



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.

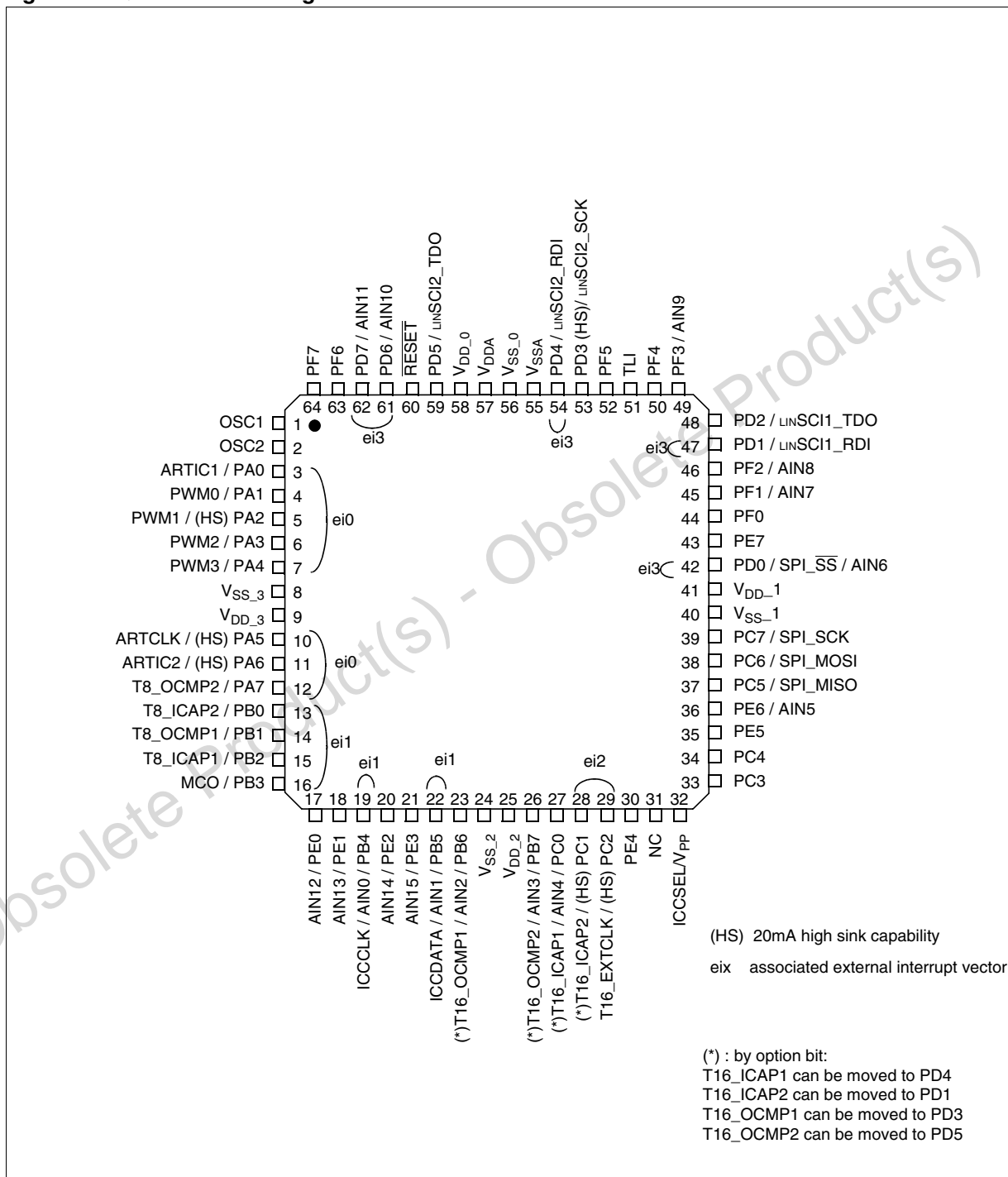
Applications of "[Embedded - Microcontrollers](#)"

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/st72f361j9tc |

2 PIN DESCRIPTION

Figure 2. LQFP 64-Pin Package Pinout



6.2 MULTI-OSCILLATOR (MO)

The main clock of the ST7 can be generated by two different source types coming from the multi-oscillator block:

- an external source
- a crystal or ceramic resonator oscillator

Each oscillator is optimized for a given frequency range in terms of consumption and is selectable through the option byte. The associated hardware configuration are shown in Table 6. Refer to the electrical characteristics section for more details.

Caution: The OSC1 and/or OSC2 pins must not be left unconnected. For the purposes of Failure Mode and Effect Analysis, it should be noted that if the OSC1 and/or OSC2 pins are left unconnected, the ST7 main oscillator may start and, in this configuration, could generate an f_{OSC} clock frequency in excess of the allowed maximum (> 16 MHz), putting the ST7 in an unsafe/undefined state. The product behavior must therefore be considered undefined when the OSC pins are left unconnected.

External Clock Source

In external clock mode, a clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC1 pin while the OSC2 pin is tied to ground.

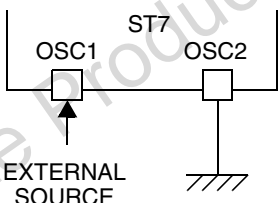
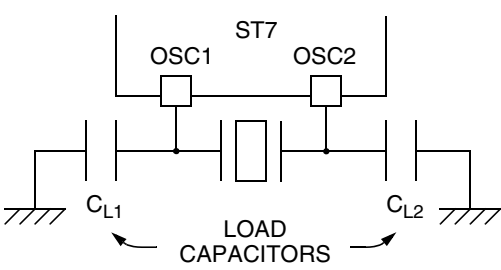
Crystal/Ceramic Oscillators

This family of oscillators has the advantage of producing a very accurate rate on the main clock of the ST7. The selection within a list of five oscillators with different frequency ranges must be done by option byte in order to reduce consumption (refer to Section 14.1 on page 210 for more details on

the frequency ranges). The resonator and the load capacitors must be placed as close as possible to the oscillator pins in order to minimize output distortion and start-up stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

These oscillators are not stopped during the RESET phase to avoid losing time in the oscillator start-up phase.

Table 6. ST7 Clock Sources

| | Hardware Configuration |
|----------------------------|--|
| External Clock |  |
| Crystal/Ceramic Resonators |  |

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.

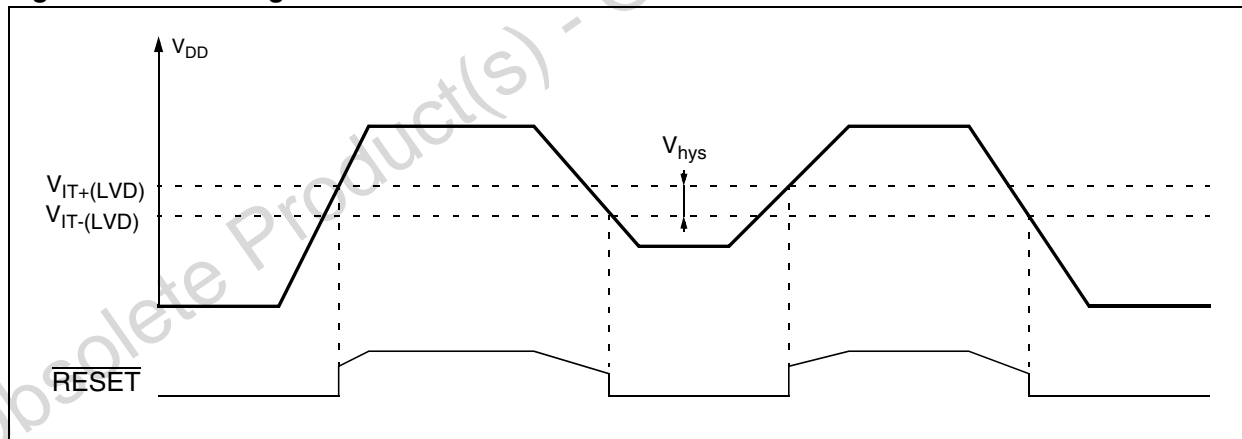
Notes:

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



POWER SAVING MODES (Cont'd)

8.4 HALT MODE

The HALT mode is the lowest power consumption mode of the MCU. It is entered by executing the 'HALT' instruction when the OIE bit of the Main Clock Controller Status register (MCCSR) is cleared (see Section 10.2 on page 58 for more details on the MCCSR register) and when the AWUEN bit in the AWUCSR register is cleared.

The MCU can exit HALT mode on reception of either a specific interrupt (see Table 9, "Interrupt Mapping," on page 33) or a RESET. When exiting HALT mode by means of a RESET or an interrupt, the oscillator is immediately turned on and the 256 or 4096 CPU cycle delay is used to stabilize the oscillator. After the start up delay, the CPU resumes operation by servicing the interrupt or by fetching the reset vector which woke it up (see Figure 26).

When entering HALT mode, the I[1:0] bits in the CC register are forced to '10b' to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes up immediately.

In HALT mode, the main oscillator is turned off causing all internal processing to be stopped, including the operation of the on-chip peripherals. All peripherals are not clocked except the ones which get their clock supply from another clock generator (such as an external or auxiliary oscillator).

The compatibility of Watchdog operation with HALT mode is configured by the "WDGHALT" option bit of the option byte. The HALT instruction when executed while the Watchdog system is enabled, can generate a Watchdog RESET (see Section 10.1 on page 52 for more details).

Figure 25. HALT Timing Overview

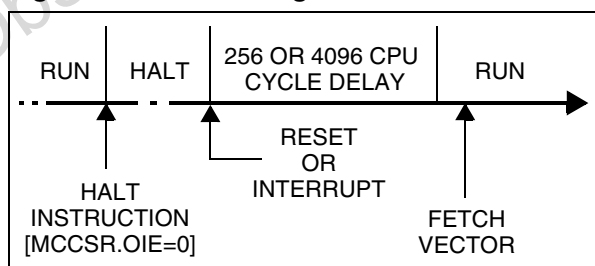
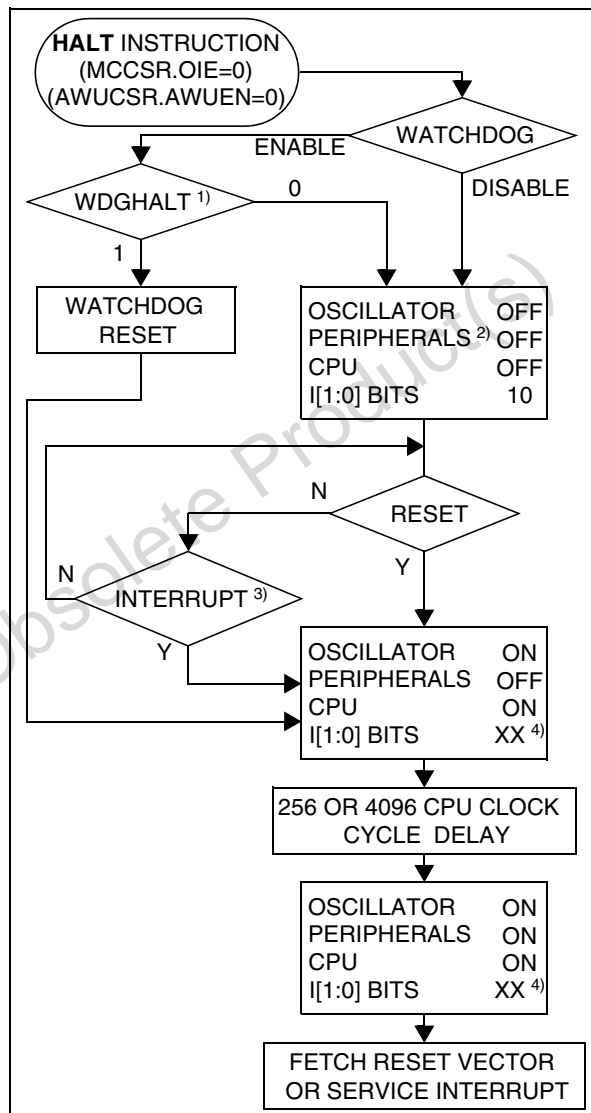


Figure 26. HALT Mode Flow-chart



Notes:

1. WDGHALT is an option bit. See option byte section for more details.
2. Peripheral clocked with an external clock source can still be active.
3. Only some specific interrupts can exit the MCU from HALT mode (such as external interrupt). Refer to Table 9, "Interrupt Mapping," on page 33 for more details.
4. Before servicing an interrupt, the CC register is pushed on the stack. The I[1:0] bits of the CC register are set to the current software priority level of the interrupt routine and recovered when the CC register is popped.

POWER SAVING MODES (Cont'd)**Halt Mode Recommendations**

- Make sure that an external event is available to wake up the microcontroller from Halt mode.
- When using an external interrupt to wake up the microcontroller, reinitialize the corresponding I/O as “Input Pull-up with Interrupt” before executing the HALT instruction. The main reason for this is that the I/O may be wrongly configured due to external interference or by an unforeseen logical condition.
- For the same reason, reinitialize the level sensitivity of each external interrupt as a precautionary measure.
- The opcode for the HALT instruction is 0x8E. To avoid an unexpected HALT instruction due to a program counter failure, it is advised to clear all occurrences of the data value 0x8E from memory. For example, avoid defining a constant in ROM with the value 0x8E.
- As the HALT instruction clears the interrupt mask in the CC register to allow interrupts, the user may choose to clear all pending interrupt bits before executing the HALT instruction. This avoids entering other peripheral interrupt routines after executing the external interrupt routine corresponding to the wake-up event (reset or external interrupt).

8.5 ACTIVE HALT MODE

ACTIVE HALT mode is the lowest power consumption mode of the MCU with a real time clock available. It is entered by executing the ‘HALT’ instruction when MCC/RTC interrupt enable flag (OIE bit in MCCSR register) is set and when the AWUEN bit in the AWUCSR register is cleared (See “Register Description” on page 44.)

| MCCSR OIE bit | Power Saving Mode entered when HALT instruction is executed |
|------------------|--|
| 0 | HALT mode |
| 1 | ACTIVE HALT mode |

The MCU can exit ACTIVE HALT mode on reception of the RTC interrupt and some specific interrupts (see Table 9, “Interrupt Mapping,” on page 33) or a RESET. When exiting ACTIVE HALT mode by means of a RESET a 4096 or 256 CPU cycle delay occurs (depending on the option byte). After the start up delay, the CPU resumes operation by servicing the interrupt or by fetching the reset vector which woke it up (see Figure 28).

When entering ACTIVE HALT mode, the I[1:0] bits in the CC register are forced to ‘10b’ to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes up immediately.

In ACTIVE HALT mode, only the main oscillator and its associated counter (MCC/RTC) are running to keep a wake-up time base. All other peripherals are not clocked except those which get their clock supply from another clock generator (such as external or auxiliary oscillator).

The safeguard against staying locked in ACTIVE HALT mode is provided by the oscillator interrupt.

Note: As soon as active halt is enabled, executing a HALT instruction while the Watchdog is active does not generate a RESET.

This means that the device cannot spend more than a defined delay in this power saving mode.

I/O PORTS (Cont'd)

9.6 I/O PORT REGISTER CONFIGURATIONS

The I/O port register configurations are summarized as follows.

9.6.1 Standard Ports

PB7:6, PC0, PC3, PC7:5, PD3:2, PD5, PE7:0, PF7:0

| MODE | DDR | OR |
|-------------------|-----|----|
| floating input | 0 | 0 |
| pull-up input | 0 | 1 |
| open drain output | 1 | 0 |
| push-pull output | 1 | 1 |

9.6.2 Interrupt Ports

PA0,2,4,6; PB0,2,4; PC1; PD0,6

(with pull-up)

| MODE | DDR | OR |
|-------------------------|-----|----|
| floating input | 0 | 0 |
| pull-up interrupt input | 0 | 1 |
| open drain output | 1 | 0 |
| push-pull output | 1 | 1 |

PA1,3,5,7; PB1,3,5; PC2; PD1,4,7

(without pull-up)

| MODE | DDR | OR |
|--------------------------|-----|----|
| floating input | 0 | 0 |
| floating interrupt input | 0 | 1 |
| open drain output | 1 | 0 |
| push-pull output | 1 | 1 |

9.6.3 Pull-up Input Port

PC4

| MODE |
|---------------|
| pull-up input |

The PC4 port cannot operate as a general purpose output. If DDR = 1 it is still possible to read the port through the DR register.

10 ON-CHIP PERIPHERALS

10.1 WINDOW WATCHDOG (WWDG)

10.1.1 Introduction

The Window Watchdog is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The Watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

10.1.2 Main Features

- Programmable free-running downcounter
- Conditional reset
 - Reset (if watchdog activated) when the downcounter value becomes less than 40h
 - Reset (if watchdog activated) if the down-

counter is reloaded outside the window (see Figure 4)

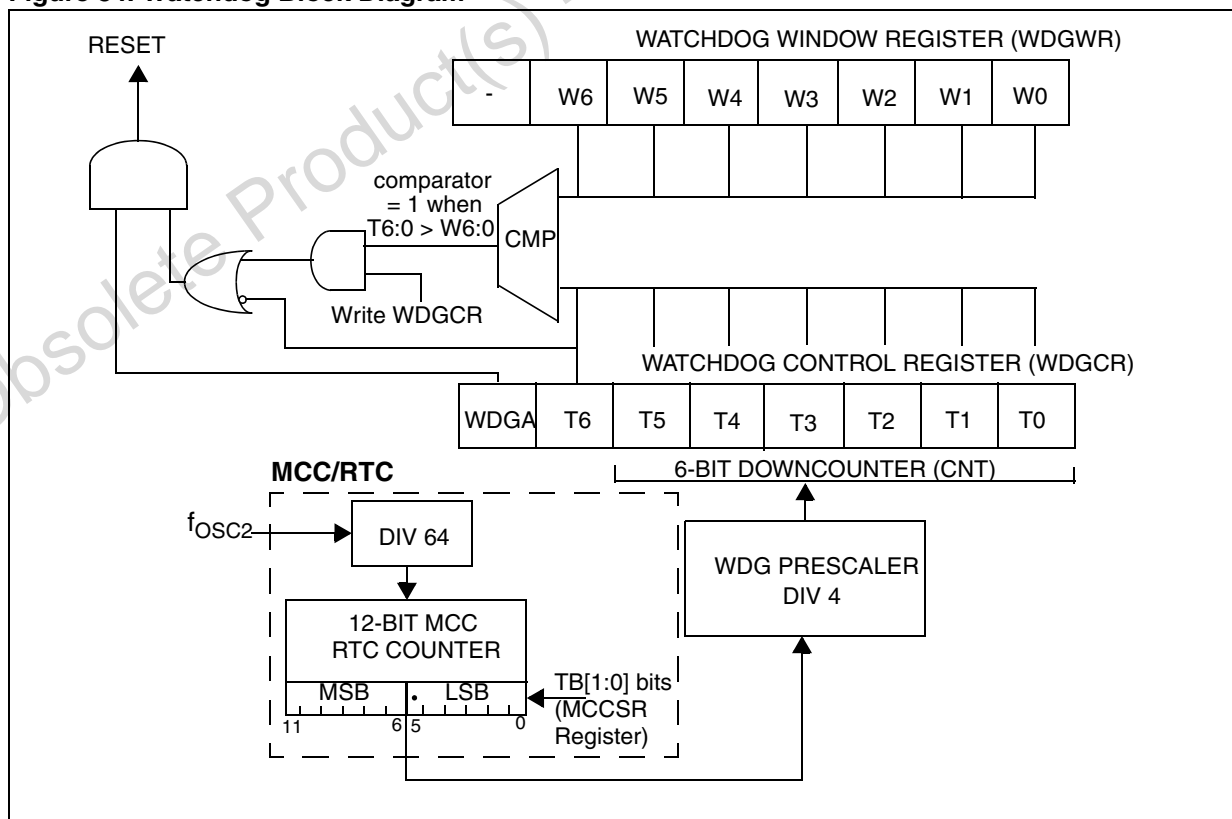
- Hardware/Software Watchdog activation (selectable by option byte)
- Optional reset on HALT instruction (configurable by option byte)

10.1.3 Functional Description

The counter value stored in the WDGCR register (bits T[6:0]), is decremented every $16384 f_{OSC2}$ cycles (approx.), and the length of the timeout period can be programmed by the user in 64 increments.

If the watchdog is activated (the WDGA bit is set) and when the 7-bit downcounter (T[6:0] bits) rolls over from 40h to 3Fh (T6 becomes cleared), it initiates a reset cycle pulling low the reset pin for typically $30\mu s$. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

Figure 34. Watchdog Block Diagram



WINDOW WATCHDOG (Cont'd)

Figure 36. Exact Timeout Duration (t_{\min} and t_{\max})**WHERE:**

$$t_{\min 0} = (\text{LSB} + 128) \times 64 \times t_{\text{OSC2}}$$

$$t_{\max 0} = 16384 \times t_{\text{OSC2}}$$

$$t_{\text{OSC2}} = 125\text{ns if } f_{\text{OSC2}} = 8 \text{ MHz}$$

CNT = Value of T[5:0] bits in the WDGCR register (6 bits)

MSB and LSB are values from the table below depending on the timebase selected by the TB[1:0] bits in the MCCR register

| TB1 Bit (MCCR Reg.) | TB0 Bit (MCCR Reg.) | Selected MCCR Timebase | MSB | LSB |
|------------------------|------------------------|---------------------------|-----|-----|
| 0 | 0 | 2ms | 4 | 59 |
| 0 | 1 | 4ms | 8 | 53 |
| 1 | 0 | 10ms | 20 | 35 |
| 1 | 1 | 25ms | 49 | 54 |

To calculate the minimum Watchdog Timeout (t_{\min}):

$$\text{IF } \text{CNT} < \left\lceil \frac{\text{MSB}}{4} \right\rceil \quad \text{THEN } t_{\min} = t_{\min 0} + 16384 \times \text{CNT} \times t_{\text{osc2}}$$

$$\text{ELSE } t_{\min} = t_{\min 0} + \left[16384 \times \left(\text{CNT} - \left\lceil \frac{4\text{CNT}}{\text{MSB}} \right\rceil \right) + (192 + \text{LSB}) \times 64 \times \left\lceil \frac{4\text{CNT}}{\text{MSB}} \right\rceil \right] \times t_{\text{osc2}}$$

To calculate the maximum Watchdog Timeout (t_{\max}):

$$\text{IF } \text{CNT} \leq \left\lceil \frac{\text{MSB}}{4} \right\rceil \quad \text{THEN } t_{\max} = t_{\max 0} + 16384 \times \text{CNT} \times t_{\text{osc2}}$$

$$\text{ELSE } t_{\max} = t_{\max 0} + \left[16384 \times \left(\text{CNT} - \left\lceil \frac{4\text{CNT}}{\text{MSB}} \right\rceil \right) + (192 + \text{LSB}) \times 64 \times \left\lceil \frac{4\text{CNT}}{\text{MSB}} \right\rceil \right] \times t_{\text{osc2}}$$

Note: In the above formulae, division results must be rounded down to the next integer value.

Example:

With 2ms timeout selected in MCCR register

| Value of T[5:0] Bits in WDGCR Register (Hex.) | Min. Watchdog Timeout (ms) t_{\min} | Max. Watchdog Timeout (ms) t_{\max} |
|--|---|---|
| 00 | 1.496 | 2.048 |
| 3F | 128 | 128.552 |

10.4 16-BIT TIMER

10.4.1 Introduction

The timer consists of a 16-bit free-running counter driven by a programmable prescaler.

It may be used for a variety of purposes, including pulse length measurement of up to two input signals (*input capture*) or generation of up to two output waveforms (*output compare* and *PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the CPU clock prescaler.

Some ST7 devices have two on-chip 16-bit timers. They are completely independent, and do not share any resources. They are synchronized after a MCU reset as long as the timer clock frequencies are not modified.

This description covers one or two 16-bit timers. In ST7 devices with two timers, register names are prefixed with TA (Timer A) or TB (Timer B).

10.4.2 Main Features

- Programmable prescaler: f_{CPU} divided by 2, 4 or 8
- Overflow status flag and maskable interrupt
- External clock input (must be at least four times slower than the CPU clock speed) with the choice of active edge
- 1 or 2 Output Compare functions each with:
 - 2 dedicated 16-bit registers
 - 2 dedicated programmable signals
 - 2 dedicated status flags
 - 1 dedicated maskable interrupt
- 1 or 2 Input Capture functions each with:
 - 2 dedicated 16-bit registers
 - 2 dedicated active edge selection signals
 - 2 dedicated status flags
 - 1 dedicated maskable interrupt
- Pulse width modulation mode (PWM)
- One Pulse mode
- Reduced Power Mode
- 5 alternate functions on I/O ports (ICAP1, ICAP2, OCMP1, OCMP2, EXTCLK)*

The Block Diagram is shown in Figure 48.

***Note:** Some timer pins may not be available (not bonded) in some ST7 devices. Refer to the device pin out description.

When reading an input signal on a non-bonded pin, the value will always be '1'.

10.4.3 Functional Description

10.4.3.1 Counter

The main block of the Programmable Timer is a 16-bit free running upcounter and its associated 16-bit registers. The 16-bit registers are made up of two 8-bit registers called high and low.

Counter Register (CR):

- Counter High Register (CHR) is the most significant byte (MS Byte).
- Counter Low Register (CLR) is the least significant byte (LS Byte).

Alternate Counter Register (ACR)

- Alternate Counter High Register (ACHR) is the most significant byte (MS Byte).
- Alternate Counter Low Register (ACLR) is the least significant byte (LS Byte).

These two read-only 16-bit registers contain the same value but with the difference that reading the ACLR register does not clear the TOF bit (Timer overflow flag), located in the Status register, (SR), (see note at the end of paragraph titled 16-bit read sequence).

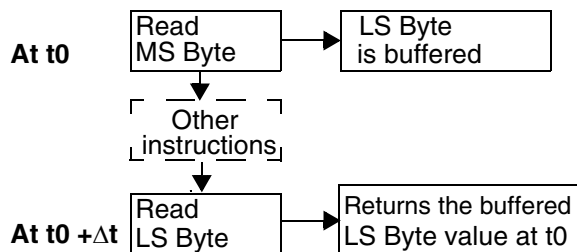
Writing in the CLR register or ACLR register resets the free running counter to the FFFCh value. Both counters have a reset value of FFFCh (this is the only value which is reloaded in the 16-bit timer). The reset value of both counters is also FFFCh in One Pulse mode and PWM mode.

The timer clock depends on the clock control bits of the CR2 register, as illustrated in Table 17 Clock Control Bits. The value in the counter register repeats every 131072, 262144 or 524288 CPU clock cycles depending on the CC[1:0] bits. The timer frequency can be $f_{CPU}/2$, $f_{CPU}/4$, $f_{CPU}/8$ or an external frequency.

16-BIT TIMER (Cont'd)

16-bit read sequence: (from either the Counter Register or the Alternate Counter Register).

Beginning of the sequence



Sequence completed

The user must read the MS Byte first, then the LS Byte value is buffered automatically.

This buffered value remains unchanged until the 16-bit read sequence is completed, even if the user reads the MS Byte several times.

After a complete reading sequence, if only the CLR register or ACLR register are read, they return the LS Byte of the count value at the time of the read.

Whatever the timer mode used (input capture, output compare, One Pulse mode or PWM mode) an overflow occurs when the counter rolls over from FFFFh to 0000h then:

- The TOF bit of the SR register is set.
- A timer interrupt is generated if:
 - TOIE bit of the CR1 register is set and
 - I bit of the CC register is cleared.

If one of these conditions is false, the interrupt remains pending to be issued as soon as they are both true.

Clearing the overflow interrupt request is done in two steps:

1. Reading the SR register while the TOF bit is set.
2. An access (read or write) to the CLR register.

Notes: The TOF bit is not cleared by accesses to ACLR register. The advantage of accessing the ACLR register rather than the CLR register is that it allows simultaneous use of the overflow function and reading the free running counter at random times (for example, to measure elapsed time) without the risk of clearing the TOF bit erroneously.

The timer is not affected by WAIT mode.

In HALT mode, the counter stops counting until the mode is exited. Counting then resumes from the previous count (MCU awakened by an interrupt) or from the reset count (MCU awakened by a Reset).

10.4.3.2 External Clock

The external clock (where available) is selected if CC0 = 1 and CC1 = 1 in the CR2 register.

The status of the EXEDG bit in the CR2 register determines the type of level transition on the external clock pin EXTCLK that will trigger the free running counter.

The counter is synchronized with the falling edge of the internal CPU clock.

A minimum of four falling edges of the CPU clock must occur between two consecutive active edges of the external clock; thus the external clock frequency must be less than a quarter of the CPU clock frequency.

8-BIT TIMER (Cont'd)

10.5.3.4 One Pulse Mode

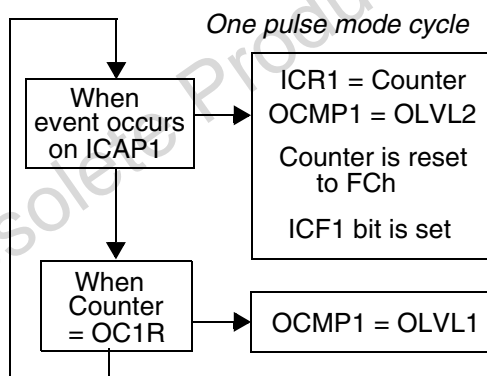
One Pulse mode enables the generation of a pulse when an external event occurs. This mode is selected via the OPM bit in the CR2 register.

The one pulse mode uses the Input Capture1 function and the Output Compare1 function.

Procedure:

To use one pulse mode:

1. Load the OC1R register with the value corresponding to the length of the pulse (see the formula in the opposite column).
2. Select the following in the CR1 register:
 - Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after the pulse.
 - Using the OLVL2 bit, select the level to be applied to the OCMP1 pin during the pulse.
 - Select the edge of the active transition on the ICAP1 pin with the IEDG1 bit (the ICAP1 pin must be configured as floating input).
3. Select the following in the CR2 register:
 - Set the OC1E bit, the OCMP1 pin is then dedicated to the Output Compare 1 function.
 - Set the OPM bit.
 - Select the timer clock CC[1:0] (see Table 19 Clock Control Bits).



Then, on a valid event on the ICAP1 pin, the counter is initialized to FCh and OLVL2 bit is loaded on the OCMP1 pin, the ICF1 bit is set and the value FFFDh is loaded in the IC1R register.

Because the ICF1 bit is set when an active edge occurs, an interrupt can be generated if the ICIE bit is set.

Clearing the Input Capture interrupt request (that is, clearing the ICF1 bit) is done in two steps:

1. Reading the SR register while the ICF1 bit is set.
2. An access (read or write) to the IC1LR register.

The OC1R register value required for a specific timing application can be calculated using the following formula:

$$\text{OC1R Value} = \frac{t \cdot f_{\text{CPU}}}{\text{PRESC}} - 5$$

Where:

t = Pulse period (in seconds)

f_{CPU} = PLL output x2 clock frequency in hertz (or $f_{\text{OSC}}/2$ if PLL is not enabled)

PRESC = Timer prescaler factor (2, 4, 8 or 8000 depending on the CC[1:0] bits, see Table 19 Clock Control Bits)

When the value of the counter is equal to the value of the contents of the OC1R register, the OLVL1 bit is output on the OCMP1 pin, (See Figure 68).

Notes:

1. The OCF1 bit cannot be set by hardware in one pulse mode but the OCF2 bit can generate an Output Compare interrupt.
2. When the Pulse Width Modulation (PWM) and One Pulse Mode (OPM) bits are both set, the PWM mode is the only active one.
3. If OLVL1=OLVL2 a continuous signal will be seen on the OCMP1 pin.
4. The ICAP1 pin can not be used to perform input capture. 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 time a valid edge occurs on the ICAP1 pin and ICF1 can also generates interrupt if ICIE is set.
5. When one pulse mode is used OC1R is dedicated to this mode. Nevertheless OC2R and OCF2 can be used to indicate a period of time has been elapsed but cannot generate an output waveform because the level OLVL2 is dedicated to the one pulse mode.

8-BIT TIMER (Cont'd)**10.5.8 8-bit Timer Register Map**

| Address (Hex.) | Register Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------------|------------------|------|------|------|-------|-------|-------|-------|-------|
| 3C | CR2 | OC1E | OC2E | OPM | PWM | CC1 | CC0 | IEDG2 | 0 |
| 3D | CR1 | ICIE | OCIE | TOIE | FOLV2 | FOLV1 | OLVL2 | IEDG1 | OLVL1 |
| 3E | CSR | ICF1 | OCF1 | TOF | ICF2 | OCF2 | TIMD | | |
| 3F | IC1R | MSB | | | | | | | LSB |
| 40 | OC1R | MSB | | | | | | | LSB |
| 41 | CTR | MSB | | | | | | | LSB |
| 42 | ACTR | MSB | | | | | | | LSB |
| 43 | IC2R | MSB | | | | | | | LSB |
| 44 | OC2R | MSB | | | | | | | LSB |

LINSI™ 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 re-

ceived 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 | SB 8 bit data STB |
| | 1 | SB 7-bit data PB STB |
| 1 | 0 | SB 9-bit data STB |
| | 1 | SB 8-bit data PB STB |

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.

LINSPI™ SERIAL COMMUNICATION INTERFACE (LIN Mode) (cont'd)**10.7.9.9 Error due to LIN Synch measurement**

The LIN Synch Field is measured over eight bit times.

This measurement is performed using a counter clocked by the CPU clock. The edge detections are performed using the CPU clock cycle.

This leads to a precision of 2 CPU clock cycles for the measurement which lasts $16 \times 8 \times \text{LDIV}$ clock cycles.

Consequently, this error (D_{MEAS}) is equal to:

$$2 / (128 \times \text{LDIV}_{\text{MIN}}).$$

LDIV_{MIN} corresponds to the minimum LIN prescaler content, leading to the maximum baud rate, taking into account the maximum deviation of +/-15%.

10.7.9.10 Error due to Baud Rate Quantization

The baud rate can be adjusted in steps of $1 / (16 \times \text{LDIV})$. The worst case occurs when the "real" baud rate is in the middle of the step.

This leads to a quantization error (D_{QUANT}) equal to $1 / (2 \times 16 \times \text{LDIV}_{\text{MIN}})$.

10.7.9.11 Impact of Clock Deviation on Maximum Baud Rate

The choice of the nominal baud rate (LDIV_{NOM}) will influence both the quantization error (D_{QUANT}) and the measurement error (D_{MEAS}). The worst case occurs for LDIV_{MIN} .

Consequently, at a given CPU frequency, the maximum possible nominal baud rate (LPR_{MIN}) should be chosen with respect to the maximum tolerated deviation given by the equation:

$$D_{\text{TRA}} + 2 / (128 \times \text{LDIV}_{\text{MIN}}) + 1 / (2 \times 16 \times \text{LDIV}_{\text{MIN}}) + D_{\text{REC}} + D_{\text{TCL}} < 3.75\%$$

Example:

A nominal baud rate of 20Kbits/s at $T_{\text{CPU}} = 125\text{ns}$ (8 MHz) leads to $\text{LDIV}_{\text{NOM}} = 25\text{d}$.

$$\text{LDIV}_{\text{MIN}} = 25 - 0.15 \times 25 = 21.25$$

$$D_{\text{MEAS}} = 2 / (128 \times \text{LDIV}_{\text{MIN}}) \times 100 = 0.00073\%$$

$$D_{\text{QUANT}} = 1 / (2 \times 16 \times \text{LDIV}_{\text{MIN}}) \times 100 = 0.0015\%$$

LIN Slave systems

For LIN Slave systems (the LINE and LSLV bits are set), receivers wake up by LIN Synch Break or LIN Identifier detection (depending on the LHDM bit).

Hot Plugging Feature for LIN Slave Nodes

In LIN Slave Mute Mode (the LINE, LSLV and RWU bits are set) it is possible to hot plug to a network during an ongoing communication flow. In this case the SCI monitors the bus on the RDI line until 11 consecutive dominant bits have been detected and discards all the other bits received.

LINSICI™ 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.

INSTRUCTION SET OVERVIEW (Cont'd)

11.1.1 Inherent

All Inherent instructions consist of a single byte. The opcode fully specifies all the required information for the CPU to process the operation.

| Inherent Instruction | Function |
|-------------------------|-------------------------------------|
| NOP | No operation |
| TRAP | S/W Interrupt |
| WFI | Wait For Interrupt (Low Power Mode) |
| HALT | Halt Oscillator (Lowest Power Mode) |
| RET | Sub-routine Return |
| IRET | Interrupt Sub-routine Return |
| SIM | Set Interrupt Mask (level 3) |
| RIM | Reset Interrupt Mask (level 0) |
| SCF | Set Carry Flag |
| RCF | Reset Carry Flag |
| RSP | Reset Stack Pointer |
| LD | Load |
| CLR | Clear |
| PUSH/POP | Push/Pop to/from the stack |
| INC/DEC | Increment/Decrement |
| TNZ | Test Negative or Zero |
| CPL, NEG | 1 or 2 Complement |
| MUL | Byte Multiplication |
| SLL, SRL, SRA, RLC, RRC | Shift and Rotate Operations |
| SWAP | Swap Nibbles |

11.1.2 Immediate

Immediate instructions have 2 bytes, the first byte contains the opcode, the second byte contains the operand value.

| Immediate Instruction | Function |
|-----------------------|-----------------------|
| LD | Load |
| CP | Compare |
| BCP | Bit Compare |
| AND, OR, XOR | Logical Operations |
| ADC, ADD, SUB, SBC | Arithmetic Operations |

11.1.3 Direct

In Direct instructions, the operands are referenced by their memory address.

The direct addressing mode consists of two sub-modes:

Direct (short)

The address is a byte, thus requires only one byte after the opcode, but only allows 00 - FF addressing space.

Direct (long)

The address is a word, thus allowing 64 Kbyte addressing space, but requires 2 bytes after the opcode.

11.1.4 Indexed (No Offset, Short, Long)

In this mode, the operand is referenced by its memory address, which is defined by the unsigned addition of an index register (X or Y) with an offset.

The indirect addressing mode consists of three submodes:

Indexed (No Offset)

There is no offset, (no extra byte after the opcode), and allows 00 - FF addressing space.

Indexed (Short)

The offset is a byte, thus requires only one byte after the opcode and allows 00 - 1FE addressing space.

Indexed (long)

The offset is a word, thus allowing 64 Kbyte addressing space and requires 2 bytes after the opcode.

11.1.5 Indirect (Short, Long)

The required data byte to do the operation is found by its memory address, located in memory (pointer).

The pointer address follows the opcode. The indirect addressing mode consists of two submodes:

Indirect (short)

The pointer address is a byte, the pointer size is a byte, thus allowing 00 - FF addressing space, and requires 1 byte after the opcode.

Indirect (long)

The pointer address is a word, the pointer size is a word, thus allowing 64 Kbyte addressing space, and requires 1 byte after the opcode.

I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 103. Connecting Unused I/O Pins

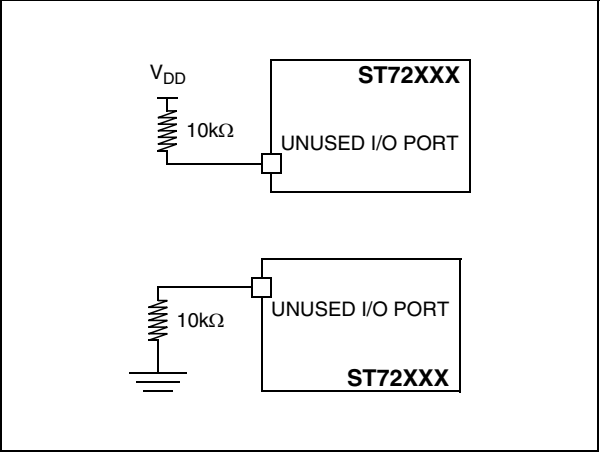


Figure 105. I_{PU} vs V_{DD} with V_{IN} = V_{SS}

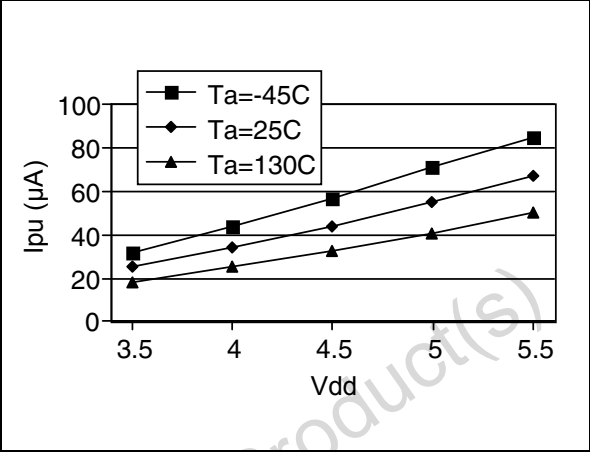
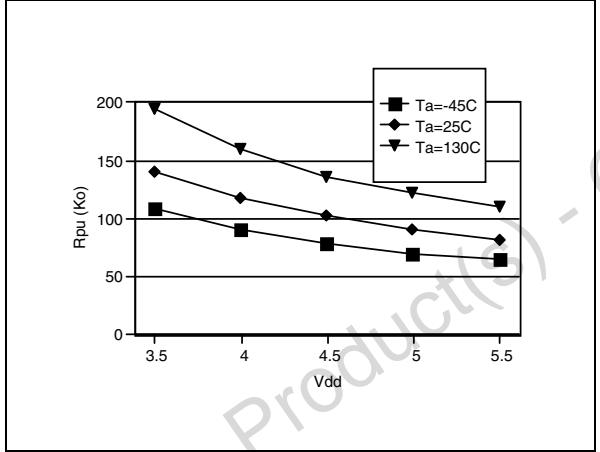


Figure 104. R_{PU} vs V_{DD} with V_{IN} = V_{SS}



12.12 COMMUNICATION INTERFACE CHARACTERISTICS

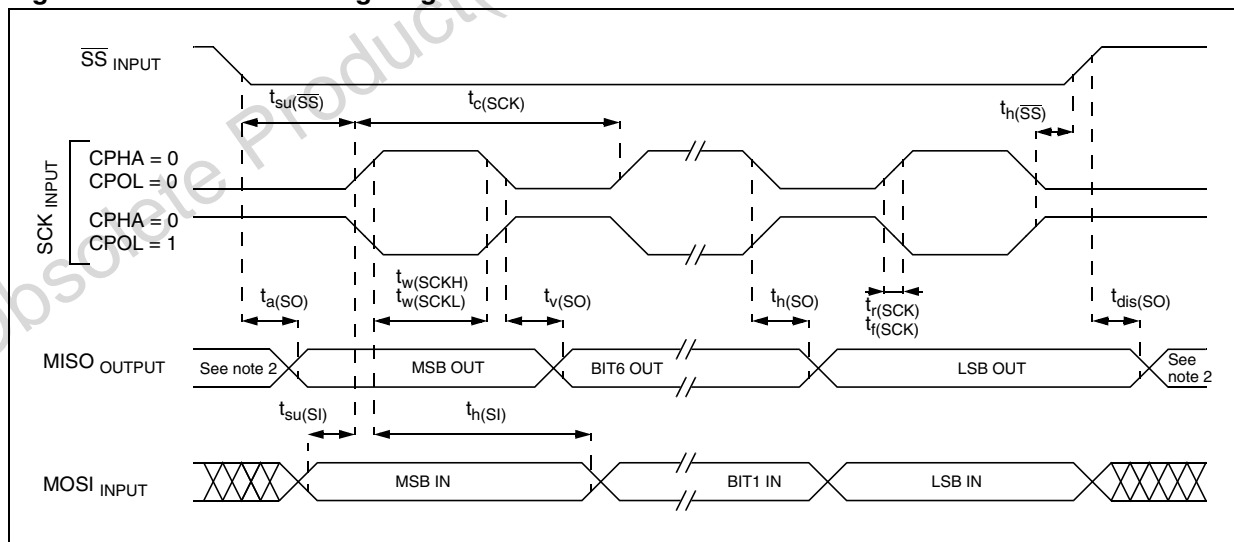
12.12.1 SPI - Serial Peripheral Interface

Subject to general operating conditions for V_{DD} , f_{OSC} , and T_A unless otherwise specified.

Refer to I/O port characteristics for more details on the input/output alternate function characteristics (\overline{SS} , SCK, MOSI, MISO).

| Symbol | Parameter | Conditions | Min | Max | Unit |
|------------------------------|--|-----------------------------------|------------------------------|-------------------|------|
| $f_{SCK} = 1 / t_c(SCK)$ | SPI clock frequency | Master, $f_{CPU} = 8 \text{ MHz}$ | $f_{CPU} / 128 = 0.0625$ | $f_{CPU} / 4 = 2$ | MHz |
| | | Slave, $f_{CPU} = 8 \text{ MHz}$ | 0 | $f_{CPU} / 2 = 4$ | |
| $t_r(SCK)$ | SPI clock rise and fall time | | See I/O port pin description | | |
| $t_f(SCK)$ | | | | | |
| $t_{su}(\overline{SS})^{1)}$ | \overline{SS} setup time ⁴⁾ | Slave | $(4 \times T_{CPU}) + 50$ | | ns |
| $t_h(\overline{SS})^{1)}$ | \overline{SS} hold time | | 120 | | |
| $t_w(SCKH)^{1)}$ | SCK high and low time | Master | 100 | | |
| $t_w(SCKL)^{1)}$ | | Slave | 90 | | |
| $t_{su}(MI)^{1)}$ | Data input setup time | Master | 100 | | |
| $t_{su}(SI)^{1)}$ | | Slave | | | |
| $t_h(MI)^{1)}$ | Data input hold time | Master | | | |
| $t_h(SI)^{1)}$ | | Slave | | | |
| $t_a(SO)^{1)}$ | Data output access time | Slave | 0 | 120 | |
| $t_{dis}(SO)^{1)}$ | Data output disable time | | 240 | | |
| $t_v(SO)^{1)}$ | Data output valid time | Slave (after enable edge) | | 90 | |
| $t_h(SO)^{1)}$ | Data output hold time | | 0 | | |
| $t_v(MO)^{1)}$ | Data output valid time | Master (after enable edge) | | 120 | |
| $t_h(MO)^{1)}$ | Data output hold time | | 0 | | |

Figure 116. SPI Slave Timing Diagram with $CPHA = 0^{(3)}$



Notes:

1. Data based on design simulation and/or characterization results, not tested in production.
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.
3. Measurement points are done at CMOS levels: $0.3 \times V_{DD}$ and $0.7 \times V_{DD}$.
4. Depends on f_{CPU} . For example, if $f_{CPU} = 8 \text{ MHz}$, then $T_{CPU} = 1 / f_{CPU} = 125 \text{ ns}$ and $t_{su}(\overline{SS}) = 550 \text{ ns}$.

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