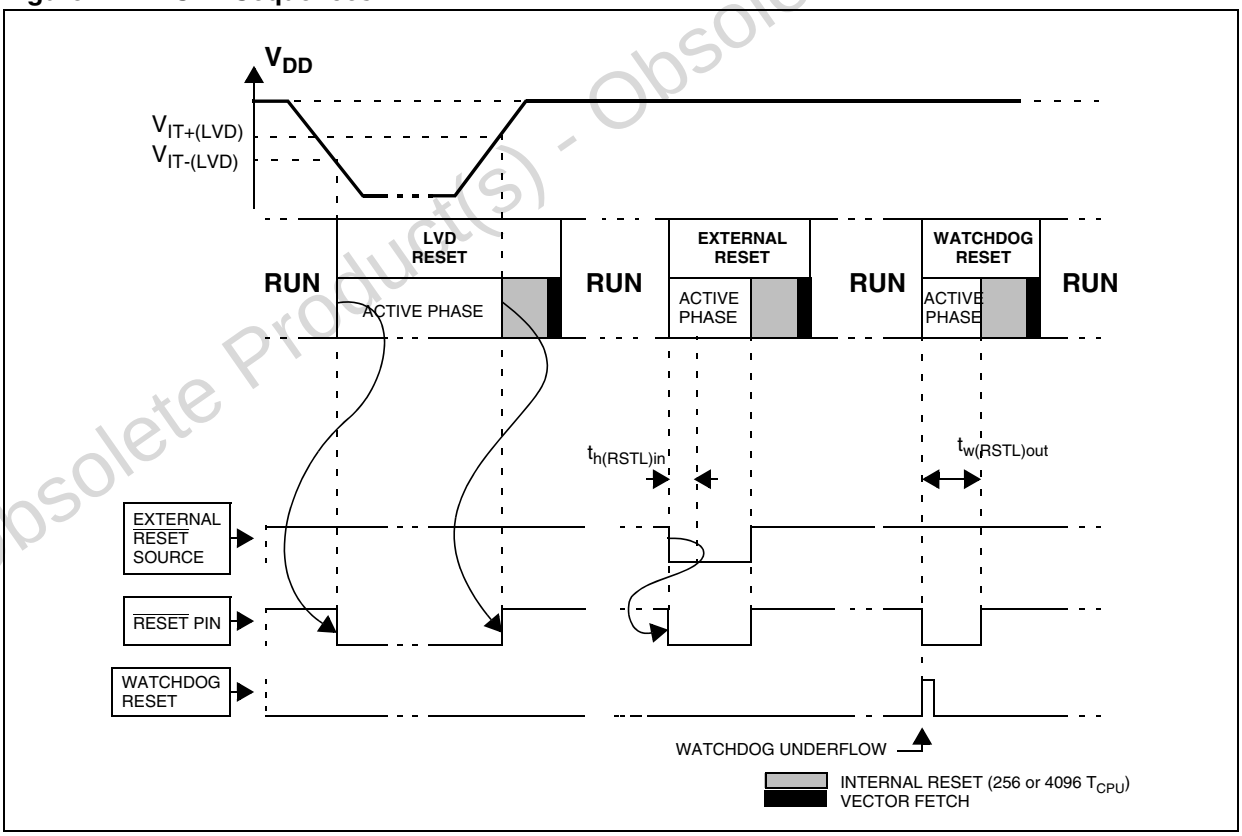
The LVD filters spikes on $V_{DD}$ larger than $t_{g(VDD)}$ to avoid parasitic resets.

### 6.3.5 Internal Watchdog RESET

The RESET sequence generated by a internal Watchdog counter overflow is shown in Figure 3.

Starting from the Watchdog counter underflow, the device $\overline{\text{RESET}}$ pin acts as an output that is pulled low during at least $t_{w(RSTL)out}$.

**Figure 14. RESET Sequences**



### 6.4 SYSTEM INTEGRITY MANAGEMENT (SI)

The System Integrity Management block contains the Low Voltage Detector (LVD) and Auxiliary Voltage Detector (AVD) functions. It is managed by the SICSR register.

#### 6.4.1 Low Voltage Detector (LVD)

The Low Voltage Detector function (LVD) generates a static reset when the $V_{DD}$ supply voltage is below a $V_{IT-(LVD)}$ reference value. This means that it secures the power-up as well as the power-down keeping the ST7 in reset.

The $V_{IT-(LVD)}$ reference value for a voltage drop is lower than the $V_{IT+(LVD)}$ reference value for power-on in order to avoid a parasitic reset when the MCU starts running and sinks current on the supply (hysteresis).

The LVD Reset circuitry generates a reset when $V_{DD}$ is below:

– $V_{IT+(LVD)}$ when $V_{DD}$ is rising
– $V_{IT-(LVD)}$ when $V_{DD}$ is falling

The LVD function is illustrated in Figure 15.

Provided the minimum $V_{DD}$ value (guaranteed for the oscillator frequency) is above $V_{IT-(LVD)}$, the MCU can only be in two modes:

– under full software control
– in static safe reset

In these conditions, secure operation is always ensured for the application without the need for external reset hardware.

During a Low Voltage Detector Reset, the $\overline{\text{RESET}}$ pin is held low, thus permitting the MCU to reset other devices.

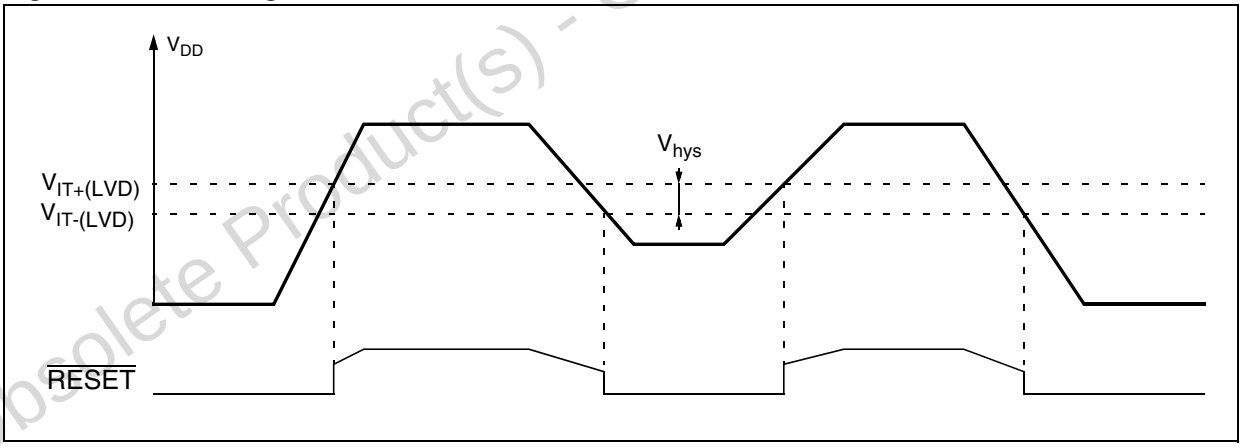
**Notes**:

The LVD allows the device to be used without any external RESET circuitry.

The LVD is an optional function which can be selected by option byte.

It is recommended to make sure that the $V_{DD}$ supply voltage rises monotonously when the device is exiting from Reset, to ensure the application functions properly.

**Figure 15. Low Voltage Detector vs Reset**



**INTERRUPTS** (Cont'd)

**Table 8. Dedicated Interrupt Instruction Set**

| Instruction | New Description | Function/Example | I1 | H | I0 | N | Z | C |
|---|---|---|---|---|---|---|---|---|
| HALT | Entering Halt mode | | 1 | | 0 | | | |
| IRET | Interrupt routine return | Pop CC, A, X, PC | I1 | H | I0 | N | Z | C |
| JRM | Jump if I1:0 = 11 (level 3) | I1:0 = 11 ? | | | | | | |
| JRNM | Jump if I1:0 <> 11 | I1:0 <> 11 ? | | | | | | |
| POP CC | Pop CC from the Stack | Mem => CC | I1 | H | I0 | N | Z | C |
| RIM | Enable interrupt (level 0 set) | Load 10 in I1:0 of CC | 1 | | 0 | | | |
| SIM | Disable interrupt (level 3 set) | Load 11 in I1:0 of CC | 1 | | 1 | | | |
| TRAP | Software trap | Software NMI | 1 | | 1 | | | |
| WFI | Wait for interrupt | | 1 | | 0 | | | |

**Note**: During the execution of an interrupt routine, the HALT, POPCC, RIM, SIM and WFI instructions change the current software priority up to the next IRET instruction or one of the previously mentioned instructions.

**INTERRUPTS** (Cont'd)

**Table 10. Nested Interrupts Register Map and Reset Values**

| Address (Hex.) | Register Label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0025h | | ei1 | | ei0 | | CLKM | | TLI | |
| | **ISPR0** | I1_3 | I0_3 | I1_2 | I0_2 | I1_1 | I0_1 | | |
| | Reset Value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0026h | | | | | | ei3 | | ei2 | |
| | **ISPR1** | I1_7 | I0_7 | I1_6 | I0_6 | I1_5 | I0_5 | I1_4 | I0_4 |
| | Reset Value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0027h | | LINSCI 2 | | TIMER 16 | | TIMER 8 | | SPI | |
| | **ISPR2** | I1_11 | I0_11 | I1_10 | I0_10 | I1_9 | I0_9 | I1_8 | I0_8 |
| | Reset Value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0028h | | | | | | ART | | LINSCI 1 | |
| | **ISPR3** | | | | | I1_13 | I0_13 | I1_12 | I0_12 |
| | Reset Value | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0029h | **EICR0** | IS31 | IS30 | IS21 | IS20 | IS11 | IS10 | IS01 | IS00 |
| | Reset Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 002Ah | **EICR1** | | | | | | | TLIS | TLIE |
| | Reset Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**I/O PORTS** (Cont'd)

**Figure 32. I/O Port General Block Diagram**

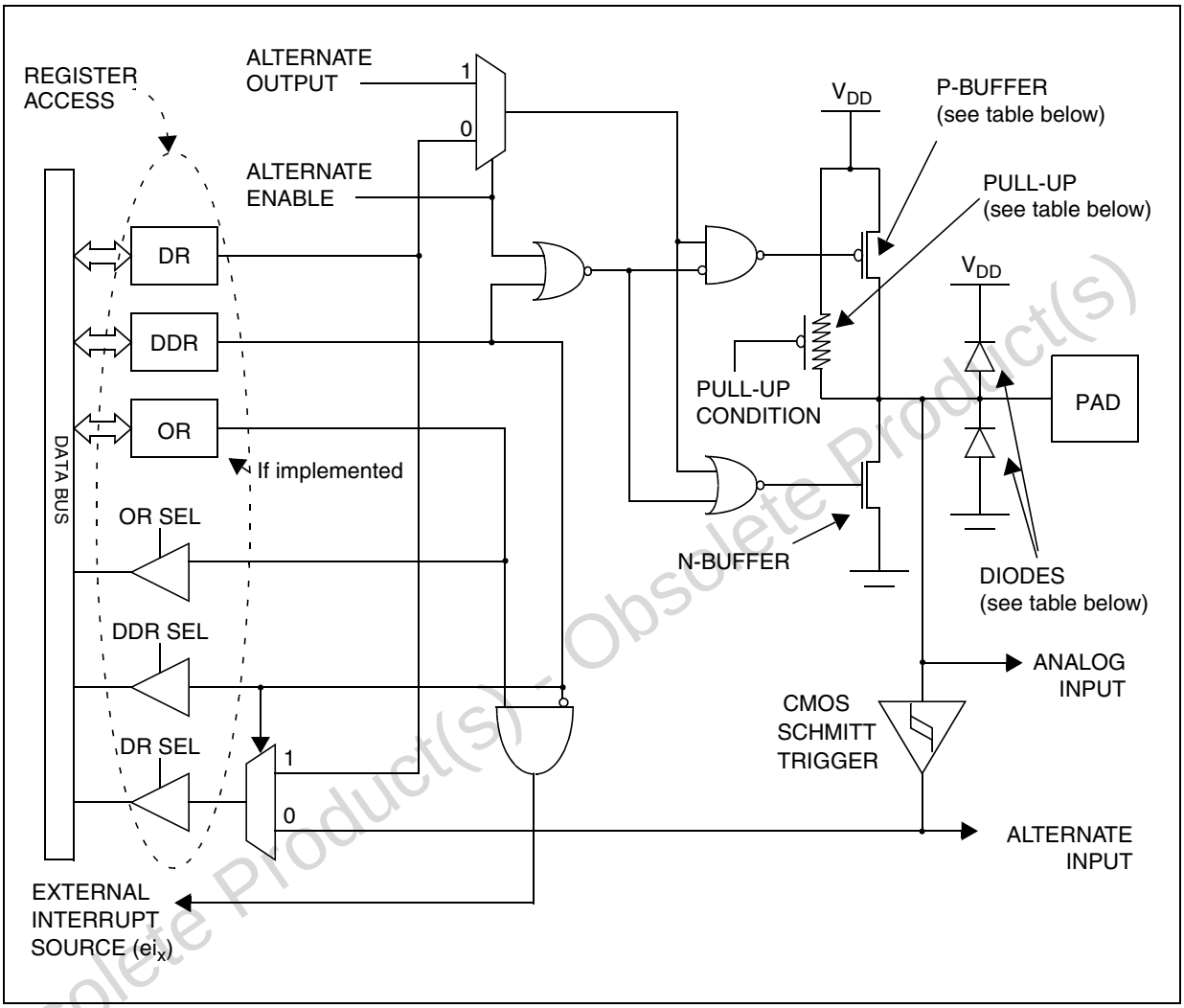





**Table 12. I/O Port Mode Options**

| Configuration Mode | | Pull-Up | P-Buffer | Diodes | |
|---|---|---|---|---|---|
| | | | | to $V_{DD}$ | to $V_{SS}$ |
| Input | Floating with/without Interrupt | Off | Off | On | On |
| | Pull-up with/without Interrupt | On | | | |
| Output | Push-pull | Off | On | | |
| | Open Drain (logic level) | | Off | | |
| | True Open Drain | NI | NI | NI (see note) | |

**Legend**: NI - not implemented
Off - implemented not activated
On - implemented and activated

**Note**: The diode to $V_{DD}$ is not implemented in the true open drain pads. A local protection between the pad and $V_{SS}$ is implemented to protect the device against positive stress.

**ON-CHIP PERIPHERALS** (Cont'd)

## 10.2 MAIN CLOCK CONTROLLER WITH REAL TIME CLOCK MCC/RTC

The Main Clock Controller consists of three different functions:

- a programmable CPU clock prescaler
- a clock-out signal to supply external devices
- a real time clock timer with interrupt capability

Each function can be used independently and simultaneously.

### 10.2.1 Programmable CPU Clock Prescaler

The programmable CPU clock prescaler supplies the clock for the ST7 CPU and its internal peripherals. It manages SLOW power saving mode (See Section 8.2 "SLOW MODE" for more details).

The prescaler selects the $f_{CPU}$ main clock frequency and is controlled by three bits in the MCCSR register: CP[1:0] and SMS.

### 10.2.2 Clock-out Capability

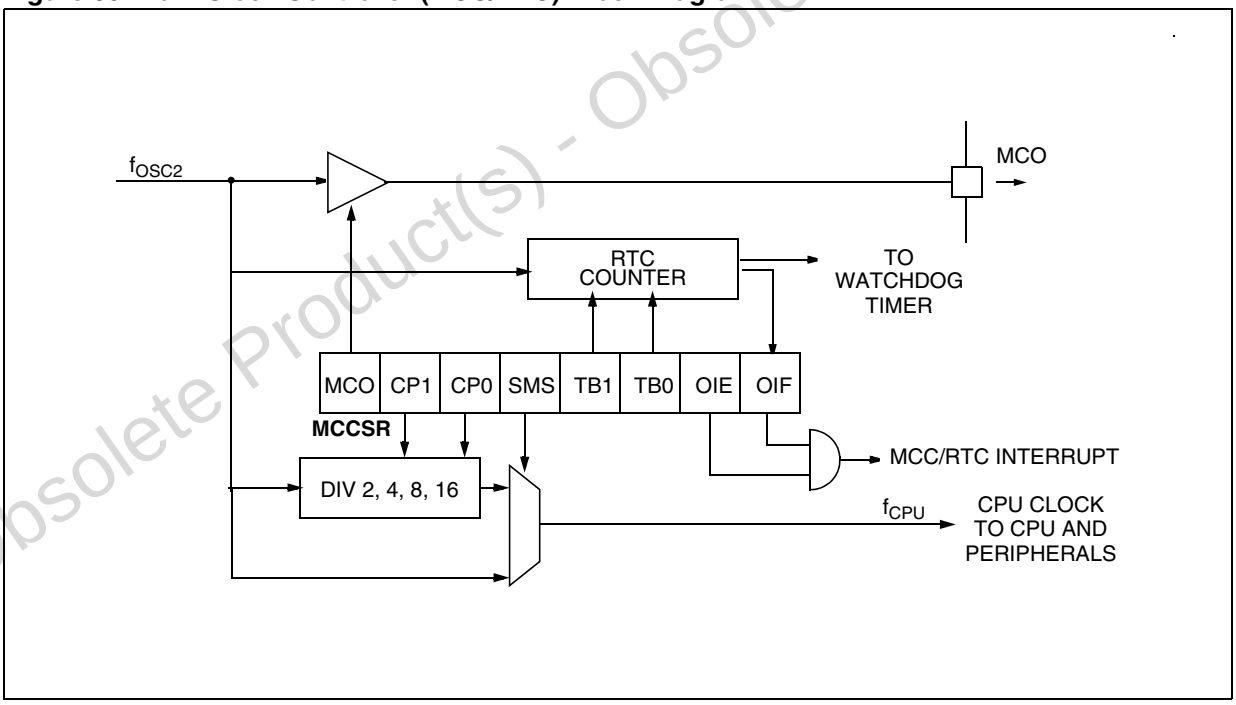
The clock-out capability is an alternate function of an I/O port pin that outputs a $f_{OSC2}$ clock to drive external devices. It is controlled by the MCO bit in the MCCSR register.

### 10.2.3 Real Time Clock Timer (RTC)

The counter of the real time clock timer allows an interrupt to be generated based on an accurate real time clock. Four different time bases depending directly on $f_{OSC2}$ are available. The whole functionality is controlled by 4 bits of the MCCSR register: TB[1:0], OIE and OIF.

When the RTC interrupt is enabled (OIE bit set), the ST7 enters ACTIVE HALT mode when the HALT instruction is executed. See Section 8.5 "ACTIVE HALT MODE" for more details.

**Figure 39. Main Clock Controller (MCC/RTC) Block Diagram**



**16-BIT TIMER** (Cont'd)

**Figure 49. Counter Timing Diagram, Internal Clock Divided by 2**

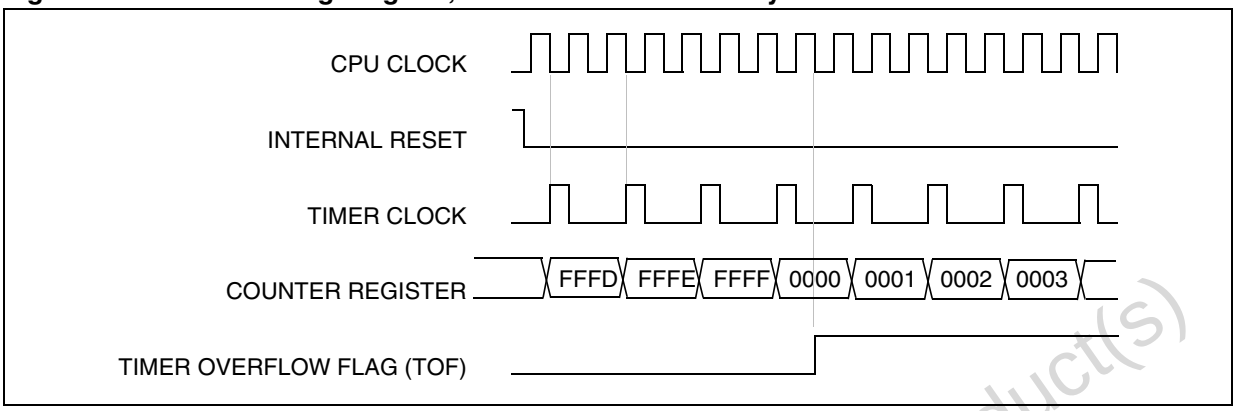



**Figure 50. Counter Timing Diagram, Internal Clock Divided by 4**

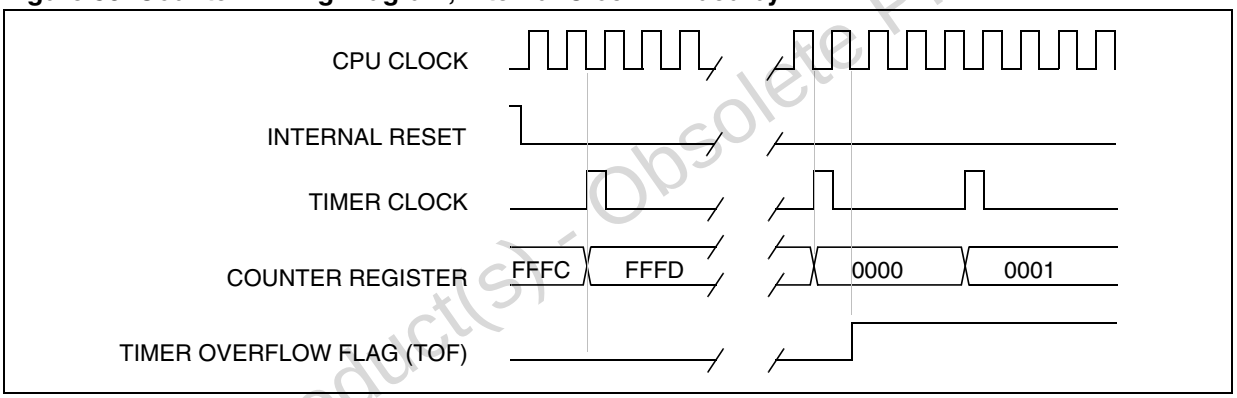
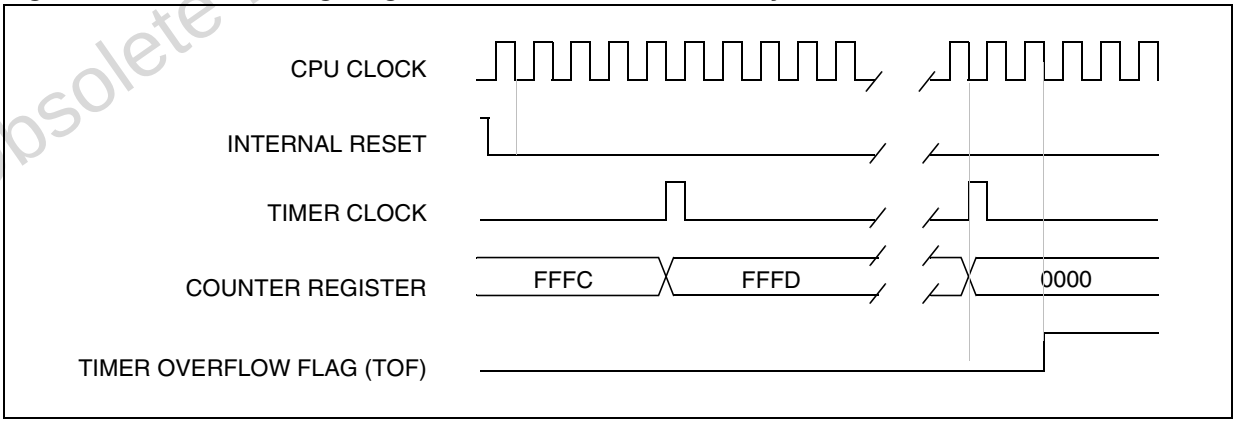



**Figure 51. Counter Timing Diagram, Internal Clock Divided By 8**




**Note:** The MCU is in reset state when the internal reset signal is high, when it is low the MCU is running.

**16-BIT TIMER** (Cont'd)

### 10.4.3.6 Pulse Width Modulation Mode

Pulse Width Modulation (PWM) mode enables the generation of a signal with a frequency and pulse length determined by the value of the OC1R and OC2R registers.

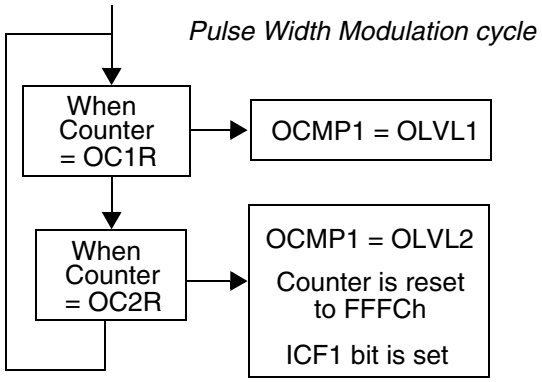
Pulse Width Modulation mode uses the complete Output Compare 1 function plus the OC2R register, and so this functionality can not be used when PWM mode is activated.

In PWM mode, double buffering is implemented on the output compare registers. Any new values written in the OC1R and OC2R registers are taken into account only at the end of the PWM period (OC2) to avoid spikes on the PWM output pin (OCMP1).

**Procedure**

To use Pulse Width Modulation mode:

1. Load the OC2R register with the value corresponding to the period of the signal using the formula in the opposite column.
2. Load the OC1R register with the value corresponding to the period of the pulse if (OLVL1 = 0 and OLVL2 = 1) using the formula in the opposite column.
3. Select the following in the CR1 register:
   - Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after a successful comparison with the OC1R register.
   - Using the OLVL2 bit, select the level to be applied to the OCMP1 pin after a successful comparison with the OC2R register.
4. Select the following in the CR2 register:
   - Set OC1E bit: the OCMP1 pin is then dedicated to the output compare 1 function.
   - Set the PWM bit.
   - Select the timer clock (CC[1:0]) (see Table 17 Clock Control Bits).




If OLVL1 = 1 and OLVL2 = 0 the length of the positive pulse is the difference between the OC2R and OC1R registers.

If OLVL1 = OLVL2 a continuous signal will be seen on the OCMP1 pin.

The OC*i*R register value required for a specific timing application can be calculated using the following formula:

$$OCiR\ Value = \frac{t * f_{CPU}}{PRESC} - 5$$

Where:

t = Signal or pulse period (in seconds)

$f_{CPU}$ = CPU clock frequency (in hertz)

PRESC = Timer prescaler factor (2, 4 or 8 depending on CC[1:0] bits, see Table 17 Clock Control Bits)

If the timer clock is an external clock the formula is:

$$OCiR = t * f_{EXT} - 5$$

Where:

t = Signal or pulse period (in seconds)

$f_{EXT}$ = External timer clock frequency (in hertz)

The Output Compare 2 event causes the counter to be initialized to FFFCh (See Figure 58)

**Notes:**

1. After a write instruction to the OC*i*HR register, the output compare function is inhibited until the OC*i*LR register is also written.
2. The OCF1 and OCF2 bits cannot be set by hardware in PWM mode therefore the Output Compare interrupt is inhibited.
3. The ICF1 bit is set by hardware when the counter reaches the OC2R value and can produce a timer interrupt if the ICIE bit is set and the I bit is cleared.
4. In PWM mode the ICAP1 pin can not be used to perform input capture because it is disconnected to the timer. The ICAP2 pin can be used to perform input capture (ICF2 can be set and IC2R can be loaded) but the user must take care that the counter is reset each period and ICF1 can also generates interrupt if ICIE is set.
5. When the Pulse Width Modulation (PWM) and One Pulse mode (OPM) bits are both set, the PWM mode is the only active one.

**SERIAL PERIPHERAL INTERFACE** (cont'd)

#### 10.6.3.3 Master Mode Operation

In master mode, the serial clock is output on the SCK pin. The clock frequency, polarity and phase are configured by software (refer to the description of the SPICSR register).

**Note:** The idle state of SCK must correspond to the polarity selected in the SPICSR register (by pulling up SCK if CPOL = 1 or pulling down SCK if CPOL = 0).

**How to operate the SPI in master mode**

To operate the SPI in master mode, perform the following steps in order:

1. Write to the SPICR register:
   - Select the clock frequency by configuring the SPR[2:0] bits.
   - Select the clock polarity and clock phase by configuring the CPOL and CPHA bits. Figure 74 shows the four possible configurations.
     **Note:** The slave must have the same CPOL and CPHA settings as the master.
2. Write to the SPICSR register:
   - Either set the SSM bit and set the SSI bit or clear the SSM bit and tie the SS pin high for the complete byte transmit sequence.
3. Write to the SPICR register:
   - Set the MSTR and SPE bits
     **Note:** MSTR and SPE bits remain set only if SS is high).

**Important note:** if the SPICSR register is not written first, the SPICR register setting (MSTR bit) may be not taken into account.

The transmit sequence begins when software writes a byte in the SPIDR register.

#### 10.6.3.4 Master Mode Transmit Sequence

When software writes to the SPIDR register, the data byte is loaded into the 8-bit shift register and then shifted out serially to the MOSI pin most significant bit first.

When data transfer is complete:

- The SPIF bit is set by hardware.
- An interrupt request is generated if the SPIE bit is set and the interrupt mask in the CCR register is cleared.

Clearing the SPIF bit is performed by the following software sequence:

1. An access to the SPICSR register while the SPIF bit is set
2. A read to the SPIDR register

**Note:** While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.

#### 10.6.3.5 Slave Mode Operation

In slave mode, the serial clock is received on the SCK pin from the master device.

To operate the SPI in slave mode:

1. Write to the SPICSR register to perform the following actions:
   - Select the clock polarity and clock phase by configuring the CPOL and CPHA bits (see Figure 74).
     **Note:** The slave must have the same CPOL and CPHA settings as the master.
   - Manage the $\overline{SS}$ pin as described in Section 10.6.3.2 and Figure 72. If CPHA = 1 SS must be held low continuously. If CPHA = 0 SS must be held low during byte transmission and pulled up between each byte to let the slave write in the shift register.
2. Write to the SPICR register to clear the MSTR bit and set the SPE bit to enable the SPI I/O functions.

#### 10.6.3.6 Slave Mode Transmit Sequence

When software writes to the SPIDR register, the data byte is loaded into the 8-bit shift register and then shifted out serially to the MISO pin most significant bit first.

The transmit sequence begins when the slave device receives the clock signal and the most significant bit of the data on its MOSI pin.

When data transfer is complete:

- The SPIF bit is set by hardware.
- An interrupt request is generated if SPIE bit is set and interrupt mask in the CCR register is cleared.

Clearing the SPIF bit is performed by the following software sequence:

1. An access to the SPICSR register while the SPIF bit is set
2. A write or a read to the SPIDR register

**Notes:** While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.

The SPIF bit can be cleared during a second transmission; however, it must be cleared before the second SPIF bit in order to prevent an Overrun condition (see Section 10.6.5.2).

**SERIAL PERIPHERAL INTERFACE** (cont'd)

### 10.6.4 Clock Phase and Clock Polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits (See Figure 74).

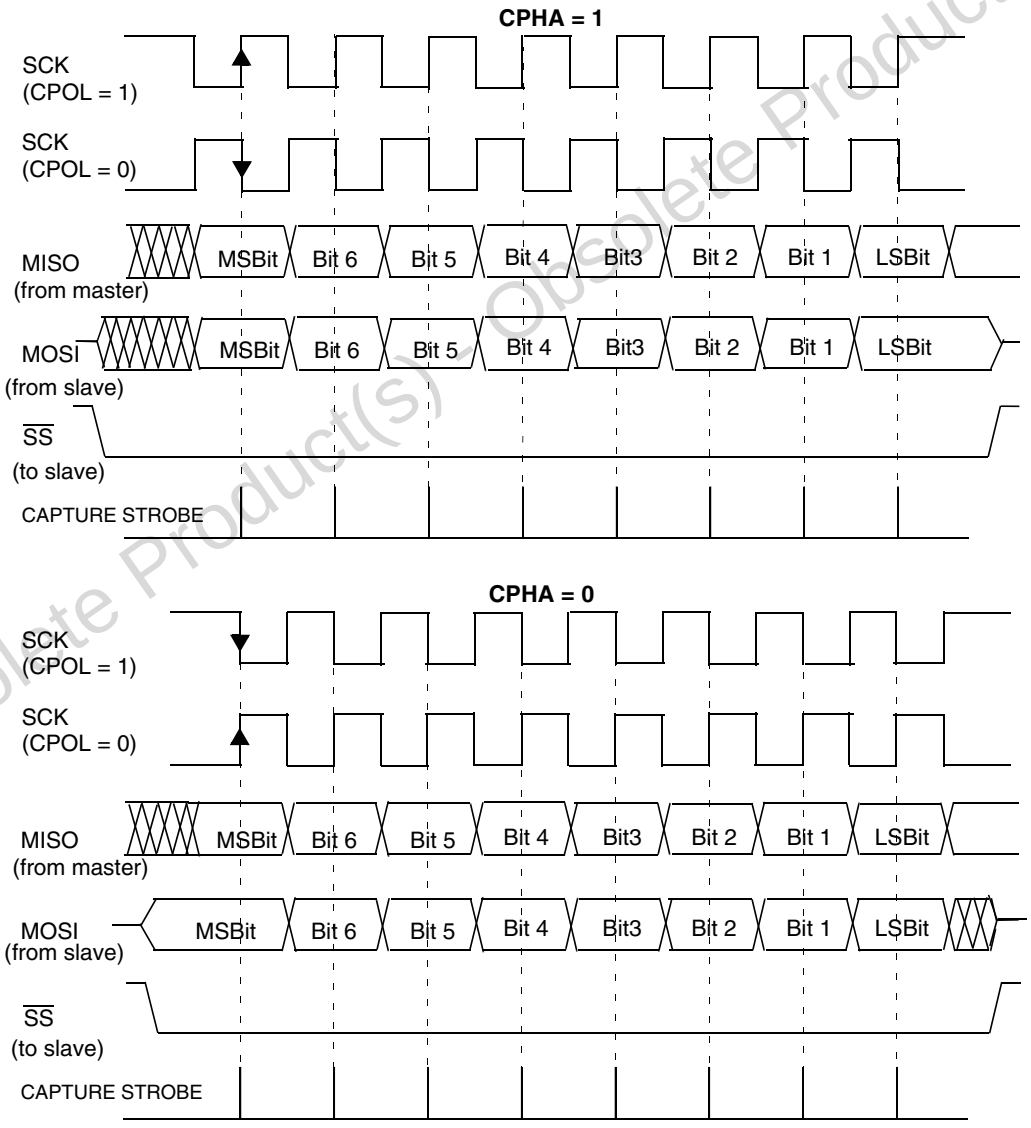
**Note:** The idle state of SCK must correspond to the polarity selected in the SPICSR register (by pulling up SCK if CPOL = 1 or pulling down SCK if CPOL = 0).

The combination of the CPOL clock polarity and CPHA (clock phase) bits selects the data capture clock edge.

Figure 74 shows an SPI transfer with the four combinations of the CPHA and CPOL bits. The diagram may be interpreted as a master or slave timing diagram where the SCK pin, the MISO pin and the MOSI pin are directly connected between the master and the slave device.

**Note**: If CPOL is changed at the communication byte boundaries, the SPI must be disabled by resetting the SPE bit.

**Figure 74. Data Clock Timing Diagram**




**Note:** This figure should not be used as a replacement for parametric information. Refer to the Electrical Characteristics chapter.

**SERIAL PERIPHERAL INTERFACE** (cont'd)

### 10.6.5 Error Flags

#### 10.6.5.1 Master Mode Fault (MODF)

Master mode fault occurs when the master device's $\overline{SS}$ pin is pulled low.

When a Master mode fault occurs:

– The MODF bit is set and an SPI interrupt request is generated if the SPIE bit is set.

– The SPE bit is reset. This blocks all output from the device and disables the SPI peripheral.

– The MSTR bit is reset, thus forcing the device into slave mode.

Clearing the MODF bit is done through a software sequence:

1. A read access to the SPICSR register while the MODF bit is set.

2. A write to the SPICR register.

**Notes:** To avoid any conflicts in an application with multiple slaves, the $\overline{SS}$ pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits may be restored to their original state during or after this clearing sequence.

Hardware does not allow the user to set the SPE and MSTR bits while the MODF bit is set except in the MODF bit clearing sequence.

In a slave device, the MODF bit can not be set, but in a multimaster configuration the device can be in slave mode with the MODF bit set.

The MODF bit indicates that there might have been a multimaster conflict and allows software to handle this using an interrupt routine and either perform a reset or return to an application default state.

#### 10.6.5.2 Overrun Condition (OVR)

An overrun condition occurs when the master device has sent a data byte and the slave device has not cleared the SPIF bit issued from the previously transmitted byte.

When an Overrun occurs:

– The OVR bit is set and an interrupt request is generated if the SPIE bit is set.

In this case, the receiver buffer contains the byte sent after the SPIF bit was last cleared. A read to the SPIDR register returns this byte. All other bytes are lost.

The OVR bit is cleared by reading the SPICSR register.

#### 10.6.5.3 Write Collision Error (WCOL)

A write collision occurs when the software tries to write to the SPIDR register while a data transfer is taking place with an external device. When this happens, the transfer continues uninterrupted and the software write will be unsuccessful.

Write collisions can occur both in master and slave mode. See also Section 10.6.3.2 "Slave Select Management".

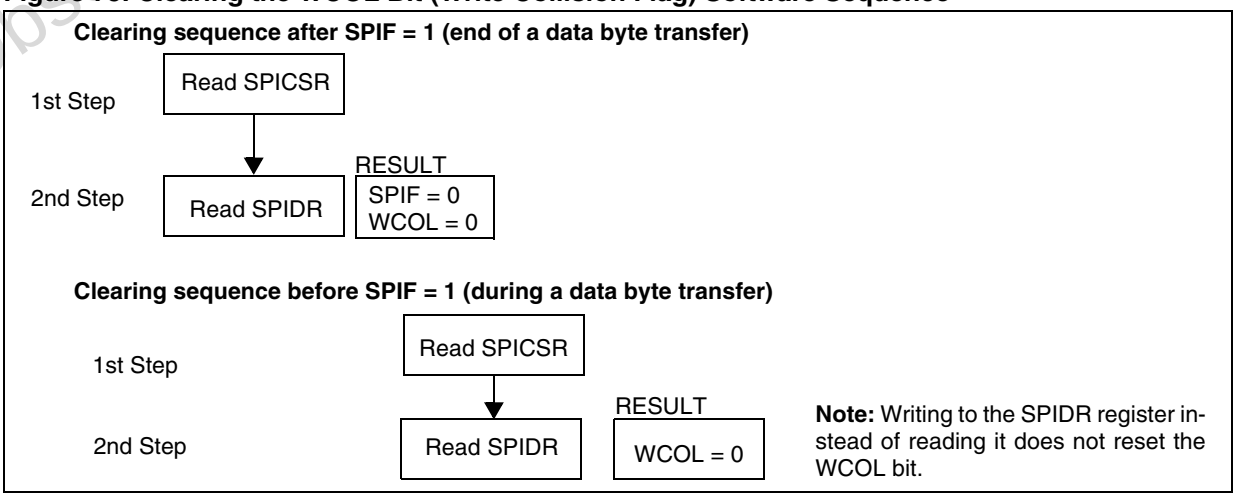
**Note:** A "read collision" will never occur since the received data byte is placed in a buffer in which access is always synchronous with the CPU operation.

The WCOL bit in the SPICSR register is set if a write collision occurs.

No SPI interrupt is generated when the WCOL bit is set (the WCOL bit is a status flag only).

Clearing the WCOL bit is done through a software sequence (see Figure 75).

**Figure 75. Clearing the WCOL Bit (Write Collision Flag) Software Sequence**



**LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode)** (cont'd)

#### 10.7.5.6 Receiver Muting and Wake-up Feature

In multiprocessor configurations it is often desirable that only the intended message recipient should actively receive the full message contents, thus reducing redundant SCI service overhead for all non-addressed receivers.

The non-addressed devices may be placed in sleep mode by means of the muting function.

Setting the RWU bit by software puts the SCI in sleep mode:

All the reception status bits can not be set.

All the receive interrupts are inhibited.

A muted receiver may be woken up in one of the following ways:

– by Idle Line detection if the WAKE bit is reset,

– by Address Mark detection if the WAKE bit is set.

**Idle Line Detection**

Receiver wakes up by Idle Line detection when the Receive line has recognized an Idle Line. Then the RWU bit is reset by hardware but the IDLE bit is not set.

This feature is useful in a multiprocessor system when the first characters of the message determine the address and when each message ends by an idle line: As soon as the line becomes idle, every receivers is waken up and analyse the first characters of the message which indicates the addressed receiver. The receivers which are not addressed set RWU bit to enter in mute mode. Consequently, they will not treat the next characters constituting the next part of the message. At the end of the message, an idle line is sent by the transmitter: this wakes up every receivers which are ready to analyse the addressing characters of the new message.

In such a system, the inter-characters space must be smaller than the idle time.

**Address Mark Detection**

Receiver wakes up by Address Mark detection when it received a "1" as the most significant bit of a word, thus indicating that the message is an address. The reception of this particular word wakes up the receiver, resets the RWU bit and sets the RDRF bit, which allows the receiver to receive this word normally and to use it as an address word.

This feature is useful in a multiprocessor system when the most significant bit of each character (except for the break character) is reserved for Address Detection. As soon as the receivers received an address character (most significant bit = '1'), the receivers are waken up. The receivers which are not addressed set RWU bit to enter in mute mode. Consequently, they will not treat the next characters constituting the next part of the message.

#### 10.7.5.7 Parity Control

Hardware byte Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the SCICR1 register. Depending on the character format defined by the M bit, the possible SCI character formats are as listed in Table 1.

**Note**: In case of wake-up by an address mark, the MSB bit of the data is taken into account and not the parity bit

**Table 23. Character Formats**

| M bit | PCE bit | Character format |
|---|---|---|
| 0 | 0 | ǀ SB ǀ 8 bit data ǀ STB ǀ |
| | 1 | ǀ SB ǀ 7-bit data ǀ PB ǀ STB ǀ |
| 1 | 0 | ǀ SB ǀ 9-bit data ǀ STB ǀ |
| | 1 | ǀ SB ǀ 8-bit data ǀ PB ǀ STB ǀ |

**Legend:** SB = Start Bit, STB = Stop Bit,
PB = Parity Bit

**Even parity:** The parity bit is calculated to obtain an even number of "1s" inside the character made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

Example: data = 00110101; 4 bits set => parity bit will be 0 if even parity is selected (PS bit = 0).

**Odd parity:** The parity bit is calculated to obtain an odd number of "1s" inside the character made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

Example: data = 00110101; 4 bits set => parity bit will be 1 if odd parity is selected (PS bit = 1).

**Transmission mode:** If the PCE bit is set then the MSB bit of the data written in the data register is not transmitted but is changed by the parity bit.

**Reception mode:** If the PCE bit is set then the interface checks if the received data byte has an even number of "1s" if even parity is selected (PS = 0) or an odd number of "1s" if odd parity is selected (PS = 1). If the parity check fails, the PE flag is set in the SCISR register and an interrupt is generated if PCIE is set in the SCICR1 register.

## LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

### 10.7.6 Low Power Modes

| Mode | Description |
|---|---|
| WAIT | No effect on SCI.<br>SCI interrupts cause the device to exit from Wait mode. |
| HALT | SCI registers are frozen.<br>In Halt mode, the SCI stops transmitting/receiving until Halt mode is exited. |

### 10.7.7 Interrupts

| Interrupt Event | Event Flag | Enable Control Bit | Exit from Wait | Exit from Halt |
|---|---|---|---|---|
| Transmit Data Register Empty | TDRE | TIE | Yes | No |
| Transmission Complete | TC | TCIE | | |
| Received Data Ready to be Read | RDRF | RIE | | |
| Overrun Error or LIN Synch Error Detected | OR/ LHE | | | |
| Idle Line Detected | IDLE | ILIE | | |
| Parity Error | PE | PIE | | |
| LIN Header Detection | LHDF | LHIE | | |

The SCI interrupt events are connected to the same interrupt vector (see Interrupts chapter).

These events generate an interrupt if the corresponding Enable Control Bit is set and the interrupt mask in the CC register is reset (RIM instruction).

**LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode)** (cont'd)

**10.7.8 SCI Mode Register Description**

**STATUS REGISTER (SCISR)**

Read Only

Reset Value: 1100 0000 (C0h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| TDRE | TC | RDRF | IDLE | OR[1)] | NF[1)] | FE[1)] | PE[1)] |

Bit 7 = **TDRE** *Transmit data register empty*

This bit is set by hardware when the content of the TDR register has been transferred into the shift register. An interrupt is generated if the TIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a write to the SCIDR register).

0: Data is not transferred to the shift register

1: Data is transferred to the shift register

Bit 6 = **TC** *Transmission complete*

This bit is set by hardware when transmission of a character containing Data is complete. An interrupt is generated if TCIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a write to the SCIDR register).

0: Transmission is not complete

1: Transmission is complete

**Note:** TC is not set after the transmission of a Preamble or a Break.

Bit 5 = **RDRF** *Received data ready flag*

This bit is set by hardware when the content of the RDR register has been transferred to the SCIDR register. An interrupt is generated if RIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).

0: Data is not received

1: Received data is ready to be read

Bit 4 = **IDLE** *Idle line detected*

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if the ILIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).

0: No Idle Line is detected

1: Idle Line is detected

**Note:** The IDLE bit will not be set again until the RDRF bit has been set itself (that is, a new idle line occurs).

Bit 3 = **OR** *Overrun error*

The OR bit is set by hardware when the word currently being received in the shift register is ready to be transferred into the RDR register whereas RDRF is still set. An interrupt is generated if RIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).

0: No Overrun error

1: Overrun error detected

**Note:** When this bit is set, RDR register contents will not be lost but the shift register will be overwritten.

Bit 2 = **NF** *Character Noise flag*

This bit is set by hardware when noise is detected on a received character. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).

0: No noise

1: Noise is detected

**Note:** This bit does not generate interrupt as it appears at the same time as the RDRF bit which itself generates an interrupt.

Bit 1 = **FE** *Framing error*

This bit is set by hardware when a desynchronization, excessive noise or a break character is detected. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).

0: No Framing error

1: Framing error or break character detected

**Note:** This bit does not generate an interrupt as it appears at the same time as the RDRF bit which itself generates an interrupt. If the word currently being transferred causes both a frame error and an overrun error, it will be transferred and only the OR bit will be set.

Bit 0 = **PE** *Parity error*

This bit is set by hardware when a byte parity error occurs (if the PCE bit is set) in receiver mode. It is cleared by a software sequence (a read to the status register followed by an access to the SCIDR data register). An interrupt is generated if PIE = 1 in the SCICR1 register.

0: No parity error

1: Parity error detected

**LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode)** (cont'd)

### 10.7.9.3 LIN Reception

In LIN mode the reception of a byte is the same as in SCI mode but the LINSCI has features for handling the LIN Header automatically (identifier detection) or semiautomatically (Synch Break detection) depending on the LIN Header detection mode. The detection mode is selected by the LHDM bit in the SCICR3.

Additionally, an automatic resynchronization feature can be activated to compensate for any clock deviation, for more details please refer to Section 0.1.9.5 LIN Baud Rate.

**LIN Header Handling by a Slave**

Depending on the LIN Header detection method the LINSCI will signal the detection of a LIN Header after the LIN Synch Break or after the Identifier has been successfully received.

**Note:**

It is recommended to combine the Header detection function with Mute mode. Putting the LINSCI in Mute mode allows the detection of Headers only and prevents the reception of any other characters.

This mode can be used to wait for the next Header without being interrupted by the data bytes of the current message in case this message is not relevant for the application.

**Synch Break Detection (LHDM = 0):**

When a LIN Synch Break is received:

– The RDRF bit in the SCISR register is set. It indicates that the content of the shift register is transferred to the SCIDR register, a value of 0x00 is expected for a Break.

– The LHDF flag in the SCICR3 register indicates that a LIN Synch Break Field has been detected.

– An interrupt is generated if the LHIE bit in the SCICR3 register is set and the I[1:0] bits are cleared in the CCR register.

– Then the LIN Synch Field is received and measured.

  – If automatic resynchronization is enabled (LASE bit = 1), the LIN Synch Field is not transferred to the shift register: There is no need to clear the RDRF bit.

  – If automatic resynchronization is disabled (LASE bit = 0), the LIN Synch Field is received as a normal character and transferred to the SCIDR register and RDRF is set.

**Note:**

In LIN slave mode, the FE bit detects all frame error which does not correspond to a break.

**Identifier Detection (LHDM = 1):**

This case is the same as the previous one except that the LHDF and the RDRF flags are set only after the entire header has been received (this is true whether automatic resynchronization is enabled or not). This indicates that the LIN Identifier is available in the SCIDR register.

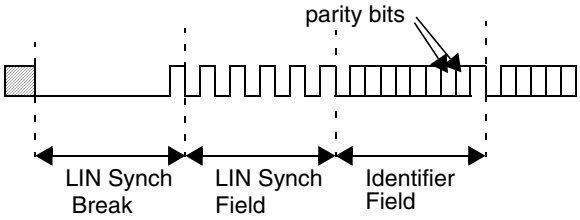
**Notes:**

During LIN Synch Field measurement, the SCI state machine is switched off: No characters are transferred to the data register.

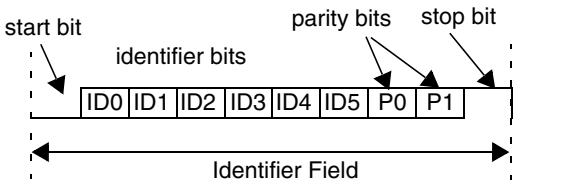
**LIN Slave parity**

In LIN Slave mode (LINE and LSLV bits are set) LIN parity checking can be enabled by setting the PCE bit.

In this case, the parity bits of the LIN Identifier Field are checked. The identifier character is recognized as the third received character after a break character (included):




The bits involved are the two MSB positions (7th and 8th bits if M = 0; 8th and 9th bits if M = 1) of the identifier character. The check is performed as specified by the LIN specification:




$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4$

$P1 = \overline{ID1 \oplus ID3 \oplus ID4 \oplus ID5}$

M = 0

**LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only)** (Cont'd)

**DATA REGISTER (SCIDR)**

Read/Write

Reset Value: Undefined

Contains the Received or Transmitted data character, depending on whether it is read from or written to.

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| DR7 | DR6 | DR5 | DR4 | DR3 | DR2 | DR1 | DR0 |

The Data register performs a double function (read and write) since it is composed of two registers, one for transmission (TDR) and one for reception (RDR).
The TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 88 on page 153).
The RDR register provides the parallel interface between the input shift register and the internal bus (see Figure 88).

**BAUD RATE REGISTER (SCIBRR)**

Read/Write

Reset Value: 0000 0000 (00h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| SCP1 | SCP0 | SCT2 | SCT1 | SCT0 | SCR2 | SCR1 | SCR0 |

Bits 7:6 = **SCP[1:0]** *First SCI Prescaler*
These 2 prescaling bits allow several standard clock division ranges:

| PR Prescaling factor | SCP1 | SCP0 |
|---|---|---|
| 1 | 0 | 0 |
| 3 | | 1 |
| 4 | 1 | 0 |
| 13 | | 1 |

Bits 5:3 = **SCT[2:0]** *SCI Transmitter rate divisor*
These 3 bits, in conjunction with the SCP1 and SCP0 bits define the total division applied to the bus clock to yield the transmit rate clock in conventional Baud Rate Generator mode.

| TR dividing factor | SCT2 | SCT1 | SCT0 |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | | | 1 |
| 4 | | 1 | 0 |
| 8 | | | 1 |
| 16 | 1 | 0 | 0 |
| 32 | | | 1 |
| 64 | | 1 | 0 |
| 128 | | | 1 |

**Note:** This TR factor is used only when the ETPR fine tuning factor is equal to 00h; otherwise, TR is replaced by the (TR*ETPR) dividing factor.

Bits 2:0 = **SCR[2:0]** *SCI Receiver rate divisor.*
These 3 bits, in conjunction with the SCP1 and SCP0 bits define the total division applied to the bus clock to yield the receive rate clock in conventional Baud Rate Generator mode.

| RR dividing factor | SCR2 | SCR1 | SCR0 |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | | | 1 |
| 4 | | 1 | 0 |
| 8 | | | 1 |
| 16 | 1 | 0 | 0 |
| 32 | | | 1 |
| 64 | | 1 | 0 |
| 128 | | | 1 |

**Note:** This RR factor is used only when the ERPR fine tuning factor is equal to 00h; otherwise, RR is replaced by the (RR*ERPR) dividing factor.

**I/O PORT PIN CHARACTERISTICS** (Cont'd)

**Figure 109. Typical $V_{OL}$ vs $V_{DD}$ (Standard I/Os)**

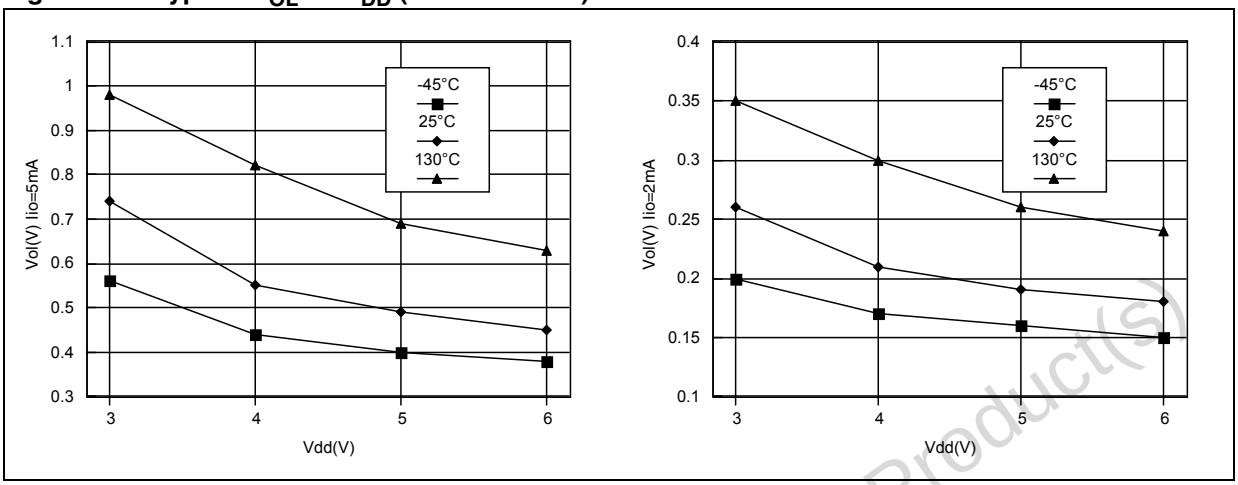
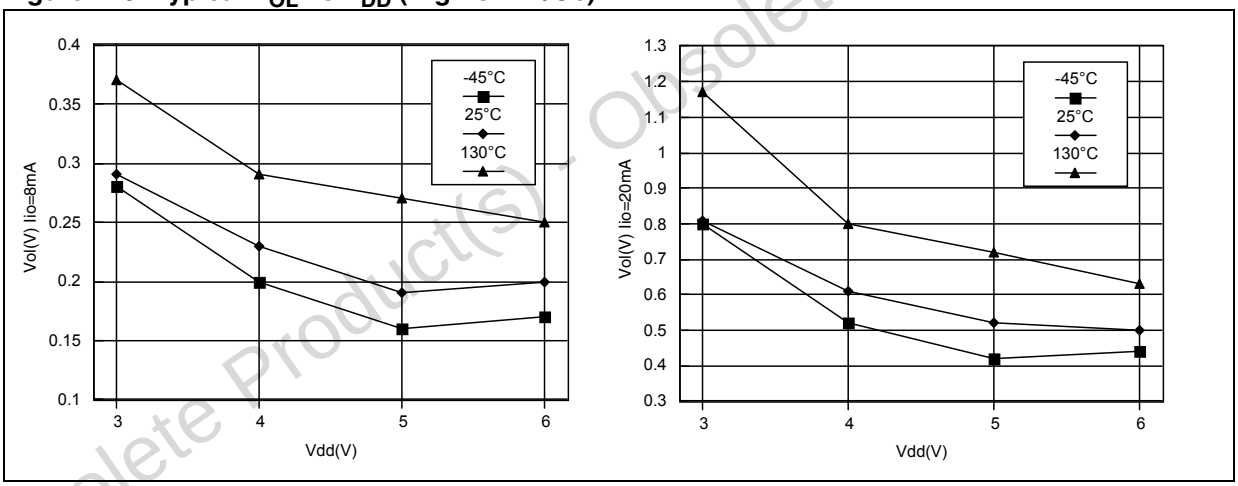



**Figure 110. Typical $V_{OL}$ vs $V_{DD}$ (High-sink I/Os)**



**COMMUNICATION INTERFACE CHARACTERISTICS** (Cont'd)

**Figure 117. SPI Slave Timing Diagram with CPHA = 1**[1)]

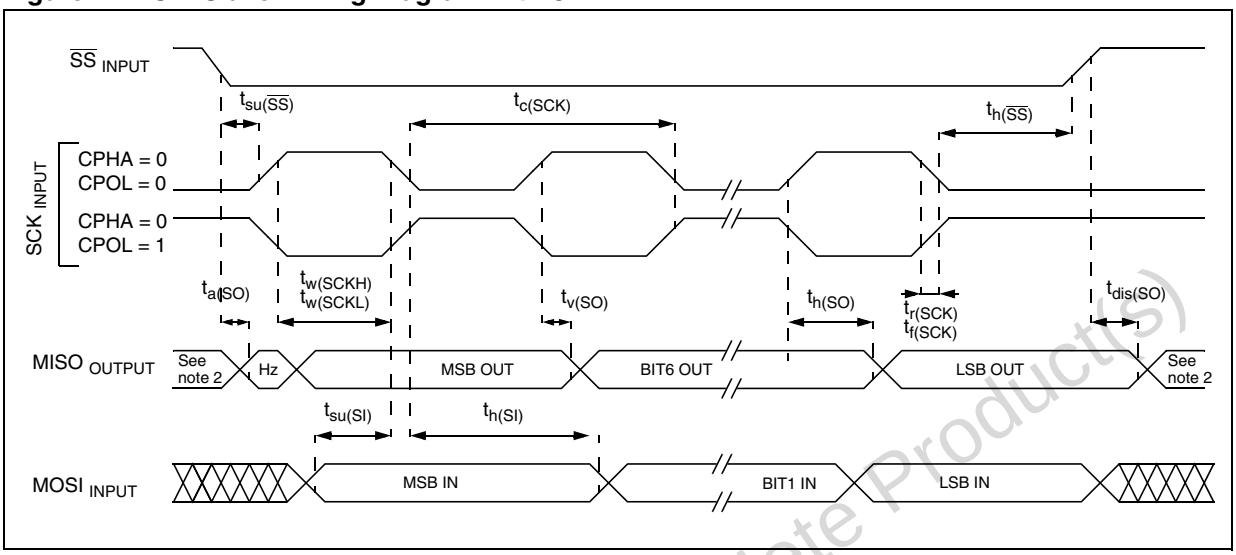
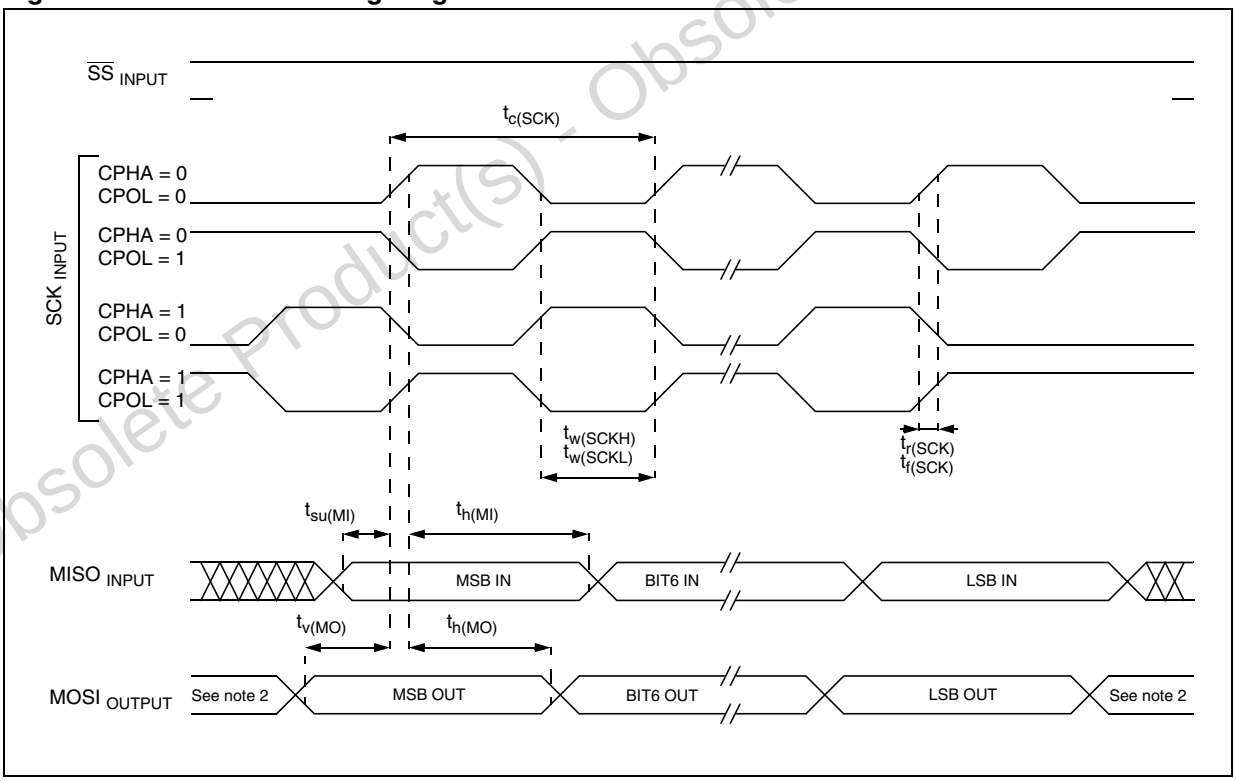



**Figure 118. SPI Master Timing Diagram**[1)]






**Notes:**

1. Measurement points are done at CMOS levels: 0.3 x $V_{DD}$ and 0.7 x $V_{DD}$.

2. When no communication is on-going the data output line of the SPI (MOSI in master mode, MISO in slave mode) has its alternate function capability released. In this case, the pin status depends on the I/O port configuration.

**ADC CHARACTERISTICS** (Cont'd)

**Figure 119. $R_{AIN}$ Max vs $f_{ADC}$ with $C_{AIN}$ = 0pF[1)2)]**

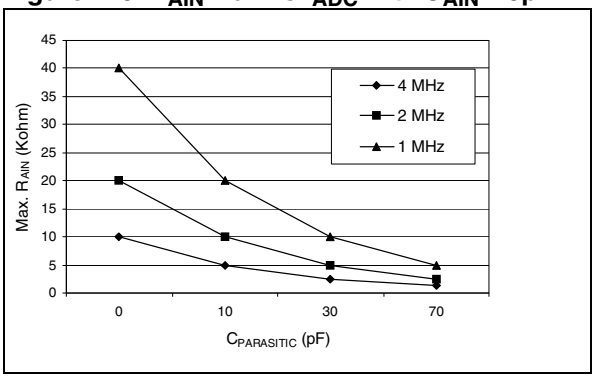



**Figure 120. Recommended $C_{AIN}/R_{AIN}$ Values[3)]**

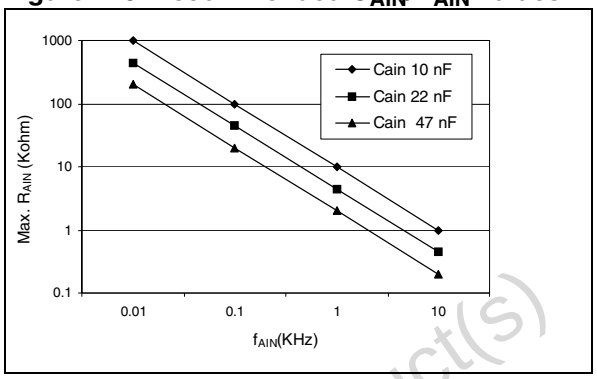
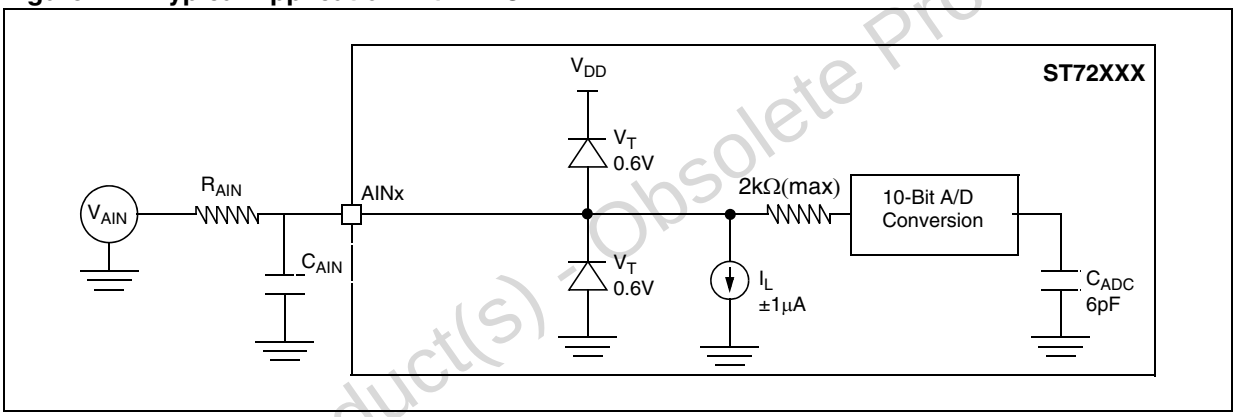



**Figure 121. Typical Application with ADC**




**Notes:**

1. $C_{PARASITIC}$ represents the capacitance of the PCB (dependent on soldering and PCB layout quality) plus the pad capacitance (3pF). A high $C_{PARASITIC}$ value will downgrade conversion accuracy. To remedy this, $f_{ADC}$ should be reduced.

2. Any added external serial resistor will downgrade the ADC accuracy (especially for resistance greater than 10kΩ). Data based on characterization results, not tested in production.

3. This graph shows that depending on the input signal variation ($f_{AIN}$), $C_{AIN}$ can be increased for stabilization time and reduced to allow the use of a larger serial resistor ($R_{AIN}$). It is valid for all $f_{ADC}$ frequencies ≤ 4 MHz.

# 16 IMPORTANT NOTES

## 16.1 ALL DEVICES

### 16.1.1 RESET Pin Protection with LVD Enabled

As mentioned in note 2 below Figure 112 on page 199, when the LVD is enabled, it is recommended not to connect a pull-up resistor or capacitor. A 10nF pull-down capacitor is required to filter noise on the reset line.

### 16.1.2 Clearing Active Interrupts Outside Interrupt Routine

When an active interrupt request occurs at the same time as the related flag or interrupt mask is being cleared, the CC register may be corrupted.

**Concurrent interrupt context**

The symptom does not occur when the interrupts are handled normally, that is, when:

– The interrupt request is cleared (flag reset or interrupt mask) within its own interrupt routine

– The interrupt request is cleared (flag reset or interrupt mask) within any interrupt routine

– The interrupt request is cleared (flag reset or interrupt mask) in any part of the code while this interrupt is disabled

If these conditions are not met, the symptom can be avoided by implementing the following sequence:

Perform SIM and RIM operation before and after resetting an active interrupt request

Example:

```
SIM
reset flag or interrupt mask
RIM
```

**Nested interrupt context**

The symptom does not occur when the interrupts are handled normally, that is, when:

– The interrupt request is cleared (flag reset or interrupt mask) within its own interrupt routine

– The interrupt request is cleared (flag reset or interrupt mask) within any interrupt routine with higher or identical priority level

– The interrupt request is cleared (flag reset or interrupt mask) in any part of the code while this interrupt is disabled

If these conditions are not met, the symptom can be avoided by implementing the following sequence:

```
PUSH CC
SIM
reset flag or interrupt mask
POP CC
```

### 16.1.3 External Interrupt Missed

To avoid any risk of generating a parasitic interrupt, the edge detector is automatically disabled for one clock cycle during an access to either DDR and OR. Any input signal edge during this period will not be detected and will not generate an interrupt.

This case can typically occur if the application refreshes the port configuration registers at intervals during runtime.

**Workaround**

The workaround is based on software checking the level on the interrupt pin before and after writing to the PxOR or PxDDR registers. If there is a level change (depending on the sensitivity programmed for this pin) the interrupt routine is invoked using the call instruction with three extra PUSH instructions before executing the interrupt routine (this is to make the call compatible with the IRET instruction at the end of the interrupt service routine).

But detection of the level change does ensure that edge occurs during the critical 1 cycle duration and the interrupt has been missed. This may lead to occurrence of same interrupt twice (one hardware and another with software call).

To avoid this, a semaphore is set to '1' before checking the level change. The semaphore is changed to level '0' inside the interrupt routine. When a level change is detected, the semaphore status is checked and if it is '1' this means that the last interrupt has been missed. In this case, the interrupt routine is invoked with the call instruction.

There is another possible case, that is, if writing to PxOR or PxDDR is done with global interrupts disabled (interrupt mask bit set). In this case, the semaphore is changed to '1' when the level change is detected. Detecting a missed interrupt is done after the global interrupts are enabled (interrupt mask bit reset) and by checking the status of the semaphore. If it is '1' this means that the last interrupt was missed and the interrupt routine is invoked with the call instruction.

To implement the workaround, the following software sequence is to be followed for writing into the PxOR/PxDDR registers. The example is for Port PF1 with falling edge interrupt sensitivity. The