**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"

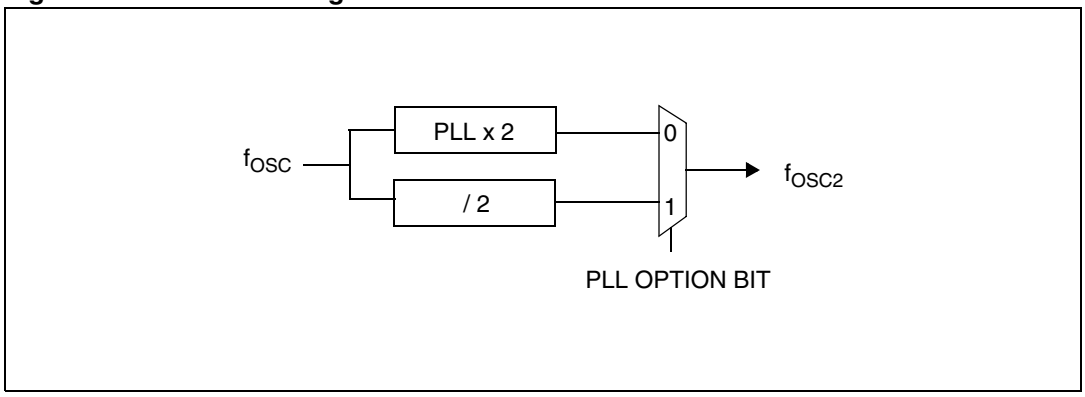
| Details | |
|---|---|
| Product Status | Obsolete |
| Core Processor | ST7 |
| Core Size | 8-Bit |
| Speed | 8MHz |
| Connectivity | I²C, SCI, SPI |
| Peripherals | LVD, POR, PWM, WDT |
| Number of I/O | 48 |
| Program Memory Size | 32KB (32K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 1K x 8 |
| Voltage - Supply (Vcc/Vdd) | 3.8V ~ 5.5V |
| Data Converters | A/D 16x10b |
| Oscillator Type | Internal |
| 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/st72f321ar6ta |

## 6.3 Phase locked loop

If the clock frequency input to the PLL is in the range 2 to 4 MHz, the PLL can be used to multiply the frequency by two to obtain an $f_{OSC2}$ of 4 to 8 MHz. The PLL is enabled by option byte. If the PLL is disabled, then $f_{OSC2} = f_{OSC}/2$.

**Caution:** The PLL is not recommended for applications where timing accuracy is required (see *Section 19.5.5: PLL characteristics on page 198*).

**Figure 10. PLL block diagram**




## 6.4 Multi-oscillator (MO)

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

- an external source
- 4 crystal or ceramic resonator oscillators
- an internal high frequency RC oscillator

Each oscillator is optimized for a given frequency range in terms of consumption and is selectable through the option byte. The associated hardware configurations are shown in *Table 10*. Refer to *Section 19: Electrical characteristics* 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 this 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.

# 8 Power saving modes

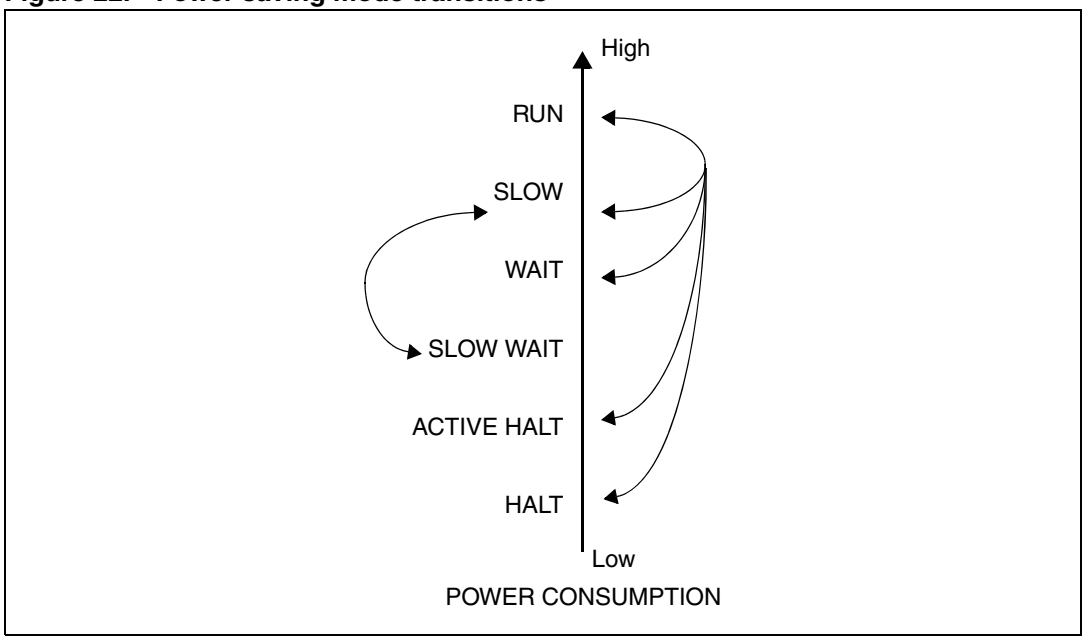
## 8.1 Introduction

To give a large measure of flexibility to the application in terms of power consumption, four main power saving modes are implemented in the ST7 (see *Figure 22*): Slow, Wait (Slow Wait), Active Halt and Halt.

After a RESET the normal operating mode is selected by default (Run mode). This mode drives the device (CPU and embedded peripherals) by means of a master clock which is based on the main oscillator frequency divided or multiplied by 2 ($f_{OSC2}$).

From Run mode, the different power saving modes may be selected by setting the relevant register bits or by calling the specific ST7 software instruction whose action depends on the oscillator status.

**Figure 22. Power saving mode transitions**




## 8.2 Slow mode

This mode has two targets:

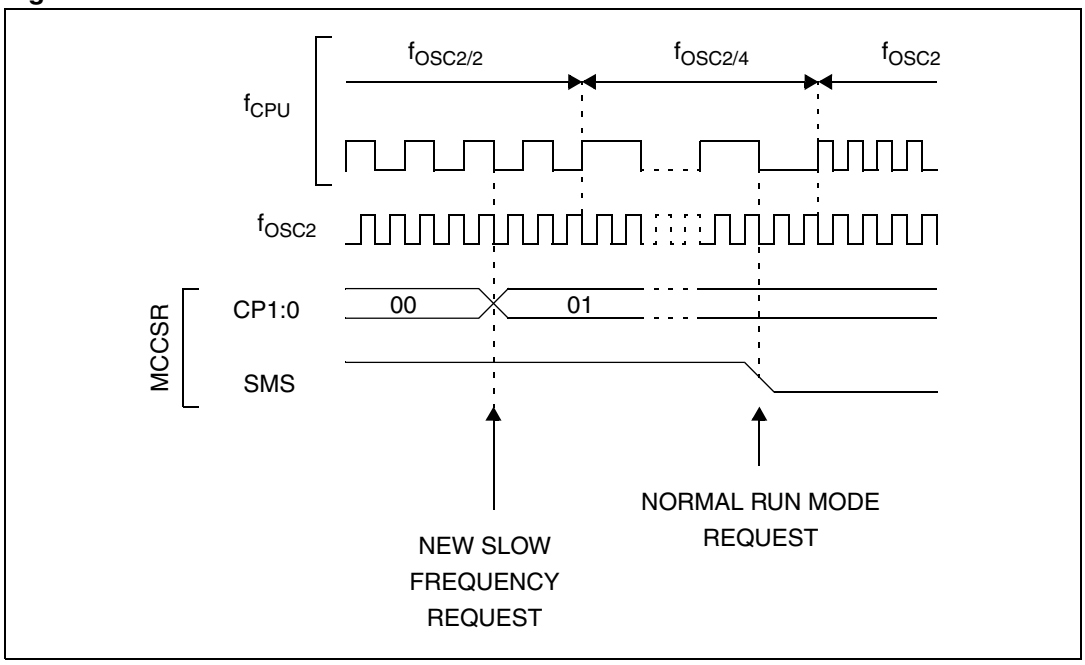
- To reduce power consumption by decreasing the internal clock in the device,
- To adapt the internal clock frequency ($f_{CPU}$) to the available supply voltage.

Slow mode is controlled by three bits in the MCCSR register: the SMS bit which enables or disables Slow mode and two CPx bits which select the internal slow frequency ($f_{CPU}$).

In this mode, the master clock frequency ($f_{OSC2}$) can be divided by 2, 4, 8 or 16. The CPU and peripherals are clocked at this lower frequency ($f_{CPU}$).

*Note: Slow Wait mode is activated when entering the Wait mode while the device is already in Slow mode.*

**Figure 23. Slow mode clock transitions**




## 8.3 Wait mode

Wait mode places the MCU in a low power consumption mode by stopping the CPU.

This power saving mode is selected by calling the 'WFI' instruction.

All peripherals remain active. During Wait mode, the I[1:0] bits of the CC register are forced to '10', to enable all interrupts. All other registers and memory remain unchanged. The MCU remains in Wait mode until an interrupt or RESET occurs, whereupon the Program Counter branches to the starting address of the interrupt or Reset service routine.

The MCU will remain in Wait mode until a Reset or an Interrupt occurs, causing it to wake up.

Refer to the following *Figure 24*.

Table 34. I/O port register map and reset values (continued)

| Address (Hex.) | Register label | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Reset value of all I/O port registers | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 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 | | | | | | | | |

## Related documentation

*SPI Communication between ST7 and EEPROM* (AN 970)

*S/W implementation of I2C bus master* (AN1045)

*Software LCD driver* (AN1048)

## 11.8 Main clock controller registers

### 11.8.1 MCC control/status register (MCCSR)

MCCSR Reset value: 0000 0000 (00h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MCO | CP[1:0] | | SMS | TB[1:0] | | OIE | OIF |
| RW | RW | | RW | RW | | RW | RW |

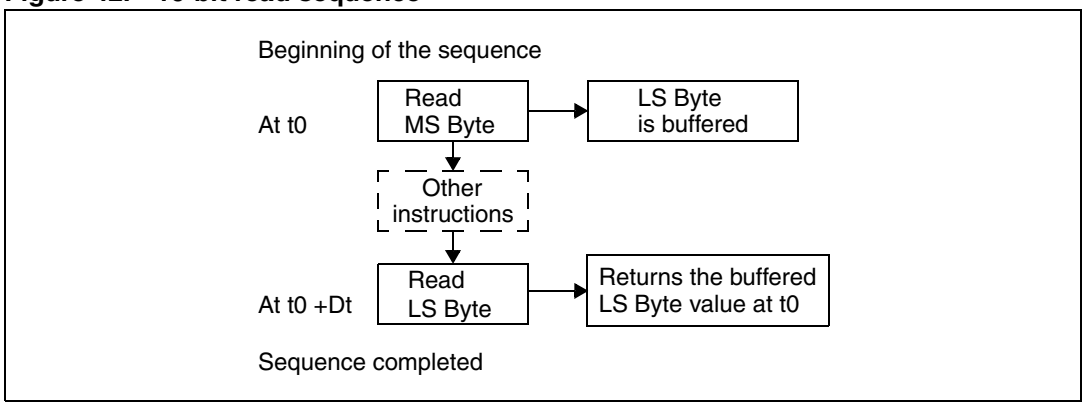
**Table 40. MCCSR register description**

| Bit | Name | Function |
|---|---|---|
| 7 | MCO | *Main clock out selection*<br>This bit enables the MCO alternate function on the PF0 I/O port. It is set and cleared by software.<br>0: MCO alternate function disabled (I/O pin free for general-purpose I/O)<br>1: MCO alternate function enabled ($f_{CPU}$ on I/O port)<br>*Note: To reduce power consumption, the MCO function is not active in Active Halt mode.* |
| 6:5 | CP[1:0] | *CPU clock prescaler*<br>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.<br>00: $f_{CPU}$ in Slow mode = $f_{OSC2}/2$<br>01: $f_{CPU}$ in Slow mode = $f_{OSC2}/4$<br>10: $f_{CPU}$ in Slow mode = $f_{OSC2}/8$<br>11: $f_{CPU}$ in Slow mode = $f_{OSC2}/16$ |
| 4 | SMS | *Slow mode select*<br>This bit is set and cleared by software.<br>0: Normal mode. $f_{CPU} = f_{OSC2}$<br>1: Slow mode. $f_{CPU}$ is given by CP1, CP0<br>See *Section 8.2: Slow mode on page 62* and *Chapter 11: Main clock controller with real-time clock and beeper (MCC/RTC)* for more details. |
| 3:2 | TB[1:0] | *Time base control*<br>These bits select the programmable divider time base. They are set and cleared by software (see *Table 41*).<br>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. |
| 1 | OIE | *Oscillator interrupt enable*<br>This bit set and cleared by software.<br>0: Oscillator interrupt disabled<br>1: Oscillator interrupt enabled<br>This interrupt can be used to exit from Active Halt mode.<br>When this bit is set, calling the ST7 software HALT instruction enters the Active Halt power saving mode. |

### 16-bit read sequence

The 16-bit read sequence (from either the Counter Register or the Alternate Counter Register) is illustrated in *Figure 42*.

**Figure 42. 16-bit read sequence**




The user must read the MS Byte first; the LS Byte value is then 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 timer mode is used (input capture, output compare, one pulse mode or PWM mode) an overflow occurs when the counter rolls over from FFFFh to 0000h, after which

- the TOF bit of the SR register is set
- a timer interrupt is generated if
  - the TOIE bit of the CR1 register is set and
  - the 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

*Note: 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).

### 13.3.2 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.

**Figure 43. Counter timing diagram, internal clock divided by 2**

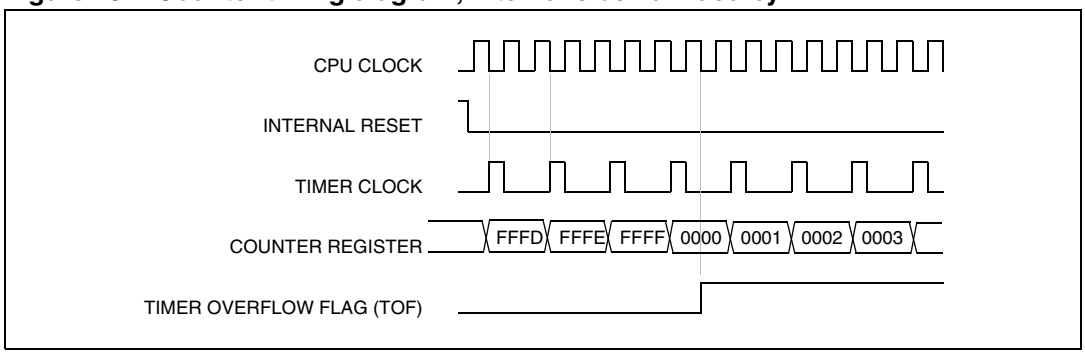



**Figure 44. Counter timing diagram, internal clock divided by 4**

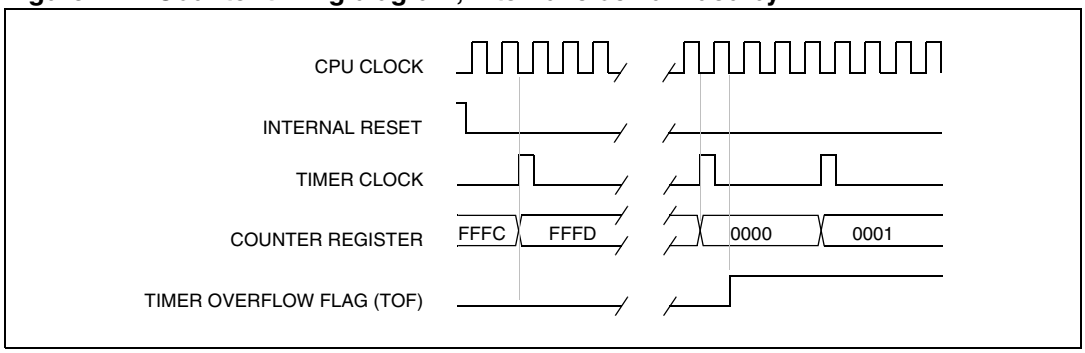
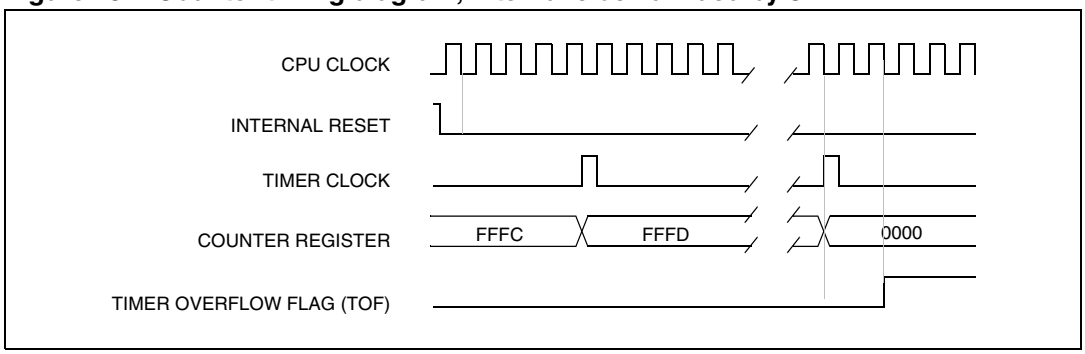



**Figure 45. Counter timing diagram, internal clock divided by 8**




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

### 13.3.6 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.

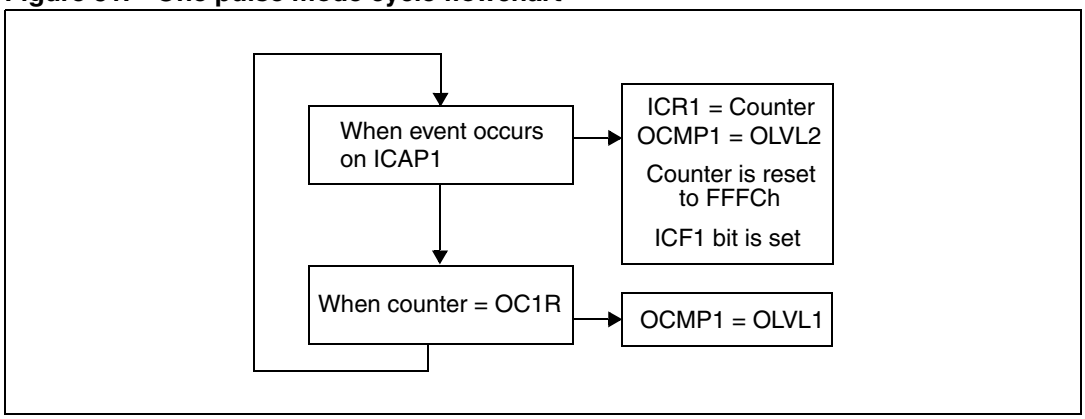
#### Procedure

To use one pulse mode:

1. Load the OC1R register with the value corresponding to the length of the pulse (using the appropriate formula below according to the timer clock source used).
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 61: Timer clock selection*).

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

**Figure 51. One pulse mode cycle flowchart**




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 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*i*LR register

**Figure 57. Generic $\overline{SS}$ timing diagram**

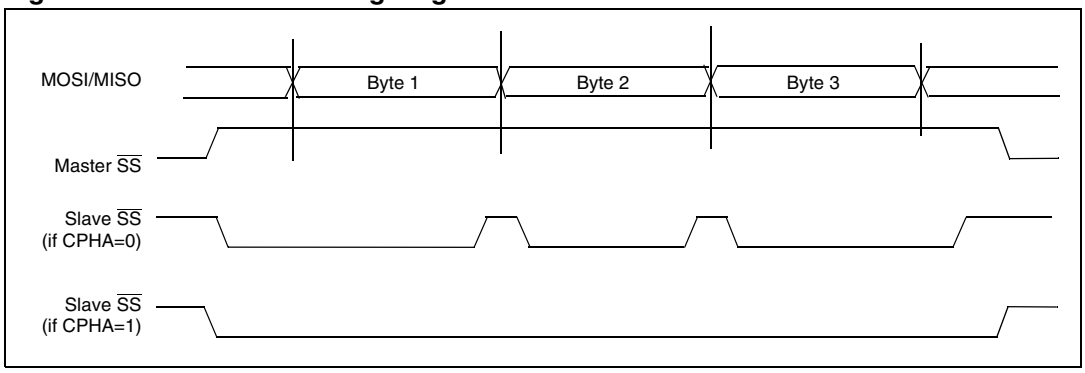
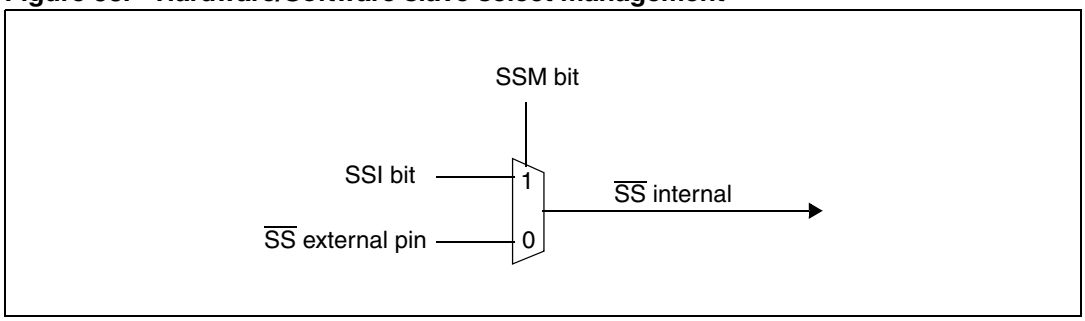



**Figure 58. Hardware/Software slave select management**




### 14.3.3 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:
   a) Select the clock frequency by configuring the SPR[2:0] bits.
   b) Select the clock polarity and clock phase by configuring the CPOL and CPHA bits. *Figure 59* 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 $\overline{SS}$ 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 $\overline{SS}$ is high).*

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

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

# 15 Serial communications interface (SCI)

## 15.1 Introduction

The Serial Communications Interface (SCI) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The SCI offers a very wide range of baud rates using two baud rate generator systems.

## 15.2 Main features

- Full duplex, asynchronous communications
- NRZ standard format (Mark/Space)
- Dual baud rate generator systems
- Independently programmable transmit and receive baud rates up to 500K baud
- Programmable data word length (8 or 9 bits)
- Receive buffer full, Transmit buffer empty and End of Transmission flags
- 2 receiver wake-up modes:
  - Address bit (MSB)
  - Idle line
- Muting function for multiprocessor configurations
- Separate enable bits for Transmitter and Receiver
- 4 error detection flags:
  - Overrun error
  - Noise error
  - Frame error
  - Parity error
- 5 interrupt sources with flags:
  - Transmit data register empty
  - Transmission complete
  - Receive data register full
  - Idle line received
  - Overrun error detected
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Reduced power consumption mode

bit by software the SCI insert a logic 1 bit at the end of the last break frame to guarantee the recognition of the start bit of the next frame.

**Idle characters**

Setting the TE bit drives the SCI to send an idle frame before the first data frame.

Clearing and then setting the TE bit during a transmission sends an idle frame after the current word.

*Note: Resetting and setting the TE bit causes the data in the TDR register to be lost. Therefore the best time to toggle the TE bit is when the TDRE bit is set, that is, before writing the next byte in the SCIDR.*

### 15.4.3 Receiver

The SCI can receive data words of either 8 or 9 bits. When the M bit is set, word length is 9 bits and the MSB is stored in the R8 bit in the SCICR1 register.

#### Character reception

During a SCI reception, data shifts in least significant bit first through the RDI pin. In this mode, the SCIDR register consists or a buffer (RDR) between the internal bus and the received shift register (see *Figure 62*).

**Procedure**

1. Select the M bit to define the word length.
2. Select the desired baud rate using the SCIBRR and the SCIERPR registers.
3. Set the RE bit, this enables the receiver which begins searching for a start bit.

When a character is received:

- The RDRF bit is set. It indicates that the content of the shift register is transferred to the RDR.
- An interrupt is generated if the RIE bit is set and the I bit is cleared in the CCR register.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.

Clearing the RDRF bit is performed by the following software sequence done by:

1. An access to the SCISR register
2. A read to the SCIDR register.

The RDRF bit must be cleared before the end of the reception of the next character to avoid an overrun error.

**Break character**

When a break character is received, the SCI handles it as a framing error.

**Idle character**

When an idle frame is detected, there is the same procedure as a data received character plus an interrupt if the ILIE bit is set and the I bit is cleared in the CCR register.

**Overrun error**

An overrun error occurs when a character is received when RDRF has not been reset. Data cannot be transferred from the shift register to the RDR register as long as the RDRF bit is not cleared.

**Table 72. SCI interrupt control/wake-up capability**

| Interrupt event | Event flag | Enable control bit | Exit from Wait | Exit from Halt |
|---|---|---|---|---|
| Idle Line Detected | IDLE | ILIE | Yes | No |
| Parity Error | PE | PIE | Yes | No |

## 15.7 SCI registers

### 15.7.1 Status register (SCISR)

SCISR Reset value: 1100 0000 (C0h)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TDRE | TC | RDRF | IDLE | OR | NF | FE | PE |
| RO | RO | RO | RO | RO | RO | RO | RO |

**Table 73. SCISR register description**

| Bit | Name | Function |
|---|---|---|
| 7 | TDRE | *Transmit data register empty*<br>This bit is set by hardware when the content of the TDR register has been transferred into the shift register. An interrupt is generated if the TIE bit = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a write to the SCIDR register).<br>0: Data is not transferred to the shift register<br>1: Data is transferred to the shift register<br>*Note: Data is not transferred to the shift register unless the TDRE bit is cleared.* |
| 6 | TC | *Transmission complete*<br>This bit is set by hardware when transmission of a frame containing Data is complete. An interrupt is generated if TCIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a write to the SCIDR register).<br>0: Transmission is not complete<br>1: Transmission is complete<br>*Note: TC is not set after the transmission of a Preamble or a Break.* |
| 5 | RDRF | *Received data ready flag*<br>This bit is set by hardware when the content of the RDR register has been transferred to the SCIDR register. An interrupt is generated if RIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).<br>0: Data is not received<br>1: Received data is ready to be read |

### Slave address transmission

Then the slave address is sent to the SDA line via the internal shift register.

- In 7-bit addressing mode, one address byte is sent.
- In 10-bit addressing mode, sending the first byte including the header sequence causes the following event:
  - The EVF bit is set by hardware with interrupt generation if the ITE bit is set.

Then the master waits for a read of the SR1 register followed by a write in the DR register, **holding the SCL line low** (see *Figure 68: Transfer sequencing* EV9).

Then the second address byte is sent by the interface.

After completion of this transfer (and acknowledge from the slave if the ACK bit is set):

- The EVF bit is set by hardware with interrupt generation if the ITE bit is set.

Then the master waits for a read of the SR1 register followed by a write in the CR register (for example set PE bit), **holding the SCL line low** (see *Figure 68: Transfer sequencing* EV6).

Next, the master must enter Receiver or Transmitter mode.

*Note: In 10-bit addressing mode, to switch the master to Receiver mode, software must generate a repeated Start condition and resend the header sequence with the least significant bit set (11110xx1).*

### Master receiver

Following the address transmission and after SR1 and CR registers have been accessed, the master receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

- Acknowledge pulse if the ACK bit is set
- EVF and BTF bits are set by hardware with an interrupt if the ITE bit is set.

Then the interface waits for a read of the SR1 register followed by a read of the DR register, **holding the SCL line low** (see *Figure 68: Transfer sequencing* EV7).

To close the communication: Before reading the last byte from the DR register, set the STOP bit to generate the Stop condition. The interface goes automatically back to slave mode (M/SL bit cleared).

*Note: In order to generate the non-acknowledge pulse after the last received data byte, the ACK bit must be cleared just before reading the second last data byte.*

### Master transmitter

Following the address transmission and after SR1 register has been read, the master sends bytes from the DR register to the SDA line via the internal shift register.

The master waits for a read of the SR1 register followed by a write in the DR register, **holding the SCL line low** (see *Figure 68: Transfer sequencing* EV8).

When the acknowledge bit is received, the interface sets:

- EVF and BTF bits with an interrupt if the ITE bit is set.

To close the communication: After writing the last byte to the DR register, set the STOP bit to generate the Stop condition. The interface goes automatically back to slave mode (M/SL bit cleared).

### Error cases

- **BERR**: Detection of a Stop or a Start condition during a byte transfer. In this case, the EVF and BERR bits are set by hardware with an interrupt if ITE is set.
  Note that BERR will not be set if an error is detected during the first or second pulse of each 9-bit transaction:
  - ***Single Master Mode***
    If a Start or Stop is issued during the first or second pulse of a 9-bit transaction, the BERR flag will not be set and transfer will continue however the BUSY flag will be reset. To work around this, slave devices should issue a NACK when they receive a misplaced Start or Stop. The reception of a NACK or BUSY by the master in the middle of communication makes it possible to re-initiate transmission.
  - ***Multimaster Mode***
    Normally the BERR bit would be set whenever unauthorized transmission takes place while transfer is already in progress. However, an issue will arise if an external master generates an unauthorized Start or Stop while the $I^2C$ master is on the first or second pulse of a 9-bit transaction. It is possible to work around this by polling the BUSY bit during $I^2C$ master mode transmission. The resetting of the BUSY bit can then be handled in a similar manner as the BERR flag being set.
- **AF**: Detection of a non-acknowledge bit. In this case, the EVF and AF bits are set by hardware with an interrupt if the ITE bit is set. To resume, set the Start or Stop bit. The AF bit is cleared by reading the I2CSR2 register. However, if read before the completion of the transmission, the AF flag will be set again, thus possibly generating a new interrupt. Software must ensure either that the SCL line is back at 0 before reading the SR2 register, or be able to correctly handle a second interrupt during the 9th pulse of a transmitted byte.
- **ARLO:** Detection of an arbitration lost condition.
  In this case the ARLO bit is set by hardware (with an interrupt if the ITE bit is set and the interface goes automatically back to slave mode (the M/SL bit is cleared).

*Note: In all these cases, the SCL line is not held low; however, the SDA line can remain low due to possible '0' bits transmitted last. It is then necessary to release both lines by software.*

# 17 10-bit A/D converter (ADC)

## 17.1 Introduction

The on-chip Analog to Digital Converter (ADC) peripheral is a 10-bit, successive approximation converter with internal sample and hold circuitry. This peripheral has up to 16 multiplexed analog input channels (refer to device pin out description) that allow the peripheral to convert the analog voltage levels from up to 16 different sources.

The result of the conversion is stored in a 10-bit data register. The A/D converter is controlled through a control/status register.

## 17.2 Main features

- 10-bit conversion
- Up to 16 channels with multiplexed input
- Linear successive approximation
- Data register (DR) which contains the results
- Conversion complete status flag
- On/off bit (to reduce consumption)

The block diagram is shown in *Figure 70*.

**Figure 70. ADC block diagram**

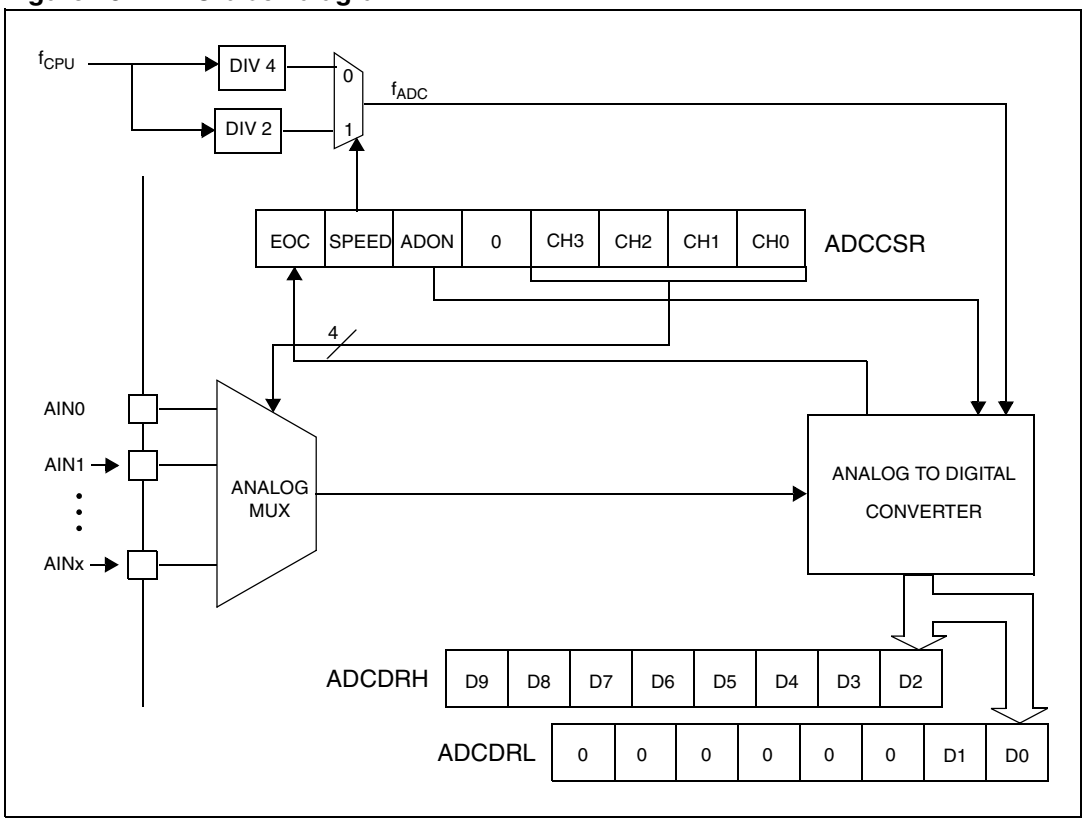


**Table 103. Instruction set overview (continued)**

| Mnemo | Description | Function/Example | Dst | Src | I1 | H | I0 | N | Z | C |
|---|---|---|---|---|---|---|---|---|---|---|
| JRUGT | Jump if (C + Z = 0) | Unsigned > | | | | | | | | |
| JRULE | Jump if (C + Z = 1) | Unsigned <= | | | | | | | | |
| LD | Load | dst <= src | reg, M | M, reg | | | | N | Z | |
| MUL | Multiply | X,A = X * A | A, X, Y | X, Y, A | | 0 | | | | 0 |
| NEG | Negate (2's compl) | neg $10 | reg, M | | | | | N | Z | C |
| NOP | No Operation | | | | | | | | | |
| OR | OR operation | A = A + M | A | M | | | | N | Z | |
| POP | Pop from the Stack | pop reg | reg | M | | | | | | |
| | | pop CC | CC | M | I1 | H | I0 | N | Z | C |
| PUSH | Push onto the Stack | push Y | M | reg, CC | | | | | | |
| RCF | Reset carry flag | C = 0 | | | | | | | | 0 |
| RET | Subroutine Return | | | | | | | | | |
| RIM | Enable Interrupts | I1:0 = 10 (level 0) | | | 1 | | 0 | | | |
| RLC | Rotate left true C | C <= A <= C | reg, M | | | | | N | Z | C |
| RRC | Rotate right true C | C => A => C | reg, M | | | | | N | Z | C |
| RSP | Reset Stack Pointer | S = Max allowed | | | | | | | | |
| SBC | Subtract with Carry | A = A - M - C | A | M | | | | N | Z | C |
| SCF | Set carry flag | C = 1 | | | | | | | | 1 |
| SIM | Disable Interrupts | I1:0 = 11 (level 3) | | | 1 | | 1 | | | |
| SLA | Shift left Arithmetic | C <= A <= 0 | reg, M | | | | | N | Z | C |
| SLL | Shift left Logic | C <= A <= 0 | reg, M | | | | | N | Z | C |
| SRL | Shift right Logic | 0 => A => C | reg, M | | | | | 0 | Z | C |
| SRA | Shift right Arithmetic | A7 => A => C | reg, M | | | | | N | Z | C |
| SUB | Subtraction | A = A - M | A | M | | | | N | Z | C |
| SWAP | SWAP nibbles | A7-A4 <=> A3-A0 | reg, M | | | | | N | Z | |
| TNZ | Test for Neg & Zero | tnz lbl1 | | | | | | N | Z | |
| TRAP | S/W trap | S/W interrupt | | | 1 | | 1 | | | |
| WFI | Wait for Interrupt | | | | 1 | | 0 | | | |
| XOR | Exclusive OR | A = A XOR M | A | M | | | | N | Z | |

### 19.7.3 Absolute maximum ratings (electrical sensitivity)

Based on two different tests (ESD and LU) using specific measurement methods, the product is stressed in order to determine its performance in terms of electrical sensitivity. For more details, refer to the application note AN1181.

#### Electrostatic discharge (ESD)

Electrostatic discharges (a positive then a negative pulse separated by 1 second) are applied to the pins of each sample according to each pin combination. The sample size depends on the number of supply pins in the device (3 parts*(n+1) supply pin). Two models can be simulated: Human Body Model and Machine Model. This test conforms to the JESD22-A114A/A115A standard.

**Table 124. ESD absolute maximum ratings**

| Symbol | Ratings | Conditions | Max. value(1) | Unit |
|---|---|---|---|---|
| $V_{ESD(HBM)}$ | Electrostatic discharge voltage (Human Body Model) | $T_A$ = +25°C | 2000 | V |
| $V_{ESD(MM)}$ | Electrostatic discharge voltage (Machine Model) | | 200 | |

1. Data based on characterization results, not tested in production.

**Static latch-up (LU)** Two complementary static tests are required on six parts to assess the latch-up performance:

- A supply overvoltage is applied to each power supply pin.
- A current injection is applied to each input, output and configurable I/O pin.

These tests are compliant with the EIA/JESD 78 IC latch-up standard.

**Table 125. Electrical sensitivities**

| Symbol | Parameter | Conditions | Class(1) |
|---|---|---|---|
| LU | Static latch-up class | $T_A$ = +25°C<br>$T_A$ = +85°C<br>$T_A$ = +125°C | A<br>A<br>A |

1. Class description: A Class is an STMicroelectronics internal specification. All its limits are higher than the JEDEC specifications, that means when a device belongs to Class A it exceeds the JEDEC standard. B Class strictly covers all the JEDEC criteria (international standard).

### 19.11.2 I²C - inter IC control interface

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

Refer to *Section 19.8: I/O port pin characteristics* for more details on the input/output alternate function characteristics (SDAI and SCLI). The ST7 I2C interface meets the requirements of the standard I2C communication protocol described in the following table.

**Table 133. I²C control interface characteristics**

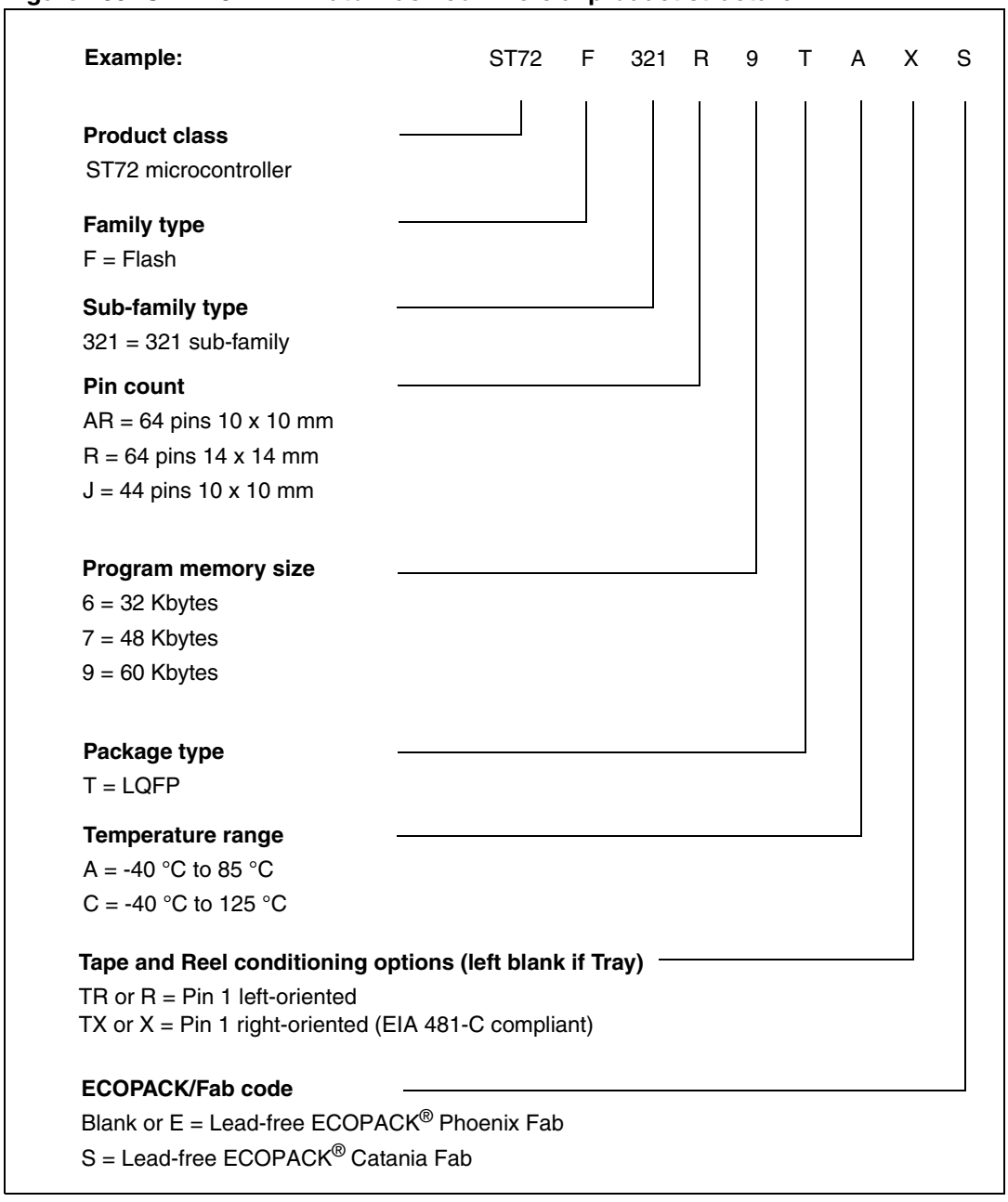
| Symbol | Parameter | Standard mode I²C | | Fast mode I²C(1) | | Unit |
|---|---|---|---|---|---|---|
| | | Min(2) | Max(2) | Min(2) | Max(2) | |
| $t_{w(SCLL)}$ | SCL clock low time | 4.7 | | 1.3 | | µs |
| $t_{w(SCLH)}$ | SCL clock high time | 4.0 | | 0.6 | | |
| $t_{su(SDA)}$ | SDA setup time | 250 | | 100 | | ns |
| $t_{h(SDA)}$ | SDA data hold time | 0(3) | | 0(4) | 900(3) | |
| $t_{r(SDA)}$ $t_{r(SCL)}$ | SDA and SCL rise time | | 1000 | 20+0.1$C_b$ | 300 | |
| $t_{f(SDA)}$ $t_{f(SCL)}$ | SDA and SCL fall time | | 300 | | | |
| $t_{h(STA)}$ | START condition hold time | 4.0 | | 0.6 | | µs |
| $t_{su(STA)}$ | Repeated START condition setup time | 4.7 | | | | |
| $t_{su(STO)}$ | STOP condition setup time | 4.0 | | | | |
| $t_{w(STO:STA)}$ | STOP to START condition time (bus free) | 4.7 | | 1.3 | | |
| $C_b$ | Capacitive load for each bus line | | 400 | | 400 | pF |

1. At 4 MHz $f_{CPU}$, maximum I²C speed (400 kHz) is not achievable. In this case, maximum I²C speed will be approximately 260 kHz.
2. Data based on standard I²C protocol requirement, not tested in production.
3. The maximum hold time of the START condition has only to be met if the interface does not stretch the low period of SCL signal.
4. The device must internally provide a hold time of at least 300ns for the SDA signal in order to bridge the undefined region of the falling edge of SCL.

### 21.1.2 Flash ordering information

The following *Figure 105* serves as a guide for ordering.

**Figure 105. ST72F321xxx-Auto Flash commercial product structure**




1. For a list of available options (e.g. memory size, package) and orderable part numbers or for further information on any aspect of this device, please go to *www.st.com* or contact the ST Sales Office nearest to you.

```
; check for falling edge
cp A,#$02

jrne OUT

TNZ Y

jrne OUT

LD A,#$01

LD sema,A
; set the semaphore to '1' if edge is detected
RIM
 ; reset the interrupt mask
LD A,sema
; check the semaphore status
CP A,#$01

jrne OUT

call call_routine
; call the interrupt routine
RIM

OUT:
RIM
JP while_loop

.call_routine
; entry to call_routine
PUSH A

PUSH X

PUSH CC

.ext1_rt
; entry to interrupt routine
LD A,#$00

LD sema,A

IRET
```