



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	CANbus, LINbusSCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	24
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 6x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	32-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st72f561k9tctr

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 59](#) 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 34](#)) 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 53](#) for more details).

Figure 25. HALT Timing Overview

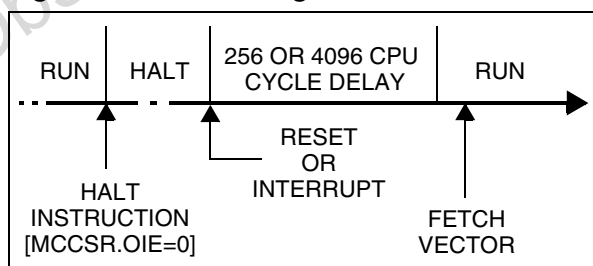
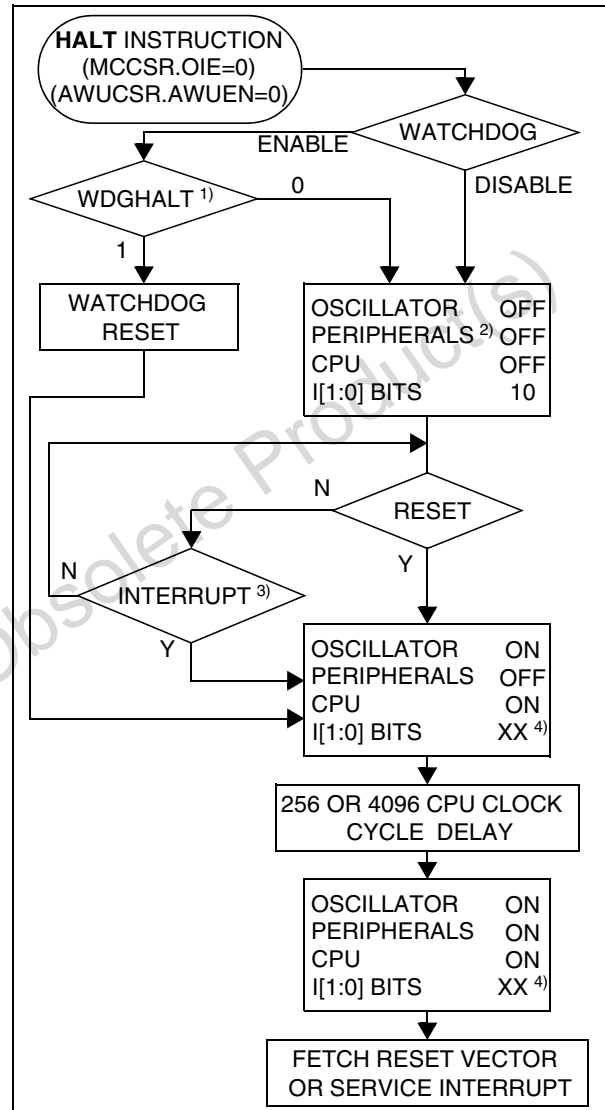


Figure 26. HALT Mode Flow-chart

**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 34](#) 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.

MAIN CLOCK CONTROLLER WITH REAL TIME CLOCK (Cont'd)

10.2.4 Low Power Modes

Mode	Description
WAIT	No effect on MCC/RTC peripheral. MCC/RTC interrupt cause the device to exit from WAIT mode.
ACTIVE HALT	No effect on MCC/RTC counter (OIE bit is set), the registers are frozen. MCC/RTC interrupt cause the device to exit from ACTIVE HALT mode.
HALT and AWUF HALT	MCC/RTC counter and registers are frozen. MCC/RTC operation resumes when the MCU is woken up by an interrupt with "exit from HALT" capability.

10.2.5 Interrupts

The MCC/RTC interrupt event generates an interrupt if the OIE bit of the MCCR register is set and the interrupt mask in the CC register is not active (RIM instruction).

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
Time base overflow event	OIF	OIE	Yes	No ¹⁾

Note:

The MCC/RTC interrupt wakes up the MCU from ACTIVE HALT mode, not from HALT or AWUF HALT mode.

10.2.6 Register Description

MCC CONTROL/STATUS REGISTER (MCCR)

Read/Write

Reset Value: 0000 0000 (00h)

7	0						
MCO	CP1	CP0	SMS	TB1	TB0	OIE	OIF

Bit 7 = **MCO** Main clock out selection

This bit enables the MCO alternate function on the corresponding I/O port. It is set and cleared by software.

0: MCO alternate function disabled (I/O pin free for general-purpose I/O)

1: MCO alternate function enabled (f_{OSC2} on I/O port)

Bits 6:5 = **CP[1:0]** CPU clock prescaler

These bits select the CPU clock prescaler which is applied in the different slow modes. Their action is conditioned by the setting of the SMS bit. These two bits are set and cleared by software

f_{CPU} in SLOW mode	CP1	CP0
$f_{OSC2} / 2$	0	0
$f_{OSC2} / 4$	0	1
$f_{OSC2} / 8$	1	0
$f_{OSC2} / 16$	1	1

Bit 4 = **SMS** Slow mode select

This bit is set and cleared by software.

0: Normal mode. $f_{CPU} = f_{OSC2}$

1: Slow mode. f_{CPU} is given by CP1, CP0

See [Section 8.2 "SLOW MODE"](#) and [Section 10.2 "MAIN CLOCK CONTROLLER WITH REAL TIME CLOCK MCC/RTC"](#) for more details.

Bits 3:2 = **TB[1:0]** Time base control

These bits select the programmable divider time base. They are set and cleared by software.

Counter Prescaler	Time Base		TB1	TB0
	$f_{OSC2} = 4 \text{ MHz}$	$f_{OSC2} = 8 \text{ MHz}$		
16000	4ms	2ms	0	0
32000	8ms	4ms	0	1
80000	20ms	10ms	1	0
200000	50ms	25ms	1	1

A modification of the time base is taken into account at the end of the current period (previously set) to avoid an unwanted time shift. This allows to use this time base as a real time clock.

Bit 1 = **OIE** Oscillator interrupt enable

This bit set and cleared by software.

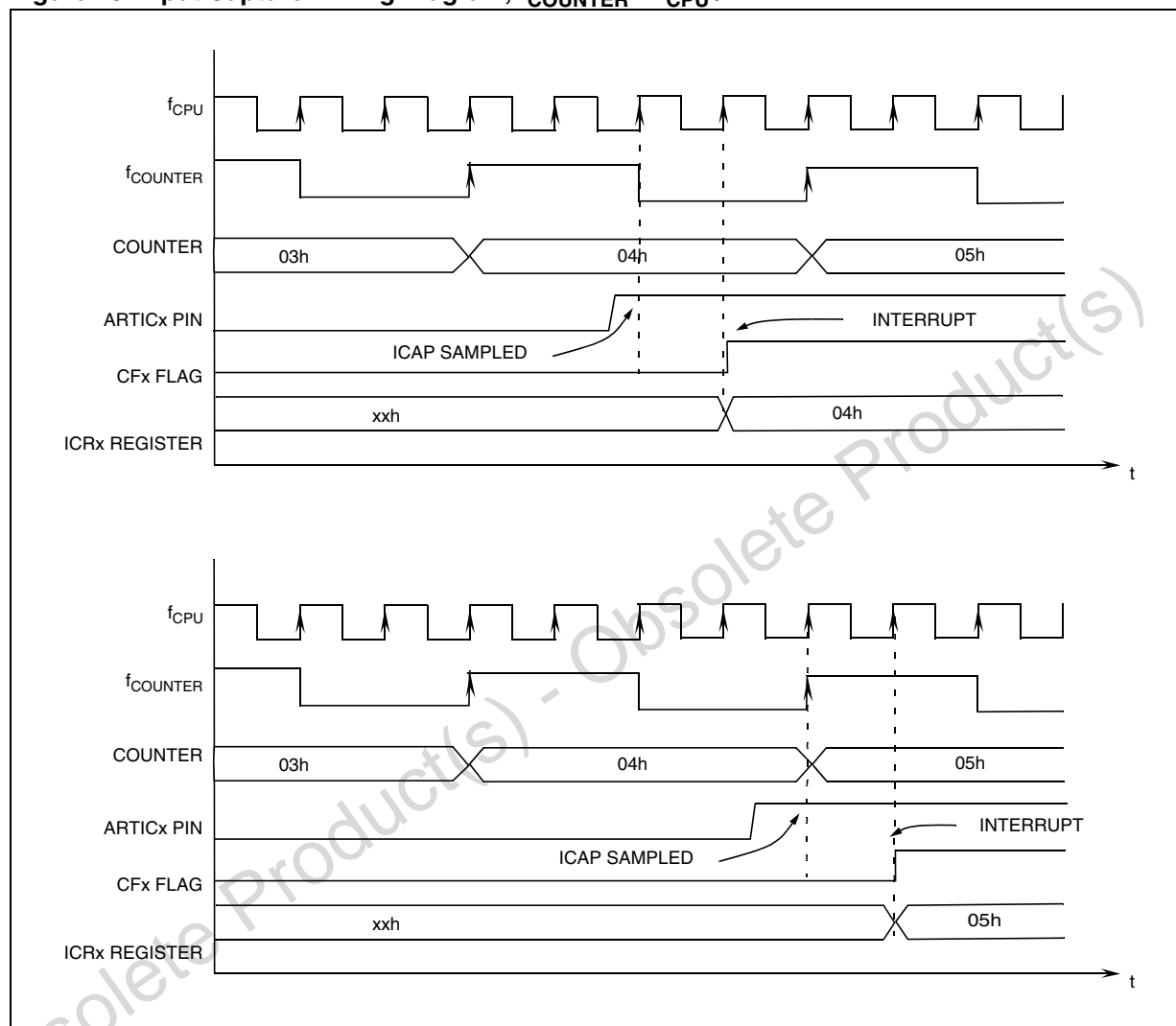
0: Oscillator interrupt disabled

1: Oscillator interrupt enabled

This interrupt can be used to exit from ACTIVE HALT mode.

When this bit is set, calling the ST7 software HALT instruction enters the ACTIVE HALT power saving mode.

PWM AUTO-RELOAD TIMER (Cont'd)

Figure 46. input Capture Timing Diagram, $f_{\text{COUNTER}} = f_{\text{CPU}} / 4$ 

16-BIT TIMER (Cont'd)

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

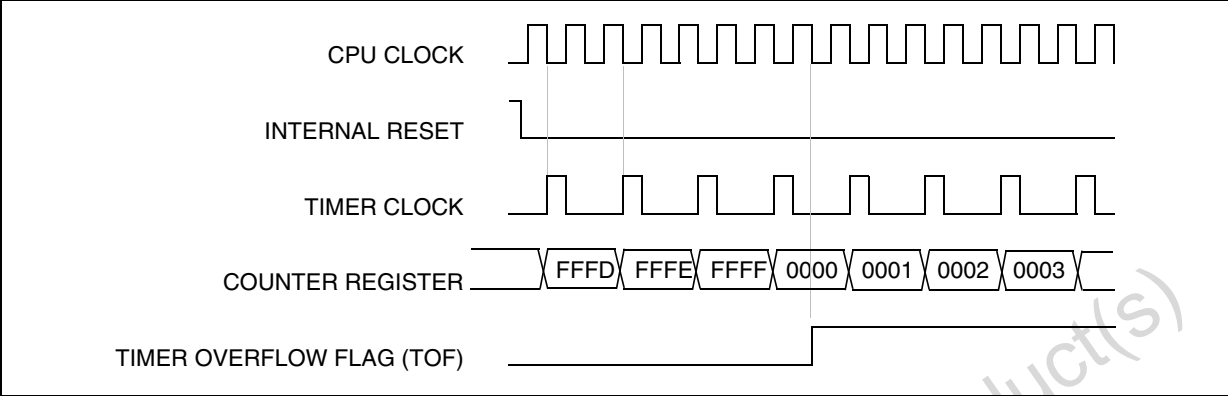


Figure 50. Counter Timing Diagram, Internal Clock Divided by 4

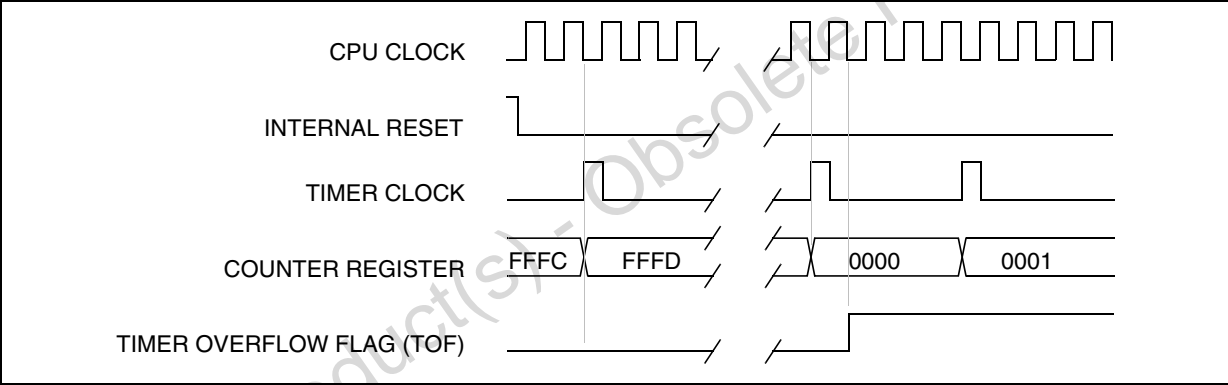
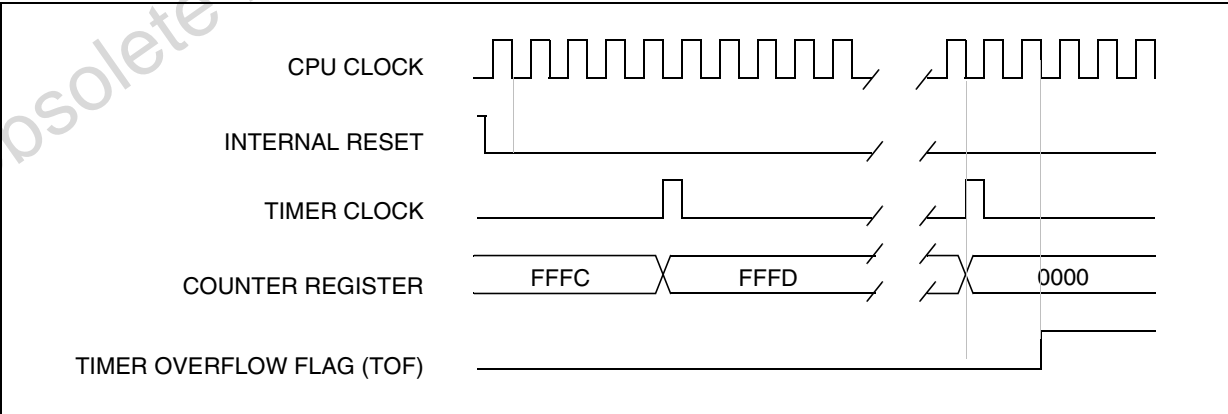


Figure 51. Counter Timing Diagram, Internal Clock Divided By 8



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

8-BIT TIMER (Cont'd)**10.5.3.2 Input Capture**

In this section, the index, i , may be 1 or 2 because there are two input capture functions in the 8-bit timer.

The two 8-bit input capture registers (IC1R and IC2R) are used to latch the value of the free running counter after a transition is detected on the ICAP i pin (see [Figure 63](#)).

IC/R register is a read-only register.

The active transition is software programmable through the IEDG i bit of Control Registers (CR i).

Timing resolution is one count of the free running counter (see [Table 19 Clock Control Bits](#)).

Procedure:

To use the input capture function select the following in the CR2 register:

- Select the timer clock (CC[1:0]) (see [Table 19 Clock Control Bits](#)).
- Select the edge of the active transition on the ICAP2 pin with the IEDG2 bit (the ICAP2 pin must be configured as floating input or input with pull-up without interrupt if this configuration is available).

And select the following in the CR1 register:

- Set the ICIE bit to generate an interrupt after an input capture coming from either the ICAP1 pin or the ICAP2 pin
- Select the edge of the active transition on the ICAP1 pin with the IEDG1 bit (the ICAP1 pin must be configured as floating input or input with pull-up without interrupt if this configuration is available).

When an input capture occurs:

- ICF i bit is set.
- The IC/R register contains the value of the free running counter on the active transition on the ICAP i pin (see [Figure 64](#)).
- A timer interrupt is generated if the ICIE bit is set and the interrupt mask is cleared in the CC register. Otherwise, the interrupt remains pending until both conditions become true.

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

1. Reading the SR register while the ICF i bit is set.
2. An access (read or write) to the IC/R register.

Notes:

1. The IC/R register contains the free running counter value which corresponds to the most recent input capture.
 2. The two input capture functions can be used together even if the timer also uses the two output compare functions.
 3. Once the ICIE bit is set both input capture features may trigger interrupt requests. If only one is needed in the application, the interrupt routine software needs to discard the unwanted capture interrupt. This can be done by checking the ICF1 and ICF2 flags and resetting them both.
 4. In One pulse Mode and PWM mode only Input Capture 2 can be used.
 5. The alternate inputs (ICAP1 and ICAP2) are always directly connected to the timer. So any transitions on these pins activates the input capture function.
- Moreover if one of the ICAP i pins is configured as an input and the second one as an output, an interrupt can be generated if the user toggles the output pin and if the ICIE bit is set.
6. The TOF bit can be used with interrupt generation in order to measure events that go beyond the timer range (FFh).

8-BIT TIMER (Cont'd)**CONTROL/STATUS REGISTER (CSR)**

Read Only (except bit 2 R/W)

Reset Value: 0000 0000 (00h)

7							0
ICF1	OCF1	TOF	ICF2	OCF2	TIMD	0	0

Bit 7 = **ICF1** *Input Capture Flag 1.*

0: No input capture (reset value).

1: An input capture has occurred on the ICAP1 pin or the counter has reached the OC2R value in PWM mode. To clear this bit, first read the SR register, then read or write the IC1R register.

Bit 6 = **OCF1** *Output Compare Flag 1.*

0: No match (reset value).

1: The content of the free running counter has matched the content of the OC1R register. To clear this bit, first read the SR register, then read or write the OC1R register.

Bit 5 = **TOF** *Timer Overflow Flag.*

0: No timer overflow (reset value).

1: The free running counter rolled over from FFh to 00h. To clear this bit, first read the SR register, then read or write the CTR register.

Note: Reading or writing the ACTR register does not clear TOF.Bit 4 = **ICF2** *Input Capture Flag 2.*

0: No input capture (reset value).

1: An input capture has occurred on the ICAP2 pin. To clear this bit, first read the SR register, then read or write the IC2R register.

Bit 3 = **OCF2** *Output Compare Flag 2.*

0: No match (reset value).

1: The content of the free running counter has matched the content of the OC2R register. To clear this bit, first read the SR register, then read or write the OC2R register.

Bit 2 = **TIMD** *Timer disable.*

This bit is set and cleared by software. When set, it freezes the timer prescaler and counter and disabled the output functions (OCMP1 and OCMP2 pins) to reduce power consumption. Access to the timer registers is still available, allowing the timer configuration to be changed, or the counter reset, while it is disabled.

0: Timer enabled

1: Timer prescaler, counter and outputs disabled

Bits 1:0 = Reserved, must be kept cleared.

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)**10.6.5 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"](#).

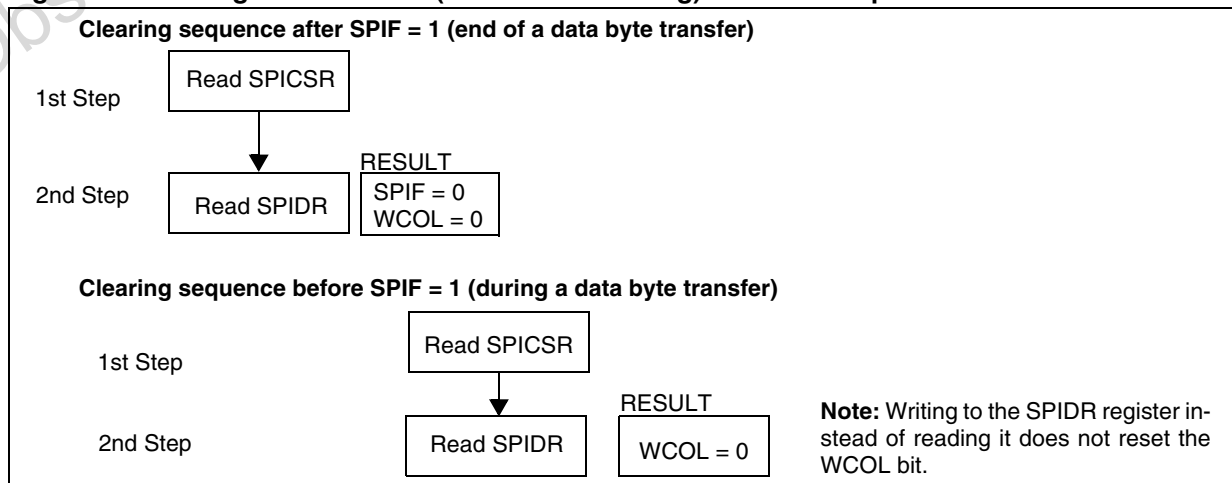
Note: A "read collision" will never occur since the received data byte is placed in a buffer in which access is always synchronous with the CPU operation.

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).

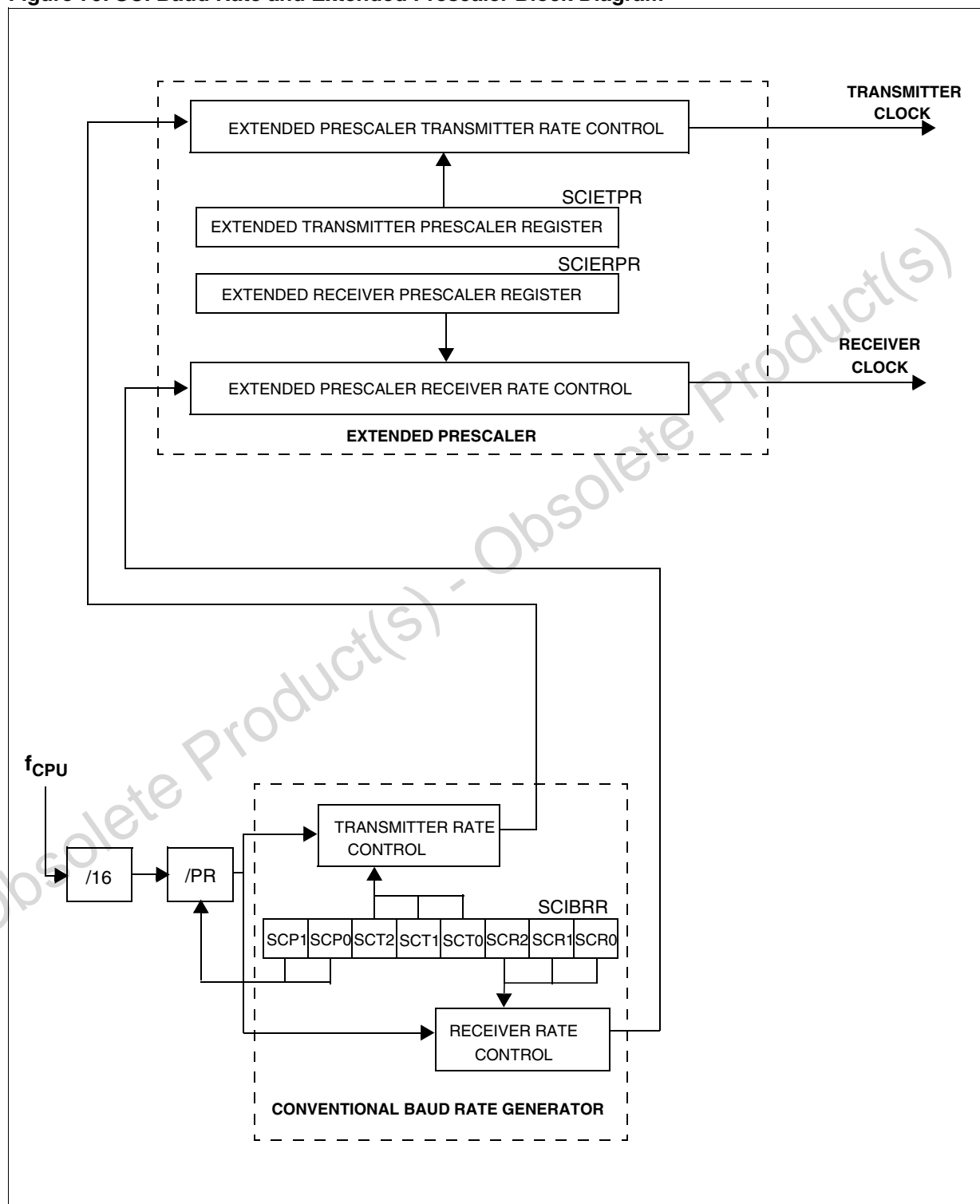
Clearing the WCOL bit is done through a software sequence (see [Figure 75](#)).

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



LINSI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

Figure 79. SCI Baud Rate and Extended Prescaler Block Diagram



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

If LHE bit is set due to this error during Fields other than LIN Synch Field or if LASE bit is reset then the current received Header is discarded and the SCI searches for a new Break Field.

Note on LIN Header Time-out Limit

According to the LIN specification, the maximum length of a LIN Header which does not cause a timeout is equal to $1.4 * (34 + 1) = 49 T_{\text{BIT_MASTER}}$.

$T_{\text{BIT_MASTER}}$ refers to the master baud rate.

When checking this timeout, the slave node is desynchronized for the reception of the LIN Break and Synch fields. Consequently, a margin must be allowed, taking into account the worst case: This occurs when the LIN identifier lasts exactly $10 T_{\text{BIT_MASTER}}$ periods. In this case, the LIN Break and Synch fields last $49 - 10 = 39 T_{\text{BIT_MASTER}}$ periods.

Assuming the slave measures these first 39 bits with a desynchronized clock of 15.5%. This leads to a maximum allowed Header Length of:

$$39 \times (1/0.845) T_{\text{BIT_MASTER}} + 10 T_{\text{BIT_MASTER}} = 56.15 T_{\text{BIT_SLAVE}}$$

A margin is provided so that the time-out occurs when the header length is greater than $57 T_{\text{BIT_SLAVE}}$ periods. If it is less than or equal to $57 T_{\text{BIT_SLAVE}}$ periods, then no timeout occurs.

LIN Header Length

Even if no timeout occurs on the LIN Header, it is possible to have access to the effective LIN header Length (T_{HEADER}) through the LHL register. This allows monitoring at software level the $T_{\text{FRAME_MAX}}$ condition given by the LIN protocol.

This feature is only available when LHDM bit = 1 or when LASE bit = 1.

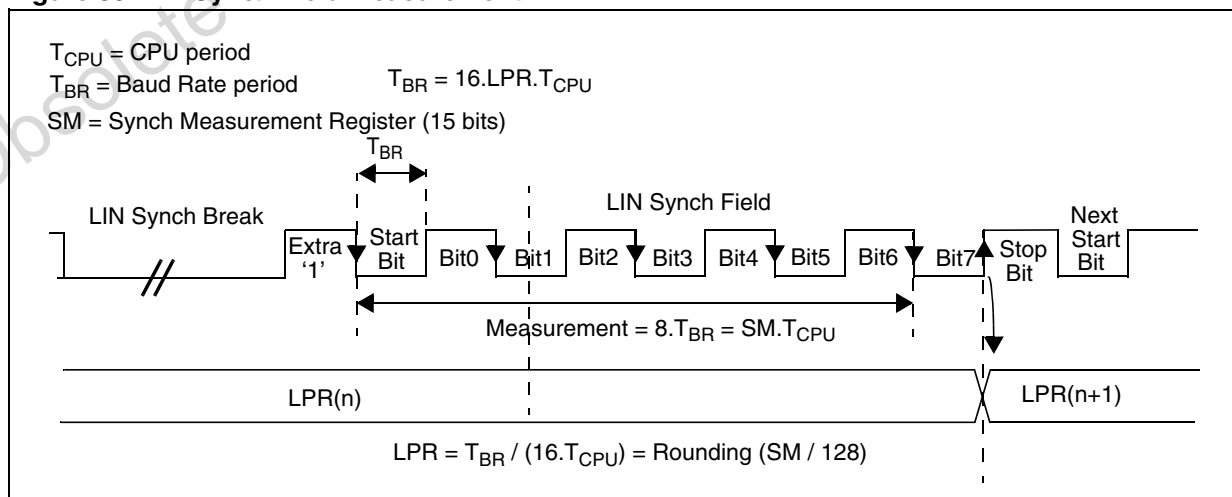
Mute Mode and Errors

In mute mode when LHDM bit = 1, if an LHE error occurs during the analysis of the LIN Synch Field or if a LIN Header Time-out occurs then the LHE bit is set but it does not wake up from mute mode. In this case, the current header analysis is discarded. If needed, the software has to reset LSF bit. Then the SCI searches for a new LIN header.

In mute mode, if a framing error occurs on a data (which is not a break), it is discarded and the FE bit is not set.

When LHDM bit = 1, any LIN header which respects the following conditions causes a wake-up from mute mode:

- A valid LIN Break Field (at least 11 dominant bits followed by a recessive bit)
- A valid LIN Synch Field (without deviation error)
- A LIN Identifier Field without framing error. Note that a LIN parity error on the LIN Identifier Field does not prevent wake-up from mute mode.
- No LIN Header Time-out should occur during Header reception.

Figure 83. LIN Synch Field Measurement

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

10.8.7 SCI Synchronous Transmission

The SCI transmitter allows the user to control a one way synchronous serial transmission. The SCLK pin is the output of the SCI transmitter clock. No clock pulses are sent to the SCLK pin during start bit and stop bit. Depending on the state of the LBCL bit in the SCICR3 register, clock pulses are or are not be generated during the last valid data bit (address mark). The CPOL bit in the SCICR3 register allows the user to select the clock polarity, and the CPHA bit in the SCICR3 register allows the user to select the phase of the external clock (see [Figure 91](#), [Figure 92](#) and [Figure 93](#)).

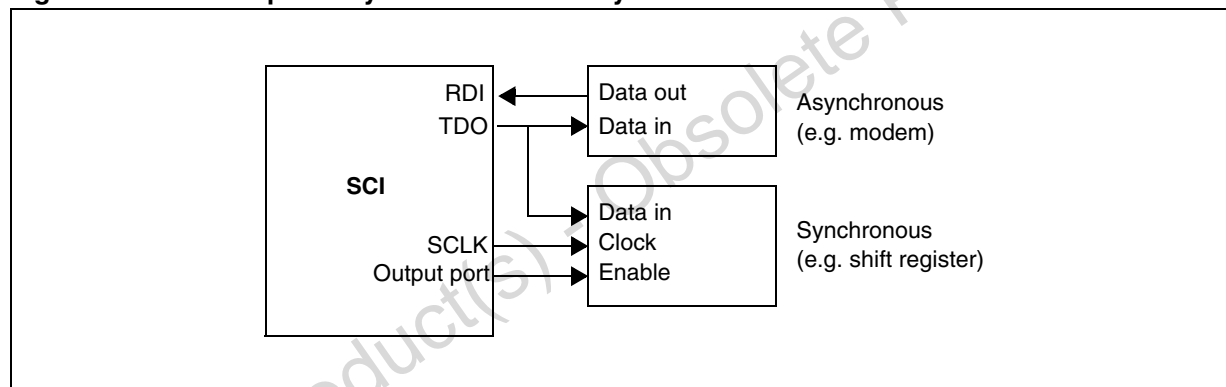
During idle, preamble and send break, the external SCLK clock is not activated.

These options allow the user to serially control peripherals which consist of shift registers, without losing any functions of the SCI transmitter which can still talk to other SCI receivers. These options do not affect the SCI receiver which is independent from the transmitter.

Note: The SCLK pin works in conjunction with the TDO pin. When the SCI transmitter is disabled (TE and RE = 0), the SCLK and TDO pins go into high impedance state.

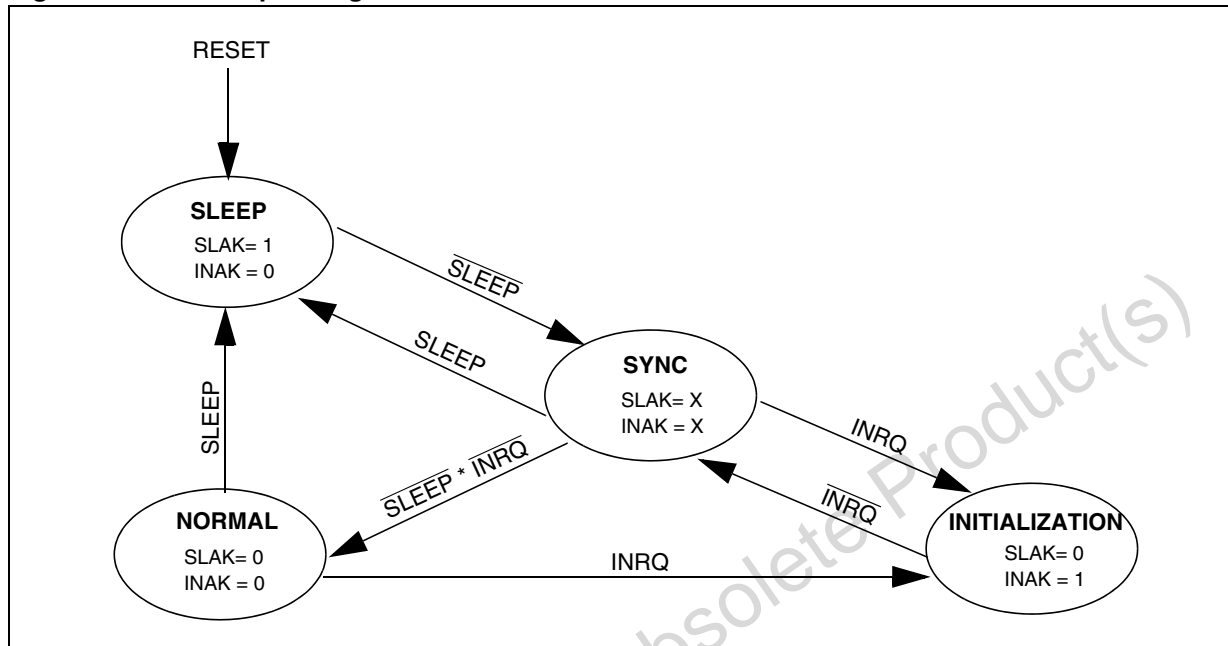
Note: The LBCL, CPOL and CPHA bits have to be selected before enabling the transmitter to ensure that the clock pulses function correctly. These bits should not be changed while the transmitter is enabled.

Figure 91. SCI Example of Synchronous and Asynchronous Transmission



beCAN CONTROLLER (Cont'd)

Figure 96. beCAN Operating Modes



10.9.3 Operating Modes

The beCAN has three main operating modes: Initialization, Normal and Sleep. After a hardware reset, beCAN is in Sleep mode to reduce power consumption. The software requests beCAN to enter Initialization or Sleep mode by setting the INRQ or SLEEP bits in the CMCR register. Once the mode has been entered, beCAN confirms it by setting the INAK or SLAK bits in the CMSR register. When neither INAK nor SLAK are set, beCAN is in Normal mode. Before entering Normal mode beCAN always has to **synchronize** on the CAN bus. To synchronize, beCAN waits until the CAN bus is idle, this means 11 consecutive recessive bits have been monitored on CANRX.

10.9.3.1 Initialization Mode

The software initialization can be done while the hardware is in Initialization mode. To enter this mode the software sets the INRQ bit in the CMCR register and waits until the hardware has confirmed the request by setting the INAK bit in the CMSR register.

To leave Initialization mode, the software clears the INQR bit. beCAN has left Initialization mode once the INAK bit has been cleared by hardware.

While in Initialization mode, all message transfers to and from the CAN bus are stopped and the sta-

tus of the CAN bus output CANTX is recessive (high).

Entering Initialization Mode does not change any of the configuration registers.

To initialize the CAN Controller, software has to set up the Bit Timing registers and the filter banks. If a filter bank is not used, it is recommended to leave it non active (leave the corresponding FACT bit cleared).

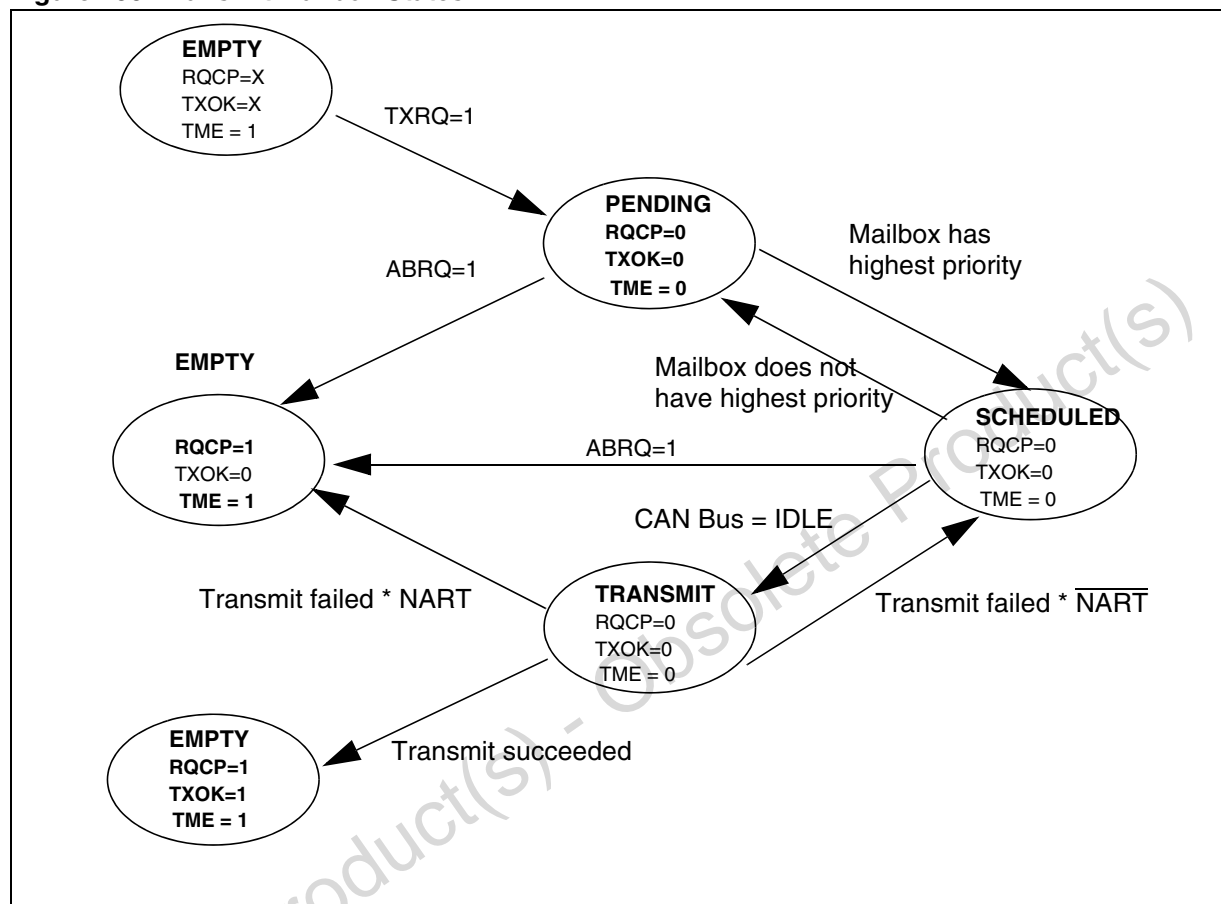
10.9.3.2 Normal Mode

Once the initialization has been done, the software must request the hardware to enter Normal mode, to synchronize on the CAN bus and start reception and transmission. Entering Normal mode is done by clearing the INRQ bit in the CMCR register and waiting until the hardware has confirmed the request by clearing the INAK bit in the CMSR register. Afterwards, the beCAN synchronizes with the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (\equiv Bus Idle) before it can take part in bus activities and start message transfer.

The initialization of the filter values is independent from Initialization mode but must be done while the filter bank is not active (corresponding FACTx bit cleared). The filter bank scale and mode configuration must be configured in initialization mode.

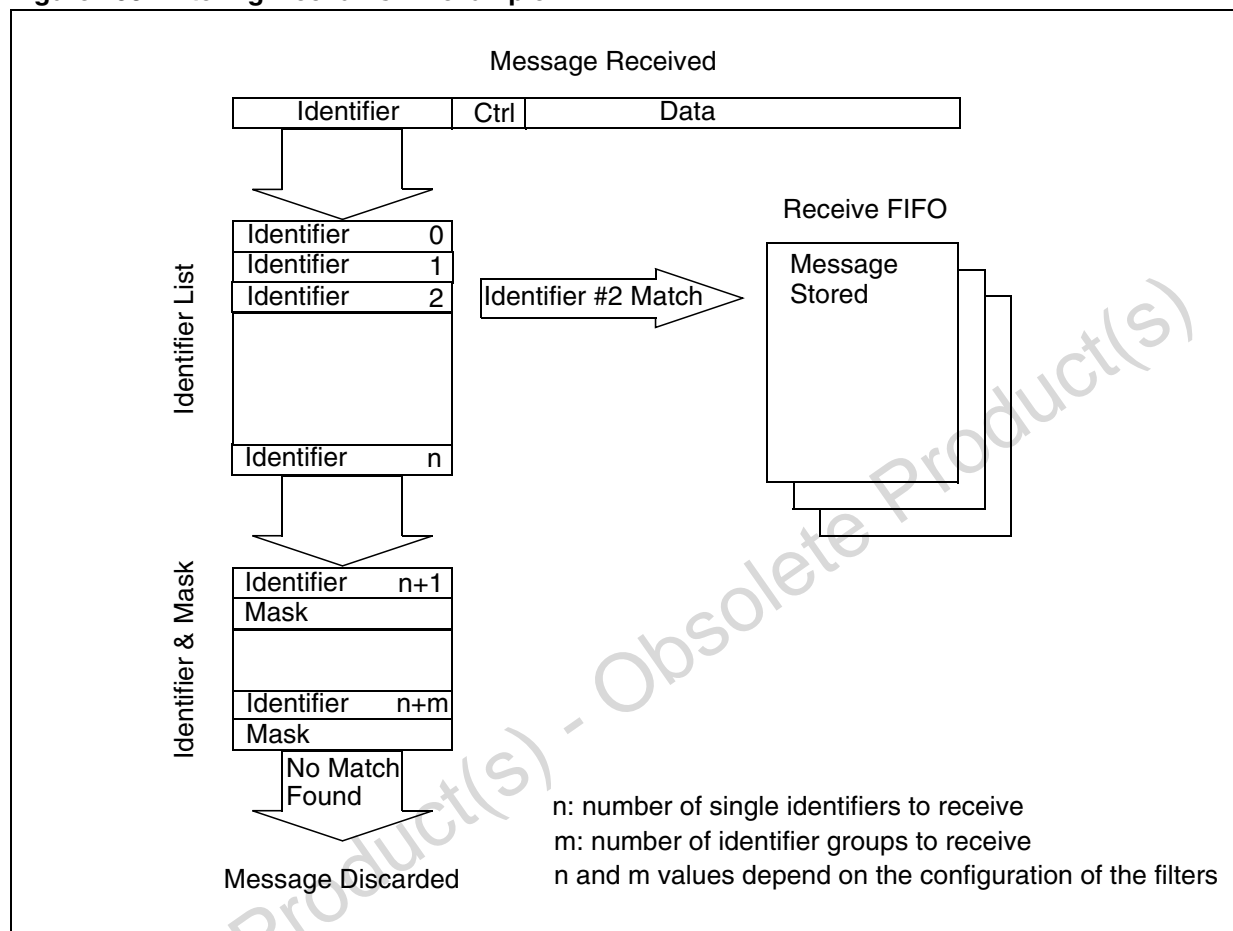
beCAN CONTROLLER (Cont'd)

Figure 100. Transmit Mailbox States



beCAN CONTROLLER (Cont'd)

Figure 103. Filtering Mechanism - example



The example above shows the filtering principle of the beCAN. On reception of a message, the identifier is compared first with the filters configured in identifier list mode. If there is a match, the message is stored in the FIFO and the index of the matching filter is stored in the Filter Match Index. As shown in the example, the identifier matches with Identifier #2 thus the message content and MFMI 2 is stored in the FIFO.

If there is no match, the incoming identifier is then compared with the filters configured in mask mode.

If the identifier does not match any of the identifiers configured in the filters, the message is discarded by hardware without software intervention.

10-BIT A/D CONVERTER (ADC) (Cont'd)

10.10.3.2 A/D Conversion

The analog input ports must be configured as input, no pull-up, no interrupt. Refer to the "I/O ports" chapter. Using these pins as analog inputs does not affect the ability of the port to be read as a logic input.

In the ADCCSR register:

- Select the CS[3:0] bits to assign the analog channel to convert.

ADC Conversion mode

In the ADCCSR register:

- Set the ADON bit to enable the A/D converter and to start the conversion. From this time on, the ADC performs a continuous conversion of the selected channel.

When a conversion is complete:

- The EOC bit is set by hardware.
- The result is in the ADCDR registers.

A read to the ADCDRH resets the EOC bit.

To read the 10 bits, perform the following steps:

1. Poll EOC bit
2. Read the ADCDRL register
3. Read the ADCDRH register. This clears EOC automatically.

To read only 8 bits, perform the following steps:

1. Poll EOC bit
2. Read the ADCDRH register. This clears EOC automatically.

10.10.3.3 Changing the conversion channel

The application can change channels during conversion. When software modifies the CH[3:0] bits

in the ADCCSR register, the current conversion is stopped, the EOC bit is cleared, and the A/D converter starts converting the newly selected channel.

10.10.3.4 ADCDR consistency

If an End Of Conversion event occurs after software has read the ADCDRLSB but before it has read the ADCDRMSB, there would be a risk that the two values read would belong to different samples.

To guarantee consistency:

- The ADCDRL and the ADCDRH registers are locked when the ADCCRL is read
- The ADCDRL and the ADCDRH registers are unlocked when the ADCDRH register is read or when ADON is reset.

This is important, as the ADCDR register will not be updated until the ADCDRH register is read.

10.10.4 Low Power Modes

Note: The A/D converter may be disabled by resetting the ADON bit. This feature allows reduced power consumption when no conversion is needed and between single shot conversions.

Mode	Description
WAIT	No effect on A/D Converter
HALT	A/D Converter disabled. After wakeup from Halt mode, the A/D Converter requires a stabilization time t_{STAB} (see Electrical Characteristics) before accurate conversions can be performed.

10.10.5 Interrupts

None.

INSTRUCTION SET OVERVIEW (Cont'd)

Mnemo	Description	Function/Example	Dst	Src	I1	H	I0	N	Z	C
ADC	Add with Carry	$A = A + M + C$	A	M		H		N	Z	C
ADD	Addition	$A = A + M$	A	M		H		N	Z	C
AND	Logical And	$A = A \cdot M$	A	M				N	Z	
BCP	Bit compare A, Memory	tst (A . M)	A	M				N	Z	
BRES	Bit Reset	bres Byte, #3	M							
BSET	Bit Set	bset Byte, #3	M							
BTJF	Jump if bit is false (0)	btjf Byte, #3, Jmp1	M							C
BTJT	Jump if bit is true (1)	btjt Byte, #3, Jmp1	M							C
CALL	Call subroutine									
CALLR	Call subroutine relative									
CLR	Clear		reg, M					0	1	
CP	Arithmetic Compare	tst(Reg - M)	reg	M				N	Z	C
CPL	One Complement	$A = \text{FFH} - A$	reg, M					N	Z	1
DEC	Decrement	dec Y	reg, M					N	Z	
HALT	Halt				1		0			
IRET	Interrupt routine return	Pop CC, A, X, PC			I1	H	I0	N	Z	C
INC	Increment	inc X	reg, M					N	Z	
JP	Absolute Jump	jp [TBL.w]								
JRA	Jump relative always									
JRT	Jump relative									
JRF	Never jump	jrf *								
JRIH	Jump if ext. INT pin = 1	(ext. INT pin high)								
JRIL	Jump if ext. INT pin = 0	(ext. INT pin low)								
JRH	Jump if H = 1	H = 1 ?								
JRNH	Jump if H = 0	H = 0 ?								
JRM	Jump if I1:0 = 11	I1:0 = 11 ?								
JRNM	Jump if I1:0 <> 11	I1:0 <> 11 ?								
JRMI	Jump if N = 1 (minus)	N = 1 ?								
JRPL	Jump if N = 0 (plus)	N = 0 ?								
JREQ	Jump if Z = 1 (equal)	Z = 1 ?								
JRNE	Jump if Z = 0 (not equal)	Z = 0 ?								
JRC	Jump if C = 1	C = 1 ?								
JRNC	Jump if C = 0	C = 0 ?								
JRULT	Jump if C = 1	Unsigned <								
JRUGE	Jump if C = 0	Jmp if unsigned >=								
JRUGT	Jump if (C + Z = 0)	Unsigned >								

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 AND TIMING CHARACTERISTICS (Cont'd)**12.5.3 Crystal and Ceramic Resonator Oscillators**

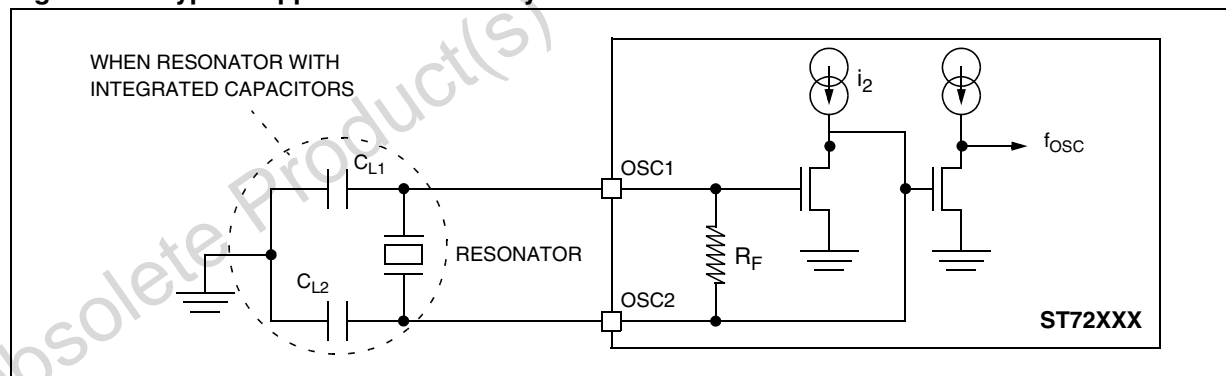
The ST7 internal clock can be supplied with four different Crystal/Ceramic resonator oscillators. All the information given in this paragraph is based on characterization results with specified typical external components. In the application, the resonator and the load capacitors have to be placed as

close as possible to the oscillator pins in order to minimize output distortion and start-up stabilization time. Refer to the crystal/ceramic resonator manufacturer for more details (frequency, package, accuracy...). ¹⁾²⁾

Symbol	Parameter	Conditions	Min	Max	Unit
f_{OSC}	Oscillator Frequency ³⁾	LP: Low power oscillator MP: Medium power oscillator MS: Medium speed oscillator HS: High speed oscillator	1 >2 >4 >8	2 4 8 16	MHz
R_F	Feedback resistor		20	40	k Ω
C_{L1} C_{L2}	Recommended load capacitance versus equivalent serial resistance of the crystal or ceramic resonator (R_S)	$R_S = 200\Omega$ LP oscillator $R_S = 200\Omega$ MP oscillator $R_S = 200\Omega$ MS oscillator $R_S = 100\Omega$ HS oscillator	22 22 18 15	56 46 33 33	pF

Symbol	Parameter	Conditions	Typ	Max	Unit
i_2	OSC2 driving current	$V_{DD} = 5V$ $V_{IN} = V_{SS}$ LP oscillator MP oscillator MS oscillator HS oscillator	80 160 310 610	150 250 460 910	μA

Figure 122. Typical Application with a Crystal or Ceramic Resonator

**Notes:**

1. Resonator characteristics given by the crystal/ceramic resonator manufacturer.
2. $t_{SU(OSC)}$ is the typical oscillator start-up time measured between $V_{DD} = 2.8V$ and the fetch of the first instruction (with a quick V_{DD} ramp-up from 0 to 5V (< 50 μs).
3. The oscillator selection can be optimized in terms of supply current using an high quality resonator with small R_S value. Refer to crystal/ceramic resonator manufacturer for more details.

12.11 TIMER PERIPHERAL CHARACTERISTICS

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 (output compare, input capture, external clock, PWM output...).

12.11.1 8-Bit PWM-ART Autoreload Timer

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{res(PWM)}$	PWM resolution time		1			t_{CPU}
		$f_{CPU} = 8 \text{ MHz}$	125			ns
f_{EXT}	ART external clock frequency		0		$f_{CPU}/2$	MHz
f_{PWM}	PWM repetition rate					
Res_{PWM}	PWM resolution				8	bit
V_{OS}	PWM/DAC output step voltage	$V_{DD} = 5V$, Res = 8-bits		20		mV
$t_{COUNTER}$	Timer clock period when internal clock is selected	$f_{CPU} = 8 \text{ MHz}$	1		128	t_{CPU}
			0.125		16	μs

12.11.2 8-Bit Timer

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{w(ICAP)in}$	Input capture pulse time		1			t_{CPU}
$t_{res(PWM)}$	PWM resolution time		2			
		$f_{CPU} = 8 \text{ MHz}$	250			ns
f_{PWM}	PWM repetition rate		0		$f_{CPU}/4$	MHz
Res_{PWM}	PWM resolution				8	bit
$t_{COUNTER}$	Timer clock period	$f_{CPU} = 8 \text{ MHz}$	2		8000	t_{CPU}
			0.250		1000	μs

12.11.3 16-Bit Timer

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{w(ICAP)in}$	Input capture pulse time		1			t_{CPU}
$t_{res(PWM)}$	PWM resolution time		2			
		$f_{CPU} = 8 \text{ MHz}$	250			ns
f_{EXT}	Timer external clock frequency		0		$f_{CPU}/4$	MHz
f_{PWM}	PWM repetition rate					
Res_{PWM}	PWM resolution				16	bit
$t_{COUNTER}$	Timer clock period when internal clock is selected	$f_{CPU} = 8 \text{ MHz}$	2		8	t_{CPU}
			0.250		1	μs

15 DEVELOPMENT TOOLS

Full details of tools available for the ST7 from third party manufacturers can be obtained from the STMicroelectronics Internet site:

→ www.st.com/mcu

Tools from iSystem and Hitex include C compilers, emulators and gang programmers.

Note: Before designing the board layout, it is recommended to check the overall dimensions of the socket as they may be greater than the dimensions of the device.

For footprint and other mechanical information about these sockets and adapters, refer to the manufacturer's datasheet.

ST Programming Tools

- ST7MDT25-EPB: For in-socket or ICC programming
- ST7-STICK: For ICC programming