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	32
Program Memory Size	48KB (48K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1.5K 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	44-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st72f321j7ta

Table 7. Arithmetic management bits (continued)

Bit	Name	Function
1	Z	*Zero* This bit is set and cleared by hardware. This bit indicates that the result of the last arithmetic, logical or data manipulation is zero. 0: The result of the last operation is different from zero. 1: The result of the last operation is zero. This bit is accessed by the JREQ and JRNE test instructions.
0	C	*Carry/borrow* This bit is set and cleared by hardware and software. It indicates an overflow or an underflow has occurred during the last arithmetic operation. 0: No overflow or underflow has occurred. 1: An overflow or underflow has occurred. This bit is driven by the SCF and RCF instructions and tested by the JRC and JRNC instructions. It is also affected by the “bit test and branch”, shift and rotate instructions.

Table 8. Interrupt management bits

Bit	Name	Function
5	I1	*Interrupt Software Priority 1* The combination of the I1 and I0 bits gives the current interrupt software priority.
3	I0	*Interrupt Software Priority 0* The combination of the I1 and I0 bits gives the current interrupt software priority.

Table 9. Interrupt software priority selection

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

These two bits are set/cleared by hardware when entering in interrupt. The loaded value is given by the corresponding bits in the interrupt software priority registers (ISPRx). They can be also set/cleared by software with the RIM, SIM, IRET, HALT, WFI and PUSH/POP instructions.

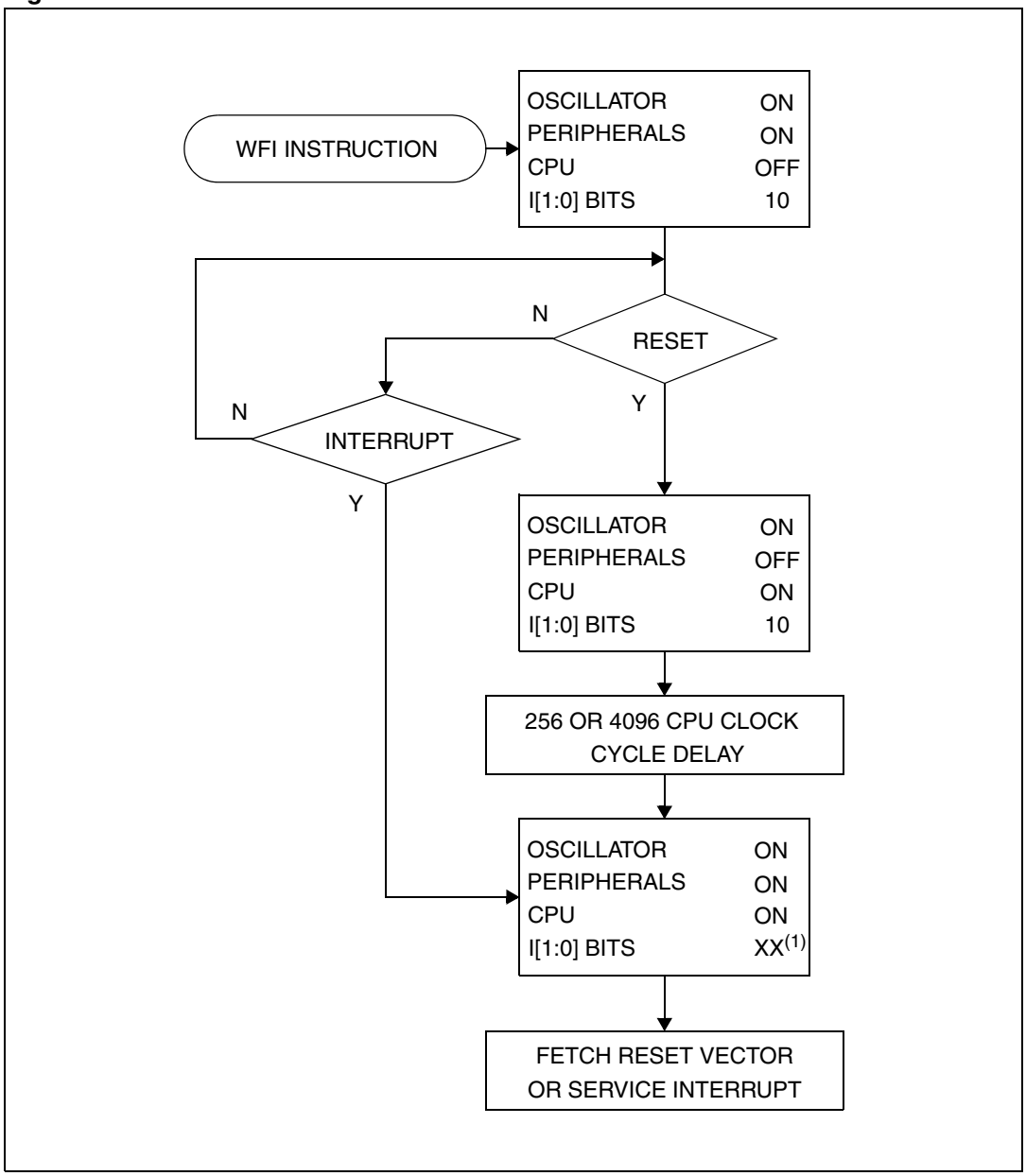
See *Chapter 7: Interrupts on page 49* for more details.

5.3.5 Stack pointer (SP) register

SP Reset value: 01 FFh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
								RW	RW	RW	RW	RW	RW	RW	RW

Figure 24. Wait mode flowchart



1. 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.

Table 31. I/O port configuration (continued)

Port	Pin name	Input (DDR = 0)		Output (DDR = 1)	
		OR = 0	OR = 1	OR = 0	OR = 1
Port B	PB7, PB3	floating	floating interrupt	open-drain	push-pull
	PB6:5, PB4, PB2:0	floating	pull-up interrupt	open-drain	push-pull
Port C	PC7:0	floating	pull-up	open-drain	push-pull
Port D	PD7:0	floating	pull-up	open-drain	push-pull
Port E	PE7:3, PE1:0	floating	pull-up	open drain	push-pull
	PE2 (Flash devices)	pull-up input only			
	PE2 (ROM devices)	floating		open drain	push-pull
Port F	PF7:3	floating	pull-up	open-drain	push-pull
	PF2	floating	floating interrupt	open-drain	push-pull
	PF1:0	floating	pull-up interrupt	open-drain	push-pull

9.4 Low power modes

Table 32. Effect of low power modes on I/O ports

Mode	Effect
Wait	No effect on I/O ports. External interrupts cause the device to exit from Wait mode.
Halt	No effect on I/O ports. External interrupts cause the device to exit from Halt mode.

9.5 Interrupts

The external interrupt event generates an interrupt if the corresponding configuration is selected with DDR and OR registers and the interrupt mask in the CC register is not active (RIM instruction).

Table 33. I/O port interrupt control/wake-up capability

Interrupt event	Event flag	Enable control bit	Exit from Wait	Exit from Halt
External interrupt on selected external event	-	DDRx, ORx	Yes	Yes

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

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
0000h	PADR	MSB							LSB
0001h	PADDR								
0002h	PAOR								

Figure 33. Exact timeout duration (t_{min} and t_{max})

WHERE:

$t_{min0} = (LSB + 128) \times 64 \times t_{OSC2}$

$t_{max0} = 16384 \times t_{OSC2}$

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

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

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

TB1 bit (MCCSR reg.)	TB0 bit (MCCSR reg.)	Selected MCCSR timebase	MSB	LSB
0	0	2ms	4	59
0	1	4ms	8	53
1	0	10ms	20	35
1	1	25ms	49	54

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

IF $CNT < \left\lceil \frac{MSB}{4} \right\rceil$ **THEN** $t_{min} = t_{min0} + 16384 \times CNT \times t_{osc2}$

ELSE $t_{min} = t_{min0} + \left[16384 \times \left(CNT - \left\lceil \frac{4CNT}{MSB} \right\rceil \right) + (192 + LSB) \times 64 \times \left\lceil \frac{4CNT}{MSB} \right\rceil \right] \times t_{osc2}$

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

IF $CNT \leq \left\lceil \frac{MSB}{4} \right\rceil$ **THEN** $t_{max} = t_{max0} + 16384 \times CNT \times t_{osc2}$

ELSE $t_{max} = t_{max0} + \left[16384 \times \left(CNT - \left\lceil \frac{4CNT}{MSB} \right\rceil \right) + (192 + LSB) \times 64 \times \left\lceil \frac{4CNT}{MSB} \right\rceil \right] \times t_{osc2}$

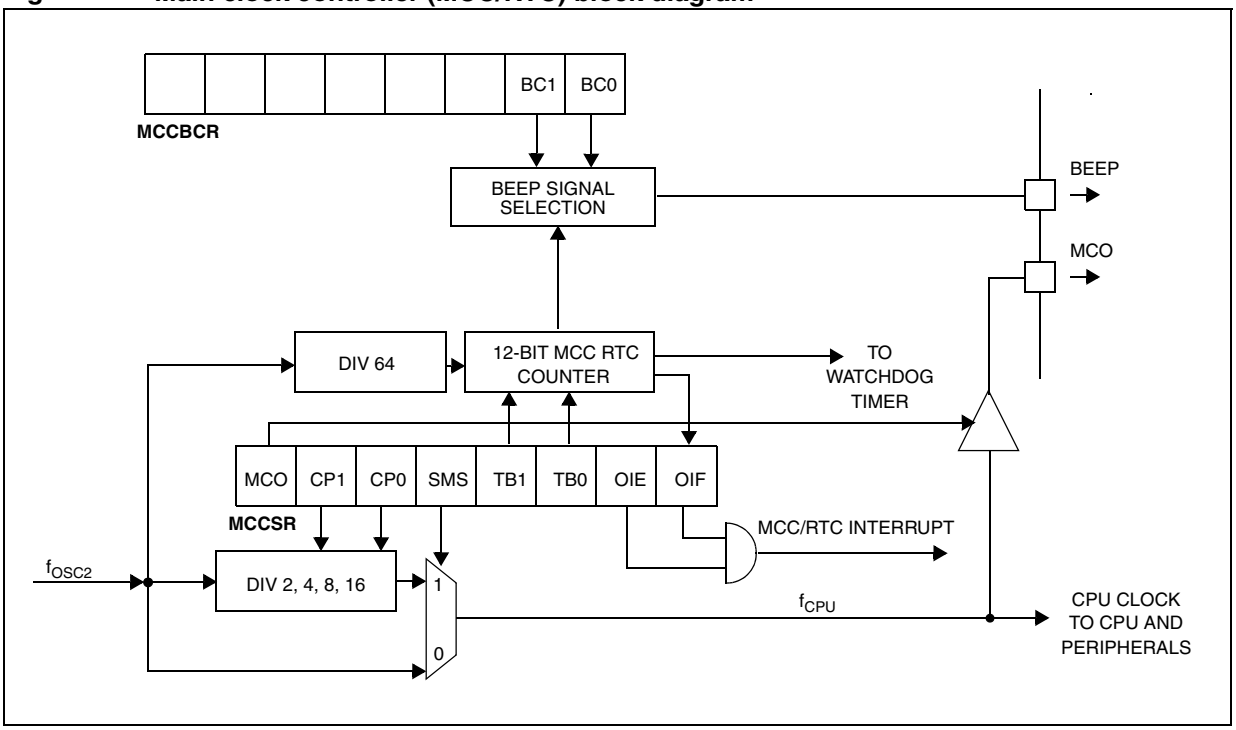
Note: In the above formulae, division results must be rounded down to the next integer value.

Example:

With 2ms timeout selected in MCCSR register

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

Figure 34. Main clock controller (MCC/RTC) block diagram



11.6 Low power modes

Table 38. Effect of low power modes on MCC/RTC

Mode	Effect
Wait	No effect on MCC/RTC peripheral. MCC/RTC interrupt causes 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 causes the device to exit from Active Halt mode.
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.

11.7 Interrupts

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

Table 39. MCC/RTC interrupt control/wake-up capability

Interrupt event	Event flag	Enable control bit	Exit from Wait	Exit from Halt
Time base overflow event	OIF	OIE	Yes	No(1)

1. The MCC/RTC interrupt wakes up the MCU from Active Halt mode, not from Halt mode.

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* This bit enables the MCO alternate function on the PF0 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_{CPU} on I/O port) *Note: To reduce power consumption, the MCO function is not active in Active Halt mode.*
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. 00: f_{CPU} in Slow mode = $f_{OSC2}/2$ 01: f_{CPU} in Slow mode = $f_{OSC2}/4$ 10: f_{CPU} in Slow mode = $f_{OSC2}/8$ 11: f_{CPU} in Slow mode = $f_{OSC2}/16$
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 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* These bits select the programmable divider time base. They are set and cleared by software (see *Table 41*). 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* 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.

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

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

$$OC\mathit{i}R \text{ value} = \frac{t * f_{CPU} - 5}{PRESC}$$

Where:

t = Pulse period (in seconds)

f_{CPU} = CPU clock frequency (in hertz)

PRESC = Timer prescaler factor (2, 4 or 8 depending on the CC[1:0] bits; see *Table 61: Timer clock selection*)

If the timer clock is an external clock the formula is:

$$OC\mathit{i}R = t * f_{EXT} - 5$$

Where:

t = Pulse period (in seconds)

f_{EXT} = External clock frequency (in hertz)

When the value of the counter is equal to the value of the contents of the OC1R register, the OLVL1 bit is output on the OCMP1 pin (see *Figure 52*).

Note: 1 The OCF1 bit cannot be set by hardware in one pulse mode but the OCF2 bit can generate an Output Compare interrupt.

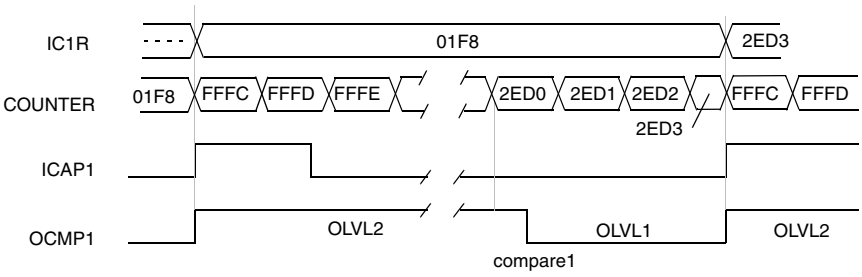
2 When the Pulse Width Modulation (PWM) and One Pulse Mode (OPM) bits are both set, the PWM mode is the only active one.

3 If OLVL1 = OLVL2 a continuous signal will be seen on the OCMP1 pin.

4 The ICAP1 pin cannot be used to perform input capture. The ICAP2 pin can be used to perform input capture (ICF2 can be set and IC2R can be loaded) but the user must take care that the counter is reset each time a valid edge occurs on the ICAP1 pin and ICF1 can also generates interrupt if ICIE is set.

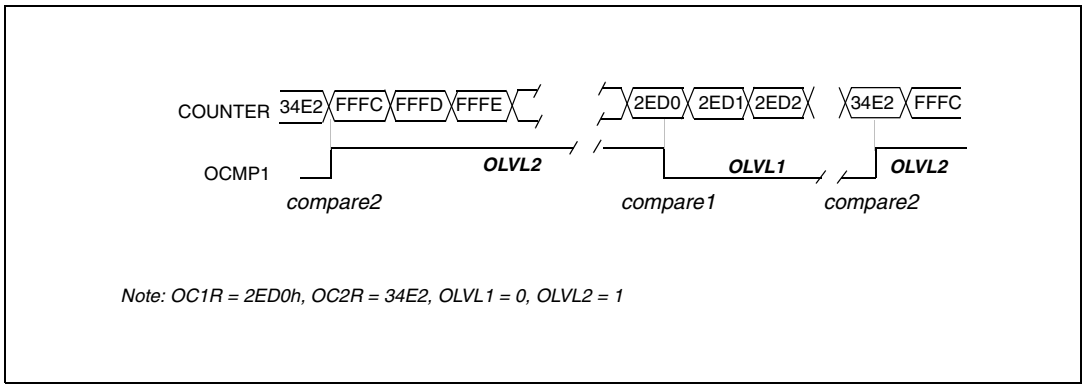
5 When one pulse mode is used OC1R is dedicated to this mode. Nevertheless OC2R and OCF2 can be used to indicate a period of time has been elapsed but cannot generate an output waveform because the level OLVL2 is dedicated to the one pulse mode.

Figure 52. One pulse mode timing example



Note: IEDG1 = 1, OC1R = 2ED0h, OLVL1 = 0, OLVL2 = 1

Figure 53. Pulse width modulation mode timing example with 2 output compare functions



Note: On timers with only one Output Compare register, a fixed frequency PWM signal can be generated using the output compare and the counter overflow to define the pulse length.

13.3.7 Pulse width modulation mode

Pulse Width Modulation (PWM) mode enables the generation of a signal with a frequency and pulse length determined by the value of the OC1R and OC2R registers.

Pulse Width Modulation mode uses the complete Output Compare 1 function plus the OC2R register, and so this functionality cannot be used when PWM mode is activated.

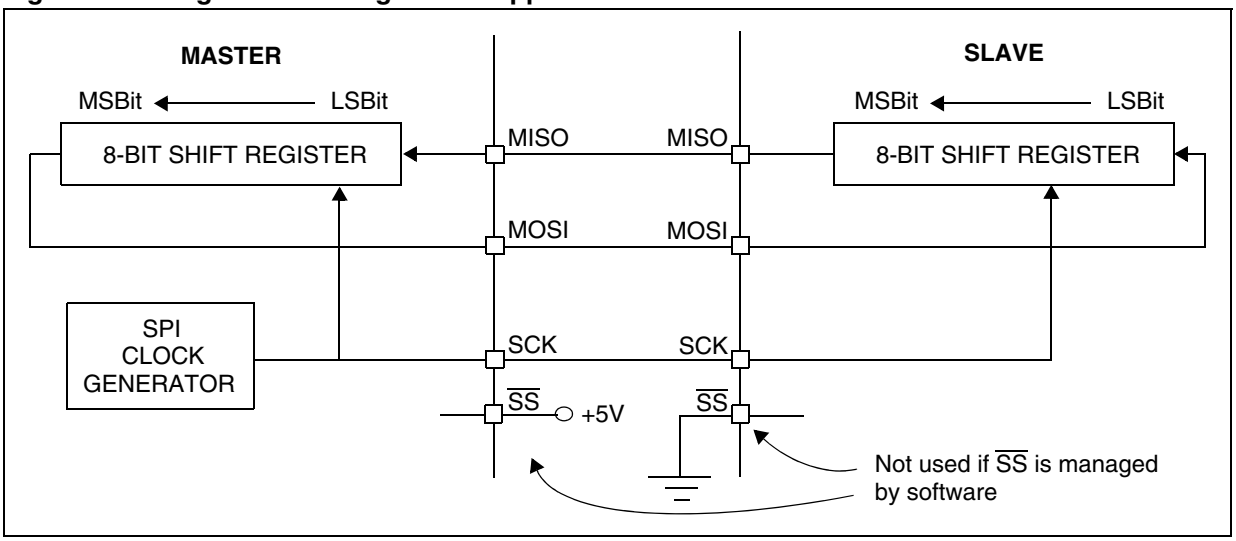
In PWM mode, double buffering is implemented on the output compare registers. Any new values written in the OC1R and OC2R registers are taken into account only at the end of the PWM period (OC2) to avoid spikes on the PWM output pin (OCMP1).

Procedure

To use pulse width modulation mode:

1. Load the OC2R register with the value corresponding to the period of the signal using the appropriate formula below according to the timer clock source used.
2. Load the OC1R register with the value corresponding to the period of the pulse if OLVL1 = 0 and OLVL2 = 1 using the appropriate formula below according to the timer clock source used.
3. Select the following in the CR1 register:
   - Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after a successful comparison with the OC1R register.
   - Using the OLVL2 bit, select the level to be applied to the OCMP1 pin after a successful comparison with the OC2R register.
4. Select the following in the CR2 register:
   - Set OC1E bit: the OCMP1 pin is then dedicated to the output compare 1 function.
   - Set the PWM bit.
   - Select the timer clock (CC[1:0]) (see *Table 61: Timer clock selection*).

Figure 56. Single master/single slave application



14.3.2 Slave select management

As an alternative to using the $\overline{SS}$ pin to control the Slave Select signal, the application can choose to manage the Slave Select signal by software. This is configured by the SSM bit in the SPICSR register (see *Figure 58*)

In software management, the external $\overline{SS}$ pin is free for other application uses and the internal $\overline{SS}$ signal level is driven by writing to the SSI bit in the SPICSR register.

In Master mode

- $\overline{SS}$ internal must be held high continuously

In Slave mode

There are two cases depending on the data/clock timing relationship (see *Figure 57*):

If CPHA = 1 (data latched on 2nd clock edge):

- $\overline{SS}$ internal must be held low during the entire transmission. This implies that in single slave applications the $\overline{SS}$ pin either can be tied to V_{SS}, or made free for standard I/O by managing the $\overline{SS}$ function by software (SSM = 1 and SSI = 0 in the in the SPICSR register)

If CPHA = 0 (data latched on 1st clock edge):

- $\overline{SS}$ internal must be held low during byte transmission and pulled high between each byte to allow the slave to write to the shift register. If $\overline{SS}$ is not pulled high, a Write Collision error will occur when the slave writes to the shift register (see *Write collision error (WCOL) on page 128*).

Warning: A write to the SPIDR register places data directly into the shift register for transmission.

A read to the SPIDR register returns the value located in the buffer and not the content of the shift register (see *Figure 55*).

Table 69. SPI register map and reset values

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0021h	SPIDR Reset value	MSB x	x	x	x	x	x	x	LSB x
0022h	SPICR Reset value	SPIE 0	SPE 0	SPR2 0	MSTR 0	CPOL x	CPHA x	SPR1 x	SPR0 x
0023h	SPICSR Reset value	SPIF 0	WCOL 0	OVR 0	MODF 0	0	SOD 0	SSM 0	SSI 0

15.7.2 Control register 1 (SCICR1)

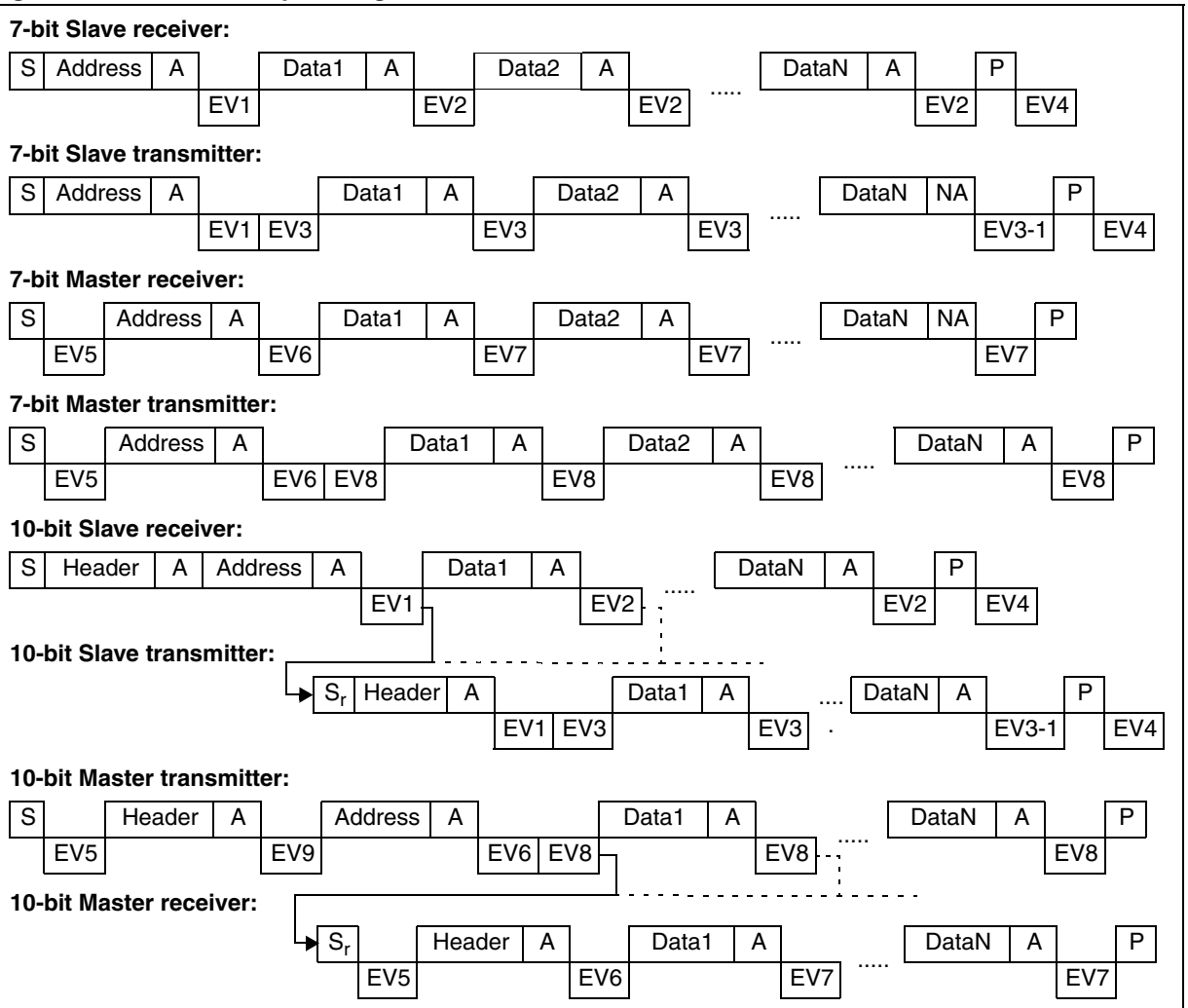
SCICR1 Reset value: X000 0000 (x0h)

7	6	5	4	3	2	1	0
R8	T8	SCID	M	WAKE	PCE	PS	PIE
RW	RW	RW	RW	RW	RW	RW	RW

Table 74. SCICR1 register description

Bit	Name	Function
7	R8	*Receive data bit 8* This bit is used to store the 9th bit of the received word when M = 1.
6	T8	*Transmit data bit 8* This bit is used to store the 9th bit of the transmitted word when M = 1.
5	SCID	*Disabled for low power consumption* When this bit is set the SCI prescalers and outputs are stopped and the end of the current byte transfer in order to reduce power consumption.This bit is set and cleared by software. 0: SCI enabled 1: SCI prescaler and outputs disabled
4	M	*Word length* This bit determines the word length. It is set or cleared by software. 0: 1 Start bit, 8 Data bits, 1 Stop bit 1: 1 Start bit, 9 Data bits, 1 Stop bit *Note: The M bit must not be modified during a data transfer (both transmission and reception).*
3	WAKE	*Wake-up method* This bit determines the SCI wake-up method. It is set or cleared by software. 0: Idle line 1: Address mark
2	PCE	*Parity control enable* This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission). 0: Parity control disabled 1: Parity control enabled
1	PS	*Parity selection* This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte. 0: Even parity 1: Odd parity

Figure 68. Transfer sequencing



Legend:

S = Start, Sr = Repeated Start, P = Stop, A = Acknowledge, NA = Non-acknowledge, EVx = Event (with interrupt if ITE = 1)

EV1: EVF = 1, ADSL = 1, cleared by reading SR1 register.

EV2: EVF = 1, BTF = 1, cleared by reading SR1 register followed by reading DR register.

EV3: EVF = 1, BTF = 1, cleared by reading SR1 register followed by writing DR register.

EV3-1: EVF = 1, AF = 1, BTF = 1; AF is cleared by reading SR1 register. BTF is cleared by releasing the lines (STOP = 1, STOP = 0) or by writing DR register (DR = FFh). ***Note:*** *If lines are released by STOP = 1, STOP = 0, the subsequent EV4 is not seen.*

EV4: EVF = 1, STOPF = 1, cleared by reading SR2 register.

EV5: EVF = 1, SB = 1, cleared by reading SR1 register followed by writing DR register.

EV6: EVF = 1, cleared by reading SR1 register followed by writing CR register (for example PE = 1).

EV7: EVF = 1, BTF = 1, cleared by reading SR1 register followed by reading DR register.

EV8: EVF = 1, BTF = 1, cleared by reading SR1 register followed by writing DR register.

EV9: EVF = 1, ADD10 = 1, cleared by reading SR1 register followed by writing DR register.

19.8 I/O port pin characteristics

19.8.1 General characteristics

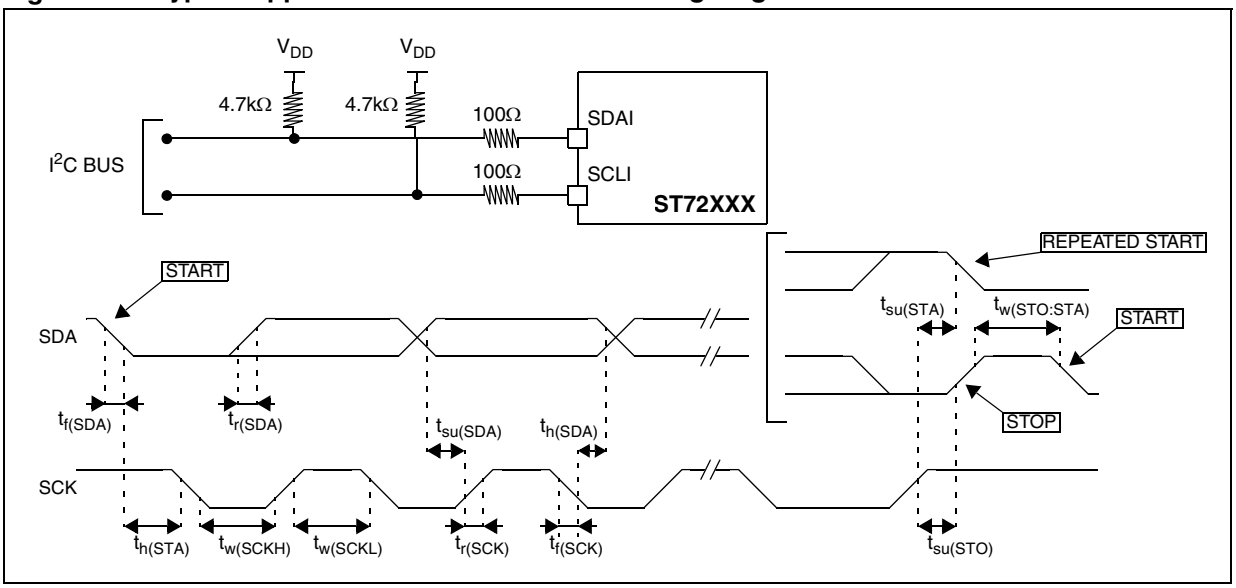
Subject to general operating conditions for V_{DD}, f_{OSC}, and T_A unless otherwise specified.

Table 126. I/O port pin general characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IL}	Input low level voltage(1)	CMOS ports			$0.3xV_{DD}$	V
V_{IH}	Input high level voltage(1)		$0.7xV_{DD}$			
V_{hys}	Schmitt trigger voltage hysteresis(2)			0.7		
$I_{INJ(PIN)}$(3)	Injected current on PC6 pin (Flash devices only)	V_{DD} = 5V	0		+4	mA
	Injected current on an I/O pin				±4	
$\Sigma I_{INJ(PIN)}$(3)	Total injected current (sum of all I/O and control pins)				±25	
I_L	Input leakage current	$V_{SS} \le V_{IN} \le V_{DD}$			±1	µA
I_S	Static current consumption	Floating input mode(4)		400		
R_{PU}	Weak pull-up equivalent resistor(5)	$V_{IN} = V_{SS}$ V_{DD} = 5V	50	120	250	kΩ
C_{IO}	I/O pin capacitance			5		pF
$t_{f(IO)out}$	Output high to low level fall time(1)	C_L = 50pF Between 10% and 90%		25		ns
$t_{r(IO)out}$	Output low to high level rise time(1)			25		
$t_{w(IT)in}$	External interrupt pulse time(6)		1			t_{CPU}

1. Data based on characterization results, not tested in production.
2. Hysteresis voltage between Schmitt trigger switching levels. Based on characterization results, not tested.
3. When the current limitation is not possible, the V_{IN} maximum must be respected, otherwise refer to $I_{INJ(PIN)}$ specification. A positive injection is induced by $V_{IN} > V_{DD}$ while a negative injection is induced by $V_{IN} < V_{SS}$. Refer to *Section 19.2.2: Current characteristics* for more details.
4. Configuration not recommended. All unused pins must be kept at a fixed voltage. This can be done by using the output mode of the I/O, for example, and leaving the I/O unconnected on the board or by an external pull-up or pull-down resistor (see *Figure 82*). Static peak current value taken at a fixed V_{IN} value, based on design simulation and technology characteristics, not tested in production. This value depends on V_{DD} and temperature values.
5. The R_{PU} pull-up equivalent resistor is based on a resistive transistor (corresponding I_{PU} current characteristics described in *Figure 83*).
6. To generate an external interrupt, a minimum pulse width has to be applied on an I/O port pin configured as an external interrupt source.

Figure 96. Typical application with I²C BUS and timing diagram(1)



1. Measurement points are done at CMOS levels: $0.3xV_{DD}$ and $0.7xV_{DD}$.

The following table provides the values to be written in the I2CCCR register to obtain the required I²C SCL line frequency.

Table 134. SCL frequency table

f_{SCL} (kHz)	I2CCCR value							
	f_{CPU} = 4 MHz				f_{CPU} = 8 MHz			
	V_{DD} = 4.1V		V_{DD} = 5V		V_{DD} = 4.1V		V_{DD} = 5V	
	R_P = 3.3kΩ	R_P = 4.7kΩ	R_P = 3.3kΩ	R_P = 4.7kΩ	R_P = 3.3kΩ	R_P = 4.7kΩ	R_P = 3.3kΩ	R_P = 4.7kΩ
400	Not achievable				83h			
300	Not achievable				85h			
200	83h				8Ah	89h	8Ah	
100	10h				24h	23h	24h	23h
50	24h				4Ch			
20	5Fh				FFh			

Legend:

R_P = External pull-up resistance

f_{SCL} = I²C speed

Note: - For speeds around 200 kHz, the achieved speed can have a ±5% tolerance.
- For other speed ranges, the achieved speed can have a ±2% tolerance.

The above variations depend on the accuracy of the external components used.

20.1 Thermal characteristics

Table 139. Thermal characteristics

Symbol	Ratings	Value	Unit
R_{thJA}	Package thermal resistance (junction to ambient) LQFP64 14x14 LQFP64 10x10 LQFP44 10x10	47 50 52	°C/W
P_D	Power dissipation(1)	500	mW
T_{Jmax}	Maximum junction temperature(2)	150	°C

1. The maximum power dissipation is obtained from the formula $P_D = (T_J - T_A) / R_{thJA}$. The power dissipation of an application can be defined by the user with the formula: $P_D = P_{INT} + P_{PORT}$ where P_{INT} is the chip internal power (I_{DD} x V_{DD}) and P_{PORT} is the port power dissipation depending on the ports used in the application.
2. The maximum chip-junction temperature is based on technology characteristics.

20.2 Ecopack information

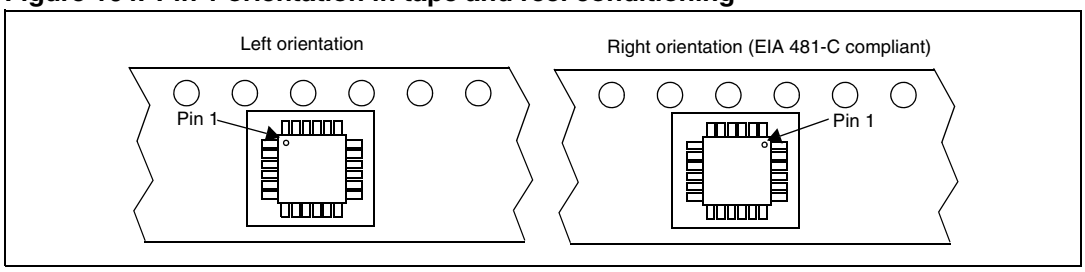
In order to meet environmental requirements, ST offers these devices in different grades of ECOPACK® packages, depending on their level of environmental compliance. ECOPACK® specifications, grade definitions and product status are available at: *www.st.com*. ECOPACK® is an ST trademark.

20.3 Packaging for automatic handling

The devices can be supplied in trays or with tape and reel conditioning.

Tape and reel conditioning can be ordered with pin 1 left-oriented or right-oriented when facing the tape sprocket holes as shown in *Figure 104*.

Figure 104. Pin 1 orientation in tape and reel conditioning



See also *Section Figure 105.: ST72F321xxx-Auto Flash commercial product structure on page 226* and *Figure 106: ST72P321xxx-Auto FastROM commercial product structure on page 228*.

```
; 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
```

22.1.6 Clearing active interrupts outside interrupt routine

When an active interrupt request occurs at the same time as the related flag is being cleared, an unwanted reset may occur.

Note: Clearing the related interrupt mask will not generate an unwanted reset.

Concurrent interrupt context

The symptom does not occur when the interrupts are handled normally, that is, when:

- The interrupt flag is cleared within its own interrupt routine
- The interrupt flag is cleared within any interrupt routine
- The interrupt flag is cleared in any part of the code while this interrupt is disabled

If these conditions are not met, the symptom can be avoided by implementing the following sequence:

Perform SIM and RIM operation before and after resetting an active interrupt request.

Example:

```
SIM
Reset interrupt flag
RIM
```

Nested interrupt context

The symptom does not occur when the interrupts are handled normally, that is, when:

- The interrupt flag is cleared within its own interrupt routine
- The interrupt flag is cleared within any interrupt routine with higher or identical priority level
- The interrupt flag is cleared in any part of the code while this interrupt is disabled

If these conditions are not met, the symptom can be avoided by implementing the following sequence:

```
PUSH CC
SIM
Reset interrupt flag
POP CC
```