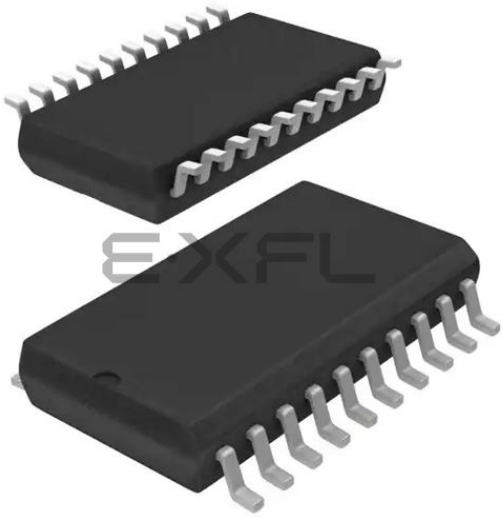


Welcome to E-XFL.COM

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
Peripherals	LVD, POR, PWM, WDT
Number of I/O	16
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	384 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 10x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SOIC (0.295", 7.50mm Width)
Supplier Device Package	20-SO
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st7foxf1m6

Table 2. Device pin description (32-pin packages) (continued)

Pin number		Pin name	Type	Level		Port/control						Main function (after reset)	Alternate function
						Input				Output			
LQFP32	SDIP32			Input	Output	float	wpu	int	ana	OD(1)	PP		
14	18	PB0/AIN0	I/O	C_T		**x**	ei1		x	x	x	**Port B0**	AIN0
15	19	PB1/AIN1/CLKIN	I/O	C_T		**x**			x	x	x	**Port B1**	AIN1/External clock source
16	20	PB2/AIN2	I/O	C_T		**x**			x	x	x	**Port B2**	AIN2
17	21	PB3/AIN3/MOSI(2)	I/O	C_T		**x**			x	x	x	**Port B3**	AIN3/SPI Master out /Slave in data
18	22	PB4/AIN4/MISO(2)	I/O	C_T		**x**			x	x	x	**Port B4**	AIN4/SPI Master in/Slave out data
19	23	PB5/AIN5/ EXTCLK_A(2)	I/O	C_T		**x**			x	x	x	**Port B5**	AIN5/Timer A input clock
20	24	PB6/AIN6/SCK(2)	I/O	C_T		**x**			x	x	x	**Port B6**	AIN6/SPI serial clock
21	25	PB7/AIN7/$\overline{SS}$(2)/ OCMP2_A(2)	I/O	C_T		**x**			x	x	x	**Port B7**	AIN7/SPI slave select (active low)/ Timer A Output Compare 2
22	26	PC0/AIN8/ ICAP1_A(2)	I/O	C_T		**x**	ei2		x	x	x	**Port C0**	AIN8/Timer A Input Capture 1
23	27	PC1/AIN9/ ICAP2_A(2)	I/O	C_T		**x**			x	x	x	**Port C1**	AIN9/Timer A Input Capture 2
24	28	PC2/ICCDATA	I/O	C_T		**x**				x	x	**Port C2**	ICCDATA
25	29	PC3/ICCCLK	I/O	C_T		x	**x**			x	x	**Port C3**	ICCCLK
26	30	PC4/LTIC	I/O	C_T		**x**	ei2			x	x	**Port C4**	LTIC
27	31	PC5/BREAK2(4)	I/O	C_T		**x**				x	x	**Port C5**	BREAK2
28	32	PC6	I/O	C_T		**x**				x	x	**Port C6**	
29	1	PC7/BREAK1	I/O	C_T		**x**				x	x	**Port C7**	BREAK1

4.5 Memory protection

There are two different types of memory protection: Read-Out Protection and Write/Erase Protection which can be applied individually.

4.5.1 Read-out protection

Read-Out Protection, when selected provides a protection against program memory content extraction and against write access to Flash memory. Even if no protection can be considered as totally unbreakable, the feature provides a very high level of protection for a general purpose microcontroller.

- In Flash devices, this protection is removed by reprogramming the option. In this case, the program memory is automatically erased and the device can be reprogrammed. The read-out protection is enabled and removed through the FMP_R bit in the option byte.

4.5.2 Flash write/erase protection

Write/Erase Protection, when set, makes it impossible to both overwrite and erase program memory. Its purpose is to provide advanced security to applications and prevent any change being made to the memory content. Write/Erase Protection is enabled through the FMP_W bit in the option byte.

Caution: **Once set, Write/Erase Protection can never be removed. A write-protected Flash device is no longer reprogrammable.**

4.6 Related documentation

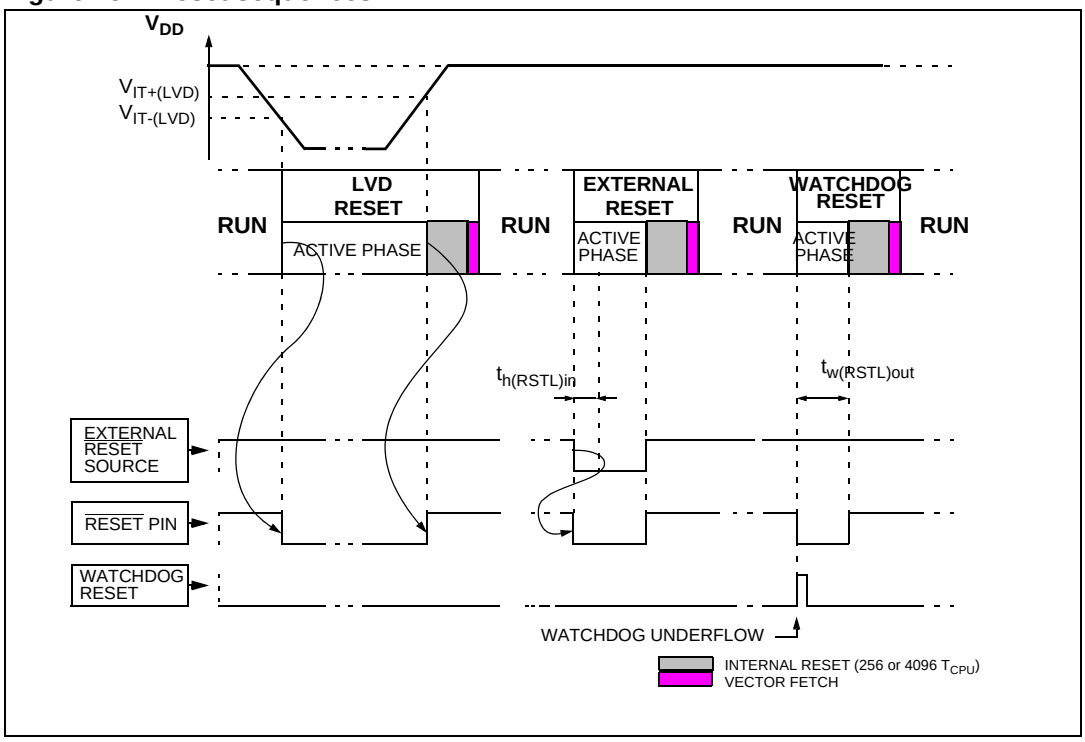
For details on Flash programming and ICC protocol, refer to the ST7 Flash Programming Reference Manual and to the ST7 ICC Protocol Reference Manual.

6.3.5 Internal watchdog reset

The Reset sequence generated by an internal watchdog counter overflow is shown in *Figure 15: Reset sequences*

Starting from the watchdog counter underflow, the device $\overline{\text{RESET}}$ pin acts as an output that is pulled low during at least $t_{w(RSTL)out}$.

Figure 15. Reset sequences



Halt mode recommendations

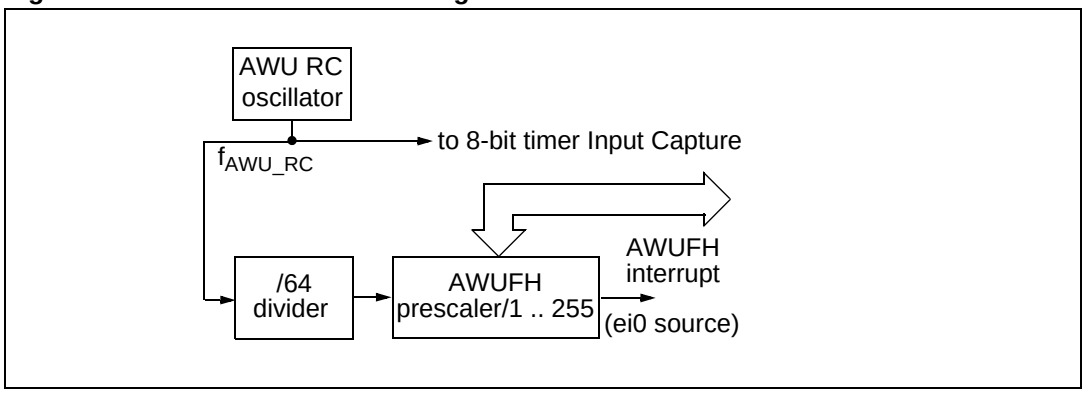
- Make sure that an external event is available to wake up the microcontroller from Halt mode.
- When using an external interrupt to wake up the microcontroller, reinitialize the corresponding I/O as “Input Pull-up with Interrupt” before executing the HALT instruction. The main reason for this is that the I/O may be wrongly configured due to external interference or by an unforeseen logical condition.
- For the same reason, reinitialize the level sensitiveness of each external interrupt as a precautionary measure.
- The opcode for the HALT instruction is 0x8E. To avoid an unexpected HALT instruction due to a Program Counter failure, it is advised to clear all occurrences of the data value 0x8E from memory. For example, avoid defining a constant in ROM with the value 0x8E.
- As the HALT instruction clears the I bit in the CC register to allow interrupts, the user may choose to clear all pending interrupt bits before executing the HALT instruction. This avoids entering other peripheral interrupt routines after executing the external interrupt routine corresponding to the wakeup event (reset or external interrupt).

8.5 Auto-wakeup from Halt mode

Auto wakeup from Halt (AWUFH) mode is similar to Halt mode with the addition of a specific internal RC oscillator for wakeup (Auto-wakeup from Halt oscillator) which replaces the main clock which was active before entering Halt mode. Compared to Active-Halt mode, AWUFH has lower power consumption (the main clock is not kept running), but there is no accurate real-time clock available.

It is entered by executing the HALT instruction when the AWUEN bit in the AWUCSR register has been set.

Figure 29. AWUFH mode block diagram



8.5.1 Register description

8.5.2 AWUFH Control/Status Register (AWUCSR)

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	0	AWUF	AWUM	AWUEN
Read/Write							

Bits 7:3 = Reserved

Bit 2 = **AWUF** *Auto wakeup flag*

This bit is set by hardware when the AWU module generates an interrupt and cleared by software on reading AWUCSR. Writing to this bit does not change its value.

0: No AWU interrupt occurred

1: AWU interrupt occurred

Bit 1 = **AWUM** *Auto wakeup Measurement bit*

This bit enables the AWU RC oscillator and connects its output to the Input Capture of the 8-bit Lite timer. This allows the timer to be used to measure the AWU RC oscillator dispersion and then compensate this dispersion by providing the right value in the AWUPRE register.

0: Measurement disabled

1: Measurement enabled

Bit 0 = **AWUEN** *Auto wakeup From Halt Enabled bit*

This bit enables the Auto wakeup from halt feature: once Halt mode is entered, the AWUFH wakes up the microcontroller after a time delay dependent on the AWU prescaler value. It is set and cleared by software.

0: AWUFH (Auto wakeup from Halt) mode disabled

1: AWUFH (Auto wakeup from Halt) mode enabled

Note: Whatever the clock source, this bit should be set to enable the AWUFH mode once the HALT instruction has been executed.

8.5.3 AWUFH prescaler register (AWUPR)

Reset value: 1111 1111 (FFh)

7							0
AWUPR7	AWUPR6	AWUPR5	AWUPR4	AWUPR3	AWUPR2	AWUPR1	AWUPR0
Read/Write							

Bits 7:0= **AWUPR[7:0]** *Auto wakeup Prescaler*

These 8 bits define the AWUPR Dividing factor (see *Table 21*).

Table 21. Configuring the dividing factor

AWUPR[7:0]	Dividing factor
00h	Forbidden
01h	1
...	...
FEh	254
FFh	255

In AWU mode, the time during which the MCU stays in Halt mode, t_{AWU}, is given by the equation below. See also *Figure 30 on page 68*.

$$t_{AWU} = 64 \times AWUPR \times \frac{1}{f_{AWURC}} + t_{RCSTRT}$$

The AWUPR prescaler register can be programmed to modify the time during which the MCU stays in Halt mode before waking up automatically.

Note: If 00h is written to AWUPR, the AWUPR remains unchanged.

Table 22. AWU register mapping and reset values

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0048h	**AWUCSR** Reset Value	0	0	0	0	0	AWUF	AWUM	AWUEN
0049h	**AWUPR** Reset Value	AWUPR7 1	AWUPR6 1	AWUPR5 1	AWUPR4 1	AWUPR3 1	AWUPR2 1	AWUPR1 1	AWUPR0 1

9 I/O ports

9.1 Introduction

The I/O ports allow data transfer. An I/O port can contain up to 8 pins. Each pin can be programmed independently either as a digital input or digital output. In addition, specific pins may have several other functions. These functions can include external interrupt, alternate signal input/output for on-chip peripherals or analog input.

9.2 Functional description

A Data register (DR) and a Data Direction register (DDR) are always associated with each port. The Option register (OR), which allows input/output options, may or may not be implemented. The following description takes into account the OR register. Refer to the Port Configuration table for device specific information.

An I/O pin is programmed using the corresponding bits in the DDR, DR and OR registers: bit x corresponding to pin x of the port.

Figure 32 shows the generic I/O block diagram.

9.2.1 Input modes

Clearing the DDRx bit selects input mode. In this mode, reading its DR bit returns the digital value from that I/O pin.

If an OR bit is available, different input modes can be configured by software: floating or pull-up. Refer to I/O Port Implementation section for configuration.

Note: 1 Writing to the DR modifies the latch value but does not change the state of the input pin.

2 Do not use read/modify/write instructions (BSET/BRES) to modify the DR register.

External interrupt function

Depending on the device, setting the ORx bit while in input mode can configure an I/O as an input with interrupt. In this configuration, a signal edge or level input on the I/O generates an interrupt request via the corresponding interrupt vector (eix).

Falling or rising edge sensitivity is programmed independently for each interrupt vector. The External Interrupt Control register (EICR) or the Miscellaneous register controls this sensitivity, depending on the device.

Each external interrupt vector is linked to a dedicated group of I/O port pins (see pinout description and interrupt section). If several I/O interrupt pins on the same interrupt vector are selected simultaneously, they are logically combined. For this reason if one of the interrupt pins is tied low, it may mask the others.

External interrupts are hardware interrupts. Fetching the corresponding interrupt vector automatically clears the request latch. Changing the sensitivity of a particular external interrupt clears this pending interrupt. This can be used to clear unwanted pending interrupts.

Spurious interrupts

When enabling/disabling an external interrupt by setting/resetting the related OR register bit, a spurious interrupt is generated if the pin level is low and its edge sensitivity includes falling/rising edge. This is due to the edge detector input which is switched to '1' when the external interrupt is disabled by the OR register.

To avoid this unwanted interrupt, a "safe" edge sensitivity (rising edge for enabling and falling edge for disabling) has to be selected before changing the OR register bit and configuring the appropriate sensitivity again.

Caution: In case a pin level change occurs during these operations (asynchronous signal input), as interrupts are generated according to the current sensitivity, it is advised to disable all interrupts before and to reenable them after the complete previous sequence in order to avoid an external interrupt occurring on the unwanted edge.

This corresponds to the following steps:

a) Set the interrupt mask with the SIM instruction (in cases where a pin level change could occur)
b) Select rising edge
c) Enable the external interrupt through the OR register
d) Select the desired sensitivity if different from rising edge
e) Reset the interrupt mask with the RIM instruction (in cases where a pin level change could occur)

2. To disable an external interrupt:

a) Set the interrupt mask with the SIM instruction SIM (in cases where a pin level change could occur)
b) Select falling edge
c) Disable the external interrupt through the OR register
d) Select rising edge
e) Reset the interrupt mask with the RIM instruction (in cases where a pin level change could occur)

9.2.2 Output modes

Setting the DDRx bit selects output mode. Writing to the DR bits applies a digital value to the I/O through the latch. Reading the DR bits returns the previously stored value.

If an OR bit is available, different output modes can be selected by software: push-pull or open-drain. Refer to I/O Port Implementation section for configuration.

Table 23. DR Value and output pin status

DR	Push-Pull	Open-Drain
0	V_{OL}	V_{OL}
1	V_{OH}	Floating

9.6 Interrupts

The external interrupt event generates an interrupt if the corresponding configuration is selected with DDR and OR registers and if the I bit in the CC register is cleared (RIM instruction).

Table 27. Description of interrupt events

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

See application notes AN1045 software implementation of I^2C bus master, and AN1048 - software LCD driver

9.7 Device-specific I/O port configuration

The I/O port register configurations are summarized in *Section 9.7.1: Standard ports* and *Section 9.7.2: Other ports*.

9.7.1 Standard ports

Table 28. PA5:0, PB7:0, PC7:4 and PC2:0 pins

Mode	DDR	OR
floating input	0	0
pull-up interrupt input	0	1
open drain output	1	0
push-pull output	1	1

9.7.2 Other ports

Table 29. PA7:6 pins

Mode	DDR	OR
floating input	0	0
interrupt input	0	1
open drain output	1	0
push-pull output	1	1

10.2.6 Register description

Timer control status register (ATCSR)

Reset value: 0x00 0000 (x0h)

7							0
0	ICF	ICIE	CK1	CK0	OVF1	OVFIE1	CMPIE
Read / Write							

Bit 7 = Reserved

Bit 6 = **ICF** *Input Capture flag*

This Bit is set by hardware and cleared by software by reading the ATICR register (a read access to ATICRH or ATICRL clears this flag). Writing to this bit does not change the bit value.

0: No input capture

1: An input capture has occurred

Bit 5 = **ICIE** *IC Interrupt Enable bit*

This bit is set and cleared by software.
0: Input Capture Interrupt Disabled

1: Input Capture Interrupt Enabled

Bits 4:3 = **CK[1:0]** *Counter Clock Selection bits*

These bits are set and cleared by software and cleared by hardware after a reset. they select the clock frequency of the counter.

Table 37. Counter clock selection

Counter clock selection	CK1	CK0
OFF	0	0
selection forbidden	1	1
f_{LTIMER} (1 ms timebase @ 8 MHz)	0	1
f_{CPU}	1	0

Bit 2 = **OVF1** *Overflow flag*

This bit is set by hardware and cleared by software by reading the ATCSR register. It indicates the transition of the Counter1 CNTR1 from FFFh to ATR1 value.

0: No Counter Overflow Occurred

1: Counter Overflow Occurred

Bit 1 = **OVFIE1** *Overflow Interrupt Enable bit*

This bit is read/write by software and cleared by hardware after a reset.

0: Overflow Interrupt Disabled.

1: Overflow Interrupt Enabled.

Timer Control register 2 (ATCSR2)

Reset value: 0000 0011 (03h)

7							0
FORCE2	FORCE1	ICS	OVFIE2	OVF2	ENCNTR2	TRAN2	TRAN1
Read/write							

Bit 7 = **FORCE2** *Force Counter 2 Overflow bit*

This bit is read/set by software. When set, it loads FFFh in the CNTR2 register. It is reset by hardware one CPU clock cycle after counter 2 overflow has occurred.

0 : No effect on CNTR2

1 : Loads FFFh in CNTR2

Note: This bit must not be reset by software

Bit 6 = **FORCE1** *Force Counter 1 Overflow bit*

This bit is read/set by software. When set, it loads FFFh in CNTR1 register. It is reset by hardware one CPU clock cycle after counter 1 overflow has occurred.

0 : No effect on CNTR1

1 : Loads FFFh in CNTR1

Note: This bit must not be reset by software

Bit 5 = **ICS** *Input Capture Shorted bit*

This bit is read/write by software. It allows the ATtimer CNTR1 to use the LTIC pin for long Input Capture.

0 : ATIC for CNTR1 Input Capture

1 : LTIC for CNTR1 Input Capture

Bit 4 = **OVFIE2** *Overflow interrupt 2 enable bit*

This bit is read/write by software and controls the overflow interrupt of counter2.

0: Overflow interrupt disabled.

1: Overflow interrupt enabled.

Bit 3 = **OVF2** *Overflow flag*

This bit is set by hardware and cleared by software by reading the ATCSR2 register. It indicates the transition of the counter2 from FFFh to ATR2 value.

0: No counter overflow occurred

1: Counter overflow occurred

Bit 2 = **ENCNTR2** *Enable counter2* for PWM2/3

This bit is read/write by software and switches the PWM2/3 operation to the CNTR2 counter. If this bit is set, PWM2/3 will be generated using CNTR2.

0: PWM2/3 is generated using CNTR1.

1: PWM2/3 is generated using CNTR2.

Note: Counter 2 gets frozen when the ENCNTR2 bit is reset. When ENCNTR2 is set again, the counter will restart from the last value.

10.3.5 Interrupts

Table 40. Description of interrupt events

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Active Halt	Exit from Halt
Timebase 1 Event	TB1F	TB1IE	Yes	Yes	No
Timebase 2 Event	TB2F	TB2IE		No	
IC Event	ICF	ICIE		No	

The TBxF and ICF interrupt events are connected to separate interrupt vectors (see *Section 7: Interrupts*).

They generate an interrupt if the enable bit is set in the LTCSR1 or LTCSR2 register and the interrupt mask in the CC register is reset (RIM instruction).

10.3.6 Register description

Lite Timer Control/Status register 2 (LTCSR2)

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	TB2IE	TB2F
Read / Write							

Bits 7:2 = Reserved, must be kept cleared.

Bit 1 = **TB2IE** *Timebase 2 Interrupt enable bit*

This bit is set and cleared by software.

0: Timebase (TB2) interrupt disabled

1: Timebase (TB2) interrupt enabled

Bit 0 = **TB2F** *Timebase 2 Interrupt flag*

This bit is set by hardware and cleared by software reading the LTCSR register. Writing to this bit has no effect.

0: No Counter 2 overflow

1: A Counter 2 overflow has occurred

Lite Timer Autoreload register (LTARR)

Reset value: 0000 0000 (00h)

7							0
AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
Read / Write							

Bits 7:0 = **AR[7:0]** *Counter 2 Reload value*

These bits register is read/write by software. The LTARR value is automatically loaded into Counter 2 (LTCNTR) when an overflow occurs.

Lite Timer Counter 2 (LTCNTR)

Reset value: 0000 0000 (00h)

7							0
CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
Read only							

Bits 7:0 = **CNT[7:0]** *Counter 2 Reload value*

This register is read by software. The LTARR value is automatically loaded into Counter 2 (LTCNTR) when an overflow occurs.

Lite Timer Control/status register (LTCSR1)

Reset value: 0x00 0000 (x0h)

7							0
ICIE	ICF	TB	TB1IE	TB1F			
Read / Write							

Bit 7 = **ICIE** *Interrupt Enable bit*

This bit is set and cleared by software.

0: Input Capture (IC) interrupt disabled

1: Input Capture (IC) interrupt enabled

Bit 6 = **ICF** *Input Capture flag*

This bit is set by hardware and cleared by software by reading the LTICR register. Writing to this bit does not change the bit value.

0: No Input Capture

1: An Input Capture has occurred

Note: After an MCU reset, software must initialize the ICF bit by reading the LTICR register

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

Equation 5

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

Where:

t = signal or pulse period (in seconds)

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

PRESC = timer prescaler factor (2, 4 or 8 depending on CC[1:0] bits, see *: Timer A Control register 2 (TACR2) on page 134*)

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

Equation 6

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

Where:

t = signal or pulse period (in seconds)

f_{EXT} = external timer clock frequency (in hertz)

The output compare 2 event causes the counter to be initialized to FFFCh (see *Figure 67*)

Note: 1 The OCF1 and OCF2 bits cannot be set by hardware in PWM mode therefore the output compare interrupt is inhibited.

2 The ICF1 bit is set by hardware when the counter reaches the OC2R value and can produce a timer interrupt if the ICIE bit is set and the I bit is cleared.

3 In PWM mode the ICAP1 pin can not be used to perform input capture because it is disconnected to the timer. 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 period and ICF1 can also generates interrupt if ICIE is set.

4 When the pulse width modulation (PWM) and one pulse mode (OPM) bits are both set, the PWM mode is the only active one.

10.4.4 Low power modes

Table 42. Effect of low power modes on 16-bit timer

Mode	Description
Wait	No effect on 16-bit timer. Timer interrupts cause the device to exit from Wait mode.
Halt	16-bit timer registers are frozen. In Halt mode, the counter stops counting until Halt mode is exited. Counting resumes from the previous count when the device is woken up by an interrupt with ‘exit from Halt mode’ capability or from the counter reset value when the device is woken up by a reset. If an input capture event occurs on the ICAP*i* pin, the input capture detection circuitry is armed. Consequently, when the device is woken up by an interrupt with ‘exit from Halt mode’ capability, the ICF*i* bit is set, and the counter value present when exiting from Halt mode is captured into the IC*i*R register.

subsequent EV4 is not seen.

6. **EV4:** EVF=1, STOPF=1, cleared by reading SR2 register.
7. **EV5:** EVF=1, SB=1, cleared by reading SR1 register followed by writing DR register.
8. **EV6:** EVF=1, cleared by reading SR1 register followed by writing CR register (for example PE=1).
9. **EV7:** EVF=1, BTF=1, cleared by reading SR1 register followed by reading DR register.
10. **EV8:** EVF=1, BTF=1, cleared by reading SR1 register followed by writing DR register.
11. **EV9:** EVF=1, ADD10=1, cleared by reading SR1 register followed by writing DR register.

10.5.5 Low power modes

Table 46. Effect of low power modes on the I²C interface

Mode	Description
Wait	No effect on I²C interface. I²C interrupts cause the device to exit from Wait mode.
Halt	I²C registers are frozen. In Halt mode, the I²C interface is inactive and does not acknowledge data on the bus. The I²C interface resumes operation when the MCU is woken up by an interrupt with “exit from Halt mode” capability.

10.5.6 Interrupts

Figure 72. Event flags and interrupt generation

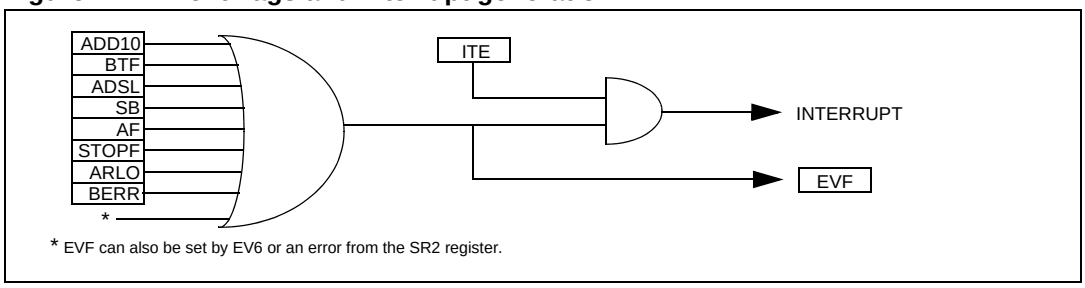


Table 47. Description of interrupt events

Interrupt Event(1)	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
10-bit Address Sent Event (Master mode)	ADD10	ITE	Yes	No
End of byte Transfer Event	BTF		Yes	No
Address Matched Event (Slave mode)	ADSL		Yes	No
Start Bit Generation Event (Master mode)	SB		Yes	No
Acknowledge Failure Event	AF		Yes	No
Stop Detection Event (Slave mode)	STOPF		Yes	No
Arbitration Lost Event (Multimaster configuration)	ARLO		Yes	No
Bus Error Event	BERR		Yes	No

1. The I²C interrupt events are connected to the same interrupt vector (see Interrupts chapter). They generate an interrupt if the corresponding Enable Control Bit is set and the I-bit in the CC register is reset (RIM instruction).

Configuring the A/D conversion

The analog input ports must be configured as input, no pull-up, no interrupt (see *Section 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.

To assign the analog channel to convert, select the CH[2:0] bits 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 ADCDRL
3. Read ADCDRH. This clears EOC automatically.

To read only 8 bits, perform the following steps:

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

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.

10.7.4 Low power modes

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.

Table 54. Effect of low power modes on the A/D converter

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.

10.7.5 Interrupts

None.

Data register High (ADCDRH)

Reset value: xxxx xxxx (xxh)

7							0
D9	D8	D7	D6	D5	D4	D3	D2
Read only							

Bits 7:0 = **D[9:2]** *MSB of Analog Converted Value*

ADC Control/data register Low (ADCDRL)

Reset value: 0000 00xx (0xh)

7							0
0	0	0	0	SLOW	0	D1	D0
Read/write							

Bits 7:4 = Reserved. Forced by hardware to 0.

Bit 3 = **SLOW** *Slow mode bit*

This bit is set and cleared by software. It is used together with the SPEED bit in the ADCCSR register to configure the ADC clock speed as shown on the table below.

Table 56. Configuring the ADC clock speed

f_{ADC}(1)	SLOW	SPEED
$f_{CPU}/2$	0	0
f_{CPU}	0	1
$f_{CPU}/4$	1	x

1. The maximum allowed value of f_{ADC} is 4 MHz (see *Section 12.11 on page 209*)

Bit 2 = Reserved. Forced by hardware to 0.

Bits 1:0 = **D[1:0]** *LSB of Analog Converted value*

Table 57. ADC register mapping and reset values

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0036h	**ADCCSR** Reset Value	EOC 0	SPEED 0	ADON 0	0 0	CH3 0	CH2 0	CH1 0	CH0 0
0037h	**ADCDRH** Reset Value	D9 x	D8 x	D7 x	D6 x	D5 x	D4 x	D3 x	D2 x
0038h	**ADCDRL** Reset Value	0 0	0 0	0 0	0	SLOW 0	0	D1 x	D0 x

12.3.3 Internal RC oscillator

To improve clock stability and frequency accuracy, it is recommended to place a decoupling capacitor, typically 100 nF, between the V_{DD} and V_{SS} pins as close as possible to the ST7 device

Internal RC oscillator calibrated at 5.0 V

The ST7 internal clock can be supplied by an internal RC oscillator (selectable by option byte).

Table 71. Internal RC oscillator characteristics (5.0 V calibration)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{RC}	Internal RC oscillator frequency	RCCR = FF (reset value), T_A = 25 °C, V_{DD} = 5 V		5.5		MHz
		RCCR=RCCR0(1), T_A = 25 °C, V_{DD} = 5 V		8		
$f_{G(RC)}$	RC trimming granularity	T_A = 25 °C, V_{DD} = 5 V		6		kHz
ACC_{RC}	Accuracy of Internal RC oscillator with RCCR=RCCR0(1)	T_A = 25 °C, V_{DD} = 5 V (2) without user calibration		±7		%
		T_A = 25 °C, V_{DD} = 4.5 to 5.5 V (2) with user calibration	-2		2	%
		T_A= 0 to +85 °C, V_{DD} = 4.5 to 5.5 V(2) with user calibration	-2.5		4	%
		T_A = -40 to 0 °C, V_{DD} = 4.5 to 5.5 V(2) with user calibration	-4		2.5	%
$t_{su(RC)}$	RC oscillator setup time	T_A = 25 °C, V_{DD} = 5 V		4 (3)		µs

1. See *Section 6.1.1: Internal RC oscillator*
2. Guaranteed by characterization
3. Not tested in production

Table 95. Development tool order codes

MCU	Debugging and programming tool	ST socket boards
ST7FOXF1 ST7FOXK1 ST7FOXK2	STX-RLINK(1)(2), ST7-STICK(3)(4), STice emulator(5)	SBX-SO20BE, SBX-DI8-20ZZ, SBX-DIP32CD, and SBX-QP32BC socket boards (3)

1. USB connection to PC.
2. Available from ST or from Raisonance, www.raisonance.com.
3. Add suffix /EU, /UK or /US for the power supply for your region.
4. Parallel port connection to PC.
5. Contact local ST sales office for sales types.

13.4 ST7 application notes

Table 96. ST7 application notes

Identification	Description
Application examples	
AN1658	Serial numbering implementation
AN1720	managing the Read-Out Protection in Flash microcontrollers
AN1755	A high resolution/precision thermometer using ST7 and NE555
AN1756	Choosing a DALI implementation strategy with ST7DALI
AN1812	A high precision, low cost, single supply ADC for positive and negative input voltages
Example drivers	
AN 969	SCI communication between ST7 and PC
AN 970	SPI communication between ST7 and EEPROM
AN 971	I²C communication between ST7 and M24Cxx EEPROM
AN 972	ST7 software SPI master communication
AN 973	SCI software communication with a PC using ST72251 16-bit timer
AN 974	Real time clock with ST7 timer Output Compare
AN 976	Driving a buzzer through ST7 timer PWM function
AN 979	Driving an analog keyboard with the ST7 ADC
AN 980	ST7 keypad decoding techniques, implementing wakeup on keystroke
AN1017	Using the ST7 Universal Serial Bus microcontroller
AN1041	Using ST7 PWM signal to generate analog output (sinusoïd)
AN1042	ST7 routine for I²C Slave mode Management
AN1044	Multiple interrupt sources management for ST7 MCUs
AN1045	ST7 S/W implementation of I²C bus master