STMicroelectronics - ST72F561AR4TAE Datasheet

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	CANbus, LINbusSCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	48
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	3.8V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st72f561ar4tae

3 REGISTER AND MEMORY MAP

As shown in Figure 5, the MCU is capable of addressing 64 Kbytes of memories and I/O registers.

The available memory locations consist of 128 bytes of register locations, up to 2 Kbytes of RAM and up to 60 Kbytes of user program memory.

The RAM space includes up to 256 bytes for the stack from 0100h to 01FFh.The highest address bytes contain the user reset and interrupt vectors.

IMPORTANT: Memory locations marked as “Reserved” must never be accessed. Accessing a reseved area can have unpredictable effects on the device.

Figure 5. Memory Map

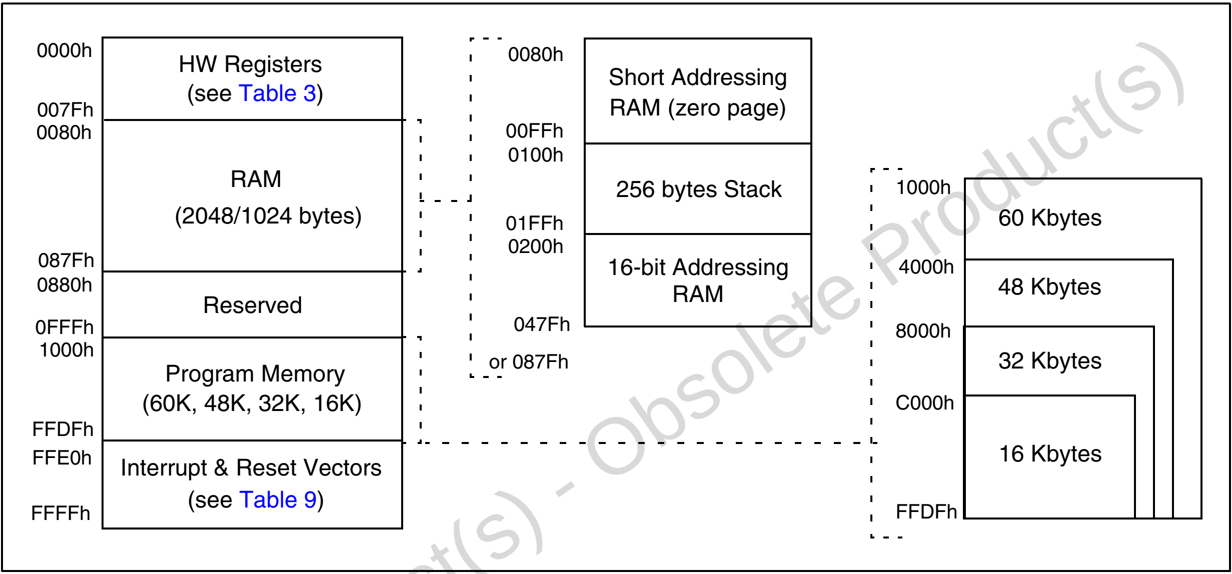


Table 3. Hardware Register Map

Address	Block	Register Label	Register Name	Reset Status	Remarks
0000h 0001h 0002h	Port A	PADR PADDR PAOR	Port A Data Register Port A Data Direction Register Port A Option Register	00h[1)] 00h 00h	R/W[2)] R/W[2)] R/W[2)]
0003h 0004h 0005h	Port B	PBDR PBDDR PBOR	Port B Data Register Port B Data Direction Register Port B Option Register	00h[1)] 00h 00h	R/W[2)] R/W[2)] R/W[2)]
0006h 0007h 0008h	Port C	PCDR PCDDR PCOR	Port C Data Register Port C Data Direction Register Port C Option Register	00h[1)] 00h 00h	R/W[2)] R/W[2)] R/W[2)]
0009h 000Ah 000Bh	Port D	PDDR PDDDR PDOR	Port D Data Register Port D Data Direction Register Port D Option Register	00h[1)] 00h 00h	R/W[2)] R/W[2)] R/W[2)]
000Ch 000Dh 000Eh	Port E	PEDR PEDDR PEOR	Port E Data Register Port E Data Direction Register Port E Option Register	00h[1)] 00h 00h	R/W[2)] R/W[2)] R/W[2)]

Address	Block	Register Label	Register Name	Reset Status	Remarks
0048h	LINSCI1 (LIN Master/ Slave)	SCI1ISR	SCI1 Status Register	C0h	Read Only
0049h		SCI1DR	SCI1 Data Register	xxh	R/W
004Ah		SCI1BRR	SCI1 Baud Rate Register	00h	R/W
004Bh		SCI1CR1	SCI1 Control Register 1	xxh	R/W
004Ch		SCI1CR2	SCI1 Control Register 2	00h	R/W
004Dh		SCI1CR3	SCI1Control Register 3	00h	R/W
004Eh		SCI1ERPR	SCI1 Extended Receive Prescaler Register	00h	R/W
004Fh		SCI1ETPR	SCI1 Extended Transmit Prescaler Register	00h	R/W
0050h	Reserved Area (1 byte)				
0051h	16-BIT TIMER	T16CR2	Timer Control Register 2	00h	R/W
0052h		T16CR1	Timer Control Register 1	00h	R/W
0053h		T16CSR	Timer Control/Status Register	00h	R/W
0054h		T16IC1HR	Timer Input Capture 1 High Register	xxh	Read Only
0055h		T16IC1LR	Timer Input Capture 1 Low Register	xxh	Read Only
0056h		T16OC1HR	Timer Output Compare 1 High Register	80h	R/W
0057h		T16OC1LR	Timer Output Compare 1 Low Register	00h	R/W
0058h		T16CHR	Timer Counter High Register	FFh	Read Only
0059h		T16CLR	Timer Counter Low Register	FCh	Read Only
005Ah		T16ACHR	Timer Alternate Counter High Register	FFh	Read Only
005Bh		T16ACLR	Timer Alternate Counter Low Register	FCh	Read Only
005Ch		T16IC2HR	Timer Input Capture 2 High Register	xxh	Read Only
005Dh		T16IC2LR	Timer Input Capture 2 Low Register	xxh	Read Only
005Eh		T16OC2HR	Timer Output Compare 2 High Register	80h	R/W
005Fh		T16OC2LR	Timer Output Compare 2 Low Register	00h	R/W
0060h	LINSCI2 (LIN Master)	SCI2SR	SCI2 Status Register	C0h	Read Only
0061h		SCI2DR	SCI2 Data Register	xxh	R/W
0062h		SCI2BRR	SCI2 Baud Rate Register	00h	R/W
0063h		SCI2CR1	SCI2 Control Register 1	xxh	R/W
0064h		SCI2CR2	SCI2 Control Register 2	00h	R/W
0065h		SCI2CR3	SCI2 Control Register 3	00h	R/W
0066h		SCI2ERPR	SCI2 Extended Receive Prescaler Register	00h	R/W
0067h		SCI2ETPR	SCI2 Extended Transmit Prescaler Register	00h	R/W

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 sensitiveness 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 45.)

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 34) 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 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.

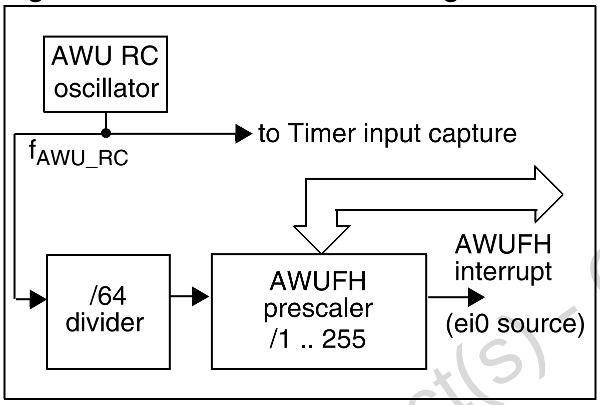
POWER SAVING MODES (Cont'd)

8.6 AUTO WAKE-UP FROM HALT MODE

Auto Wake-Up From Halt (AWUFH) mode is similar to Halt mode with the addition of an internal RC oscillator for wake-up. Compared to ACTIVE HALT mode, AWUFH has lower power consumption because the main clock is not kept running, but there is no accurate realtime clock available.

It is entered by executing the HALT instruction when the AWUEN bit in the AWUCSR register has been set and the OIE bit in the MCCSR register is cleared (see Section 10.2 on page 59 for more details).

Figure 29. AWUFH Mode Block Diagram



As soon as HALT mode is entered, and if the AWUEN bit has been set in the AWUCSR register, the AWU RC oscillator provides a clock signal (f_{AWU_RC}). Its frequency is divided by a fixed divider and a programmable prescaler controlled by the AWUPR register. The output of this prescaler provides the delay time. When the delay has elapsed the AWUF flag is set by hardware and an interrupt wakes up the MCU from Halt mode. At the same time the main oscillator is immediately turned on and a 256 or 4096 cycle delay is used to stabilize it. After this start-up delay, the CPU resumes operation by servicing the AWUFH interrupt. The AWU flag and its associated interrupt are cleared by software reading the AWUCSR register.

To compensate for any frequency dispersion of the AWU RC oscillator, it can be calibrated by measuring the clock frequency f_{AWU_RC} and then calculating the right prescaler value. Measurement mode is enabled by setting the AWUM bit in the AWUCSR register in Run mode. This connects f_{AWU_RC} to the ICAP1 input of the 16-bit timer, allowing the f_{AWU_RC} to be measured using the main oscillator clock as a reference timebase.

Similarities with Halt mode

The following AWUFH mode behavior is the same as normal Halt mode:

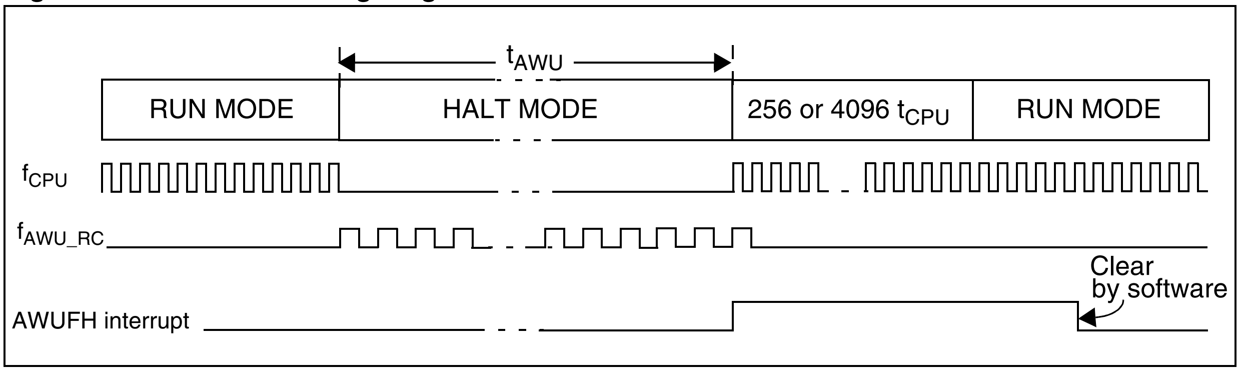
– The MCU can exit AWUFH mode by means of any interrupt with exit from Halt capability or a reset (see Section 8.4 "HALT MODE").

– When entering AWUFH 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 AWUFH mode, the main oscillator is turned off causing all internal processing to be stopped, including the operation of the on-chip peripherals. None of the peripherals are clocked except those which get their clock supply from another clock generator (such as an external or auxiliary oscillator like the AWU oscillator).

– The compatibility of Watchdog operation with AWUFH mode is configured by the WDGHALT option bit in the option byte. Depending on this setting, the HALT instruction when executed while the Watchdog system is enabled, can generate a Watchdog RESET.

Figure 30. AWUF Halt Timing Diagram



9 I/O PORTS

9.1 INTRODUCTION

The I/O ports offer different functional modes:
– transfer of data through digital inputs and outputs

and for specific pins:
– external interrupt generation
– alternate signal input/output for the on-chip peripherals.

An I/O port contains up to 8 pins. Each pin can be programmed independently as digital input (with or without interrupt generation) or digital output.

9.2 FUNCTIONAL DESCRIPTION

Each port has two main registers:

– Data Register (DR)

– Data Direction Register (DDR)

and one optional register:

– Option Register (OR)

Each I/O pin may be programmed using the corresponding register bits in the DDR and OR registers: Bit X corresponding to pin X of the port. The same correspondence is used for the DR register.

The following description takes into account the OR register, (for specific ports which do not provide this register refer to the I/O Port Implementation section). The generic I/O block diagram is shown in Figure 32

9.2.1 Input Modes

The input configuration is selected by clearing the corresponding DDR register bit.

In this case, reading the DR register returns the digital value applied to the external I/O pin.

Different input modes can be selected by software through the OR register.

Notes:
1. Writing the DR register modifies the latch value but does not affect the pin status.
2. When switching from input to output mode, the DR register has to be written first to drive the correct level on the pin as soon as the port is configured as an output.
3. Do not use read/modify/write instructions (BSET or BRES) to modify the DR register as this might corrupt the DR content for I/Os configured as input.

External interrupt function

When an I/O is configured as Input with Interrupt, an event on this I/O can generate an external interrupt request to the CPU.

Each pin can independently generate an interrupt request. The interrupt sensitivity is independently programmable using the sensitivity bits in the EICR register.

Each external interrupt vector is linked to a dedicated group of I/O port pins (see pinout description and interrupt section). If several input pins are selected simultaneously as interrupt sources, these are first detected according to the sensitivity bits in the EICR register and then logically ORed.

The external interrupts are hardware interrupts, which means that the request latch (not accessible directly by the application) is automatically cleared when the corresponding interrupt vector is fetched. To clear an unwanted pending interrupt by software, the sensitivity bits in the EICR register must be modified.

9.2.2 Output Modes

The output configuration is selected by setting the corresponding DDR register bit. In this case, writing the DR register applies this digital value to the I/O pin through the latch. Then reading the DR register returns the previously stored value.

Two different output modes can be selected by software through the OR register: Output push-pull and open-drain.

DR register value and output pin status:

DR	Push-pull	Open-drain
0	V_{SS}	Vss
1	V_{DD}	Floating

9.2.3 Alternate Functions

When an on-chip peripheral is configured to use a pin, the alternate function is automatically selected. This alternate function takes priority over the standard I/O programming.

When the signal is coming from an on-chip peripheral, the I/O pin is automatically configured in output mode (push-pull or open drain according to the peripheral).

When the signal is going to an on-chip peripheral, the I/O pin must be configured in input mode. In this case, the pin state is also digitally readable by addressing the DR register.

Note: Input pull-up configuration can cause unexpected value at the input of the alternate peripheral input. When an on-chip peripheral use a pin as input and output, this pin has to be configured in input floating mode.

I/O PORTS (Cont'd)

Table 15. I/O Port Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
Reset Value of all IO port registers		0	0	0	0	0	0	0	0
0000h	**PADR**	MSB							LSB
0001h	**PADDR**								
0002h	**PAOR**								
0003h	**PBDR**	MSB							LSB
0004h	**PBDDR**								
0005h	**PBOR**								
0006h	**PCDR**	MSB							LSB
0007h	**PCDDR**								
0008h	**PCOR**								
0009h	**PDDR**	MSB							LSB
000Ah	**PDDDR**								
000Bh	**PDOR**								
000Ch	**PEDR**	MSB							LSB
000Dh	**PEDDR**								
000Eh	**PEOR**								
000Fh	**PFDR**	MSB							LSB
0010h	**PFDDR**								
0011h	**PFOR**								

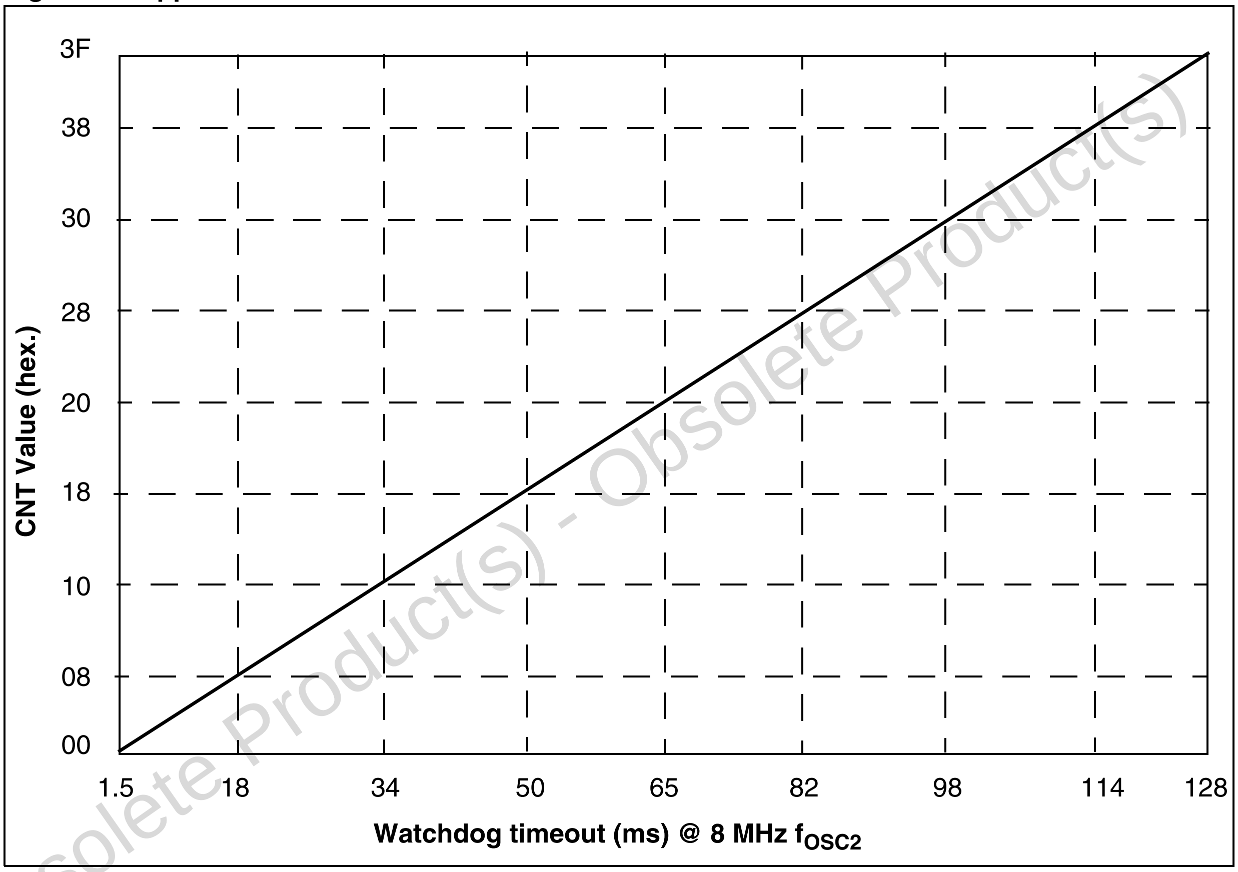
WINDOW WATCHDOG (Cont'd)

10.1.5 How to Program the Watchdog Timeout

Figure 2 shows the linear relationship between the 6-bit value to be loaded in the Watchdog Counter (CNT) and the resulting timeout duration in milliseconds. This can be used for a quick calculation without taking the timing variations into account. If more precision is needed, use the formulae in Figure 3.

Caution: When writing to the WDGCR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 35. Approximate Timeout Duration



PWM AUTO-RELOAD TIMER (Cont'd)

10.3.2 Functional Description

Counter

The free running 8-bit counter is fed by the output of the prescaler, and is incremented on every rising edge of the clock signal.

It is possible to read or write the contents of the counter on the fly by reading or writing the Counter Access register (ARTCAR).

When a counter overflow occurs, the counter is automatically reloaded with the contents of the ARTARR register (the prescaler is not affected).

Counter clock and prescaler

The counter clock frequency is given by:

$f_{COUNTER} = f_{INPUT} / 2^{CC[2:0]}$

The timer counter's input clock (f_{INPUT}) feeds the 7-bit programmable prescaler, which selects one of the eight available taps of the prescaler, as defined by CC[2:0] bits in the Control/Status Register (ARTCSR). Thus the division factor of the prescaler can be set to 2^n (where n = 0, 1,..7).

This f_{INPUT} frequency source is selected through the EXCL bit of the ARTCSR register and can be either the f_{CPU} or an external input frequency f_{EXT}.

The clock input to the counter is enabled by the TCE (Timer Counter Enable) bit in the ARTCSR register. When TCE is reset, the counter is stopped and the prescaler and counter contents are frozen. When TCE is set, the counter runs at the rate of the selected clock source.

Counter and Prescaler Initialization

After RESET, the counter and the prescaler are cleared and $f_{INPUT} = f_{CPU}$.

The counter can be initialized by:

– Writing to the ARTARR register and then setting the FCRL (Force Counter Re-Load) and the TCE (Timer Counter Enable) bits in the ARTCSR register.

– Writing to the ARTCAR counter access register,

In both cases the 7-bit prescaler is also cleared, whereupon counting will start from a known value.

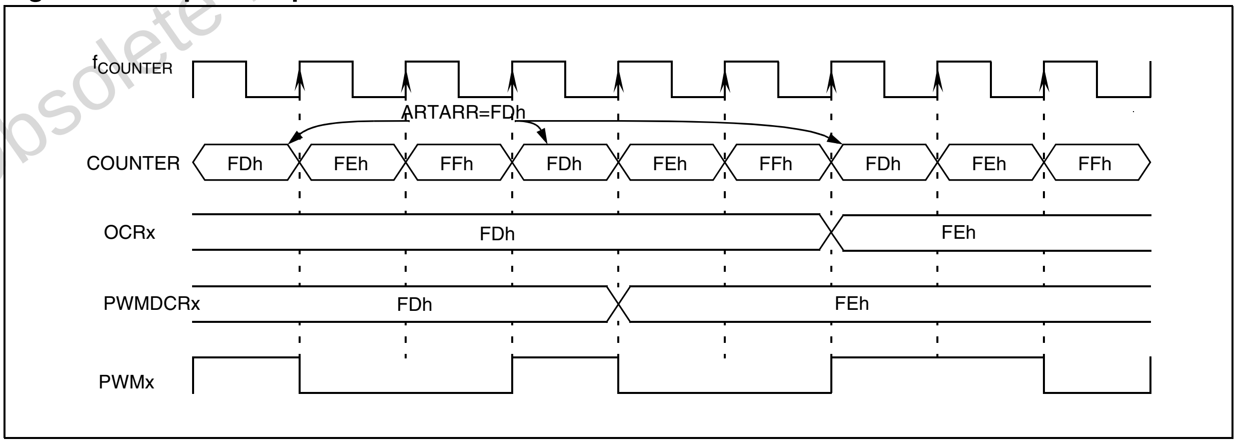
Direct access to the prescaler is not possible.

Output compare control

The timer compare function is based on four different comparisons with the counter (one for each PWMx output). Each comparison is made between the counter value and an output compare register (OCRx) value. This OCRx register can not be accessed directly, it is loaded from the duty cycle register (PWMDCRx) at each overflow of the counter.

This double buffering method avoids glitch generation when changing the duty cycle on the fly.

Figure 41. Output Compare Control



16-BIT TIMER (Cont'd)

10.4.3.4 Output Compare

In this section, the index, *i*, may be 1 or 2 because there are two output compare functions in the 16-bit timer.

This function can be used to control an output waveform or indicate when a period of time has elapsed.

When a match is found between the Output Compare register and the free running counter, the output compare function:

– Assigns pins with a programmable value if the OC*i*E bit is set

– Sets a flag in the status register

– Generates an interrupt if enabled

Two 16-bit registers Output Compare Register 1 (OC1R) and Output Compare Register 2 (OC2R) contain the value to be compared to the counter register each timer clock cycle.

	MS Byte	LS Byte
OC*i*R	OC*i*HR	OC*i*LR

These registers are readable and writable and are not affected by the timer hardware. A reset event changes the OC*i*R value to 8000h.

Timing resolution is one count of the free running counter: ($f_{CPU/CC[1:0]}$).

Procedure:

To use the output compare function, select the following in the CR2 register:

– Set the OC*i*E bit if an output is needed then the OCMP*i* pin is dedicated to the output compare *i* signal.

– Select the timer clock (CC[1:0]) (see Table 17 Clock Control Bits).

And select the following in the CR1 register:

– Select the OLVL*i* bit to applied to the OCMP*i* pins after the match occurs.

– Set the OCIE bit to generate an interrupt if it is needed.

When a match is found between OC*i*R register and CR register:

– OCF*i* bit is set.

– The OCMP*i* pin takes OLVL*i* bit value (OCMP*i* pin latch is forced low during reset).

– A timer interrupt is generated if the OCIE bit is set in the CR1 register and the I bit is cleared in the CC register (CC).

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

$$\Delta\ OCiR = \frac{\Delta t * f_{CPU}}{PRESC}$$

Where:

Δt	=	Output compare 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:

$$\Delta\ OCiR = \Delta t * f_{EXT}$$

Where:

Δt	=	Output compare period (in seconds)
f_{EXT}	=	External timer clock frequency (in hertz)

Clearing the output compare interrupt request (that is, clearing the OCF*i* bit) is done by:

1. Reading the SR register while the OCF*i* bit is set.

2. An access (read or write) to the OC*i*LR register.

The following procedure is recommended to prevent the OCF*i* bit from being set between the time it is read and the write to the OC*i*R register:

– Write to the OC*i*HR register (further compares are inhibited).

– Read the SR register (first step of the clearance of the OCF*i* bit, which may be already set).

– Write to the OC*i*LR register (enables the output compare function and clears the OCF*i* bit).

8-BIT TIMER (Cont'd)

10.5.4 Low Power Modes

Mode	Description
WAIT	No effect on 8-bit Timer. Timer interrupts cause the device to exit from WAIT mode.
HALT	8-bit Timer registers are frozen. In HALT mode, the counter stops counting until Halt mode is exited. Counting resumes from the previous count when the MCU is woken up by an interrupt with "exit from HALT mode" capability or from the counter reset value when the MCU is woken up by a RESET. If an input capture event occurs on the ICAP*i* pin, the input capture detection circuitry is armed. Consequently, when the MCU is woken up by an interrupt with "exit from HALT mode" capability, the ICF*i* bit is set, and the counter value present when exiting from HALT mode is captured into the IC*i*R register.

10.5.5 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
Input Capture 1 event/Counter reset in PWM mode	ICF1	ICIE	Yes	No
Input Capture 2 event	ICF2			
Output Compare 1 event (not available in PWM mode)	OCF1	OCIE		
Output Compare 2 event (not available in PWM mode)	OCF2			
Timer Overflow event	TOF	TOIE		

Note: The 8-bit Timer 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).

10.5.6 Summary of Timer modes

MODES	AVAILABLE RESOURCES			
	Input Capture 1	Input Capture 2	Output Compare 1	Output Compare 2
Input Capture (1 and/or 2)	Yes	Yes	Yes	Yes
Output Compare (1 and/or 2)				
One Pulse Mode	No	Not Recommended[1)]	No	Partially [2)]
PWM Mode		Not Recommended[3)]		No

1) See note 4 in "One Pulse Mode" on page 100
2) See note 5 in "One Pulse Mode" on page 100
3) See note 4 in "Pulse Width Modulation Mode" on page 102

10.6.8 Register Description

SPI CONTROL REGISTER (SPICR)

Read/Write

Reset Value: 0000 xxxx (0xh)

7							0
SPIE	SPE	SPR2	MSTR	CPOL	CPHA	SPR1	SPR0

Bit 7 = **SPIE** *Serial Peripheral Interrupt Enable*
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SPI interrupt is generated whenever an End of Transfer event, Master Mode Fault or Overrun error occurs (SPIF = 1, MODF = 1 or OVR = 1 in the SPICSR register)

Bit 6 = **SPE** *Serial Peripheral Output Enable*
This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$ = 0 (see Section 10.6.5.1 "Master Mode Fault (MODF)"). The SPE bit is cleared by reset, so the SPI peripheral is not initially connected to the external pins.
0: I/O pins free for general purpose I/O
1: SPI I/O pin alternate functions enabled

Bit 5 = **SPR2** *Divider Enable*
This bit is set and cleared by software and is cleared by reset. It is used with the SPR[1:0] bits to set the baud rate. Refer to Table 20 SPI Master Mode SCK Frequency.
0: Divider by 2 enabled
1: Divider by 2 disabled

Note: This bit has no effect in slave mode.

Bit 4 = **MSTR** *Master Mode*
This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$ = 0 (see Section 10.6.5.1 "Master Mode Fault (MODF)").
0: Slave mode
1: Master mode. The function of the SCK pin changes from an input to an output and the functions of the MISO and MOSI pins are reversed.

Bit 3 = **CPOL** *Clock Polarity*
This bit is set and cleared by software. This bit determines the idle state of the serial Clock. The CPOL bit affects both the master and slave modes.
0: SCK pin has a low level idle state
1: SCK pin has a high level idle state

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

Bit 2 = **CPHA** *Clock Phase*
This bit is set and cleared by software.
0: The first clock transition is the first data capture edge.
1: The second clock transition is the first capture edge.

Note: The slave must have the same CPOL and CPHA settings as the master.

Bits 1:0 = **SPR[1:0]** *Serial Clock Frequency*
These bits are set and cleared by software. Used with the SPR2 bit, they select the baud rate of the SPI serial clock SCK output by the SPI in master mode.

Note: These 2 bits have no effect in slave mode.

Table 21. SPI Master Mode SCK Frequency

Serial Clock	SPR2	SPR1	SPR0
$f_{CPU}/4$	1	0	0
$f_{CPU}/8$	0		
$f_{CPU}/16$			1
$f_{CPU}/32$	1	1	0
$f_{CPU}/64$	0		
$f_{CPU}/128$			1

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

CONTROL REGISTER 2 (SCICR2)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK

Bit 7 = **TIE** *Transmitter interrupt enable*.
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SCI interrupt is generated whenever TDRE = 1 in the SCISR register

Bit 6 = **TCIE** *Transmission complete interrupt enable*
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SCI interrupt is generated whenever TC = 1 in the SCISR register

Bit 5 = **RIE** *Receiver interrupt enable*.
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SCI interrupt is generated whenever OR = 1 or RDRF = 1 in the SCISR register

Bit 4 = **ILIE** *Idle line interrupt enable.*
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SCI interrupt is generated whenever IDLE = 1 in the SCISR register.

Bit 3 = **TE** *Transmitter enable.*
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled

Notes:

– During transmission, a "0" pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word.

– When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 = **RE** *Receiver enable.*
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 = **RWU** *Receiver wake-up.*
This bit determines if the SCI is in mute mode or not. It is set and cleared by software and can be cleared by hardware when a wake-up sequence is recognized.
0: Receiver in active mode
1: Receiver in mute mode

Notes:

– Before selecting Mute mode (by setting the RWU bit) the SCI must first receive a data byte, otherwise it cannot function in Mute mode with wake-up by Idle line detection.

– In Address Mark Detection Wake-Up configuration (WAKE bit = 1) the RWU bit cannot be modified by software while the RDRF bit is set.

Bit 0 = **SBK** *Send break.*
This bit set is used to send break characters. It is set and cleared by software.
0: No break character is transmitted
1: Break characters are transmitted

Note: If the SBK bit is set to "1" and then to "0", the transmitter sends a BREAK word at the end of the current word.

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

CONTROL REGISTER 3 (SCICR3)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
-	LINE	-	-	CLKEN	CPOL	CPHA	LBCL

Bit 7 = Reserved, must be kept cleared.

Bit 6 = **LINE** *LIN Mode Enable.*
This bit is set and cleared by software.
0: LIN Mode disabled
1: LIN Master mode enabled

The LIN Master mode enables the capability to send LIN Synch Breaks (13 low bits) using the SBK bit in the SCICR2 register

.In transmission, the LIN Synch Break low phase duration is shown as below:

LINE	M	Number of low bits sent during a LIN Synch Break
0	0	10
	1	11
1	0	13
	1	14

Bits 5:4 = Reserved, forced by hardware to 0.
These bits are not used.

Bit 3 = **CLKEN** *Clock Enable.*
This bit allows the user to enable the SCLK pin.
0: SLK pin disabled
1: SLK pin enabled

Bit 2 = **CPOL** *Clock Polarity.*
This bit allows the user to select the polarity of the clock output on the SCLK pin. It works in conjunction with the CPHA bit to produce the desired clock/data relationship (see Figure 92 and Figure 93).
0: Steady low value on SCLK pin outside transmission window.
1: Steady high value on SCLK pin outside transmission window.

Bit 1 = **CPHA** *Clock Phase.*
This bit allows the user to select the phase of the clock output on the SCLK pin. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see Figure 92 and Figure 93)
0: SCLK clock line activated in middle of data bit.
1: SCLK clock line activated at beginning of data bit.

Bit 0 = **LBCL** *Last bit clock pulse.*
This bit allows the user to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the SCLK pin.
0: The clock pulse of the last data bit is not output to the SCLK pin.
1: The clock pulse of the last data bit is output to the SCLK pin.

Note: The last bit is the 8th or 9th data bit transmitted depending on the 8 or 9 bit format selected by the M bit in the SCICR1 register.

Table 26. SCI clock on SCLK pin

Data format	M bit	LBCL bit	Number of clock pulses on SCLK
8 bit	0	0	7
		1	8
9 bit	1	0	8
		1	9

Note: These 3 bits (**CPOL**, **CPHA**, **LBCL**) should not be written while the transmitter is enabled.

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

EXTENDED RECEIVE PRESCALER DIVISION REGISTER (SCIERPR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
ERPR 7	ERPR 6	ERPR 5	ERPR 4	ERPR 3	ERPR 2	ERPR 1	ERPR 0

Bits 7:0 = **ERPR[7:0]** *8-bit Extended Receive Prescaler Register.*

The extended Baud Rate Generator is activated when a value other than 00h is stored in this register. The clock frequency from the 16 divider (see Figure 90) is divided by the binary factor set in the SCIERPR register (in the range 1 to 255).

The extended baud rate generator is not active after a reset.

EXTENDED TRANSMIT PRESCALER DIVISION REGISTER (SCIETPR)

Read/Write

Reset Value:0000 0000 (00h)

7							0
ETPR 7	ETPR 6	ETPR 5	ETPR 4	ETPR 3	ETPR 2	ETPR 1	ETPR 0

Bits 7:0 = **ETPR[7:0]** *8-bit Extended Transmit Prescaler Register.*

The extended Baud Rate Generator is activated when a value other than 00h is stored in this register. The clock frequency from the 16 divider (see Figure 90) is divided by the binary factor set in the SCIETPR register (in the range 1 to 255).

The extended baud rate generator is not active after a reset.

Table 27. Baud Rate Selection

Symbol	Parameter	Conditions			Standard	Baud Rate	Unit
		f_{CPU}	Accuracy vs. Standard	Prescaler			
f_{Tx} f_{Rx}	Communication frequency	8 MHz	~0.16%	Conventional Mode			Hz
				TR (or RR) = 128, PR = 13	300	~300.48	
				TR (or RR) = 32, PR = 13	1200	~1201.92	
				TR (or RR) = 16, PR =13	2400	~2403.84	
				TR (or RR) = 8, PR = 13	4800	~4807.69	
				TR (or RR) = 4, PR = 13	9600	~9615.38	
				TR (or RR) = 16, PR = 3	10400	~10416.67	
				TR (or RR) = 2, PR = 13	19200	~19230.77	
				TR (or RR) = 1, PR =13	38400	~38461.54	
			~0.79%	Extended Mode ETPR (or ERPR) = 35, TR (or RR) = 1, PR = 1	14400	~14285.71	

beCAN CONTROLLER (Cont'd)

Filter Bank Scale and Mode Configuration

The filter banks are configured by means of the corresponding CFCRx register. To configure a filter bank this must be deactivated by clearing the FACT bit in the CFCR register. The filter scale is configured by means of the FSC[1:0] bits in the corresponding CFCR register, refer to Figure 9. Filter Bank Scale Configuration - Register Organisation. The **identifier list** or **identifier mask** mode for the corresponding Mask/Identifier registers is configured by means of the FMCLx and FMCHx bits in the CFMR register. The FMCLx bit defines the mode for the two least significant bytes, and the FMCHx bit the mode for the two most significant bytes of filter bank x. **Examples**:

– If filter bank 1 is configured as two 16-bit filters, then the FMCL1 bit defines the mode of the CF1R2 and CF1R3 registers and the FMCH1 bit defines the mode of the CF1R6 and CF1R7 registers.

– If filter bank 2 is configured as four 8-bit filters, then the FMCL2 bit defines the mode of the CF2R1 and CF2R3 registers and the FMCH2 bit defines the mode of the CF2R5 and CF2R7 registers.

Note: In 32-bit configuration, the FMCLx and FMCHx bits must have the same value to ensure that the four Mask/Identifier registers are in the same mode.

To filter a group of identifiers, configure the Mask/Identifier registers in mask mode.

To select single identifiers, configure the Mask/Identifier registers in identifier list mode.

Filters not used by the application should be left deactivated.

Filter Match Index

Once a message has been received in the FIFO it is available to the application. Typically application data are copied into RAM locations. To copy the data to the right location the application has to identify the data by means of the identifier. To avoid this and to ease the access to the RAM locations, the CAN controller provides a Filter Match Index.

This index is stored in the mailbox together with the message according to the filter priority rules. Thus each received message has its associated Filter Match Index.

The Filter Match Index can be used in two ways:

– Compare the Filter Match Index with a list of expected values.

– Use the Filter Match Index as an index on an array to access the data destination location.

For non-masked filters, the software no longer has to compare the identifier.

If the filter is masked the software reduces the comparison to the masked bits only.

Filter Priority Rules

Depending on the filter combination it may occur that an identifier passes successfully through several filters. In this case the filter match value stored in the receive mailbox is chosen according to the following rules:

– A filter in identifier list mode prevails on an filter in mask mode.

– A filter with full identifier coverage prevails over filters covering part of the identifier, e.g. 16-bit filters prevail over 8-bit filters.

– Filters configured in the same mode and with identical coverage are prioritized by filter number and register number. The lower the number the higher the priority.

beCAN CONTROLLER (Cont'd)

– The **FIFO interrupt** can be generated by the following events:

  – Reception of a new message, FMP bits in the CRFR0 register incremented.

  – FIFO0 full condition, FULL bit in the CRFR0 register set.

  – FIFO0 overrun condition, FOVR bit in the CRFR0 register set.

– The **transmit, error and status change interrupt** can be generated by the following events:

  – Transmit mailbox 0 becomes empty, RQCP0 bit in the CTSR register set.

  – Transmit mailbox 1 becomes empty, RQCP1 bit in the CTSR register set.

  – Error condition, for more details on error conditions please refer to the CAN Error Status register (CESR).

  – Wake-up condition, SOF monitored on the CAN Rx signal.

10.9.6 Register Access Protection

Erroneous access to certain configuration registers can cause the hardware to temporarily disturb the whole CAN network. Therefore the following registers can be modified by software only while the hardware is in initialization mode:

CBTR0, CBTR1, CFCR0, CFCR1, CFMR and CDGR registers.

Although the transmission of incorrect data will not cause problems at the CAN network level, it can severely disturb the application. A transmit mailbox can be only modified by software while it is in empty state (refer to Figure 7. Transmit Mailbox States).

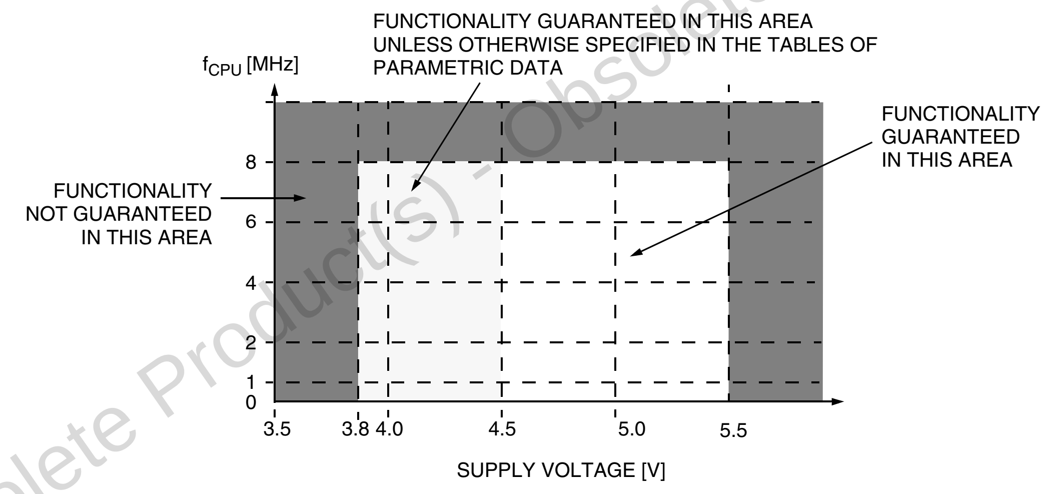
The filters must be deactivated before their value can be modified by software. The modification of the filter configuration (scale or mode) can be done by software only in initialization mode.

12.3 OPERATING CONDITIONS

12.3.1 General Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
f_{CPU}	Internal clock frequency		0	8	MHz
V_{DD}	Extended Operating voltage	No Flash Write/Erase. Analog parameters not guaranteed.	3.8	4.5	V
	Standard Operating Voltage		4.5	5.5	
	Operating Voltage for Flash Write/Erase	V_{PP} = 11.4 to 12.6V	4.5	5.5	
T_A	Ambient temperature range	1 Suffix Version	0	70	°C
		5 Suffix Version	-10	85	
		6 Suffix Version	-40	85	
		7 Suffix Version		105	
		3 Suffix Version		125	

Figure 119. f_{CPU} Maximum vs V_{DD}



Note: It is mandatory to connect all available V_{DD} and V_{DDA} pins to the supply voltage and all V_{SS} and V_{SSA} pins to ground.

SUPPLY CURRENT CHARACTERISTICS (Cont'd)

12.4.1 Supply and Clock Managers

The previous current consumption specified for the ST7 functional operating modes over temperature range does not take into account the clock source current consumption. To obtain the total device consumption, the two current values must be added (except for HALT mode).

Symbol	Parameter	Conditions	Typ	Max[1]	Unit
$I_{DD(RES)}$	Supply current of resonator oscillator[2)3)]		See Section 12.5.3 on page 227		μA
$I_{DD(PLL)}$	PLL supply current	V_{DD} = 5V	360		
$I_{DD(LVD)}$	LVD supply current	HALT mode, V_{DD} = 5V	150	300	

Notes:
1. Data based on characterization results, not tested in production.
2. Data based on characterization results done with the external components specified in Section 12.5.3, not tested in production.
3. As the oscillator is based on a current source, the consumption does not depend on the voltage.

CLOCK CHARACTERISTICS (Cont'd)

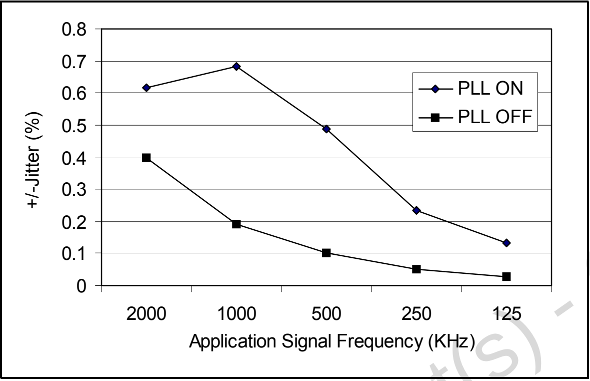
12.5.4 PLL Characteristics

Operating conditions: V_{DD} 3.8 to 5.5V @ T_A 0 to 70°C[1] or V_{DD} 4.5 to 5.5V @ T_A -40 to 125°C

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{DD(PLL)}$	PLL Voltage Range	T_A = 0 to +70°C	3.8		5.5	
		T_A = -40 to +125°C	4.5			
f_{OSC}	PLL input frequency range		2		4	MHz
$\Delta f_{CPU}/f_{CPU}$	PLL jitter[1]	f_{OSC} = 4 MHz, V_{DD} = 4.5 to 5.5V		Note 2		%
		f_{OSC} = 2 MHz, V_{DD} = 4.5 to 5.5V				

Notes:
1. Data characterized but not tested.
2. Under characterization

Figure 123. PLL Jitter vs Signal Frequency[1]



Notes:
1. Measurement conditions: f_{CPU} = 4 MHz, T_A = 25°C

The user must take the PLL jitter into account in the application (for example in serial communication or sampling of high frequency signals). The PLL jitter is a periodic effect, which is integrated over several CPU cycles. Therefore, the longer the period of the application signal, the less it is impacted by the PLL jitter.

Figure 123 shows the PLL jitter integrated on application signals in the range 125 kHz to 2 MHz. At frequencies of less than 125 kHz, the jitter is negligible.

COMMUNICATION INTERFACE CHARACTERISTICS (Cont'd)

Figure 139. SPI Slave Timing Diagram with CPHA = 1[1)]

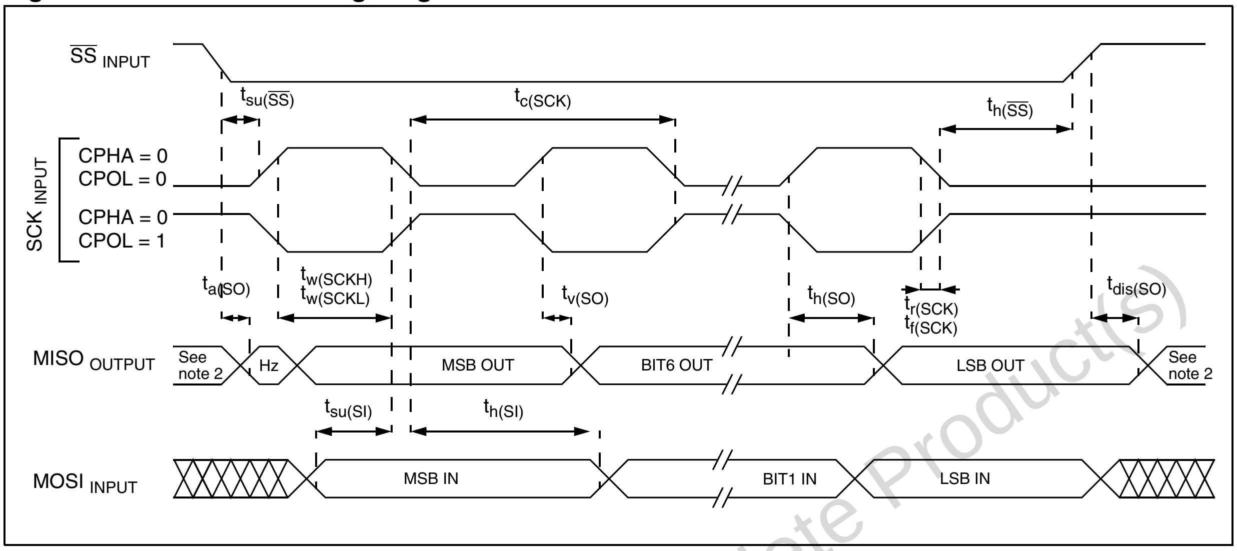
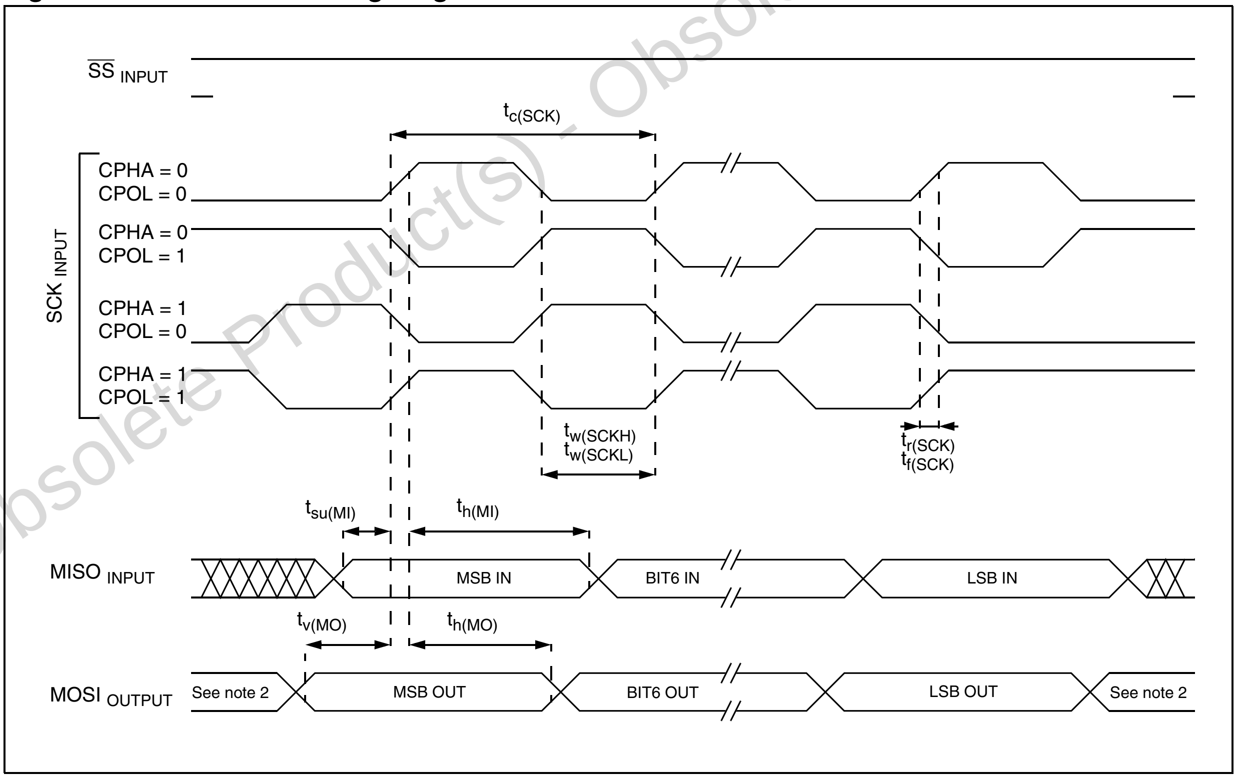


Figure 140. SPI Master Timing Diagram[1)]



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.