Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

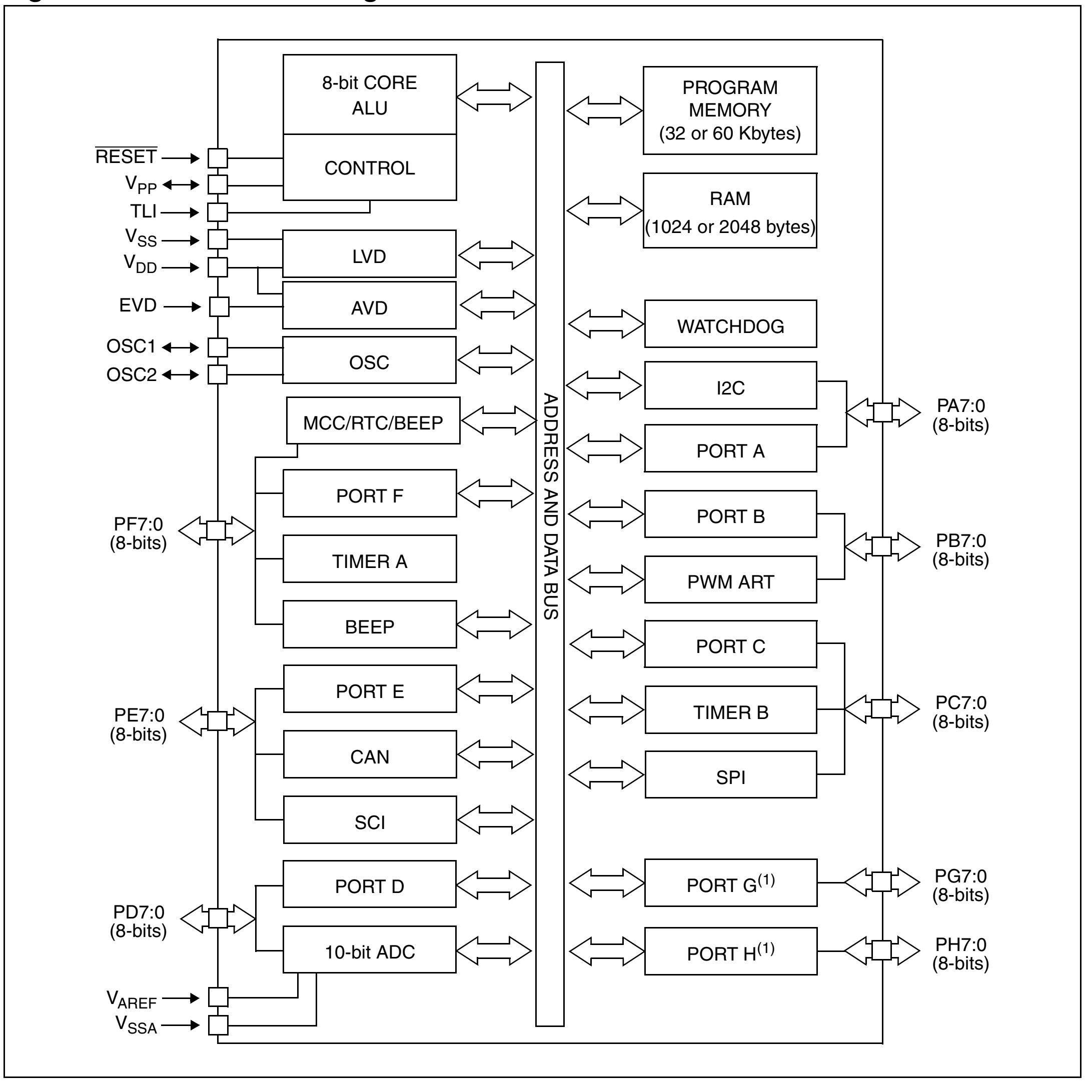
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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, I²C, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	48
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 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/st72f521r9tatr

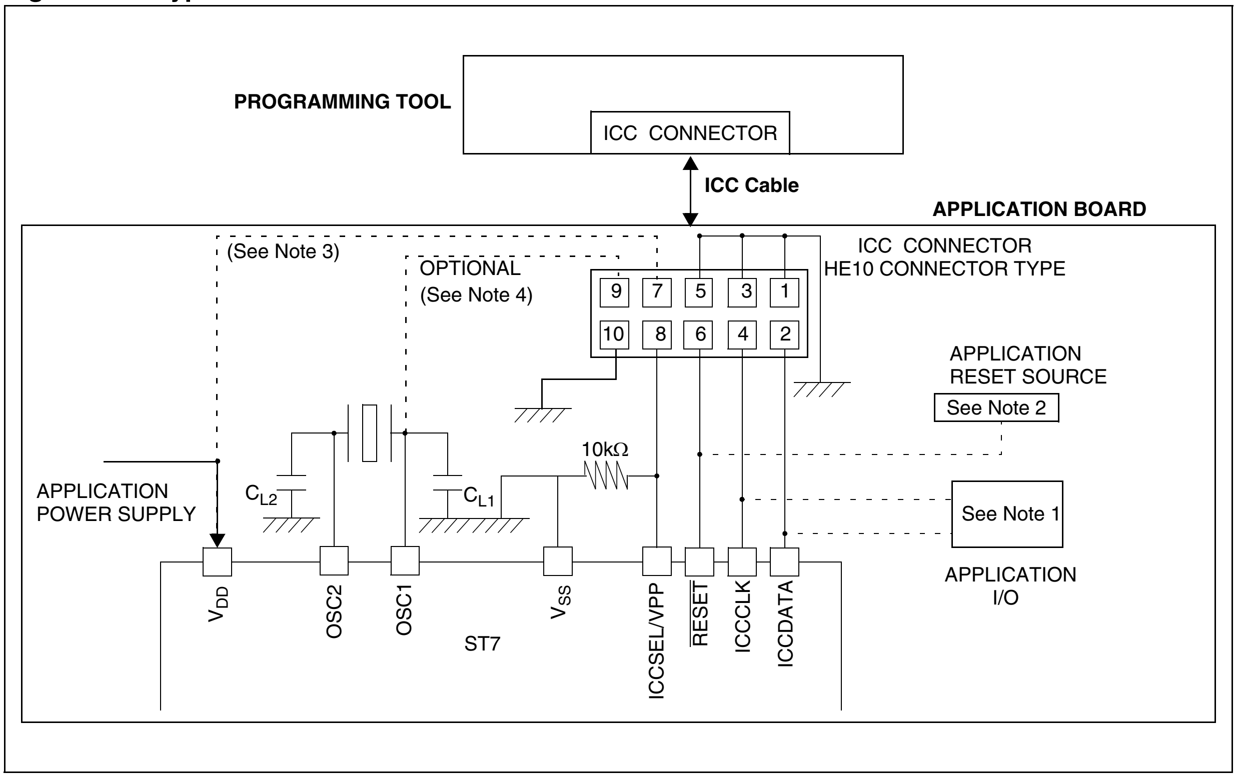
List of tables

Figure 1. Device block diagram



1. On certain devices only (see *Section Table 3.: Device pin description on page 23*)

Figure 6. Typical ICC interface



1. If the ICCCLK or ICCDATA pins are only used as outputs in the application, no signal isolation is necessary. As soon as the programming tool is plugged to the board, even if an ICC session is not in progress, the ICCCLK and ICCDATA pins are not available for the application. If they are used as inputs by the application, isolation such as a serial resistor has to implemented in case another device forces the signal. Refer to the programming tool documentation for recommended resistor values.
2. During the ICC session, the programming tool must control the $\overline{\text{RESET}}$ pin. This can lead to conflicts between the programming tool and the application reset circuit if it drives more than 5mA at high level (push-pull output or pull-up resistor < 1K). A schottky diode can be used to isolate the application RESET circuit in this case. When using a classical RC network with R > 1K or a reset management IC with open-drain output and pull-up resistor > 1K, no additional components are needed. In all cases the user must ensure that no external reset is generated by the application during the ICC session.
3. The use of Pin 7 of the ICC connector depends on the programming tool architecture. This pin must be connected when using most ST programming tools (it is used to monitor the application power supply). Please refer to the programming tool manual.
4. Pin 9 has to be connected to the OSC1 or OSCIN pin of the ST7 when the clock is not available in the application or if the selected clock option is not programmed in the option byte. ST7 devices with multi-oscillator capability need to have OSC2 grounded in this case.

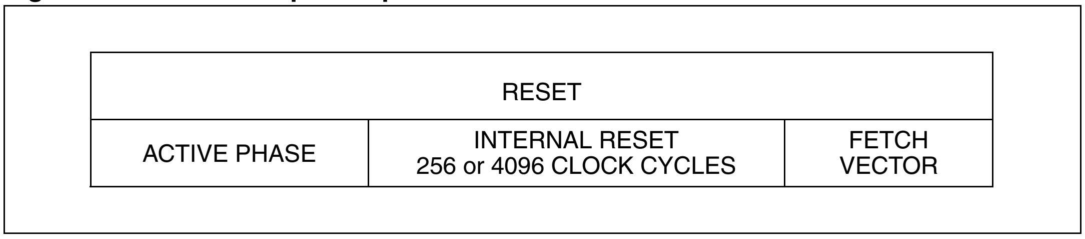
4.5 ICP (in-circuit programming)

To perform ICP the microcontroller must be switched to ICC (in-circuit communication) mode by an external controller or programming tool.

Depending on the ICP code downloaded in RAM, Flash memory programming can be fully customized (number of bytes to program, program locations, or selection serial communication interface for downloading).

When using an STMicroelectronics or third-party programming tool that supports ICP and the specific microcontroller device, the user needs only to implement the ICP hardware interface on the application board (see *Figure 6*). For more details on the pin locations, refer to the device pinout description.

Figure 12. RESET sequence phases



6.5.2 Asynchronous external $\overline{\text{RESET}}$ pin

The $\overline{\text{RESET}}$ pin is both an input and an open-drain output with integrated R_{ON} weak pull-up resistor. This pull-up has no fixed value but varies in accordance with the input voltage. It can be pulled low by external circuitry to reset the device. See *Section 20.9: Control pin characteristics on page 240*for more details.

A RESET signal originating from an external source must have a duration of at least $t_{h(RSTL)in}$ in order to be recognized (see *Figure 13*). This detection is asynchronous and therefore the MCU can enter reset state even in Halt mode.

The $\overline{\text{RESET}}$ pin is an asynchronous signal which plays a major role in EMS performance. In a noisy environment, it is recommended to follow the guidelines mentioned in *Section 20: Electrical characteristics*.

If the external $\overline{\text{RESET}}$ pulse is shorter than $t_{w(RSTL)out}$ (see short ext. Reset in *Figure 13*), the signal on the $\overline{\text{RESET}}$ pin may be stretched. Otherwise the delay will not be applied (see long ext. Reset in *Figure 13*). Starting from the external RESET pulse recognition, the device $\overline{\text{RESET}}$ pin acts as an output that is pulled low during at least $t_{w(RSTL)out}$.

6.5.3 External power-on RESET

If the LVD is disabled by option byte, to start up the microcontroller correctly, the user must ensure by means of an external reset circuit that the reset signal is held low until V_{DD} is over the minimum level specified for the selected f_{OSC} frequency (see *Section 20.3: Operating conditions on page 221*).

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

6.5.4 Internal low voltage detector (LVD) RESET

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

- Power-on RESET
- Voltage drop RESET

The device $\overline{\text{RESET}}$ pin acts as an output that is pulled low when $V_{DD} < V_{IT+}$ (rising edge) or $V_{DD} < V_{IT-}$ (falling edge) as shown in *Figure 13*.

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

Table 17. Interrupt software priority levels

Interrupt software priority	Level	I1	I0
Level 0 (main)	Low	1	0
Level 1	↓	0	1
Level 2		0	0
Level 3 (= interrupt disable[1])	High	1	1

1. TLI, TRAP and RESET events can interrupt a level 3 program.

7.5.2 Interrupt software priority registers (ISPRx)

These four registers are read/write, with the exception of bits 7:4 of ISPR3, which are read only.

ISPRx Reset value: 1111 1111 (FFh)

	7	6	5	4	3	2	1	0
ISPR0	I1_3	I0_3	I1_2	I0_2	I1_1	I0_1	I1_0	I0_0
ISPR1	I1_7	I0_7	I1_6	I0_6	I1_5	I0_5	I1_4	I0_4
ISPR2	I1_11	I0_11	I1_10	I0_10	I1_9	I0_9	I1_8	I0_8
ISPR3	1	1	1	1	I1_13	I0_13	I1_12	I0_12

These four registers contain the interrupt software priority of each interrupt vector.

- Each interrupt vector (except RESET and TRAP) has corresponding bits in these registers where its own software priority is stored. This correspondence is shown in the following *Table 18*.

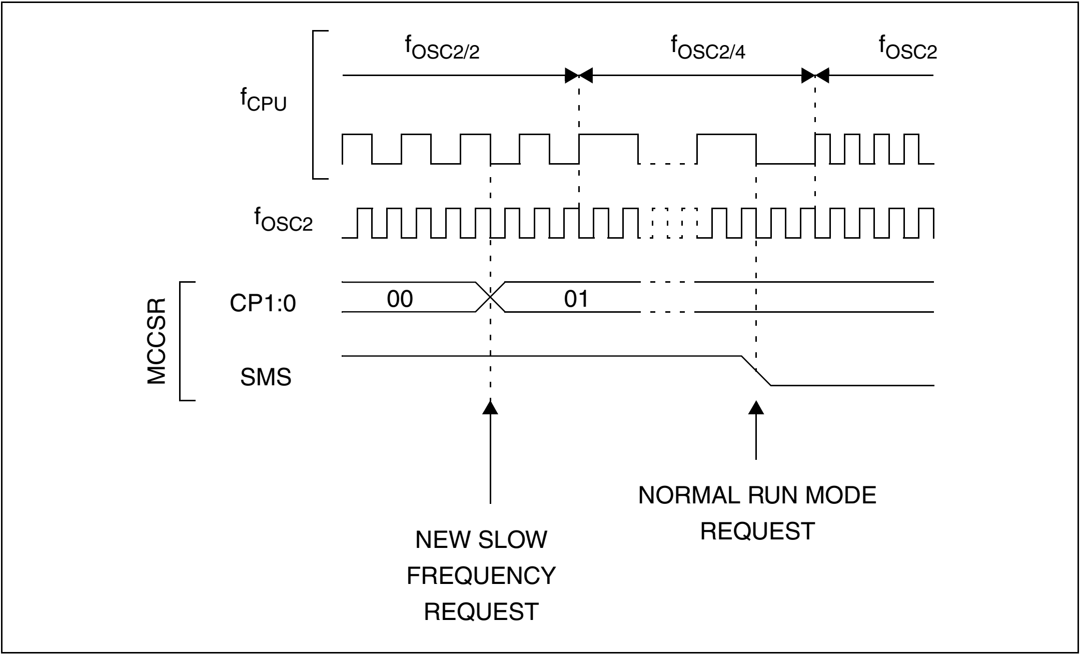
Table 18. Interrupt priority bits

Vector address	ISPRx bits
FFFBh-FFFAh	I1_0 and I0_0 bits[1]
FFF9h-FFF8h	I1_1 and I0_1 bits
...	...
FFE1h-FFE0h	I1_13 and I0_13 bits

1. Bits in the ISPRx registers which correspond to the TLI can be read and written but they are not significant in the interrupt process management.

- Each I1_x and I0_x bit value in the ISPRx registers has the same meaning as the I1 and I0 bits in the CC register.
- Level 0 cannot be written (I1_x = 1, I0_x = 0). In this case, the previously stored value is kept (Example: previous = CFh, write = 64h, result = 44h).

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*.

8.4 Active Halt and Halt modes

Active Halt and Halt modes are the two lowest power consumption modes of the MCU. They are both entered by executing the ‘HALT’ instruction. The decision to enter either in Active Halt or Halt mode is given by the MCC/RTC interrupt enable flag (OIE bit in MCCSR register) as shown in *Table 27*.

Table 27. MCC/RTC low power mode selection

MCCSR OIE bit	Power saving mode entered when HALT instruction is executed
0	Halt
1	Active Halt

8.4.1 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 the OIE bit of the Main Clock Controller Status register (MCCSR) is set (see *Section 12.3: ART registers on page 96* for more details on the MCCSR register).

The MCU can exit Active Halt mode on reception of an MCC/RTC interrupt or a RESET. In ROM devices, external interrupts can be used to wake up the MCU. When exiting Active Halt mode by means of an interrupt, no 256 or 4096 CPU cycle delay occurs. The CPU resumes operation by servicing the interrupt or by fetching the reset vector which woke it up (see *Figure 26*).

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

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

The safeguard against staying locked in Active Halt mode is provided by the oscillator interrupt.

Note: *As soon as the interrupt capability of one of the oscillators is selected (MCCSR.OIE bit set), entering Active Halt mode 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.

Caution: When exiting Active Halt mode following an MCC/RTC interrupt, OIE bit of MCCSR register must not be cleared before t_{DELAY} after the interrupt occurs (t_{DELAY} = 256 or 4096 t_{CPU} delay depending on option byte). Otherwise, the ST7 enters Halt mode for the remaining t_{DELAY} period.

Table 46. Prescaler selection for ART (continued)

$f_{COUNTER}$	With f_{INPUT} = 8 MHz	CC2	CC1	CC0
f_{INPUT} / 8	1 MHz	0	1	1
f_{INPUT} / 16	500 kHz	1	0	0
f_{INPUT} / 32	250 kHz	1	0	1
f_{INPUT} / 64	125 kHz	1	1	0
f_{INPUT} / 128	62.5 kHz	1	1	1

12.3.2 Counter access register (ARTCAR)

ARTCAR Reset value: 0000 0000 (00h)

7	6	5	4	3	2	1	0
CA[7:0]							
RW							

Table 47. ARTCAR register description

Bit	Name	Function
7:0	CA[7:0]	*Counter Access Data* These bits can be set and cleared either by hardware or by software. The ARTCAR register is used to read or write the auto-reload counter “on the fly” (while it is counting).

12.3.3 Auto-reload register (ARTARR)

ARTARR Reset value: 0000 0000 (00h)

7	6	5	4	3	2	1	0
AR[7:0]							
RW							

Table 48. ARTAAR register description

Bit	Name	Function
7:0	AR[7:0]	*Counter Auto-Reload Data* These bits are set and cleared by software. They are used to hold the auto-reload value which is automatically loaded in the counter when an overflow occurs. At the same time, the PWM output levels are changed according to the corresponding OPx bit in the PWMCR register.

This register has two PWM management functions:

– Adjusting the PWM frequency

– Setting the PWM duty cycle resolution

14 Serial peripheral interface (SPI)

14.1 Introduction

The Serial Peripheral Interface (SPI) allows full-duplex, synchronous, serial communication with external devices. An SPI system may consist of a master and one or more slaves however the SPI interface cannot be a master in a multimaster system.

14.2 Main features

- Full duplex synchronous transfers (on 3 lines)
- Simplex synchronous transfers (on 2 lines)
- Master or slave operation
- 6 master mode frequencies ($f_{CPU}/4$ max.)
- $f_{CPU}/2$ max. slave mode frequency (see note)
- $\overline{SS}$ Management by software or hardware
- Programmable clock polarity and phase
- End of transfer interrupt flag
- Write collision, Master Mode Fault and Overrun flags

Note: In slave mode, continuous transmission is not possible at maximum frequency due to the software overhead for clearing status flags and to initiate the next transmission sequence.

14.3 General description

Figure 55 shows the serial peripheral interface (SPI) block diagram. There are three registers:

- SPI Control Register (SPICR)
- SPI Control/Status Register (SPICSR)
- SPI Data Register (SPIDR)

The SPI is connected to external devices through four pins:

- MISO (Master In / Slave Out data)
- MOSI (Master Out / Slave In data)
- SCK (Serial Clock out by SPI masters and input by SPI slaves)
- $\overline{SS}$ (Slave select): This input signal acts as a ‘chip select’ to let the SPI master communicate with slaves individually and to avoid contention on the data lines. Slave $\overline{SS}$ inputs can be driven by standard I/O ports on the master MCU.

17.3.3 Modes of operation

The CAN core unit assumes one of the seven states described below.

Standby

Standby mode is entered either on a chip reset or on resetting the RUN bit in the Control/Status Register (CSR). Any on-going transmission or reception operation is not interrupted and completes normally before the Bit Time Logic and the clock prescaler are turned off for minimum power consumption. This state is signalled by the RUN bit being read-back as 0.

Once in standby, the only event monitored is the reception of a dominant bit which causes a wake-up interrupt if the SCIE bit of the Interrupt Control Register (ICR) is set.

The Standby mode is left by setting the RUN bit. If the WKPS bit is set in the CSR register, then the controller passes through WAKE-UP, otherwise it enters RESYNC directly.

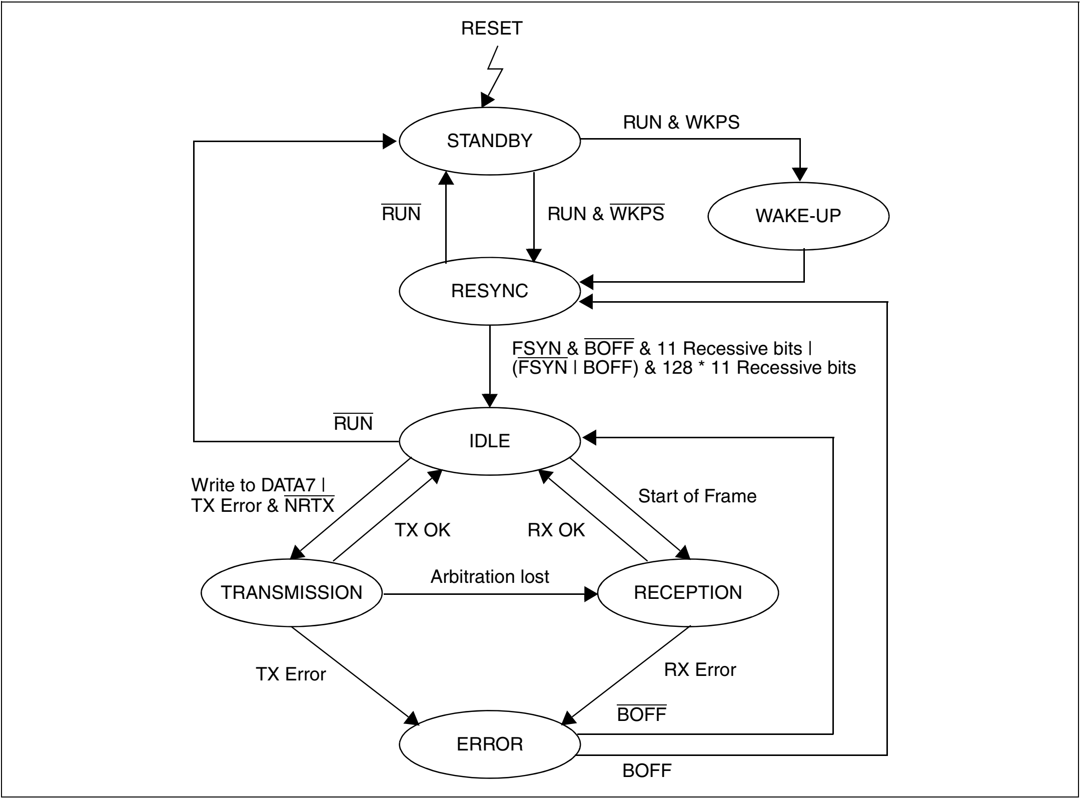
It is important to note that the wake-up mechanism is software-driven and therefore carries a significant time overhead. All messages received after the wake-up bit and before the controller is set to run and has completed synchronization are ignored.

Note: Standby mode is not entered on resetting the RUN bit in the Control/Status register (CSR) if the CANRX pin is shorted to GND.

Wake-up

The CAN bus line is forced to dominant for one bit time signalling the wake-up condition to all other bus members.

Figure 72. CAN controller state diagram



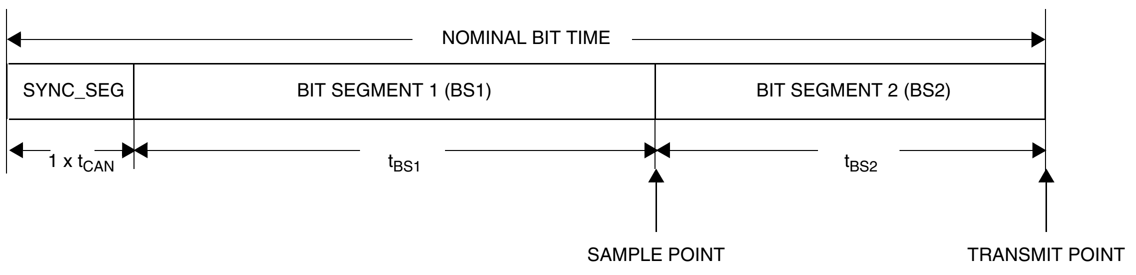
To guarantee the correct behavior of the CAN controller, SYNC_SEG + BS1 + BS2 must be greater than or equal to 5 time quanta.

The CAN controller resynchronizes on recessive to dominant edges only.

For a detailed description of the CAN resynchronization mechanism and other bit timing configuration constraints, please refer to the Bosch CAN standard 2.0.

As a safeguard against programming errors, the configuration of the Bit Timing Register (BTR) is only possible while the device is in Standby mode.

Figure 74. Bit timing



17.4 Register description

The CAN registers are organized as 6 general purpose registers plus 5 pages of 16 registers spanning the same address space and primarily used for message and filter storage. The page actually selected is defined by the content of the Page Selection Register.

17.4.1 General purpose registers

Interrupt status register (ISR)

ISR Reset value: 00h

7	6	5	4	3	2	1	0
RXIF3	RXIF2	RXIF1	TXIF	SCIF	ORIF	TEIF	EPND
RC	RC	RC	RC	RC	RC	RC	RO

Table 91. ISR register description

Bit	Name	Function
7	RXIF3	*Receive Interrupt Flag for Buffer 3* Set by hardware to signal that a new error-free message is available in buffer 3. Cleared by software to release buffer 3. Also cleared by resetting bit RDY of BCSR3.

18.3 Functional description

The conversion is monotonic, meaning that the result never decreases if the analog input does not and never increases if the analog input does not.

If the input voltage (V_{AIN}) is greater than V_{AREF} (high-level voltage reference) then the conversion result is FFh in the ADCDRH register and 03h in the ADCDRL register (without overflow indication).

If the input voltage (V_{AIN}) is lower than V_{SSA} (low-level voltage reference) then the conversion result in the ADCDRH and ADCDRL registers is 00 00h.

The A/D converter is linear and the digital result of the conversion is stored in the ADCDRH and ADCDRL registers. The accuracy of the conversion is described in *Section 20: Electrical characteristics*.

R_{AIN} is the maximum recommended impedance for an analog input signal. If the impedance is too high, this will result in a loss of accuracy due to leakage and sampling not being completed in the allotted time.

18.3.1 A/D converter configuration

The analog input ports must be configured as input, no pull-up, no interrupt. Refer to the *Chapter 9: I/O ports*. 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.

18.3.2 Starting the conversion

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 or a write to any bit of the ADCCSR register resets the EOC bit.

To read the 10 bits, perform the following steps:

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

Note: *The data is not latched, so both the low and the high data register must be read before the next conversion is complete, so it is recommended to disable interrupts while reading the conversion result.*

To read only 8 bits, perform the following steps:

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

19.1.5 Indirect (short, long)

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

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

Indirect (short)

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

Indirect (long)

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

19.1.6 Indirect indexed (short, long)

This is a combination of indirect and short indexed addressing modes. The operand is referenced by its memory address, which is defined by the unsigned addition of an index register value (X or Y) with a pointer value located in memory. The pointer address follows the opcode.

The indirect indexed addressing mode consists of two submodes:

Indirect indexed (short)

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

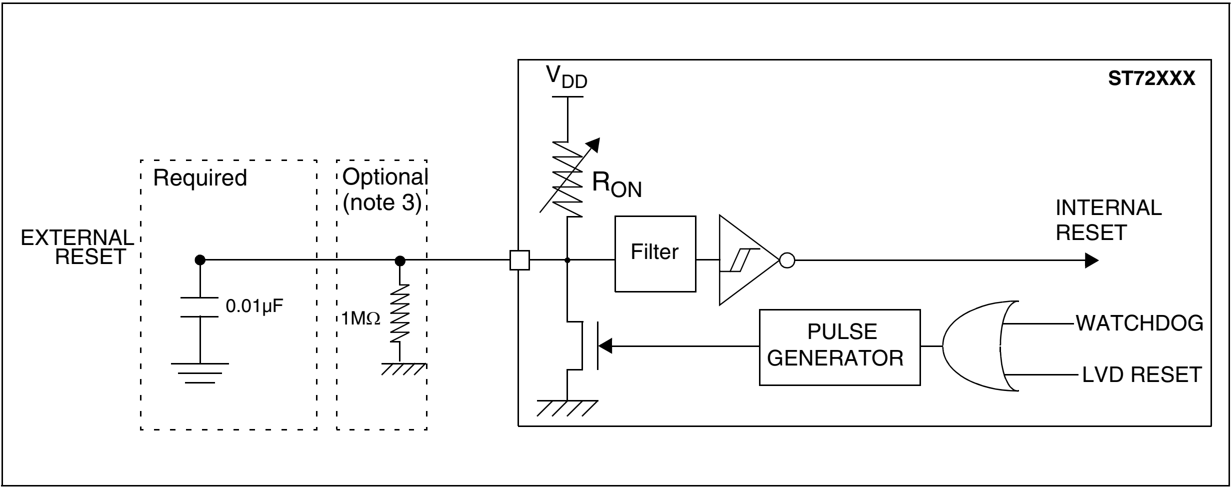
Indirect indexed (long)

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

Table 120. Instructions supporting direct, indexed, indirect, and indirect indexed addressing modes

Type	Instruction	Function
Long and short instructions	LD	Load
	CP	Compare
	AND, OR, XOR	Logical operations
	ADC, ADD, SUB, SBC	Arithmetic Additions/Subtractions operations
	BCP	Bit Compare

Figure 105. $\overline{\text{RESET}}$ pin protection when LVD is enabled



Note: 1 *The reset network protects the device against parasitic resets.*

The output of the external reset circuit must have an open-drain output to drive the ST7 reset pad. Otherwise the device can be damaged when the ST7 generates an internal reset (LVD or watchdog).

Whether the reset source is internal or external, the user must ensure that the level on the $\overline{\text{RESET}}$ pin can go below the V_{IL} maximum level specified in Section 20.9.1 on page 240. Otherwise the reset will not be taken into account internally.

Because the reset circuit is designed to allow the internal RESET to be output in the $\overline{\text{RESET}}$ pin, the user must ensure that the current sunk on the $\overline{\text{RESET}}$ pin is less than the absolute maximum value specified for $I_{INJ(RESET)}$ in Section 20.2.2 on page 220.

2 *When the LVD is enabled, it is recommended not to connect a pull-up resistor or capacitor. A 10nF pull-down capacitor is required to filter noise on the reset line.*

3 *In case a capacitive power supply is used, it is recommended to connect a 1MΩ pull-down resistor to the $\overline{\text{RESET}}$ pin to discharge any residual voltage induced by the capacitive effect of the power supply (this will add 5µA to the power consumption of the MCU).*

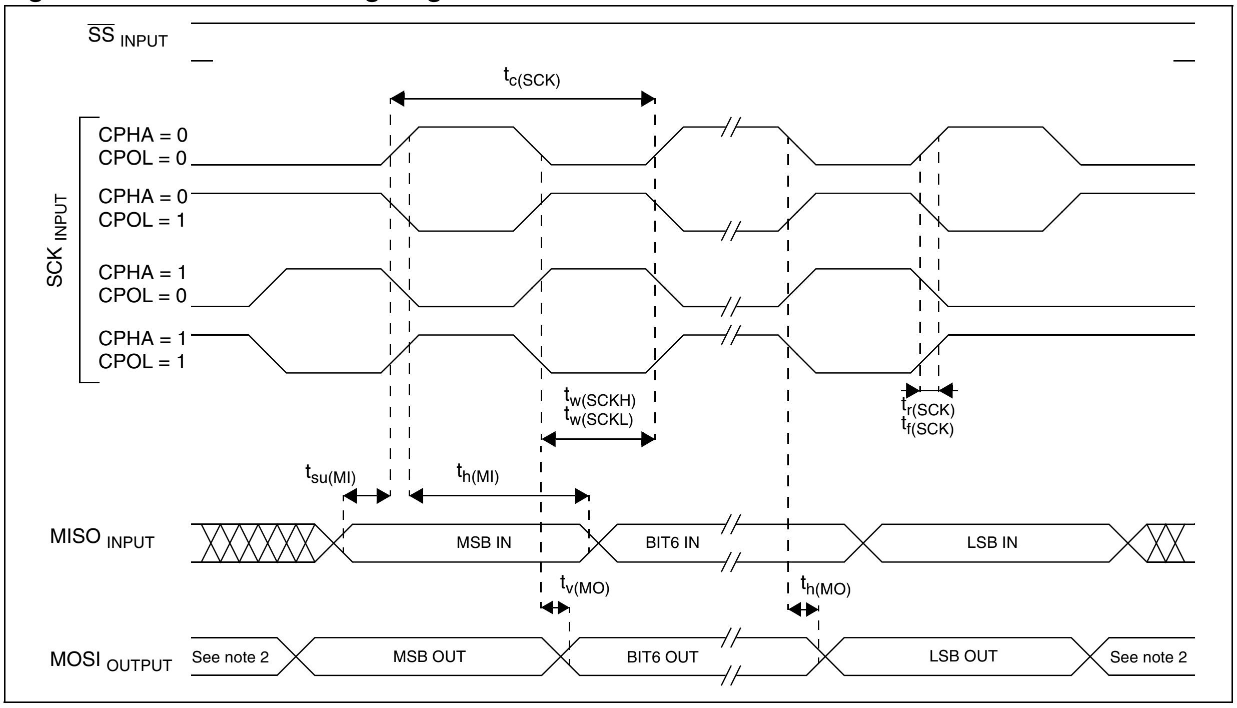
4 *Tips when using the LVD:*

A. Check that all recommendations related to reset circuit have been applied (see notes above).

B. Check that the power supply is properly decoupled (100nF + 10µF close to the MCU). Refer to AN1709 and AN2017. If this cannot be done, it is recommended to put a 100nF + 1MΩ pull-down on the $\overline{\text{RESET}}$ pin.

C. The capacitors connected on the $\overline{\text{RESET}}$ pin and also the power supply are key to avoid any start-up marginality. In most cases, steps A and B above are sufficient for a robust solution. Otherwise, replace 10nF pull-down on the $\overline{\text{RESET}}$ pin with a 5µF to 20µF capacitor.

Figure 110. SPI master timing diagram[1]



1. Measurement points are done at CMOS levels: $0.3xV_{DD}$ and $0.7xV_{DD}$.
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 of the I/O port configuration.

20.11.2 I^2C - inter IC control interface

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

Refer to *Section 20.8: 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 153. I^2C control interface characteristics

Symbol	Parameter	Standard mode I^2C		Fast mode I^2C[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		µs
$t_{su(SDA)}$	SDA setup time	250		100		ns
$t_{h(SDA)}$	SDA data hold time	0[3]		0[4]	900[3]	ns
$t_{r(SDA)}$ $t_{r(SCL)}$	SDA and SCL rise time		1000	$20+0.1C_b$	300	ns
$t_{f(SDA)}$ $t_{f(SCL)}$	SDA and SCL fall time		300	$20+0.1C_b$	300	ns
$t_{h(STA)}$	START condition hold time	4.0		0.6		µs
$t_{su(STA)}$	Repeated START condition setup time	4.7		0.6		µs
$t_{su(STO)}$	STOP condition setup time	4.0		0.6		µs
$t_{w(STO:STA)}$	STOP to START condition time (bus free)	4.7		1.3		µs
C_b	Capacitive load for each bus line		400		400	pF

1. At 4 MHz f_{CPU}, maximum I^2C speed (400 kHz) is not achievable. In this case, maximum I^2C speed will be approximately 260 kHz.
2. Data based on standard I^2C 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.

Figure 119. 64-pin (10x10) low profile quad flat package outline

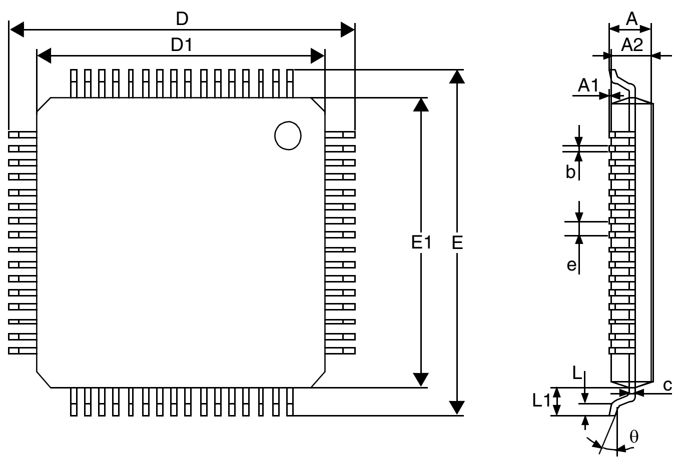


Table 160. 64-pin (10x10) low profile quad flat package mechanical data

Dimension	mm			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A			1.600			0.0630
A1	0.050		0.150	0.0020		0.0059
A2	1.350	1.400	1.450	0.0531	0.0551	0.0571
b	0.170	0.220	0.270	0.0067	0.0087	0.0106
c	0.090		0.200	0.0035		0.0079
D		12.000			0.4724	
D1		10.000			0.3937	
E		12.000			0.4724	
E1		10.000			0.3937	
e		0.500			0.0197	
θ	0°	3.5°	7°	0°	3.5°	7°
L	0.450	0.600	0.750	0.0177	0.0236	0.0295
L1		1.000			0.0394	

1. Values in inches are converted from mm and rounded to 4 decimal digits.

```
; check the semaphore status if edge is detected
CP A,#01

jrne OUT

call call_routine
; call the interrupt routine
OUT:LD A,#00

LD sema,A

.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
```

Case 2: Writing to PxOR or PxDDR with global interrupts disabled:

```
SIM
; set the interrupt mask
LD A,PFDR

AND A,#$02

LD  X,A
; store the level before writing to PxOR/PxDDR
LD A,#$90

LD PFDDR,A
; Write into PFDDR
LD A,#$ff

LD PFOR,A
          ; Write to PFOR
LD A,PFDR

AND A,#$02

LD Y,A
; store the level after writing to PxOR/PxDDR
LD A,X
```

23.1.9 TIMD set simultaneously with OC interrupt

If the 16-bit timer is disabled at the same time the output compare event occurs, the output compare flag then gets locked and cannot be cleared before the timer is enabled again.

Impact on the application

If the output compare interrupt is enabled, then the output compare flag cannot be cleared in the timer interrupt routine. Consequently, the interrupt service routine is called repeatedly.

Workaround

Disable the timer interrupt before disabling the timer. Again while enabling, first enable the timer, then the timer interrupts.

- Perform the following to disable the timer:
  - TACR1 or TBCR1 = 0x00h; // Disable the compare interrupt
  - TACSR | or TBCSR | = 0x40; // Disable the timer
- Perform the following to enable the timer again:
  - TACSR & or TBCSR & = ~0x40; // Enable the timer
  - TACR1 or TBCR1 = 0x40; // Enable the compare interrupt

23.1.10 CAN cell limitations

Table 169. CAN cell limitations

Limitation(1)	Flash	ROM
Omitted SOF bit	x	x
CPU write access (more than one cycle) corrupts CAN frame	x	x
Unexpected Message transmission	x(2)	
Bus Off State not entered		x(3)
WKPS functionality		x(4)

1. For details see *Section 17.5: List of CAN cell limitations on page 196*
2. Software workaround possible using modified WKPS bit.
3. Limitation present on ROM Rev W and Rev Z. Not present in Flash and ROM Rev Y.
4. Functionality modified for Unexpected Message Transmission workaround in Flash.

Legend:

x = limitation present

23.1.11 I^{2}C multimaster

In multimaster configurations, if the ST7 I2C receives a START condition from another I2C master after the START bit is set in the I2CCR register and before the START condition is generated by the ST7 I2C, it may ignore the START condition from the other I2C master. In this case, the ST7 master will receive a NACK from the other device. On reception of the NACK, ST7 can send a restart and Slave address to re-initiate communication.