



## ST7DALIF2

8-bit MCU family with single voltage Flash memory,  
data EEPROM, ADC, timers, SPI, DALI

### Features

#### ■ Memories

- 8 Kbytes single voltage Flash Program memory with readout protection, In-Circuit Programming and In-Application programming (ICP and IAP). 10K write/erase cycles guaranteed, data retention: 20 years at 55°C.
- 384 bytes RAM
- 256 bytes data EEPROM with readout protection. 300K write/erase cycles guaranteed, data retention: 20 yrs at 55°C.

#### ■ Clock, reset and supply management

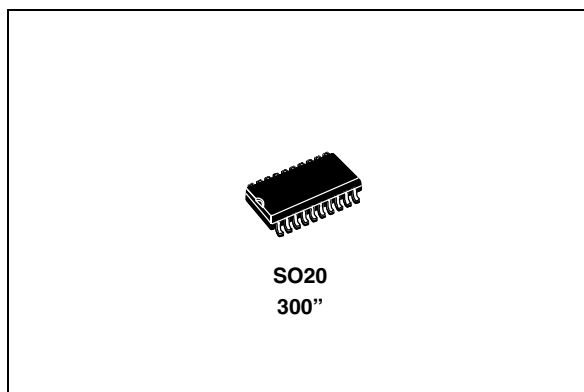
- Enhanced reset system
- Enhanced low voltage supervisor (LVD) for main supply and an auxiliary voltage detector (AVD) with interrupt capability for implementing safe power-down procedures
- Clock sources: Internal 1% RC oscillator, crystal/ceramic resonator or external clock
- Internal 32 MHz input clock for Auto-reload timer
- Optional x4 or x8 PLL for 4 or 8 MHz internal clock
- 5 power saving modes: Halt, Active-halt, Wait and Slow, Auto Wake Up From Halt

#### ■ I/O ports

- Up to 15 multifunctional bidirectional I/Os
- 7 high sink outputs

#### ■ 4 timers

- Configurable watchdog timer
- Two 8-bit Lite timers with prescaler, watchdog, 1 real-time base and 1 input capture
- 12-bit auto-reload timer with 4 PWM outputs, input capture and output compare functions



#### ■ 2 communication interfaces

- SPI synchronous serial interface
- DALI communication interface

#### ■ Interrupt management

- 10 interrupt vectors plus TRAP and RESET
- 15 external interrupt lines (on 4 vectors)

#### ■ A/D converter

- 7 input channels
- Fixed gain op-amp
- 13-bit resolution for 0 to 430 mV (@ 5 V  $V_{DD}$ )
- 10-bit resolution for 430 mV to 5 V (@ 5 V  $V_{DD}$ )

#### ■ 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 hardware/software development package
- DM (Debug module)

# Contents

<b>1</b>	<b>Description</b> .....	<b>10</b>
<b>2</b>	<b>Device summary</b> .....	<b>11</b>
<b>3</b>	<b>Block diagram</b> .....	<b>12</b>
<b>4</b>	<b>Pin description</b> .....	<b>13</b>
<b>5</b>	<b>Register and memory map</b> .....	<b>16</b>
<b>6</b>	<b>Flash program memory</b> .....	<b>19</b>
6.1	Introduction .....	19
6.2	Main features .....	19
6.3	Programming modes .....	19
6.3.1	In-circuit programming (ICP) .....	19
6.3.2	In application programming (IAP) .....	20
6.4	ICC interface .....	20
6.5	Memory protection .....	21
6.5.1	Readout protection .....	21
6.5.2	Flash write/erase protection .....	21
6.6	Related documentation .....	22
6.7	Register description .....	22
6.7.1	Flash control/status register (FCSR) .....	22
<b>7</b>	<b>Data EEPROM</b> .....	<b>23</b>
7.1	Introduction .....	23
7.2	Main features .....	23
7.3	Memory access .....	23
7.4	Power saving modes .....	25
7.5	Access error handling .....	25
7.6	Data EEPROM readout protection .....	26
7.7	Register description .....	27
7.8	EEPROM control/status register (EECSR) .....	27

<b>8</b>	<b>Central processing unit (CPU)</b>	<b>28</b>
8.1	Introduction	28
8.2	Main features	28
8.3	CPU registers	28
8.3.1	Accumulator (A)	29
8.3.2	Index registers (X and Y)	29
8.3.3	Program counter (PC)	29
8.3.4	Condition code register (CC)	29
8.3.5	Stack pointer register (SP)	31
<b>9</b>	<b>Supply, reset and clock management</b>	<b>32</b>
9.1	Main features	32
9.2	Internal RC oscillator adjustment	32
9.3	Phase locked loop	33
9.4	Register description	34
9.4.1	Main clock control/status register (MCCSR)	34
9.4.2	RC control register (RCCR)	35
9.5	Multi-oscillator (MO)	36
9.6	Reset sequence manager (RSM)	38
9.6.1	Introduction	38
9.6.2	Asynchronous external RESET pin	39
9.6.3	External power-on RESET	39
9.6.4	Internal low voltage detector (LVD) RESET	39
9.6.5	Internal watchdog RESET	40
9.7	System integrity management (SI)	40
9.7.1	Low voltage detector (LVD)	40
9.7.2	Auxiliary voltage detector (AVD)	42
9.7.3	Low power modes	43
9.7.4	Register description	44
<b>10</b>	<b>Interrupts</b>	<b>45</b>
10.1	Non maskable software interrupt	45
10.2	External interrupts	45
10.3	Peripheral interrupts	46
10.4	Interrupt registers	48

10.4.1	External interrupt control register (EICR) .....	48
10.4.2	External interrupt selection register (EISR) .....	48
<b>11</b>	<b>Power saving modes .....</b>	<b>50</b>
11.1	Introduction .....	50
11.2	Slow mode .....	50
11.3	Wait mode .....	51
11.4	Halt mode .....	52
11.4.1	Halt mode recommendations .....	54
11.5	Active-halt mode .....	54
11.6	Auto wakeup from Halt mode .....	56
11.6.1	Register description .....	58
<b>12</b>	<b>I/O ports .....</b>	<b>61</b>
12.1	Introduction .....	61
12.2	Functional description .....	61
12.2.1	Input modes .....	61
12.2.2	Output modes .....	62
12.2.3	Alternate functions .....	63
12.3	I/O port implementation .....	66
12.4	Unused I/O pins .....	66
12.5	Low power modes .....	66
12.6	Interrupts .....	66
12.7	Device-specific I/O port configuration .....	67
<b>13</b>	<b>Watchdog timer (WDG) .....</b>	<b>69</b>
13.1	Introduction .....	69
13.2	Main features .....	69
13.3	Functional description .....	69
13.4	Hardware watchdog option .....	70
13.4.1	Using Halt mode with the WDG (WDGHALT option) .....	70
13.5	Interrupts .....	70
13.6	Register description .....	71
13.6.1	Control register (CR) .....	71

<b>14</b>	<b>12-bit autoreload timer 2 (AT2)</b>	<b>72</b>
14.1	Introduction	72
14.2	Main features	72
14.3	Functional description	73
14.3.1	PWM mode	73
14.3.2	Output compare mode	75
14.3.3	Break function	75
14.3.4	Input capture	76
14.4	Low power modes	77
14.5	Interrupts	77
14.6	Register description	77
14.6.1	Timer control status register (ATCSR)	77
14.6.2	Counter register high (CNTRH)	78
14.6.3	Counter register low (CNTRL)	78
14.6.4	Autoreload register (ATRH)	79
14.6.5	Autoreload register (ATRL)	79
14.6.6	PWM output control register (PWMCR)	79
14.6.7	PWMx control status register (PWMxCSR)	79
14.6.8	Break control register (BREAKCR)	80
14.6.9	PWMx duty cycle register high (DCRxH)	80
14.6.10	PWMx duty cycle register low (DCRxL)	81
14.6.11	Input capture register high (ATICRH)	81
14.6.12	Input capture register low (ATICRL)	81
14.6.13	Transfer control register (TRANCR)	82
<b>15</b>	<b>Lite timer 2 (LT2)</b>	<b>84</b>
15.1	Introduction	84
15.2	Main features	84
15.3	Functional description	85
15.3.1	Timebase counter 1	85
15.3.2	Input capture	85
15.3.3	Timebase counter 2	85
15.4	Low power modes	86
15.5	Interrupts	86
15.6	Register description	86
15.6.1	Lite timer control/status register 2 (LTCSR2)	86

15.6.2	Lite timer autoreload register (LTARR) .....	87
15.6.3	Lite timer counter 2 (LTCNTR) .....	87
15.6.4	Lite timer control/status register (LTCSR1) .....	87
15.6.5	Lite timer input capture register (LTICR) .....	88
<b>16</b>	<b>DALI communication module .....</b>	<b>90</b>
16.1	Introduction .....	90
16.2	Main features .....	90
16.3	DALI standard protocol .....	91
16.4	General description .....	92
16.5	Functional description .....	92
16.6	Special functions .....	93
16.6.1	Forced transmission (test mode) .....	93
16.6.2	Normal transmission .....	93
16.6.3	DCM enable .....	93
16.7	DALI interface failure .....	93
16.8	Low power modes .....	94
16.9	Interrupts .....	94
16.10	Bi-phase bit detection .....	94
16.11	Register description .....	96
16.11.1	DCM data rate control register (DCMCLK) .....	96
16.11.2	DCM forward address register (DCMFA) .....	96
16.11.3	DCM forward data register (DCMFD) .....	97
16.11.4	DCM backward data register (DCMBD) .....	97
16.11.5	DCM control register (DCMCR) .....	97
16.11.6	DCM control/status register (DCMCSR) .....	98
<b>17</b>	<b>Serial peripheral interface (SPI) .....</b>	<b>100</b>
17.1	Introduction .....	100
17.2	Main features .....	100
17.3	General description .....	100
17.4	Functional description .....	101
17.4.1	Slave select management .....	102
17.4.2	Master mode operation .....	103
17.4.3	Slave mode operation .....	104

17.4.4	Clock phase and clock polarity	104
17.4.5	Error flags	105
17.4.6	Single master and multimaster configurations	107
17.5	Low power modes	108
17.5.1	Using the SPI to wake-up the device from Halt mode	108
17.6	Interrupts	109
17.7	Register description	109
17.7.1	Control register (SPICR)	109
17.7.2	Control/status register (SPICSR)	110
17.7.3	Data I/O register (SPIDR)	112
<b>18</b>	<b>10-bit A/D converter (ADC)</b>	<b>113</b>
18.1	Introduction	113
18.2	Main features	113
18.3	Functional description	113
18.3.1	Analog power supply	113
18.3.2	Input voltage amplifier	114
18.3.3	Digital A/D conversion result	114
18.3.4	A/D conversion	115
18.4	Changing the conversion channel	115
18.5	Low power modes	115
18.6	Interrupts	116
18.7	Register description	116
18.7.1	Control/status register (ADCCSR)	116
18.7.2	Data register high (ADCDRH)	117
18.7.3	AMP control/data register low (ADCDRL)	117
<b>19</b>	<b>Instruction set</b>	<b>119</b>
19.1	CPU addressing modes	119
19.1.1	Inherent	120
19.1.2	Immediate	120
19.1.3	Direct	121
19.1.4	Indexed (no offset, short, long)	121
19.1.5	Indirect (short, long)	121
19.1.6	Indirect indexed (short, long)	122
19.1.7	Relative mode (direct, indirect)	123

19.2	Instruction groups	123
<b>20</b>	<b>Electrical characteristics</b>	<b>127</b>
20.1	Parameter conditions	127
20.1.1	Minimum and maximum values	127
20.1.2	Typical values	127
20.1.3	Typical curves	127
20.1.4	Loading capacitor	127
20.1.5	Pin input voltage	127
20.2	Absolute maximum ratings	128
20.3	Operating conditions	130
20.3.1	Operating conditions with low voltage detector (LVD)	131
20.3.2	Auxiliary Voltage Detector (AVD) Thresholds	131
20.3.3	Internal RC oscillator and PLL	131
20.4	Supply current characteristics	135
20.4.1	Supply current	136
20.5	Clock and timing characteristics	138
20.5.1	Crystal and ceramic resonator oscillators	138
20.6	Memory characteristics	139
20.7	EMC characteristics	141
20.7.1	Functional EMS (electromagnetic susceptibility)	141
20.7.2	Electromagnetic interference (EMI)	142
20.7.3	Absolute maximum ratings (electrical sensitivity)	142
20.8	I/O port pin characteristics	143
20.8.1	General characteristics	143
20.9	Control pin characteristics	149
20.10	Communication interface characteristics	151
20.10.1	SPI - serial peripheral interface	151
20.11	10-bit ADC characteristics	154
20.11.1	Amplifier output offset variation	157
<b>21</b>	<b>Package characteristics</b>	<b>158</b>
21.1	Package mechanical data	159
<b>22</b>	<b>Device configuration</b>	<b>161</b>
22.1	Option bytes	161



---

22.1.1	Option byte 0 .....	161
22.1.2	Option byte 1 .....	162
22.2	Device ordering information and transfer of customer code .....	164
<b>23</b>	<b>Important notes .....</b>	<b>166</b>
23.1	Execution of BTJX instruction .....	166
23.2	ADC conversion spurious results .....	166
23.3	A/ D converter accuracy for first conversion .....	166
23.4	Negative injection impact on ADC accuracy .....	166
23.5	Clearing active interrupts outside interrupt routine .....	166
23.6	Using PB4 as external interrupt .....	167
23.7	Timebase 2 interrupt in Slow mode .....	167
<b>24</b>	<b>Revision history .....</b>	<b>168</b>

# 1 Description

The ST7DALIF2 device is a member of the ST7 microcontroller family designed for DALI applications running from 2.4 to 5.5 V. Different package options offer up to 15 I/O pins.

All devices are based on a common industry-standard 8-bit core, featuring an enhanced instruction set and are available with Flash or ROM program memory. The ST7 family architecture offers both power and flexibility to software developers, enabling the design of highly efficient and compact application code.

The on-chip peripherals include a DALI communication interface and an SPI. For power economy, the microcontroller can switch dynamically into, Slow, Wait, Active-halt, Auto Wakeup from Halt or Halt mode when the application is in idle or stand-by state.

Typical applications include consumer, home, office, lighting and industrial products.

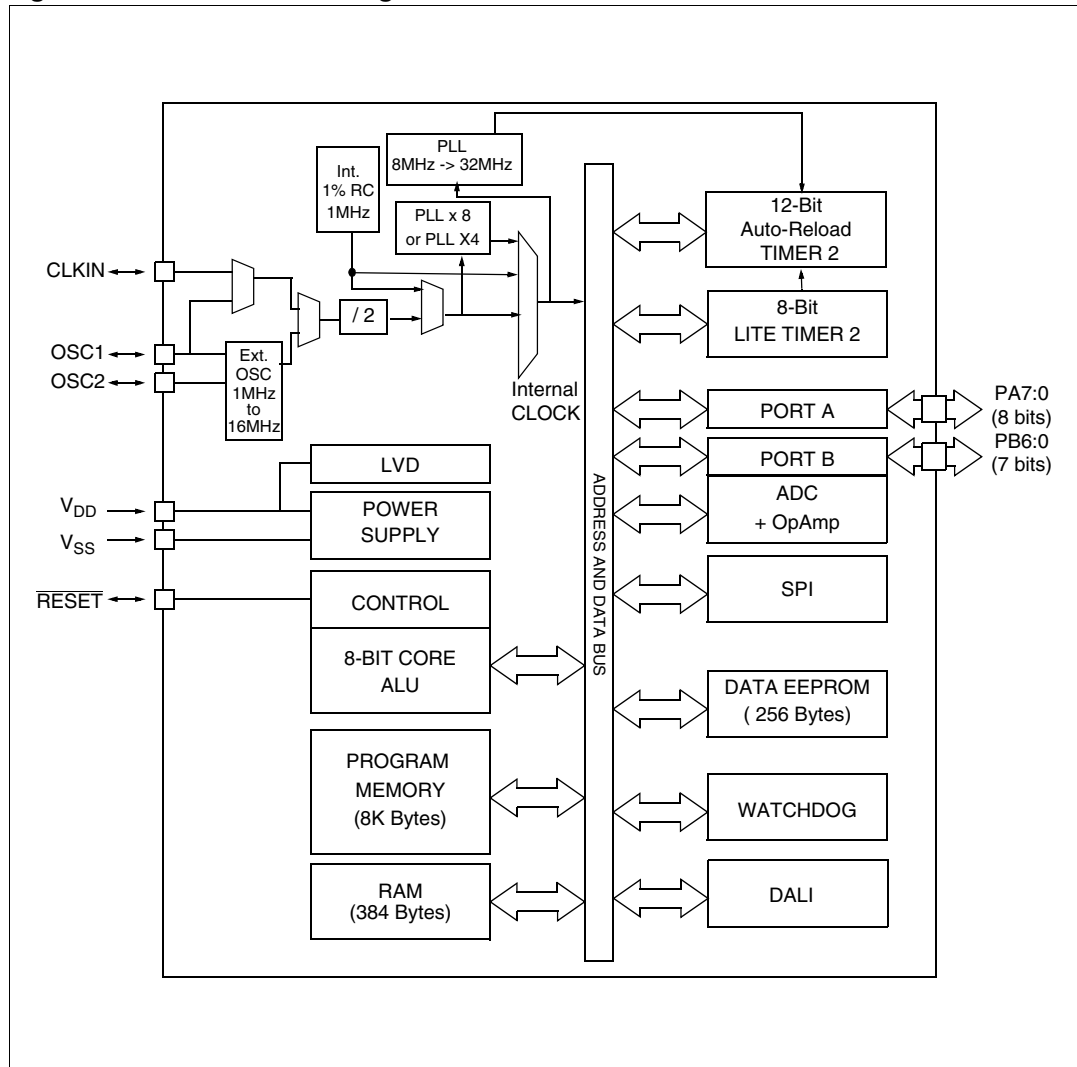
## 2 Device summary

**Table 1. Device summary**

Features	ST7DALIF2
Program memory	8 Kbytes
RAM (stack)	384 (128) bytes
Data EEPROM	256 bytes
Peripherals	Lite Timer with Watchdog, Autoreload Timer with 32 MHz input clock, SPI, 10-bit ADC with Op-Amp, DALI
Operating supply	2.4V to 5.5V
CPU frequency	Up to 8 MHz (with external OSC up to 16 MHz and internal 1 MHz RC 1% PLLx8/4 MHz)
Operating temperature	-40°C to +85°C
Packages	SO20 300"

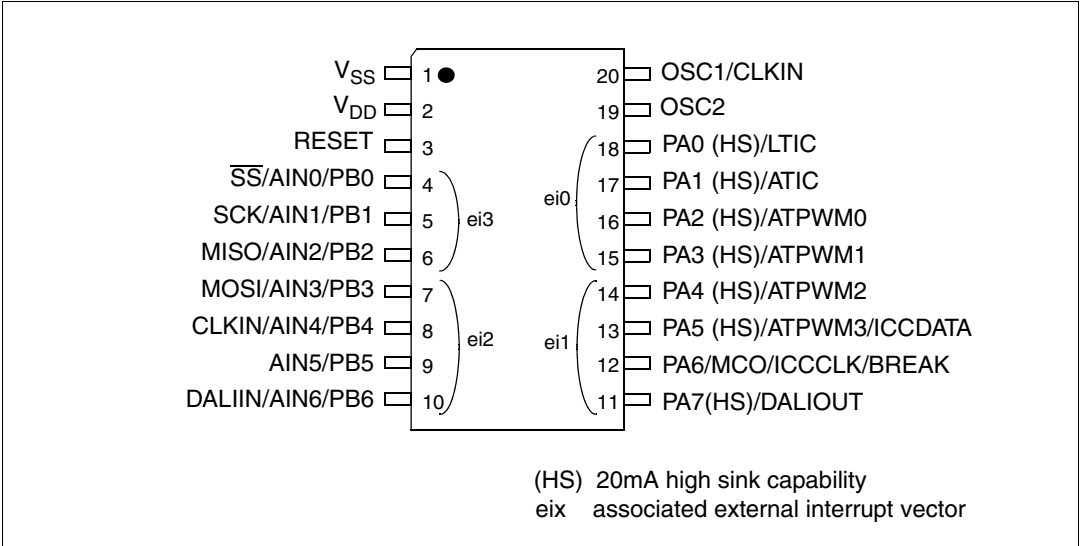
### 3 Block diagram

Figure 1. General block diagram



# 4 Pin description

Figure 2. 20-pin SO package pinout



**Legend / Abbreviations** for [Table 2](#):

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

In/Output level:  $C_T = \text{CMOS } 0.3V_{DD}/0.7V_{DD}$  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

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**

Pin no.	Pin name	Type	Level		Port / control						Main function (after reset)	Alternate function
			Input	Output	Input				Output			
					float	wpu	int	ana	OD	PP		
1	V <sub>SS</sub>	S										Ground
2	V <sub>DD</sub>	S										Main power supply
3	RESET	I/O	C <sub>T</sub>			X			X			Top priority non maskable interrupt (active low)
4	PB0/AIN0/SS	I/O	C <sub>T</sub>	X	ei3		X	X	X	Port B0	ADC Analog Input 0 or SPI Slave Select (active low) <b>Caution:</b> No negative current injection allowed on this pin. For details, refer to <a href="#">Section 20.2 on page 128</a>	
5	PB1/AIN1/SCK	I/O	C <sub>T</sub>	X			X	X	X	Port B1	ADC Analog Input 1 or SPI Serial Clock <b>Caution:</b> No negative current injection allowed on this pin. For details, refer to <a href="#">Section 20.2 on page 128</a>	
6	PB2/AIN2/MISO	I/O	C <sub>T</sub>	X			X	X	X	Port B2	ADC Analog Input 2 or SPI Master In/ Slave Out Data	
7	PB3/AIN3/MOSI	I/O	C <sub>T</sub>	X	ei2		X	X	X	Port B3	ADC Analog Input 3 or SPI Master Out / Slave In Data	
8	PB4/AIN4/CLKIN	I/O	C <sub>T</sub>	X			X	X	X	Port B4	ADC Analog Input 4 or External clock input	
9	PB5/AIN5	I/O	C <sub>T</sub>	X			X	X	X	Port B5	ADC Analog Input 5	
10	PB6/AIN6/DALIIN	I/O	C <sub>T</sub>	X			X	X	X	Port B6	ADC Analog Input 6 or DALI Input	

Table 2. Device pin description (continued)

Pin no.	Pin name	Type	Level		Port / control						Main function (after reset)	Alternate function
			Input	Output	Input				Output			
					float	wpu	int	ana	OD	PP		
11	PA7/DALIOUT	I/O	C <sub>T</sub>	HS	X	ei1			X	X	Port A7	DALI Output
12	PA6 /MCO/ ICCCLK/BREAK	I/O	C <sub>T</sub>		X				X	X	Port A6	Main Clock Output or In Circuit Communication Clock or External BREAK <b>Caution:</b> During normal operation this 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. In the application, even if the pin is configured as output, any reset will put it back in input pull-up.
13	PA5 /ATPWM3/ ICCDATA	I/O	C <sub>T</sub>	HS	X				X	X	Port A5	Auto-Reload Timer PWM3 or In Circuit Communication Data
14	PA4/ATPWM2	I/O	C <sub>T</sub>	HS	X				X	X	Port A4	Auto-Reload Timer PWM2
15	PA3/ATPWM1	I/O	C <sub>T</sub>	HS	X	ei0			X	X	Port A3	Auto-Reload Timer PWM1
16	PA2/ATPWM0	I/O	C <sub>T</sub>	HS	X				X	X	Port A2	Auto-Reload Timer PWM0
17	PA1/ATIC	I/O	C <sub>T</sub>	HS	X				X	X	Port A1	Auto-Reload Timer Input Capture
18	PA0 /LTIC	I/O	C <sub>T</sub>	HS	X				X	X	Port A0	Lite Timer Input Capture
19	OSC2	O									Resonator oscillator inverter output	
20	OSC1/CLKIN	I									Resonator oscillator inverter input or External clock input	

# 5 Register and memory map

As shown in [Figure 3](#), the MCU is capable of addressing 64K bytes of memories and I/O registers.

The available memory locations consist of 128 bytes of register locations, 384 bytes of RAM, 256 bytes of data EEPROM and 8 Kbytes of user program memory. The RAM space includes up to 128 bytes for the stack from 180h to 1FFh.

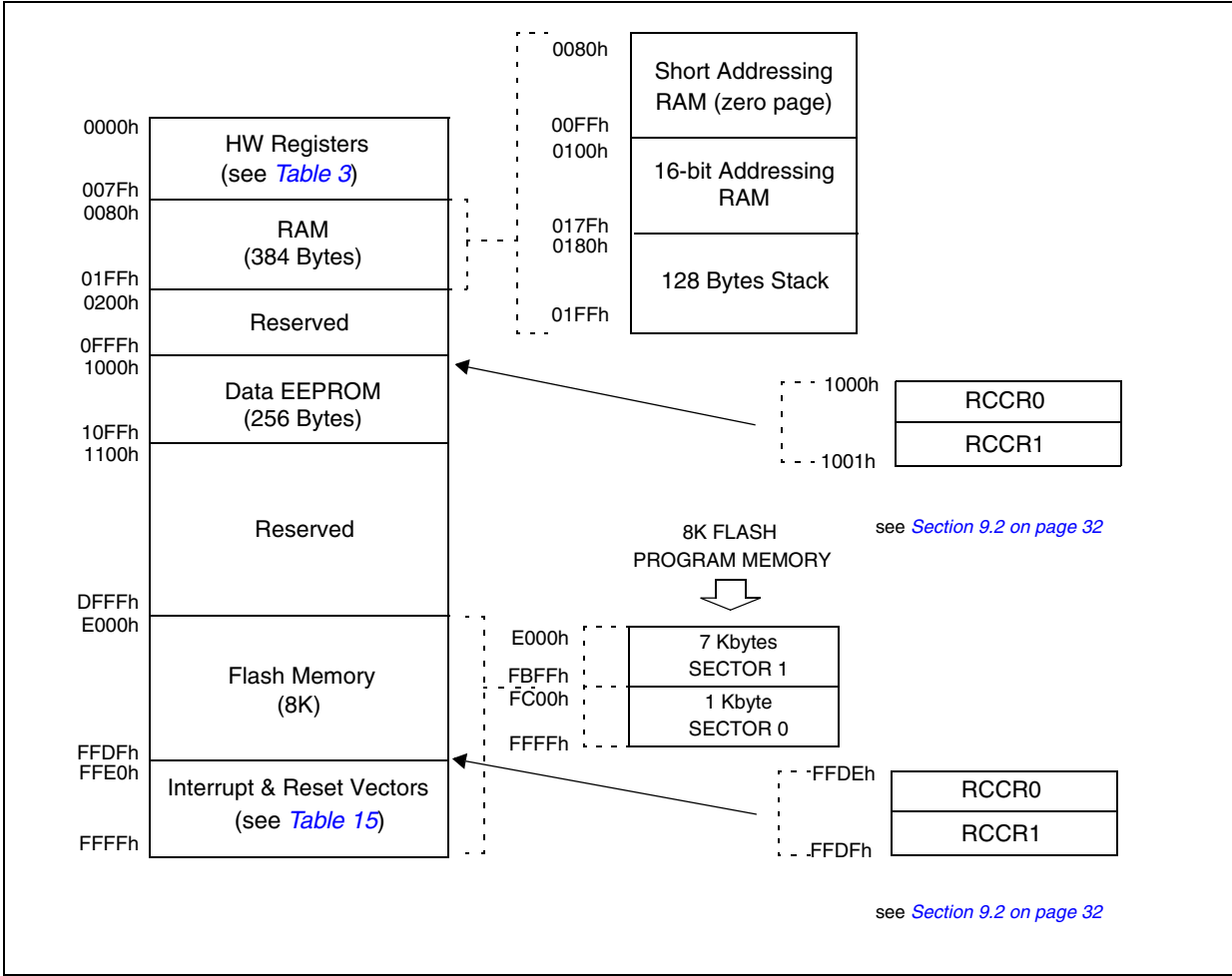
The highest address bytes contain the user reset and interrupt vectors.

The Flash memory contains two sectors (see [Figure 3](#)) mapped in the upper part of the ST7 addressing space so the reset and interrupt vectors are located in Sector 0 (F000h-FFFFh).

The size of Flash Sector 0 and other device options are configurable by Option byte (refer to [Section 22.1 on page 161](#)).

**Note:** **IMPORTANT:** memory locations marked as “Reserved” must never be accessed. Accessing a reserved area can have unpredictable effects on the device.

**Figure 3. Memory map**





**Table 3. Hardware register map**

Address	Block	Register label	Register name	Reset status	Remarks
0000h 0001h 0002h	Port A	PADR	Port A Data Register	FFh <sup>(1)</sup>	R/W
		PADDR	Port A Data Direction Register	00h	R/W
		PAOR	Port A Option Register	40h	R/W
0003h 0004h 0005h	Port B	PBDR	Port B Data Register	FFh <sup>1)</sup>	R/W
		PBDDR	Port B Data Direction Register	00h	R/W
		PBOR	Port B Option Register	00h	R/W <sup>(2)</sup>
0006h 0007h	Reserved Area (2 bytes)				
0008h 0009h 000Ah 000Bh 000Ch	LITE TIMER 2	LTCSR2	Lite Timer Control/Status Register 2	00h	R/W
		LTARR	Lite Timer Auto-reload Register	00h	R/W
		LTCNTR	Lite Timer Counter Register	00h	Read Only
		LTCSR1	Lite Timer Control/Status Register 1	0x00 0000b	R/W
		LTICR	Lite Timer Input Capture Register	00h	Read Only
000Dh 000Eh 000Fh 0010h 0011h 0012h 0013h 0014h 0015h 0016h 0017h 0018h 0019h 001Ah 001Bh 001Ch 001Dh 001Eh 001Fh 0020h 0021h 0022h	AUTO- RELOAD TIMER 2	ATCSR	Timer Control/Status Register	0x00 0000b	R/W
		CNTRH	Counter Register High	00h	Read Only
		CNTRL	Counter Register Low	00h	Read Only
		ATRH	Auto-Reload Register High	00h	R/W
		ATRL	Auto-Reload Register Low	00h	R/W
		PWMCR	PWM Output Control Register	00h	R/W
		PWM0CSR	PWM 0 Control/Status Register	00h	R/W
		PWM1CSR	PWM 1 Control/Status Register	00h	R/W
		PWM2CSR	PWM 2 Control/Status Register	00h	R/W
		PWM3CSR	PWM 3 Control/Status Register	00h	R/W
		DCR0H	PWM 0 Duty Cycle Register High	00h	R/W
		DCR0L	PWM 0 Duty Cycle Register Low	00h	R/W
		DCR1H	PWM 1 Duty Cycle Register High	00h	R/W
		DCR1L	PWM 1 Duty Cycle Register Low	00h	R/W
		DCR2H	PWM 2 Duty Cycle Register High	00h	R/W
		DCR2L	PWM 2 Duty Cycle Register Low	00h	R/W
		DCR3H	PWM 3 Duty Cycle Register High	00h	R/W
		DCR3L	PWM 3 Duty Cycle Register Low	00h	R/W
		ATICRH	Input Capture Register High	00h	Read Only
		ATICRL	Input Capture Register Low	00h	Read Only
		TRANC	Transfer Control Register	01h	R/W
		BREAKCR	Break Control Register	00h	R/W
0023h to 002Dh	Reserved area (11 bytes)				
002Eh	WDG	WDGCR	Watchdog Control Register	7Fh	R/W
0002Fh	FLASH	FCSR	Flash Control/Status Register	00h	R/W
00030h	EEPROM	EECSR	Data EEPROM Control/Status Register	00h	R/W
0031h 0032h 0033h	SPI	SPIDR	SPI Data I/O Register	xxh	R/W
		SPICR	SPI Control Register	0xh	R/W
		SPICSR	SPI Control Status Register	00h	R/W

**Table 3. Hardware register map (continued)**

Address	Block	Register label	Register name	Reset status	Remarks
0034h 0035h 0036h	ADC	ADCCSR ADCDRH ADCDRL	A/D Control Status Register A/D Data Register High A/D Amplifier Control/Data Low Register	00h xxh 0xh	R/W Read Only R/W
0037h	ITC	EICR	External Interrupt Control Register	00h	R/W
0038h	MCC	MCCSR	Main Clock Control/Status Register	00h	R/W
0039h 003Ah	Clock and Reset	RCCR SICSR	RC oscillator Control Register System Integrity Control/Status Register	FFh 0000 0xx0b	R/W R/W
003Bh	Reserved area (1 byte)				
003Ch	ITC	EISR	External Interrupt Selection Register	0Ch	R/W
003Dh to 003Fh	Reserved area (3 bytes)				
0040h 0041h 0042h 0043h 0044h 0045h	DALI	DCMCLK DCMFSA DCMFD DCMBD DCMCR DCMCSR	DALI Clock Register DALI Forward Address Register DALI Forward Data Register DALI Backward Data Register DALI Control Register DALI Control/Status Register	00h 00h 00h 00h 00h 00h	R/W R/W R/W R/W R/W R/W
0046h to 0048h	Reserved area (3 bytes)				
0049h 004Ah	AWU	AWUPR AWUCSR	AWU Prescaler Register AWU Control/Status Register	FFh 00h	R/W R/W
004Bh 004Ch 004Dh 004Eh 004Fh 0050h	DM <sup>(3)</sup>	DMCR DMSR DMBK1H DMBK1L DMBK2H DMBK2L	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	00h 00h 00h 00h 00h 00h	R/W R/W R/W R/W R/W R/W
0051h to 007Fh	Reserved area (47 bytes)				

1. The contents of the I/O port DR registers are readable only in output configuration. In input configuration, the values of the I/O pins are returned instead of the DR register contents.
2. The bits associated with unavailable pins must always keep their reset value.
3. For a description of the Debug Module registers, see ST7 ICC Protocol Reference Manual.

**Legend:** x=undefined, R/W=read/write

## 6 Flash program memory

### 6.1 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.

### 6.2 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
- Readout and write protection

### 6.3 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 and data EEPROM (if present) can be programmed or erased.
- In-Circuit Programming. In this mode, FLASH sectors 0 and 1, option byte row and data EEPROM (if present) can be programmed or erased without removing the device from the application board.
- In-Application Programming. In this mode, sector 1 and data EEPROM (if present) can be programmed or erased without removing the device from the application board and while the application is running.

#### 6.3.1 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 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).

### 6.3.2 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.

## 6.4 ICC interface

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

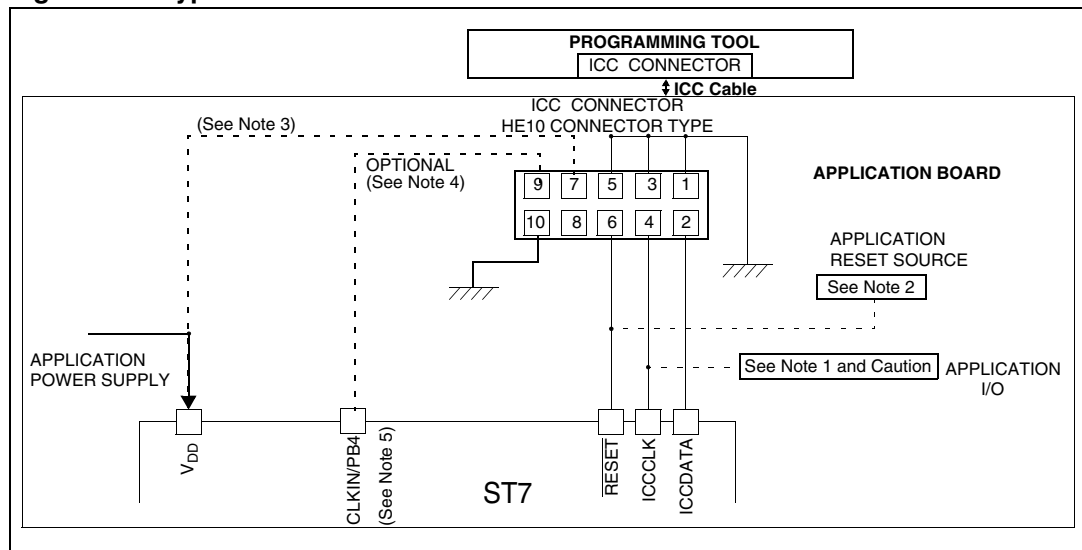
- $\overline{\text{RESET}}$ : device reset
- $V_{SS}$ : device power supply ground
- ICCCLK: ICC output serial clock pin
- ICCDATA: ICC input serial data pin
- CLKIN/PB4: main clock input for external source
- $V_{DD}$ : application board power supply (optional, see Note 3)

- 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 < 1K). A schottky diode can be used to isolate the application RESET circuit in this case. When using a classical RC network with  $R > 1K$  or a reset management IC with open drain output and pull-up resistor > 1K, no additional components are needed. In all cases the user must ensure that no external reset is generated by the application during the ICC session.*
  - 3 *The use of Pin 7 of the ICC connector depends on the Programming Tool architecture. This pin must be connected when using most ST Programming Tools (it is used to monitor the application power supply). Please refer to the Programming Tool manual.*
  - 4 *Pin 9 has to be connected to the CLKIN/PB4 pin of the ST7 when the clock is not available in the application or if the selected clock option is not programmed in the option byte. ST7 devices with multi-oscillator capability need to have OSC1 and OSC2 grounded in this case.*
  - 5 *With any programming tool, while the ICP option is disabled, the external clock has to be provided on PB4.*

**Caution:** 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. In the application, even if the pin is configured as output, any reset will put it back in input pull-up.

**Figure 4. Typical ICC interface**



## 6.5 Memory protection

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

### 6.5.1 Readout protection

Readout 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. Both program and data E<sup>2</sup> memory are protected.

In flash devices, this protection is removed by reprogramming the option. In this case, both program and data E<sup>2</sup> memory are automatically erased and the device can be reprogrammed.

Readout protection selection depends on the device type:

- In Flash devices it is enabled and removed through the FMP\_R bit in the option byte.
- In ROM devices it is enabled by mask option specified in the Option List.

### 6.5.2 Flash write/erase protection

Write/erase protection, when set, makes it impossible to both overwrite and erase program memory. It does not apply to E<sup>2</sup> data. Its purpose is to provide advanced security to applications and prevent any change being made to the memory content.

**Caution:** Once set, Write/erase protection can never be removed. A write-protected flash device is no longer reprogrammable.

Write/erase protection is enabled through the FMP\_W bit in the option byte.

## 6.6 Related documentation

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

## 6.7 Register description

### 6.7.1 Flash control/status register (FCSR)

This register is reserved for programming using ICP, IAP or other programming methods. It controls the XFlash programming and erasing operations.

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

1st RASS Key: 0101 0110 (56h)

2nd RASS Key: 1010 1110 (AEh)

FCSR					Reset value:0000 0000 (00h)		
7	6	5	4	3	2	1	0
0	0	0	0	0	OPT	LAT	PGM
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**Table 4. Flash control/status register address and reset value**

Address (Hex)	Register label	7	6	5	4	3	2	1	0
002Fh	FCSR reset value	0	0	0	0	0	0	0	0

## 7 Data EEPROM

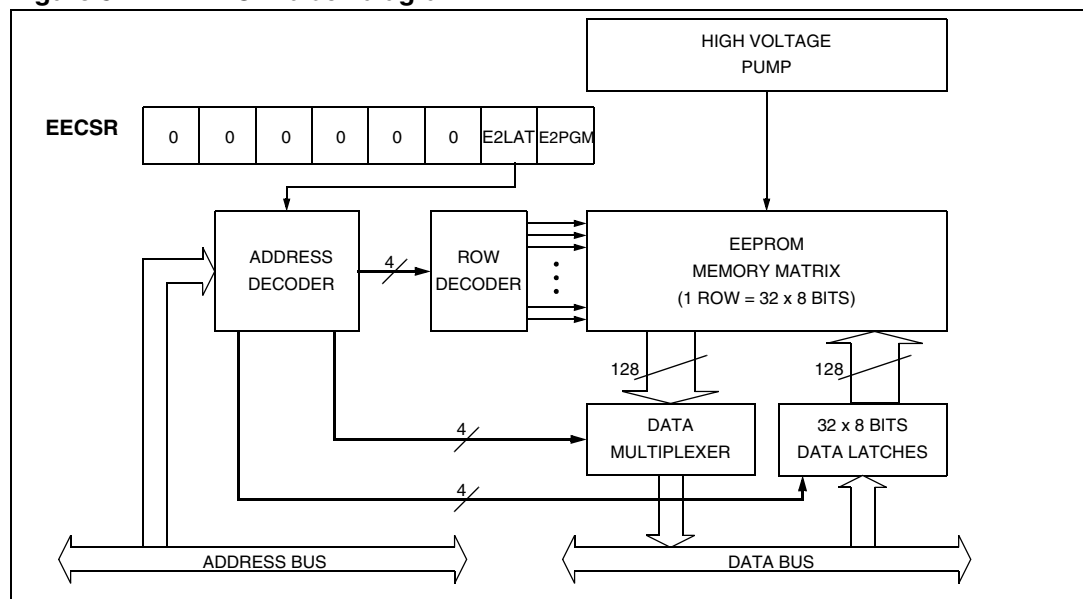
### 7.1 Introduction

The Electrically Erasable Programmable Read Only Memory can be used as a non volatile back-up for storing data. Using the EEPROM requires a basic access protocol described in this chapter.

### 7.2 Main features

- Up to 32 Bytes programmed in the same cycle
- EEPROM mono-voltage (charge pump)
- Chained erase and programming cycles
- Internal control of the global programming cycle duration
- Wait mode management
- Readout protection

**Figure 5. EEPROM block diagram**



### 7.3 Memory access

The Data EEPROM memory read/write access modes are controlled by the E2LAT bit of the EEPROM Control/Status register (EECSR). The flowchart in [Figure 6](#) describes these different memory access modes.

#### Read operation (E2LAT=0)

The EEPROM can be read as a normal ROM location when the E2LAT bit of the EECSR register is cleared.

On this device, Data EEPROM can also be used to execute machine code. Take care not to write to the Data EEPROM while executing from it. This would result in an unexpected code being executed.

#### Write operation (E2LAT=1)

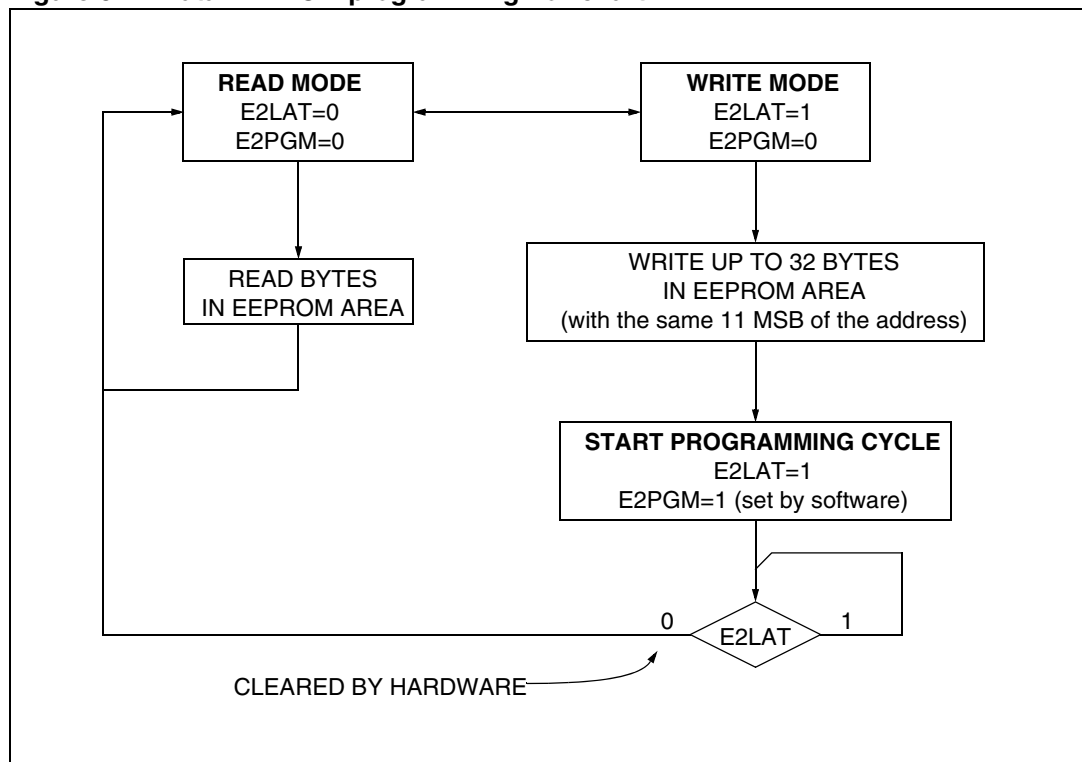
To access the write mode, the E2LAT bit has to be set by software (the E2PGM bit remains cleared). When a write access to the EEPROM area occurs, the value is latched inside the 32 data latches according to its address.

When PGM bit is set by the software, all the previous bytes written in the data latches (up to 32) are programmed in the EEPROM cells. The effective high address (row) is determined by the last EEPROM write sequence. To avoid wrong programming, the user must take care that all the bytes written between two programming sequences have the same high address: only the five Least Significant Bits of the address can change.

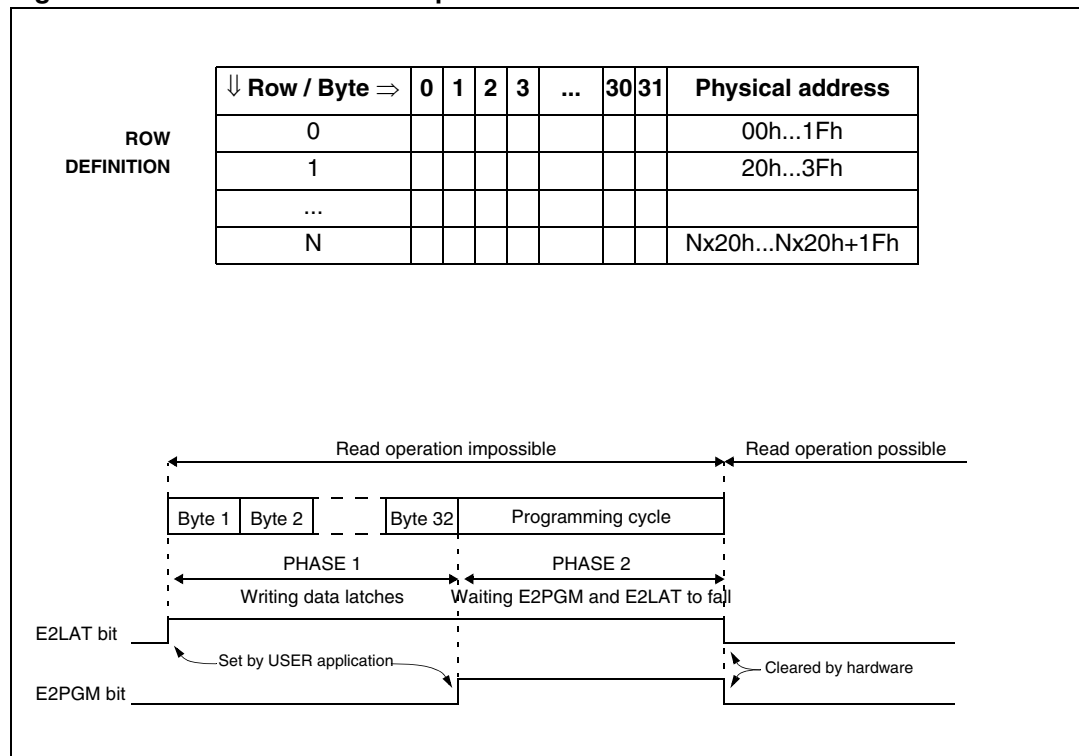
At the end of the programming cycle, the PGM and LAT bits are cleared simultaneously.

*Note: Care should be taken during the programming cycle. Writing to the same memory location will over-program the memory (logical AND between the two write access data result) because the data latches are only cleared at the end of the programming cycle and by the falling edge of the E2LAT bit. It is not possible to read the latched data. This note is illustrated by Figure 8.*

**Figure 6. Data EEPROM programming flowchart**





**Figure 7. Data EEPROM write operation**

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

## 7.4 Power saving modes

### Wait mode

The data EEPROM can enter Wait mode on execution of the WFI instruction of the microcontroller or when the microcontroller enters Active-Halt mode. The data EEPROM will immediately enter this mode if there is no programming in progress, otherwise the data EEPROM will finish the cycle and then enter Wait mode.

### Active-halt mode

Refer to Wait mode.

### Halt mode

The data EEPROM immediately enters Halt mode if the microcontroller executes the HALT instruction. Therefore the EEPROM will stop the function in progress, and data may be corrupted.

## 7.5 Access error handling

If a read access occurs while E2LAT=1, then the data bus will not be driven.

If a write access occurs while E2LAT=0, then the data on the bus will not be latched.

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

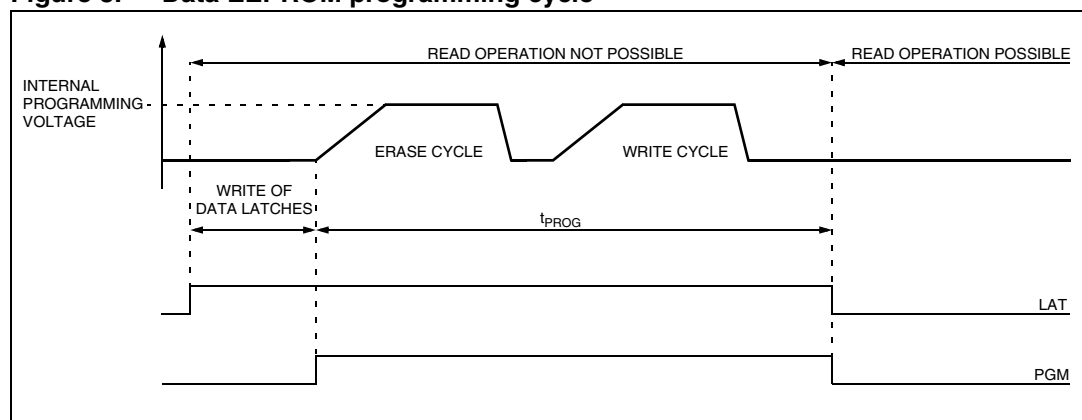
## 7.6 Data EEPROM readout protection

The readout protection is enabled through an option bit (see [Section 22.1 on page 161](#)).

When this option is selected, the programs and data stored in the EEPROM memory are protected against readout (including a re-write protection). In Flash devices, when this protection is removed by reprogramming the Option Byte, the entire Program memory and EEPROM is first automatically erased.

*Note: Both Program Memory and data EEPROM are protected using the same option bit.*

**Figure 8. Data EEPROM programming cycle**



## 7.7 Register description

### 7.8 EEPROM control/status register (EECSR)

Read/Write

Reset Value: 0000 0000 (00h)

7						0	
0	0	0	0	0	0	E2LAT	E2PGM

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

Bit 1 = **E2LAT** *Latch Access Transfer*

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 E2PGM bit is cleared.

0: Read mode

1: Write mode

Bit 0 = **E2PGM** *Programming control and status*

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 E2PGM bit is cleared during the programming cycle, the memory data is not guaranteed*

**Table 5. Data EEPROM register map and reset values**

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0030h	EECSR Reset Value	0	0	0	0	0	0	E2LAT 0	E2PGM 0

# 8 Central processing unit (CPU)

## 8.1 Introduction

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

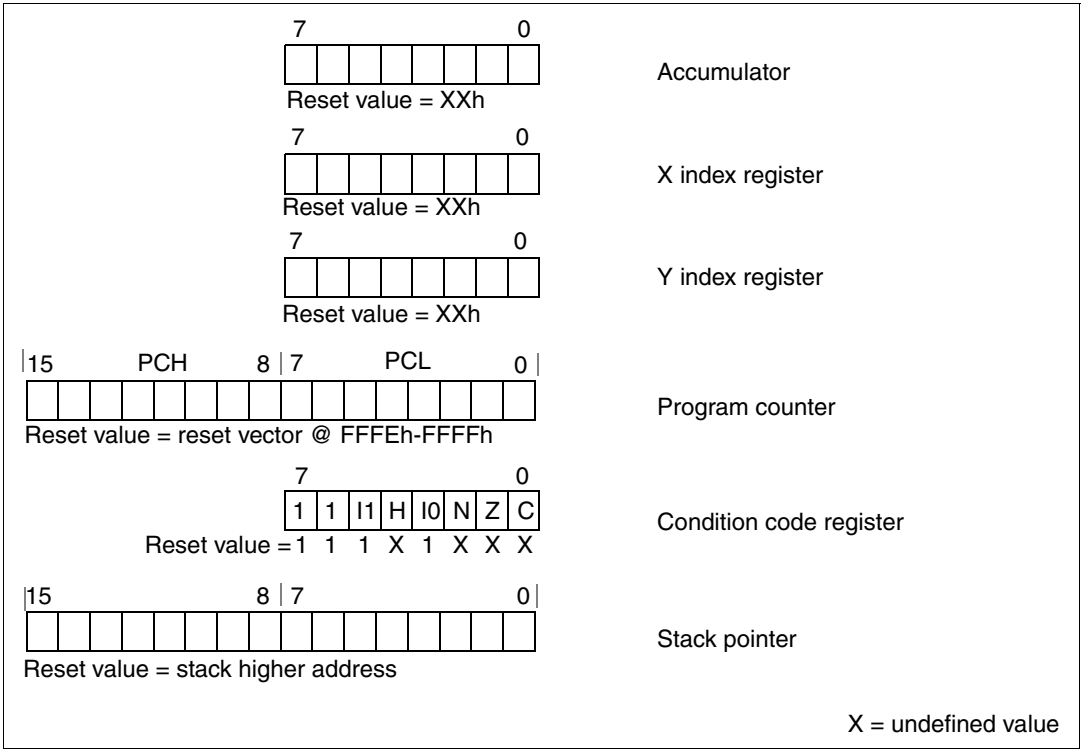
## 8.2 Main features

- Enable executing 63 basic instructions
- Fast 8-bit by 8-bit multiply
- 17 main addressing modes (with indirect addressing mode)
- Two 8-bit index registers
- 16-bit stack pointer
- Low power Halt and Wait modes
- Priority maskable hardware interrupts
- Non-maskable software/hardware interrupts

## 8.3 CPU registers

The six CPU registers shown in [Figure 9](#) are not present in the memory mapping and are accessed by specific instructions.

**Figure 9. CPU registers**



### 8.3.1 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.

### 8.3.2 Index registers (X and Y)

These 8-bit registers are used to create effective addresses or as 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.

### 8.3.3 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).

### 8.3.4 Condition code register (CC)

The 8-bit Condition Code register contains the interrupt masks and four flags representative of the result of the instruction just executed. This register can also be handled by the PUSH and POP instructions. These bits can be individually tested and/or controlled by specific instructions.

CC				Reset value: 111x1xxx			
7	6	5	4	3	2	1	0
1	1	I1	H	I0	N	Z	C
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**Table 6. Arithmetic management bits**

Bit	Name	Function
4	H	<p>Half carry</p> <p>This bit is set by hardware when a carry occurs between bits 3 and 4 of the ALU during an ADD or ADC instructions. It is reset by hardware during the same instructions.</p> <p>0: No half carry has occurred. 1: A half carry has occurred.</p> <p>This bit is tested using the JRH or JRNH instruction. The H bit is useful in BCD arithmetic subroutines.</p>
2	N	<p>Negative</p> <p>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 result 7th bit.</p> <p>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).</p> <p>This bit is accessed by the JRMI and JRPL instructions.</p>

**Table 6. Arithmetic management bits (continued)**

Bit	Name	Function
1	Z	Zero (Arithmetic Management 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.
0	C	Carry/borrow This bit is set and cleared by hardware and software. It indicates an overflow or an underflow has occurred during the last arithmetic operation. 0: No overflow or underflow has occurred. 1: An overflow or underflow has occurred. This bit is driven by the SCF and RCF instructions and tested by the JRC and JRNC instructions. It is also affected by the 'bit test and branch', shift and rotate instructions.

**Table 7. Software interrupt bits**

Bit	Name	Function
5	I1	Software Interrupt Priority 1 The combination of the I1 and I0 bits determines the current interrupt software priority (see <a href="#">Table 8</a> ).
3	I0	Software Interrupt Priority 0 The combination of the I1 and I0 bits determines the current interrupt software priority (see <a href="#">Table 8</a> ).

**Table 8. Interrupt software priority selection**

Interrupt software priority	Level	I1	I0
Level 0 (main)	<div style="text-align: center;"> Low ↓ High </div>	1	0
Level 1		0	1
Level 2		0	0
Level 3 (= interrupt disable)		1	1

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

See [Section 10: Interrupts on page 45](#) for more details.

### 8.3.5 Stack pointer register (SP)

SP																Reset value: 01 FFh	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	1	1	SP6	SP5	SP4	SP3	SP2	SP1	SP0		
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	R/W		

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 10](#)).

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 SP7 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 an LD instruction.

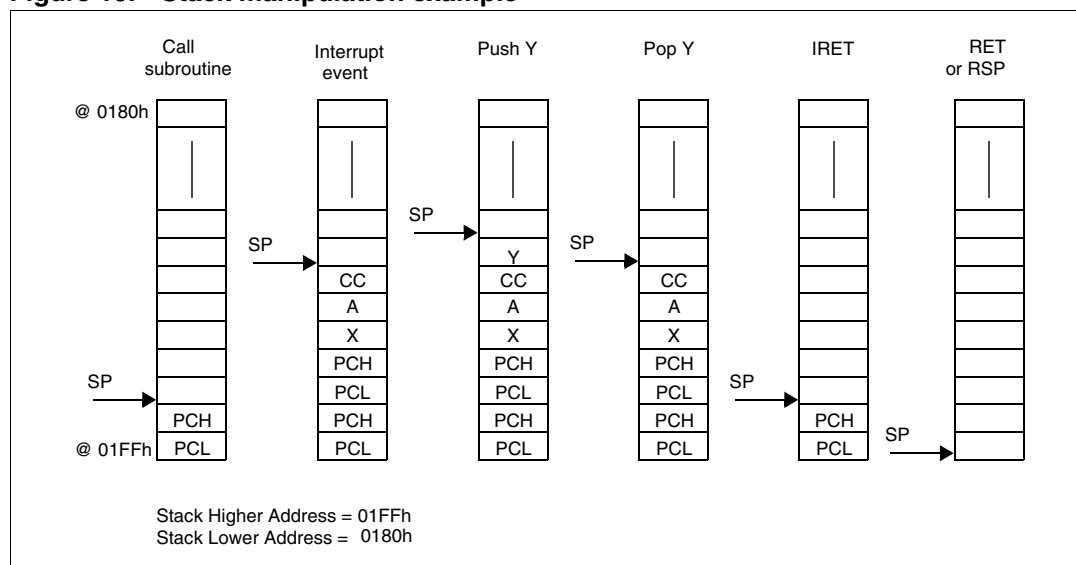
**Note:** *When the lower limit is exceeded, the Stack Pointer wraps around to the stack upper limit, without indicating the stack overflow. The previously stored information is then overwritten and therefore lost. The stack also wraps in case of an underflow.*

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 10](#).

- 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 10. Stack manipulation example**



## 9 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.

### 9.1 Main features

- Clock management
  - 1 MHz internal 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)
  - PLL for multiplying the frequency by 8 or 4 (enabled by option byte)
  - For clock ART counter only: PLL32 for multiplying the 8 MHz frequency by 4 (enabled by option byte). The 8 MHz input frequency is mandatory and can be obtained in the following ways:
    - 1 MHz RC + PLLx8
    - 16 MHz external clock (internally divided by 2)
    - 2 MHz. external clock (internally divided by 2) + PLLx8
    - Crystal oscillator with 16 MHz output frequency (internally divided by 2)
- Reset Sequence Manager (RSM)
- System Integrity Management (SI)
  - Main supply Low voltage detection (LVD) with reset generation (enabled by option byte)
  - Auxiliary Voltage detector (AVD) with interrupt capability for monitoring the main supply (enabled by option byte)

### 9.2 Internal RC oscillator adjustment

The device contains an internal RC oscillator with an accuracy of 1% 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 calibration value in the RCCR (RC Control Register).

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 in EEPROM for 3 and 5 V  $V_{DD}$  supply voltages at 25° C, as shown in the following table.



**Table 9. RC control registers**

RCCR	Conditions	ST7DALI address
RCCR0	$V_{DD}=5V$ $T_A=25^{\circ}C$ $f_{RC}=1MHz$	1000h and FFDEh
RCCR1	$V_{DD}=3V$ $T_A=25^{\circ}C$ $f_{RC}=700KHz$	1001h and FFDFh

- Note:**
- 1 [Section 20: Electrical characteristics on page 127](#) for more information on the frequency and accuracy of the RC oscillator.
  - 2 To improve clock stability and frequency accuracy, it is recommended to place a decoupling capacitor, typically 100nF, between the  $V_{DD}$  and  $V_{SS}$  pins as close as possible to the ST7 device.
  - 3 These two bytes are systematically programmed by ST, including on FASTROM devices. Consequently, customers intending to use FASTROM service must not use these two bytes.
  - 4 RCCR0 and RCCR1 calibration values will be erased if the readout protection bit is reset after it has been set. See [Readout protection on page 21](#)

**Caution:** If the voltage or temperature conditions change in the application, the frequency may need to be recalibrated.

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

## 9.3 Phase locked loop

The PLL can be used to multiply a 1 MHz frequency from the RC oscillator or the external clock by 4 or 8 to obtain  $f_{OSC2}$  of 4 or 8 MHz. The PLL is enabled and the multiplication factor of 4 or 8 is selected by 2 option bits.

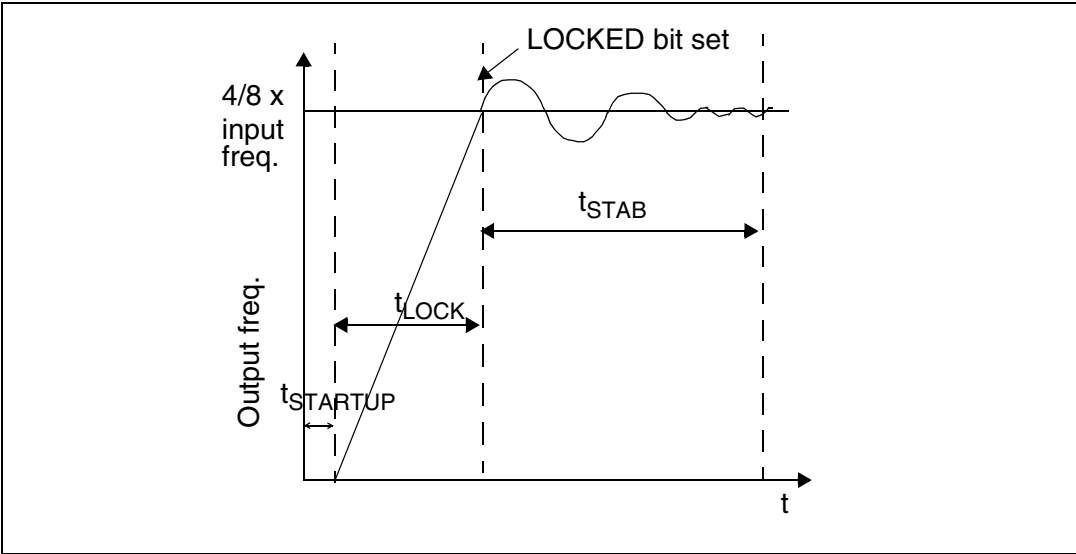
- The x4 PLL is intended for operation with  $V_{DD}$  in the 2.4 V to 3.3 V range
- The x8 PLL is intended for operation with  $V_{DD}$  in the 3.3 V to 5.5 V range

Refer to [Section 22.1 on page 161](#) for the option byte description.

If the PLL is disabled and the RC oscillator is enabled, then  $f_{OSC2} = 1\text{ MHz}$ .

If both the RC oscillator and the PLL are disabled,  $f_{OSC}$  is driven by the external clock.

Figure 11. PLL output frequency timing diagram



When the PLL is started, after reset or wakeup from Halt mode or AWUFH mode, it outputs the clock after a delay of  $t_{\text{STARTUP}}$

When the PLL output signal reaches the operating frequency, the LOCKED bit in the SICSCR register is set. Full PLL accuracy ( $\text{ACC}_{\text{PLL}}$ ) is reached after a stabilization time of  $t_{\text{STAB}}$  (see [Figure 11](#) and [Section 20.3.3: Internal RC oscillator and PLL on page 131](#))

Refer to [Section 9.7.4 on page 44](#) for a description of the LOCKED bit in the SICSCR register.

## 9.4 Register description

### 9.4.1 Main clock control/status register (MCCSR)

Read / Write

Reset Value: 0000 0000 (00h)

7						0	
0	0	0	0	0	0	MCO	SMS

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

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

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 select

This bit is read/write by software and cleared by hardware after a reset. This bit selects the input clock  $f_{\text{OSC2}}$  or  $f_{\text{OSC2}}/32$ .

0: Normal mode ( $f_{\text{CPU}} = f_{\text{OSC2}}$ )

1: Slow mode ( $f_{\text{CPU}} = f_{\text{OSC2}}/32$ )

### 9.4.2 RC control register (RCCR)

Read / Write

Reset Value: 1111 1111 (FFh)

7							0
CR70	CR60	CR50	CR40	CR30	CR20	CR10	CR0

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

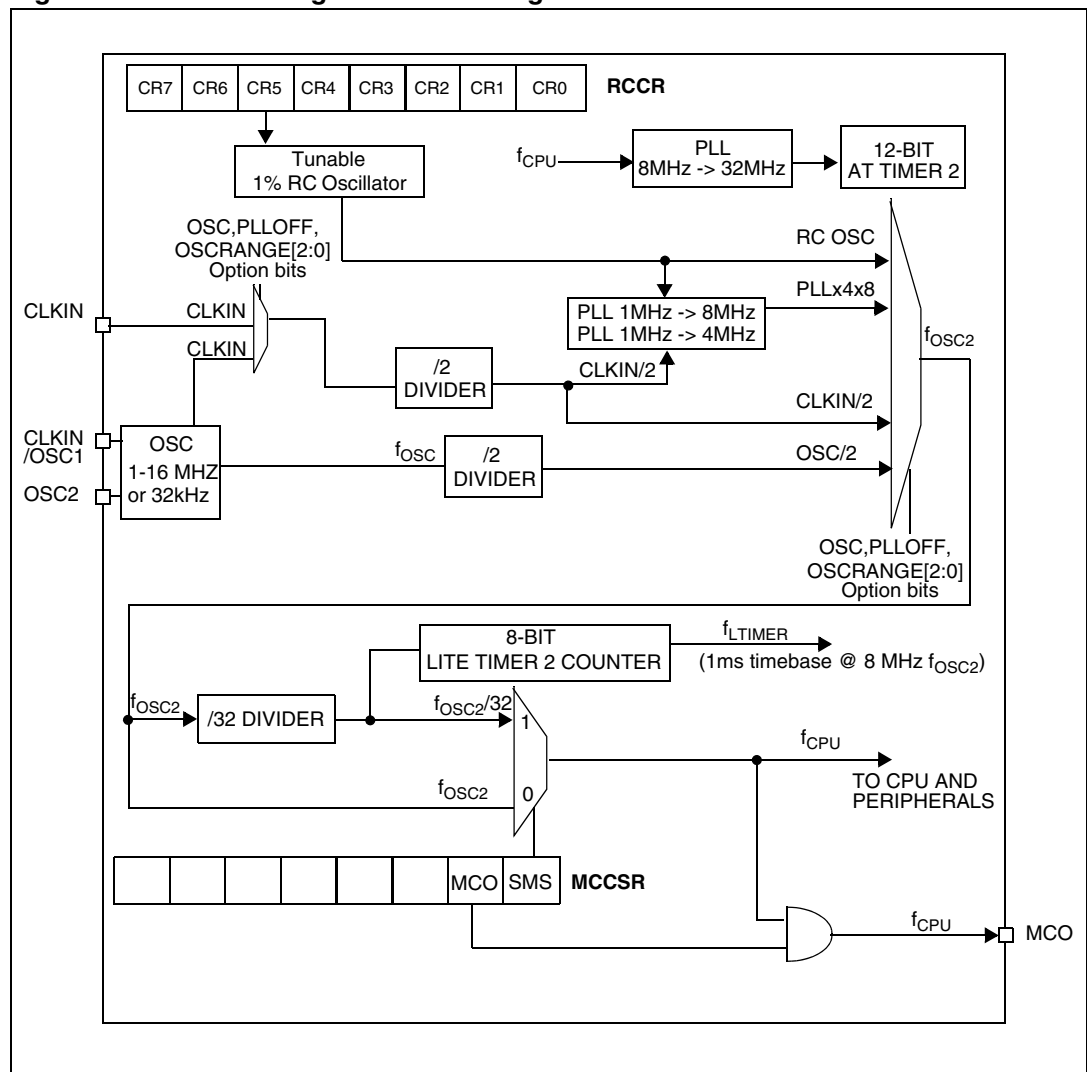
These bits must be written immediately after reset to adjust the RC oscillator frequency and to obtain an accuracy of 1%. The application can store the correct value for each voltage range in EEPROM and write it to this register at start-up.

00h = maximum available frequency

FFh = lowest available frequency

**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.

**Figure 12. Clock management block diagram**



## 9.5 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 or 32 kHz):

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

Each oscillator is optimized for a given frequency range in terms of consumption and is selectable through the option byte. The associated hardware configurations are shown in [Table 10](#). Refer to the electrical characteristics section for more details.

### 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, PB4 is selected by default as external clock.*

### Crystal/ceramic oscillators

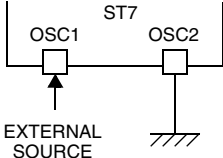
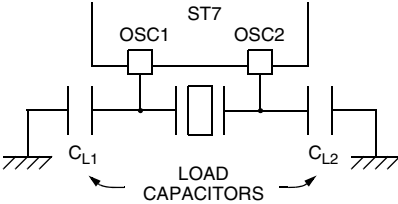
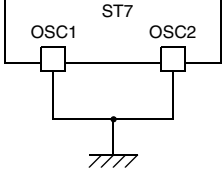
This family of oscillators has the advantage of producing a very accurate rate on the main clock of the ST7. The selection within a list of 4 oscillators with different frequency ranges has to be done by option byte in order to reduce consumption (refer to [Section 22.1 on page 161](#) for more details on the frequency ranges). 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.

### Internal RC oscillator

In this mode, the tunable 1%RC oscillator is used as main clock source. The two oscillator pins have to be tied to ground.

Table 10. ST7 clock sources

Clock source	Hardware configuration
External Clock	
Crystal/Ceramic Resonators	
Internal RC Oscillator or External Clock on PB4	

## 9.6 Reset sequence manager (RSM)

### 9.6.1 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 19.2](#) 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 map.

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 below)
- RESET vector fetch

The 256 or 4096 CPU clock cycle delay allows the oscillator to stabilise 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:

**Table 11. Oscillator delay**

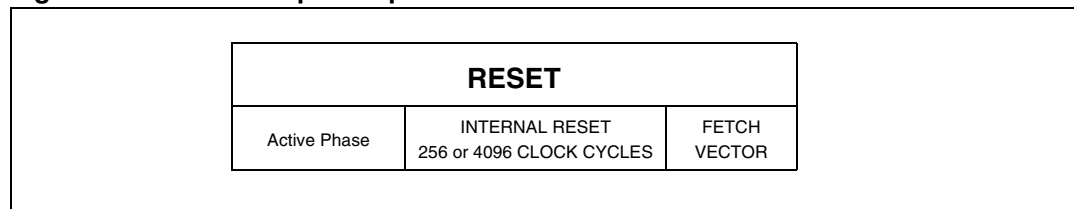
Clock source	CPU clock cycle delay
Internal RC Oscillator	256
External clock (connected to CLKIN pin)	256
External Crystal/Ceramic Oscillator (connected to OSC1/OSC2 pins)	4096

The RESET vector fetch phase duration is 2 clock cycles.

**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 RESET pin in low state until programming mode is entered, in order to avoid unwanted behavior.

If the PLL is enabled by option byte, it outputs the clock after an additional delay of  $t_{\text{STARTUP}}$  (see [Figure 11](#)).

**Figure 13. RESET sequence phases**

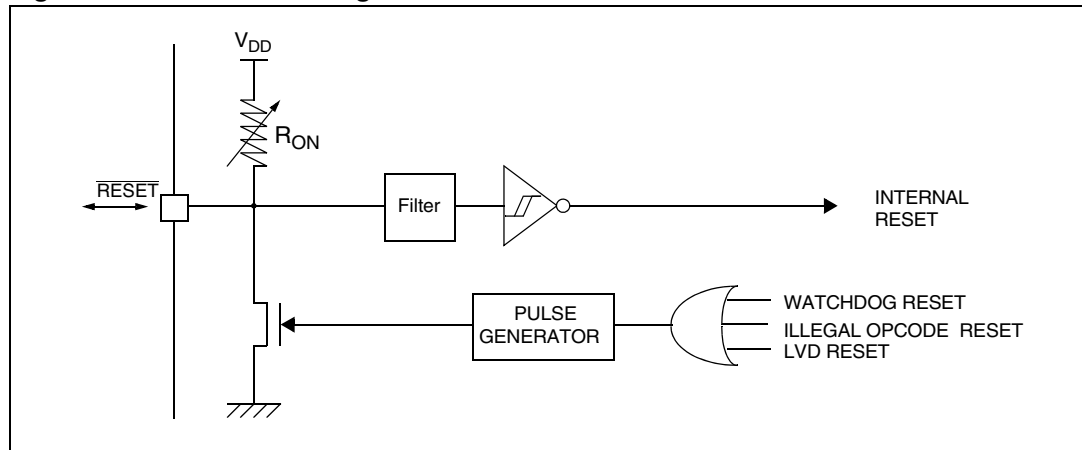


### 9.6.2 Asynchronous external $\overline{\text{RESET}}$ pin

The  $\overline{\text{RESET}}$  pin is both an input and an open-drain output with integrated  $R_{\text{ON}}$  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  $t_{\text{h(RSTL)}}_{\text{in}}$  in order to be recognized (see [Figure 15](#)). This detection is asynchronous and therefore the MCU can enter reset state even in Halt mode.

**Figure 14. Reset block diagram**



*Note:* [Illegal Opcode Reset on page 124](#) for more details on illegal opcode reset conditions.

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.

### 9.6.3 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  $V_{\text{DD}}$  is over the minimum level specified for the selected  $f_{\text{OSC}}$  frequency.

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

### 9.6.4 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  $V_{\text{DD}} < V_{\text{IT+}}$  (rising edge) or  $V_{\text{DD}} < V_{\text{IT-}}$  (falling edge) as shown in [Figure 15](#).

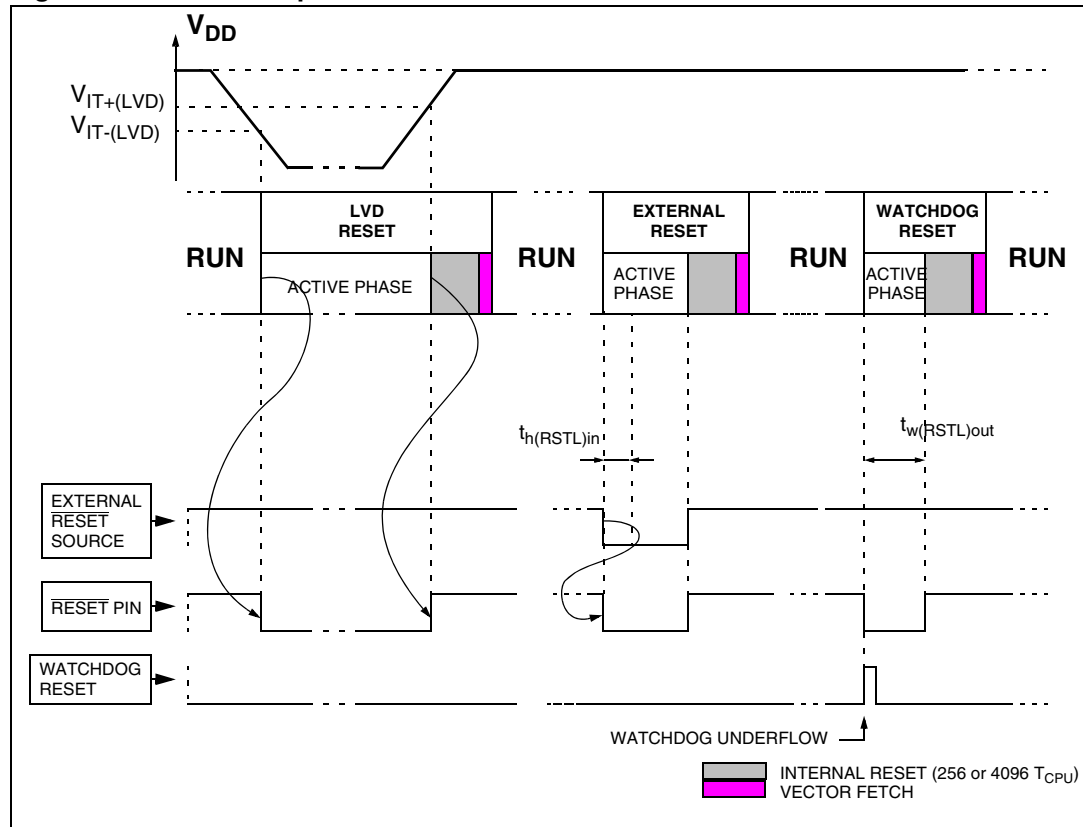
The LVD filters spikes on  $V_{\text{DD}}$  larger than  $t_{\text{g(VDD)}}$  to avoid parasitic resets.

## 9.6.5 Internal watchdog RESET

The RESET sequence generated by a internal Watchdog counter overflow is shown in [Figure 15](#).

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

**Figure 15. RESET sequences**



## 9.7 System integrity management (SI)

The System Integrity Management block contains the Low voltage Detector (LVD) and Auxiliary Voltage Detector (AVD) functions. It is managed by the SICSR register.

*Note:* A reset can also be triggered following the detection of an illegal opcode or prebyte code. Refer to [Illegal Opcode Reset on page 124](#) for further details.

### 9.7.1 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 configured by option byte to be low, medium or high.

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:**
- 1 The LVD allows the device to be used without any external RESET circuitry.
  - 2 The LVD is an optional function which can be selected by option byte.
  - 3 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 0V to ensure optimum restart conditions. Refer to circuit example in [Figure 87 on page 150](#) and note 4.
  - 4 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.

**Figure 16. Low voltage detector vs reset**

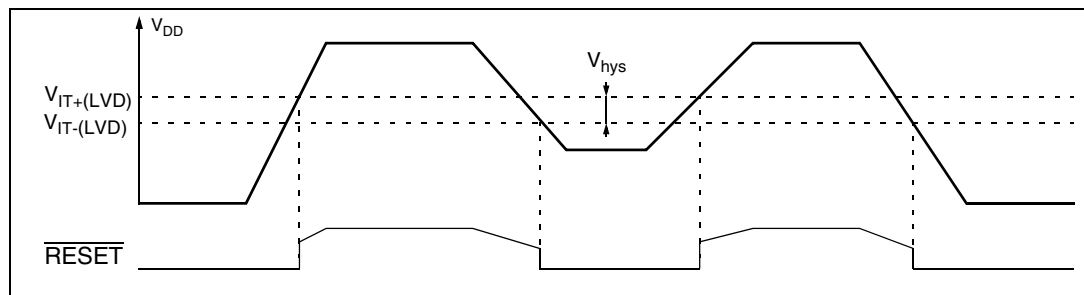
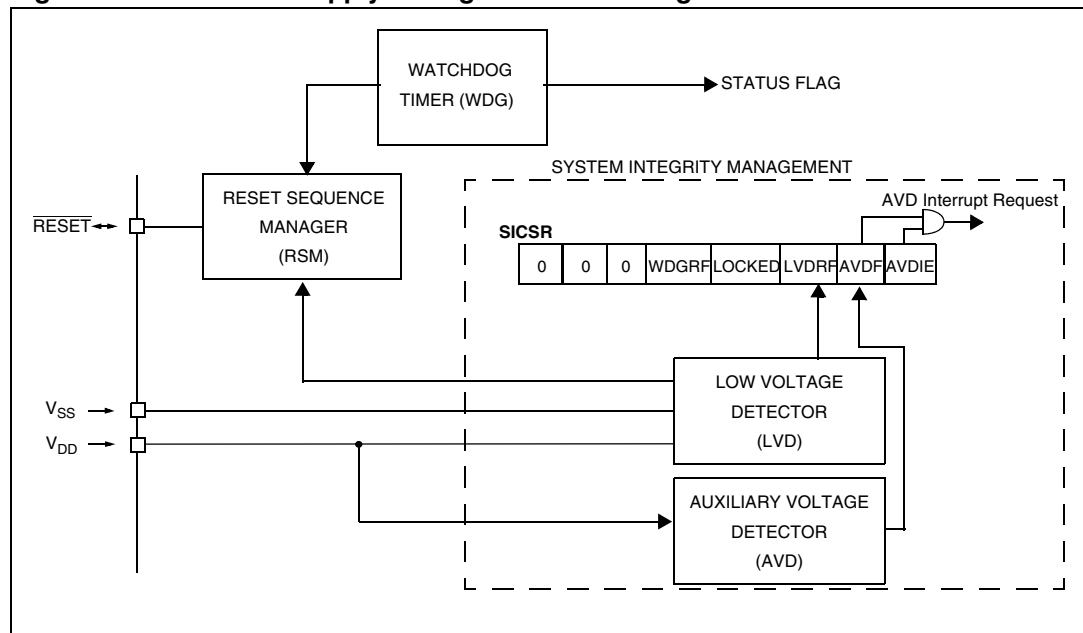


Figure 17. Reset and supply management block diagram



### 9.7.2 Auxiliary voltage detector (AVD)

The Voltage Detector function (AVD) is based on an analog comparison between a  $V_{IT-(AVD)}$  and  $V_{IT+(AVD)}$  reference value and the  $V_{DD}$  main supply voltage ( $V_{AVD}$ ). The  $V_{IT-(AVD)}$  reference value for falling voltage is lower than the  $V_{IT+(AVD)}$  reference value for rising voltage in order to avoid parasitic detection (hysteresis).

The output of the AVD comparator is directly readable by the application software through a real time status bit (AVDF) in the SICS register. This bit is read only.

**Caution:** The AVD functions only if the LVD is enabled through the option byte.

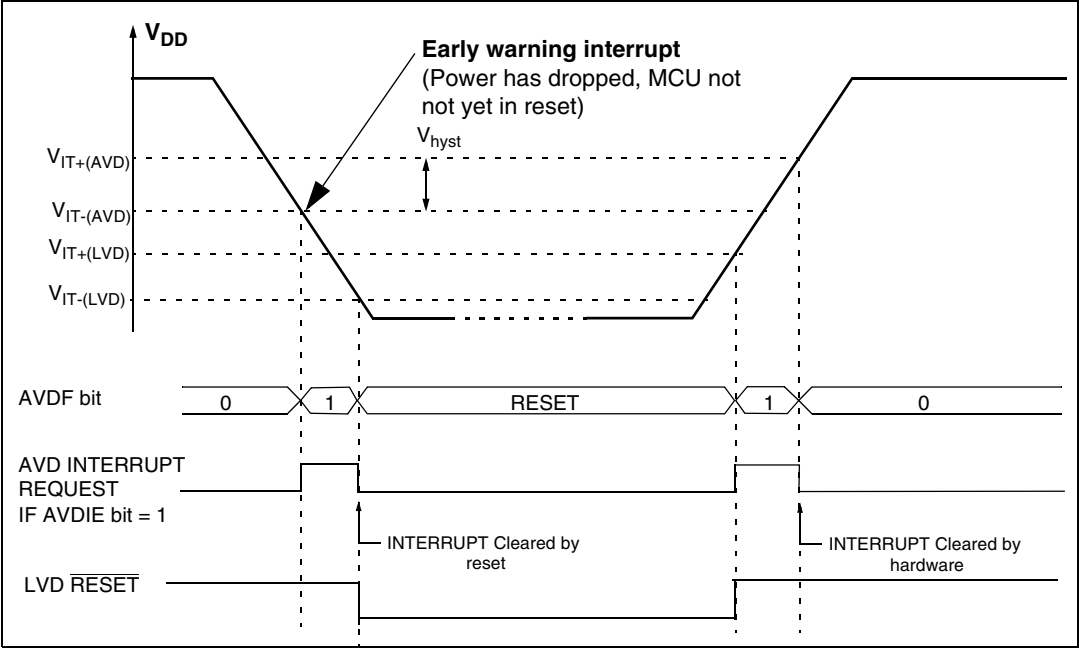
#### Monitoring the $V_{DD}$ main supply

The AVD voltage threshold value is relative to the selected LVD threshold configured by option byte (see [Section 22.1 on page 161](#)).

If the AVD interrupt is enabled, an interrupt is generated when the voltage crosses the  $V_{IT+(LVD)}$  or  $V_{IT-(AVD)}$  threshold (AVDF bit is set).

In the case of a drop in voltage, the AVD interrupt acts as an early warning, allowing software to shut down safely before the LVD resets the microcontroller. See [Figure 18](#).

**Figure 18. Using the AVD to monitor  $V_{DD}$**



### 9.7.3 Low power modes

**Table 12. Effect of low power modes on SI**

Mode	Description
Wait	No effect on SI. AVD interrupts cause the device to exit from Wait mode.
Halt	The SICSR register is frozen. The AVD remains active.

#### Interrupts

The AVD interrupt event generates an interrupt if the corresponding Enable Control Bit (AVDIE) is set and the interrupt mask in the CC register is reset (RIM instruction).

**Table 13. Interrupt control bits**

Interrupt event	Event flag	Enable control bit	Exit from Wait	Exit from Halt
AVD event	AVDF	AVDIE	Yes	No

## 9.7.4 Register description

### System integrity (SI) control/status register (SICSR)

Read/Write

Reset Value: 0000 0xx0 (0xh)

7							0
0	0	0	WDGRF	LOCKED	LVDRF	AVDF	AVDIE

Bit 7:5 = Reserved, must be kept cleared.

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).

Combined with the LVDRF flag information, the flag description is given by the following table.

**Table 14. Reset flags**

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

Bit 3 = **LOCKED** *PLL Locked Flag*

This bit is set and cleared by hardware. It is set automatically when the PLL reaches its operating frequency.

0: PLL not locked

1: PLL locked

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.

Bit 1 = **AVDF** *Voltage Detector flag*

This read-only bit is set and cleared by hardware. If the AVDIE bit is set, an interrupt request is generated when the AVDF bit is set. Refer to [Figure 18](#) and to [Monitoring the VDD main supply on page 42](#) for additional details.

0:  $V_{DD}$  over AVD threshold

1:  $V_{DD}$  under AVD threshold

Bit 0 = **AVDIE** *Voltage Detector interrupt enable*

This bit is set and cleared by software. It enables an interrupt to be generated when the AVDF flag is set. The pending interrupt information is automatically cleared when software enters the AVD interrupt routine.

0: AVD interrupt disabled

1: AVD interrupt enabled

**Note:** *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.*

## 10 Interrupts

The ST7 core may be interrupted by one of two different methods: maskable hardware interrupts as listed in the [Table 15: Interrupt mapping](#) and a non-maskable software interrupt (TRAP). The Interrupt processing flowchart is shown in [Figure 19](#).

The maskable interrupts must be enabled by clearing the I bit in order to be serviced. However, disabled interrupts may be latched and processed when they are enabled (see [External interrupt function on page 61](#)).

*Note:* After reset, all interrupts are disabled.

When an interrupt 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.
- The I bit of the CC register is set to prevent additional interrupts.
- 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 [Table 15: Interrupt mapping](#) for vector addresses).

The interrupt service routine should finish 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 I bit will be cleared and the main program will resume.

### Priority management

By default, a servicing interrupt cannot be interrupted because the I bit is set by hardware entering in interrupt routine.

In the case when several interrupts are simultaneously pending, an hardware priority defines which one will be serviced first (see [Table 15: Interrupt mapping](#)).

### Interrupts and low power mode

All interrupts allow the processor to leave the Wait low power mode. Only external and specifically mentioned interrupts allow the processor to leave the Halt low power mode (refer to the “Exit from Halt” column in [Table 15: Interrupt mapping](#)).

## 10.1 Non maskable software interrupt

This interrupt is entered when the TRAP instruction is executed regardless of the state of the I bit. It will be serviced according to the flowchart on [Figure 19](#).

## 10.2 External interrupts

External interrupt vectors can be loaded into the PC register if the corresponding external interrupt occurred and if the I bit is cleared. These interrupts allow the processor to leave the Halt low power mode.

The external interrupt polarity is selected through the [External interrupt control register \(EICR\)](#).

An external interrupt triggered on edge will be latched and the interrupt request automatically cleared upon entering the interrupt service routine.

**Caution:** The type of sensitivity defined in the [External interrupt control register \(EICR\)](#) applies to the ei source. In case of a NAnDED source (as described in [Section 12: I/O ports](#)), a low level on an I/O pin configured as input with interrupt, masks the interrupt request even in case of rising-edge sensitivity.

### 10.3 Peripheral interrupts

Different peripheral interrupt flags in the status register are able to cause an interrupt when they are active if both:

- The I bit of the CC register is cleared.
- The corresponding enable bit is set in the control register.

If any of these two conditions is false, the interrupt is latched and thus remains pending.

Clearing an interrupt request is done by:

- Writing “0” to the corresponding bit in the status register or
- Access to the status register while the flag is set followed by a read or write of an associated register.

**Note:** *The clearing sequence resets the internal latch. A pending interrupt (i.e. waiting for being enabled) will therefore be lost if the clear sequence is executed.*

**Figure 19. Interrupt processing flowchart**

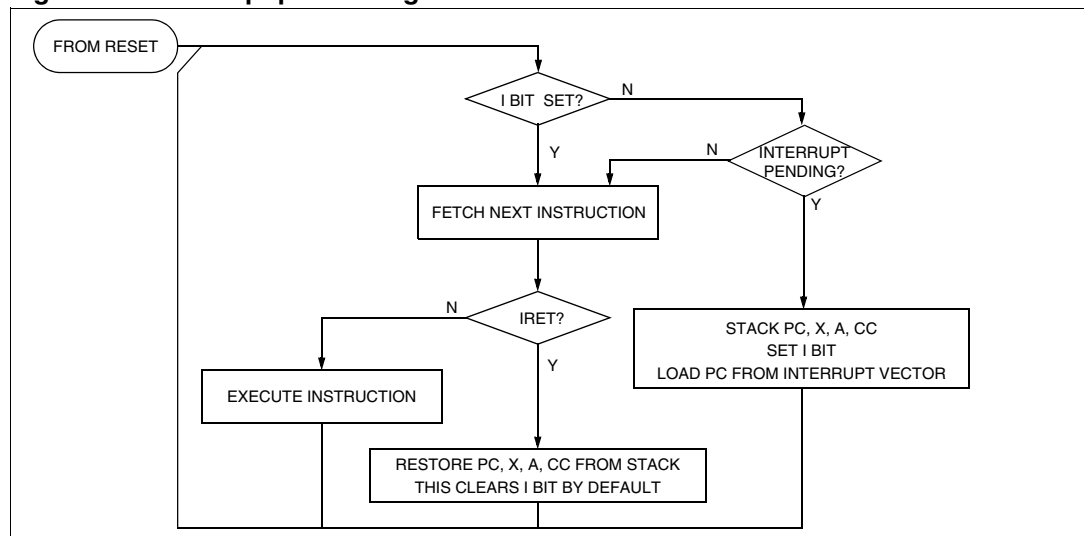


Table 15. Interrupt mapping

N°	Source block	Description	Register label	Priority order	Exit from Halt or AWUFH	Exit from Active-halt	Address vector
	RESET	Reset	N/A	<div>Highest Priority</div> <div>↓</div> <div>Lowest Priority</div>	yes	yes	FFFEh-FFFFh
	TRAP	Software Interrupt			no	no	FFFCh-FFFDh
0	AWU	Auto Wake Up Interrupt	AWUCSR		yes <sup>(1)</sup>		FFFAh-FFFBh
1	ei0	External Interrupt 0	N/A		yes		FFF8h-FFF9h
2	ei1	External Interrupt 1					FFF6h-FFF7h
3	ei2	External Interrupt 2					FFF4h-FFF5h
4	ei3	External Interrupt 3					FFF2h-FFF3h
5	LITE TIMER	LITE TIMER RTC2 interrupt	LTCSR2		no		FFF0h-FFF1h
6	DALI	DALI	DCMCSR		no		FFEEh-FFEFh
7	SI	AVD interrupt	SICSR		no	no	FFEC h-FFEDh
8	AT TIMER	AT TIMER Output Compare Interrupt or Input Capture Interrupt	PWMxCSR or ATCSR				FFEAh-FFEBh
9		AT TIMER Overflow Interrupt	ATCSR			yes	FFE8h-FFE9h
10	LITE TIMER	LITE TIMER Input Capture Interrupt	LTCSR			no	FFE6h-FFE7h
11		LITE TIMER RTC1 Interrupt	LTCSR			yes	FFE4h-FFE5h
12	SPI	SPI Peripheral Interrupts	SPICSR		yes	no	FFE2h-FFE3h
13		Not used					FFE0h-FFE1h

1. This interrupt exits the MCU from "Auto Wake-up from Halt" mode only.

## 10.4 Interrupt registers

### 10.4.1 External interrupt control register (EICR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
IS31	IS30	IS21	IS20	IS11	IS10	IS01	IS00

Bit 7:6 = **IS3[1:0]** *ei3 sensitivity*

These bits define the interrupt sensitivity for ei3 (Port B0) according to [Table 16](#).

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

These bits define the interrupt sensitivity for ei2 (Port B3) according to [Table 16](#).

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

These bits define the interrupt sensitivity for ei1 (Port A7) according to [Table 16](#).

Bit 1:0 = **IS0[1:0]** *ei0 sensitivity*

These bits define the interrupt sensitivity for ei0 (Port A0) according to [Table 16](#).

- 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 [Section 10.4: Interrupt registers](#).

**Table 16. 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

### 10.4.2 External interrupt selection register (EISR)

Read/Write

Reset Value: 0000 1100 (0Ch)

7							0
ei31	ei30	ei21	ei20	ei11	ei10	ei01	ei00

Bit 7:6 = **ei3[1:0]** *ei3 pin selection*

These bits are written by software. They select the Port B I/O pin used for the ei3 external interrupt according to the table below.



**Table 17. External interrupt I/O pin selection**

ei31	ei30	I/O pin
0	0	PB0 <sup>(1)</sup>
0	1	PB1
1	0	PB2

1. Reset state

Bits 5:4 = **ei2[1:0]** *ei2 pin selection*

These bits are written by software. They select the Port B I/O pin used for the ei2 external interrupt according to the table below.

**Table 18. External interrupt I/O pin selection**

ei21	ei20	I/O pin
0	0	PB3 <sup>(1)</sup>
0	1	PB4 <sup>(2)</sup>
1	0	PB5
1	1	PB6

1. Reset state

2. PB4 cannot be used as an external interrupt in Halt mode.

Bit 3:2 = **ei1[1:0]** *ei1 pin selection*

These bits are written by software. They select the Port A I/O pin used for the ei1 external interrupt according to the table below.

**Table 19. External interrupt I/O pin selection**

ei11	ei10	I/O pin
0	0	PA4
0	1	PA5
1	0	PA6
1	1	PA7 <sup>(1)</sup>

1. Reset state

Bits 1:0 = **ei0[1:0]** *ei0 pin selection*

These bits are written by software. They select the Port A I/O pin used for the ei0 external interrupt according to the table below.

**Table 20. External interrupt I/O pin selection**

ei01	ei00	I/O pin
0	0	PA0 <sup>(1)</sup>
0	1	PA1
1	0	PA2
1	1	PA3

1. Reset state

## 11 Power saving modes

### 11.1 Introduction

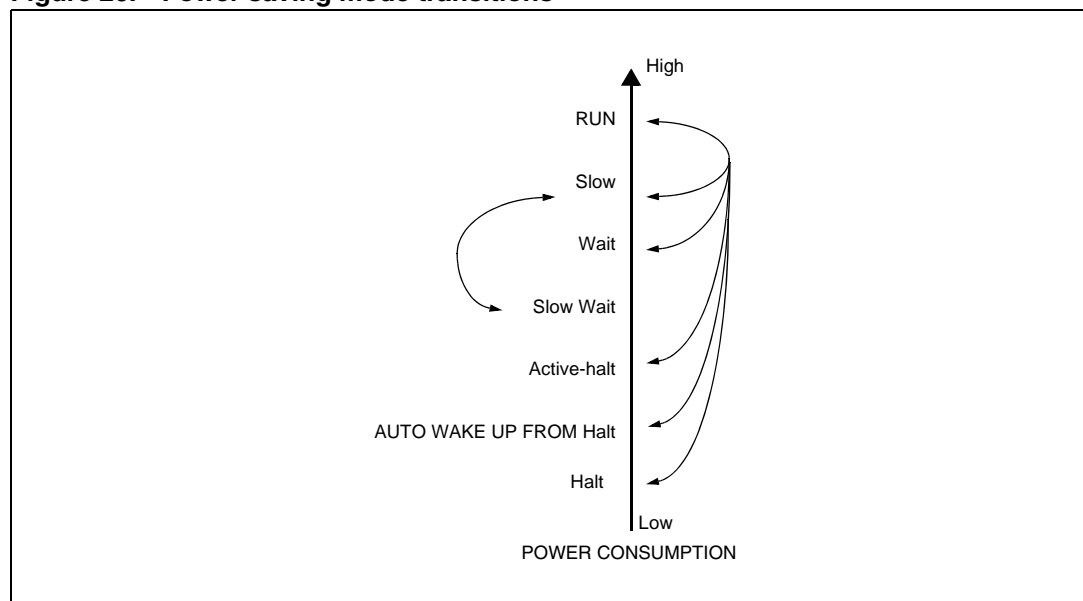
To give a large measure of flexibility to the application in terms of power consumption, five main power saving modes are implemented in the ST7 (see [Figure 20](#)):

- Slow
- Wait (and Slow-Wait)
- Active-halt
- Auto Wake up 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 divided or multiplied by 2 ( $f_{OSC2}$ ).

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

**Figure 20. Power saving mode transitions**



### 11.2 Slow mode

This mode has two targets:

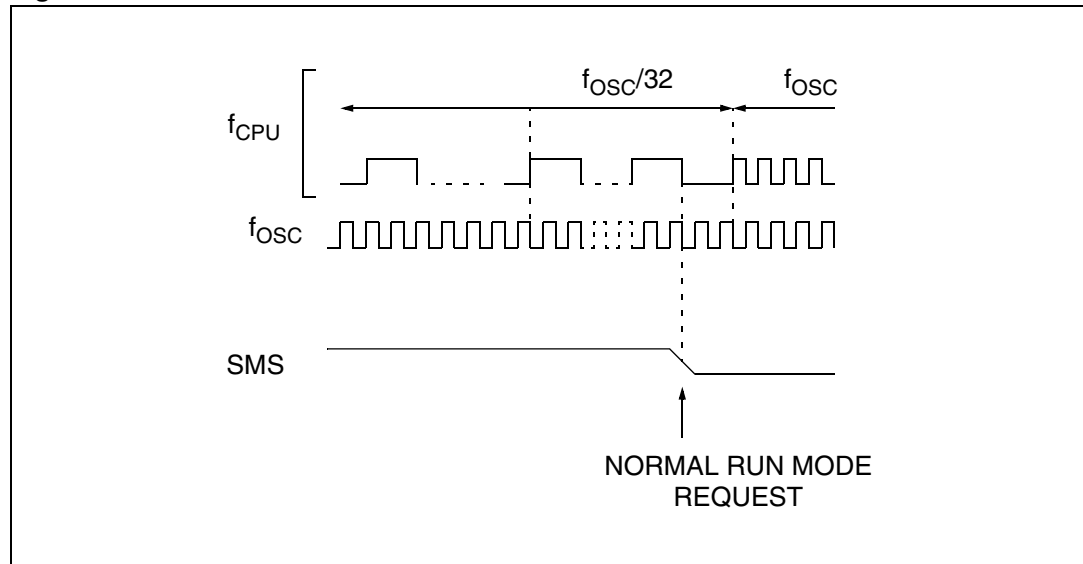
- To reduce power consumption by decreasing the internal clock in the device,
- To adapt the internal clock frequency ( $f_{CPU}$ ) to the available supply voltage.

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 21. Slow mode clock transition**



## 11.3 Wait mode

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

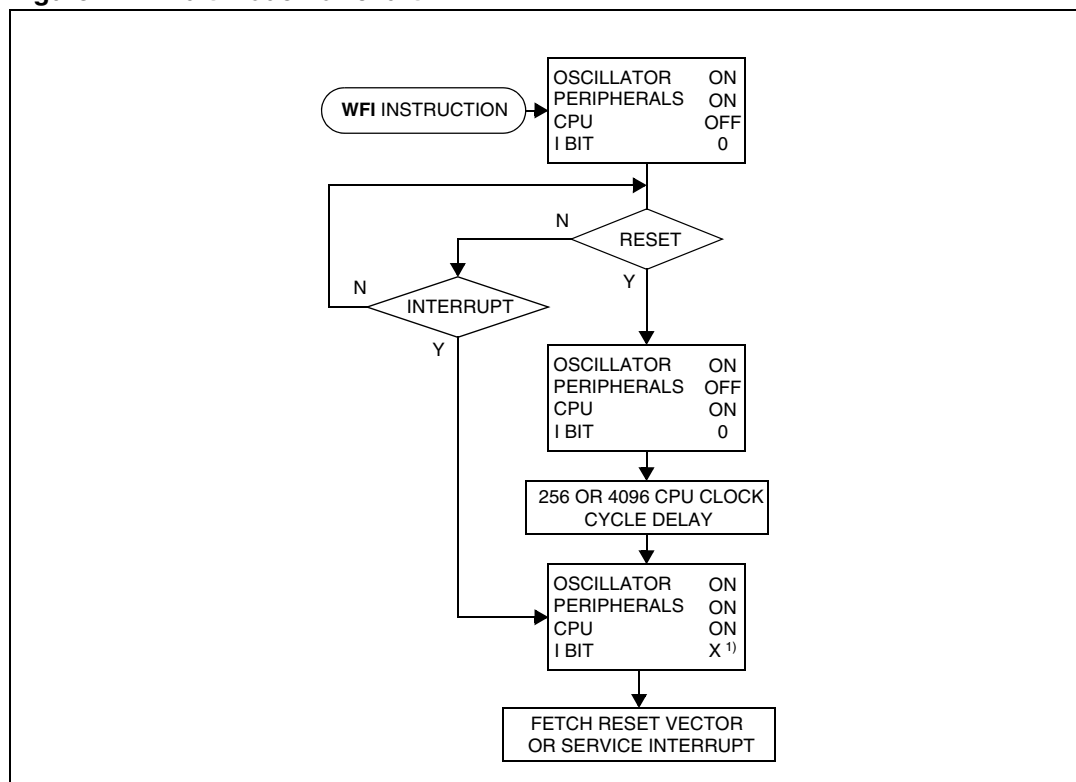
This power saving mode is selected by calling the 'WFI' instruction.

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 22](#).

Figure 22. 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.

## 11.4 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 is disabled (see [Section 11.5 on page 54](#) for more details) and when the AWUEN bit in the AWUCSR register is cleared.

The MCU can exit Halt mode on reception of either a specific interrupt (see [Table 15: Interrupt mapping on page 47](#)) or a RESET. When exiting Halt mode by means of a RESET or an interrupt, the oscillator is immediately turned on and the 256 or 4096 CPU cycle delay is used to stabilize the oscillator. 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 24](#)).

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 up 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 22.1 on page 161](#) for more details).

Figure 23. Halt mode timing overview

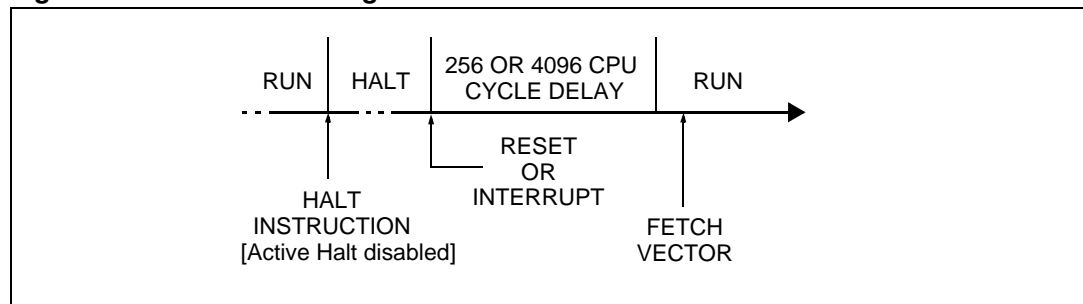
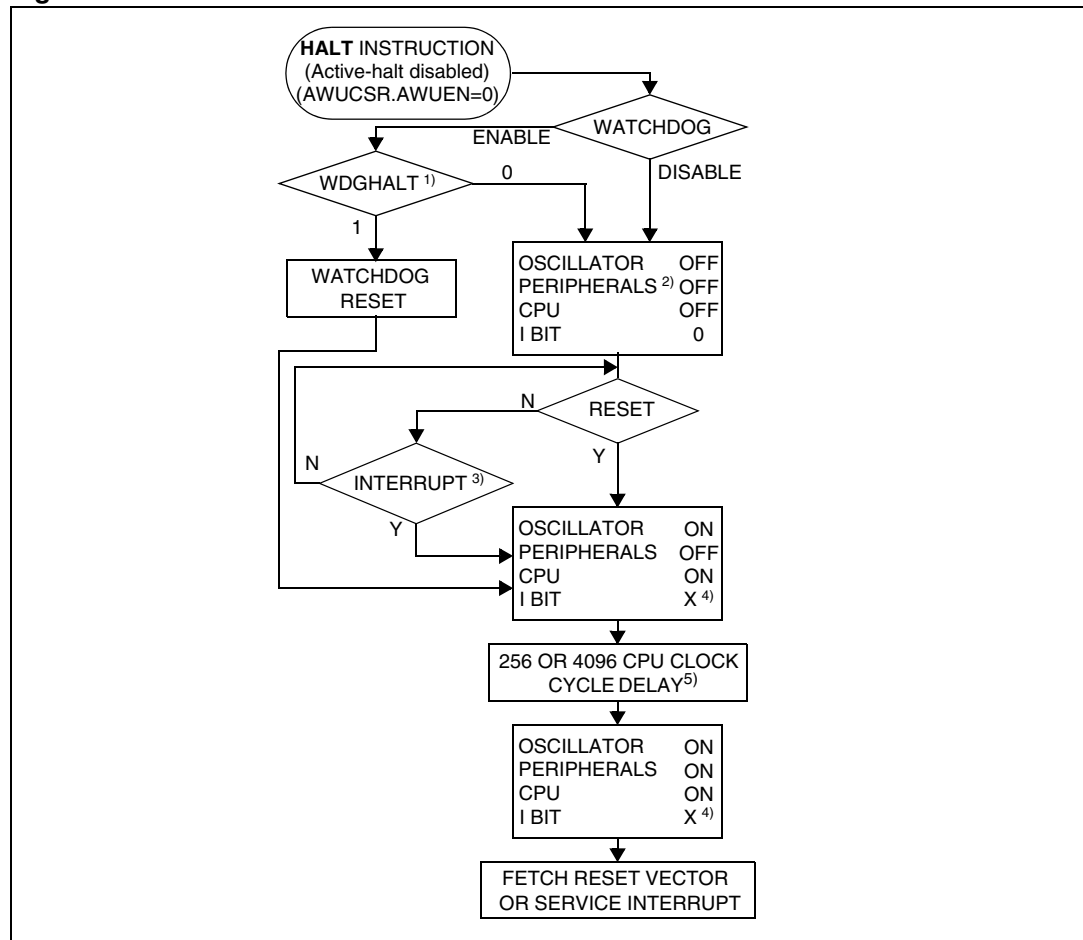


Figure 24. Halt mode flowchart



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 15: 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. If the PLL is enabled by option byte, it outputs the clock after a delay of  $t_{STARTUP}$  (see [Figure 11](#)).

### 11.4.1 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 program memory with the value 0x8E.
- As the HALT instruction clears the interrupt mask 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 wake-up event (reset or external interrupt).

## 11.5 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. 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 21. Active-Halt control**

LTCSR1 TB1IE bit	ATCSR OVFI bit	ATCSRCK1 bit	ATCSRCK0 bit	Meaning
0	x	x	0	Active-halt mode disabled
0	0	x	x	
1	x	x	x	Active-halt mode enabled
x	1	0	1	

The MCU can exit Active-halt mode on reception of a specific interrupt (see [Table 15: Interrupt mapping on page 47](#)) or a RESET.

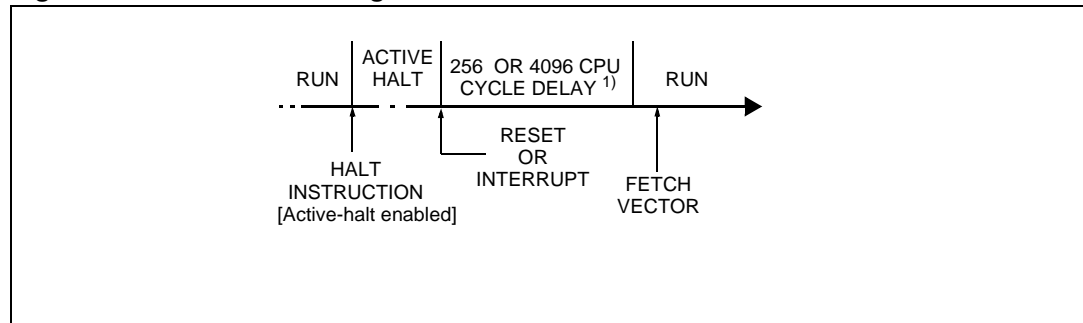
- When exiting Active-halt mode by means of a RESET, a 256 or 4096 CPU cycle delay occurs. After the start up delay, the CPU resumes operation by fetching the reset vector which woke it up (see [Figure 26](#)).
- When exiting Active-halt mode by means of an interrupt, the CPU immediately resumes operation 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 (see Note 3).

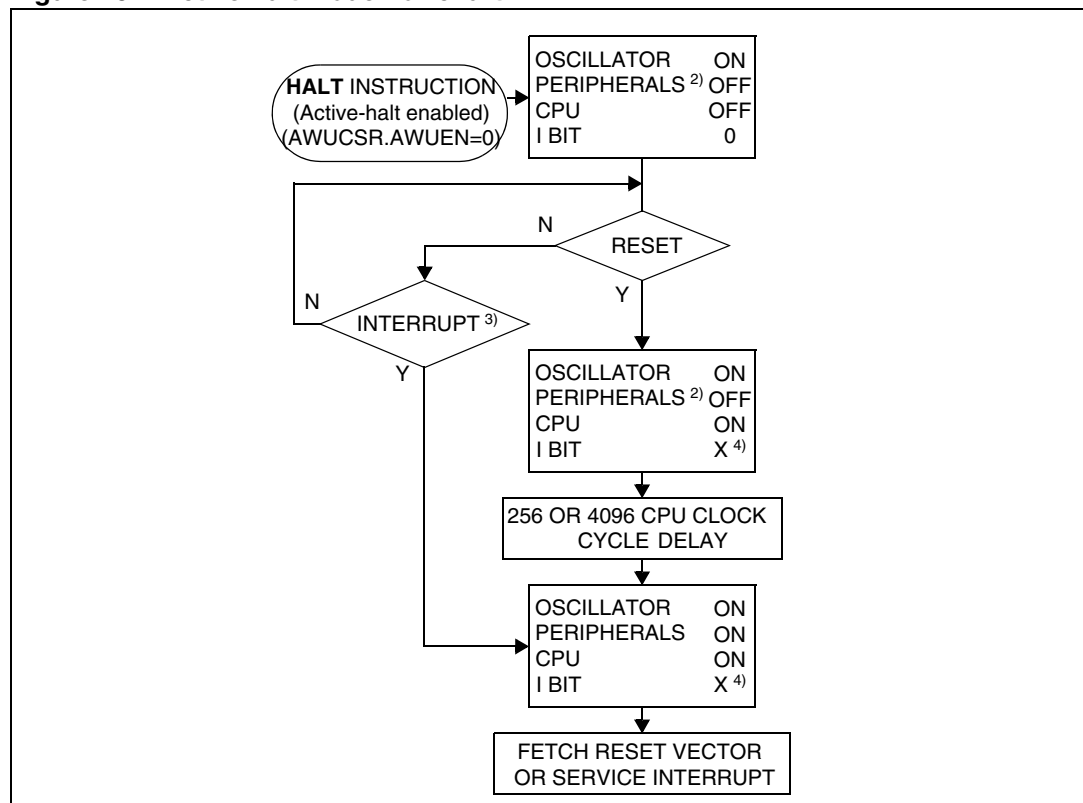
In Active-halt mode, only the main oscillator and the selected timer counter (LT/AT) are running to keep a wake-up time base. All other peripherals are not clocked except those which get their clock supply from another clock generator (such as external or auxiliary oscillator).

**Note:** As soon as Active-halt is enabled, executing a HALT instruction while the Watchdog is active does not generate a RESET.  
This means that the device cannot spend more than a defined delay in this power saving mode.

**Figure 25. Active-halt timing overview**



**Figure 26. Active-halt mode flowchart**



**Notes:**

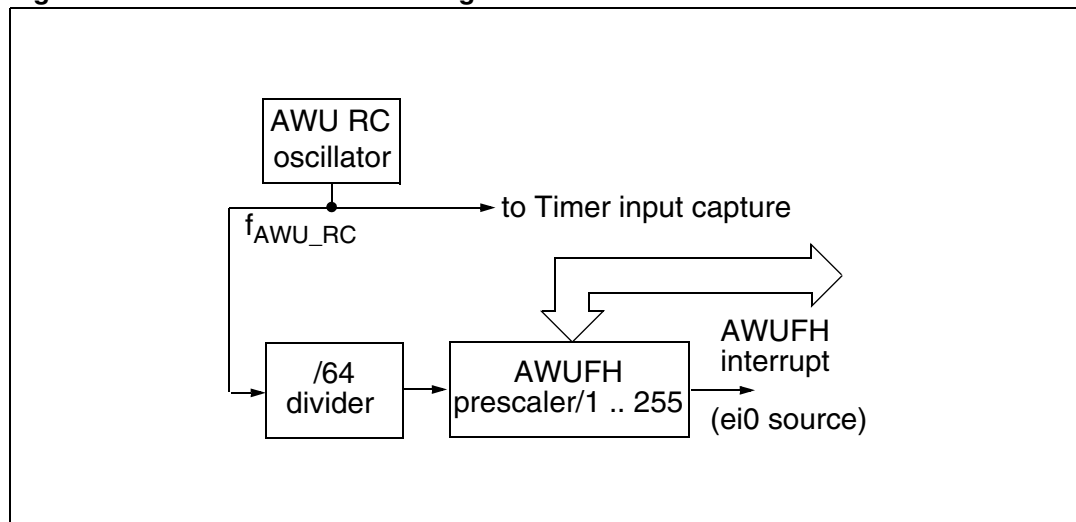
1. This delay occurs only if the MCU exits Active-halt mode by means of a RESET.
2. Peripherals clocked with an external clock source can still be active.
3. Only the RTC1 interrupt and some specific interrupts can exit the MCU from Active-halt mode. Refer to [Table 15: Interrupt mapping on page 47](#) 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.

## 11.6 Auto wakeup from Halt mode

Auto Wake Up From Halt (AWUFH) mode is similar to Halt mode with the addition of a specific internal RC oscillator for wake-up (Auto Wake Up from Halt Oscillator). 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 27. 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 AWUF flag is set by hardware and an interrupt wakes-up the MCU from Halt mode. At the same time the main oscillator is immediately turned on and a 256 or 4096 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 12-bit Auto-Reload 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 11.4: Halt mode](#)).
- 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 28. AWUF Halt timing diagram**

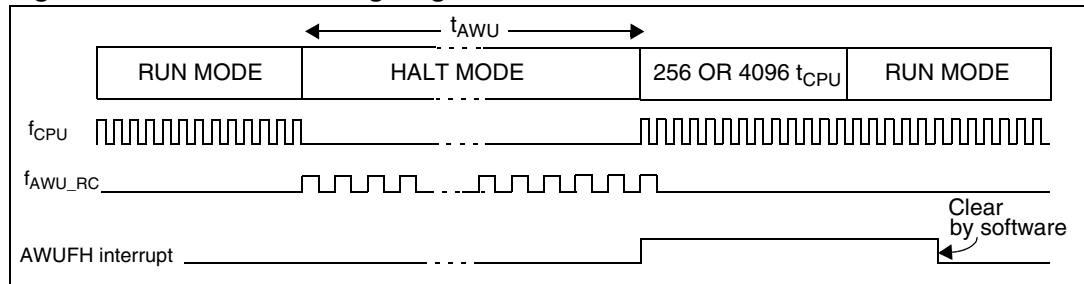
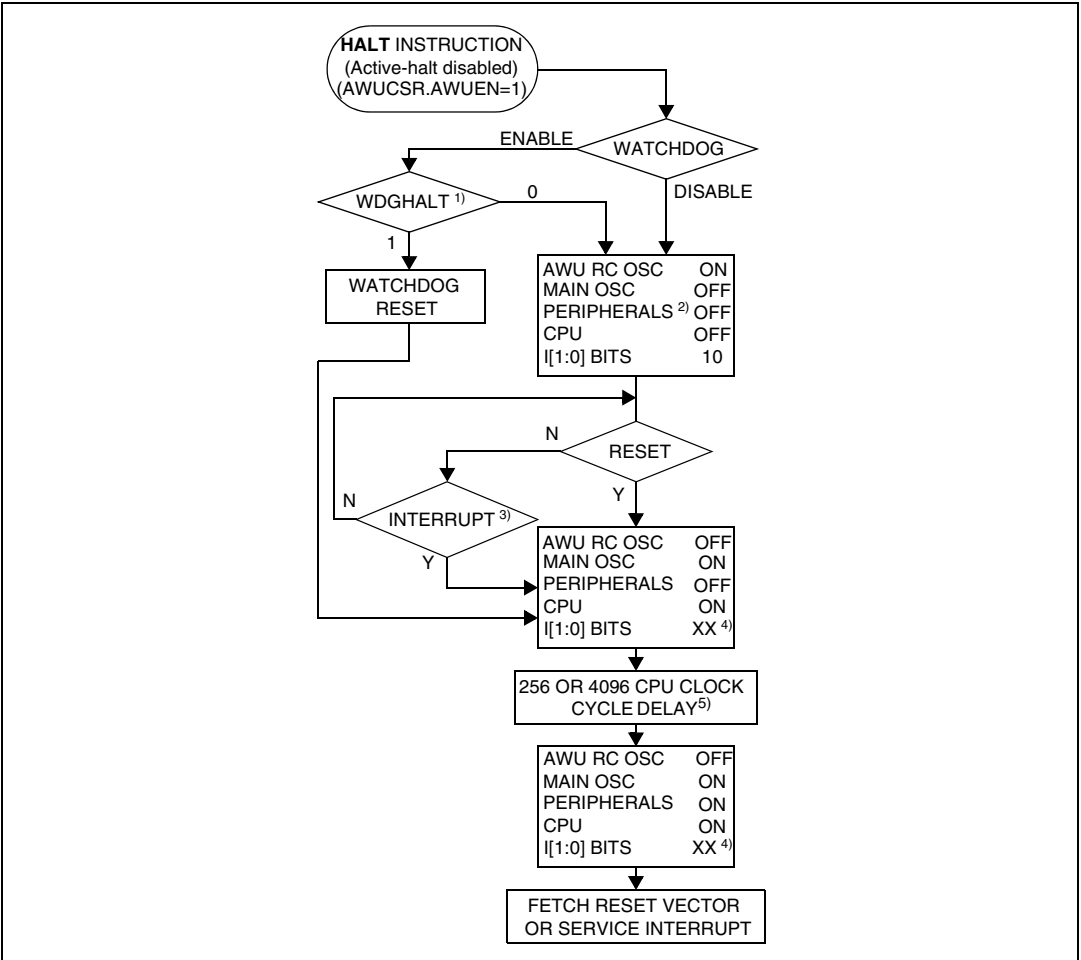


Figure 29. AWUFH mode flowchart



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 15: Interrupt mapping on page 47](#) 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.
5. If the PLL is enabled by option byte, it outputs the clock after an additional delay of  $t_{STARTUP}$  (see [Figure 11](#)).

### 11.6.1 Register description

#### AWUFH control/status register (AWUCSR)

Read/Write

Reset Value: 0000 0000 (00h)

7					0		
0	0	0	0	0	AWU F	AWUM	AWUEN

Bits 7:3 = Reserved.

**Bit 2 = AWUF Auto Wake Up 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 Wake Up Measurement**

This bit enables the AWU RC oscillator and connects its output to the input capture of the 12-bit Auto-Reload 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 Wake Up From Halt Enabled**

This bit enables the Auto Wake Up 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 Wake Up From Halt) mode disabled

1: AWUFH (Auto Wake Up From Halt) mode enabled

**AWUFH prescaler register (AWUPR)**

Read/Write

Reset Value: 1111 1111 (FFh)

7							0
AWUPR7	AWUPR6	AWUPR5	AWUPR4	AWUPR3	AWUPR2	AWUPR1	AWUPR0

**Bits 7:0 = AWUPR[7:0] Auto Wake Up Prescaler**

These 8 bits define the AWUPR Dividing factor (as explained below):

**Table 22. AWU prescaler**

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

In AWU mode, the period that the MCU stays in Halt Mode ( $t_{AWU}$  in [Figure 28 on page 57](#)) is defined by

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

This prescaler register can be programmed to modify the time that the MCU stays in Halt mode before waking up automatically.

*Note:* If 00h is written to AWUPR, depending on the product, an interrupt is generated immediately after a HALT instruction, or the AWUPR remains unchanged.

**Table 23. AWU register map and reset values**

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0049h	<b>AWUPR</b> Reset Value	AWUPR7 1	AWUPR6 1	AWUPR5 1	AWUPR4 1	AWUPR3 1	AWUPR2 1	AWUPR1 1	AWUPR0 1
004Ah	<b>AWUCSR</b> Reset Value	0	0	0	0	0	AWUF	AWUM	AWUEN

## 12 I/O ports

### 12.1 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.

### 12.2 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 30](#) shows the generic I/O block diagram.

#### 12.2.1 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 \(EICR\) on page 48](#) controls this sensitivity.

Each external interrupt vector is linked to a dedicated group of I/O port pins (see [Section 4: Pin description on page 13](#) and [Section 10: Interrupts on page 45](#)).

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 reenale them after the complete previous sequence in order to avoid an external interrupt occurring on the unwanted edge.

This corresponds to the following steps:

1. To enable an external interrupt:
  - set the interrupt mask with the SIM instruction (in cases where a pin level change could occur)
  - select rising edge
  - enable the external interrupt through the OR register
  - select the desired sensitivity if different from rising edge
  - reset the interrupt mask with the RIM instruction (in cases where a pin level change could occur)
2. To disable an external interrupt:
  - set the interrupt mask with the SIM instruction SIM (in cases where a pin level change could occur)
  - select falling edge
  - disable the external interrupt through the OR register
  - select rising edge
  - reset the interrupt mask with the RIM instruction (in cases where a pin level change could occur)

## 12.2.2 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 24. DR value and output pin status**

DR	Push-pull	Open-drain
0	$V_{OL}$	$V_{OL}$
1	$V_{OH}$	Floating

### 12.2.3 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. The [Table 2: Device pin description on page 14](#) describes which peripheral signals can be input/output to which ports.

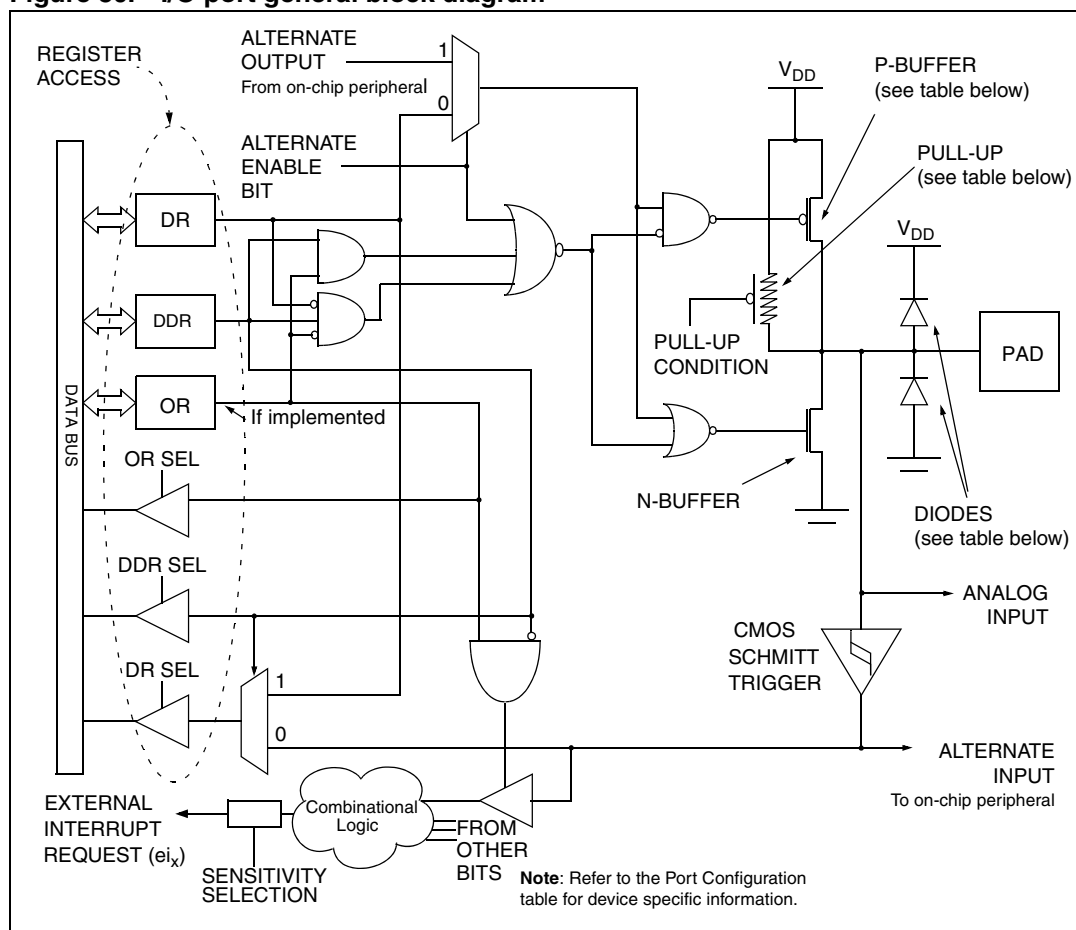
A signal coming from an on-chip peripheral can be output on an I/O. To do this, enable the on-chip peripheral as an output (enable bit in the peripheral's control register). The peripheral configures the I/O as an output and takes priority over standard I/O programming. The I/O's state is readable by 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 30. I/O port general block diagram**



**Table 25. I/O port mode options**

Configuration mode		Pull-up	P-buffer	Diodes	
				to V <sub>DD</sub>	to V <sub>SS</sub>
Input	Floating with/without Interrupt	Off	Off	On	On
	Pull-up with/without Interrupt	On			
Output	Push-pull	Off	On	On	On
	Open Drain (logic level)		Off		
	True Open Drain	NI	NI	NI <sup>(1)</sup>	

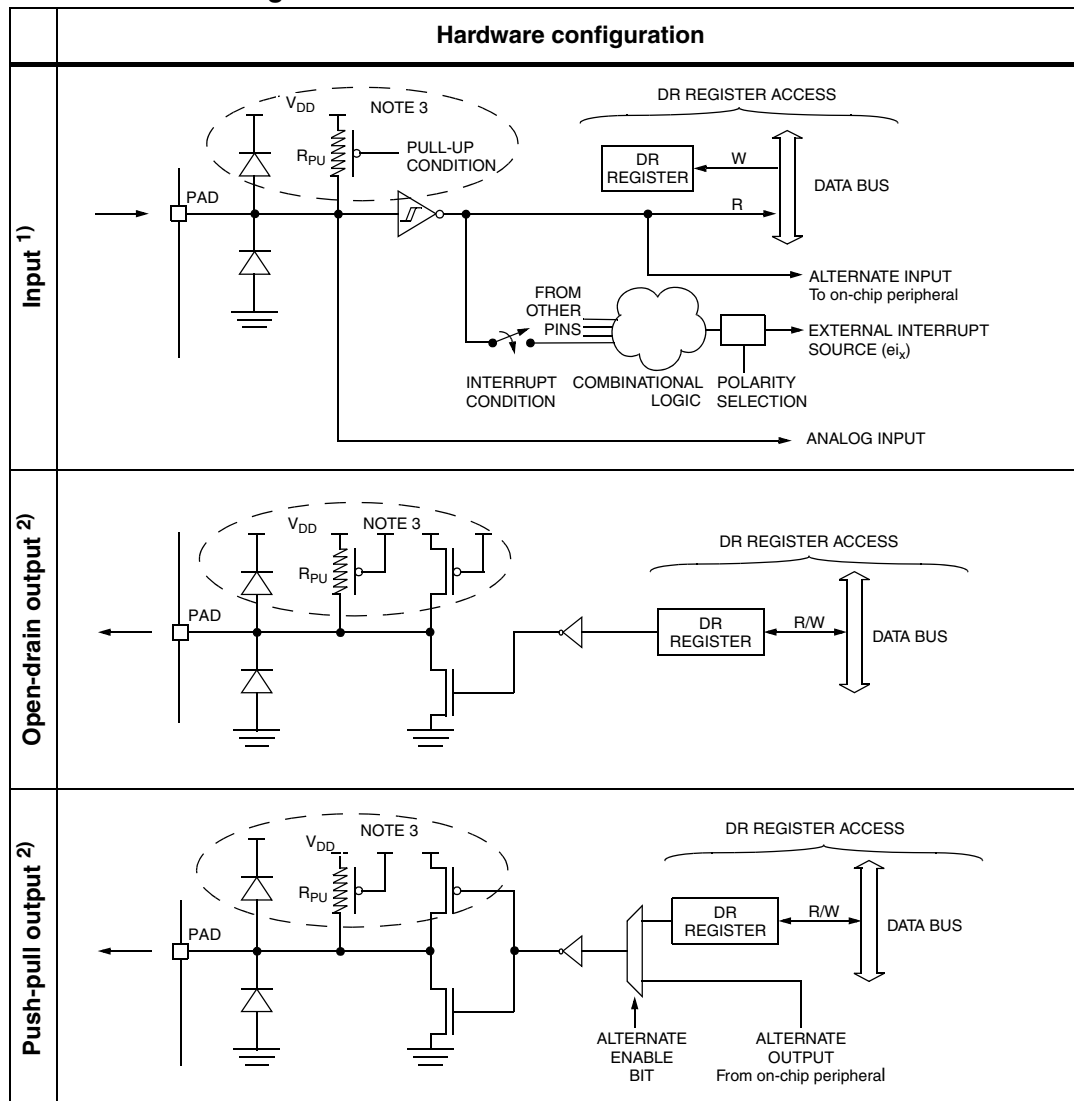
1. The diode to VDD is not implemented in the true open drain pads. A local protection between the pad and VOL is implemented to protect the device against positive stress.

**Legend:**

- NI - not implemented
- Off - implemented not activated
- On - implemented and activated



Table 26. I/O configurations



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.
3. For true open drain, these elements are not implemented.

### 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.

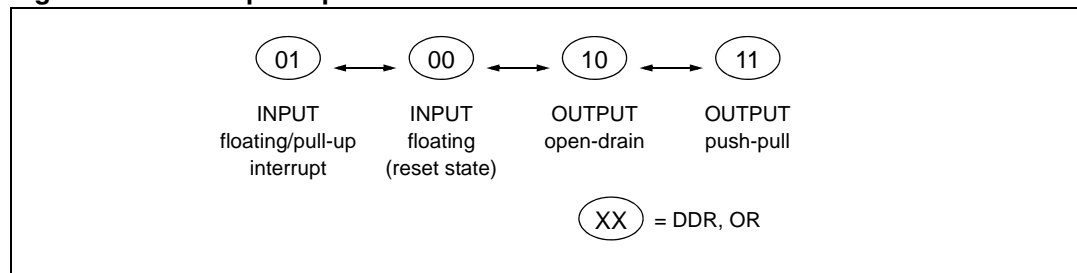
**Caution:** The analog input voltage level must be within the limits stated in the absolute maximum ratings.

## 12.3 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 31](#). Other transitions are potentially risky and should be avoided, since they may present unwanted side-effects such as spurious interrupt generation.

**Figure 31. Interrupt I/O port state transitions**



## 12.4 Unused I/O pins

Unused I/O pins must be connected to fixed voltage levels. Refer to [Section 20](#).

## 12.5 Low power modes

**Table 27. 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.

## 12.6 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 28. I/O port interrupt control/wake-up capability**

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

## 12.7 Device-specific I/O port configuration

The I/O port register configurations are summarized as follows.

**Table 29. Ports PA7:0, PB6:0 with interrupt capability not selected in EISR register**

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

**Table 30. Ports PA7:0, PB6:0 with interrupt capability selected in EISR register**

Mode	DDR	OR
Floating input	0	0
Pull-up interrupt input	0	1
Open drain output	1	0
Push-pull output	1	1

**Table 31. Port configuration (interrupt capability not selected in the EISR register)**

Port	Pin name	Input		Output	
		OR = 0	OR = 1	OR = 0	OR = 1
Port A	PA7:0	floating	pull-up	open drain	push-pull
Port B	PB6:0	floating	pull-up	open drain	push-pull

**Table 32. Port configuration (interrupt capability selected in the EISR register)**

Port	Pin name	Input		Output	
		OR = 0	OR = 1	OR = 0	OR = 1
Port A	PA7:0	floating	pull-up interrupt	open drain	push-pull
Port B	PB6:0	floating	pull-up interrupt	open drain	push-pull

**Table 33. I/O port register map and reset values**

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0000h	PADR Reset Value	MSB 1	1	1	1	1	1	1	LSB 1
0001h	PADDR Reset Value	MSB 0	0	0	0	0	0	0	LSB 0
0002h	PAOR Reset Value	MSB 0	1	0	0	0	0	0	LSB 0

**Table 33. I/O port register map and reset values (continued)**

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0003h	PBDR Reset Value	MSB 1	1	1	1	1	1	1	LSB 1
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

## 13 Watchdog timer (WDG)

### 13.1 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.

### 13.2 Main features

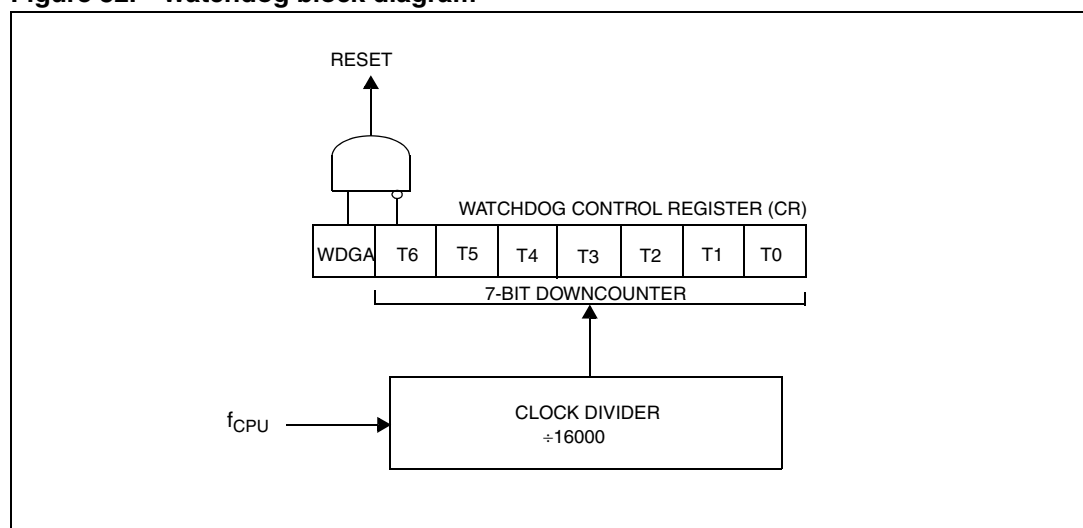
- Programmable free-running downcounter (64 increments of 16000 CPU cycles)
- Programmable reset
- Reset (if watchdog activated) when the T6 bit reaches zero
- Optional reset on HALT instruction (configurable by option byte)
- Hardware Watchdog selectable by option byte

### 13.3 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 reset pin for typically 30  $\mu$ s.

**Figure 32. 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 34](#)):

- 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 34. Watchdog timing ( $f_{CPU} = 8 \text{ MHz.}$ )**

WDG counter code	min [ms]	max [ms]
C0h	1	2
FFh	127	128

- Note:*
- 1 The timing variation shown in [Table 34](#) 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.

## 13.4 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 22.1 on page 161](#).

### 13.4.1 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. The same behavior occurs in Active-halt mode.

## 13.5 Interrupts

None.

## 13.6 Register description

### 13.6.1 Control register (CR)

Read/Write

Reset Value: 0111 1111 (7Fh)

7							0
WDGA	T6	T5	T4	T3	T2	T1	T0

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 35. Watchdog timer register map and reset values**

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
002Eh	WDGCR Reset Value	WDGA 0	T6 1	T5 1	T4 1	T3 1	T2 1	T1 1	T0 1

## 14 12-bit autoreload timer 2 (AT2)

### 14.1 Introduction

The 12-bit Autoreload Timer can be used for general-purpose timing functions. It is based on a free-running 12-bit upcounter with an input capture register and four PWM output channels. There are 6 external pins:

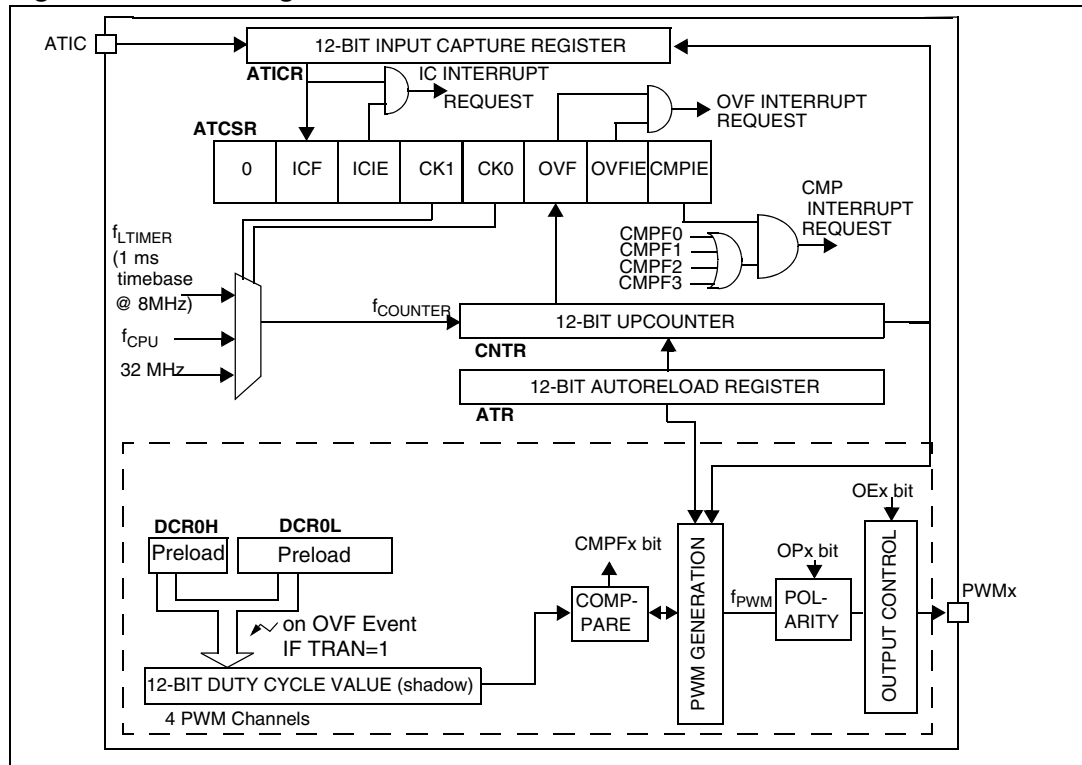
- Four PWM outputs
- ATIC pin for the Input Capture function
- BREAK pin for forcing a break condition on the PWM outputs

### 14.2 Main features

- 12-bit upcounter with 12-bit autoreload register (ATR)
- Maskable overflow interrupt
- Generation of four independent PWMx signals
- Frequency 2 kHz-4 MHz (@ 8 MHz  $f_{CPU}$ )
  - Programmable duty-cycles
  - Polarity control
  - Programmable output modes
  - Maskable Compare interrupt
- Input Capture
  - 12-bit input capture register (ATICR)
  - Triggered by rising and falling edges
  - Maskable IC interrupt



Figure 33. Block diagram



## 14.3 Functional description

### 14.3.1 PWM mode

This mode allows up to four Pulse Width Modulated signals to be generated on the PWMx output pins. The PWMx output signals can be enabled or disabled using the OEx bits in the PWMCR register.

#### PWM frequency and duty cycle

The four PWM signals have the same frequency ( $f_{PWM}$ ) which is controlled by the counter period and the ATR register value.

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

Following the above formula,

- If  $f_{COUNTER}$  is 32 MHz, the maximum value of  $f_{PWM}$  is 8 MHz (ATR register value = 4092), the minimum value is 8 kHz (ATR register value = 0)
- If  $f_{COUNTER}$  is 4 Mhz, the maximum value of  $f_{PWM}$  is 2 MHz (ATR register value = 4094), the minimum value is 1 KHz (ATR register value = 0).

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

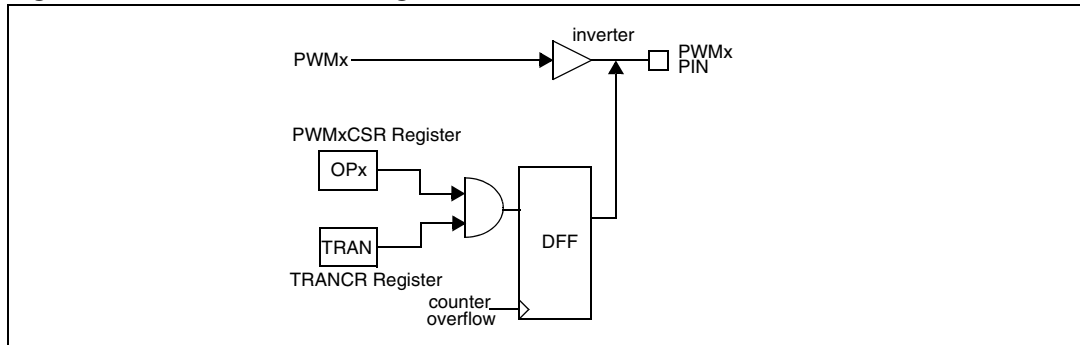
At reset, the counter starts counting from 0.

When a upcounter overflow occurs (OVF event), the preloaded Duty cycle values are transferred to the Duty Cycle registers and the PWMx signals are set to a high level. When

the upcounter matches the DCRx value the PWMx signals are set to a low level. To obtain a signal on a PWMx pin, the contents of the corresponding DCRx register must be greater than the contents of the ATR register.

The polarity bits can be used to invert any of the four output signals. The inversion is synchronized with the counter overflow if the TRAN bit in the TRANCN register is set (reset value). See [Figure 34](#).

**Figure 34. PWM inversion diagram**

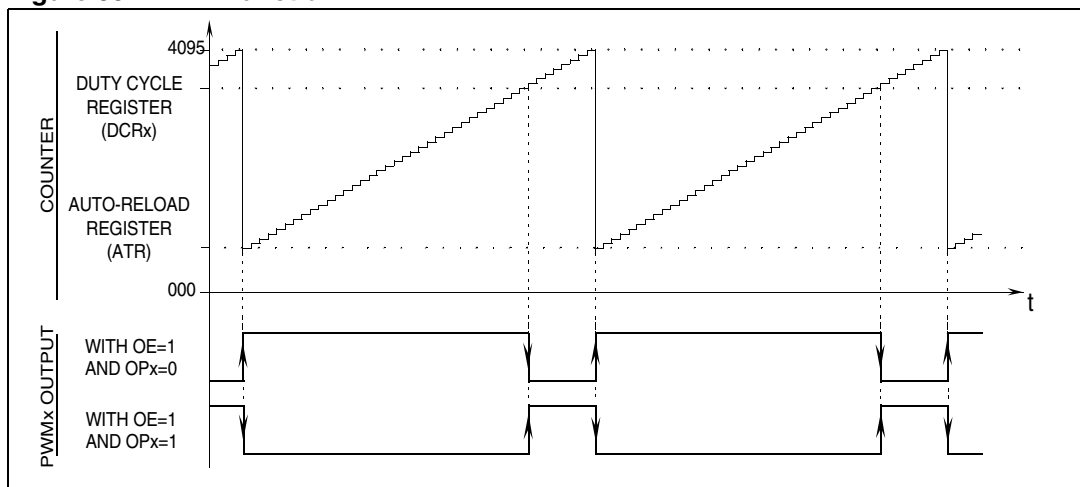


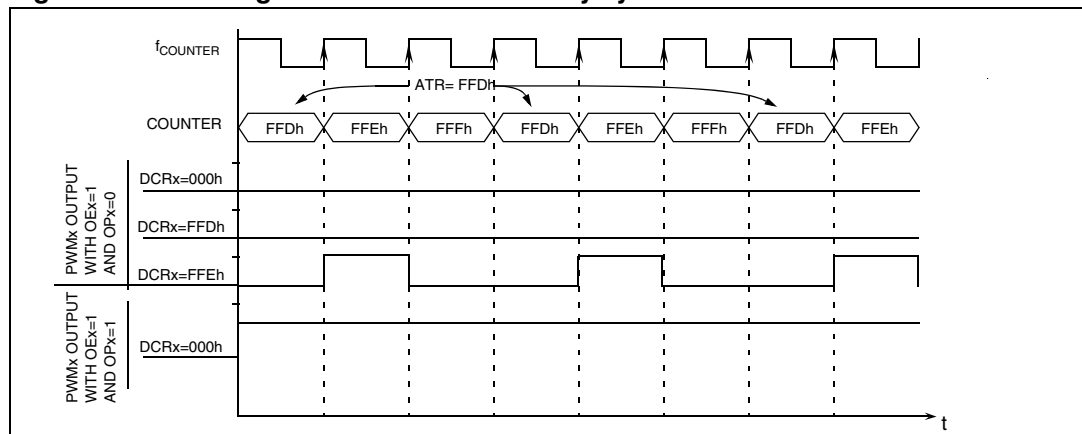
The maximum available resolution for the PWMx duty cycle is:

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

**Note:** To get the maximum resolution (1/4096), the ATR register must be 0. With this maximum resolution, 0% and 100% can be obtained by changing the polarity.

**Figure 35. PWM function**



**Figure 36. PWM signal from 0% to 100% duty cycle**

### 14.3.2 Output compare mode

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

When the 12-bit upcounter (CNTR) reaches the value stored in the DCRxH and DCRxL registers, the CMPF bit in the PWMxCSR register is set and an interrupt request is generated if the CMPIE bit is set.

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

### 14.3.3 Break function

The break function is used to perform an emergency shutdown of the power converter.

The break function is activated by the external BREAK pin (active low). In order to use the BREAK pin it must be previously enabled by software setting the BPEN bit in the BREAKCR register.

When a low level is detected on the BREAK pin, the BA bit is set and the break function is activated.

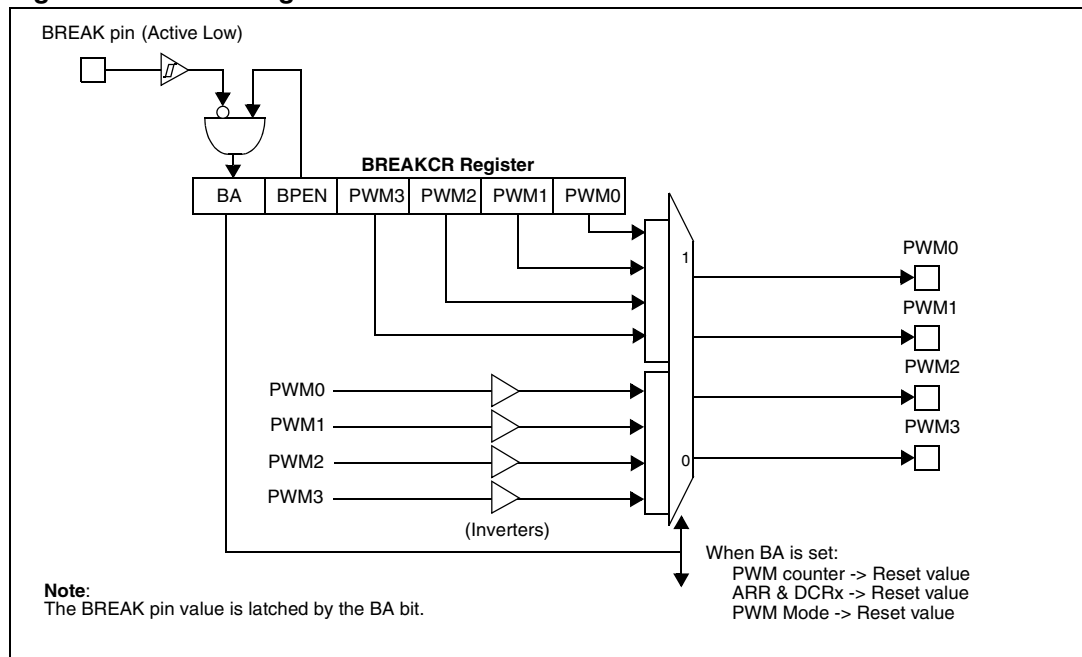
Software can set the BA bit to activate the break function without using the BREAK pin.

When the break function is activated (BA bit =1):

- The break pattern (PWM[3:0] bits in the BREAKCR) is forced directly on the PWMx output pins (after the inverter).
- The 12-bit PWM counter is set to its reset value.
- The ARR, DCRx and the corresponding shadow registers are set to their reset values.
- The PWMCR register is reset.

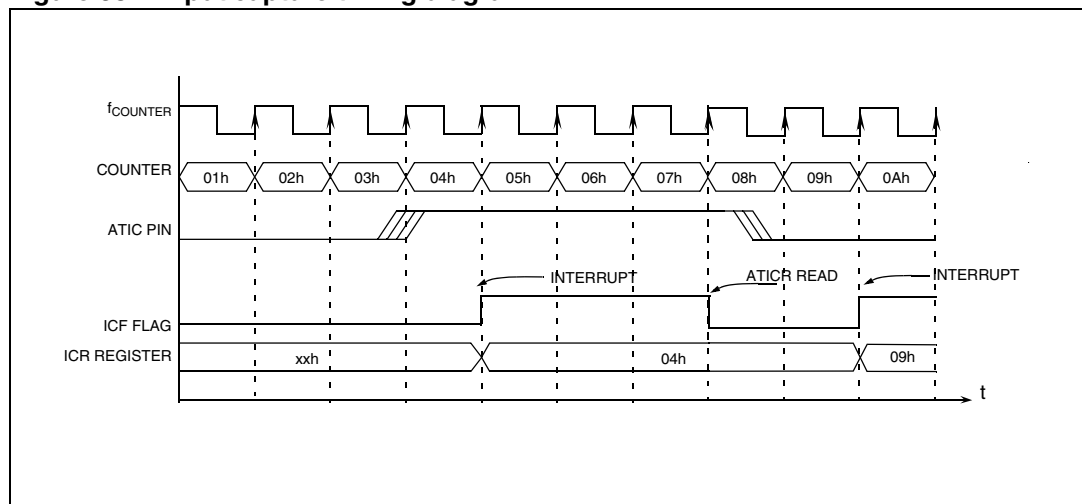
When the break function is deactivated after applying the break (BA bit goes from 1 to 0 by software):

- The control of PWM outputs is transferred to the port registers.

**Figure 37. Block diagram of break function**

### 14.3.4 Input capture

The 12-bit ATICR register is used to latch the value of the 12-bit free running upcounter 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 ATICR 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 38. Input capture timing diagram**

## 14.4 Low power modes

**Table 36. Effect of low power modes on AT2 timer**

Mode	Description
Slow	The input frequency is divided by 32
Wait	No effect on AT timer
Active-halt	AT timer halted except if CK0=1, CK1=0 and OVFIE=1
Halt	AT timer halted

## 14.5 Interrupts

**Table 37. AT2 timer interrupt control bits**

Interrupt event <sup>(1)</sup>	Event flag	Enable control bit	Exit from Wait	Exit from Halt	Exit from Active-halt
Overflow event	OVF	OVIE	Yes	No	Yes <sup>(2)</sup>
IC event	ICF	ICIE	Yes	No	No
CMP event	CMPF0	CMPIE	Yes	No	No

1. The CMP and IC events are connected to the same interrupt vector. The OVF event is mapped on a separate vector (see [Table 15: Interrupt mapping](#)). 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).

2. Only if CK0=1 and CK1=0 ( $f_{\text{COUNTER}} = f_{\text{TIMER}}$ )

## 14.6 Register description

### 14.6.1 Timer control status register (ATCSR)

Read / Write

Reset Value: 0x00 0000 (x0h)

7	6						0
0	ICF	ICIE	CK1	CK0	OVF	OVFIE	CMPIE

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 will clear 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*.

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*.

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

**Table 38. Counter clock frequency**

Counter clock selection	CK1	CK0
OFF	0	0
$f_{\text{TIMER}}$ (1 ms timebase @ 8 MHz) <sup>(1)</sup>	0	1
$f_{\text{CPU}}$	1	0
32 MHz <sup>(2)</sup>	1	1

1. PWM mode and Output Compare modes are not available at this frequency.
2. ATICR counter may return inaccurate results when read. It is therefore not recommended to use Input Capture mode at this frequency.

Bit 2 = **OVF** *Overflow Flag*.

This bit is set by hardware and cleared by software by reading the TCSR register. It indicates the transition of the counter from FFFh to ATR value.

0: No counter overflow occurred

1: Counter overflow occurred

Bit 1 = **OVFIE** *Overflow Interrupt Enable*.

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

0: OVF interrupt disabled.

1: OVF interrupt enabled.

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

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

0: CMPF interrupt disabled.

1: CMPF interrupt enabled.

## 14.6.2 Counter register high (CNTRH)

Read only

Reset Value: 0000 0000 (000h)

15				8			
0	0	0	0	CNTR 11	CNTR 10	CNTR9	CNTR8

## 14.6.3 Counter register low (CNTRL)

Read only

Reset Value: 0000 0000 (000h)

7				0			
CNTR7	CNTR6	CNTR5	CNTR4	CNTR3	CNTR2	CNTR1	CNTR0

Bits 15:12 = Reserved.

Bits 11:0 = **CNTR[11:0]** *Counter Value*.

This 12-bit register is read by software and cleared by hardware after a reset. The counter is incremented 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. The CNTRH register can be incremented between the two reads, and in order to be accurate when  $f_{\text{TIMER}}=f_{\text{CPU}}$ , the software should take this into account when CNTRL and CNTRH are read. If CNTRL is close to its highest value, CNTRH could be incremented before it is read

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

#### 14.6.4 Autoreload register (ATRH)

Read / Write

Reset Value: 0000 0000 (00h)

15				8			
0	0	0	0	ATR11	ATR10	ATR9	ATR8

#### 14.6.5 Autoreload register (ATRL)

Read / Write

Reset Value: 0000 0000 (00h)

7				0			
ATR7	ATR6	ATR5	ATR4	ATR3	ATR2	ATR1	ATR0

Bits 11:0 = **ATR[11:0]** *Autoreload Register*.

This is a 12-bit register which is written by software. The ATR register value is automatically loaded into the upcounter when an overflow occurs. The register value is used to set the PWM frequency.

#### 14.6.6 PWM output control register (PWMCR)

Read/Write

Reset Value: 0000 0000 (00h)

7				0			
0	OE3	0	OE2	0	OE1	0	OE0

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

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 after an overflow event.

1: PWM mode enabled

#### 14.6.7 PWMx control status register (PWMxCSR)

Read / Write

Reset Value: 0000 0000 (00h)

7				6		0	
0	0	0	0	0	0	OPx	CMPFx

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

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

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 DCRx register value.

0: Upcounter value does not match DCR value.

1: Upcounter value matches DCR value.

### 14.6.8 Break control register (BREAKCR)

Read/Write

Reset Value: 0000 0000 (00h)

7				0			
0	0	BA	BPEN	PWM3	PWM2	PWM1	PWM0

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

Bit 5 = **BA** Break Active.

This bit is read/write by software, cleared by hardware after reset and set by hardware when the BREAK pin is low. It activates/deactivates the Break function.

0: Break not active

1: Break active

Bit 4 = **BPEN** Break Pin Enable.

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

0: Break pin disabled

1: Break pin enabled

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

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.

### 14.6.9 PWMx duty cycle register high (DCRxH)

Read / Write

Reset Value: 0000 0000 (00h)

15				8			
0	0	0	0	DCR11	DCR10	DCR9	DCR8



### 14.6.10 PWMx duty cycle register low (DCRxL)

Read / Write

Reset Value: 0000 0000 (00h)

7							0
DCR7	DCR6	DCR5	DCR4	DCR3	DCR2	DCR1	DCR0

Bits 15:12 = Reserved.

Bits 11:0 = **DCR[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 35](#)).

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 35](#)). In Output Compare mode, they define the value to be compared with the 12-bit upcounter value.

### 14.6.11 Input capture register high (ATICRH)

Read only

Reset Value: 0000 0000 (00h)

15							8
0	0	0	0	ICR11	ICR10	ICR9	ICR8

### 14.6.12 Input capture register low (ATICRL)

Read only

Reset Value: 0000 0000 (00h)

7							0
ICR7	ICR6	ICR5	ICR4	ICR3	ICR2	ICR1	ICR0

Bits 15:12 = Reserved.

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 CNTR register when a rising or falling edge occurs on the ATIC pin. Capture will only be performed when the ICF flag is cleared.

### 14.6.13 Transfer control register (TRANCN)

Read/Write

Reset Value: 0000 0001 (01h)

7							0
0	0	0	0	0	0	0	TRAN

Bits 7:1 Reserved. Forced by hardware to 0.

Bit 0 = **TRAN** *Transfer enable*

This bit is read/write by software, cleared by hardware after each completed transfer and set by hardware after reset.

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

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

**Table 39. Register map and reset values**

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0D	<b>ATCSR</b> Reset Value	0	ICF 0	ICIE 0	CK1 0	CK0 0	OVF 0	OVFIE 0	CMPIE 0
0E	<b>CNTRH</b> Reset Value	0	0	0	0	CNTR1 1 0	CNTR1 0 0	CNTR9 0	CNTR8 0
0F	<b>CNTRL</b> Reset Value	CNTR7 0	CNTR8 0	CNTR7 0	CNTR6 0	CNTR3 0	CNTR2 0	CNTR1 0	CNTR0 0
10	<b>ATRH</b> Reset Value	0	0	0	0	ATR11 0	ATR10 0	ATR9 0	ATR8 0
11	<b>ATRL</b> Reset Value	ATR7 0	ATR6 0	ATR5 0	ATR4 0	ATR3 0	ATR2 0	ATR1 0	ATR0 0
12	<b>PWMCR</b> Reset Value	0	OE3 0	0	OE2 0	0	OE1 0	0	OE0 0
13	<b>PWM0CSR</b> Reset Value	0	0	0	0	0	0	OP0 0	CMPF0 0
14	<b>PWM1CSR</b> Reset Value	0	0	0	0	0	0	OP1 0	CMPF1 0
15	<b>PWM2CSR</b> Reset Value	0	0	0	0	0	0	OP2 0	CMPF2 0

Table 39. Register map and reset values (continued)

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
16	<b>PWM3CSR</b> Reset Value	0	0	0	0	0	0	OP3 0	CMPF3 0
17	<b>DCR0H</b> Reset Value	0	0	0	0	DCR11 0	DCR10 0	DCR9 0	DCR8 0
18	<b>DCR0L</b> Reset Value	DCR7 0	DCR6 0	DCR5 0	DCR4 0	DCR3 0	DCR2 0	DCR1 0	DCR0 0
19	<b>DCR1H</b> Reset Value	0	0	0	0	DCR11 0	DCR10 0	DCR9 0	DCR8 0
1A	<b>DCR1L</b> Reset Value	DCR7 0	DCR6 0	DCR5 0	DCR4 0	DCR3 0	DCR2 0	DCR1 0	DCR0 0
1B	<b>DCR2H</b> Reset Value	0	0	0	0	DCR11 0	DCR10 0	DCR9 0	DCR8 0
1C	<b>DCR2L</b> Reset Value	DCR7 0	DCR6 0	DCR5 0	DCR4 0	DCR3 0	DCR2 0	DCR1 0	DCR0 0
1D	<b>DCR3H</b> Reset Value	0	0	0	0	DCR11 0	DCR10 0	DCR9 0	DCR8 0
1E	<b>DCR3L</b> Reset Value	DCR7 0	DCR6 0	DCR5 0	DCR4 0	DCR3 0	DCR2 0	DCR1 0	DCR0 0
1F	<b>ATICRH</b> Reset Value	0	0	0	0	ICR11 0	ICR10 0	ICR9 0	ICR8 0
20	<b>ATICRL</b> Reset Value	ICR7 0	ICR6 0	ICR5 0	ICR4 0	ICR3 0	ICR2 0	ICR1 0	ICR0 0
21	<b>TRANC</b> Reset Value	0	0	0	0	0	0	0	TRAN 1
22	<b>BREAKCR</b> Reset Value	0	0	BA 0	BPEN 0	PWM3 0	PWM2 0	PWM1 0	PWM0 0



## 15.3 Functional description

### 15.3.1 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  $f_{OSC2}/32$ . An overflow event occurs when the counter rolls over from F9h to 00h. If  $f_{OSC2} = 8$  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.

### 15.3.2 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 LTICR1 register contains the MSB of Counter 1. An interrupt is generated if the ICIE bit is set. The ICF bit is cleared by reading the LTICR register.

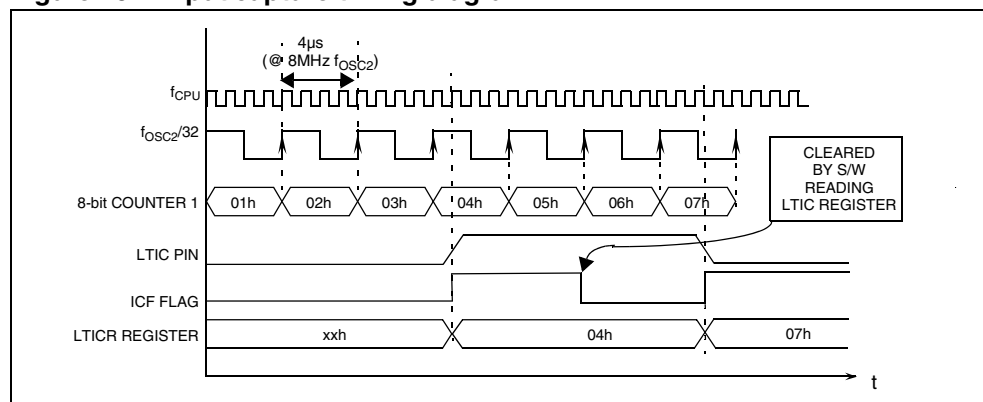
The LTICR is a read-only register and always contains the data from the last input capture. Input capture is inhibited if the ICF bit is set.

### 15.3.3 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  $f_{OSC2}/32$  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 anytime 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 40. Input capture timing diagram.**



## 15.4 Low power modes

**Table 40. Effect of low power modes on Lite timer**

Mode	Description
Slow	No effect on Lite timer (this peripheral is driven directly by $f_{OSC2}/32$ )
Wait	No effect on Lite timer
Active-halt	No effect on Lite timer
Halt	Lite timer stops counting

## 15.5 Interrupts

**Table 41. Interrupt control bits**

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	Yes	No	No
IC Event	ICF	ICIE	Yes	No	No

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

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).

## 15.6 Register description

### 15.6.1 Lite timer control/status register 2 (LTCSR2)

Read / Write

Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	TB2IE	TB2F

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

Bit 1 = TB2IE *Timebase 2 Interrupt enable*.  
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

### 15.6.2 Lite timer autoreload register (LTARR)

Read / Write

Reset Value: 0000 0000 (00h)

7							0
AR7	AR7	AR7	AR7	AR3	AR2	AR1	AR0

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.

### 15.6.3 Lite timer counter 2 (LTCNTR)

Read only

Reset Value: 0000 0000 (00h)

7							0
CNT7	CNT7	CNT7	CNT7	CNT3	CNT2	CNT1	CNT0

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.

### 15.6.4 Lite timer control/status register (LTCSR1)

Read / Write

Reset Value: 0x00 0000 (x0h)

7							0
ICIE	ICF	TB	TB1IE	TB1F	-	-	-

Bit 7 = ICIE *Interrupt Enable*.

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 initialise the ICF bit by reading the LTICR register

Bit 5 = TB *Timebase period selection*.  
This bit is set and cleared by software.

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

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

Bit 4 = TB1IE *Timebase Interrupt enable*.  
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

### 15.6.5 Lite timer input capture register (LTICR)

Read only

Reset Value: 0000 0000 (00h)

7				0			
ICR7	ICR6	ICR5	ICR4	ICR3	ICR2	ICR1	ICR0

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 42. Lite timer register map and reset values**

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
08	<b>LTCSR2</b> Reset Value	0	0	0	0	0	0	TB2IE 0	TB2F 0
09	<b>LTARR</b> Reset Value	AR7 0	AR6 0	AR5 0	AR4 0	AR3 0	AR2 0	AR1 0	AR0 0
0A	<b>LTCNTR</b> Reset Value	CNT7 0	CNT6 0	CNT5 0	CNT4 0	CNT3 0	CNT2 0	CNT1 0	CNT0 0



Table 42. Lite timer register map and reset values (continued)

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0B	<b>LTCSR1</b> Reset Value	ICIE 0	ICF x	TB 0	TB1IE 0	TB1F 0	0	0	0
0C	<b>LTICR</b> Reset Value	ICR7 0	ICR6 0	ICR5 0	ICR4 0	ICR3 0	ICR2 0	ICR1 0	ICR0 0

## 16 DALI communication module

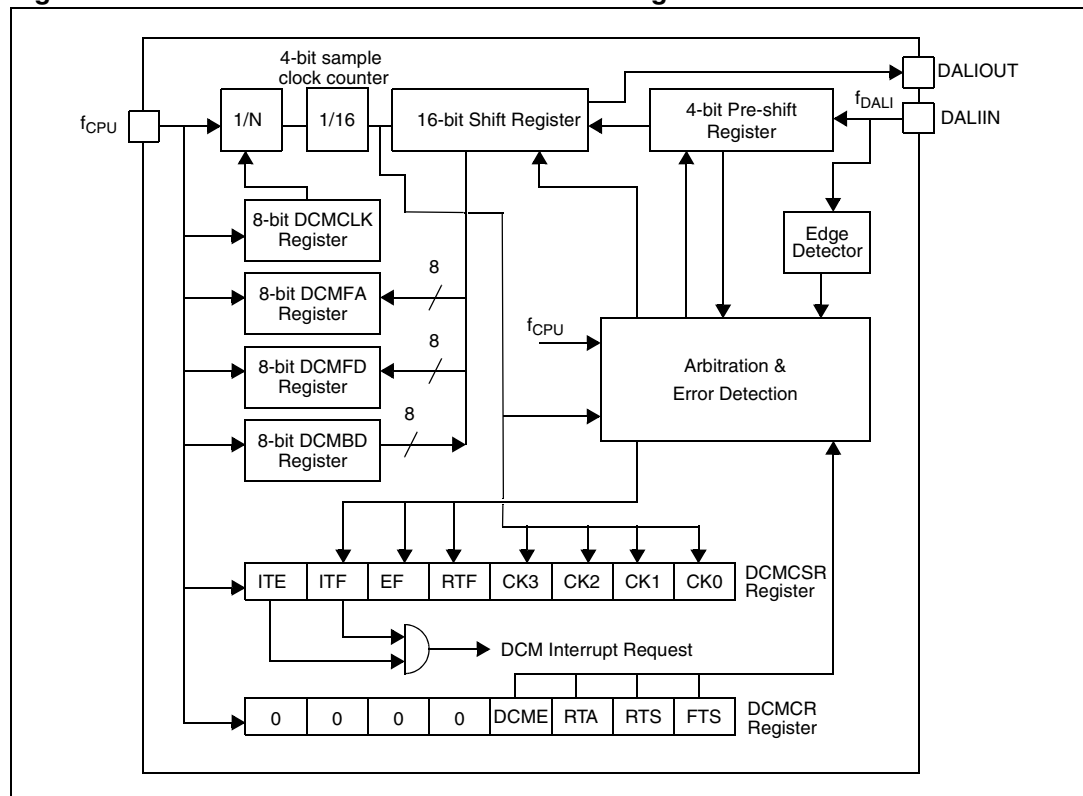
### 16.1 Introduction

The DALI Communication Module (DCM) is a serial communication circuit designed for controllable electronic ballasts. Ballasts are the devices used to provide the required starting voltage and operating current for fluorescent, mercury, or other electric-discharge lamps. The DCM supports the DALI (Digital Addressable Lighting Interface) communications standard (IEC standard).

### 16.2 Main features

- 8-bit forward address register for addressing up to 64 digital ballasts
- 1.2 kHz transmission rate  $\pm 10\%$
- 8-bit forward and backward data registers for bi-directional communications
- Maskable interrupt

**Figure 41. DALI communication module block diagram**



*Note:* The 4-bit preshift register is always active, except when in Halt mode or if the DCME bit = 0.

## 16.3 DALI standard protocol

The DALI protocol uses the bi-phase Manchester asynchronous serial data format. All the bits of the frame are bi-phase encoded except the two stop bits.

- The transmission rate is about 1.2 kHz. The bi-phase bit period is  $833.33 \mu\text{s} \pm 10\%$ .
- A forward frame consists of 19 bi-phase encoded bits:
  - 1 start bit (0→1: logical '1')
  - 1 address byte (8-bit address)
  - 1 data byte (8-bit data)
  - 2 high level stop bits (no change of phase)
- A backward frame consists of 11 bi-phase encoded bits:
  - 1 start bit (0→1: logical '1')
  - 1 data byte (8-bit data)
  - 2 high level stop bits (no change of phase)

A forward frame consists of 19 bi-phase encoded bits: 1 start bit (logical '1'), 1 address byte and 1 data byte. The frame is terminated by 2 stop bits (idle). The stop bits do not contain any change of phase.

A backward frame consists of 11 bi-phase encoded bits: 1 start bit (logical '1') and 1 data byte. The frame is terminated by 2 stop bits (idle). The stop bits do not contain any change of phase.

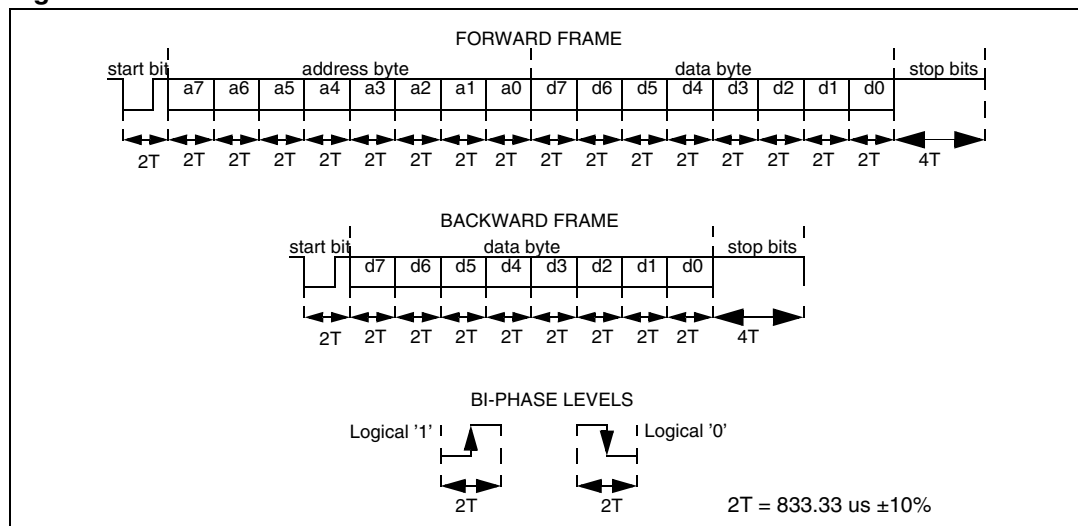
The transmission rate, expressed as a bandwidth, is specified at 1.2 kHz for the forward channel and for the backward channel.

The settling time between two subsequent forward frames is 9.17 ms (minimum).

The settling time between forward and backward frames is between 2.92 ms and 9.17 ms. If a backward frame has not been started after 9.17 ms, this is interpreted as "no answer".

In the event of code violation, the frame is ignored. After a code violation has occurred, the system is ready again for data reception.

**Figure 42. DALI standard frame**



## 16.4 General description

The DCM is able to receive or transmit a serial DALI signal using a 16-bit shift register, an edge detector, several data/control registers and arbitration logic.

The DCM receives the DALI standard signal from the lighting control network, checks for errors and loads the address/data bytes of a "forward frame" to the corresponding DCMFA/DCMFD registers and sends back the data byte of the "backward frame" (written by software to the DCMBD register) in DALI standard format.

The data rate can be changed by writing in the DCMCLK register ( $f_{\text{DATA}} = 2 * f_{\text{DALI}}$ ).

$$f_{\text{DATA}} = f_{\text{CPU}} / [(N+1) * 16]$$

The DALI standard data rate  $f_{\text{DALI}}$  is 1.2 kHz. N is the integer value of the DCMCLK register. Following the above formula, if  $f_{\text{CPU}}$  is 8 MHz, the integer value of the DCMCLK register is "207". The bi-phase bit period is 833.33 us  $\pm 10\%$ .

The polarity of the bi-phase start bit is not configurable. The start bit is a logical '1'.

The polarity of the 2 stop bits is not configurable. The 2 stop bits are set to high level.

If an error is detected during reception, the frame will be ignored and the DCM will return to Receive state.

## 16.5 Functional description

The user must write to the DCMCLK register to select the data rate according to the DALI signal frequency.

After Reset, the DCM is in Receive state and waits for the bi-phase start bit (logical '1') of the "forward frame".

The DCM checks the data format of the "forward frame" with the 4-bit pre-shift register. If an error occurs during reception, the DCM will skip the data and return to the Receive state.

If there is no error in the "forward frame", the data will be shifted Most Significant Bit-first into the 16-bit shift register. The address byte and the data byte will be loaded to the corresponding DCMFA and DCMFD registers. The DCM will send an interrupt signal by setting the ITF bit in the DCMCSR register.

If the software receives an interrupt signal from the DCM, it reads the DCMFA and DCMFD registers.

Depending on the command, the DCM is able to send back or receive data.

In an interrupt routine, the RTS bit has to be set either before or at the same time as the RTA bit.

If the software asks the DCM to send back a "backward frame", the software must first write to the DCMBD register and switch the DCM to Transmit state by setting the RTS and RTA bits in the DCMCR register during the interrupt routine. The DCMBD register will be shifted out from the 16-bit shift register in DALI format, the Most Significant Bit-first.

When the "backward frame" has been transmitted, the DCM will send an interrupt signal by setting the ITF bit in the DCMCSR register.

If the software asks the DCM to receive a "forward frame", the software must switch the DCM to Receive state by clearing the RTS bit and setting the RTA bit in the DCMCR register during the interrupt routine.

If the ITF interrupt flag is set in the DCMCSR register, the software must set the RTA bit in the DCMCR register to allow the DCM to perform the next DALI signal reception or transmission.

The DALIIN signal is always taken into account by the 4-bit pre-shifter.

## 16.6 Special functions

### 16.6.1 Forced transmission (test mode)

The DCM must receive a "forward frame" before sending back a "backward frame". But it is possible to force the DCM into Transmit state by setting the FTS bit in the DCMCR register. The DCMBD register will be shifted out in DALI format, the Most Significant Bit-first. Preferably before forcing the DCM into Transmit state, the user should reset/set the DCME bit in the DCMCR register. An interrupt flag will be generated after a forced transmission (the ITF bit in the DCMCSR register).

**Procedure:**

- Reset the DCME bit in the DCMCR register.
- Write the backward value in the DCMBD register.
- Set both the DCME and the FTS bits in the DCMCR register.
- When an interrupt is generated (end of transmission, the ITF bit is set in the DCMCSR register), set the RTA bit in the DCMCR register to re-start a transmission.
- To return to normal DALI communications, reset/set the DCME bit and reset the FTS bit in the DCMCR register.

### 16.6.2 Normal transmission

After the "forward frame" reception, the software must write the backward data byte to the DCMBD register and set both the RTS and RTA bits in the DCMCR register to start the transmission.

It is not possible to send a backward frame just after having sent a backward frame (see DALI standard protocol).

### 16.6.3 DCM enable

The user can enable or disable the DCM by writing the DCME bit in the DCMCR register. This bit is also used to reset the entire internal finite state machine.

## 16.7 DALI interface failure

If the DALI input signal is set to low level for a 2-bit period (1.66 ms), then the DCM generates an error flag by setting the EF bit in the DCMCSR register. This bit can be cleared by reading the DCMCSR register. The interface failure is detected if the DCM is in Receive state only.

## 16.8 Low power modes

**Table 43. Effect of low power modes on DCM**

Mode	Description
Wait	No effect on DCM
Halt	DCM registers are frozen
Active-halt	No effect on DCM

## 16.9 Interrupts

**Table 44. Interrupt control bits**

Interrupt event	Event flag	Enable control bit	Exit from Wait	Exit from Halt
EOT	ITF	ITE	Yes	No

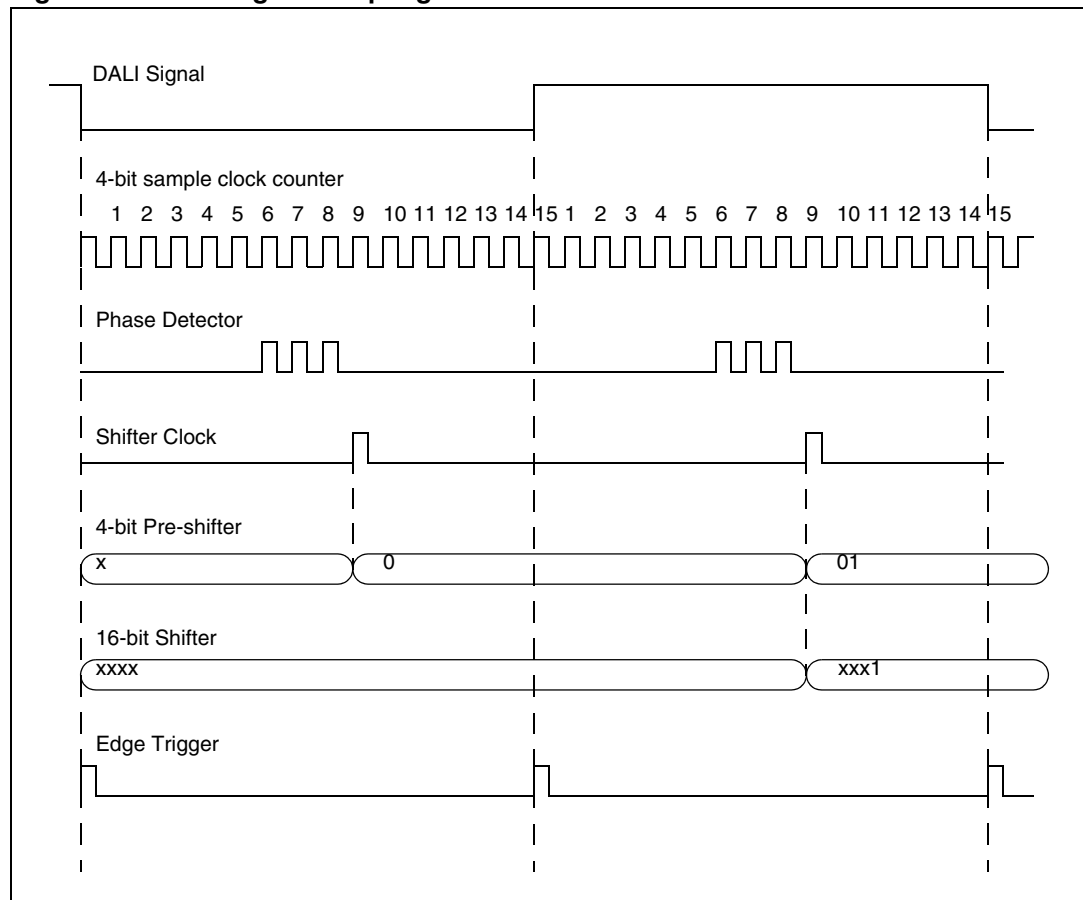
### 16.10 Bi-phase bit detection

The clock used for sampling the DALI signal is programmed by the DCMCLK register. Each bit phase is sampled 16 times. The bit phase level is determined by two of three sample clock pulses (pulses 6,7,8). The two phase levels of the bi-phase bit are shifted into the 4-bit pre-shift register at the 9th sample clock pulse.

Only the second phase level of the bi-phase bit is shifted into the 16-bit shifter.

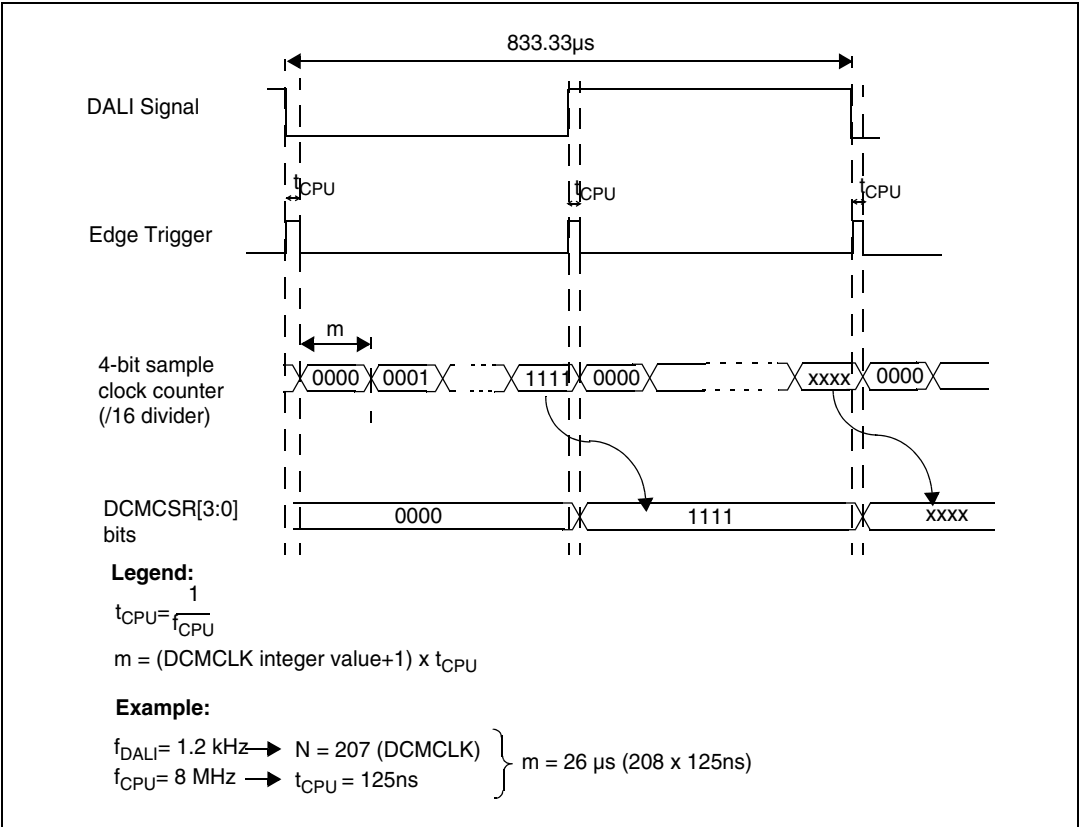
The 4-bit pre-shifter is used to detect any errors in the received frame.

When a change of phase is detected (edge trigger), the 4-bit sample clock counter (integer range 0 to15) is cleared.

**Figure 43. DALI signal sampling**

In the example shown in [Figure 44](#), the DCMCSR[3:0] bits are updated automatically at each edge trigger (DALI signal change of phase). At the same time the value of the 4-bit sample clock counter is reset. By reading the DCMCSR[3:0] bits software can detect changes in the DALI signal pulse length.

Figure 44. Example of DALI signal sampling



## 16.11 Register description

### 16.11.1 DCM data rate control register (DCMCLK)

Read / Write

Reset Value: 0000 0000 (00h)

7							0
CK7	CK6	CK5	CK4	CK3	CK2	CK1	CK0

Bits 7:0 = **DCMCLK[7:0]** *Clock Prescaler.*

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

These 8 bits are used for tuning the DALI data rate.  $f_{DATA} = f_{CPU} / [(N+1) \times 16]$  where N is the integer value of the DCMCLK register.

### 16.11.2 DCM forward address register (DCMFA)

Read only

Reset Value: 0000 0000 (00h)

7							0
FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0



Bits 7:0 = **DCMFA[7:0]** *Forward Address*.

These bits are read by software and set/cleared by hardware.

These 8 bits are used to store the "forward frame" address byte.

### 16.11.3 DCM forward data register (DCMFD)

Read only

Reset Value: 0000 0000 (00h)

7							0
FD7	FD6	FD5	FD4	FD3	FD2	FD1	FD0

Bits 7:0 = **DCMFD[7:0]** *Forward Data*.

These bits are read by software and set/cleared by hardware.

These 8 bits are used to store the "forward frame" data byte.

### 16.11.4 DCM backward data register (DCMBD)

Read / Write

Reset Value: 0000 0000 (00h)

7							0
BD7	BD6	BD5	BD4	BD3	BD2	BD1	BD0

Bits 7:0 = **DCMBD[7:0]** *Backward Data*.

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

These 8 bits are used to store the "backward frame" data byte. The software writes to this register before enabling the transmit operation.

### 16.11.5 DCM control register (DCMCR)

Read / Write

Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	DCME	RTA	RTS	FTS

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

Bit 3 = **DCME** *DALI Communication Enable*.

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

When set, it enables DALI communication. It also resets the entire internal finite state machine.

0: The DCM is not enable to receive/transmit

1: The DCM is enable to receive/transmit

Bit 2 = **RTA** *Receive/Transmit Acknowledge*.

This bit is reset by hardware after it has been set by software. It is cleared after a reset.

This bit must be set, after a first DALI frame reception or transmission, to allow the DCM to perform the next DALI communication.

0: No acknowledge

1: Acknowledge

Bit 1 = **RTS** *Receive/Transmit state*.

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

This bit must be set to '1' after a forward frame is received, if a backward frame is required.

This bit must be cleared after a backward frame is transmitted, if a forward frame is required.

0: The DCM is set to Receive state

1: The DCM is set to Transmit state

Bit 0 = **FTS** *Force Transmit state*.

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

When this bit is set, the DCM is forced into Transmit state. Preferably before forcing the

DCM into Transmit state, the user should reset and set the DCME bit in the DCMCR

register. An interrupt flag

(ITF) is generated after a forced transmission.

0: The DCM is not forced to Transmit state

1: The DCM is forced to Transmit state

### 16.11.6 DCM control/status register (DCMCSR)

Read only (except for bit 7)

Reset Value: 0000 0000 (00h)

7							0
ITE	ITF	EF	RTF	CK3	CK2	CK1	CK0

Bit 7 = **ITE** *Interrupt Enable*.

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

When set, this bit allows the generation of DALI interrupts.

0: DCM interrupt (ITF) disabled

1: DCM interrupt (ITF) enabled

Bit 6 = **ITF** *Interrupt Flag*. (Read only)

This bit is set/cleared by hardware and read by software.

This bit is set after the end of the "backward frame" transmission or the "forward frame"

reception. It is cleared by setting the RTA bit in the DCMCR register. It is set after a forced transmission (see the FTS bit).

0: Not the end of reception/transmission

1: End of reception/transmission

Bit 5 = **EF** *Error Flag*. (Read only)

This bit is set/cleared by hardware. It is cleared by reading the DCMCSR register.

This bit is set when either the DALI data format received is wrong or an interface failure is detected.

0: No data format error during reception

1: Data format error during reception

Bit 4 = **RTF** *Receive/Transmit Flag*. (Read only)

This bit is set/reset by hardware and read by software.

0: The DCM is in Transmit state

1: The DCM is in Receive state

Bits 3:0 = **DCMCSR[3:0]** *Clock counter value*. (Read only) These bits are set/cleared by hardware and read by software.

The value of the 4-bit sample clock counter (integer range 0 to 15). The clock counter value is loaded in the DCMCSR register when a DALI change of phase signal is detected (edge trigger). Refer to [Figure 44](#).

**Table 45. Register map and reset values**

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0040h	<b>DCMCLK</b> Reset Value	CK7 0	CK6 0	CK5 0	CK4 0	CK3 0	CK2 0	CK1 0	CK0 0
0041h	<b>DCMFA</b> Reset Value	FA7 0	FA6 0	FA5 0	FA4 0	FA3 0	FA2 0	FA1 0	FA0 0
0042h	<b>DCMFD</b> Reset Value	FD7 0	FD6 0	FD5 0	FD4 0	FD3 0	FD2 0	FD1 0	FD0 0
0043h	<b>DCMBD</b> Reset Value	BD7 0	BD6 0	BD5 0	BD4 0	BD3 0	BD2 0	BD1 0	BD0 0
0044h	<b>DCMCR</b> Reset Value	0	0	0	0	DCME 0	RTA 0	RTS 0	FTS 0
0045h	<b>DCMCSR</b> Reset Value	ITE 0	ITF 0	EF 0	RTF 0	CK3 0	CK2 0	CK1 0	CK0 0

## 17 Serial peripheral interface (SPI)

### 17.1 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.

### 17.2 Main features

- Full duplex synchronous transfers (on 3 lines)
- Simplex synchronous transfers (on 2 lines)
- Master or slave operation
- Six master mode frequencies ( $f_{\text{CPU}}/4$  max.)
- $f_{\text{CPU}}/2$  max. slave mode frequency (see note)
- $\overline{\text{SS}}$  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.

### 17.3 General description

Figure 45 shows the serial peripheral interface (SPI) block diagram. There are 3 registers:

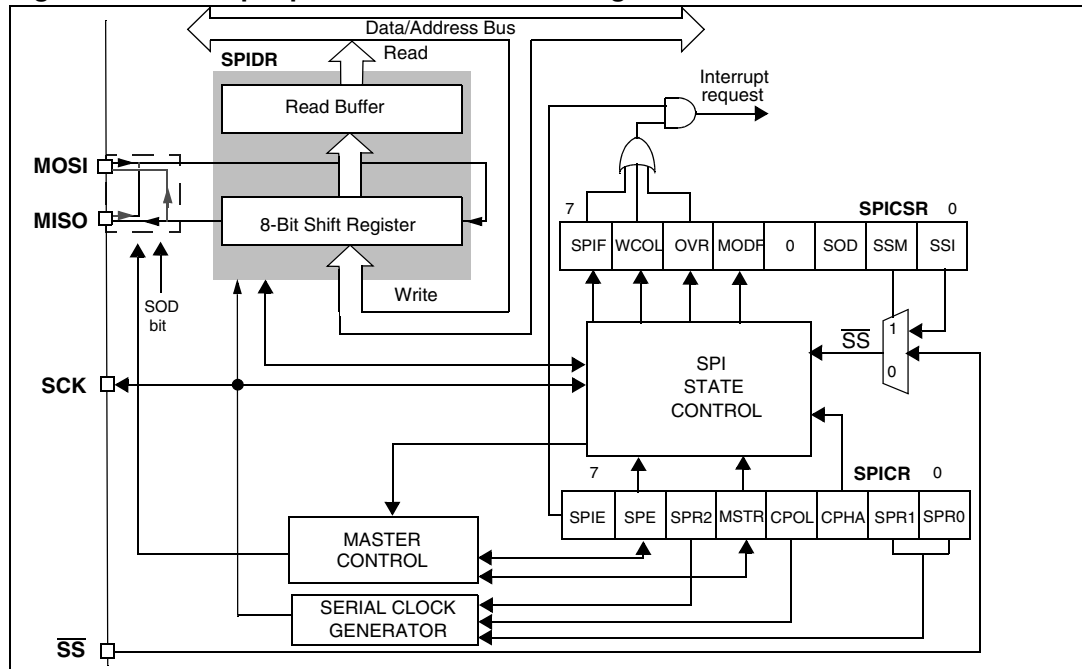
- SPI Control Register (SPICR)
- SPI Control/Status Register (SPICSR)
- SPI Data Register (SPIDR)

The SPI is connected to external devices through 3 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
- $\overline{\text{SS}}$ : 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  $\overline{\text{SS}}$  inputs can be driven by standard I/O ports on the master device.

Figure 45. Serial peripheral interface block diagram



## 17.4 Functional description

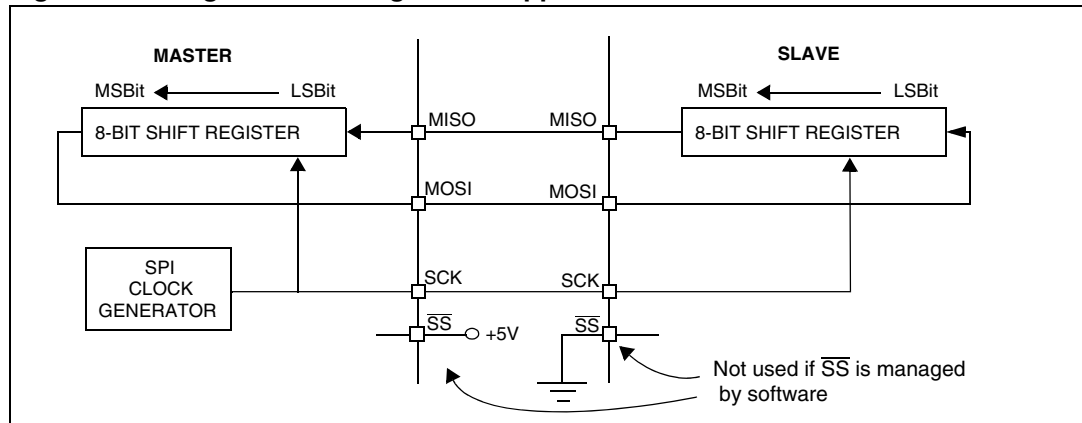
A basic example of interconnections between a single master and a single slave is illustrated in [Figure 46](#).

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 49](#)) but master and slave must be programmed with the same timing mode.

**Figure 46. Single master/ single slave application**

### 17.4.1 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 48](#))

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:

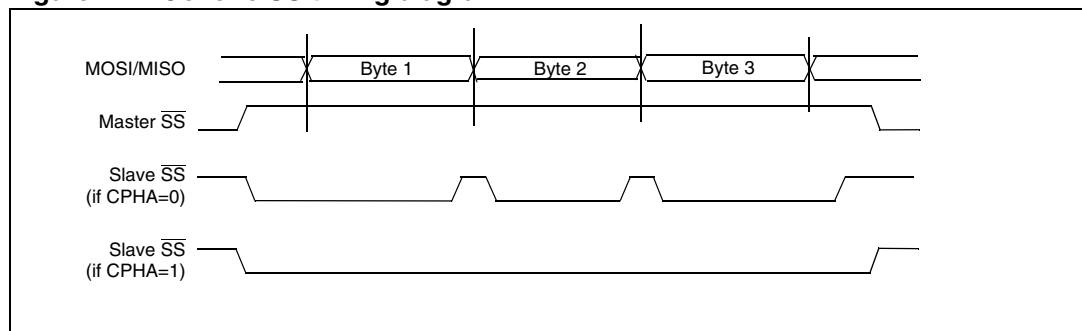
There are two cases depending on the data/clock timing relationship (see [Figure 47](#)):

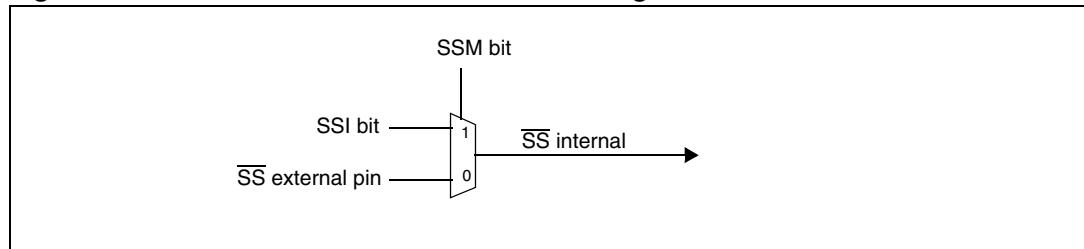
If CPHA=1 (data latched on 2nd clock edge):

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

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

- $\overline{SS}$  internal must be held low during byte transmission and pulled high between each byte to allow the slave to write to the shift register. If  $\overline{SS}$  is not pulled high, a Write Collision error will occur when the slave writes to the shift register (see [Write collision error \(WCOL\) on page 106](#)).

**Figure 47. Generic  $\overline{SS}$  timing diagram**

**Figure 48. Hardware/software slave select management**

### 17.4.2 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 (if the SPICSR register is not written first, the SPICR register setting (MSTR bit) may be not taken into account):

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 49* 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.

*Note:* **Important:** 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.

### 17.4.3 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 49](#)).

*Note:* The slave must have the same CPOL and CPHA settings as the master.

- Manage the  $\overline{SS}$  pin as described in [Section 17.4.1](#) and [Figure 47](#). 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 [Overrun condition \(OVR\)](#) on page 106).*

### 17.4.4 Clock phase and clock polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits (See [Figure 49](#)).

*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).

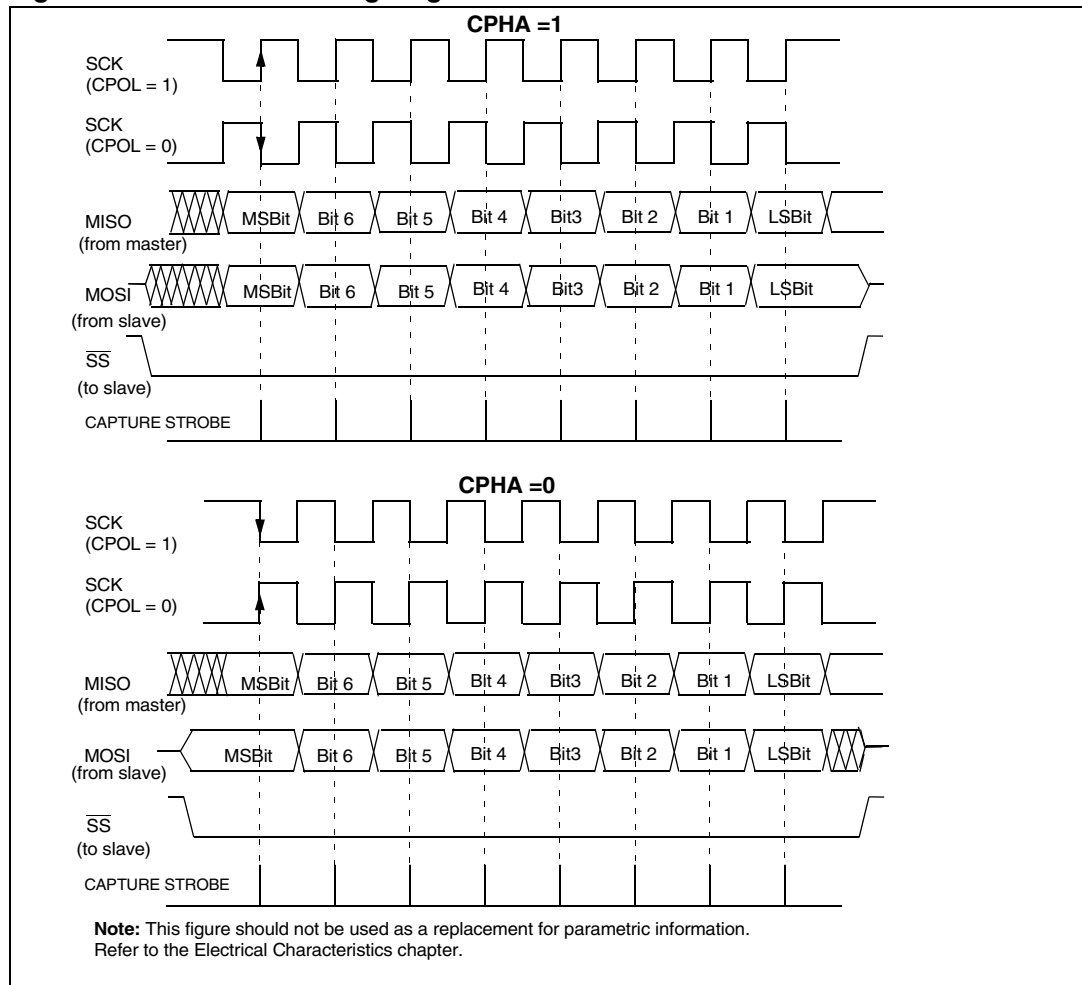
The combination of the CPOL clock polarity and CPHA (clock phase) bits selects the data capture clock edge



Figure 49, 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, 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 49. Data clock timing diagram**



## 17.4.5 Error flags

### Master mode fault (MODF)

Master mode fault occurs when the master device has its  $\overline{SS}$  pin 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 multi master configuration the Device can be in slave mode with the MODF bit set.

The MODF bit indicates that there might have been a multi-master 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.

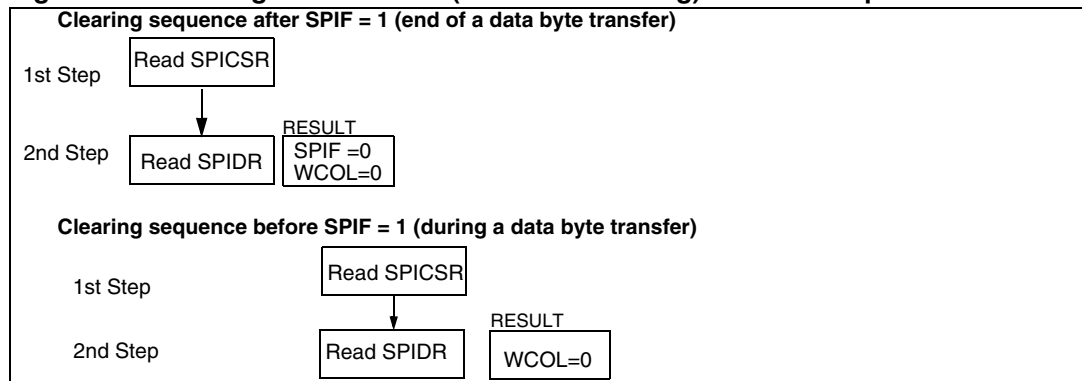
Write collisions can occur both in master and slave mode. See also [Section 17.4.1: Slave select management](#).

*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 50](#)).

**Figure 50. Clearing the WCOL bit (write collision flag) software sequence**

*Note:* Writing to the SPIDR register instead of reading it does not reset the WCOL bit.

### 17.4.6 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 51](#)).

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.

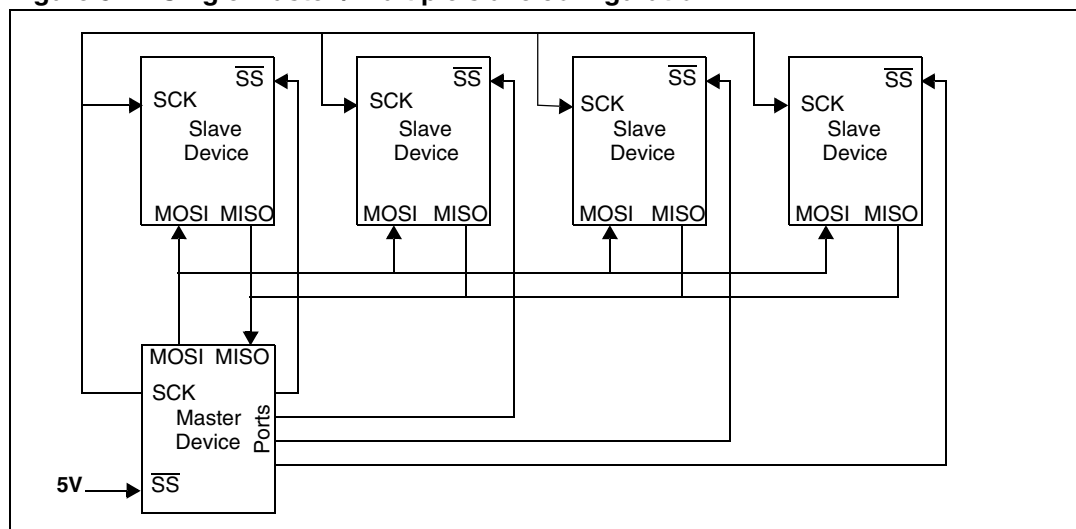
For more security, the slave device may respond to the master with the received data byte. Then the master will receive the previous byte back from the slave device if all MISO and MOSI pins are connected and the slave has not written to its SPIDR register.

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 51. Single master / multiple slave configuration**

## 17.5 Low power modes

**Table 46. Effect of low power modes on SPI**

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 fetch). If several data are received before the wake-up event, then an overrun error is generated. This error can be detected after the fetch of the interrupt routine that woke up the Device.

### 17.5.1 Using the SPI to wake-up the device from Halt mode

In slave configuration, the SPI is able to wake-up the Device from Halt mode through a SPIF interrupt. The data received is subsequently read from the SPIDR register when the software is running (interrupt vector fetch). If multiple data transfers have been performed before software clears the SPIF bit, then the OVR bit is set by hardware.

**Note:** When waking up from Halt mode, if the SPI remains in Slave mode, it is recommended to perform an extra communications cycle to bring the SPI from Halt mode state to normal state. If the SPI exits from Slave mode, it returns to normal state immediately.

**Caution:** The SPI can wake-up the Device from Halt mode only if the Slave Select signal (external  $\overline{SS}$  pin or the SSI bit in the SPICSR register) is low when the Device enters Halt mode. So if Slave selection is configured as external (see [Section 17.4.1](#)), make sure the master drives a low level on the  $\overline{SS}$  pin when the slave enters Halt mode.

## 17.6 Interrupts

**Table 47. Interrupt control bits**

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
SPI End of Transfer Event	SPIF	SPIE	Yes	Yes
Master Mode Fault Event	MODF		Yes	No
Overrun Error	OVR		Yes	No

*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).

## 17.7 Register description

### 17.7.1 Control register (SPICR)

Read/Write

Reset Value: 0000 xxxx (0xh)

7				0			
SPIE	SPE	SPR2	MSTR	CPOL	CPHA	SPR1	SPR0

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 [Master mode fault \(MODF\) on page 105](#)). 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 48: SPI master mode SCK frequency](#).

0: Divider by 2 enabled

1: Divider by 2 disabled

*Note:* 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 [Master mode fault \(MODF\) on page 105](#)).

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

*Note:* 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.

*Note:* 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.

*Note:* These 2 bits have no effect in slave mode.

**Table 48. 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

### 17.7.2 Control/status register (SPICSR)

Read/Write (some bits Read Only)

Reset Value: 0000 0000 (00h)

7							0
SPIF	WCOL	OVR	MODF	-	SOD	SSM	SSI

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.

*Note:* 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 50](#)).

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 [Overrun condition \(OVR\) on page 106](#)). 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  $\overline{SS}$  pin is pulled low in master mode (see [Master mode fault \(MODF\) on page 105](#)). 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**  $\overline{SS}$  Management.

This bit is set and cleared by software. When set, it disables the alternate function of the SPI  $\overline{SS}$  pin and uses the SSI bit value instead. See [Section 17.4.1: Slave select management](#).

0: Hardware management ( $\overline{SS}$  managed by external pin)

1: Software management (internal  $\overline{SS}$  signal controlled by SSI bit. External  $\overline{SS}$  pin free for general-purpose I/O)

Bit 0 = **SSI**  $\overline{SS}$  Internal Mode.

This bit is set and cleared by software. It acts as a 'chip select' by controlling the level of the  $\overline{SS}$  slave select signal when the SSM bit is set.

0: Slave selected

1: Slave deselected

### 17.7.3 Data I/O register (SPIDR)

Read/Write

Reset Value: Undefined

7							0
D7	D6	D5	D4	D3	D2	D1	D0

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.*

**Caution:** 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 45](#)).

**Table 49. SPI register map and reset values**

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0031h	SPIDR Reset Value	MSB x	x	x	x	x	x	x	LSB x
0032h	SPICR Reset Value	SPIE 0	SPE 0	SPR2 0	MSTR 0	CPOL x	CPHA x	SPR1 x	SPR0 x
0033h	SPICSR Reset Value	SPIF 0	WCOL 0	OVR 0	MODF 0	0	SOD 0	SSM 0	SSI 0



## 18 10-bit A/D converter (ADC)

### 18.1 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 7 multiplexed analog input channels (refer to device pin out description) that allow the peripheral to convert the analog voltage levels from up to 7 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.

### 18.2 Main features

- 10-bit conversion
- Up to 7 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 52](#).

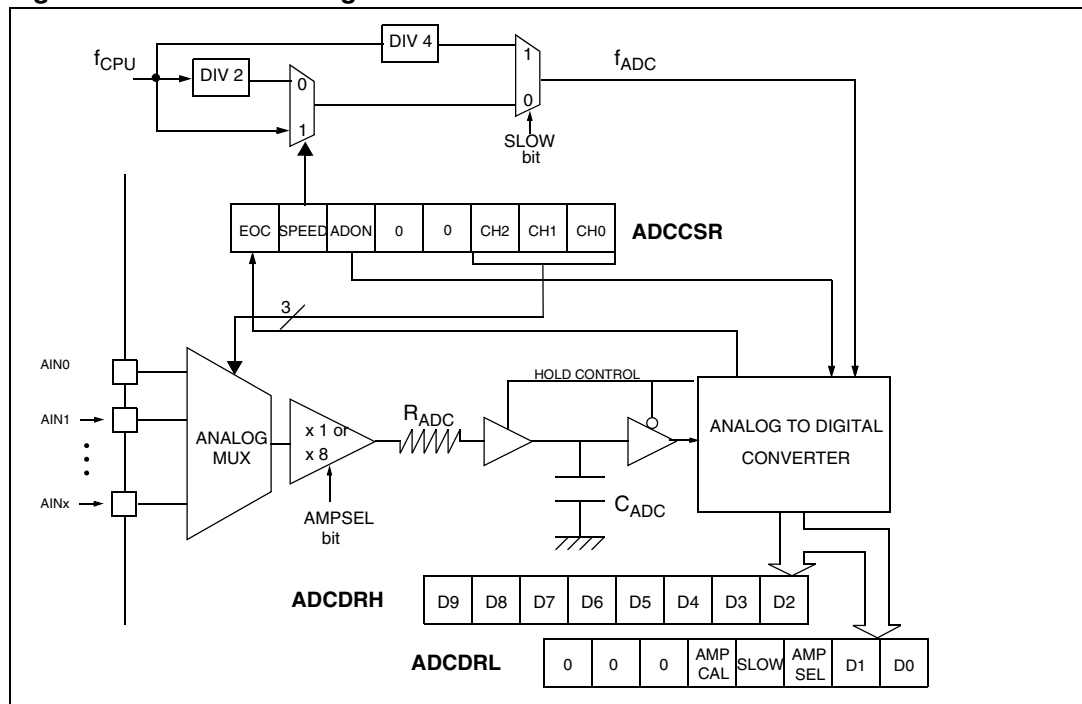
### 18.3 Functional description

#### 18.3.1 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 52. ADC block diagram



### 18.3.2 Input voltage amplifier

The input voltage can be amplified by a factor of 8 by enabling the **AMPSEL** bit in the **ADCDRL** register.

When the amplifier is enabled, the input range is 0V to  $V_{DD}/8$ .

For example, if  $V_{DD} = 5V$ , then the ADC can convert voltages in the range 0V to 430mV with an ideal resolution of 0.6mV (equivalent to 13-bit resolution with reference to a  $V_{SS}$  to  $V_{DD}$  range).

For more details, refer to the Electrical characteristics section.

**Note:** The amplifier is switched on by the **ADON** bit in the **ADCCSR** register, so no additional startup time is required when the amplifier is selected by the **AMPSEL** bit.

### 18.3.3 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 **ADCDRL** 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 **ADCDRL** and **ADCDRL** registers is 00 00h.

The A/D converter is linear and the digital result of the conversion is stored in the **ADCDRL** 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 allotted time.

### 18.3.4 A/D conversion

The analog input ports must be configured as input, no pull-up, no interrupt. Refer to [Section 12: I/O ports on page 61](#). Using these pins as analog inputs does not affect the ability of the port to be read as a logic input.

In the ADCCSR register:

- Select the CS[2:0] bits to assign the analog channel to convert.

#### ADC conversion mode

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 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.

## 18.4 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.

## 18.5 Low power modes

*Note: 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 50. Effect of low power modes on ADC**

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.

## 18.6 Interrupts

None.

## 18.7 Register description

### 18.7.1 Control/status register (ADCCSR)

Read/Write (Except bit 7 read only)

Reset Value: 0000 0000 (00h)

7							0
EOC	SPEED	ADON	0	CH3	CH2	CH1	CH0

Bit 7 = **EOC** *End of Conversion*

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*

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.

Bit 5 = **ADON** *A/D Converter on*

This bit is set and cleared by software.

0: A/D converter and amplifier are switched off

1: A/D converter and amplifier are switched on

Bits 4:3 = **Reserved**. Must be kept cleared.

Bits 2:0 = **CH[2:0]** *Channel Selection*

These bits are set and cleared by software. They select the analog input to convert.

**Table 51. Channel selection bits**

Channel pin <sup>(1)</sup>	CH2	CH1	CH0
AIN0	0	0	0
AIN1	0	0	1
AIN2	0	1	0

**Table 51. Channel selection bits (continued)**

Channel pin <sup>(1)</sup>	CH2	CH1	CH0
AIN3	0	1	1
AIN4	1	0	0
AIN5	1	0	1
AIN6	1	1	0

1. The number of channels is device dependent. Refer to [Section 4: Pin description on page 13](#).

### 18.7.2 Data register high (ADCDRH)

Read Only

Reset Value: xxxx xxxx (xxh)

7							0
D9	D8	D7	D6	D5	D4	D3	D2

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

### 18.7.3 AMP control/data register low (ADCDRL)

Read/Write

Reset Value: 0000 00xx (0xh)

7							0
0	0	0	AMP CAL	SLOW	AMPSEL	D1	D0

Bits 7:5 = Reserved. Forced by hardware to 0.

Bit 4 = **AMPCAL** *Amplifier Calibration Bit*

This bit is set and cleared by software. User is suggested to use this bit to calibrate the ADC when amplifier is ON. Setting this bit internally connects amplifier input to 0 V. Hence, corresponding ADC output can be used in software to eliminate amplifier-offset error.

0: Calibration off

1: Calibration on (The input voltage of the amp is set to 0V)

*Note: It is advised to use this bit to calibrate the ADC when the amplifier is ON. Setting this bit internally connects the amplifier input to 0v. Hence, the corresponding ADC output can be used in software to eliminate an amplifier-offset error.*

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

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

**Table 52. ADC clock speed selection**

f <sub>ADC</sub>	SLOW	SPEED
f <sub>CPU</sub> /2	0	0
f <sub>CPU</sub>	0	1
f <sub>CPU</sub> /4	1	x

This bit is set and cleared by software.

Bit 2 = **AMPSEL** Amplifier Selection Bit

0: Amplifier is not selected

1: Amplifier is selected

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

*Note:* When AMPSEL=1 it is mandatory that  $f_{ADC}$  be less than or equal to 2 MHz.

**Table 53. ADC register map and reset values**

Address (Hex.)	Register label	7	6	5	4	3	2	1	0
0034h	<b>ADCCSR</b> Reset Value	EOC 0	SPEED 0	ADON 0	0 0	0 0	CH2 0	CH1 0	CH0 0
0035h	<b>ADCDRH</b> Reset Value	D9 x	D8 x	D7 x	D6 x	D5 x	D4 x	D3 x	D2 x
0036h	<b>ADCDRL</b> Reset Value	0 0	0 0	0 0	AMPC AL 0	SLOW 0	AMPSE L 0	D1 x	D0 x

## 19 Instruction set

### 19.1 CPU addressing modes

The CPU features 17 different addressing modes which can be classified in 7 main groups (see [Table 54](#)).

**Table 54. Addressing mode groups**

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 CPU Instruction Set is designed to minimize the number of bytes required per instruction: To do so, most of the addressing modes may be divided 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 55. CPU addressing mode overview**

Mode			Syntax	Destination	Pointer address (Hex.)	Pointer size (Hex.)	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
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
Long	Indirect	Indexed	ld A,([\$10.w],X)	0000..FFFF	00..FF	word	+ 2

**Table 55. CPU addressing mode overview (continued)**

Relative	Direct		jrne loop	PC+/-127			+ 1
Relative	Indirect		jrne [\$10]	PC+/-127	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

### 19.1.1 Inherent

All Inherent instructions consist of a single byte. The opcode fully specifies all the required information for the CPU to process the operation.

**Table 56. Inherent instructions**

Instruction	Function
NOP	No Operation
TRAP	S/W Interrupt
WFI	Wait for Interrupt (low power mode)
HALT	Halt oscillator (lowest power mode)
RET	Sub-routine Return
IRET	Interrupt sub-routine Return
SIM	Set Interrupt Mask (level 3)
RIM	Reset Interrupt Mask (level 0)
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
MUL	Byte Multiplication
SLL, SRL, SRA, RLC, RRC	Shift and Rotate operations
SWAP	Swap nibbles

### 19.1.2 Immediate

Immediate instructions have two bytes: The first byte contains the opcode and the second byte contains the operand value.



**Table 57. Immediate instructions**

Instruction	Function
LD	Load
CP	Compare
BCP	Bit Compare
AND, OR, XOR	Logical operations
ADC, ADD, SUB, SBC	Arithmetic operations

### 19.1.3 Direct

In Direct instructions, the operands are referenced by their memory address. The direct addressing mode consists of two submodes:

#### Direct (short)

The address is a byte, thus requiring only one byte after the opcode, but only allows 00 - FF addressing space.

#### Direct (long)

The address is a word, thus allowing 64 Kbyte addressing space, but requires 2 bytes after the opcode.

### 19.1.4 Indexed (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 indexed addressing mode consists of three submodes:

#### Indexed (no offset)

There is no offset, (no extra byte after the opcode), and it allows 00 - FF addressing space.

#### Indexed (short)

The offset is a byte, thus requiring only one byte after the opcode and allows 00 - 1FE addressing space.

#### Indexed (long)

The offset is a word, thus allowing 64 Kbyte addressing space and requires 2 bytes after the opcode.

### 19.1.5 Indirect (short, long)

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

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

**Indirect (short)**

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

**Indirect (long)**

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

**19.1.6 Indirect indexed (short, long)**

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

The indirect indexed addressing mode consists of two submodes:

**Indirect indexed (short)**

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

**Indirect indexed (long)**

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

**Table 58. Instructions supporting direct, indexed, indirect and indirect indexed addressing modes**

Instructions		Function
Long and short	LD	Load
	CP	Compare
	AND, OR, XOR	Logical operations
	ADC, ADD, SUB, SBC	Arithmetic Additions/Subtractions operations
	BCP	Bit Compare
Short 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 sub-routine

### 19.1.7 Relative mode (direct, indirect)

This addressing mode is used to modify the PC register value, by adding an 8-bit signed offset to it.

**Table 59. Relative direct and indirect instructions and functions**

Available relative direct/indirect instructions	Function
JRxx	Conditional Jump
CALLR	Call Relative

The relative addressing mode consists of two submodes:

#### Relative (direct)

The offset follows the opcode.

#### Relative (indirect)

The offset is defined in the memory, the address of which follows the opcode.

## 19.2 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 60. Instruction groups**

Group	Instructions							
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 Rotates	SLL	SRL	SRA	RLC	RRC	SWAP	SLA	
Unconditional Jump or Call	JRA	JRT	JRF	JP	CALL	CALLR	NOP	RET
Conditional Branch	JRxx							
Interrupt management	TRAP	WFI	HALT	IRET				
Condition Code Flag modification	SIM	RIM	SCF	RCF				

### Using a prebyte

The instructions are described with one to four opcodes.

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:

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 the 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.

### Illegal Opcode Reset

In order to provide enhanced robustness to the device against unexpected behavior, a system of illegal opcode detection is implemented. If a code to be executed does not correspond to any opcode or prebyte value, a reset is generated. This, combined with the Watchdog, allows the detection and recovery from an unexpected fault or interference.

*Note: A valid prebyte associated with a valid opcode forming an unauthorized combination does not generate a reset.*

Table 61. Instruction set overview

Mnemo	Description	Function/example	Dst	Src	I1	H	I0	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	$A = A \cdot M$	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 sub-routine									
CALLR	Call sub-routine 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				1		0			
IRET	Interrupt routine return	Pop CC, A, X, PC			I1	H	I0	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. INT pin = 1	(ext. INT pin high)								
JRIL	Jump if ext. INT pin = 0	(ext. INT pin low)								
JRH	Jump if H = 1	H = 1 ?								
JRNH	Jump if H = 0	H = 0 ?								
JRM	Jump if I1:0 = 11	I1:0 = 11 ?								
JRNM	Jump if I1:0 <> 11	I1:0 <> 11 ?								
JRMI	Jump if N = 1 (minus)	N = 1 ?								
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 >=								

Table 61. Instruction set overview (continued)

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

## 20 Electrical characteristics

### 20.1 Parameter conditions

Unless otherwise specified, all voltages are referred to  $V_{SS}$ .

#### 20.1.1 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  $T_A=25^{\circ}\text{C}$  and  $T_A=T_A\text{max}$  (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 ( $\text{mean} \pm 3\sigma$ ).

#### 20.1.2 Typical values

Unless otherwise specified, typical data are based on  $T_A=25^{\circ}\text{C}$ ,  $V_{DD}=5\text{V}$  (for the  $4.5\text{V} \leq V_{DD} \leq 5.5\text{V}$  voltage range) and  $V_{DD}=3.3\text{V}$  (for the  $3\text{V} \leq V_{DD} \leq 4\text{V}$  voltage range). They are given only as design guidelines and are not tested.

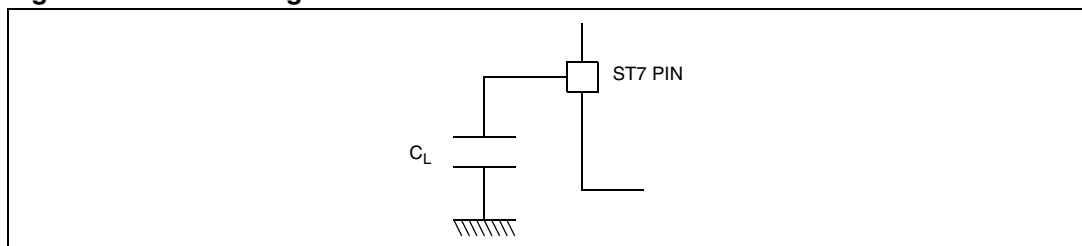
#### 20.1.3 Typical curves

Unless otherwise specified, all typical curves are given only as design guidelines and are not tested.

#### 20.1.4 Loading capacitor

The loading conditions used for pin parameter measurement are shown in [Figure 53](#).

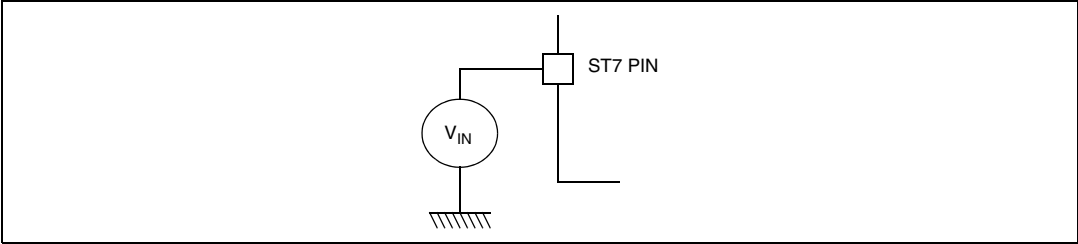
**Figure 53. Pin loading conditions**



#### 20.1.5 Pin input voltage

The input voltage measurement on a pin of the device is described in [Figure 54](#).

Figure 54. Pin input voltage



20.2 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 62. Voltage characteristics

Symbol	Ratings	Maximum value	Unit
$V_{DD} - V_{SS}$	Supply voltage <sup>(1)</sup>	7.0	V
$V_{IN}$	Input voltage on any pin <sup>(2)</sup>	$V_{SS}-0.3$ to $V_{DD}+0.3$	
$V_{ESD(HBM)}$	Electrostatic discharge voltage (Human Body Model)	see <a href="#">Section 20.7.3 on page 142</a>	V

- 1. All power ( $V_{DD}$ ) and ground ( $V_{SS}$ ) lines must always be connected to the external supply.
- 2. Directly connecting the I/O pins to  $V_{DD}$  or  $V_{SS}$  could damage the device if 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: 10kW 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.



**Table 63. Current characteristics**

Symbol	Ratings	Maximum value	Unit
$I_{VDD}$	Total current into $V_{DD}$ power lines (source)	75	mA
$I_{VSS}$	Total current out of $V_{SS}$ ground lines (sink)	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	
$I_{INJ(PIN)}^{(1) \text{ \& } (2)}$	Injected current on $\overline{RESET}$ pin	$\pm 5$	
	Injected current on OSC1 and OSC2 pins	$\pm 5$	
	Injected current on PB0 and PB1 pins <sup>(3)</sup>	+5	
	Injected current on any other pin <sup>(4)</sup>	$\pm 5$	
$\Sigma I_{INJ(PIN)}^{(2)}$	Total injected current (sum of all I/O and control pins) <sup>(4)</sup>	$\pm 20$	

1.  $I_{INJ(PIN)}$  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
2. 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 1.6mA. In addition, it is recommended to inject the current as far as possible from the analog input pins.
3. No negative current injection allowed on PB0 and PB1 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 64. Thermal characteristics**

Symbol	Ratings	Value	Unit
$T_{STG}$	Storage temperature range	-65 to +150	°C
$T_J$	Maximum junction temperature (see <a href="#">Table 94 on page 160</a> )		

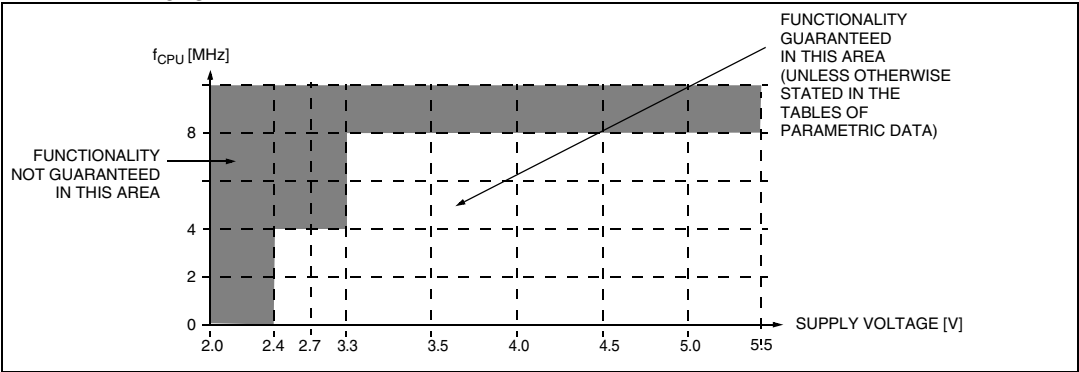
20.3 Operating conditions

T<sub>A</sub> = -40 to +85°C unless otherwise specified.

Table 65. General operating conditions

Symbol	Parameter	Conditions	Min	Max	Unit
V <sub>DD</sub>	Supply voltage	f <sub>CPU</sub> = 4 MHz. max.,	2.4	5.5	V
		f <sub>CPU</sub> = 8 MHz. max.	3.3	5.5	
f <sub>CPU</sub>	CPU clock frequency	3.3V≤ V <sub>DD</sub> ≤5.5V	up to 8		MHz
		2.4V≤V <sub>DD</sub> <3.3V	up to 4		

Figure 55. f<sub>CPU</sub> Maximum operating frequency versus V<sub>DD</sub> supply voltage



### 20.3.1 Operating conditions with low voltage detector (LVD)

$T_A = -40$  to  $85^\circ\text{C}$ , unless otherwise specified

**Table 66. Power on/power down operating conditions**

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{IT+(LVD)}$	Reset release threshold ( $V_{DD}$ rise)	High Threshold Med. Threshold Low Threshold	4.00 <sup>(1)</sup> 3.40 <sup>(1)</sup> 2.65 <sup>(1)</sup>	4.25 3.60 2.90	4.50 3.80 3.15	V
$V_{IT-(LVD)}$	Reset generation threshold ( $V_{DD}$ fall)	High Threshold Med. Threshold Low Threshold	3.80 3.20 2.40	4.05 3.40 2.70	4.30 <sup>(1)</sup> 3.65 <sup>(1)</sup> 2.90 <sup>(1)</sup>	
$V_{hys}$	LVD voltage threshold hysteresis	$V_{IT+(LVD)} - V_{IT-(LVD)}$		200		mV
$V_{tPOR}$	$V_{DD}$ rise time rate <sup>(2)(3)</sup>		20		20000	$\mu\text{s/V}$
$t_{g(VDD)}$	Filtered glitch delay on $V_{DD}$	Not detected by the LVD			150	ns
$I_{DD(LVD)}$	LVD/AVD current consumption			220		$\mu\text{A}$

1. Not tested in production.

2. Not tested in production. The  $V_{DD}$  rise time rate condition is needed to insure a correct device power-on and LVD reset. When the  $V_{DD}$  slope is outside these values, the LVD may not ensure a proper reset of the MCU.

3. 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 0V to ensure optimum restart conditions. Refer to circuit example in [Figure 87 on page 150](#) and note 4.

### 20.3.2 Auxiliary Voltage Detector (AVD) Thresholds

$T_A = -40$  to  $85^\circ\text{C}$ , unless otherwise specified

**Table 67. AVD thresholds**

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{IT+(AVD)}$	1=>0 AVDF flag toggle threshold ( $V_{DD}$ rise)	High Threshold Med. Threshold Low Threshold	4.40 <sup>(1)</sup> 3.90 <sup>(1)</sup> 3.20 <sup>(1)</sup>	4.70 4.10 3.40	5.00 4.30 3.60	V
$V_{IT-(AVD)}$	0=>1 AVDF flag toggle threshold ( $V_{DD}$ fall)	High Threshold Med. Threshold Low Threshold	4.30 3.70 2.90	4.60 3.90 3.20	4.90 <sup>(1)</sup> 4.10 <sup>(1)</sup> 3.40 <sup>(1)</sup>	
$V_{hys}$	AVD voltage threshold hysteresis	$V_{IT+(AVD)} - V_{IT-(AVD)}$		150		mV
$\Delta V_{IT-}$	Voltage drop between AVD flag set and LVD reset activation	$V_{DD}$ fall		0.45		V

1. Not tested in production.

### 20.3.3 Internal RC oscillator and PLL

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

**Table 68. Operating voltage and startup time**

Symbol	Parameter	Min	Typ	Max	Unit
$V_{DD(RC)}$	Internal RC Oscillator operating voltage	2.4		5.5	V
$V_{DD(x4PLL)}$	x4 PLL operating voltage	2.4		3.3	
$V_{DD(x8PLL)}$	x8 PLL operating voltage	3.3		5.5	
$t_{STARTUP}$	PLL Startup time		60		PLL input clock ( $f_{PLL}$ ) cycles

The RC oscillator and PLL characteristics are temperature-dependent and are grouped in four tables.

**Table 69. RC oscillator and PLL characteristics (tested for  $T_A = -40$  to  $+85^\circ\text{C}$ ) @  $V_{DD} = 4.5$  to  $5.5\text{ V}$** 

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$f_{RC}^{1)}$	Internal RC oscillator frequency <sup>(1)</sup>	RCCR = FF (reset value), $T_A=25^\circ\text{C}$ , $V_{DD}=5\text{ V}$		760		kHz
		RCCR = RCCR0 <sup>(2)</sup> , $T_A=25^\circ\text{C}$ , $V_{DD}=5\text{ V}$		1000		
$ACC_{RC}$	Accuracy of Internal RC oscillator with RCCR=RCCR0 <sup>(2)</sup>	$T_A=25^\circ\text{C}$ , $V_{DD}=4.5$ to $5.5\text{ V}$	-1		+1	%
		$T_A=-40$ to $+85^\circ\text{C}$ , $V_{DD}=5\text{ V}$	-5		+2	%
		$T_A=0$ to $+85^\circ\text{C}$ , $V_{DD}=4.5$ to $5.5\text{ V}$	-2 <sup>(3)</sup>		+2 <sup>(3)</sup>	%
$I_{DD(RC)}$	RC oscillator current consumption	$T_A=25^\circ\text{C}$ , $V_{DD}=5\text{ V}$		970 <sup>(3)</sup>		$\mu\text{A}$
$t_{su(RC)}$	RC oscillator setup time	$T_A=25^\circ\text{C}$ , $V_{DD}=5\text{ V}$			10 <sup>(2)</sup>	$\mu\text{s}$
$f_{PLL}$	x8 PLL input clock			1 <sup>(3)</sup>		MHz
$t_{LOCK}$	PLL Lock time <sup>(4)</sup>			2		ms
$t_{STAB}$	PLL Stabilization time <sup>(4)</sup>			4		ms
$ACC_{PLL}$	x8 PLL Accuracy	$f_{RC} = 1\text{ MHz}$ @ $T_A=25^\circ\text{C}$ , $V_{DD}=4.5$ to $5.5\text{ V}$		0.1 <sup>(5)</sup>		%
		$f_{RC} = 1\text{ MHz}$ @ $T_A=-40$ to $+85^\circ\text{C}$ , $V_{DD}=5\text{ V}$		0.1 <sup>(5)</sup>		%
$t_{w(JIT)}$	PLL jitter period	$f_{RC} = 1\text{ MHz}$		125 <sup>(6)</sup>		$\mu\text{s}$
$JIT_{PLL}$	PLL jitter ( $\Delta f_{CPU}/f_{CPU}$ )			1 <sup>(6)</sup>		%
$I_{DD(PLL)}$	PLL current consumption	$T_A=25^\circ\text{C}$		600 <sup>(3)</sup>		$\mu\text{A}$

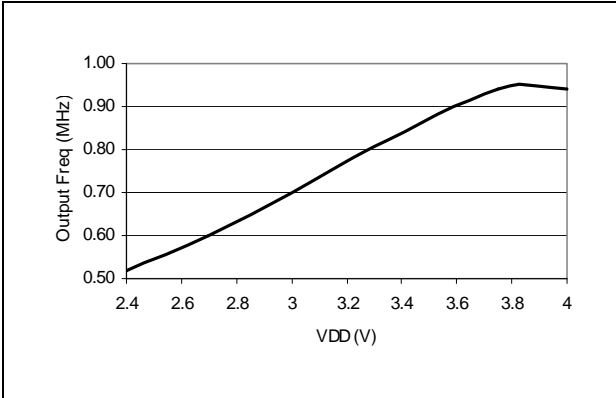
1. If the RC oscillator clock is selected, to improve clock stability and frequency accuracy, it is recommended to place a decoupling capacitor, typically 100nF, between the  $V_{DD}$  and  $V_{SS}$  pins as close as possible to the ST7 device.
2. See [Section 9.2: Internal RC oscillator adjustment on page 32](#)
3. Data based on characterization results, not tested in production
4. After the LOCKED bit is set  $ACC_{PLL}$  is max. 10% until  $t_{STAB}$  has elapsed. See [Figure 11 on page 34](#).
5. Averaged over a 4ms period. After the LOCKED bit is set, a period of  $t_{STAB}$  is required to reach  $ACC_{PLL}$  accuracy.
6. Guaranteed by design.

**Table 70. RC oscillator and PLL characteristics (tested for  $T_A = -40$  to  $+85^\circ\text{C}$ ) @  $V_{DD} = 2.7$  to  $3.3\text{ V}$** 

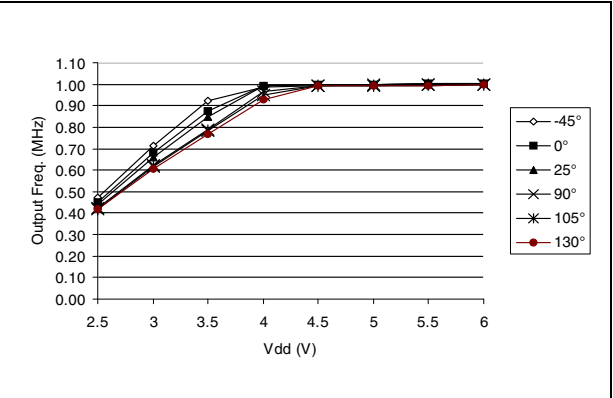
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$f_{RC}^{(1)}$	Internal RC oscillator frequency	RCCR = FF (reset value), $T_A=25^\circ\text{C}$ , $V_{DD}=3.0\text{ V}$		560		kHz
		RCCR=RCCR1 <sup>(2)</sup> , $T_A=25^\circ\text{C}$ , $V_{DD}=3\text{ V}$		700		
$ACC_{RC}$	Accuracy of Internal RC oscillator when calibrated with RCCR=RCCR1 <sup>(3)(2)</sup>	$T_A=25^\circ\text{C}$ , $V_{DD}=3\text{ V}$	-2		+2	%
		$T_A=25^\circ\text{C}$ , $V_{DD}=2.7$ to $3.3\text{ V}$	-25		+25	%
		$T_A=-40$ to $+85^\circ\text{C}$ , $V_{DD}=3\text{ V}$	-15		15	%
$I_{DD(RC)}$	RC oscillator current consumption	$T_A=25^\circ\text{C}$ , $V_{DD}=3\text{ V}$		700 <sup>(3)</sup>		$\mu\text{A}$
$t_{su(RC)}$	RC oscillator setup time	$T_A=25^\circ\text{C}$ , $V_{DD}=3\text{ V}$			10 <sup>(2)</sup>	$\mu\text{s}$
$f_{PLL}$	x4 PLL input clock			0.7 <sup>(3)</sup>		MHz
$t_{LOCK}$	PLL Lock time <sup>(4)</sup>			2		ms
$t_{STAB}$	PLL Stabilization time <sup>(4)</sup>			4		ms
$ACC_{PLL}$	x4 PLL Accuracy	$f_{RC} = 1\text{ MHz}$ @ $T_A=25^\circ\text{C}$ , $V_{DD}=2.7$ to $3.3\text{ V}$		0.1 <sup>(5)</sup>		%
		$f_{RC} = 1\text{ MHz}$ @ $T_A=40$ to $+85^\circ\text{C}$ , $V_{DD}=3\text{ V}$		0.1 <sup>(5)</sup>		%
$t_{w(JIT)}$	PLL jitter period	$f_{RC} = 1\text{ MHz}$		125 <sup>(6)</sup>		$\mu\text{s}$
$JIT_{PLL}$	PLL jitter ( $\Delta f_{CPU}/f_{CPU}$ )			1 <sup>(6)</sup>		%
$I_{DD(PLL)}$	PLL current consumption	$T_A=25^\circ\text{C}$		190 <sup>(3)</sup>		$\mu\text{A}$

1. If the RC oscillator clock is selected, to improve clock stability and frequency accuracy, it is recommended to place a decoupling capacitor, typically 100nF, between the  $V_{DD}$  and  $V_{SS}$  pins as close as possible to the ST7 device.
2. See [Section 9.2: Internal RC oscillator adjustment on page 32](#).
3. Data based on characterization results, not tested in production
4. After the LOCKED bit is set  $ACC_{PLL}$  is max. 10% until  $t_{STAB}$  has elapsed. See [Figure 11 on page 34](#).
5. Averaged over a 4ms period. After the LOCKED bit is set, a period of  $t_{STAB}$  is required to reach  $ACC_{PLL}$  accuracy
6. Guaranteed by design.

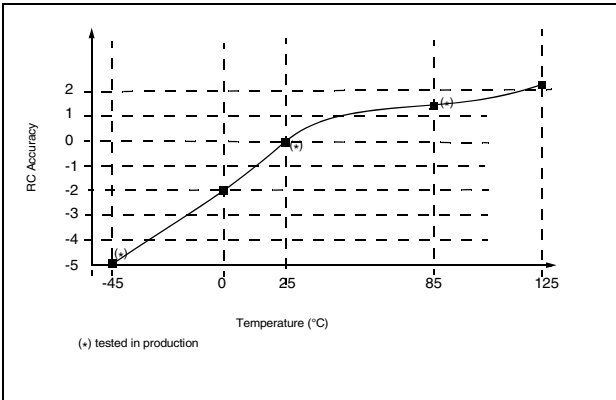
**Figure 56. RC Osc Freq vs  $V_{DD}$  @  $T_A=25^\circ\text{C}$   
(Calibrated with RCCR1: 3V @  $25^\circ\text{C}$ )**



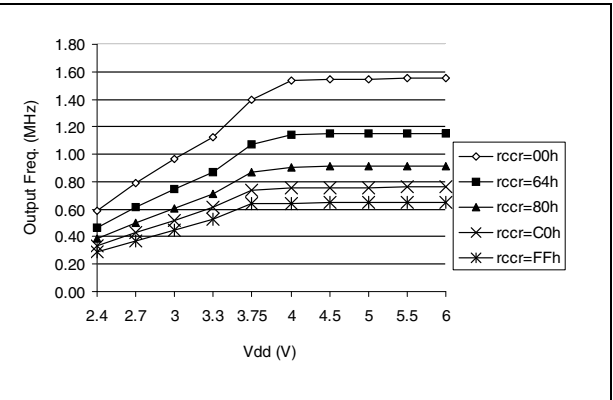
**Figure 57. RC Osc Freq vs  $V_{DD}$   
(Calibrated with RCCR0: 5V @  $25^\circ\text{C}$ )**



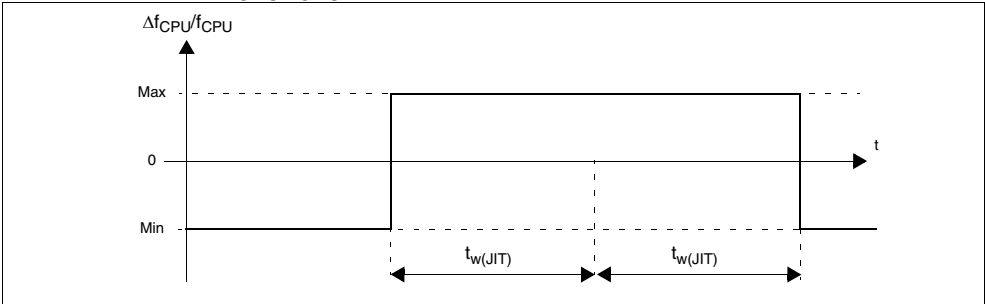
**Figure 58. Typical RC oscillator Accuracy vs temperature @  $V_{DD}=5\text{V}$   
(Calibrated with RCCR0: 5V @  $25^\circ\text{C}$ )**



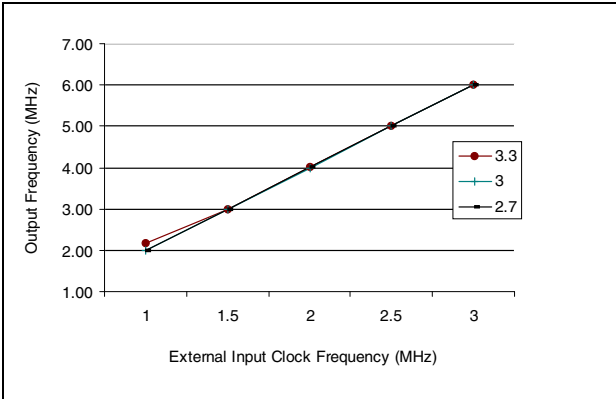
**Figure 59. RC Osc Freq vs  $V_{DD}$  and RCCR Value**



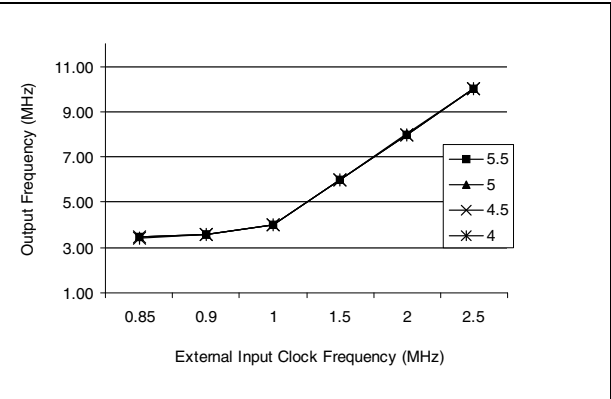
**Figure 60. PLL  $\Delta f_{\text{CPU}}/f_{\text{CPU}}$  versus time**



**Figure 61. PLLx4 Output vs CLKIN frequency**  
( $f_{osc} = f_{CLKIN}/2 \cdot PLL4$ )



**Figure 62. PLLx8 Output vs CLKIN frequency**  
( $f_{osc} = f_{CLKIN}/2 \cdot PLL8$ )



$T_A = -40$  to  $85^\circ\text{C}$ , unless otherwise specified

**Table 71. 32 MHz PLL**

Symbol	Parameter	Min	Typ	Max	Unit
$V_{DD}$	Voltage <sup>(1)</sup>	4.5	5	5.5	V
$f_{PLL32}$	Frequency <sup>(1)</sup>		32		MHz
$f_{INPUT}$	Input Frequency	7	8	9	MHz

1. 32 MHz is guaranteed within this voltage range.

## 20.4 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).

### 20.4.1 Supply current

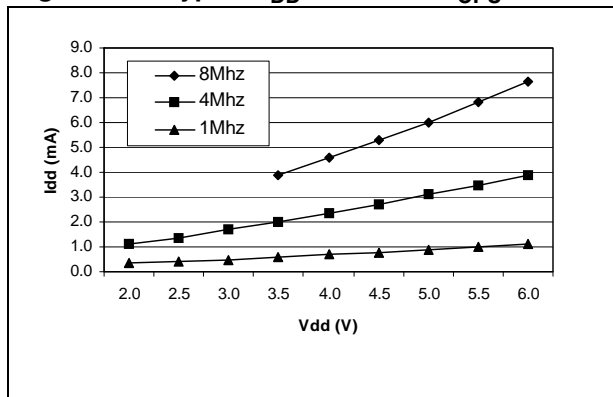
$T_A = -40$  to  $+85^\circ\text{C}$  unless otherwise specified,  $V_{DD}=5.5\text{ V}$

**Table 72. Supply current characteristics**

Symbol	Parameter	Conditions	Typ	Max	Unit
$I_{DD}$	Supply current in RUN mode	External Clock, $f_{CPU}=1\text{ MHz}$ <sup>(1)</sup>	1		mA
		Internal RC, $f_{CPU}=1\text{ MHz}$	2.2		
		$f_{CPU}=8\text{ MHz}$ <sup>(1)</sup>	7.5	12	
	Supply current in Wait mode	External Clock, $f_{CPU}=1\text{ MHz}$ <sup>(2)</sup>	0.8		
		Internal RC, $f_{CPU}=1\text{ MHz}$	1.8		
		$f_{CPU}=8\text{ MHz}$ <sup>(2)</sup>	3.7	6	
	Supply current in Slow mode	$f_{CPU}=250\text{ kHz}$ <sup>(3)</sup>	1.6	2.5	
	Supply current in Slow Wait mode	$f_{CPU}=250\text{ kHz}$ <sup>(4)</sup>	1.6	2.5	
	Supply current in Halt mode <sup>(5)</sup>	$-40^\circ\text{C} \leq T_A \leq +85^\circ\text{C}$	1	10	$\mu\text{A}$
		$T_A = +125^\circ\text{C}$	15	50	
	Supply current in AWUFH mode <sup>(6)(7)</sup>	$T_A = +25^\circ\text{C}$	20	30	

- 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.
- 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.
- Slow mode selected with  $f_{CPU}$  based on  $f_{OSC2}$  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.
- Slow-Wait mode selected with  $f_{CPU}$  based on  $f_{OSC2}$  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.
- 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.
- 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.
- This consumption refers to the Halt period only and not the associated run period which is software dependent.

**Figure 63. Typical  $I_{DD}$  in RUN vs.  $f_{CPU}$**



**Figure 64. Typical  $I_{DD}$  in Slow vs.  $f_{CPU}$**

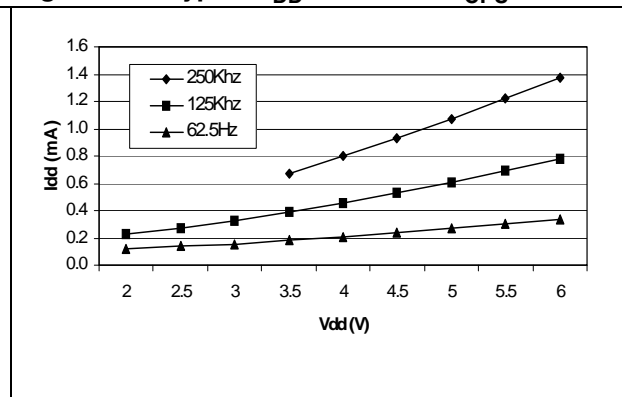




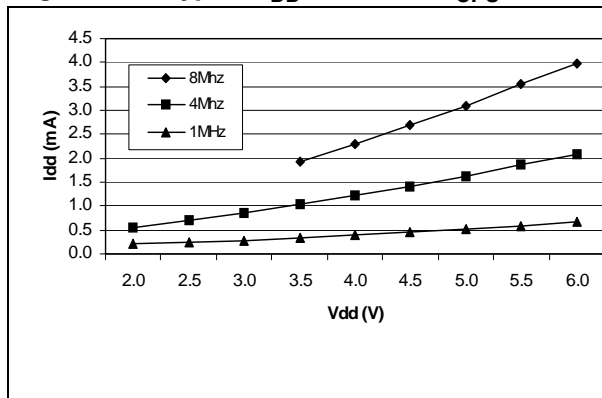
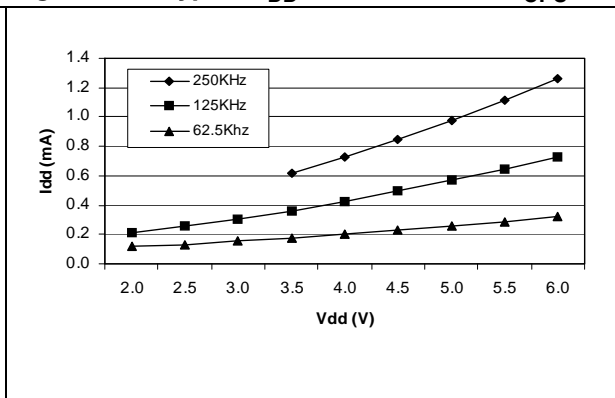
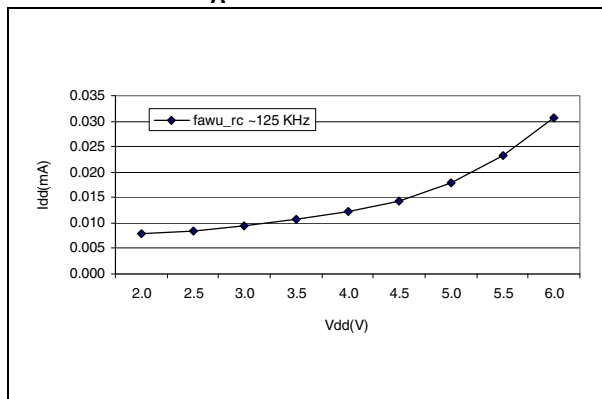
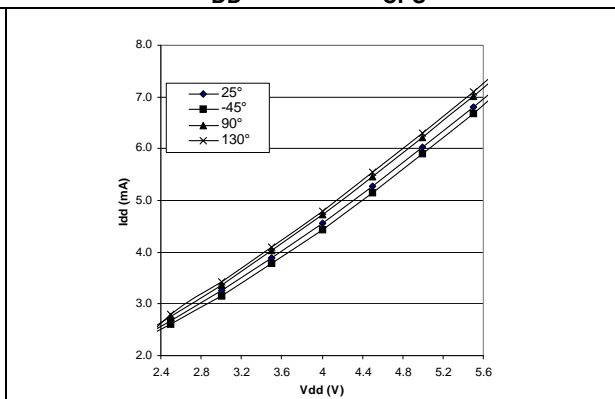
Figure 65. Typical  $I_{DD}$  in Wait vs.  $f_{CPU}$ Figure 66. Typical  $I_{DD}$  in Slow-Wait vs.  $f_{CPU}$ Figure 67. Typical  $I_{DD}$  in AWUFH mode at  $T_A=25^\circ\text{C}$ Figure 68. Typical  $I_{DD}$  vs. Temperature at  $V_{DD} = 5\text{ V}$  and  $f_{CPU} = 8\text{ MHz}$ 

Table 73. On-chip peripherals

Symbol	Parameter	Conditions		Typ	Unit
$I_{DD(AT)}$	12-bit Auto-Reload Timer supply current <sup>(1)</sup>	$f_{CPU}=4\text{ MHz}$	$V_{DD}=3.0\text{ V}$	300	$\mu\text{A}$
		$f_{CPU}=8\text{ MHz}$	$V_{DD}=5.0\text{ V}$	1000	
$I_{DD(SPI)}$	SPI supply current <sup>(2)</sup>	$f_{CPU}=4\text{ MHz}$	$V_{DD}=3.0\text{ V}$	50	
		$f_{CPU}=8\text{ MHz}$	$V_{DD}=5.0\text{ V}$	300	
$I_{DD(ADC)}$	ADC supply current when converting <sup>(3)</sup>	$f_{ADC}=4\text{ MHz}$	$V_{DD}=3.0\text{ V}$	250	
			$V_{DD}=5.0\text{ V}$	1100	

1. Data based on a differential  $I_{DD}$  measurement between reset configuration (timer stopped) and a timer running in PWM mode at  $f_{CPU}=8\text{MHz}$ .
2. Data based on a differential  $I_{DD}$  measurement between reset configuration and a permanent SPI master communication (data sent equal to 55h)
3. Data based on a differential  $I_{DD}$  measurement between reset configuration and continuous A/D conversions with amplifier off.

## 20.5 Clock and timing characteristics

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

**Table 74. General timings**

Symbol	Parameter <sup>(1)</sup>	Conditions	Min	Typ <sup>(2)</sup>	Max	Unit
$t_{C(INST)}$	Instruction cycle time	$f_{CPU}=8\text{ MHz}$	2	3	12	$t_{CPU}$
			250	375	1500	ns
$t_{V(IT)}$	Interrupt reaction time <sup>(3)</sup> $t_{V(IT)} = \Delta t_{C(INST)} + 10$	$f_{CPU}=8\text{ MHz}$	10		22	$t_{CPU}$
			1.25		2.75	$\mu\text{s}$

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 75. Auto wakeup from halt oscillator (AWU)**

Symbol	Parameter <sup>(1)</sup>	Conditions	Min	Typ	Max	Unit
$f_{AWU}$	AWU Oscillator Frequency		50	125	250	kHz
$t_{RCSRT}$	AWU Oscillator startup time				50	$\mu\text{s}$

1. Guaranteed by design.

### 20.5.1 Crystal and ceramic resonator oscillators

The ST7 internal clock can be supplied with eight 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 76. Oscillator characteristics**

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$f_{CROSC}$	Crystal Oscillator Frequency <sup>(1)</sup>		2		16	MHz
$C_{L1}$ $C_{L2}$	Recommended load capacitance versus equivalent serial resistance of the crystal or ceramic resonator ( $R_S$ )		See table below			pF

1. When PLL is used, please refer to the PLL characteristics chapter and to [Section 9: Supply, reset and clock management on page 32](#) ( $f_{CROSC}$  min. is 8 MHz with PLL).

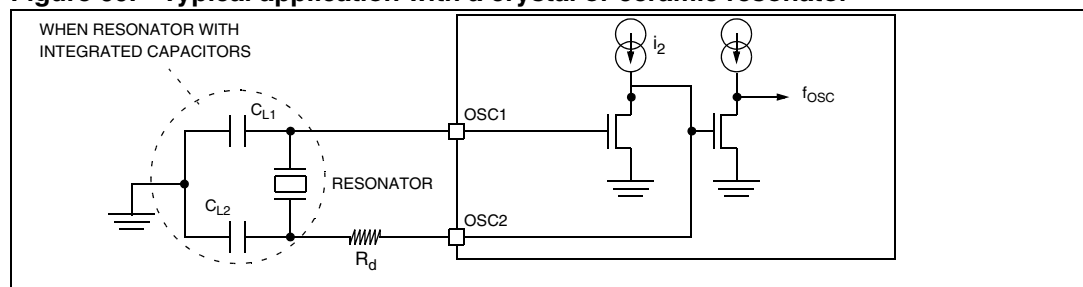
**Table 77. Typical ceramic resonators**

Supplier	f <sub>CrOSC</sub> [MHz]	Typical ceramic resonators <sup>(1)</sup>		CL1 <sup>(2)</sup> [pF]	CL2 <sup>(2)</sup> [pF]	Rd [Ω]	Supply voltage range [V]	Temperature range [°C]
		Type <sup>(3)</sup>	Reference					
Murata	2	SMD	CSTCC2M00G56-R0	(47)	(47)	0	2.4 V to 5.5 V	-40 to 85
	4	SMD	CSTCR4M00G53-R0	(15)	(15)	0		
		LEAD	CSTLS4M00G53-B0	(15)	(15)	0		
	8	SMD	CSTCE8M00G52-R0	(10)	(10)	0		
		LEAD	CSTLS8M00G53-B0	(15)	(15)	0		
	16	SMD	CSTCE16M0V51-R0	(5)	(5)	0	3.3 V to 5.5 V	
		LEAD	CSTLS16M0X53-B0	(15)	(15)	0	4.5 V to 5.5 V	
		LEAD	CSALS16M0X55-B0	7	7	1.5k	3.8 V to 5.5 V	

1. Resonator characteristics given by the ceramic resonator manufacturer. For more information on these resonators, please consult [www.murata.com](http://www.murata.com)

2. () means load capacitor built in resonator

3. SMD = -R0: Plastic tape package ( $\varnothing$  =180mm). LEAD = -B0: Bulk

**Figure 69. Typical application with a crystal or ceramic resonator**

## 20.6 Memory characteristics

$T_A$  = -40°C to 85°C, unless otherwise specified

**Table 78. RAM and hardware registers**

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{RM}$	Data retention mode <sup>(1)</sup>	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 79. FLASH program memory**

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{DD}$	Operating voltage for Flash write/erase		2.4		5.5	V
$t_{prog}$	Programming time for 1~32 bytes <sup>(1)</sup>	$T_A = -40$ to $+85^\circ\text{C}$		5	10	ms
	Programming time for 1.5 kBytes	$T_A = +25^\circ\text{C}$		0.24	0.48	s
$t_{RET}$	Data retention <sup>(2)</sup>	$T_A = +55^\circ\text{C}^{(3)}$	20			years
$N_{RW}$	Write erase cycles	$T_A = +25^\circ\text{C}$	10K <sup>(4)</sup>			cycles
$I_{DD}$	Supply current	Read / Write / Erase modes $f_{CPU} = 8\text{ MHz}$ , $V_{DD} = 5.5\text{ V}$			2.6 <sup>(5)</sup>	mA
		No Read/No Write Mode			100	$\mu\text{A}$
		Power down mode / Halt		0	0.1	$\mu\text{A}$

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. Design target value pending full product characterization.
5. Guaranteed by design. Not tested in production.

**Table 80. EEPROM data memory**

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{DD}$	Operating voltage for EEPROM write/erase		2.4		5.5	V
$t_{prog}$	Programming time for 1~32 bytes	$T_A = -40$ to $+85^\circ\text{C}$		5	10	ms
$t_{ret}$	Data retention <sup>(1)</sup>	$T_A = +55^\circ\text{C}^{(2)}$	20			years
$N_{RW}$	Write erase cycles	$T_A = +25^\circ\text{C}$	300K <sup>(3)</sup>			cycles

1. Data based on reliability test results and monitored in production.
2. The data retention time increases when the  $T_A$  decreases.
3. Design target value pending full product characterization.

## 20.7 EMC characteristics

Susceptibility tests are performed on a sample basis during product characterization.

### 20.7.1 Functional EMS (electromagnetic susceptibility)

Based on a simple running application on the product (toggling 2 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 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 81. EMS data**

Symbol	Parameter	Conditions	Level/class
$V_{FESD}$	Voltage limits to be applied on any I/O pin to induce a functional disturbance	$V_{DD}=5\text{ V}$ , $T_A=+25^\circ\text{C}$ , $f_{OSC2}=8\text{ MHz}$ conforms to IEC 1000-4-2	3B
$V_{FFTB}$	Fast transient voltage burst limits to be applied through 100 pF on $V_{DD}$ and $V_{SS}$ pins to induce a functional disturbance	$V_{DD}=5\text{ V}$ , $T_A=+25^\circ\text{C}$ , $f_{OSC2}=8\text{ MHz}$ conforms to IEC 1000-4-4	3B

## 20.7.2 Electromagnetic interference (EMI)

Based on a simple application running on the product (toggling 2 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 82. EMI data<sup>(1)</sup>**

Symbol	Parameter	Conditions	Monitored frequency band	Max vs. [f <sub>osc2</sub> /f <sub>cpu</sub> ]		Unit
				8/4 MHz	16/8 MHz	
S <sub>EMI</sub>	Peak level	V <sub>DD</sub> =5 V, T <sub>A</sub> =+25° C, SO20 package, conforming to SAE J 1752/3	0.1 MHz to 30 MHz	9	17	dB $\mu$ V
			30 MHz to 130 MHz	31	36	
			130 MHz to 1 GHz	25	27	
			SAE EMI Level	3.5	4	-

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

## 20.7.3 Absolute maximum ratings (electrical sensitivity)

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

### 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). This test conforms to the AEC-Q100-002 standard.

**Table 83. Absolute maximum ratings**

Symbol	Ratings	Conditions	Class	Maximum value <sup>(1)</sup>	Unit
V <sub>ESD(HBM)</sub>	Electrostatic discharge voltage (Human Body Model)	T <sub>A</sub> =+25°C conforming to AEC-Q100-002	H1C	4000	V

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

### Static latch-up (LU)

2 complementary static tests are required on six parts to assess the latch-up performance.

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

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

**Table 84. Electrical sensitivities**

Symbol	Parameter	Conditions	Class
LU	Static latch-up class	T <sub>A</sub> =+25° C conforming to JESD 78	II level A

## 20.8 I/O port pin characteristics

### 20.8.1 General characteristics

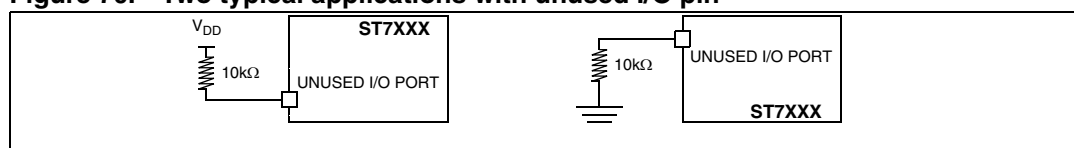
Subject to general operating conditions for  $V_{DD}$ ,  $f_{OSC}$ , and  $T_A$  unless otherwise specified.

**Table 85. General characteristics**

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{IL}$	Input low level voltage		$V_{SS} - 0.3$		$0.3 \times V_{DD}$	V
$V_{IH}$	Input high level voltage		$0.7 \times V_{DD}$		$V_{DD} + 0.3$	
$V_{hys}$	Schmitt trigger voltage hysteresis <sup>(1)</sup>			400		mV
$I_L$	Input leakage current	$V_{SS} \leq V_{IN} \leq V_{DD}$			$\pm 1$	$\mu A$
$I_S$	Static current consumption induced by each floating input pin <sup>(2)</sup>	Floating input mode		400		
$R_{PU}$	Weak pull-up equivalent resistor <sup>(3)</sup>	$V_{IN} = V_{SS}$ $V_{DD} = 5 V$	50	120	250	k $\Omega$
		$V_{DD} = 3 V$		160		
$C_{IO}$	I/O pin capacitance			5		pF
$t_{r(I/O)out}$	Output high to low level fall time <sup>(1)</sup>	$C_L = 50 pF$ Between 10% and 90%		25		ns
$t_{r(I/O)out}$	Output low to high level rise time <sup>(1)</sup>			25		
$t_{w(IT)in}$	External interrupt pulse time <sup>(4)</sup>		1			$t_{CPU}$

1. Data based on characterization results, not tested in production.
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 70](#)). 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 (corresponding  $I_{PU}$  current characteristics described in [Figure 71](#)).
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 70. Two typical applications with unused I/O pin**



**Caution:** 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.

**Note:** 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.

Figure 71. Typical  $I_{PU}$  vs.  $V_{DD}$  with  $V_{IN}=V_{SS}$

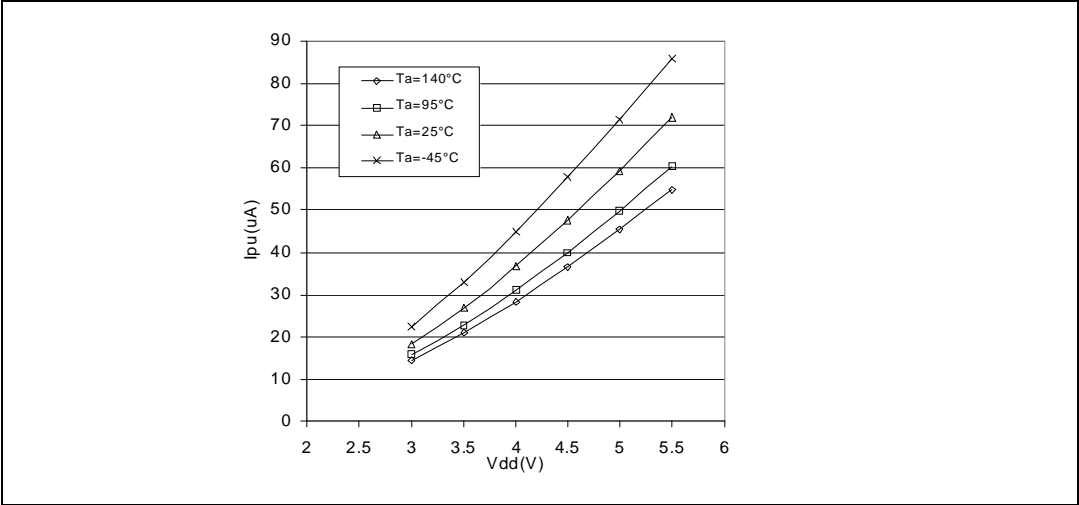




Table 86. Output driving current

Symbol	Parameter	Conditions <sup>(1)</sup>	Min	Max	Unit
$V_{OL}^{(2)}$	Output low level voltage for a standard I/O pin when 8 pins are sunk at same time (see <a href="#">Figure 75</a> )	$I_{IO}=+5\text{ mA}$ $T_A\leq 85^\circ\text{C}$ $T_A\geq 85^\circ\text{C}$		1.0 1.2	V
		$I_{IO}=+2\text{ mA}$ $T_A\leq 85^\circ\text{C}$ $T_A\geq 85^\circ\text{C}$		0.4 0.5	
	Output low level voltage for a high sink I/O pin when 4 pins are sunk at same time (see <a href="#">Figure 77</a> )	$I_{IO}=+20\text{ mA}$ , $T_A\leq 85^\circ\text{C}$ $T_A\geq 85^\circ\text{C}$		1.3 1.5	
		$I_{IO}=+8\text{ mA}$ $T_A\leq 85^\circ\text{C}$ $T_A\geq 85^\circ\text{C}$		0.75 0.85	
$V_{OH}^{(3)}$	Output high level voltage for an I/O pin when 4 pins are sourced at same time (see <a href="#">Figure 83</a> )	$I_{IO}=-5\text{ mA}$ , $T_A\leq 85^\circ\text{C}$ $T_A\geq 85^\circ\text{C}$	$V_{DD}-1.5$ $V_{DD}-1.6$		
		$I_{IO}=-2\text{ mA}$ $T_A\leq 85^\circ\text{C}$ $T_A\geq 85^\circ\text{C}$	$V_{DD}-0.8$ $V_{DD}-1.0$		
$V_{OL}^{(2)(4)}$	Output low level voltage for a standard I/O pin when 8 pins are sunk at same time (see <a href="#">Figure 74</a> )	$I_{IO}=+2\text{ mA}$ $T_A\leq 85^\circ\text{C}$ $T_A\geq 85^\circ\text{C}$		0.5 0.6	
		$I_{IO}=+8\text{ mA}$ $T_A\leq 85^\circ\text{C}$ $T_A\geq 85^\circ\text{C}$		0.5 0.6	
$V_{OH}^{(3)(4)}$	Output high level voltage for an I/O pin when 4 pins are sourced at same time	$I_{IO}=-2\text{ mA}$ $T_A\leq 85^\circ\text{C}$ $T_A\geq 85^\circ\text{C}$	$V_{DD}-0.8$ $V_{DD}-1.0$		
$V_{OL}^{(2)(4)}$	Output low level voltage for a standard I/O pin when 8 pins are sunk at same time (see <a href="#">Figure 73</a> )	$I_{IO}=+2\text{ mA}$ $T_A\leq 85^\circ\text{C}$ $T_A\geq 85^\circ\text{C}$		0.6 0.7	
		$I_{IO}=+8\text{ mA}$ $T_A\leq 85^\circ\text{C}$ $T_A\geq 85^\circ\text{C}$		0.6 0.7	
$V_{OH}^{(3)(4)}$	Output high level voltage for an I/O pin when 4 pins are sourced at same time (see <a href="#">Figure 80</a> )	$I_{IO}=-2\text{ mA}$ $T_A\leq 85^\circ\text{C}$ $T_A\geq 85^\circ\text{C}$	$V_{DD}-0.9$ $V_{DD}-1.0$		

1. Subject to general operating conditions for  $V_{DD}$ ,  $f_{CPU}$ , and  $T_A$  unless otherwise specified.
2. The  $I_{IO}$  current sunk must always respect the absolute maximum rating specified in [Section 20.2](#) and the sum of  $I_{IO}$  (I/O ports and control pins) must not exceed  $I_{VSS}$ .
3. The  $I_{IO}$  current sourced must always respect the absolute maximum rating specified in [Section 20.2](#) and the sum of  $I_{IO}$  (I/O ports and control pins) must not exceed  $I_{VDD}$ .
4. Not tested in production, based on characterization results.

Figure 72. Typical  $V_{OL}$  at  $V_{DD}=2.4$  V (standard) Figure 73. Typical  $V_{OL}$  at  $V_{DD}=2.7$  V (standard)

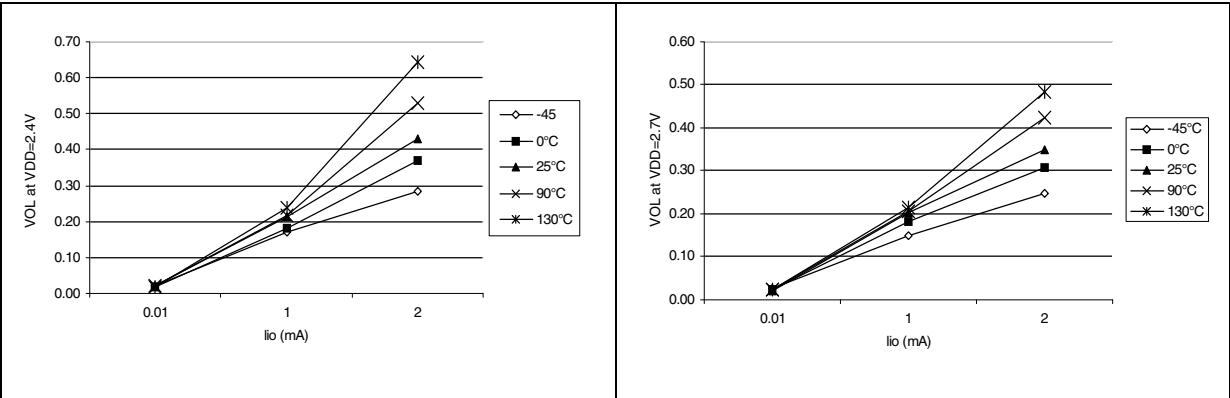


Figure 74. Typical  $V_{OL}$  at  $V_{DD}=3.3$  V (standard) Figure 75. Typical  $V_{OL}$  at  $V_{DD}=5$  V (standard)

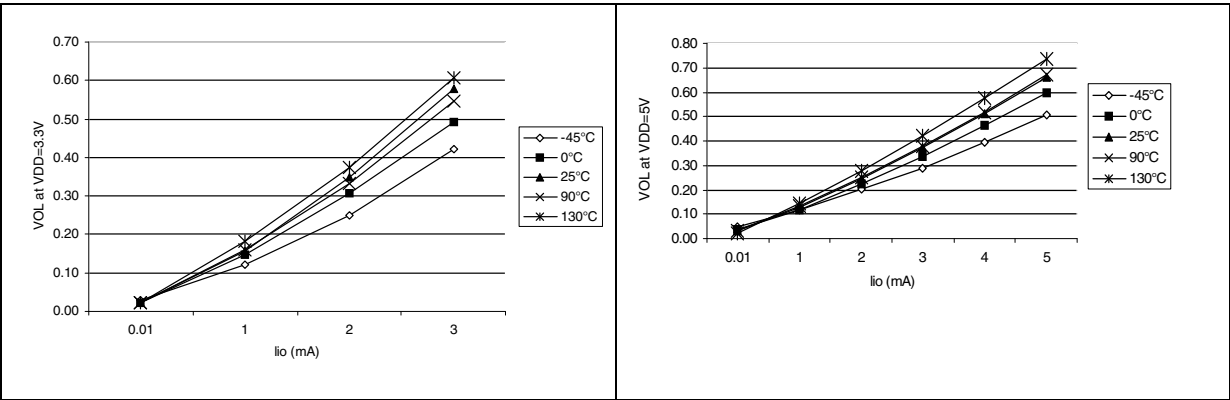


Figure 76. Typical  $V_{OL}$  at  $V_{DD}=2.4$  V (high-sink) Figure 77. Typical  $V_{OL}$  at  $V_{DD}=5$  V (high-sink)

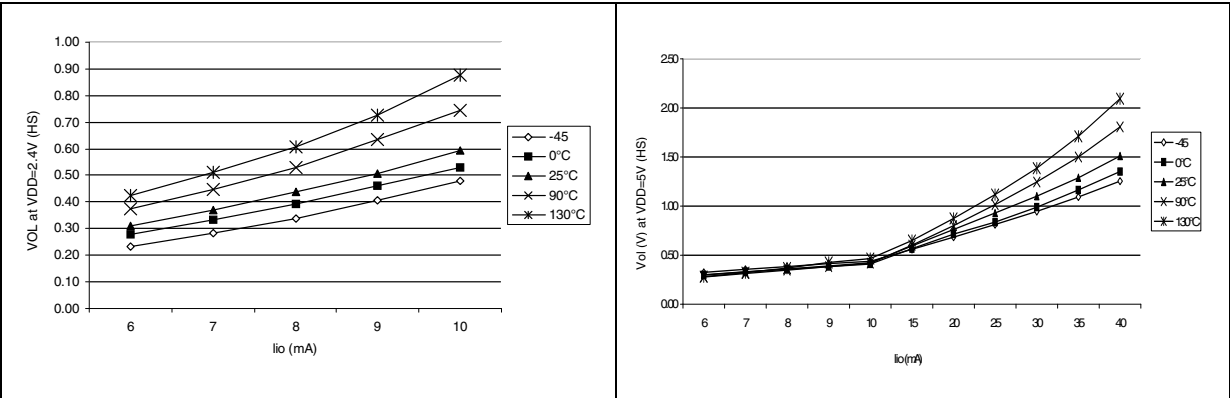


Figure 78. Typical  $V_{OL}$  at  $V_{DD}=3$  V (high-sink)

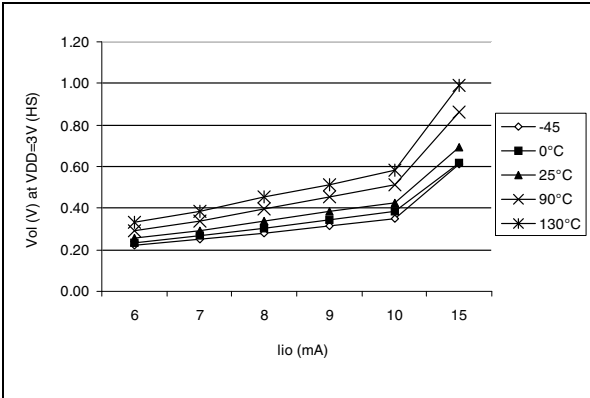


Figure 79. Typical  $V_{DD}-V_{OH}$  at  $V_{DD}=2.4$  V

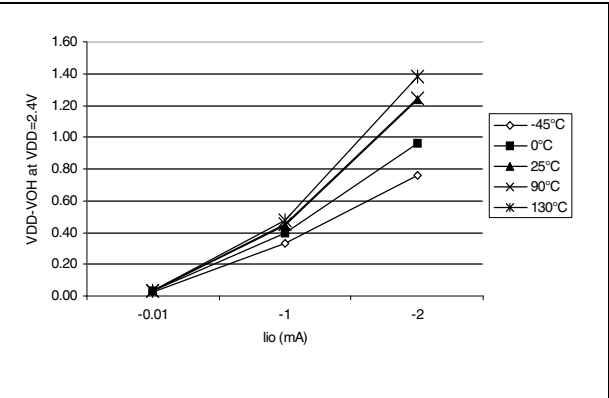


Figure 80. Typical  $V_{DD}-V_{OH}$  at  $V_{DD}=2.7$  V

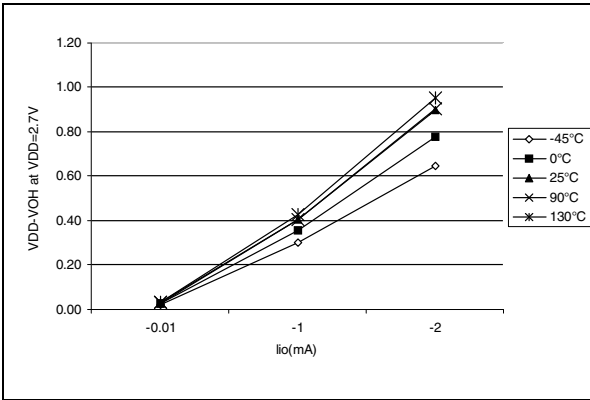


Figure 81. Typical  $V_{DD}-V_{OH}$  at  $V_{DD}=3$  V

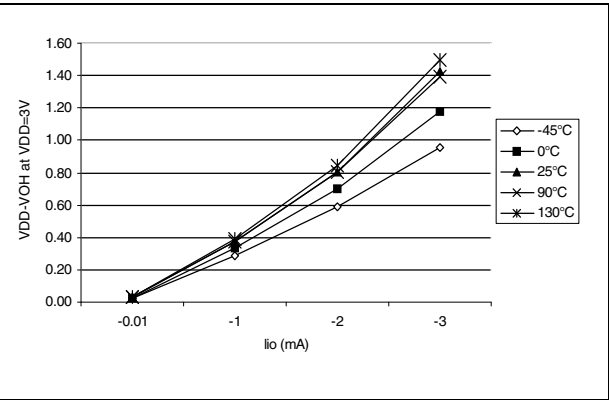


Figure 82. Typical  $V_{DD}-V_{OH}$  at  $V_{DD}=4$  V

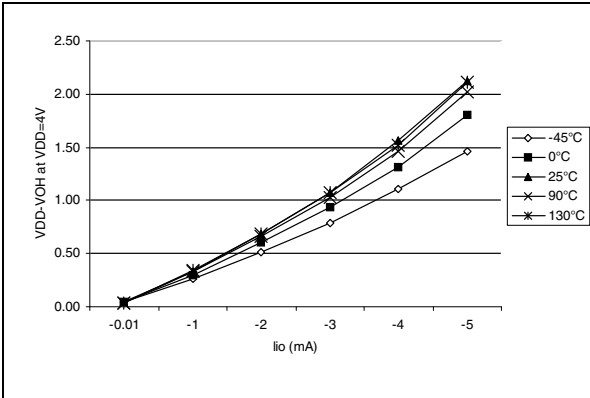


Figure 83. Typical  $V_{DD}-V_{OH}$  at  $V_{DD}=5$  V

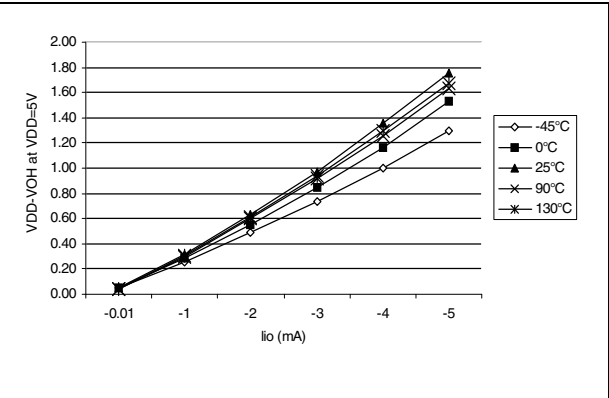


Figure 84. Typical  $V_{OL}$  vs.  $V_{DD}$  (standard I/Os)

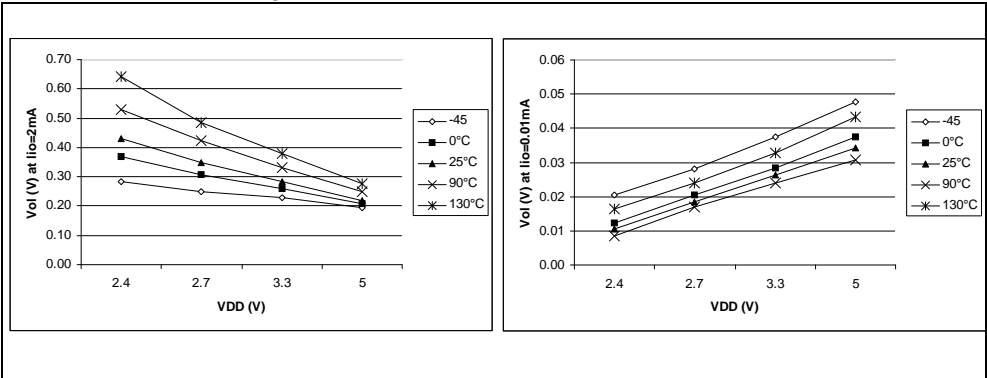


Figure 85. Typical  $V_{OL}$  vs.  $V_{DD}$  (high-sink I/Os)

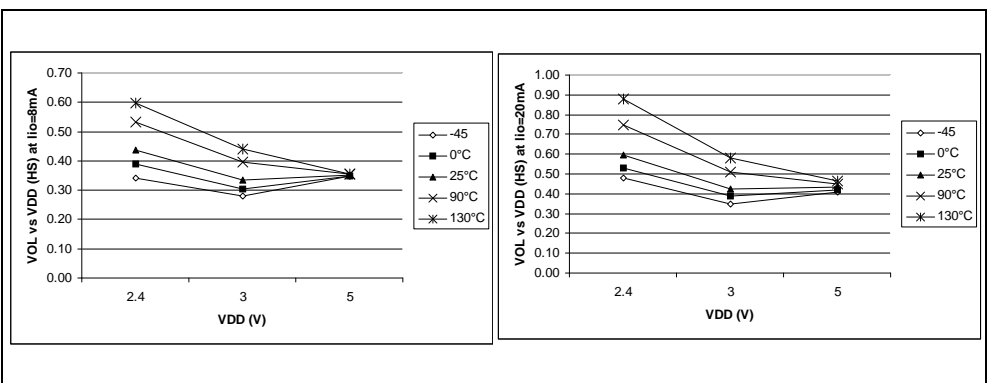
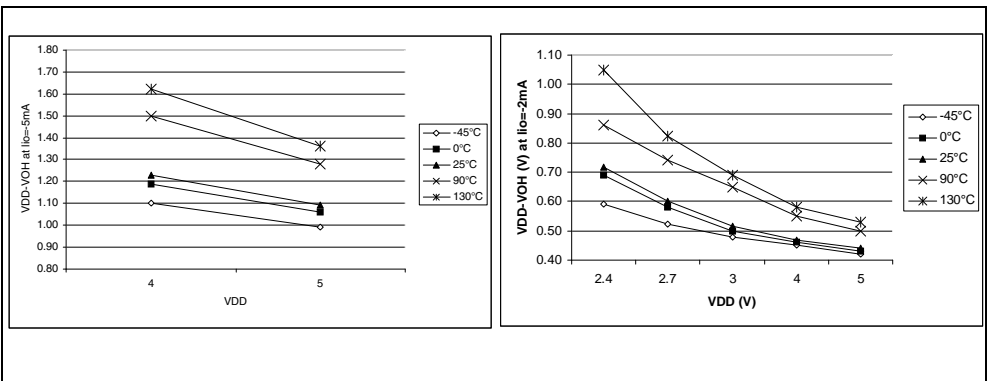


Figure 86. Typical  $V_{DD}-V_{OH}$  vs.  $V_{DD}$



## 20.9 Control pin characteristics

$T_A = -40^{\circ}\text{C}$  to  $85^{\circ}\text{C}$ , unless otherwise specified

**Table 87. Asynchronous  $\overline{\text{RESET}}$  pin characteristics**

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{IL}$	Input low level voltage		$V_{SS} - 0.3$		$0.3 \times V_{DD}$	V
$V_{IH}$	Input high level voltage		$0.7 \times V_{DD}$		$V_{DD} + 0.3$	
$V_{hys}$	Schmitt trigger voltage hysteresis <sup>(1)</sup>			2		V
$V_{OL}$	Output low level voltage <sup>(2)</sup>	$V_{DD}=5\text{ V}$ $I_{IO}=+5\text{ mA}$ $T_A \leq 85^{\circ}\text{C}$ $T_A \geq 85^{\circ}\text{C}$		0.5	1.0 1.2	V
		$I_{IO}=+2\text{ mA}$ $T_A \leq 85^{\circ}\text{C}$ $T_A \geq 85^{\circ}\text{C}$		0.2	0.4 0.5	
$R_{ON}$	Pull-up equivalent resistor <sup>(3) (1)</sup>	$V_{DD}=5\text{ V}$	20	40	80	$k\Omega$
		$V_{DD}=3\text{ V}$	40	70	120	
$t_{w(RSTL)out}$	Generated reset pulse duration	Internal reset sources		30		$\mu\text{s}$
$t_{h(RSTL)in}$	External reset pulse hold time <sup>(4)</sup>		20			$\mu\text{s}$
$t_{g(RSTL)in}$	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 13.2.2 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.

### **$\overline{\text{RESET}}$ pin protection when LVD is enabled**

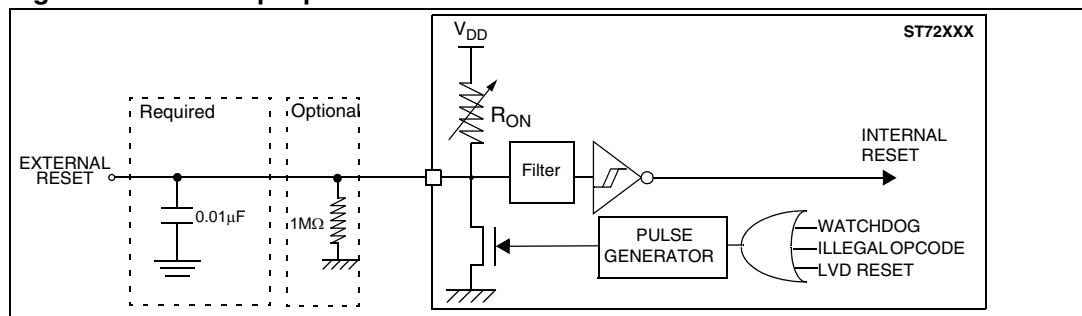
When the LVD is enabled, it is recommended to protect the  $\overline{\text{RESET}}$  pin as shown in [Figure 87](#) and follow these guidelines:

1. The reset network protects the device against parasitic resets.
2. 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).
3. 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  $V_{IL \text{ max.}}$  level specified in [Section 20.8](#). Otherwise the reset will not be taken into account internally.
4. Because the reset circuit is designed to allow the internal  $\overline{\text{RESET}}$  to be output in the  $\overline{\text{RESET}}$  pin, the user must ensure that the current sunk on the  $\overline{\text{RESET}}$  pin (by an external pull-up for example) is less than the absolute maximum value specified for  $I_{INJ}(\overline{\text{RESET}})$  in [Section 20.2 on page 128](#).
5. When the LVD is enabled, it is mandatory not to connect a pull-up resistor. A 10nF pull-down capacitor is recommended to filter noise on the reset line.
6. In case a capacitive power supply is used, it is recommended to connect a 1M ohm pull-down resistor to the  $\overline{\text{RESET}}$  pin to discharge any residual voltage induced by this capacitive power supply (this will add 5  $\mu\text{A}$  to the power consumption of the MCU).

#### **Tips when using the LVD:**

- Check that all recommendations related to reset circuit have been applied (see section above)
- Check that the power supply is properly decoupled (100 nF + 10  $\mu\text{F}$  close to the MCU). Refer to AN1709. If this cannot be done, it is recommended to put a 100 nF + 1M Ohm 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 avoiding any start-up marginality. In most cases, steps 1 and 2 above are sufficient for a robust solution. Otherwise: Replace 10 nF pull-down on the  $\overline{\text{RESET}}$  pin with a 5  $\mu\text{F}$  to 20  $\mu\text{F}$  capacitor.

**Figure 87.  $\overline{\text{RESET}}$  pin protection when LVD is enabled.**

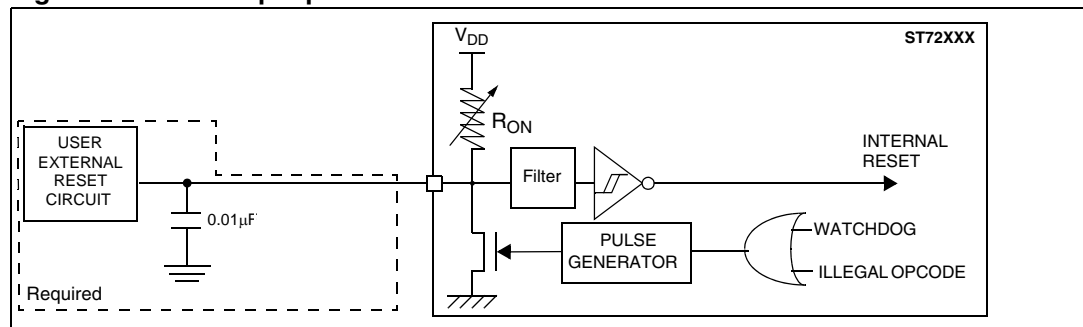


### **$\overline{\text{RESET}}$ pin protection when LVD is disabled**

When the LVD is disabled, it is recommended to protect the  $\overline{\text{RESET}}$  pin as shown in [Figure 88](#) and follow these guidelines:

1. The reset network protects the device against parasitic resets.
2. 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).
3. 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  $V_{IL \text{ max.}}$  level specified in [Section 20.8](#). Otherwise the reset will not be taken into account internally.
4. 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 (by an external pull-up for example) is less than the absolute maximum value specified for  $I_{\text{INJ}}(\text{RESET})$  in [Section 20.2 on page 128](#).

**Figure 88.  $\overline{\text{RESET}}$  pin protection when LVD is disabled.**



## **20.10 Communication interface characteristics**

### **20.10.1 SPI - serial peripheral 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 ( $\overline{\text{SS}}$ , SCK, MOSI, MISO).

**Table 88. SPI characteristics**

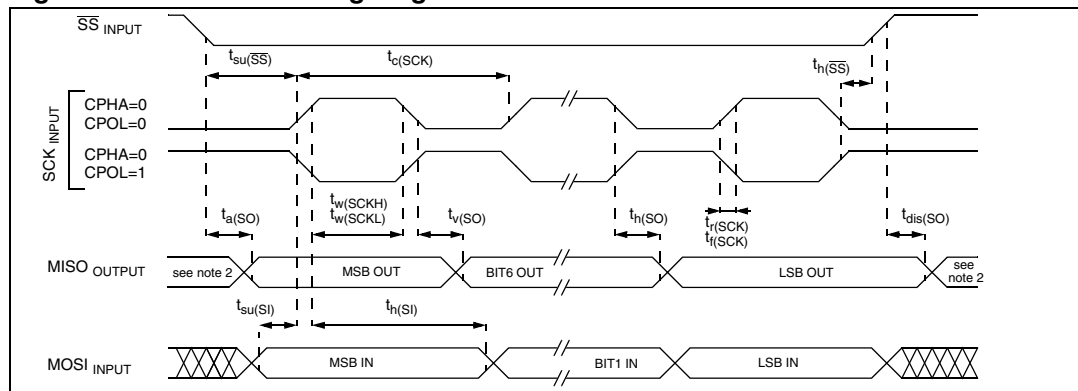
Symbol	Parameter	Conditions	Min	Max	Unit
$f_{\text{SCK}} = 1/t_{\text{c}}(\text{SCK})$	SPI clock frequency	Master $f_{\text{CPU}}=8 \text{ MHz}$	$f_{\text{CPU}}/128 =0.0625$	$f_{\text{CPU}}/4 =2$	MHz
		Slave $f_{\text{CPU}}=8 \text{ MHz}$	0	$f_{\text{CPU}}/2 =4$	
$t_{\text{r}}(\text{SCK})$ $t_{\text{f}}(\text{SCK})$	SPI clock rise and fall time		see I/O port pin description		

**Table 88. SPI characteristics (continued)**

Symbol	Parameter	Conditions	Min	Max	Unit
$t_{su}(\overline{SS})^{(1)}$	$\overline{SS}$ setup time <sup>(2)</sup>	Slave	$(4 \times T_{CPU}) + 150$		ns
$t_{h}(\overline{SS})^{(1)}$	$\overline{SS}$ hold time	Slave	120		
$t_{w(SCKH)}$ $t_{w(SCKL)}$	SCK high and low time	Master Slave	100 90		
$t_{su(MI)}$ $t_{su(SI)}$	Data input setup time	Master Slave	100 100		
$t_{h(MI)}$ $t_{h(SI)}$	Data input hold time	Master Slave	100 100		
$t_{a(SO)}$	Data output access time	Slave	0	120	
$t_{dis(SO)}$	Data output disable time	Slave		240	
$t_{v(SO)}$	Data output valid time	Slave (after enable edge)		120	
$t_{h(SO)}$	Data output hold time		0		
$t_{v(MO)}$	Data output valid time	Master (after enable edge)		120	
$t_{h(MO)}$	Data output hold time		0		

1. Data based on design simulation and/or characterisation results, not tested in production.

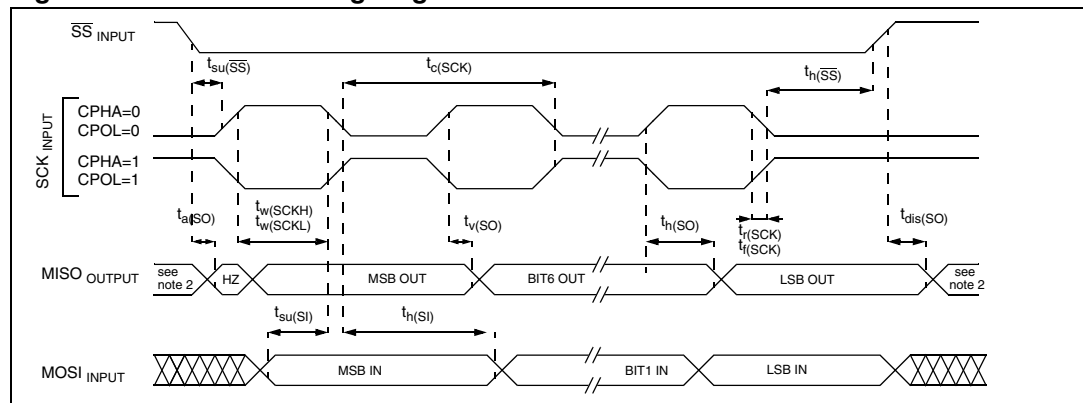
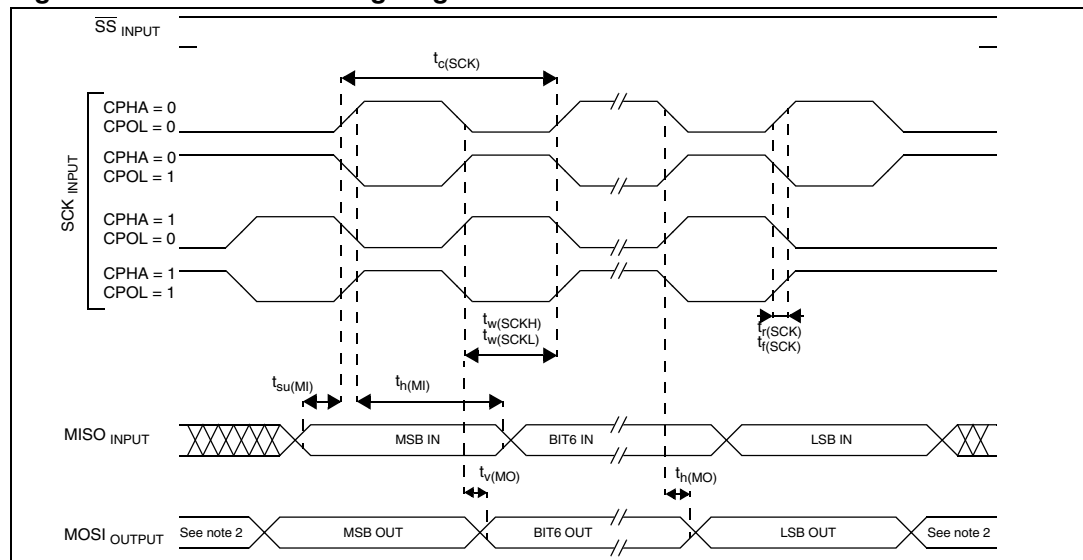
2. Depends on  $f_{CPU}$ . For example, if  $f_{CPU} = 8 \text{ MHz}$ , then  $T_{CPU} = 1 / f_{CPU} = 125 \text{ ns}$  and  $t_{su}(\overline{SS}) = 550 \text{ ns}$

**Figure 89. SPI slave timing diagram with CPHA=0**

Note: 1 Measurement points are done at CMOS levels:  $0.3 \times V_{DD}$  and  $0.7 \times V_{DD}$ .

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 90. SPI slave timing diagram with CPHA=1****Figure 91. SPI master timing diagram**

- Note:**
- 1 Measurement points are done at CMOS levels:  $0.3 \times V_{DD}$  and  $0.7 \times V_{DD}$
  - 2 When no communication is on-going the data output line of the SPI (MOSI in master mode, MISO in slave mode) has its alternate function capability released. In this case, the pin status depends of the I/O port configuration.

## 20.11 10-bit ADC characteristics

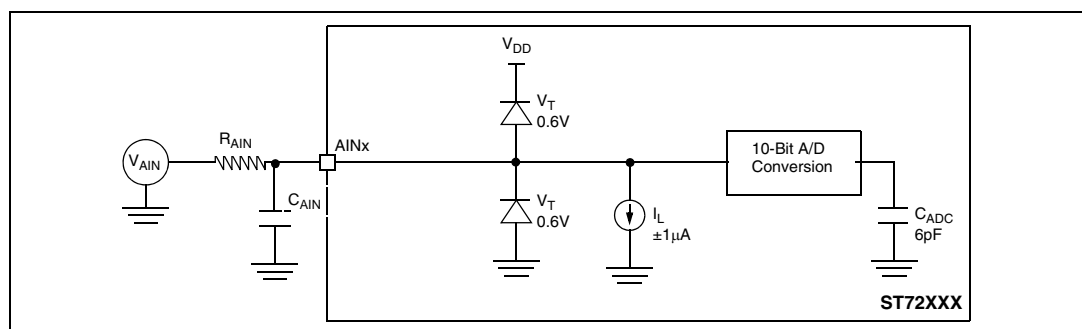
Subject to general operating condition for  $V_{DD}$ ,  $f_{OSC}$ , and  $T_A$  unless otherwise specified.

**Table 89. ADC characteristics**

Symbol	Parameter	Conditions	Min	Typ <sup>(1)</sup>	Max	Unit
f <sub>ADC</sub>	ADC clock frequency				4	MHz
V <sub>AIN</sub>	Conversion voltage range <sup>(2)</sup>		V <sub>SSA</sub>		V <sub>DDA</sub>	V
R <sub>AIN</sub>	External input resistor				10 <sup>(3)</sup>	kΩ
C <sub>ADC</sub>	Internal sample and hold capacitor			6		pF
t <sub>STAB</sub>	Stabilization time after ADC enable	f <sub>CPU</sub> =8MHz, f <sub>ADC</sub> =4MHz	0 <sup>(4)</sup>			μs
t <sub>ADC</sub>	Conversion time (Sample+Hold)		3.5			
	- Sample capacitor loading time - Hold conversion time		4 10			1/f <sub>ADC</sub>
I <sub>ADC</sub>	Analog Part				1	mA
	Digital Part				0.2	

1. Unless otherwise specified, typical data are based on  $T_A=25^\circ\text{C}$  and  $V_{DD}-V_{SS}=5\text{V}$ . They are given only as design guidelines and are not tested.
2. When  $V_{DDA}$  and  $V_{SSA}$  pins are not available on the pinout, the ADC refers to  $V_{DD}$  and  $V_{SS}$ .
3. Any added external serial resistor will downgrade the ADC accuracy (especially for resistance greater than 10k $\Omega$ ). Data based on characterization results, not tested in production.
4. The stabilization time of the AD converter is masked by the first  $t_{LOAD}$ . The first conversion after the enable is then always valid.

**Figure 92. Typical Application with ADC**

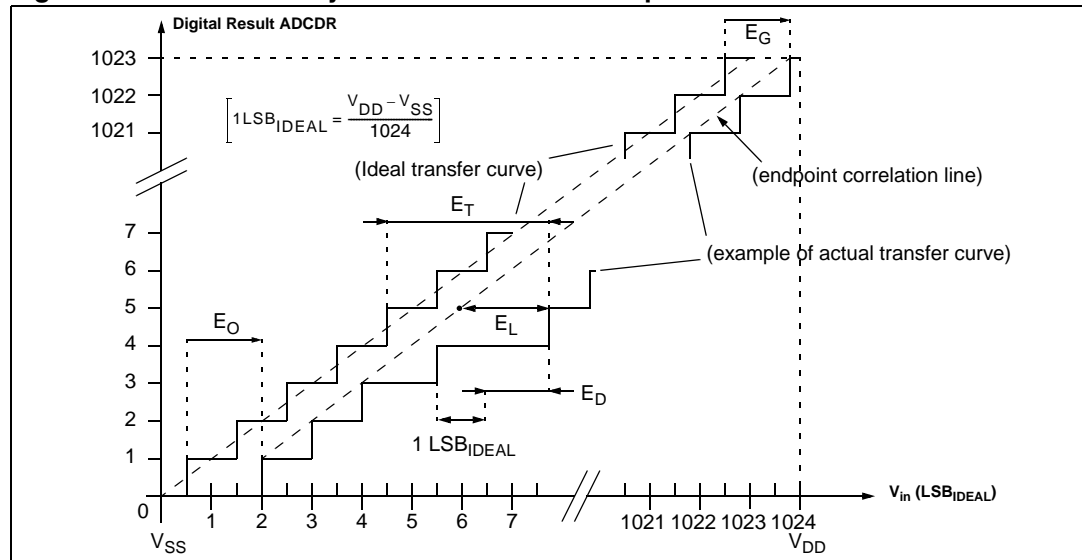


**Table 90. ADC Accuracy with  $V_{DD}=5.0\text{ V}$** 

Symbol	Parameter	Conditions	Typ	Max <sup>1)</sup>	Unit
$ E_T $	Total unadjusted error <sup>(1)</sup>	$f_{CPU}=8\text{ MHz}$ , $f_{ADC}=4\text{ MHz}$ <sup>(2)</sup> , $V_{DD}=5.0\text{ V}$	3	6	LSB
$ E_O $	Offset error <sup>(2)</sup>		1.5	5	
$ E_G $	Gain Error <sup>(2)</sup>		2	4.5	
$ E_D $	Differential linearity error <sup>(2)</sup>		2.5	4.5	
$ E_L $	Integral linearity error <sup>(2)</sup>		2.5	4.5	

1. Injecting negative current on any of the analog input pins significantly reduces the accuracy of any conversion being performed on any analog input.  
Analog pins can be protected against negative injection by adding a Schottky diode (pin to ground). Injecting negative current on digital input pins degrades ADC accuracy especially if performed on a pin close to the analog input pins.  
Any positive injection current within the limits specified for  $I_{INJ(PIN)}$  and  $\Sigma I_{INJ(PIN)}$  in [Section 20.2](#) does not affect the ADC accuracy.

2. Data based on characterization results over the whole temperature range, not tested in production.

**Figure 93. ADC accuracy characteristics with amplifier disabled****Glossary:**

**$E_T$** =Total Unadjusted Error: maximum deviation between the actual and the ideal transfer curves.

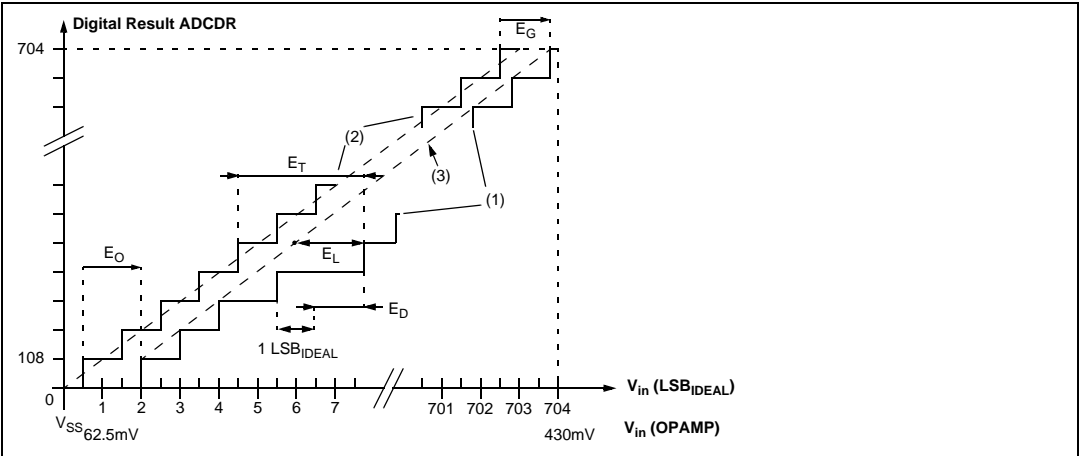
**$E_O$** =Offset Error: deviation between the first actual transition and the first ideal one.

**$E_G$** =Gain Error: deviation between the last ideal transition and the last actual one.

**$E_D$** =Differential Linearity Error: maximum deviation between actual steps and the ideal one.

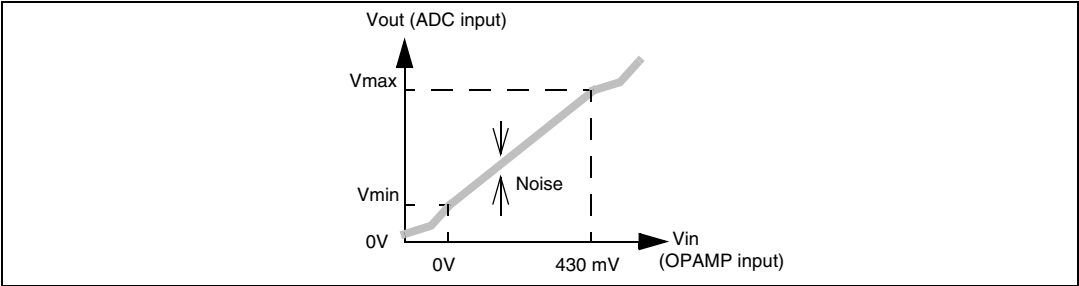
**$E_L$** =Integral Linearity Error: maximum deviation between any actual transition and the endpoint correlation line.

Figure 94. ADC accuracy characteristics with amplifier enabled



Note: When the AMPSEL bit in the ADCDRL register is set, it is mandatory that  $f_{ADC}$  be less than or equal to 2 MHz. (if  $f_{CPU}=8$  MHz. then SPEED=0, SLOW=1).

Figure 95. Amplifier output voltage vs. input voltage



**Table 91. Amplifier characteristics**

Symbol	Parameter	Conditions <sup>(1)</sup>	Min	Typ	Max	Unit
V <sub>DD(AMP)</sub>	Amplifier operating voltage		3.6		5.5	V
V <sub>IN</sub>	Amplifier input voltage <sup>(2)</sup>	V <sub>DD</sub> =3.6 V	0		350	mV
		V <sub>DD</sub> =5 V	0		500	
V <sub>OFFSET</sub>	Amplifier output offset voltage <sup>(3)</sup>	V <sub>DD</sub> =5 V		200		mV
V <sub>STEP</sub>	Step size for monotonicity <sup>(4)</sup>	V <sub>DD</sub> =3.6 V	3.5			mV
		V <sub>DD</sub> =5 V	4.89			
Linearity	Output Voltage Response		Linear			
Gain factor	Amplified Analog input Gain <sup>(5)</sup>			8		
V <sub>max</sub>	Output Linearity Max Voltage	V <sub>INmax</sub> = 430 mV, V <sub>DD</sub> =5 V		3.65	3.94	V
V <sub>min</sub>	Output Linearity Min Voltage			200		mV

1. Data based on characterization results over the whole temperature range, not tested in production.
2. Please refer to the Application Note AN1830 for details of TE% vs  $V_{IN}$ .
3. Refer to the offset variation in temperature below
4. Monotonicity guaranteed if  $V_{IN}$  increases or decreases in steps of min. 5 mV.
5. For precise conversion results it is recommended to calibrate the amplifier at the following two points:
  - Offset at  $V_{INmin} = 0\text{V}$
  - Gain at full scale (for example  $V_{IN}=430\text{ mV}$ )

### 20.11.1 Amplifier output offset variation

The offset is quite sensitive to temperature variations. In order to ensure a good reliability in measurements, the offset must be recalibrated periodically i.e. during power on or whenever the device is reset depending on the customer application and during temperature variation. The table below gives the typical offset variation over temperature:

**Table 92. Amplifier offset variation over temperature**

Typical offset variation (LSB)				Unit
-45	-20	+25	+90	°C
-12	-7	-	+13	LSB

## 21 Package characteristics

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

## 21.1 Package mechanical data

Figure 96. 20-Pin plastic small outline package, 300-mil width

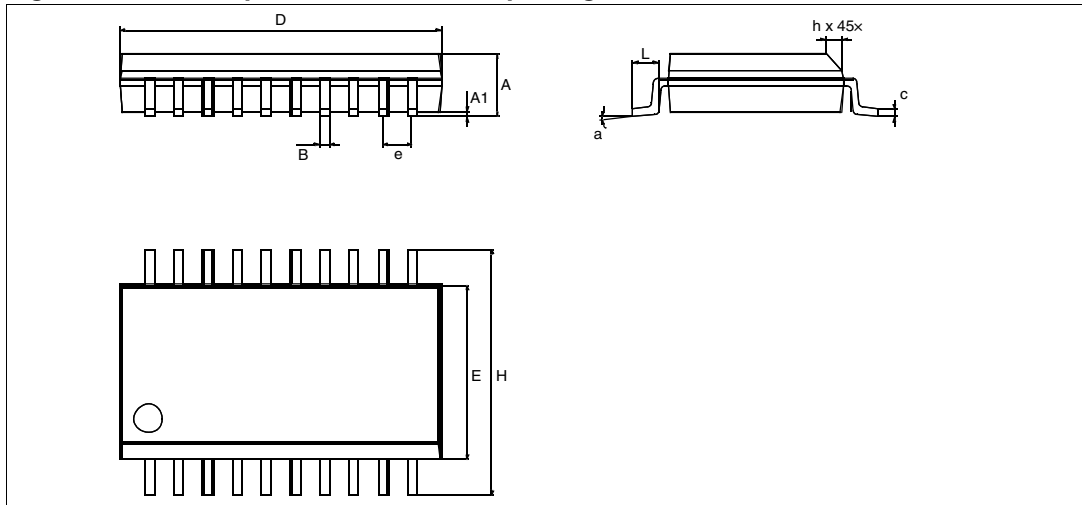


Table 93. 20-pin plastic small outline package dimensions

Dim.	mm			Inches		
	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			0.050	
H	10.00		10.65	0.3937		0.4193
h	0.25		0.75	0.0098		0.0295
$\alpha$	0°		8°	0°		8°
L	0.40		1.27	0.0157		0.0500
	Number of pins					
N	20					

**Table 94. Thermal characteristics**

Symbol	Ratings	Value	Unit
$R_{thJA}$	Package thermal resistance (junction to ambient) SO20 DIP20	125 63	°C/W
$T_{Jmax}$	Maximum junction temperature <sup>(1)</sup>	150	°C
$P_{Dmax}$	Power dissipation <sup>(2)</sup>	500	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 ( $I_{DD} \times V_{DD}$ ) and  $P_{PORT}$  is the port power dissipation depending on the ports used in the application.



## 22 Device configuration

Each device is available for production in user programmable versions (FLASH) as well as in factory coded versions (FASTROM).

ST7FDALI devices are FLASH versions. ST7PDALI devices are Factory Advanced Service Technique ROM (FASTROM) versions: they are factory programmed FLASH devices.

ST7FDALI devices are shipped to customers with a default program memory content (FFh), while FASTROM factory coded parts contain the code supplied by the customer. This implies that FLASH devices have to be configured by the customer using the Option Bytes while the FASTROM devices are factory configured.

### 22.1 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).

	OPTION BYTE 0								OPTION BYTE 1							
	7	0						7	0							
	Res.	OSCRANGE 2:0			SEC1	SEC0	FMP R	FMP W	PLL x4x8	PLL OFF	PLL32 OFF	OSC	LVD1	LVD0	WDG SW	WDG HALT
Default Value	1	1	1	1	1	1	0	0	1	1	1	0	1	1	1	1

#### 22.1.1 Option byte 0

OPT7 = Reserved, must always be 1.

OPT6:4 = **OSCRANGE[2:0]** *Oscillator range*

When the internal RC oscillator is not selected (Option OSC=1), these option bits select the range of the resonator oscillator current source or the external clock source.

**Table 95. Oscillator source configuration**

			OSCRANGE		
			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 source: CLKIN	on OSC1		1	0	1
	on PB4		1	1	1
Reserved			1	1	0

**Note:** When the internal RC oscillator is selected, the *OSCRANGE* option bits must be kept at their default value in order to select the 256 clock cycle delay (see [Section 9.6](#)).

OPT3:2 = **SEC[1:0]** Sector 0 size definition

These option bits indicate the size of sector 0 according to the following table.

**Table 96. Program memory sector size configuration**

Sector 0 Size	SEC1	SEC0
0.5k	0	0
1k	0	1
2k	1	0
4k	1	1

OPT1 = **FMP\_R** Readout protection

Readout 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 the ST7 Flash Programming Reference Manual and [Section 6.5 on page 21](#) for more details

0: Readout protection off

1: Readout protection on

OPT0 = **FMP\_W** FLASH write protection

This option indicates if the FLASH program memory is write protected.

**Caution:** When this option is selected, the program memory (and the option bit itself) can never be erased or programmed again.

0: Write protection off

1: Write protection on

## 22.1.2 Option byte 1

OPT7 = **PLLx4x8** PLL Factor selection.

0: PLLx4

1: PLLx8

OPT6 = **PLLOFF** PLL disable.

0: PLL enabled

1: PLL disabled (by-passed)

OPT5 = **PLL32OFF** 32MHz PLL disable.

0: PLL32 enabled

1: PLL32 disabled (by-passed)

OPT4 = **OSC** RC Oscillator selection

0: RC oscillator on

1: RC oscillator off

**Note:** If the RC oscillator is selected, then 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.

OPT3:2 = **LVD[1:0]** *Low voltage detection selection*

These option bits enable the LVD block with a selected threshold as shown in [Table 97](#).

**Table 97. LVD threshold configuration**

Configuration	LVD1	LVD0
LVD Off	1	1
Highest Voltage Threshold (~4.1V)	1	0
Medium Voltage Threshold (~3.5V)	0	1
Lowest Voltage Threshold (~2.8V)	0	0

OPT1 = **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)

OPT0 = **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

**Table 98. List of valid option combinations**

Operating conditions				Option bits		
V <sub>DD</sub> range	Clock source	PLL	Typ f <sub>CPU</sub>	OSC	PLLOFF	PLLx4x8
2.4 V - 3.3 V	Internal RC 1%	off	0.7 MHz @3 V	0	1	1
		x4	2.8 MHz @3 V	0	0	0
		x8	-	-	-	-
	External clock or oscillator (depending on OPT6:4 selection)	off	0-4 MHz	1	1	1
		x4	4 MHz	1	0	0
		x8	-	-	-	-
3.3 V - 5.5 V	Internal RC 1%	off	1 MHz @5 V	0	1	1
		x4	-	-	-	-
		x8	8 MHz @5 V	0	0	1
	External clock or oscillator (depending on OPT6:4 selection)	off	0-8 MHz	1	1	1
		x4	-	-	-	-
		x8	8 MHz	1	0	1

*Note:* See Clock Management Block diagram in [Figure 12](#)

## 22.2 Device ordering information and transfer of customer code

Customer code is made up of the FASTROM contents and the list of the selected options (if any). The FASTROM contents are to be sent on diskette, or by electronic means, with the S19 hexadecimal file generated by the development tool. All unused bytes must be set to FFh. The selected options are communicated to STMicroelectronics using the correctly completed OPTION LIST appended [on page 165](#).

Refer to application note AN1635 for information on the counter listing returned by ST after code has been transferred.

The STMicroelectronics Sales Organization will be pleased to provide detailed information on contractual points.

**Table 99. Order codes**

Order code	Program memory (bytes)	RAM (bytes)	Temp. range	Package
ST7FDALIF2M6	8K Flash	384	-40° C to 85° C	SO20
ST7PDALIF2M6	8K FASTROM	384	-40° C to 85° C	SO20

*Note:* Contact ST sales office for product availability.

ST7DALI FASTROM MICROCONTROLLER OPTION LIST			
(Last update: November 2004)			
Customer Address	.....		
Contact	.....		
Phone No	.....		
Reference/FASTROM Code*	.....		
*FASTROM code name is assigned by STMicroelectronics.			
FASTROM code must be sent in .S19 format. Hex extension cannot be processed.			
Device Type/Memory Size/Package (check only one option):			
FASTROM DEVICE:	8K		
SO20:	<input type="checkbox"/> ST7PDALIF2M6		
<b>Warning:</b> Addresses 1000h, 1001h, FFDEh and FFDFh are reserved areas for ST to program RCCR0 and RCCR1 (see <a href="#">Section 9.2 on page 32</a> ).			
Conditioning (check only one option):			
<input type="checkbox"/> Tape & Reel	<input type="checkbox"/> Tube		
Special Marking: <input type="checkbox"/> No <input type="checkbox"/> Yes " _____ "			
Authorized characters are letters, digits, '.', '-', '/' and spaces only.			
Maximum character count:			
DIP20/S020 (8 char. max) : _____			
Watchdog Selection:	<input type="checkbox"/> Software Activation	<input type="checkbox"/> Hardware Activation	
Watchdog Reset on Halt:	<input type="checkbox"/> Reset	<input type="checkbox"/> No Reset	
LVD Reset	<input type="checkbox"/> Disabled	<input type="checkbox"/> Enabled <input type="checkbox"/> Highest threshold <input type="checkbox"/> Medium threshold <input type="checkbox"/> Lowest threshold	
Sector 0 size:	<input type="checkbox"/> 0.5K	<input type="checkbox"/> 1K	<input type="checkbox"/> 2K <input type="checkbox"/> 4K
Readout Protection:	<input type="checkbox"/> Disabled	<input type="checkbox"/> Enabled	
FLASH write Protection:	<input type="checkbox"/> Disabled	<input type="checkbox"/> Enabled	
Clock Source Selection:	<input type="checkbox"/> Resonator: <input type="checkbox"/> VLP: Very Low power resonator (32 to 100 kHz) <input type="checkbox"/> LP: Low power resonator (1 to 2 MHz) <input type="checkbox"/> MP: Medium power resonator (2 to 4 MHz) <input type="checkbox"/> MS: Medium speed resonator (4 to 8 MHz) <input type="checkbox"/> HS: High speed resonator (8 to 16 MHz) <input type="checkbox"/> External Clock: <input type="checkbox"/> On OSC1 <input type="checkbox"/> On PB4 <input type="checkbox"/> Internal RC Oscillator		
PLL	<input type="checkbox"/> Disabled	<input type="checkbox"/> PLLx4	<input type="checkbox"/> PLLx8
PLL32	<input type="checkbox"/> Disabled	<input type="checkbox"/> Enabled	
Comments :	.....		
Supply Operating Range in the application:	.....		
Notes	.....		
Date:	.....		
Signature:	.....		
<b>Important note:</b> Not all configurations are available. See <a href="#">Table 98 on page 163</a> for authorized option byte combinations.			
Please download the latest version of this option list from: <a href="http://www.st.com">http://www.st.com</a>			

## 23 Important notes

### 23.1 Execution of BTJX instruction

When testing the address \$FF with the "BTJT" or "BTJF" instructions, the CPU may perform an incorrect operation when the relative jump is negative and performs an address page change.

To avoid this issue, including when using a C compiler, it is recommended to never use address \$00FF as a variable (using the linker parameter for example).

### 23.2 ADC conversion spurious results

Spurious conversions occur with a rate lower than 50 per million. Such conversions happen when the measured voltage is just between 2 consecutive digital values.

#### Workaround

A software filter should be implemented to remove erratic conversion results whenever they may cause unwanted consequences.

### 23.3 A/ D converter accuracy for first conversion

When the ADC is enabled after being powered down (for example when waking up from Halt, Active-halt or setting the ADON bit in the ADCCSR register), the first conversion (8-bit or 10-bit) accuracy does not meet the accuracy specified in the datasheet.

#### Workaround

In order to have the accuracy specified in the datasheet, the first conversion after a ADC switch-on has to be ignored.

### 23.4 Negative injection impact on ADC accuracy

Injecting a negative current on an analog input pins significantly reduces the accuracy of the AD Converter. Whenever necessary, the negative injection should be prevented by the addition of a Schottky diode between the concerned I/Os and ground.

Injecting a negative current on digital input pins degrades ADC accuracy especially if performed on a pin close to ADC channel in use.

### 23.5 Clearing active interrupts outside interrupt routine

When an active interrupt request occurs at the same time as the related flag or interrupt mask is being cleared, the CC register may be corrupted.

Concurrent interrupt context

The symptom does not occur when the interrupts are handled normally, i.e. when:

- The interrupt request is cleared (flag reset or interrupt mask) within its own interrupt routine
- The interrupt request is cleared (flag reset or interrupt mask) within any interrupt routine
- The interrupt request is cleared (flag reset or interrupt mask) in any part of the code while this interrupt is disabled

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

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

Ex:

```
SIM
reset flag or interrupt mask
RIM
```

## 23.6 Using PB4 as external interrupt

PB4 cannot be used as an external interrupt in Halt mode because the port pin PB4 is not active in this mode.

## 23.7 Timebase 2 interrupt in Slow mode

Timebase 2 interrupt is not available in slow mode.

## 24 Revision history

**Table 100. Document revision history**

Date	Revision	Changes
01-Aug-2003	1.3	<p>Changed status of the document</p> <p>Modified Caution to pin n°12 in Table 1, "Device Pin Description," on page 7</p> <p>Modified note 5 in section 4.4 on page 13</p> <p>Added "and the device can be reprogrammed" in section 4.5.1 on page 14</p> <p>Changed Figure 12 on page 25 (CLKIN/2, OSC/2)</p> <p>Added note in section 7.4 on page 26 (external clock source paragraph)</p> <p>Added note in the description of AWUPR[7:0] bits in section 9.6.0.1 on page 45</p> <p>Added text specifying that the watchdog counter is a free-running downcounter: Section 11.1.2 and section 11.1.3 on page 51</p> <p>Replaced ITFE by ITE in Figure 41 on page 70 (DCMCSR register) and in section 11.4.9 on page 73</p> <p>Changed section 13.3.1 on page 102: <math>f_{CLKIN}</math> instead of <math>f_{OSC}</math></p> <p>Changed section 13.7 on page 112</p> <p>Changed description of WDG HALT option bit (section 15.1 on page 132)</p> <p>Changed description of FMP_R option bit (section 15.1 on page 132)</p> <p>Changed Table 27, "Dedicated STMicroelectronics Development Tools," on page 135</p>



Table 100. Document revision history (continued)

Date	Revision	Changes
19-Nov-2004	2	<p>Reset delay in section 11.1.3 on page 51 changed to 30 <math>\mu</math>s</p> <p>Altered note 1 for section 13.2.3 on page 101 removing references to RESET</p> <p>Removed sentence relating to an effective change only after overflow for CK[1:0], <a href="#">page 60</a></p> <p>MOD00 replaced by 0Ex in <a href="#">Figure 36 on page 57</a></p> <p>Added Note 2 related to Exit from Active halt, <a href="#">section 11.2.5 on page 59</a></p> <p>Added illegal opcode detection to page 1, <a href="#">section 7.6 on page 29</a>, <a href="#">section 12 on page 94</a></p> <p>Clarification of Flash readout protection, <a href="#">section 4.5.1 on page 14</a></p> <p>Added note 4 and description relating to Total Percentage in Error and Amplifier Output Offset</p> <p>Variation to the ADC Characteristics subsection and table, <a href="#">page 126</a></p> <p>Added note 5 and description relating to Offset Variation in Temperature to ADC Characteristics subsection and table, <a href="#">page 126</a></p> <p>FPLL value of 1MHz quoted as Typical instead of a Minimum in <a href="#">section 13.3.4.1 on page 104</a></p> <p>Updated FSCK in <a href="#">section 13.10.1 on page 121</a> to <math>f_{CPU}/4</math> and <math>f_{CPU}/2</math></p> <p>Corrected <math>f_{CPU}</math> in Slow and slow wait modes in <a href="#">section 13.4.1 on page 108</a></p> <p>Max values updated for ADC Accuracy, <a href="#">page 124</a></p> <p>Notes indicating that PB4 cannot be used as an external interrupt in HALT mode, <a href="#">section 16.6 on page 138</a> and <a href="#">Section 8.3 PERIPHERAL INTERRUPTS on page 138</a></p> <p>Changed <a href="#">section 11.5.2 on page 79</a></p> <p>Changed <a href="#">section 11.5.3.3 on page 82</a></p> <p>Removed "optional" referring to VDD in <a href="#">Figure 4 on page 13</a></p> <p>Changed FMP_R option bit description in <a href="#">section 15.1 on page 130</a></p> <p>Added "Clearing active interrupts outside interrupt routine" on page 138</p> <p>Changed "Development Tools" on page 134</p> <p>Changed <a href="#">Figure 41 on page 70</a>: <math>f_{CPU}</math> instead of 8 MHz <math>f_{CPU}</math></p>

Table 100. Document revision history (continued)

Date	Revision	Changes
05-Feb-2009	3	<p>Updated <a href="#">Section 7.5: Access error handling on page 25</a></p> <p>Added caution in <a href="#">Section 9.6: Reset sequence manager (RSM) on page 38</a></p> <p>Renamed <math>f_{OSC}</math> to <math>f_{OSC2}</math> in <a href="#">Figure 12: Clock management block diagram on page 35</a></p> <p>Modified <a href="#">Section 14.6.3: Counter register low (CNTRL) on page 78</a></p> <p>Updated description of EOC bit in <a href="#">Section 18: 10-bit A/D converter (ADC) and added Section 18.4: Changing the conversion channel on page 115</a></p> <p>Updated EMC characteristics <a href="#">Section 20.7 on page 141</a></p> <p>Updated <a href="#">Table 63: Current characteristics on page 129</a></p> <p>Removed EMC protective circuitry in <a href="#">Figure 87</a> and <a href="#">Figure 88 on page 151</a> (device works correctly without these components)</p> <p>Replaced soldering information with ECOPACK reference in <a href="#">Section 21 on page 158</a></p> <p>Increased precision of package dimensions in inches to 4 decimal digits in <a href="#">Table 93 on page 159</a>.</p>

**Please Read Carefully:**

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

**UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.**

**UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER'S OWN RISK.**

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2009 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

[www.st.com](http://www.st.com)

