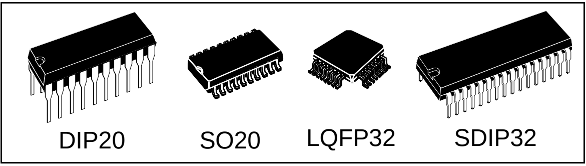
ST7FOXF1, ST7FOXK1, ST7FOXK2

8-bit MCU with single voltage Flash memory, SPI, I²C, ADC, timers

Features

- Memories
  - 4 to 8 Kbytes single voltage extended Flash (XFlash) Program memory with Read-Out Protection
    In-Circuit Programming and In-Application programming (ICP and IAP)
    Endurance: 1K write/erase cycles guaranteed
    Data retention: 20 years at 55 °C
  - 384 bytes RAM
- Clock, Reset and Supply Management
  - Low voltage supervisor (LVD) for safe power-on/off
  - Clock sources: Internal trimmable 8 MHz RC oscillator, auto wakeup internal low power - low frequency oscillator, crystal/ceramic resonator or external clock
  - External reset source and watchdog reset
  - Five power saving modes: Halt, Active-Halt, Auto Wakeup from Halt, Wait and Slow
- I/O Ports
  - Up to 24 multifunctional bidirectional I/Os
  - Up to 8 high sink outputs



- 6 timers
  - Configurable watchdog timer
  - Dual 8-bit Lite timers with prescaler, 1 real time base and 1 input capture
  - Dual 12-bit Auto-reload timers with 4 PWM outputs, input capture, output compare, dead-time generation and enhanced one pulse mode functions
  - One 16-bit timer
- Communication interfaces:
  - I²C multimaster interface
  - SPI synchronous serial interface
- A/D converter: up to 10 input channels
- Interrupt management
  - 13 interrupt vectors plus TRAP and RESET
- Instruction set
  - 8-bit data manipulation
  - 63 basic instructions with illegal opcode detection
  - 17 main addressing modes
  - 8 x 8 unsigned multiply instructions
- Development tools
  - Full HW/SW development package
  - DM (Debug Module)

Contents

List of tables

List of figures

1 Description

The ST7FOX is a member of the ST7 microcontroller family. All ST7 devices are based on a common industry-standard 8-bit core, featuring an enhanced instruction set.

The device is positioned at the entry level of the 8-bit microcontroller range providing an attractive cost while at the same time embedding the most advanced features.

The ST7FOX features Flash memory with byte-by-byte In-Circuit Programming (ICP) and In-Application Programming (IAP) capability.

Under software control, the ST7FOX device can be placed in Wait, Slow, or Halt mode, reducing power consumption when the application is in idle or standby state.

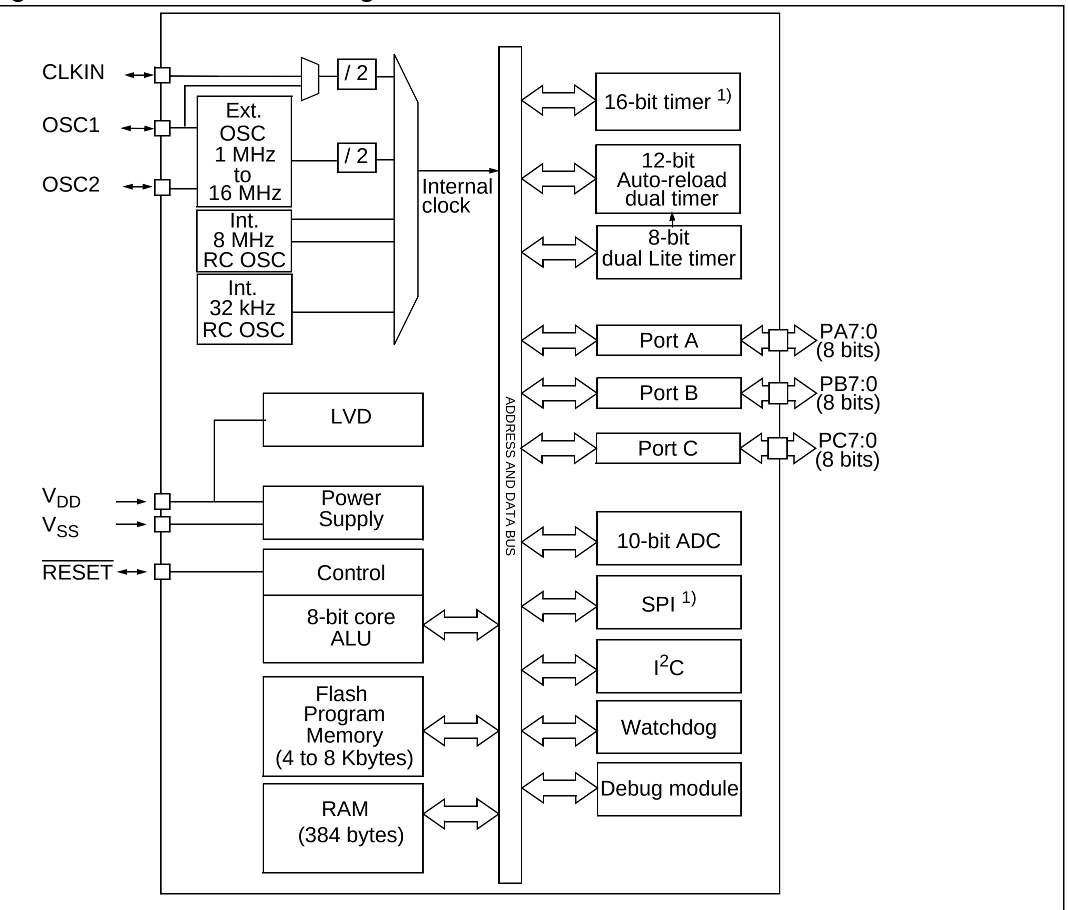
The enhanced instruction set and addressing modes of the ST7 offer both power and flexibility to software developers, enabling the design of highly efficient and compact application code. In addition to standard 8-bit data management, all ST7 microcontrollers feature true bit manipulation, 8x8 unsigned multiplication and indirect addressing modes.

The ST7FOX features an on-chip Debug Module (DM) to support In-Circuit Debugging (ICD). For a description of the DM registers, refer to the ST7 ICC Protocol Reference Manual.

Table 1. Device summary

Features	ST7FOXF1 / ST7FOXK1	ST7FOXK2
Program memory - bytes	4K	8K
RAM (stack) - bytes	384 (128)	
Timers	Dual 8-bit timer, dual 12-bit AT (4 PWM)	Dual 8-bit timer, dual 12-bit AT (4 PWM), 1 x 16-bit timer
ADC	1 x 10-bit	
Peripherals	I²C	I²C and SPI
Packages	DIP20, SO20, LQFP32, SDIP32	LQFP32, SDIP32

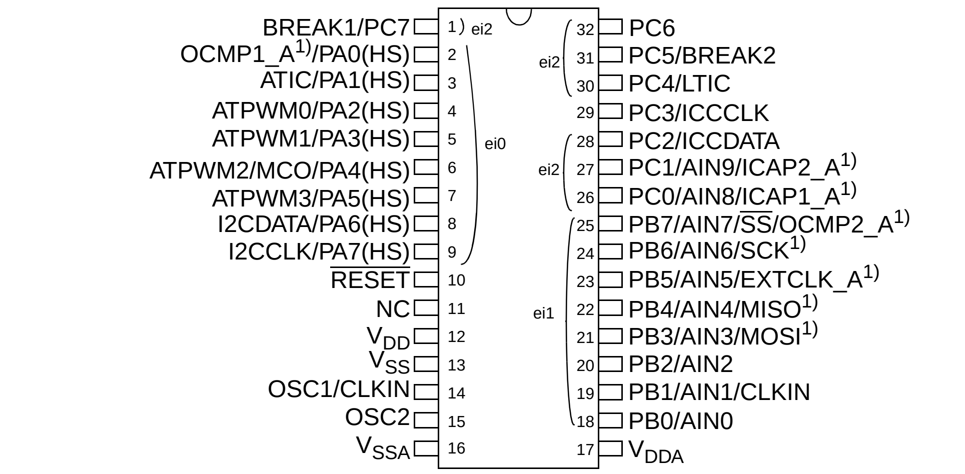
Figure 1. General block diagram



Note 1 : available on 8K version only

2 Pin description

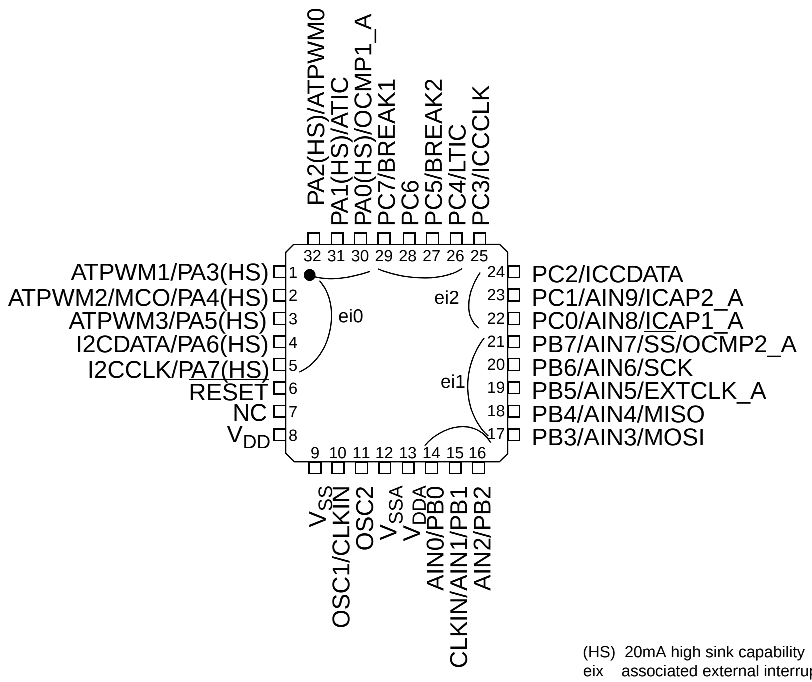
Figure 2. 32-pin SDIP package pinout



(HS) 20mA high sink capability
eix associated external interrupt vector

Note 1: Available on 8K version only

Figure 3. 32-pin LQFP 7x7 package pinout



(HS) 20mA high sink capability
eix associated external interrupt vector

Legend / Abbreviations for *Table 2*:

Type: I = input, O = output, S = supply

In/Output level: C_T = CMOS $0.3V_{DD}/0.7V_{DD}$ with input trigger

Output level: HS = 20 mA high sink (on N-buffer only)

Port and control configuration:

- Input: float = floating, wpu = weak pull-up, int = interrupt, ana = analog
- Output: OD = open drain, PP = push-pull

The RESET configuration of each pin is shown in bold which is valid as long as the device is in reset state.

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

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**		
1	5	PA3(HS)/ATPWM1	I/O	C_T	HS	**x**				x	x	**Port A3 (HS)**	ATPWM1
2	6	PA4(HS)/ ATPWM2/MCO	I/O	C_T	HS	**x**	ei0			x	x	**Port A4 (HS)**	ATPWM2/ MCO
3	7	PA5 (HS)ATPWM3	I/O	C_T	HS	**x**				x	x	**Port A5 (HS)**	ATPWM3
4	8	PA6(HS)/ I2CDATA/SCK(2)	I/O	C_T	HS	**x**		ei0		T		**Port A6 (HS)**	I2CDATA/SPI serial clock
5	9	PA7(HS)/I2CCLK/$\overline{SS}$ (2)	I/O	C_T	HS	**x**				T		**Port A7 (HS)**	I2CCLK/SPI slave select (active low)
6	10	RESET					**x**			x		Reset	
8	12	V_{DD}(3)	S									Digital Supply Voltage	
9	13	V_{SS}(3)	S									Digital Ground Voltage	
10	14	OSC1/CLKIN	I									Resonator oscillator inverter input or External clock input	
11	15	OSC2	O									Resonator oscillator output	
12	16	V_{SSA}(3)	S									Analog Ground Voltage	
13	17	V_{DDA}(3)	S									Analog Supply Voltage	

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

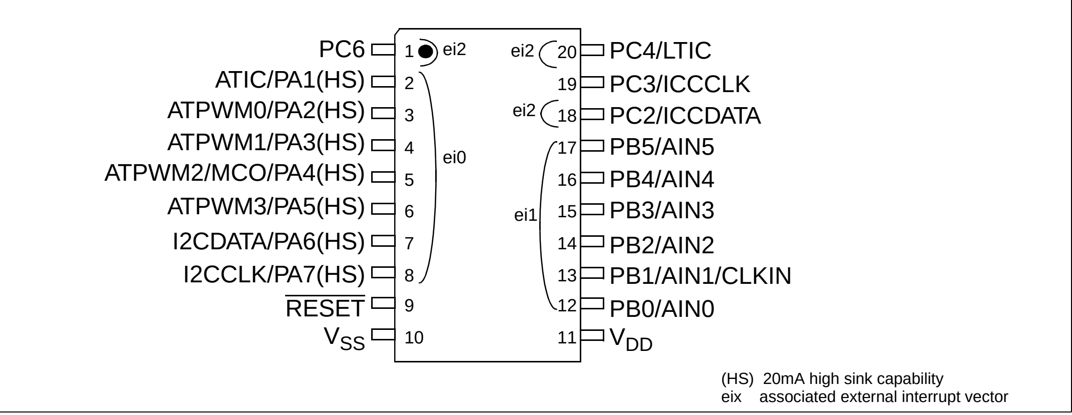
Pin number		Pin name	Type	Level		Port/control						Main function (after reset)	Alternate function
						Input				Output			
LQFP32	SDIP32			Input	Output	float	wpu	int	ana	OD[1]	PP		
14	18	PB0/AIN0	I/O	C_T		x		ei1	x	x	x	**Port B0**	AIN0
15	19	PB1/AIN1/CLKIN	I/O	C_T		x		ei1	x	x	x	**Port B1**	AIN1/External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

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		
30	2	PA0 (HS)[5]/OCMP1_A[2]	I/O	C_T	HS (5)	**x**		ei0		x	x	**Port A0** (HS)[5]/ Timer A Output Compare 1	
31	3	PA1 (HS)/ATIC	I/O	C_T	HS	**x**		ei0		x	x	**Port A1 (HS)**	ATIC
32	4	PA2 (HS)/ATPWM0	I/O	C_T	HS	**x**		ei0		x	x	**Port A2 (HS)**	ATPWM0

1. In the open-drain output column, T defines a true open-drain I/O (P-Buffer and protection diode to V_{DD} are not implemented).
2. Available on ST7FOXK2 only.
3. It is mandatory to connect all available V_{DD} and V_{DDA} pins to the supply voltage and all V_{SS} and V_{SSA} pins to ground.
4. BREAK2 available on ST7FOXK2 only
5. Available on ST7FOXK1 only.

Figure 4. 20-pin SO and DIP package pinout



Legend / Abbreviations for *Table 3*: Type: I = input, O = output, S = supply

In/Output level:C_T= CMOS 0.3V_{DD}/0.7V_{DD} with input trigger

Output level: HS = 20mA high sink (on N-buffer only)

Port and control configuration:

- Input: float = floating, wpu = weak pull-up, int = interrupt, ana = analog
- Output: OD = open drain, PP = push-pull

Note: The RESET configuration of each pin is shown in bold which is valid as long as the device is in reset state.

Table 3. Device pin description (20-pin package)

Pin Number	Pin Name	Type	Level		Port / Control						Main function (after reset)	Alternate function
			Input	Output	Input				Output [1]			
					float	wpu	int	ana	OD	PP		
1	PC6	I/O	C_T		x	ei2			x	x	Port C6	
2	PA1 (HS)/ATIC	I/O	C_T	HS	x	ei0			x	x	Port A1 (HS)	ATIC
3	PA2 (HS)/ATPWM0	I/O	C_T	HS	x	ei0			x	x	Port A2 (HS)	ATPWM0
4	PA3 (HS)/ATPWM1	I/O	C_T	HS	x	ei0			x	x	Port A3 (HS)	ATPWM1
5	PA4 (HS)ATPWM2/MCO	I/O	C_T	HS	x	ei0			x	x	Port A4 (HS)	ATPWM2/MCO
6	PA5 (HS)ATPWM3	I/O	C_T	HS	x	ei0			x	x	Port A5 (HS)	ATPWM3
7	PA6 (HS)/I2CDATA	I/O	C_T	HS	x	ei0			T		Port A6 (HS)	I2CDATA
8	PA7 (HS)/ I2CCLK	I/O	C_T	HS	x	ei0			T		Port A7 (HS)	I2CCLK
9	RESET					x			x		Reset	
10	V_{SS}[3]	S									Digital Ground Voltage	
11	V_{DD}[2]	S									Digital Supply Voltage	
12	PB0/AIN0	I/O	C_T		x	ei1		x	x	x	Port B0	AIN0
13	PB1/AIN1/CLKIN	I/O	C_T		x	ei1		x	x	x	Port B1	AIN1/External clock source
14	PB2/AIN2	I/O	C_T		x	ei1		x	x	x	Port B2	AIN2
15	PB3/AIN3	I/O	C_T		x	ei1		x	x	x	Port B3	AIN3
16	PB4/AIN4	I/O	C_T		x	ei1		x	x	x	Port B4	AIN4
17	PB5/AIN5	I/O	C_T		x	ei1		x	x	x	Port B5	AIN5
18	PC2/ICCDATA	I/O	C_T		x	ei2			x	x	Port C2	ICCDATA
19	PC3/ICCCLK	I/O	C_T		x				x	x	Port C3	ICCCLK
20	PC4/LTIC	I/O	C_T		x	ei2			x	x	Port C4	LTIC

1. In the open-drain output column, T defines a true open-drain I/O (P-Buffer and protection diode to V_{DD} not implemented).
2. It is mandatory to connect all available V_{DD} and V_{DDA} pins to the supply voltage and all V_{SS} and V_{SSA} pins to ground.

3 Register and memory mapping

As shown in *Figure 5*, the MCU is capable of addressing 64 Kbytes of memories and I/O registers.

The available memory locations consist of 128 bytes of register locations, 384 bytes of RAM and 4 to 8 Kbytes of Flash program memory. The RAM space includes up to 128 bytes for the stack from 180h to 1FFh.

The highest address bytes contain the user reset and interrupt vectors.

The Flash memory contains two sectors (see *Figure 5*) mapped in the upper part of the ST7 addressing space so the reset and interrupt vectors are located in Sector 0 (FFE0h-FFFFh).

The size of Flash Sector 0 and other device options are configurable by option bytes (refer to *Section 13.1 on page 211*).

Caution: Memory locations marked as “Reserved” must never be accessed. Accessing a reserved area can have unpredictable effects on the device.

Figure 5. ST7FOXF1/ST7FOXK1/ST7FOXK2 memory map

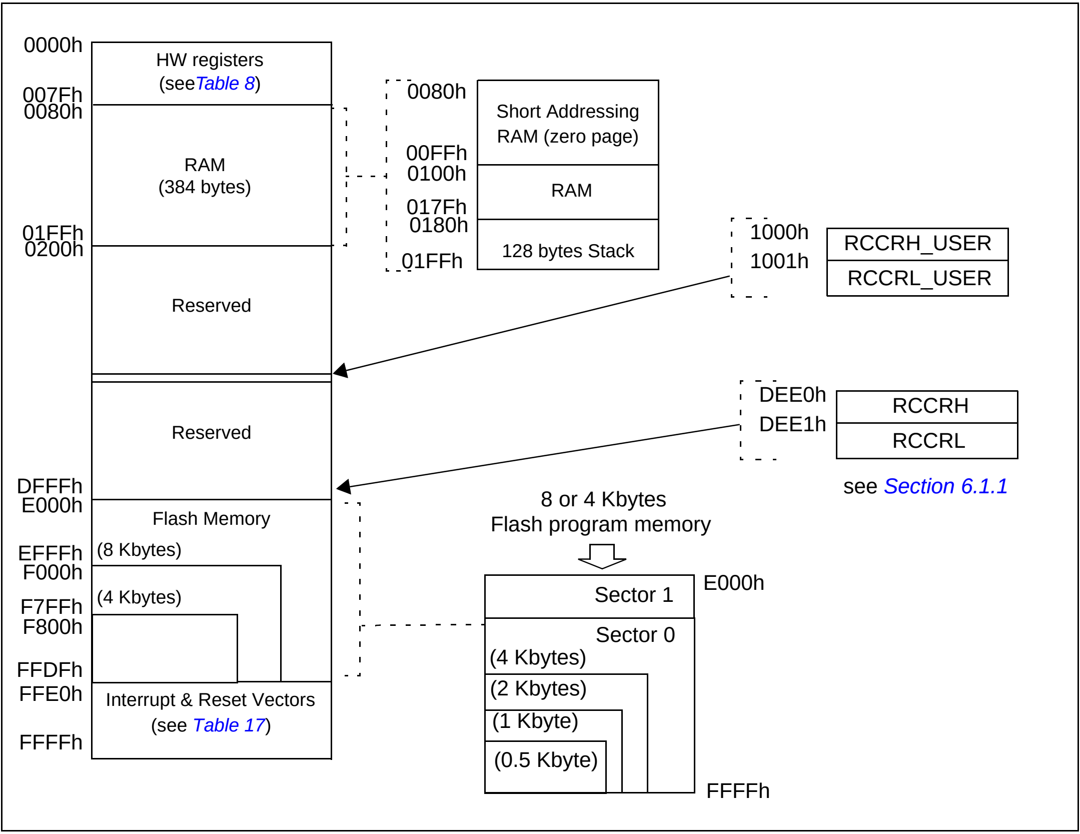


Table 4. Hardware register map[1]

Address	Block	Register label	Register name	Reset status	Remarks
0000h 0001h 0002h	Port A	PADR PADDR PAOR	Port A Data register Port A Data Direction register Port A Option register	00h 00h 00h	R/W R/W R/W
0003h 0004h 0005h	Port B	PBDR PBDDR PBOR	Port B Data register Port B Data Direction register Port B Option register	00h 00h 00h	R/W R/W R/W
0006h 0007h 0008h	Port C	PCDR PCDDR PCOR	Port C Data register Port C Data Direction register Port C Option register	00h 00h 08h	R/W R/W R/W
0009h to 000Bh	Reserved area (3 bytes)				
000Ch 000Dh 000Eh 000Fh 0010h	LITE TIMER	LTCSR2 LTARR LTCNTR LTCSR1 LTICR	Lite Timer Control/Status register 2 Lite Timer Auto-reload register Lite Timer Counter register Lite Timer Control/Status register 1 Lite Timer Input Capture register	0Fh 00h 00h 0x00 0000b xxh	R/W R/W Read Only R/W Read Only
0011h 0012h 0013h 0014h 0015h 0016h 0017h 0018h 0019h 001Ah 001Bh 001Ch 001Dh 001Eh 001Fh 0020h 0021h 0022h 0023h 0024h 0025h 0026h 0027h 0028h 0029h 002Ah	AUTO-RELOAD TIMER	ATCSR CNTR1H CNTR1L ATR1H ATR1L PWMCR PWM0CSR PWM1CSR PWM2CSR PWM3CSR DCR0H DCR0L DCR1H DCR1L DCR2H DCR2L DCR3H DCR3L ATICRH ATICRL ATCSR2 BREAKCR1 ATR2H ATR2L DTGR BREAKEN	Timer Control/Status register Counter register 1 High Counter register 1 Low Auto-Reload register 1 High Auto-Reload register 1 Low PWM Output Control register PWM 0 Control/Status register PWM 1 Control/Status register PWM 2 Control/Status register PWM 3 Control/Status register PWM 0 Duty Cycle register High PWM 0 Duty Cycle register Low PWM 1 Duty Cycle register High PWM 1 Duty Cycle register Low PWM 2 Duty Cycle register High PWM 2 Duty Cycle register Low PWM 3 Duty Cycle register High PWM 3 Duty Cycle register Low Input Capture register High Input Capture register Low Timer Control/Status register 2 Break Control register Auto-Reload register 2 High Auto-Reload register 2 Low Dead Time Generation register Break Enable register	0x00 0000b 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 03h 00h 00h 00h 00h 03h	R/W Read Only Read Only R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W Read Only Read Only R/W R/W R/W R/W R/W R/W
002Bh	Reserved area (1 byte)				
002Ch	AUTO-RELOAD TIMER	BREAKCR2 [4]	Break Control register 2 [4]	00h	R/W

Table 4. Hardware register map(1) (continued)

Address	Block	Register label	Register name	Reset status	Remarks
002Dh 002Eh 002Fh 0030h 0031h	ITC	ISPR0 ISPR1 ISPR2 ISPR3 EICR	Interrupt Software Priority register 0 Interrupt Software Priority register 1 Interrupt Software Priority register 2 Interrupt Software Priority register 3 External Interrupt Control register	FFh FFh FFh FFh 00h	R/W R/W R/W R/W R/W
0032h	Reserved area (1 byte)				
0033h	WDG	WDGCR	Watchdog Control register	7Fh	R/W
0034h	FLASH	FCSR	Flash Control/Status register	00h	R/W
0035h	RC Calibration	RCC_CSR	RC calibration Control/Status register	00h	R/W
0036h 0037h 0038h	ADC	ADCCSR ADCDRH ADCDRL	A/D Control Status register A/D Data register High A/D Data Low / test register	00h xxh 0xh	R/W Read Only R/W
0039h	Reserved area (1 byte)				
003Ah	MCC	MCCSR	Main Clock Control/Status register	00h	R/W
003Bh 003Ch 003Dh	Clock and Reset	RCCRH RCCRL PSCR	RC oscillator Control register High RC oscillator Control register Low Prescaler register	FFh 011x 0x00b 00h or 03h(2)	R/W R/W R/W
003Eh to 0047h	Reserved area (10 bytes)				
0048h 0049h	AWU	AWUCSR AWUPR	AWU Control/Status register AWU Preload register	FFh 00h	R/W R/W
004Ah 004Bh 004Ch 004Dh 004Eh 004Fh 0050h	DM(3)	DMCR DMSR DMBK1H DMBK1L DMBK2H DMBK2L DMCR2	DM Control register DM Status register DM Breakpoint register 1 High DM Breakpoint register 1 Low DM Breakpoint register 2 High DM Breakpoint register 2 Low DM Control register 2	00h 00h 00h 00h 00h 00h 00h	R/W R/W R/W R/W R/W R/W R/W
0051h	Clock Controller	CKCNTCSR	Clock Controller Status register	09h	R/W
0052h to 0054h	Reserved area (3 bytes)				

Table 4. Hardware register map(1) (continued)

Address	Block	Register label	Register name	Reset status	Remarks
0055h	16-bit Timer(4)	TACR2	Timer A Control register 2	00h	R/W
0056h		TACR1	Timer A Control register 1	00h	R/W
0057h		TACSR	Timer A Control/status register	00h	Read Only
0058h		TAICHR1	Timer A Input capture 1 high register	xxh	Read Only
0059h		TAICLR1	Timer A Input capture 1 low register	xxh	Read Only
005Ah		TAOCHR1	Timer A Output compare 1 high register	80h	R/W
005Bh		TAOCLR1	Timer A Output compare 1 low register	00h	R/W
005Ch		TACHR	Timer A Output counter high register	FFh	Read Only
005Dh		TACLR	Timer A Output counter low register	FCh	Read Only
005Eh		TAACHR	Timer A Alternate counter high register	FFh	Read Only
005Fh		TAACLR	Timer A Alternate counter low register	FCh	Read Only
0060h		TAICHR2	Timer A Input capture 2 high register	xxh	Read Only
0061h		TAICLR2	Timer A Input capture 2 low register	xxh	Read Only
0062h		TAOCHR2	Timer A Output compare 2 high register	80h	R/W
0063h		TAOCLR2	Timer A Output compare 2 low register	00h	R/W
0064h	I2C	I2CCR	I^2C Control register	00h	R/W
0065h		I2CSR1	I^2C Status register 1	00h	Read only
0066h		I2CSR2	I^2C Status register 2	00h	Read only
0067h		I2CCCR	I^2C Clock Control register	00h	R/W
0068h		I2COAR1	I^2C Own Address register 1	00h	R/W
0069h		I2COAR2	I^2C Own Address register 2	40h	R/W
006Ah		I2CDR	I^2C Data register	00h	R/W
0070h	SPI(4)	SPIDR	SPI Data register	0xh	R/W
0071h		SPICR	SPI Control register	00h	R/W
0072h		SPISR	SPI Status register	xxh	R/W

1. Legend: x=undefined, R/W=read/write.
2. Reset status is 03h for ST7FOXK2 and 00h for ST7FOXF1 and ST7FOXK1
3. For a description of the Debug Module registers, see ICC protocol reference manual.
4. Available on ST7FOXK2 only

4 Flash programmable memory

4.1 Introduction

The ST7 single voltage extended Flash (XFlash) is a non-volatile memory that can be electrically erased and programmed either on a byte-by-byte basis or up to 32 bytes in parallel.

The XFlash devices can be programmed off-board (plugged in a programming tool) or on-board using In-Circuit Programming or In-Application Programming.

The array matrix organization allows each sector to be erased and reprogrammed without affecting other sectors.

4.2 Main features

- ICP (In-Circuit Programming)
- IAP (In-Application Programming)
- ICT (In-Circuit Testing) for downloading and executing user application test patterns in RAM
- Sector 0 size configurable by option byte
- Read-out and write protection

4.3 Programming modes

The ST7 can be programmed in three different ways:

- Insertion in a programming tool. In this mode, Flash sectors 0 and 1, option byte row can be programmed or erased.
- In-Circuit Programming. In this mode, Flash sectors 0 and 1, option byte row can be programmed or erased without removing the device from the application board.
- In-Application Programming. In this mode, sector 1 can be programmed or erased without removing the device from the application board and while the application is running.

4.3.1 In-Circuit Programming (ICP)

ICP uses a protocol called ICC (In-Circuit Communication) which allows an ST7 plugged on a printed circuit board (PCB) to communicate with an external programming device connected via cable. ICP is performed in three steps:

Switch the ST7 to ICC mode (In-Circuit Communications). This is done by driving a specific signal sequence on the ICCCLK/DATA pins while the $\overline{\text{RESET}}$ pin is pulled low. When the ST7 enters ICC mode, it fetches a specific Reset vector which points to the ST7 System Memory containing the ICC protocol routine. This routine enables the ST7 to receive bytes from the ICC interface.

- Download ICP Driver code in RAM from the ICCDATA pin
- Execute ICP Driver code in RAM to program the Flash memory

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

4.3.2 In Application Programming (IAP)

This mode uses an IAP Driver program previously programmed in Sector 0 by the user (in ICP mode).

This mode is fully controlled by user software. This allows it to be adapted to the user application, (user-defined strategy for entering programming mode, choice of communications protocol used to fetch the data to be stored etc.)
IAP mode can be used to program any memory areas except Sector 0, which is Write/Erase protected to allow recovery in case errors occur during the programming operation.

4.4 ICC interface

ICP needs a minimum of 4 and up to 6 pins to be connected to the programming tool. These pins are:

- $\overline{\text{RESET}}$: device reset
- V_{SS}: device power supply ground
- ICCCLK: ICC output serial clock pin
- ICCDATA: ICC input serial data pin
- OSC1: main clock input for external source
- V_{DD}: application board power supply (optional, see Note 3)

Note: 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 be implemented in case another device forces the signal. Refer to the Programming Tool documentation for recommended resistor values.*

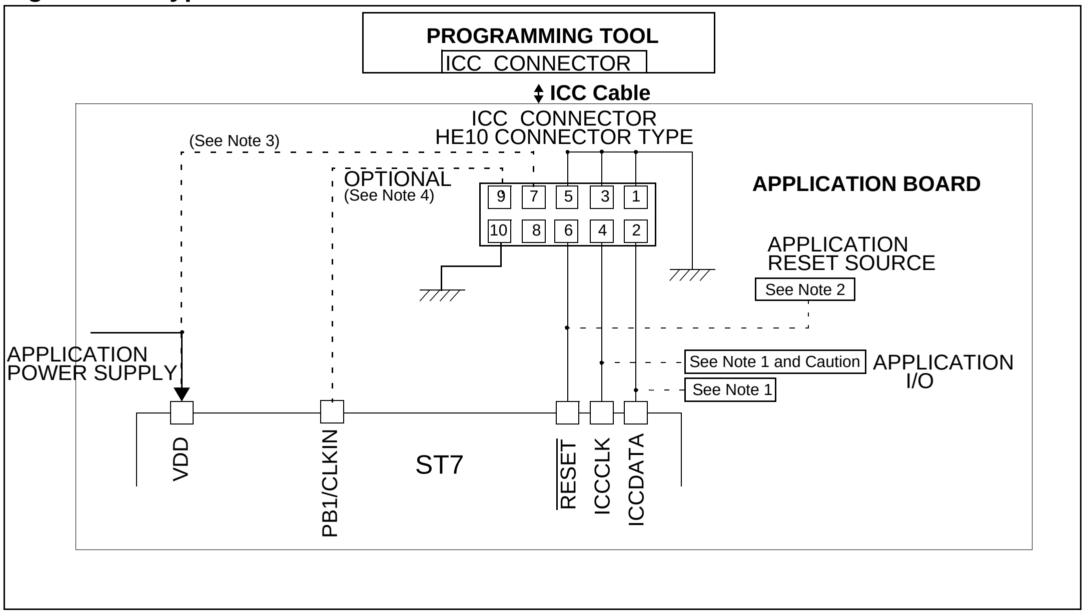
2 *During the ICP session, the programming tool must control the $\overline{\text{RESET}}$ pin. This can lead to conflicts between the programming tool and the application reset circuit if it drives more than 5mA at high level (push pull output or pull-up resistor<1 kΩ). A schottky diode can be used to isolate the application RESET circuit in this case. When using a classical RC network with R>1 kΩ or a reset management IC with open drain output and pull-up resistor>1 kΩ, 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 *In “enabled option byte” mode (38-pulse ICC mode), the internal RC oscillator is forced as a clock source, regardless of the selection in the option byte. In “disabled option byte” mode (35-pulse ICC mode), pin 9 has to be connected to the PB1/CLKIN 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.*

Caution: During normal operation the ICCCLK pin must be internally or externally pulled- up (external pull-up of 10 kΩ mandatory in noisy environment) to avoid entering ICC mode unexpectedly during a reset. In the application, even if the pin is configured as output, any reset will put it back in input pull-up.

Figure 6. Typical ICC Interface



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.

4.7 Description of Flash Control/Status register (FCSR)

This register controls the XFlash erasing and programming using ICP, IAP or other programming methods.

1st RASS Key: 0101 0110 (56h)

2nd RASS Key: 1010 1110 (AEh)

When an EPB or another programming tool is used (in socket or ICP mode), the RASS keys are sent automatically.

Reset value: 000 0000 (00h)

7							0
0	0	0	0	0	OPT	LAT	PGM
Read/write							

Table 5. Flash register mapping and reset values

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0034	FCSR Reset Value	- 0	- 0	- 0	- 0	- 0	OPT 0	LAT 0	PGM 0

5 Central processing unit

5.1 Introduction

This CPU has a full 8-bit architecture and contains six internal registers allowing efficient 8-bit data manipulation.

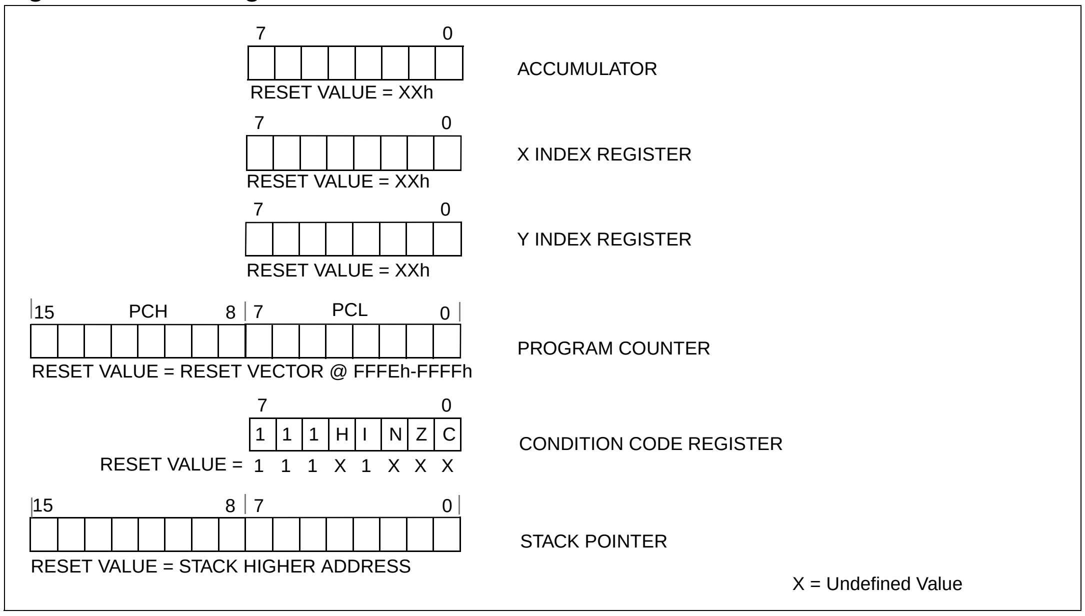
5.2 Main features

- 63 basic instructions
- Fast 8-bit by 8-bit multiply
- 17 main addressing modes
- Two 8-bit index registers
- 16-bit stack pointer
- Low power modes
- Maskable hardware interrupts
- Non-maskable software interrupt

5.3 CPU registers

The six CPU registers shown in *Figure 7*. They are not present in the memory mapping and are accessed by specific instructions.

Figure 7. CPU registers



5.3.1 Accumulator (A)

The Accumulator is an 8-bit general purpose register used to hold operands and the results of the arithmetic and logic calculations and to manipulate data.

5.3.2 Index registers (X and Y)

In indexed addressing modes, these 8-bit registers are used to create either effective addresses or temporary storage areas for data manipulation. (The Cross-Assembler generates a precede instruction (PRE) to indicate that the following instruction refers to the Y register.)

The Y register is not affected by the interrupt automatic procedures (not pushed to and popped from the stack).

5.3.3 Program Counter (PC)

The Program Counter is a 16-bit register containing the address of the next instruction to be executed by the CPU. It is made of two 8-bit registers PCL (Program Counter low which is the LSB) and PCH (Program Counter high which is the MSB).

5.3.4 Condition Code register (CC)

The 8-bit Condition Code register contains the interrupt mask and four flags representative of the result of the instruction just executed. This register can also be handled by the PUSH and POP instructions.

Reset value: 111x 1xxx

7							0
1	1	I1	H	I0	N	Z	C
Read/write							

These bits can be individually tested and/or controlled by specific instructions.

Arithmetic management bits

Bit 4 = **H** *Half carry bit*

This bit is set by hardware when a carry occurs between bits 3 and 4 of the ALU during an ADD or ADC instruction. It is reset by hardware during the same instructions.

0: No half carry has occurred.

1: A half carry has occurred.

This bit is tested using the JRH or JRNH instruction. The H bit is useful in BCD arithmetic subroutines.

Bit 3 = **I** *Interrupt mask bit*

This bit is set by hardware when entering in interrupt or by software to disable all interrupts except the TRAP software interrupt. This bit is cleared by software.

0: Interrupts are enabled.

1: Interrupts are disabled.

This bit is controlled by the RIM, SIM and IRET instructions and is tested by the JRM and JRNM instructions.

Note: Interrupts requested while I is set are latched and can be processed when I is cleared. By default an interrupt routine is not interruptible because the I bit is set by hardware at the start of the routine and reset by the IRET instruction at the end of the routine. If the I bit is cleared by software in the interrupt routine, pending interrupts are serviced regardless of the priority level of the current interrupt routine.

Bit 2 = **N** *Negative bit*

This bit is set and cleared by hardware. It is representative of the result sign of the last arithmetic, logical or data manipulation. It is a copy of the 7th bit of the result.

0: The result of the last operation is positive or null.

1: The result of the last operation is negative (that is, the most significant bit is a logic 1).

This bit is accessed by the JRMI and JRPL instructions.

Bit 1 **= Z** *Zero bit*

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.

Bit 0 = **C** *Carry/borrow bit*

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.

Interrupt management bits

Bits 5,3 = **I1, I0** *Interrupt bits*

The combination of the I1 and I0 bits gives the current interrupt software priority.

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 (IxSPR). They can be also set/cleared by software with the RIM, SIM, IRET, HALT, WFI and PUSH/POP instructions. See *Section 9.6: Interrupts* for more details.

Table 6. Interrupt software priority truth table

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

5.3.5 Stack Pointer (SP)

Reset value: 01FFh

15							8	7							0
0	0	0	0	0	0	0	1	1	SP6	SP5	SP4	SP3	SP2	SP1	SP0
Read/write															

The Stack Pointer is a 16-bit register which is always pointing to the next free location in the stack. It is then decremented after data has been pushed onto the stack and incremented before data is popped from the stack (see *Figure 8*).

Since the stack is 128 bytes deep, the 9 most significant bits are forced by hardware. Following an MCU Reset, or after a Reset Stack Pointer instruction (RSP), the Stack Pointer contains its reset value (the SP6 to SP0 bits are set) which is the stack higher address.

The least significant byte of the Stack Pointer (called S) can be directly accessed by a LD instruction.

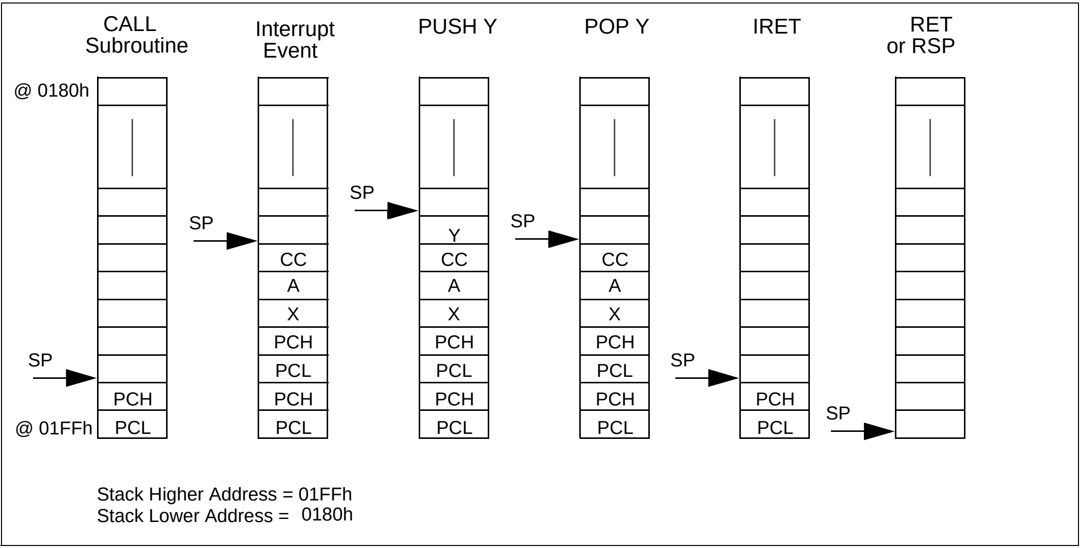
Note: When the lower limit is exceeded, the Stack Pointer wraps around to the stack upper limit, without indicating the stack overflow. The previously stored information is then overwritten and therefore lost. The stack also wraps in case of an underflow.

The stack is used to save the return address during a subroutine call and the CPU context during an interrupt. The user may also directly manipulate the stack by means of the PUSH and POP instructions. In the case of an interrupt, the PCL is stored at the first location pointed to by the SP. Then the other registers are stored in the next locations as shown in *Figure 8*.

- When an interrupt is received, the SP is decremented and the context is pushed on the stack.
- On return from interrupt, the SP is incremented and the context is popped from the stack.

A subroutine call occupies two locations and an interrupt five locations in the stack area.

Figure 8. Stack manipulation example



6 Supply, reset and clock management

The device includes a range of utility features for securing the application in critical situations (for example in case of a power brown-out), and reducing the number of external components. The main features are the following:

- Clock management
  - 8 MHz internal RC oscillator (enabled by option byte)
  - Auto wakeup RC oscillator (enabled by option byte)
  - 1 to 16 MHz or 32 kHz External crystal/ceramic resonator (selected by option byte)
  - External clock input (enabled by option byte)
- Reset Sequence Manager (RSM)
- System Integrity management (SI)
  - Main supply Low voltage detection (LVD) with reset generation (enabled by option byte)

6.1 RC oscillator adjustment

6.1.1 Internal RC oscillator

The device contains an internal RC oscillator with a specific accuracy for a given device, temperature and voltage range (4.5 V - 5.5 V). It must be calibrated to obtain the frequency required in the application. This is done by software writing a 10-bit calibration value in the RCCRH (RC Control register High) and in the bits 6:5 in the RCCRL (RC Control register Low).

Whenever the microcontroller is reset, the RCCR returns to its default value (FFh), i.e. each time the device is reset, the calibration value must be loaded in the RCCR. Predefined calibration values are stored for 5 V V_{DD} supply voltage at 25 °C (see *Table 7*).

Table 7. Predefined RC oscillator calibration values

RCCR	Conditions	ST7FOX Address
RCCRH	V_{DD}= 5V T_A= 25°C f_{RC} = 8 MHz	DEE0h(1) (CR[9:2])
RCCRL		DEE1h(1) (CR[1:0])

1. The DEE0h and DEE1h addresses are located in a reserved area in non-volatile memory. They are read-only bytes for the application code. This area cannot be erased or programmed by any ICC operations. For compatibility reasons with the RCCRL register, CR[1:0] bits are stored in the 5th and 6th position of DEE1 address.

In 38-pulse ICC mode, the internal RC oscillator is forced as a clock source, regardless of the selection in the option byte.

Section 12: Electrical characteristics on page 187 for more information on the frequency and accuracy of the 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 and also between the V_{DDA} and V_{SSA} pins as close as possible to the ST7 device.

These bytes are systematically programmed by ST.

6.1.2 Customized RC calibration

If the application requires a higher frequency accuracy or if the voltage or temperature conditions change in the application, the frequency may need to be recalibrated. Two non-volatile bytes (RCCRH_USER and RCCRL_USER) are reserved for storing these new values. These two-byte area is Electrically Erasable Programmable Read Only Memory.

Note: Refer to application note AN1324 for information on how to calibrate the RC frequency using an external reference signal.

How to program RCCRH_USER and RCCRL_USER

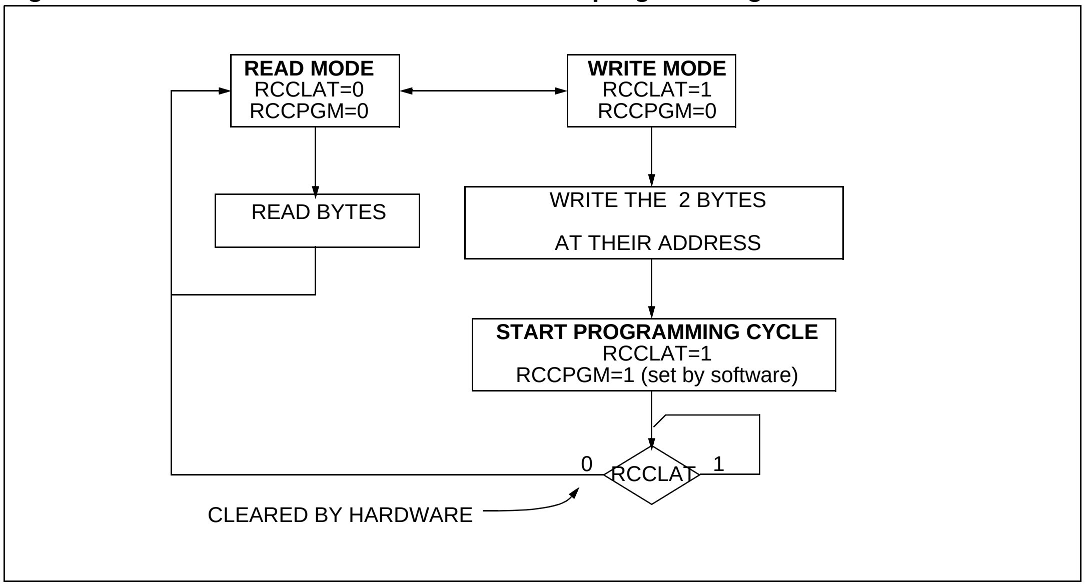
To access the write mode, the RCCLAT bit has to be set by software (the RCCPGM bit remains cleared). When a write access to this two-byte area occurs, the values are latched.

When RCCPGM bit is set by the software, the latched data are programmed in the EEPROM cells. To avoid wrong programming, the user must take care to only access these two-byte addresses.

At the end of the programming cycle, the RCCPGM and RCCLAT bits are cleared simultaneously.

Note: During the programming cycle, it is forbidden to access the latched data (see Figure 9).

Figure 9. RCCRH_USER and RCCRL_USER programming flowchart



Note: If a programming cycle is interrupted (by a reset action), the integrity of the data in memory is not guaranteed.

Access error handling

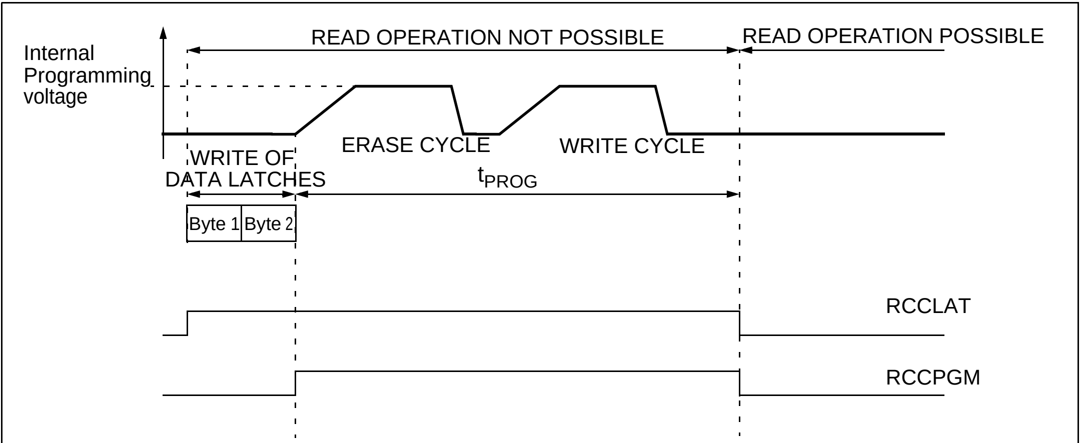
If a read access occurs while RCCLAT=1, then the data bus will not be driven.

If a write access occurs while RCCLAT=0, then the data on the bus will not be latched.

If a programming cycle is interrupted (by a RESET action), the integrity of the data in memory will not be guaranteed.

Caution: When the Read-Out Protection is enabled through an option bit (see *Section 13.1: Option bytes*), these two bytes are protected against Read-out (including a re-write protection). In Flash devices, when this protection is removed by reprogramming the option byte, these two bytes are automatically erased.

Figure 10. RC user calibration programming cycle



t_{PROG} is typically 5 ms and max 10 ms

6.1.3 Auto wakeup RC oscillator

The ST7FOX also contains an Auto wakeup RC oscillator. This RC oscillator should be enabled to enter Auto wakeup from halt mode.

The Auto wakeup (AWU) RC oscillator can also be configured as the startup clock through the CKSEL[1:0] option bits (see *Section 13.1: Option bytes on page 211*).

This is recommended for applications where very low power consumption is required.

Switching from one startup clock to another can be done in run mode as follows (see *Figure 11*):

Case 1 Switching from internal RC to AWU

1. Set the RC/AWU bit in the CKCNTCSR register to enable the AWU RC oscillator
2. The RC_FLAG is cleared and the clock output is at 1.
3. Wait 3 AWU RC cycles till the AWU_FLAG is set
4. The switch to the AWU clock is made at the positive edge of the AWU clock signal
5. Once the switch is made, the internal RC is stopped

Case 2 Switching from AWU RC to internal RC

1. Reset the RC/AWU bit to enable the internal RC oscillator
2. Using a 4-bit counter, wait until 8 internal RC cycles have elapsed. The counter is running on internal RC clock.
3. Wait till the AWU_FLAG is cleared (1AWU RC cycle) and the RC_FLAG is set (2 RC cycles)
4. The switch to the internal RC clock is made at the positive edge of the internal RC clock signal
5. Once the switch is made, the AWU RC is stopped

Note: 1 *When the internal RC is not selected, it is stopped so as to save power consumption.*

2 *When the internal RC is selected, the AWU RC is turned on by hardware when entering Auto wakeup from Halt mode.*

3 *When the external clock is selected, the AWU RC oscillator is always on.*

Figure 11. Clock switching

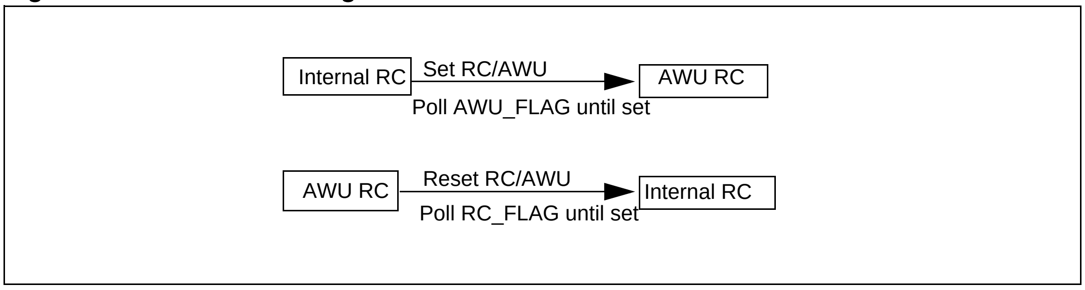
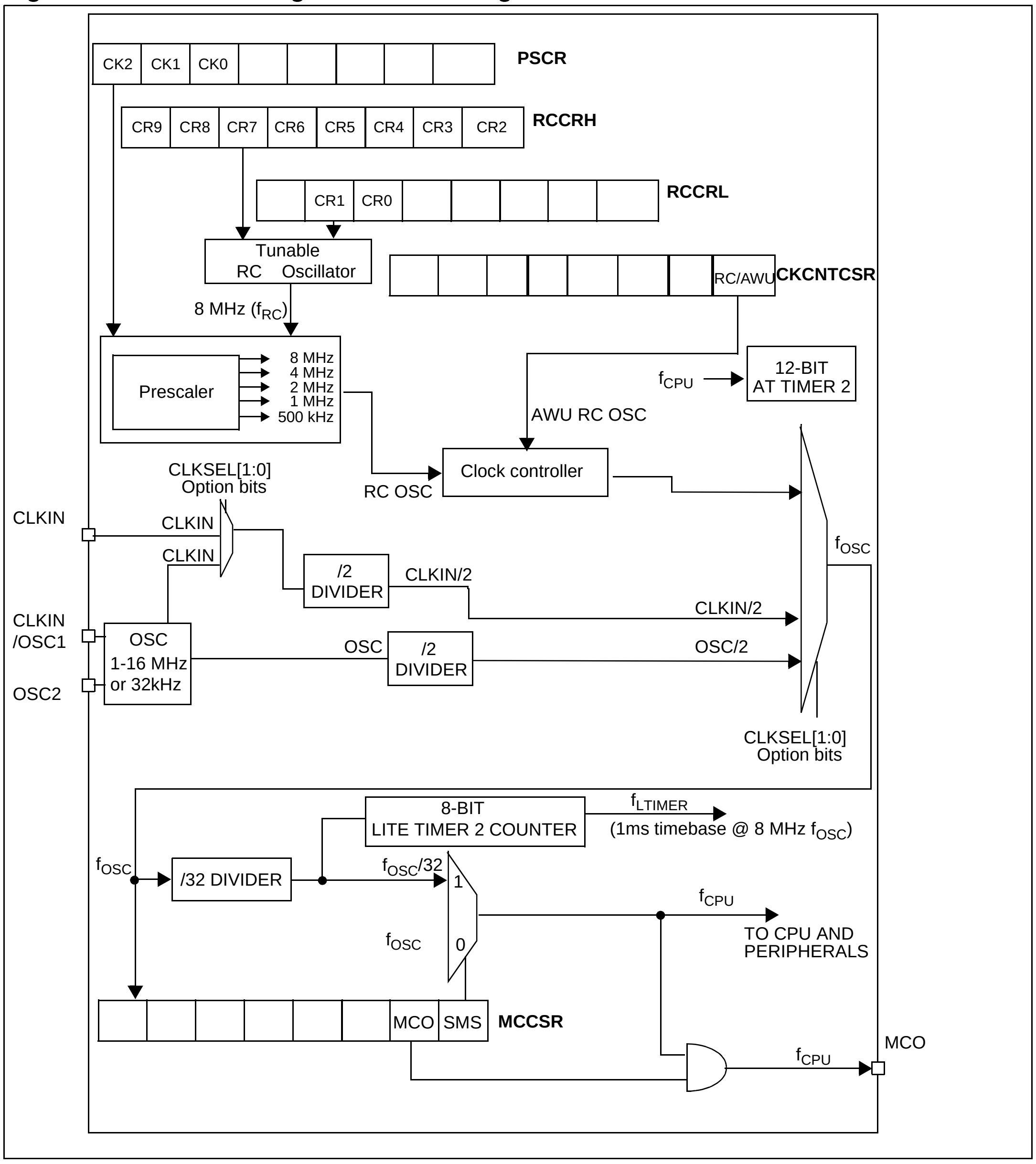


Figure 12. Clock management block diagram



6.2 Multi-oscillator (MO)

The main clock of the ST7 can be generated by four different source types coming from the multi-oscillator block (1 to 16 MHz):

- An external source
- 5 different configurations for 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 8*. Refer to the electrical characteristics section for more details.

6.2.1 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.

Note: When the Multi-Oscillator is not used OSCI1 and OSCI2 must be tied to ground, and PB1 is selected by default as the external clock.

6.2.2 Crystal/ceramic oscillators

In this mode, with a self-controlled gain feature, oscillator of any frequency from 1 to 16 MHz can be placed on OSC1 and OSC2 pins. This family of oscillators has the advantage of producing a very accurate rate on the main clock of the ST7. In this mode of the multi-oscillator, the resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and start-up stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

These oscillators are not stopped during the RESET phase to avoid losing time in the oscillator start-up phase.

6.2.3 Internal RC oscillator

In this mode, the tunable RC oscillator is used as main clock source. The two oscillator pins have to be tied to ground.

The calibration is done through the RCCRH[7:0] and RCCRL[6:5] registers.

Table 8. ST7 clock sources

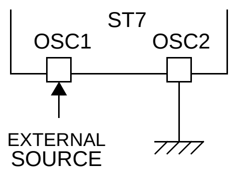
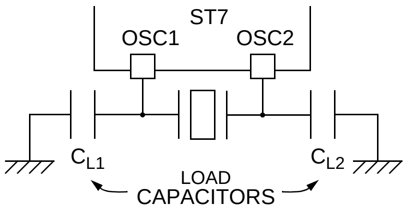
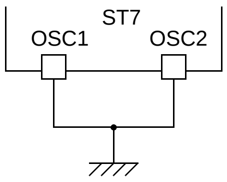
	Hardware configuration
External Clock	

Table 8. ST7 clock sources

	Hardware configuration
Crystal/Ceramic Resonators	 ST7 OSC1 OSC2 C_{L1} C_{L2} LOAD CAPACITORS
Internal RC Oscillator	

6.3 Reset sequence manager (RSM)

6.3.1 Introduction

The reset sequence manager includes three RESET sources as shown in *Figure 14*:

- External $\overline{\text{RESET}}$ source pulse
- Internal LVD RESET (Low Voltage Detection)
- Internal WATCHDOG RESET

Note: A reset can also be triggered following the detection of an illegal opcode or prebyte code. Refer to Section 11.2.1 on page 184 for further details.

These sources act on the $\overline{\text{RESET}}$ pin and it is always kept low during the delay phase.

The RESET service routine vector is fixed at addresses FFFEh-FFFFh in the ST7 memory mapping.

The basic RESET sequence consists of 3 phases as shown in *Figure 13*:

- Active Phase depending on the RESET source
- 256 or 4096 CPU clock cycle delay (see *Table 13*)

Caution: When the ST7 is unprogrammed or fully erased, the Flash is blank and the Reset vector is not programmed. For this reason, it is recommended to keep the $\overline{\text{RESET}}$ pin in low state until programming mode is entered, in order to avoid unwanted behavior.

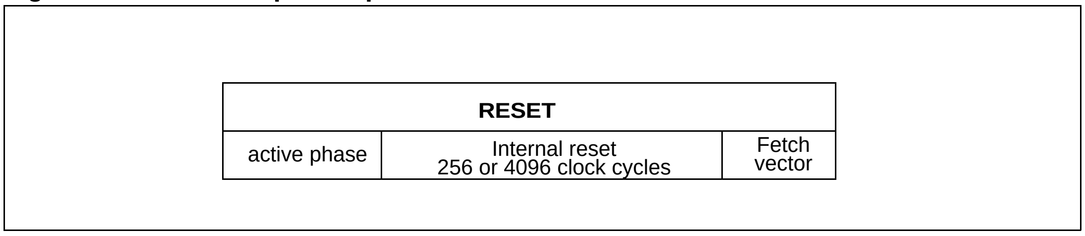
The 256 or 4096 CPU clock cycle delay allows the oscillator to stabilize and ensures that recovery has taken place from the Reset state. The shorter or longer clock cycle delay is automatically selected depending on the clock source chosen by option byte.

The Reset vector fetch phase duration is 2 clock cycles.

Table 9. CPU clock delay during Reset sequence

Clock source	CPU clock cycle delay
Internal RC 8 MHz Oscillator	4096
Internal RC 32 kHz Oscillator	256
External clock (connected to CLKIN/PB1 pin)	4096
External Crystal/Ceramic Oscillator (connected to OSC1/OSC2 pins)	4096
External Crystal/Ceramic 1-16 MHz Oscillator	4096
External Crystal/Ceramic 32 kHz Oscillator	256

Figure 13. Reset sequence phases



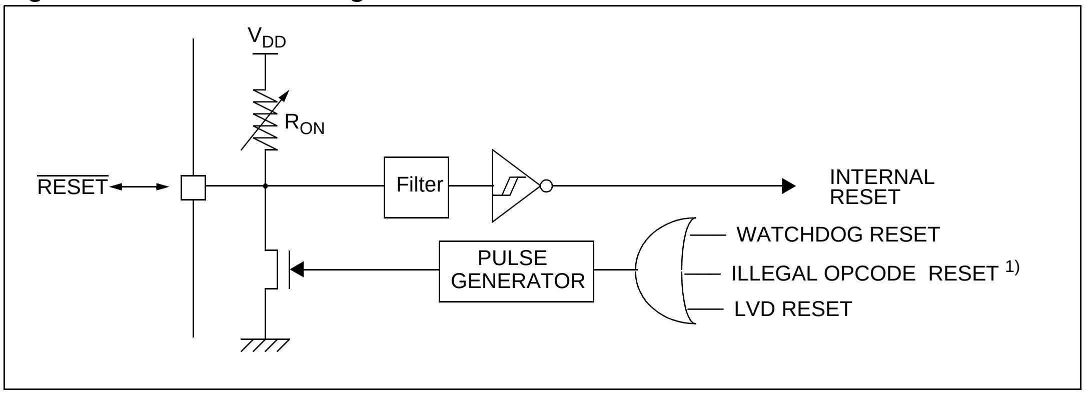
6.3.2 Asynchronous external $\overline{\text{RESET}}$ pin

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 Electrical Characteristic section 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 15: Reset sequences*). 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 the electrical characteristics section.

Figure 14. Reset block diagram



1. See *Section 11.2.1: Illegal opcode reset on page 184* for more details on illegal opcode reset conditions.

6.3.3 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.

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.

6.3.4 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} is lower than V_{IT+} (rising edge) or V_{DD} lower than V_{IT-} (falling edge) as shown in *Figure 15.*

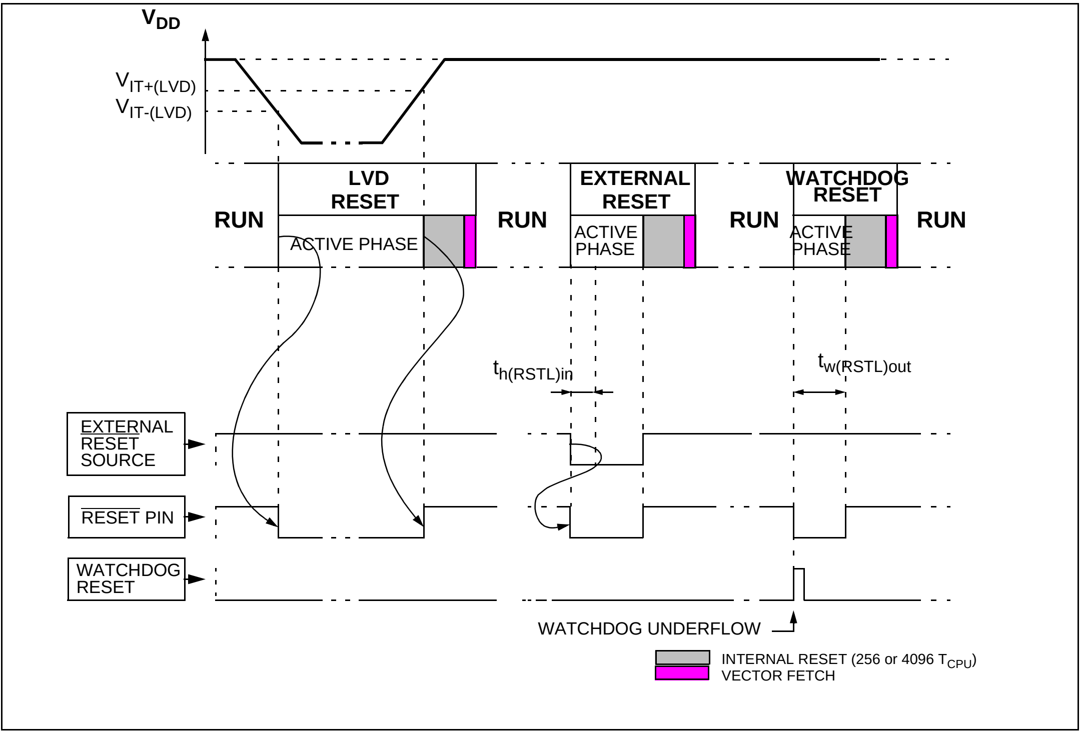
The LVD filters spikes on V_{DD} larger than $t_{g(VDD)}$ to avoid parasitic resets.

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



6.4 System Integrity management (SI)

The System Integrity Management block contains the Low voltage Detector (LVD).

Note: A reset can also be triggered following the detection of an illegal opcode or prebyte code. Refer to Section 11.2.1 on page 184 for further details.

6.4.1 Low Voltage Detector (LVD)

The Low Voltage Detector function (LVD) generates a static reset when the V_{DD} supply voltage is below a $V_{IT-(LVD)}$ reference value. This means that it secures the power-up as well as the power-down keeping the ST7 in reset.

The $V_{IT-(LVD)}$ reference value for a voltage drop is lower than the $V_{IT+(LVD)}$ reference value for power-on in order to avoid a parasitic reset when the MCU starts running and sinks current on the supply (hysteresis).

The LVD Reset circuitry generates a reset when V_{DD} is below:

- $V_{IT+(LVD)}$ when V_{DD} is rising
- $V_{IT-(LVD)}$ when V_{DD} is falling

The LVD function is illustrated in *Figure 16*.

The voltage threshold can be enabled/disabled by option byte. See *Section 13.1 on page 211*.

Provided the minimum V_{DD} value (guaranteed for the oscillator frequency) is above $V_{IT-(LVD)}$, the MCU can only be in two modes:

- Under full software control
- In static safe reset

In these conditions, secure operation is always ensured for the application without the need for external reset hardware.

During a Low Voltage Detector Reset, the $\overline{\text{RESET}}$ pin is held low, thus permitting the MCU to reset other devices.

Note: Use of LVD with capacitive power supply: with this type of power supply, if power cuts occur in the application, it is recommended to pull V_{DD} down to 0 V to ensure optimum restart conditions. Refer to circuit example in Figure 89 on page 207 and note 4.

The LVD is an optional function which can be selected by option byte. See Section 13.1 on page 211.

It allows the device to be used without any external RESET circuitry.

If the LVD is disabled, an external circuitry must be used to ensure a proper power-on reset.

It is recommended to make sure that the V_{DD} supply voltage rises monotonously when the device is exiting from Reset, to ensure the application functions properly.

Caution: If an LVD reset occurs after a watchdog reset has occurred, the LVD will take priority and will clear the watchdog flag.

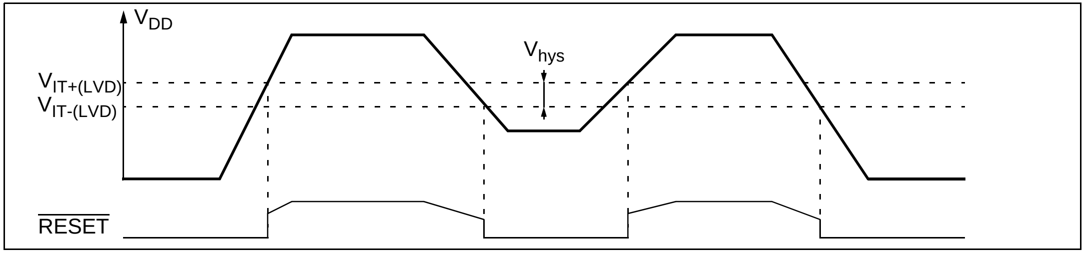
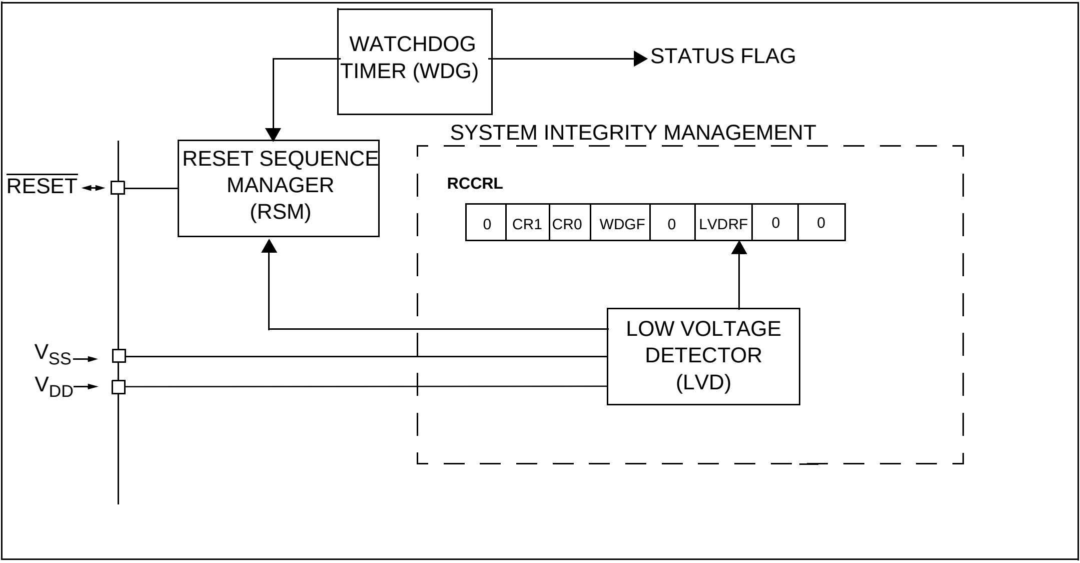


Figure 16. Low voltage detector vs reset



WATCHDOG TIMER (WDG)
STATUS FLAG
SYSTEM INTEGRITY MANAGEMENT
RESET SEQUENCE MANAGER (RSM)
$\overline{\text{RESET}}$
RCCRL
0
CR1
CR0
WDGF
0
LVDRF
0
0
LOW VOLTAGE DETECTOR (LVD)
V_{SS}
V_{DD}

Figure 17. Reset and supply management block diagram

6.5 Register description

6.5.1 RC calibration control/status register (RCC_CSR)

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	RCCLAT	RCCPGM
Read/write							

Bits 7:2 = Reserved, forced by hardware to 0

0: Read mode

1: Write mode

Bit 1 = **RCCLAT** *Latch Access Transfer bit:* this bit is set by software.

It is cleared by hardware at the end of the programming cycle. It can only be cleared by software if the RCCPGM bit is cleared

Bit 0 = **RCCPGM** *Programming Control and Status bit*

This bit is set by software to begin the programming cycle. At the end of the programming cycle, this bit is cleared by hardware.

0: Programming finished or not yet started

1: Programming cycle is in progress

Note: *If the RCCPGM bit is cleared during the programming cycle, the memory data is not guaranteed.*

6.5.2 Main Clock Control/Status Register (MCCSR)

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	MCO	SMS
Read/write							

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

Bit 1 = **MCO** *Main Clock Out enable bit*

This bit is read/write by software and cleared by hardware after a reset. This bit allows to enable the MCO output clock.

0: MCO clock disabled, I/O port free for general purpose I/O.

1: MCO clock enabled.

Bit 0 = **SMS** *Slow mode selection bit*

This bit is read/write by software and cleared by hardware after a reset. This bit selects the input clock f_{OSC} or $f_{OSC}/32$.

0: Normal mode ($f_{CPU} = f_{OSC}$

1: Slow mode ($f_{CPU} = f_{OSC}/32$)

6.5.3 RC Control Register High (RCCRH)

Reset value: 1111 1111 (FFh)

7							0
CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2
Read/write							

Bits 7:0 = **CR[9:2]** *RC Oscillator Frequency Adjustment bits*

These bits must be written immediately after reset to adjust the RC oscillator frequency. The application can store the correct value for each voltage range in Flash memory and write it to this register at start-up.

00h = maximum available frequency

FFh = lowest available frequency

These bits are used with the CR[1:0] bits in the RCCRL register. Refer to *Chapter 6.5.4*.

Note: To tune the oscillator, write a series of different values in the register until the correct frequency is reached. The fastest method is to use a dichotomy starting with 80h.

6.5.4 RC Control Register Low (RCCRL)

Reset value: 011x 0x00 (xxh)

7							0
0	CR1	CR0	WDGRF	0	LVDRF	0	0
Read/write							

Bit 7 = Reserved, must be kept cleared

Bits 6:5 = **CR[1:0]** *RC Oscillator Frequency Adjustment bits*

These bits, as well as CR[9:2] bits in the RCCRH register must be written immediately after reset to adjust the RC oscillator frequency. Refer to *Section 6.1.1: Internal RC oscillator on page 34*.

Bit 4 = **WDGRF** *Watchdog Reset flag*

This bit indicates that the last reset was generated by the watchdog peripheral. It is set by hardware (watchdog reset) and cleared by software (writing zero) or an LVD Reset (to ensure a stable cleared state of the WDGRF flag when CPU starts). The WDGRF and the LVDRF flags areis used to select the reset source (see *Table 10: Reset source selection on page 48*).

Table 10. Reset source selection

RESET source	LVDRF	WDGRF
External $\overline{\text{RESET}}$ pin	0	0
Watchdog	0	1
LVD	1	X

Bit 3 = Reserved, must be kept cleared

Bit 2 = **LVDRF** *LVD reset flag*

This bit indicates that the last Reset was generated by the LVD block. It is set by hardware (LVD reset) and cleared by software (by reading). When the LVD is disabled by option byte, the LVDRF bit value is undefined.

The LVDRF flag is not cleared when another RESET type occurs (external or watchdog), the LVDRF flag remains set to keep trace of the original failure.
In this case, a watchdog reset can be detected by software while an external reset can not.

Bits 1:0 = Reserved, must be kept cleared

6.5.5 Prescaler register (PSCR)

Reset value: 0000 0000 (00h) for ST7FOXF1 and ST7FOXK1

Reset value: 0000 0011 (03h) for ST7FOXK2

7							0
CK2	CK1	CK0	0	0	0	0 or 1	0 or 1
Read/write							

Bits 7:5 = **CK[2:0]** internal RC Prescaler Selection

These bits are set by software and cleared by hardware after a reset. These bits select the prescaler of the internal RC oscillator. See *Table 11*.

If the internal RC is used with a supply operating range below 3.3 V, a division ratio of at least 2 must be enabled in the RC prescaler.

Table 11. Internal RC prescaler selection bits

CK2	CK1	CK0	f_{OSC}
0	0	1	$f_{RC/2}$
0	1	0	$f_{RC/4}$
0	1	1	$f_{RC/8}$
1	0	0	$f_{RC/16}$
others			f_{RC}

Bits 4:2 = Reserved, must be cleared.

Bits 1:0 = Must be set.

Caution: For ST7FOXF1 and ST7FOXK1 devices, PRSC[1:0] bits must be forced to 1 by software in order to reduce current consumption.

6.5.6 Clock controller control/status register (CKCNTCSR)

Reset value: 0000 1001 (09h)

7							0
0	0	0	0	AWU_FLAG	RC_FLAG	0	RC/AWU
Read/write							

Bits 7:4 = Reserved, must be kept cleared.

Bit 3 = **AWU_FLAG** *AWU Selection bit*

This bit is set and cleared by hardware.

0: No switch from AWU to RC requested

1: AWU clock activated and temporization completed

Bit 2 = **RC_FLAG** *RC Selection bit*

This bit is set and cleared by hardware.

0: No switch from RC to AWU requested

1: RC clock activated and temporization completed

Bit 1 = Reserved, must be kept cleared.

Bit 0 = **RC/AWU** *RC/AWU Selection bit*

0: RC enabled

1: AWU enabled (default value)

Table 12. Clock register mapping and reset values

Address (Hex.)	**Register label**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0035h	RCC_CSR	- 0	- 0	- 0	- 0	- 0	- 0	RCCLAT 0	RCCPGM 0
003Ah	MCCSR Reset Value	- 0	- 0	- 0	- 0	- 0	- 0	MCO 0	SMS 0
003Bh	RCCRH Reset Value	CR9 1	CR8 1	CR7 1	CR6 1	CR5 1	CR4 1	CR3 1	CR2 1
003Ch	RCCRL Reset Value	- 0	CR1 1	CR0 1	WDGRF 0	- 0	LVDRF x	- 0	- 0
003Dh	PSCR Reset Value	CK2 0	CK1 0	CK0 0	- 0	- 0	- 0	- 0 or 1	- 0 or 1
0051h	CKCNTCSR Reset Value	- 0	- 0	- 0	- 0	AWU_FLAG 1	RC_FLAG 0	- 0	RC/AWU 1

7 Interrupts

7.1 Introduction

The ST7 enhanced interrupt management provides the following features:

- Hardware interrupts
- Software interrupt (TRAP)
- Nested or concurrent interrupt management with flexible interrupt priority and level management:
  - Up to 4 software programmable nesting levels
  - 13 interrupt vectors fixed by hardware
  - 2 non maskable events: RESET, TRAP

This interrupt management is based on:

- Bit 5 and bit 3 of the CPU CC register (I1:0),
- Interrupt software priority registers (ISPRx),
- Fixed interrupt vector addresses located at the high addresses of the memory mapping (FFE0h to FFFFh) sorted by hardware priority order.

This enhanced interrupt controller guarantees full upward compatibility with the standard (not nested) ST7 interrupt controller.

7.2 Masking and processing flow

The interrupt masking is managed by the I1 and I0 bits of the CC register and the ISPRx registers which give the interrupt software priority level of each interrupt vector (see *Table 13*). The processing flow is shown in *Figure 18*.

When an interrupt request has to be serviced:

- Normal processing is suspended at the end of the current instruction execution.
- The PC, X, A and CC registers are saved onto the stack.
- I1 and I0 bits of CC register are set according to the corresponding values in the ISPRx registers of the serviced interrupt vector.
- The PC is then loaded with the interrupt vector of the interrupt to service and the first instruction of the interrupt service routine is fetched (refer to interrupt mapping table for vector addresses).

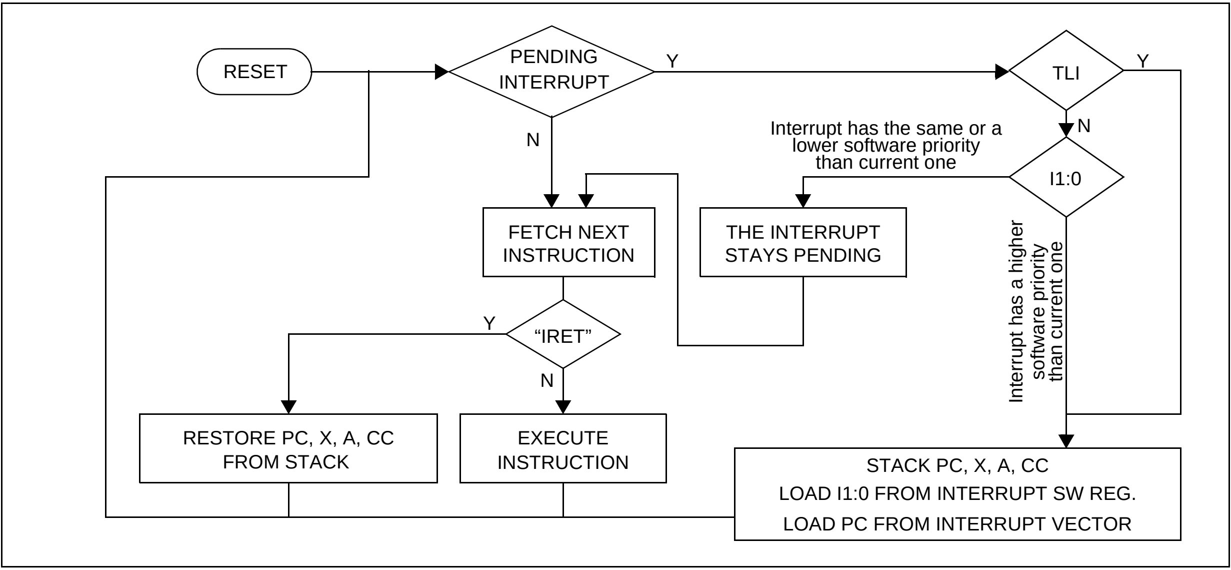
The interrupt service routine should end with the IRET instruction which causes the contents of the saved registers to be recovered from the stack.

Note: *As a consequence of the IRET instruction, the I1 and I0 bits will be restored from the stack and the program in the previous level will resume.*

Table 13. Interrupt software priority levels

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

Figure 18. Interrupt processing flowchart


RESET
PENDING INTERRUPT
Y
N
TLI
Y
N
Interrupt has the same or a lower software priority than current one
I1:0
Interrupt has a higher software priority than current one
FETCH NEXT INSTRUCTION
THE INTERRUPT STAYS PENDING
"IRET"
Y
N
RESTORE PC, X, A, CC FROM STACK
EXECUTE INSTRUCTION
STACK PC, X, A, CC
LOAD I1:0 FROM INTERRUPT SW REG.
LOAD PC FROM INTERRUPT VECTOR

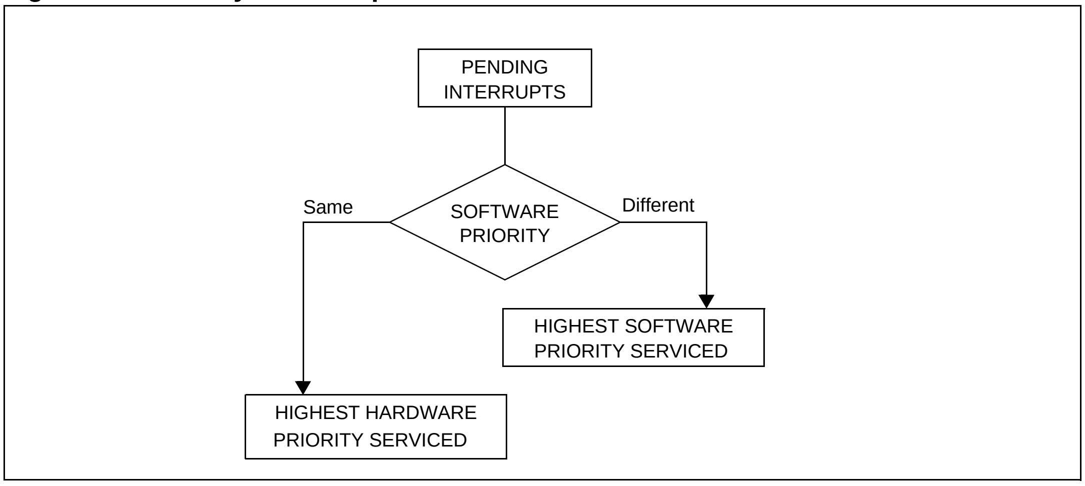
7.2.1 Servicing pending interrupts

As several interrupts can be pending at the same time, the interrupt to be taken into account is determined by the following two-step process:

- The highest software priority interrupt is serviced,
- If several interrupts have the same software priority then the interrupt with the highest hardware priority is serviced first.

Figure 19 describes this decision process.

Figure 19. Priority decision process


PENDING INTERRUPTS
Same
SOFTWARE PRIORITY
Different
HIGHEST SOFTWARE PRIORITY SERVICED
HIGHEST HARDWARE PRIORITY SERVICED

When an interrupt request is not serviced immediately, it is latched and then processed when its software priority combined with the hardware priority becomes the highest one.

Note: 1 *The hardware priority is exclusive while the software one is not. This allows the previous process to succeed with only one interrupt.*

2 *RESET and TRAP can be considered as having the highest software priority in the decision process.*

7.2.2 Interrupt vector sources

Two interrupt source types are managed by the ST7 interrupt controller: the non-maskable type (RESET, TRAP) and the maskable type (external or from internal peripherals).

Non-maskable sources

These sources are processed regardless of the state of the I1 and I0 bits of the CC register (see *Figure 18*). After stacking the PC, X, A and CC registers (except for RESET), the corresponding vector is loaded in the PC register and the I1 and I0 bits of the CC are set to disable interrupts (level 3). These sources allow the processor to exit Halt mode.

- TRAP (non maskable software interrupt)

  This software interrupt is serviced when the TRAP instruction is executed. It will be serviced according to the flowchart in *Figure 18*.
- RESET

  The RESET source has the highest priority in the ST7. This means that the first current routine has the highest software priority (level 3) and the highest hardware priority. See the RESET chapter for more details.

Maskable sources

Maskable interrupt vector sources can be serviced if the corresponding interrupt is enabled and if its own interrupt software priority (in ISPRx registers) is higher than the one currently being serviced (I1 and I0 in CC register). If any of these two conditions is false, the interrupt is latched and thus remains pending.

- External interrupts

  External interrupts allow the processor to exit from Halt low power mode.
  External interrupt sensitivity is software selectable through the External Interrupt Control register (EICR).
  External interrupt triggered on edge will be latched and the interrupt request automatically cleared upon entering the interrupt service routine.
  If several input pins of a group connected to the same interrupt line are selected simultaneously, these will be logically ORed.
- Peripheral interrupts

  Usually the peripheral interrupts cause the MCU to exit from Halt mode except those mentioned in *Table 17: ST7FOXF1/ST7FOXK1 Interrupt mapping*.
  A peripheral interrupt occurs when a specific flag is set in the peripheral status registers and if the corresponding enable bit is set in the peripheral control register. The general sequence for clearing an interrupt is based on an access to the status register followed by a read or write to an associated register.

Note: The clearing sequence resets the internal latch. A pending interrupt (that is, waiting for being serviced) will therefore be lost if the clear sequence is executed.

7.3 Interrupts and low power modes

All interrupts allow the processor to exit the Wait low power mode. On the contrary, only external and other specified interrupts allow the processor to exit from the Halt modes (see column “Exit from Halt” in *Table 17: ST7FOXF1/ST7FOXK1 Interrupt mapping*). When several pending interrupts are present while exiting Halt mode, the first one serviced can only be an interrupt with exit from Halt mode capability and it is selected through the same decision process shown in *Figure 19*.

Note: If an interrupt, that is not able to Exit from Halt mode, is pending with the highest priority when exiting Halt mode, this interrupt is serviced after the first one serviced.

7.4 Concurrent and nested management

The following *Figure 20* and *Figure 21* show two different interrupt management modes. The first is called concurrent mode and does not allow an interrupt to be interrupted, unlike the nested mode in *Figure 21*. The interrupt hardware priority is given in this order from the lowest to the highest: MAIN, IT5, IT4, IT3, IT2, IT1, IT0. The software priority is given for each interrupt.

Caution: A stack overflow may occur without notifying the software of the failure.

Figure 20. Concurrent interrupt management

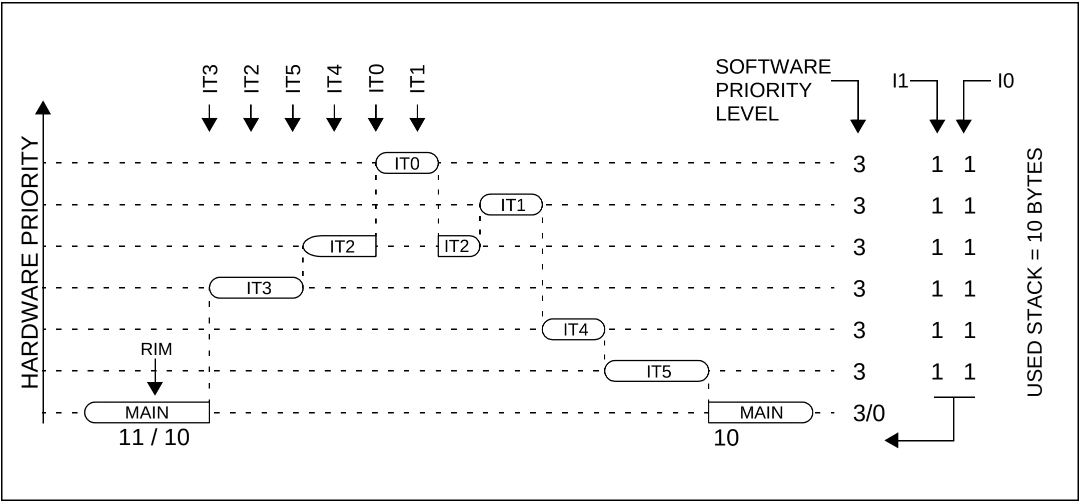
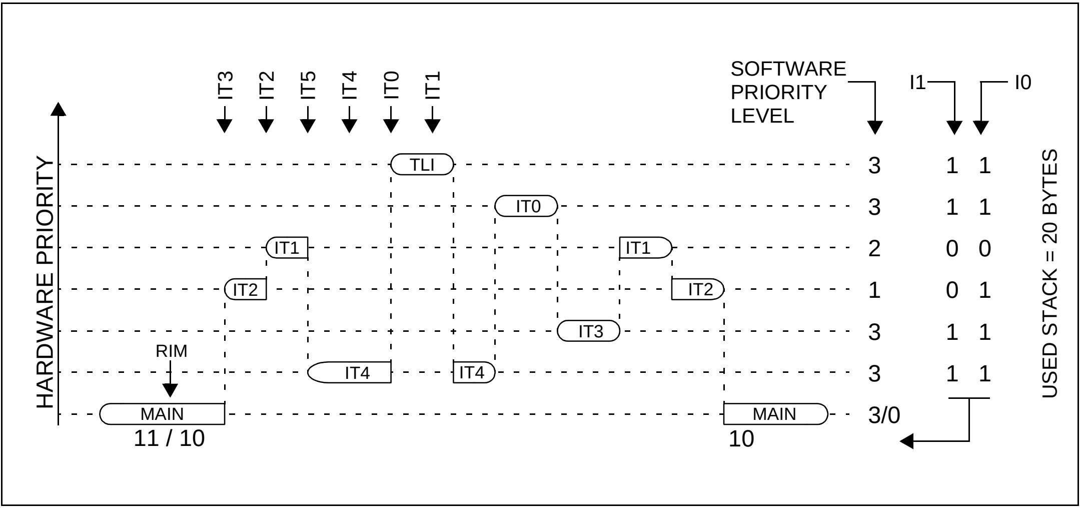


Figure 21. Nested interrupt management



7.5 Description of interrupt registers

7.5.1 CPU CC register interrupt bits

Reset value: 111x 1010(xAh)

7							0
1	1	I1	H	I0	N	Z	C
Read/write							

Bits 5, 3 = **I1, I0** *Software Interrupt Priority bits*

These two bits indicate the current interrupt software priority (see *Table 14*).

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, HALT, WFI, IRET and PUSH/POP instructions (see *Table 16: Dedicated interrupt instruction set*).

TRAP and RESET events can interrupt a level 3 program.

Table 14. Setting the interrupt software priority

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

7.5.2 Interrupt software priority registers (ISPRx)

All ISPRx register bits are read/write except bit 7:4 of **ISPR3** which are read only.

Reset value: 1111 1111 (FFh)

	7							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	1	1	I1_12	I0_12

ISPRx registers contain the interrupt software priority of each interrupt vector. Each interrupt vector (except RESET and TRAP) has corresponding bits in these registers to define its software priority. This correspondence is shown in *Table 15*.

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.

The RESET and TRAP vectors have no software priorities. When one is serviced, the I1 and I0 bits of the CC register are both set.

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).

Table 15. Interrupt vector vs ISPRx 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 can be read and written but they are not significant in the interrupt process management.

Caution: If the I1_x and I0_x bits are modified while the interrupt x is executed the following behavior has to be considered: If the interrupt x is still pending (new interrupt or flag not cleared) and the new software priority is higher than the previous one, the interrupt x is re-entered. Otherwise, the software priority stays unchanged up to the next interrupt request (after the IRET of the interrupt x).

Table 16. Dedicated interrupt instruction set(1)

Instruction	New description	Function/Example	I1	H	I0	N	Z	C
HALT	Entering Halt mode		1		0			
IRET	Interrupt routine return	Pop CC, A, X, PC	I1	H	I0	N	Z	C
JRM	Jump if I1:0 = 11 (level 3)	I1:0 = 11						
JRNM	Jump if I1:0 <> 11	I1:0 <> 11						
POP CC	Pop CC from the Stack	Mem => CC	I1	H	I0	N	Z	C
RIM	Enable interrupt (level 0 set)	Load 10 in I1:0 of CC	1		0			
SIM	Disable interrupt (level 3 set)	Load 11 in I1:0 of CC	1		1			
TRAP	Software trap	Software NMI	1		1			
WFI	Wait for interrupt		1		0			

1. During the execution of an interrupt routine, the HALT, POPCC, RIM, SIM and WFI instructions change the current software priority up to the next IRET instruction or one of the previously mentioned instructions.

Table 17. ST7FOXF1/ST7FOXK1 Interrupt mapping

Number	Source block	Description	Register label	Priority order	Exit from HALT or AWUFH (1)	Address vector
	RESET	Reset	N/A	Highest Priority	yes	FFFEh-FFFFh
	TRAP	Software interrupt			no	FFFCh-FFFDh
0	AWU	Auto Wake Up interrupt	AWUCSR		yes(2)	FFFAh-FFFBh
1	Reserved				-	FFF8h-FFF9h
2	ei0	External interrupt 0 (Port A)	N/A		yes	FFF6h-FFF7
3	ei1	External interrupt 1 (Port B)				FFF4h-FFF5h
4	ei2	External interrupt 2 (Port C)				FFF2h-FFF3h
5	AT TIMER	AT timer Output Compare interrupt	ATCSR		no	FFF0h-FFF1h
6		AT timer input Capture interrupt			no	FFEEh-FFEFh
7(3)		AT timer overflow 1 interrupt			no	FFECh-FFEDh
8		AT timer Overflow 2 interrupt			no	FFEAh-FFEBh
9	I^2C	I^2C interrupt	N/A	↓	no	FFE8h-FFE9h
10 (3)	LITE TIMER	Lite timer RTC interrupt	LTCSR2	Lowest Priority	yes	FFE6h-FFE7h
11		Lite timer Input Capture interrupt			no	FFE4h-FFE5h
12		Lite timer RTC2 interrupt			no	FFE2h-FFE3h

1. For an interrupt, all events do not have the same capability to wake up the MCU from Halt, Active-Halt or Auto Wake-up from Halt modes. Refer to the description of interrupt events for more details.
2. This interrupt exits the MCU from Auto Wake-up from Halt mode only.
3. These interrupts exit the MCU from Active-Halt mode only.

Table 18. ST7FOXK2 interrupt mapping

Number	Source block	Description	Register label	Priority order	Exit from HALT or AWUFH (1)	Address vector
	RESET	Reset	N/A	Highest Priority ↓ Lowest Priority	yes	FFFEh-FFFFh
	TRAP	Software interrupt			no	FFFCh-FFFDh
0	AWU	Auto Wake Up interrupt	AWUCSR		yes (2)	FFFAh-FFFBh
1	Reserved				-	FFF8h-FFF9h
2	Reserved				-	FFF6h-FFF7h
3	Reserved				-	FFF4h-FFF5h
4	ei0	External interrupt 0 (Port A)	N/A		yes	FFF2h-FFF3h
5	ei1	External interrupt 1 (Port B)				FFF0h-FFF1h
6	ei2	External interrupt 2 (Port C)				FFEEh-FFEFh
7	AT TIMER	AT timer input Capture/Output Compare interrupt	ATCSR		no	FFECh-FFEDh
8(3)		AT timer overflow 1 interrupt			no	FFEAh-FFEBh
9		AT timer Overflow 2 interrupt			no	FFE8h-FFE9h
10	I^2C	I^2C interrupt	I2CSR1/ I2CSR2		no	FFE6h-FFE7h
11	SPI	SPI interrupt	SPICSR		no	FFE4h-FFE5h
12	TIM16	16-bit timer peripheral interrupt	TACSR		no	FFE2h-FFE3h
13(3)	LITE TIMER	Lite timer RTC/IC/RTC2 interrupt	LTCSR2		yes	FFE0h-FFE1h

1. For an interrupt, all events do not have the same capability to wake up the MCU from Halt, Active-Halt or Auto-wakeup from Halt modes. Refer to the description of interrupt events for more details.
2. This interrupt exits the MCU from Auto-wakeup from Halt mode only.
3. These interrupts exit the MCU from Active-Halt mode only.

7.5.3 External Interrupt Control register (EICR)

Reset value: 0000 0000 (00h)

7							0
0	0	IS21	IS20	IS11	IS10	IS01	IS00
Read/write							

Bits 7:6 = Reserved, must be kept cleared.

Bits 5:4 = **IS2[1:0]** *ei2 sensitivity bits*

These bits define the interrupt sensitivity for ei2 (Port C) according to *Table 19*.

Bits 3:2 = **IS1[1:0]** *ei1 sensitivity bits*

These bits define the interrupt sensitivity for ei1 (Port B) according to *Table 19*.

Bits 1:0 = **IS0[1:0]** *ei0 sensitivity bits*

These bits define the interrupt sensitivity for ei0 (Port A) according to *Table 19*.

Note: 1 These 8 bits can be written only when the I bit in the CC register is set.

2 Changing the sensitivity of a particular external interrupt clears this pending interrupt. This can be used to clear unwanted pending interrupts. Refer to Section : External interrupt function.

Table 19. Interrupt sensitivity bits

ISx1	ISx0	External interrupt sensitivity
0	0	Falling edge & low level
0	1	Rising edge only
1	0	Falling edge only
1	1	Rising and falling edge

8 Power saving modes

8.1 Introduction

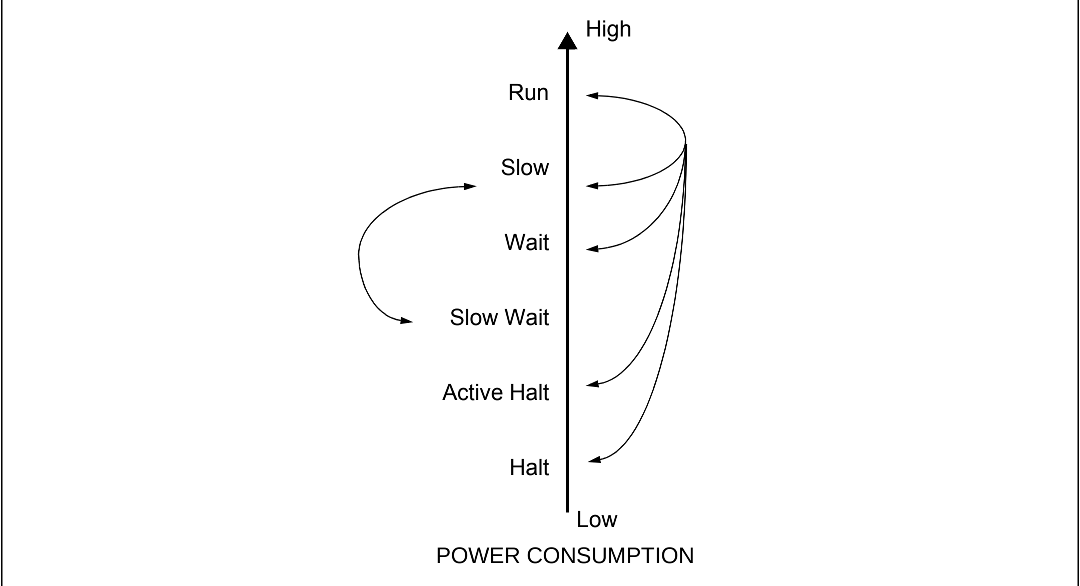
To give a large measure of flexibility to the application in terms of power consumption, four main power saving modes are implemented in the ST7 (see *Figure 22*):

- Slow
- Wait (and Slow-Wait)
- Active Halt
- Auto wakeup From Halt (AWUFH)
- 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 (f_{OSC}).

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:

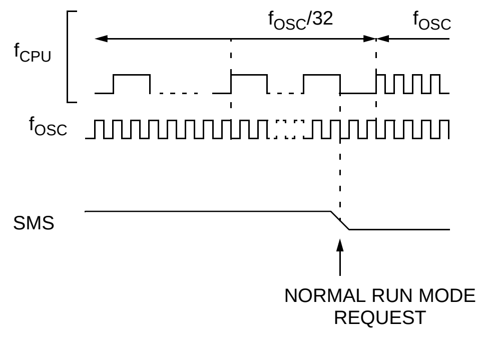
- To reduce power consumption by decreasing the internal clock in the device,
- To adapt the internal clock frequency (f_{CPU}) to the available supply voltage.

Slow mode is controlled by the SMS bit in the MCCSR register which enables or disables Slow mode.

In this mode, the oscillator frequency is divided by 32. The CPU and peripherals are clocked at this lower frequency.

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

Figure 23. Slow mode clock transition



8.3 Wait mode

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

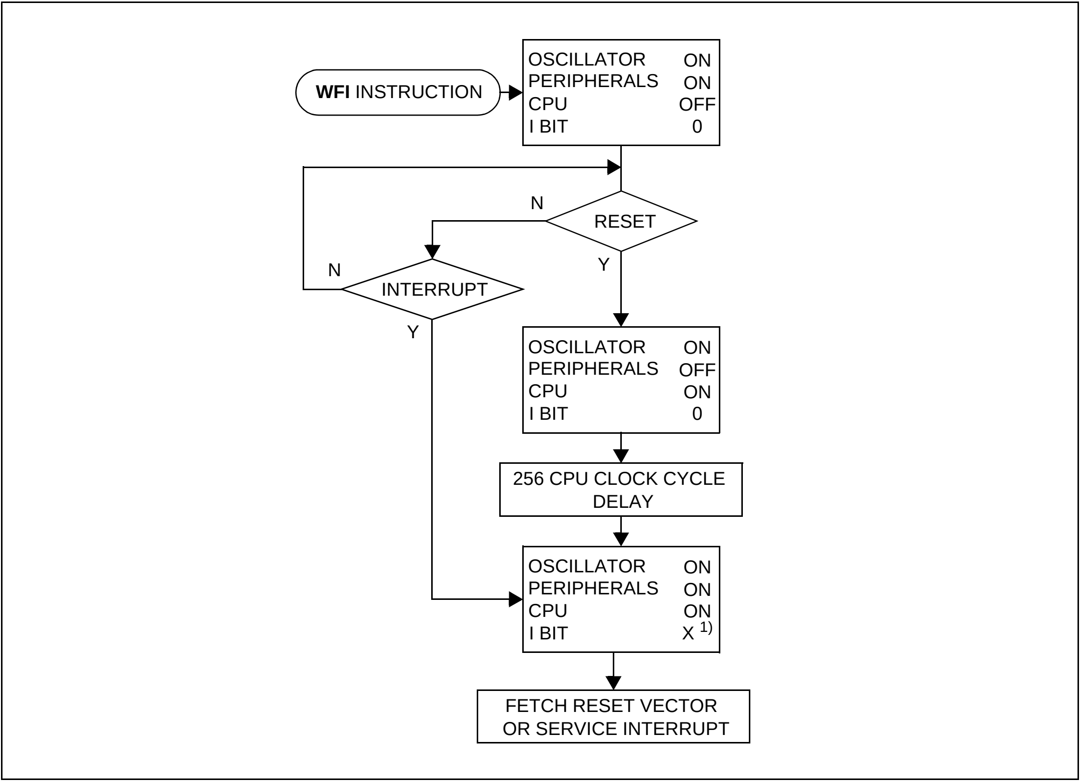
This power saving mode is selected by calling the 'WFI' instruction.

All peripherals remain active. During Wait mode, the I bit of the CC register is cleared, 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 *Figure 24* for a description of the Wait mode flowchart.

Figure 24. Wait mode flowchart



1. Before servicing an interrupt, the CC register is pushed on the stack. The I bit of the CC register is set during the interrupt routine and cleared when the CC register is popped.

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 LTCSR/ATCSR register status as shown in the following table:

Table 20. Enabling/disabling active-halt and halt modes

LTCSR TBIE bit	ATCSR OVFIE bit	ATCSRCK1 bit	ATCSRCK0 bit	Meaning
0	x	x	0	Active-Halt mode disabled
0	0	x	x	
0	1	1	1	
1	x	x	x	Active-Halt mode enabled
x	1	0	1	

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 active halt mode is enabled.

The MCU can exit Active-Halt mode on reception of a Lite timer/ AT timer interrupt or a Reset.

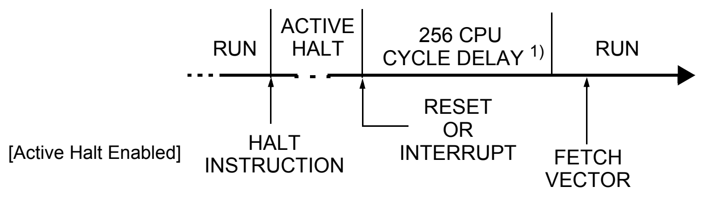
- When exiting Active-Halt mode by means of a Reset, a 256 CPU cycle delay occurs. After the start up delay, the CPU resumes operation by fetching the Reset vector which woke it up (see *Figure 26*).
- When exiting Active-Halt mode by means of an interrupt, the CPU immediately resumes operation by servicing the interrupt vector which woke it up (see *Figure 26*).

When entering Active-Halt mode, the I bit in the CC register is cleared to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes up immediately.

In Active-Halt mode, only the main oscillator and the selected timer counter (LT/AT) are running to keep a wakeup 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).

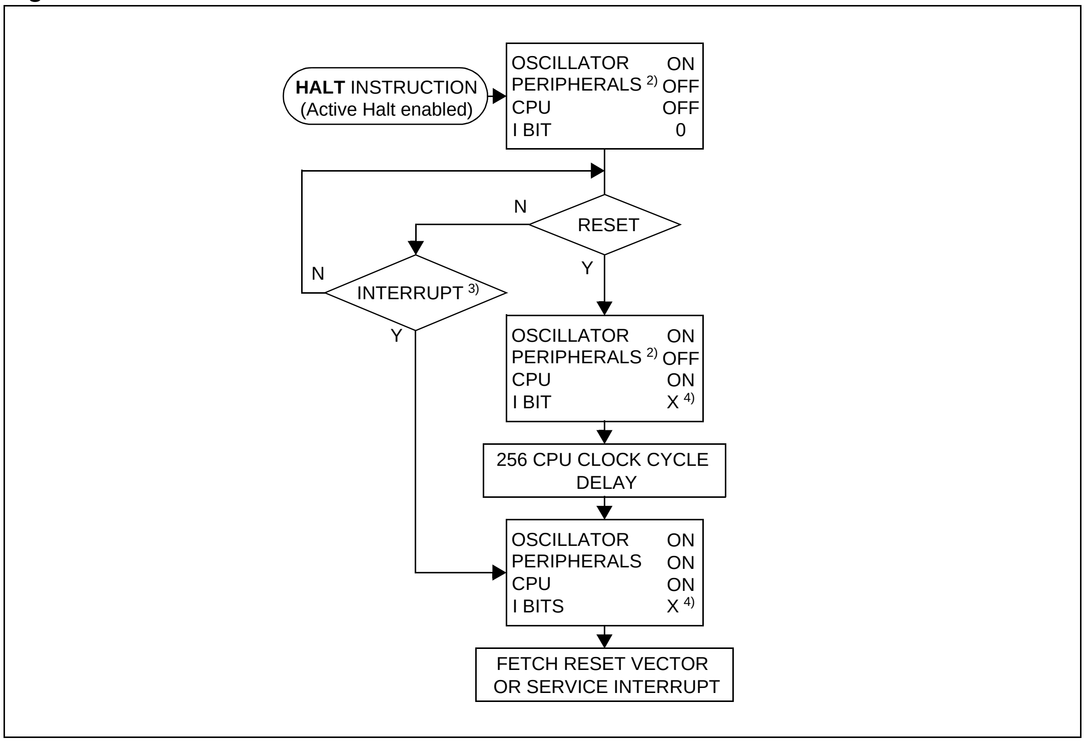
Caution: As soon as Active-Halt is enabled, executing a HALT instruction while the Watchdog is active does not generate a Reset if the WDGHALT bit is reset.
This means that the device cannot spend more than a defined delay in this power saving mode.

Figure 25. Active-halt timing overview



1. This delay occurs only if the MCU exits Active-Halt mode by means of a RESET.

Figure 26. Active-halt mode flowchart



1. This delay occurs only if the MCU exits Active-Halt mode by means of a RESET.
2. Peripherals clocked with an external clock source can still be active.
3. Only the Lite timer RTC and AT timer interrupts can exit the MCU from Active-Halt mode.
4. Before servicing an interrupt, the CC register is pushed on the stack. The I bit of the CC register is set during the interrupt routine and cleared when the CC register is popped.

8.4.2 Halt mode

The Halt mode is the lowest power consumption mode of the MCU. It is entered by executing the HALT instruction when active halt mode is disabled.

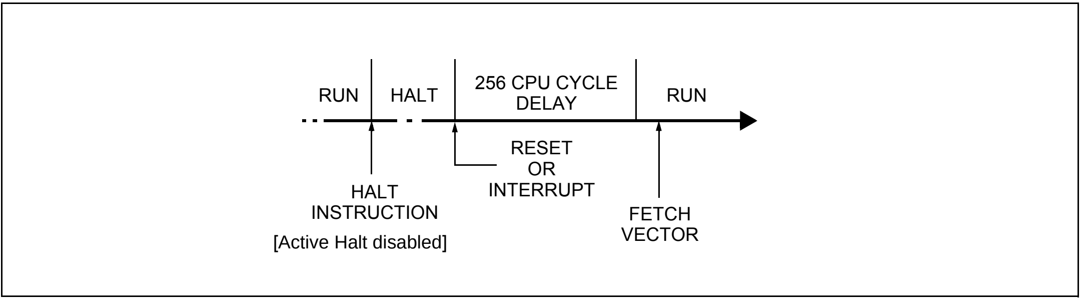
The MCU can exit Halt mode on reception of either a specific interrupt (see*Table 17: ST7FOXF1/ST7FOXK1 Interrupt mapping*) or a Reset. When exiting Halt mode by means of a Reset or an interrupt, the main oscillator is immediately turned on and the 256 CPU cycle delay is used to stabilize it. After the start up delay, the CPU resumes operation by servicing the interrupt or by fetching the Reset vector which woke it up (see *Figure 28*).

When entering Halt mode, the I bit in the CC register is forced to 0 to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes immediately.

In Halt mode, the main oscillator is turned off causing all internal processing to be stopped, including the operation of the on-chip peripherals. All peripherals are not clocked except the ones which get their clock supply from another clock generator (such as an external or auxiliary oscillator).

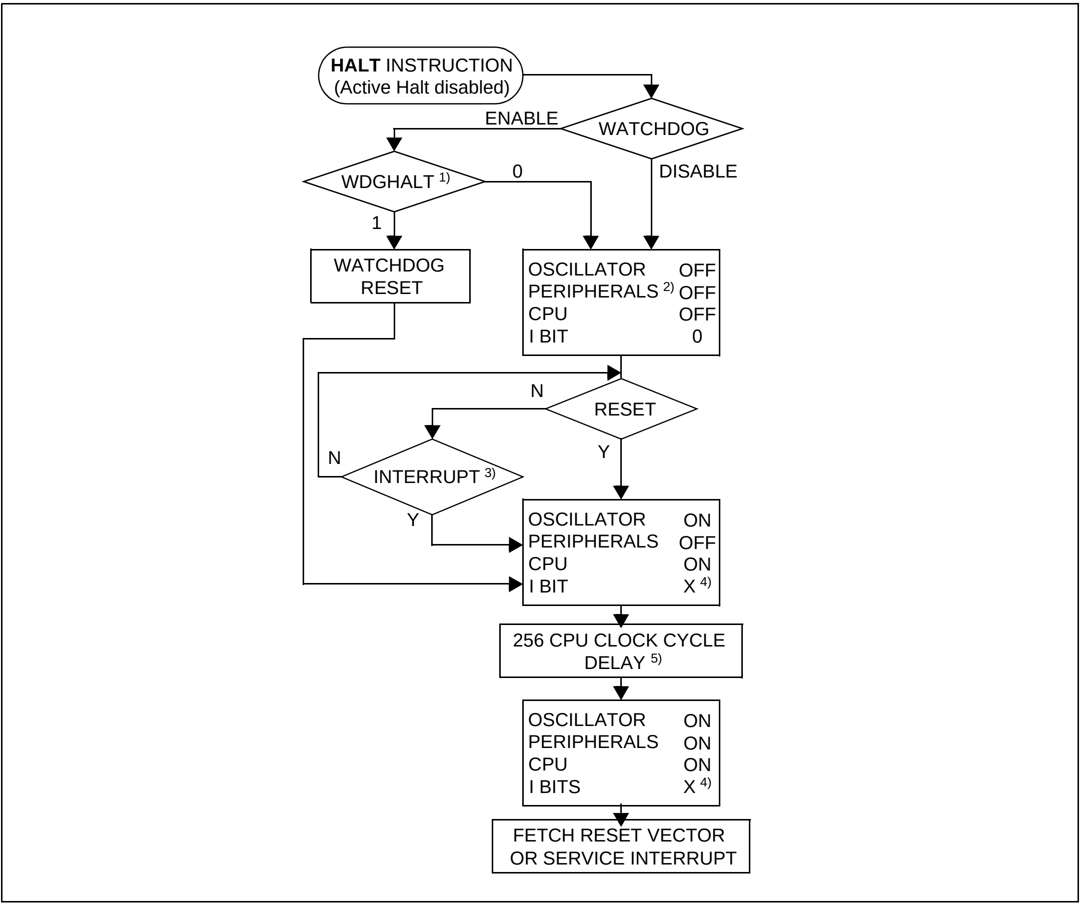
The compatibility of Watchdog operation with Halt mode is configured by the “WDGHALT” option bit of the option byte. The HALT instruction when executed while the Watchdog system is enabled, can generate a Watchdog Reset (see *Section 13.1: Option bytes* for more details).

Figure 27. Halt timing overview



1. A reset pulse of at least 42 µs must be applied when exiting from Halt mode.

Figure 28. Halt mode flowchart



1. WDGHALT is an option bit. See option byte section for more details.
2. Peripheral clocked with an external clock source can still be active.
3. Only some specific interrupts can exit the MCU from Halt mode (such as external interrupt). Refer to *Table 17: ST7FOXF1/ST7FOXK1 Interrupt mapping*for more details.
4. Before servicing an interrupt, the CC register is pushed on the stack. The I bit of the CC register is set during the interrupt routine and cleared when the CC register is popped.
5. The CPU clock must be switched to 1 MHz (RC/8) or AWU RC before entering Halt mode.

Halt mode recommendations

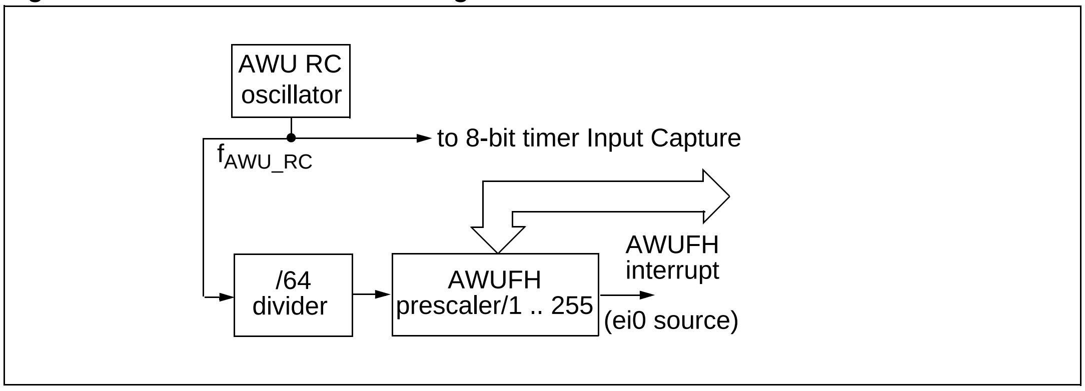
- Make sure that an external event is available to wake up the microcontroller from Halt mode.
- When using an external interrupt to wake up the microcontroller, reinitialize the corresponding I/O as “Input Pull-up with Interrupt” before executing the HALT instruction. The main reason for this is that 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



As soon as Halt mode is entered, and if the AWUEN bit has been set in the AWUCSR register, the AWU RC oscillator provides a clock signal (f_{AWU_RC}). Its frequency is divided by a fixed divider and a programmable prescaler controlled by the AWUPR register. The output of this prescaler provides the delay time. When the delay has elapsed, the following actions are performed:

- the AWUF flag is set by hardware,
- an interrupt wakes-up the MCU from Halt mode,
- the main oscillator is immediately turned on and the 256 CPU cycle delay is used to stabilize it.

After this start-up delay, the CPU resumes operation by servicing the AWUFH interrupt. The AWU flag and its associated interrupt are cleared by software reading the AWUCSR register.

To compensate for any frequency dispersion of the AWU RC oscillator, it can be calibrated by measuring the clock frequency f_{AWU_RC} and then calculating the right prescaler value. Measurement mode is enabled by setting the AWUM bit in the AWUCSR register in Run mode. This connects f_{AWU_RC} to the Input Capture of the 8-bit Lite timer, allowing the f_{AWU_RC} to be measured using the main oscillator clock as a reference timebase.

Similarities with halt mode

The following AWUFH mode behavior is the same as normal Halt mode:

- The MCU can exit AWUFH mode by means of any interrupt with exit from Halt capability or a reset (see *Section 8.4: Active-halt and halt modes*).
- When entering AWUFH mode, the I bit in the CC register is forced to 0 to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes up immediately.
- In AWUFH mode, the main oscillator is turned off causing all internal processing to be stopped, including the operation of the on-chip peripherals. None of the peripherals are clocked except those which get their clock supply from another clock generator (such as an external or auxiliary oscillator like the AWU oscillator).
- The compatibility of watchdog operation with AWUFH mode is configured by the WDGHALT option bit in the option byte. Depending on this setting, the HALT instruction when executed while the watchdog system is enabled, can generate a watchdog Reset.

Figure 30. AWUF halt timing diagram

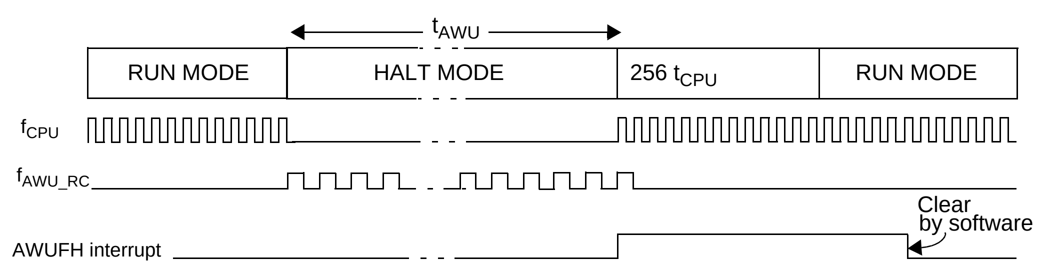
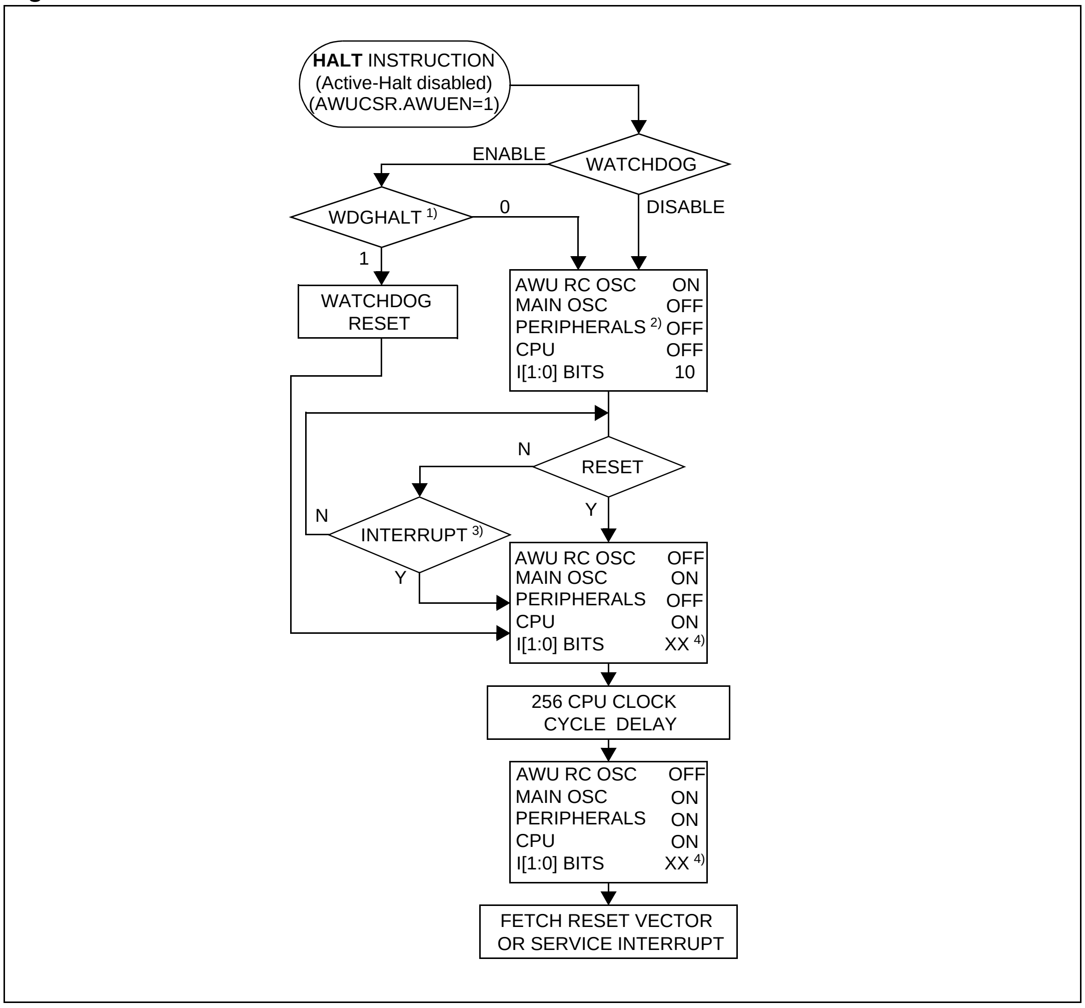


Figure 31. AWUFH mode flowchart



1. WDGHALT is an option bit. See option byte section for more details.
2. Peripheral clocked with an external clock source can still be active.
3. Only an AWUFH interrupt and some specific interrupts can exit the MCU from Halt mode (such as external interrupt). Refer to *Table 17: ST7FOXF1/ST7FOXK1 Interrupt mapping* for more details.
4. Before servicing an interrupt, the CC register is pushed on the stack. The I[1:0] bits of the CC register are set to the current software priority level of the interrupt routine and recovered when the CC register is popped.

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	AWU F	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

9.2.3 Alternate functions

Many ST7s I/Os have one or more alternate functions. These may include output signals from, or input signals to, on-chip peripherals. *Table 2* and *Table 3* describe which peripheral signals can be input/output to which ports.

A signal coming from an on-chip peripheral can be output on an I/O. To do this, enable the on-chip peripheral as an output (enable bit in the peripheral's control register). The peripheral configures the I/O as an output and takes priority over standard I/O programming. The I/O's state is readable by addressing the corresponding I/O data register.

Configuring an I/O as floating enables alternate function input. It is not recommended to configure an I/O as pull-up as this will increase current consumption. Before using an I/O as an alternate input, configure it without interrupt. Otherwise spurious interrupts can occur.

Configure an I/O as input floating for an on-chip peripheral signal which can be input and output.

Caution: I/Os which can be configured as both an analog and digital alternate function need special attention. The user must control the peripherals so that the signals do not arrive at the same time on the same pin. If an external clock is used, only the clock alternate function should be employed on that I/O pin and not the other alternate function.

Figure 32. I/O port general block diagram

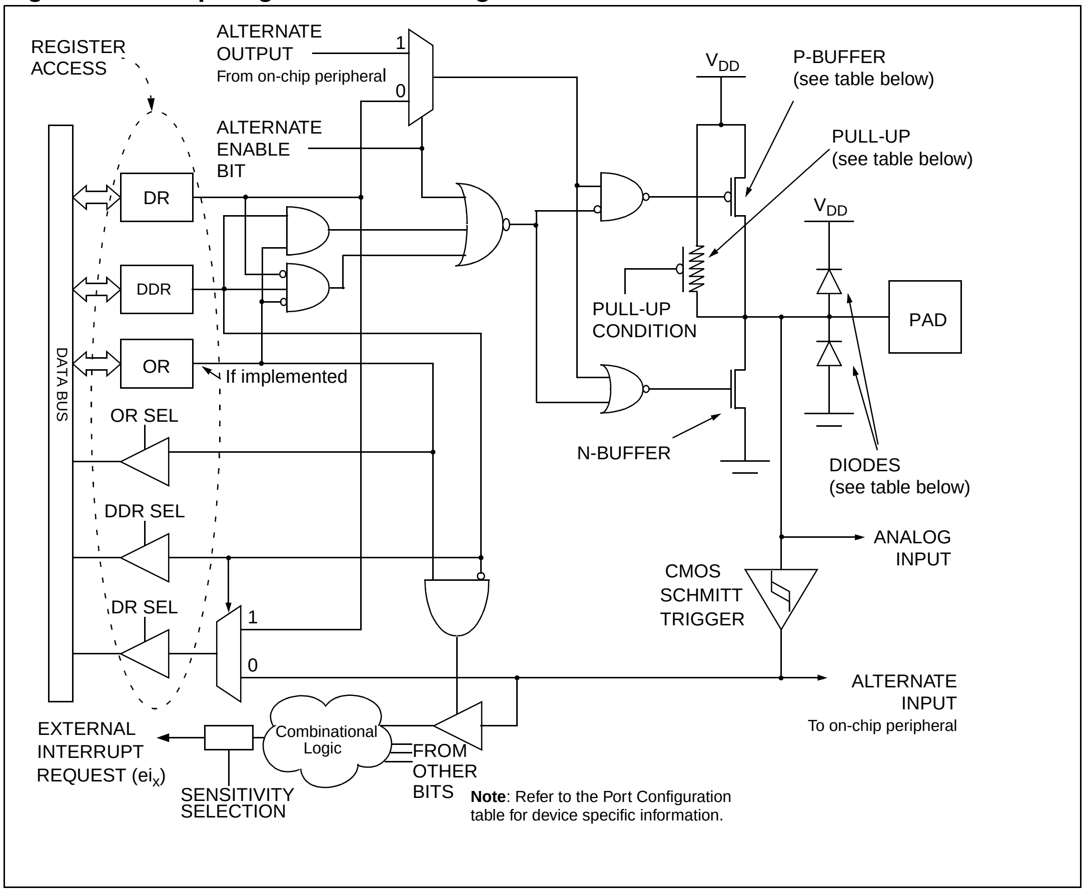
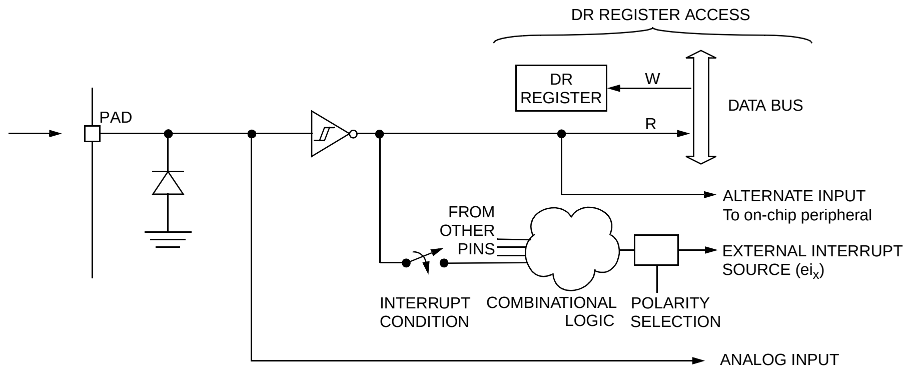
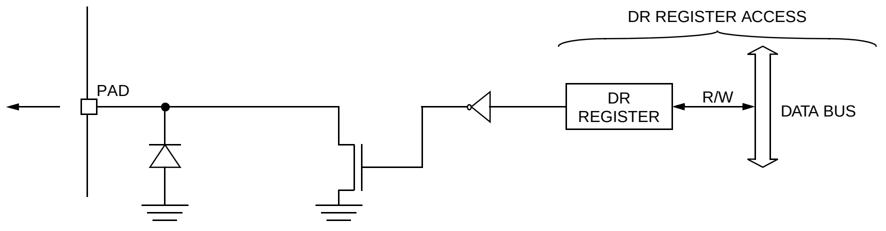
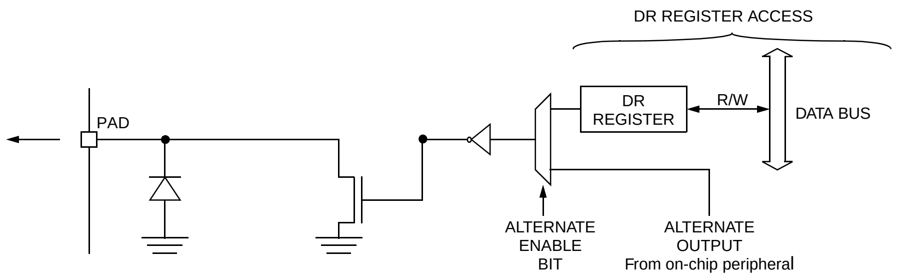


Table 24. I/O port mode options [(1)]

Configuration mode		Pull-Up	P-Buffer	Diodes	
				to V_{DD}	to V_{SS}
Input	Floating with/without Interrupt	Off	Off	On	On
	Pull-up with Interrupt	On			
Output	Push-pull	Off	On		
	Open Drain (logic level)		Off		

1. Off means implemented not activated, On means implemented and activated.

Table 25. ST7FOXF1/ST7FOXK1/ST7FOXK2 I/O port configuration

	Hardware configuration
INPUT [(1)]	 DR REGISTER ACCESS DR REGISTER W R DATA BUS PAD ALTERNATE INPUT To on-chip peripheral FROM OTHER PINS EXTERNAL INTERRUPT SOURCE (ei_x) INTERRUPT CONDITION COMBINATIONAL LOGIC POLARITY SELECTION ANALOG INPUT
OPEN-DRAIN OUTPUT[(2)]	
PUSH-PULL OUTPUT[(2)]	 DR REGISTER ACCESS PAD DR REGISTER R/W DATA BUS ALTERNATE ENABLE BIT ALTERNATE OUTPUT From on-chip peripheral

1. When the I/O port is in input configuration and the associated alternate function is enabled as an output, reading the DR register will read the alternate function output status.
2. When the I/O port is in output configuration and the associated alternate function is enabled as an input, the alternate function reads the pin status given by the DR register content.

9.2.4 Analog alternate function

Configure the I/O as floating input to use an ADC input. The analog multiplexer (controlled by the ADC registers) switches the analog voltage present on the selected pin to the common analog rail, connected to the ADC input.

Analog Recommendations

Do not change the voltage level or loading on any I/O while conversion is in progress. Do not have clocking pins located close to a selected analog pin.

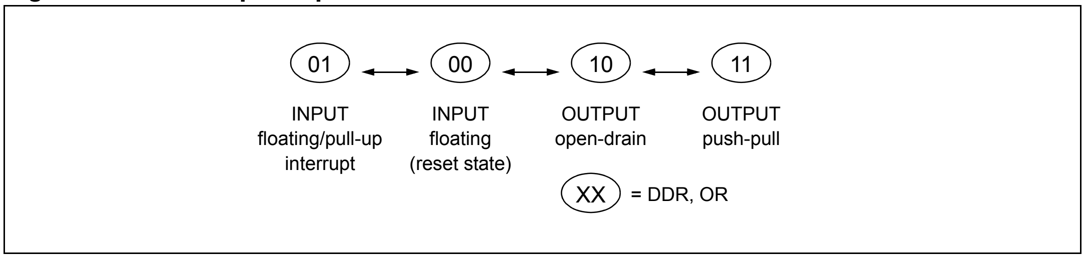
Caution: The analog input voltage level must be within the limits stated in the absolute maximum ratings.

9.3 I/O port implementation

The hardware implementation on each I/O port depends on the settings in the DDR and OR registers and specific I/O port features such as ADC input or open drain.

Switching these I/O ports from one state to another should be done in a sequence that prevents unwanted side effects. Recommended safe transitions are illustrated in *Figure 33*. Other transitions are potentially risky and should be avoided, since they may present unwanted side-effects such as spurious interrupt generation.

Figure 33. Interrupt I/O port state transitions


01
00
10
11
INPUT floating/pull-up interrupt
INPUT floating (reset state)
OUTPUT open-drain
OUTPUT push-pull
XX = DDR, OR

9.4 Unused I/O pins

Unused I/O pins must be connected to fixed voltage levels. Refer to *Section 12.9: I/O port pin characteristics*.

9.5 Low power modes

Table 26. Effect of low power modes on I/O ports

Mode	Description
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.

9.6 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^{2}C 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

Table 30. PC3 pin

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

Table 31. Port configuration

Port	Pin name	Input		Output	
		OR = 0	OR = 1	OR = 0	OR = 1
Port A	PA5:0	floating	pull-up interrupt	open drain	push-pull
	PA7:6	floating	interrupt	true open drain	
Port B	PB7:0	floating	pull-up interrupt	open drain	push-pull
Port C	PC7:4, PC2:0	floating	pull-up interrupt	open drain	push-pull
	PC3	floating	pull-up	open drain	push-pull

Table 32. I/O port register mapping and reset values

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0000h	PADR Reset Value	MSB 0	0	0	0	0	0	0	LSB 0
0001h	PADDR Reset Value	MSB 0	0	0	0	0	0	0	LSB 0
0002h	PAOR Reset Value	MSB 0	0	0	0	0	0	0	LSB 0
0003h	PBDR Reset Value	MSB 0	0	0	0	0	0	0	LSB 0
0004h	PBDDR Reset Value	MSB 0	0	0	0	0	0	0	LSB 0
0005h	PBOR Reset Value	MSB 0	0	0	0	0	0	0	LSB 0
0006h	PCDR Reset Value	MSB 0	0	0	0	0	0	0	LSB 0
0007h	PCDDR Reset Value	MSB 0	0	0	0	0	0	0	LSB 0
0008h	PCOR Reset Value	MSB 0	0	0	0	1	0	0	LSB 0

10 On-chip peripherals

10.1 Watchdog timer (WDG)

10.1.1 Introduction

The Watchdog timer is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The Watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the counter's contents before the T6 bit becomes cleared.

10.1.2 Main features

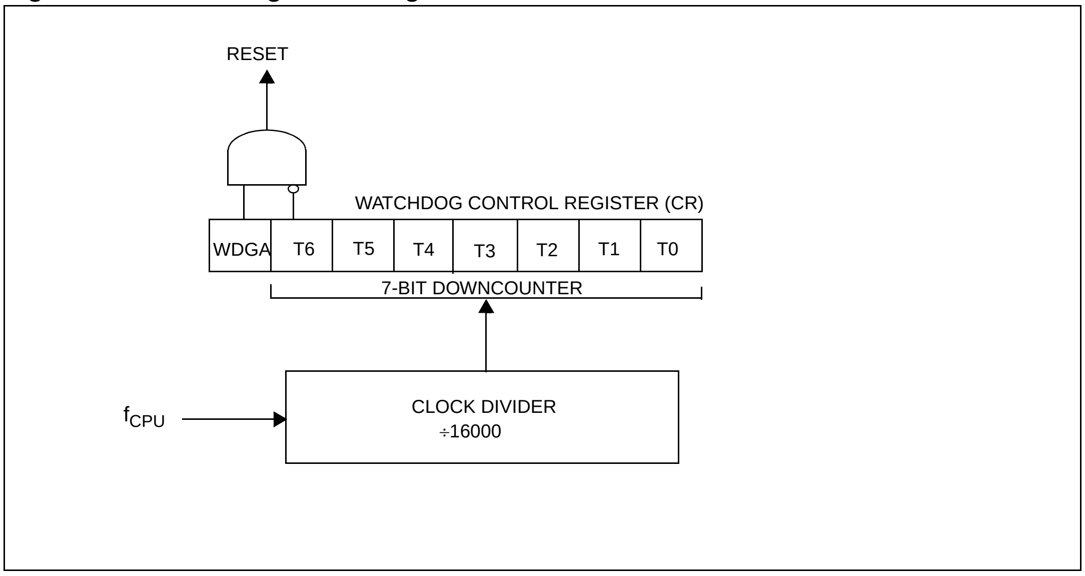
- Programmable free-running downcounter (64 increments of 16000 CPU cycles)
- Programmable reset
- Reset (if watchdog activated) when the T6 bit reaches zero
- Optional reset on HALT instruction (configurable by option byte)
- Hardware Watchdog selectable by option byte

10.1.3 Functional description

The counter value stored in the CR register (bits T[6:0]), is decremented every 16000 machine cycles, and the length of the timeout period can be programmed by the user in 64 increments.

If the watchdog is activated (the WDGA bit is set) and when the 7-bit timer (bits T[6:0]) rolls over from 40h to 3Fh (T6 becomes cleared), it initiates a reset cycle pulling low the $\overline{\text{RESET}}$ pin for typically 30µs.

Figure 34. Watchdog block diagram



The application program must write in the CR register at regular intervals during normal operation to prevent an MCU reset. This downcounter is free-running: it counts down even if the watchdog is disabled. The value to be stored in the CR register must be between FFh and C0h (see *Table 33: Watchdog timing*):

- The WDGA bit is set (watchdog enabled)
- The T6 bit is set to prevent generating an immediate reset
- The T[5:0] bits contain the number of increments which represents the time delay before the watchdog produces a reset.

Following a reset, the watchdog is disabled. Once activated it cannot be disabled, except by a reset.

The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

If the watchdog is activated, the HALT instruction will generate a Reset.

Table 33. Watchdog timing (1)(2)

f_{CPU} = 8 MHz		
WDG counter code	**min [ms]**	**max [ms]**
C0h	1	2
FFh	127	128

1. The timing variation shown in *Table 33* is due to the unknown status of the prescaler when writing to the CR register.
2. The number of CPU clock cycles applied during the Reset phase (256 or 4096) must be taken into account in addition to these timings.

10.1.4 Hardware watchdog option

If Hardware Watchdog is selected by option byte, the watchdog is always active and the WDGA bit in the CR is not used.

Refer to the option byte description in *Section 13 on page 211*.

Using Halt mode with the WDG (WDGHALT option)

If Halt mode with Watchdog is enabled by option byte (No watchdog reset on HALT instruction), it is recommended before executing the HALT instruction to refresh the WDG counter, to avoid an unexpected WDG reset immediately after waking up the microcontroller. Same behavior in active-halt mode.

10.1.5 Interrupts

None.

10.1.6 Register description

Control register (WDGCR)

Reset value: 0111 1111 (7Fh)

7							0
WDGA	T6	T5	T4	T3	T2	T1	T0
Read/Write							

Bit 7 = **WDGA** *Activation bit*

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Note: This bit is not used if the hardware watchdog option is enabled by option byte.

Bits 6:0 = **T[6:0]** *7-bit timer (MSB to LSB)*

These bits contain the decremented value. A reset is produced when it rolls over from 40h to 3Fh (T6 becomes cleared).

Table 34. Watchdog timer register mapping and reset values

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0033h	WDGCR Reset Value	WDGA 0	T6 1	T5 1	T4 1	T3 1	T2 1	T1 1	T0 1

10.2 Dual 12-bit autoreload timer

10.2.1 Introduction

The 12-bit Autoreload timer can be used for general-purpose timing functions. It is based on one or two free-running 12-bit upcounters with an Input Capture register and four PWM output channels. There are 7 external pins:

- Four PWM outputs
- ATIC/LTIC pins for the Input Capture function
- BREAK pins for forcing a break condition on the PWM outputs

10.2.2 Main features

- Single Timer or Dual Timer mode with two 12-bit upcounters (CNTR1/CNTR2) and two 12-bit autoreload registers (ATR1/ATR2)
- Maskable overflow interrupts
- PWM mode
  - Generation of four independent PWMx signals
  - Dead time generation for Half bridge driving mode with programmable dead time
  - Frequency 2 kHz - 4 MHz (@ 8 MHz f_{CPU})
  - Programmable duty-cycles
  - Polarity control
  - Programmable output modes
- Output Compare mode
- Input Capture mode
  - 12-bit Input Capture register (ATICR)
  - Triggered by rising and falling edges
  - Maskable IC interrupt
  - Long range Input Capture
- Internal/External Break control
- Flexible Clock control
- One Pulse mode on PWM2/3
- Force update

Figure 35. Single timer mode (ENCNTR2=0)

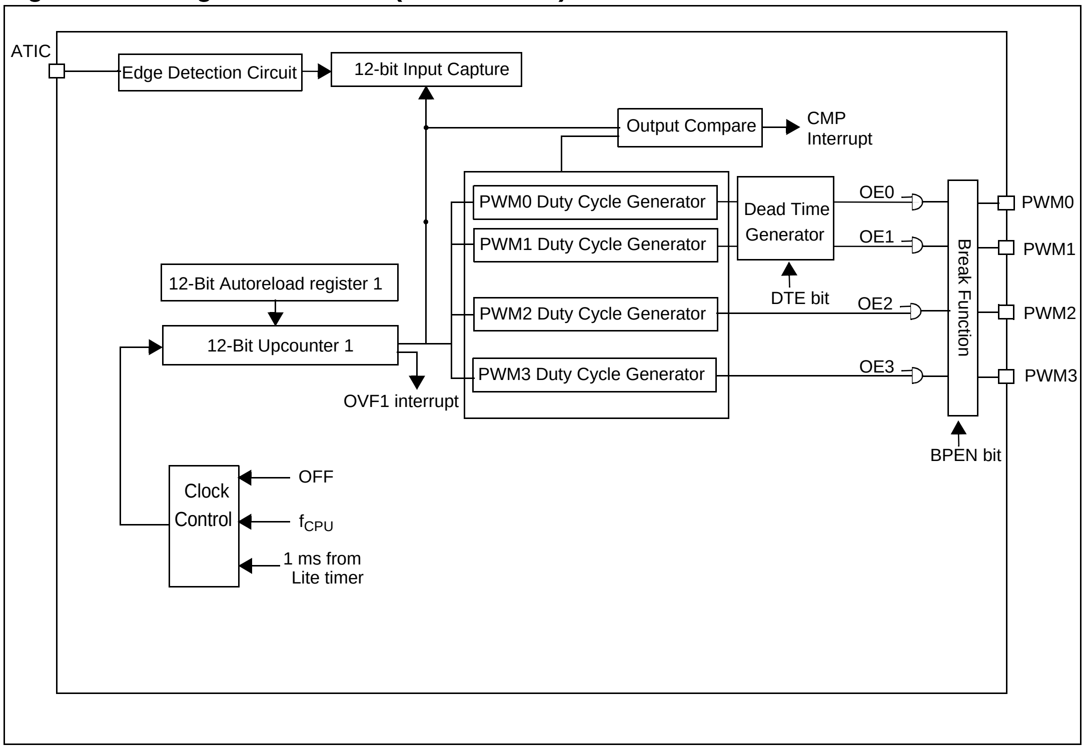
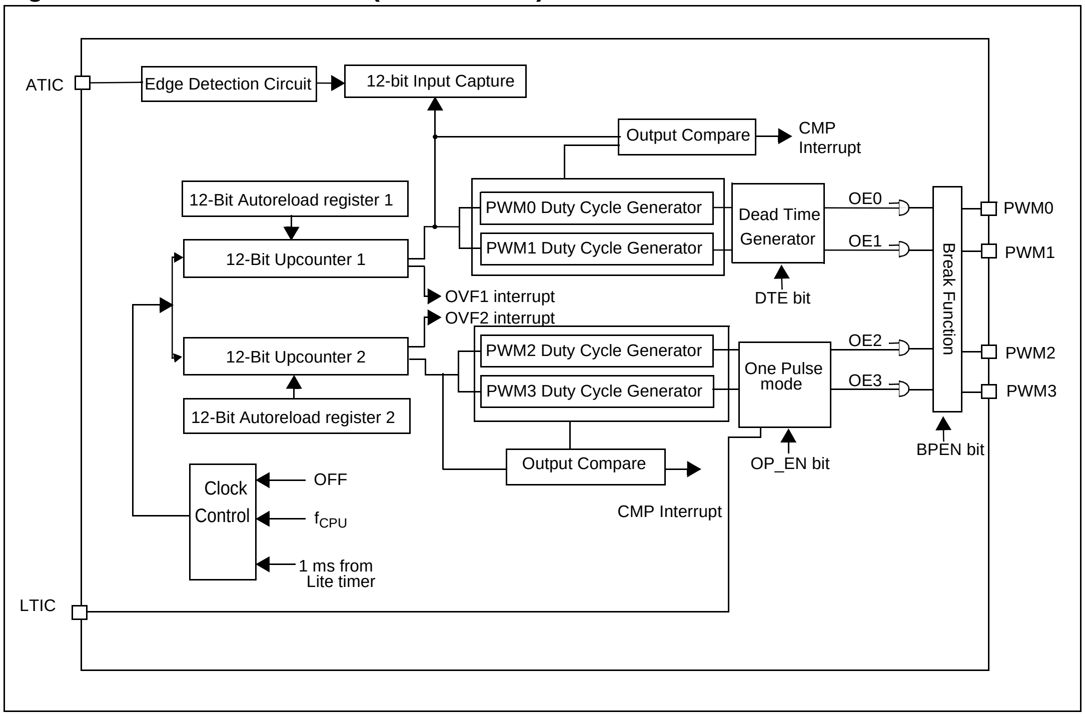


Figure 36. Dual timer mode (ENCNTR2=1)



10.2.3 Functional description

PWM mode

This mode allows up to four Pulse Width Modulated signals to be generated on the PWMx output pins.

- **PWM frequency**

  The four PWM signals can have the same frequency (f_{PWM}) or can have two different frequencies. This is selected by the ENCNTR2 bit which enables Single Timer or Dual Timer mode (see *Figure 35* and *Figure 36*). The frequency is controlled by the counter period and the ATR register value. In Dual Timer mode, PWM2 and PWM3 can be generated with a different frequency controlled by CNTR2 and ATR2.

  $$f_{PWM} = f_{COUNTER} / (4096 - ATR)$$

  Following the above formula, if $f_{COUNTER}$ equals 4 MHz, the maximum value of f_{PWM} is 2 MHz (ATR register value = 4094), and the minimum value is 1 kHz (ATR register value = 0).

  The maximum value of ATR is 4094 because it must be lower than the DC4R value which must be 4095 in this case.

- **Duty cycle**

  The duty cycle is selected by programming the DCRx registers. These are preload registers. The DCRx values are transferred in Active duty cycle registers after an overflow event if the corresponding transfer bit (TRANx bit) is set.

  The TRAN1 bit controls the PWMx outputs driven by counter 1 and the TRAN2 bit controls the PWMx outputs driven by counter 2.

  PWM generation and output compare are done by comparing these active DCRx values with the counter.

  The maximum available resolution for the PWMx duty cycle is:

  $$\text{Resolution} = 1 / (4096 - ATR)$$

  where ATR is equal to 0. With this maximum resolution, 0% and 100% duty cycle can be obtained by changing the polarity.

  At reset, the counter starts counting from 0.

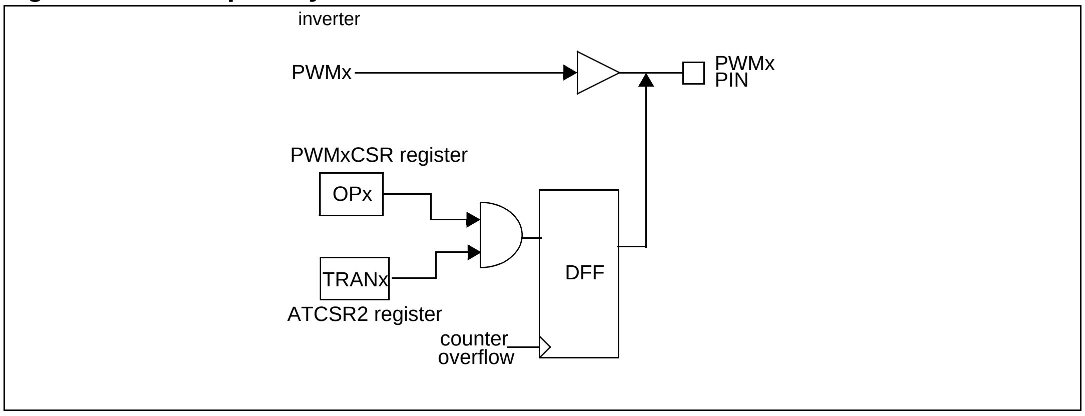
  When a upcounter overflow occurs (OVF event), the preloaded Duty cycle values are transferred to the active Duty Cycle registers and the PWMx signals are set to a high level. When the upcounter matches the active DCRx value the PWMx signals are set to a low level. To obtain a signal on a PWMx pin, the contents of the corresponding active DCRx register must be greater than the contents of the ATR register.

  The maximum value of ATR is 4094 because it must be lower than the DCR value which must be 4095 in this case.

- **Polarity inversion**

  The polarity bits can be used to invert any of the four output signals. The inversion is synchronized with the counter overflow if the corresponding transfer bit in the ATCSR2 register is set (reset value). See *Figure 37*.

Figure 37. PWM polarity inversion



The Data Flip Flop (DFF) applies the polarity inversion when triggered by the counter overflow input.

- **Output control**

  The PWMx output signals can be enabled or disabled using the OEx bits in the PWMCR register.

Figure 38. PWM function

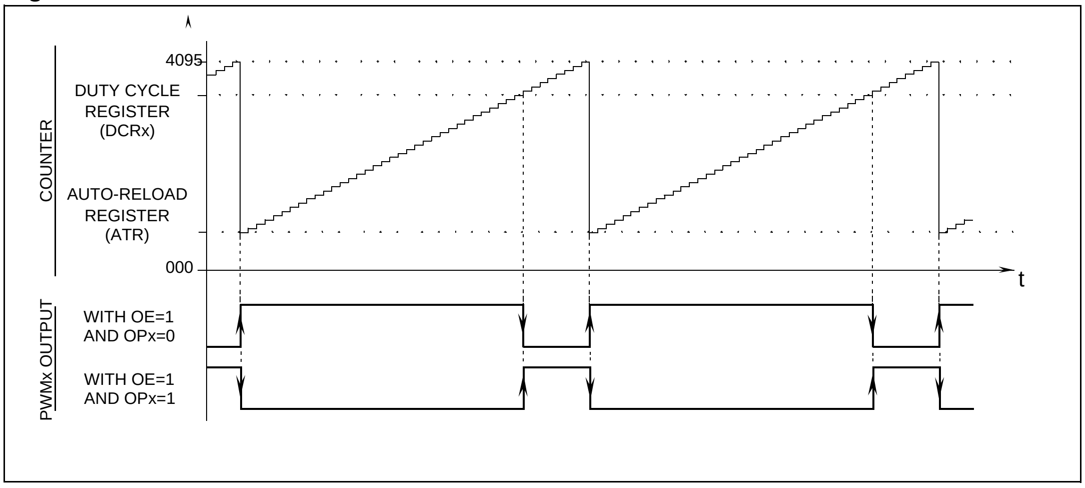
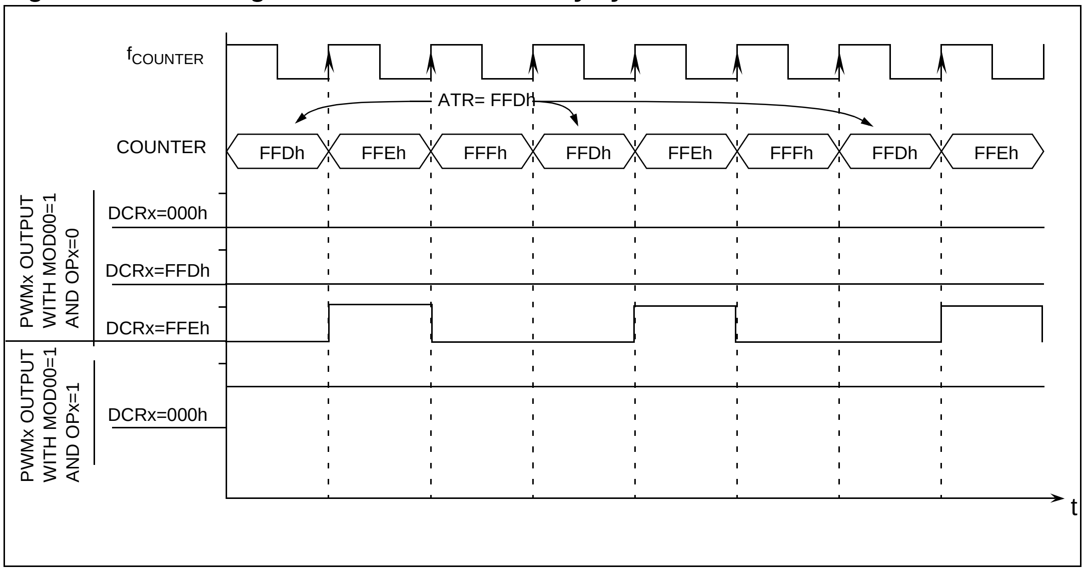


Figure 39. PWM signal from 0% to 100% duty cycle



Dead time generation

A dead time can be inserted between PWM0 and PWM1 using the DTGR register. This is required for half-bridge driving where PWM signals must not be overlapped. The non-overlapping PWM0/PWM1 signals are generated through a programmable dead time by setting the DTE bit.

$$\text{Dead time} = DT[6:0] \times Tcounter1$$

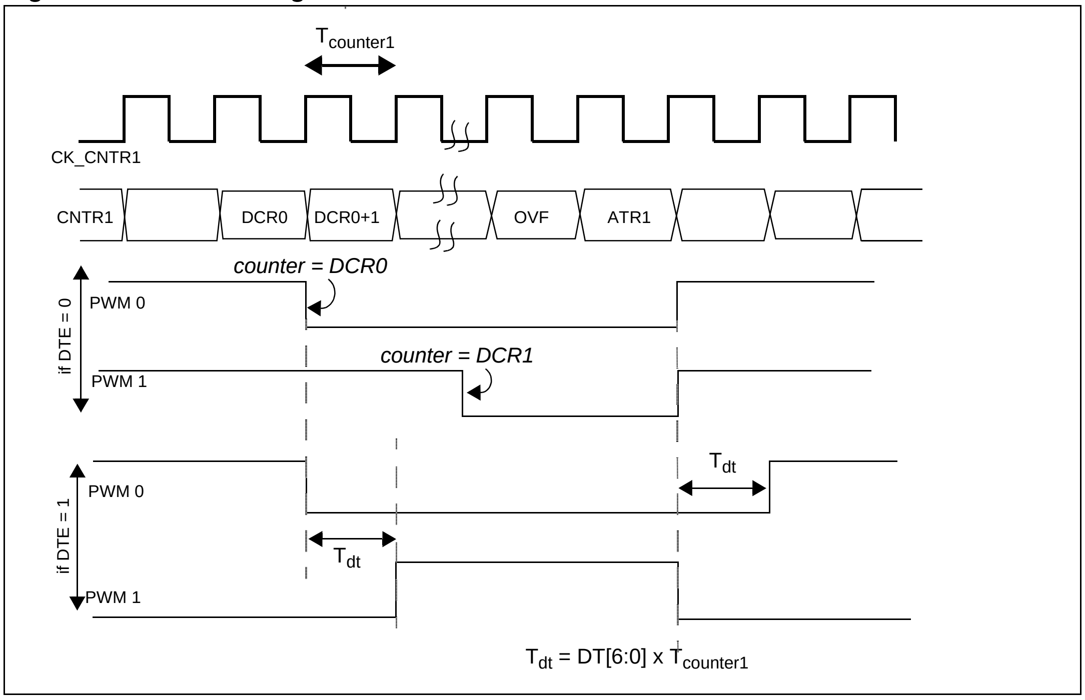
DTGR[7:0] is buffered inside so as to avoid deforming the current PWM cycle. The DTGR effect will take place only after an overflow.

Note: 1 *Dead time is generated only when DTE=1 and DT[6:0] $\neq$ 0. If DTE is set and DT[6:0]=0, PWM output signals will be at their reset state.*

2 *Half Bridge driving is possible only if polarities of PWM0 and PWM1 are not inverted, i.e. if OP0 and OP1 are not set. If polarity is inverted, overlapping PWM0/PWM1 signals will be generated.*

3 *Dead Time generation does not work at 1msec timebase.*

Figure 40. Dead time generation



In the above example, when the DTE bit is set:

- PWM goes low at DCR0 match and goes high at ATR1+Tdt
- PWM1 goes high at DCR0+Tdt and goes low at ATR match.

With this programmable delay (Tdt), the PWM0 and PWM1 signals which are generated are not overlapped.

Break function

The break function can be used to perform an emergency shutdown of the application being driven by the PWM signals.

The break function is activated by the external BREAK pin. This can be selected by using the BRSEL bit in BREAKCR register. In order to use the break function it must be previously enabled by software setting the BPEN bit in the BREAKCR register.

The Break active level can be programmed by the BREDGE bit in the BREAKCR register. When an active level is detected on the BREAK pin, the BA bit is set and the break function is activated. In this case, the PWM signals are forced to BREAK value if respective OEx bit is set in PWMCR register.

Software can set the BA bit to activate the break function without using the BREAK pin. The BREN1 and BREN2 bits in the BREAKEN register are used to enable the break activation on the 2 counters respectively. In Dual Timer mode, the break for PWM2 and PWM3 is enabled by the BREN2 bit. In Single Timer mode, the BREN1 bit enables the break for all PWM channels.

When a break function is activated (BA bit =1 and BREN1/BREN2 =1):

- The break pattern (PWM[3:0] bits in the BREAKCR) is forced directly on the PWMx output pins if respective OEx is set. (after the inverter).
- The 12-bit PWM counter CNTR1 is put to its reset value, i.e. 00h (if BREN1 = 1).
- The 12-bit PWM counter CNTR2 is put to its reset value,i.e. 00h (if BREN2 = 1).
- ATR1, ATR2, Preload and Active DCRx are put to their reset values.
- Counters stop counting.

When the break function is deactivated after applying the break (BA bit goes from 1 to 0 by software), Timer takes the control of PWM ports.

Note: The break function of the ST7FOXK2 is different from the break function of the ST7FOXF1/ST7FOXK1. Refer to Figure 41: ST7FOXF1/ST7FOXK1 Block diagram of break function on page 88 and Figure 42: ST7FOXK2 Block diagram of break function on page 89

Figure 41. ST7FOXF1/ST7FOXK1 Block diagram of break function

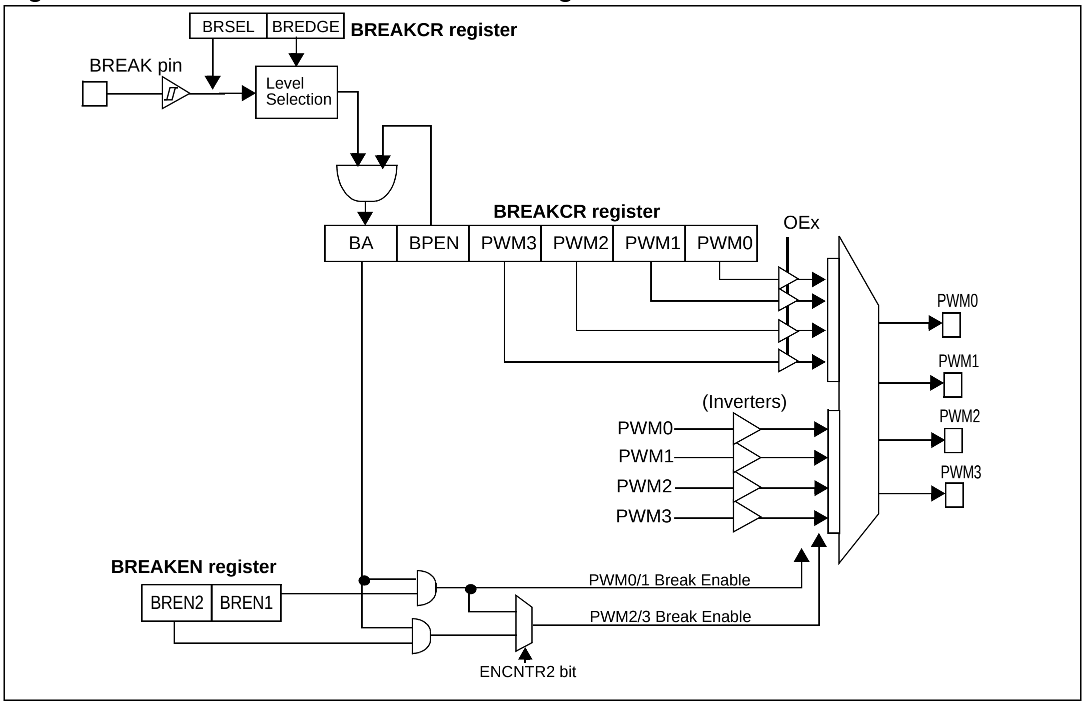
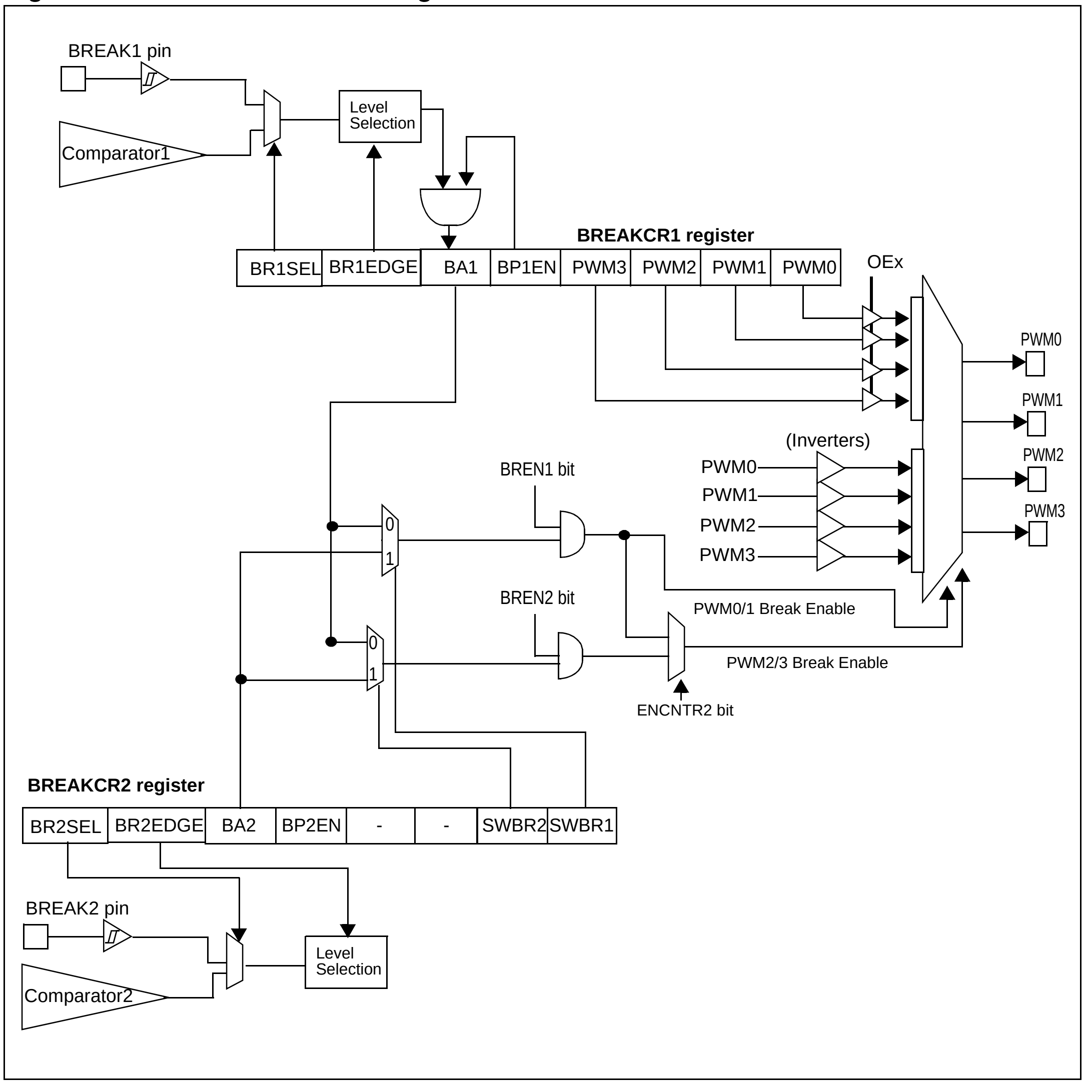


Figure 42. ST7FOXK2 Block diagram of break function



Output compare mode

To use this function, load a 12-bit value in the Preload DCRxH and DCRxL registers.

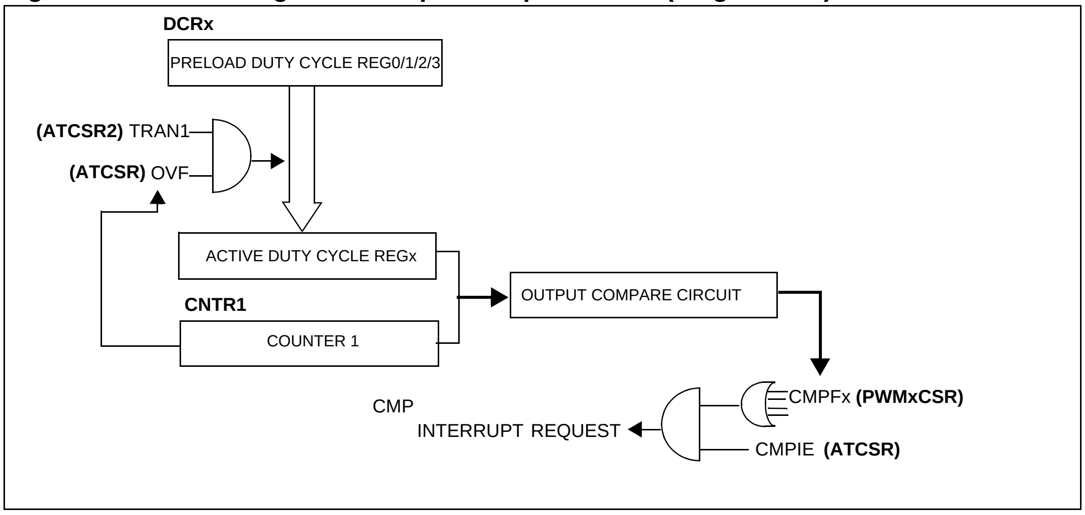
When the 12-bit upcounter CNTR1 reaches the value stored in the Active DCRxH and DCRxL registers, the CMPFx bit in the PWMxCSR register is set and an interrupt request is generated if the CMPIE bit is set.

In Single Timer mode the output compare function is performed only on CNTR1. The difference between both the modes is that, in Single Timer mode, CNTR1 can be compared with any of the four DCR registers, and in Dual Timer mode, CNTR1 is compared with DCR0 or DCR1 and CNTR2 is compared with DCR2 or DCR3.

Note: 1 *The output compare function is only available for DCRx values other than 0 (reset value).*

2 *Duty cycle registers are buffered internally. The CPU writes in Preload Duty Cycle registers and these values are transferred in Active Duty Cycle registers after an overflow event if the corresponding transfer bit (TRANx bit) is set. Output compare is done by comparing these active DCRx values with the counters.*

Figure 43. Block diagram of output compare mode (single timer)



Input capture mode

The 12-bit ATICR register is used to latch the value of the 12-bit free running upcounter CNTR1 after a rising or falling edge is detected on the ATIC pin. When an Input Capture occurs, the ICF bit is set and the ATICR register contains the value of the free running upcounter. An IC interrupt is generated if the ICIE bit is set. The ICF bit is reset by reading the ATICRH/ATICRL register when the ICF bit is set. The ATICR is a read only register and always contains the free running upcounter value which corresponds to the most recent Input Capture. Any further Input Capture is inhibited while the ICF bit is set.

Figure 44. Block diagram of input capture mode

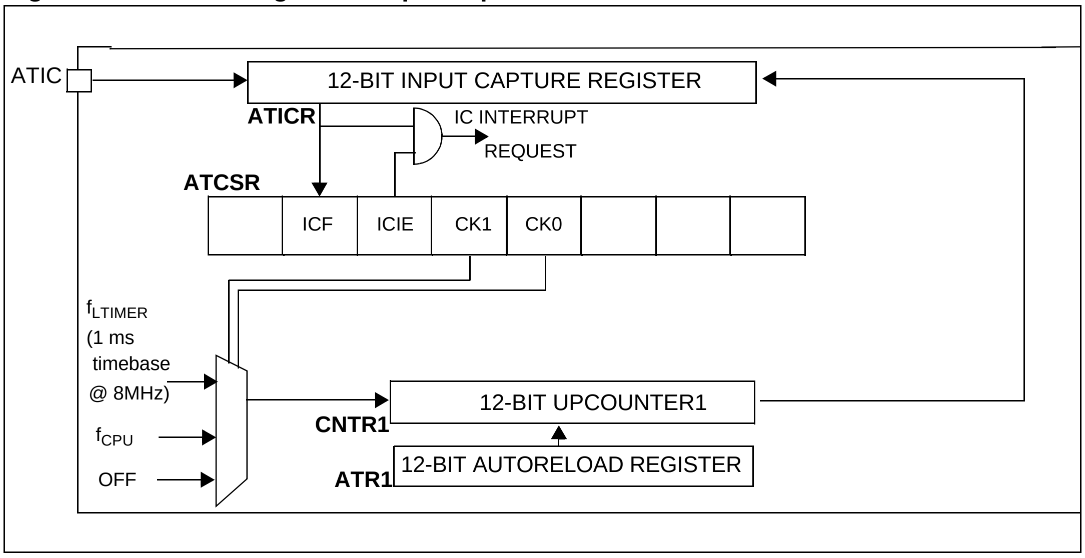
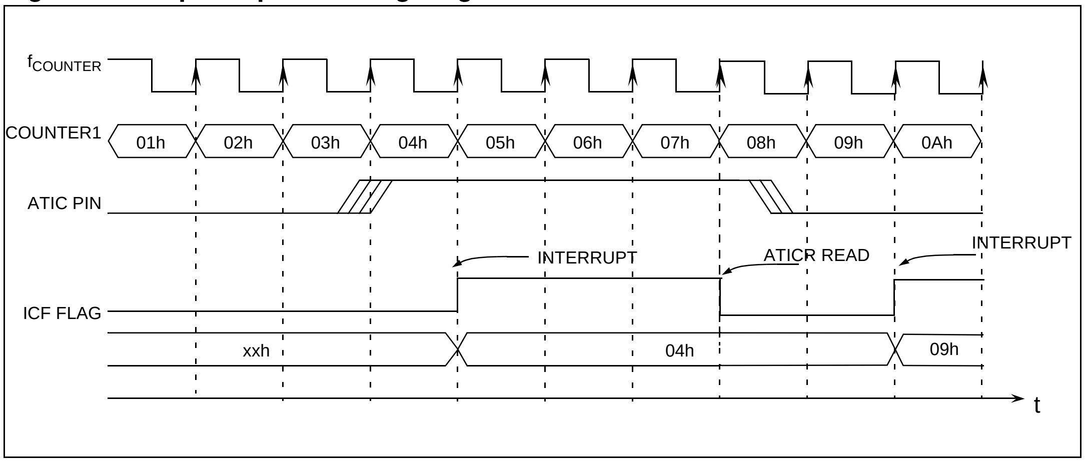

ATIC
12-BIT INPUT CAPTURE REGISTER
ATICR
IC INTERRUPT REQUEST
ATCSR
ICF
ICIE
CK1
CK0
fLTIMER (1 ms timebase @ 8MHz)
fCPU
OFF
12-BIT UPCOUNTER1
CNTR1
12-BIT AUTORELOAD REGISTER
ATR1

Figure 45. Input capture timing diagram



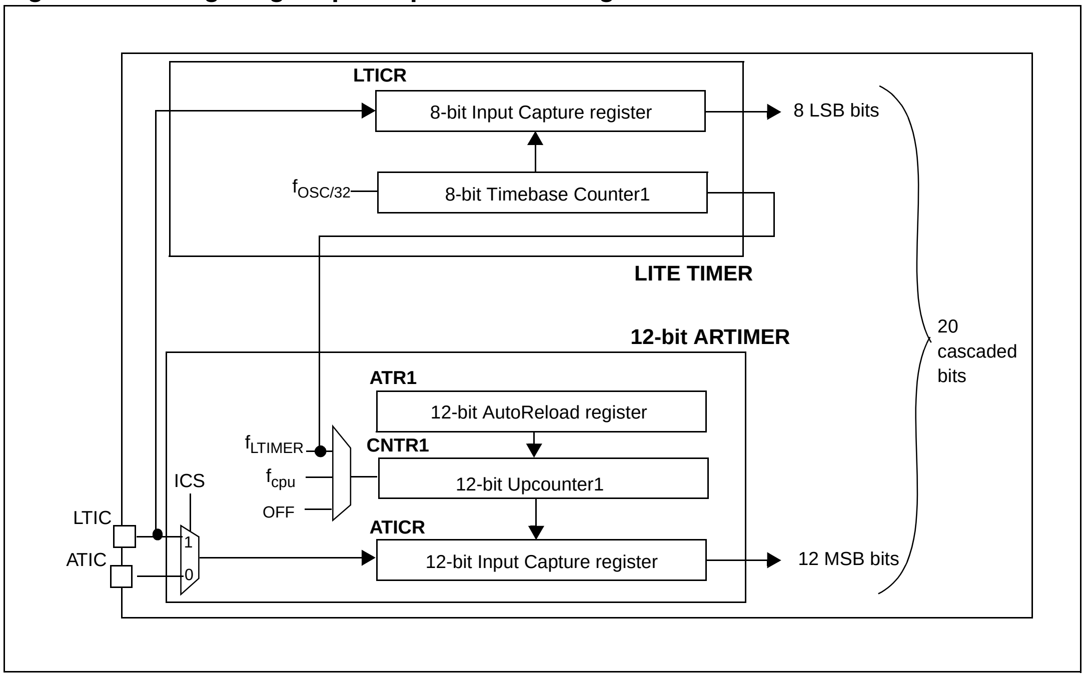
Long range input capture

Pulses that last more than 8 µs can be measured with an accuracy of 4 µs if f_{OSC} equals 8 MHz in the following conditions:

- The 12-bit AT4 timer is clocked by the Lite timer (RTC pulse: CK[1:0] = 01 in the ATCSR register)
- The ICS bit in the ATCSR2 register is set so that the LTIC pin is used to trigger the AT4 timer capture.
- The signal to be captured is connected to LTIC pin
- Input Capture registers LTICR, ATICRH and ATICRL are read

This configuration allows to cascade the Lite timer and the 12-bit AT4 timer to get a 20-bit Input Capture value. Refer to *Figure 46*.

Figure 46. Long range input capture block diagram



Since the Input Capture flags (ICF) for both timers (AT4 timer and LT timer) are set when signal transition occurs, software must mask one interrupt by clearing the corresponding ICIE bit before setting the ICS bit.

If the ICS bit changes (from 0 to 1 or from 1 to 0), a spurious transition might occur on the Input Capture signal because of different values on LTIC and ATIC. To avoid this situation, it is recommended to do as follows:

1. First, reset both ICIE bits.
2. Then set the ICS bit.
3. Reset both ICF bits.
4. And then set the ICIE bit of desired interrupt.

Computing a pulse length in long Input Capture mode is not straightforward since both timers are used. The following steps are required:

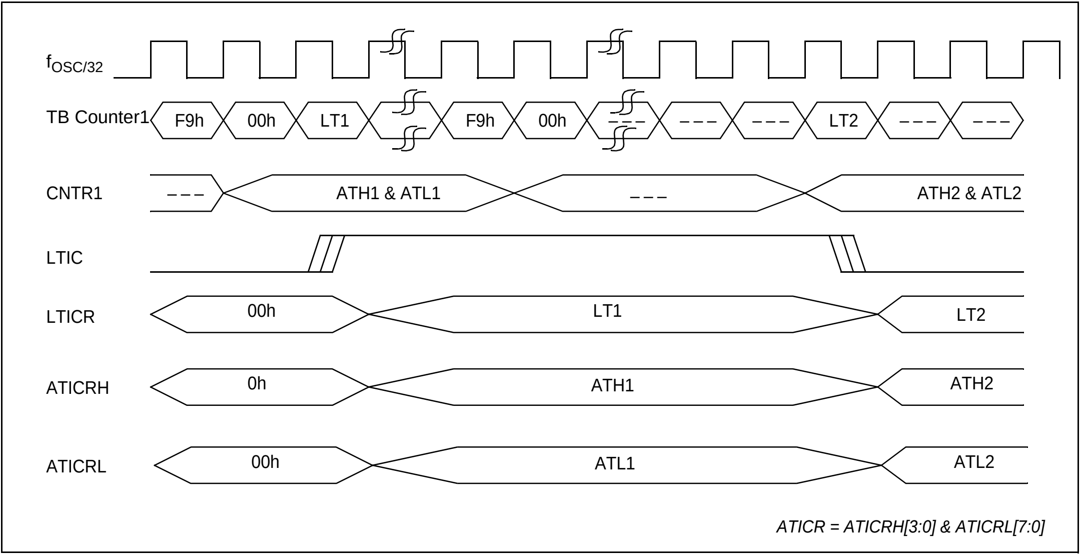
1. At the first Input Capture on the rising edge of the pulse, we assume that values in the registers are the following:
   - LTICR = LT1
   - ATICRH = ATH1
   - ATICRL = ATL1
   - Hence ATICR1 [11:0] = ATH1 & ATL1. Refer to *Figure 47 on page 93*.
2. At the second Input Capture on the falling edge of the pulse, we assume that the values in the registers are as follows:
   - LTICR = LT2
   - ATICRH = ATH2
   - ATICRL = ATL2
   - Hence ATICR2 [11:0] = ATH2 & ATL2.

Now pulse width P between first capture and second capture is given by:

$$P = \mathrm{decimal} \times (F9 - LT1 + LT2 + 1) \times 0.004\mathrm{ms} \\ + \mathrm{decimal}((FFF \times N) + N + ATICR2 - ATICR1 - 1) \times 1\mathrm{ms}$$

where N is the number of overflows of 12-bit CNTR1.

Figure 47. Long range input capture timing diagram



One pulse mode

One Pulse mode can be used to control PWM2/3 signal with an external LTIC pin. This mode is available only in Dual Timer mode i.e. only for CNTR2, when the OP_EN bit in PWM3CSR register is set.

One Pulse mode is activated by the external LTIC input. The active edge of the LTIC pin is selected by the OPEDGE bit in the PWM3CSR register.

After getting the active edge of the LTIC pin, CNTR2 is reset (000h) and PWM3 is set to high. CNTR2 starts counting from 000h, when it reaches the active DCR3 value then PWM3 goes low. Till this time, any further transitions on the LTIC signal will have no effect. If there are LTIC transitions after CNTR2 reaches DCR3 value, CNTR2 is reset again and PWM3 goes high.

If there is no LTIC active edge, CNTR2 counts until it reaches the ATR2 value, then it is reset again and PWM3 is set to high. The counter again starts counting from 000h, when it reaches the active DCR3 value PWM3 goes low, the counter counts until it reaches ATR2, it resets and PWM3 is set to high and so on.

The same operation applies for PWM2, but in this case the comparison is done on DCR2. OP_EN and OPEDGE bits take effect on the fly and are not synchronized with Counter 2 overflow. The output bit OP2/3 can be used to inverse the polarity of PWM2/3 in one-pulse mode. The update of these bits (OP2/3) is synchronized with the counter 2 overflow, they will be updated if the TRAN2 bit is set.

The time taken from activation of LTIC input and CNTR2 reset is between 1 and 2 t_{CPU} cycles, that is, 125 ns to 250 ns (with 8-MHz f_{CPU}).

Lite timer Input Capture interrupt should be disabled while 12-bit ARtimer is in One Pulse mode. This is to avoid spurious interrupts.

The priority of the various conditions for PWM3 is the following: Break > one-pulse mode with active LTIC edge > Forced overflow by s/w > one-pulse mode without active LTIC edge > normal PWM operation.

It is possible to update DCR2/3 and OP2/3 at the counter 2 reset, the update is synchronized with the counter reset. This is managed by the overflow interrupt which is generated if counter is reset either due to ATR match or active pulse at LTIC pin. DCR2/3 and OP2/3 update in one-pulse mode is performed dynamically using a software force update. DCR3 update in this mode is not synchronized with any event. That may lead to a longer next PWM3 cycle duration than expected just after the change.

In One Pulse mode ATR2 value must be greater than DCR2/3 value for PWM2/3. (opposite to normal PWM mode).

If there is an active edge on the LTIC pin after the counter has reset due to an ATR2 match, then the timer again gets reset and appears as modified Duty cycle depending on whether the new DCR value is less than or more than the previous value.

The TRAN2 bit should be set along with the FORCE2 bit with the same instruction after a write to the DCR register.

ATR2 value should be changed after an overflow in one pulse mode to avoid any irregular PWM cycle.

When exiting from one pulse mode, the OP_EN bit in the PWM3CSR register should be reset first and then the ENCNTR2 bit (if counter 2 must be stopped).

How to enter one pulse mode

The steps required to enter One Pulse mode are the following:

1. Load ATR2H/ATR2L with required value.
2. Load DCR3H/DCR3L for PWM3. ATR2 value must be greater than DCR3.
3. Set OP3 in PWM3CSR if polarity change is required.
4. Select CNTR2 by setting ENCNTR2 bit in ATCSR2.
5. Set TRAN2 bit in ATCSR2 to enable transfer.
6. "Wait for Overflow" by checking the OVF2 flag in ATCSR2.
7. Select counter clock using CK<1:0> bits in ATCSR.
8. Set OP_EN bit in PWM3CSR to enable one-pulse mode.
9. Enable PWM3 by OE3 bit of PWMCR.

The "Wait for Overflow" in step 6 can be replaced by a forced update.

Follow the same procedure for PWM2 with the bits corresponding to PWM2.

Note: When break is applied in one-pulse mode, CNTR2, DCR2/3 & ATR2 registers are reset. So, these registers have to be initialized again when break is removed.

Figure 48. Block diagram of one pulse mode

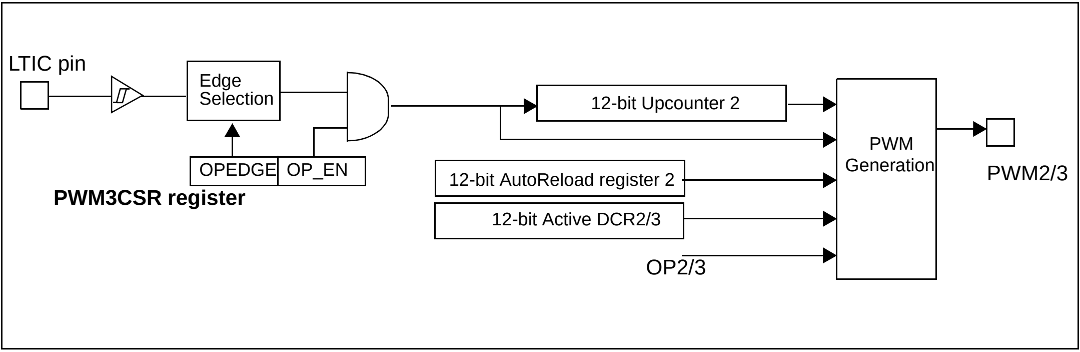
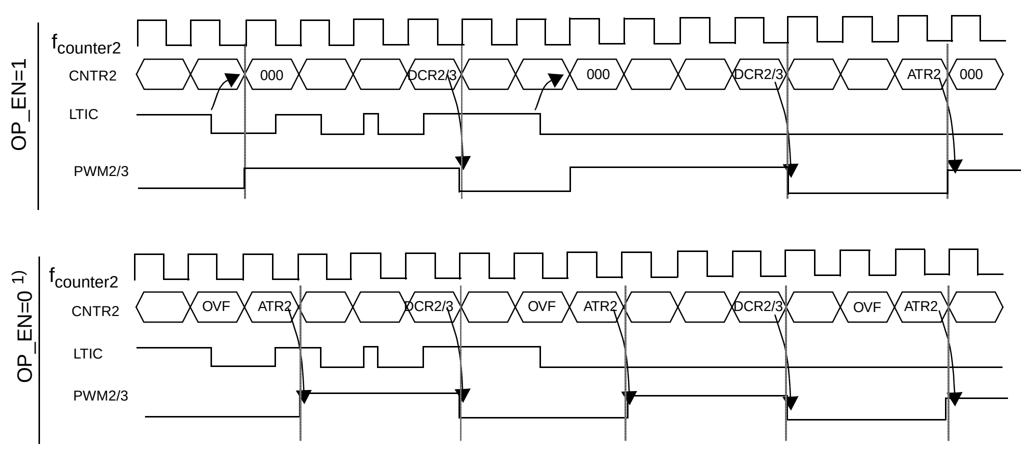
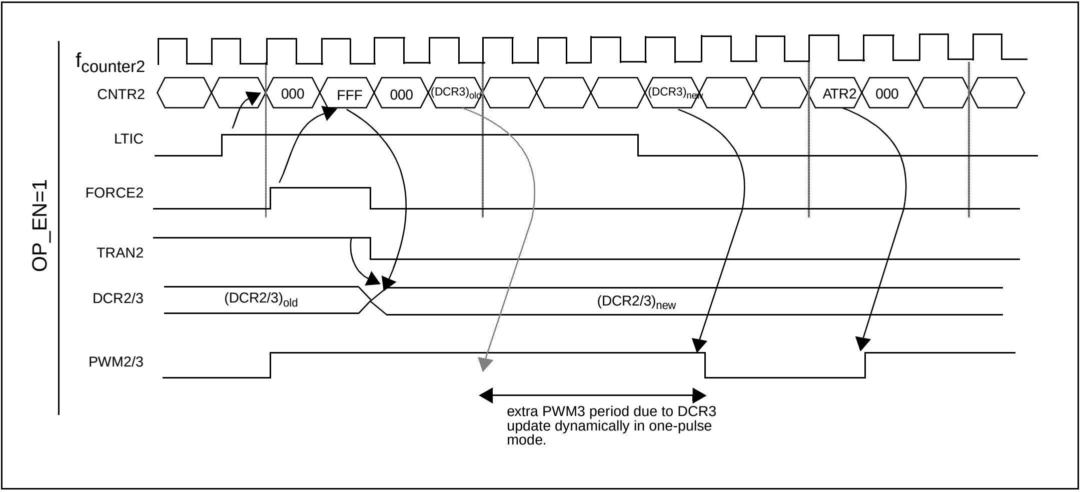


Figure 49. One pulse mode and PWM timing diagram



Note 1: When OP_EN=0, LTIC edges are not taken into account as the timer runs in PWM mode.

Figure 50. Dynamic DCR2/3 update in one pulse mode



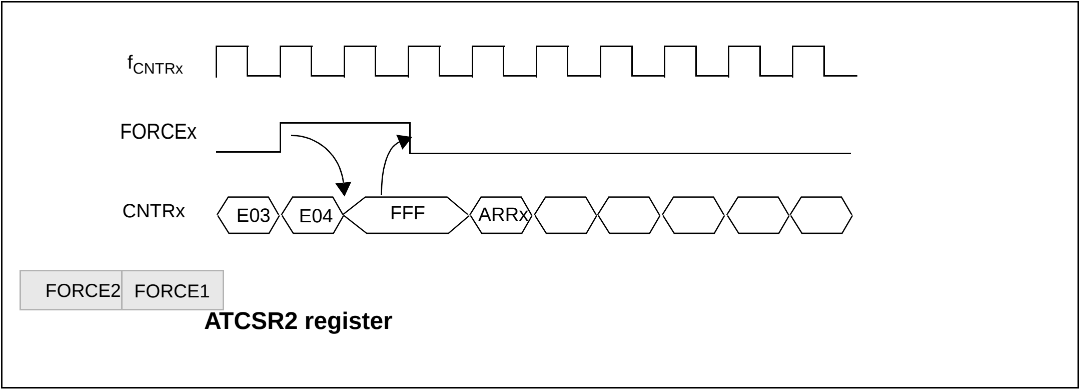
Force update

In order not to wait for the counter$_x$ overflow to load the value into active DCRx registers, a programmable counter$_x$ overflow is provided. For both counters, a separate bit is provided which when set, make the counters start with the overflow value, i.e. FFFh. After overflow, the counters start counting from their respective auto reload register values.

These bits are FORCE1 and FORCE2 in the ATCSR2 register. FORCE1 is used to force an overflow on Counter 1 and, FORCE2 is used for Counter 2. These bits are set by software and reset by hardware after the respective counter overflow event has occurred.

This feature can be used at any time. All related features such as PWM generation, Output Compare, Input Capture, One-pulse (refer to *Figure 50: Dynamic DCR2/3 update in one pulse mode*) etc. can be used this way.

Figure 51. Force overflow timing diagram



10.2.4 Low power modes

Table 35. Effect of low power modes on autoreload timer

Mode	Description
Wait	No effect on AT timer
Halt	AT timer halted.

10.2.5 Interrupts

Table 36. Description of interrupt events

Interrupt Event	Event Flag	Enable Control bit	Exit from Wait	Exit from Halt	Exit from Active-Halt
Overflow Event	OVF1	OVIE1	Yes	No	Yes
AT4 IC Event	ICF	ICIE	Yes	No	No
Overflow Event2	OVF2	OVIE2	Yes	No	No

Note: *The AT4 IC is connected to an interrupt vector. The OVF event is mapped on a separate vector (see Interrupts chapter).*
They generate an interrupt if the enable bit is set in the ATCSR register and the interrupt mask in the CC register is reset (RIM instruction).

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.

Bit 0 = **CMPIE** *Compare Interrupt Enable bit*

This bit is read/write by software and cleared by hardware after a reset. it can be used to mask the interrupt generated when any of the cmpfx bit is set.

0: Output Compare Interrupt Disabled.

1: Output Compare Interrupt Enabled.

Counter register 1 High (CNTR1H)

Reset value: 0000 0000 (00h)

15							8
0	0	0	0	CNTR1_ 11	CNTR1_ 10	CNTR1_9	CNTR1_8
Read only							

Counter register 1 Low (CNTR1L)

Reset value: 0000 0000 (00h)

7							0
CNTR1_7	CNTR1_6	CNTR1_5	CNTR1_4	CNTR1_3	CNTR1_2	CNTR1_1	CNTR1_0
Read only							

Bits 15:12 = Reserved

Bits 11:0 = **CNTR1[11:0]** *Counter value*

This 12-bit register is read by software and cleared by hardware after a reset. The counter CNTR1 increments continuously as soon as a counter clock is selected. To obtain the 12-bit value, software should read the counter value in two consecutive read operations. As there is no latch, it is recommended to read LSB first. In this case, CNTR1H can be incremented between the two read operations and to have an accurate result when $f_{timer}=f_{CPU}$, special care must be taken when CNTR1L values close to FFh are read.

When a counter overflow occurs, the counter restarts from the value specified in the ATR1 register.

Autoreload register (ATR1H)

Reset value: 0000 0000 (00h)

15							8
0	0	0	0	ATR11	ATR10	ATR9	ATR8
Read/write							

Autoreload register (ATR1L)

Reset value: 0000 0000 (00h)

7							0
ATR7	ATR6	ATR5	ATR4	ATR3	ATR2	ATR1	ATR0
Read/write							

Bits 11:0 = **ATR1[11:0]** *Autoreload register 1:*
This is a 12-bit register which is written by software. The ATR1 register value is automatically loaded into the upcounter CNTR1 when an overflow occurs. The register value is used to set the PWM frequency.

PWM output control register (PWMCR)

Reset value: 0000 0000 (00h)

7							0
0	OE3	0	OE2	0	OE1	0	OE0
Read/write							

Bits 7:0 = **OE[3:0]** *PWMx output enable* bits

These bits are set and cleared by software and cleared by hardware after a reset.

0: PWM mode disabled. PWMx Output Alternate function disabled (I/O pin free for general purpose I/O)

1: PWM mode enabled

PWMX control status register (PWMxCSR)

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	OP_EN	OPEDGE	OPx	CMPFx
Read/write							

Bits 7:4= Reserved, must be kept cleared.

Bit 3 = OP_EN *One Pulse Mode Enable* bit

This bit is read/write by software and cleared by hardware after a reset. This bit enables the One Pulse feature for PWM2 and PWM3 **(only available for PWM3CSR)**

0: One Pulse mode disable for PWM2/3.

1: One Pulse mode enable for PWM2/3.

Bit 2 = OPEDGE *One Pulse Edge Selection bit*

This bit is read/write by software and cleared by hardware after a reset. This bit selects the polarity of the LTIC signal for One Pulse feature. This bit will be effective only if OP_EN bit is set **(only available for PWM3CSR)**

0: Falling edge of LTIC is selected.

1: Rising edge of LTIC is selected.

Bit 1 = **OPx** *PWMx Output Polarity bit*

This bit is read/write by software and cleared by hardware after a reset. This bit selects the polarity of the PWM signal.

0: The PWM signal is not inverted.

1: The PWM signal is inverted.

Bit 0 = **CMPFx** PWMx *Compare flag*

This bit is set by hardware and cleared by software by reading the PWMxCSR register. It indicates that the upcounter value matches the Active DCRx register value.

0: Upcounter value does not match DCRx value.

1: Upcounter value matches DCRx value.

Break control register 1 (BREAKCR1)

Reset value: 0000 0000 (00h)

7							0
BR1SEL	BR1EDGE	BA1	BP1EN	PWM3	PWM2	PWM1	PWM0
Read/write							

Bit 7 = **BR1SEL** *Break 1 input selection bit* (**only available on ST7FOXK2, reserved for ST7FOXF1/ST7FOXK1**)

This bit is read/write by software and cleared by hardware after reset. It selects the active Break 1 signal from external BREAK1 pin and the output of the comparator.

0: External BREAK1 pin is selected for break mode.

1: Comparator 1 output is selected for break mode.

Bit 6 = **BR1EDGE** *Break 1 input edge selection bit* (**BREDGE** on ST7FOXF1/ST7FOXK1)

This bit is read/write by software and cleared by hardware after reset. It selects the active level of Break 1 signal.

0: Low level of Break 1selected as active level

1: High level of Break 1selected as active level

Bit 5 = **BA1** *Break 1 Active bit* (**BA** on ST7FOXF1/ST7FOXK1)

This bit is read/write by software, cleared by hardware after reset and set by hardware when the active level defined by the BR1EDGE bit is applied on the BREAK1 pin. It activates/deactivates the Break 1function.

0: Break 1not active

1: Break 1active

Bit 4 = **BP1EN** *Break 1Pin Enable bit* (**BPEN** on ST7FOXF1/ST7FOXK1)

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

0: Break 1pin disabled

1: Break 1pin enabled

Bits 3:0 = **PWM[3:0]** *Break Pattern bits*

These bits are read/write by software and cleared by hardware after a reset. They are used to force the four PWMx output signals into a stable state when the Break function is active and corresponding OEx bit is set.

Break control register 2 (BREAKCR2)

Reset value: 0000 0000 (00h)

7							0
BR2SEL	BR2EDGE	BA2	BP2EN	-	-	SWBR2	SWBR1
Read/write							

Note: This register is available on ST7FOXK2 only

Bit 7 = **BR2SEL** *Break 2 input selection bit*

This bit is read/write by software and cleared by hardware after reset. It selects the active Break 2 signal from external BREAK2 pin and the output of the comparator.

0: External BREAK2 pin is selected for break mode.

1: Comparator 2 output is selected for break mode.

Bit 6 = **BR2EDGE** *Break 2 input edge selection bit*

This bit is read/write by software and cleared by hardware after reset. It selects the active level of Break 2 signal.

0: Low level of Break 2 selected as active level

1: High level of Break 2 selected as active level

Bit 5 = **BA2** *Break 2 Active bit*

This bit is read/write by software, cleared by hardware after reset and set by hardware when the active level defined by the BR2EDGE bit is applied on the BREAK2 pin. It activates/deactivates the Break 2 function.

0: Break 2 not active

1: Break 2 active

Bit 4 = **BP2EN** *Break 2 pin enable bit*

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

0: BREAK2 pin disabled

1: BREAK2 pin enabled

Bits 3:2 = Reserved, must be kept cleared

Bit 1 = **SWBR2** *Switch Break for counter 2 bit*

This bit is read/write by software. While BREN2 is set, it selects BA1 or BA2 to control PWM2/3 if ENCNTR2 bit is set.

0: BA1 selected

1: BA2 selected

Bit 0 = **SWBR1** *Switch Break for counter 1 bit*

This bit is read/write by software. While BREN1 is set, it selects BA1 or BA2 to control PWM0/1 by default and also PWM2/3 if ENCNTR2 bit is reset.

0: BA1 selected

1: BA2 selected

PWMx Duty Cycle register High (DCRxH)

Reset value: 0000 0000 (00h)

15							8
0	0	0	0	DCR11	DCR10	DCR9	DCR8
Read/write							

Bits 15:12 = Reserved.

PWMx Duty Cycle register Low (DCRxL)

Reset value: 0000 0000 (00h)

7							0
DCR7	DCR6	DCR5	DCR4	DCR3	DCR2	DCR1	DCR0
Read/write							

Bits 11:0 = **DCRx[11:0]** *PWMx Duty Cycle Value:* this 12-bit value is written by software. It defines the duty cycle of the corresponding PWM output signal (see *Figure 38*).

In PWM mode (OEx=1 in the PWMCR register) the DCR[11:0] bits define the duty cycle of the PWMx output signal (see *Figure 38*). In Output Compare mode, they define the value to be compared with the 12-bit upcounter value.

Input Capture register High (ATICRH)

Reset value: 0000 0000 (00h)

15							8
0	0	0	0	ICR11	ICR10	ICR9	ICR8
Read only							

Bits 15:12 = Reserved.

Input Capture register Low (ATICRL)

Reset value: 0000 0000 (00h)

7							0
ICR7	ICR6	ICR5	ICR4	ICR3	ICR2	ICR1	ICR0
Read only							

Bits 11:0 = **ICR[11:0]** *Input Capture Data*.

This is a 12-bit register which is readable by software and cleared by hardware after a reset. The ATICR register contains captured the value of the 12-bit CNTR1 register when a rising or falling edge occurs on the ATIC or LTIC pin (depending on ICS). Capture will only be performed when the ICF flag is cleared.

Break Enable register (BREAKEN)

Reset value: 0000 0011 (03h)

7							0
0	0	0	0	0	0	BREN2	BREN1
Read/write							

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

Bit 1 = **BREN2** *Break Enable for Counter 2 bit*

This bit is read/write by software. It enables the break functionality for Counter2 if BA bit is set in BREAKCR. It controls PWM2/3 if ENCNTR2 bit is set.

0: No Break applied for CNTR2

1: Break applied for CNTR2

Bit 0 = **BREN1** *Break Enable for Counter 1 bit*

This bit is read/write by software. It enables the break functionality for Counter1. If BA bit is set, it controls PWM0/1 by default, and controls PWM2/3 also if ENCNTR2 bit is reset.

0: No Break applied for CNTR1

1: Break applied for CNTR1

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.

Bit 1= **TRAN2** *Transfer enable2 bit*

This bit is read/write by software, cleared by hardware after each completed transfer and set by hardware after reset. It controls the transfers on CNTR2.

It allows the value of the Preload DCRx registers to be transferred to the Active DCRx registers after the next overflow event.

The OPx bits are transferred to the shadow OPx bits in the same way.

Note: 1 DCR2/3 transfer will be controlled using this bit if ENCNTR2 bit is set.

2 This bit must not be reset by software

Bit 0 = **TRAN1** *Transfer enable 1 bit*

This bit is read/write by software, cleared by hardware after each completed transfer and set by hardware after reset. It controls the transfers on CNTR1. It allows the value of the Preload DCRx registers to be transferred to the Active DCRx registers after the next overflow event.

The OPx bits are transferred to the shadow OPx bits in the same way.

Note: 1 DCR0,1 transfers are always controlled using this bit.

2 DCR2/3 transfer will be controlled using this bit if ENCNTR2 is reset.

3 This bit must not be reset by software

Autoreload register 2 (ATR2H)

Reset value: 0000 0000 (00h)

15							8
0	0	0	0	ATR11	ATR10	ATR9	ATR8
Read/write							

Autoreload register (ATR2L)

Reset value: 0000 0000 (00h)

7							0
ATR7	ATR6	ATR5	ATR4	ATR3	ATR2	ATR1	ATR0
Read/write							

Bits 11:0 = **ATR2[11:0]** *Autoreload register 2*

This is a 12-bit register which is written by software. The ATR2 register value is automatically loaded into the upcounter CNTR2 when an overflow of CNTR2 occurs. The register value is used to set the PWM2/PWM3 frequency when ENCNTR2 is set.

Dead Time Generator register (DTGR)

Reset value: 0000 0000 (00h)

7							0
DTE	DT6	DT5	DT4	DT3	DT2	DT1	DT0
Read/write							

Bit 7 = **DTE** *Dead Time Enable bit*

This bit is read/write by software. It enables a dead time generation on PWM0/PWM1.

0: No Dead time insertion.

1: Dead time insertion enabled.

Bits 6:0 = **DT[6:0]** *Dead Time value*

These bits are read/write by software. They define the dead time inserted between PWM0/PWM1. Dead time is calculated as follows:

Dead Time = DT[6:0] x Tcounter1

Note: If DTE is set and DT[6:0]=0, PWM output signals will be at their reset state.

Table 38. Register mapping and reset values

Add. (Hex)	Register label	7	6	5	4	3	2	1	0
0011	**ATCSR** Reset Value	0	ICF 0	ICIE 0	CK1 0	CK0 0	OVF1 0	OVFIE1 0	CMPIE 0
0012	**CNTR1H** Reset Value	0	0	0	0	CNTR1_11 0	CNTR1_10 0	CNTR1_9 0	CNTR1_8 0
0013	**CNTR1L** Reset Value	CNTR1_7 0	CNTR1_8 0	CNTR1_7 0	CNTR1_6 0	CNTR1_3 0	CNTR1_2 0	CNTR1_1 0	CNTR1_0 0
0014	**ATR1H** Reset Value	0	0	0	0	ATR11 0	ATR10 0	ATR9 0	ATR8 0
0015	**ATR1L** Reset Value	ATR7 0	ATR6 0	ATR5 0	ATR4 0	ATR3 0	ATR2 0	ATR1 0	ATR0 0
0016	**PWMCR** Reset Value	0	OE3 0	0	OE2 0	0	OE1 0	0	OE0 0
0017	**PWM0CSR** Reset Value	0	0	0	0	0	0	OP0 0	CMPF0 0
0018	**PWM1CSR** Reset Value	0	0	0	0	0	0	OP1 0	CMPF1 0
0019	**PWM2CSR** Reset Value	0	0	0	0	0	0	OP2 0	CMPF2 0
001A	**PWM3CSR** Reset Value	0	0	0	0	OP_EN 0	OPEDGE 0	OP3 0	CMPF3 0
001B	**DCR0H** Reset Value	0	0	0	0	DCR11 0	DCR10 0	DCR9 0	DCR8 0

Table 38. Register mapping and reset values (continued)

Add. (Hex)	Register label	7	6	5	4	3	2	1	0
001C	**DCR0L** Reset Value	DCR7 0	DCR6 0	DCR5 0	DCR4 0	DCR3 0	DCR2 0	DCR1 0	DCR0 0
001D	**DCR1H** Reset Value	0	0	0	0	DCR11 0	DCR10 0	DCR9 0	DCR8 0
001E	**DCR1L** Reset Value	DCR7 0	DCR6 0	DCR5 0	DCR4 0	DCR3 0	DCR2 0	DCR1 0	DCR0 0
001F	**DCR2H** Reset Value	0	0	0	0	DCR11 0	DCR10 0	DCR9 0	DCR8 0
0020	**DCR2L** Reset Value	DCR7 0	DCR6 0	DCR5 0	DCR4 0	DCR3 0	DCR2 0	DCR1 0	DCR0 0
0021	**DCR3H** Reset Value	0	0	0	0	DCR11 0	DCR10 0	DCR9 0	DCR8 0
0022	**DCR3L** Reset Value	DCR7 0	DCR6 0	DCR5 0	DCR4 0	DCR3 0	DCR2 0	DCR1 0	DCR0 0
0023	**ATICRH** Reset Value	0	0	0	0	ICR11 0	ICR10 0	ICR9 0	ICR8 0
0024	**ATICRL** Reset Value	ICR7 0	ICR6 0	ICR5 0	ICR4 0	ICR3 0	ICR2 0	ICR1 0	ICR0 0
0025	**ATCSR2** Reset Value	FORCE2 0	FORCE1 0	ICS 0	OVFIE2 0	OVF2 0	ENCNTR2 0	TRAN2 1	TRAN1 1
0026	**BREAKCR1** (1) Reset Value	0	BREDGE 0	BA 0	BPEN 0	PWM3 0	PWM2 0	PWM1 0	PWM0 0
0027	**ATR2H** Reset Value	0	0	0	0	ATR11 0	ATR10 0	ATR9 0	ATR8 0
0028	**ATR2L** Reset Value	ATR7 0	ATR6 0	ATR5 0	ATR4 0	ATR3 0	ATR2 0	ATR1 0	ATR0 0
0029	**DTGR** Reset Value	DTE 0	DT6 0	DT5 0	DT4 0	DT3 0	DT2 0	DT1 0	DT0 0
002A	**BREAKEN** Reset Value	0	0	0	0	0	0	BREN2 1	BREN1 1
002B	Reserved area								
002C	**BREAKCR2** (2) Reset Value	BR2SEL 0	BR2EDGE 0	BA2 0	BP2EN 0	0	0	SWBR2 0	SWBR1 0

1.

2.

10.3 Lite timer 2 (LT2)

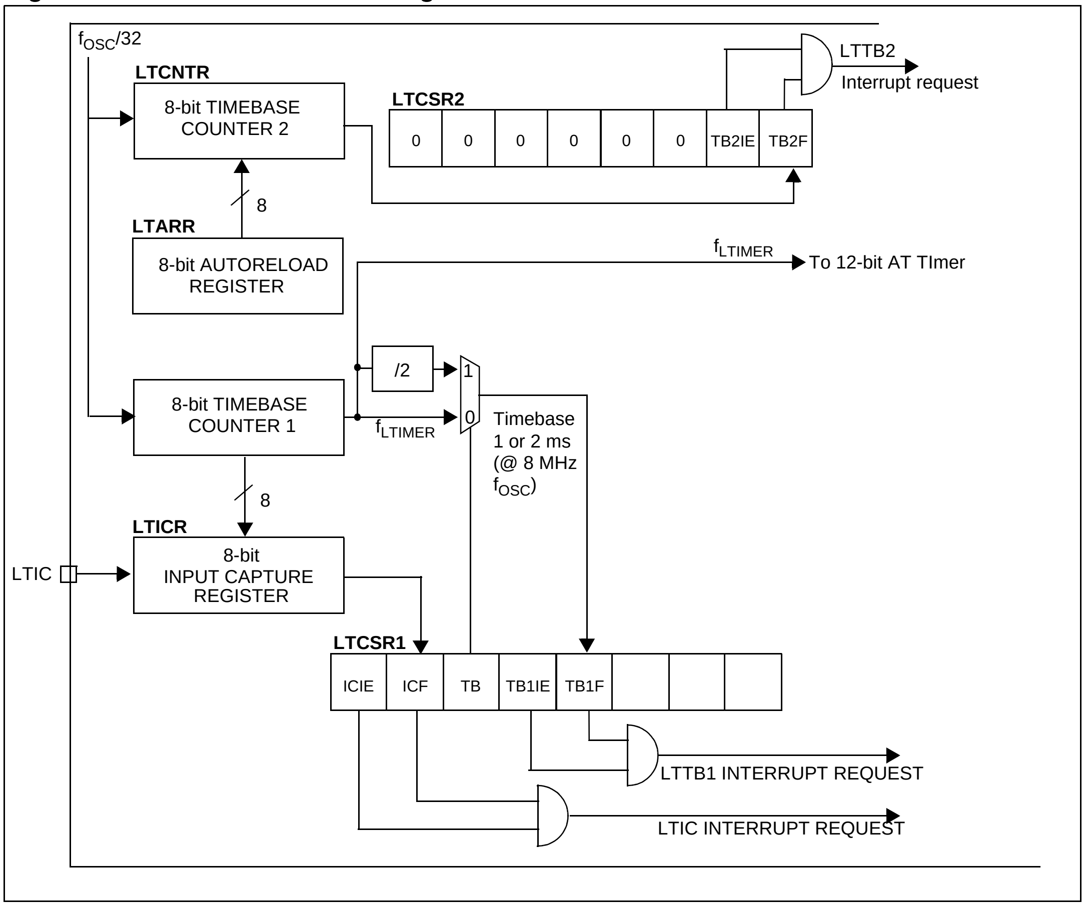
10.3.1 Introduction

The Lite timer can be used for general-purpose timing functions. It is based on two free-running 8-bit upcounters, a watchdog function and an 8-bit Input Capture register.

10.3.2 Main features

- Real-time Clock
  - One 8-bit upcounter 1 ms or 2 ms timebase period (@ 8 MHz f_{OSC})
  - One 8-bit upcounter with autoreload and programmable timebase period from 4µs to 1.024 ms in 4 µs increments (@ 8 MHz f_{OSC})
  - 2 Maskable timebase interrupts
- Input Capture
  - 8-bit Input Capture register (LTICR)
- Maskable interrupt with wakeup from Halt mode capability
- Watchdog
  - Enabled by hardware or software (configurable by option byte)
  - Optional reset on HALT instruction (configurable by option byte)
  - Automatically resets the device unless disable bit is refreshed
  - Software reset (Forced Watchdog reset)
  - Watchdog reset status flag

Figure 52. Lite timer 2 block diagram



10.3.3 Functional description

Timebase Counter 1

The 8-bit value of Counter 1 cannot be read or written by software. After an MCU reset, it starts incrementing from 0 at a frequency of $f_{OSC}/32$. An overflow event occurs when the counter rolls over from F9h to 00h. If f_{OSC} = 8 MHz, then the time period between two counter overflow events is 1 ms. This period can be doubled by setting the TB bit in the LTCSR1 register.

When Counter 1 overflows, the TB1F bit is set by hardware and an interrupt request is generated if the TB1IE bit is set. The TB1F bit is cleared by software reading the LTCSR1 register.

Input Capture

The 8-bit Input Capture register is used to latch the free-running upcounter (Counter 1) 1 after a rising or falling edge is detected on the LTIC pin. When an Input Capture occurs, the ICF bit is set and the LTICR register contains the counter 1 value. An interrupt is generated if the ICIE bit is set. The ICF bit is cleared by reading the LTICR register.

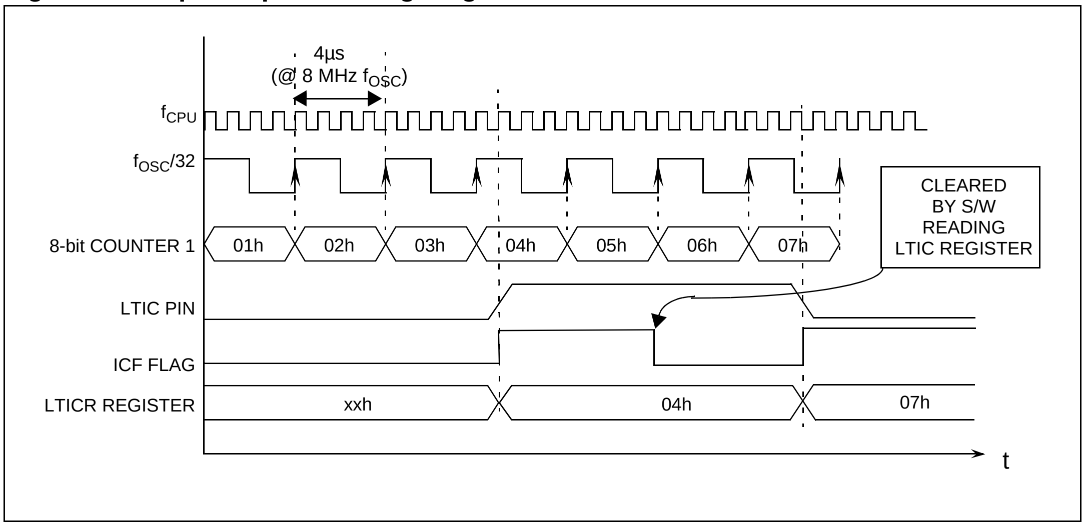
The LTICR is a read-only register and always contains the data from the last Input Capture. Input Capture is inhibited if the ICF bit is set.

Timebase Counter 2

Counter 2 is an 8-bit autoreload upcounter. It can be read by accessing the LTCNTR register. After an MCU reset, it increments at a frequency of $f_{OSC}/32$ starting from the value stored in the LTARR register. A counter overflow event occurs when the counter rolls over from FFh to the LTARR reload value. Software can write a new value at any time in the LTARR register, this value will be automatically loaded in the counter when the next overflow occurs.

When Counter 2 overflows, the TB2F bit in the LTCSR2 register is set by hardware and an interrupt request is generated if the TB2IE bit is set. The TB2F bit is cleared by software reading the LTCSR2 register.

Figure 53. Input Capture timing diagram



Watchdog

When enabled using the WDGE bit, the Watchdog generates a reset after 2 ms (@ = 8 MHz f_{OSC}).

To prevent this watchdog reset occurring, software must set the WDGD bit. The WDGD bit is cleared by hardware after t_{WDG}. This means that software must write to the WDGD bit at regular intervals to prevent a watchdog reset occurring. Refer to *Figure 54*.

Note: Software can use the timebase feature to set the WDGD bit at 1 or 2 ms intervals.

A Watchdog reset can be forced at any time by setting the WDGRF bit.

The WDGRF bit also acts as a flag, indicating that the Watchdog was the source of the reset. It is automatically cleared after it has been read.

Hardware Watchdog Option

If Hardware Watchdog is selected by option byte, the watchdog is always active and the WDGE bit in the LTCSR1 is not used.

Refer to the Option byte description.

Using Halt mode with the Watchdog (option)

If the Watchdog reset on HALT option is not selected by option byte, the Halt mode can be used when the watchdog is enabled.

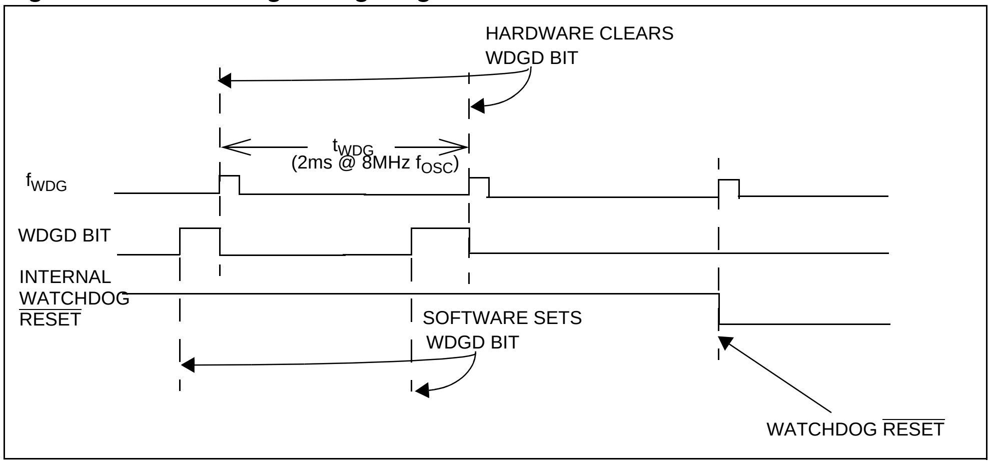
In this case, the HALT instruction stops the oscillator. When the oscillator is stopped, the Lite timer stops counting and is no longer able to generate a Watchdog reset until the microcontroller receives an external interrupt or a reset.

If an external interrupt is received, the WDG restarts counting after 4096 CPU clocks. If a reset is generated, the Watchdog is disabled (reset state).

Recommendations

- Make sure that an external event is available to wake up the microcontroller from Halt mode.
- Before executing the HALT instruction, refresh the WDGD bit, to avoid an unexpected Watchdog reset immediately after waking up the microcontroller.
- 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.

Figure 54. Watchdog timing diagram



10.3.4 Low power modes

Table 39. Effect of low power modes on Lite timer 2

Mode	Description
Slow	No effect on Lite timer (this peripheral is driven directly by $f_{OSC}/32$)
Wait	No effect on Lite timer
Active Halt	No effect on Lite timer
Halt	Lite timer stops counting

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

Bit 5 = **TB** *Timebase period selection bit*

This bit is set and cleared by software.

0: Timebase period = t_{OSC} * 8000 (1 ms @ 8 MHz)

1: Timebase period = t_{OSC} * 16000 (2 ms @ 8 MHz)

Bit 4 = **TB1IE** *Timebase Interrupt enable bit*

This bit is set and cleared by software.

0: Timebase (TB1) interrupt disabled

1: Timebase (TB1) interrupt enabled

Bit 3 = **TB1F** *Timebase 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 overflow

1: A counter overflow has occurred

Bits 2:0 = Reserved, must be kept cleared.

Lite Timer Input Capture register (LTICR)

Reset value: 0000 0000 (00h)

7							0
ICR7	ICR6	ICR5	ICR4	ICR3	ICR2	ICR1	ICR0
Read only							

Bits 7:0 = **ICR[7:0]** *Input Capture value*

These bits are read by software and cleared by hardware after a reset. If the ICF bit in the LTCSR is cleared, the value of the 8-bit up-counter will be captured when a rising or falling edge occurs on the LTIC pin.

Table 41. Lite Timer register mapping and reset values

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0C	**LTCSR2** Reset Value	0	0	0	0	0	0	TB2IE 0	TB2F 0
0D	**LTARR** Reset Value	AR7 0	AR6 0	AR5 0	AR4 0	AR3 0	AR2 0	AR1 0	AR0 0
0E	**LTCNTR** Reset Value	CNT7 0	CNT6 0	CNT5 0	CNT4 0	CNT3 0	CNT2 0	CNT1 0	CNT0 0
0F	**LTCSR1** Reset Value	ICIE 0	ICF x	TB 0	TB1IE 0	TB1F 0	0	0	0
10	**LTICR** Reset Value	ICR7 0	ICR6 0	ICR5 0	ICR4 0	ICR3 0	ICR2 0	ICR1 0	ICR0 0

10.4 16-bit timer

10.4.1 Introduction

The timer consists of a 16-bit free-running counter driven by a programmable prescaler.

It may be used for a variety of purposes, including pulse length measurement of up to two input signals (*input capture*) or generation of up to two output waveforms (*output compare* and *PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the CPU clock prescaler.

10.4.2 Main features

- Programmable prescaler: f_{CPU} divided by 2, 4 or 8.
- Overflow status flag and maskable interrupt
- External clock input (must be at least 4 times slower than the CPU clock speed) with the choice of active edge
- Output compare functions with
  - 2 dedicated 16-bit registers
  - 2 dedicated programmable signals
  - 2 dedicated status flags
  - 1 dedicated maskable interrupt
- Input capture functions with
  - 2 dedicated 16-bit registers
  - 2 dedicated active edge selection signals
  - 2 dedicated status flags
  - 1 dedicated maskable interrupt
- Pulse width modulation mode (PWM)
- One pulse mode
- Reduced power mode
- 5 alternate functions on I/O ports (ICAP1, ICAP2, OCMP1, OCMP2, EXTCLK)

Note: Some timer pins may not be available (not bonded) in some devices. Refer to Section 2: Pin description on page 15.

The block diagram is shown in *Figure 55*.

When reading an input signal on a non-bonded pin, the value is always ‘1’.

10.4.3 Functional description

Counter

The main block of the programmable timer is a 16-bit free running upcounter and its associated 16-bit registers. The 16-bit registers are made up of two 8-bit registers called high and low.

Counter register (CR)

- Counter high register (CHR) is the most significant byte (MSB).
- Counter low register (CLR) is the least significant byte (LSB).

Alternate counter register (ACR)

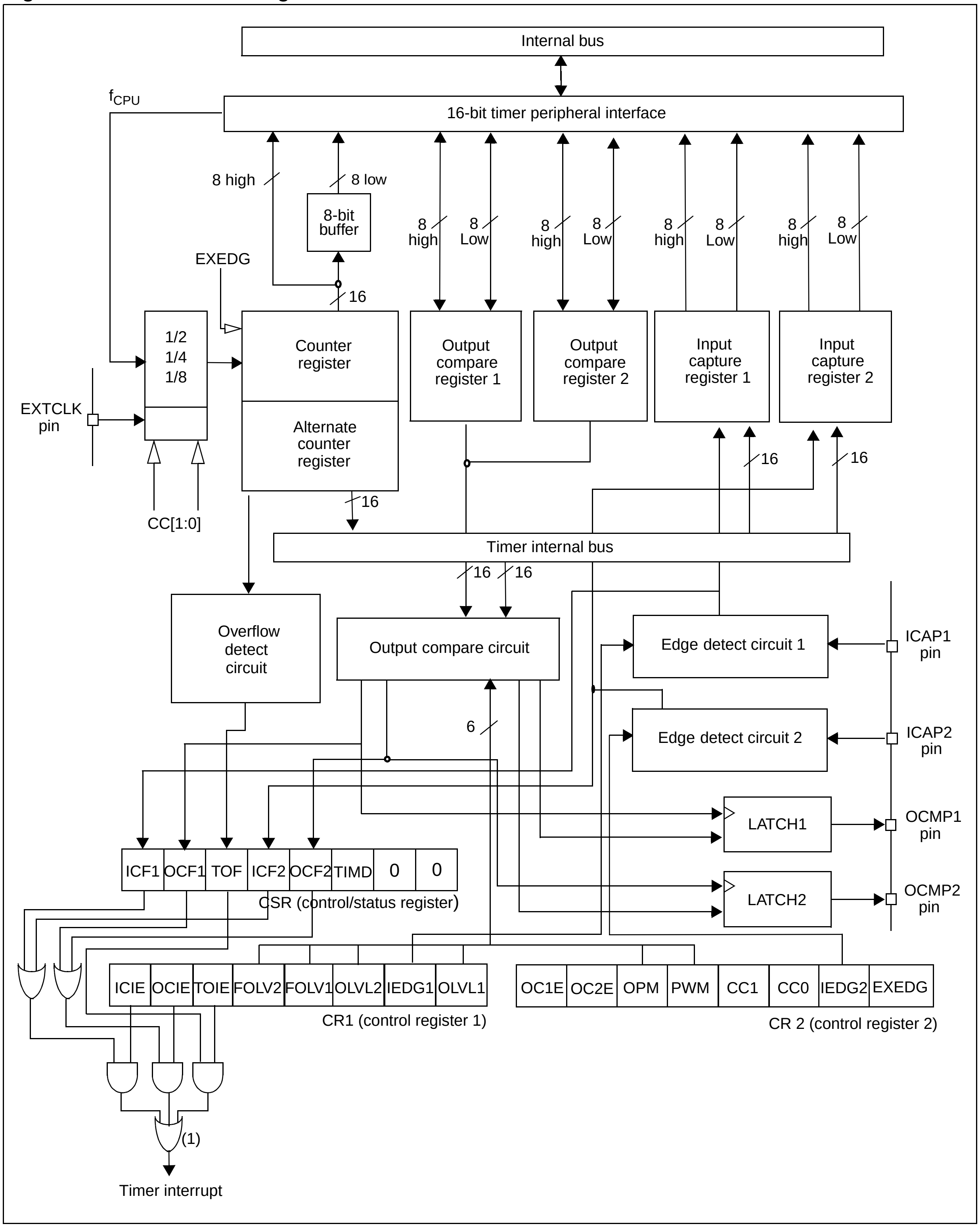
- Alternate counter high register (ACHR) is the MSB.
- Alternate counter low register (ACLR) is the LSB.

These two read-only 16-bit registers contain the same value but with the difference that reading the ACLR register does not clear the TOF bit (timer overflow flag), located in the status register, (SR), (see*16-bit read sequence (from either the counter register or the alternate counter register) on page 120*).

Writing in the CLR register or ACLR register resets the free running counter to the FFFCh value. Both counters have a reset value of FFFCh (this is the only value which is reloaded in the 16-bit timer). The reset value of both counters is also FFFCh in One Pulse mode and PWM mode.

The timer clock depends on the clock control bits of the CR2 register, as illustrated in *Table 37*. The value in the counter register repeats every 131 072, 262 144 or 524 288 CPU clock cycles depending on the CC[1:0] bits. The timer frequency can be $f_{CPU}/2$, $f_{CPU}/4$, $f_{CPU}/8$ or an external frequency.

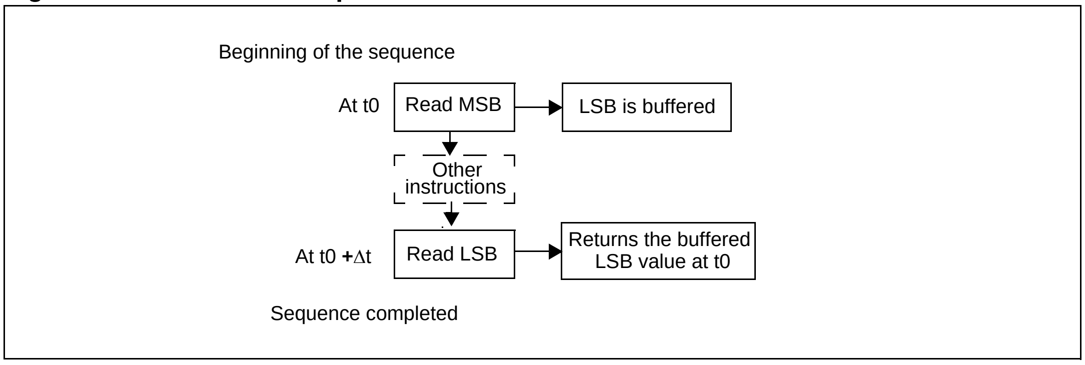
Figure 55. Timer block diagram



1. If IC, OC and TO interrupt requests have separate vectors then the last OR is not present (see *Table 17: ST7FOXF1/ST7FOXK1 Interrupt mapping*)

16-bit read sequence (from either the counter register or the alternate counter register)

Figure 56. 16-bit read sequence



The user must read the MSB first, then the LSB value is buffered automatically.

This buffered value remains unchanged until the 16-bit read sequence is completed, even if the user reads the MSB several times.

After a complete reading sequence, if only the CLR register or ACLR register are read, they return the LSB of the count value at the time of the read.

Whatever the timer mode used (input capture, output compare, one pulse mode or PWM mode) an overflow occurs when the counter rolls over from FFFFh to 0000h then:

- The TOF bit of the SR register is set.
- A timer interrupt is generated if:
  - TOIE bit of the CR1 register is set and
  - 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 (device awakened by an interrupt) or from the reset count (device awakened by a reset).

External clock

The external clock (where available) is selected if CC0 = 1 and CC1 = 1 in 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 triggers 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 57. Counter timing diagram, internal clock divided by 2

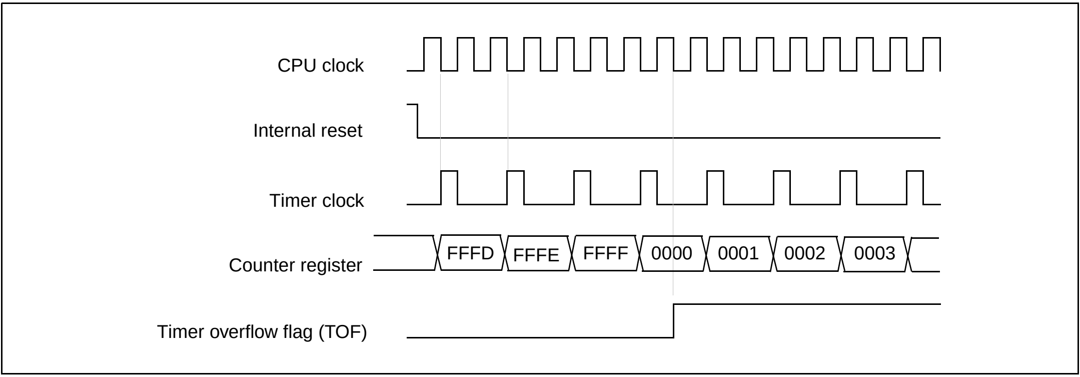


Figure 58. Counter timing diagram, internal clock divided by 4

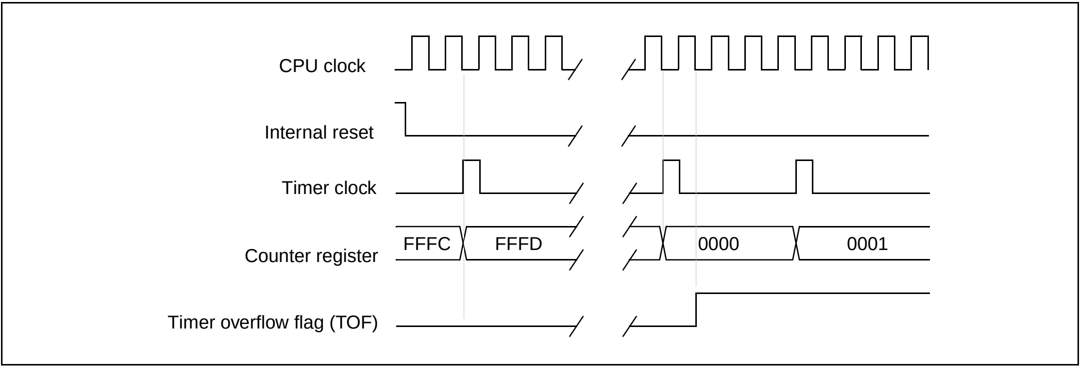
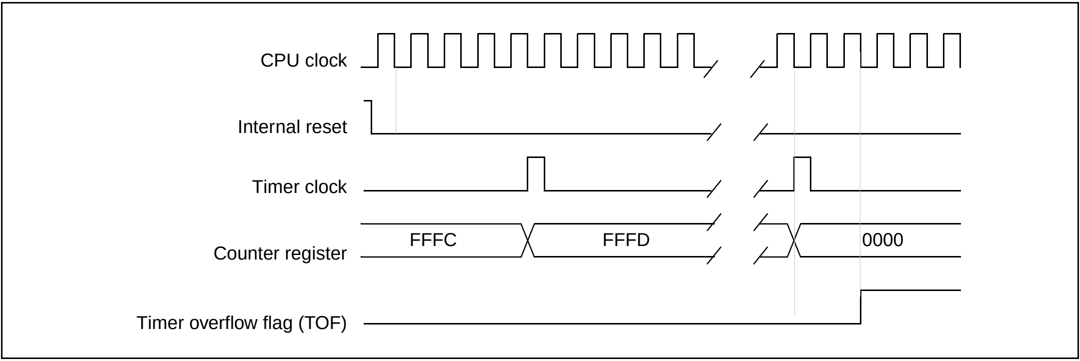


Figure 59. Counter timing diagram, internal clock divided by 8



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

Input capture

In this section, the index, *i*, may be 1 or 2 because there are two input capture functions in the 16-bit timer.

The two input capture 16-bit registers (IC1R and IC2R) are used to latch the value of the free-running counter after a transition detected by the ICAP*i* pin (see below).

	MSB	LSB
ICiR	IC*i*HR	IC*i*LR

IC*i*R register is a read-only register.

The active transition is software programmable through the IEDG*i* bit of control registers (CR*i*).

Timing resolution is one count of the free running counter: (f_{CPU}/CC[1:0]).

Procedure

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

- Select the timer clock (CC[1:0]).
- Select the edge of the active transition on the ICAP2 pin with the IEDG2 bit (the ICAP2 pin must be configured as floating input).

Select the following in the CR1 register:

- Set the ICIE bit to generate an interrupt after an input capture coming from either the ICAP1 pin or the ICAP2 pin
- Select the edge of the active transition on the ICAP1 pin with the IEDG1 bit (the ICAP1pin must be configured as floating input).

When an input capture occurs:

- The ICF*i* bit is set
- The IC*i*R register contains the value of the free running counter on the active transition on the ICAP*i* pin (see *Figure 61*).
- A timer interrupt is generated if the ICIE bit is set and the I bit is cleared in the CC register. Otherwise, the interrupt remains pending until both conditions become true.

Clearing the input capture interrupt request (that is, clearing the ICF*i* bit) is done in two steps:

1. By reading the SR register while the ICF*i* bit is set.
2. By accessing (reading or writing) the IC*i*LR register.

Note: 1 *After reading the ICiHR register, transfer of input capture data is inhibited and ICFi is never set until the ICiLR register is also read.*

2 *The ICiR register contains the free running counter value which corresponds to the most recent input capture.*

3 *The two input capture functions can be used together even if the timer also uses the two output compare functions.*

4 *In One Pulse mode and PWM mode only the input capture 2 can be used.*

5 *The alternate inputs (ICAP1 and ICAP2) are always directly connected to the timer. So any transitions on these pins activate the input capture function. Moreover if one of the ICAPi pin is configured as an input and the second one as an output, an interrupt can be generated if*

the user toggle the output pin and if the ICIE bit is set. This can be avoided if the input capture function i is disabled by reading the ICiHR (see note 1).

6 The TOF bit can be used with interrupt in order to measure event that go beyond the timer range (FFFFh).

Figure 60. Input capture block diagram

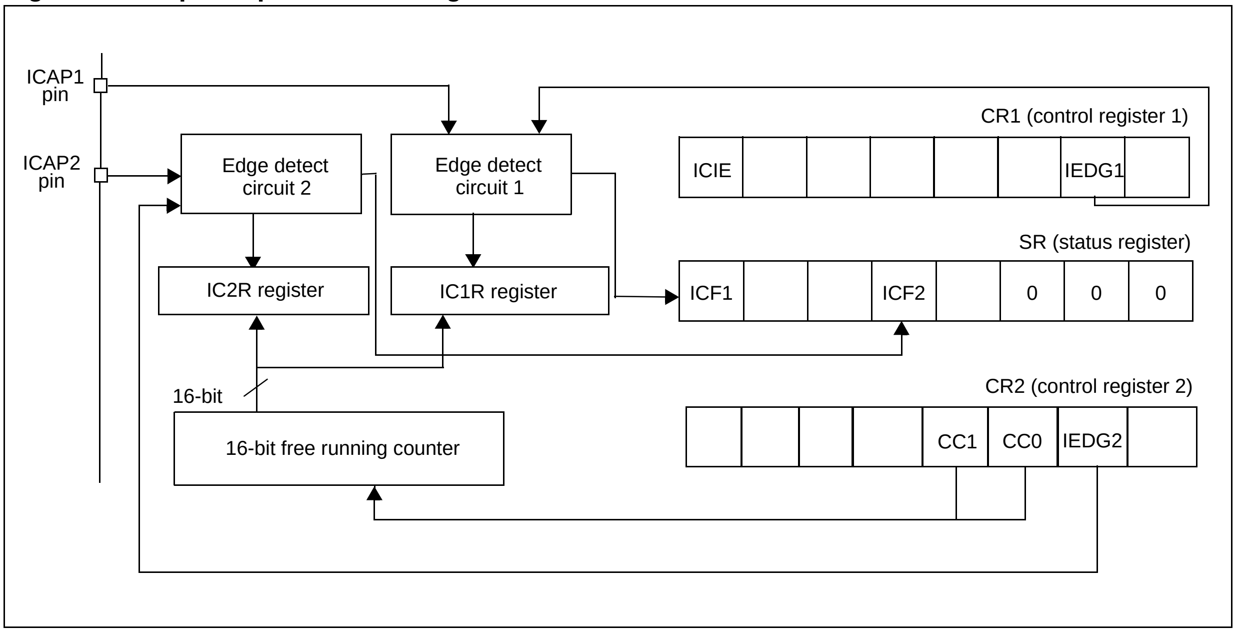
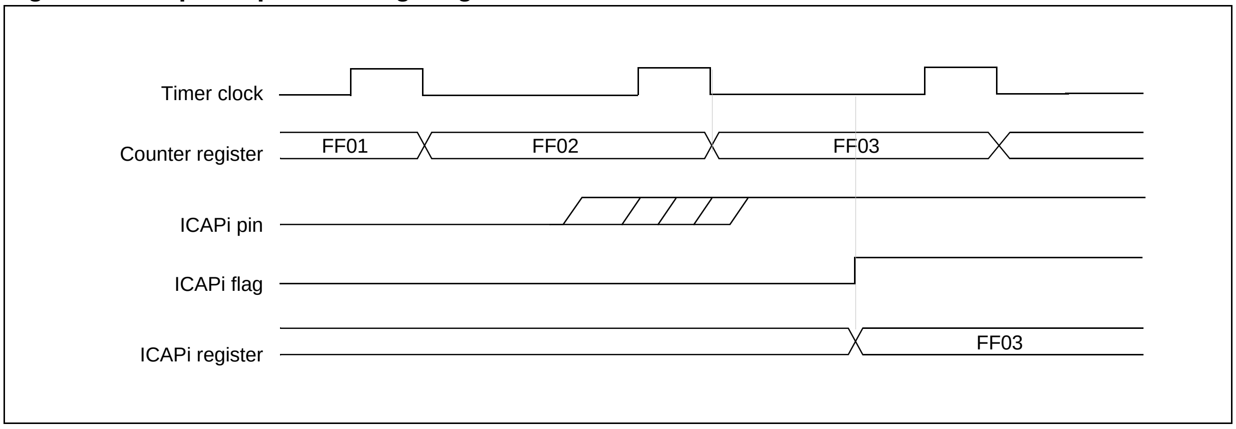


Figure 61. Input capture timing diagram



1. The active edge is the rising edge.
2. The time between an event on the ICAPi pin and the appearance of the corresponding flag is from 2 to 3 CPU clock cycles. This depends on the moment when the ICAP event happens relative to the timer clock.

Output compare

In this section, the index, *i*, may be 1 or 2 because there are two output compare functions in the 16-bit timer.

This function can be used to control an output waveform or indicate when a period of time has elapsed.

When a match is found between the output compare register and the free running counter, the output compare function:

- Assigns pins with a programmable value if the OCIE bit is set
- Sets a flag in the status register
- Generates an interrupt if enabled

Two 16-bit registers output compare register 1 (OC1R) and output compare register 2 (OC2R) contain the value to be compared to the counter register each timer clock cycle.

	MSB	LSB
OC*i*R	OC*i*HR	OC*i*LR

These registers are readable and writable and are not affected by the timer hardware. A reset event changes the OC*i*R value to 8000h.

Timing resolution is one count of the free running counter: ($f_{CPU/CC[1:0]}$).

Procedure:

To use the output compare function, select the following in the CR2 register:

- Set the OC*i*E bit if an output is needed then the OCMP*i* pin is dedicated to the output compare *i* signal.
- Select the timer clock (CC[1:0]) (see *: Timer A Control register 2 (TACR2) on page 134*).

In the CR1 register select the following:

- Select the OLVL*i* bit to be applied to the OCMP*i* pins after the match occurs.
- Set the OCIE bit to generate an interrupt if it is needed.

When a match is found between OCRi register and CR register:

- Set the OCF*i* bit.
- The OCMP*i* pin takes OLVL*i* bit value (OCMP*i* pin latch is forced low during reset).
- A timer interrupt is generated if the OCIE bit is set in the CR2 register and the I bit is cleared in the CC register (CC).

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

Equation 1

$$\Delta \text{ OC}i\text{R} = \frac{\Delta t * f_{CPU}}{\text{PRESC}}$$

Where:

Δt =	output compare 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 2

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

Where:

Δt =	output compare period (in seconds)
f_{EXT} =	external timer clock frequency (in hertz)

Clearing the output compare interrupt request (that is, clearing the OCF*i* bit) is done by:

1. Reading the SR register while the OCF*i* bit is set.
2. Accessing (reading or writing) the OC*i*LR register.

The following procedure is recommended to prevent the OCF*i* bit from being set between the time it is read and the time it is written to the OC*i*R register:

- Write to the OC*i*HR register (further compares are inhibited).
- Read the SR register (first step in the clearance of the OCF*i* bit, which may be already set).
- Write to the OC*i*LR register (enables the output compare function and clears the OCF*i* bit).

Note: 1 *After a processor write cycle to the OCiHR register, the output compare function is inhibited until the OCiLR register is also written.*

2 *If the OCiE bit is not set, the OCMPi pin is a general I/O port and the OLVLi bit does not appear when a match is found but an interrupt could be generated if the OCIE bit is set.*

3 *In both internal and external clock modes, OCFi and OCMPi are set while the counter value equals the OCiR register value (see Figure 63 for an example with $f_{CPU}/2$ and Figure 64 for an example with $f_{CPU}/4$). This behavior is the same in OPM or PWM mode.*

4 *The output compare functions can be used both for generating external events on the OCMPi pins even if the input capture mode is also used.*

5 *The value in the 16-bit OCiR register and the OLVi bit should be changed after each successful comparison in order to control an output waveform or establish a new elapsed timeout.*

Forced compare output capability

When the FOLV*i* bit is set by software, the OLVL*i* bit is copied to the OCMP*i* pin. The OLV*i* bit has to be toggled in order to toggle the OCMP*i* pin when it is enabled (OC*i*E bit = 1). The OCF*i* bit is then not set by hardware, and thus no interrupt request is generated.

FOLVL*i* bits have no effect in both one pulse mode and PWM mode.

Figure 62. Output compare block diagram

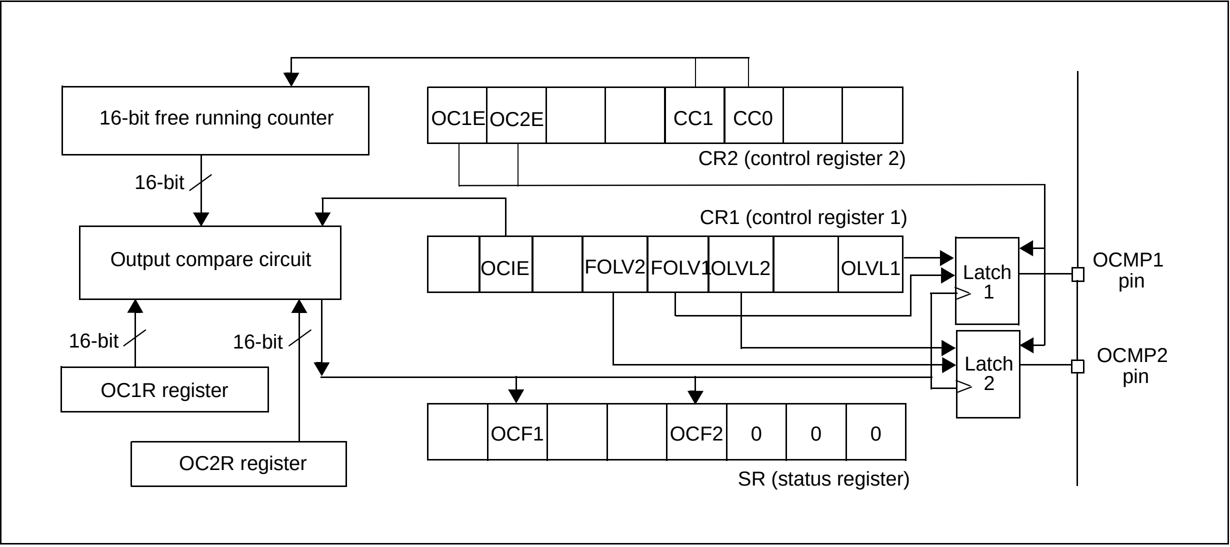


Figure 63. Output compare timing diagram, $f_{TIMER} = f_{CPU}/2$

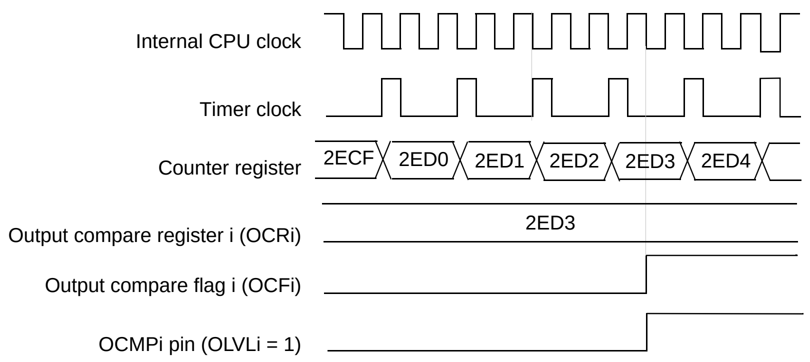
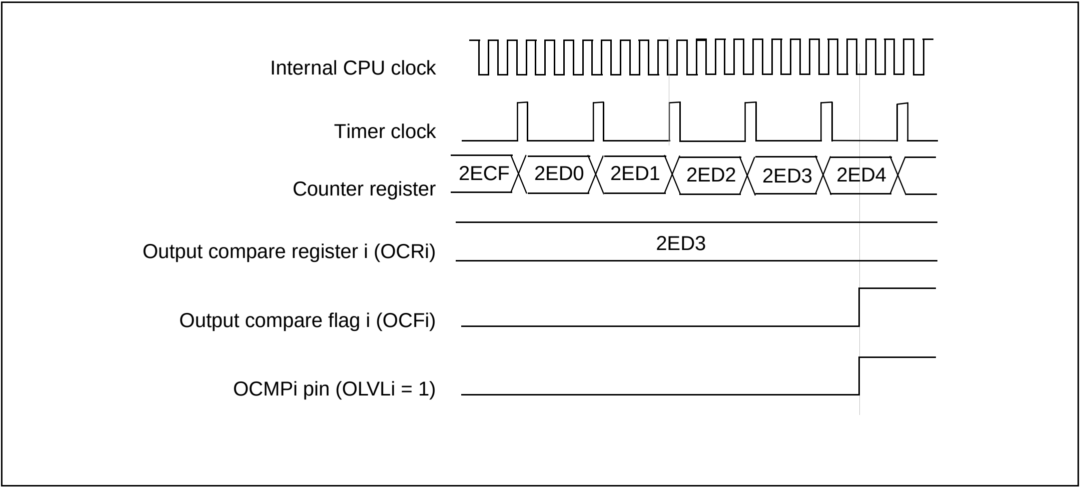


Figure 64. Output compare timing diagram, $f_{TIMER} = f_{CPU}/4$



One pulse mode

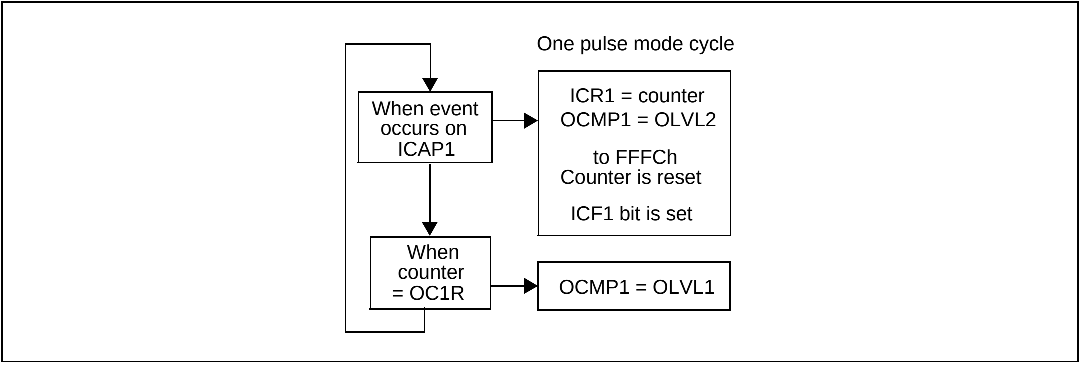
One pulse mode enables the generation of a pulse when an external event occurs. This mode is selected via the OPM bit in the CR2 register.

The one pulse mode uses the input capture1 function and the output compare1 function.

Procedure

1. Load the OC1R register with the value corresponding to the length of the pulse (see *Equation 3* below).
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 *: Timer A Control register 2 (TACR2) on page 134*).

Figure 65. One pulse mode sequence



When a valid event occurs on the ICAP1 pin, the counter value is loaded in the ICR1 register. The counter is then initialized to FFFCh, the OLVL2 bit is output on the OCMP1 pin and the ICF1 bit is set.

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. Accessing (reading or writing) the IC*i*LR register.

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

Equation 3

$$OCiR\ value = \frac{t * f_{CPU}}{PRESC} - 5$$

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 *: Timer A Control register 2 (TACR2) on page 134*)

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

Equation 4

$$OCiR = t * f_{EXT} - 5$$

Where:

t =	pulse period (in seconds)
f_{EXT} =	external timer 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 66*).

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

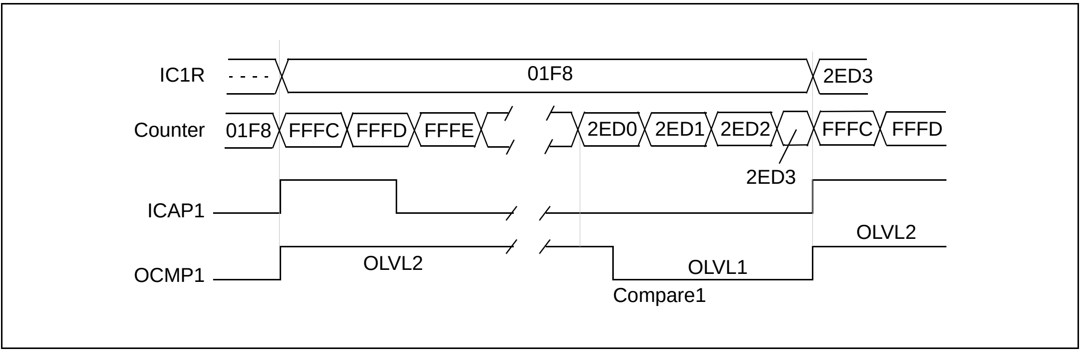
2 *When the pulse width modulation (PWM) and one pulse mode (OPM) bits are both set, the PWM mode is the only active one.*

3 *If OLVL1 = OLVL2 a continuous signal is seen on the OCMP1 pin.*

4 *The ICAP1 pin can not 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.*

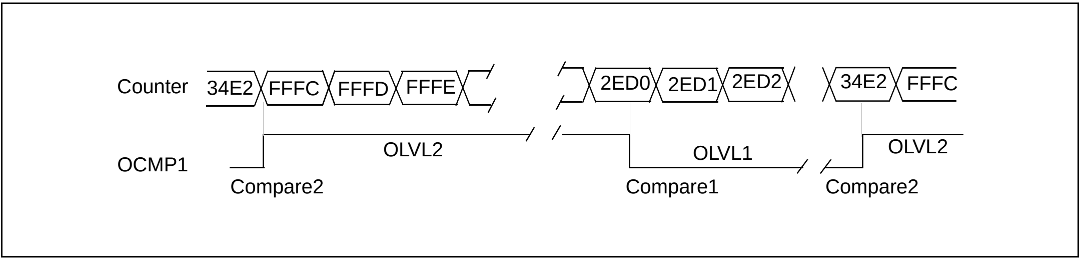
5 *When one pulse mode is used OC1R is dedicated to this mode. Nevertheless OC2R and OCF2 can be used to indicate a period of time has been elapsed but cannot generate an output waveform because the level OLVL2 is dedicated to the one pulse mode.*

Figure 66. One pulse mode timing example



1. IEDG1 = 1, OC1R = 2ED0h, OLVL1 = 0, OLVL2 = 1

Figure 67. Pulse width modulation mode timing example



1. OC1R = 2ED0h, OC2R = 34E2, OLVL1 = 0, OLVL2 = 1

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 can not 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 loaded in their respective shadow registers (double buffer) only at the end of the PWM period (OC2) to avoid spikes on the PWM output pin (OCMP1). The shadow registers contain the reference values for comparison in PWM 'double buffering' mode.

Note: There is a locking mechanism for transferring the OCiR value to the buffer. After a write to the OCiHR register, transfer of the new compare value to the buffer is inhibited until OCiLR is also written.

Unlike in output compare mode, the compare function is always enabled in PWM mode.

Procedure

To use pulse width modulation mode:

1. Load the OC2R register with the value corresponding to the period of the signal using the formula in the opposite column.
2. Load the OC1R register with the value corresponding to the period of the pulse if (OLVL1 = 0 and OLVL2 = 1) using *Equation 5*.
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 OC1R register.

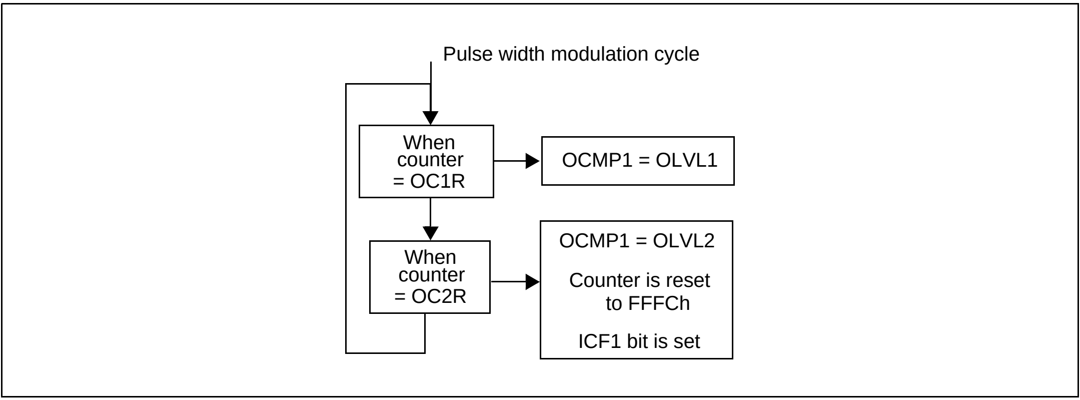
   Using the OLVL2 bit, select the level to be applied to the OCMP1 pin after a successful comparison with OC2R register.
4. Select the following in the CR2 register:

   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*: Timer A Control register 2 (TACR2) on page 134*).

Figure 68. Pulse width modulation cycle



If OLVL = 1 and OLVL2 = 0 the length of the positive pulse is the difference between the OC2R and OC1R registers.

If OLVL1 = OLVL2 a continuous signal is seen on the OCMP1 pin.

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

Equation 5

$$OCiR\ value = \frac{t * f_{CPU}}{PRESC} - 5$$

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

$$OCiR = t * f_{EXT} - 5$$

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.

10.4.5 Interrupts

Table 43. 16-bit timer interrupt control/wakeup capability[(1)]

Interrupt event	Event flag	Enable control bit	Exit from WAIT	Exit from HALT
Input capture 1 event/counter reset in PWM mode	ICF1	ICIE	Yes	No
Input capture 2 event	ICF2		Yes	No
Output compare 1 event (not available in PWM mode)	OCF1	OCIE	Yes	No
Output compare 2 event (not available in PWM mode)	OCF2		Yes	No
Timer overflow event	TOF	TOIE	Yes	No

1. The 16-bit timer interrupt events are connected to the same interrupt vector (see *Section 7: Interrupts*). These events generate an interrupt if the corresponding enable control bit is set and the interrupt mask in the CC register is reset (RIM instruction).

10.4.6 Summary of 16-bit timer modes

Table 44. Summary of 16-bit timer modes

Modes	Available resources			
	Input capture 1	Input capture 2	Output compare 1	Output compare 2
Input capture (1) and/or (2)	Yes	Yes	Yes	Yes
Output compare (1) and/or (2)	Yes	Yes	Yes	Yes
One pulse mode	No	Not recommended[(1)]	No	Partially[(2)]
PWM mode	No	Not recommended[(3)]	No	No

1. See note 4 in *One pulse mode on page 127*.
2. See note 5 in *One pulse mode on page 127*.
3. See note 4 in *Pulse width modulation mode on page 129*.

10.4.7 16-bit timer registers

Each timer is associated with three control and status registers, and with six pairs of data registers (16-bit values) relating to the two input captures, the two output compares, the counter and the alternate counter.

TIMA Control register 1 (TACR1)

Reset value: 0000 0000 (00h)

7							0
ICIE	OCIE	TOIE	FOLV2	FOLV1	OLVL2	IEDG1	OLVL1
Read / Write							

Bit 7 = **ICIE** *Input capture interrupt enable*

0: Interrupt is inhibited

1: A timer interrupt is generated whenever the ICF1 or ICF2 bit of the SR register is set

Bit 6 = **OCIE** *Output compare interrupt enable*

0: Interrupt is inhibited

1: A timer interrupt is generated whenever the OCF1 or OCF2 bit of the SR register is set

Bit 5 = **TOIE** *Timer overflow interrupt enable*

0: Interrupt is inhibited

1: A timer interrupt is enabled whenever the TOF bit of the SR register is set

Bit 4 = **FOLV2** *Forced output compare 2*

This bit is set and cleared by software.

0: No effect on the OCMP2 pin

1: Forces the OLVL2 bit to be copied to the OCMP2 pin, if the OC2E bit is set and even if there is no successful comparison

Bit 3 = **FOLV1** *Forced output compare 1*

This bit is set and cleared by software.

0: No effect on the OCMP1 pin

1: Forces OLVL1 to be copied to the OCMP1 pin, if the OC1E bit is set and even if there is no successful comparison

Bit 2 = **OLVL2** *Output level 2*

This bit is copied to the OCMP2 pin whenever a successful comparison occurs with the OC2R register and OCxE is set in the CR2 register. This value is copied to the OCMP1 pin in one pulse mode and pulse width modulation mode.

Bit 1 = **IEDG1** *Input edge 1*

This bit determines which type of level transition on the ICAP1 pin triggers the capture.

0: A falling edge triggers the capture

1: A rising edge triggers the capture

Bit 0 = **OLVL1** *Output level 1*

The OLVL1 bit is copied to the OCMP1 pin whenever a successful comparison occurs with the OC1R register and the OC1E bit is set in the CR2 register.

Timer A Control register 2 (TACR2)

Reset value: 0000 0000 (00h)

7							0
OC1E	OC2E	OPM	PWM	CC[1:0]		IEDG2	EXEDG
Read / Write							

Bit 7 = **OC1E** *Output compare 1 pin enable*

This bit is used only to output the signal from the timer on the OCMP1 pin (OLV1 in output compare mode, both OLV1 and OLV2 in PWM and one-pulse mode). Whatever the value of the OC1E bit, the output compare 1 function of the timer remains active.

0: OCMP1 pin alternate function disabled (I/O pin free for general-purpose I/O)
1: OCMP1 pin alternate function enabled

Bit 6 = **0C2E** *Output compare 2 pin enable*

This bit is used only to output the signal from the timer on the OCMP2 pin (OLV2 in output compare mode). Whatever the value of the OC2E bit, the output compare 2 function of the timer remains active.

0: OCMP2 pin alternate function disabled (I/O pin free for general-purpose I/O)
1: OCMP2 pin alternate function enabled

Bit 5 = **OPM** *One pulse mode*

0: One pulse mode is not active
1: One pulse mode is active, the ICAP1 pin can be used to trigger one pulse on the OCMP1 pin; the active transition is given by the IEDG1 bit. The length of the generated pulse depends on the contents of the OC1R register.

Bit 4 = **PWM** *Pulse width modulation*

0: PWM mode is not active
1: PWM mode is active, the OCMP1 pin outputs a programmable cyclic signal; the length of the pulse depends on the value of OC1R register; the period depends on the value of OC2R register.

Bits 3:2 **CC[1:0]** *Clock control*

The timer clock mode depends on the following bits:

00: Timer clock = $f_{CPU}/4$
01: Timer clock = $f_{CPU}/2$

10: Timer clock = $f_{CPU}/8$
11: Timer clock = external clock (where available)

Note: *If the external clock pin is not available, programming the external clock configuration stops the counter.*

Bit 1 = **IEDG2** *Input edge 2*

This bit determines which type of level transition on the ICAP2 pin triggers the capture.

0: A falling edge triggers the capture
1: A rising edge triggers the capture

Bit 0 = **EXEDG** *External clock edge*

This bit determines which type of level transition on the external clock pin EXTCLK triggers the counter register.

0: A falling edge triggers the counter register
1: A rising edge triggers the counter register

Timer A Control/status register (TACSR)

Reset value: 0000 0000 (00h)

The 3 least significant bits are not used.

7						0
ICF1	OCF1	TOF	ICF2	OCF2	TIMD	Reserved
Read Only						

Bit 7 = **ICF1** *Input capture flag 1*

0: No input capture (reset value)

1: An input capture has occurred on the ICAP1 pin or the counter has reached the OC2R value in PWM mode. To clear this bit, first read the SR register, then read or write the low byte of the IC1R (IC1LR) register.

Bit 6 = **OCF1** *Output compare flag 1*

0: No match (reset value)

1: The content of the free running counter has matched the content of the OC1R register. To clear this bit, first read the SR register, then read or write the low byte of the OC1R (OC1LR) register.

Bit 5 = **TOF** *Timer overflow flag*

0: No timer overflow (reset value)

1: The free running counter rolled over from FFFFh to 0000h. To clear this bit, first read the SR register, then read or write the low byte of the CR (CLR) register.

Note: *Reading or writing the ACLR register does not clear TOF.*

Bit 4 = **ICF2** *Input capture flag 2*

0: No input capture (reset value)

1: An input capture has occurred on the ICAP2 pin. To clear this bit, first read the SR register, then read or write the low byte of the IC2R (IC2LR) register.

Bit 3 = **OCF2** *Output compare flag 2*

0: No match (reset value)

1: The content of the free running counter has matched the content of the OC2R register. To clear this bit, first read the SR register, then read or write the low byte of the OC2R (OC2LR) register.

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

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

0: Timer enabled

1: Timer prescaler, counter and outputs disabled

Bits 1:0 = Reserved, must be kept cleared

Timer A Input capture 1 high register (TAIC1HR)

Reset value: undefined

This is an 8-bit read-only register that contains the high part of the counter value (transferred by the input capture 1 event).

7							0
MSB							LSB
Read Only							

Timer A Input capture 1 low register (TAIC1LR)

Reset value: undefined

This is an 8-bit read-only register that contains the low part of the counter value (transferred by the input capture 1 event).

7							0
MSB							LSB
Read Only							

Timer A Output compare 1 high register (TAOC1HR)

Reset value: 1000 0000 (80h)

This is an 8-bit register that contains the high part of the value to be compared to the CHR register.

7							0
MSB							LSB
Read / Write							

Timer A Output compare 1 low register (TAOC1LR)

Reset value: 0000 0000 (00h)

This is an 8-bit register that contains the low part of the value to be compared to the CLR register.

7							0
MSB							LSB
Read / Write							

Output compare 2 high register (OC2HR)

Reset value: 1000 0000 (80h)

This is an 8-bit register that contains the high part of the value to be compared to the CHR register.

7							0
MSB							LSB
Read / Write							

Output compare 2 low register (OC2LR)

Reset value: 0000 0000 (00h)

This is an 8-bit register that contains the low part of the value to be compared to the CLR register.

7							0
MSB							LSB
Read / Write							

Counter high register (CHR)

Reset value: 1111 1111 (FFh)

This is an 8-bit read-only register that contains the high part of the counter value.

7							0
MSB							LSB
Read Only							

Counter low register (CLR)

Reset value: 1111 1100 (FCh)

This is an 8-bit read-only register that contains the low part of the counter value. A write to this register resets the counter. An access to this register after accessing the CSR register clears the TOF bit.

7							0
MSB							LSB
Read Only							

Alternate counter high register (ACHR)

Reset value: 1111 1111 (FFh)

This is an 8-bit read-only register that contains the high part of the counter value.

7							0
MSB							LSB
Read Only							

Alternate counter low register (ACLR)

Reset value: 1111 1100 (FCh)

This is an 8-bit read-only register that contains the low part of the counter value. A write to this register resets the counter. An access to this register after an access to CSR register does not clear the TOF bit in the CSR register.

7							0
MSB							LSB
Read Only							

Input capture 2 high register (IC2HR)

Reset value: undefined

This is an 8-bit read-only register that contains the high part of the counter value (transferred by the input capture 2 event).

7							0
MSB							LSB
Read Only							

Input capture 2 low register (IC2LR)

Reset value: undefined

This is an 8-bit read-only register that contains the low part of the counter value (transferred by the input capture 2 event).

7							0
MSB							LSB
Read Only							

10.4.8 16-bit timer register map and reset values

Table 45. 16-bit timer register map and reset values

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
55	TACR2 Reset value	OC1E 0	OC2E 0	OPM 0	PWM 0	CC1 0	CC0 0	IEDG2 0	EXEDG 0
56	TACR1 Reset value	ICIE 0	OCIE 0	TOIE 0	FOLV2 0	FOLV1 0	OLVL2 0	IEDG1 0	OLVL1 0
57	TACSR Reset value	ICF1 0	OCF1 0	TOF 0	ICF2 0	OCF2 0	TIMD 0	- 0	- 0
58	TAICHR1 Reset value	MSB -	-	-	-	-	-	-	LSB -
59	TAICLR1 Reset value	MSB -	-	-	-	-	-	-	LSB -
5A	TAOCHR1 Reset value	MSB -	-	-	-	-	-	-	LSB -
5B	TAOCLR1 Reset value	MSB -	-	-	-	-	-	-	LSB -
5C	TACHR Reset value	MSB 1	1	1	1	1	1	1	LSB 1
5D	TACLR Reset value	MSB 1	1	1	1	1	1	0	LSB 0

Table 45. 16-bit timer register map and reset values (continued)

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
5E	TAACHR Reset value	MSB 1	1	1	1	1	1	1	LSB 1
5F	TAACLR Reset value	MSB 1	1	1	1	1	1	0	LSB 0
60	TAICHR2 Reset value	MSB -	-	-	-	-	-	-	LSB -
61	TAICLR2 Reset value	MSB -	-	-	-	-	-	-	LSB -
62	TAOCHR2 Reset value	MSB -	-	-	-	-	-	-	LSB -
63	TAOCLR2 Reset value	MSB -	-	-	-	-	-	-	LSB -

10.5 I^{2}C bus interface (I^{2}C)

10.5.1 Introduction

The I^{2}C Bus Interface serves as an interface between the microcontroller and the serial I^{2}C bus. It provides both multimaster and slave functions, and controls all I^{2}C bus-specific sequencing, protocol, arbitration and timing. It supports fast I^{2}C mode (400 kHz).

10.5.2 Main features

- Parallel-bus/I^{2}C protocol converter
- Multi-master capability
- 7-bit/10-bit Addressing
- Transmitter/Receiver flag
- End-of-byte transmission flag
- Transfer problem detection

I^{2}C master features:

- Clock generation
- I^{2}C bus busy flag
- Arbitration Lost Flag
- End of byte transmission flag
- Transmitter/Receiver Flag
- Start bit detection flag
- Start and Stop generation

I^{2}C slave features:

- Stop bit detection
- I^{2}C bus busy flag
- Detection of misplaced start or stop condition
- Programmable I^{2}C Address detection
- Transfer problem detection
- End-of-byte transmission flag
- Transmitter/Receiver flag

10.5.3 General description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa, using either an interrupt or polled handshake. The interrupts are enabled or disabled by software. The interface is connected to the I^{2}C bus by a data pin (SDAI) and by a clock pin (SCLI). It can be connected both with a standard I^{2}C bus and a Fast I^{2}C bus. This selection is made by software.

Mode selection

The interface can operate in the four following modes:

- Slave transmitter/receiver
- Master transmitter/receiver

By default, it operates in slave mode.

The interface automatically switches from slave to master after it generates a START condition and from master to slave in case of arbitration loss or a STOP generation, allowing then Multi-Master capability.

Communication flow

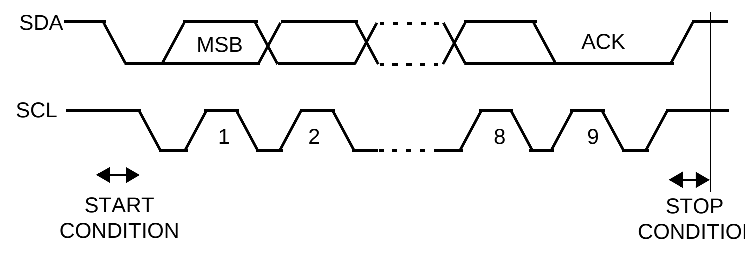
In Master mode, it initiates a data transfer and generates the clock signal. A serial data transfer always begins with a start condition and ends with a stop condition. Both start and stop conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own address (7 or 10-bit), and the General Call address. The General Call address detection may be enabled or disabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the start condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to *Figure 69*.

Figure 69. I^2C bus protocol



Acknowledge may be enabled and disabled by software.

The I^2C interface address and/or general call address can be selected by software.

The speed of the I^2C interface may be selected between Standard (up to 100 kHz) and Fast I^2C (up to 400 kHz).

SDA/SCL line control

Transmitter mode: the interface holds the clock line low before transmission to wait for the microcontroller to write the byte in the Data register.

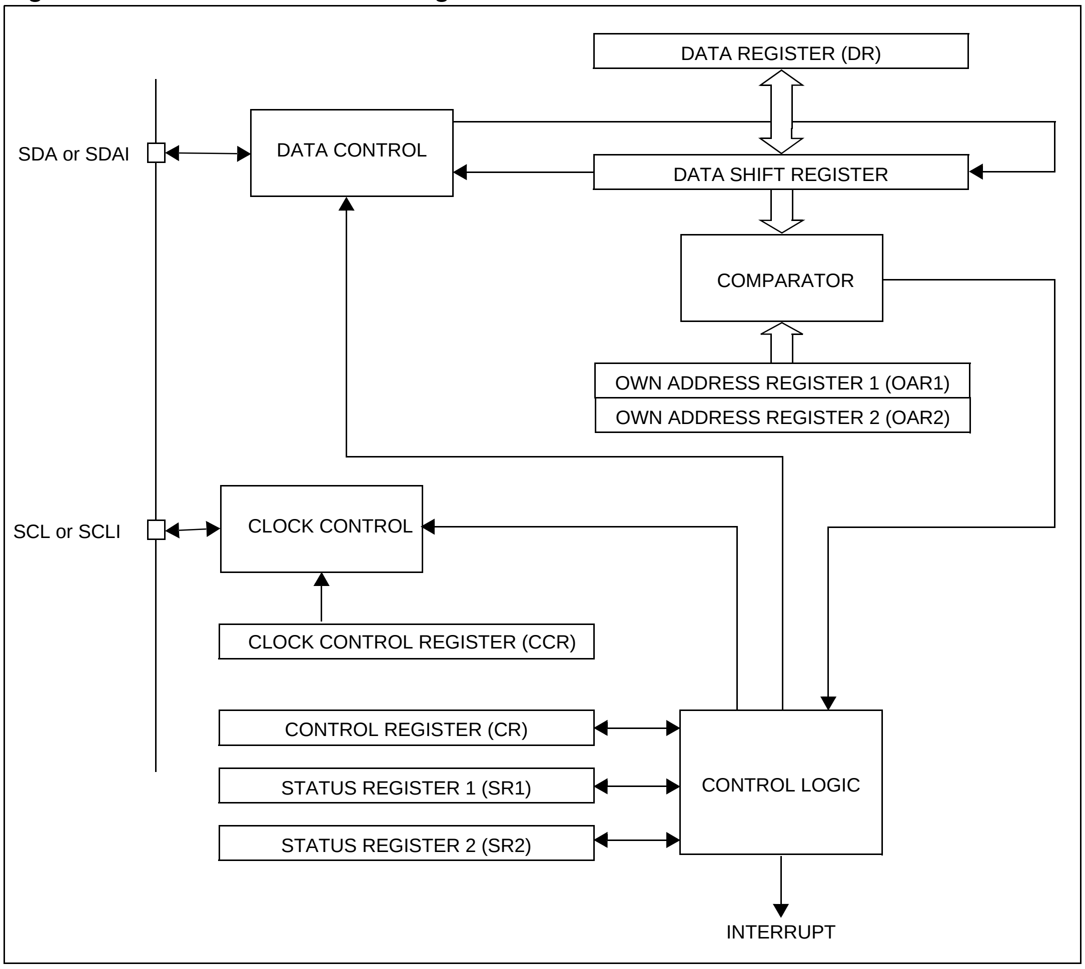
Receiver mode: the interface holds the clock line low after reception to wait for the microcontroller to read the byte in the Data register.

The SCL frequency (F_{scl}) is controlled by a programmable clock divider which depends on the I^2C bus mode.

When the I^2C cell is enabled, the SDA and SCL ports must be configured as floating inputs. In this case, the value of the external pull-up resistor used depends on the application.

When the I^2C cell is disabled, the SDA and SCL ports revert to being standard I/O port pins.

Figure 70. I^2C interface block diagram



10.5.4 Functional description

Refer to the CR, SR1 and SR2 registers in *Section 10.5.7*. for the bit definitions.

By default the I^2C interface operates in Slave mode (M/SL bit is cleared) except when it initiates a transmit or receive sequence.

First the interface frequency must be configured using the FRi bits in the OAR2 register.

Slave mode

As soon as a start condition is detected, the address is received from the SDA line and sent to the shift register; then it is compared with the address of the interface or the General Call address (if selected by software).

Note: In 10-bit addressing mode, the comparison includes the header sequence (11110xx0) and the two most significant bits of the address.

- **Header matched** (10-bit mode only): the interface generates an acknowledge pulse if the ACK bit is set.
- **Address not matched**: the interface ignores it and waits for another Start condition.
- **Address matched**: the interface generates in sequence:
  - Acknowledge pulse if the ACK bit is set.
  - EVF and ADSL bits are set with an interrupt if the ITE bit is set.

Then the interface waits for a read of the SR1 register, **holding the SCL line low** (see *Figure 71* Transfer sequencing EV1).
Next, in 7-bit mode read the DR register to determine from the least significant bit (Data Direction Bit) if the slave must enter Receiver or Transmitter mode.

In 10-bit mode, after receiving the address sequence the slave is always in receive mode. It will enter transmit mode on receiving a repeated Start condition followed by the header sequence with matching address bits and the least significant bit set (11110xx1).

Slave receiver

Following the address reception and after SR1 register has been read, the **slave 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 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 71* Transfer sequencing EV2).

Slave transmitter

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

The slave waits for a read of the SR1 register followed by a write in the DR register, **holding the SCL line low** (see *Figure 71* Transfer sequencing EV3).

When the acknowledge pulse is received the EVF and BTF bits are set by hardware with an interrupt if the ITE bit is set.

Closing slave communication

After the last data byte is transferred a Stop Condition is generated by the master. The interface detects this condition and sets:

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

Then the interface waits for a read of the SR2 register (see *Figure 71* Transfer sequencing EV4).

Error cases

- **BERR**: Detection of a Stop or a Start condition during a byte transfer. In this case, the EVF and the BERR bits are set with an interrupt if the ITE bit is set.
  If it is a Stop then the interface discards the data, released the lines and waits for another Start condition.
  If it is a Start then the interface discards the data and waits for the next slave address on the bus.
- **AF**: Detection of a non-acknowledge bit. In this case, the EVF and AF bits are set with an interrupt if the ITE bit is set.
  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.

Note: *In both cases, SCL line is not held low; however, the SDA line can remain low if the last bits transmitted are all 0. It is then necessary to release both lines by software. The SCL line is not held low while AF=1 but by other flags (SB or BTF) that are set at the same time.*

How to release the SDA / SCL lines

Set and subsequently clear the STOP bit while BTF is set. The SDA/SCL lines are released after the transfer of the current byte.

SMBus compatibility

ST7 I^{2}C is compatible with SMBus V1.1 protocol. It supports all SMBus addressing modes, SMBus bus protocols and CRC-8 packet error checking. Refer to AN1713: SMBus Slave Driver For ST7 I^{2}C Peripheral.

Master mode

To switch from default Slave mode to Master mode a Start condition generation is needed.

Start condition

Setting the START bit while the BUSY bit is cleared causes the interface to switch to Master mode (M/SL bit set) and generates a Start condition.

Once the Start condition is sent, the EVF and SB bits are set by hardware with an interrupt if the ITE bit is set.

The master then waits for a read of the SR1 register followed by a write in the DR register with the Slave address, **holding the SCL line low** (see *Figure 71* Transfer sequencing EV5).

Slave address transmission

1. The slave address is then 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.
2. The master then waits for a read of the SR1 register followed by a write in the DR register, **holding the SCL line low** (see *Figure 71* Transfer sequencing EV9).
3. Then the second address byte is sent by the interface.
4. 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.
5. 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 71* Transfer sequencing EV6).
6. 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 71* 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 71* 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.

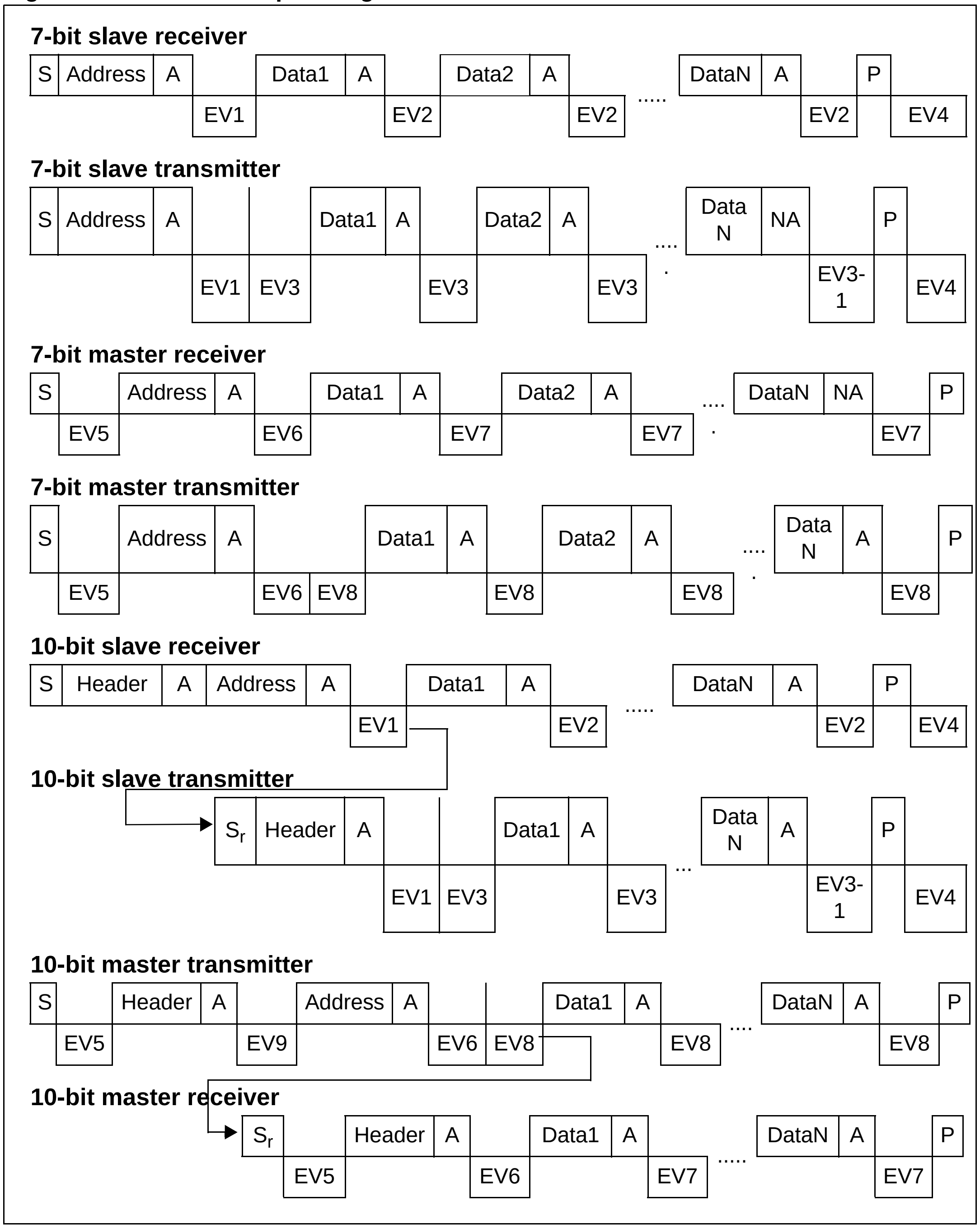
To close the communication: after writing the last byte to the DR register, set the STOP bit to generate the Stop condition. The interface goes automatically back to slave mode (M/SL bit cleared).

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 pulse of each 9-bit transaction:
  Single Master mode
  If a Start or Stop is issued during the first 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 gives the possibility to reinitiate 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 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 if the last bits transmitted are all 0. It is then necessary to release both lines by software. The SCL line is not held low while AF=1 but by other flags (SB or BTF) that are set at the same time.*

Figure 71. Transfer sequencing


7-bit slave receiver
S Address A EV1 Data1 A EV2 Data2 A EV2 ..... DataN A EV2 P EV4
7-bit slave transmitter
S Address A EV1 EV3 Data1 A EV3 Data2 A EV3 ..... Data N NA EV3-1 P EV4
7-bit master receiver
S EV5 Address A EV6 Data1 A EV7 Data2 A EV7 ..... DataN NA EV7 P
7-bit master transmitter
S EV5 Address A EV6 EV8 Data1 A EV8 Data2 A EV8 ..... Data N A EV8 P
10-bit slave receiver
S Header A Address A EV1 Data1 A EV2 ..... DataN A EV2 P EV4
10-bit slave transmitter
Sr Header A EV1 EV3 Data1 A EV3 ... Data N A EV3-1 P EV4
10-bit master transmitter
S EV5 Header A EV9 Address A EV6 EV8 Data1 A EV8 .... DataN A EV8 P
10-bit master receiver
Sr EV5 Header A EV6 Data1 A EV7 ..... DataN A EV7 P

1. S=Start, S_r = Repeated Start, P=Stop, A=Acknowledge, NA=Non-acknowledge, EVx=Event (with interrupt if ITE=1).
2. **EV1:** EVF=1, ADSL=1, cleared by reading SR1 register.
3. **EV2:** EVF=1, BTF=1, cleared by reading SR1 register followed by reading DR register.
4. **EV3:** EVF=1, BTF=1, cleared by reading SR1 register followed by writing DR register.
5. **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). If lines are released by STOP=1, STOP=0, the

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 I^{2}C interface

Mode	Description
Wait	No effect on I^{2}C interface. I^{2}C interrupts cause the device to exit from Wait mode.
Halt	I^{2}C registers are frozen. In Halt mode, the I^{2}C interface is inactive and does not acknowledge data on the bus. The I^{2}C 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

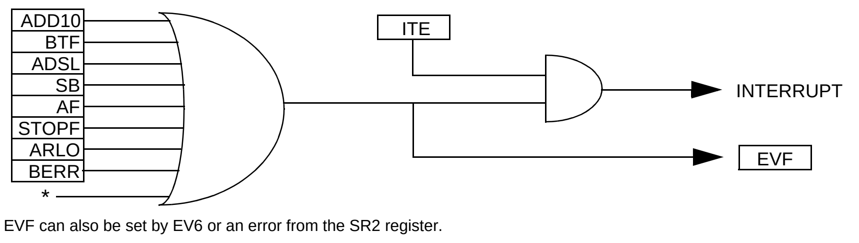

ADD10
BTF
ADSL
SB
AF
STOPF
ARLO
BERR
*
ITE
INTERRUPT
EVF
* EVF can also be set by EV6 or an error from the SR2 register.

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 I^{2}C 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).

10.5.7 Register description

I^2C Control register (I2CCR)

Reset value: 0000 0000 (00h)

7							0
0	0	PE	ENGC	START	ACK	STOP	ITE
Read / Write							

Bits 7:6 = Reserved. Forced to 0 by hardware.

Bit 5 = **PE** *Peripheral Enable bit*

This bit is set and cleared by software.

0: Peripheral disabled

1: Master/Slave capability

Note: *When PE=0, all the bits of the CR register and the SR register except the Stop bit are reset. All outputs are released while PE=0*

When PE=1, the corresponding I/O pins are selected by hardware as alternate functions.

To enable the I^2C interface, write the CR register ***TWICE*** *with PE=1 as the first write only activates the interface (only PE is set).*

Bit 4 = **ENGC** *Enable General Call bit*

This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE=0). The 00h General Call address is acknowledged (01h ignored).

0: General Call disabled

1: General Call enabled

Note: *In accordance with the I^2C standard, when GCAL addressing is enabled, an I^2C slave can only receive data. It will not transmit data to the master.*

Bit 3 = **START** *Generation of a Start condition bit*. This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE=0) or when the Start condition is sent (with interrupt generation if ITE=1).

- In master mode:

  0: No start generation

  1: Repeated start generation
- In slave mode:

  0: No start generation

  1: Start generation when the bus is free

Bit 2 = **ACK** *Acknowledge enable bit*

This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE=0).

0: No acknowledge returned

1: Acknowledge returned after an address byte or a data byte is received

Bit 1 = **STOP** *Generation of a Stop condition bit*

This bit is set and cleared by software. It is also cleared by hardware in master mode. Note: This bit is not cleared when the interface is disabled (PE=0).

- In master mode:

  0: No stop generation

  1: Stop generation after the current byte transfer or after the current Start condition is sent. The STOP bit is cleared by hardware when the Stop condition is sent.
- In slave mode:

  0: No stop generation

  1: Release the SCL and SDA lines after the current byte transfer (BTF=1). In this mode the STOP bit has to be cleared by software.

Bit 0 = **ITE** *Interrupt Enable bit*

This bit is set and cleared by software and cleared by hardware when the interface is disabled (PE=0).

0: Interrupts disabled

1: Interrupts enabled

Refer to *Figure 72* for the relationship between the events and the interrupt.

SCL is held low when the ADD10, SB, BTF or ADSL flags or an EV6 event (See *Figure 71*) is detected.

I²C Status register 1 (I2CSR1)

Reset value: 0000 0000 (00h)

7							0
EVF	ADD10	TRA	BUSY	BTF	ADSL	M/SL	SB
Read Only							

Bit 7 = **EVF** *Event flag*

This bit is set by hardware as soon as an event occurs. It is cleared by software reading SR2 register in case of error event or as described in *Figure 71*. It is also cleared by hardware when the interface is disabled (PE=0).

0: No event

1: One of the following events has occurred:

- BTF=1 (byte received or transmitted)
- ADSL=1 (Address matched in Slave mode while ACK=1)
- SB=1 (Start condition generated in Master mode)
- AF=1 (No acknowledge received after byte transmission)
- STOPF=1 (Stop condition detected in Slave mode)
- ARLO=1 (Arbitration lost in Master mode)
- BERR=1 (Bus error, misplaced Start or Stop condition detected)
- ADD10=1 (Master has sent header byte)
- Address byte successfully transmitted in Master mode.

Bit 6 = **ADD10** *10-bit addressing in Master mode*

This bit is set by hardware when the master has sent the first byte in 10-bit address mode. It is cleared by software reading SR2 register followed by a write in the DR register of the second address byte. It is also cleared by hardware when the peripheral is disabled (PE=0).

0: No ADD10 event occurred.

1: Master has sent first address byte (header)

Bit 5 = **TRA** *Transmitter/Receiver bit*

When BTF is set, TRA=1 if a data byte has been transmitted. It is cleared automatically when BTF is cleared. It is also cleared by hardware after detection of Stop condition (STOPF=1), loss of bus arbitration (ARLO=1) or when the interface is disabled (PE=0).

0: Data byte received (if BTF=1)

1: Data byte transmitted

Bit 4 = **BUSY** *Bus busy* bit

This bit is set by hardware on detection of a Start condition and cleared by hardware on detection of a Stop condition. It indicates a communication in progress on the bus. The BUSY flag of the I2CSR1 register is cleared if a Bus Error occurs.

0: No communication on the bus

1: Communication ongoing on the bus

Bit 3 = **BTF** *Byte Transfer Finished bit*

This bit is set by hardware as soon as a byte is correctly received or transmitted with interrupt generation if ITE=1. It is cleared by software reading SR1 register followed by a read or write of DR register. It is also cleared by hardware when the interface is disabled (PE=0).

– Following a byte transmission, this bit is set after reception of the acknowledge clock pulse. In case an address byte is sent, this bit is set only after the EV6 event (See *Figure 71*). BTF is cleared by reading SR1 register followed by writing the next byte in DR register.

– Following a byte reception, this bit is set after transmission of the acknowledge clock pulse if ACK=1. BTF is cleared by reading SR1 register followed by reading the byte from DR register.

The SCL line is held low while BTF=1.

0: byte transfer not done

1: byte transfer succeeded

Bit 2 = **ADSL** *Address matched bit (slave mode).* This bit is set by hardware as soon as the received slave address matched with the OAR register content or a general call is recognized. An interrupt is generated if ITE=1. It is cleared by software reading SR1 register or by hardware when the interface is disabled (PE=0).

The SCL line is held low while ADSL=1.

0: Address mismatched or not received

1: Received address matched

Bit 1 = **M/SL** *Master/Slave bit*

This bit is set by hardware as soon as the interface is in Master mode (writing START=1). It is cleared by hardware after detecting a Stop condition on the bus or a loss of arbitration (ARLO=1). It is also cleared when the interface is disabled (PE=0).

0: Slave mode

1: Master mode

Bit 0 = **SB** *Start bit (master mode).*

This bit is set by hardware as soon as the Start condition is generated (following a write START=1). An interrupt is generated if ITE=1. It is cleared by software reading SR1 register followed by writing the address byte in DR register. It is also cleared by hardware when the interface is disabled (PE=0).

0: No Start condition

1: Start condition generated

I^{2}C Status register 2 (I2CSR2)

Reset value: 0000 0000 (00h)

7							0
0	0	0	AF	STOPF	ARLO	BERR	GCAL
Read Only							

Bits 7:5 = Reserved. Forced to 0 by hardware.

Bit 4 = **AF** *Acknowledge failure* bit

This bit is set by hardware when no acknowledge is returned. An interrupt is generated if ITE=1. It is cleared by software reading SR2 register or by hardware when the interface is disabled (PE=0).

The SCL line is not held low while AF=1 but by other flags (SB or BTF) that are set at the same time.

0: No acknowledge failure

1: Acknowledge failure

Bit 3 = **STOPF** *Stop detection bit (slave mode)*

This bit is set by hardware when a Stop condition is detected on the bus after an acknowledge (if ACK=1). An interrupt is generated if ITE=1. It is cleared by software reading SR2 register or by hardware when the interface is disabled (PE=0).

The SCL line is not held low while STOPF=1.

0: No Stop condition detected

1: Stop condition detected

Bit 2 = **ARLO** *Arbitration lost bit*

This bit is set by hardware when the interface loses the arbitration of the bus to another master. An interrupt is generated if ITE=1. It is cleared by software reading SR2 register or by hardware when the interface is disabled (PE=0).

After an ARLO event the interface switches back automatically to Slave mode (M/SL=0).

The SCL line is not held low while ARLO=1.

0: No arbitration lost detected

1: Arbitration lost detected

Note: In a Multimaster environment, when the interface is configured in Master Receive mode it does not perform arbitration during the reception of the Acknowledge Bit. Mishandling of the ARLO bit from the I2CSR2 register may occur when a second master simultaneously requests the same data from the same slave and the I^{2}C master does not acknowledge the data. The ARLO bit is then left at 0 instead of being set.

Bit 1 = **BERR** *Bus error bit*

This bit is set by hardware when the interface detects a misplaced Start or Stop condition. An interrupt is generated if ITE=1. It is cleared by software reading SR2 register or by hardware when the interface is disabled (PE=0).

The SCL line is not held low while BERR=1.

0: No misplaced Start or Stop condition

1: Misplaced Start or Stop condition

Note: *If a Bus Error occurs, a Stop or a repeated Start condition should be generated by the Master to re-synchronize communication, get the transmission acknowledged and the bus released for further communication*

Bit 0 = **GCAL** *General Call bit (slave mode).*

This bit is set by hardware when a general call address is detected on the bus while ENGC=1. It is cleared by hardware detecting a Stop condition (STOPF=1) or when the interface is disabled (PE=0).

0: No general call address detected on bus

1: general call address detected on bus

I^{2}C Clock Control register (I2CCCR)

Reset value: 0000 0000 (00h)

7							0
FM/SM	CC6	CC5	CC4	CC3	CC2	CC1	CC0
Read / Write							

Bit 7 = **FM/SM** *Fast/Standard I^{2}C mode bit*

This bit is set and cleared by software. It is not cleared when the interface is disabled (PE=0).

0: Standard I^{2}C mode

1: Fast I^{2}C mode

Bits 6:0 = **CC[6:0]** *7-bit clock divider bits*

These bits select the speed of the bus (F_{SCL}) depending on the I^{2}C mode. They are not cleared when the interface is disabled (PE=0).

Refer to the Electrical Characteristics section for the table of values.

Note: *The programmed F_{SCL} assumes no load on SCL and SDA lines.*

I^{2}C Data register (I2CDR)

Reset Value: 0000 0000 (00h)

7							0
D7	D6	D5	D4	D3	D2	D1	D0
Read / Write							

Bits 7:0 = **D[7:0]** *8-bit Data register*

These bits contain the byte to be received or transmitted on the bus.

– Transmitter mode: byte transmission start automatically when the software writes in the DR register.

– Receiver mode: the first data byte is received automatically in the DR register using the least significant bit of the address. Then, the following data bytes are received one by one after reading the DR register.

I²C Own Address register (I2COAR1)

Reset value: 0000 0000 (00h)

7							0
ADD7	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0
Read / Write							

- In 7-bit addressing mode

  Bits 7:1 = **ADD[7:1]** *Interface address*. These bits define the I²C bus address of the interface. They are not cleared when the interface is disabled (PE=0).

  Bit 0 = **ADD0** *Address direction bit.*
  This bit is don't care, the interface acknowledges either 0 or 1. It is not cleared when the interface is disabled (PE=0).

Note: Address 01h is always ignored.

- In 10-bit addressing mode

  Bits 7:0 = **ADD[7:0]** *Interface address*. These are the least significant bits of the I²C bus address of the interface. They are not cleared when the interface is disabled (PE=0).

I²C Own Address register (I2COAR2)

Reset value: 0100 0000 (40h)

7							0
FR1	FR0	0	0	0	ADD9	ADD8	0
Read / Write							

Bits 7:6 = **FR[1:0]** *Frequency bits*

These bits are set by software only when the interface is disabled (PE=0). To configure the interface to I²C specified delays select the value corresponding to the microcontroller frequency f_{CPU}.

Table 48. Configuration of I²C delay times

f_{CPU}	FR1	FR0
< 6 MHz	0	0
6 to 8 MHz	0	1

Bits 5:3 = Reserved

Bits 2:1 = **ADD[9:8]** *Interface address*

These are the most significant bits of the I²C bus address of the interface (10-bit mode only). They are not cleared when the interface is disabled (PE=0).

Bit 0 = Reserved.

Table 49. I^2C register mapping and reset values

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0064h	**I2CCR** Reset Value	 0	 0	PE 0	ENGC 0	START 0	ACK 0	STOP 0	ITE 0
0065h	**I2CSR1** Reset Value	EVF 0	ADD10 0	TRA 0	BUSY 0	BTF 0	ADSL 0	M/SL 0	SB 0
0066h	**I2CSR2** Reset Value	 0	 0	 0	AF 0	STOPF 0	ARLO 0	BERR 0	GCAL 0
0067h	**I2CCCR** Reset Value	FM/SM 0	CC6 0	CC5 0	CC4 0	CC3 0	CC2 0	CC1 0	CC0 0
0068h	**I2COAR1** Reset Value	ADD7 0	ADD6 0	ADD5 0	ADD4 0	ADD3 0	ADD2 0	ADD1 0	ADD0 0
0069h	**I2COAR2** Reset Value	FR1 0	FR0 1	 0	 0	 0	ADD9 0	ADD8 0	 0
006Ah	**I2CDR** Reset Value	MSB 0	 0	 0	 0	 0	 0	 0	LSB 0

10.6 Serial peripheral interface (SPI)

10.6.1 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 or a system in which devices may be either masters or slaves.

10.6.2 Main features

- Full duplex synchronous transfers (on three lines)
- Simplex synchronous transfers (on two 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.

10.6.3 General description

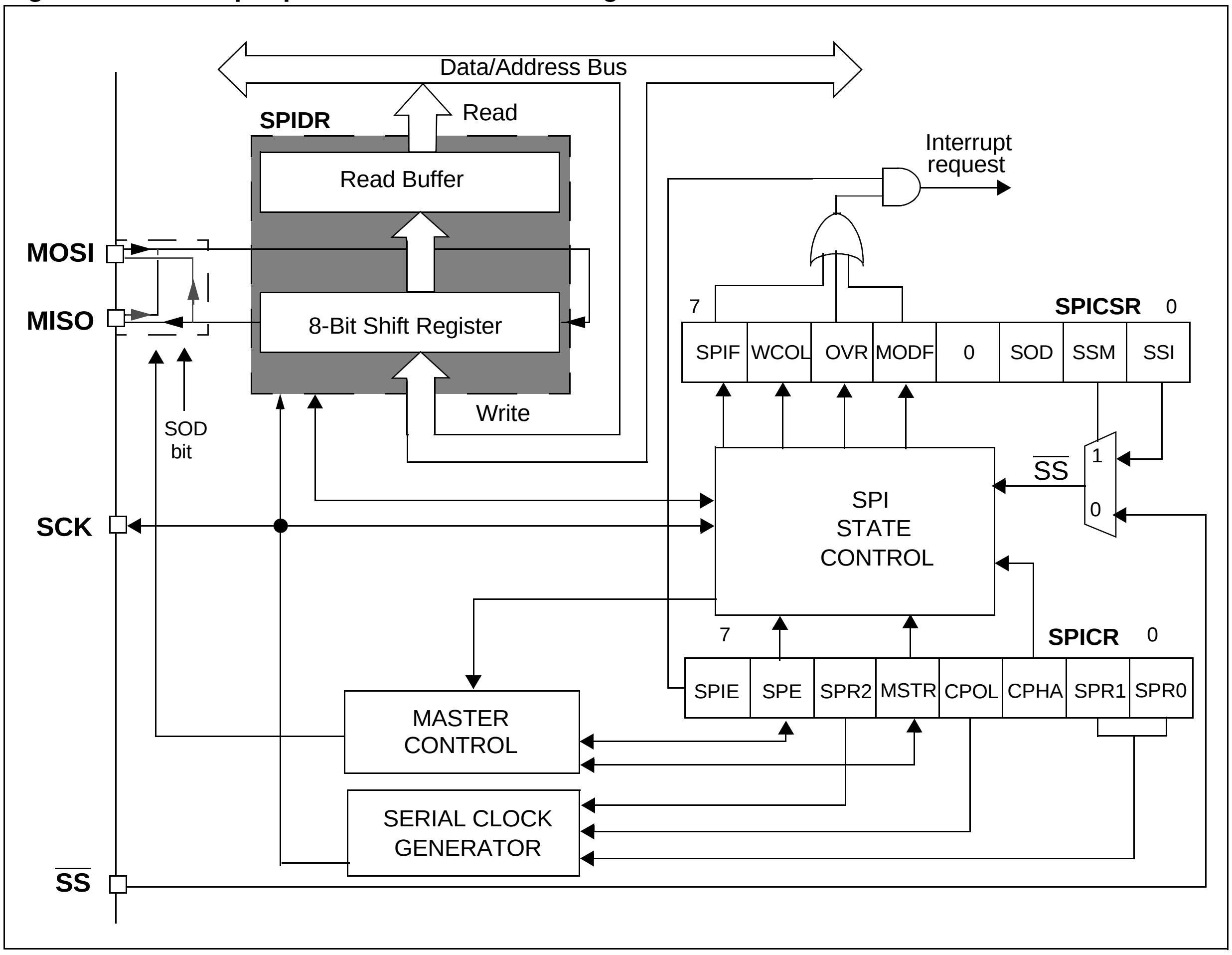
Figure 73 on page 159 shows the serial peripheral interface (SPI) block diagram. There are three registers:

- 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 Device.

Figure 73. Serial peripheral interface block diagram



10.6.4 Functional description

A basic example of interconnections between a single master and a single slave is illustrated in *Figure 74*.

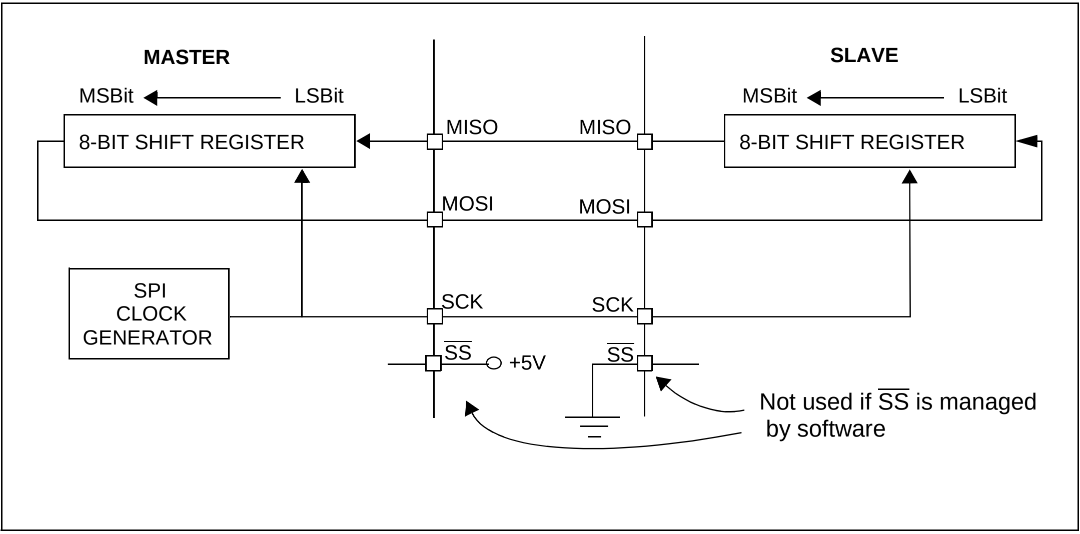
The MOSI pins are connected together and the MISO pins are connected together. In this way data is transferred serially between master and slave (most significant bit first).

The communication is always initiated by the master. When the master device transmits data to a slave device via MOSI pin, the slave device responds by sending data to the master device via the MISO pin. This implies full duplex communication with both data out and data in synchronized with the same clock signal (which is provided by the master device via the SCK pin).

To use a single data line, the MISO and MOSI pins must be connected at each node (in this case only simplex communication is possible).

Four possible data/clock timing relationships may be chosen (see *Figure 77 on page 164*) but master and slave must be programmed with the same timing mode.

Figure 74. Single master/ single slave application



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 76*).

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 75*):

If CPHA = 1 (data latched on second 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 SPICSR register)

If CPHA = 0 (data latched on first 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 *Section : Write collision error (WCOL)*).

Figure 75. Generic $\overline{SS}$ timing diagram

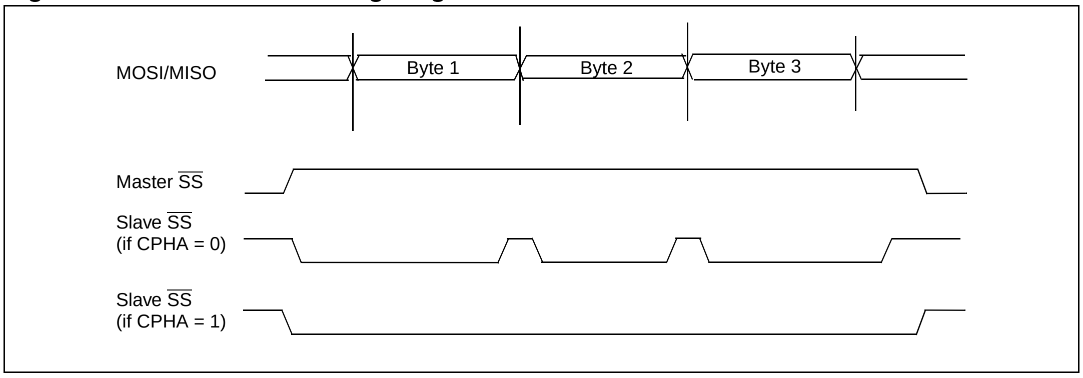
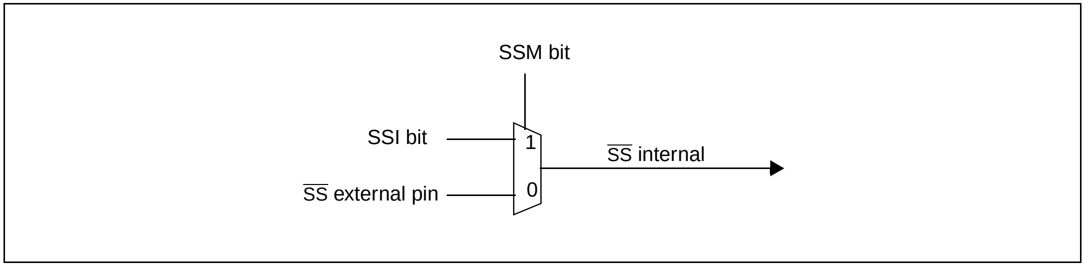


Figure 76. Hardware/software slave select management



Master mode operation

In master mode, the serial clock is output on the SCK pin. The clock frequency, polarity and phase are configured by software (refer to the description of the SPICSR register).

Note: The idle state of SCK must correspond to the polarity selected in the SPICSR register (by pulling up SCK if CPOL = 1 or pulling down SCK if CPOL = 0).

How to operate the SPI in master mode

To operate the SPI in master mode, perform the following steps in order:

1. Write to the SPICR register:
   - Select the clock frequency by configuring the SPR[2:0] bits.
   - Select the clock polarity and clock phase by configuring the CPOL and CPHA bits. *Figure 77* 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).

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

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

Master mode transmit sequence

When software writes to the SPIDR register, the data byte is loaded into the 8-bit shift register and then shifted out serially to the MOSI pin most significant bit first.

When data transfer is complete:

- The SPIF bit is set by hardware.
- An interrupt request is generated if the SPIE bit is set and the interrupt mask in the CCR register is cleared.

Clearing the SPIF bit is performed by the following software sequence:

1. An access to the SPICSR register while the SPIF bit is set
2. A read to the SPIDR register

Note: While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.

Slave mode operation

In slave mode, the serial clock is received on the SCK pin from the master device.

To operate the SPI in slave mode:

1. Write to the SPICSR register to perform the following actions:
   – Select the clock polarity and clock phase by configuring the CPOL and CPHA bits (see *Figure 77*).

Note: The slave must have the same CPOL and CPHA settings as the master.

   – Manage the $\overline{SS}$ pin as described in *Section : Slave select management* and *Figure 75*. If CPHA = 1 $\overline{SS}$ must be held low continuously. If CPHA = 0 $\overline{SS}$ must be held low during byte transmission and pulled up between each byte to let the slave write in the shift register.
2. Write to the SPICR register to clear the MSTR bit and set the SPE bit to enable the SPI I/O functions.

Slave mode transmit sequence

When software writes to the SPIDR register, the data byte is loaded into the 8-bit shift register and then shifted out serially to the MISO pin most significant bit first.

The transmit sequence begins when the slave device receives the clock signal and the most significant bit of the data on its MOSI pin.

When data transfer is complete:

- The SPIF bit is set by hardware.
- An interrupt request is generated if SPIE bit is set and interrupt mask in the CCR register is cleared.

Clearing the SPIF bit is performed by the following software sequence:

1. An access to the SPICSR register while the SPIF bit is set
2. A write or a read to the SPIDR register

Note: While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.

The SPIF bit can be cleared during a second transmission; however, it must be cleared before the second SPIF bit in order to prevent an Overrun condition (see *Section : Overrun condition (OVR)*).

10.6.5 Clock phase and clock polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits (See *Figure 77*).

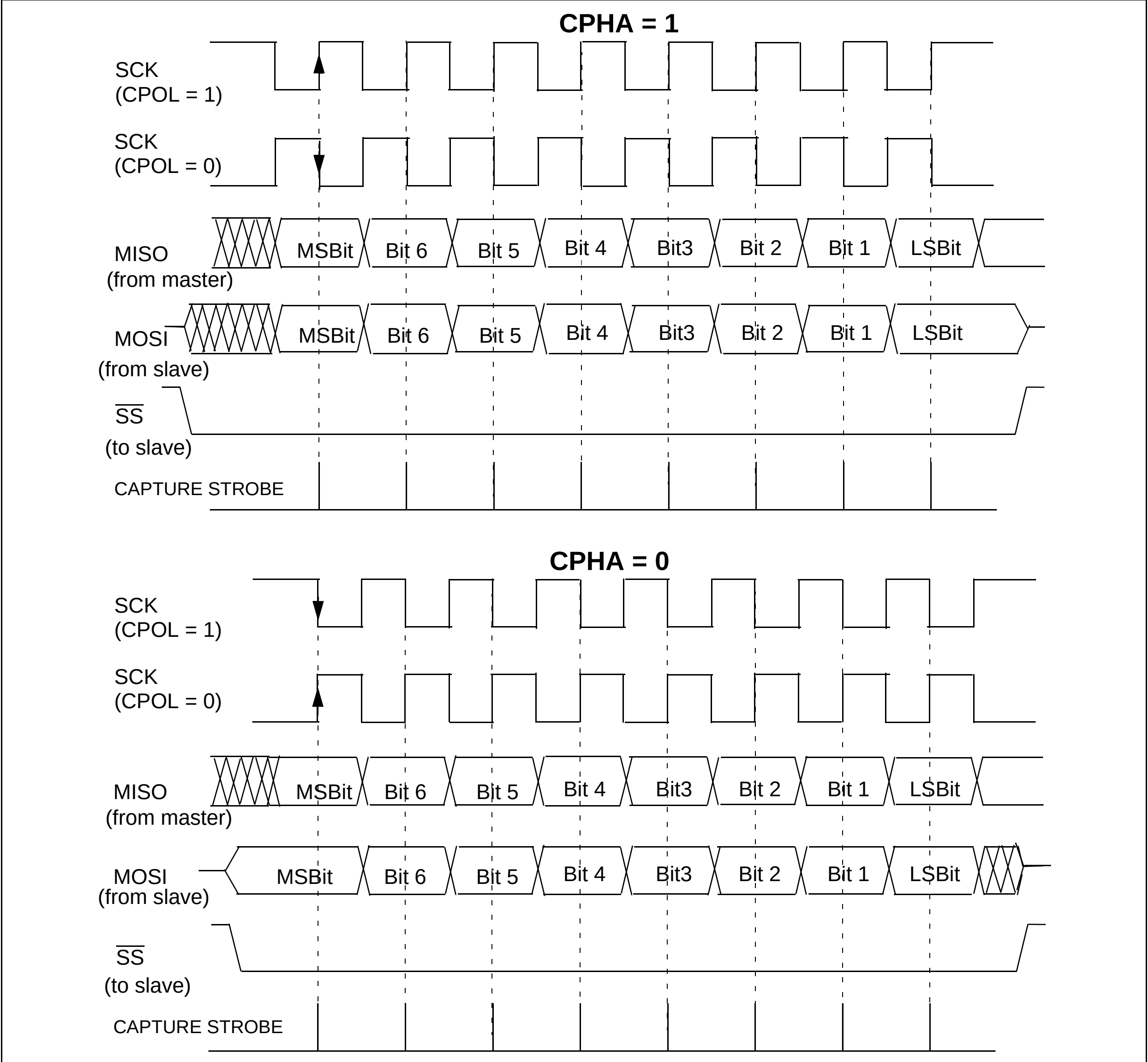
Note: The idle state of SCK must correspond to the polarity selected in the SPICSR register (by pulling up SCK if CPOL = 1 or pulling down SCK if CPOL = 0).

The combination of the CPOL clock polarity and CPHA (clock phase) bits selects the data capture clock edge.

Figure 77 shows an SPI transfer with the four combinations of the CPHA and CPOL bits. The diagram may be interpreted as a master or slave timing diagram where the SCK pin, the MISO pin and the MOSI pin are directly connected between the master and the slave device.

Note: If CPOL is changed at the communication byte boundaries, the SPI must be disabled by resetting the SPE bit.

Figure 77. Data clock timing diagram



Note: This figure should not be used as a replacement for parametric information. Refer to the Electrical Characteristics chapter.

10.6.6 Error flags

Master mode fault (MODF)

Master mode fault occurs when the master device's $\overline{SS}$ pin is pulled low.

When a Master mode fault occurs:

- The MODF bit is set and an SPI interrupt request is generated if the SPIE bit is set.
- The SPE bit is reset. This blocks all output from the device and disables the SPI peripheral.
- The MSTR bit is reset, thus forcing the device into slave mode.

Clearing the MODF bit is done through a software sequence:

1. A read access to the SPICSR register while the MODF bit is set.
2. A write to the SPICR register.

Note: To avoid any conflicts in an application with multiple slaves, the $\overline{SS}$ pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits may be restored to their original state during or after this clearing sequence.

Hardware does not allow the user to set the SPE and MSTR bits while the MODF bit is set except in the MODF bit clearing sequence.

In a slave device, the MODF bit can not be set, but in a multimaster configuration the device can be in slave mode with the MODF bit set.

The MODF bit indicates that there might have been a multimaster conflict and allows software to handle this using an interrupt routine and either perform a reset or return to an application default state.

Overrun condition (OVR)

An overrun condition occurs when the master device has sent a data byte and the slave device has not cleared the SPIF bit issued from the previously transmitted byte.

When an Overrun occurs:

The OVR bit is set and an interrupt request is generated if the SPIE bit is set.

In this case, the receiver buffer contains the byte sent after the SPIF bit was last cleared. A read to the SPIDR register returns this byte. All other bytes are lost.

The OVR bit is cleared by reading the SPICSR register.

Write collision error (WCOL)

A write collision occurs when the software tries to write to the SPIDR register while a data transfer is taking place with an external device. When this happens, the transfer continues uninterrupted and the software write will be unsuccessful.

Write collisions can occur both in master and slave mode. See also *Section : Slave select management*.

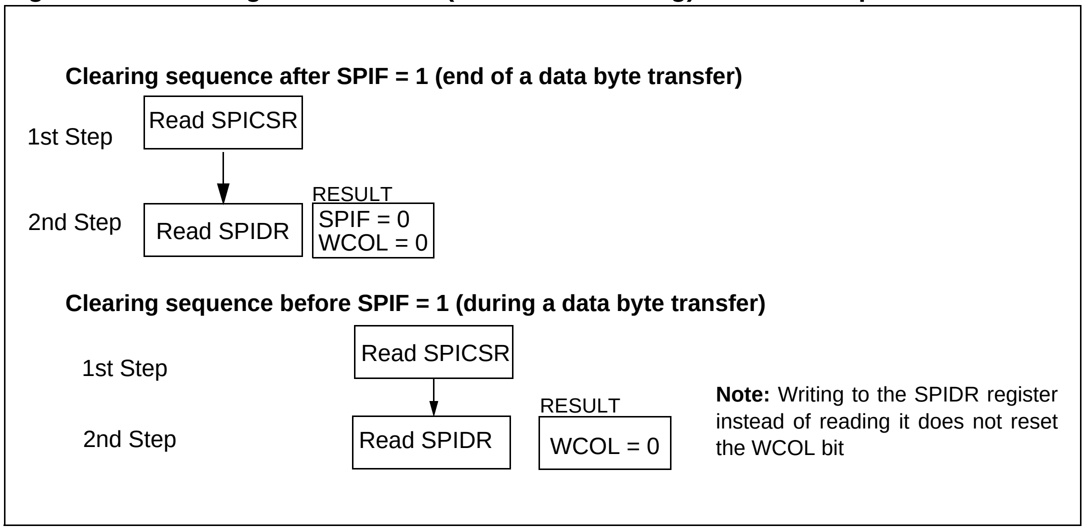
Note: A "read collision" will never occur since the received data byte is placed in a buffer in which access is always synchronous with the CPU operation.

The WCOL bit in the SPICSR register is set if a write collision occurs.

No SPI interrupt is generated when the WCOL bit is set (the WCOL bit is a status flag only).

Clearing the WCOL bit is done through a software sequence (see *Figure 78*).

Figure 78. Clearing the WCOL bit (write collision flag) software sequence


Clearing sequence after SPIF = 1 (end of a data byte transfer)
1st Step
Read SPICSR
2nd Step
Read SPIDR
RESULT
SPIF = 0
WCOL = 0
Clearing sequence before SPIF = 1 (during a data byte transfer)
1st Step
Read SPICSR
2nd Step
Read SPIDR
RESULT
WCOL = 0
Note: Writing to the SPIDR register instead of reading it does not reset the WCOL bit

Single master and multimaster configurations

There are two types of SPI systems:

- Single Master System
- Multimaster System

Single Master System

A typical single master system may be configured using a device as the master and four devices as slaves (see *Figure 79*).

The master device selects the individual slave devices by using four pins of a parallel port to control the four $\overline{SS}$ pins of the slave devices.

The $\overline{SS}$ pins are pulled high during reset since the master device ports will be forced to be inputs at that time, thus disabling the slave devices.

Note: To prevent a bus conflict on the MISO line, the master allows only one active slave device during a transmission.

For more security, the slave device may respond to the master with the received data byte. Then the master will receive the previous byte back from the slave device if all MISO and MOSI pins are connected and the slave has not written to its SPIDR register.

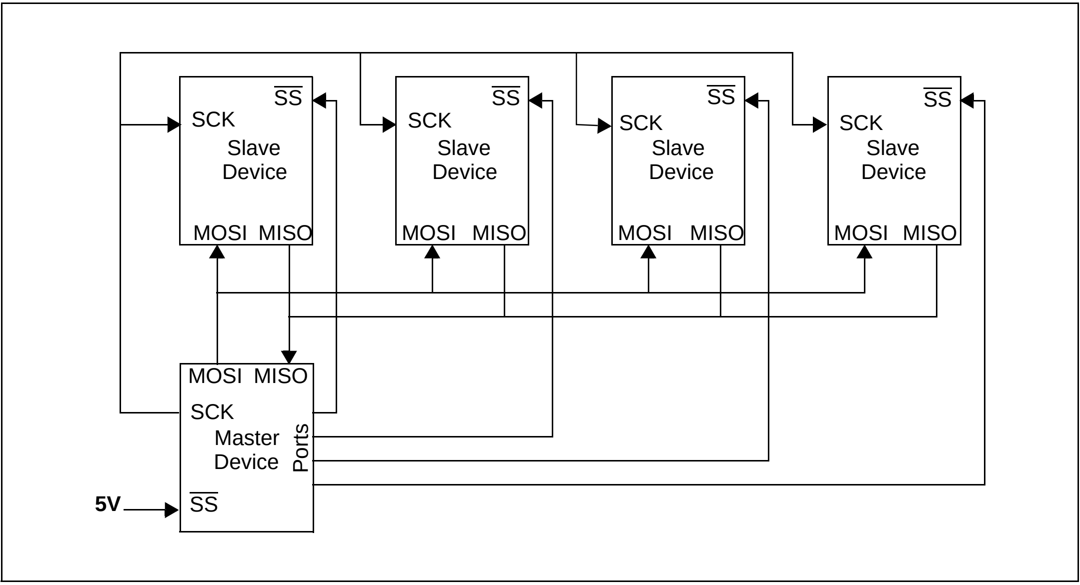
Other transmission security methods can use ports for handshake lines or data bytes with command fields.

Multimaster system

A multimaster system may also be configured by the user. Transfer of master control could be implemented using a handshake method through the I/O ports or by an exchange of code messages through the serial peripheral interface system.

The multimaster system is principally handled by the MSTR bit in the SPICR register and the MODF bit in the SPICSR register.

Figure 79. Single master / multiple slave configuration



10.6.7 Low power modes

Table 50. Low power mode descriptions

Mode	Description
WAIT	No effect on SPI. SPI interrupt events cause the device to exit from WAIT mode.
HALT	SPI registers are frozen. In HALT mode, the SPI is inactive. SPI operation resumes when the device is woken up by an interrupt with “exit from HALT mode” capability. The data received is subsequently read from the SPIDR register when the software is running (interrupt vector fetching). If several data are received before the wakeup event, then an overrun error is generated. This error can be detected after the fetch of the interrupt routine that woke up the Device.

10.6.8 Interrupts

Table 51. Interrupt events

Interrupt event	Event flag	Enable control bit	Exit from Wait	Exit from Halt
SPI End of Transfer Event	SPIF	SPIE	Yes	No
Master Mode Fault Event	MODF			No
Overrun Error	OVR			

Note: The SPI 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 interrupt mask in the CC register is reset (RIM instruction).

10.6.9 Register description

SPI Control register (SPICR)

Reset value: 0000 xxxx (0xh)

7							0
SPIE	SPE	SPR2	MSTR	CPOL	CPHA	SPR1	SPR0
Read / Write							

Bit 7 = **SPIE** *Serial Peripheral interrupt enable.*

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An SPI interrupt is generated whenever an End of Transfer event, Master Mode Fault or Overrun error occurs (SPIF = 1, MODF = 1 or OVR = 1 in the SPICSR register)

Bit 6 = **SPE** *Serial Peripheral output enable.*

This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$ = 0 (see *Section : Master mode fault (MODF)*). The SPE bit is cleared by reset, so the SPI peripheral is not initially connected to the external pins.

0: I/O pins free for general purpose I/O

1: SPI I/O pin alternate functions enabled

Bit 5 = **SPR2** *Divider enable*.

This bit is set and cleared by software and is cleared by reset. It is used with the SPR[1:0] bits to set the baud rate. Refer to *Table 52: SPI Master mode SCK Frequency*.

0: Divider by 2 enabled

1: Divider by 2 disabled

This bit has no effect in slave mode.

Bit 4 = **MSTR** *Master mode.*

This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$ = 0 (see *Section : Master mode fault (MODF)*).

0: Slave mode

1: Master mode. The function of the SCK pin changes from an input to an output and the functions of the MISO and MOSI pins are reversed.

Bit 3 = **CPOL** *Clock polarity.*

This bit is set and cleared by software. This bit determines the idle state of the serial Clock. The CPOL bit affects both the master and slave modes.

0: SCK pin has a low level idle state

1: SCK pin has a high level idle state

If CPOL is changed at the communication byte boundaries, the SPI must be disabled by resetting the SPE bit.

Bit 2 = **CPHA** *Clock phase.*

This bit is set and cleared by software.

0: The first clock transition is the first data capture edge.

1: The second clock transition is the first capture edge.

The slave must have the same CPOL and CPHA settings as the master.

Bits 1:0 = **SPR[1:0]** *Serial clock frequency.*

These bits are set and cleared by software. Used with the SPR2 bit, they select the baud rate of the SPI serial clock SCK output by the SPI in master mode.

These 2 bits have no effect in slave mode.

Table 52. SPI master mode SCK frequency

Serial Clock	SPR2	SPR1	SPR0
$f_{CPU}/4$	1	0	0
$f_{CPU}/8$	0	0	0
$f_{CPU}/16$	0	0	1
$f_{CPU}/32$	1	1	0
$f_{CPU}/64$	0	1	0
$f_{CPU}/128$	0	1	1

SPI control/status register (SPICSR)

Reset Value: 0000 0000 (00h)

7							0
SPIF	WCOL	OVR	MODF	-	SOD	SSM	SSI
Read / Write (some bits Read only)							

Bit 7 = **SPIF** *Serial Peripheral Data Transfer Flag (Read only)*.

This bit is set by hardware when a transfer has been completed. An interrupt is generated if SPIE = 1 in the SPICR register. It is cleared by a software sequence (an access to the SPICSR register followed by a write or a read to the SPIDR register).

0: Data transfer is in progress or the flag has been cleared.

1: Data transfer between the device and an external device has been completed.

While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.

Bit 6 = **WCOL** *Write Collision status (Read only)*.

This bit is set by hardware when a write to the SPIDR register is done during a transmit sequence. It is cleared by a software sequence (see Figure 75).

0: No write collision occurred

1: A write collision has been detected

Bit 5 = **OVR** *SPI Overrun error (Read only)*.

This bit is set by hardware when the byte currently being received in the shift register is ready to be transferred into the SPIDR register while SPIF = 1 (See Section Overrun condition (OVR)). An interrupt is generated if SPIE = 1 in the SPICR register. The OVR bit is cleared by software reading the SPICSR register.

0: No overrun error

1: Overrun error detected

Bit 4 = **MODF** *Mode Fault flag (Read only)*.

This bit is set by hardware when the SS pin is pulled low in master mode (see Section Master mode fault (MODF)). An SPI interrupt can be generated if SPIE = 1 in the SPICR register. This bit is cleared by a software sequence (An access to the SPICSR register while MODF = 1 followed by a write to the SPICR register).

0: No master mode fault detected

1: A fault in master mode has been detected

Bit 3 = Reserved, must be kept cleared.

Bit 2 = **SOD** *SPI Output Disable*.

This bit is set and cleared by software. When set, it disables the alternate function of the SPI output (MOSI in master mode / MISO in slave mode)

0: SPI output enabled (if SPE = 1)

1: SPI output disabled

Bit 1 = **SSM** *SS Management*.

This bit is set and cleared by software. When set, it disables the alternate function of the SPI SS pin and uses the SSI bit value instead. See Section Slave select management.

0: Hardware management (SS managed by external pin)

1: Software management (internal SS signal controlled by SSI bit. External SS pin free for general-purpose I/O)

Bit 0 = **SSI** *SS Internal Mode*.

This bit is set and cleared by software. It acts as a 'chip select' by controlling the level of the SS slave select signal when the SSM bit is set.

0: Slave selected

1: Slave deselected

SPI data I/O register (SPIDR)

Reset Value: Undefined

7							0
D7	D6	D5	D4	D3	D2	D1	D0
Read / Write							

The SPIDR register is used to transmit and receive data on the serial bus. In a master device, a write to this register will initiate transmission/reception of another byte.

Note: *During the last clock cycle the SPIF bit is set, a copy of the received data byte in the shift register is moved to a buffer. When the user reads the serial peripheral data I/O register, the buffer is actually being read.*

While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.

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 73*).

Table 53. SPI register map and reset values

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

10.7 10-bit A/D converter (ADC)

10.7.1 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 10 multiplexed analog input channels (refer to device pin out description) that allow the peripheral to convert the analog voltage levels from up to 10 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.

10.7.2 Main features

- 10-bit conversion
- Up to 10 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 80*.

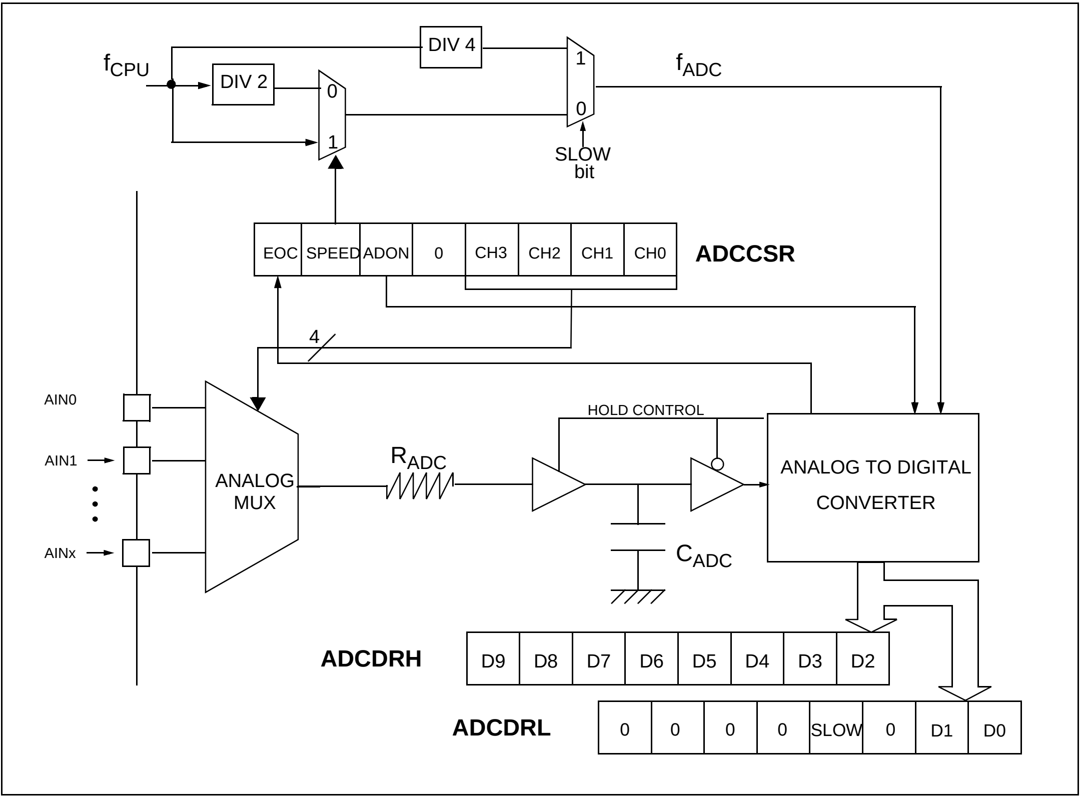
10.7.3 Functional description

Analog power supply

V_{DDA} and V_{SSA} are the high and low level reference voltage pins. In some devices (refer to device pin out description) they are internally connected to the V_{DD} and V_{SS} pins.

Conversion accuracy may therefore be impacted by voltage drops and noise in the event of heavily loaded or badly decoupled power supply lines.

Figure 80. ADC block diagram



Digital A/D conversion result

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_{DDA} (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 the Electrical Characteristics Section.

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 alloted time.

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.

10.7.6 Register description

Control/status register (ADCCSR)

Reset value: 0000 0000 (00h)

7							0
EOC	SPEED	ADON	0	CH3	CH2	CH1	CH0
Read only	Read/write						

Bit 7 = **EOC** *End of Conversion bit*

This bit is set by hardware. It is cleared by hardware when software reads the ADCDRH register or writes to any bit of the ADCCSR register.

0: Conversion is not complete

1: Conversion complete

Bit 6 = **SPEED** *ADC clock selection bit*

This bit is set and cleared by software. It is used together with the SLOW bit to configure the ADC clock speed. Refer to the table in the SLOW bit description (ADCDRL register).

Bit 5 = **ADON** *A/D Converter on bit*

This bit is set and cleared by software.

0: A/D converter is switched off

1: A/D converter is switched on

Bit 4 = Reserved**,** must be kept cleared.

Bits 3:0 = **CH[3:0]** *Channel Selection*

These bits select the analog input to convert. They are set and cleared by software.

Table 55. Channel selection using CH[3:0]

Channel Pin[(1)]	CH3	CH2	CH1	CH0
AIN0	0	0	0	0
AIN1	0	0	0	1
AIN2	0	0	1	0
AIN3	0	0	1	1
AIN4	0	1	0	0
AIN5	0	1	0	1
AIN6	0	1	1	0
AIN7	0	1	1	1
AIN8	1	0	0	0
AIN9	1	0	0	1

1. The number of channels is device dependent. Refer to the device pinout description.

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

11 Instruction set

11.1 ST7 addressing modes

The ST7 core features 17 different addressing modes which can be classified in seven main groups:

Table 58. Description of addressing modes

Addressing mode	Example
Inherent	nop
Immediate	ld A,#$55
Direct	ld A,$55
Indexed	ld A,($55,X)
Indirect	ld A,([$55],X)
Relative	jrne loop
Bit operation	bset byte,#5

The ST7 instruction set is designed to minimize the number of bytes required per instruction: To do so, most of the addressing modes may be subdivided in two submodes called long and short:

- Long addressing mode is more powerful because it can use the full 64 Kbyte address space, however it uses more bytes and more CPU cycles.
- Short addressing mode is less powerful because it can generally only access page zero (0000h - 00FFh range), but the instruction size is more compact, and faster. All memory to memory instructions use short addressing modes only (CLR, CPL, NEG, BSET, BRES, BTJT, BTJF, INC, DEC, RLC, RRC, SLL, SRL, SRA, SWAP)

The ST7 Assembler optimizes the use of long and short addressing modes.

Table 59. ST7 addressing mode overview

Mode			Syntax	Destination/ source	Pointer address	Pointer size	Length (bytes)
Inherent			nop				+ 0
Immediate			ld A,#$55				+ 1
Short	Direct		ld A,$10	00..FF			+ 1
Long	Direct		ld A,$1000	0000..FFFF			+ 2
No Offset	Direct	Indexed	ld A,(X)	00..FF			+ 0 (with X register) + 1 (with Y register)
Short	Direct	Indexed	ld A,($10,X)	00..1FE			+ 1
Long	Direct	Indexed	ld A,($1000,X)	0000..FFFF			+ 2
Short	Indirect		ld A,[$10]	00..FF	00..FF	byte	+ 2
Long	Indirect		ld A,[$10.w]	0000..FFFF	00..FF	word	+ 2
Short	Indirect	Indexed	ld A,([$10],X)	00..1FE	00..FF	byte	+ 2

Table 59. ST7 addressing mode overview (continued)

Mode			Syntax	Destination/ source	Pointer address	Pointer size	Length (bytes)
Long	Indirect	Indexed	ld A,([$10.w],X)	0000..FFFF	00..FF	word	+ 2
Relative	Direct		jrne loop	PC-128/PC+127[1]			+ 1
Relative	Indirect		jrne [$10]	PC-128/PC+127[1]	00..FF	byte	+ 2
Bit	Direct		bset $10,#7	00..FF			+ 1
Bit	Indirect		bset [$10],#7	00..FF	00..FF	byte	+ 2
Bit	Direct	Relative	btjt $10,#7,skip	00..FF			+ 2
Bit	Indirect	Relative	btjt [$10],#7,skip	00..FF	00..FF	byte	+ 3

1. At the time the instruction is executed, the Program Counter (PC) points to the instruction following JRxx.

11.1.1 Inherent mode

All Inherent instructions consist of a single byte. The opcode fully specifies all the required information for the CPU to process the operation.

Table 60. Instructions supporting inherent addressing mode

Instruction	Function
NOP	No operation
TRAP	S/W interrupt
WFI	Wait for interrupt (low power mode)
HALT	Halt oscillator (lowest power mode)
RET	Subroutine return
IRET	Interrupt subroutine return
SIM	Set interrupt mask
RIM	Reset interrupt mask
SCF	Set carry flag
RCF	Reset carry flag
RSP	Reset stack pointer
LD	Load
CLR	Clear
PUSH/POP	Push/Pop to/from the stack
INC/DEC	Increment/decrement
TNZ	Test negative or zero
CPL, NEG	1 or 2 complement

Table 60. Instructions supporting inherent addressing mode (continued)

Instruction	Function
MUL	Byte multiplication
SLL, SRL, SRA, RLC, RRC	Shift and rotate operations
SWAP	Swap nibbles

11.1.2 Immediate mode

Immediate instructions have 2 bytes, the first byte contains the opcode, the second byte contains the operand value.

Table 61. Instructions supporting inherent immediate addressing mode

Immediate Instruction	Function
LD	Load
CP	Compare
BCP	Bit compare
AND, OR, XOR	Logical operations
ADC, ADD, SUB, SBC	Arithmetic operations

11.1.3 Direct modes

In Direct instructions, the operands are referenced by their memory address.

The direct addressing mode consists of two submodes:

Direct (Short) addressing mode

The address is a byte, thus requires only 1 byte after the opcode, but only allows 00 - FF addressing space.

Direct (Long) addressing mode

The address is a word, thus allowing 64 Kbyte addressing space, but requires 2 bytes after the opcode.

11.1.4 Indexed modes (No Offset, Short, Long)

In this mode, the operand is referenced by its memory address, which is defined by the unsigned addition of an index register (X or Y) with an offset.

The indirect addressing mode consists of three submodes:

Indexed mode (No Offset)

There is no offset (no extra byte after the opcode), and allows 00 - FF addressing space.

Indexed mode (Short)

The offset is a byte, thus requires only 1 byte after the opcode and allows 00 - 1FE addressing space.

Indexed mode (long)

The offset is a word, thus allowing 64 Kbyte addressing space and requires 2 bytes after the opcode.

11.1.5 Indirect modes (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 mode (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 mode (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.

11.1.6 Indirect indexed modes (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 mode (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 mode (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 62. Instructions supporting direct, indexed, indirect and indirect indexed addressing modes

Instructions	Function
Long and short instructions	
LD	Load
CP	Compare
AND, OR, XOR	Logical operations
ADC, ADD, SUB, SBC	Arithmetic addition/subtraction operations
BCP	Bit compare

Table 62. Instructions supporting direct, indexed, indirect and indirect indexed addressing modes (continued)

Instructions	Function
Short instructions only	
CLR	Clear
INC, DEC	Increment/decrement
TNZ	Test negative or zero
CPL, NEG	1 or 2 complement
BSET, BRES	Bit operations
BTJT, BTJF	Bit test and jump operations
SLL, SRL, SRA, RLC, RRC	Shift and rotate operations
SWAP	Swap nibbles
CALL, JP	Call or jump subroutine

11.1.7 Relative modes (direct, indirect)

This addressing mode is used to modify the PC register value by adding an 8-bit signed offset to it.

Table 63. Instructions supporting relative modes

Available Relative Direct/Indirect instructions	Function
JRxx	Conditional jump
CALLR	Call relative

The relative addressing mode consists of two submodes:

Relative mode (Direct)

The offset follows the opcode.

Relative mode (Indirect)

The offset is defined in memory, of which the address follows the opcode.

11.2 Instruction groups

The ST7 family devices use an Instruction Set consisting of 63 instructions. The instructions may be subdivided into 13 main groups as illustrated in the following table:

Table 64. ST7 instruction set

Load and Transfer	LD	CLR						
Stack operation	PUSH	POP	RSP					
Increment/decrement	INC	DEC						
Compare and tests	CP	TNZ	BCP					
Logical operations	AND	OR	XOR	CPL	NEG			
Bit operation	BSET	BRES						
Conditional bit test and branch	BTJT	BTJF						
Arithmetic operations	ADC	ADD	SUB	SBC	MUL			
Shift and rotate	SLL	SRL	SRA	RLC	RRC	SWAP	SLA	
Unconditional jump or call	JRA	JRT	JRF	JP	CALL	CALLR	NOP	RET
Conditional branch	JRxx							
Interruption management	TRAP	WFI	HALT	IRET				
Condition Code Flag modification	SIM	RIM	SCF	RCF				

Using a prebyte

The instructions are described with 1 to 4 bytes.

In order to extend the number of available opcodes for an 8-bit CPU (256 opcodes), three different prebyte opcodes are defined. These prebytes modify the meaning of the instruction they precede.

The whole instruction becomes by:

PC-2 End of previous instruction

PC-1 Prebyte

PC Opcode

PC+1 Additional word (0 to 2) according to the number of bytes required to compute the effective address

These prebytes enable instruction in Y as well as indirect addressing modes to be implemented. They precede the opcode of the instruction in X or the instruction using direct addressing mode. The prebytes are:

PDY 90 Replace an X based instruction using immediate, direct, indexed, or inherent addressing mode by a Y one.

PIX 92 Replace an instruction using direct, direct bit or direct relative addressing mode to an instruction using the corresponding indirect addressing mode.
It also changes an instruction using X indexed addressing mode to an instruction using indirect X indexed addressing mode.

PIY 91 Replace an instruction using X indirect indexed addressing mode by a Y one.

11.2.1 Illegal opcode reset

In order to provide enhanced robustness to the device against unexpected behavior, a system of illegal opcode detection is implemented: a reset is generated if the code to be executed does not correspond to any opcode or prebyte value. This, combined with the Watchdog, allows the detection and recovery from an unexpected fault or interference.

A valid prebyte associated with a valid opcode forming an unauthorized combination does not generate a reset.

Table 65. Illegal opcode detection

Mnemo	Description	Function/Example	Dst	Src		H	I	N	Z	C
ADC	Add with Carry	A = A + M + C	A	M		H		N	Z	C
ADD	Addition	A = A + M	A	M		H		N	Z	C
AND	Logical And	A = A . M	A	M				N	Z	
BCP	Bit compare A, Memory	tst (A . M)	A	M				N	Z	
BRES	Bit Reset	bres Byte, #3	M							
BSET	Bit Set	bset Byte, #3	M							
BTJF	Jump if bit is false (0)	btjf Byte, #3, Jmp1	M							C
BTJT	Jump if bit is true (1)	btjt Byte, #3, Jmp1	M							C
CALL	Call subroutine									
CALLR	Call subroutine relative									
CLR	Clear		reg, M					0	1	
CP	Arithmetic Compare	tst(Reg - M)	reg	M				N	Z	C
CPL	One Complement	A = FFH-A	reg, M					N	Z	1
DEC	Decrement	dec Y	reg, M					N	Z	
HALT	Halt						0			
IRET	Interrupt routine return	Pop CC, A, X, PC				H	I	N	Z	C
INC	Increment	inc X	reg, M					N	Z	
JP	Absolute Jump	jp [TBL.w]								
JRA	Jump relative always									
JRT	Jump relative									
JRF	Never jump	jrf *								
JRIH	Jump if ext. interrupt = 1									
JRIL	Jump if ext. interrupt = 0									
JRH	Jump if H = 1	H = 1 ?								
JRNH	Jump if H = 0	H = 0 ?								
JRM	Jump if I = 1	I = 1 ?								
JRNM	Jump if I = 0	I = 0 ?								
JRMI	Jump if N = 1 (minus)	N = 1 ?								

Table 65. Illegal opcode detection (continued)

Mnemo	Description	Function/Example	Dst	Src		H	I	N	Z	C
JRPL	Jump if N = 0 (plus)	N = 0 ?								
JREQ	Jump if Z = 1 (equal)	Z = 1 ?								
JRNE	Jump if Z = 0 (not equal)	Z = 0 ?								
JRC	Jump if C = 1	C = 1 ?								
JRNC	Jump if C = 0	C = 0 ?								
JRULT	Jump if C = 1	Unsigned <								
JRUGE	Jump if C = 0	Jmp if unsigned >=								
JRUGT	Jump if (C + Z = 0)	Unsigned >								
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		H	I	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	I = 0					0			
RLC	Rotate left true C	C <= Dst <= C	reg, M					N	Z	C
RRC	Rotate right true C	C => Dst => 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	I = 1					1			
SLA	Shift left Arithmetic	C <= Dst <= 0	reg, M					N	Z	C
SLL	Shift left Logic	C <= Dst <= 0	reg, M					N	Z	C
SRL	Shift right Logic	0 => Dst => C	reg, M					0	Z	C
SRA	Shift right Arithmetic	Dst7 => Dst => C	reg, M					N	Z	C
SUB	Subtraction	A = A - M	A	M				N	Z	C
SWAP	SWAP nibbles	Dst[7..4]<=>Dst[3..0]	reg, M					N	Z	
TNZ	Test for Neg & Zero	tnz lbl1						N	Z	
TRAP	S/W trap	S/W interrupt					1			

Table 65. Illegal opcode detection (continued)

Mnemo	Description	Function/Example	Dst	Src		H	I	N	Z	C
WFI	Wait for Interrupt						0			
XOR	Exclusive OR	A = A XOR M	A	M				N	Z	

12 Electrical characteristics

12.1 Parameter conditions

Unless otherwise specified, all voltages are referred to V_{SS}.

12.1.1 Minimum and maximum values

Unless otherwise specified the minimum and maximum values are guaranteed in the worst conditions of ambient temperature, supply voltage and frequencies by tests in production on 100% of the devices with an ambient temperature at T_A = 25 °C and T_A = $T_A$max (given by the selected temperature range).

Data based on characterization results, design simulation and/or technology characteristics are indicated in the table footnotes and are not tested in production. Based on characterization, the minimum and maximum values refer to sample tests and represent the mean value plus or minus three times the standard deviation (mean±3Σ).

12.1.2 Typical values

Unless otherwise specified, typical data are based on T_A = 25 °C, V_{DD} = 5 V (for the 4.5 V≤ V_{DD}≤ 5.5 V voltage range). They are given only as design guidelines and are not tested.

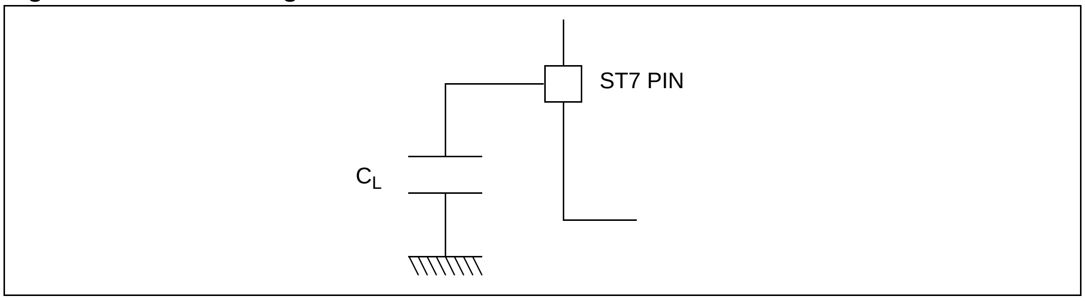
12.1.3 Typical curves

Unless otherwise specified, all typical curves are given only as design guidelines and are not tested.

12.1.4 Loading capacitor

The loading conditions used for pin parameter measurement are shown in *Figure 81*.

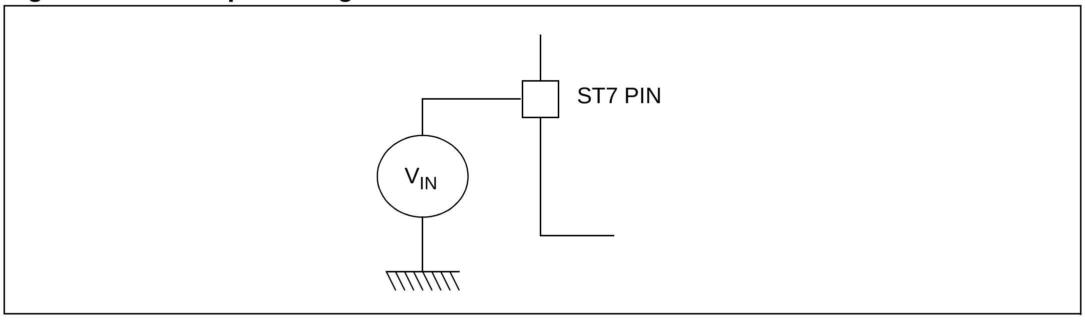
Figure 81. Pin loading conditions



12.1.5 Pin input voltage

The input voltage measurement on a pin of the device is described in *Figure 82*.

Figure 82. Pin input voltage



12.2 Absolute maximum ratings

Stresses above those listed as “absolute maximum ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device under these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

Table 66. Voltage characteristics

Symbol	Ratings	Maximum value	Unit
V_{DD} - V_{SS}	Supply voltage	7.0	V
V_{IN}	Input voltage on any pin(1)(2)	V_{SS}-0.3 to V_{DD}+0.3	
$V_{ESD(HBM)}$	Electrostatic discharge voltage (Human Body model)	see *Section 12.8.3 on page 203*	
$V_{ESD(CDM)}$	Electrostatic discharge voltage (Charge Device model)		

1. Directly connecting the $\overline{\text{RESET}}$ and I/O pins to V_{DD} or V_{SS} could damage the device if an unintentional internal reset is generated or an unexpected change of the I/O configuration occurs (for example, due to a corrupted Program Counter). To guarantee safe operation, this connection has to be done through a pull-up or pull-down resistor (typical: 4.7 kΩ for $\overline{\text{RESET}}$, 10 kΩ for I/Os). Unused I/O pins must be tied in the same way to V_{DD} or V_{SS} according to their reset configuration.

2. $I_{INJ(PIN)}$ must never be exceeded. This is implicitly insured if V_{IN} maximum is respected. If V_{IN} maximum cannot be respected, the injection current must be limited externally to the $I_{INJ(PIN)}$ value. A positive injection is induced by $V_{IN}>V_{DD}$ while a negative injection is induced by $V_{IN}<V_{SS}$. For true open-drain pads, there is no positive injection current, and the corresponding V_{IN} maximum must always be respected

Table 67. Current characteristics

Symbol	Ratings	Maximum value	Unit
I_{VDD}	Total current into V_{DD} power lines (source)(1)	75	mA
I_{VSS}	Total current out of V_{SS} ground lines (sink)(1)	150	
I_{IO}	Output current sunk by any standard I/O and control pin	20	
	Output current sunk by any high sink I/O pin	40	
	Output current source by any I/Os and control pin	- 25	
$I_{INJ(PIN)}$(2)(3)	Injected current on $\overline{\text{RESET}}$ pin	± 5	
	Injected current on OSC1/CLKIN and OSC2 pins	± 5	
	Injected current on any other pin(4)	± 5	
$\Sigma I_{INJ(PIN)}$(2)	Total injected current (sum of all I/O and control pins)(4)	± 20	

1. All power (V_{DD}) and ground (V_{SS}) lines must always be connected to the external supply.
2. $I_{INJ(PIN)}$ must never be exceeded. This is implicitly insured if V_{IN} maximum is respected. If V_{IN} maximum cannot be respected, the injection current must be limited externally to the $I_{INJ(PIN)}$ value. A positive injection is induced by $V_{IN}>V_{DD}$ while a negative injection is induced by $V_{IN}<V_{SS}$. For true open-drain pads, there is no positive injection current, and the corresponding V_{IN} maximum must always be respected
3. Negative injection disturbs the analog performance of the device. In particular, it induces leakage currents throughout the device including the analog inputs. To avoid undesirable effects on the analog functions, care must be taken:
   - Analog input pins must have a negative injection less than 0.8 mA (assuming that the impedance of the analog voltage is lower than the specified limits)
   - Pure digital pins must have a negative injection less than 1.6 mA. In addition, it is recommended to inject the current as far as possible from the analog input pins.
4. When several inputs are submitted to a current injection, the maximum $\Sigma I_{INJ(PIN)}$ is the absolute sum of the positive and negative injected currents (instantaneous values). These results are based on characterization with $\Sigma I_{INJ(PIN)}$ maximum current injection on four I/O port pins of the device.

Table 68. Thermal characteristics

Symbol	Ratings	Value	Unit
T_{STG}	Storage temperature range	-65 to +150	°C
T_J	Maximum junction temperature (see *Table 101: Thermal characteristics on page 224*)		

12.3 Operating conditions

12.3.1 General operating conditions

T_A = -40 to +85 °C unless otherwise specified.

Table 69. General operating conditions

Symbol	Parameter	Conditions	Min	Max	Unit
V_{DD}	Supply voltage	f_{CPU} = 8 MHz max.	4.5	5.5	V
f_{CPU}	CPU clock frequency	4.5 V≤ V_{DD}≤5.5 V	up to 8		MHz

12.3.2 Operating conditions with Low Voltage Detector (LVD)

T_A = -40 to 85 °C unless otherwise specified.

Table 70. Operating characteristics with LVD

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{IT+(LVD)}$	Reset release threshold (V_{DD} rise)		3.9	4.2	4.5	V
$V_{IT-(LVD)}$	Reset generation threshold (V_{DD} fall)		3.7	4.0	4.3	
V_{hys}	LVD voltage threshold hysteresis	$V_{IT+(LVD)}$-$V_{IT-(LVD)}$		150		mV
V_{tPOR}	V_{DD} rise time rate(1)(2)		2			µs/V
$I_{DD(LVD)}$	LVD current consumption	V_{DD} = 5 V		80	140	µA

1. Not tested in production. The V_{DD} rise time rate condition is needed to ensure a correct device power-on and LVD reset release. When the V_{DD} slope is outside these values, the LVD may not release properly the reset of the MCU.

2. Use of LVD with capacitive power supply: with this type of power supply, if power cuts occur in the application, it is recommended to pull V_{DD} down to 0 V to ensure optimum restart conditions. Refer to circuit example in *Figure 89 on page 207*.

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=RCCR01)	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

12.4 Supply current characteristics

The following current consumption specified for the ST7 functional operating modes over temperature range does not take into account the clock source current consumption. To get the total device consumption, the two current values must be added (except for Halt mode for which the clock is stopped).

12.4.1 Supply current

T_A = -40 to +85 °C unless otherwise specified.

Table 72. Supply current characteristics

Symbol	Parameter	Conditions		Typ	Max	Unit
I_{DD}	Supply current in Run mode(1)	V_{DD}=5V	f_{CPU} = 4 MHz	2.5	4.5(2)	mA
			f_{CPU} = 8 MHz	5.0	9	
	Supply current in Wait mode(3)		f_{CPU} = 4 MHz	1.1	2(2)	
			f_{CPU} = 8 MHz	2	3.5	
	Supply current in Slow mode(4)		f_{CPU}/32 = 250 kHz	550	950	µA
	Supply current in Slow-Wait mode(5)		f_{CPU}/32 = 250 kHz	450	750	
	Supply current in AWUFH mode(6)(7)			50	100(2)	
	Supply current in Active Halt mode			120	250	
	Supply current in Halt mode(8)		T_A = 85 °C	0.5	5	

1. CPU running with memory access, all I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (CLKIN) driven by external square wave, LVD disabled.
2. Data based on characterization, not tested in production.
3. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (CLKIN) driven by external square wave, LVD disabled.
4. Slow mode selected with f_{CPU} based on f_{OSC} divided by 32. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (CLKIN) driven by external square wave, LVD disabled.
5. Slow-Wait mode selected with f_{CPU} based on f_{OSC} divided by 32. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (CLKIN) driven by external square wave, LVD disabled.
6. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load). Data tested in production at V_{DD} max. and f_{CPU} max.
7. This consumption refers to the Halt period only and not the associated run period which is software dependent.
8. All I/O pins in output mode with a static value at V_{SS} (no load), LVD disabled. Data based on characterization results, tested in production at V_{DD} max and f_{CPU} max.

12.4.2 On-chip peripherals

Table 73. On-chip peripheral characteristics

Symbol	Parameter	Conditions		Typ	Unit
$I_{DD(SPI)}$	SPI supply current(1)	f_{CPU}=8 MHz	V_{DD}=5.0 V	200	µA
$I_{DD(AT)}$	12-bit Auto-Reload timer supply current(2)	f_{CPU}=8 MHz	V_{DD}=5.0 V	50	µA
$I_{DD(I2C)}$	I^2C supply current(3)	f_{CPU}=8 MHz	V_{DD}=5.0 V	1000	µA
$I_{DD(ADC)}$	ADC supply current when converting(4)	f_{ADC}=4 MHz	V_{DD}=5.0 V	600	µA

1. Data based on a differential I_{DD} measurement between reset configuration and a permanent SPI master communication (data sent equal to 55h).
2. Data based on a differential I_{DD} measurement between reset configuration (timer stopped) and a timer running in PWM mode at f_{cpu}= 8 MHz.
3. Data based on a differential I_{DD} measurement between reset configuration (I^2C disabled) and a permanent I^2C master communication at 100 kHz (data sent equal to 55h). This measurement include the pad toggling consumption (4.7 kOhm external pull-up on clock and data lines).
4. Data based on a differential I_{DD} measurement between reset configuration and continuous A/D conversions.

12.5 Communication interface characteristics

12.5.1 I²C interface

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

Refer to I/O port characteristics for more details on the input/output alternate function characteristics (SDAI and SCLI). The ST7 I²C interface meets the electrical and timing requirements of the Standard I²C communication protocol.

T_A = -40°C to 85 °C, unless otherwise specified.

Table 74. I²C interface characteristics

Symbol	Parameter	Conditions	Min	Max	Unit
f_{SCL}[1]	I²C SCL frequency	f_{CPU}=4 MHz to 8 MHz, V_{DD}= 4.5 to 5.5 V		400	kHz

1. The I²C interface will not function below the minimum clock speed of 4 MHz (see *Table 75*).

Table 102 gives the values to be written in the I2CCCR register to obtain the required I²C SCL line frequency.

Table 75. SCL frequency (multimaster I²C interface)[1][2][3]

f_{SCL}	I2CCCR Value			
	f_{CPU} = 4 MHz, V_{DD} = 5 V		f_{CPU} = 8 MHz, V_{DD} = 5 V	
	R_P=3.3kΩ	R_P=4.7kΩ	R_P=3.3kΩ	R_P=4.7kΩ
400	NA	NA	84h	84h
300	NA	NA	86h	86h
200	84h	84h	8Ah	8Ah
100	11h	11h	25h	24h
50	25h	25h	4Dh	4Ch
20	61h	62h	FFh	FFh

1. R_P = External pull-up resistance, f_{SCL} = I²C speed.
2. For fast mode speeds, achieved speed can have ±5% tolerance. For other speed ranges, achieved speed can have ±2% tolerance.
3. The above variations depend on the accuracy of the external components used.

12.5.2 SPI interface

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

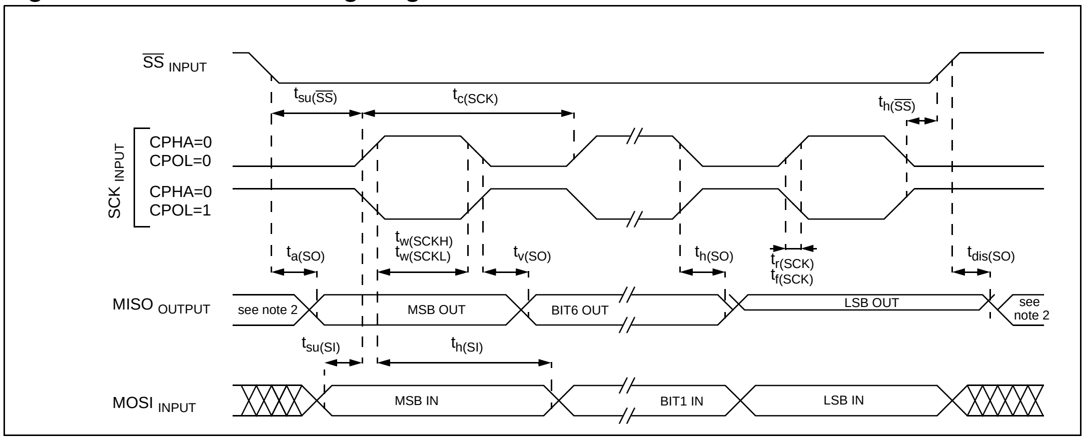
Refer to I/O port characteristics for more details on the input/output alternate function characteristics ($\overline{SS}$, SCK, MOSI, MISO).

Table 76. SPI interface characteristics

Symbol	Parameter	Conditions	Min	Max	Unit
f_{SCK} $1/t_{c(SCK)}$	SPI clock frequency	Master f_{CPU}=8MHz	$f_{CPU}/128$ 0.0625	$f_{CPU}/4$ 2	MHz
		Slave f_{CPU}=8MHz	0	$f_{CPU}/2$ 4	
$t_{r(SCK)}$ $t_{f(SCK)}$	SPI clock rise and fall time		see I/O port pin description		
$t_{su(\overline{SS})}$ (1)	$\overline{SS}$ setup time (2)	Slave	$(4 \times T_{CPU}) + 50$		ns
$t_{h(\overline{SS})}$ (1)	$\overline{SS}$ hold time	Slave	120		
$t_{w(SCKH)}$ (1) $t_{w(SCKL)}$ (1)	SCK high and low time	Master Slave	100 90		
$t_{su(MI)}$ (1) $t_{su(SI)}$ (1)	Data input setup time	Master Slave	100 100		
$t_{h(MI)}$ (1) $t_{h(SI)}$ (1)	Data input hold time	Master Slave	100 100		
$t_{a(SO)}$ (1)	Data output access time	Slave	0	120	
$t_{dis(SO)}$ (1)	Data output disable time	Slave		240	
$t_{v(SO)}$ (1)	Data output valid time	Slave (after enable edge)		120	
$t_{h(SO)}$ (1)	Data output hold time		0		
$t_{v(MO)}$ (1)	Data output valid time	Master (after enable edge)		120	
$t_{h(MO)}$ (1)	Data output hold time		0		

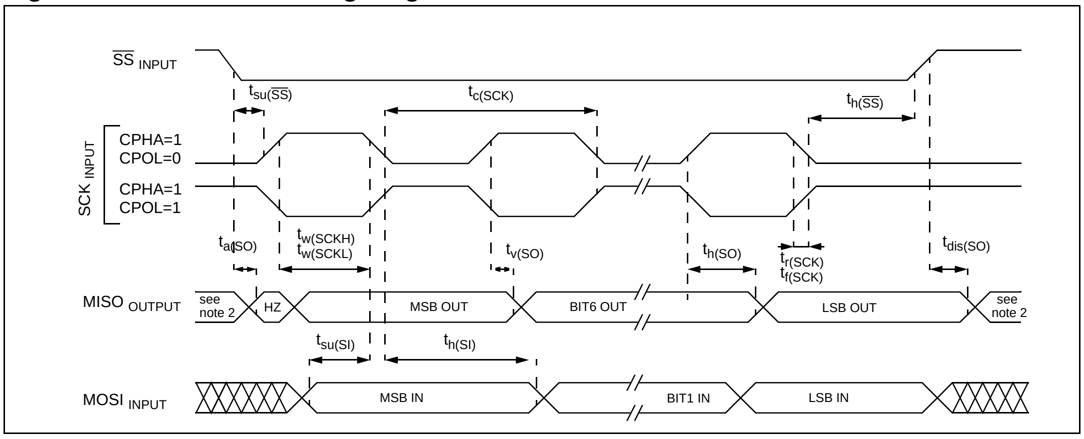
1. Data based on design simulation, not tested in production.
2. Depends on f_{CPU}. For example, if f_{CPU}=8 MHz, then $T_{CPU} = 1/f_{CPU}$ =125 ns and $t_{su(\overline{SS})}$=550 ns

Figure 83. SPI slave timing diagram with CPHA=0



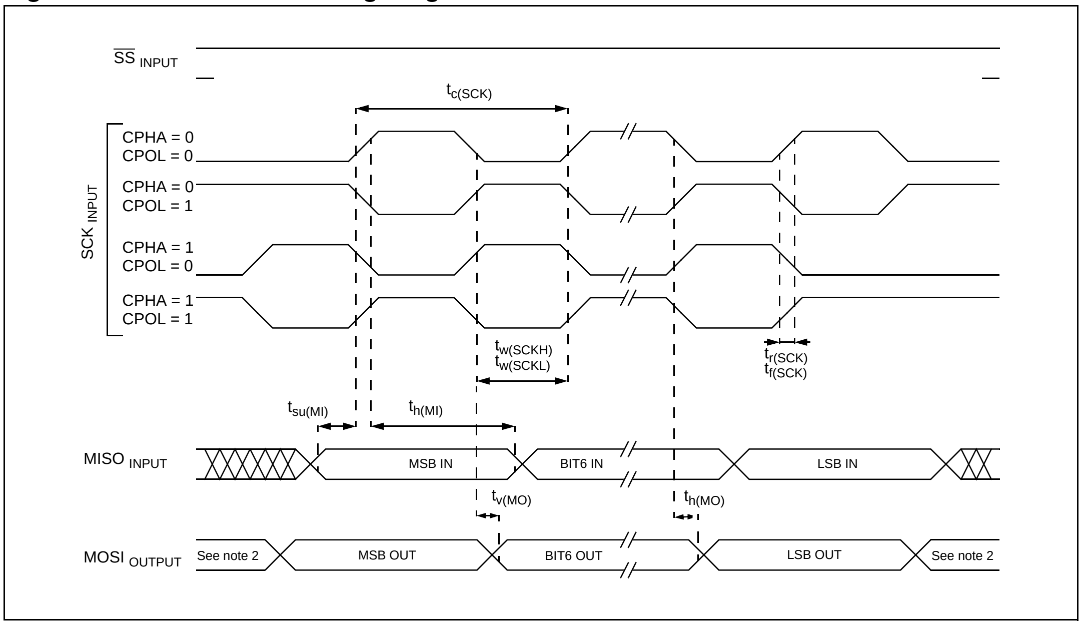
1. Measurement points are done at CMOS levels: $0.3xV_{DD}$ and $0.7xV_{DD}$
2. When no communication is on-going the data output line of the SPI (MOSI in master mode, MISO in slave mode) has its alternate function capability released. In this case, the pin status depends on the I/O port configuration.

Figure 84. SPI slave timing diagram with CPHA=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 on the I/O port configuration.

Figure 85. SPI master timing diagram



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.

12.6 Clock and timing characteristics

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

Table 77. General timings

Symbol	Parameter[(1)]	Conditions	Min	Typ[(2)]	Max	Unit
$t_{c(INST)}$	Instruction cycle time	f_{CPU} = 8 MHz	2	3	12	t_{CPU}
			250	375	1500	ns
$t_{v(IT)}$	Interrupt reaction time[(3)] $t_{v(IT)} = \Delta t_{c(INST)} + 10$	f_{CPU} = 8 MHz	10		22	t_{CPU}
			1.25		2.75	µs

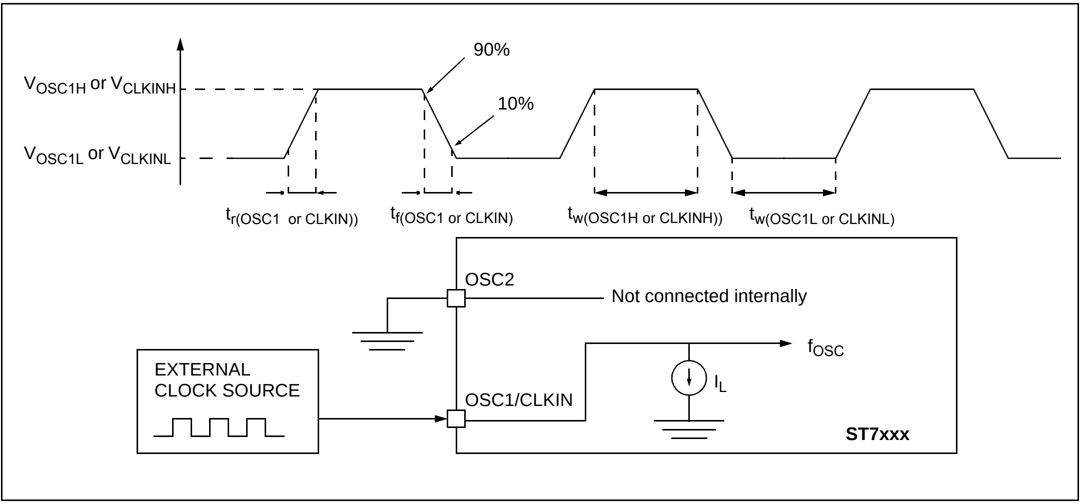
1. Guaranteed by Design. Not tested in production.
2. Data based on typical application software.
3. Time measured between interrupt event and interrupt vector fetch. $\Delta t_{c(INST)}$ is the number of t_{CPU} cycles needed to finish the current instruction execution.

Table 78. External clock source characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{OSC1H} or V_{CLKIN_H}	OSC1/CLKIN input pin high level voltage	see *Figure 86*	$0.7xV_{DD}$		V_{DD}	V
V_{OSC1L} or V_{CLKIN_L}	OSC1/CLKIN input pin low level voltage		V_{SS}		$0.3xV_{DD}$	
$t_{w(OSC1H)}$ or $t_{w(CLKINH)}$ $t_{w(OSC1L)}$ or $t_{w(CLKINL)}$	OSC1/CLKIN high or low time(1)		15			ns
$t_{r(OSC1)}$ or $t_{r(CLKIN)}$ $t_{f(OSC1)}$ or $t_{f(CLKIN)}$	OSC1/CLKIN rise or fall time(1)				15	
I_L	OSCx/CLKIN Input leakage current	$V_{SS} \leq V_{IN} \leq V_{DD}$			±1	µA

1. Data based on design simulation and/or technology characteristics, not tested in production.

Figure 86. Typical application with an external clock source



90%
10%
V_{OSC1H} or V_{CLKINH}
V_{OSC1L} or V_{CLKINL}
$t_{r(OSC1\ or\ CLKIN))}$
$t_{f(OSC1\ or\ CLKIN)}$
$t_{w(OSC1H\ or\ CLKINH))}$
$t_{w(OSC1L\ or\ CLKINL)}$
OSC2
Not connected internally
f_{OSC}
I_L
EXTERNAL CLOCK SOURCE
OSC1/CLKIN
ST7xxx

12.6.1 Auto wakeup from Halt oscillator (AWU)

Table 79. AWU from Halt characteristics

Symbol	Parameter(1)	Conditions	Min	Typ	Max	Unit
f_{AWU}	AWU Oscillator Frequency		16	32	64	kHz
t_{RCSRT}	AWU Oscillator startup time				50	µs

1. Guaranteed by Design. Not tested in production.

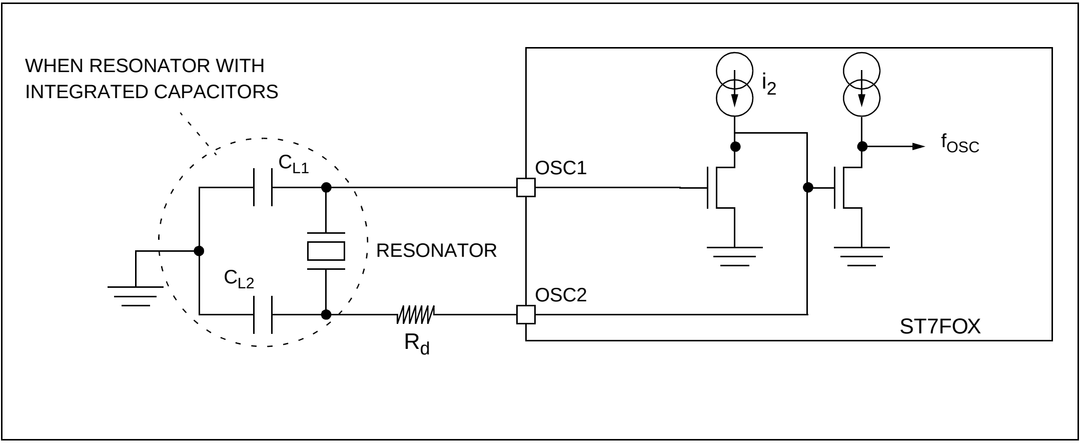
12.6.2 Crystal and ceramic resonator oscillators

The ST7 internal clock can be supplied with ten different Crystal/Ceramic resonator oscillators. All the information given in this paragraph are based on characterization results with specified typical external components. In the application, the resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and start-up stabilization time. Refer to the crystal/ceramic resonator manufacturer for more details (frequency, package, accuracy...).

Table 80. Crystal/ceramic resonator oscillator characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{CrOSC}	Crystal oscillator frequency		2		16	MHz
C_{L1} C_{L2}	Recommended load capacitance versus equivalent serial resistance of the crystal or ceramic resonator (R_S)		TBD			pF

Figure 87. Typical application with a crystal or ceramic resonator



12.7 Memory characteristics

T_A = -40 °C to 85 °C, unless otherwise specified.

Table 81. RAM and hardware registers characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{RM}	Data retention mode(1)	Halt mode (or Reset)	1.6			V

1. Minimum V_{DD} supply voltage without losing data stored in RAM (in Halt mode or under Reset) or in hardware registers (only in Halt mode). Guaranteed by construction, not tested in production.

Table 82. Flash program memory characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DD}	Operating voltage for Flash Write/Erase	Refer to operating range of V_{DD} with T_A, *Section 12.3.1 on page 190*	4.5		5.5	V
t_{prog}	Programming time for 1~32 bytes(1)	T_A=–40 to +85 °C		5	10	ms
	Programming time for 4 kbytes	T_A=+25 °C		0.64	1.28	s
t_{RET}	Data retention(2)	T_A=+55 °C(3)	20			years
N_{RW}	Write erase cycles	T_A=+25 °C			1k	cycles
I_{DD}	Supply current(4)	Read / Write / Erase modes f_{CPU} = 8 MHz, V_{DD} = 5.5 V			2.6	mA
		No Read/No Write mode			100	µA
		Power down mode / Halt		0	0.1	µA

1. Up to 32 bytes can be programmed at a time.
2. Data based on reliability test results and monitored in production.
3. The data retention time increases when the T_A decreases.
4. Guaranteed by Design. Not tested in production.

12.8 EMC (electromagnetic compatibility) characteristics

Susceptibility tests are performed on a sample basis during product characterization.

12.8.1 Functional EMS (electromagnetic susceptibility)

Based on a simple running application on the product (toggling two LEDs through I/O ports), the product is stressed by two electromagnetic events until a failure occurs (indicated by the LEDs).

- **ESD**: Electrostatic Discharge (positive and negative) is applied on all pins of the device until a functional disturbance occurs. This test conforms with the IEC 1000-4-2 standard.
- **FTB**: A Burst of Fast Transient voltage (positive and negative) is applied to V_{DD} and V_{SS} through a 100 pF capacitor, until a functional disturbance occurs. This test conforms with the IEC 1000-4-4 standard.

A device reset allows normal operations to be resumed. The test results are given in the table below based on the EMS levels and classes defined in application note AN1709.

Designing hardened software to avoid noise problems

EMC characterization and optimization are performed at component level with a typical application environment and simplified MCU software. It should be noted that good EMC performance is highly dependent on the user application and the software in particular.

Therefore it is recommended that the user applies EMC software optimization and prequalification tests in relation with the EMC level requested for his application.

- Software recommendations

  The software flowchart must include the management of runaway conditions such as:
  - Corrupted Program Counter
  - Unexpected reset
  - Critical Data corruption (control registers...)
- Prequalification trials

  Most of the common failures (unexpected reset and Program Counter corruption) can be reproduced by manually forcing a low state on the $\overline{\text{RESET}}$ pin or the Oscillator pins for 1 second.

  To complete these trials, ESD stress can be applied directly on the device, over the range of specification values. When unexpected behavior is detected, the software can be hardened to prevent unrecoverable errors occurring (see application note AN1015).

Table 83. EMS test results

Symbol	Parameter	Conditions	Level/ Class
V_{FESD}	Voltage limits to be applied on any I/O pin to induce a functional disturbance	V_{DD}=5 V, T_A=+25 °C, f_{OSC}=8 MHz conforms to IEC 1000-4-2	2B
V_{FFTB}	Fast transient voltage burst limits to be applied through 100pF on V_{DD} and V_{SS} pins to induce a functional disturbance	V_{DD}=5 V, T_A=+25 °C, f_{OSC}=8 MHz conforms to IEC 1000-4-4	3B

12.8.2 EMI (Electromagnetic interference)

Based on a simple application running on the product (toggling two LEDs through the I/O ports), the product is monitored in terms of emission. This emission test is in line with the norm SAE J 1752/3 which specifies the board and the loading of each pin.

Table 84. EMI emissions

Symbol	Parameter	Conditions	Monitored frequency band	Max vs. [f_{OSC}/f_{CPU}]		Unit
				8/4MHz	16/8MHz	
S_{EMI}	Peak level	V_{DD}=5 V, T_A = +25 °C, conforming to SAE J 1752/3	0.1 MHz to 30 MHz	28	32	dBµV
			30 MHz to 130 MHz	31	34	
			130 MHz to 1 GHz	18	26	
			SAE EMI Level	3	3.5	-

12.8.3 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.

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. For more details, refer to the application note AN1181.

Table 85. ESD absolute maximum ratings

Symbol	Ratings	Conditions	Maximum value[1]	Unit
$V_{ESD(HBM)}$	Electrostatic discharge voltage (Human Body model)	T_A=+25 °C	4000	V
$V_{ESD(CDM)}$	Electrostatic discharge voltage (Charge Device model)	T_A=+25 °C	500	

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 86. Electrical sensitivities

Symbol	Parameter	Conditions	Class
LU	Static latch-up class	T_A = +85 °C	A

12.9 I/O port pin characteristics

12.9.1 General characteristics

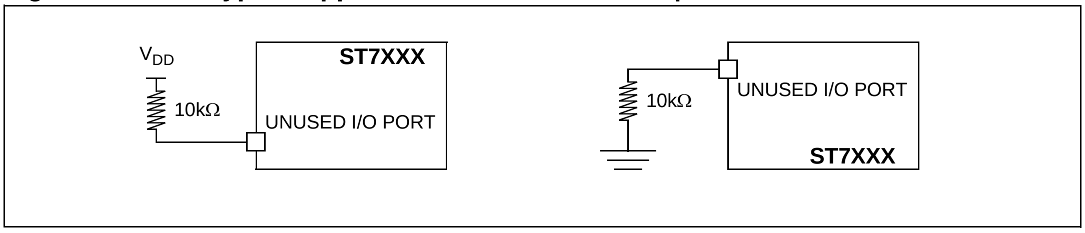
Subject to general operating conditions for V_{DD}, f_{OSC}, and T_A unless otherwise specified.

Table 87. General characteristics

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{IL}	Input low level voltage			V_{SS} - 0.3		$0.3V_{DD}$	V
V_{IH}	Input high level voltage			$0.7V_{DD}$		$V_{DD}+0.3$	
V_{hys}	Schmitt trigger voltage hysteresis(1)				400		mV
I_L	Input leakage current	$V_{SS} \leq V_{IN} \leq V_{DD}$				±1	µA
I_S	Static current consumption induced by each floating input pin(2)	Floating input mode			400		
R_{PU}	Weak pull-up equivalent resistor(3)	$V_{IN}=V_{SS}$	V_{DD}=5 V	100	120	140	kΩ
C_{IO}	I/O pin capacitance				5		pF
$t_{f(IO)out}$	Output high to low level fall time(1)	C_L= 50 pF 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(4)			1			t_{CPU}

1. Data based on validation/design results.
2. Configuration not recommended, all unused pins must be kept at a fixed voltage: using the output mode of the I/O for example or an external pull-up or pull-down resistor (see *Figure 88*). 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.
3. The R_{PU} pull-up equivalent resistor is based on a resistive transistor.
4. 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 88. Two typical applications with unused I/O pin



1. During normal operation the ICCCLK pin must be pulled-up, internally or externally (external pull-up of 10k mandatory in noisy environment). This is to avoid entering ICC mode unexpectedly during a reset.
2. I/O can be left unconnected if it is configured as output (0 or 1) by the software. This has the advantage of greater EMC robustness and lower cost.

12.9.2 Output driving current

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

Table 88. Output driving current characteristics

Symbol	Parameter	Conditions		Min	Max	Unit
V_{OL} [(1)]	Output low level voltage for a standard I/O pin when 8 pins are sunk at same time	$V_{DD} = 5$ V	I_{IO}=+5 mA, $T_A \leq 85°C$		1.0	V
			I_{IO}=+2mA, $T_A \leq 85°C$		0.4	
	Output low level voltage for a high sink I/O pin when 4 pins are sunk at same time		I_{IO}=+20mA,$T_A \leq 85°C$		1.3	
			I_{IO}=+8mA$T_A \leq 85°C$		0.75	
V_{OH} [(2)]	Output high level voltage for an I/O pin when 4 pins are sourced at same time		I_{IO}=-5mA,$T_A \leq 85°C$	V_{DD}-1.5		
			I_{IO}=-2mA$T_A \leq 85°C$	V_{DD}-0.8		

1. The I_{IO} current sunk must always respect the absolute maximum rating specified in *Section Table 67.* and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VSS}.
2. The I_{IO} current sourced must always respect the absolute maximum rating specified in *Section Table 67.* and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VDD}.

12.10 Control pin characteristics

12.10.1 Asynchronous $\overline{\text{RESET}}$ pin

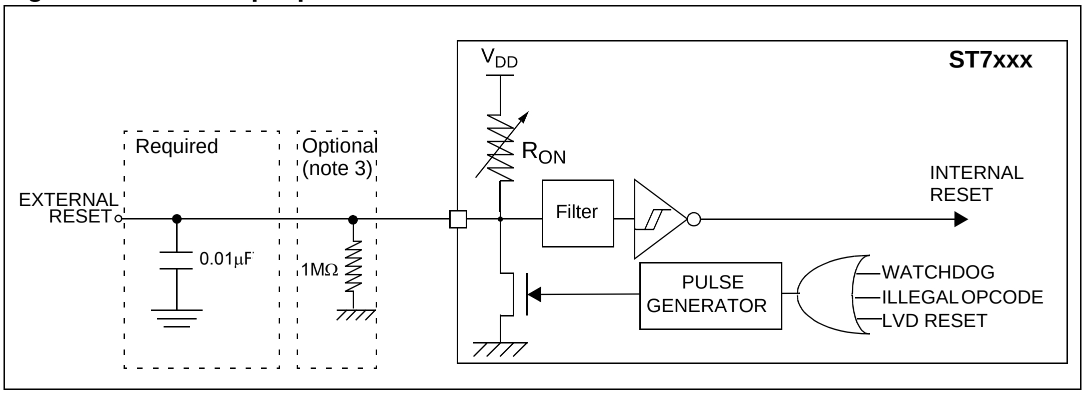
T_A = -40 to 85 °C, unless otherwise specified.

Table 89. Asynchronous $\overline{\text{RESET}}$ pin characteristics

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{IL}	Input low level voltage			V_{SS} - 0.3		$0.3V_{DD}$	V
V_{IH}	Input high level voltage			$0.7V_{DD}$		V_{DD}+0.3	
V_{hys}	Schmitt trigger voltage hysteresis[(1)]				2		V
V_{OL}	Output low level voltage [(2)]	V_{DD}= 5 V	I_{IO} = +2 mA		200		mV
R_{ON}	Pull-up equivalent resistor[(3)]	V_{IN}=V_{SS}	V_{DD} = 5 V	30	50	70	kΩ
$t_{w(RSTL)out}$	Generated reset pulse duration	Internal reset sources			90[(1)]		µs
$t_{h(RSTL)in}$	External reset pulse hold time[(4)]			20			µs
$t_{g(RSTL)in}$	Filtered glitch duration				200		ns

1. Data based on characterization results, not tested in production
2. The I_{IO} current sunk must always respect the absolute maximum rating specified in *Section Table 67. on page 189* and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VSS}.
3. The R_{ON} pull-up equivalent resistor is based on a resistive transistor. Specified for voltages on $\overline{\text{RESET}}$ pin between V_{ILmax} and V_{DD}
4. To guarantee the reset of the device, a minimum pulse has to be applied to the $\overline{\text{RESET}}$ pin. All short pulses applied on $\overline{\text{RESET}}$ pin with a duration below $t_{h(RSTL)in}$ can be ignored.

Figure 89. $\overline{\text{RESET}}$ pin protection when LVD is enabled

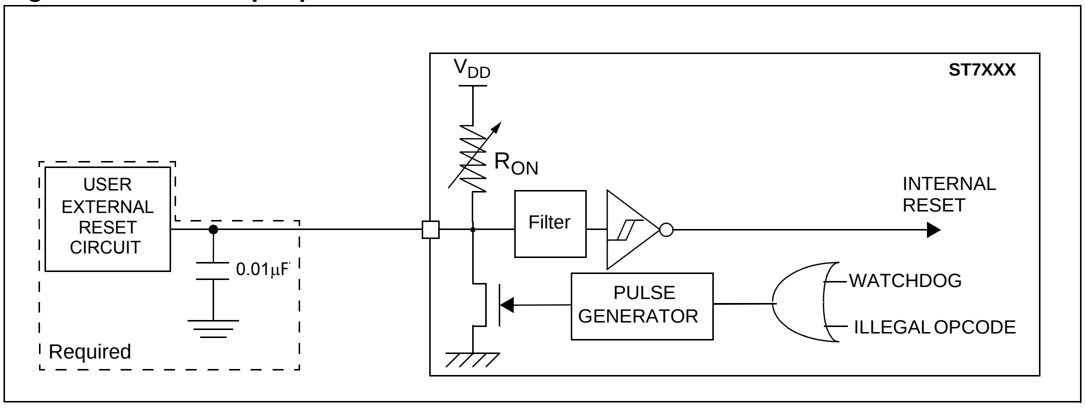


1. The reset network protects the device against parasitic resets. The output of the external reset circuit must have an open-drain output to drive the ST7 reset pad. Otherwise the device can be damaged when the ST7 generates an internal reset (LVD or watchdog). Whatever the reset source is (internal or external), the user must ensure that the level on the $\overline{\text{RESET}}$ pin can go below the V_{IL} max. level specified in *Section 12.10.1 on page 206*. 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 Table 67. on page 189*.
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).

Tips when using the LVD

- Check that all recommendations related to ICCCLK and reset circuit have been applied (see caution in *Table 2 on page 16* and notes above).
- 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.
- 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 1 and 2 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 90. $\overline{\text{RESET}}$ pin protection when LVD is disabled



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).
   Whatever 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} max. level specified in *Section 12.10.1 on page 206*. 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 Table 67. on page 189*.
2. Please refer to *Section 11.2.1 on page 184* for more details on illegal opcode reset conditions.

12.11 10-bit ADC characteristics

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

Table 90. ADC characteristics

Symbol	Parameter	Conditions	Min	Typ[1]	Max	Unit
f_{ADC}	ADC clock frequency				4	MHz
V_{AIN}	Conversion voltage range		V_{SSA}		V_{DDA}	V
R_{AIN}	External input resistor	V_{DD} = 5 V, f_{ADC} = 4 MHz			8k[2]	Ω
		4.5 V ≤ V_{DD} ≤ 5.5 V, f_{ADC} = 2 MHz			10k[2]	
C_{ADC}	Internal sample and hold capacitor			6		pF
t_{STAB}	Stabilization time after ADC enable	f_{CPU} = 8 MHz, f_{ADC} = 4 MHz		0[3]		µs
t_{ADC}	Conversion time (Sample+Hold)			3.5		
	- Sample capacitor loading time - Hold conversion time			4 10		$1/f_{ADC}$

1. Unless otherwise specified, typical data are based on T_A = 25 °C and V_{DD}-V_{SS} = 5 V. They are given only as design guidelines and are not tested.
2. Any added external serial resistor will downgrade the ADC accuracy (especially for resistance greater than the maximum value). Data guaranteed by Design, not tested in production.
3. The stabilization time of the A/D converter is masked by the first t_{LOAD}. The first conversion after the enable is then always valid.

Figure 91. Typical application with ADC

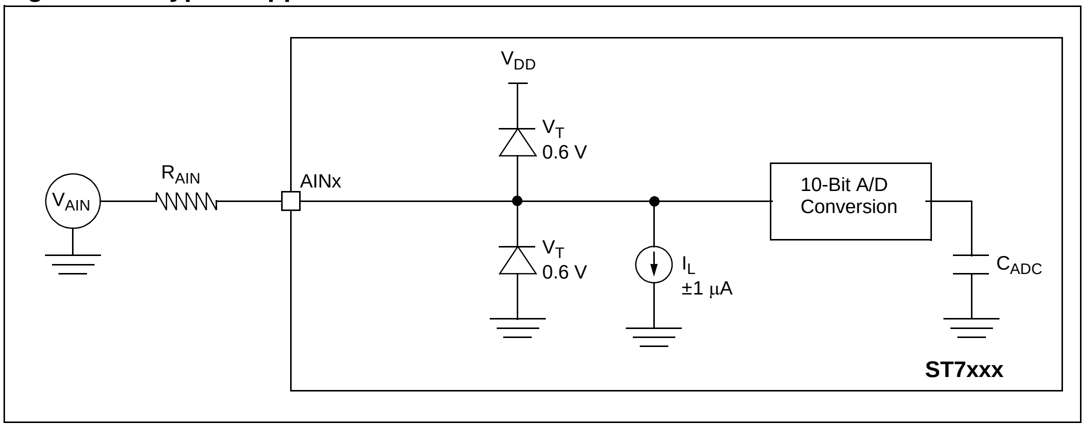
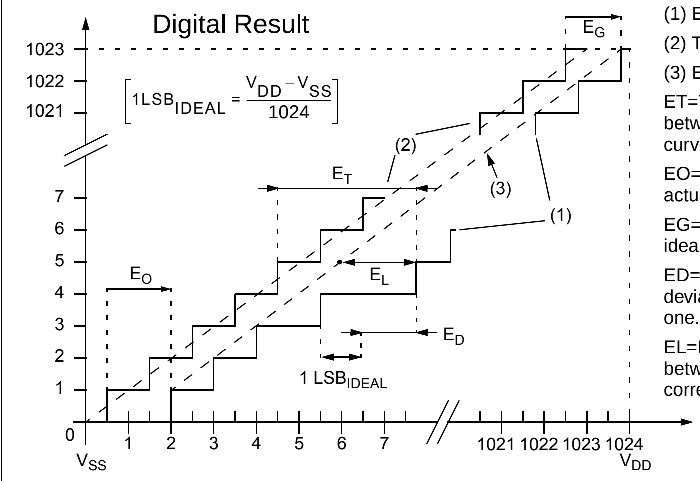


Table 91. ADC accuracy with V_{DD} = 4.5 to 5.5 V

Symbol (1)	Parameter	Conditions	Typ	Max	Unit
$\lvert E_T \rvert$	Total unadjusted error	f_{CPU}=8 MHz, f_{ADC}=4 MHz(1)	2.0	5.0	LSB
$\lvert E_O \rvert$	Offset error		0.9	2.5	
$\lvert E_G \rvert$	Gain Error		1.0	1.5	
$\lvert E_D \rvert$	Differential linearity error		1.2	3.5	
$\lvert E_L \rvert$	Integral linearity error		1.1	4.5	

1. Data based on characterization results over the whole temperature range.

Figure 92. ADC accuracy characteristics



(1) Example of an actual transfer curve

(2) The ideal transfer curve

(3) End point correlation line

ET=Total Unadjusted Error: maximum deviation between the actual and the ideal transfer curves.

EO=Offset Error: deviation between the first actual transition and the first ideal one.

EG=Gain Error: deviation between the last ideal transition and the last actual one.

ED=Differential Linearity Error: maximum deviation between actual steps and the ideal one.

EL=Integral Linearity Error: maximum deviation between any actual transition and the end point correlation line.

13 Device configuration and ordering information

This device is available for production in user programmable version (Flash).

ST7FOX XFlash devices are shipped to customers with a default program memory content (FFh).

13.1 Option bytes

The two option bytes allow the hardware configuration of the microcontroller to be selected. The option bytes can be accessed only in programming mode (for example using a standard ST7 programming tool).

13.1.1 Option byte 1

Bits 7:6 = **CKSEL[1:0]** *Start-up clock selection.*

These bits are used to select the startup frequency. By default, the internal RC is selected.

Table 92. Startup clock selection

Configuration	CKSEL1	CKSEL0
Internal RC as Startup Clock	0	0
AWU RC as a Startup Clock	0	1
External crystal/ceramic resonator	1	0
External Clock	1	1

Bits 5:3 = Reserved, must always be 1.

Bit 2 = **LVD** *Low Voltage Detection selection.*

This option bit enables the low voltage detection block (LVD).

0: LVD on
1: LVD off (default value)

Bit 1 = **WDG SW** *Hardware or software watchdog*

This option bit selects the watchdog type.
0: Hardware (watchdog always enabled)
1: Software (watchdog to be enabled by software)

Bit 0 = **WDG HALT** *Watchdog Reset on Halt*

This option bit determines if a Reset is generated when entering Halt mode while the Watchdog is active.
0: No Reset generation when entering Halt mode
1: Reset generation when entering Halt mode

13.1.2 Option byte 0

OPT 7 = **AWUCK** *Auto Wake Up Clock Selection*

0: 32-kHz Oscillator (VLP) selected as AWU clock

1: AWU RC Oscillator selected as AWU clock.

Note: If this bit is reset, OSCRANGE[2:0] must be set to 100.

OPT6:4 = **OSCRANGE[2:0]** *Oscillator Range*

When the internal RC oscillator is not selected (CKSEL1=1), these option bits (and CKSEL0) select the range of the resonator oscillator current source or the external clock source.

Table 93. Selection of the resonator oscillator range

			OSCRANGE[(1)]		
			2	**1**	**0**
Typ. frequency range with Resonator	LP	1~2 MHz	0	0	0
	MP	2~4 MHz	0	0	1
	MS	4~8 MHz	0	1	0
	HS	8~16 MHz	0	1	1
	VLP	32.768 kHz	1	0	0
External Clock on OSC1/CLKIN			1	0	1
Reserved			1	1	0
External Clock on PB1			1	1	1

1. When the internal RC oscillator is selected, the CLKSEL option bits must be kept at their default value in order to select the 256 clock cycle delay (see *Section 6.3*).

OPT 3:2 = **SEC[1:0]** *Sector 0 size definition*

These option bits indicate the size of sector 0 according to *Table 94*.

Table 94. Configuration of sector size

Sector 0 Size	SEC1	SEC0
0.5k	0	0
1k	0	1
2k	1	0
4k	1	1

Bit 1 = **FMP_R** *Read-Out Protection*

Read-Out Protection, when selected provides a protection against program memory content extraction and against write access to Flash memory. Erasing the option bytes when the FMP_R option is selected will cause the whole memory to be erased first, and the device can be reprogrammed. Refer to *Section 4.5 on page 27* and the ST7 Flash Programming Reference Manual for more details.

0: Read-Out Protection off

1: Read-Out Protection on

Bit 0 = **FMP_W** *Flash write protection*

This option indicates if the Flash program memory is write protected.

0: Write protection off

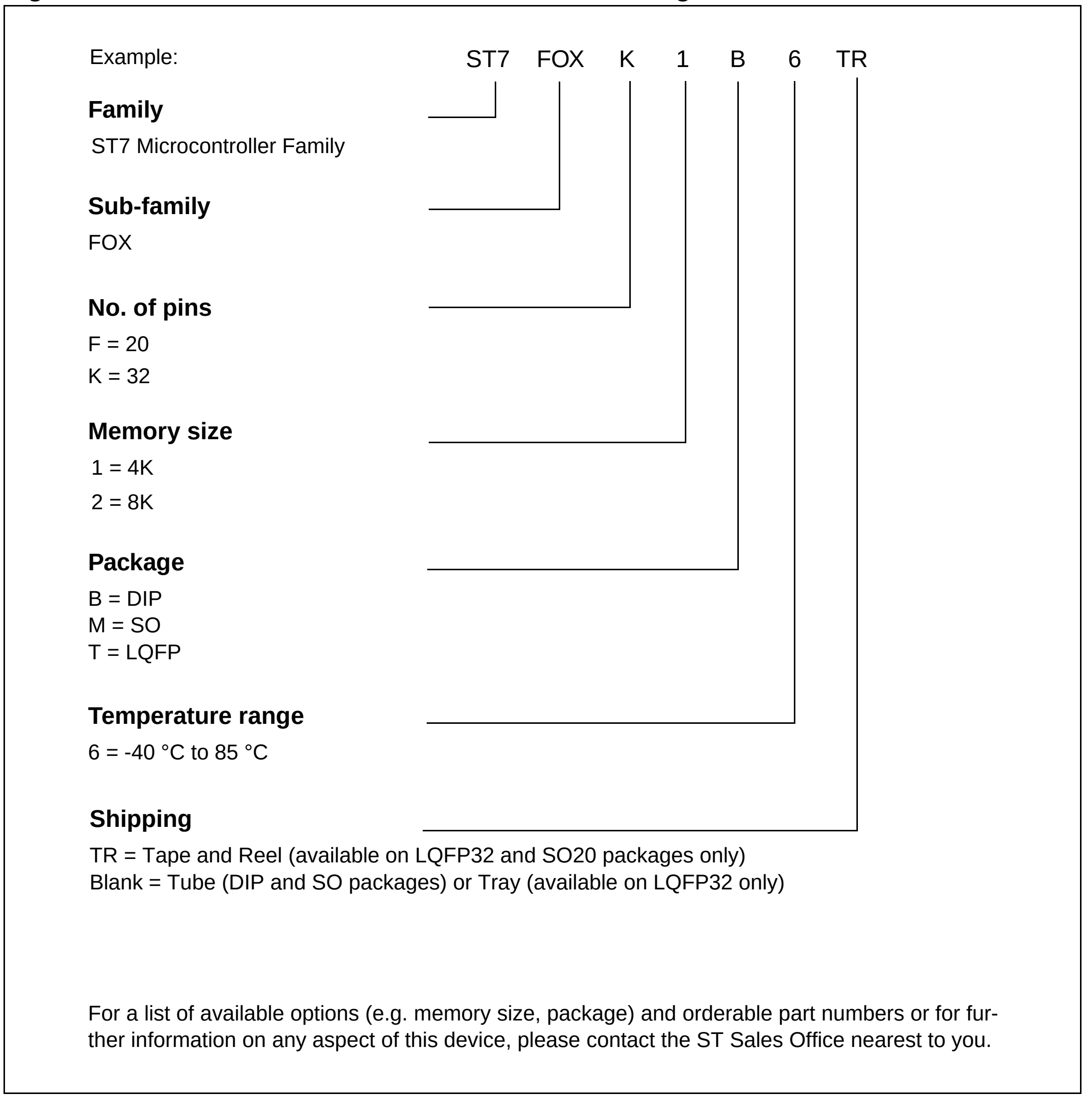
1: Write protection on

Warning: When the Flash write protection is selected, the program memory (and the option bit itself) can never be erased or programmed again.

	Option byte 0								Option byte 1							
	7							0	7							0
	AWU CK	OSCRANGE[2:0]			SEC 1	SEC 0	FMP R	FMP W	CK SEL1	CK SEL0	Res	Res	Res	LVD	WDG SW	WDG HALT
Default value	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1

13.2 Device ordering information

Figure 93. ST7FOXF1/ST7FOXK1/ST7FOXK2 ordering information scheme


Example:
ST7 FOX K 1 B 6 TR
Family
ST7 Microcontroller Family
Sub-family
FOX
No. of pins
F = 20
K = 32
Memory size
1 = 4K
2 = 8K
Package
B = DIP
M = SO
T = LQFP
Temperature range
6 = -40 °C to 85 °C
Shipping
TR = Tape and Reel (available on LQFP32 and SO20 packages only)
Blank = Tube (DIP and SO packages) or Tray (available on LQFP32 only)
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 contact the ST Sales Office nearest to you.

ST7FOX failure analysis service

For ST7FOX family devices, STMicroelectronics agrees to accept return of defective parts subject to the FAR (Failure Analysis Report) procedure only if the customer reject rate exceeds 0.35 % for each delivered batch.

A batch is identified with a single trace code located on the top side marking.

13.3 Development tools

Development tools for the ST7 microcontrollers include a complete range of hardware systems and software tools from STMicroelectronics and third-party tool suppliers. The range of tools includes solutions to help you evaluate microcontroller peripherals, develop and debug your application, and program your microcontrollers.

13.3.1 Starter kits

ST offers complete, affordable **starter kits**. Starter kits are complete hardware/software tool packages that include features and samples to help you quickly start developing your application.

13.3.2 Development and debugging tools

Application development for ST7 is supported by fully optimizing **C Compilers** and the **ST7 Assembler-Linker** toolchain, which are all seamlessly integrated in the ST7 integrated development environments in order to facilitate the debugging and fine-tuning of your application. The Cosmic C Compiler is available in a free version that outputs up to 16Kbytes of code.

The range of hardware tools includes a full-featured **STice**Emulator**,** the low-cost **RLink** and the **ST7-STICK** in-circuit debugger/programmer. These tools are supported by the **ST7 Toolset** from STMicroelectronics, which includes the STVD7 integrated development environment (IDE) with high-level language debugger, editor, project manager and integrated programming interface.

13.3.3 Programming tools

During the development cycle, the **STice** emulator, the **ST7-STICK** and the **RLink** provide in-circuit programming capability for programming the Flash microcontroller on your application board.

ST also provides a low-cost dedicated in-circuit programmer and **ST7 Socket Boards,** which provide all the sockets required for programming any of the devices in a specific ST7 sub-family with any tool with in-circuit programming capability for ST7.

For production programming of ST7 devices, ST's third-party tool partners also provide a complete range of gang and automated programming solutions, which are ready to integrate into your production environment.

13.3.4 Order codes for development and programming tools

Table 95 below lists the ordering codes for the ST7FOX development and programming tools. For additional ordering codes for spare parts and accessories, refer to the online product selector at www.st.com/mcu.

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

Table 96. ST7 application notes (continued)

Identification	Description
AN1046	UART emulation software
AN1047	Managing reception errors with the ST7 SCI peripherals
AN1048	ST7 software LCD Driver
AN1078	PWM duty cycle switch implementing true 0% & 100% duty cycle
AN1082	Description of the ST72141 motor control peripherals registers
AN1083	ST72141 BLDC motor control software and flowchart example
AN1105	ST7 pCAN peripheral driver
AN1129	PWM management for BLDC motor drives using the ST72141
AN1130	An introduction to sensorless brushless DC motor drive applications with the ST72141
AN1148	Using the ST7263 for designing a USB mouse
AN1149	Handling Suspend mode on a USB mouse
AN1180	Using the ST7263 kit to implement a USB game pad
AN1276	BLDC motor start routine for the ST72141 microcontroller
AN1321	Using the ST72141 motor control MCU in Sensor mode
AN1325	Using the ST7 USB low-speed firmware V4.x
AN1445	Emulated 16-bit slave SPI
AN1475	Developing an ST7265X mass storage application
AN1504	Starting a PWM signal directly at high level using the ST7 16-bit timer
AN1602	16-bit timing operations using ST7262 or ST7263B ST7 USB MCUs
AN1633	Device firmware upgrade (DFU) implementation in ST7 non-USB applications
AN1712	Generating a high resolution sinewave using ST7 PWMART
AN1713	SMBus slave driver for ST7 I^{2}C peripherals
AN1753	Software UART using 12-bit ART
AN1947	ST7MC PMAC sine wave motor control software library
General purpose	
AN1476	Low cost power supply for home appliances
AN1526	ST7FLITE0 quick reference note
AN1709	EMC design for ST microcontrollers
AN1752	ST72324 quick reference note
Product evaluation	
AN 910	Performance benchmarking
AN 990	ST7 benefits vs industry standard
AN1077	Overview of enhanced CAN controllers for ST7 and ST9 MCUs
AN1086	U435 can-do solutions for car multiplexing

Table 96. ST7 application notes (continued)

Identification	Description
AN1103	Improved B-EMF detection for low speed, low voltage with ST72141
AN1150	Benchmark ST72 vs PC16
AN1151	Performance comparison between ST72254 & PC16F876
AN1278	LIN (Local Interconnect Network) solutions
Product migration	
AN1131	Migrating applications from ST72511/311/214/124 to ST72521/321/324
AN1322	Migrating an application from ST7263 Rev.B to ST7263B
AN1365	Guidelines for migrating ST72C254 applications to ST72F264
AN1604	How to use ST7MDT1-TRAIN with ST72F264
AN2200	Guidelines for migrating ST7LITE1x applications to ST7FLITE1xB
Product optimization	
AN 982	Using ST7 with ceramic resonator
AN1014	How to minimize the ST7 power consumption
AN1015	Software techniques for improving microcontroller EMC performance
AN1040	Monitoring the Vbus signal for USB self-powered devices
AN1070	ST7 checksum self-checking capability
AN1181	Electrostatic discharge sensitive measurement
AN1324	Calibrating the RC oscillator of the ST7FLITE0 MCU using the mains
AN1502	Emulated data EEPROM with ST7 HD Flash memory
AN1529	Extending the current & voltage capability on the ST7265 V_{DDF} supply
AN1530	Accurate timebase for low-cost ST7 applications with internal RC oscillator
AN1605	Using an active RC to wake up the ST7LITE0 from power saving mode
AN1636	Understanding and minimizing ADC conversion errors
AN1828	PIR (passive infrared) detector using the ST7FLITE05/09/SUPERLITE
AN1946	Sensorless BLDC motor control and BEMF sampling methods with ST7MC
AN1953	PFC for ST7MC starter kit
AN1971	ST7LITE0 microcontrolled ballast
Programming and tools	
AN 978	ST7 Visual DeVELOP software key debugging features
AN 983	Key features of the Cosmic ST7 C-compiler package
AN 985	Executing code In ST7 RAM
AN 986	Using the indirect addressing mode with ST7
AN 987	ST7 serial test controller programming
AN 988	Starting with ST7 assembly tool chain

Table 96. ST7 application notes (continued)

Identification	Description
AN1039	ST7 math utility routines
AN1071	Half duplex USB-to-serial bridge using the ST72611 USB microcontroller
AN1106	Translating assembly code from HC05 to ST7
AN1179	Programming ST7 Flash microcontrollers in remote ISP mode (In-situ programming)
AN1446	Using the ST72521 emulator to debug an ST72324 target application
AN1477	Emulated data EEPROM with XFlash memory
AN1527	Developing a USB smartcard reader with ST7SCR
AN1575	On-board programming methods for XFlash and HD Flash ST7 MCUs
AN1576	In-application programming (IAP) drivers for ST7 HD Flash or XFlash MCUs
AN1577	Device firmware upgrade (DFU) Implementation for ST7 USB applications
AN1601	Software implementation for ST7DALI-EVAL
AN1603	Using the ST7 USB device firmware upgrade development kit (DFU-DK)
AN1635	ST7 customer ROM code release information
AN1754	Data logging program for testing ST7 applications via ICC
AN1796	Field updates for Flash memory based ST7 applications using a PC comm port
AN1900	Hardware implementation for ST7DALI-EVAL
AN1904	ST7MC three-phase AC induction motor control software library
AN1905	ST7MC three-phase BLDC motor control software library
System optimization	
AN1711	Software techniques for compensating ST7 ADC errors
AN1827	Implementation of SIGMA-DELTA ADC with ST7FLITE05/09
AN2009	PWM management for 3-phase BLDC motor drives using the ST7FMC
AN2030	Back EMF detection during PWM on time by ST7MC

14 Package mechanical data

In order to meet environmental requirements, ST offers these devices in ECOPACK® packages. These packages have a lead-free second level interconnect. The category of second Level Interconnect is marked on the package and on the inner box label, in compliance with JEDEC standard JESD97. The maximum ratings related to soldering conditions are also marked on the inner box label.

ECOPACK is an ST trademark. ECOPACK specifications are available at: www.st.com.

Figure 94. 20-pin plastic small outline package, 300-mil width, package outline

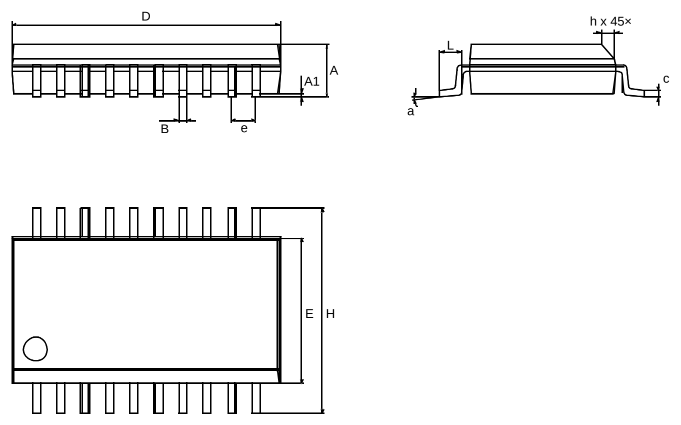


Table 97. 20-pin plastic small outline package, 300-mil width, mechanical data

Dim.	mm			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A	2.35		2.65	0.0925		0.1043
A1	0.10		0.30	0.0039		0.0118
B	0.33		0.51	0.0130		0.0201
C	0.23		0.32	0.0091		0.0126
D	12.60		13.00	0.4961		0.5118
E	7.40		7.60	0.2913		0.2992
e		1.27			0.0500	
H	10.00		10.65	0.3937		0.4193
h	0.25		0.75	0.0098		0.0295
α	0°		8°	0°		8°
L	0.40		1.27	0.0157		0.0500
	Number of Pins					
N	20					

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

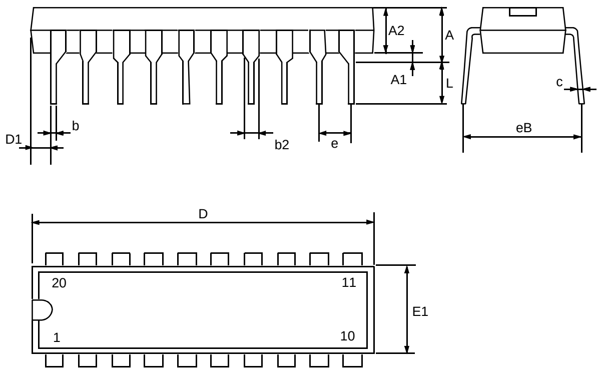


Figure 95. 20-pin plastic dual in-line package, 300-mil width, package outline

Table 98. 20-pin plastic dual in-line package, 300-mil width, mechanical data

Dim.	mm			inches[(1)]		
	Min	Typ	Max	Min	Typ	Max
A			5.33			0.2098
A1	0.38			0.0150		
A2	2.92	3.30	4.95	0.1150	0.1299	0.1949
b	0.36	0.46	0.56	0.0142	0.0181	0.0220
b2	1.14	1.52	1.78	0.0449	0.0598	0.0701
c	0.20	0.25	0.36	0.0079	0.0098	0.0142
D	24.89	26.16	26.92	0.9799	1.0299	1.0598
D1	0.13			0.0051		
e		2.54			0.1000	
eB			10.92			0.4299
E1	6.10	6.35	7.11	0.2402	0.2500	0.2799
L	2.92	3.30	3.81	0.1150	0.1299	0.1500
	Number of Pins					
N	20					

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

Figure 96. 32-pin plastic dual in-line package, shrink 400-mil width, package outline

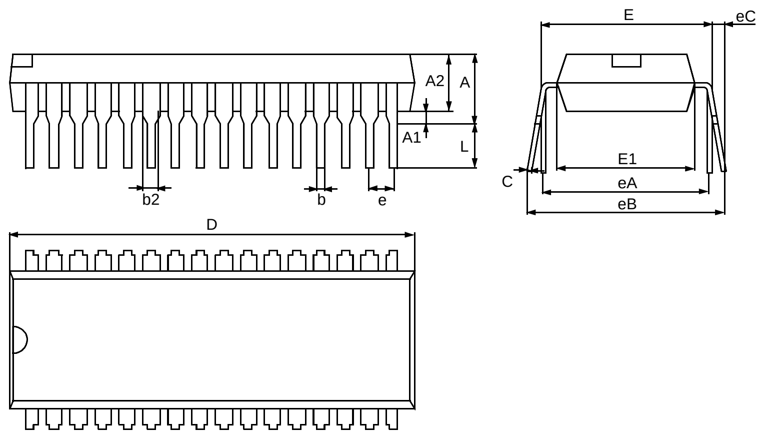


Table 99. 32-pin plastic dual in-line package, shrink 400-mil width, mechanical data

Dim.	mm			inches[(1)]		
	Min	Typ	Max	Min	Typ	Max
A	3.56	3.76	5.08	0.1402	0.1480	0.2000
A1	0.51			0.0201		
A2	3.05	3.56	4.57	0.1201	0.1402	0.1799
b	0.36	0.46	0.58	0.0142	0.0181	0.0228
b1	0.76	1.02	1.40	0.0299	0.0402	0.0551
C	0.20	0.25	0.36	0.0079	0.0098	0.0142
D	27.43		28.45	1.0799		1.1201
E	9.91	10.41	11.05	0.3902	0.4098	0.4350
E1	7.62	8.89	9.40	0.3000	0.3500	0.3701
e		1.78			0.0701	
eA		10.16			0.4000	
eB			12.70			0.5000
eC			1.40			0.0551
L	2.54	3.05	3.81	0.1000	0.1201	0.1500
	Number of pins					
N	32					

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

Figure 97. 32-pin low profile quad flat package (7x7), package outline

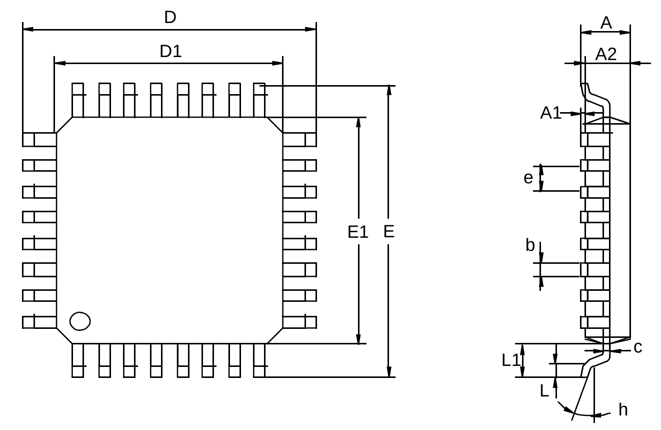


Table 100. 32-pin low profile quad flat package (7x7), package mechanical data

Dim.	mm			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A			1.60			0.0630
A1	0.05		0.15	0.0020		0.0059
A2	1.35	1.40	1.45	0.0531	0.0551	0.0571
b	0.30	0.37	0.45	0.0118	0.0146	0.0177
C	0.09		0.20	0.0035		0.0079
D		9.00			0.3543	
D1		7.00			0.2756	
E		9.00			0.3543	
E1		7.00			0.2756	
e		0.80			0.0315	
θ	0°	3.5°	7°	0°	3.5°	7°
L	0.45	0.60	0.75	0.0177	0.0236	0.0295
L1		1.00			0.0394	
	Number of pins					
N	32					

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

14.1 Thermal characteristics

Table 101. Thermal characteristics

Symbol	Ratings		Value	Unit
R_{thJA}	Package thermal resistance (junction to ambient)	LQFP32 SDIP32 SO20 DIP20	55 58 76 63	°C/W
T_{Jmax}	Maximum junction temperature(1)		150	°C
P_{Dmax}	Power dissipation(2)		160	mW

1. The maximum chip-junction temperature is based on technology characteristics.
2. 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.

15 Revision history

Table 102. Document revision history

Date	Revision	Changes
15-Oct-2007	1	Initial release
23-Oct-2007	2	Interrupt mapping modified (*Table 17 on page 58* and *Table 18 on page 59*) *Section 12.8: EMC (electromagnetic compatibility) characteristics on page 201* modified
23-Nov-2007	3	Added LVD function Modified *Figure 5: ST7FOXF1/ST7FOXK1/ST7FOXK2 memory map on page 20* Modified reset configuration for ICCCLK pin (*Table 2: Device pin description (32-pin packages) on page 16* Modified *Table 3 on page 19* (added AIN5 on pin n°17) Modified information on eix in *Figure 4: 20-pin SO and DIP package pinout on page 18* and *Table 3 on page 19* Added BREAK2 (*Table 2: Device pin description (32-pin packages) on page 16* and *Section 10.2: Dual 12-bit autoreload timer on page 82*) Added PB5 in *Table 3 on page 19* Modified note on SPI in *Table 4: Hardware register map on page 21* Modified *Figure 44.: Block diagram of input capture mode on page 90* Modified *Figure 46.: Long range input capture block diagram on page 91* Modified note 4 in *Section 4.4: ICC interface on page 25* Modified *Figure 6: Typical ICC Interface on page 26* Modified *Figure 52: Lite timer 2 block diagram on page 111* Removed bits 2:0 in LTCSR1 register in *: Lite Timer Control/status register (LTCSR1) on page 115* Modified *Figure 12: Clock management block diagram on page 38* Added RCC_CSR in *Table 4: Hardware register map on page 21* Added Customized RC calibration section Modified *Section 10.2.3: Functional description on page 84* (32 MHz references removed from PWM frequency and one pulse mode paragraphs) Modified *Table 73: On-chip peripheral characteristics on page 193* Modified *Section 10.7.1: Introduction on page 173* Modified *Section 13.3: Development tools on page 215* Modified *Section 13.2: Device ordering information on page 214*
07-Feb-08	4	Added DIP20 package Added LVD in *Figure 1: General block diagram on page 14* Modified *Figure 93: ST7FOXF1/ST7FOXK1/ST7FOXK2 ordering information scheme on page 214* Modified *Table 95: Development tool order codes on page 216* Modified *Table 101: Thermal characteristics on page 224*

Компания «Life Electronics» занимается поставками электронных компонентов импортного и отечественного производства от производителей и со складов крупных дистрибьюторов Европы, Америки и Азии.

С конца 2013 года компания активно расширяет линейку поставок компонентов по направлению коаксиальный кабель, кварцевые генераторы и конденсаторы (керамические, пленочные, электролитические), за счёт заключения дистрибьюторских договоров

Мы предлагаем:

- *Конкурентоспособные цены и скидки постоянным клиентам.*
- *Специальные условия для постоянных клиентов.*
- *Подбор аналогов.*
- *Поставку компонентов в любых объемах, удовлетворяющих вашим потребностям.*

- *Приемлемые сроки поставки, возможна ускоренная поставка.*
- *Доставку товара в любую точку России и стран СНГ.*
- *Комплексную поставку.*
- *Работу по проектам и поставку образцов.*
- *Формирование склада под заказчика.*

- *Сертификаты соответствия на поставляемую продукцию (по желанию клиента).*
- *Тестирование поставляемой продукции.*
- *Поставку компонентов, требующих военную и космическую приемку.*
- *Входной контроль качества.*
- *Наличие сертификата ISO.*

В составе нашей компании организован Конструкторский отдел, призванный помогать разработчикам, и инженерам.

Конструкторский отдел помогает осуществить:

- *Регистрацию проекта у производителя компонентов.*
- *Техническую поддержку проекта.*
- *Защиту от снятия компонента с производства.*
- *Оценку стоимости проекта по компонентам.*
- *Изготовление тестовой платы монтаж и пусконаладочные работы.*

Тел: +7 (812) 336 43 04 (многоканальный)
Email: org@lifeelectronics.ru

www.lifeelectronics.ru