



# ST7LITE1

## 8-BIT MCU WITH SINGLE VOLTAGE FLASH MEMORY, DATA EEPROM, ADC, 4 TIMERS, SPI

### ■ Memories

- 4 Kbytes single voltage extended Flash (XFlash) Program memory with read-out protection, In-Circuit Programming and In-Application programming (ICP and IAP). 10K write/erase cycles guaranteed, data retention: 20 years at 55°C.
- 256 bytes RAM
- 128 bytes data E<sup>2</sup>PROM with read-out protection. 300K write/erase cycles guaranteed, data retention: 20 years at 55°C.

### ■ Clock, Reset and Supply Management

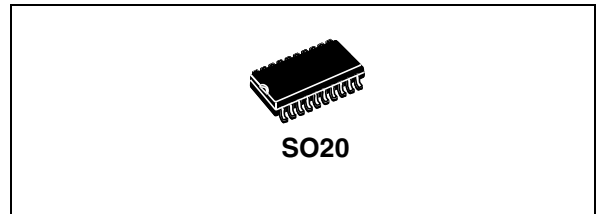
- Enhanced reset system
- Enhanced low voltage supervisor (LVD) for main supply and an auxiliary voltage detector (AVD) with interrupt capability for implementing safe power-down procedures
- Clock sources: Internal 1% RC oscillator (on some devices), crystal/ceramic resonator or external clock
- Internal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

### ■ I/O Ports

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

### ■ 4 Timers

- Configurable watchdog timer
- Two 8-bit Lite Timers with prescaler,



1 realtime base and 1 input capture

- One 12-bit Auto-reload Timer with 4 PWM outputs, input capture and output compare functions

### ■ Communication Interface

- SPI synchronous serial 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 precision for 0 to 430 mV (@ 5V V<sub>DD</sub>)
- 10-bit precision for 430 mV to 5V (@ 5V V<sub>DD</sub>)

### ■ 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)

### Device Summary

Features	ST7LITE10	ST7LITE15	ST7LITE19
Program memory - bytes		4K	
RAM (stack) - bytes		256 (128)	
Data EEPROM - bytes	-	-	128
Peripherals	Lite Timer with Watchdog, Autoreload Timer, SPI, 10-bit ADC with Op-Amp	Lite Timer with Watchdog, Autoreload Timer with 32-MHz input clock, SPI, 10-bit ADC with Op-Amp	
Operating Supply	2.4V to 5.5V		
CPU Frequency	Up to 8Mhz (w/ ext OSC up to 16MHz)	Up to 8Mhz (w/ ext OSC up to 16MHz and int 1MHz RC 1% PLLx8/4MHz)	
Operating Temperature	-40°C to +85°C		
Packages	SO20 300°		

---

# Table of Contents

---

<b>1 INTRODUCTION</b>	<b>5</b>
<b>2 PIN DESCRIPTION</b>	<b>6</b>
<b>3 REGISTER &amp; MEMORY MAP</b>	<b>10</b>
<b>4 FLASH PROGRAM MEMORY</b>	<b>13</b>
4.1 INTRODUCTION	13
4.2 MAIN FEATURES	13
4.3 PROGRAMMING MODES	13
4.4 ICC INTERFACE	14
4.5 MEMORY PROTECTION	15
4.6 RELATED DOCUMENTATION	15
4.7 REGISTER DESCRIPTION	15
<b>5 DATA EEPROM</b>	<b>16</b>
5.1 INTRODUCTION	16
5.2 MAIN FEATURES	16
5.3 MEMORY ACCESS	17
5.4 POWER SAVING MODES	19
5.5 ACCESS ERROR HANDLING	19
5.6 DATA EEPROM READ-OUT PROTECTION	19
5.7 REGISTER DESCRIPTION	20
<b>6 CENTRAL PROCESSING UNIT</b>	<b>21</b>
6.1 INTRODUCTION	21
6.2 MAIN FEATURES	21
6.3 CPU REGISTERS	21
<b>7 SUPPLY, RESET AND CLOCK MANAGEMENT</b>	<b>24</b>
7.1 INTERNAL RC OSCILLATOR ADJUSTMENT	24
7.2 PHASE LOCKED LOOP	24
7.3 REGISTER DESCRIPTION	25
7.4 MULTI-OSCILLATOR (MO)	27
7.5 RESET SEQUENCE MANAGER (RSM)	28
7.6 SYSTEM INTEGRITY MANAGEMENT (SI)	30
<b>8 INTERRUPTS</b>	<b>35</b>
8.1 NON MASKABLE SOFTWARE INTERRUPT	35
8.2 EXTERNAL INTERRUPTS	35
8.3 PERIPHERAL INTERRUPTS	35
<b>9 POWER SAVING MODES</b>	<b>39</b>
9.1 INTRODUCTION	39
9.2 SLOW MODE	39
9.3 WAIT MODE	40
9.4 HALT MODE	41
9.5 ACTIVE-HALT MODE	42
9.6 AUTO WAKE UP FROM HALT MODE	43
<b>10 I/O PORTS</b>	<b>47</b>

---

# Table of Contents

---

10.1	INTRODUCTION	47
10.2	FUNCTIONAL DESCRIPTION	47
10.3	I/O PORT IMPLEMENTATION	50
10.4	UNUSED I/O PINS	50
10.5	LOW POWER MODES	50
10.6	INTERRUPTS	50
10.7	DEVICE-SPECIFIC I/O PORT CONFIGURATION	51
<b>11</b>	<b>ON-CHIP PERIPHERALS</b>	<b>52</b>
11.1	WATCHDOG TIMER (WDG)	52
11.2	12-BIT AUTORELOAD TIMER 2 (AT2)	56
11.3	LITE TIMER 2 (LT2)	66
11.4	SERIAL PERIPHERAL INTERFACE (SPI)	71
11.5	10-BIT A/D CONVERTER (ADC)	82
<b>12</b>	<b>INSTRUCTION SET</b>	<b>86</b>
12.1	ST7 ADDRESSING MODES	86
12.2	INSTRUCTION GROUPS	89
<b>13</b>	<b>ELECTRICAL CHARACTERISTICS</b>	<b>92</b>
13.1	PARAMETER CONDITIONS	92
13.2	ABSOLUTE MAXIMUM RATINGS	93
13.3	OPERATING CONDITIONS	94
13.4	SUPPLY CURRENT CHARACTERISTICS	100
13.5	CLOCK AND TIMING CHARACTERISTICS	102
13.6	MEMORY CHARACTERISTICS	103
13.7	EMC CHARACTERISTICS	104
13.8	I/O PORT PIN CHARACTERISTICS	106
13.9	CONTROL PIN CHARACTERISTICS	111
13.10	COMMUNICATION INTERFACE CHARACTERISTICS	113
13.11	10-BIT ADC CHARACTERISTICS	115
<b>14</b>	<b>PACKAGE CHARACTERISTICS</b>	<b>119</b>
14.1	PACKAGE MECHANICAL DATA	119
14.2	SOLDERING AND GLUEABILITY INFORMATION	121
<b>15</b>	<b>DEVICE CONFIGURATION</b>	<b>122</b>
15.1	OPTION BYTES	122
15.2	DEVICE ORDERING INFORMATION	124
15.3	DEVELOPMENT TOOLS	125
15.4	ST7 APPLICATION NOTES	126
<b>16</b>	<b>IMPORTANT NOTES</b>	<b>129</b>
16.1	EXECUTION OF BTJX INSTRUCTION	129
16.2	ADC CONVERSION SPURIOUS RESULTS	129
16.3	A/D CONVERTER ACCURACY FOR FIRST CONVERSION	129
16.4	NEGATIVE INJECTION IMPACT ON ADC ACCURACY	129

16.5	CLEARING ACTIVE INTERRUPTS OUTSIDE INTERRUPT ROUTINE .....	129
16.6	USING PB4 AS EXTERNAL INTERRUPT .....	129
<b>17</b>	<b>REVISION HISTORY .....</b>	<b>130</b>

To obtain the most recent version of this datasheet,  
please check at [www.st.com>products>technical literature>datasheet](http://www.st.com>products>technical literature>datasheet)

Please also pay special attention to the Section “IMPORTANT NOTES” on page 128.

## 1 INTRODUCTION

The ST7LITE1 is a member of the ST7 microcontroller family. All ST7 devices are based on a common industry-standard 8-bit core, featuring an enhanced instruction set.

The ST7LITE1 features FLASH memory with byte-by-byte In-Circuit Programming (ICP) and In-Application Programming (IAP) capability.

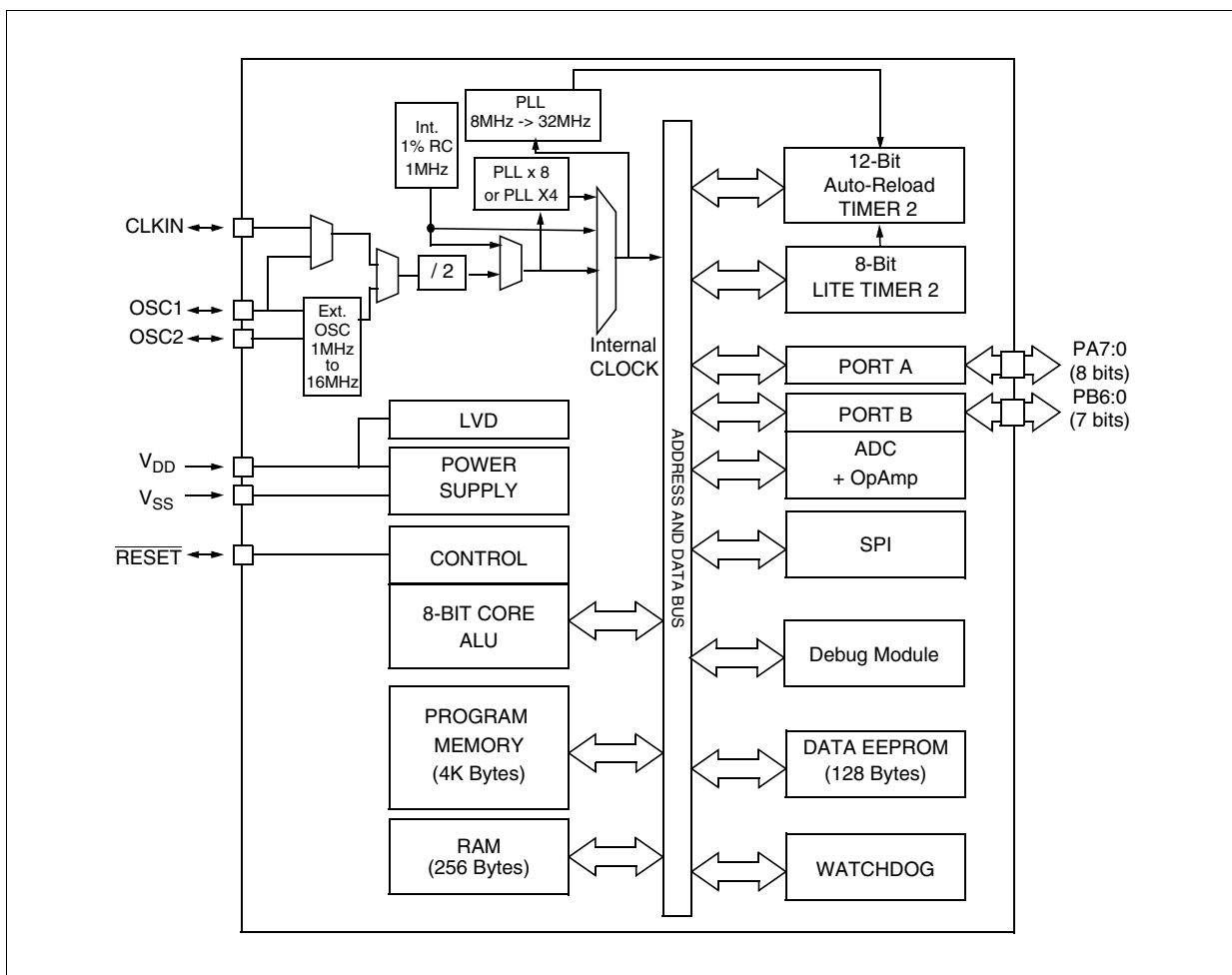
Under software control, the ST7LITE1 device can be placed in WAIT, SLOW, or HALT mode, reducing power consumption when the application is in idle or standby state.

The enhanced instruction set and addressing modes of the ST7 offer both power and flexibility to

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

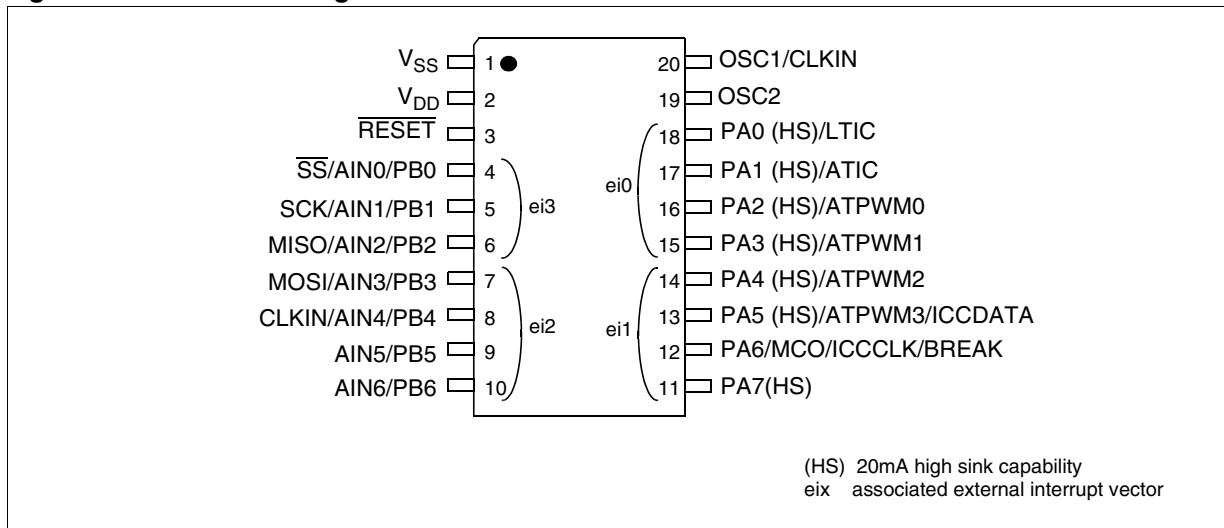
For easy reference, all parametric data are located in [section 13 on page 91](#). The devices feature an on-chip Debug Module (DM) to support in-circuit debugging (ICD). For a description of the DM registers, refer to the ST7 ICC Protocol Reference Manual.

**Figure 1. General Block Diagram**



## 2 PIN DESCRIPTION

Figure 2. 20-Pin SO Package Pinout



## PIN DESCRIPTION (Cont'd)

## Legend / Abbreviations for Table 1:

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 1. 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	<b>RESET</b>	I/O	<b>C<sub>T</sub></b>			X				X		Top priority non maskable interrupt (active low)
4	PB0/AIN0/ $\overline{SS}$	I/O	<b>C<sub>T</sub></b>	<b>X</b>	ei3			X	X	X	<b>Port B0</b>	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 13.2.2 on page 92</a>
5	PB1/AIN1/SCK	I/O	<b>C<sub>T</sub></b>	<b>X</b>				X	X	X	<b>Port B1</b>	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 13.2.2 on page 92</a>
6	PB2/AIN2/MISO	I/O	<b>C<sub>T</sub></b>	<b>X</b>				X	X	X	<b>Port B2</b>	ADC Analog Input 2 or SPI Master In/ Slave Out Data
7	PB3/AIN3/MOSI	I/O	<b>C<sub>T</sub></b>	<b>X</b>	ei2			X	X	X	<b>Port B3</b>	ADC Analog Input 3 or SPI Master Out / Slave In Data
8	PB4/AIN4/CLKIN	I/O	<b>C<sub>T</sub></b>	<b>X</b>				X	X	X	<b>Port B4</b>	ADC Analog Input 4 or External clock input
9	PB5/AIN5	I/O	<b>C<sub>T</sub></b>	<b>X</b>				X	X	X	<b>Port B5</b>	ADC Analog Input 5
10	PB6/AIN6	I/O	<b>C<sub>T</sub></b>	<b>X</b>				X	X	X	<b>Port B6</b>	ADC Analog Input 6
11	PA7	I/O	<b>C<sub>T</sub></b>	HS	<b>X</b>	ei1			X	X	<b>Port A7</b>	

Pin No.	Pin Name	Type	Level		Port / Control						Main Function (after reset)	Alternate Function		
			Input	Output	Input				Output					
					float	wpu	int	ana	OD	PP				
12	PA6/MCO/ ICCCLK/BREAK	I/O		C <sub>T</sub>	X						X	X	<b>Port A6</b>	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/ ICCDATA	I/O	C <sub>T</sub>	HS	X					X	X	<b>Port A5</b>	In Circuit Communication Data	
14	PA4	I/O	C <sub>T</sub>	HS	X					X	X	<b>Port A4</b>		
15	PA3/ATPWM1	I/O	C <sub>T</sub>	HS	X					X	X	<b>Port A3</b>	Auto-Reload Timer PWM1	
16	PA2/ATPWM0	I/O	C <sub>T</sub>	HS	X					X	X	<b>Port A2</b>	Auto-Reload Timer PWM0	
17	PA1/ATIC	I/O	C <sub>T</sub>	HS	X					X	X	<b>Port A1</b>	Auto-Reload Timer Input Capture	
18	PA0/LTIC	I/O	C <sub>T</sub>	HS	X					X	X	<b>Port A0</b>	Lite Timer Input Capture	
19	OSC2	O											Resonator oscillator inverter output	
20	OSC1/CLKIN	I											Resonator oscillator inverter input or External clock input	



### 3 REGISTER & 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, 256 bytes of RAM, 128 bytes of data EEPROM and 4 Kbytes of flash program memory. The RAM space includes up to 128 bytes for the stack from 180h to 1FFh.

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

The Flash memory contains two sectors (see [Figure 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 15.1 on page 121](#)).

**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**

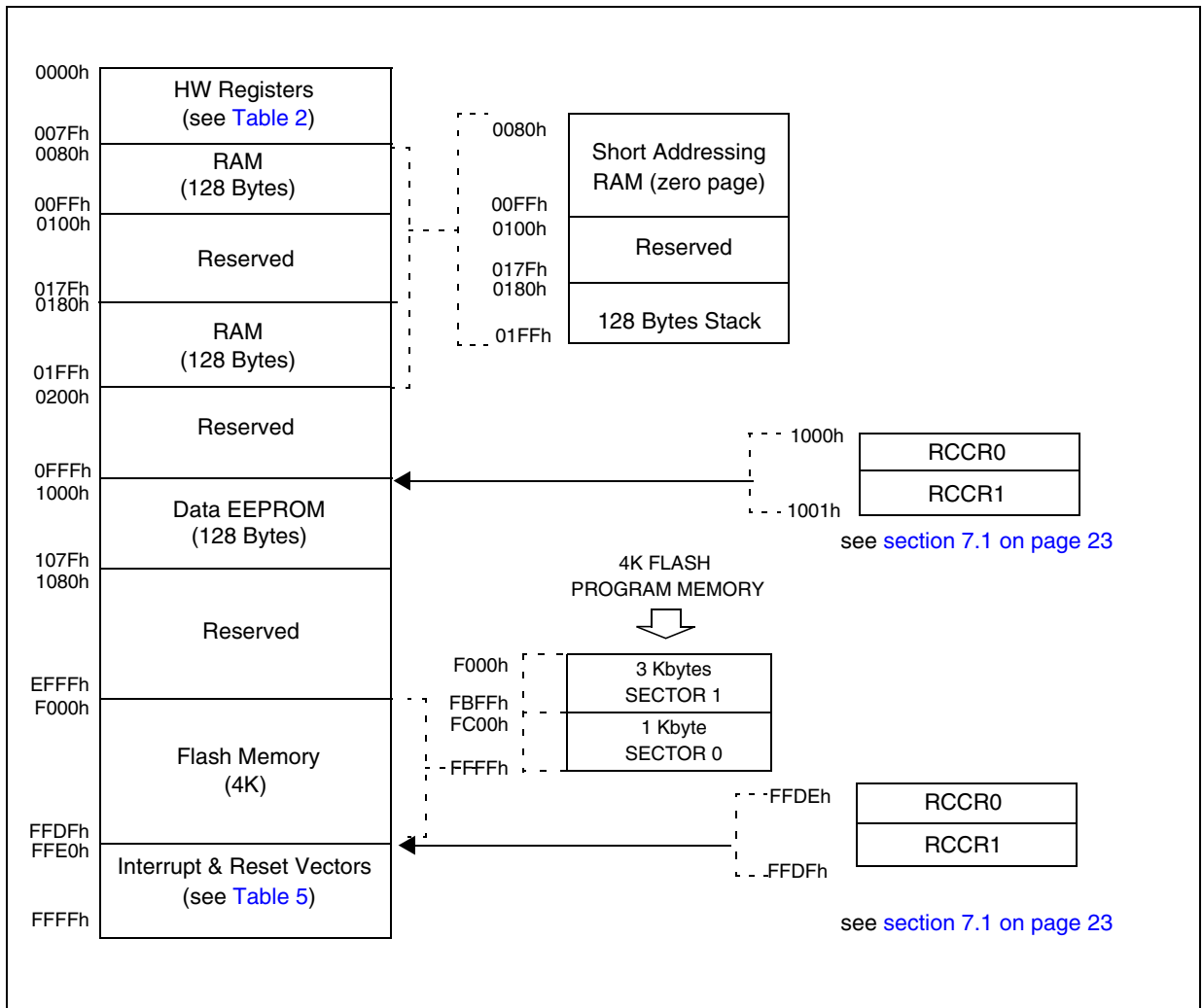


Table 2. Hardware Register Map

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	0Fh	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 0000h	R/W
		LTICR	Lite Timer Input Capture Register	xxh	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 0000h	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	
	TRANCR	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	SPI DR	SPI Data I/O Register	xxh	R/W
		SPI CR	SPI Control Register	0xh	R/W
		SPI CSR	SPI Control Status Register	00h	R/W
0034h 0035h 0036h	ADC	ADCCSR	A/D Control Status Register	00h	R/W
		ADCDRH	A/D Data Register High	xxh	Read Only
		ADC DRL	A/D Amplifier Control/Data Low Register	0xh	R/W

Address	Block	Register Label	Register Name	Reset Status	Remarks
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 0XX0h	R/W R/W
003Bh	Reserved area (1 byte)				
003Ch	ITC	EISR	External Interrupt Selection Register	0Ch	R/W
003Dh to 0048h	Reserved area (12 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)				

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

**Notes:**

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 ICC reference manual.

## 4 FLASH PROGRAM MEMORY

### 4.1 Introduction

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

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

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

### 4.2 Main Features

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

### 4.3 PROGRAMMING MODES

The ST7 can be programmed in three different ways:

- Insertion in a programming tool. In this mode, FLASH sectors 0 and 1, option byte row 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.

#### 4.3.1 In-Circuit Programming (ICP)

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

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

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

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

#### 4.3.2 In Application Programming (IAP)

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

This mode is fully controlled by user software. This allows it to be adapted to the user application, (user-defined strategy for entering programming mode, choice of communications protocol used to fetch the data to be stored etc).

IAP mode can be used to program any memory areas except Sector 0, which is write/erase protected to allow recovery in case errors occur during the programming operation.

## FLASH PROGRAM MEMORY (Cont'd)

### 4.4 ICC interface

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

- $\overline{\text{RESET}}$ : device reset
- $V_{SS}$ : device power supply ground
- ICCCLK: ICC output serial clock pin
- ICCDATA: ICC input serial data pin
- OSC1: main clock input for external source (not required on devices without OSC1/OSC2 pins)
- $V_{DD}$ : application board power supply (optional, see Note 3)

#### Notes:

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.

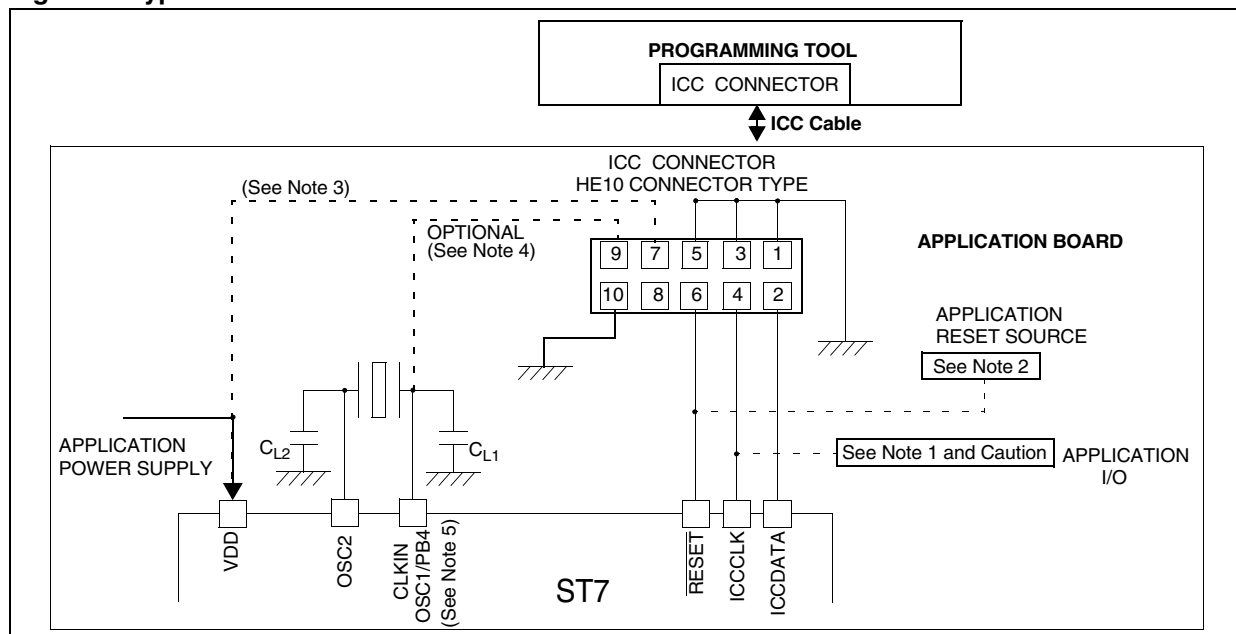
3. The use of Pin 7 of the ICC connector depends on the Programming Tool architecture. This pin must be connected when using most ST Programming Tools (it is used to monitor the application power supply). Please refer to the Programming Tool manual.

4. Pin 9 has to be connected to the OSC1 pin of the ST7 when the clock is not available in the application or if the selected clock option is not programmed in the option byte. ST7 devices with multi-oscillator capability need to have OSC2 grounded in this case.

5. With the ICP option disabled with ST7 MDT10-EPB that 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



**FLASH PROGRAM MEMORY** (Cont'd)

**4.5 Memory Protection**

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

**4.5.1 Read out Protection**

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 re-programming the option. In this case, both program and data E<sup>2</sup> memory are automatically erased and the device can be reprogrammed.

Read-out 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.

**4.5.2 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.

**Warning:** 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.

**4.6 Related Documentation**

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

**4.7 Register Description**

**FLASH CONTROL/STATUS REGISTER (FCSR)**

Read/Write

Reset Value: 000 0000 (00h)

1st RASS Key: 0101 0110 (56h)

2nd RASS Key: 1010 1110 (AEh)

7							0
0	0	0	0	0	OPT	LAT	PGM

**Note:** 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.

## 5 DATA EEPROM

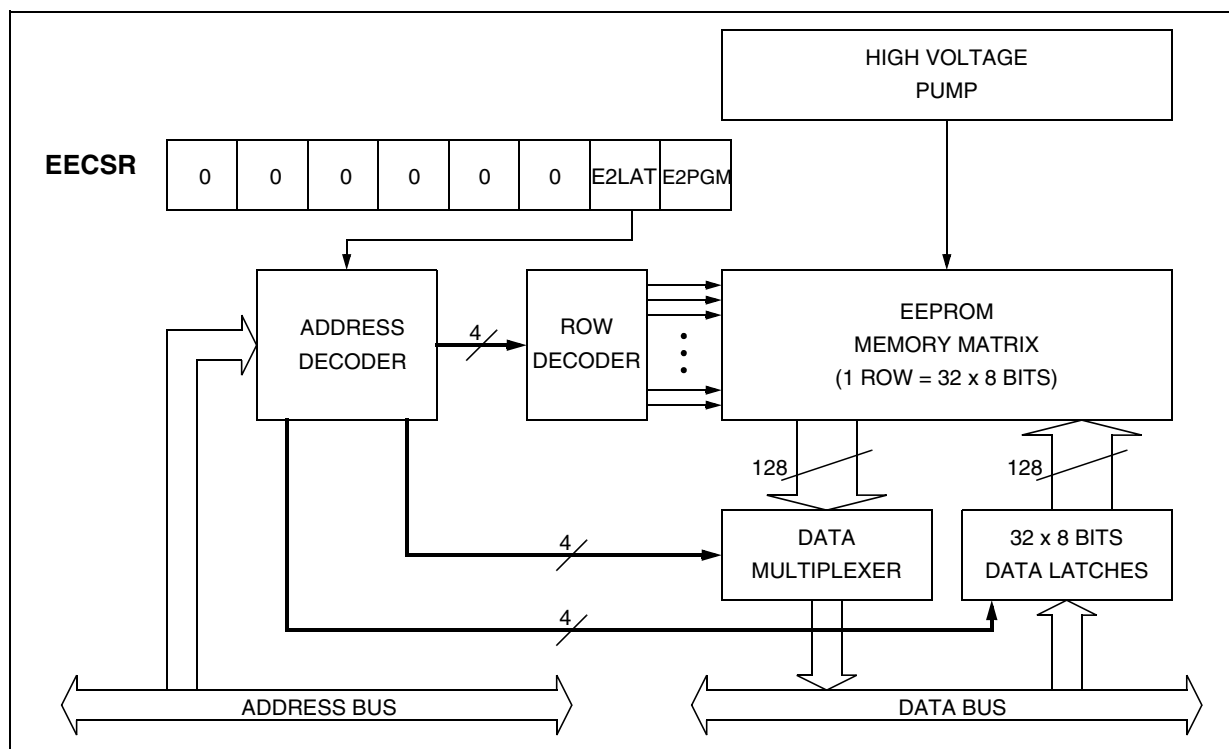
### 5.1 INTRODUCTION

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.

### 5.2 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



DATA EEPROM (Cont'd)

5.3 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. In a read cycle, the byte to be accessed is put on the data bus in less than 1 CPU clock cycle. This means that reading data from EEPROM takes the same time as reading data from EPROM, but this memory cannot be used to execute machine code.

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.

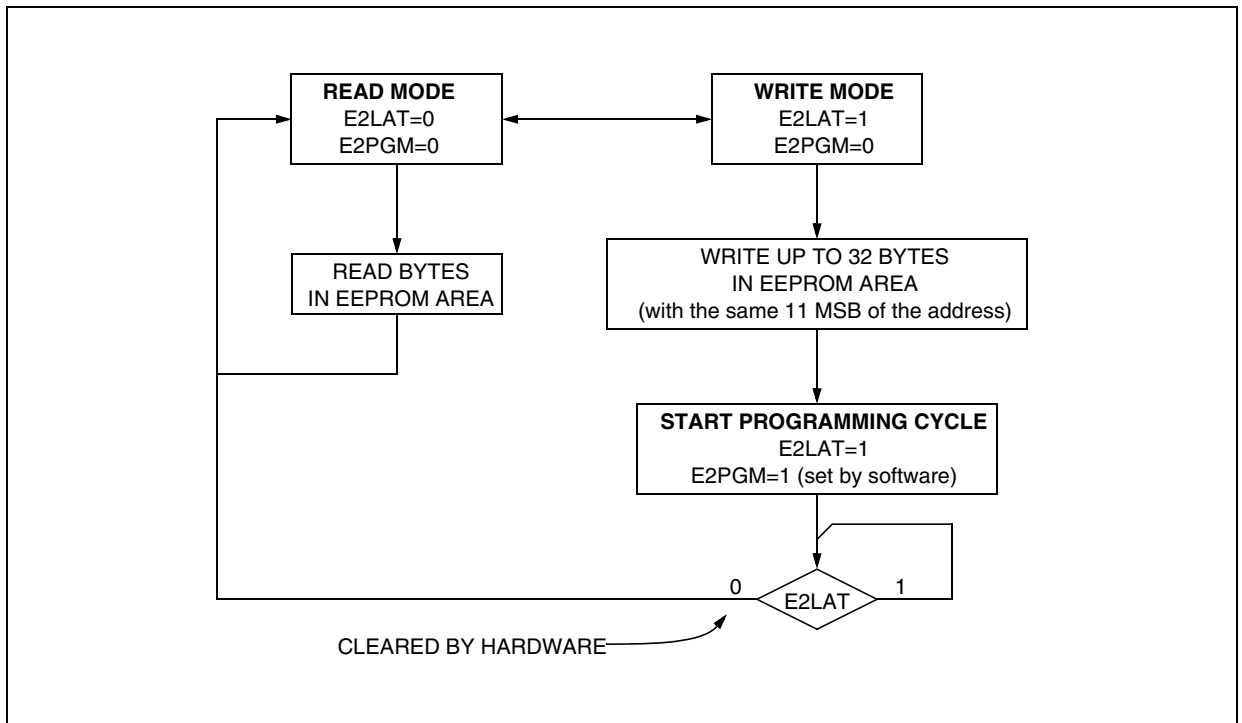
When PGM bit is set by the software, all the previous bytes written in the data latches (up to 32) are programmed in the EEPROM cells. The effective high address (row) is determined by the last EEPROM write sequence. To avoid wrong programming, the user must take care that all the bytes written between 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.

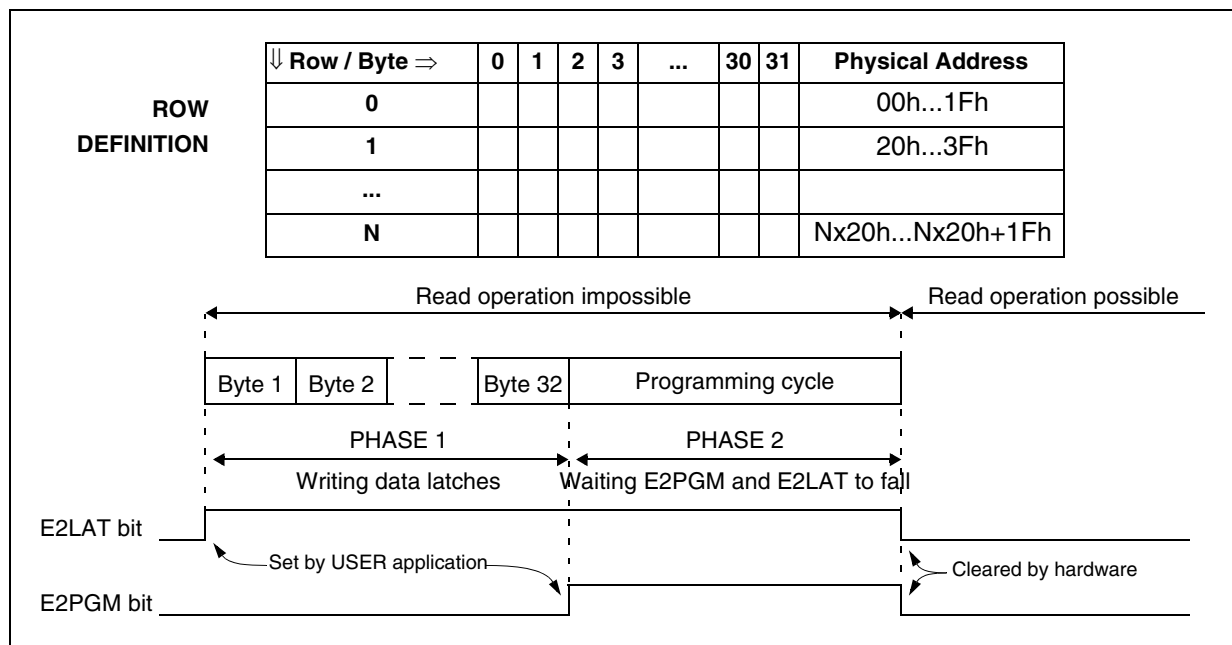
It is not possible to read the latched data. This note is illustrated by the Figure 8.

Figure 6. Data EEPROM Programming Flowchart





## DATA EEPROM (Cont'd)

Figure 7. Data E<sup>2</sup>PROM Write Operation

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

DATA EEPROM (Cont'd)

5.4 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.

5.5 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 software/RESET action), the memory data will not be guaranteed.

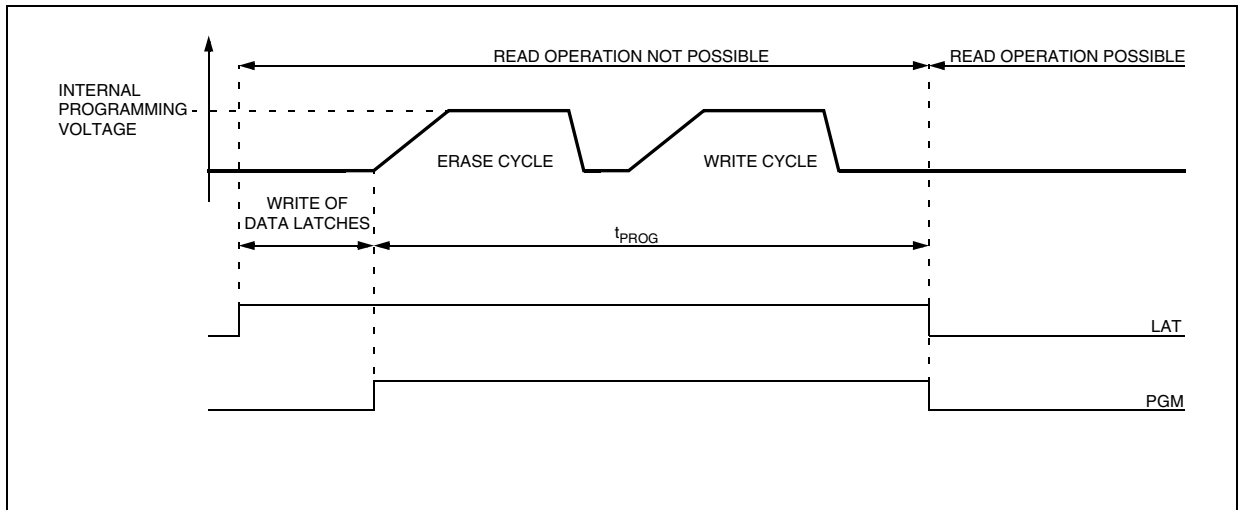
5.6 Data EEPROM Read-out Protection

The read-out protection is enabled through an option bit (see section 15.1 on page 121).

When this option is selected, the programs and data stored in the EEPROM memory are protected against read-out (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



**DATA EEPROM (Cont'd)****5.7 REGISTER DESCRIPTION****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 3. 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

## 6 CENTRAL PROCESSING UNIT

### 6.1 INTRODUCTION

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

### 6.2 MAIN FEATURES

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

### 6.3 CPU REGISTERS

The 6 CPU registers shown in [Figure 9](#) are not present in the memory mapping and are accessed by specific instructions.

#### 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.

#### Index Registers (X and Y)

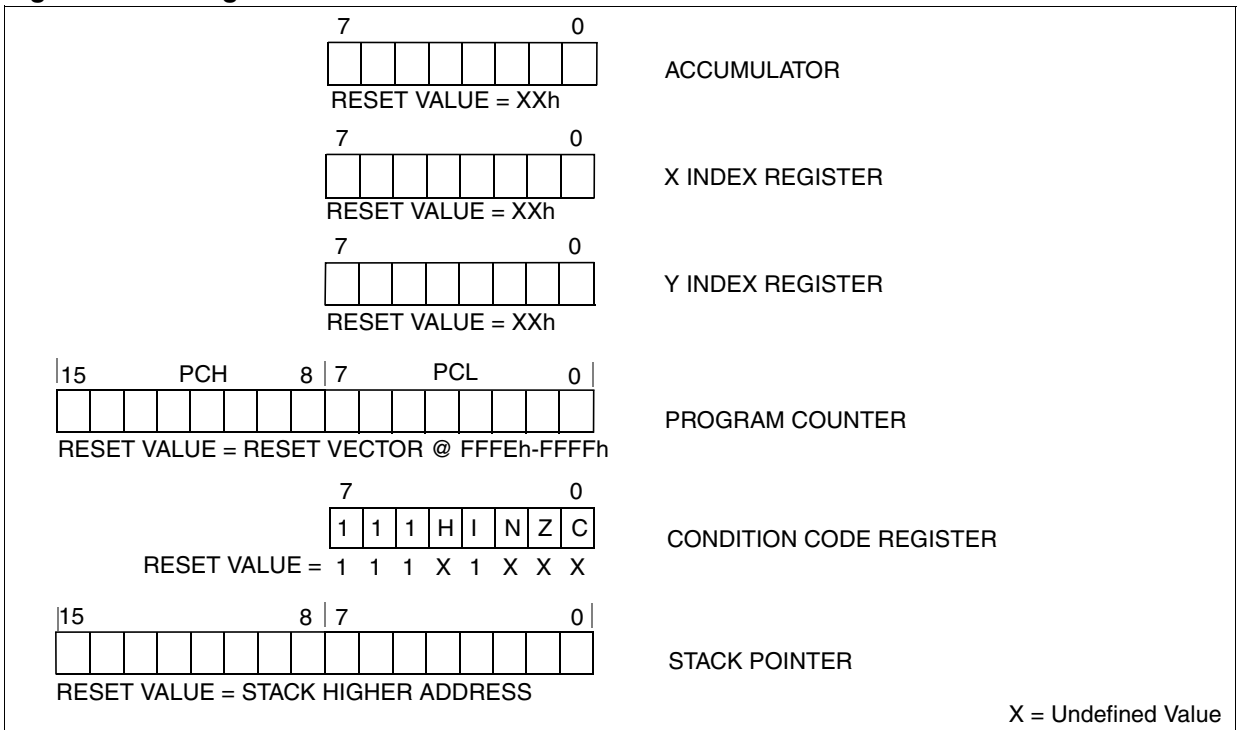
In indexed addressing modes, these 8-bit registers are used to create either effective addresses or temporary storage areas for data manipulation. (The Cross-Assembler generates a precede instruction (PRE) to indicate that the following instruction refers to the Y register.)

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

#### 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).

**Figure 9. CPU Registers**



**CPU REGISTERS** (Cont'd)**CONDITION CODE REGISTER (CC)**

Read/Write

Reset Value: 111x1xxx

7							0
1	1	1	H	I	N	Z	C

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

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

**Bit 4 = H Half carry.**

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

0: No half carry has occurred.

1: A half carry has occurred.

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

**Bit 3 = I Interrupt mask.**

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

0: Interrupts are enabled.

1: Interrupts are disabled.

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

**Note:** Interrupts requested while I is set are latched and can be processed when I is cleared. By default an interrupt routine is not interruptable

because the I bit is set by hardware at the start of the routine and reset by the IRET instruction at the end of the routine. If the I bit is cleared by software in the interrupt routine, pending interrupts are serviced regardless of the priority level of the current interrupt routine.

**Bit 2 = N Negative.**

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 7<sup>th</sup> bit of the result.

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

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

This bit is accessed by the JRMI and JRPL instructions.

**Bit 1 = Z Zero.**

This bit is set and cleared by hardware. This bit indicates that the result of the last arithmetic, logical or data manipulation is zero.

0: The result of the last operation is different from zero.

1: The result of the last operation is zero.

This bit is accessed by the JREQ and JRNE test instructions.

**Bit 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.

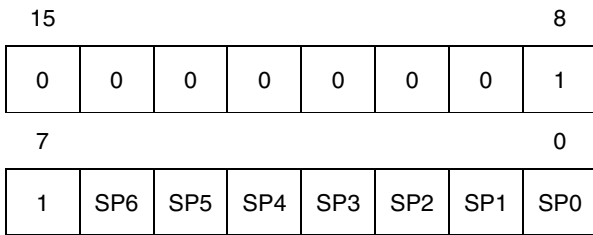
This bit is driven by the SCF and RCF instructions and tested by the JRC and JRNC instructions. It is also affected by the "bit test and branch", shift and rotate instructions.

**CPU REGISTERS** (Cont'd)

**STACK POINTER (SP)**

Read/Write

Reset Value: 01FFh



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 SP6 to SP0 bits are set) which is the stack higher address.

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

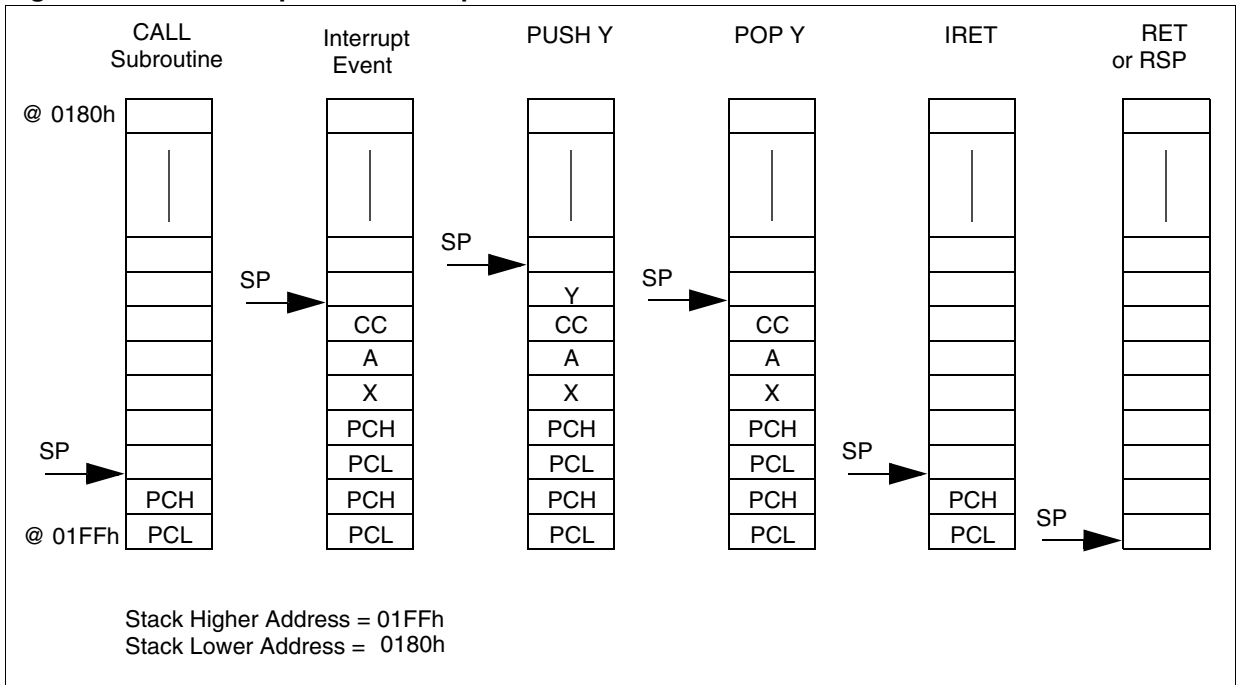
**Note:** When the lower limit is exceeded, the Stack Pointer wraps around to the stack upper limit, without indicating the stack overflow. The previously stored information is then overwritten and therefore lost. The stack also wraps in case of an underflow.

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



## 7 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.

### Main features

- Clock Management
  - 1 MHz internal RC oscillator (enabled by option byte, available on ST7LITE15 and ST7LITE19 devices only)
  - 1 to 16 MHz or 32kHz 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)

### 7.1 INTERNAL RC OSCILLATOR ADJUSTMENT

The device contains an internal RC oscillator with an accuracy of 1% for a given device, temperature and voltage range (4.5V-5.5V). 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 5V  $V_{DD}$  supply voltages at 25°C, as shown in the following table.

RCCR	Conditions	ST7LITE19 Address	ST7LITE15 Address
RCCR0	$V_{DD}=5V$ $T_A=25^{\circ}C$ $f_{RC}=1MHz$	1000h and FFDEh	FFDEh
RCCR1	$V_{DD}=3V$ $T_A=25^{\circ}C$ $f_{RC}=700KHz$	1001h and FFDFh	FFDFh

### Note:

- See “ELECTRICAL CHARACTERISTICS” on page 91. for more information on the frequency and accuracy of the RC oscillator.
- To improve clock stability, it is recommended to place a decoupling capacitor between the  $V_{DD}$  and  $V_{SS}$  pins.
- 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.
- RCCR0 and RCCR1 calibration values will be erased if the read-out protection bit is reset after it has been set. See “Read out Protection” on page 14.

**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.

### 7.2 PHASE LOCKED LOOP

The PLL can be used to multiply a 1MHz frequency from the RC oscillator or the external clock by 4 or 8 to obtain  $f_{OSC}$  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 2.4V to 3.3V range
- The x8 PLL is intended for operation with  $V_{DD}$  in the 3.3V to 5.5V range

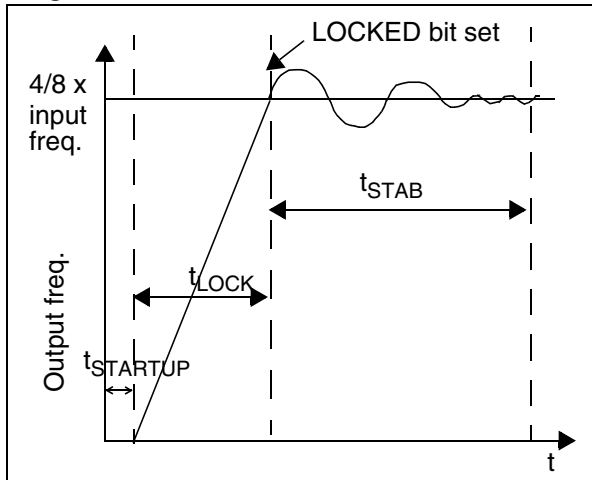
Refer to [Section 15.1](#) for the option byte description.

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

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

PHASE LOCKED LOOP (Cont'd)

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  $t_{STARTUP}$ .

When the PLL output signal reaches the operating frequency, the LOCKED bit in the SICSCR register is set. Full PLL accuracy ( $ACC_{PLL}$ ) is reached after a stabilization time of  $t_{STAB}$  (see Figure 11 and 13.3.4 Internal RC Oscillator and PLL)

Refer to section 7.6.4 on page 33 for a description of the LOCKED bit in the SICSCR register.

7.3 REGISTER DESCRIPTION

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  $f_{OSC}$  or  $f_{OSC}/32$ .

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

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

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.

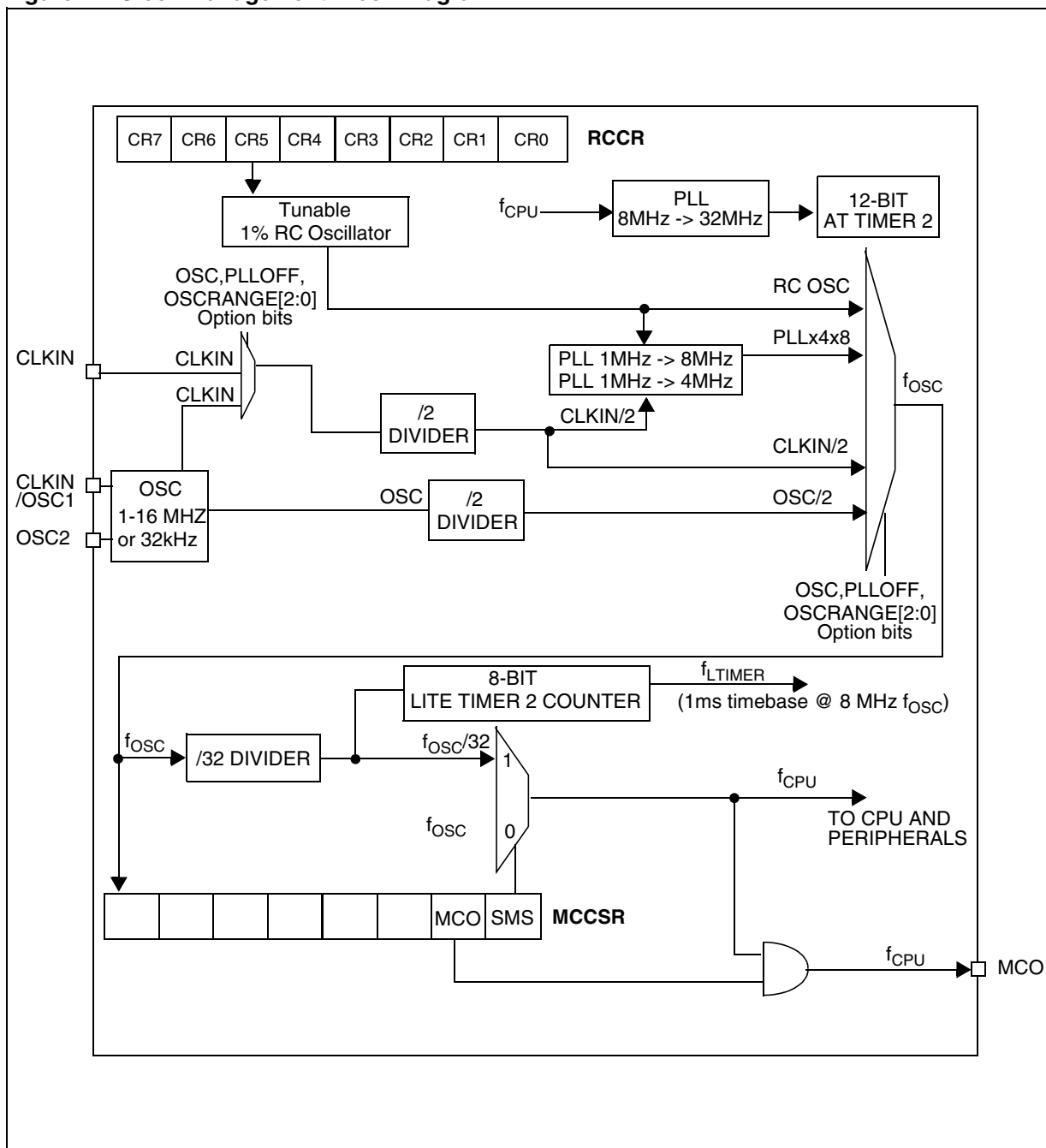
00h = maximum available frequency

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



7.4 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 16MHz or 32kHz):

- 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 4. 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 15.1 on page 121 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 4. ST7 Clock Sources

Hardware Configuration	
External Clock	
Crystal/Ceramic Resonators	
Internal RC Oscillator	

## 7.5 RESET SEQUENCE MANAGER (RSM)

### 7.5.1 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

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.

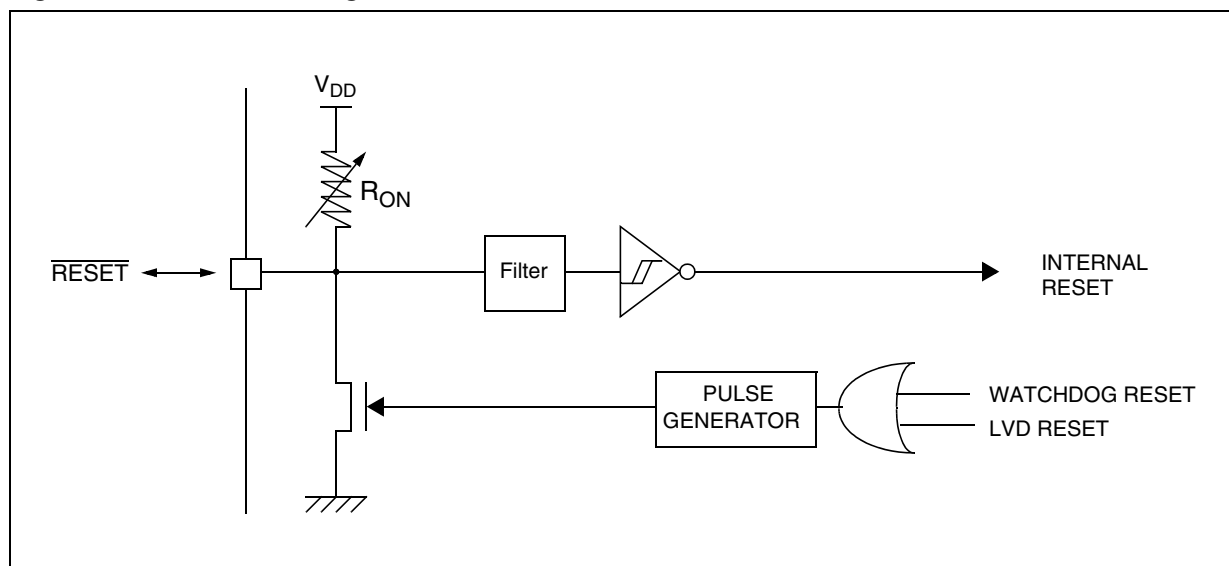
The basic RESET sequence consists of 3 phases as shown in Figure 13:

- 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:

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

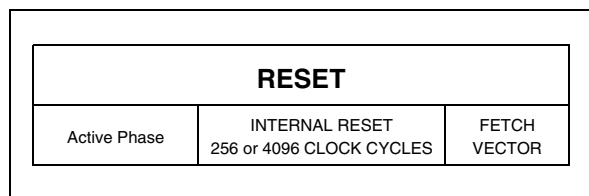
Figure 14. Reset Block Diagram



The RESET vector fetch phase duration is 2 clock cycles.

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



### 7.5.2 Asynchronous External $\overline{\text{RESET}}$ pin

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

**RESET SEQUENCE MANAGER (Cont'd)**

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.

**7.5.3 External Power-On RESET**

If the LVD is disabled by option byte, to start up the microcontroller correctly, the user must ensure by means of an external reset circuit that the reset signal is held low until  $V_{DD}$  is over the minimum level specified for the selected  $f_{OSC}$  frequency.

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

**7.5.4 Internal Low Voltage Detector (LVD) RESET**

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

- Power-On RESET
- Voltage Drop RESET

The device  $\overline{\text{RESET}}$  pin acts as an output that is pulled low when  $V_{DD} < V_{IT+}$  (rising edge) or  $V_{DD} < V_{IT-}$  (falling edge) as shown in Figure 15.

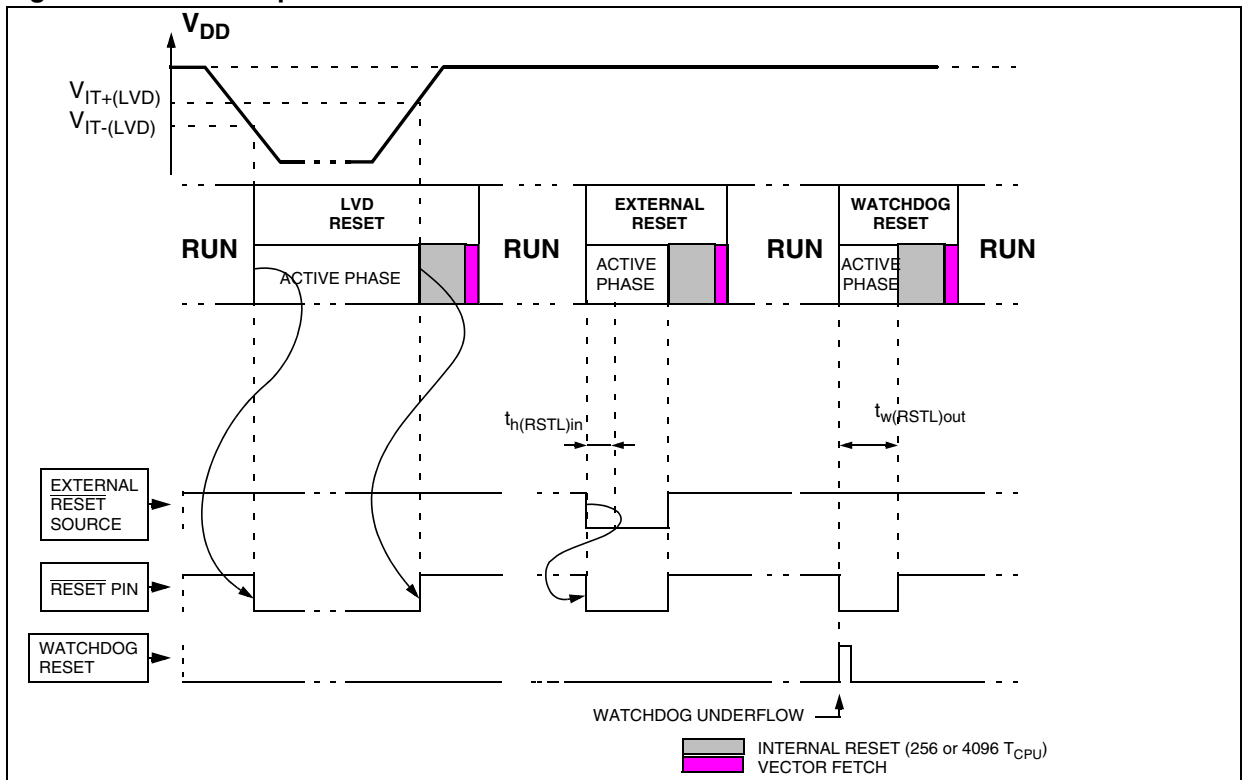
The LVD filters spikes on  $V_{DD}$  larger than  $t_{g(VDD)}$  to avoid parasitic resets.

**7.5.5 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  $t_{w(RSTL)out}$ .

**Figure 15. RESET Sequences**



## 7.6 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 [section 12.2.1 on page 88](#) for further details.

### 7.6.1 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.

#### Notes:

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

The LVD is an optional function which can be selected by option byte.

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**

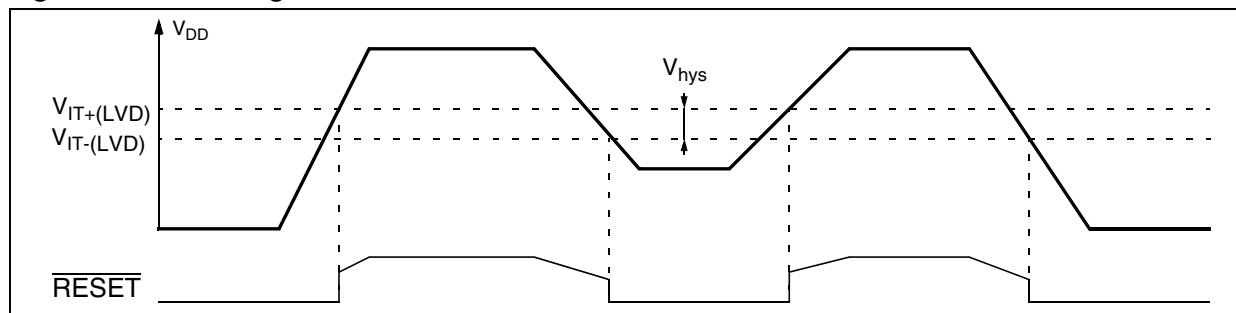
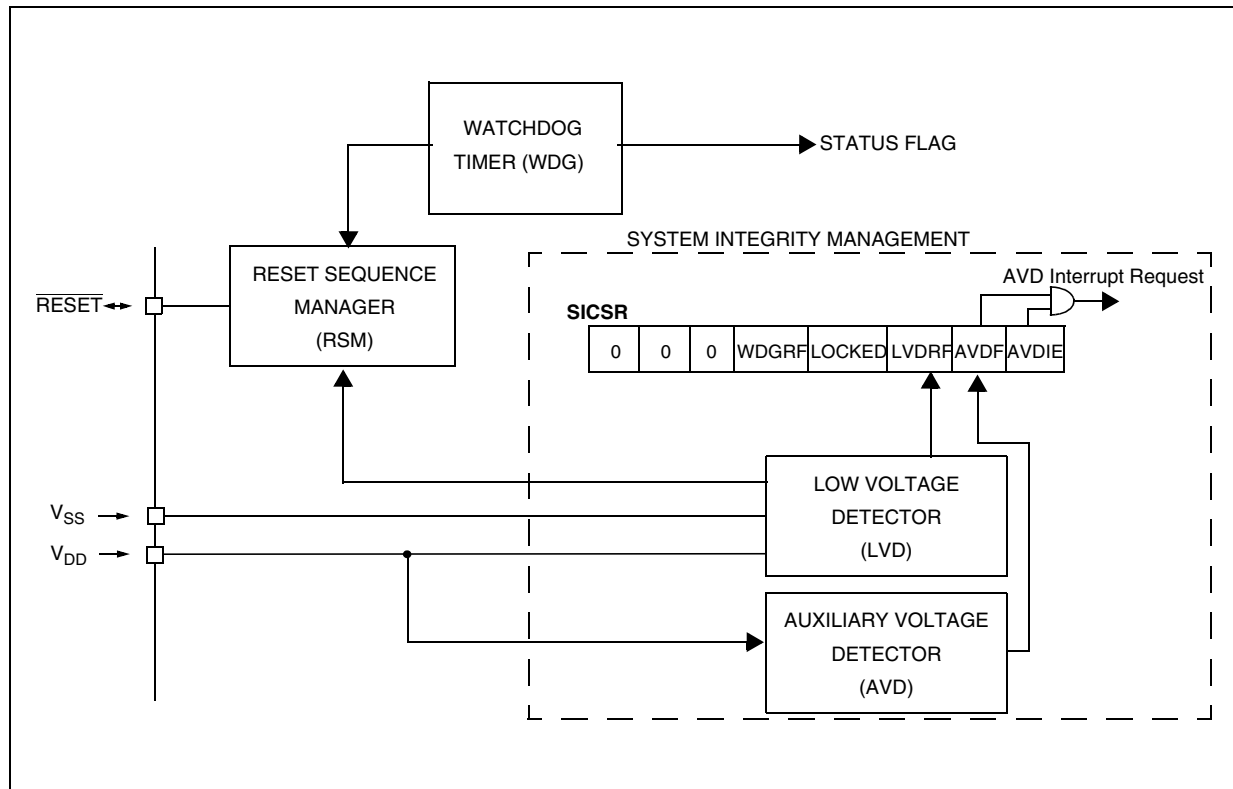


Figure 17. Reset and Supply Management Block Diagram



## SYSTEM INTEGRITY MANAGEMENT (Cont'd)

### 7.6.2 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 SICSR register. This bit is read only.

**Caution:** The AVD functions only if the LVD is en-

abled through the option byte.

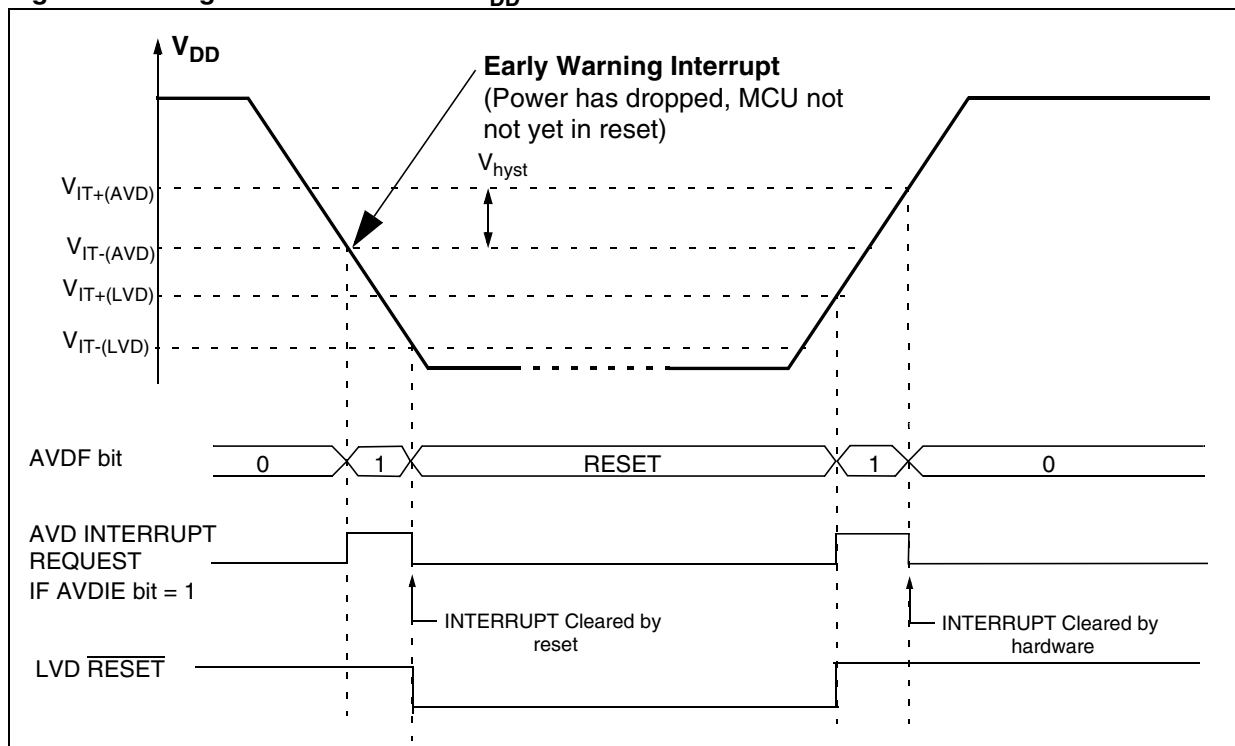
#### 7.6.2.1 Monitoring the $V_{DD}$ Main Supply

The AVD voltage threshold value is relative to the selected LVD threshold configured by option byte (see [section 15.1 on page 121](#)).

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  $V_{DD}$**



**SYSTEM INTEGRITY MANAGEMENT (Cont'd)****7.6.3 Low Power Modes**

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.

set and the interrupt mask in the CC register is reset (RIM instruction).

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
AVD event	AVDF	AVDIE	Yes	No

**7.6.3.1 Interrupts**

The AVD interrupt event generates an interrupt if the corresponding Enable Control Bit (AVDIE) is



## SYSTEM INTEGRITY MANAGEMENT (Cont'd)

### 7.6.4 Register Description

#### SYSTEM INTEGRITY (SI) CONTROL/STATUS REGISTER (SICSR)

Read/Write

Reset Value: 0000 0xx0 (0xh)

7							0
0	0	0	WDG RF	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.

RESET Sources	LVDRF	WDGRF
External 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 [Section 7.6.2.1](#) 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

### Application notes

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.

## 8 INTERRUPTS

The ST7 core may be interrupted by one of two different methods: maskable hardware interrupts as listed in the Interrupt Mapping Table 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 interrupts subsection).

**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 the Interrupt Mapping Table 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 the Interrupt Mapping Table).

### 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 the Interrupt Mapping Table).

### 8.1 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](#).

### 8.2 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 miscellaneous register or interrupt register (if available).

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 Miscellaneous or Interrupt register (if available) applies to the ei source.

### 8.3 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.

## INTERRUPTS (Cont'd)

Figure 19. Interrupt Processing Flowchart

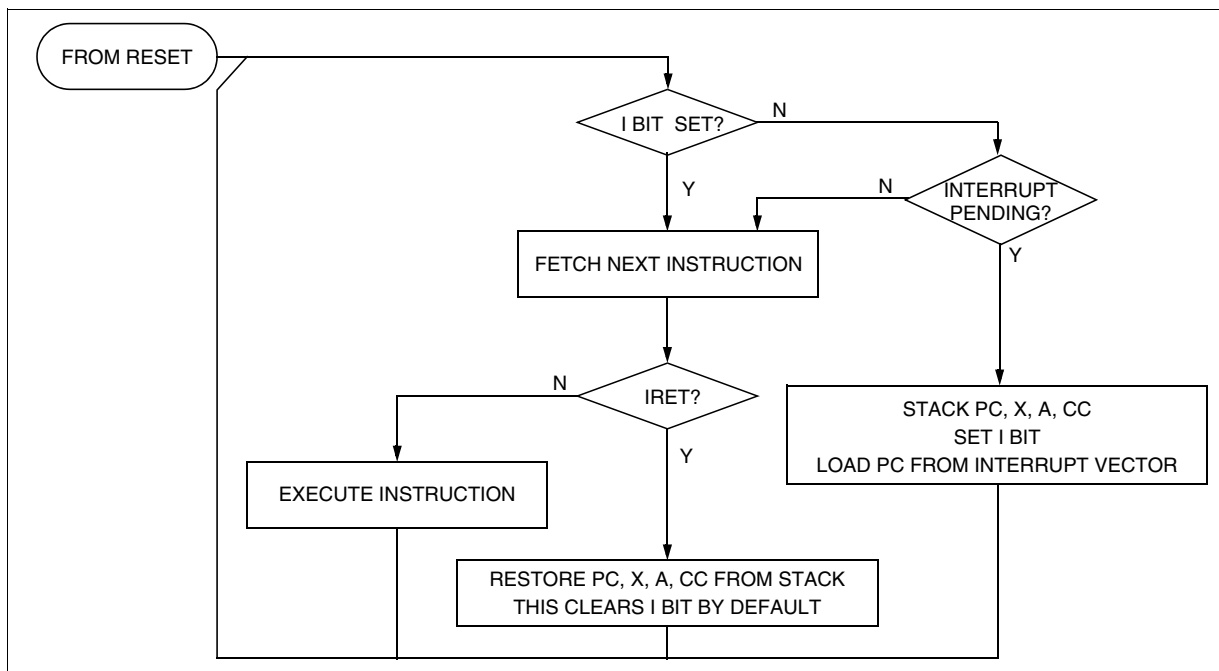


Table 5. Interrupt Mapping

N°	Source Block	Description	Register Label	Priority Order	Exit from HALT or AWUFH	Exit from ACTIVE -HALT	Address Vector	
	RESET	Reset	N/A	Highest Priority ↓ Lowest Priority	yes	yes	FFFEh-FFFFh	
	TRAP	Software Interrupt	N/A		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		Not used					FFEEh-FFEFh	
7	SI	AVD interrupt	SICSR		no		no	FFECh-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	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		

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

**INTERRUPTS** (Cont'd)

**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 6](#).

Bit 5:4 = **IS2[1:0]** *ei2 sensitivity*  
 These bits define the interrupt sensitivity for ei2 (Port B3) according to [Table 6](#).

Bit 3:2 = **IS1[1:0]** *ei1 sensitivity*  
 These bits define the interrupt sensitivity for ei1 (Port A7) according to [Table 6](#).

Bit 1:0 = **IS0[1:0]** *ei0 sensitivity*  
 These bits define the interrupt sensitivity for ei0 (Port A0) according to [Table 6](#).

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

**Table 6. 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

**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.

**External Interrupt I/O pin selection**

ei31	ei30	I/O Pin
0	0	PB0 *
0	1	PB1
1	0	PB2

\* Reset State

Bit 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.

**External Interrupt I/O pin selection**

ei21	ei20	I/O Pin
0	0	PB3 *
0	1	PB4 <sup>1)</sup>
1	0	PB5
1	1	PB6

\* Reset State

1) Note that PB4 cannot be used as an external interrupt in HALT mode.

**INTERRUPTS** (Cont'd)

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.

**External Interrupt I/O pin selection**

ei11	ei10	I/O Pin
0	0	PA4
0	1	PA5
1	0	PA6
1	1	PA7*

\* Reset State

Bit 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.

**External Interrupt I/O pin selection**

ei01	ei00	I/O Pin
0	0	PA0 *
0	1	PA1
1	0	PA2
1	1	PA3

\* Reset State

Bits 1:0 = Reserved.

## 9 POWER SAVING MODES

### 9.1 INTRODUCTION

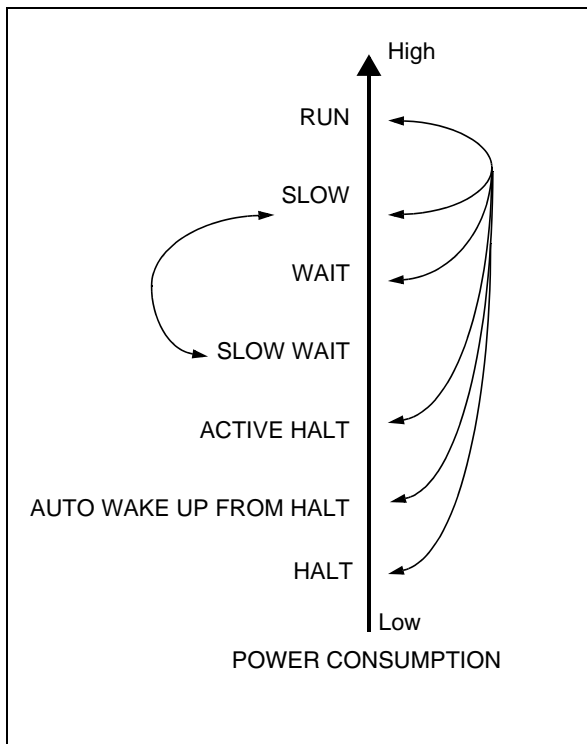
To give a large measure of flexibility to the application in terms of power consumption, five main power saving modes are implemented in the ST7 (see Figure 20):

- 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**



### 9.2 SLOW MODE

This mode has two targets:

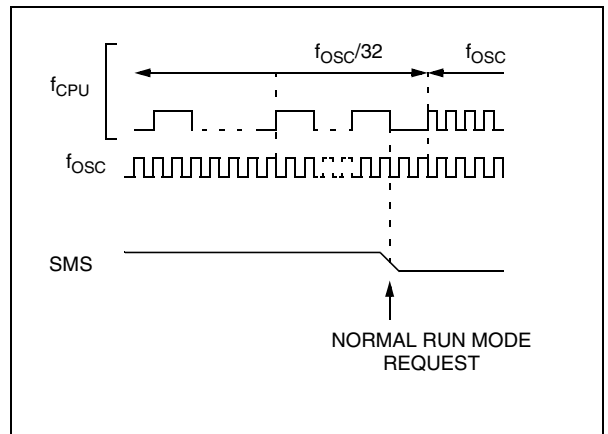
- To reduce power consumption by decreasing the internal clock in the device,
- To adapt the internal clock frequency ( $f_{CPU}$ ) to the available supply voltage.

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**



## POWER SAVING MODES (Cont'd)

## 9.3 WAIT MODE

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

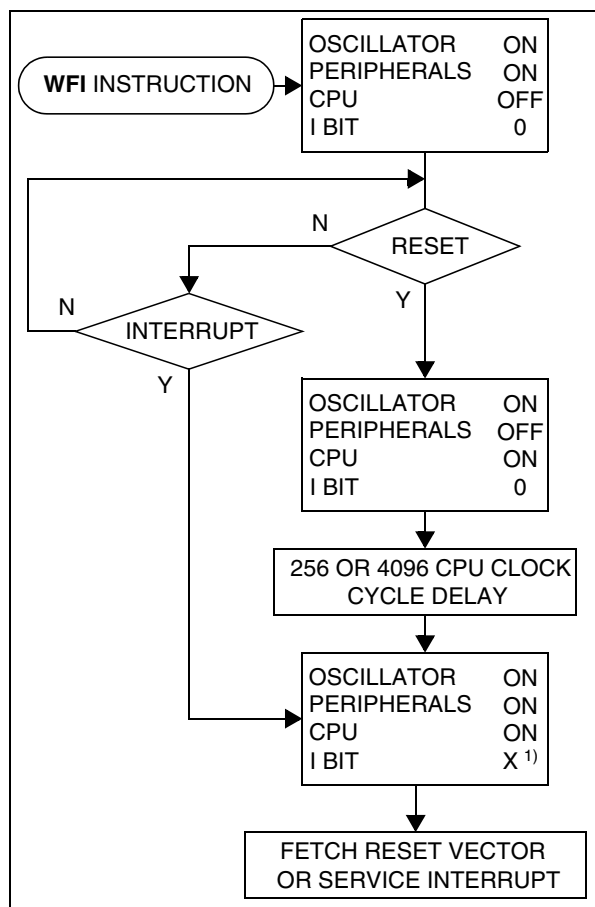
This power saving mode is selected by calling the 'WFI' instruction.

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 Flow-chart

**Note:**

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.

POWER SAVING MODES (Cont'd)

9.4 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 9.5 on page 41 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 5, "Interrupt Mapping," on page 35) 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 15.1 on page 121 for more details).

Figure 23. HALT Timing Overview

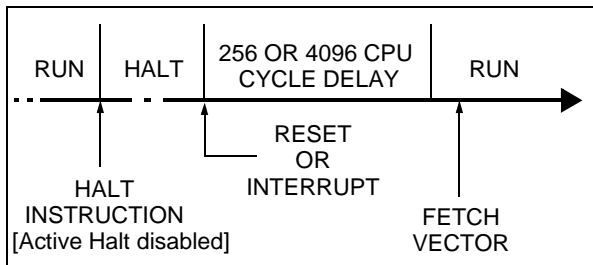
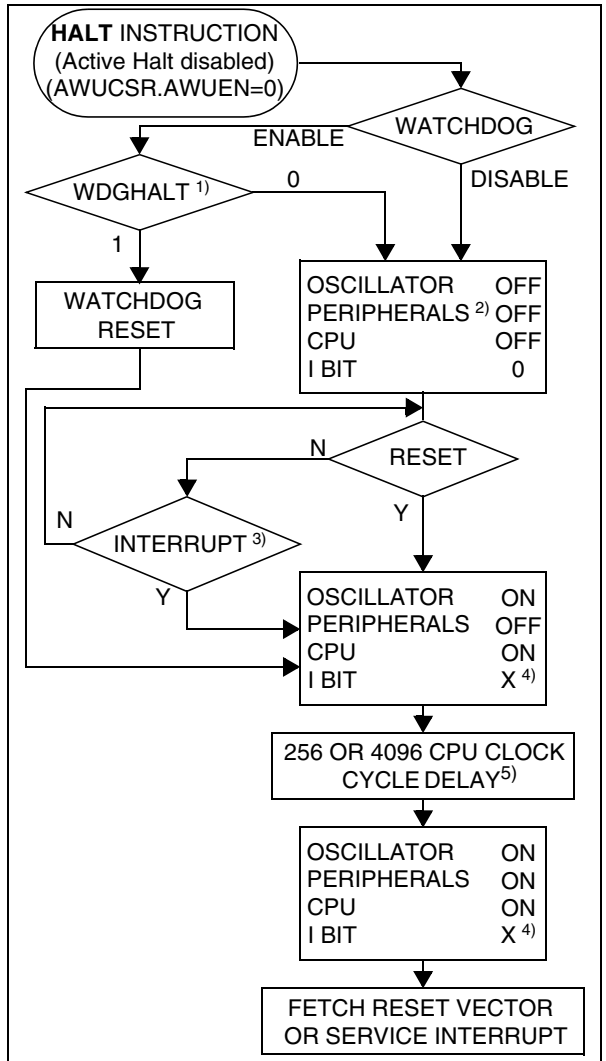


Figure 24. HALT Mode Flow-chart



Notes:

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 5 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<sub>STARTUP</sub> (see Figure 11).



## POWER SAVING MODES (Cont'd)

### 9.4.1 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 sensitivity 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).

### 9.5 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:

LTCSR1 TB11E bit	ATCSR OVFIE bit	ATCSR CK1 bit	ATCSR CK0 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 5, “Interrupt Mapping,” on page 35) 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.

POWER SAVING MODES (Cont'd)

Figure 25. ACTIVE-HALT Timing Overview

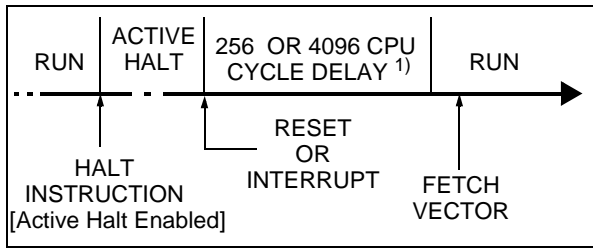
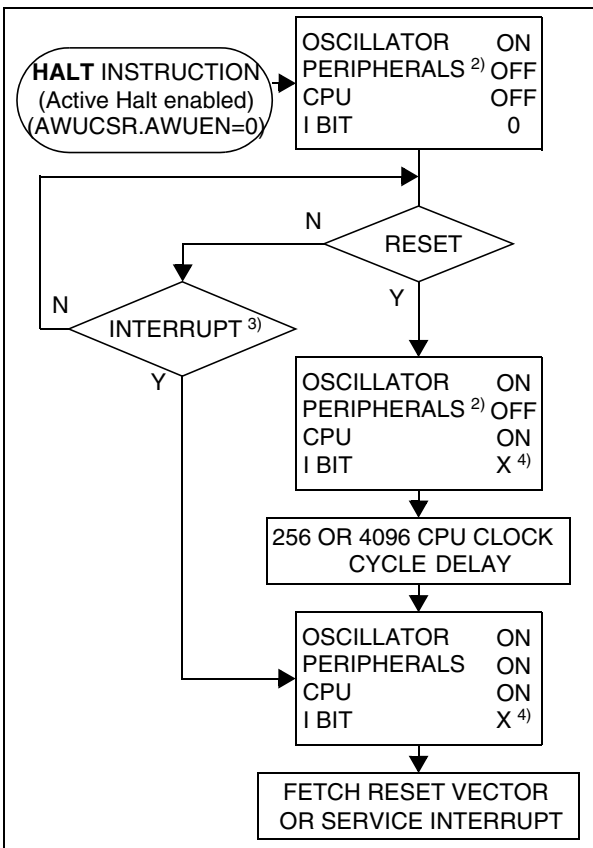


Figure 26. ACTIVE-HALT Mode Flow-chart



Notes:

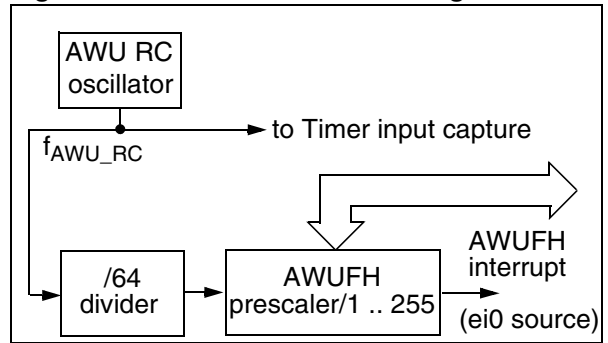
1. This delay occurs only if the MCU exits ACTIVE-HALT mode by means of a RESET.
2. Peripherals clocked with an external clock source can still be active.
3. Only the RTC1 interrupt and some specific interrupts can exit the MCU from ACTIVE-HALT mode. Refer to Table 5, "Interrupt Mapping," on page 35 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.

9.6 AUTO WAKE UP 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 realtime 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 time-base.

## POWER SAVING MODES (Cont'd)

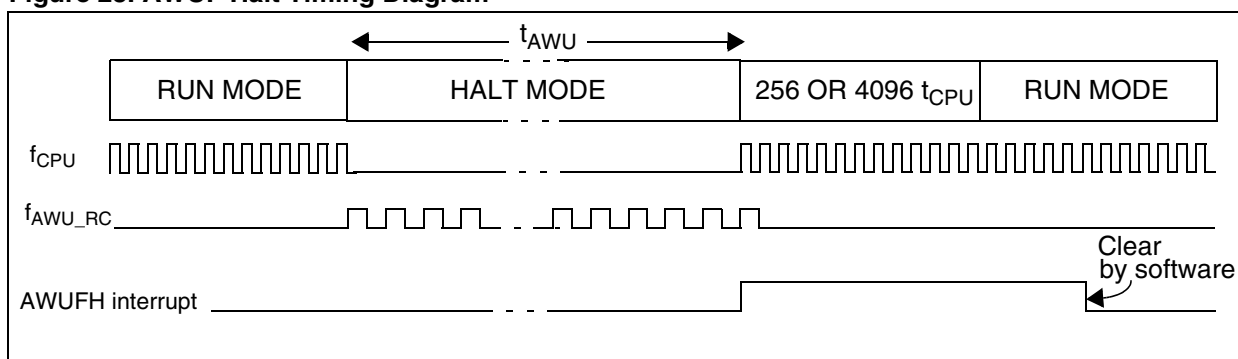
### Similarities with Halt mode

The following AWUFH mode behaviour 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 9.4 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.

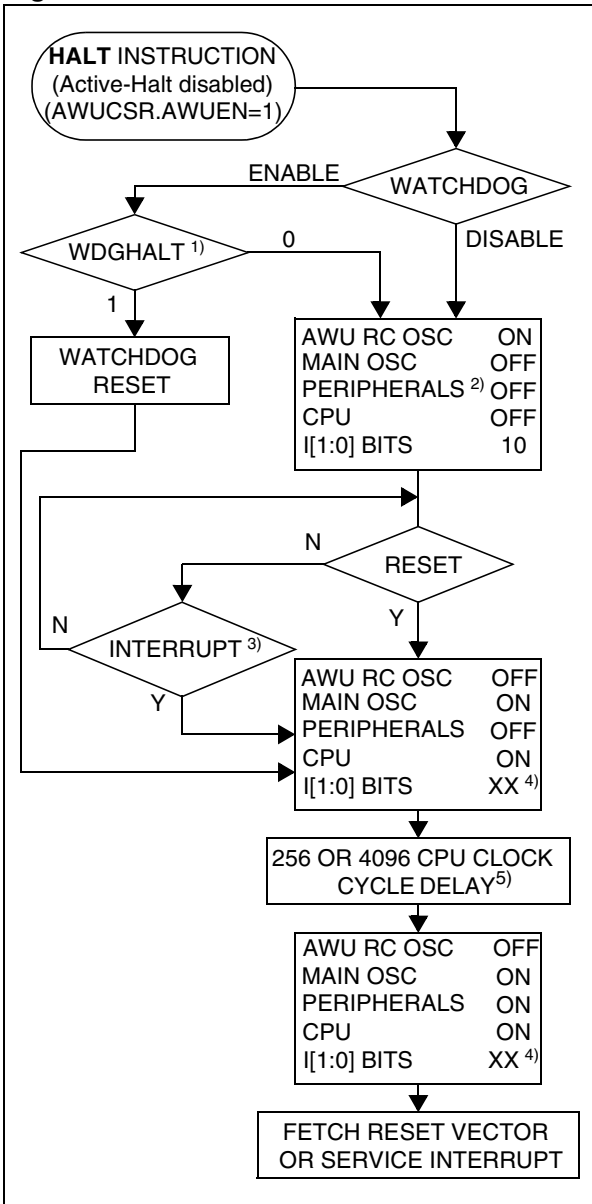
- In AWUFH mode, the main oscillator is turned off causing all internal processing to be stopped, including the operation of the on-chip peripherals. None of the peripherals are clocked except those which get their clock supply from another clock generator (such as an external or auxiliary oscillator 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**



POWER SAVING MODES (Cont'd)

Figure 29. AWUFH Mode Flow-chart



Notes:

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 5, "Interrupt Mapping," on page 35 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).

## POWER SAVING MODES (Cont'd)

## 9.6.0.1 Register Description

AWUFH CONTROL/STATUS REGISTER  
(AWUCSR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0	
0	0	0	0	0	0	AWU F	AWU M	AWU EN

Bits 7:3 = Reserved.

Bit 1= **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 inputcapture 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

Table 7. 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

7

0

AWU PR7	AWU PR6	AWU PR5	AWU PR4	AWU PR3	AWU PR2	AWU PR1	AWU PR0
------------	------------	------------	------------	------------	------------	------------	------------

Bits 7:0= **AWUPR[7:0]** *Auto Wake Up Prescaler*  
These 8 bits define the AWUPR Dividing factor (as explained below:

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 43](#)) 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.

## 10 I/O PORTS

### 10.1 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.

### 10.2 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.

#### 10.2.1 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.

#### Notes:

1. Writing to the DR modifies the latch value but does not change the state of the input pin.
2. Do not use read/modify/write instructions (BSET/BRES) to modify the DR register.

#### External Interrupt Function

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

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

External interrupts are hardware interrupts. Fetching the corresponding interrupt vector automatically clears the request latch. Modifying the sensitivity bits will clear any pending interrupts.

#### 10.2.2 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.

#### DR Value and Output Pin Status

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

#### 10.2.3 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 Device Pin Description table 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.

## I/O PORTS (Cont'd)

Figure 30. I/O Port General Block Diagram

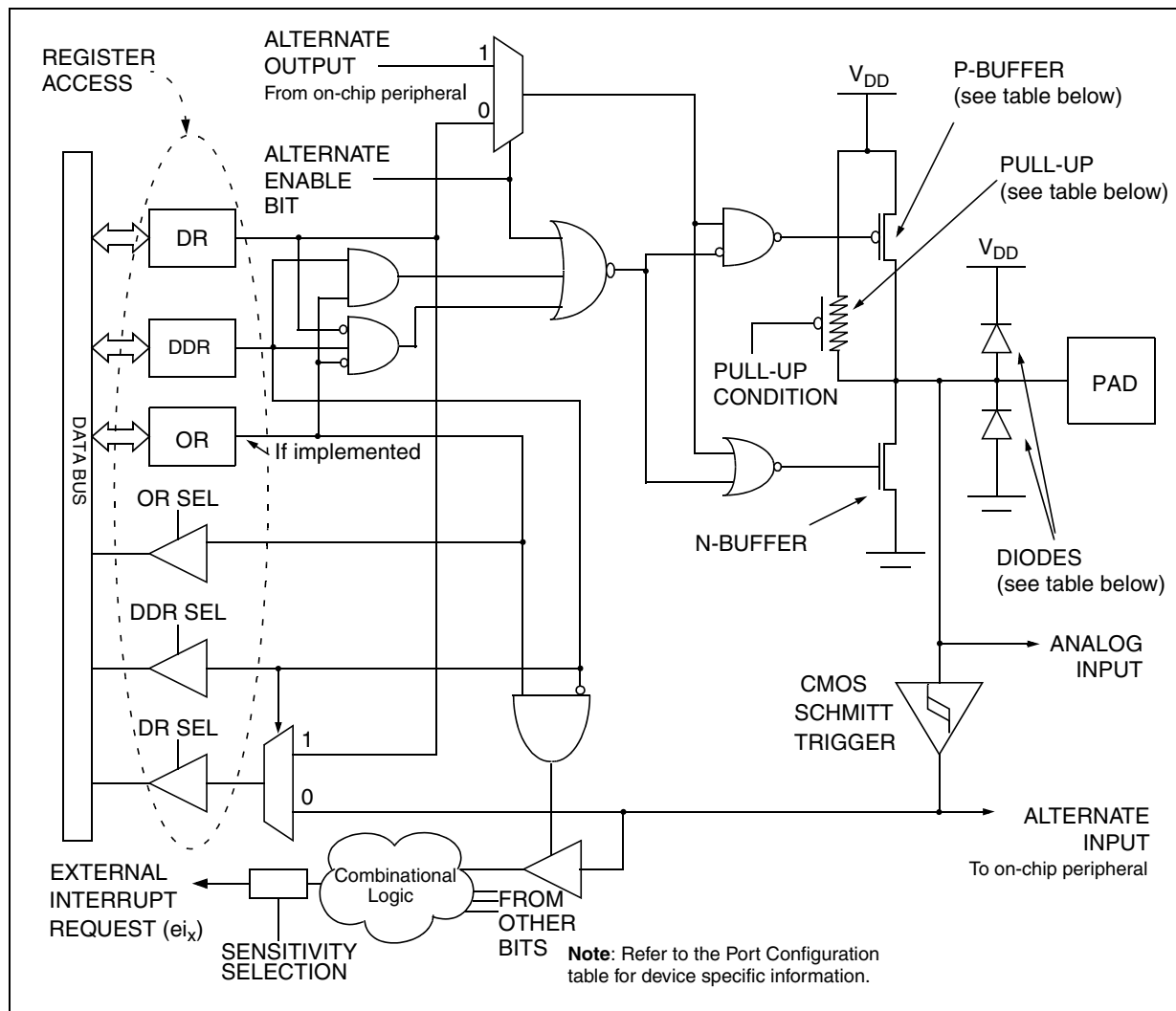


Table 8. I/O Port Mode Options

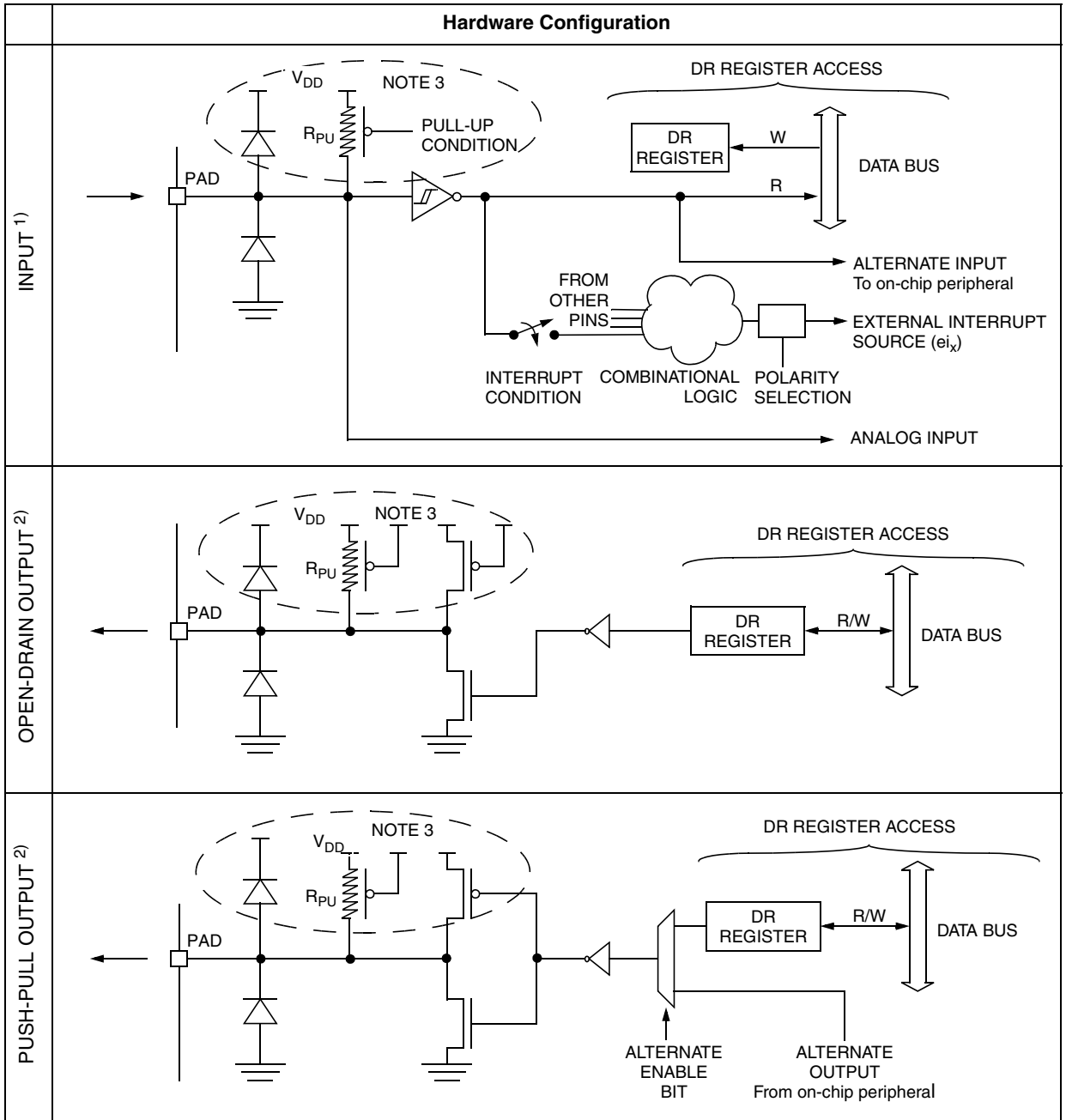
Configuration Mode		Pull-Up	P-Buffer	Diodes	
				to $V_{DD}$	to $V_{SS}$
Input	Floating with/without Interrupt	Off	Off	On	On
	Pull-up with/without Interrupt	On			
Output	Push-pull	Off	On	NI (see note)	On
	Open Drain (logic level)		Off		
	True Open Drain	NI	NI		

**Legend:** NI - not implemented  
 Off - implemented not activated  
 On - implemented and activated

**Note:** The diode to  $V_{DD}$  is not implemented in the true open drain pads. A local protection between the pad and  $V_{OL}$  is implemented to protect the device against positive stress.

I/O PORTS (Cont'd)

Table 9. I/O Configurations



Notes:

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 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.



## I/O PORTS (Cont'd)

### 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.

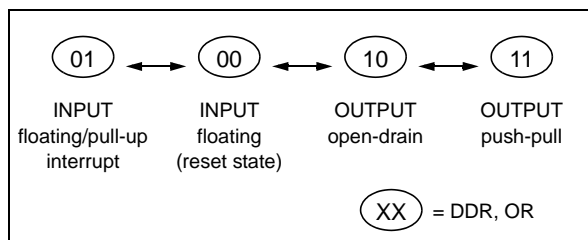
**WARNING:** The analog input voltage level must be within the limits stated in the absolute maximum ratings.

### 10.3 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



### 10.4 UNUSED I/O PINS

Unused I/O pins must be connected to fixed voltage levels. Refer to [Section 13.8](#).

### 10.5 LOW POWER MODES

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.

### 10.6 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).

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

I/O PORTS (Cont'd)

10.7 DEVICE-SPECIFIC I/O PORT CONFIGURATION

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

Standard Ports

PA7:0, PB6:0

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

Interrupt Ports

Ports where the external interrupt capability is selected using the 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 10. Port Configuration (Standard ports)

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

**Note:** On ports where the external interrupt capability is selected using the EISR register, the configuration will be as follows:

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 11. 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
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

## 11 ON-CHIP PERIPHERALS

### 11.1 WATCHDOG TIMER (WDG)

#### 11.1.1 Introduction

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

#### 11.1.2 Main Features

- Programmable free-running downcounter (64 increments of 16000 CPU cycles)
- Programmable reset
- Reset (if watchdog activated) when the T6 bit reaches zero

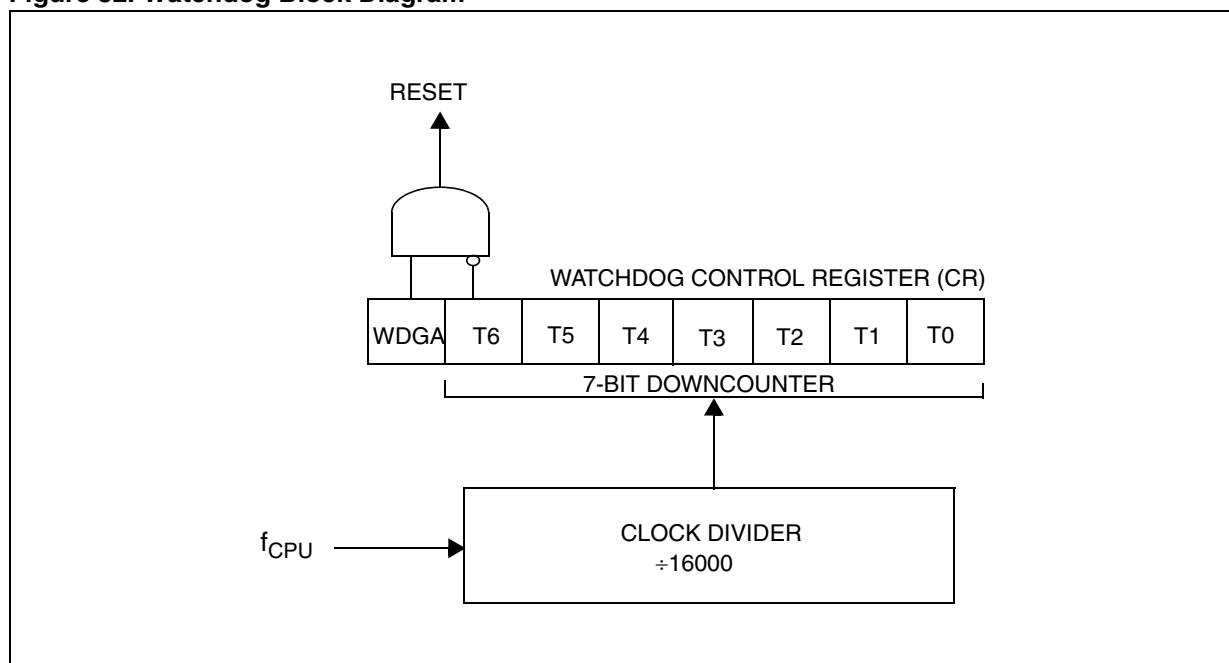
- Optional reset on HALT instruction (configurable by option byte)
- Hardware Watchdog selectable by option byte

#### 11.1.3 Functional Description

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

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

Figure 32. Watchdog Block Diagram



**WATCHDOG TIMER** (Cont'd)

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 12 .Watchdog Timing](#)):

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

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

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

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

**Table 12. Watchdog Timing**

$f_{CPU} = 8MHz$		
WDG Counter Code	min [ms]	max [ms]
C0h	1	2
FFh	127	128

Notes:

1. The timing variation shown in [Table 12](#) 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.

**11.1.4 Hardware Watchdog Option**

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

Refer to the Option Byte description in [section 15 on page 121](#).

**11.1.4.1 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. Same behavior in active-halt mode.

**WATCHDOG TIMER** (Cont'd)**11.1.5 Interrupts**

None.

**11.1.6 Register Description****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.

Bit 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).

**WATCHDOG TIMER (Cont'd)****Table 13. Watchdog Timer Register Map and Reset Values**

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

## 11.2 12-BIT AUTORELOAD TIMER 2 (AT2)

### 11.2.1 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:

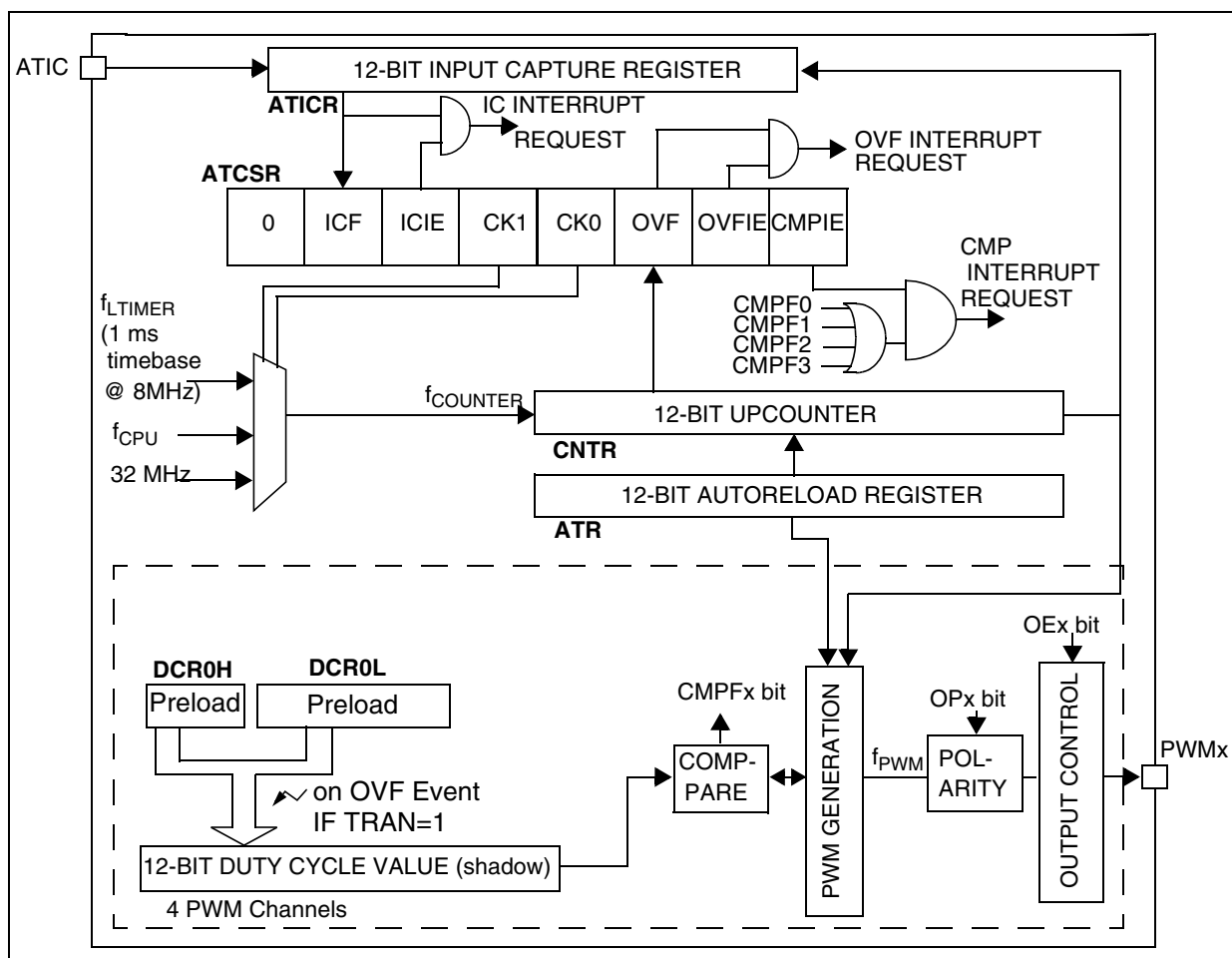
- Four PWM outputs
- ATIC pin for the Input Capture function
- BREAK pin for forcing a break condition on the PWM outputs

### 11.2.2 Main Features

- 12-bit upcounter with 12-bit autoreload register (ATR)

- Maskable overflow interrupt
- Generation of four independent PWMx signals
- Frequency 2KHz-4MHz (@ 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



12-BIT AUTORELOAD TIMER (Cont'd)

11.2.3 Functional Description

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,

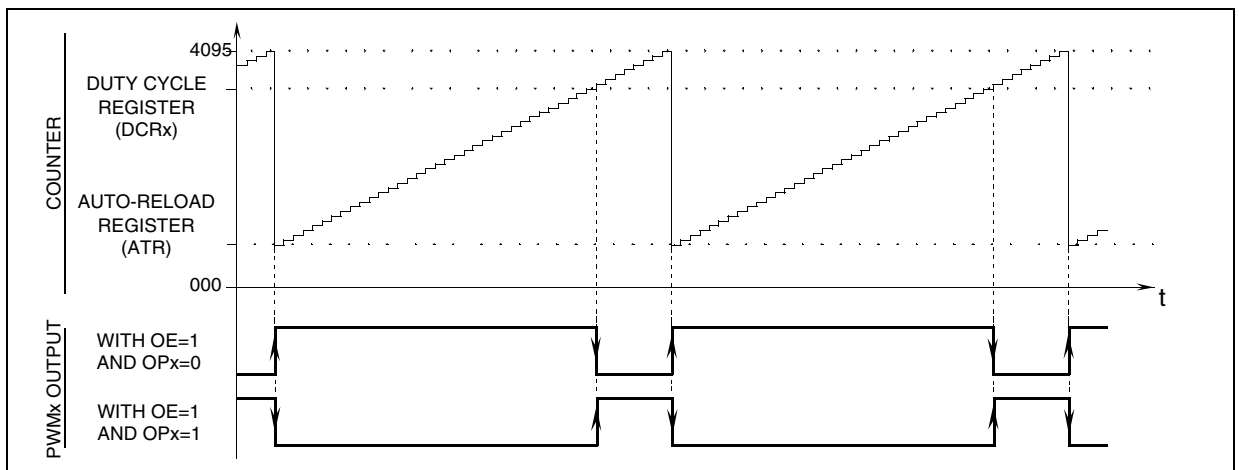
- If  $f_{COUNTER}$  is 32 MHz, the maximum value of  $f_{PWM}$  is 8 MHz (ATR register value = 4092), the minimum value is 8 KHz (ATR register value = 0)
- If  $f_{COUNTER}$  is 4 Mhz the maximum value of  $f_{PWM}$  is 2 MHz (ATR register value = 4094), the minimum value is 1 KHz (ATR register value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

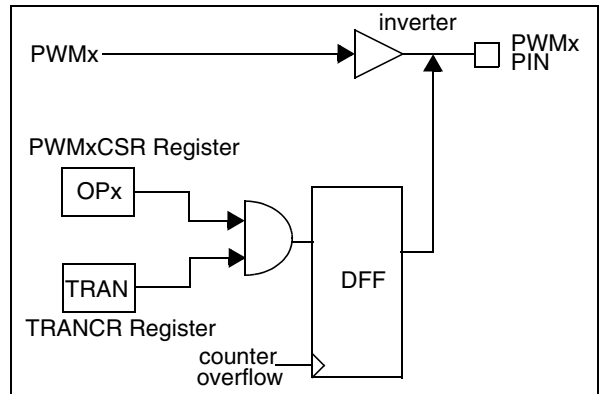
Figure 35. PWM Function



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 TRANCR register is set (reset value). See Figure 34.

Figure 34. PWM Inversion Diagram



The maximum available resolution for the PWMx duty cycle is:

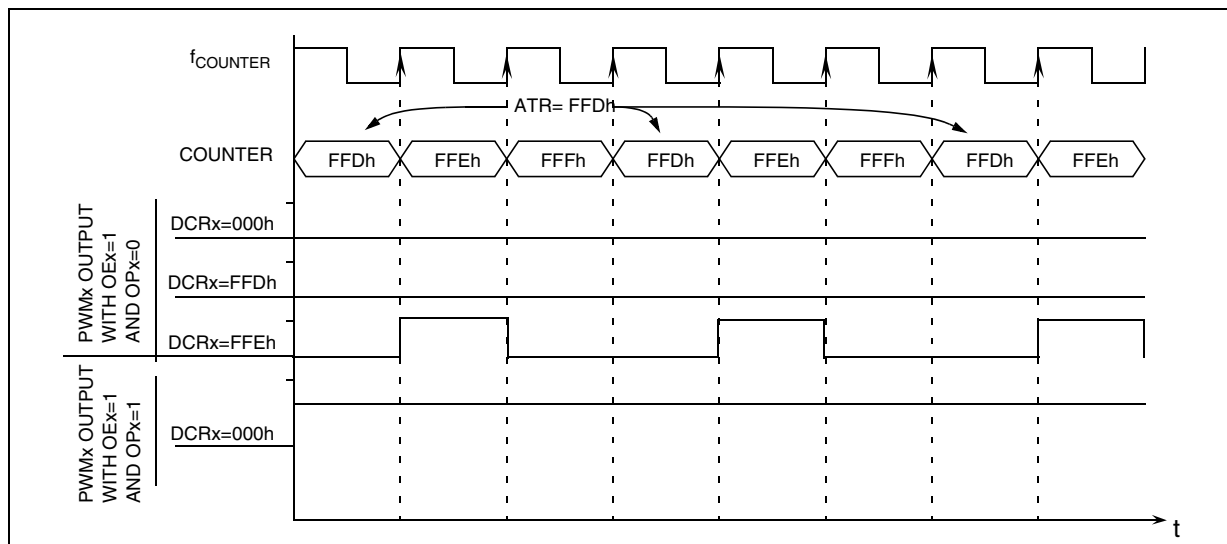
$$\text{Resolution} = 1 / (4096 - ATR)$$

**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.



## 12-BIT AUTORELOAD TIMER (Cont'd)

Figure 36. PWM Signal from 0% to 100% Duty Cycle

**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).

**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.

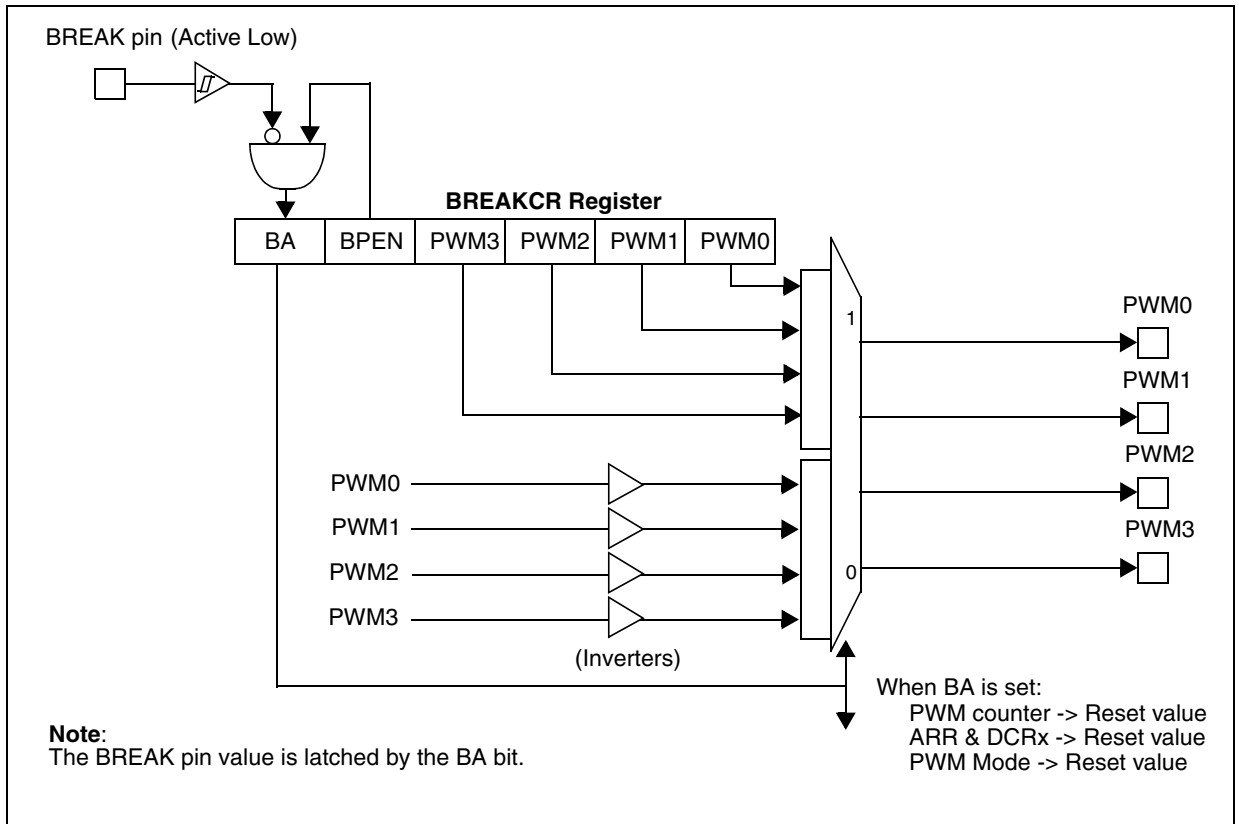
When the break function is activated (BA bit =1):

- The break pattern (PWM[3:0] bits in the BREAK-CR) 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

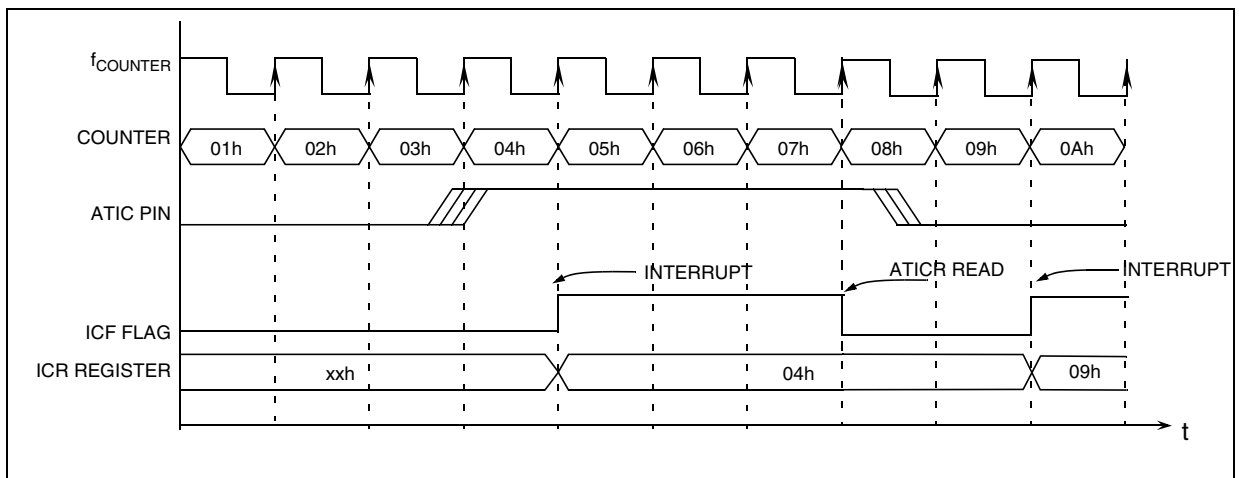


### 11.2.3.1 Input Capture

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



## 12-BIT AUTORELOAD TIMER (Cont'd)

### 11.2.4 Low Power Modes

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

### 11.2.5 Interrupts

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

**Note 1:** The CMP and IC events are connected to the same interrupt vector.

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

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

12-BIT AUTORELOAD TIMER (Cont'd)

11.2.6 Register Description

**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.

Counter Clock Selection	CK1	CK0
OFF	0	0
$f_{LTIMER}$ (1 ms timebase @ 8 MHz) <sup>1)</sup>	0	1
$f_{CPU}$	1	0
32 MHz <sup>2)</sup>	1	1

**Note 1:** PWM mode and Output Compare modes are not available at this frequency.

**Note 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.

**COUNTER REGISTER HIGH (CNTRH)**

Read only

Reset Value: 0000 0000 (000h)

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

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

**12-BIT AUTORELOAD TIMER (Cont'd)****AUTORELOAD REGISTER (ATRH)**

Read / Write

Reset Value: 0000 0000 (00h)

15							8
0	0	0	0	ATR11	ATR10	ATR9	ATR8

**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.

**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)

1: PWM mode enabled

**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.

**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

**12-BIT AUTORELOAD TIMER (Cont'd)**

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

Bit 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.

**PWMx DUTY CYCLE REGISTER HIGH (DCRxH)**

Read / Write

Reset Value: 0000 0000 (00h)

15								8
0	0	0	0	DCR11	DCR10	DCR9	DCR8	

**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.

**INPUT CAPTURE REGISTER HIGH (ATICRH)**

Read only

Reset Value: 0000 0000 (00h)

15								8
0	0	0	0	ICR11	ICR10	ICR9	ICR8	

**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.

**TRANSFER CONTROL REGISTER (TRANC R)**

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.

## 12-BIT AUTORELOAD TIMER (Cont'd)

Table 14. 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	CNTR11 0	CNTR10 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
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

## ST7LITE1

---

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
21	<b>TRANC</b> Reset Value	0	0	0	0	0	0	0	TRAN 1
22	<b>BREAKC</b> Reset Value	0	0	BA 0	BPEN 0	PWM3 0	PWM2 0	PWM1 0	PWM0 0



## 11.3 LITE TIMER 2 (LT2)

### 11.3.1 Introduction

The Lite Timer can be used for general-purpose timing functions. It is based on two free-running 8-bit upcounters, an 8-bit input capture register.

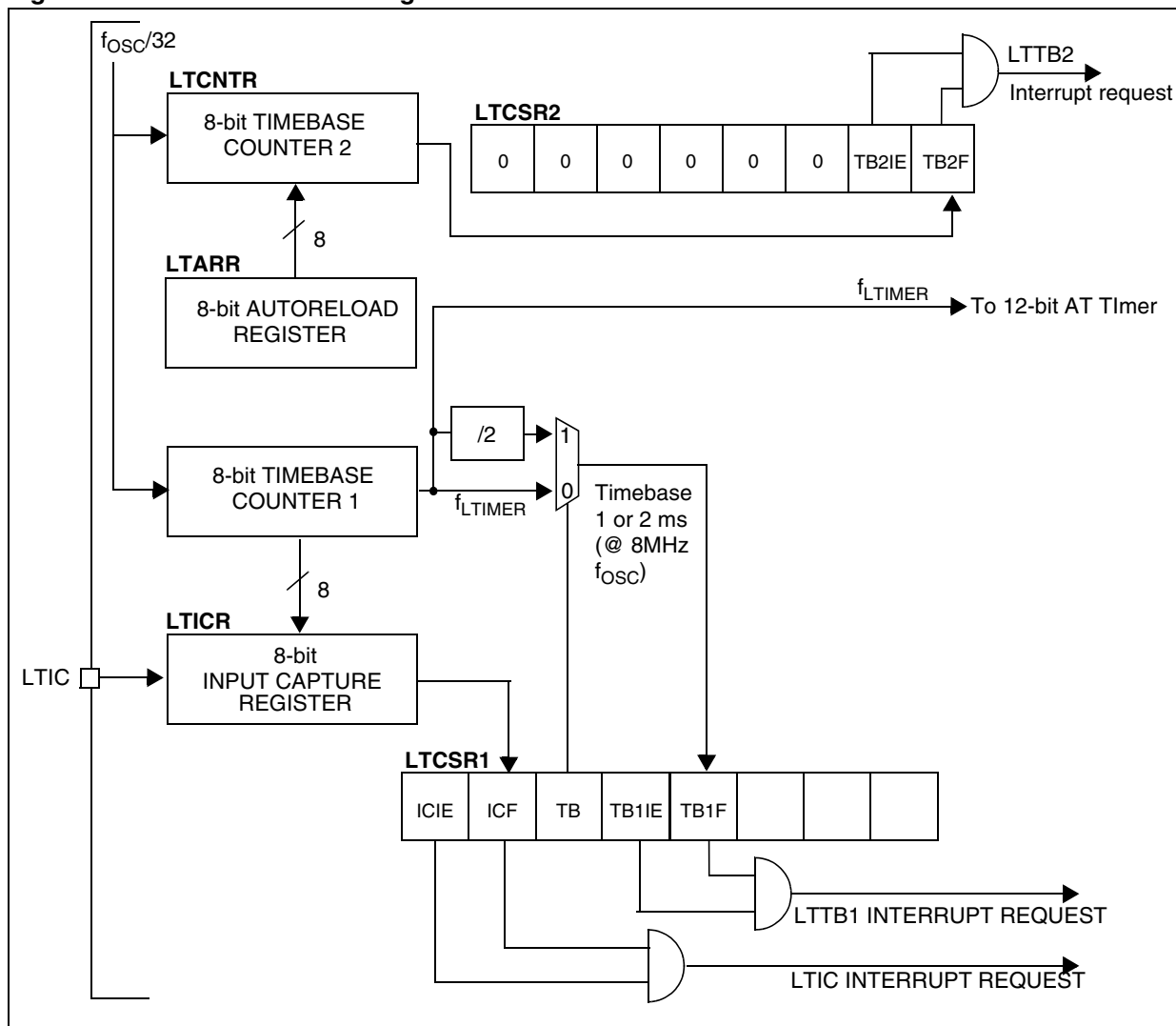
### 11.3.2 Main Features

#### ■ Realtime Clock

- One 8-bit upcounter 1 ms or 2 ms timebase period (@ 8 MHz  $f_{OSC}$ )

- One 8-bit upcounter with autoreload and programmable timebase period from 4 $\mu$ s to 1.024ms in 4 $\mu$ s increments (@ 8 MHz  $f_{OSC}$ )
- 2 Maskable timebase interrupts
  - Input Capture
    - 8-bit input capture register (LTICR)
    - Maskable interrupt with wakeup from Halt Mode capability

Figure 39. Lite Timer 2 Block Diagram



LITE TIMER (Cont'd)

11.3.3 Functional Description

11.3.3.1 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  $f_{OSC}/32$ . An overflow event occurs when the counter rolls over from F9h to 00h. If  $f_{OSC} = 8\text{ MHz}$ , 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.

11.3.3.2 Timebase Counter 2

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

LTARR reload value. Software can write a new value at anytime in the LTARR register, this value will be automatically loaded in the counter when the next overflow occurs.

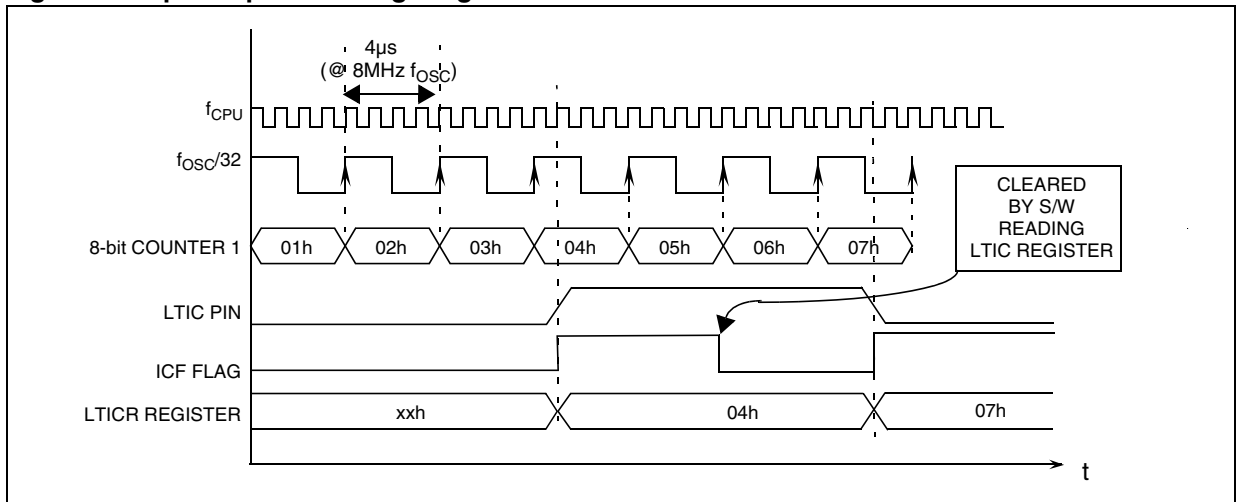
When Counter 2 overflows, the TB2F bit in the LTCSR2 register is set by hardware and an interrupt request is generated if the TB2IE bit is set. The TB2F bit is cleared by software reading the LTCSR2 register.

11.3.3.3 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.

Figure 40. Input Capture Timing Diagram.



**LITE TIMER** (Cont'd)

- The opcode for the HALT instruction is 0x8E. To avoid an unexpected HALT instruction due to a program counter failure, it is advised to clear all occurrences of the data value 0x8E from memory. For example, avoid defining a constant in ROM with the value 0x8E.
- As the HALT instruction clears the I bit in the CC register to allow interrupts, the user may choose to clear all pending interrupt bits before executing the HALT instruction. This avoids entering other peripheral interrupt routines after executing the external interrupt routine corresponding to the wake-up event (reset or external interrupt).

**11.3.4 Low Power Modes**

Mode	Description
SLOW	No effect on Lite timer (this peripheral is driven directly by $f_{OSC}/32$ )
WAIT	No effect on Lite timer
ACTIVE-HALT	No effect on Lite timer
HALT	Lite timer stops counting

**11.3.5 Interrupts**

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).

**11.3.6 Register Description****LITE TIMER CONTROL/STATUS REGISTER 2 (LTCSR2)**

Read / Write

Reset Value: 0x00 0000 (x0h)

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

**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.

**LITE TIMER (Cont'd)**

**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.

**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

**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.

## LITE TIMER (Cont'd)

Table 15. 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
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

### 11.4 SERIAL PERIPHERAL INTERFACE (SPI)

#### 11.4.1 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.

#### 11.4.2 Main Features

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

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

#### 11.4.3 General Description

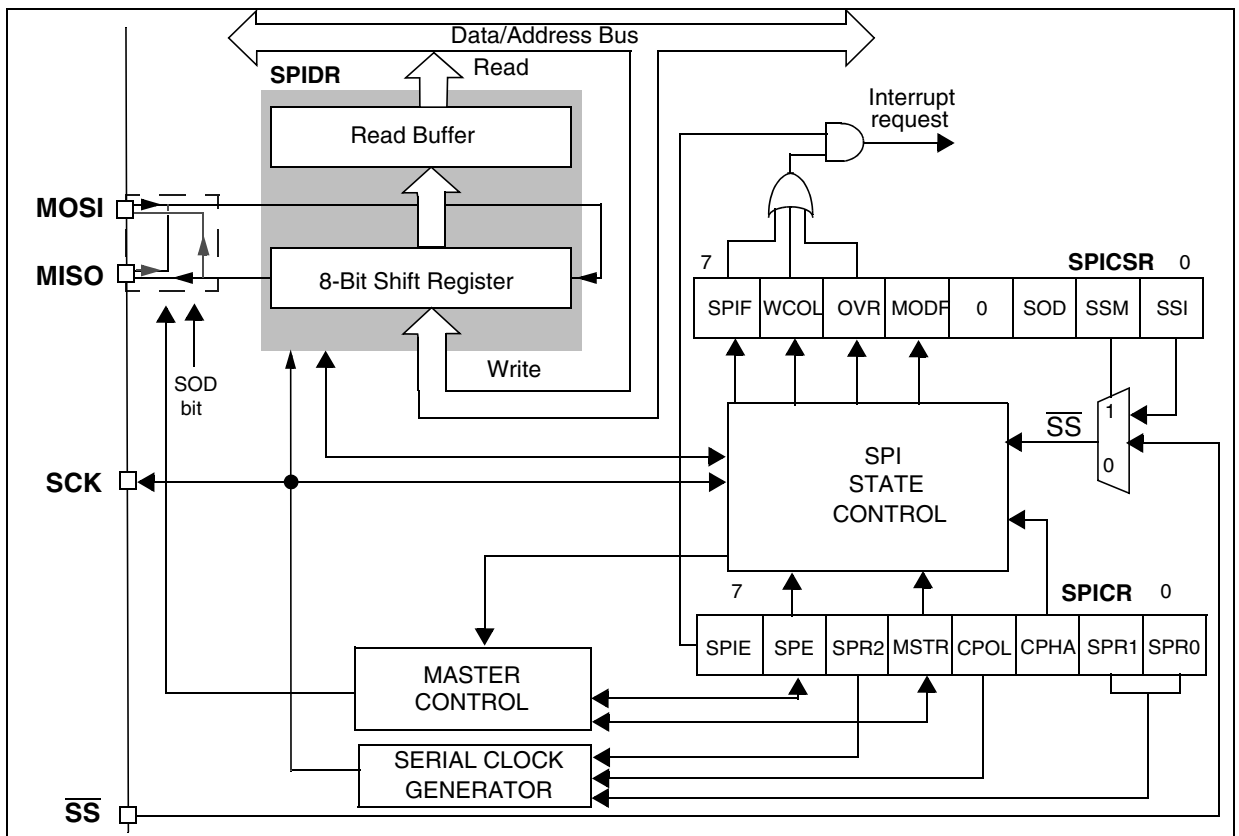
Figure 41 shows the serial peripheral interface (SPI) block diagram. There are 3 registers:

- 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
- $\overline{SS}$ : Slave select:  
This input signal acts as a 'chip select' to let the SPI master communicate with slaves individually and to avoid contention on the data lines. Slave  $\overline{SS}$  inputs can be driven by standard I/O ports on the master Device.

Figure 41. Serial Peripheral Interface Block Diagram



## SERIAL PERIPHERAL INTERFACE (Cont'd)

### 11.4.3.1 Functional Description

A basic example of interconnections between a single master and a single slave is illustrated in Figure 42.

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).

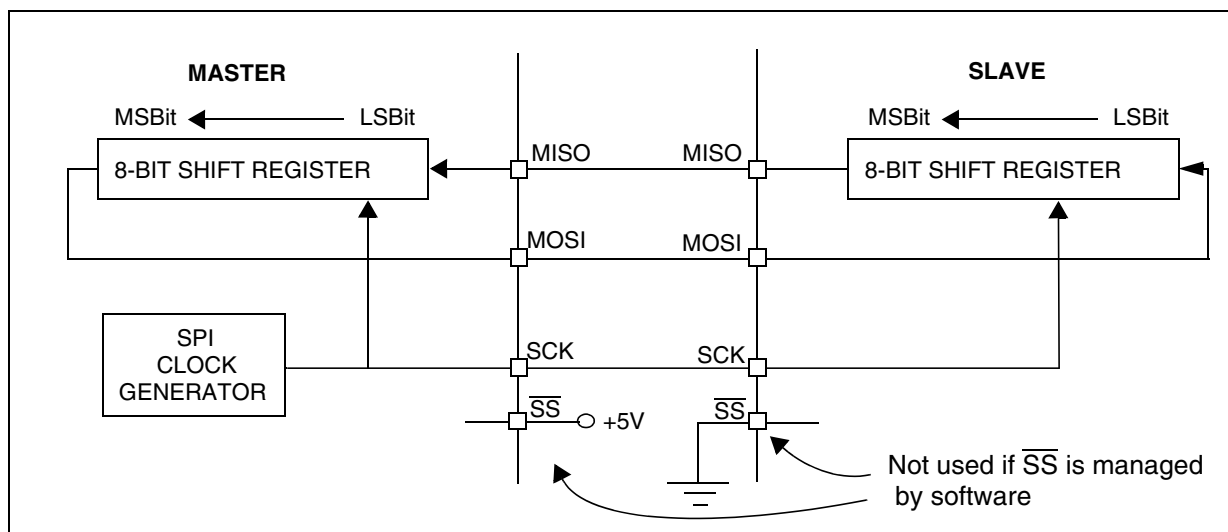
The communication is always initiated by the master. When the master device transmits data to a slave device via MOSI pin, the slave device re-

sponds 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 45) but master and slave must be programmed with the same timing mode.

**Figure 42. Single Master/ Single Slave Application**



**SERIAL PERIPHERAL INTERFACE (Cont'd)**

**11.4.3.2 Slave Select Management**

As an alternative to using the  $\overline{SS}$  pin to control the Slave Select signal, the application can choose to manage the Slave Select signal by software. This is configured by the SSM bit in the SPICSR register (see Figure 44)

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 43):

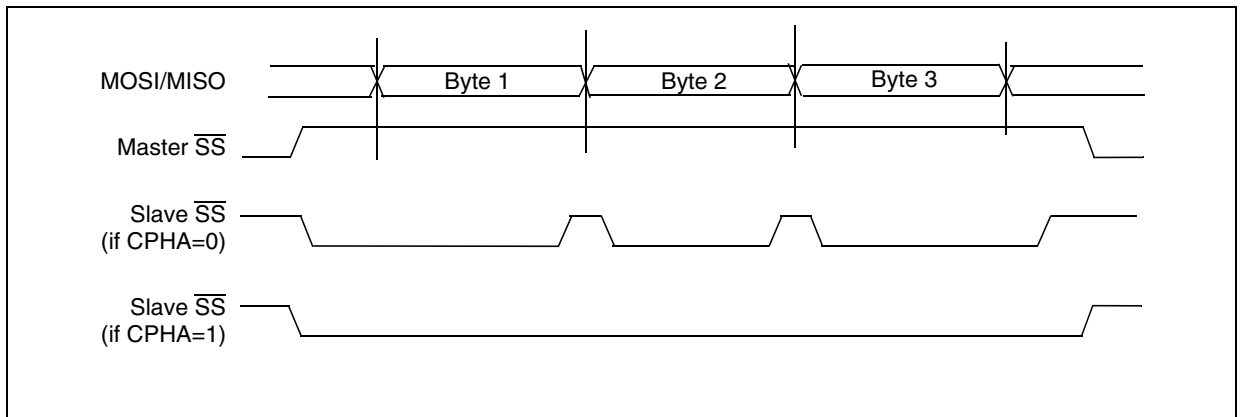
If CPHA=1 (data latched on 2nd clock edge):

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

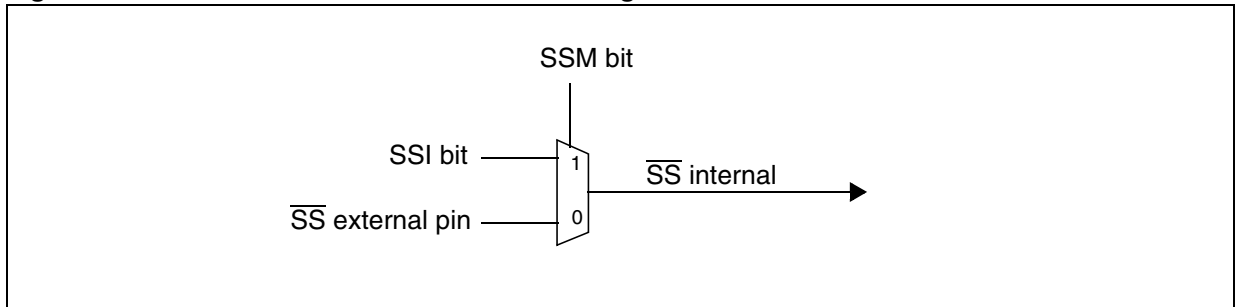
If CPHA=0 (data latched on 1st clock edge):

- $\overline{SS}$  internal must be held low during byte transmission and pulled high between each byte to allow the slave to write to the shift register. If  $\overline{SS}$  is not pulled high, a Write Collision error will occur when the slave writes to the shift register (see Section 11.4.5.3).

**Figure 43. Generic  $\overline{SS}$  Timing Diagram**



**Figure 44. Hardware/Software Slave Select Management**





## SERIAL PERIPHERAL INTERFACE (Cont'd)

### 11.4.3.3 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).

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 45](#) 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 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 SS is high.

The transmit sequence begins when software writes a byte in the SPIDR register.

### 11.4.3.4 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.

### 11.4.3.5 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 45](#)).

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

- Manage the  $\overline{SS}$  pin as described in [Section 11.4.3.2](#) and [Figure 43](#). 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.

### 11.4.3.6 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.

**Notes:** While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.

The SPIF bit can be cleared during a second transmission; however, it must be cleared before the second SPIF bit in order to prevent an Overrun condition (see [Section 11.4.5.2](#)).

**SERIAL PERIPHERAL INTERFACE (Cont'd)**

**11.4.4 Clock Phase and Clock Polarity**

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits (See Figure 45).

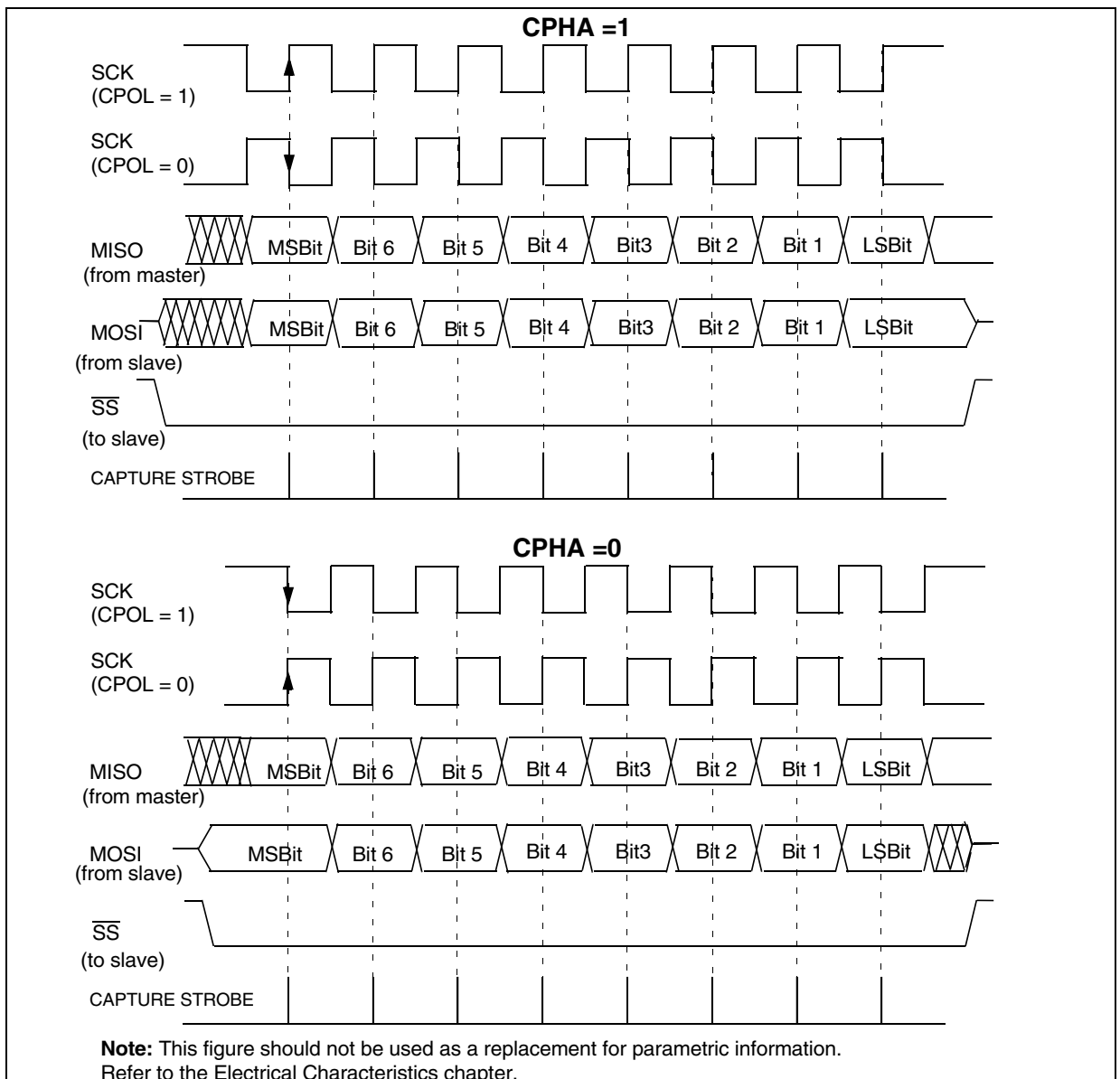
**Note:** The idle state of SCK must correspond to the polarity selected in the SPICSR register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).

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

Figure 45, 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 re-setting the SPE bit.

**Figure 45. Data Clock Timing Diagram**



## SERIAL PERIPHERAL INTERFACE (Cont'd)

### 11.4.5 Error Flags

#### 11.4.5.1 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.

**Notes:** 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 to a reset or return to an application default state.

#### 11.4.5.2 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.

#### 11.4.5.3 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 11.4.3.2 Slave Select Management](#).

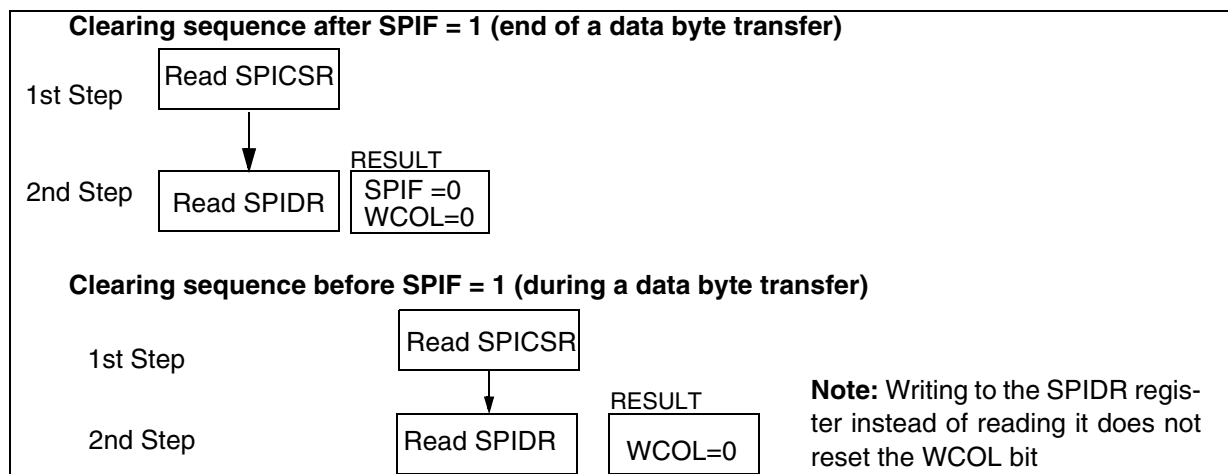
**Note:** a "read collision" will never occur since the received data byte is placed in a buffer in which access is always synchronous with the CPU operation.

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 46](#)).

**Figure 46. Clearing the WCOL bit (Write Collision Flag) Software Sequence**



**SERIAL PERIPHERAL INTERFACE (Cont'd)**

**11.4.5.4 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 47).

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.

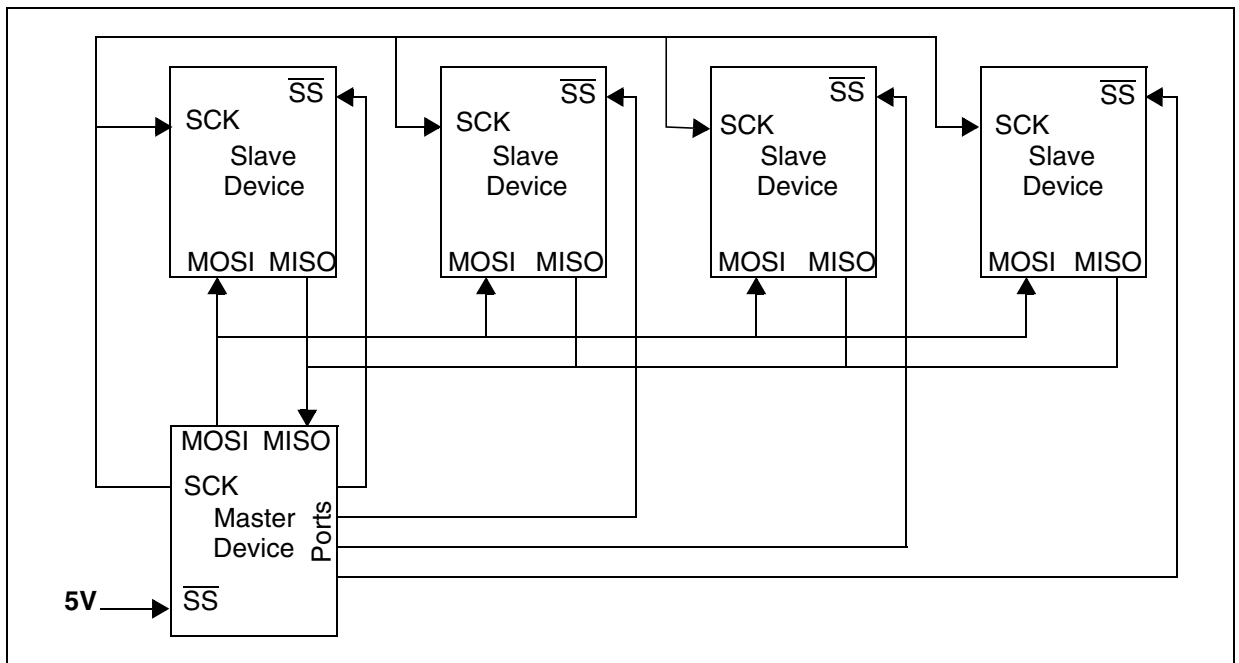
Other transmission security methods can use ports for handshake lines or data bytes with command fields.

**Multi-Master System**

A multi-master 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 multi-master system is principally handled by the MSTR bit in the SPICR register and the MODF bit in the SPICSR register.

**Figure 47. Single Master / Multiple Slave Configuration**



## SERIAL PERIPHERAL INTERFACE (Cont'd)

### 11.4.6 Low Power Modes

Mode	Description
WAIT	No effect on SPI. SPI interrupt events cause the Device to exit from WAIT mode.
HALT	SPI registers are frozen. In HALT mode, the SPI is inactive. SPI operation resumes when the Device is woken up by an interrupt with “exit from HALT mode” capability. The data received is subsequently read from the SPIDR register when the software is running (interrupt vector fetching). If several data are received before the 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.

#### 11.4.6.1 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 11.4.3.2](#)), make sure the master drives a low level on the  $\overline{SS}$  pin when the slave enters Halt mode.

### 11.4.7 Interrupts

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).

**SERIAL PERIPHERAL INTERFACE (Cont'd)**

**11.4.8 Register Description**

**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 Over-run error occurs (SPIF=1, MODF=1 or OVR=1 in the SPICSR register)

Bit 6 = **SPE** *Serial Peripheral Output Enable*.

This bit is set and cleared by software. It is also cleared by hardware when, in master mode,  $\overline{SS}=0$  (see [Section 11.4.5.1 Master Mode Fault \(MODF\)](#)). The SPE bit is cleared by reset, so the SPI peripheral is not initially connected to the external pins.

0: I/O pins free for general purpose I/O

1: SPI I/O pin alternate functions enabled

Bit 5 = **SPR2** *Divider Enable*.

This bit is set and cleared by software and is cleared by reset. It is used with the SPR[1:0] bits to set the baud rate. Refer to [Table 16 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 [Section 11.4.5.1 Master Mode Fault \(MODF\)](#)).

0: Slave mode

1: Master mode. The function of the SCK pin changes from an input to an output and the functions of the MISO and MOSI pins are reversed.

Bit 3 = **CPOL** *Clock Polarity*.

This bit is set and cleared by software. This bit determines the idle state of the serial Clock. The CPOL bit affects both the master and slave modes.

0: SCK pin has a low level idle state

1: SCK pin has a high level idle state

**Note:** If CPOL is changed at the communication byte boundaries, the SPI must be disabled by re-setting 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 16. 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

**SERIAL PERIPHERAL INTERFACE (Cont'd)****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 46](#)).

0: No write collision occurred

1: A write collision has been detected

Bit 5 = **OVR** *SPI Overrun error (Read only)*.

This bit is set by hardware when the byte currently being received in the shift register is ready to be transferred into the SPIDR register while SPIF = 1 (See [Section 11.4.5.2](#)). 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 [Section 11.4.5.1 Master Mode Fault \(MODF\)](#)). An SPI interrupt can be generated if SPIE=1 in the SPICR register. This bit is cleared by a software sequence (An access to the SPICSR register while MODF=1 followed by a write to the SPICR register).

0: No master mode fault detected

1: A fault in master mode has been detected

Bit 3 = Reserved, must be kept cleared.

Bit 2 = **SOD** *SPI Output Disable*.

This bit is set and cleared by software. When set, it disables the alternate function of the SPI output (MOSI in master mode / MISO in slave mode)

0: SPI output enabled (if SPE=1)

1: SPI output disabled

Bit 1 = **SSM**  $\overline{SS}$  *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 11.4.3.2 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

**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.

**Notes:** During the last clock cycle the SPIF bit is set, a copy of the received data byte in the shift register is moved to a buffer. When the user reads the serial peripheral data I/O register, the buffer is actually being read.

While the SPIF bit is set, all writes to the SPIDR register are inhibited until the SPICSR register is read.

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

A read to the SPIDR register returns the value located in the buffer and not the content of the shift register (see [Figure 41](#)).

Table 17. SPI Register Map and Reset Values

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



## 11.5 10-BIT A/D CONVERTER (ADC)

### 11.5.1 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.

### 11.5.2 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 48](#).

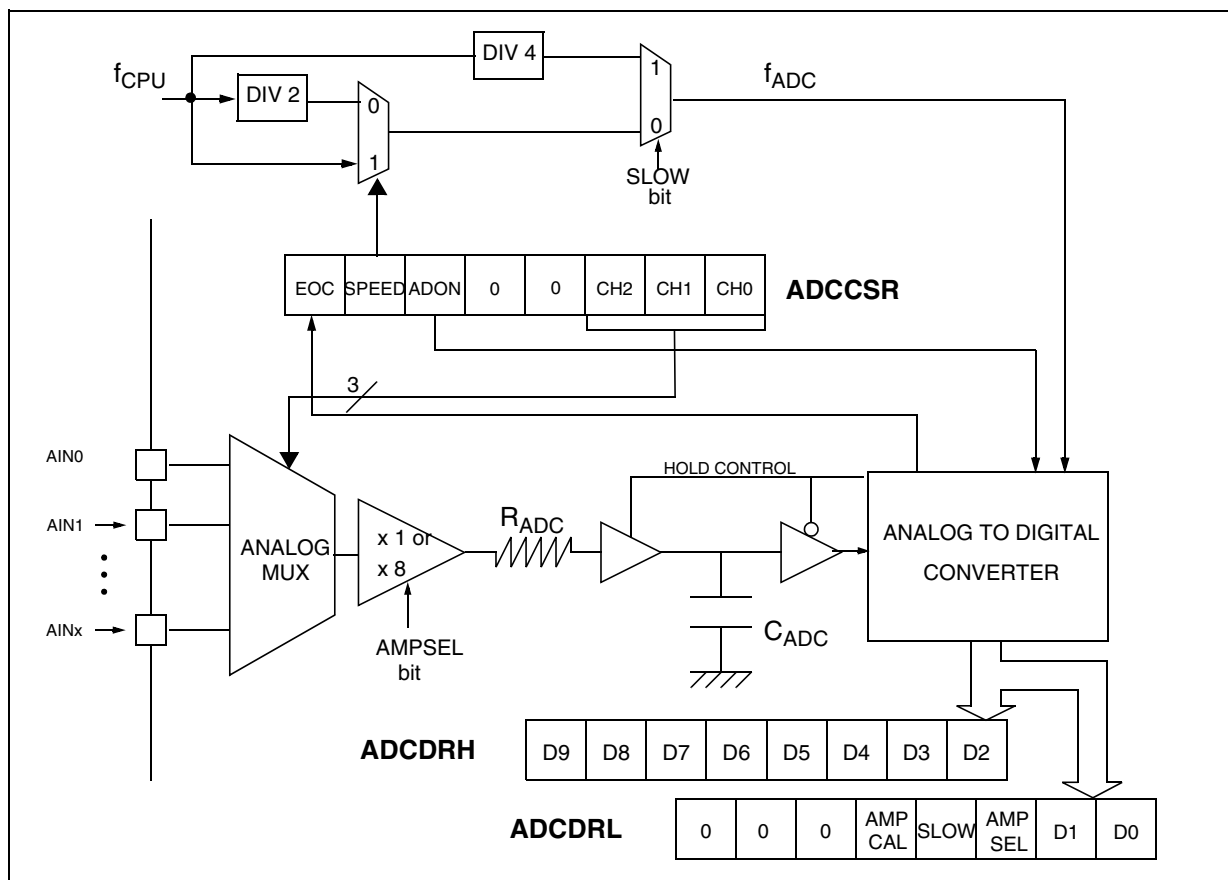
### 11.5.3 Functional Description

#### 11.5.3.1 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 48. ADC Block Diagram**



**10-BIT A/D CONVERTER (ADC) (Cont'd)**

**11.5.3.2 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 start-up time is required when the amplifier is selected by the AMPSEL bit.

**11.5.3.3 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 ADCDRH register and 03h in the ADCDRL register (without overflow indication).

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

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

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

**11.5.3.4 A/D Conversion**

The analog input ports must be configured as input, no pull-up, no interrupt. Refer to the «I/O ports» chapter. 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 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.

**11.5.4 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.

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.

**11.5.5 Interrupts**

None.

## 10-BIT A/D CONVERTER (ADC) (Cont'd)

### 11.5.6 Register Description

#### 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

This bit is set by hardware. It is cleared by software reading the ADCDRH 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

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

#### Bit 2:0 = **CH[2:0]** Channel Selection

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

Channel Pin*	CH2	CH1	CH0
AIN0	0	0	0
AIN1	0	0	1
AIN2	0	1	0
AIN3	0	1	1
AIN4	1	0	0
AIN5	1	0	1
AIN6	1	1	0

\*The number of channels is device dependent. Refer to the device pinout description.

#### DATA REGISTER HIGH (ADCDRH)

Read Only

Reset Value: xxxx xxxx (xxh)

7							0
D9	D8	D7	D6	D5	D4	D3	D2

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

#### AMP CONTROL/DATA REGISTER LOW (ADCURL)

Read/Write

Reset Value: 0000 00xx (0xh)

7							0
0	0	0	AMP CAL	SLOW	AMP- SEL	D1	D0

Bit 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 0v. 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.

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

Bit 2 = **AMPSEL** Amplifier Selection Bit

This bit is set and cleared by software.

0: Amplifier is not selected

1: Amplifier is selected

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

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

**Table 18. 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>ADCRL</b> Reset Value	0 0	0 0	0 0	AMPCAL 0	SLOW 0	AMPSEL 0	D1 x	D0 x

## 12 INSTRUCTION SET

### 12.1 ST7 ADDRESSING MODES

The ST7 Core features 17 different addressing modes which can be classified in 7 main 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 ST7 Instruction set is designed to minimize the number of bytes required per instruction: To do

so, most of the addressing modes may be subdivided in two sub-modes 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 19. ST7 Addressing Mode Overview**

Mode		Syntax	Destination/ Source	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 (with X register) + 1 (with Y register)	
Short	Direct	Indexed	ld A,(\$10,X)	00..1FE		+ 1	
Long	Direct	Indexed	ld A,(\$1000,X)	0000..FFFF		+ 2	
Short	Indirect		ld A,[\$10]	00..FF	00..FF	byte	+ 2
Long	Indirect		ld A,[\$10.w]	0000..FFFF	00..FF	word	+ 2
Short	Indirect	Indexed	ld A,([\$10],X)	00..1FE	00..FF	byte	+ 2
Long	Indirect	Indexed	ld A,([\$10.w],X)	0000..FFFF	00..FF	word	+ 2
Relative	Direct		jrne loop	PC-128/PC+127 <sup>1)</sup>		+ 1	
Relative	Indirect		jrne [\$10]	PC-128/PC+127 <sup>1)</sup>	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

**Note 1.** At the time the instruction is executed, the Program Counter (PC) points to the instruction following JRxx.

**ST7 ADDRESSING MODES (Cont'd)****12.1.1 Inherent**

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

Inherent 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
RIM	Reset Interrupt Mask
SCF	Set Carry Flag
RCF	Reset Carry Flag
RSP	Reset Stack Pointer
LD	Load
CLR	Clear
PUSH/POP	Push/Pop to/from the stack
INC/DEC	Increment/Decrement
TNZ	Test Negative or Zero
CPL, NEG	1 or 2 Complement
MUL	Byte Multiplication
SLL, SRL, SRA, RLC, RRC	Shift and Rotate Operations
SWAP	Swap Nibbles

**12.1.2 Immediate**

Immediate instructions have two bytes, the first byte contains the opcode, the second byte contains the operand value.

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

**12.1.3 Direct**

In Direct instructions, the operands are referenced by their memory address.

The direct addressing mode consists of two sub-modes:

**Direct (short)**

The address is a byte, thus requires 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.

**12.1.4 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 indirect addressing mode consists of three sub-modes:

**Indexed (No Offset)**

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

**Indexed (Short)**

The offset is a byte, thus requires 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.

**12.1.5 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 sub-modes:

**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.

## ST7 ADDRESSING MODES (Cont'd)

### 12.1.6 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 sub-modes:

#### 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 20. Instructions Supporting Direct, Indexed, Indirect and Indirect Indexed Addressing Modes**

Long and Short Instructions	Function
LD	Load
CP	Compare
AND, OR, XOR	Logical Operations
ADC, ADD, SUB, SBC	Arithmetic Addition/subtraction operations
BCP	Bit Compare

Short Instructions Only	Function
CLR	Clear
INC, DEC	Increment/Decrement
TNZ	Test Negative or Zero
CPL, NEG	1 or 2 Complement
BSET, BRES	Bit Operations
BTJT, BTJF	Bit Test and Jump Operations
SLL, SRL, SRA, RLC, RRC	Shift and Rotate Operations

SWAP	Swap Nibbles
CALL, JP	Call or Jump subroutine

### 12.1.7 Relative Mode (Direct, Indirect)

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

Available Relative Direct/Indirect Instructions	Function
JRxx	Conditional Jump
CALLR	Call Relative

The relative addressing mode consists of two sub-modes:

#### Relative (Direct)

The offset follows the opcode.

#### Relative (Indirect)

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

**12.2 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:

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							
Interruption management	TRAP	WFI	HALT	IRET				
Condition Code Flag modification	SIM	RIM	SCF	RCF				

**Using a pre-byte**

The instructions are described with one to four bytes.

In order to extend the number of available opcodes for an 8-bit CPU (256 opcodes), three different prebyte opcodes are defined. These prebytes modify the meaning of the instruction they precede.

The whole instruction becomes:

- PC-2 End of previous instruction
- PC-1 Prebyte
- PC Opcode
- PC+1 Additional word (0 to 2) according to the number of bytes required to compute the effective address

These prebytes enable instruction in Y as well as indirect addressing modes to be implemented. They precede the opcode of the instruction in X or the instruction using direct addressing mode. The prebytes are:

PDY 90 Replace an X based instruction using immediate, direct, indexed, or inherent addressing mode by a Y one.

PIX 92 Replace an instruction using direct, direct bit, or direct relative addressing mode to an instruction using the corresponding indirect addressing mode. It also changes an instruction using X indexed addressing mode to an instruction using indirect X indexed addressing mode.

PIY 91 Replace an instruction using X indirect indexed addressing mode by a Y one.

**12.2.1 Illegal Opcode Reset**

In order to provide enhanced robustness to the device against unexpected behaviour, 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.



## INSTRUCTION GROUPS (Cont'd)

Mnemo	Description	Function/Example	Dst	Src	H	I	N	Z	C
ADC	Add with Carry	$A = A + M + C$	A	M	H		N	Z	C
ADD	Addition	$A = A + M$	A	M	H		N	Z	C
AND	Logical And	$A = A . M$	A	M			N	Z	
BCP	Bit compare A, Memory	tst (A . M)	A	M			N	Z	
BRES	Bit Reset	bres Byte, #3	M						
BSET	Bit Set	bset Byte, #3	M						
BTJF	Jump if bit is false (0)	btjf Byte, #3, Jmp1	M						C
BTJT	Jump if bit is true (1)	btjt Byte, #3, Jmp1	M						C
CALL	Call subroutine								
CALLR	Call subroutine relative								
CLR	Clear		reg, M				0	1	
CP	Arithmetic Compare	tst(Reg - M)	reg	M			N	Z	C
CPL	One Complement	$A = FFH-A$	reg, M				N	Z	1
DEC	Decrement	dec Y	reg, M				N	Z	
HALT	Halt					0			
IRET	Interrupt routine return	Pop CC, A, X, PC			H	I	N	Z	C
INC	Increment	inc X	reg, M				N	Z	
JP	Absolute Jump	jp [TBL.w]							
JRA	Jump relative always								
JRT	Jump relative								
JRF	Never jump	jrf *							
JRIH	Jump if ext. interrupt = 1								
JRIL	Jump if ext. interrupt = 0								
JRH	Jump if H = 1	H = 1 ?							
JRNH	Jump if H = 0	H = 0 ?							
JRM	Jump if I = 1	I = 1 ?							
JRNM	Jump if I = 0	I = 0 ?							
JRMI	Jump if N = 1 (minus)	N = 1 ?							
JRPL	Jump if N = 0 (plus)	N = 0 ?							
JREQ	Jump if Z = 1 (equal)	Z = 1 ?							
JRNE	Jump if Z = 0 (not equal)	Z = 0 ?							
JRC	Jump if C = 1	C = 1 ?							
JRNC	Jump if C = 0	C = 0 ?							
JRULT	Jump if C = 1	Unsigned <							
JRUGE	Jump if C = 0	Jmp if unsigned >=							
JRUGT	Jump if (C + Z = 0)	Unsigned >							

## INSTRUCTION GROUPS (Cont'd)

Mnemo	Description	Function/Example	Dst	Src	H	I	N	Z	C
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 pop CC	reg CC	M M					
PUSH	Push onto the Stack	push Y	M	reg, CC	H	I	N	Z	C
RCF	Reset carry flag	C = 0							0
RET	Subroutine Return								
RIM	Enable Interrupts	I = 0				0			
RLC	Rotate left true C	C <= Dst <= C	reg, M				N	Z	C
RRC	Rotate right true C	C => Dst => C	reg, M				N	Z	C
RSP	Reset Stack Pointer	S = Max allowed							
SBC	Subtract with Carry	A = A - M - C	A	M			N	Z	C
SCF	Set carry flag	C = 1							1
SIM	Disable Interrupts	I = 1				1			
SLA	Shift left Arithmetic	C <= Dst <= 0	reg, M				N	Z	C
SLL	Shift left Logic	C <= Dst <= 0	reg, M				N	Z	C
SRL	Shift right Logic	0 => Dst => C	reg, M				0	Z	C
SRA	Shift right Arithmetic	Dst7 => Dst => C	reg, M				N	Z	C
SUB	Subtraction	A = A - M	A	M			N	Z	C
SWAP	SWAP nibbles	Dst[7..4] <=> Dst[3..0]	reg, M				N	Z	
TNZ	Test for Neg & Zero	tnz  b 1					N	Z	
TRAP	S/W trap	S/W interrupt				1			
WFI	Wait for Interrupt					0			
XOR	Exclusive OR	A = A XOR M	A	M			N	Z	

## 13 ELECTRICAL CHARACTERISTICS

### 13.1 PARAMETER CONDITIONS

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

#### 13.1.1 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 ( $\text{mean} \pm 3\Sigma$ ).

#### 13.1.2 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.

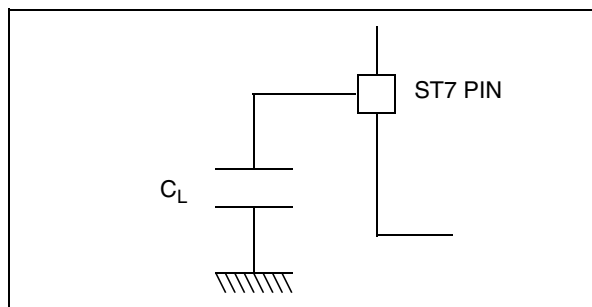
#### 13.1.3 Typical curves

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

#### 13.1.4 Loading capacitor

The loading conditions used for pin parameter measurement are shown in [Figure 49](#).

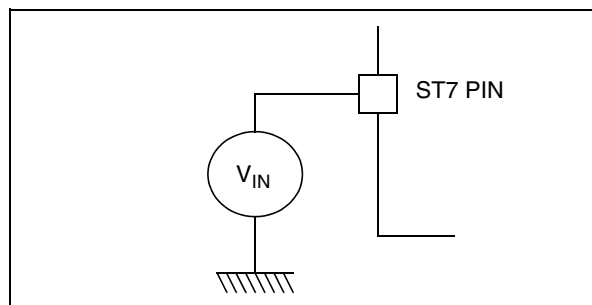
**Figure 49. Pin loading conditions**



#### 13.1.5 Pin input voltage

The input voltage measurement on a pin of the device is described in [Figure 50](#).

**Figure 50. Pin input voltage**



## 13.2 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.

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.

### 13.2.1 Voltage Characteristics

Symbol	Ratings	Maximum value	Unit
$V_{DD} - V_{SS}$	Supply voltage	7.0	V
$V_{IN}$	Input voltage on any pin <sup>1) &amp; 2)</sup>	$V_{SS}-0.3$ to $V_{DD}+0.3$	
$V_{ESD}(HBM)$	Electrostatic discharge voltage (Human Body Model)	see <a href="#">section 13.7.3 on page 104</a>	V

### 13.2.2 Current Characteristics

Symbol	Ratings	Maximum value	Unit
$I_{VDD}$	Total current into $V_{DD}$ power lines (source) <sup>3)</sup>	100	mA
$I_{VSS}$	Total current out of $V_{SS}$ ground lines (sink) <sup>3)</sup>	100	
$I_{IO}$	Output current sunk by any standard I/O and control pin	25	
	Output current sunk by any high sink I/O pin	50	
	Output current source by any I/Os and control pin	- 25	
$I_{INJ(PIN)}$ <sup>2) &amp; 4)</sup>	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>5)</sup>	+5	
	Injected current on any other pin <sup>6)</sup>	$\pm 5$	
$\Sigma I_{INJ(PIN)}$ <sup>2)</sup>	Total injected current (sum of all I/O and control pins) <sup>6)</sup>	$\pm 20$	

### 13.2.3 Thermal Characteristics

Symbol	Ratings	Value	Unit
$T_{STG}$	Storage temperature range	-65 to +150	°C
$T_J$	Maximum junction temperature (see Table 21, “THERMAL CHARACTERISTICS,” on page 119)		

#### Notes:

- 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: 10k $\Omega$  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.
- $I_{INJ(PIN)}$  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
- All power ( $V_{DD}$ ) and ground ( $V_{SS}$ ) lines must always be connected to the external supply.
- 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.
- No negative current injection allowed on PB0 and PB1 pins.
- 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 characterisation with  $\Sigma I_{INJ(PIN)}$  maximum current injection on four I/O port pins of the device.

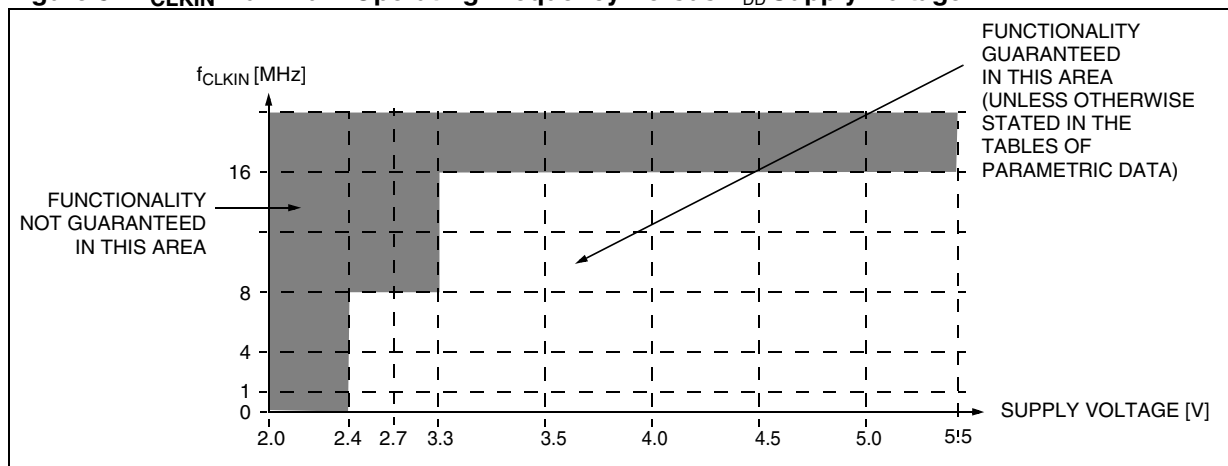
### 13.3 OPERATING CONDITIONS

#### 13.3.1 General Operating Conditions: Suffix 6 Devices

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

Symbol	Parameter	Conditions	Min	Max	Unit
$V_{DD}$	Supply voltage	$f_{OSC} = 8 \text{ MHz. max.},$	2.4	5.5	V
		$f_{OSC} = 16 \text{ MHz. max.}$	3.3	5.5	
$f_{CLKIN}$	External clock frequency on CLKIN pin	$3.3\text{V} \leq V_{DD} \leq 5.5\text{V}$	up to 16		MHz
		$2.4\text{V} \leq V_{DD} < 3.3\text{V}$	up to 8		

**Figure 51.  $f_{CLKIN}$  Maximum Operating Frequency Versus  $V_{DD}$  Supply Voltage**



### 13.3.2 Operating Conditions with Low Voltage Detector (LVD)

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

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)</sup>		20		20000	$\mu\text{s/V}$
$t_g(V_{DD})$	Filtered glitch delay on $V_{DD}$	Not detected by the LVD			150	ns
$I_{DD(LVD)}$	LVD/AVD current consumption			220		$\mu\text{A}$

**Note:**

- Not tested in production.
- 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.

### 13.3.3 Auxiliary Voltage Detector (AVD) Thresholds

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

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

**Note:**

- Not tested in production.

### 13.3.4 Internal RC Oscillator and PLL

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

Symbol	Parameter	Conditions	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

**OPERATING CONDITIONS** (Cont'd)

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

**13.3.4.1 Devices with "6" order code suffix (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
$f_{RC}$	Internal RC oscillator frequency	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 <sub>RC</sub>	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>1)</sup>		+2 <sup>1)</sup>	%
I <sub>DD(RC)</sub>	RC oscillator current consumption	$T_A=25^\circ\text{C}, V_{DD}=5\text{V}$		970 <sup>1)</sup>		μA
t <sub>su(RC)</sub>	RC oscillator setup time	$T_A=25^\circ\text{C}, V_{DD}=5\text{V}$			10 <sup>2)</sup>	μs
f <sub>PLL</sub>	x8 PLL input clock			1 <sup>1)</sup>		MHz
t <sub>LOCK</sub>	PLL Lock time <sup>5)</sup>			2		ms
t <sub>STAB</sub>	PLL Stabilization time <sup>5)</sup>			4		ms
ACC <sub>PLL</sub>	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>4)</sup>		%
		$f_{RC} = 1\text{MHz} @ T_A=-40$ to $+85^\circ\text{C}, V_{DD}=5\text{V}$		0.1 <sup>4)</sup>		%
t <sub>w(JIT)</sub>	PLL jitter period	$f_{RC} = 1\text{MHz}$		8 <sup>3)</sup>		kHz
JIT <sub>PLL</sub>	PLL jitter ( $\Delta f_{CPU}/f_{CPU}$ )			1 <sup>3)</sup>		%
I <sub>DD(PLL)</sub>	PLL current consumption	$T_A=25^\circ\text{C}$		600 <sup>1)</sup>		μA

**Notes:**

1. Data based on characterization results, not tested in production
2. RCCR0 is a factory-calibrated setting for 1000kHz with  $\pm 0.2$  accuracy @  $T_A = 25^\circ\text{C}, V_{DD} = 5\text{V}$ . See "INTERNAL RC OSCILLATOR ADJUSTMENT" on page 23
3. Guaranteed by design.
4. Averaged over a 4ms period. After the LOCKED bit is set, a period of t<sub>STAB</sub> is required to reach ACC<sub>PLL</sub> accuracy.
5. After the LOCKED bit is set ACC<sub>PLL</sub> is max. 10% until t<sub>STAB</sub> has elapsed. See [Figure 11 on page 24](#).

## OPERATING CONDITIONS (Cont'd)

13.3.4.2 Devices with "'6" order code suffix (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
$f_{RC}$	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 <sub>RC</sub>	Accuracy of Internal RC oscillator when calibrated with RCCR=RCCR1 <sup>1)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>1)</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>1)</sup>		MHz
$t_{LOCK}$	PLL Lock time <sup>5)</sup>			2		ms
$t_{STAB}$	PLL Stabilization time <sup>5)</sup>			4		ms
ACC <sub>PLL</sub>	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>4)</sup>		%
		$f_{RC} = 1\text{MHz}$ @ $T_A=40$ to $+85^\circ\text{C}$ , $V_{DD}= 3\text{V}$		0.1 <sup>4)</sup>		%
$t_{w(JIT)}$	PLL jitter period	$f_{RC} = 1\text{MHz}$		8 <sup>3)</sup>		kHz
JIT <sub>PLL</sub>	PLL jitter ( $\Delta f_{CPU}/f_{CPU}$ )			1 <sup>3)</sup>		%
$I_{DD(PLL)}$	PLL current consumption	$T_A=25^\circ\text{C}$		190 <sup>1)</sup>		$\mu\text{A}$

**Notes:**

1. Data based on characterization results, not tested in production
2. RCCR1 is a factory-calibrated setting for 700MHz with  $\pm 0.2$  accuracy @  $T_A = 25^\circ\text{C}$ ,  $V_{DD}=3\text{V}$ . See "INTERNAL RC OSCILLATOR ADJUSTMENT" on page 23.
3. Guaranteed by design.
4. Averaged over a 4ms period. After the LOCKED bit is set, a period of  $t_{STAB}$  is required to reach ACC<sub>PLL</sub> accuracy
5. After the LOCKED bit is set ACC<sub>PLL</sub> is max. 10% until  $t_{STAB}$  has elapsed. See [Figure 11 on page 24](#).



OPERATING CONDITIONS (Cont'd)

Figure 52. RC Osc Freq vs  $V_{DD}$  @  $T_A=25^\circ\text{C}$   
(Calibrated with RCCR1: 3V @ 25°C)

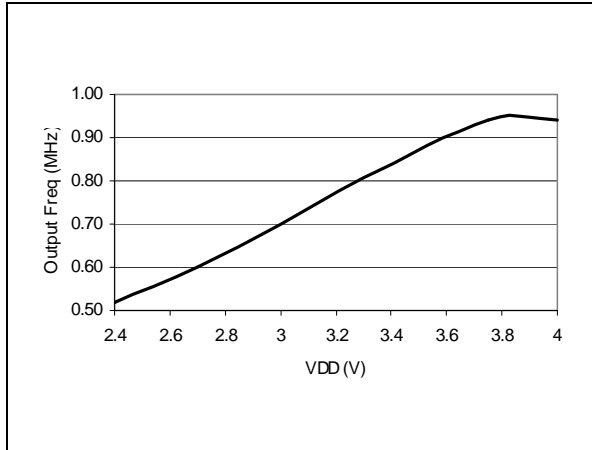


Figure 53. RC Osc Freq vs  $V_{DD}$   
(Calibrated with RCCR0: 5V @ 25°C)

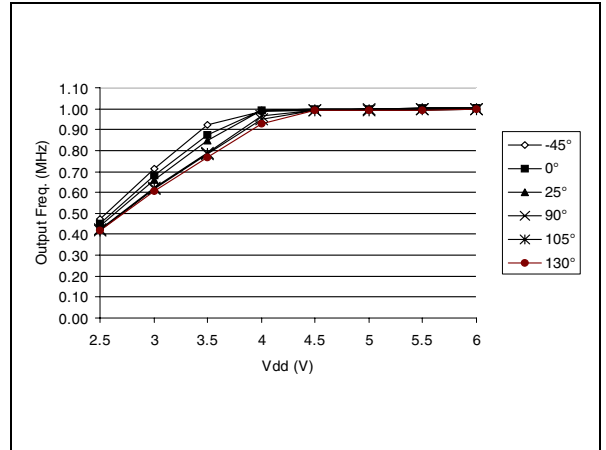


Figure 54. Typical RC oscillator Accuracy vs temperature @  $V_{DD}=5\text{V}$   
(Calibrated with RCCR0: 5V @ 25°C)

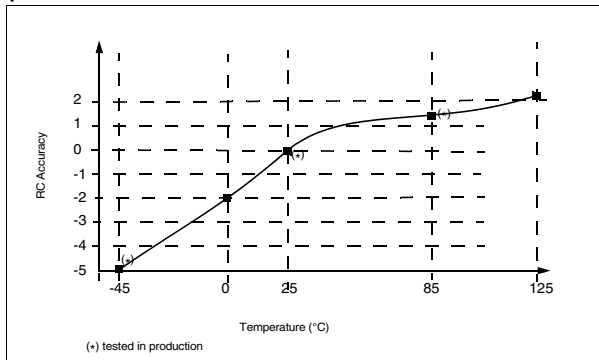
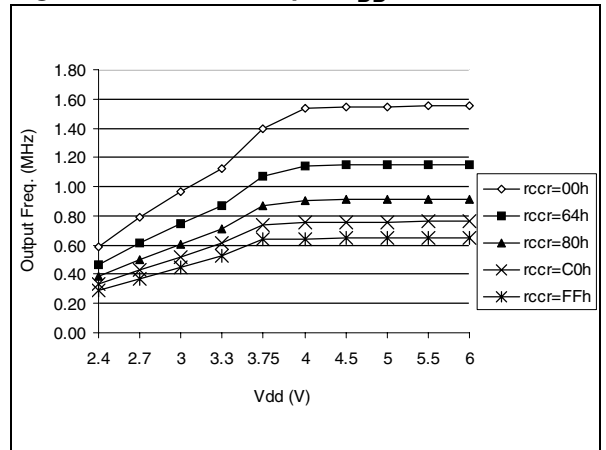


Figure 55. RC Osc Freq vs  $V_{DD}$  and RCCR Value



OPERATING CONDITIONS (Cont'd)

Figure 56. PLL  $\Delta f_{CPU}/f_{CPU}$  versus time

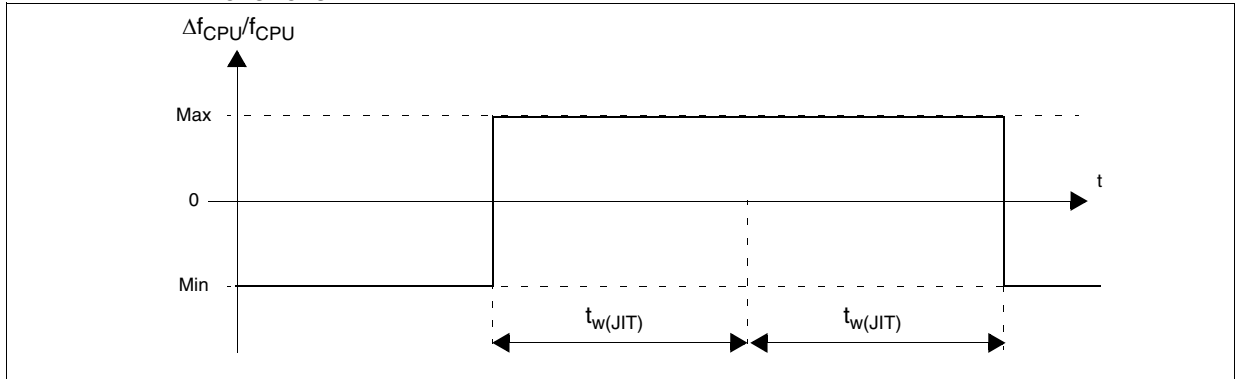
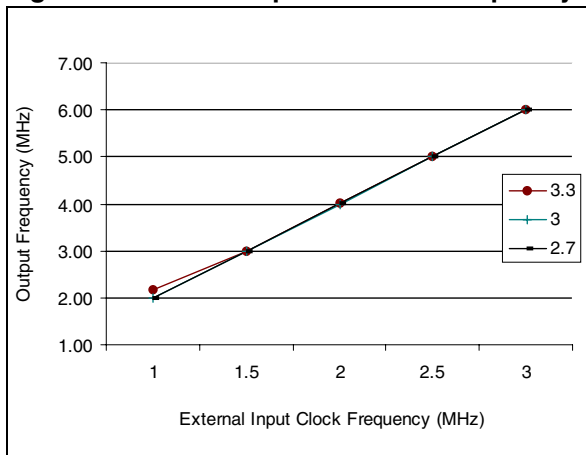
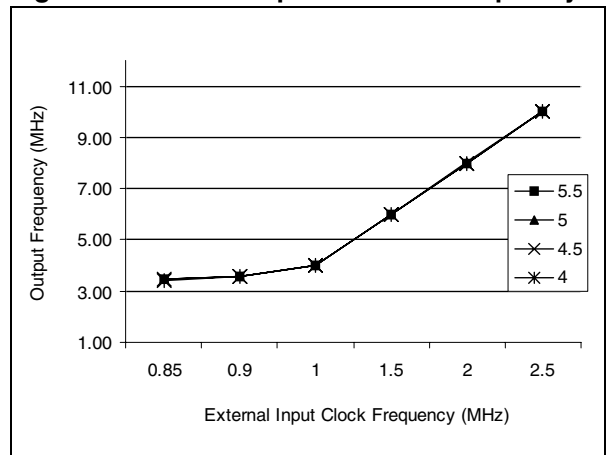


Figure 57. PLLx4 Output vs CLKIN frequency



Note:  $f_{OSC} = f_{CLKIN}/2 * PLL4$

Figure 58. PLLx8 Output vs CLKIN frequency



Note:  $f_{OSC} = f_{CLKIN}/2 * PLL8$

13.3.4.3 32MHz PLL

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

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

Note 1: 32 MHz is guaranteed within this voltage range.

### 13.4 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 de-

vice consumption, the two current values must be added (except for HALT mode for which the clock is stopped).

#### 13.4.1 Supply Current

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

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		
$T_A = +25^\circ\text{C}$		20	30			
Supply current in AWUFH mode <sup>6)7)</sup>	$T_A = +25^\circ\text{C}$					

#### Notes:

- 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  $f_{OSC}$  divided by 32. All I/O pins in input mode with a static value at  $V_{DD}$  or  $V_{SS}$  (no load), all peripherals in reset state; clock input (CLKIN) driven by external square wave, LVD disabled.
- SLOW-WAIT mode selected with  $f_{CPU}$  based on  $f_{OSC}$  divided by 32. All I/O pins in input mode with a static value at  $V_{DD}$  or  $V_{SS}$  (no load), all peripherals in reset state; clock input (CLKIN) driven by external square wave, LVD disabled.
- 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 59. Typical  $I_{DD}$  in RUN vs.  $f_{CPU}$

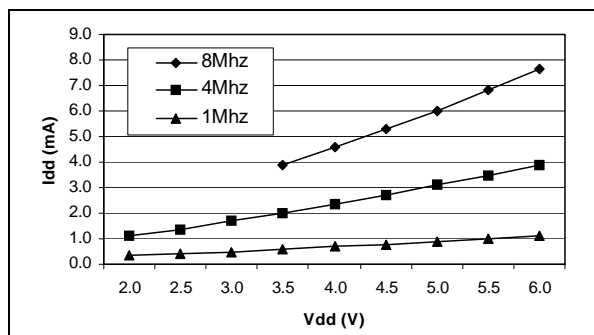
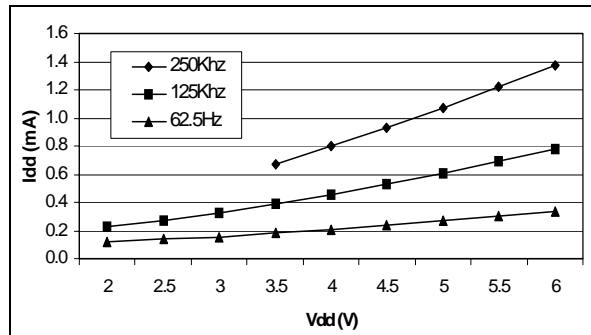


Figure 60. Typical  $I_{DD}$  in SLOW vs.  $f_{CPU}$



SUPPLY CURRENT CHARACTERISTICS (Cont'd)

Figure 61. Typical  $I_{DD}$  in WAIT vs.  $f_{CPU}$

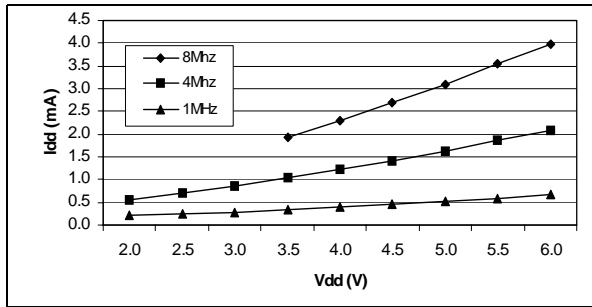


Figure 62. Typical  $I_{DD}$  in SLOW-WAIT vs.  $f_{CPU}$

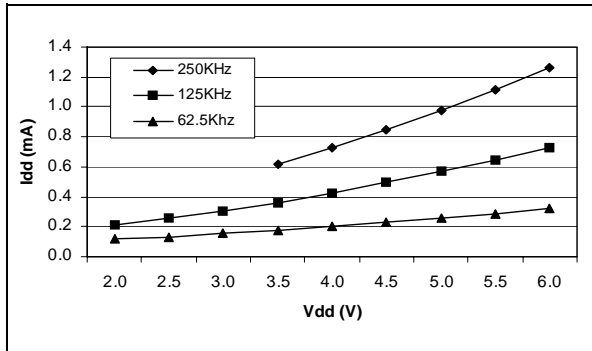


Figure 63. Typical  $I_{DD}$  in AWUFH mode at  $T_A=25^\circ\text{C}$

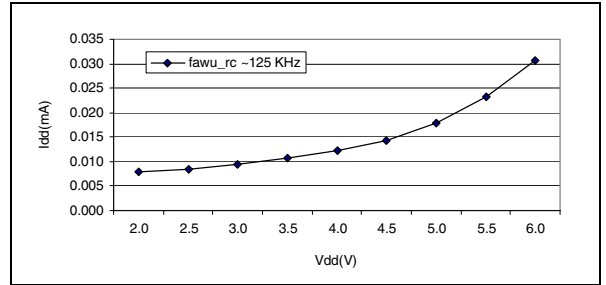
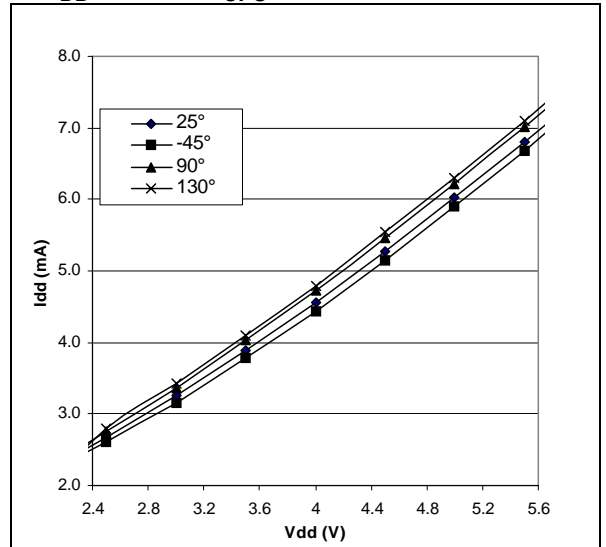


Figure 64. Typical  $I_{DD}$  vs. Temperature at  $V_{DD} = 5\text{V}$  and  $f_{CPU} = 8\text{MHz}$



13.4.2 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	
		$f_{ADC}=4\text{MHz}$ $V_{DD}=5.0\text{V}$	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.

### 13.5 CLOCK AND TIMING CHARACTERISTICS

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

#### 13.5.1 General Timings

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

#### Notes:

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.

#### 13.5.2 Auto Wakeup from Halt Oscillator (AWU)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$f_{AWU}$	AWU Oscillator Frequency		50	125	250	kHz
$t_{RCSRT}$	AWU Oscillator startup time				50	$\mu s$

### 13.6 MEMORY CHARACTERISTICS

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

#### 13.6.1 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

#### 13.6.2 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>2)</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>4)</sup>	$T_A = +55^{\circ}\text{C}$ <sup>3)</sup>	20			years
$N_{RW}$	Write erase cycles	$T_A = +25^{\circ}\text{C}$	10K <sup>7)</sup>			cycles
$I_{DD}$	Supply current	Read / Write / Erase modes $f_{CPU} = 8\text{MHz}$ , $V_{DD} = 5.5\text{V}$			2.6 <sup>6)</sup>	mA
		No Read/No Write Mode			100	$\mu\text{A}$
		Power down mode / HALT		0	0.1	$\mu\text{A}$

#### 13.6.3 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>4)</sup>	$T_A = +55^{\circ}\text{C}$ <sup>3)</sup>	20			years
$N_{RW}$	Write erase cycles	$T_A = +25^{\circ}\text{C}$	300K <sup>7)</sup>			cycles

#### Notes:

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.
2. Up to 32 bytes can be programmed at a time.
3. The data retention time increases when the  $T_A$  decreases.
4. Data based on reliability test results and monitored in production.
5. Data based on characterization results, not tested in production.
6. Guaranteed by Design. Not tested in production.
7. Design target value pending full product characterization.

## 13.7 EMC CHARACTERISTICS

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

### 13.7.1 Functional EMS (Electro Magnetic Susceptibility)

Based on a simple running application on the product (toggling 2 LEDs through I/O ports), the product is stressed by two electro magnetic events until a failure occurs (indicated by the LEDs).

- **ESD:** Electro-Static 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 100pF 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.

#### 13.7.1.1 Designing hardened software to avoid noise problems

EMC characterization and optimization are performed at component level with a typical applica-

tion 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 behaviour is detected, the software can be hardened to prevent unrecoverable errors occurring (see application note AN1015).

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

### 13.7.2 Electro Magnetic 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.

Symbol	Parameter	Conditions	Monitored Frequency Band	Max vs. [ $f_{OSC}/f_{CPU}$ ]		Unit
				8/4MHz	16/8MHz	
$S_{EMI}$	Peak level	$V_{DD}=5V$ , $T_A=+25^{\circ}C$ , SO20 package, conforming to SAE J 1752/3	0.1MHz to 30MHz	9	17	dB $\mu$ V
			30MHz to 130MHz	31	36	
			130MHz to 1GHz	25	27	
			SAE EMI Level	3.5	4	-

#### Notes:

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

**EMC CHARACTERISTICS (Cont'd)****13.7.3 Absolute Maximum Ratings (Electrical Sensitivity)**

Based on three different tests (ESD, LU and DLU) using specific measurement methods, the product is stressed in order to determine its performance in terms of electrical sensitivity. For more details, refer to the application note AN1181.

**13.7.3.1 Electro-Static Discharge (ESD)**

Electro-Static 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 JESD22-A114A/A115A standard.

**Absolute Maximum Ratings**

Symbol	Ratings	Conditions	Maximum value <sup>1)</sup>	Unit
V <sub>ESD(HBM)</sub>	Electro-static discharge voltage (Human Body Model)	T <sub>A</sub> =+25°C	4000	V

**Notes:**

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

**13.7.3.2 Static and Dynamic Latch-Up**

■ **LU:** 3 complementary static tests are required on 10 parts to assess the latch-up performance. A supply overvoltage (applied to each power supply pin) and a current injection (applied to each input, output and configurable I/O pin) are performed on each sample. This test conforms to the EIA/JESD 78 IC latch-up standard. For more details, refer to the application note AN1181.

■ **DLU:** Electro-Static Discharges (one positive then one negative test) are applied to each pin of 3 samples when the micro is running to assess the latch-up performance in dynamic mode. Power supplies are set to the typical values, the oscillator is connected as near as possible to the pins of the micro and the component is put in reset mode. This test conforms to the IEC1000-4-2 and SAEJ1752/3 standards. For more details, refer to the application note AN1181.

**Electrical Sensitivities**

Symbol	Parameter	Conditions	Class <sup>1)</sup>
LU	Static latch-up class	T <sub>A</sub> =+25°C	A
DLU	Dynamic latch-up class	V <sub>DD</sub> =5.5V, f <sub>OSC</sub> =4MHz, T <sub>A</sub> =+25°C	A

**Notes:**

1. Class description: A Class is an STMicroelectronics internal specification. All its limits are higher than the JEDEC specifications, that means when a device belongs to Class A it exceeds the JEDEC standard. B Class strictly covers all the JEDEC criteria (international standard).



## 13.8 I/O PORT PIN CHARACTERISTICS

### 13.8.1 General Characteristics

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

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 <sup>2)</sup>	Floating input mode			200	
$R_{PU}$	Weak pull-up equivalent resistor <sup>3)</sup>	$V_{IN} = V_{SS}$	50	120 160	250	$k\Omega$
		$V_{DD} = 5V$ $V_{DD} = 3V$				
$C_{IO}$	I/O pin capacitance			5		pF
$t_{f(I/O)out}$	Output high to low level fall time <sup>1)</sup>	$C_L = 50pF$ 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			$t_{CPU}$

#### Notes:

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 65). Data based on design simulation and/or technology characteristics, not tested in production.
3. The  $R_{PU}$  pull-up equivalent resistor is based on a resistive transistor (corresponding  $I_{PU}$  current characteristics described in Figure 66).
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 65. Two typical Applications with unused I/O Pin

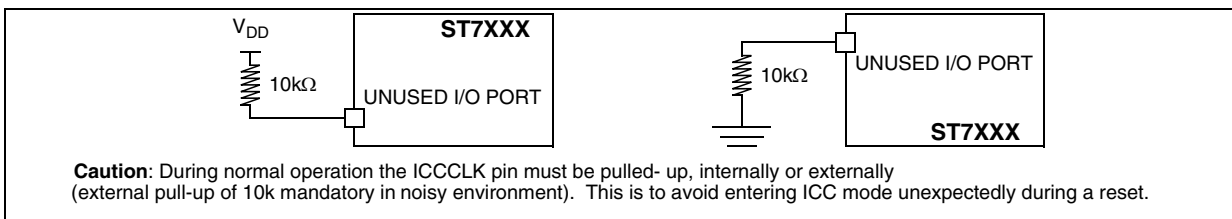
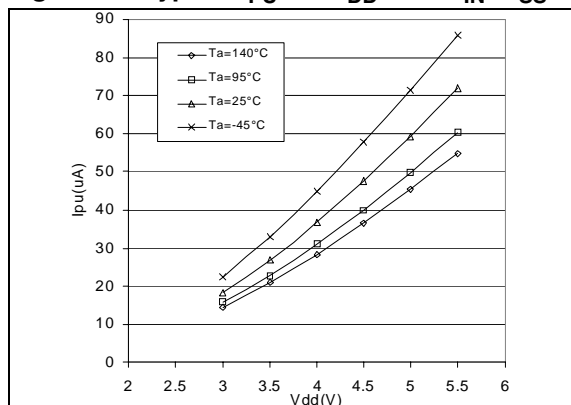


Figure 66. Typical  $I_{PU}$  vs.  $V_{DD}$  with  $V_{IN} = V_{SS}$



I/O PORT PIN CHARACTERISTICS (Cont'd)

13.8.2 Output Driving Current

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

Symbol	Parameter	Conditions	Min	Max	Unit	
$V_{OL}^{1)}$	Output low level voltage for a standard I/O pin when 8 pins are sunk at same time (see Figure 70)	$I_{IO}=+5mA$ $T_A \leq 85^\circ C$		1.0	V	
		$T_A \geq 85^\circ C$		1.2		
	Output low level voltage for a high sink I/O pin when 4 pins are sunk at same time (see Figure 72)	$I_{IO}=+2mA$ $T_A \leq 85^\circ C$	$V_{DD}=5V$			0.4
		$T_A \geq 85^\circ C$				0.5
		$I_{IO}=+20mA, T_A \leq 85^\circ C$				1.3
		$T_A \geq 85^\circ C$				1.5
Output high level voltage for an I/O pin when 4 pins are sourced at same time (see Figure 78)	$I_{IO}=-5mA, T_A \leq 85^\circ C$	$V_{DD}-1.5$				
	$T_A \geq 85^\circ C$	$V_{DD}-1.6$				
$V_{OH}^{2)}$	Output high level voltage for an I/O pin when 4 pins are sourced at same time (see Figure 78)	$I_{IO}=-2mA$ $T_A \leq 85^\circ C$	$V_{DD}-0.8$			
		$T_A \geq 85^\circ C$	$V_{DD}-1.0$			
$V_{OL}^{1)3)}$	Output low level voltage for a standard I/O pin when 8 pins are sunk at same time (see Figure 69)	$V_{DD}=3.3V$		0.5		
	Output low level voltage for a high sink I/O pin when 4 pins are sunk at same time			0.6		
$V_{OH}^{2)3)}$	Output high level voltage for an I/O pin when 4 pins are sourced at same time			$I_{IO}=+2mA$ $T_A \leq 85^\circ C$		0.5
				$T_A \geq 85^\circ C$		0.6
$V_{OL}^{1)3)}$	Output low level voltage for a standard I/O pin when 8 pins are sunk at same time (see Figure 68)			$V_{DD}=2.7V$		$V_{DD}-0.8$
	Output low level voltage for a high sink I/O pin when 4 pins are sunk at same time					$V_{DD}-1.0$
$V_{OH}^{2)3)}$	Output high level voltage for an I/O pin when 4 pins are sourced at same time (see Figure 75)	$I_{IO}=+2mA$ $T_A \leq 85^\circ C$				0.6
		$T_A \geq 85^\circ C$				0.7
$V_{OL}^{1)3)}$	Output low level voltage for a standard I/O pin when 8 pins are sunk at same time (see Figure 68)	$I_{IO}=+8mA$ $T_A \leq 85^\circ C$				0.6
		$T_A \geq 85^\circ C$				0.7
$V_{OH}^{2)3)}$	Output high level voltage for an I/O pin when 4 pins are sourced at same time (see Figure 75)	$I_{IO}=-2mA$ $T_A \leq 85^\circ C$	$V_{DD}-0.9$			
		$T_A \geq 85^\circ C$	$V_{DD}-1.0$			

Notes:

1. 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  $I_{VSS}$ .
2. The  $I_{IO}$  current sourced 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  $I_{VDD}$ .
3. Not tested in production, based on characterization results.

Figure 67. Typical  $V_{OL}$  at  $V_{DD}=2.4V$  (standard)

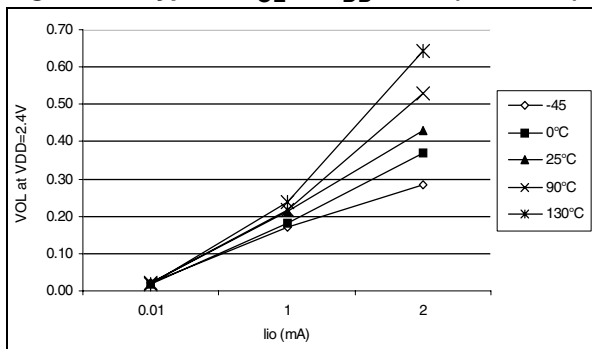
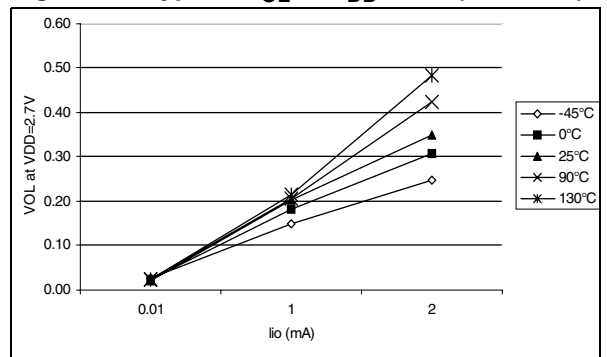


Figure 68. Typical  $V_{OL}$  at  $V_{DD}=2.7V$  (standard)



I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 69. Typical  $V_{OL}$  at  $V_{DD}=3.3V$  (standard)

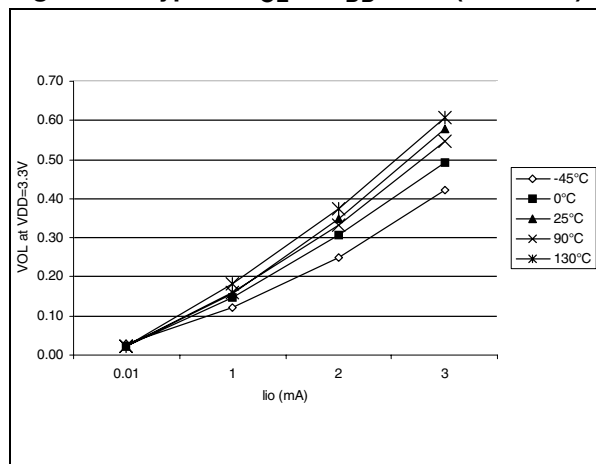


Figure 70. Typical  $V_{OL}$  at  $V_{DD}=5V$  (standard)

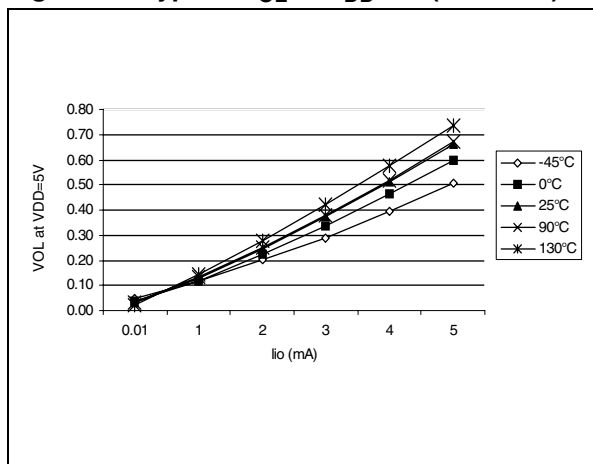


Figure 71. Typical  $V_{OL}$  at  $V_{DD}=2.4V$  (high-sink)

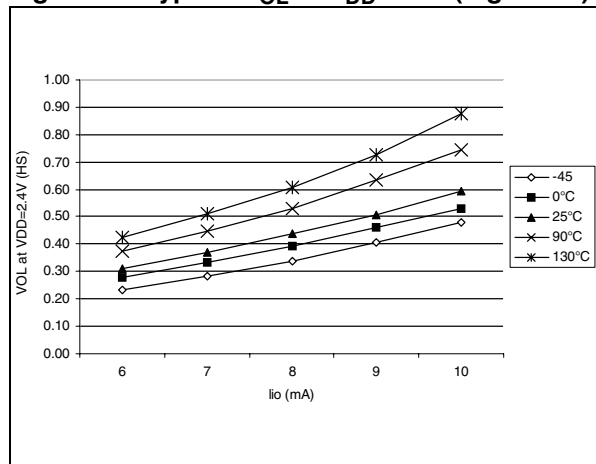


Figure 73. Typical  $V_{OL}$  at  $V_{DD}=3V$  (high-sink)

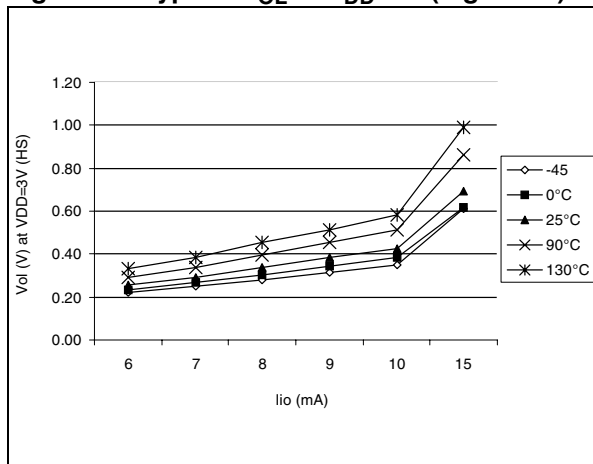
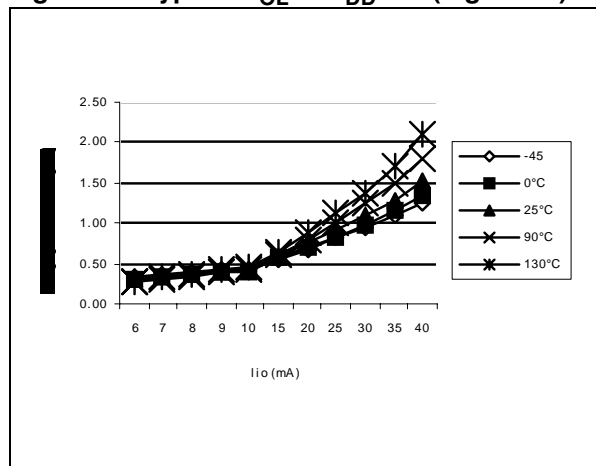


Figure 72. Typical  $V_{OL}$  at  $V_{DD}=5V$  (high-sink)



I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 74. Typical  $V_{DD}-V_{OH}$  at  $V_{DD}=2.4V$

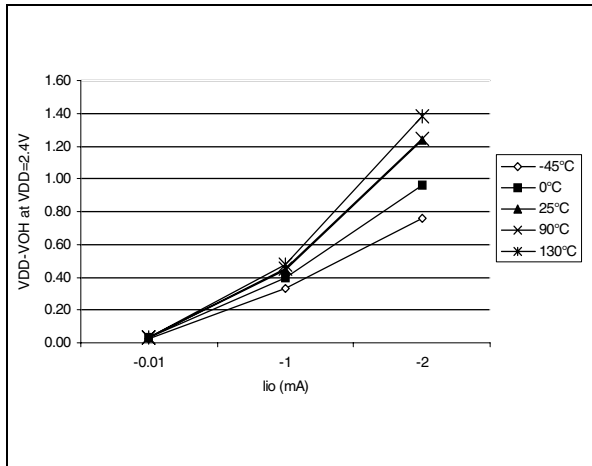


Figure 76. Typical  $V_{DD}-V_{OH}$  at  $V_{DD}=3V$

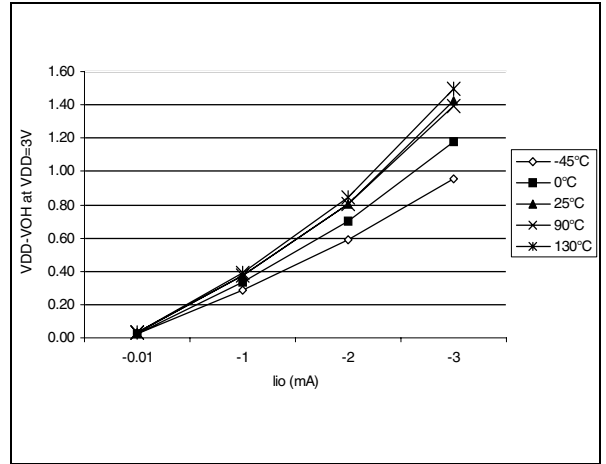


Figure 75. Typical  $V_{DD}-V_{OH}$  at  $V_{DD}=2.7V$

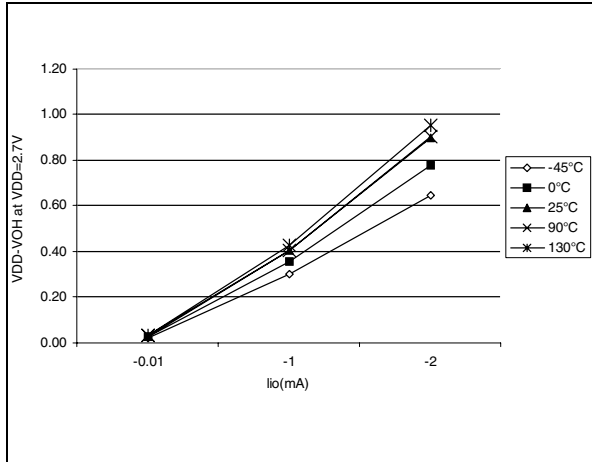


Figure 77. Typical  $V_{DD}-V_{OH}$  at  $V_{DD}=4V$

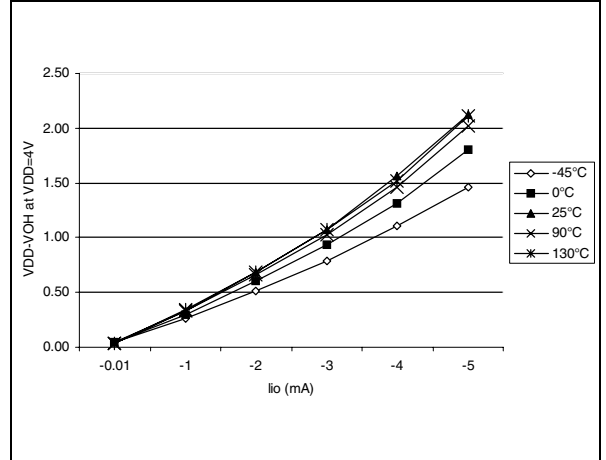
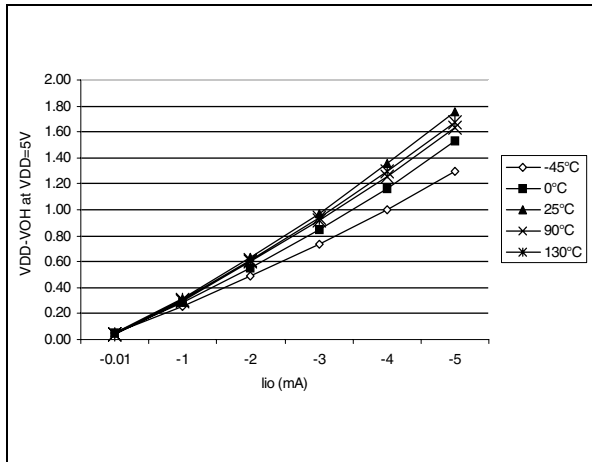


Figure 78. Typical  $V_{DD}-V_{OH}$  at  $V_{DD}=5V$



I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 79. Typical  $V_{OL}$  vs.  $V_{DD}$  (standard I/Os)

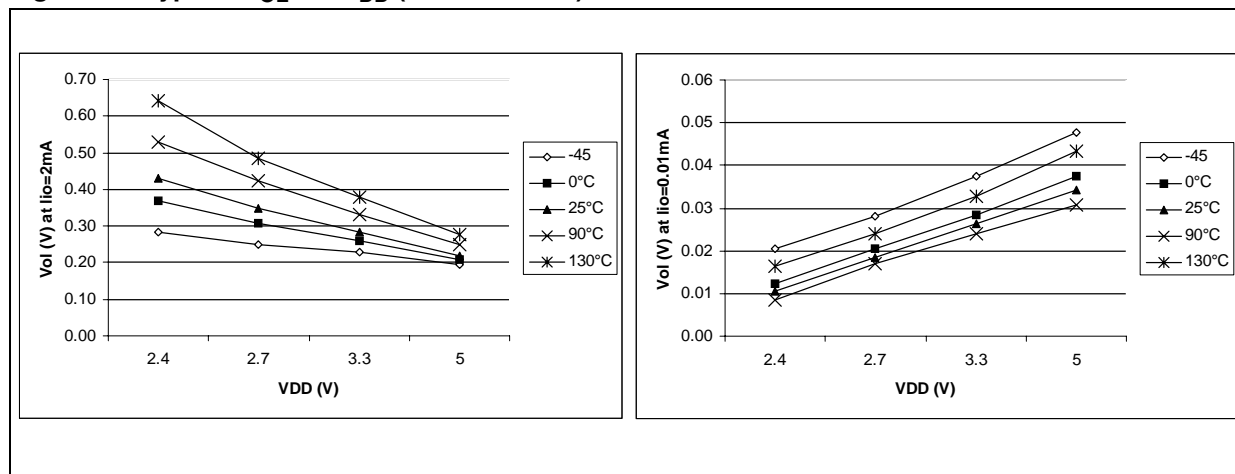


Figure 80. Typical  $V_{OL}$  vs.  $V_{DD}$  (high-sink I/Os)

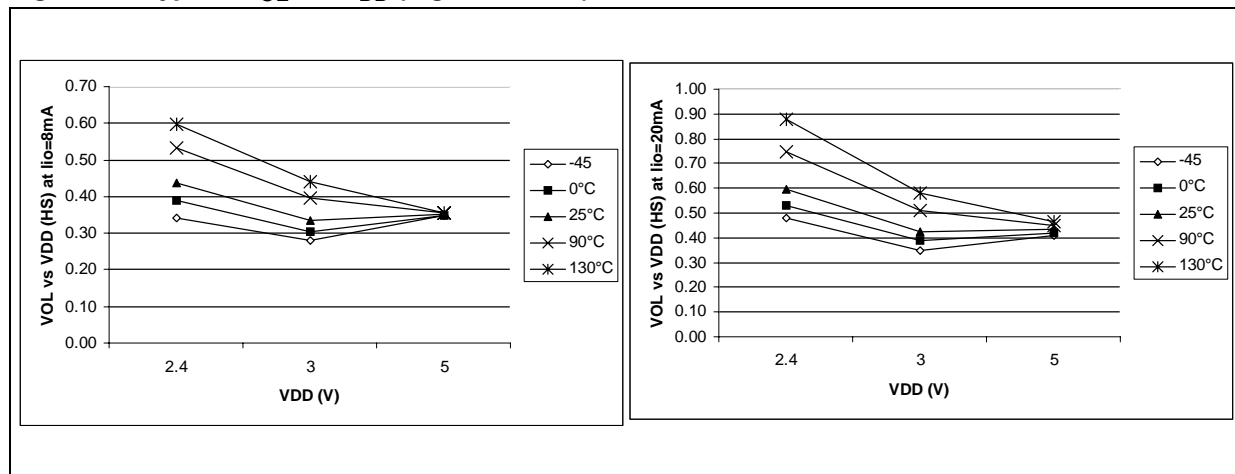
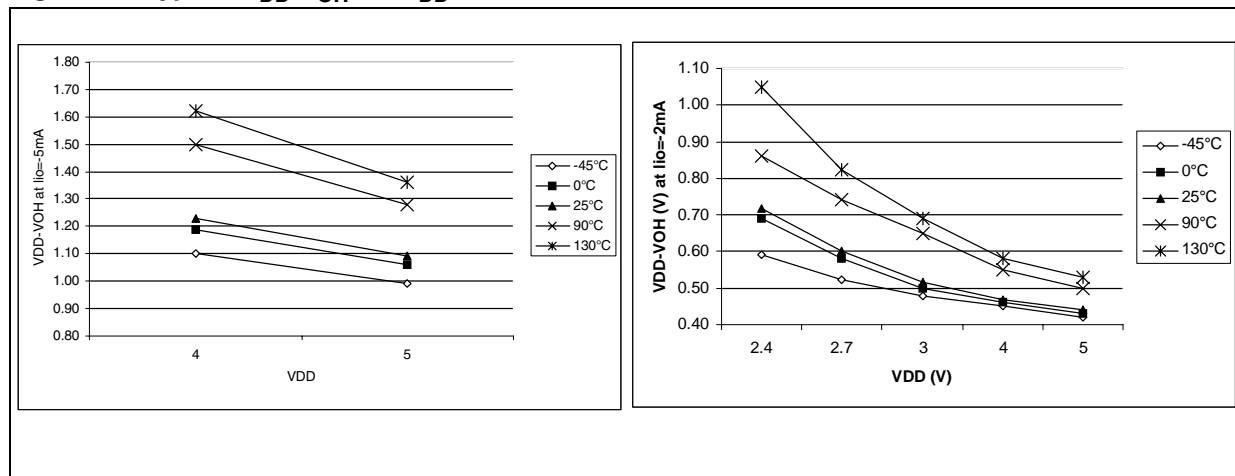


Figure 81. Typical  $V_{DD}-V_{OH}$  vs.  $V_{DD}$



## 13.9 CONTROL PIN CHARACTERISTICS

### 13.9.1 Asynchronous $\overline{\text{RESET}}$ Pin

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

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>	$V_{DD}=5\text{V}$	$I_{IO}=+5\text{mA}$ $T_A \leq 85^\circ\text{C}$	0.5	1.0	V
			$T_A \geq 85^\circ\text{C}$		1.2	
$R_{ON}$	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	
$t_{w(\text{RSTL})\text{out}}$	Generated reset pulse duration	Internal reset sources		30		$\mu\text{s}$
$t_{h(\text{RSTL})\text{in}}$	External reset pulse hold time <sup>4)</sup>		20			$\mu\text{s}$
$t_{g(\text{RSTL})\text{in}}$	Filtered glitch duration			200		ns

#### Notes:

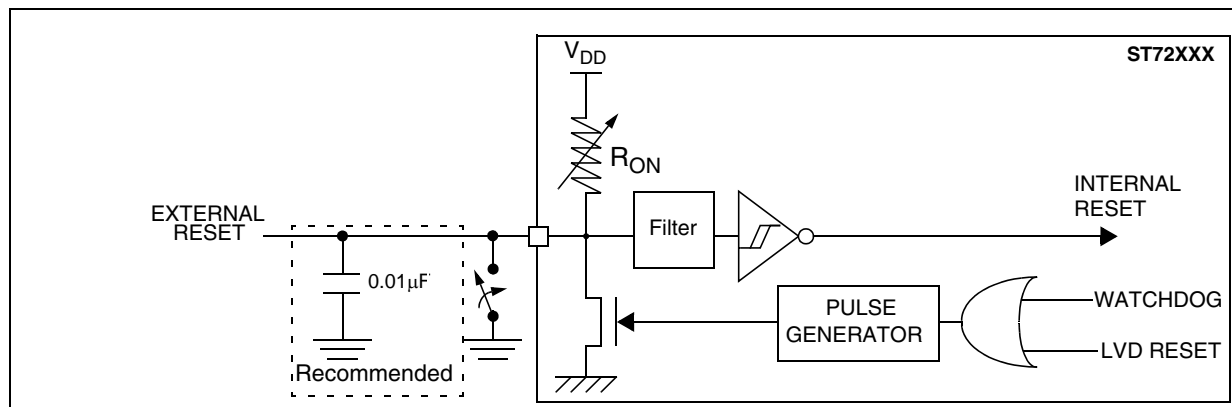
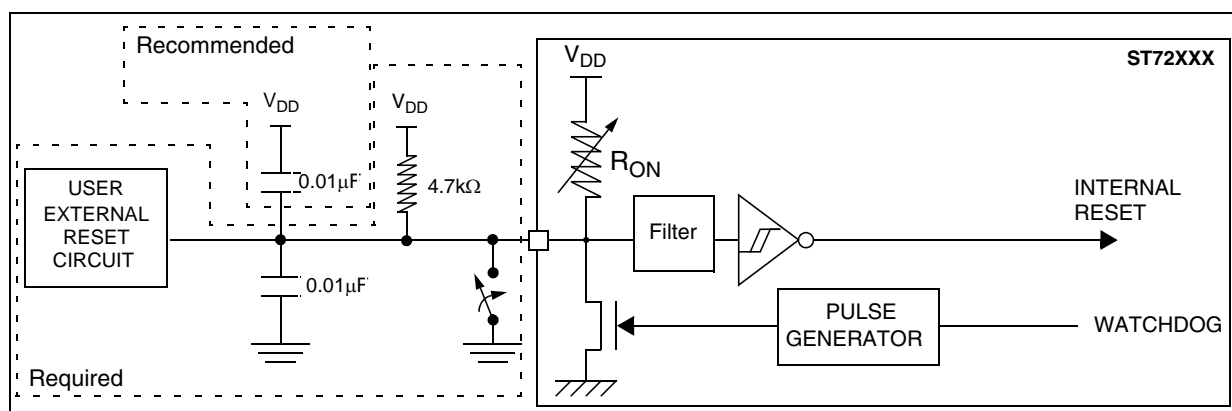
1. Data based on characterization results, not tested in production.

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  $V_{IL\text{max}}$  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  $t_{h(\text{RSTL})\text{in}}$  can be ignored.

## CONTROL PIN CHARACTERISTICS (Cont'd)

Figure 82.  $\overline{\text{RESET}}$  pin protection when LVD is enabled.<sup>1)2)3)4)5)</sup>Figure 83.  $\overline{\text{RESET}}$  pin protection when LVD is disabled.<sup>1)2)3)</sup>

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  $V_{\text{IL}}$  max. level specified in [section 13.9.1 on page 110](#). 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  $I_{\text{INJ}}(\overline{\text{RESET}})$  in [section 13.2.2 on page 92](#).
5. When the LVD is enabled, it is mandatory not to connect a pull-up resistor and a capacitor to  $V_{\text{DD}}$  on the  $\overline{\text{RESET}}$  pin.

13.10 COMMUNICATION INTERFACE CHARACTERISTICS

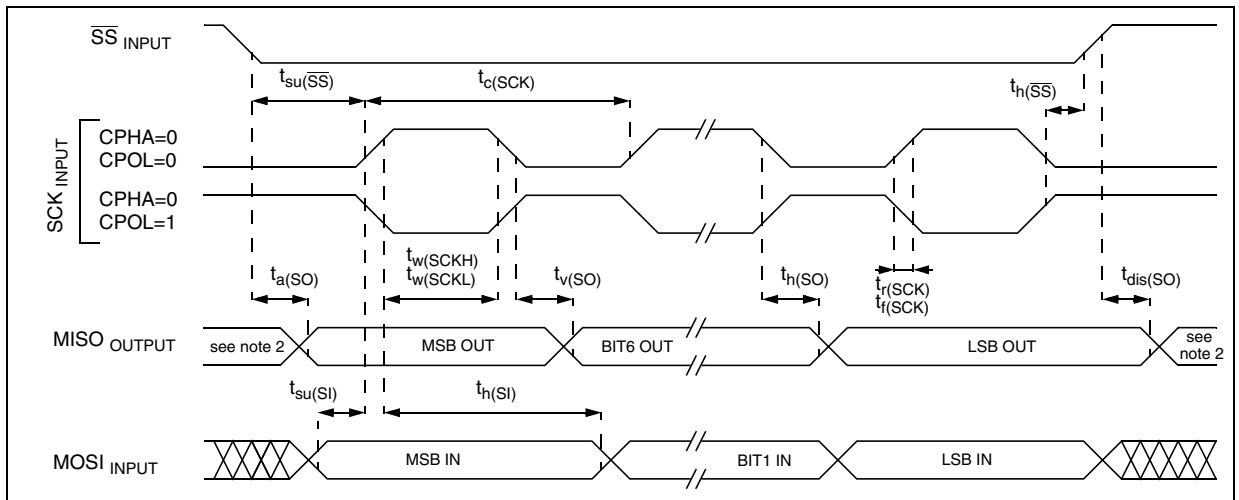
13.10.1 SPI - Serial Peripheral Interface

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 ( $\overline{SS}$ , SCK, MOSI, MISO).

Symbol	Parameter	Conditions	Min	Max	Unit
$f_{SCK} = 1/t_c(SCK)$	SPI clock frequency	Master $f_{CPU}=8MHz$	$f_{CPU}/128 = 0.0625$	$f_{CPU}/4 = 2$	MHz
		Slave $f_{CPU}=8MHz$	0	$f_{CPU}/24$	
$t_r(SCK)$ $t_f(SCK)$	SPI clock rise and fall time		see I/O port pin description		
$t_{su}(\overline{SS})$	$\overline{SS}$ setup time	Slave	120		ns
$t_h(\overline{SS})$	$\overline{SS}$ hold time	Slave	120		
$t_w(SCKH)$ $t_w(SCKL)$	SCK high and low time	Master	100		
		Slave	90		
$t_{su}(MI)$ $t_{su}(SI)$	Data input setup time	Master	100		
		Slave	100		
$t_h(MI)$ $t_h(SI)$	Data input hold time	Master	100		
		Slave	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 (before capture edge)	0.25		$t_{CPU}$
$t_h(MO)$	Data output hold time		0.25		

Figure 84. SPI Slave Timing Diagram with  $CPHA=0^3$



Notes:

1. Data based on design simulation and/or characterisation results, not tested in production.
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.
3. Measurement points are done at CMOS levels:  $0.3xV_{DD}$  and  $0.7xV_{DD}$ .



COMMUNICATION INTERFACE CHARACTERISTICS (Cont'd)

Figure 85. SPI Slave Timing Diagram with CPHA=1<sup>1)</sup>

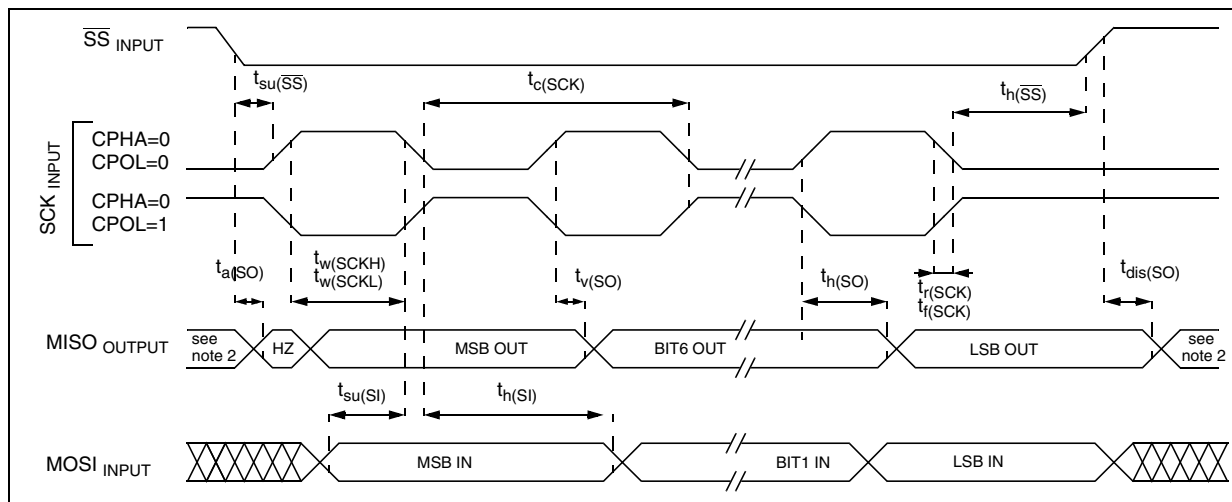
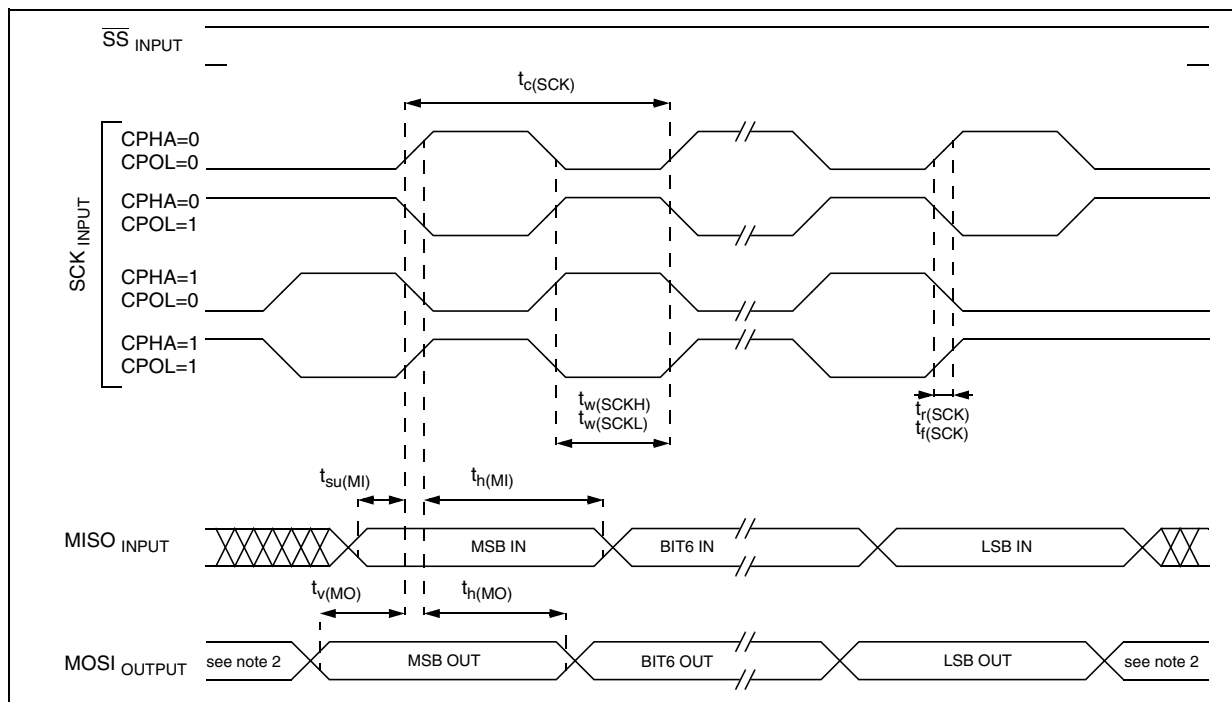


Figure 86. SPI Master Timing Diagram <sup>1)</sup>



Notes:

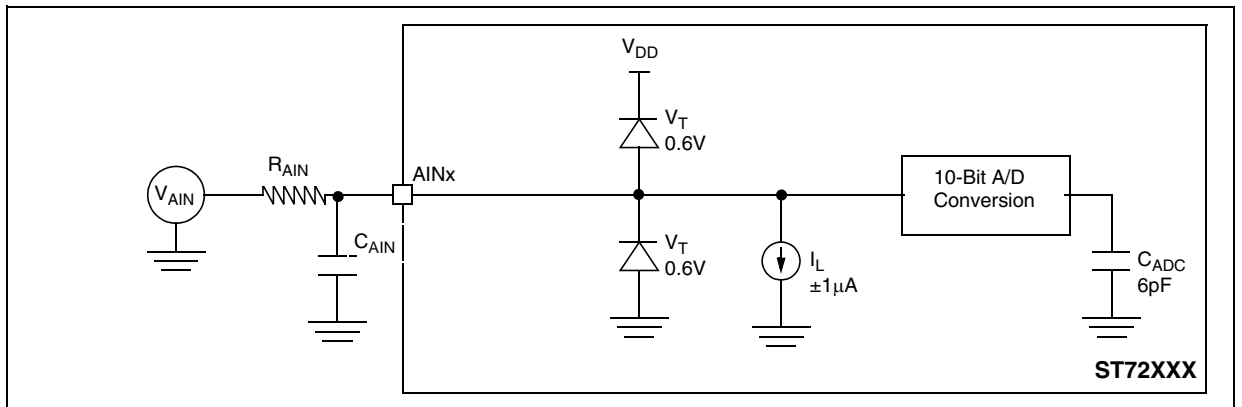
1. Measurement points are done at CMOS levels:  $0.3xV_{DD}$  and  $0.7xV_{DD}$ .
2. When no communication is on-going the data output line of the SPI (MOSI in master mode, MISO in slave mode) has its alternate function capability released. In this case, the pin status depends of the I/O port configuration.

13.11 10-BIT ADC CHARACTERISTICS

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

Symbol	Parameter	Conditions	Min	Typ <sup>1)</sup>	Max	Unit
$f_{ADC}$	ADC clock frequency				4	MHz
$V_{AIN}$	Conversion voltage range <sup>2)</sup>		$V_{SSA}$		$V_{DDA}$	V
$R_{AIN}$	External input resistor				$10^{3)}$	k $\Omega$
$C_{ADC}$	Internal sample and hold capacitor			6		pF
$t_{STAB}$	Stabilization time after ADC enable	$f_{CPU}=8MHz, f_{ADC}=4MHz$		0 <sup>4)</sup>		$\mu s$
$t_{ADC}$	Conversion time (Sample+Hold)			3.5		
	- Sample capacitor loading time - Hold conversion time			4 10		1/ $f_{ADC}$
$I_{ADC}$	Analog Part				1	mA
	Digital Part				0.2	

Figure 87. Typical Application with ADC



Notes:

1. Unless otherwise specified, typical data are based on  $T_A=25^\circ C$  and  $V_{DD}-V_{SS}=5V$ . 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.

## ADC CHARACTERISTICS (Cont'd)

ADC Accuracy with  $V_{DD}=5.0V$ 

Symbol	Parameter	Conditions	Typ	Max	Unit
$ E_T $	Total unadjusted error <sup>2)</sup>	$f_{CPU}=8MHz, f_{ADC}=4MHz$ <sup>1)</sup> , $V_{DD}=5.0V$	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	

## Notes:

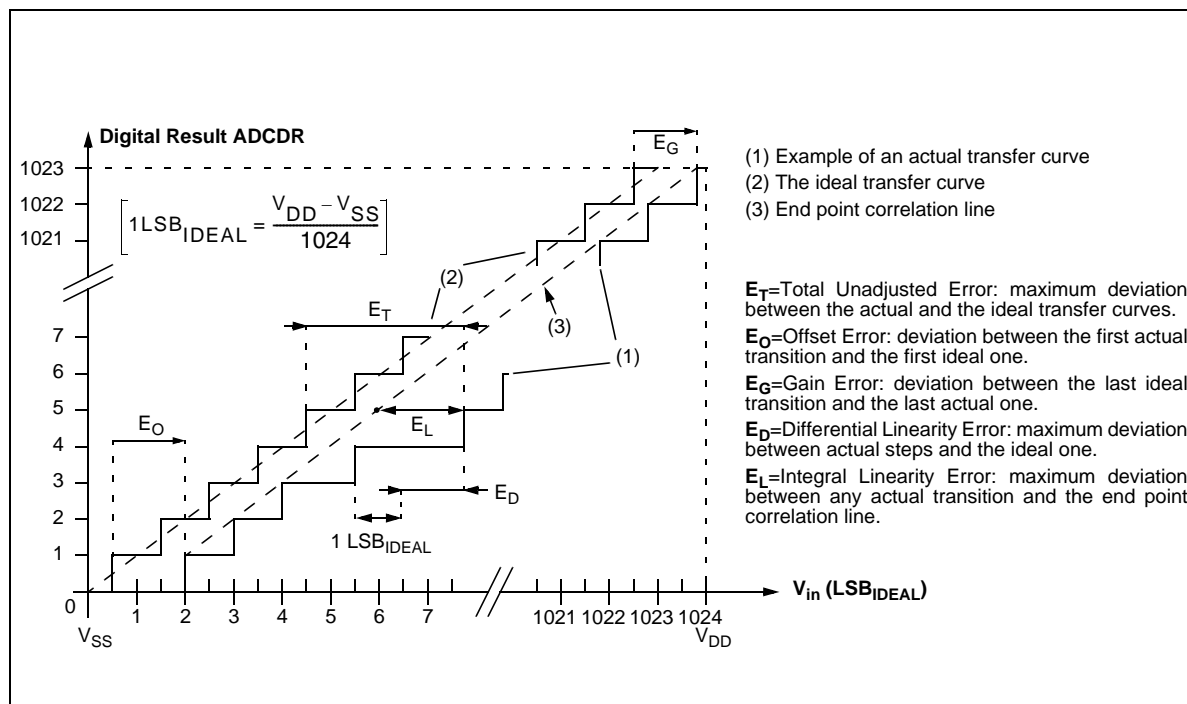
1) Data based on characterization results over the whole temperature range, monitored in production.

2) 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.

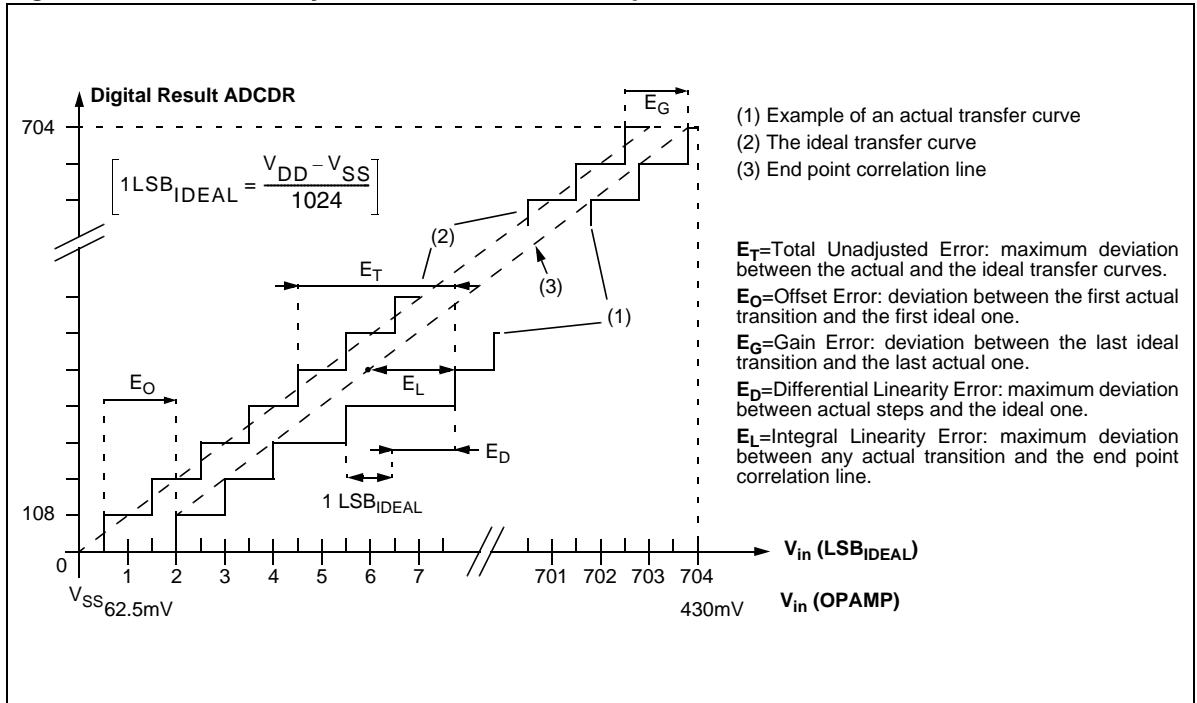
Any positive injection current within the limits specified for  $I_{INJ(PIN)}$  and  $\Sigma I_{INJ(PIN)}$  in Section 13.8 does not affect the ADC accuracy.

Figure 88. ADC Accuracy Characteristics with amplifier disabled

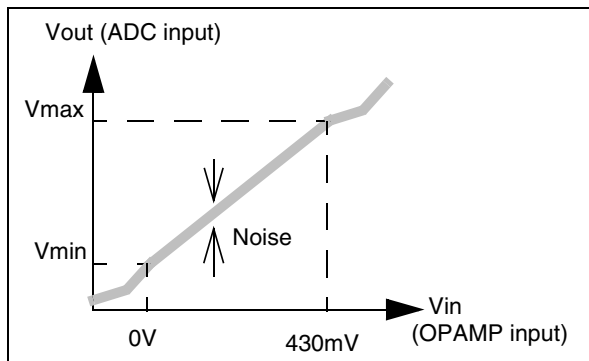


ADC CHARACTERISTICS (Cont'd)

Figure 89. 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  $f_{CPU} = 8MHz$ , then SPEED=0, SLOW=1).



**ADC CHARACTERISTICS** (Cont'd)

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

**Notes:**

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

**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

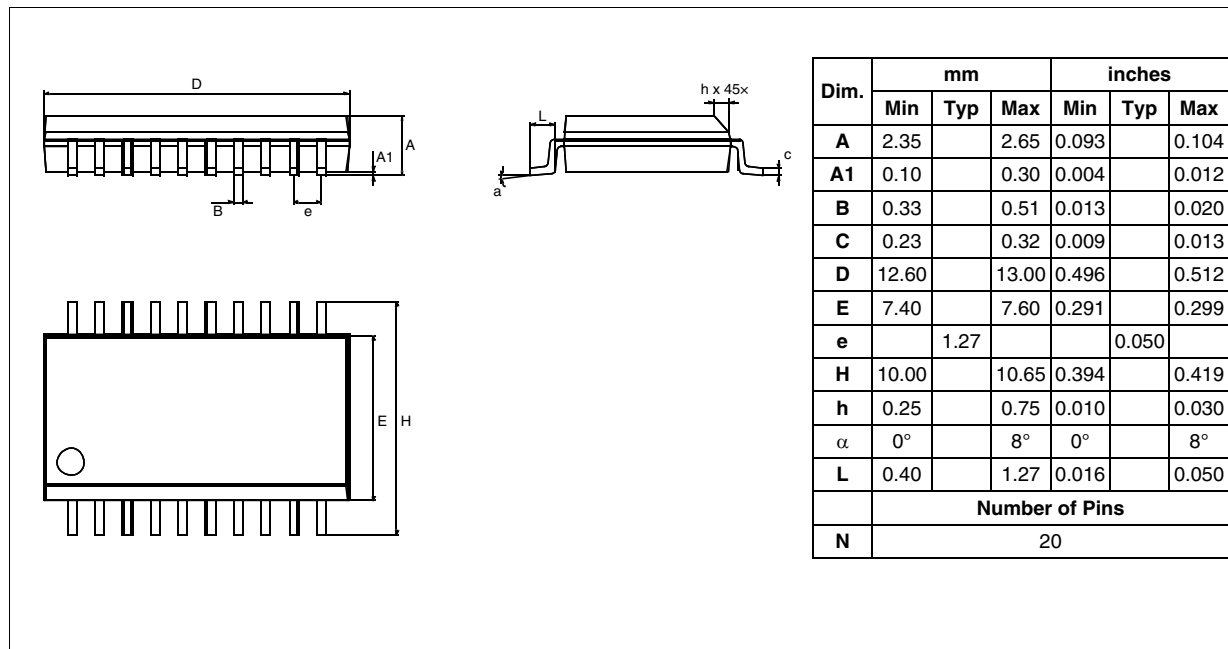
and during temperature variation. The table below gives the typical offset variation over temperature:

Typical Offset Variation (LSB)				UNIT
-45	-20	+25	+90	°C
-12	-7	-	+13	LSB

## 14 PACKAGE CHARACTERISTICS

## 14.1 PACKAGE MECHANICAL DATA

Figure 90. 20-Pin Plastic Small Outline Package, 300-mil Width



**PACKAGE CHARACTERISTICS** (Cont'd)**Table 21. THERMAL CHARACTERISTICS**

Symbol	Ratings	Value	Unit	
$R_{thJA}$	Package thermal resistance (junction to ambient)	SO20 DIP20	125 TBD	°C/W
$P_D$	Power dissipation <sup>1)</sup>		500	mW
$T_{Jmax}$	Maximum junction temperature <sup>2)</sup>		150	°C

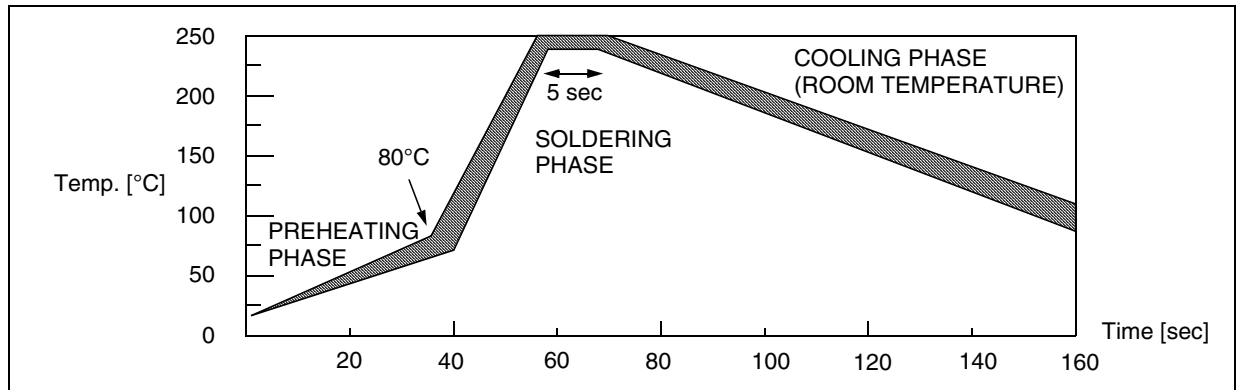
**Notes:**

1. The power dissipation is obtained from 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 determined by the user.
2. The average chip-junction temperature can be obtained from the formula  $T_J = T_A + P_D \times R_{thJA}$ .

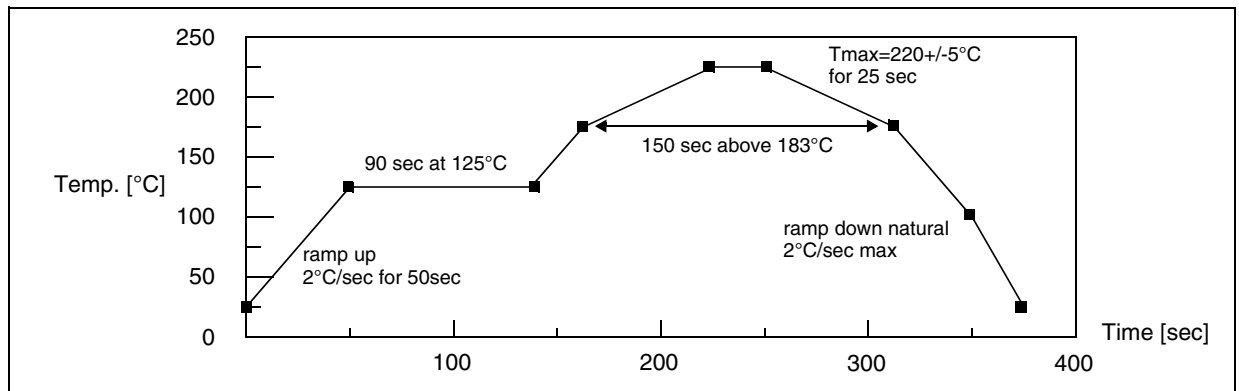
## 14.2 SOLDERING AND GLUEABILITY INFORMATION

Recommended soldering information given only as design guidelines.

**Figure 91. Recommended Wave Soldering Profile (with 37% Sn and 63% Pb)**



**Figure 92. Recommended Reflow Soldering Oven Profile (MID JEDEC)**



Recommended glue for SMD plastic packages:

- Heraeus: PD945, PD955
- Loctite: 3615, 3298



## 15 DEVICE CONFIGURATION

Each device is available for production in user programmable versions (FLASH).

### 15.1 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

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.

			OSCRANGE		
			2	1	0
Typ. frequency range with Resonator	LP	1~2MHz	0	0	0
	MP	2~4MHz	0	0	1
	MS	4~8MHz	0	1	0
	HS	8~16MHz	0	1	1
	VLP	32.768kHz	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 7.5](#)).

ST7FLITE1 devices are shipped to customers with a default program memory content (FFh). This implies that FLASH devices have to be configured by the customer using the Option Bytes .

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

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

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

#### OPT1 = **FMP\_R** *Read-out 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 4.5 on page 14](#) for more details

0: Read-out protection off

1: Read-out protection on

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

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

**Warning:** 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

	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

**OPTION BYTES** (Cont'd)

**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:** 1% RC oscillator available on ST7LITE15 and ST7LITE19 devices only

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

These option bits enable the LVD block with a selected threshold as shown in [Table 22](#).

**Table 23. 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
2.4V - 3.3V	Internal RC 1% <sup>1)</sup>	off	0.7MHz @3V	0	1	1
		x4	2.8MHz @3V	0	0	0
		x8	-	-	-	-
	External clock or oscillator (depending on OPT6:4 selection)	off	0-4MHz	1	1	1
		x4	4MHz	1	0	0
		x8	-	-	-	-
3.3V - 5.5V	Internal RC 1% <sup>1)</sup>	off	1MHz @5V	0	1	1
		x4	-	-	-	-
		x8	8MHz @5V	0	0	1
	External clock or oscillator (depending on OPT6:4 selection)	off	0-8MHz	1	1	1
		x4	-	-	-	-
		x8	8 MHz	1	0	1

Note 1: Configuration available on ST7LITE15 and ST7LITE19 devices only

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

**Table 22. 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

**15.2 DEVICE ORDERING INFORMATION****Table 24. Supported part numbers**

Part Number	Program Memory (Bytes)	RAM (Bytes)	Data EEPROM (Bytes)	Temp. Range	Package
ST7FLITE10F1M6	4K FLASH	256	-	-40°C to 85°C	SO20
ST7FLITE15F1M6			-		SO20
ST7FLITE19F1M6			128		SO20

15.3 DEVELOPMENT TOOLS

STMicroelectronics offers a range of hardware and software development tools for the ST7 micro-controller family. Full details of tools available for the ST7 from third party manufacturers can be obtained from the STMicroelectronics Internet site: <http://www.st.com>.

Tools from these manufacturers include C compilers, evaluation tools, emulators and programmers.

**Emulators**

Two types of emulators are available from ST for the ST7LITE1 family:

- **ST7 DVP3** entry-level emulator offers a flexible and modular debugging and programming solution. SO20 packages need a specific connection kit (refer to [Table 25](#))
- **ST7 EMU3** high-end emulator is delivered with everything (probes, TEB, adapters etc.) needed to start emulating the ST7LITE1. To configure it to emulate other ST7 subfamily devices, the active probe for the ST7EMU3 can be changed and the ST7EMU3 probe is designed for easy interchange of TEBs (Target Emulation Board). See [Table 25](#).

**In-circuit Debugging Kit**

Two configurations are available from ST:

- **ST7FLIT2-IND/USB:** Low-cost In-Circuit Debugging kit from Softec Microsystems. Includes STX-InDART/USB board (USB port) (A promotion package of 15 ST7FLIT2-IND/USB can be ordered with the following order code: ST7FLIT2-IND/15)
- **STxF-INDART/USB** (A promotion package of 15 STxF-INDART/USB can be ordered with the following order code: STxF-INDART)

**Flash Programming tools**

- **ST7-STICK** ST7 In-circuit Communication Kit, a complete software/hardware package for programming ST7 Flash devices. It connects to a host PC parallel port and to the target board or socket board via ST7 ICC connector.
- **ICC Socket Boards** provide an easy to use and flexible means of programming ST7 Flash devices. They can be connected to any tool that supports the ST7 ICC interface, such as ST7 EMU3, ST7-DVP3, inDART, ST7-STICK, or many third-party development tools.

**Evaluation boards**

One evaluation tool is available from ST:

- **ST7FLIT2-COS/COM:** STReal time starter kit from Cosmic software for ST7FLITE2 and ST7FLITE1

**Table 25. STMicroelectronics Development Tools**

Supported Products	Emulation				Programming
	ST7 DVP3 Series		ST7 EMU3 series		
	Emulator	Connection kit	Emulator	Active Probe & T.E.B.	ICC Socket Board
ST7FLITE10 ST7FLITE15 ST7FLITE19	ST7MDT10-DVP3	ST7MDT10-20/ DVP	ST7MDT10-EMU3	ST7MDT10-TEB	ST7SB10/123 <sup>1)</sup>

Note 1: Add suffix /EU, /UK, /US for the power supply of your region.

## 15.4 ST7 APPLICATION NOTES

**Table 26. ST7 Application Notes**

IDENTIFICATION	DESCRIPTION
<b>APPLICATION EXAMPLES</b>	
AN1658	SERIAL NUMBERING IMPLEMENTATION
AN1720	MANAGING THE READ-OUT PROTECTION IN FLASH MICROCONTROLLERS
AN1755	A HIGH RESOLUTION/PRECISION THERMOMETER USING ST7 AND NE555
AN1756	CHOOSING A DALI IMPLEMENTATION STRATEGY WITH ST7DALI
<b>EXAMPLE DRIVERS</b>	
AN 969	SCI COMMUNICATION BETWEEN ST7 AND PC
AN 970	SPI COMMUNICATION BETWEEN ST7 AND EEPROM
AN 971	I <sup>2</sup> C COMMUNICATION BETWEEN ST7 AND M24CXX EEPROM
AN 972	ST7 SOFTWARE SPI MASTER COMMUNICATION
AN 973	SCI SOFTWARE COMMUNICATION WITH A PC USING ST72251 16-BIT TIMER
AN 974	REAL TIME CLOCK WITH ST7 TIMER OUTPUT COMPARE
AN 976	DRIVING A BUZZER THROUGH ST7 TIMER PWM FUNCTION
AN 979	DRIVING AN ANALOG KEYBOARD WITH THE ST7 ADC
AN 980	ST7 KEYPAD DECODING TECHNIQUES, IMPLEMENTING WAKE-UP ON KEYSTROKE
AN1017	USING THE ST7 UNIVERSAL SERIAL BUS MICROCONTROLLER
AN1041	USING ST7 PWM SIGNAL TO GENERATE ANALOG OUTPUT (SINUSOĪD)
AN1042	ST7 ROUTINE FOR I <sup>2</sup> C SLAVE MODE MANAGEMENT
AN1044	MULTIPLE INTERRUPT SOURCES MANAGEMENT FOR ST7 MCUS
AN1045	ST7 S/W IMPLEMENTATION OF I <sup>2</sup> C BUS MASTER
AN1046	UART EMULATION SOFTWARE
AN1047	MANAGING RECEPTION ERRORS WITH THE ST7 SCI PERIPHERALS
AN1048	ST7 SOFTWARE LCD DRIVER
AN1078	PWM DUTY CYCLE SWITCH IMPLEMENTING TRUE 0% & 100% DUTY CYCLE
AN1082	DESCRIPTION OF THE ST72141 MOTOR CONTROL PERIPHERALS REGISTERS
AN1083	ST72141 BLDC MOTOR CONTROL SOFTWARE AND FLOWCHART EXAMPLE
AN1105	ST7 PCAN PERIPHERAL DRIVER
AN1129	PWM MANAGEMENT FOR BLDC MOTOR DRIVES USING THE ST72141
AN1130	AN INTRODUCTION TO SENSORLESS BRUSHLESS DC MOTOR DRIVE APPLICATIONS WITH THE ST72141
AN1148	USING THE ST7263 FOR DESIGNING A USB MOUSE
AN1149	HANDLING SUSPEND MODE ON A USB MOUSE
AN1180	USING THE ST7263 KIT TO IMPLEMENT A USB GAME PAD
AN1276	BLDC MOTOR START ROUTINE FOR THE ST72141 MICROCONTROLLER
AN1321	USING THE ST72141 MOTOR CONTROL MCU IN SENSOR MODE
AN1325	USING THE ST7 USB LOW-SPEED FIRMWARE V4.X
AN1445	EMULATED 16 BIT SLAVE SPI
AN1475	DEVELOPING AN ST7265X MASS STORAGE APPLICATION
AN1504	STARTING A PWM SIGNAL DIRECTLY AT HIGH LEVEL USING THE ST7 16-BIT TIMER
AN1602	16-BIT TIMING OPERATIONS USING ST7262 OR ST7263B ST7 USB MCUS
AN1633	DEVICE FIRMWARE UPGRADE (DFU) IMPLEMENTATION IN ST7 NON-USB APPLICATIONS
AN1712	GENERATING A HIGH RESOLUTION SINEWAVE USING ST7 PWMART
AN1713	SMBUS SLAVE DRIVER FOR ST7 I <sup>2</sup> C PERIPHERALS
AN1753	SOFTWARE UART USING 12-BIT ART
<b>GENERAL PURPOSE</b>	

Table 26. ST7 Application Notes

IDENTIFICATION	DESCRIPTION
AN1476	LOW COST POWER SUPPLY FOR HOME APPLIANCES
AN1526	ST7FLITE0 QUICK REFERENCE NOTE
AN1709	EMC DESIGN FOR ST MICROCONTROLLERS
AN1752	ST72324 QUICK REFERENCE NOTE
<b>PRODUCT EVALUATION</b>	
AN 910	PERFORMANCE BENCHMARKING
AN 990	ST7 BENEFITS VERSUS INDUSTRY STANDARD
AN1077	OVERVIEW OF ENHANCED CAN CONTROLLERS FOR ST7 AND ST9 MCUS
AN1086	U435 CAN-DO SOLUTIONS FOR CAR MULTIPLEXING
AN1103	IMPROVED B-EMF DETECTION FOR LOW SPEED, LOW VOLTAGE WITH ST72141
AN1150	BENCHMARK ST72 VS PC16
AN1151	PERFORMANCE COMPARISON BETWEEN ST72254 & PC16F876
AN1278	LIN (LOCAL INTERCONNECT NETWORK) SOLUTIONS
<b>PRODUCT MIGRATION</b>	
AN1131	MIGRATING APPLICATIONS FROM ST72511/311/214/124 TO ST72521/321/324
AN1322	MIGRATING AN APPLICATION FROM ST7263 REV.B TO ST7263B
AN1365	GUIDELINES FOR MIGRATING ST72C254 APPLICATIONS TO ST72F264
AN1604	HOW TO USE ST7MDT1-TRAIN WITH ST72F264
<b>PRODUCT OPTIMIZATION</b>	
AN 982	USING ST7 WITH CERAMIC RENATOR
AN1014	HOW TO MINIMIZE THE ST7 POWER CONSUMPTION
AN1015	SOFTWARE TECHNIQUES FOR IMPROVING MICROCONTROLLER EMC PERFORMANCE
AN1040	MONITORING THE VBUS SIGNAL FOR USB SELF-POWERED DEVICES
AN1070	ST7 CHECKSUM SELF-CHECKING CAPABILITY
AN1181	ELECTROSTATIC DISCHARGE SENSITIVE MEASUREMENT
AN1324	CALIBRATING THE RC OSCILLATOR OF THE ST7FLITE0 MCU USING THE MAINS
AN1502	EMULATED DATA EEPROM WITH ST7 HDFLASH MEMORY
AN1529	EXTENDING THE CURRENT & VOLTAGE CAPABILITY ON THE ST7265 VDDF SUPPLY
AN1530	ACCURATE TIMEBASE FOR LOW-COST ST7 APPLICATIONS WITH INTERNAL RC OSCILLATOR
AN1605	USING AN ACTIVE RC TO WAKEUP THE ST7LITE0 FROM POWER SAVING MODE
AN1636	UNDERSTANDING AND MINIMIZING ADC CONVERSION ERRORS
AN1828	PIR (PASSIVE INFRARED) DETECTOR USING THE ST7FLITE05/09/SUPERLITE
AN1971	ST7LITE0 MICROCONTROLLED BALLAST
<b>PROGRAMMING AND TOOLS</b>	
AN 978	ST7 VISUAL DEVELOP SOFTWARE KEY DEBUGGING FEATURES
AN 983	KEY FEATURES OF THE COSMIC ST7 C-COMPILER PACKAGE
AN 985	EXECUTING CODE IN ST7 RAM
AN 986	USING THE INDIRECT ADDRESSING MODE WITH ST7
AN 987	ST7 SERIAL TEST CONTROLLER PROGRAMMING
AN 988	STARTING WITH ST7 ASSEMBLY TOOL CHAIN
AN 989	GETTING STARTED WITH THE ST7 HIWARE C TOOLCHAIN
AN1039	ST7 MATH UTILITY ROUTINES
AN1064	WRITING OPTIMIZED HIWARE C LANGUAGE FOR ST7
AN1071	HALF DUPLEX USB-TO-SERIAL BRIDGE USING THE ST72611 USB MICROCONTROLLER
AN1106	TRANSLATING ASSEMBLY CODE FROM HC05 TO ST7

Table 26. ST7 Application Notes

IDENTIFICATION	DESCRIPTION
AN1179	PROGRAMMING ST7 FLASH MICROCONTROLLERS IN REMOTE ISP MODE (IN-SITU PROGRAMMING)
AN1446	USING THE ST72521 EMULATOR TO DEBUG A ST72324 TARGET APPLICATION
AN1477	EMULATED DATA EEPROM WITH XFLASH MEMORY
AN1478	PORTING AN ST7 PANTA PROJECT TO CODEWARRIOR IDE
AN1527	DEVELOPING A USB SMARTCARD READER WITH ST7SCR
AN1575	ON-BOARD PROGRAMMING METHODS FOR XFLASH AND HDFLASH ST7 MCUS
AN1576	IN-APPLICATION PROGRAMMING (IAP) DRIVERS FOR ST7 HDFLASH OR XFLASH MCUS
AN1577	DEVICE FIRMWARE UPGRADE (DFU) IMPLEMENTATION FOR ST7 USB APPLICATIONS
AN1601	SOFTWARE IMPLEMENTATION FOR ST7DALI-EVAL
AN1603	USING THE ST7 USB DEVICE FIRMWARE UPGRADE DEVELOPMENT KIT (DFU-DK)
AN1635	ST7 CUSTOMER ROM CODE RELEASE INFORMATION
AN1754	DATA LOGGING PROGRAM FOR TESTING ST7 APPLICATIONS VIA ICC
AN1796	FIELD UPDATES FOR FLASH BASED ST7 APPLICATIONS USING A PC COMM PORT
AN1900	HARDWARE IMPLEMENTATION FOR ST7DALI-EVAL
AN1904	ST7MC THREE-PHASE AC INDUCTION MOTOR CONTROL SOFTWARE LIBRARY
<b>SYSTEM OPTIMIZATION</b>	
AN1711	SOFTWARE TECHNIQUES FOR COMPENSATING ST7 ADC ERRORS
AN1827	IMPLEMENTATION OF SIGMA-DELTA ADC WITH ST7FLITE05/09

## 16 IMPORTANT NOTES

### 16.1 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).

### 16.2 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.

### 16.3 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.

### 16.4 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 in-

jection 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.

### 16.5 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

### 16.6 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.



## 17 REVISION HISTORY

**Table 27. Revision History**

Date	Revision	Description of Changes
July 03	1.3	<p>Changed <a href="#">section 3 on page 9</a> and <a href="#">Figure 3</a>            Added note on RC oscillator in <a href="#">section 7 on page 23</a> (main features) and changed <a href="#">section 7.1 on page 23</a>: removed reference to ST7LITE10 in RCCR table            Changed <a href="#">Figure 12 on page 25</a> (CLKIN/2, OSC/2)            Added note in <a href="#">section 7.4 on page 26</a> (external clock source paragraph)            Changed <a href="#">section 13.3.1 on page 93</a>: <math>f_{CLKIN}</math> instead of <math>f_{OSC}</math>            Added note in the description of OSC option bit and in <a href="#">Table 23 on page 122</a></p>
Dec-2004	2.0	<p>Revision number incremented from 1.3 to 2.0 due to Internal Document Management System change            Modified Caution to pin n°12 (SO20) or pin n°7 (DIP20) and added caution to PB0 and PB1 in <a href="#">Table 1 on page 7</a>            Changed Caution in <a href="#">section 4.4 on page 13</a>            Removed “optional” referring to VDD in <a href="#">Figure 4 on page 13</a>            In <a href="#">section 4.5.1 on page 14</a>: Changed 1st sentence and Clarification of Flash read-out protection            Replaced CRSR register by SICSR register in <a href="#">section 7.6.3 on page 32</a>            Added note in <a href="#">section 7.6.1 on page 29</a>            Reset delay in <a href="#">section 11.1.3 on page 51</a> changed to 30<math>\mu</math>s            MOD00 replaced by 0Ex in <a href="#">Figure 36 on page 57</a>            Added Note 2 related to Exit from Active Halt, <a href="#">section 11.2.5 on page 59</a>            Changed “Output Compare Mode” on page 57 and note 1 in <a href="#">section 11.2.6 on page 60</a>            Replaced FFh by FFFh in the description of OVF bit in <a href="#">section 11.2.6 on page 60</a>            Removed sentence relating to an effective change only after overflow for CK[1:0], <a href="#">page 60</a>            Replaced ICAP1 pin by LTIC Pin in <a href="#">section 11.3.3.3 on page 66</a>            Changed <a href="#">section 11.4.2 on page 70</a>            Changed <a href="#">section 11.4.3.3 on page 73</a>            Changed “An interrupt is generated if SPIE = 1 in the SPICSR register” to “An interrupt is generated if SPIE = 1 in the SPICR register” in description of OVR and MODF bits in <a href="#">section 11.4.8 on page 78</a>            Added illegal opcode detection to page 1, <a href="#">section 7.6 on page 29</a>, <a href="#">section 12 on page 85</a>            Removed references to “-40°C to +125°C” temperature range in <a href="#">section 13 on page 91</a>            Altered note 1 for <a href="#">section 13.2.3 on page 92</a> removing references to RESET            Changed note 2 in <a href="#">section 13.2.1 on page 92</a>            Added one row in <a href="#">section 13.2.2 on page 92</a> (PB0 and PB1)            Changed <a href="#">section 13.3 on page 93</a>  <math>f_{PLL}</math> value of 1MHz quoted as Typical instead of a Minimum in <a href="#">section 13.3.4.1 on page 95</a>            In <a href="#">section 13.4.1 on page 99</a>: Added note 5 and corrected <math>f_{CPU}</math> in SLOW and SLOW WAIT modes            Added data for Fcpu @ 1MHz into <a href="#">Section 13.4.1 Supply Current</a> table.            Updated <a href="#">Figure 61. Typical IDD in WAIT vs. fCPU</a> with correct data            Added <math>V_{DD}</math> row in <a href="#">section 13.6.3 on page 102</a>            Changed <a href="#">section 13.7 on page 103</a>            Added caution to <a href="#">Figure 65 on page 105</a>            Added <math>V_{IL}</math> min value and <math>V_{IH}</math> max value in <a href="#">section 13.8.1 on page 105</a> and in <a href="#">section 13.9.1 on page 110</a>            Modified “Asynchronous RESET Pin” on page 110 (<a href="#">Figure 82</a> and <a href="#">Figure 83</a>)            Updated <math>f_{SCK}</math> in <a href="#">section 13.10.1 on page 112</a> to <math>f_{CPU}/4</math> and <math>f_{CPU}/2</math>            Updated ADC accuracy table values on <a href="#">page 115</a>            Changed values in ADC Characteristics table on <a href="#">page 117</a>            Added note 4 and description relating to Total Percentage in Error and Amplifier Output Offset Variation to the ADC Characteristics subsection and table, <a href="#">page 117</a>            Added note 5 and description relating to Offset Variation in Temperature to ADC Characteristics subsection and table, <a href="#">page 117</a></p>

Dec-2004	2.0	<p>Changed FMP_R option bit description in <a href="#">section 15.1 on page 121</a></p> <p>Changed "DEVELOPMENT TOOLS" on page 124</p> <p>Added notes indicating that PB4 cannot be used as an external interrupt in HALT mode, <a href="#">section 16.6 on page 128</a> and <a href="#">Section 8.3 PERIPHERAL INTERRUPTS</a></p> <p>Added "NEGATIVE INJECTION IMPACT ON ADC ACCURACY" on page 128</p> <p>Added "CLEARING ACTIVE INTERRUPTS OUTSIDE INTERRUPT ROUTINE" on page 128</p> <p>Changed Table 23, "List of valid option combinations," on page 122: PLLx4x8 selection when PLL off</p>
----------	-----	---

**Notes:**

Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

The ST logo is a registered trademark of STMicroelectronics.

All other names are the property of their respective owners

© 2004 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)