

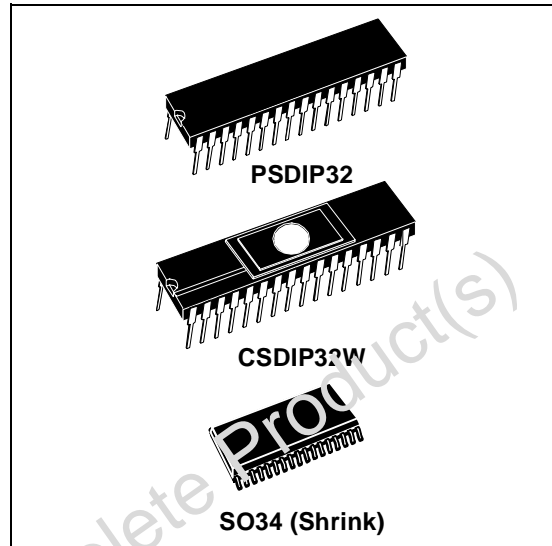


# ST7263

## LOW SPEED USB 8-BIT MCU FAMILY with up to 16K MEMORY, up to 512 BYTES RAM, 8-BIT ADC, WDG, TIMER, SCI & I<sup>2</sup>C

DATASHEET

- Up to 16Kbytes program memory
- Data RAM: up to 512 bytes with 64 bytes stack
- Run, Wait and Halt CPU modes
- 12 or 24 MHz oscillator
- RAM retention mode
- USB (Universal Serial Bus) Interface with DMA for low speed applications compliant with USB 1.5 Mbs specification (version 1.1) and USB HID specifications (version 1.0)
- Integrated 3.3V voltage regulator and transceivers
- Suspend and Resume operations
- 3 endpoints with programmable in/out configuration
- 19 programmable I/O lines with:
  - 8 high current I/Os (10mA at 1.3V)
  - 2 very high current pure Open Drain I/Os (25mA at 1.5V)
  - 8 lines individually programmable as interrupt inputs
- Optional Low Voltage Detector (LVD)
- Programmable Watchdog for system reliability
- 16-bit Timer with:
  - 2 Input Captures
  - 2 Output Compares
  - PWM Generation capabilities
  - External Clock input
- Asynchronous Serial Communications Interface (8K and 16K program memory versions only)
- I<sup>2</sup>C Multi Master Interface up to 400 KHz (16K program memory version only)



- 8-bit A/D Converter (ADC) with 8 channels
- Fully static operation
- 63 basic instructions
- 17 main addressing modes
- 8x8 unsigned multiply instruction
- True bit manipulation
- Versatile Development Tools (under Windows) including assembler, linker, C-compiler, archiver, source level debugger, software library, hardware emulator, programming boards and gang programmers

**Table 1. Device Summary**

Features	ST72631	ST72632	ST72633
ROM - OTP (bytes)	16K	8K	4K
RAM (stack) - bytes	512 (64)	256 (64)	
Peripherals	Watchdog, 16-bit timer, SCI, I <sup>2</sup> C, ADC, USB	Watchdog, 16-bit timer, SCI, ADC, USB	Watchdog, 16-bit timer, ADC, USB
Operating Supply	4.0V to 5.5V		
CPU frequency	8 Mhz (with 24 MHz oscillator) or 4 MHz (with 12 MHz oscillator)		
Operating temperature	0°C to +70°C		
Packages	SO34/SDIP32		
EPROM device	ST72E631 <sup>1</sup> (CSDIP32W)		

Note 1: EPROM version for development only

Rev. 1.8

---

# Table of Contents

---

<b>1 GENERAL DESCRIPTION</b> .....	<b>5</b>
1.1 INTRODUCTION .....	5
1.2 PIN DESCRIPTION .....	6
1.3 EXTERNAL CONNECTIONS .....	9
1.4 REGISTER & MEMORY MAP .....	10
1.5 EPROM/OTP PROGRAM MEMORY .....	13
1.5.1 EPROM ERASURE .....	13
<b>2 CENTRAL PROCESSING UNIT</b> .....	<b>14</b>
2.1 INTRODUCTION .....	14
2.2 MAIN FEATURES .....	14
2.3 CPU REGISTERS .....	14
<b>3 CLOCKS AND RESET</b> .....	<b>17</b>
3.1 CLOCK SYSTEM .....	17
3.1.1 General Description .....	17
3.1.2 External Clock .....	17
3.2 RESET .....	18
3.2.1 Low Voltage Detector (LVD) .....	18
3.2.2 Watchdog Reset .....	18
3.2.3 External Reset .....	18
<b>4 INTERRUPTS AND POWER SAVING MODES</b> .....	<b>20</b>
4.1 INTERRUPTS .....	20
4.1.1 Interrupt Register .....	22
4.2 POWER SAVING MODES .....	23
4.2.1 Introduction .....	23
4.2.2 HALT mode .....	23
4.2.3 WAIT mode .....	24
<b>5 ON-CHIP PERIPHERALS</b> .....	<b>25</b>
5.1 I/O PORTS .....	25
5.1.1 Introduction .....	25
5.1.2 Functional description .....	25
5.1.3 I/O Port Implementation .....	26
5.1.4 Port A .....	27
5.1.5 Port B .....	29
5.1.6 Port C .....	30
5.1.7 Register Description .....	31
5.2 MISCELLANEOUS REGISTER .....	32
5.3 WATCHDOG TIMER (WDG) .....	33
5.3.1 Introduction .....	33
5.3.2 Main Features .....	33
5.3.3 Functional Description .....	34
5.3.4 Interrupts .....	34
5.3.5 Register Description .....	34
5.4 16-BIT TIMER .....	36
5.4.1 Introduction .....	36
5.4.2 Main Features .....	36

---

## Table of Contents

---

5.4.3	Functional Description	36
5.4.4	Low Power Modes	48
5.4.5	Interrupts	48
5.4.6	Summary of Timer modes	48
5.4.7	Register Description	49
5.5	SERIAL COMMUNICATIONS INTERFACE (SCI)	54
5.5.1	Introduction	54
5.5.2	Main Features	54
5.5.3	General Description	54
5.5.4	Functional Description	56
5.5.5	Low Power Modes	60
5.5.6	Interrupts	60
5.5.7	Register Description	61
5.6	USB INTERFACE (USB)	65
5.6.1	Introduction	65
5.6.2	Main Features	65
5.6.3	Functional Description	65
5.6.4	Register Description	66
5.6.5	Programming Considerations	71
5.7	I <sup>2</sup> C BUS INTERFACE (I <sup>2</sup> C)	73
5.7.1	Introduction	73
5.7.2	Main Features	73
5.7.3	General Description	73
5.7.4	Functional Description	75
5.7.5	Low Power Modes	78
5.7.6	Interrupts	78
5.7.7	Register Description	79
5.8	8-BIT A/D CONVERTER (ADC)	84
5.8.1	Introduction	84
5.8.2	Main Features	84
5.8.3	Functional Description	85
5.8.4	Low Power Modes	85
5.8.5	Interrupts	85
5.8.6	Register Description	86
<b>6</b>	<b>INSTRUCTION SET</b>	<b>87</b>
6.1	ST7 ADDRESSING MODES	87
6.1.1	Inherent	88
6.1.2	Immediate	88
6.1.3	Direct	88
6.1.4	Indexed (No Offset, Short, Long)	88
6.1.5	Indirect (Short, Long)	88
6.1.6	Indirect Indexed (Short, Long)	89
6.1.7	Relative Mode (Direct, Indirect)	89
6.2	INSTRUCTION GROUPS	90
<b>7</b>	<b>ELECTRICAL CHARACTERISTICS</b>	<b>93</b>
7.1	ABSOLUTE MAXIMUM RATINGS	93
7.2	THERMAL CHARACTERISTICS	94

7.3 OPERATING CONDITIONS .....	95
7.4 POWER CONSUMPTION .....	96
7.5 I/O PORT CHARACTERISTICS .....	97
7.6 LOW VOLTAGE DETECTOR (LVD) CHARACTERISTICS .....	98
7.7 CONTROL TIMING CHARACTERISTICS .....	98
7.8 COMMUNICATION INTERFACE CHARACTERISTICS .....	99
7.8.1 USB - Universal Bus Interface .....	99
7.8.2 I2C - Inter IC Control Interface .....	101
7.9 8-BIT ADC CHARACTERISTICS .....	102
<b>8 PACKAGE CHARACTERISTICS .....</b>	<b>104</b>
8.1 PACKAGE MECHANICAL DATA .....	104
<b>9 DEVICE CONFIGURATION AND ORDERING INFORMATION .....</b>	<b>106</b>
9.1 DEVICE ORDERING INFORMATION AND TRANSFER OF CUSTOMER CODE .....	106
9.2 ST7 APPLICATION NOTES .....	108
9.3 TO GET MORE INFORMATION .....	108
<b>10 SUMMARY OF CHANGES .....</b>	<b>109</b>



## 1 GENERAL DESCRIPTION

### 1.1 INTRODUCTION

The ST7263 Microcontrollers form a sub family of the ST7 dedicated to USB applications. The devices are based on an industry-standard 8-bit core and feature an enhanced instruction set. They operate at a 24MHz or 12 MHz oscillator frequency. Under software control, the ST7263 MCUs may be placed in either Wait or Halt modes, thus reducing power consumption. The enhanced instruction set and addressing modes afford real programming potential. In addition to standard 8-bit data management, the ST7263 MCUs feature true bit manipulation, 8x8 unsigned multiplication and indirect addressing modes. The devices include an ST7 Core, up to 16K program memory, up to 512 bytes RAM, 19 I/O lines and the following on-chip peripherals:

- USB low speed interface with 3 endpoints with programmable in/out configuration using the DMA architecture with embedded 3.3V voltage regulator and transceivers (no external components are needed).
- 8-bit Analog-to-Digital converter (ADC) with 8 multiplexed analog inputs

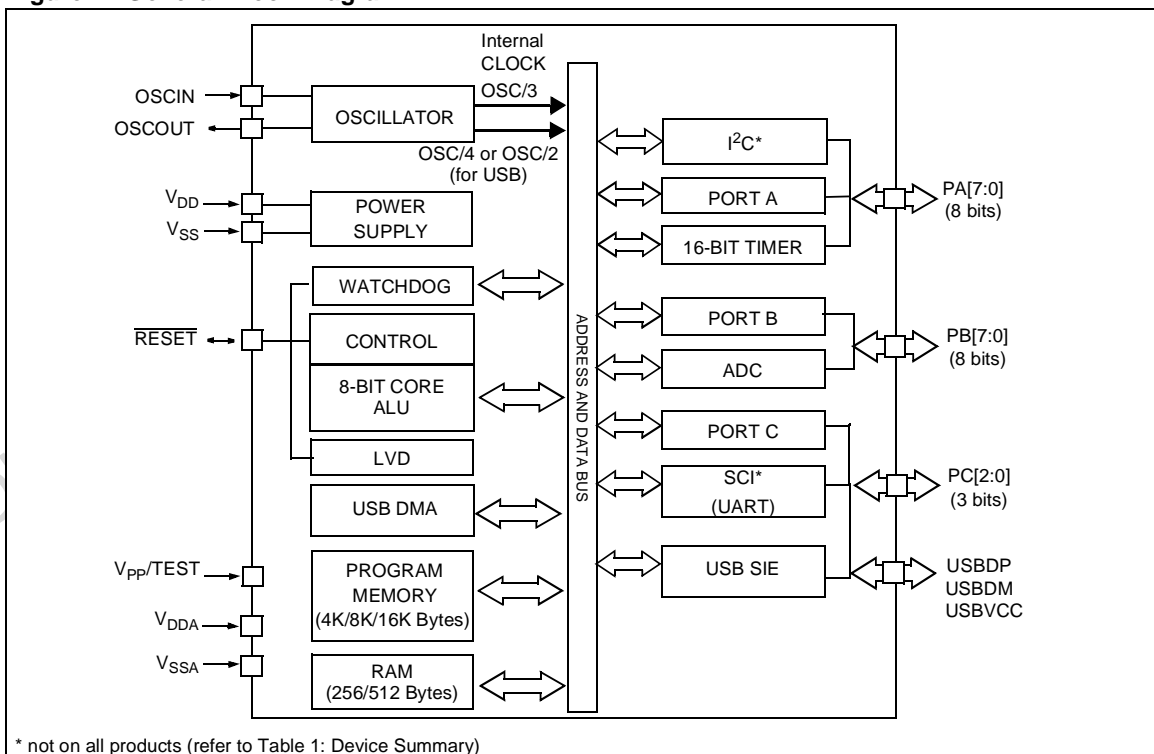
- industry standard asynchronous SCI serial interface (not on all products - see device summary below)
- digital Watchdog
- 16-bit Timer featuring an External clock input, 2 Input Captures, 2 Output Compares with Pulse Generator capabilities
- fast I2C Multi Master interface (not on all products - see device summary)
- Low voltage (LVD) reset ensuring proper power-on or power-off of the device

All ST7263 MCUs are available in ROM and OTP versions.

The ST72E631 is the EPROM version of the ST7263 in CSDIP32 windowed packages.

A specific mode is available to allow programming of the EPROM user memory array. This is set by a specific voltage source applied to the V<sub>PP</sub>/TEST pin.

**Figure 1. General Block Diagram**



## 1.2 PIN DESCRIPTION

Figure 2. 34-Pin SO Package Pinout

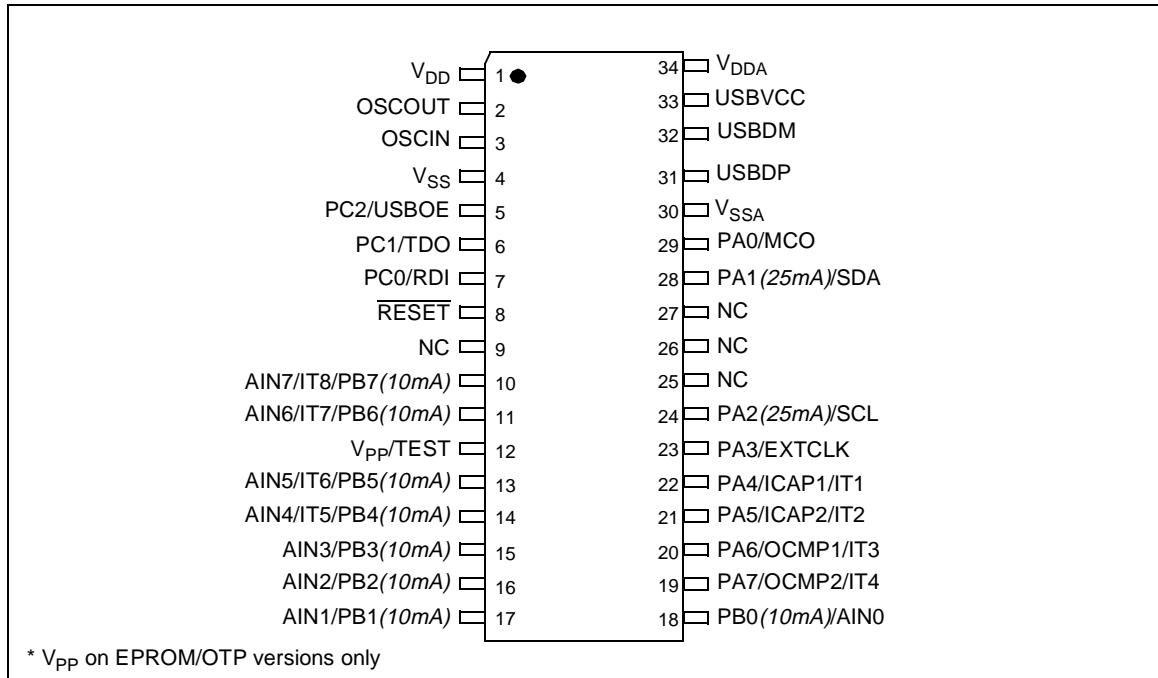
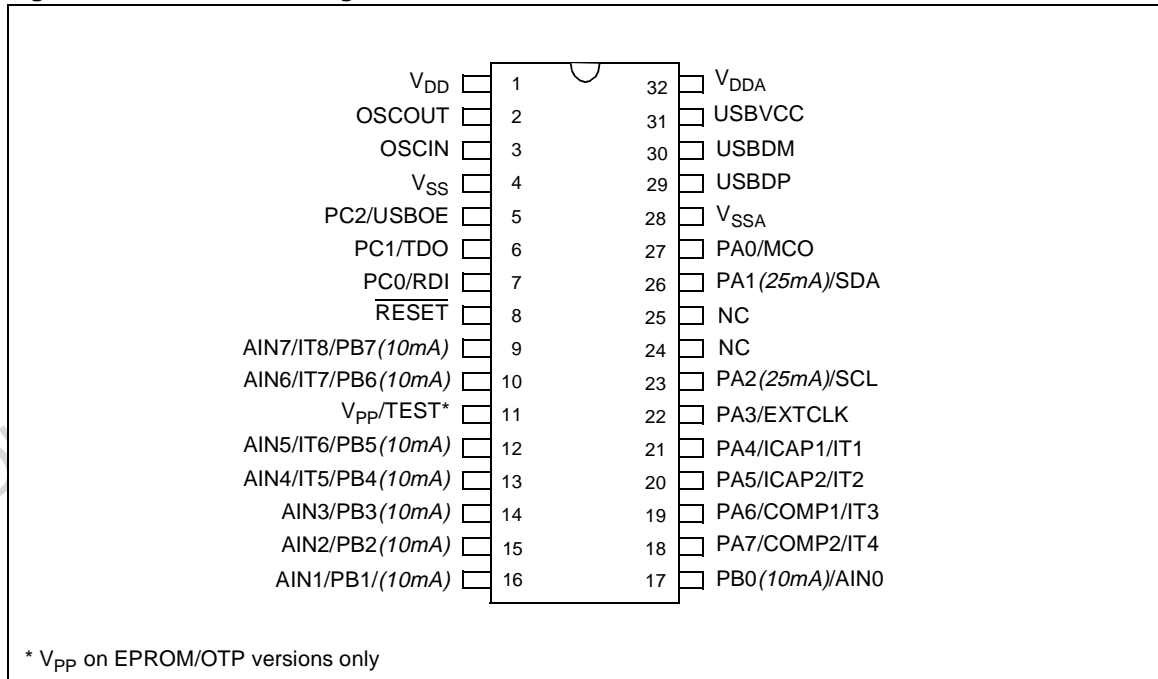


Figure 3. 32-Pin SDIP Package Pinout



**PIN DESCRIPTION** (Cont'd)

**RESET** (see Note 1): Bidirectional. This active low signal forces the initialization of the MCU. This event is the top priority non maskable interrupt. This pin is switched low when the Watchdog has triggered or  $V_{DD}$  is low. It can be used to reset external peripherals.

**OSCIN/OSCOUT**: Input/Output Oscillator pin. These pins connect a parallel-resonant crystal, or an external source to the on-chip oscillator.

**V<sub>PP</sub>/TEST**: EPROM programming input. This pin must be held low during normal operating modes.

**V<sub>DD</sub>/V<sub>SS</sub>** (see Note 2): Main power supply and Ground voltages.

**V<sub>DDA</sub>/V<sub>SSA</sub>** (see Note 2): Power Supply and Ground for analog peripherals.

**Alternate Functions**: Several pins of the I/O ports assume software programmable alternate functions as shown in the pin description.

**Note 1**: Adding two 100nF decoupling capacitors on Reset pin (respectively connected to  $V_{DD}$  and  $V_{SS}$ ) will significantly improve product electromagnetic susceptibility performances.

**Note 2**: To enhance reliability of operation, it is recommended to connect  $V_{DDA}$  and  $V_{DD}$  together on the application board. The same recommendations apply to  $V_{SSA}$  and  $V_{SS}$ .

**Table 2. Device Pin Description**

Pin n°		Pin Name	Type	Level		Port / Control						Main Function (after reset)	Alternate Function
SDIP32	SO34			Input	Output	Input				Output			
						float	wpu	int	ana	OD	PP		
1	1	V <sub>DD</sub>	S										Power supply voltage (4V - 5.5V)
2	2	OSCOUT	O										Oscillator output
3	3	OSCIN	I										Oscillator input
4	4	V <sub>SS</sub>	S										Digital ground
5	5	PC2/USBOE	I/O	C <sub>T</sub>		X				X		<b>Port C2</b>	USB Output Enable
6	6	PC1/TDO	I/O	C <sub>T</sub>		X				X		<b>Port C1</b>	SCI transmit data output <sup>*)</sup>
7	7	PC0/RDI	I/O	C <sub>T</sub>		X				X		<b>Port C0</b>	SCI Receive Data Input <sup>*)</sup>
8	8	RESET	I/O			X				X			Reset
--	9	NC	--										Not connected
9	10	PB7/AIN7/IT8	I/O	C <sub>T</sub>	10mA	X		X	X		X	<b>Port B7</b>	ADC analog input 7
10	11	PB6/AIN6/IT7	I/O	C <sub>T</sub>	10mA	X		X	X		X	<b>Port B6</b>	ADC analog input 6
11	12	V <sub>PP</sub> /TEST	S										Supply for EPROM and test input
12	13	PB5/AIN5/IT6	I/O	C <sub>T</sub>	10mA	X		X	X		X	<b>Port B5</b>	ADC analog input 5
13	14	PB4/AIN4/IT5	I/O	C <sub>T</sub>	10mA	X		X	X		X	<b>Port B4</b>	ADC analog input 4
14	15	PB3/AIN3	I/O	C <sub>T</sub>	10mA	X			X		X	<b>Port B3</b>	ADC analog input 3
15	16	PB2/AIN2	I/O	C <sub>T</sub>	10mA	X			X		X	<b>Port B2</b>	ADC analog input 2
16	17	PB1/AIN1	I/O	C <sub>T</sub>	10mA	X			X		X	<b>Port B1</b>	ADC analog input 1
17	18	PB0/AIN0	I/O	C <sub>T</sub>	10mA	X			X		X	<b>Port B0</b>	ADC Analog Input 0
18	19	PA7/OCMP2/IT4	I/O	C <sub>T</sub>		X	X				X	<b>Port A7</b>	Timer Output Compare 2
19	20	PA6/OCMP1/IT3	I/O	C <sub>T</sub>		X	X				X	<b>Port A6</b>	Timer Output Compare 1

Pin n°		Pin Name	Type	Level		Port / Control						Main Function (after reset)	Alternate Function
SDIP32	SO34			Input	Output	Input				Output			
						float	wpu	int	ana	OD	PP		
20	21	PA5/ICAP2/IT2	I/O	C <sub>T</sub>		X	X				X	<b>Port A5</b>	Timer Input Capture 2
21	22	PA4/ICAP1/IT1	I/O	C <sub>T</sub>		X	X				X	<b>Port A4</b>	Timer Input Capture 1
22	23	PA3/EXTCLK	I/O	C <sub>T</sub>		X					X	<b>Port A3</b>	Timer External Clock
23	24	PA2/SCL	I/O	C <sub>T</sub>	25mA	X				T		<b>Port A2</b>	I <sup>2</sup> C serial clock <sup>*)</sup>
--	25	NC	--										Not connected
24	26	NC	--										Not connected
25	27	NC	--										Not connected
26	28	PA1/SDA	I/O	C <sub>T</sub>	25mA	X				T		<b>Port A1</b>	I <sup>2</sup> C serial data <sup>*)</sup>
27	29	PA0/MCO	I/O	C <sub>T</sub>				X			X	<b>Port A0</b>	Main Clock Output
28	30	V <sub>SSA</sub>	S										Analog ground
29	31	USBDP	I/O										USB bidirectional data (data +)
30	32	USBDM	I/O										USB bidirectional data (data -)
31	33	USBVCC	O										USB power supply
32	34	V <sub>DDA</sub>	S										Analog supply voltage

\*: if the peripheral is present on the device (see Table 1 Device Summary)

#### Legend / Abbreviations for Figure 2 and Table 2:

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

In/Output level: C<sub>T</sub> = CMOS 0.3V<sub>DD</sub>/0.7V<sub>DD</sub> with input trigger

Output level: 10mA = 10mA high sink (on N-buffer only)

25mA = 25mA very 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, T = True open drain

Refer to "I/O PORTS" on page 25 for more details on the software configuration of the I/O ports.

The RESET configuration of each pin is shown in bold. This configuration is kept as long as the device is under reset state.





### 1.3 EXTERNAL CONNECTIONS

The following figure shows the recommended external connections for the device.

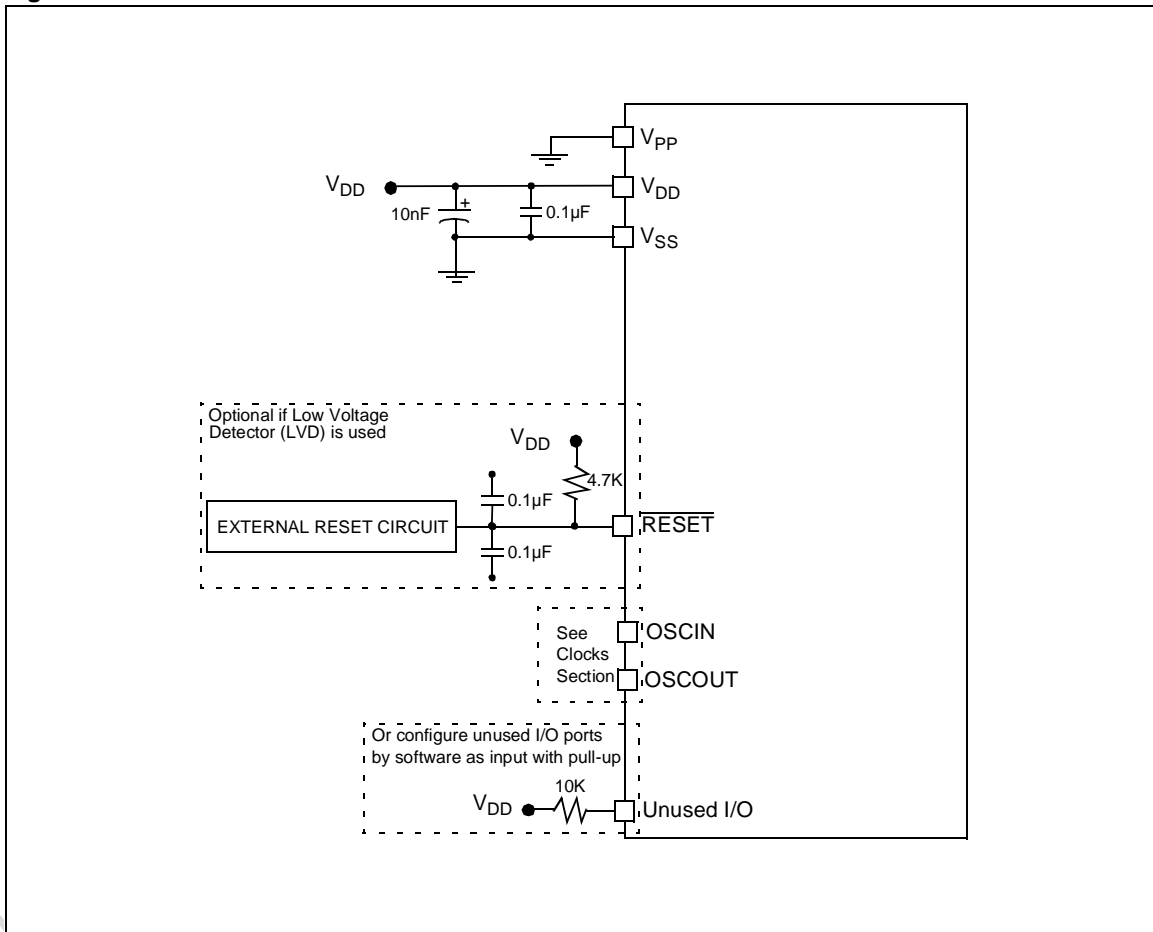
The  $V_{PP}$  pin is only used for programming OTP and EPROM devices and must be tied to ground in user mode.

The 10 nF and 0.1  $\mu$ F decoupling capacitors on the power supply lines are a suggested EMC performance/cost tradeoff.

The external reset network is intended to protect the device against parasitic resets, especially in noisy environments.

Unused I/Os should be tied high to avoid any unnecessary power consumption on floating lines. An alternative solution is to program the unused ports as inputs with pull-up.

**Figure 4. Recommended External Connections**



**1.4 REGISTER & MEMORY MAP**

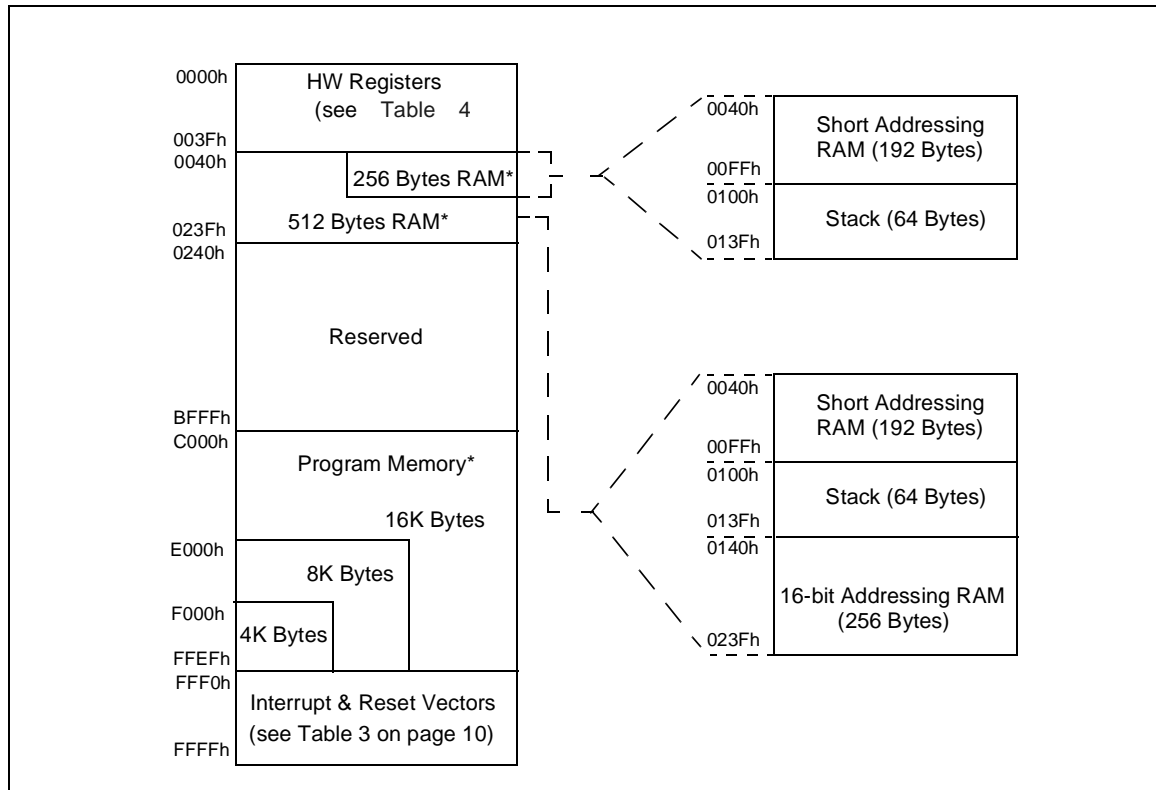
As shown in Figure 5, the MCU is capable of addressing 64K bytes of memories and I/O registers.

The available memory locations consist of 192 bytes of register location, up to 512 bytes of RAM and up to 16K bytes of user program memory. The RAM space includes up to 64 bytes for the stack from 0100h to 013Fh.

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

**IMPORTANT:** Memory locations noted “Reserved” must never be accessed. Accessing a reserved area can have unpredictable effects on the device

**Figure 5. Memory Map**



\* Program memory and RAM sizes are product dependent (see Table 1 Device Summary)

**Table 3. Interrupt Vector Map**

Vector Address	Description	Masked by	Remarks	Exit from Halt Mode
FFF0-FFF1h	USB Interrupt Vector	I- bit	Internal Interrupt	No
FFF2-FFF3h	SCI Interrupt Vector*	I- bit	Internal Interrupt	No
FFF4-FFF5h	I <sup>2</sup> C Interrupt Vector*	I- bit	Internal Interrupt	No
FFF6-FFF7h	TIMER Interrupt Vector	I- bit	Internal Interrupt	No
FFF8-FFF9h	IT1 to IT8 Interrupt Vector	I- bit	External Interrupts	Yes
FFFA-FFFBh	USB End Suspend Mode Interrupt Vector	I- bit	Internal Interrupt	Yes
FFFC-FFFDh	TRAP (software) Interrupt Vector	none	CPU Interrupt	No
FFFE-FFFFh	RESET Vector	none		Yes

\* If the peripheral is present on the device (see Table 1 Device Summary)

Table 4. Hardware Register Memory Map

Address	Block	Register Label	Register name	Reset Status	Remarks
0000h		PADR	Port A Data Register	00h	R/W
0001h		PADDR	Port A Data Direction Register	00h	R/W
0002h		PBDR	Port B Data Register	00h	R/W
0003h		PBDDR	Port B Data Direction Register	00h	R/W
0004h		PCDR	Port C Data Register	1111 x000b	R/W
0005h		PCDDR	Port C Data Direction Register	1111 x000b	R/W
0006h 0007h	Reserved (2 Bytes)				
0008h		ITIFRE	Interrupt Register	00h	R/W
0009h		MISCR	Miscellaneous Register	F0h	R/W
000Ah	ADC	DR	ADC Data Register	00h	Read only
000Bh		CSR	ADC control Status register	00h	R/W
000Ch	WDG	CR	Watchdog Control Register	7Fh	R/W
000Dh 0010h	Reserved (4 Bytes)				
0011h	TIM	CR2	Timer Control Register 2	00h	R/W
0012h		CR1	Timer Control Register 1	00h	R/W
0013h		SR	Timer Status Register	00h	Read only
0014h		IC1HR	Timer Input Capture High Register 1	xxh	Read only
0015h		IC1LR	Timer Input Capture Low Register 1	xxh	Read only
0016h		OC1HR	Timer Output Compare High Register 1	80h	R/W
0017h		OC1LR	Timer Output Compare Low Register 1	00h	R/W
0018h		CHR	Timer Counter High Register	FFh	Read only
0019h		CLR	Timer Counter Low Register	FCh	R/W
001Ah		ACHR	Timer Alternate Counter High Register	FFh	Read only
001Bh		ACLRL	Timer Alternate Counter Low Register	FCh	R/W
001Ch		IC2HR	Timer Input Capture High Register 2	xxh	Read only
001Dh		IC2LR	Timer Input Capture Low Register 2	xxh	Read only
001Eh		OC2HR	Timer Output Compare High Register 2	80h	R/W
001Fh		OC2LR	Timer Output Compare Low Register 2	00h	R/W
0020h	SCI <sup>1)</sup>	SR	SCI Status Register	C0h	Read only
0021h		DR	SCI Data Register	xxh	R/W
0022h		BRR	SCI Baud Rate Register	00xx xxxxb	R/W
0023h		CR1	SCI Control Register 1	xxh	R/W
0024h		CR2	SCI Control Register 2	00h	R/W

Address	Block	Register Label	Register name	Reset Status	Remarks
0025h	USB	PIDR	USB PID Register	xxh	Read only
0026h		DMAR	USB DMA address Register	xxh	R/W
0027h		IDR	USB Interrupt/DMA Register	xxh	R/W
0028h		ISTR	USB Interrupt Status Register	00h	R/W
0029h		IMR	USB Interrupt Mask Register	00h	R/W
002Ah		CTLR	USB Control Register	xxxx 0110b	R/W
002Bh		DADDR	USB Device Address Register	00h	R/W
002Ch		EP0RA	USB Endpoint 0 Register A	0000 xxxxb	R/W
002Dh		EP0RB	USB Endpoint 0 Register B	80h	R/W
002Eh		EP1RA	USB Endpoint 1 Register A	0000 xxxxb	R/W
002Fh		EP1RB	USB Endpoint 1 Register B	0000 xxxxb	R/W
0030h		EP2RA	USB Endpoint 2 Register A	0000 xxxxb	R/W
0031h		EP2RB	USB Endpoint 2 Register B	0000 xxxxb	R/W
0032h 0038h		Reserved (7 Bytes)			
0039h	I <sup>2</sup> C <sup>1)</sup>	DR	I <sup>2</sup> C Data Register	00h	R/W
003Ah		Reserved	-		
003Bh		OAR	I2C (7 Bits) Slave Address Register	00h	R/W
003Ch		CCR	I <sup>2</sup> C Clock Control Register	00h	R/W
003Dh		SR2	I <sup>2</sup> C 2nd Status Register	00h	Read only
003Eh		SR1	I <sup>2</sup> C 1st Status Register	00h	Read only
003Fh		CR	I <sup>2</sup> C Control Register	00h	R/W

Note 1. If the peripheral is present on the device (see Table 1 Device Summary)

## 1.5 EPROM/OTP PROGRAM MEMORY

The program memory of the ST72T63 may be programmed using the EPROM programming boards available from STMicroelectronics (see Table 26).

### 1.5.1 EPROM ERASURE

ST72Exxx EPROM devices are erased by exposure to high intensity UV light admitted through the transparent window. This exposure discharges the floating gate to its initial state through induced photo current.

It is recommended that the ST72Exxx devices be kept out of direct sunlight, since the UV content of sunlight can be sufficient to cause functional failure. Extended exposure to room level fluorescent lighting may also cause erasure.

An opaque coating (paint, tape, label, etc...) should be placed over the package window if the product is to be operated under these lighting conditions. Covering the window also reduces  $I_{DD}$  in power-saving modes due to photo-diode leakage currents.

An Ultraviolet source of wave length 2537 Å yielding a total integrated dosage of 15 Watt-sec/cm<sup>2</sup> is required to erase the ST72Exxx. The device will be erased in 15 to 30 minutes if such a UV lamp with a 12mW/cm<sup>2</sup> power rating is placed 1 inch from the device window without any interposed filters.



## 2 CENTRAL PROCESSING UNIT

### 2.1 INTRODUCTION

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

### 2.2 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

### 2.3 CPU REGISTERS

The 6 CPU registers shown in Figure 1 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)

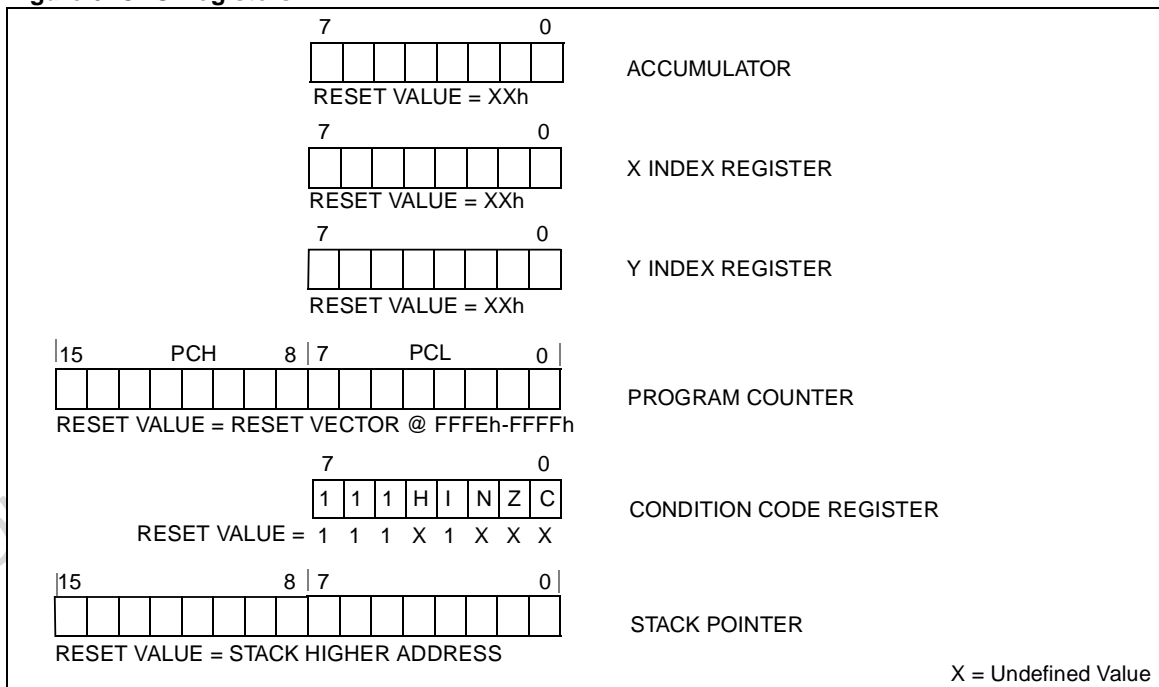
In indexed addressing modes, these 8-bit registers are used to create either effective addresses or temporary storage areas for data manipulation. (The Cross-Assembler generates a precede instruction (PRE) to indicate that the following instruction refers to the Y register.)

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 6. 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.

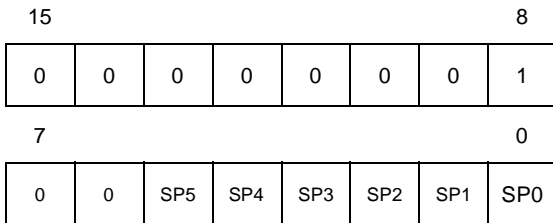
This bit is driven by the SCF and RCF instructions and tested by the JRC and JRNC instructions. It is also affected by the "bit test and branch", shift and rotate instructions.

**CPU REGISTERS** (Cont'd)

**Stack Pointer (SP)**

Read/Write

Reset Value: 01 3Fh



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 7).

Since the stack is 64 bytes deep, the 10 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 (SP5 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.

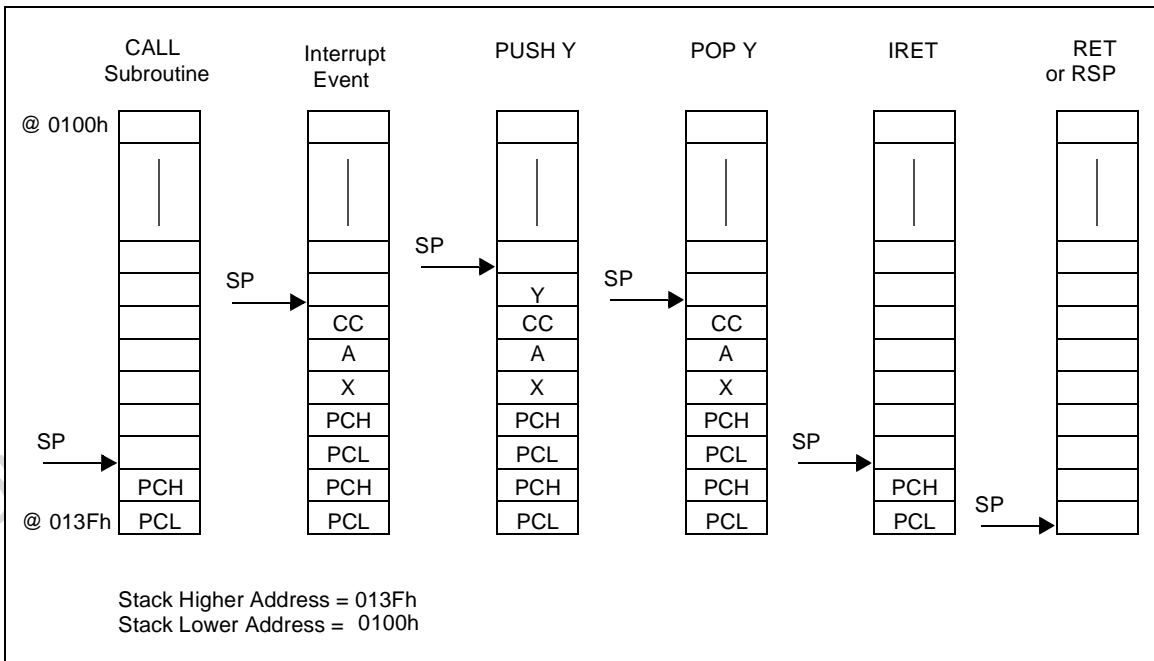
**Note:** When the lower limit is exceeded, the Stack Pointer wraps around to the stack upper limit, without indicating the stack overflow. The previously stored information is then overwritten and therefore lost. The stack also wraps in case of an underflow.

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 7.

- 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 7. Stack Manipulation Example**





## 3 CLOCKS AND RESET

### 3.1 CLOCK SYSTEM

#### 3.1.1 General Description

The MCU accepts either a Crystal or Ceramic resonator, or an external clock signal to drive the internal oscillator. The internal clock ( $f_{CPU}$ ) is derived from the external oscillator frequency ( $f_{OSC}$ ), which is divided by 3 (and by 2 or 4 for USB, depending on the external clock used).

By setting the CLKDIV bit in the Miscellaneous Register, a 12 MHz external clock can be used giving an internal frequency of 4 MHz while maintaining a 6 MHz for USB (refer to Figure 10).

The internal clock signal ( $f_{CPU}$ ) is also routed to the on-chip peripherals. The CPU clock signal consists of a square wave with a duty cycle of 50%.

The internal oscillator is designed to operate with an AT-cut parallel resonant quartz or ceramic resonator in the frequency range specified for  $f_{OSC}$ . The circuit shown in Figure 9 is recommended when using a crystal, and Table 5 Recommended Values for 24 MHz Crystal Resonator lists the recommended capacitance. The crystal and associated components should be mounted as close as possible to the input pins in order to minimize output distortion and start-up stabilisation time.

**Table 5. Recommended Values for 24 MHz Crystal Resonator**

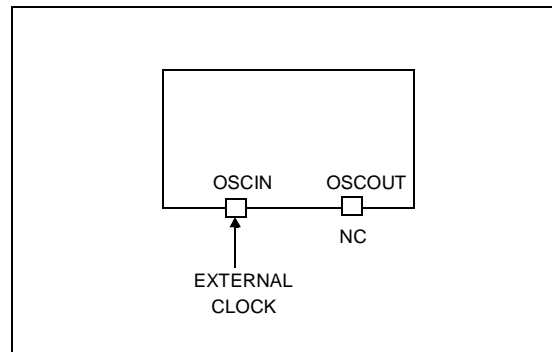
$R_{SMAX}$	20 $\Omega$	25 $\Omega$	70 $\Omega$
$C_{OSCIN}$	56pF	47pF	22pF
$C_{OSCOUT}$	56pF	47pF	22pF
$R_p$	1-10 M $\Omega$	1-10 M $\Omega$	1-10 M $\Omega$

**Note:**  $R_{SMAX}$  is the equivalent serial resistor of the crystal (see crystal specification).

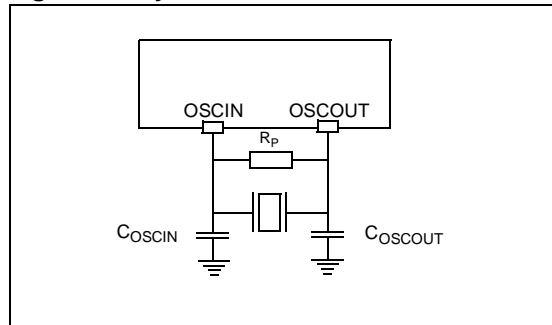
#### 3.1.2 External Clock

An external clock may be applied to the OSCIN input with the OSCOUT pin not connected, as shown on Figure 8. The  $t_{OXOV}$  specifications does not apply when using an external clock input. The equivalent specification of the external clock source should be used instead of  $t_{OXOV}$  (see Section 6.5 CONTROL TIMING).

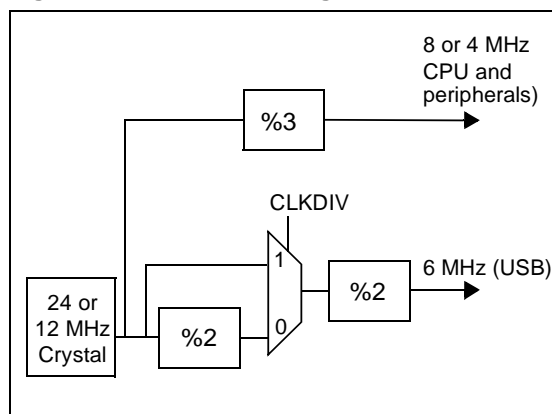
**Figure 8. External Clock Source Connections**



**Figure 9. Crystal/Ceramic Resonator**



**Figure 10. Clock block diagram**



### 3.2 RESET

The Reset procedure is used to provide an orderly software start-up or to exit low power modes.

Three reset modes are provided: a low voltage (LVD) reset, a watchdog reset and an external reset at the  $\overline{\text{RESET}}$  pin.

A reset causes the reset vector to be fetched from addresses FFFEh and FFFFh in order to be loaded into the PC and with program execution starting from this point.

An internal circuitry provides a 4096 CPU clock cycle delay from the time that the oscillator becomes active.

#### 3.2.1 Low Voltage Detector (LVD)

Low voltage reset circuitry generates a reset when  $V_{DD}$  is:

- below  $V_{IT+}$  when  $V_{DD}$  is rising,
- below  $V_{IT-}$  when  $V_{DD}$  is falling.

During low voltage reset, the  $\overline{\text{RESET}}$  pin is held low, thus permitting the MCU to reset other devices.

The Low Voltage Detector can be disabled by setting the LVD bit of the Miscellaneous Register.

#### 3.2.2 Watchdog Reset

When a watchdog reset occurs, the  $\overline{\text{RESET}}$  pin is pulled low permitting the MCU to reset other devices in the same way as the low voltage reset (Figure 11).

#### 3.2.3 External Reset

The external reset is an active low input signal applied to the  $\overline{\text{RESET}}$  pin of the MCU.

As shown in Figure 14, the  $\overline{\text{RESET}}$  signal must stay low for a minimum of one and a half CPU clock cycles.

An internal Schmitt trigger at the  $\overline{\text{RESET}}$  pin is provided to improve noise immunity.

**Table 6. List of sections affected by RESET, WAIT and HALT (Refer to 3.5 for Wait and Halt Modes)**

Section	RESET	WAIT	HALT
CPU clock running at 8 MHz	X		
Timer Prescaler reset to zero	X		
Timer Counter set to FFFCh	X		
All Timer enable bit set to 0 (disable)	X		
Data Direction Registers set to 0 (as Inputs)	X		
Set Stack Pointer to 013Fh	X		
Force Internal Address Bus to restart vector FFFEh,FFFFh	X		
Set Interrupt Mask Bit (I-Bit, CCR) to 1 (Interrupt Disable)	X		
Set Interrupt Mask Bit (I-Bit, CCR) to 0 (Interrupt Enable)		X	X
Reset HALT latch	X		
Reset WAIT latch	X		
Disable Oscillator (for 4096 cycles)	X		X
Set Timer Clock to 0	X		X
Watchdog counter reset	X		
Watchdog register reset	X		
Port data registers reset	X		
Other on-chip peripherals: registers reset	X		

Figure 11. Low Voltage Detector functional Diagram

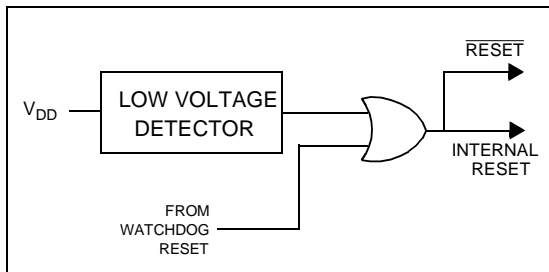
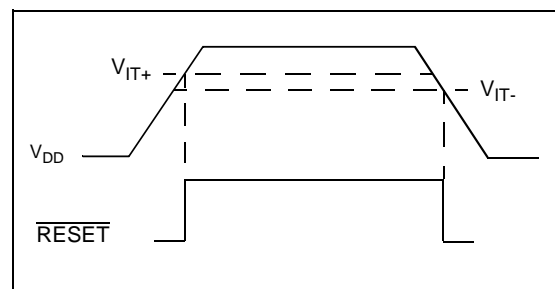


Figure 12. Low Voltage Reset Signal Output



Note: Hysteresis ( $V_{IT+} - V_{IT-}$ ) =  $V_{hys}$

Figure 13. Temporization timing diagram after an internal Reset

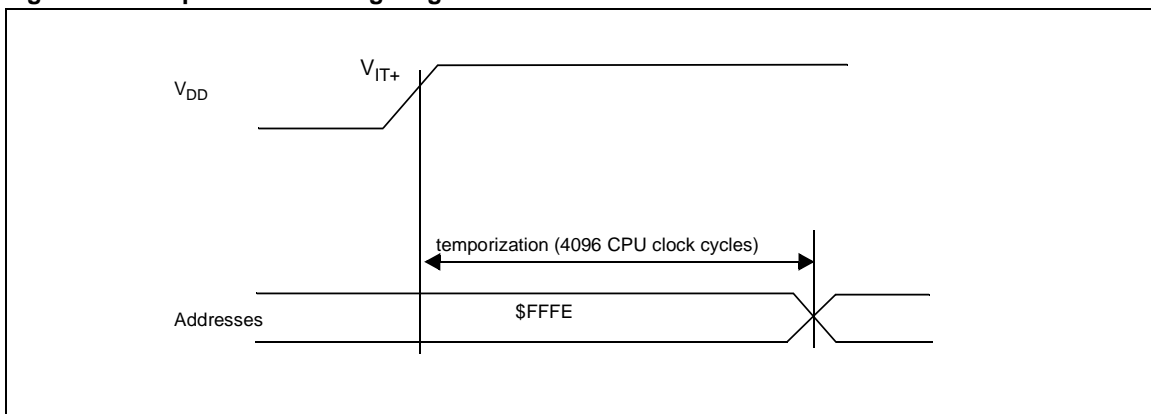
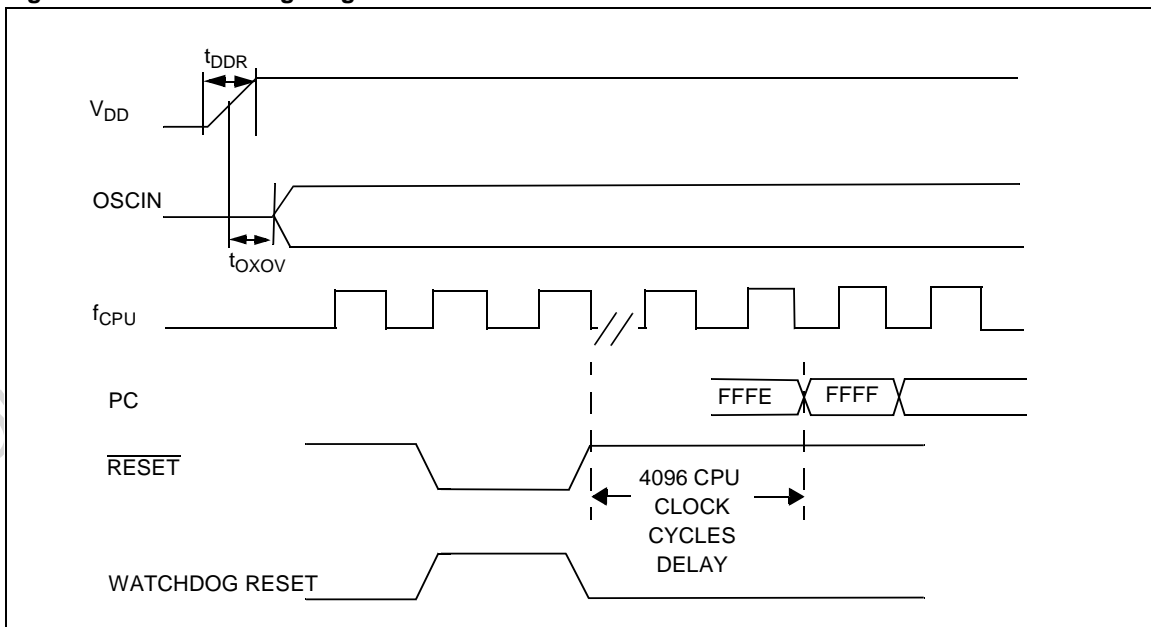


Figure 14. Reset Timing Diagram



Note: Refer to Electrical Characteristics for values of  $t_{DDR}$ ,  $t_{OXOV}$ ,  $V_{IT+}$ ,  $V_{IT-}$  and  $V_{hys}$

## 4 INTERRUPTS AND POWER SAVING MODES

### 4.1 INTERRUPTS

The ST7 core may be interrupted by one of two different methods: maskable hardware interrupts as listed in Table 7 Interrupt Mapping and a non-maskable software interrupt (TRAP). The Interrupt processing flowchart is shown in Figure 15.

The maskable interrupts must be enabled 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).

When an interrupt has to be serviced:

- Normal processing is suspended at the end of the current instruction execution.
- The PC, X, A and CC registers are saved onto the stack.
- The I bit of the CC register is set to prevent additional interrupts.
- The PC is then loaded with the interrupt vector of the interrupt to service and the first instruction of the interrupt service routine is fetched (refer to Table 7 Interrupt Mapping for vector addresses).

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

**Note:** As a consequence of the IRET instruction, the I bit will be cleared and the main program will resume.

#### Priority management

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

In the case several interrupts are simultaneously pending, a hardware priority defines which one will be serviced first (see Table 7 Interrupt Mapping).

#### Non maskable software interrupts

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 15.

#### Interrupts and Low power mode

All interrupts allow the processor to leave the Wait low power mode. Only external and specific mentioned interrupts allow the processor to leave the Halt low power mode (refer to the “Exit from HALT” column in Table 7 Interrupt Mapping).

#### External interrupts

The pins ITi/PAk and ITj/PBk (i=1,2; j= 5,6; k=4,5) can generate an interrupt when a rising edge occurs on this pin. Conversely, pins ITi/PAn and ITm/PBn (l=3,4; m= 7,8; n=6,7) can generate an interrupt when a falling edge occurs on this pin.

Interrupt generation will occur if it is enabled with the ITiE bit (i=1 to 8) in the ITRFRE register and if the I bit of the CCR is reset.

#### 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
- an access to the status register while the flag is set followed by a read or write of an associated register.

#### Notes:

1. 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.
2. All interrupts allow the processor to leave the Wait low power mode.
3. Exit from Halt mode may only be triggered by an External Interrupt on one of the ITi ports (PA4-PA7 and PB4-PB7), an end suspend mode Interrupt coming from USB peripheral, or a reset.

## INTERRUPTS (Cont'd)

Figure 15. Interrupt Processing Flowchart

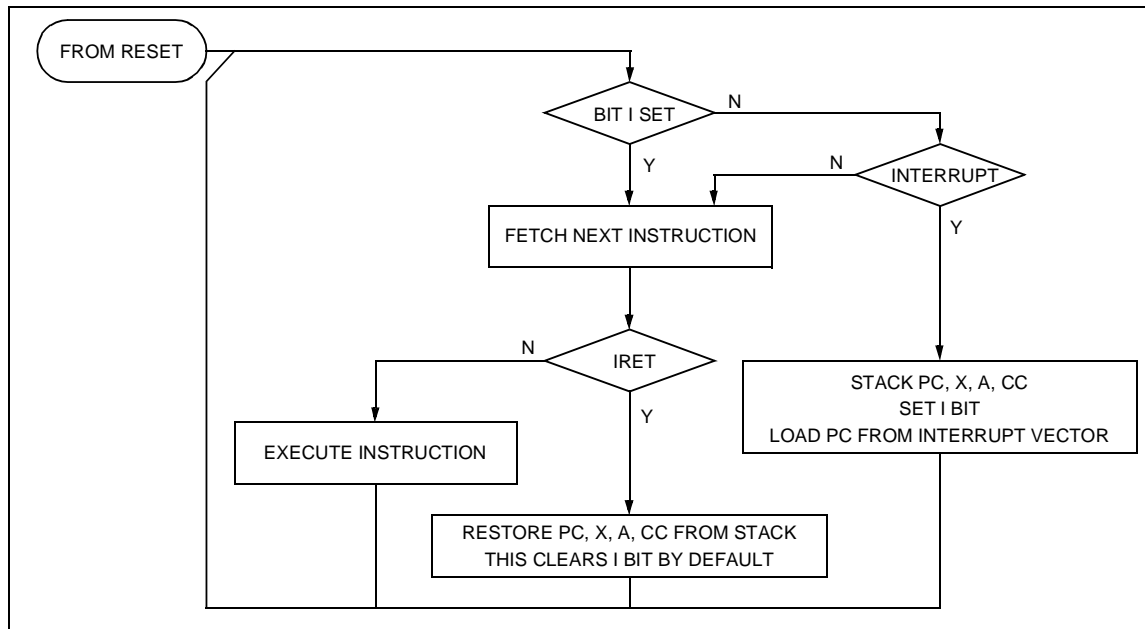


Table 7. Interrupt Mapping

N°	Source Block	Description	Register Label	Priority Order	Exit from HALT	Vector Address
	RESET	Reset	N/A	Highest Priority	yes	FFFEh-FFFFh
	TRAP	Software Interrupt			no	FFFCh-FFFDh
	USB	End Suspend Mode	ISTR	↓ Lowest Priority	yes	FFFAh-FFFBh
1	ITi	External Interrupts	ITRFRE		no	FFF8h-FFF9h
2	TIMER	Timer Peripheral Interrupts	TIMSR		no	FFF6h-FFF7h
3	I <sup>2</sup> C	I <sup>2</sup> C Peripheral Interrupts	I2CSR1		no	FFF4h-FFF5h
			I2CSR2		no	FFF2h-FFF3h
4	SCI	SCI Peripheral Interrupts	SCISR	no	FFF0h-FFF1h	
5	USB	USB Peripheral Interrupts	ISTR	no	FFF0h-FFF1h	

**INTERRUPTS** (Cont'd)**4.1.1 Interrupt Register****INTERRUPTS REGISTER (ITRFRE)**

Address: 0008h — Read/Write

Reset Value: 0000 0000 (00h)

7								0
IT8E	IT7E	IT6E	IT5E	IT4E	IT3E	IT2E	IT1E	

Bit 7:0 = **ITiE (i=1 to 8)**. *Interrupt Enable Control Bits.*

If an ITiE bit is set, the corresponding interrupt is generated when

– a rising edge occurs on the pin PA4/IT1 or PA5/IT2 or PB4/IT5 or PB5/IT6

or

– a falling edge occurs on the pin PA6/IT3 or PA7/IT4 or PB6/IT7 or PB7/IT8

No interrupt is generated elsewhere.

**Note:** Analog input must be disabled for interrupts coming from port B.

## 4.2 POWER SAVING MODES

### 4.2.1 Introduction

To give a large measure of flexibility to the application in terms of power consumption, two main power saving modes are implemented in the ST7.

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 by 3 ( $f_{CPU}$ ).

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.

### 4.2.2 HALT mode

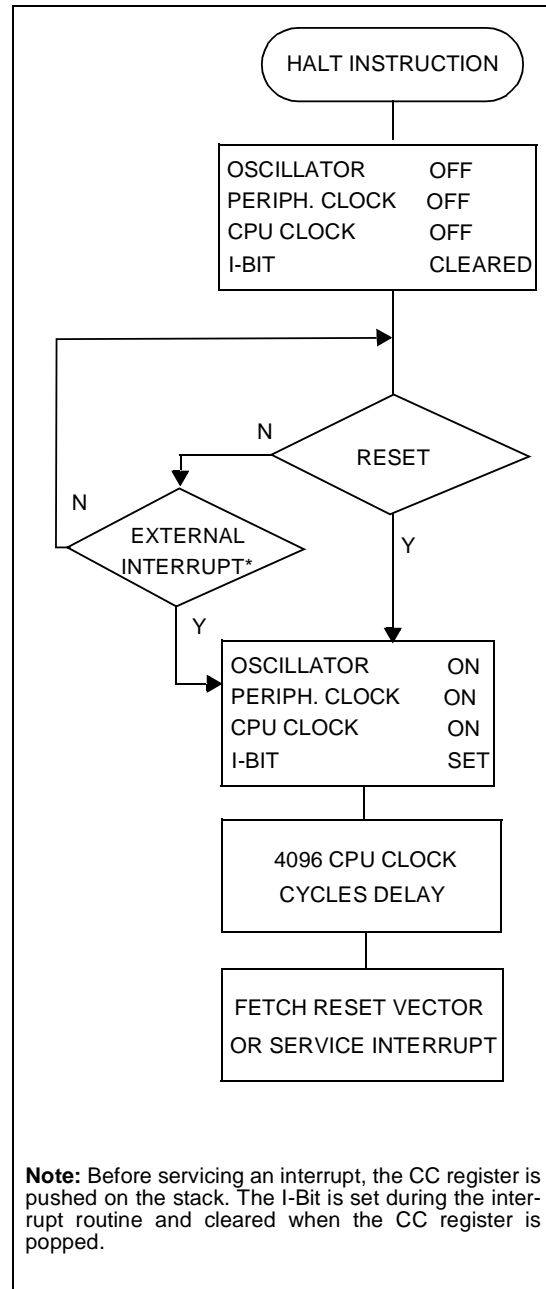
The HALT mode is the MCU lowest power consumption mode. The HALT mode is entered by executing the HALT instruction. The internal oscillator is then turned off, causing all internal processing to be stopped, including the operation of the on-chip peripherals.

When entering HALT mode, the I bit in the Condition Code Register is cleared. Thus, any of the external interrupts (ITi or USB end suspend mode), are allowed and if an interrupt occurs, the CPU clock becomes active.

The MCU can exit HALT mode on reception of either an external interrupt on ITi, an end suspend mode interrupt coming from USB peripheral, or a reset. The oscillator is then turned on and a stabilization time is provided before releasing CPU operation. The stabilization time is 4096 CPU clock cycles.

After the start up delay, the CPU continues operation by servicing the interrupt which wakes it up or by fetching the reset vector if a reset wakes it up.

Figure 16. HALT Mode Flow Chart



**POWER SAVING MODES (Cont'd)****4.2.3 WAIT mode**

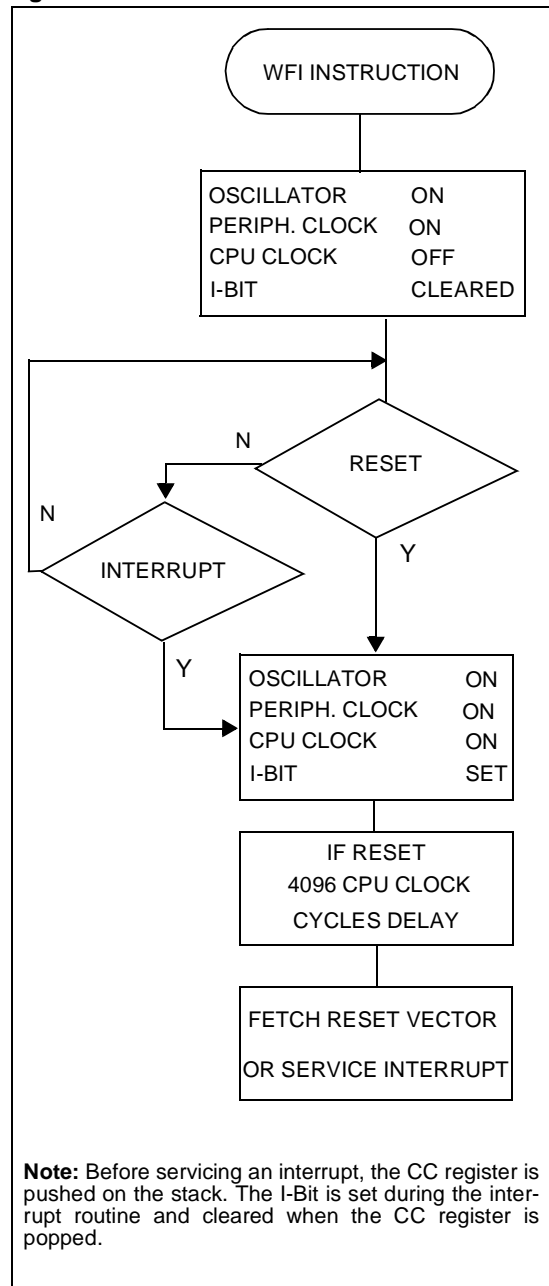
WAIT mode places the MCU in a low power consumption mode by stopping the CPU.

This power saving mode is selected by calling the "WFI" ST7 software instruction.

All peripherals remain active. During WAIT mode, the I bit of the CC register is forced to 0, 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 17.

**Figure 17. WAIT Mode Flow Chart**



## 5 ON-CHIP PERIPHERALS

### 5.1 I/O PORTS

#### 5.1.1 Introduction

The I/O ports offer different functional modes:

- transfer of data through digital inputs and outputs and for specific pins:
- analog signal input (ADC)
- alternate signal input/output for the on-chip peripherals.
- external interrupt generation

An I/O port is composed of up to 8 pins. Each pin can be programmed independently as digital input (with or without interrupt generation) or digital output.

#### 5.1.2 Functional description

Each port is associated to 2 main registers:

- Data Register (DR)
- Data Direction Register (DDR)

Each I/O pin may be programmed using the corresponding register bits in DDR register: bit X corresponding to pin X of the port. The same correspondence is used for the DR register.

**Table 8. I/O Pin Functions**

DDR	MODE
0	Input
1	Output

#### Input Modes

The input configuration is selected by clearing the corresponding DDR register bit.

In this case, reading the DR register returns the digital value applied to the external I/O pin.

**Note 1:** All the inputs are triggered by a Schmitt trigger.

**Note 2:** When switching from input mode to output mode, the DR register should be written first to output the correct value as soon as the port is configured as an output.

#### Interrupt function

When an I/O is configured in Input with Interrupt, an event on this I/O can generate an external In-

terrupt request to the CPU. The interrupt sensitivity is given independently according to the description mentioned in the ITRFRE interrupt register.

Each pin can independently generate an Interrupt request.

Each external interrupt vector is linked to a dedicated group of I/O port pins (see Interrupts section). If more than one input pin is selected simultaneously as interrupt source, this is logically ORed. For this reason if one of the interrupt pins is tied low, it masks the other ones.

#### Output Mode

The pin is configured in output mode by setting the corresponding DDR register bit (see Table 7).

In this mode, writing “0” or “1” to the DR register applies this digital value to the I/O pin through the latch. Then reading the DR register returns the previously stored value.

**Note:** In this mode, the interrupt function is disabled.

#### Digital Alternate Function

When an on-chip peripheral is configured to use a pin, the alternate function is automatically selected. This alternate function takes priority over standard I/O programming. When the signal is coming from an on-chip peripheral, the I/O pin is automatically configured in output mode (push-pull or open drain according to the peripheral).

When the signal is going to an on-chip peripheral, the I/O pin has to be configured in input mode. In this case, the pin’s state is also digitally readable by addressing the DR register.

Notes:

1. Input pull-up configuration can cause an unexpected value at the input of the alternate peripheral input.
2. When the on-chip peripheral uses a pin as input and output, this pin must be configured as an input (DDR = 0).

**Warning:** The alternate function must not be activated as long as the pin is configured as input with interrupt, in order to avoid generating spurious interrupts.

**I/O PORTS** (Cont'd)

**Analog Alternate Function**

When the pin is used as an ADC input the I/O must be configured as input, floating. The analog multiplexer (controlled by the ADC registers) switches the analog voltage present on the selected pin to the common analog rail which is connected to the ADC input.

It is recommended not to change the voltage level or loading on any port pin while conversion is in progress. Furthermore it is recommended not to

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.

**5.1.3 I/O Port Implementation**

The hardware implementation on each I/O port depends on the settings in the DDR register and specific feature of the I/O port such as ADC Input or true open drain.



## I/O PORTS (Cont'd)

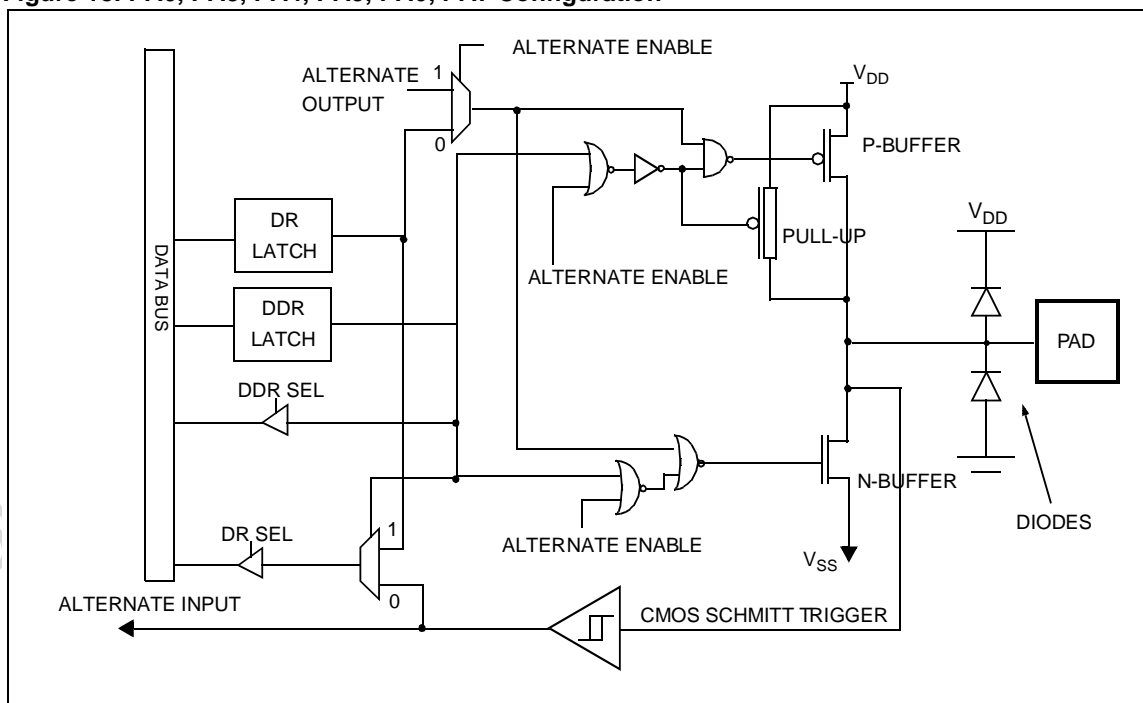
## 5.1.4 Port A

Table 9. Port A0, A3, A4, A5, A6, A7 Description

PORT A	I/O		Alternate Function	
	Input*	Output	Signal	Condition
PA0	with pull-up	push-pull	MCO (Main Clock Output)	MCO = 1 (MISCR)
PA3	with pull-up	push-pull	Timer EXTCLK	CC1 = 1 CC0 = 1 (Timer CR2)
PA4	with pull-up	push-pull	Timer ICAP1	
			IT1 Schmitt triggered input	IT1E = 1 (ITIFRE)
PA5	with pull-up	push-pull	Timer ICAP2	
			IT2 Schmitt triggered input	IT2E = 1 (ITIFRE)
PA6	with pull-up	push-pull	Timer OCMP1	OC1E = 1
			IT3 Schmitt triggered input	IT3E = 1 (ITIFRE)
PA7	with pull-up	push-pull	Timer OCMP2	OC2E = 1
			IT4 Schmitt triggered input	IT4E = 1 (ITIFRE)

\*Reset State

Figure 18. PA0, PA3, PA4, PA5, PA6, PA7 Configuration



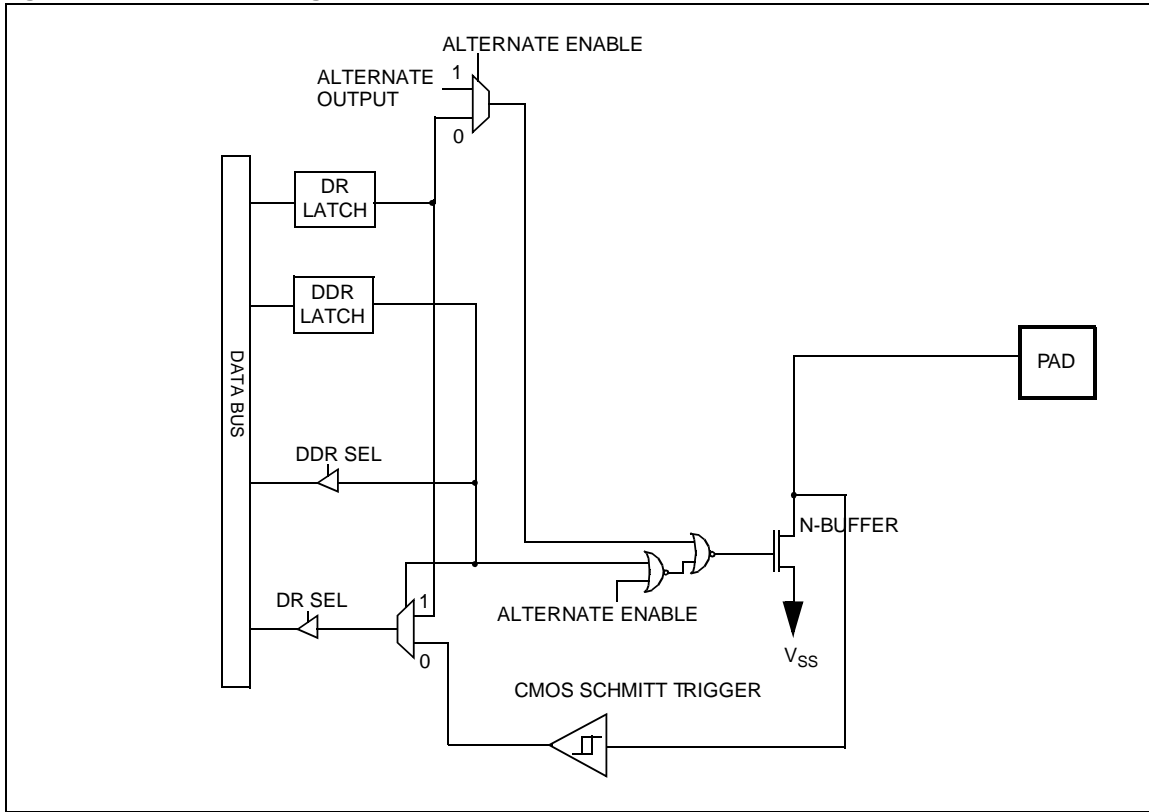
I/O PORTS (Cont'd)

Table 10. PA1, PA2 Description

PORT A	I/O		Alternate Function	
	Input*	Output	Signal	Condition
PA1	without pull-up	Very High Current open drain	SDA (I2C data)	I2C enable
PA2	without pull-up	Very High Current open drain	SCL (I2C clock)	I2C enable

\*Reset State

Figure 19. PA1, PA2 Configuration



## I/O PORTS (Cont'd)

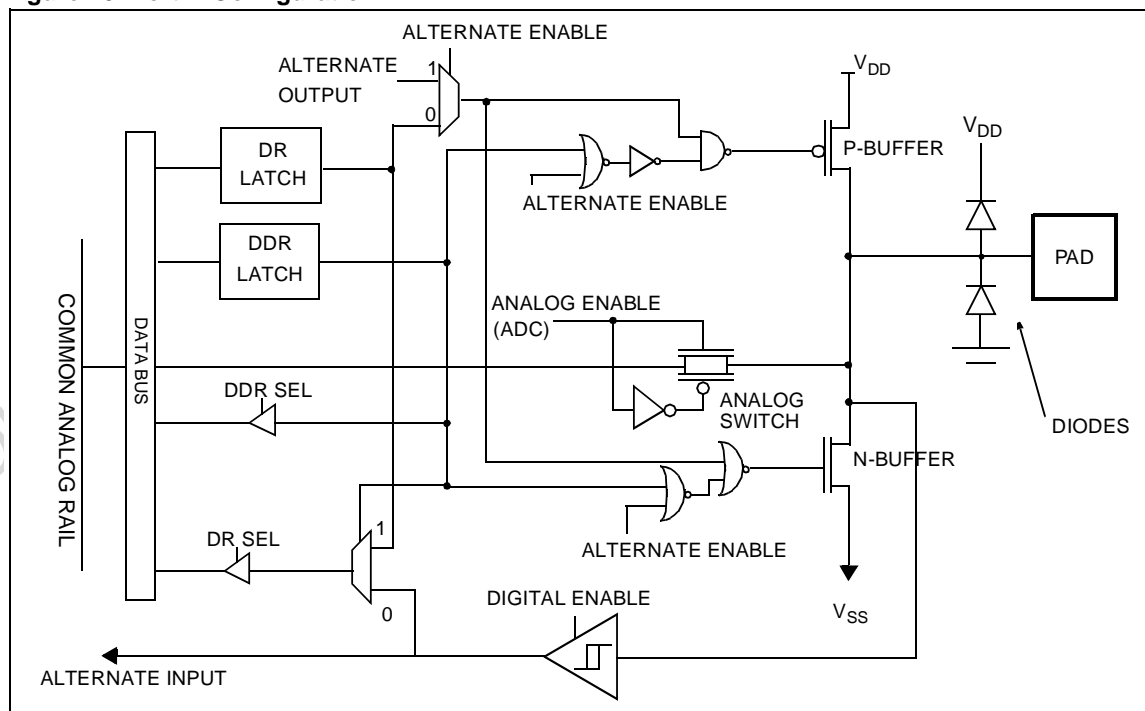
## 5.1.5 Port B

Table 11. Port B Description

PORT B	I/O		Alternate Function	
	Input*	Output	Signal	Condition
PB0	without pull-up	push-pull	Analog input (ADC)	CH[2:0] = 000 (ADCCSR)
PB1	without pull-up	push-pull	Analog input (ADC)	CH[2:0] = 001 (ADCCSR)
PB2	without pull-up	push-pull	Analog input (ADC)	CH[2:0] = 010 (ADCCSR)
PB3	without pull-up	push-pull	Analog input (ADC)	CH[2:0] = 011 (ADCCSR)
PB4	without pull-up	push-pull	Analog input (ADC)	CH[2:0] = 100 (ADCCSR)
			IT5 Schmitt triggered input	IT4E = 1 (ITIFRE)
PB5	without pull-up	push-pull	Analog input (ADC)	CH[2:0] = 101 (ADCCSR)
			IT6 Schmitt triggered input	IT5E = 1 (ITIFRE)
PB6	without pull-up	push-pull	Analog input (ADC)	CH[2:0] = 110 (ADCCSR)
			IT7 Schmitt triggered input	IT6E = 1 (ITIFRE)
PB7	without pull-up	push-pull	Analog input (ADC)	CH[2:0] = 111 (ADCCSR)
			IT8 Schmitt triggered input	IT7E = 1 (ITIFRE)

\*Reset State

Figure 20. Port B Configuration



I/O PORTS (Cont'd)

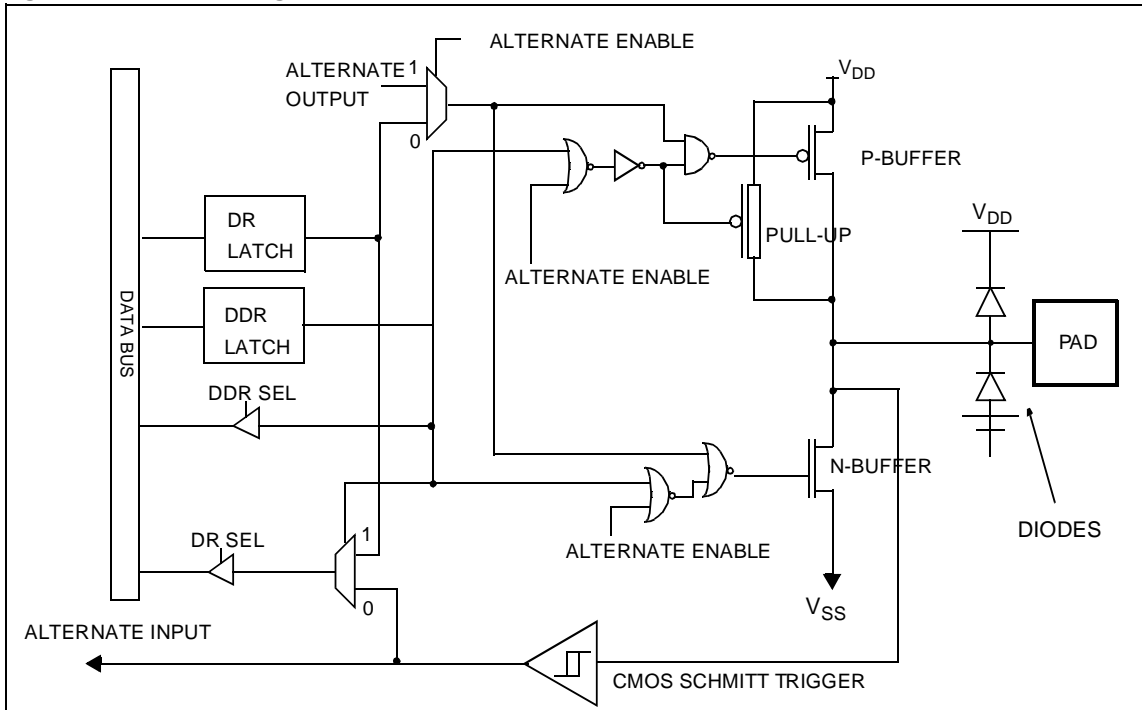
5.1.6 Port C

Table 12. Port C Description

PORT C	I / O		Alternate Function	
	Input*	Output	Signal	Condition
PC0	with pull-up	push-pull	RDI (SCI input)	
PC1	with pull-up	push-pull	TDO (SCI output)	SCI enable
PC2	with pull-up	push-pull	USBOE (USB output enable)	USBOE =1 (MISCR)

\*Reset State

Figure 21. Port C Configuration



**I/O PORTS** (Cont'd)**5.1.7 Register Description****DATA REGISTERS (PxDR)**

Port A Data Register (PADR): 0000h

Port B Data Register (PBDR): 0002h

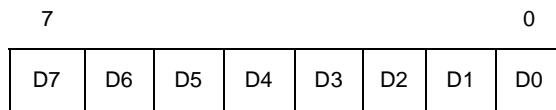
Port C Data Register (PCDR): 0004h

Read/Write

Reset Value Port A: 0000 0000 (00h)

Reset Value Port B: 0000 0000 (00h)

Reset Value Port C: 1111 x000 (FXh)

**Note:** for Port C, unused bits (7-3) are not accessible.**Bit 7:0 = D7-D0 Data Register 8 bits.**

The DR register has a specific behaviour according to the selected input/output configuration. Writing the DR register is always taken in account even if the pin is configured as an input. Reading the DR register returns either the DR register latch content (pin configured as output) or the digital value applied to the I/O pin (pin configured as input).

**DATA DIRECTION REGISTER (PxDDR)**

Port A Data Direction Register (PADDR): 0001h

Port B Data Direction Register (PBDDR): 0003h

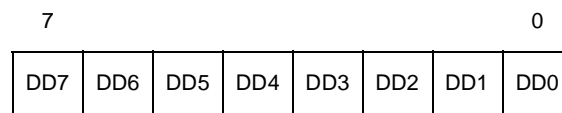
Port C Data Direction Register (PCDDR): 0005h

Read/Write

Reset Value Port A: 0000 0000 (00h)

Reset Value Port B: 0000 0000 (00h)

Reset Value Port C: 1111 x000 (FXh)

**Note:** for Port C, unused bits (7-3) are not accessible**Bit 7:0 = DD7-DD0 Data Direction Register 8 bits.**

The DDR register gives the input/output direction configuration of the pins. Each bits is set and cleared by software.

0: Input mode

1: Output mode

**Table 13. I/O Ports Register Map**

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
00	<b>PADR</b>	MSB							LSB
01	<b>PADDR</b>	MSB							LSB
02	<b>PBDR</b>	MSB							LSB
03	<b>PBDDR</b>	MSB							LSB
04	<b>PCDR</b>	MSB							LSB
05	<b>PCDDR</b>	MSB							LSB

## 5.2 MISCELLANEOUS REGISTER

Address: 0009h — Read/Write  
Reset Value: 1111 0000 (F0h)

7							0
-	-	-	-	LVD	CLKDIV	USBOE	MCO

Bit 7:4 = Reserved

Bit 3 = **LVD** *Low Voltage Detector*.

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

0: LVD enabled  
1: LVD disabled

Bit 2 = **CLKDIV** *Clock Divider*.

This bit is set by software and only cleared by hardware after a reset. If this bit is set, it enables the use of a 12 MHz external oscillator (refer to Figure 10 on page 17).

0: 24 MHz external oscillator  
1: 12 MHz external oscillator.

Bit 1 = **USBOE** *USB enable*.

If this bit is set, the port PC2 outputs the USB output enable signal (at "1" when the ST7 USB is transmitting data).

Unused bits 7-4 are set.

Bit 0 = **MCO** *Main Clock Out selection*

This bit enables the MCO alternate function on the PA0 I/O port. It is set and cleared by software.

0: MCO alternate function disabled (I/O pin free for general-purpose I/O)  
1: MCO alternate function enabled ( $f_{CPU}$  on I/O port)





## 5.3 WATCHDOG TIMER (WDG)

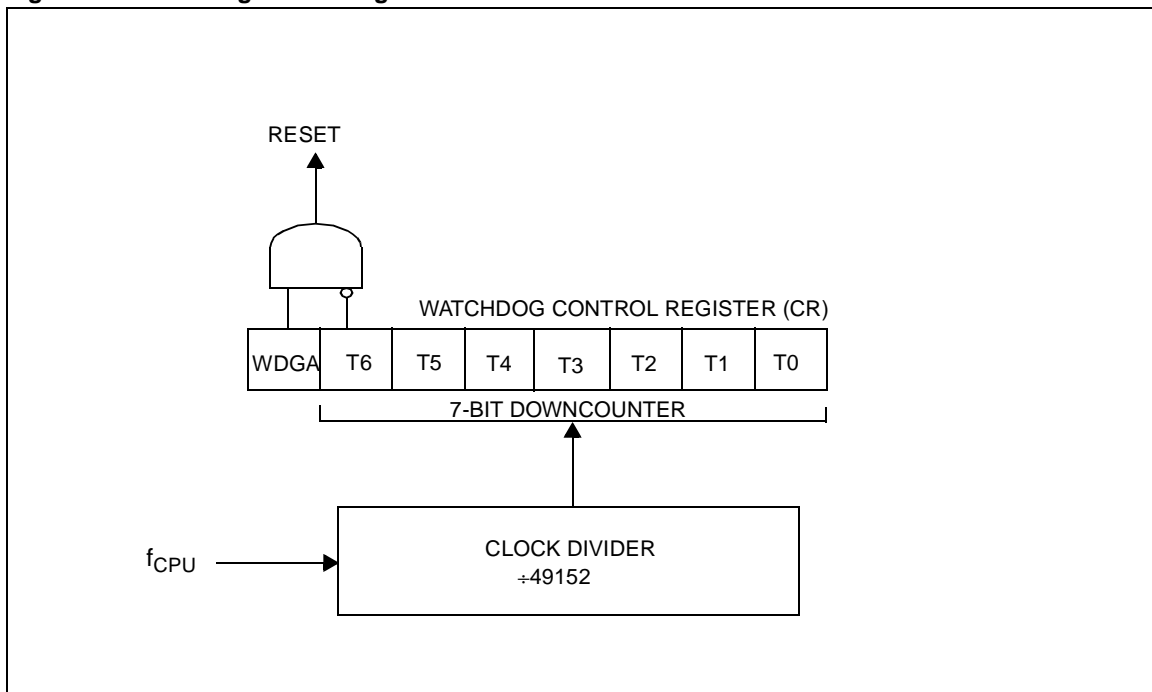
### 5.3.1 Introduction

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.

### 5.3.2 Main Features

- Programmable timer (64 increments of 49152 CPU cycles)
- Programmable reset
- Reset (if watchdog activated) when the T6 bit reaches zero

Figure 22. Watchdog Block Diagram



**WATCHDOG TIMER (Cont'd)****5.3.3 Functional Description**

The counter value stored in the CR register (bits T6:T0), is decremented every 49,152 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 T6:T0) rolls over from 40h to 3Fh (T6 becomes cleared), it initiates a reset cycle pulling low the reset pin for typically 500ns.

The application program must write in the CR register at regular intervals during normal operation to prevent an MCU reset. The value to be stored in the CR register must be between FFh and C0h (see Table 14 . Watchdog Timing (f<sub>CPU</sub> = 8 MHz)):

- The WDGA bit is set (watchdog enabled)
- The T6 bit is set to prevent generating an immediate reset
- The T5:T0 bits contain the number of increments which represents the time delay before the watchdog produces a reset.

**Table 14. Watchdog Timing (f<sub>CPU</sub> = 8 MHz)**

	CR Register initial value	WDG timeout period (ms)
Max	FFh	393.216
Min	C0h	6.144

**Notes:** 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).

**5.3.3.1 Using Halt Mode with the WDG**

The HALT instruction stops the oscillator. When the oscillator is stopped, the WDG stops counting and is no longer able to generate a reset until the microcontroller receives an external interrupt or a reset.

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

**Recommendations**

- Make sure that an external event is available to wake up the microcontroller from Halt mode.
- Before executing the HALT instruction, refresh the WDG counter, to avoid an unexpected WDG

reset immediately after waking up the microcontroller.

- When using an external interrupt to wake up the microcontroller, reinitialize the corresponding I/O as "Input Pull-up with Interrupt" before executing the HALT instruction. The main reason for this is that the I/O may be wrongly configured due to external interference or by an unforeseen logical condition.
- For the same reason, reinitialize the level 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 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).

**5.3.4 Interrupts**

None.

**5.3.5 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

**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 15. Watchdog Timer Register Map and Reset Values**

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

## 5.4 16-BIT TIMER

### 5.4.1 Introduction

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

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

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

Some ST7 devices have two on-chip 16-bit timers. They are completely independent, and do not share any resources. They are synchronized after a MCU reset as long as the timer clock frequencies are not modified.

This description covers one or two 16-bit timers. In ST7 devices with two timers, register names are prefixed with TA (Timer A) or TB (Timer B).

### 5.4.2 Main Features

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

The Block Diagram is shown in Figure 1.

**\*Note:** Some timer pins may not be available (not bonded) in some ST7 devices. Refer to the device pin out description.

When reading an input signal on a non-bonded pin, the value will always be '1'.

### 5.4.3 Functional Description

#### 5.4.3.1 Counter

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

Counter Register (CR):

- Counter High Register (CHR) is the most significant byte (MS Byte).
- Counter Low Register (CLR) is the least significant byte (LS Byte).

Alternate Counter Register (ACR)

- Alternate Counter High Register (ACHR) is the most significant byte (MS Byte).
- Alternate Counter Low Register (ACLR) is the least significant byte (LS Byte).

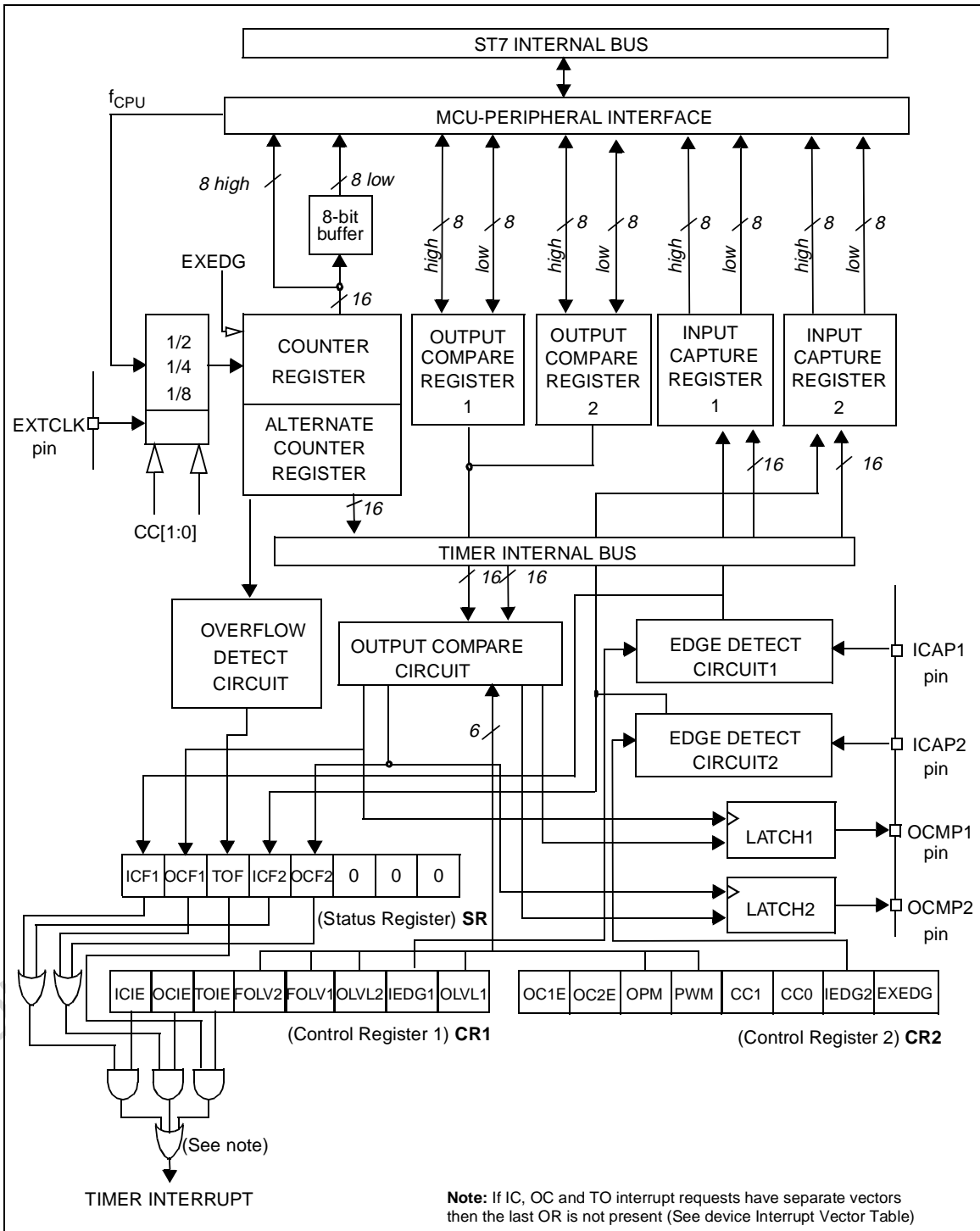
These two read-only 16-bit registers contain the same value but with the difference that reading the ACLR register does not clear the TOF bit (Timer overflow flag), located in the Status register (SR). (See note at the end of paragraph titled 16-bit read sequence).

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

The timer clock depends on the clock control bits of the CR2 register, as illustrated in Table 1. The value in the counter register repeats every 131.072, 262.144 or 524.288 CPU clock cycles depending on the CC[1:0] bits. The timer frequency can be  $f_{CPU}/2$ ,  $f_{CPU}/4$ ,  $f_{CPU}/8$  or an external frequency.

16-BIT TIMER (Cont'd)

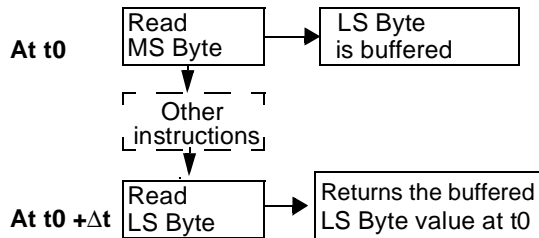
Figure 23. Timer Block Diagram



**16-BIT TIMER** (Cont'd)

**16-bit Read Sequence:** (from either the Counter Register or the Alternate Counter Register).

*Beginning of the sequence*



*Sequence completed*

The user must read the MS Byte first, then the LS Byte value is buffered automatically.

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

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

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

- The TOF bit of the SR register is set.
- A timer interrupt is generated if:
  - TOIE bit of the CR1 register is set and
  - I bit of the CC register is cleared.

If one of these conditions is false, the interrupt remains pending to be issued as soon as they are both true.

Clearing the overflow interrupt request is done in two steps:

1. Reading the SR register while the TOF bit is set.
2. An access (read or write) to the CLR register.

**Note:** The TOF bit is not cleared by accessing the ACLR register. The advantage of accessing the ACLR register rather than the CLR register is that it allows simultaneous use of the overflow function and reading the free running counter at random times (for example, to measure elapsed time) without the risk of clearing the TOF bit erroneously.

The timer is not affected by WAIT mode.

In HALT mode, the counter stops counting until the mode is exited. Counting then resumes from the previous count (MCU awakened by an interrupt) or from the reset count (MCU awakened by a Reset).

#### 5.4.3.2 External Clock

The external clock (where available) is selected if CC0=1 and CC1=1 in the CR2 register.

The status of the EXEDG bit in the CR2 register determines the type of level transition on the external clock pin EXTCLK that will trigger the free running counter.

The counter is synchronised with the falling edge of the internal CPU clock.

A minimum of four falling edges of the CPU clock must occur between two consecutive active edges of the external clock; thus the external clock frequency must be less than a quarter of the CPU clock frequency.



## 16-BIT TIMER (Cont'd)

Figure 24. Counter Timing Diagram, internal clock divided by 2

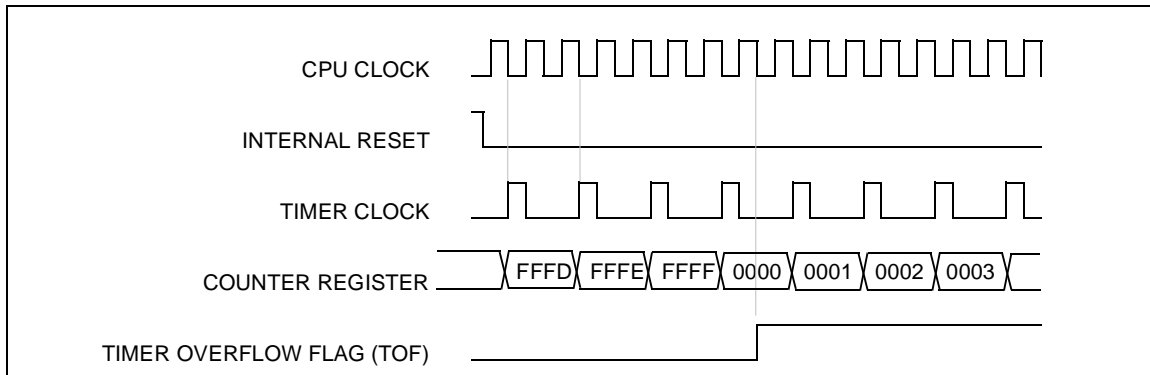


Figure 25. Counter Timing Diagram, internal clock divided by 4

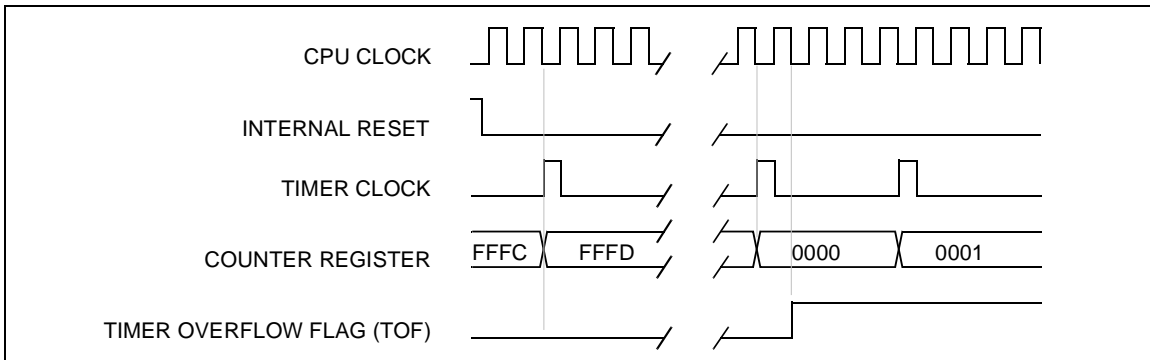
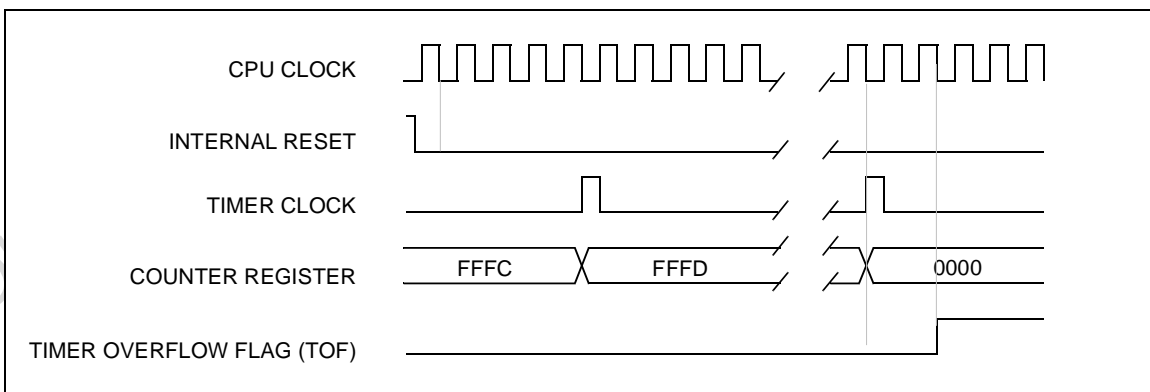


Figure 26. Counter Timing Diagram, internal clock divided by 8

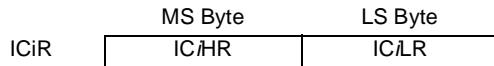


**Note:** The MCU is in reset state when the internal reset signal is high. When it is low, the MCU is running.

**16-BIT TIMER (Cont'd)****5.4.3.3 Input Capture**

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

The two input capture 16-bit registers (IC1R and IC2R) are used to latch the value of the free running counter after a transition is detected by the ICAP $i$  pin (see figure 5).



The ICiR register is a read-only register.

The active transition is software programmable through the IEDG $i$  bit of Control Registers (CR $i$ ).

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

**Procedure:**

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

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

And select the following in the CR1 register:

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

When an input capture occurs:

- The ICF $i$  bit is set.
- The ICiR register contains the value of the free running counter on the active transition on the ICAP $i$  pin (see Figure 6).
- A timer interrupt is generated if the ICIE bit is set and the I bit is cleared in the CC register. Otherwise, the interrupt remains pending until both conditions become true.

Clearing the Input Capture interrupt request (i.e. clearing the ICF $i$  bit) is done in two steps:

1. Reading the SR register while the ICF $i$  bit is set.
2. An access (read or write) to the ICiLR register.

**Notes:**

1. After reading the ICiHR register, the transfer of input capture data is inhibited and ICF $i$  will never be set until the ICiLR register is also read.
2. The ICiR register contains the free running counter value which corresponds to the most recent input capture.
3. The 2 input capture functions can be used together even if the timer also uses the 2 output compare functions.
4. In One Pulse mode and PWM mode only the input capture 2 function can be used.
5. The alternate inputs (ICAP1 & ICAP2) are always directly connected to the timer. So any transitions on these pins activate the input capture function.  
Moreover if one of the ICAP $i$  pin is configured as an input and the second one as an output, an interrupt can be generated if the user toggles the output pin and if the ICIE bit is set.  
This can be avoided if the input capture function  $i$  is disabled by reading the ICiHR (see note 1).
6. The TOF bit can be used with an interrupt in order to measure events that exceed the timer range (FFFFh).



## 16-BIT TIMER (Cont'd)

Figure 27. Input Capture Block Diagram

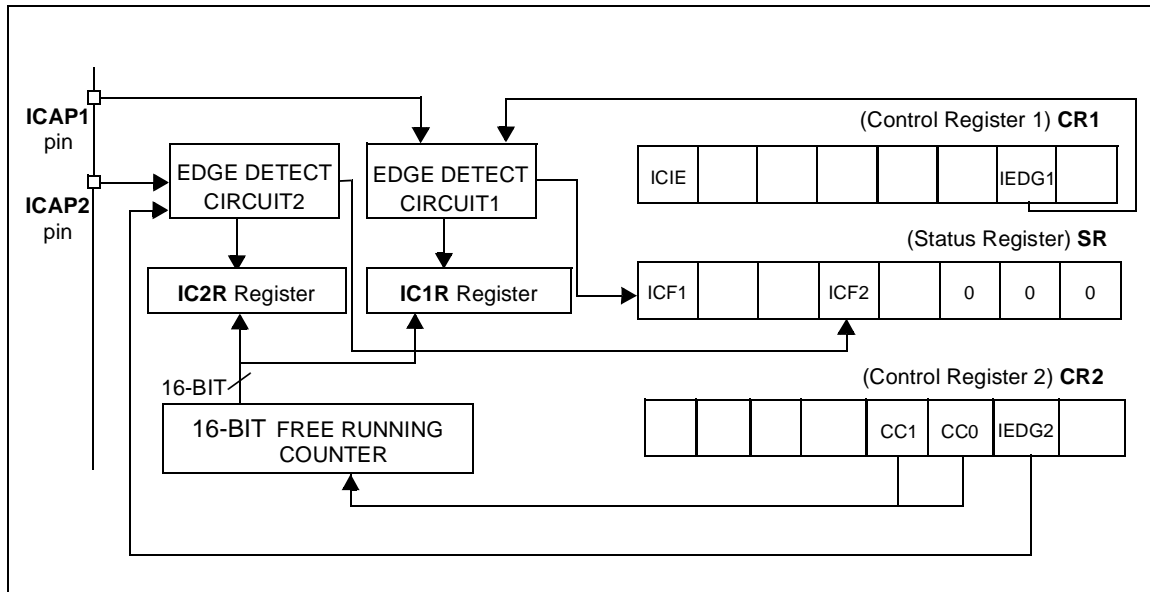
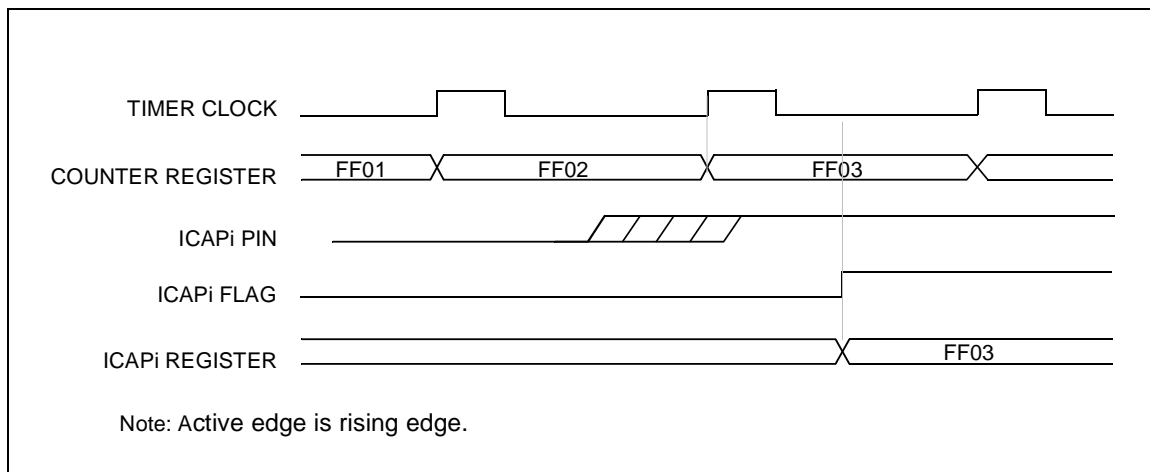


Figure 28. Input Capture Timing Diagram



**16-BIT TIMER (Cont'd)****5.4.3.4 Output Compare**

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

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

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

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

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

	MS Byte	LS Byte
OC <i>R</i>	OC <i>HR</i>	OC <i>LR</i>

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

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

**Procedure:**

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

- Set the OC*E* bit if an output is needed then the OCMP*i* pin is dedicated to the output compare  $i$  signal.
- Select the timer clock (CC[1:0]) (see Table 1).

And select the following in the CR1 register:

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

When a match is found between OC*R* <sub>$i$</sub>  register and CR register:

- OCF*i* bit is set.

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

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

$$\Delta OCiR = \frac{\Delta t * f_{CPU}}{PRESC}$$

Where:

- $\Delta t$  = Output compare period (in seconds)
- $f_{CPU}$  = CPU clock frequency (in hertz)
- PRESC = Timer prescaler factor (2, 4 or 8 depending on CC[1:0] bits, see Table 1)

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

$$\Delta OCiR = \Delta t * f_{EXT}$$

Where:

- $\Delta t$  = Output compare period (in seconds)
- $f_{EXT}$  = External timer clock frequency (in hertz)

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

1. Reading the SR register while the OCF*i* bit is set.
2. An access (read or write) to the OC*LR* register.

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

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

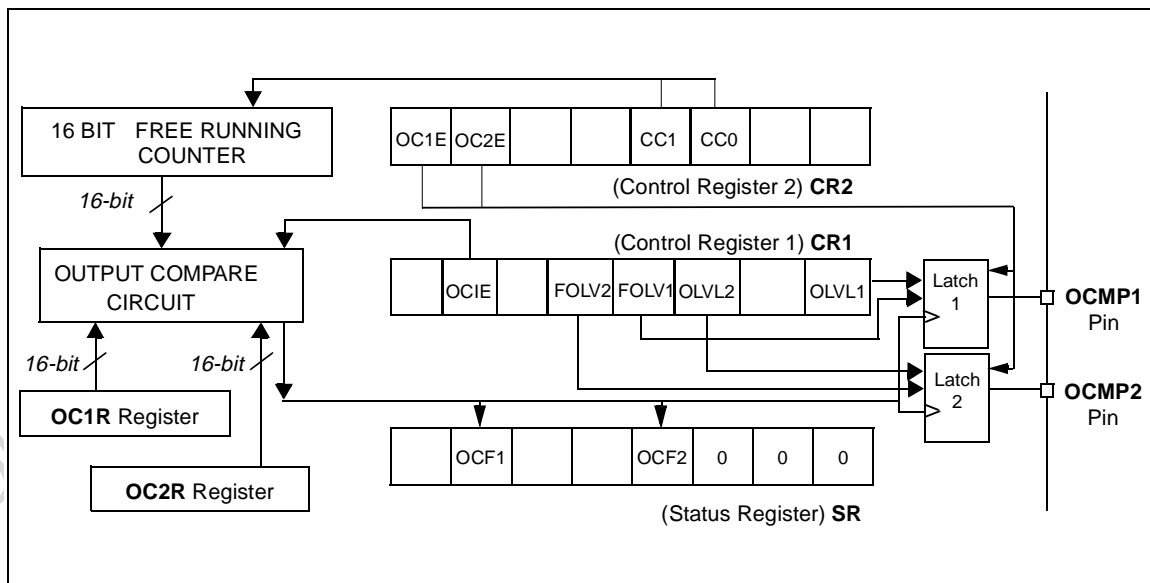
**16-BIT TIMER (Cont'd)****Notes:**

1. After a processor write cycle to the OC<sub>HR</sub> register, the output compare function is inhibited until the OC<sub>LR</sub> register is also written.
2. If the OC<sub>iE</sub> bit is not set, the OC<sub>MPi</sub> pin is a general I/O port and the OLV<sub>Li</sub> bit will not appear when a match is found but an interrupt could be generated if the OC<sub>IE</sub> bit is set.
3. When the timer clock is  $f_{CPU}/2$ , OC<sub>Fi</sub> and OC<sub>MPi</sub> are set while the counter value equals the OC<sub>R</sub> register value (see Figure 8). This behaviour is the same in OPM or PWM mode. When the timer clock is  $f_{CPU}/4$ ,  $f_{CPU}/8$  or in external clock mode, OC<sub>Fi</sub> and OC<sub>MPi</sub> are set while the counter value equals the OC<sub>R</sub> register value plus 1 (see Figure 9).
4. The output compare functions can be used both for generating external events on the OC<sub>MPi</sub> pins even if the input capture mode is also used.
5. The value in the 16-bit OC<sub>R</sub> register and the OLV<sub>i</sub> bit should be changed after each successful comparison in order to control an output waveform or establish a new elapsed timeout.

**Forced Compare Output capability**

When the FOLV<sub>i</sub> bit is set by software, the OLV<sub>Li</sub> bit is copied to the OC<sub>MPi</sub> pin. The OLV<sub>i</sub> bit has to be toggled in order to toggle the OC<sub>MPi</sub> pin when it is enabled (OC<sub>iE</sub> bit=1). The OC<sub>Fi</sub> bit is then not set by hardware, and thus no interrupt request is generated.

FOLV<sub>Li</sub> bits have no effect in either One-Pulse mode or PWM mode.

**Figure 29. Output Compare Block Diagram**

16-BIT TIMER (Cont'd)

Figure 30. Output Compare Timing Diagram,  $f_{\text{TIMER}} = f_{\text{CPU}}/2$

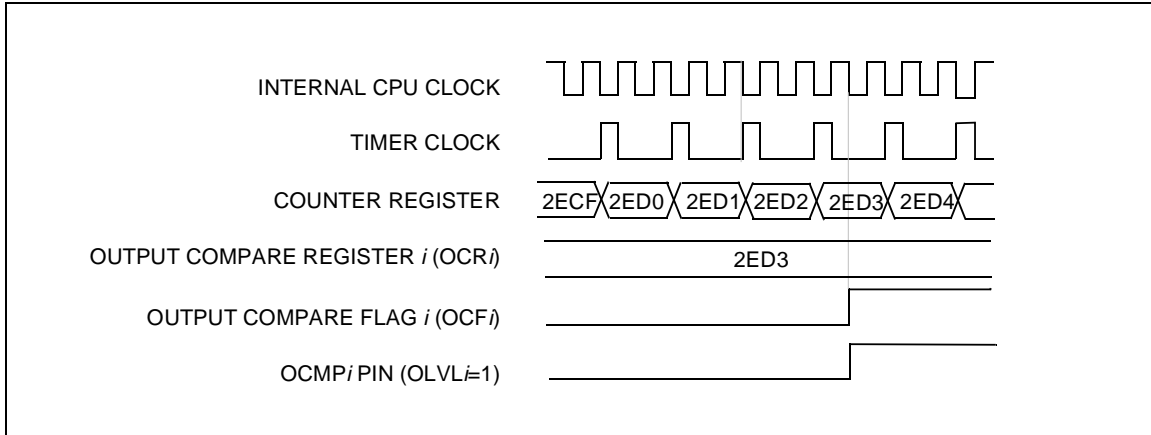
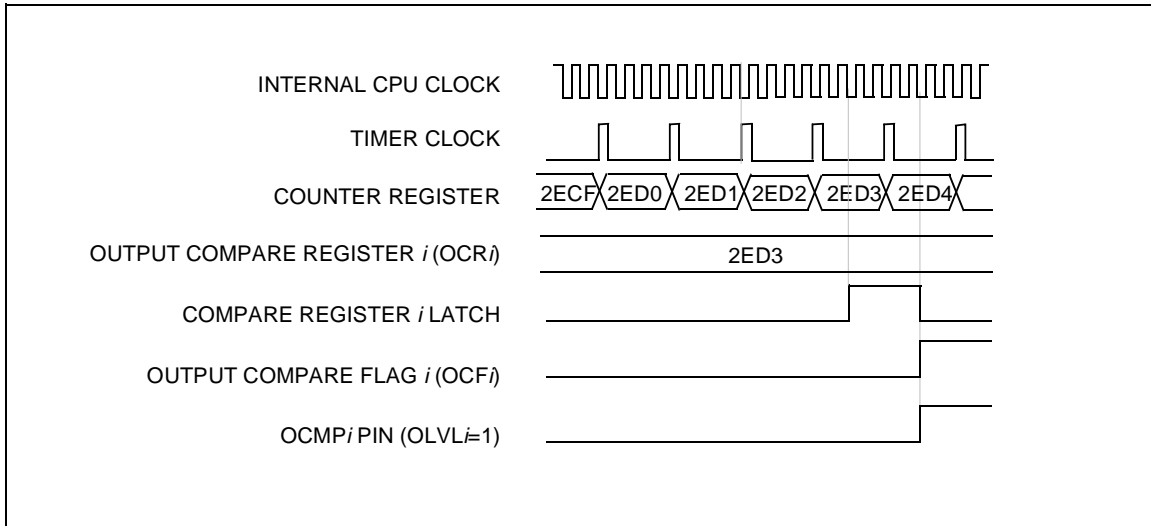


Figure 31. Output Compare Timing Diagram,  $f_{\text{TIMER}} = f_{\text{CPU}}/4$



## 16-BIT TIMER (Cont'd)

### 5.4.3.5 One Pulse Mode

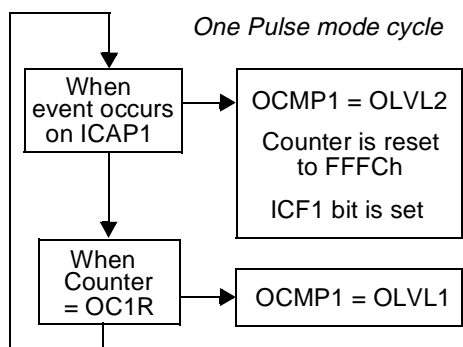
One Pulse mode enables the generation of a pulse when an external event occurs. This mode is selected via the OPM bit in the CR2 register.

The One Pulse mode uses the Input Capture1 function and the Output Compare1 function.

#### Procedure:

To use One Pulse mode:

1. Load the OC1R register with the value corresponding to the length of the pulse (see the formula in the opposite column).
2. Select the following in the CR1 register:
  - Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after the pulse.
  - Using the OLVL2 bit, select the level to be applied to the OCMP1 pin during the pulse.
  - Select the edge of the active transition on the ICAP1 pin with the IEDG1 bit (the ICAP1 pin must be configured as floating input).
3. Select the following in the CR2 register:
  - Set the OC1E bit, the OCMP1 pin is then dedicated to the Output Compare 1 function.
  - Set the OPM bit.
  - Select the timer clock CC[1:0] (see Table 1).



Then, on a valid event on the ICAP1 pin, the counter is initialized to FFFCh and the OLVL2 bit is loaded on the OCMP1 pin, the ICF1 bit is set and the value FFFDh is loaded in the IC1R register.

Because the ICF1 bit is set when an active edge occurs, an interrupt can be generated if the ICIE bit is set.

Clearing the Input Capture interrupt request (i.e. clearing the ICF $i$  bit) is done in two steps:

1. Reading the SR register while the ICF $i$  bit is set.
2. An access (read or write) to the IC $i$ LR register.

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

$$\text{OC1R Value} = \frac{t \cdot f_{\text{CPU}}}{\text{PRESC}} - 5$$

Where:

$t$  = Pulse period (in seconds)

$f_{\text{CPU}}$  = CPU clock frequency (in hertz)

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

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

$$\text{OC1R} = t \cdot f_{\text{EXT}} - 5$$

Where:

$t$  = Pulse period (in seconds)

$f_{\text{EXT}}$  = External timer clock frequency (in hertz)

When the value of the counter is equal to the value of the contents of the OC1R register, the OLVL1 bit is output on the OCMP1 pin (see Figure 10).

#### Notes:

1. The OCF1 bit cannot be set by hardware in One Pulse mode but the OCF2 bit can generate an Output Compare interrupt.
2. When the Pulse Width Modulation (PWM) and One Pulse mode (OPM) bits are both set, the PWM mode is the only active one.
3. If OLVL1=OLVL2 a continuous signal will be seen on the OCMP1 pin.
4. The ICAP1 pin can not be used to perform input capture. The ICAP2 pin can be used to perform input capture (ICF2 can be set and IC2R can be loaded) but the user must take care that the counter is reset each time a valid edge occurs on the ICAP1 pin and ICF1 can also generate interrupt if ICIE is set.
5. When One Pulse mode is used OC1R is dedicated to this mode. Nevertheless OC2R and OCF2 can be used to indicate that a period of time has elapsed but cannot generate an output waveform because the OLVL2 level is dedicated to One Pulse mode.

16-BIT TIMER (Cont'd)

Figure 32. One Pulse Mode Timing Example

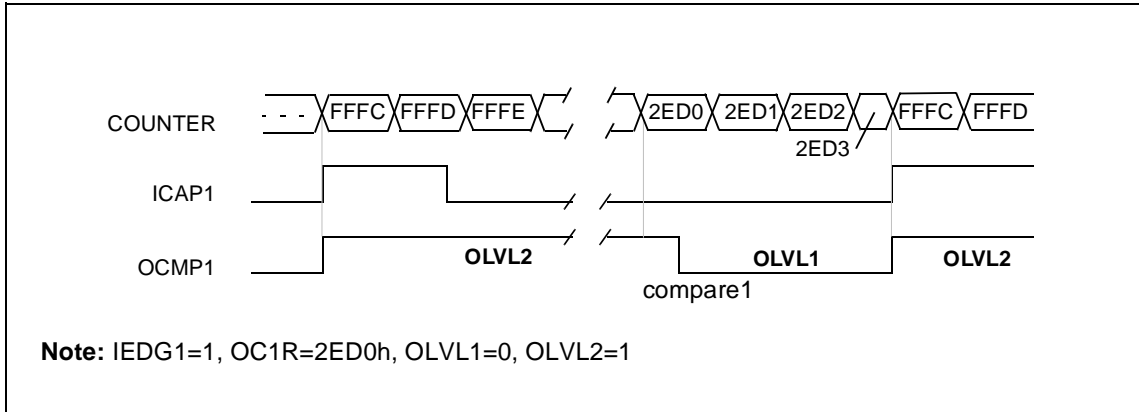
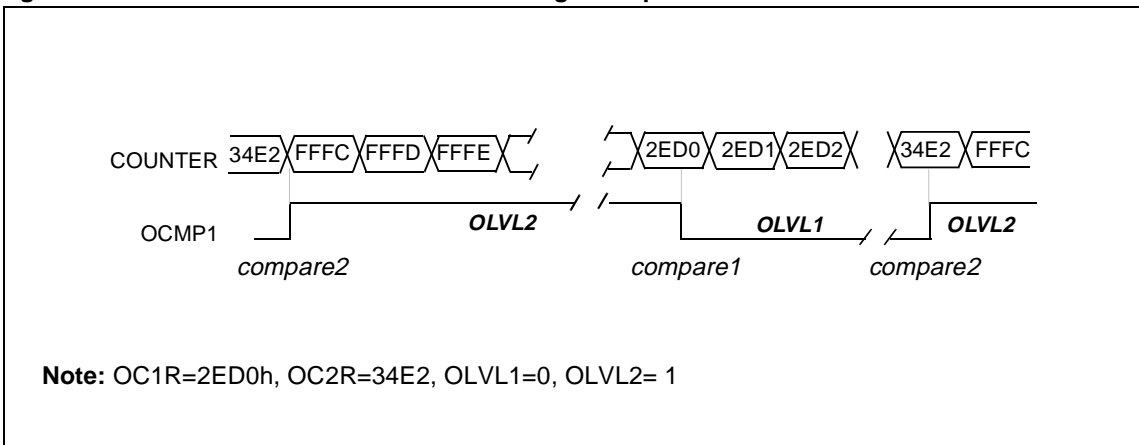


Figure 33. Pulse Width Modulation Mode Timing Example



## 16-BIT TIMER (Cont'd)

### 5.4.3.6 Pulse Width Modulation Mode

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

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

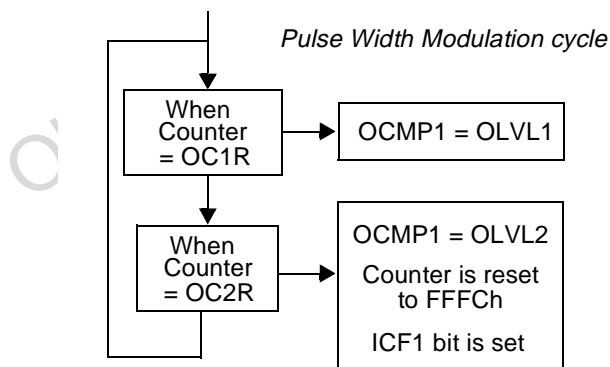
#### Procedure

To use Pulse Width Modulation mode:

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

If OLVL1=1 and OLVL2=0, the length of the positive pulse is the difference between the OC2R and OC1R registers.

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



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

$$\text{OC/R Value} = \frac{t \cdot f_{\text{CPU}}}{\text{PRESC}} - 5$$

Where:

$t$  = Signal or pulse period (in seconds)

$f_{\text{CPU}}$  = CPU clock frequency (in hertz)

PRESC = Timer prescaler factor (2, 4 or 8 depending on CC[1:0] bits, see Table 1)

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

$$\text{OC/R} = t \cdot f_{\text{EXT}} - 5$$

Where:

$t$  = Signal or pulse period (in seconds)

$f_{\text{EXT}}$  = External timer clock frequency (in hertz)

The Output Compare 2 event causes the counter to be initialized to FFFCh (See Figure 11)

#### Notes:

1. After a write instruction to the OC/HR register, the output compare function is inhibited until the OC/LR register is also written.
2. The OCF1 and OCF2 bits cannot be set by hardware in PWM mode, therefore the Output Compare interrupt is inhibited.
3. The ICF1 bit is set by hardware when the counter reaches the OC2R value and can produce a timer interrupt if the ICIE bit is set and the I bit is cleared.
4. In PWM mode the ICAP1 pin can not be used to perform input capture because it is disconnected from the timer. The ICAP2 pin can be used to perform input capture (ICF2 can be set and IC2R can be loaded) but the user must take care that the counter is reset after each period and ICF1 can also generate an interrupt if ICIE is set.
5. When the Pulse Width Modulation (PWM) and One Pulse mode (OPM) bits are both set, the PWM mode is the only active one.

## 16-BIT TIMER (Cont'd)

## 5.4.4 Low Power Modes

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

## 5.4.5 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
Input Capture 1 event/Counter reset in PWM mode	ICF1	ICIE	Yes	No
Input Capture 2 event	ICF2		Yes	No
Output Compare 1 event (not available in PWM mode)	OCF1	OCIE	Yes	No
Output Compare 2 event (not available in PWM mode)	OCF2		Yes	No
Timer Overflow event	TOF	TOIE	Yes	No

**Note:** The 16-bit Timer interrupt events are connected to the same interrupt vector (see Interrupts chapter). These events generate an interrupt if the corresponding Enable Control Bit is set and the interrupt mask in the CC register is reset (RIM instruction).

## 5.4.6 Summary of Timer modes

MODES	AVAILABLE RESOURCES			
	Input Capture 1	Input Capture 2	Output Compare 1	Output Compare 2
Input Capture (1 and/or 2)	Yes	Yes	Yes	Yes
Output Compare (1 and/or 2)	Yes	Yes	Yes	Yes
One Pulse mode	No	Not Recommended <sup>1)</sup>	No	Partially <sup>2)</sup>
PWM Mode	No	Not Recommended <sup>3)</sup>	No	No

<sup>1)</sup> See note 4 in Section 0.1.3.5 One Pulse Mode

<sup>2)</sup> See note 5 in Section 0.1.3.5 One Pulse Mode

<sup>3)</sup> See note 4 in Section 0.1.3.6 Pulse Width Modulation Mode



**16-BIT TIMER (Cont'd)****5.4.7 Register Description**

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

**CONTROL REGISTER 1 (CR1)**

Read/Write

Reset Value: 0000 0000 (00h)

7							0
ICIE	OCIE	TOIE	FOLV2	FOLV1	OLVL2	IEDG1	OLVL1

Bit 7 = **ICIE** *Input Capture Interrupt Enable*.

0: Interrupt is inhibited.

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

Bit 6 = **OCIE** *Output Compare Interrupt Enable*.

0: Interrupt is inhibited.

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

Bit 5 = **TOIE** *Timer Overflow Interrupt Enable*.

0: Interrupt is inhibited.

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

Bit 4 = **FOLV2** *Forced Output Compare 2*.

This bit is set and cleared by software.

0: No effect on the OCMP2 pin.

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

Bit 3 = **FOLV1** *Forced Output Compare 1*.

This bit is set and cleared by software.

0: No effect on the OCMP1 pin.

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

Bit 2 = **OLVL2** *Output Level 2*.

This bit is copied to the OCMP2 pin whenever a successful comparison occurs with the OC2R register and OCxE is set in the CR2 register. This value is copied to the OCMP1 pin in One Pulse mode and Pulse Width Modulation mode.

Bit 1 = **IEDG1** *Input Edge 1*.

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

0: A falling edge triggers the capture.

1: A rising edge triggers the capture.

Bit 0 = **OLVL1** *Output Level 1*.

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

**16-BIT TIMER (Cont'd)****CONTROL REGISTER 2 (CR2)**

Read/Write

Reset Value: 0000 0000 (00h)

7							0
OC1E	OC2E	OPM	PWM	CC1	CC0	IEDG2	EXEDG

**Bit 7 = OC1E Output Compare 1 Pin Enable.**

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

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

1: OCMP1 pin alternate function enabled.

**Bit 6 = OC2E Output Compare 2 Pin Enable.**

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

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

1: OCMP2 pin alternate function enabled.

**Bit 5 = OPM One Pulse mode.**

0: One Pulse mode is not active.

1: One Pulse mode is active, the ICAP1 pin can be used to trigger one pulse on the OCMP1 pin; the active transition is given by the IEDG1 bit. The length of the generated pulse depends on the contents of the OC1R register.

**Bit 4 = PWM Pulse Width Modulation.**

0: PWM mode is not active.

1: PWM mode is active, the OCMP1 pin outputs a programmable cyclic signal; the length of the pulse depends on the value of OC1R register; the period depends on the value of OC2R register.

**Bits 3:2 = CC[1:0] Clock Control.**

The timer clock mode depends on these bits:

**Table 16. Clock Control Bits**

Timer Clock	CC1	CC0
$f_{\text{CPU}} / 4$	0	0
$f_{\text{CPU}} / 2$	0	1
$f_{\text{CPU}} / 8$	1	0
External Clock (where available)	1	1

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

**Bit 1 = IEDG2 Input Edge 2.**

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

0: A falling edge triggers the capture.

1: A rising edge triggers the capture.

**Bit 0 = EXEDG External Clock Edge.**

This bit determines which type of level transition on the external clock pin (EXTCLK) will trigger the counter register.

0: A falling edge triggers the counter register.

1: A rising edge triggers the counter register.

**16-BIT TIMER (Cont'd)****STATUS REGISTER (SR)**

Read Only

Reset Value: 0000 0000 (00h)

The three least significant bits are not used.

7							0
ICF1	OCF1	TOF	ICF2	OCF2	0	0	0

Bit 7 = **ICF1** *Input Capture Flag 1.*

0: No input capture (reset value).

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

Bit 6 = **OCF1** *Output Compare Flag 1.*

0: No match (reset value).

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

Bit 5 = **TOF** *Timer Overflow Flag.*

0: No timer overflow (reset value).

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

**Note:** Reading or writing the ACLR register does not clear TOF.Bit 4 = **ICF2** *Input Capture Flag 2.*

0: No input capture (reset value).

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

Bit 3 = **OCF2** *Output Compare Flag 2.*

0: No match (reset value).

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

Bit 2-0 = Reserved, forced by hardware to 0.

**INPUT CAPTURE 1 HIGH REGISTER (IC1HR)**

Read Only

Reset Value: Undefined

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

7							0
MSB							LSB

**INPUT CAPTURE 1 LOW REGISTER (IC1LR)**

Read Only

Reset Value: Undefined

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

7							0
MSB							LSB

**OUTPUT COMPARE 1 HIGH REGISTER (OC1HR)**

Read/Write

Reset Value: 1000 0000 (80h)

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

7							0
MSB							LSB

**OUTPUT COMPARE 1 LOW REGISTER (OC1LR)**

Read/Write

Reset Value: 0000 0000 (00h)

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

7							0
MSB							LSB

16-BIT TIMER (Cont'd)

**OUTPUT COMPARE 2 HIGH REGISTER (OC2HR)**

Read/Write  
Reset Value: 1000 0000 (80h)

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



**ALTERNATE COUNTER HIGH REGISTER (ACHR)**

Read Only  
Reset Value: 1111 1111 (FFh)

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



**OUTPUT COMPARE 2 LOW REGISTER (OC2LR)**

Read/Write  
Reset Value: 0000 0000 (00h)

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



**ALTERNATE COUNTER LOW REGISTER (ACLRL)**

Read Only  
Reset Value: 1111 1100 (FCh)

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



**COUNTER HIGH REGISTER (CHR)**

Read Only  
Reset Value: 1111 1111 (FFh)

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



**INPUT CAPTURE 2 HIGH REGISTER (IC2HR)**

Read Only  
Reset Value: Undefined

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



**COUNTER LOW REGISTER (CLR)**

Read Only  
Reset Value: 1111 1100 (FCh)

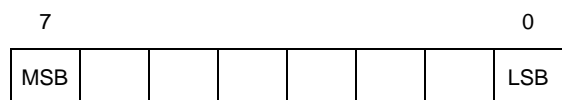
This is an 8-bit register that contains the low part of the counter value. A write to this register resets the counter. An access to this register after accessing the SR register clears the TOF bit.



**INPUT CAPTURE 2 LOW REGISTER (IC2LR)**

Read Only  
Reset Value: Undefined

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



## 16-BIT TIMER (Cont'd)

Table 17. 16-Bit Timer Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
11	<b>CR2</b> Reset Value	OC1E 0	OC2E 0	OPM 0	PWM 0	CC1 0	CC0 0	IEDG2 0	EXEDG 0
12	<b>CR1</b> Reset Value	ICIE 0	OCIE 0	TOIE 0	FOLV2 0	FOLV1 0	OLVL2 0	IEDG1 0	OLVL1 0
13	<b>SR</b> Reset Value	ICF1 0	OCF1 0	TOF 0	ICF2 0	OCF2 0	0 0	0 0	0 0
14	<b>IC1HR</b> Reset Value	MSB							LSB
15	<b>IC1LR</b> Reset Value	MSB							LSB
16	<b>OC1HR</b> Reset Value	MSB 1	- 0	- 0	- 0	- 0	- 0	- 0	LSB 0
17	<b>OC1LR</b> Reset Value	MSB 0	- 0	- 0	- 0	- 0	- 0	- 0	LSB 0
18	<b>CHR</b> Reset Value	MSB 1	- 1	- 1	- 1	- 1	- 1	- 1	LSB 1
19	<b>CLR</b> Reset Value	MSB 1	- 1	- 1	- 1	- 1	- 1	- 0	LSB 0
1A	<b>ACHR</b> Reset Value	MSB 1	- 1	- 1	- 1	- 1	- 1	- 1	LSB 1
1B	<b>ACLR</b> Reset Value	MSB 1	- 1	- 1	- 1	- 1	- 1	- 0	LSB 0
1C	<b>IC2HR</b> Reset Value	MSB							LSB
1D	<b>IC2LR</b> Reset Value	MSB							LSB
1E	<b>OC2HR</b> Reset Value	MSB 1	- 0	- 0	- 0	- 0	- 0	- 0	LSB 0
1F	<b>OC2LR</b> Reset Value	MSB 0	- 0	- 0	- 0	- 0	- 0	- 0	LSB 0

## 5.5 SERIAL COMMUNICATIONS INTERFACE (SCI)

### 5.5.1 Introduction

The Serial Communications Interface (SCI) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format.

### 5.5.2 Main Features

- Full duplex, asynchronous communications
- NRZ standard format (Mark/Space)
- Independently programmable transmit and receive baud rates up to 250K baud.
- Programmable data word length (8 or 9 bits)
- Receive buffer full, Transmit buffer empty and End of Transmission flags
- Two receiver wake-up modes:
  - Address bit (MSB)
  - Idle line
- Muting function for multiprocessor configurations
- Separate enable bits for Transmitter and Receiver
- Three error detection flags:
  - Overrun error
  - Noise error
  - Frame error
- Five interrupt sources with flags:
  - Transmit data register empty
  - Transmission complete
  - Receive data register full
  - Idle line received
  - Overrun error detected

### 5.5.3 General Description

The interface is externally connected to another device by two pins (see Figure 1):

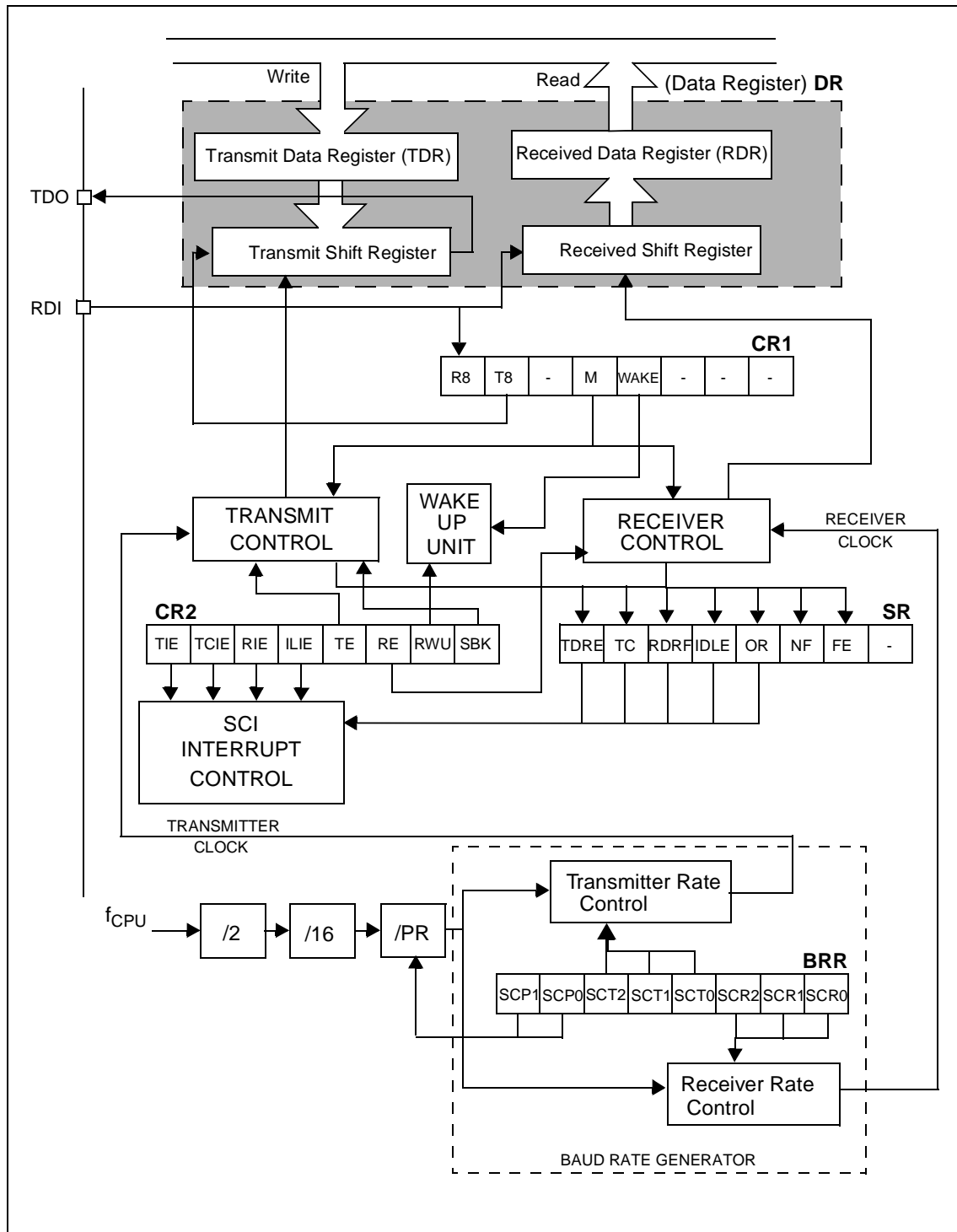
- TDO: Transmit Data Output. When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TDO pin is at high level.
- RDI: Receive Data Input is the serial data input. Oversampling techniques are used for data recovery by discriminating between valid incoming data and noise.

Through this pins, serial data is transmitted and received as frames comprising:

- An Idle Line prior to transmission or reception
- A start bit
- A data word (8 or 9 bits) least significant bit first
- A Stop bit indicating that the frame is complete.



**SERIAL COMMUNICATIONS INTERFACE (Cont'd)**  
**Figure 34. SCI Block Diagram**



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

**5.5.4 Functional Description**

The block diagram of the Serial Control Interface, is shown in Figure 1. It contains 4 dedicated registers:

- Two control registers (CR1 & CR2)
- A status register (SR)
- A baud rate register (BRR)

Refer to the register descriptions in Section 0.1.7 for the definitions of each bit.

**5.5.4.1 Serial Data Format**

Word length may be selected as being either 8 or 9 bits by programming the M bit in the CR1 register (see Figure 1).

The TDO pin is in low state during the start bit.

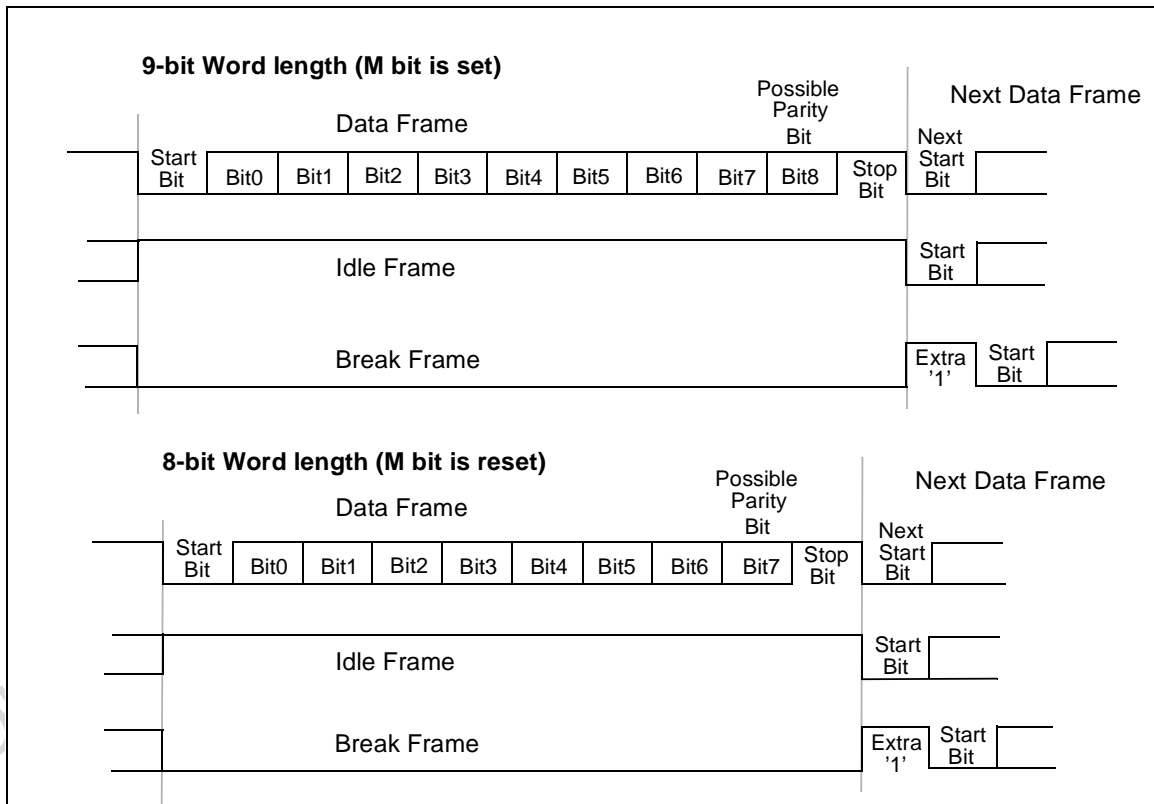
The TDO pin is in high state during the stop bit.

An Idle character is interpreted as an entire frame of "1"s followed by the start bit of the next frame which contains data.

A Break character is interpreted on receiving "0"s for some multiple of the frame period. At the end of the last break frame the transmitter inserts an extra "1" bit to acknowledge the start bit.

Transmission and reception are driven by their own baud rate generator.

**Figure 35. Word Length Programming**





## SERIAL COMMUNICATIONS INTERFACE (Cont'd)

### 5.5.4.2 Transmitter

The transmitter can send data words of either 8 or 9 bits depending on the M bit status. When the M bit is set, word length is 9 bits and the 9th bit (the MSB) has to be stored in the T8 bit in the CR1 register.

#### Character Transmission

During an SCI transmission, data shifts out least significant bit first on the TDO pin. In this mode, the DR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see Figure 1).

#### Procedure

- Select the M bit to define the word length.
- Select the desired baud rate using the BRR register.
- Set the TE bit to assign the TDO pin to the alternate function and to send a idle frame as first transmission.
- Access the SR register and write the data to send in the DR register (this sequence clears the TDRE bit). Repeat this sequence for each data to be transmitted.

The following software sequence is always to clear the TDRE bit:

1. An access to the SR register
2. A write to the DR register

The TDRE bit is set by hardware and it indicates that:

- The TDR register is empty.
- The data transfer is beginning.
- The next data can be written in the DR register without overwriting the previous data.

This flag generates an interrupt if the TIE bit is set and the I bit is cleared in the CC register.

When a transmission is taking place, a write instruction to the DR register stores the data in the TDR register which is copied in the shift register at the end of the current transmission.

When no transmission is taking place, a write instruction to the DR register places the data directly in the shift register, the data transmission starts, and the TDRE bit is immediately set.

When a frame transmission is complete (after the stop bit or after the break frame) the TC bit is set and an interrupt is generated if the TCIE is set and the I bit is cleared in the CC register.

The following software sequence is always to clear the TC bit:

1. An access to the SR register
2. A write to the DR register

**Note:** The TDRE and TC bits are cleared by the same software sequence.

#### Break Characters

Setting the SBK bit loads the shift register with a break character. The break frame length depends on the M bit (see Figure 2).

As long as the SBK bit is set, the SCI sends break frames to the TDO pin. After clearing this bit by software, the SCI inserts a logic 1 bit at the end of the last break frame to guarantee the recognition of the start bit of the next frame.

#### Idle Characters

Setting the TE bit drives the SCI to send an idle frame before the first data frame.

Clearing and then setting the TE bit during a transmission sends an idle frame after the current word.

**Note:** Resetting and setting the TE bit causes the data in the TDR register to be lost. Therefore the best time to toggle the TE bit is when the TDRE bit is set, i.e. before writing the next byte in the DR.

**SERIAL COMMUNICATIONS INTERFACE (Cont'd)****5.5.4.3 Receiver**

The SCI can receive data words of either 8 or 9 bits. When the M bit is set, word length is 9 bits and the MSB is stored in the R8 bit in the CR1 register.

**Character reception**

During a SCI reception, data shifts in least significant bit first through the RDI pin. In this mode, the DR register consists of a buffer (RDR) between the internal bus and the received shift register (see Figure 1).

**Procedure**

- Select the M bit to define the word length.
- Select the desired baud rate using the BRR register.
- Set the RE bit to enable the receiver to begin searching for a start bit.

When a character is received:

- The RDRF bit is set. It indicates that the content of the shift register is transferred to the RDR.
- An interrupt is generated if the RIE bit is set and the I bit is cleared in the CC register.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.

Clearing the RDRF bit is performed by the following software sequence done by:

1. An access to the SR register
2. A read to the DR register.

The RDRF bit must be cleared before the end of the reception of the next character to avoid an overrun error.

**Break Character**

When a break character is received, the SCI handles it as a framing error.

**Idle Character**

When an idle frame is detected, there is the same procedure as a data received character plus an interrupt if the ILIE bit is set and the I bit is cleared in the CC register.

**Overrun Error**

An overrun error occurs when a character is received when RDRF has not been reset. Data can not be transferred from the shift register to the TDR register as long as the RDRF bit is not cleared.

When an overrun error occurs:

- The OR bit is set.
- The RDR content will not be lost.
- The shift register will be overwritten.
- An interrupt is generated if the RIE bit is set and the I bit is cleared in the CC register.

The OR bit is reset by an access to the SR register followed by a DR register read operation.

**Noise Error**

Oversampling techniques are used for data recovery by discriminating between valid incoming data and noise.

When noise is detected in a frame:

- The NF is set at the rising edge of the RDRF bit.
- Data is transferred from the Shift register to the DR register.
- No interrupt is generated. However this bit rises at the same time as the RDRF bit which itself generates an interrupt.

The NF bit is reset by a SR register read operation followed by a DR register read operation.

**Framing Error**

A framing error is detected when:

- The stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.
- A break is received.

When the framing error is detected:

- The FE bit is set by hardware
- Data is transferred from the Shift register to the DR register.
- No interrupt is generated. However this bit rises at the same time as the RDRF bit which itself generates an interrupt.

The FE bit is reset by a SR register read operation followed by a DR register read operation.

## SERIAL COMMUNICATIONS INTERFACE (Cont'd)

### 5.5.4.4 Baud Rate Generation

The baud rates for the receiver and transmitter (Rx and Tx) are set independently and calculated as follows:

$$Tx = \frac{f_{CPU}}{(32 \cdot PR) \cdot TR} \quad Rx = \frac{f_{CPU}}{(32 \cdot PR) \cdot RR}$$

with:

PR = 1, 3, 4 or 13 (see SCP0 & SCP1 bits)

TR = 1, 2, 4, 8, 16, 32, 64, 128

(see SCT0, SCT1 & SCT2 bits)

RR = 1, 2, 4, 8, 16, 32, 64, 128

(see SCR0, SCR1 & SCR2 bits)

All these bits are in the BRR register.

**Example:** If  $f_{CPU}$  is 8 MHz and if PR=13 and TR=RR=1, the transmit and receive baud rates are 19200 bauds.

**Note:** The baud rate registers MUST NOT be changed while the transmitter or the receiver is enabled.

### 5.5.4.5 Receiver Muting and Wake-up Feature

In multiprocessor configurations it is often desirable that only the intended message recipient

should actively receive the full message contents, thus reducing redundant SCI service overhead for all non addressed receivers.

The non addressed devices may be placed in sleep mode by means of the muting function.

Setting the RWU bit by software puts the SCI in sleep mode:

All the reception status bits can not be set.

All the receive interrupt are inhibited.

A muted receiver may be awakened by one of the following two ways:

- by Idle Line detection if the WAKE bit is reset,
- by Address Mark detection if the WAKE bit is set.

The Receiver wakes-up by Idle Line detection when the Receive line has recognised an Idle Frame. Then the RWU bit is reset by hardware but the IDLE bit is not set.

The Receiver wakes-up by Address Mark detection when it received a "1" as the most significant bit of a word, thus indicating that the message is an address. The reception of this particular word wakes up the receiver, resets the RWU bit and sets the RDRF bit, which allows the receiver to receive this word normally and to use it as an address word.

## SERIAL COMMUNICATIONS INTERFACE (Cont'd)

## 5.5.5 Low Power Modes

Mode	Description
WAIT	No effect on SCI. SCI interrupts exit from Wait mode.
HALT	SCI registers are frozen. In Halt mode, the SCI stops transmitting/receiving until Halt mode is exited.

## 5.5.6 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
Transmit Data Register Empty	TDRE	TIE	Yes	No
Transmission Complete	TC	TCIE	Yes	No
Received Data Ready to be Read	RDRF	RIE	Yes	No
Overrun Error Detected	OR		Yes	No
Idle Line Detected	IDLE	ILIE	Yes	No

The SCI interrupt events are connected to the same interrupt vector (see Interrupts chapter).

These events generate an interrupt if the corresponding Enable Control Bit is set and the inter-

rupt mask in the CC register is reset (RIM instruction).

## SERIAL COMMUNICATIONS INTERFACE (Cont'd)

## 5.5.7 Register Description

## STATUS REGISTER (SR)

Read Only

Reset Value: 1100 0000 (C0h)

7							0
TDRE	TC	RDRF	IDLE	OR	NF	FE	0

Bit 7 = **TDRE** *Transmit data register empty.*

This bit is set by hardware when the content of the TDR register has been transferred into the shift register. An interrupt is generated if TIE =1 in the CR2 register. It is cleared by a software sequence (an access to the SR register followed by a write to the DR register).

0: Data is not transferred to the shift register

1: Data is transferred to the shift register

**Note:** data will not be transferred to the shift register as long as the TDRE bit is not reset.

Bit 6 = **TC** *Transmission complete.*

This bit is set by hardware when transmission of a frame containing Data, a Preamble or a Break is complete. An interrupt is generated if TCIE=1 in the CR2 register. It is cleared by a software sequence (an access to the SR register followed by a write to the DR register).

0: Transmission is not complete

1: Transmission is complete

Bit 5 = **RDRF** *Received data ready flag.*

This bit is set by hardware when the content of the RDR register has been transferred into the DR register. An interrupt is generated if RIE=1 in the CR2 register. It is cleared by hardware when RE=0 or by a software sequence (an access to the SR register followed by a read to the DR register).

0: Data is not received

1: Received data is ready to be read

Bit 4 = **IDLE** *Idle line detect.*

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if ILIE=1 in the CR2 register. It is cleared by hardware when RE=0 by a software sequence (an access to the SR register followed by a read to the DR register).

0: No Idle Line is detected

1: Idle Line is detected

**Note:** The IDLE bit will not be set again until the RDRF bit has been set itself (i.e. a new idle line occurs). This bit is not set by an idle line when the receiver wakes up from wake-up mode.

Bit 3 = **OR** *Overrun error.*

This bit is set by hardware when the word currently being received in the shift register is ready to be transferred into the RDR register while RDRF=1. An interrupt is generated if RIE=1 in the CR2 register. It is cleared by hardware when RE=0 by a software sequence (an access to the SR register followed by a read to the DR register).

0: No Overrun error

1: Overrun error is detected

**Note:** When this bit is set the RDR register content will not be lost but the shift register will be overwritten.

Bit 2 = **NF** *Noise flag.*

This bit is set by hardware when noise is detected on a received frame. It is cleared by hardware when RE=0 by a software sequence (an access to the SR register followed by a read to the DR register).

0: No noise is detected

1: Noise is detected

**Note:** This bit does not generate interrupt as it appears at the same time as the RDRF bit which itself generates an interrupt.

Bit 1 = **FE** *Framing error.*

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by hardware when RE=0 by a software sequence (an access to the SR register followed by a read to the DR register).

0: No Framing error is detected

1: Framing error or break character is detected

**Note:** This bit does not generate interrupt as it appears at the same time as the RDRF bit which itself generates an interrupt. If the word currently being transferred causes both frame error and overrun error, it will be transferred and only the OR bit will be set.

Bit 0 = Reserved, forced by hardware to 0.

**SERIAL COMMUNICATIONS INTERFACE (Cont'd)****CONTROL REGISTER 1 (CR1)**

Read/Write

Reset Value: Undefined

7							0
R8	T8	0	M	WAKE	0	0	0

**Bit 7 = R8 Receive data bit 8.**

This bit is used to store the 9th bit of the received word when M=1.

**Bit 6 = T8 Transmit data bit 8.**

This bit is used to store the 9th bit of the transmitted word when M=1.

**Bit 5 = Reserved, forced by hardware to 0.****Bit 4 = M Word length.**

This bit determines the data length. It is set or cleared by software.

0: 1 Start bit, 8 Data bits, 1 Stop bit

1: 1 Start bit, 9 Data bits, 1 Stop bit

**Bit 3 = WAKE Wake-Up method.**

This bit determines the SCI Wake-Up method, it is set or cleared by software.

0: Idle Line

1: Address Mark

**Bits 2:0 = Reserved, forced by hardware to 0.****CONTROL REGISTER 2 (CR2)**

Read/Write

Reset Value: 0000 0000 (00h)

7							0
TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK

**Bit 7 = TIE Transmitter interrupt enable.**

This bit is set and cleared by software.

0: interrupt is inhibited

1: An SCI interrupt is generated whenever TDRE=1 in the SR register.

**Bit 6 = TCIE Transmission complete interrupt enable**

This bit is set and cleared by software.

0: interrupt is inhibited

1: An SCI interrupt is generated whenever TC=1 in the SR register

**Bit 5 = RIE Receiver interrupt enable.**

This bit is set and cleared by software.

0: interrupt is inhibited

1: An SCI interrupt is generated whenever OR=1 or RDRF=1 in the SR register

**Bit 4 = ILIE Idle line interrupt enable.**

This bit is set and cleared by software.

0: interrupt is inhibited

1: An SCI interrupt is generated whenever IDLE=1 in the SR register.

**Bit 3 = TE Transmitter enable.**

This bit enables the transmitter and assigns the TDO pin to the alternate function. It is set and cleared by software.

0: Transmitter is disabled, the TDO pin is back to the I/O port configuration.

1: Transmitter is enabled

**Note:** During transmission, a "0" pulse on the TE bit ("0" followed by "1") sends a preamble after the current word.**Bit 2 = RE Receiver enable.**

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled, it resets the RDRF, IDLE, OR, NF and FE bits of the SR register.

1: Receiver is enabled and begins searching for a start bit.

**Bit 1 = RWU Receiver wake-up.**

This bit determines if the SCI is in mute mode or not. It is set and cleared by software and can be cleared by hardware when a wake-up sequence is recognized.

0: Receiver in active mode

1: Receiver in mute mode

**Bit 0 = SBK Send break.**

This bit set is used to send break characters. It is set and cleared by software.

0: No break character is transmitted

1: Break characters are transmitted

**Note:** If the SBK bit is set to "1" and then to "0", the transmitter will send a BREAK word at the end of the current word.

**SERIAL COMMUNICATIONS INTERFACE (Cont'd)****DATA REGISTER (DR)**

Read/Write

Reset Value: Undefined

Contains the Received or Transmitted data character, depending on whether it is read from or written to.

7							0
DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0

The Data register performs a double function (read and write) since it is composed of two registers, one for transmission (TDR) and one for reception (RDR).

The TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 1).

The RDR register provides the parallel interface between the input shift register and the internal bus (see Figure 1).

**BAUD RATE REGISTER (BRR)**

Read/Write

Reset Value: 00xx xxxx (XXh)

7							0
SCP1	SCP0	SCT2	SCT1	SCT0	SCR2	SCR1	SCR0

Bits 7:6= **SCP[1:0]** First SCI Prescaler

These 2 prescaling bits allow several standard clock division ranges:

PR Prescaling Factor	SCP1	SCP0
1	0	0
3	0	1
4	1	0
13	1	1

Bits 5:3 = **SCT[2:0]** SCI Transmitter rate divisor

These 3 bits, in conjunction with the SCP1 & SCP0 bits, define the total division applied to the bus clock to yield the transmit rate clock in conventional Baud Rate Generator mode.

TR Dividing Factor	SCT2	SCT1	SCT0
1	0	0	0
2	0	0	1
4	0	1	0
8	0	1	1
16	1	0	0
32	1	0	1
64	1	1	0
128	1	1	1

Bits 2:0 = **SCR[2:0]** SCI Receiver rate divisor.

These 3 bits, in conjunction with the SCP1 & SCP0 bits, define the total division applied to the bus clock to yield the receive rate clock in conventional Baud Rate Generator mode.

RR Dividing Factor	SCR2	SCR1	SCR0
1	0	0	0
2	0	0	1
4	0	1	0
8	0	1	1
16	1	0	0
32	1	0	1
64	1	1	0
128	1	1	1

## SERIAL COMMUNICATIONS INTERFACE (Cont'd)

Table 18. SCI Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
20	<b>SR</b> Reset Value	TDRE 1	TC 1	RDRF 0	IDLE 0	OR 0	NF 0	FE 0	0 0
21	<b>DR</b> Reset Value	DR7 x	DR6 x	DR5 x	DR4 x	DR3 x	DR2 x	DR1 x	DR0 x
22	<b>BRR</b> Reset Value	SCP1 0	SCP0 0	SCT2 x	SCT1 x	SCT0 x	SCR2 x	SCR1 x	SCR0 x
23	<b>CR1</b> Reset Value	R8 x	T8 x	0 0	M x	WAKE x	0 0	0 0	0 0
24	<b>CR2</b> Reset Value	TIE 0	TCIE 0	RIE 0	ILIE 0	TE 0	RE 0	RWU 0	SBK 0



## 5.6 USB INTERFACE (USB)

### 5.6.1 Introduction

The USB Interface implements a low-speed function interface between the USB and the ST7 microcontroller. It is a highly integrated circuit which includes the transceiver, 3.3 voltage regulator, SIE and DMA. No external components are needed apart from the external pull-up on USBDM for low speed recognition by the USB host. The use of DMA architecture allows the endpoint definition to be completely flexible. Endpoints can be configured by software as in or out.

### 5.6.2 Main Features

- USB Specification Version 1.1 Compliant
- Supports Low-Speed USB Protocol
- Two or Three Endpoints (including default one) depending on the device (see device feature list and register map)
- CRC generation/checking, NRZI encoding/decoding and bit-stuffing
- USB Suspend/Resume operations
- DMA Data transfers
- On-Chip 3.3V Regulator
- On-Chip USB Transceiver

### 5.6.3 Functional Description

The block diagram in Figure 1, gives an overview of the USB interface hardware.

For general information on the USB, refer to the “Universal Serial Bus Specifications” document available at <http://www.usb.org>.

### Serial Interface Engine

The SIE (Serial Interface Engine) interfaces with the USB, via the transceiver.

The SIE processes tokens, handles data transmission/reception, and handshaking as required by the USB standard. It also performs frame formatting, including CRC generation and checking.

### Endpoints

The Endpoint registers indicate if the microcontroller is ready to transmit/receive, and how many bytes need to be transmitted.

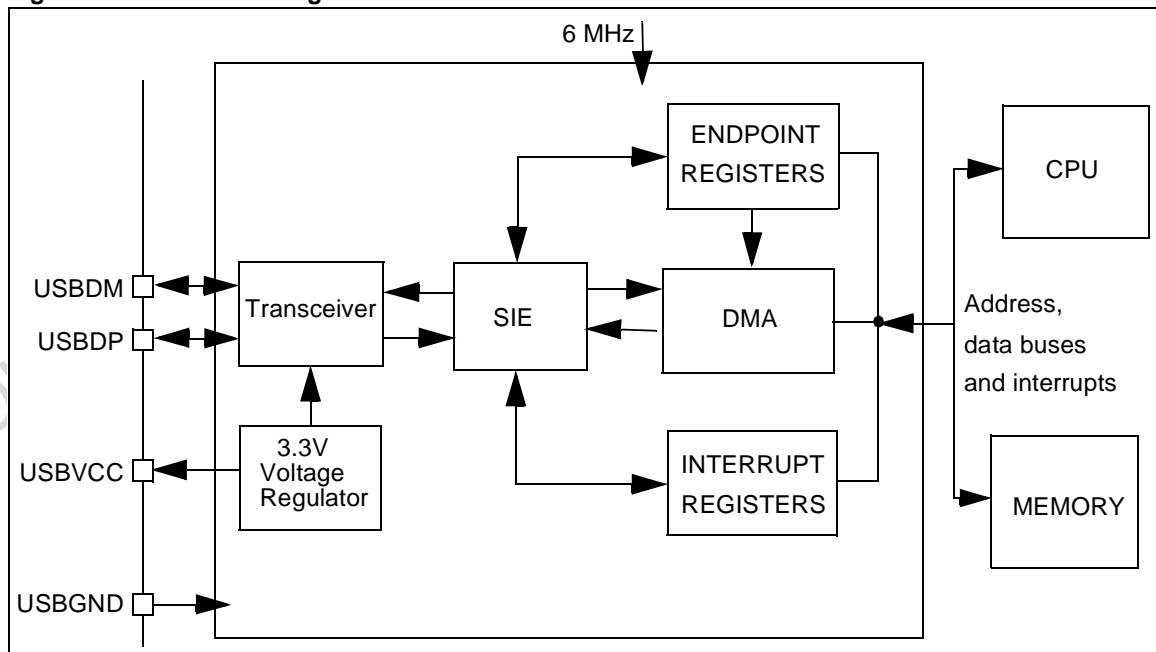
### DMA

When a token for a valid Endpoint is recognized by the USB interface, the related data transfer takes place, using DMA. At the end of the transaction, an interrupt is generated.

### Interrupts

By reading the Interrupt Status register, application software can know which USB event has occurred.

Figure 36. USB Block Diagram



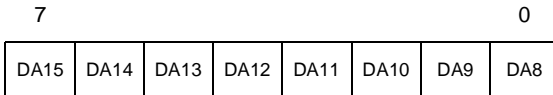
USB INTERFACE (Cont'd)

5.6.4 Register Description

DMA ADDRESS REGISTER (DMAR)

Read / Write

Reset Value: Undefined

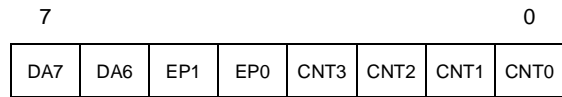


Bits 7:0=**DA[15:8]** DMA address bits 15-8. Software must write the start address of the DMA memory area whose most significant bits are given by DA15-DA6. The remaining 6 address bits are set by hardware. See the description of the IDR register and Figure 2.

INTERRUPT/DMA REGISTER (IDR)

Read / Write

Reset Value: xxxx 0000 (x0h)



Bits 7:6 = **DA[7:6]** DMA address bits 7-6. Software must reset these bits. See the description of the DMAR register and Figure 2.

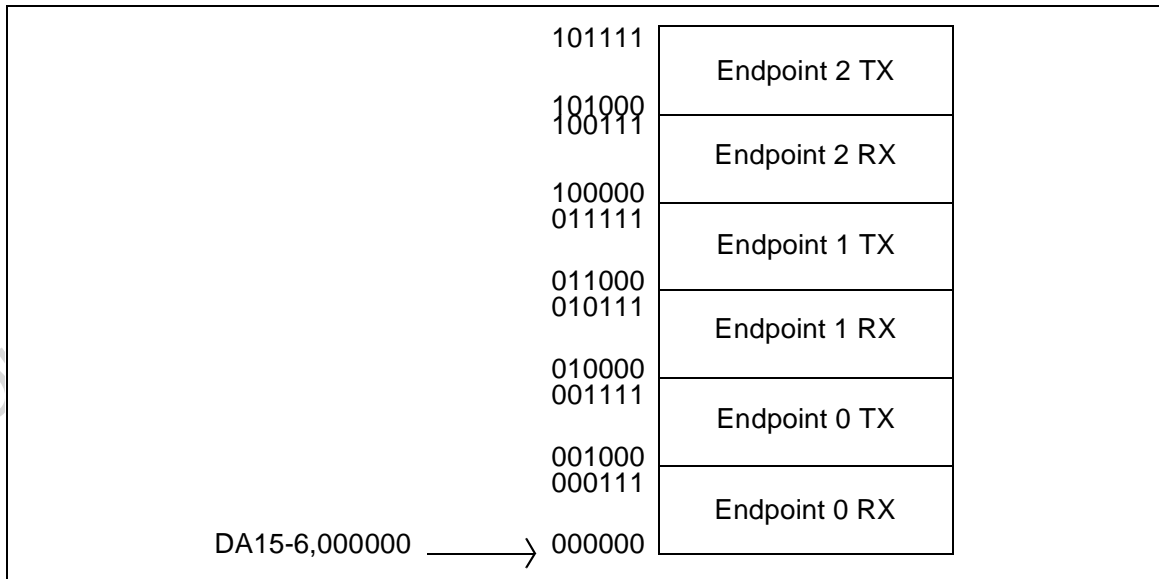
Bits 5:4 = **EP[1:0]** Endpoint number (read-only). These bits identify the endpoint which required attention.  
 00: Endpoint 0  
 01: Endpoint 1  
 10: Endpoint 2

When a CTR interrupt occurs (see register ISTR) the software should read the EP bits to identify the endpoint which has sent or received a packet.

Bits 3:0 = **CNT[3:0]** Byte count (read only). This field shows how many data bytes have been received during the last data reception.

**Note:** Not valid for data transmission.

Figure 37. DMA Buffers



**USB INTERFACE (Cont'd)****PID REGISTER (PIDR)**

Read only

Reset Value: xx00 0000 (x0h)

7							0
TP3	TP2	0	0	0	RX_SEZ	RXD	0

Bits 7:6 = **TP[3:2]** *Token PID bits 3 & 2.*USB token PIDs are encoded in four bits. **TP[3:2]** correspond to the variable token PID bits 3 & 2.**Note:** PID bits 1 & 0 have a fixed value of 01.

When a CTR interrupt occurs (see register ISTR) the software should read the TP3 and TP2 bits to retrieve the PID name of the token received.

The USB standard defines TP bits as:

TP3	TP2	PID Name
0	0	OUT
1	0	IN
1	1	SETUP

Bits 5:3 Reserved. Forced by hardware to 0.

Bit 2 = **RX\_SEZ** *Received single-ended zero*

This bit indicates the status of the RX\_SEZ transceiver output.

0: No SE0 (single-ended zero) state

1: USB lines are in SE0 (single-ended zero) state

Bit 1 = **RXD** *Received data*

0: No K-state

1: USB lines are in K-state

This bit indicates the status of the RXD transceiver output (differential receiver output).

**Note:** If the environment is noisy, the RX\_SEZ and RXD bits can be used to secure the application. By interpreting the status, software can distinguish a valid End Suspend event from a spurious wake-up due to noise on the external USB line. A valid End Suspend is followed by a Resume or Reset sequence. A Resume is indicated by RXD=1, a Reset is indicated by RX\_SEZ=1.

Bit 0 = Reserved. Forced by hardware to 0.

**INTERRUPT STATUS REGISTER (ISTR)**

Read / Write

Reset Value: 0000 0000 (00h)

7							0
SUSP	DOVR	CTR	ERR	IOVR	ESUSP	RESET	SOF

When an interrupt occurs these bits are set by hardware. Software must read them to determine the interrupt type and clear them after servicing.

**Note:** These bits cannot be set by software.Bit 7 = **SUSP** *Suspend mode request.*

This bit is set by hardware when a constant idle state is present on the bus line for more than 3 ms, indicating a suspend mode request from the USB bus. The suspend request check is active immediately after each USB reset event and its disabled by hardware when suspend mode is forced (FSUSP bit of CTLR register) until the end of resume sequence.

Bit 6 = **DOVR** *DMA over/underrun.*

This bit is set by hardware if the ST7 processor can't answer a DMA request in time.

0: No over/underrun detected

1: Over/underrun detected

Bit 5 = **CTR** *Correct Transfer.* This bit is set by hardware when a correct transfer operation is performed. The type of transfer can be determined by looking at bits TP3-TP2 in register PIDR. The Endpoint on which the transfer was made is identified by bits EP1-EP0 in register IDR.

0: No Correct Transfer detected

1: Correct Transfer detected

**Note:** A transfer where the device sent a NAK or STALL handshake is considered not correct (the host only sends ACK handshakes). A transfer is considered correct if there are no errors in the PID and CRC fields, if the DATA0/DATA1 PID is sent as expected, if there were no data overruns, bit stuffing or framing errors.

Bit 4 = **ERR** *Error.*

This bit is set by hardware whenever one of the errors listed below has occurred:

0: No error detected

1: Timeout, CRC, bit stuffing or nonstandard framing error detected

**USB INTERFACE (Cont'd)**

Bit 3 = **IOVR** *Interrupt overrun.*

This bit is set when hardware tries to set ERR, or SOF before they have been cleared by software.

- 0: No overrun detected
- 1: Overrun detected

Bit 2 = **ESUSP** *End suspend mode.*

This bit is set by hardware when, during suspend mode, activity is detected that wakes the USB interface up from suspend mode.

This interrupt is serviced by a specific vector, in order to wake up the ST7 from HALT mode.

- 0: No End Suspend detected
- 1: End Suspend detected

Bit 1 = **RESET** *USB reset.*

This bit is set by hardware when the USB reset sequence is detected on the bus.

- 0: No USB reset signal detected
- 1: USB reset signal detected

**Note:** The DADDR, EP0RA, EP0RB, EP1RA, EP1RB, EP2RA and EP2RB registers are reset by a USB reset.

Bit 0 = **SOF** *Start of frame.*

This bit is set by hardware when a low-speed SOF indication (keep-alive strobe) is seen on the USB bus. It is also issued at the end of a resume sequence.

- 0: No SOF signal detected
- 1: SOF signal detected

**Note:** To avoid spurious clearing of some bits, it is recommended to clear them using a load instruction where all bits which must not be altered are set, and all bits to be cleared are reset. Avoid read-modify-write instructions like AND , XOR..

**INTERRUPT MASK REGISTER (IMR)**

Read / Write

Reset Value: 0000 0000 (00h)

7								0
SUS PM	DOV RM	CTR M	ERR M	IOVR M	ESU SPM	RES ETM	SOF M	

Bits 7:0 = These bits are mask bits for all interrupt condition bits included in the ISTR. Whenever one of the IMR bits is set, if the corresponding ISTR bit is set, and the I bit in the CC register is cleared, an interrupt request is generated. For an explanation

of each bit, please refer to the corresponding bit description in ISTR.

**CONTROL REGISTER (CTRL)**

Read / Write

Reset Value: 0000 0110 (06h)

7						0	
0	0	0	0	RESUME	PDWN	FSUSP	FRES

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

Bit 3 = **RESUME** *Resume.*

This bit is set by software to wake-up the Host when the ST7 is in suspend mode.

- 0: Resume signal not forced
- 1: Resume signal forced on the USB bus.

Software should clear this bit after the appropriate delay.

Bit 2 = **PDWN** *Power down.*

This bit is set by software to turn off the 3.3V on-chip voltage regulator that supplies the external pull-up resistor and the transceiver.

- 0: Voltage regulator on
- 1: Voltage regulator off

**Note:** After turning on the voltage regulator, software should allow at least 3 μs for stabilisation of the power supply before using the USB interface.

Bit 1 = **FSUSP** *Force suspend mode.*

This bit is set by software to enter Suspend mode. The ST7 should also be halted allowing at least 600 ns before issuing the HALT instruction.

- 0: Suspend mode inactive
- 1: Suspend mode active

When the hardware detects USB activity, it resets this bit (it can also be reset by software).

Bit 0 = **FRES** *Force reset.*

This bit is set by software to force a reset of the USB interface, just as if a RESET sequence came from the USB.

- 0: Reset not forced
- 1: USB interface reset forced.

The USB is held in RESET state until software clears this bit, at which point a "USB-RESET" interrupt will be generated if enabled.

**USB INTERFACE (Cont'd)****DEVICE ADDRESS REGISTER (DADDR)**

Read / Write

Reset Value: 0000 0000 (00h)

7							0
0	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0

Bit 7 = Reserved. Forced by hardware to 0.

Bits 6:0 = **ADD[6:0]** Device address, 7 bits.

Software must write into this register the address sent by the host during enumeration.

**Note:** This register is also reset when a USB reset is received from the USB bus or forced through bit FRES in the CTLR register.

**ENDPOINT n REGISTER A (EPnRA)**

Read / Write

Reset Value: 0000 xxxx (0xh)

7							0
ST_OUT	DTOG_TX	STAT_TX1	STAT_TX0	TBC 3	TBC 2	TBC 1	TBC 0

These registers (**EP0RA**, **EP1RA** and **EP2RA**) are used for controlling data transmission. They are also reset by the USB bus reset.

**Note:** Endpoint 2 and the EP2RA register are not available on some devices (see device feature list and register map).

Bit 7 = **ST\_OUT** Status out.

This bit is set by software to indicate that a status out packet is expected: in this case, all nonzero OUT data transfers on the endpoint are STALLED instead of being ACKed. When ST\_OUT is reset, OUT transactions can have any number of bytes, as needed.

Bit 6 = **DTOG\_TX** Data Toggle, for transmission transfers.

It contains the required value of the toggle bit (0=DATA0, 1=DATA1) for the next transmitted data packet. This bit is set by hardware at the reception of a SETUP PID. DTOG\_TX toggles only when the transmitter has received the ACK signal from the USB host. DTOG\_TX and also DTOG\_RX (see EPnRB) are normally updated by hardware, at the receipt of a relevant PID. They can be also written by software.

Bits 5:4 = **STAT\_TX[1:0]** Status bits, for transmission transfers.

These bits contain the information about the endpoint status, which are listed below:

STAT_TX1	STAT_TX0	Meaning
0	0	<b>DISABLED:</b> transmission transfers cannot be executed.
0	1	<b>STALL:</b> the endpoint is stalled and all transmission requests result in a STALL handshake.
1	0	<b>NAK:</b> the endpoint is naked and all transmission requests result in a NAK handshake.
1	1	<b>VALID:</b> this endpoint is enabled for transmission.

These bits are written by software. Hardware sets the STAT\_TX bits to NAK when a correct transfer has occurred (CTR=1) related to a IN or SETUP transaction addressed to this endpoint; this allows the software to prepare the next set of data to be transmitted.

Bits 3:0 = **TBC[3:0]** Transmit byte count for Endpoint n.

Before transmission, after filling the transmit buffer, software must write in the TBC field the transmit packet size expressed in bytes (in the range 0-8).

**USB INTERFACE (Cont'd)****ENDPOINT n REGISTER B (EPnRB)**

Read / Write

Reset Value: 0000 xxxx (0xh)

7							0
CTRL	DTOG_RX	STAT_RX1	STAT_RX0	EA3	EA2	EA1	EA0

These registers (**EP1RB** and **EP2RB**) are used for controlling data reception on Endpoints 1 and 2. They are also reset by the USB bus reset.

**Note:** Endpoint 2 and the EP2RB register are not available on some devices (see device feature list and register map).

Bit 7 = **CTRL Control**.

This bit should be 0.

**Note:** If this bit is 1, the Endpoint is a control endpoint. (Endpoint 0 is always a control Endpoint, but it is possible to have more than one control Endpoint).

Bit 6 = **DTOG\_RX Data toggle, for reception transfers**.

It contains the expected value of the toggle bit (0=DATA0, 1=DATA1) for the next data packet. This bit is cleared by hardware in the first stage (Setup Stage) of a control transfer (SETUP transactions start always with DATA0 PID). The receiver toggles DTOG\_RX only if it receives a correct data packet and the packet's data PID matches the receiver sequence bit.

Bits 5:4 = **STAT\_RX [1:0] Status bits, for reception transfers**.

These bits contain the information about the endpoint status, which are listed below:

STAT_RX1	STAT_RX0	Meaning
0	0	<b>DISABLED:</b> reception transfers cannot be executed.
0	1	<b>STALL:</b> the endpoint is stalled and all reception requests result in a STALL handshake.

STAT_RX1	STAT_RX0	Meaning
1	0	<b>NAK:</b> the endpoint is nacked and all reception requests result in a NAK handshake.
1	1	<b>VALID:</b> this endpoint is enabled for reception.

These bits are written by software. Hardware sets the STAT\_RX bits to NAK when a correct transfer has occurred (CTR=1) related to an OUT or SETUP transaction addressed to this endpoint, so the software has the time to elaborate the received data before acknowledging a new transaction.

Bits 3:0 = **EA[3:0] Endpoint address**.

Software must write in this field the 4-bit address used to identify the transactions directed to this endpoint. Usually EP1RB contains "0001" and EP2RB contains "0010".

**ENDPOINT 0 REGISTER B (EP0RB)**

Read / Write

Reset Value: 1000 0000 (80h)

7							0
1	DTOG_RX	STAT_RX1	STAT_RX0	0	0	0	0

This register is used for controlling data reception on Endpoint 0. It is also reset by the USB bus reset.

Bit 7 = Forced by hardware to 1.

Bits 6:4 = Refer to the EPnRB register for a description of these bits.

Bits 3:0 = Forced by hardware to 0.

## USB INTERFACE (Cont'd)

### 5.6.5 Programming Considerations

The interaction between the USB interface and the application program is described below. Apart from system reset, action is always initiated by the USB interface, driven by one of the USB events associated with the Interrupt Status Register (ISTR) bits.

#### 5.6.5.1 Initializing the Registers

At system reset, the software must initialize all registers to enable the USB interface to properly generate interrupts and DMA requests.

1. Initialize the DMAR, IDR, and IMR registers (choice of enabled interrupts, address of DMA buffers). Refer the paragraph titled initializing the DMA Buffers.
2. Initialize the EP0RA and EP0RB registers to enable accesses to address 0 and endpoint 0 to support USB enumeration. Refer to the paragraph titled Endpoint Initialization.
3. When addresses are received through this channel, update the content of the DADDR.
4. If needed, write the endpoint numbers in the EA fields in the EP1RB and EP2RB register.

#### 5.6.5.2 Initializing DMA buffers

The DMA buffers are a contiguous zone of memory whose maximum size is 48 bytes. They can be placed anywhere in the memory space to enable the reception of messages. The 10 most significant bits of the start of this memory area are specified by bits DA15-DA6 in registers DMAR and IDR, the remaining bits are 0. The memory map is shown in Figure 2.

Each buffer is filled starting from the bottom (last 3 address bits=000) up.

#### 5.6.5.3 Endpoint Initialization

To be ready to receive:

Set STAT\_RX to VALID (11b) in EP0RB to enable reception.

To be ready to transmit:

1. Write the data in the DMA transmit buffer.
2. In register EPnRA, specify the number of bytes to be transmitted in the TBC field
3. Enable the endpoint by setting the STAT\_TX bits to VALID (11b) in EPnRA.

**Note:** Once transmission and/or reception are enabled, registers EPnRA and/or EPnRB (respectively) must not be modified by software, as the hardware can change their value on the fly.

When the operation is completed, they can be accessed again to enable a new operation.

#### 5.6.5.4 Interrupt Handling

##### Start of Frame (SOF)

The interrupt service routine may monitor the SOF events for a 1 ms synchronization event to the USB bus. This interrupt is generated at the end of a resume sequence and can also be used to detect this event.

##### USB Reset (RESET)

When this event occurs, the DADDR register is reset, and communication is disabled in all endpoint registers (the USB interface will not respond to any packet). Software is responsible for reenabling endpoint 0 within 10 ms of the end of reset. To do this, set the STAT\_RX bits in the EP0RB register to VALID.

##### Suspend (SUSP)

The CPU is warned about the lack of bus activity for more than 3 ms, which is a suspend request. The software should set the USB interface to suspend mode and execute an ST7 HALT instruction to meet the USB-specified power constraints.

##### End Suspend (ESUSP)

The CPU is alerted by activity on the USB, which causes an ESUSP interrupt. The ST7 automatically terminates HALT mode.

##### Correct Transfer (CTR)

1. When this event occurs, the hardware automatically sets the STAT\_TX or STAT\_RX to NAK.
 

**Note:** Every valid endpoint is NAKed until software clears the CTR bit in the ISTR register, independently of the endpoint number addressed by the transfer which generated the CTR interrupt.

**Note:** If the event triggering the CTR interrupt is a SETUP transaction, both STAT\_TX and STAT\_RX are set to NAK.
2. Read the PIDR to obtain the token and the IDR to get the endpoint number related to the last transfer.
 

**Note:** When a CTR interrupt occurs, the TP3-TP2 bits in the PIDR register and EP1-EP0 bits in the IDR register stay unchanged until the CTR bit in the ISTR register is cleared.
3. Clear the CTR bit in the ISTR register.

## USB INTERFACE (Cont'd)

Table 19. USB Register Map and Reset Values

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
25	PIDR Reset Value	TP3 x	TP2 x	0 0	0 0	0 0	RX_SEZ 0	RXD 0	0 0
26	DMAR Reset Value	DA15 x	DA14 x	DA13 x	DA12 x	DA11 x	DA10 x	DA9 x	DA8 x
27	IDR Reset Value	DA7 x	DA6 x	EP1 x	EP0 x	CNT3 0	CNT2 0	CNT1 0	CNT0 0
28	ISTR Reset Value	SUSP 0	DOVR 0	CTR 0	ERR 0	IOVR 0	ESUSP 0	RESET 0	SOF 0
29	IMR Reset Value	SUSPM 0	DOVRM 0	CTRM 0	ERRM 0	IOVRM 0	ESUSPM 0	RESETM 0	SOFM 0
2A	CTLR Reset Value	0 0	0 0	0 0	0 0	RESUME 0	PDWN 1	FSUSP 1	FRES 0
2B	DADDR Reset Value	0 0	ADD6 0	ADD5 0	ADD4 0	ADD3 0	ADD2 0	ADD1 0	ADD0 0
2C	EP0RA Reset Value	ST_OUT 0	DTOG_TX 0	STAT_TX1 0	STAT_TX0 0	TBC3 x	TBC2 x	TBC1 x	TBC0 x
2D	EP0RB Reset Value	1 1	DTOG_RX 0	STAT_RX1 0	STAT_RX0 0	0 0	0 0	0 0	0 0
2E	EP1RA Reset Value	ST_OUT 0	DTOG_TX 0	STAT_TX1 0	STAT_TX0 0	TBC3 x	TBC2 x	TBC1 x	TBC0 x
2F	EP1RB Reset Value	CTRL 0	DTOG_RX 0	STAT_RX1 0	STAT_RX0 0	EA3 x	EA2 x	EA1 x	EA0 x
30	EP2RA Reset Value	ST_OUT 0	DTOG_TX 0	STAT_TX1 0	STAT_TX0 0	TBC3 x	TBC2 x	TBC1 x	TBC0 x
31	EP2RB Reset Value	CTRL 0	DTOG_RX 0	STAT_RX1 0	STAT_RX0 0	EA3 x	EA2 x	EA1 x	EA0 x



## 5.7 I<sup>2</sup>C BUS INTERFACE (I<sup>2</sup>C)

### 5.7.1 Introduction

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

### 5.7.2 Main Features

- Parallel-bus/I<sup>2</sup>C protocol converter
- Multi-master capability
- 7-bit Addressing
- Transmitter/Receiver flag
- End-of-byte transmission flag
- Transfer problem detection

#### I<sup>2</sup>C Master Features:

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

#### I<sup>2</sup>C Slave Features:

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

### 5.7.3 General Description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa, using either an interrupt or polled

handshake. The interrupts are enabled or disabled by software. The interface is connected to the I<sup>2</sup>C bus by a data pin (SDAI) and by a clock pin (SCLI). It can be connected both with a standard I<sup>2</sup>C bus and a Fast I<sup>2</sup>C bus. This selection is made by software.

#### Mode Selection

The interface can operate in the four following modes:

- Slave transmitter/receiver
- Master transmitter/receiver

By default, it operates in slave mode.

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

#### Communication Flow

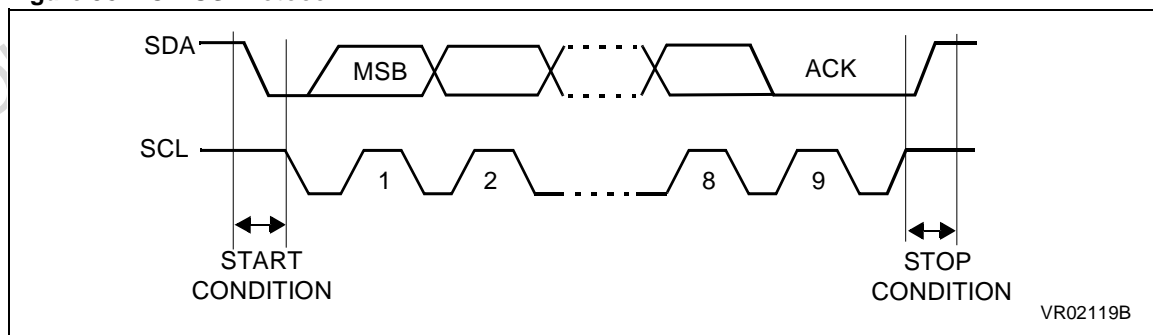
In Master mode, it initiates a data transfer and generates the clock signal. A serial data transfer always begins with a start condition and ends with a stop condition. Both start and stop conditions are generated in master mode by software.

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

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte following the start condition is the address byte; it is always transmitted in Master mode.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to Figure 1.

Figure 38. I<sup>2</sup>C BUS Protocol



**I<sup>2</sup>C BUS INTERFACE (Cont'd)**

The Acknowledge function may be enabled and disabled by software.

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

The speed of the I<sup>2</sup>C interface may be selected between Standard I<sup>2</sup>C (0-100 kHz) and Fast I<sup>2</sup>C (100-400 kHz).

**SDA/SCL Line Control**

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

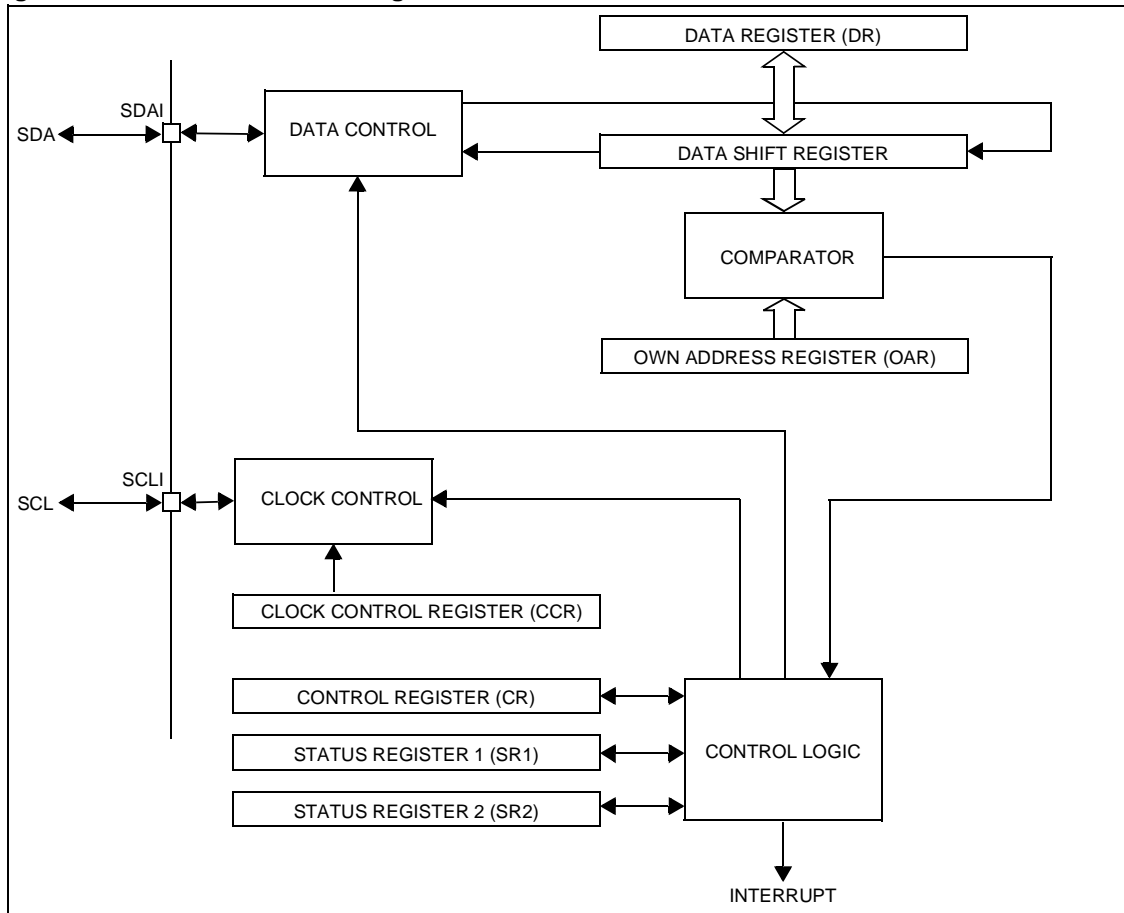
Receiver mode: the interface holds the clock line low after reception to wait for the microcontroller to read the byte in the Data Register.

The SCL frequency ( $F_{SCL}$ ) is controlled by a programmable clock divider which depends on the I<sup>2</sup>C bus mode.

When the I<sup>2</sup>C cell is enabled, the SDA and SCL ports must be configured as floating open-drain output or floating input. In this case, the value of the external pull-up resistor used depends on the application.

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

**Figure 39. I<sup>2</sup>C Interface Block Diagram**



## I<sup>2</sup>C BUS INTERFACE (Cont'd)

### 5.7.4 Functional Description

Refer to the CR, SR1 and SR2 registers in Section 0.1.7. for the bit definitions.

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

#### 5.7.4.1 Slave Mode

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

**Address not matched:** the interface ignores it and waits for another Start condition.

**Address matched:** the interface generates in sequence:

- An Acknowledge pulse is generated if the ACK bit is set.
- EVF and ADSL bits are set with an interrupt if the ITE bit is set.

Then the interface waits for a read of the SR1 register, **holding the SCL line low** (see Figure 3 Transfer sequencing EV1).

Next, software must read the DR register to determine from the least significant bit if the slave must enter Receiver or Transmitter mode.

#### Slave Receiver

Following the address reception and after SR1 register has been read, the slave receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

- An Acknowledge pulse is generated if the ACK bit is set
- EVF and BTF bits are set with an interrupt if the ITE bit is set.

Then the interface waits for a read of the SR1 register followed by a read of the DR register, **holding the SCL line low** (see Figure 3 Transfer sequencing EV2).

#### Slave Transmitter

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

The slave waits for a read of the SR1 register followed by a write in the DR register, **holding the SCL line low** (see Figure 3 Transfer sequencing EV3).

When the acknowledge pulse is received:

- The EVF and BTF bits are set by hardware with an interrupt if the ITE bit is set.

#### Closing Slave Communication

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

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

Then the interface waits for a read of the SR2 register (see Figure 3 Transfer sequencing EV4).

#### Error Cases

- **BERR:** Detection of a Stop or a Start condition during a byte transfer. In this case, the EVF and BERR bits are set with an interrupt if the ITE bit is set.

If it is a Stop condition, then the interface discards the data, released the lines and waits for another Start condition.

If it is a Start condition, then the interface discards the data and waits for the next slave address on the bus.

- **AF:** Detection of a non-acknowledge bit. In this case, the EVF and AF bits are set with an interrupt if the ITE bit is set.

**Note:** In both cases, the SCL line is not held low; however, the SDA line can remain low due to possible "0" bits transmitted last. It is then necessary to release both lines by software.

#### How to Release the SDA / SCL lines

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

## I<sup>2</sup>C BUS INTERFACE (Cont'd)

### 5.7.4.2 Master Mode

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

#### Start Condition and Transmit Slave Address

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

Once the Start condition is sent:

- The EVF and SB bits are set by hardware with an interrupt if the ITE bit is set.

Then the master waits for a read of the SR1 register followed by a write in the DR register with the Slave address byte, **holding the SCL line low** (see Figure 3 Transfer sequencing EV5).

Then the slave address byte is sent to the SDA line via the internal shift register.

After completion of this transfer (and acknowledge from the slave if the ACK bit is set):

- The EVF bit is set by hardware with interrupt generation if the ITE bit is set.

Then the master waits for a read of the SR1 register followed by a write in the CR register (for example set PE bit), **holding the SCL line low** (see Figure 3 Transfer sequencing EV6).

Next the master must enter Receiver or Transmitter mode.

#### Master Receiver

Following the address transmission and after the SR1 and CR registers have been accessed, the master receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

- An Acknowledge pulse is generated if the ACK bit is set
- EVF and BTF bits are set by hardware with an interrupt if the ITE bit is set.

Then the interface waits for a read of the SR1 register followed by a read of the DR register, **holding the SCL line low** (see Figure 3 Transfer sequencing EV7).

To close the communication: before reading the last byte from the DR register, set the STOP bit to generate the Stop condition. The interface returns automatically to slave mode (M/SL bit cleared).

**Note:** In order to generate the non-acknowledge pulse after the last received data byte, the ACK bit must be cleared just before reading the second last data byte.

#### Master Transmitter

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

The master waits for a read of the SR1 register followed by a write in the DR register, **holding the SCL line low** (see Figure 3 Transfer sequencing EV8).

When the acknowledge bit is received, the interface sets:

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

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

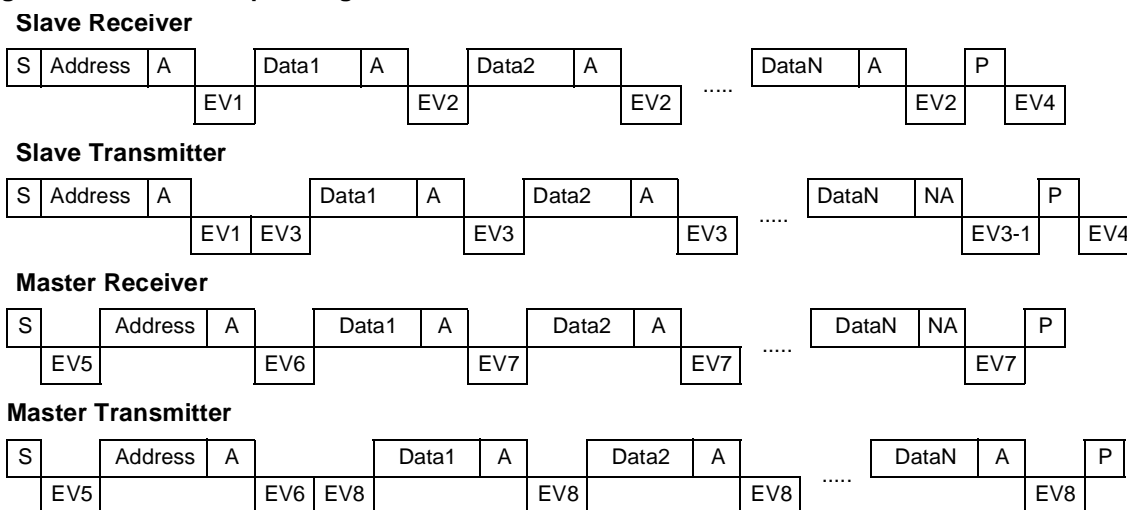
#### Error Cases

- **BERR:** Detection of a Stop or a Start condition during a byte transfer. In this case, the EVF and BERR bits are set by hardware with an interrupt if the ITE bit is set.
- **AF:** Detection of a non-acknowledge bit. In this case, the EVF and AF bits are set by hardware with an interrupt if the ITE bit is set. To resume, set the START or STOP bit.
- **ARLO:** Detection of an arbitration lost condition. In this case the ARLO bit is set by hardware (with an interrupt if the ITE bit is set and the interface goes automatically back to slave mode (the M/SL bit is cleared).

**Note:** In all these cases, the SCL line is not held low; however, the SDA line can remain low due to possible "0" bits transmitted last. It is then necessary to release both lines by software.

I<sup>2</sup>C BUS INTERFACE (Cont'd)

Figure 40. Transfer Sequencing

**Legend:**

S=Start, P=Stop, A=Acknowledge, NA=Non-acknowledge

EVx=Event (with interrupt if ITE=1)

**EV1:** EVF=1, ADSL=1, cleared by reading the SR1 register.**EV2:** EVF=1, BTF=1, cleared by reading the SR1 register followed by reading the DR register.**EV3:** EVF=1, BTF=1, cleared by reading the SR1 register followed by writing the DR register.**EV3-1:** EVF=1, AF=1, BTF=1; AF is cleared by reading the SR1 register. The BTF is cleared by releasing the lines (STOP=1, STOP=0) or by writing the DR register (DR=FFh).**Note:** If lines are released by STOP=1, STOP=0, the subsequent EV4 is not seen.**EV4:** EVF=1, STOPF=1, cleared by reading the SR2 register.**EV5:** EVF=1, SB=1, cleared by reading the SR1 register followed by writing the DR register.**EV6:** EVF=1, cleared by reading the SR1 register followed by writing the CR register (for example PE=1).**EV7:** EVF=1, BTF=1, cleared by reading the SR1 register followed by reading the DR register.**EV8:** EVF=1, BTF=1, cleared by reading the SR1 register followed by writing the DR register.

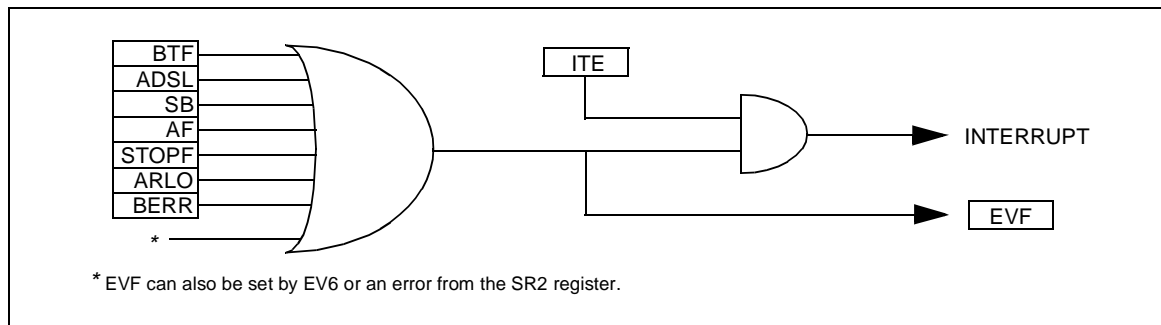
**I<sup>2</sup>C BUS INTERFACE (Cont'd)**

**5.7.5 Low Power Modes**

Mode	Description
WAIT	No effect on I <sup>2</sup> C interface. I <sup>2</sup> C interrupts exit from Wait mode.
HALT	I <sup>2</sup> C registers are frozen. In Halt mode, the I <sup>2</sup> C interface is inactive and does not acknowledge data on the bus. The I <sup>2</sup> C interface resumes operation when the MCU is woken up by an interrupt with "exit from Halt mode" capability.

**5.7.6 Interrupts**

**Figure 41. Event Flags and Interrupt Generation**



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

The I<sup>2</sup>C interrupt events are connected to the same interrupt vector (see Interrupts chapter).

They generate an interrupt if the corresponding Enable Control Bit is set and the I-bit in the CC register is reset (RIM instruction).

**I<sup>2</sup>C BUS INTERFACE** (Cont'd)**5.7.7 Register Description****I<sup>2</sup>C CONTROL REGISTER (CR)**

Read / Write

Reset Value: 0000 0000 (00h)

7							0
0	0	PE	ENGC	START	ACK	STOP	ITE

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

Bit 5 = **PE** *Peripheral enable*.

This bit is set and cleared by software.

0: Peripheral disabled

1: Master/Slave capability

**Notes:**

- When PE=0, all the bits of the CR register and the SR register except the Stop bit are reset. All outputs are released while PE=0
- When PE=1, the corresponding I/O pins are selected by hardware as alternate functions.
- To enable the I<sup>2</sup>C interface, write the CR register **TWICE** with PE=1 as the first write only activates the interface (only PE is set).

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

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

0: General Call disabled

1: General Call enabled

Bit 3 = **START** *Generation of a Start condition*.

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

– In master mode:

0: No start generation

1: Repeated start generation

– In slave mode:

0: No start generation

1: Start generation when the bus is free

Bit 2 = **ACK** *Acknowledge enable*.

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

0: No acknowledge returned

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

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

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

– In Master mode:

0: No stop generation

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

– In Slave mode:

0: No stop generation

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

Bit 0 = **ITE** *Interrupt enable*.

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

0: Interrupts disabled

1: Interrupts enabled

Refer to Figure 4 for the relationship between the events and the interrupt.

SCL is held low when the SB, BTF or ADSL flags or an EV6 event (See Figure 3) is detected.

**I<sup>2</sup>C BUS INTERFACE** (Cont'd)**I<sup>2</sup>C STATUS REGISTER 1 (SR1)**

Read Only

Reset Value: 0000 0000 (00h)

7							0
EVF	0	TRA	BUSY	BTF	ADSL	M/SL	SB

**Bit 7 = EVF Event flag.**

This bit is set by hardware as soon as an event occurs. It is cleared by software reading SR2 register in case of error event or as described in Figure 3. It is also cleared by hardware when the interface is disabled (PE=0).

0: No event

1: One of the following events has occurred:

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

Bit 6 = Reserved. Forced to 0 by hardware.

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

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

0: Data byte received (if BTF=1)

1: Data byte transmitted

**Bit 4 = BUSY Bus busy.**

This bit is set by hardware on detection of a Start condition and cleared by hardware on detection of a Stop condition. It indicates a communication is still updated when the interface is disabled (PE=0).

0: No communication on the bus

1: Communication ongoing on the bus

**Bit 3 = BTF Byte transfer finished.**

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

– Following a byte transmission, this bit is set after reception of the acknowledge clock pulse. In case an address byte is sent, this bit is set only after the EV6 event (See Figure 3). BTF is cleared by reading SR1 register followed by writing the next byte in DR register.

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

The SCL line is held low while BTF=1.

0: Byte transfer not done

1: Byte transfer succeeded

**Bit 2 = ADSL Address matched (Slave mode).**

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

The SCL line is held low while ADSL=1.

0: Address mismatched or not received

1: Received address matched

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

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

0: Slave mode

1: Master mode

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

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

0: No Start condition

1: Start condition generated



**I<sup>2</sup>C BUS INTERFACE** (Cont'd)**I<sup>2</sup>C STATUS REGISTER 2 (SR2)**

Read Only

Reset Value: 0000 0000 (00h)

7							0
0	0	0	AF	STOPF	ARLO	BERR	GCAL

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

**Bit 4 = AF Acknowledge failure.**

This bit is set by hardware when no acknowledge is returned. An interrupt is generated if ITE=1. It is cleared by software reading SR2 register or by hardware when the interface is disabled (PE=0).

The SCL line is not held low while AF=1.

0: No acknowledge failure  
1: Acknowledge failure

**Bit 3 = STOPF Stop detection (Slave mode).**

This bit is set by hardware when a Stop condition is detected on the bus after an acknowledge (if ACK=1). An interrupt is generated if ITE=1. It is cleared by software reading SR2 register or by hardware when the interface is disabled (PE=0).

The SCL line is not held low while STOPF=1.

0: No Stop condition detected  
1: Stop condition detected

**Bit 2 = ARLO Arbitration lost.**

This bit is set by hardware when the interface loses

the arbitration of the bus to another master. An interrupt is generated if ITE=1. It is cleared by software reading SR2 register or by hardware when the interface is disabled (PE=0).

After an ARLO event the interface switches back automatically to Slave mode (M/SL=0).

The SCL line is not held low while ARLO=1.

0: No arbitration lost detected  
1: Arbitration lost detected

**Bit 1 = BERR Bus error.**

This bit is set by hardware when the interface detects a misplaced Start or Stop condition. An interrupt is generated if ITE=1. It is cleared by software reading SR2 register or by hardware when the interface is disabled (PE=0).

The SCL line is not held low while BERR=1.

0: No misplaced Start or Stop condition  
1: Misplaced Start or Stop condition

**Bit 0 = GCAL General Call (Slave mode).**

This bit is set by hardware when a general call address is detected on the bus while ENGC=1. It is cleared by hardware detecting a Stop condition (STOPF=1) or when the interface is disabled (PE=0).

0: No general call address detected on bus  
1: general call address detected on bus

**I<sup>2</sup>C BUS INTERFACE (Cont'd)****I<sup>2</sup>C CLOCK CONTROL REGISTER (CCR)**

Read / Write

Reset Value: 0000 0000 (00h)

7							0
FM/SM	CC6	CC5	CC4	CC3	CC2	CC1	CC0

Bit 7 = **FM/SM** *Fast/Standard I<sup>2</sup>C mode*.

This bit is set and cleared by software. It is not cleared when the interface is disabled (PE=0).

0: Standard I<sup>2</sup>C mode

1: Fast I<sup>2</sup>C mode

Bits 6:0 = **CC6-CC0** *7-bit clock divider*.

These bits select the speed of the bus ( $F_{SCL}$ ) depending on the I<sup>2</sup>C mode. They are not cleared when the interface is disabled (PE=0).

– Standard mode (FM/SM=0):  $F_{SCL} \leq 100\text{kHz}$

$$F_{SCL} = f_{CPU} / (2 \times ([CC6..CC0] + 2))$$

– Fast mode (FM/SM=1):  $F_{SCL} > 100\text{kHz}$

$$F_{SCL} = f_{CPU} / (3 \times ([CC6..CC0] + 2))$$

Note: The programmed  $F_{SCL}$  assumes no load on SCL and SDA lines.

**I<sup>2</sup>C DATA REGISTER (DR)**

Read / Write

Reset Value: 0000 0000 (00h)

7							0
D7	D6	D5	D4	D3	D2	D1	D0

Bits 7:0 = **D7-D0** *8-bit Data Register*.

These bits contains the byte to be received or transmitted on the bus.

– Transmitter mode: Byte transmission start automatically when the software writes in the DR register.

– Receiver mode: the first data byte is received automatically in the DR register using the least significant bit of the address.

Then, the next data bytes are received one by one after reading the DR register.

**I<sup>2</sup>C OWN ADDRESS REGISTER (OAR)**

Read / Write

Reset Value: 0000 0000 (00h)

7							0
ADD7	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0

Bits 7:1 = **ADD7-ADD1** *Interface address*.

These bits define the I<sup>2</sup>C bus address of the interface. They are not cleared when the interface is disabled (PE=0).

Bit 0 = **ADD0** *Address direction bit*.

This bit is don't care, the interface acknowledges either 0 or 1. It is not cleared when the interface is disabled (PE=0).

Note: Address 01h is always ignored.

Table 20. I<sup>2</sup>C Register Map

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
39	DR	DR7 .. DR0							
3B	OAR	ADD7 .. ADD0							
3C	CCR	FM/SM	CC6 .. CC0						
3D	SR2				AF	STOPF	ARLO	BERR	GCAL
3E	SR1	EVF		TRA	BUSY	BTF	ADSL	M/SL	SB
3F	CR			PE	ENG C	START	ACK	STOP	ITE

## 5.8 8-BIT A/D CONVERTER (ADC)

### 5.8.1 Introduction

The on-chip Analog to Digital Converter (ADC) peripheral is a 8-bit, successive approximation converter with internal sample and hold circuitry. This peripheral has up to 8 multiplexed analog input channels (refer to device pin out description) that allow the peripheral to convert the analog voltage levels from up to 8 different sources.

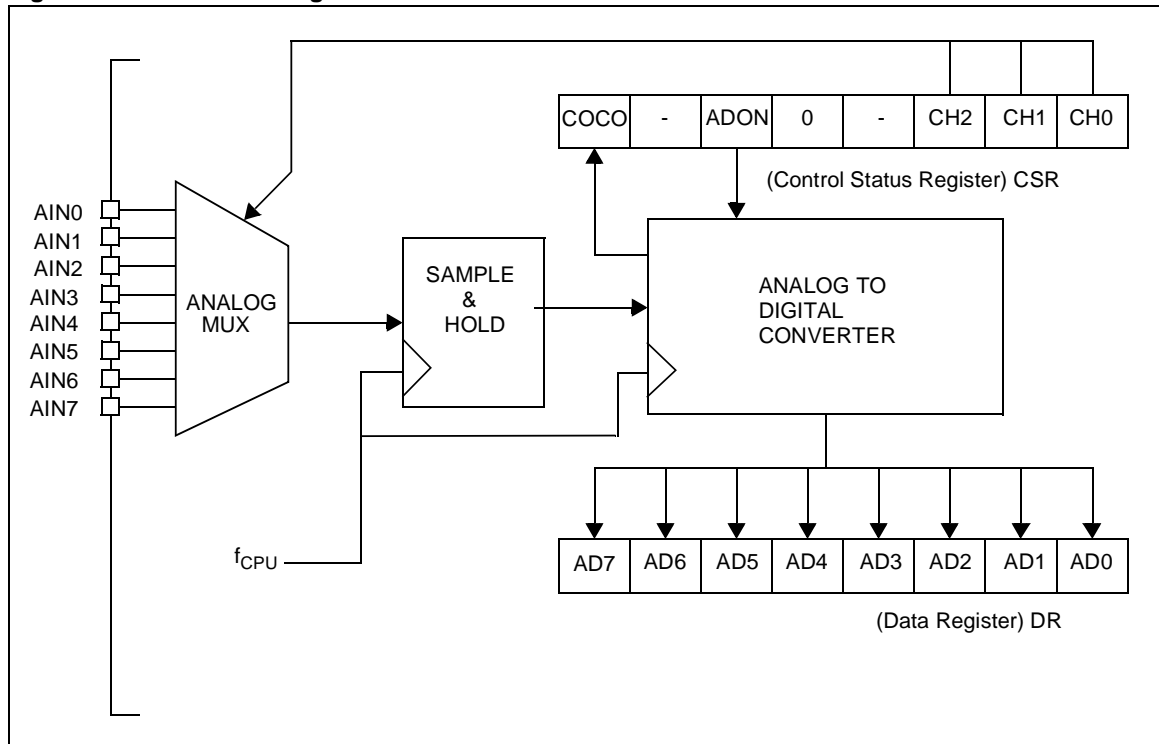
The result of the conversion is stored in a 8-bit Data Register. The A/D converter is controlled through a Control/Status Register.

### 5.8.2 Main Features

- 8-bit conversion
- Up to 8 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 42.

Figure 42. ADC Block Diagram



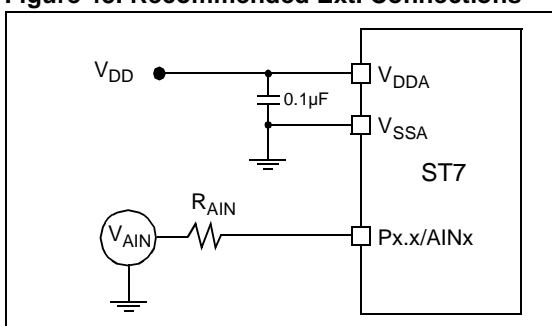
## 8-BIT A/D CONVERTER (ADC) (Cont'd)

### 5.8.3 Functional Description

The high level reference voltage  $V_{DDA}$  must be connected externally to the  $V_{DD}$  pin. The low level reference voltage  $V_{SSA}$  must be connected externally to the  $V_{SS}$  pin. In some devices (refer to device pin out description) high and low level reference voltages are internally connected to the  $V_{DD}$  and  $V_{SS}$  pins.

Conversion accuracy may therefore be degraded by voltage drops and noise in the event of heavily loaded or badly decoupled power supply lines.

**Figure 43. Recommended Ext. Connections**



#### Characteristics:

The conversion is monotonic, meaning the result never decreases if the analog input does not and never increases if the analog input does not.

If input voltage is greater than or equal to  $V_{DD}$  (voltage reference high) then results = FFh (full scale) without overflow indication.

If input voltage  $\leq V_{SS}$  (voltage reference low) then the results = 00h.

The conversion time is 64 CPU clock cycles including a sampling time of 31.5 CPU clock cycles.

$R_{AIN}$  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.

The A/D converter is linear and the digital result of the conversion is given by the formula:

$$\text{Digital result} = \frac{255 \times \text{Input Voltage}}{\text{Reference Voltage}}$$

Where Reference Voltage is  $V_{DD} - V_{SS}$ .

The accuracy of the conversion is described in the Electrical Characteristics Section.

#### Procedure:

Refer to the CSR and DR register description section for the bit definitions.

Each analog input pin 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 CSR register:

- Select the CH2 to CH0 bits to assign the analog channel to convert. Refer to Table 21 Channel Selection.
- Set the ADON bit. Then the A/D converter is enabled after a stabilization time (typically 30  $\mu$ s). It then performs a continuous conversion of the selected channel.

When a conversion is complete

- The COCO bit is set by hardware.
- No interrupt is generated.
- The result is in the DR register.

A write to the CSR register aborts the current conversion, resets the COCO bit and starts a new conversion.

### 5.8.4 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.

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 stabilisation time before accurate conversions can be performed.

### 5.8.5 Interrupts

None.

**8-BIT A/D CONVERTER (ADC) (Cont'd)****5.8.6 Register Description****CONTROL/STATUS REGISTER (CSR)**

Read/Write

Reset Value: 0000 0000 (00h)

7							0
COCO	-	ADON	0	-	CH2	CH1	CH0

Bit 7 = **COCO** *Conversion Complete*

This bit is set by hardware. It is cleared by software reading the result in the DR register or writing to the CSR register.

0: Conversion is not complete.

1: Conversion can be read from the DR register.

Bit 6 = **Reserved**. Must always be cleared.Bit 5 = **ADON** *A/D converter On*

This bit is set and cleared by software.

0: A/D converter is switched off.

1: A/D converter is switched on.

**Note:** A typical 30  $\mu$ s delay time is necessary for the ADC to stabilize when the ADON bit is set.

Bit 4 = **Reserved**. Forced by hardware to 0.Bit 3 = **Reserved**. Must always be cleared.Bits 2:0: **CH[2:0]** *Channel Selection*

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

**Table 21.** Channel Selection

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
AIN7	1	1	1

**\*IMPORTANT NOTE:** The number of pins AND the channel selection vary according to the device. REFER TO THE DEVICE PINOUT).

**DATA REGISTER (DR)**

Read Only

Reset Value: 0000 0000 (00h)

7							0
AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0

Bit 7:0 = **AD[7:0]** *Analog Converted Value*

This register contains the converted analog value in the range 00h to FFh.

Reading this register resets the COCO flag.



## 6 INSTRUCTION SET

### 6.1 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 22. 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)****6.1.1 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

**6.1.2 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

**6.1.3 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.

**6.1.4 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.

**6.1.5 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)**6.1.6 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 23. 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

**6.1.7 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.

## 6.2 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				
Code Condition 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.

## 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	$A = A \cdot M$	A	M			N	Z	
BCP	Bit compare A, Memory	tst (A . M)	A	M			N	Z	
BRES	Bit Reset	bres Byte, #3	M						
BSET	Bit Set	bset Byte, #3	M						
BTJF	Jump if bit is false (0)	btjf Byte, #3, Jmp1	M						C
BTJT	Jump if bit is true (1)	btjt Byte, #3, Jmp1	M						C
CALL	Call 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	H	I	N	Z	C
PUSH	Push onto the Stack	push Y	M	reg, CC					
RCF	Reset carry flag	C = 0							0
RET	Subroutine Return								
RIM	Enable Interrupts	I = 0				0			
RLC	Rotate left true C	C <= Dst <= C	reg, M				N	Z	C
RRC	Rotate right true C	C => Dst => C	reg, M				N	Z	C
RSP	Reset Stack Pointer	S = Max allowed							
SBC	Subtract with Carry	A = A - M - C	A	M			N	Z	C
SCF	Set carry flag	C = 1							1
SIM	Disable Interrupts	I = 1				1			
SLA	Shift left Arithmetic	C <= Dst <= 0	reg, M				N	Z	C
SLL	Shift left Logic	C <= Dst <= 0	reg, M				N	Z	C
SRL	Shift right Logic	0 => Dst => C	reg, M				0	Z	C
SRA	Shift right Arithmetic	Dst7 => Dst => C	reg, M				N	Z	C
SUB	Subtraction	A = A - M	A	M			N	Z	C
SWAP	SWAP nibbles	Dst[7..4] <=> Dst[3..0]	reg, M				N	Z	
TNZ	Test for Neg & Zero	tnz lbl1					N	Z	
TRAP	S/W trap	S/W interrupt				1			
WFI	Wait for Interrupt					0			
XOR	Exclusive OR	A = A XOR M	A	M			N	Z	

## 7 ELECTRICAL CHARACTERISTICS

### 7.1 ABSOLUTE MAXIMUM RATINGS

Devices of the ST72 family contain circuitry to protect the inputs against damage due to high static voltage or electric fields. Nevertheless, it is recommended that normal precautions be observed in order to avoid subjecting this high-impedance circuit to voltages above those quoted in the Absolute Maximum Ratings. For proper operation, it is recommended that the input voltage  $V_{IN}$  be constrained within the range:

$$(V_{SS} - 0.3V) \leq V_{IN} \leq (V_{DD} + 0.3V)$$

To enhance reliability of operation, it is recommended to configure unused I/Os as inputs and to

connect them to an appropriate logic voltage level such as  $V_{SS}$  or  $V_{DD}$ . It is also recommended to connect  $V_{DDA}$  and  $V_{DD}$  together on application. (same remark for  $V_{SSA}$  and  $V_{SS}$ ).

All the voltage in the following tables are referenced to  $V_{SS}$ .

Stresses above those listed as "Absolute Maximum Ratings" may cause permanent damage to the device. Functional operation of the device at these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

**Table 24. Absolute Maximum Ratings (Voltage Referenced to  $V_{SS}$ )**

Symbol	Ratings	Value	Unit
$V_{DD}$	Recommended Supply Voltage	- 0.3 to +6.0	V
$V_{DDA}$	Analog Reference Voltage	- 0.3 to +6.0	V
$ V_{DDA} - V_{DD} $	Max. variations on Power Line	50	mV
$ V_{SSA} - V_{SS} $	Max. variations on Ground Line	50	mV
$I_{VDD} - I_{VSS}$	Total current into $V_{DD}/V_{SS}$	80/80	mA
$V_{IN}$	Input Voltage	$V_{SS} - 0.3$ to $V_{DD} + 0.3$	V
$V_{OUT}$	Output Voltage	$V_{SS} - 0.3$ to $V_{DD} + 0.3$	V
$T_A$	Ambient Temperature Range	$T_L$ to $T_H$ 0 to + 70	°C
$T_{STG}$	Storage Temperature Range	-65 to +150	°C
$T_J$	Junction Temperature	150	°C
PD	Power Dissipation	350	mW
ESD	ESD susceptibility	2000	V

## 7.2 THERMAL CHARACTERISTICS

The average chip-junction temperature,  $T_J$ , in degrees Celsius, may be calculated using the following equation:

$$T_J = T_A + (P_D \times \theta_{JA}) \quad (1)^*$$

Where:

- $T_A$  is the Ambient Temperature in °C,
- $\theta_{JA}$  is the Package Junction-to-Ambient Thermal Resistance, in °C/W,
- $P_D$  is the sum of  $P_{INT}$  and  $P_{I/O}$ ,
- $P_{INT}$  is the product of  $I_{DD}$  and  $V_{DD}$ , expressed in Watts. This is the Chip Internal Power
- $P_{I/O}$  represents the Power Dissipation on Input and Output Pins; User Determined.

For most applications  $P_{I/O} < P_{INT}$  and may be neglected.  $P_{I/O}$  may be significant if the device is configured to drive Darlington bases or sink LED Loads.

An approximate relationship between  $P_D$  and  $T_J$  (if  $P_{I/O}$  is neglected) is given by:

$$P_D = K \div (T_J + 273^\circ\text{C}) \quad (2)$$

Therefore:

$$K = P_D \times (T_A + 273^\circ\text{C}) + \theta_{JA} \times P_D^2 \quad (3)$$

Where:

- $K$  is a constant for the particular part, which may be determined from equation (3) by measuring  $P_D$  (at equilibrium) for a known  $T_A$ . Using this value of  $K$ , the values of  $P_D$  and  $T_J$  may be obtained by solving equations (1) and (2) iteratively for any value of  $T_A$ .

**Table 25. Thermal Characteristics**

Symbol	Package	Typical Value	Unit
$\theta_{JA}$	SO34	70	°C/W
	PSDIP32	50	

(\*): Maximum chip dissipation can directly be obtained from  $T_j$  (max),  $\theta_{JA}$  and  $T_A$  parameters.



### 7.3 OPERATING CONDITIONS

#### General Operating Conditions

( $T_A = 0$  to  $+70^\circ\text{C}$  unless otherwise specified)

Symbol	Parameter	Conditions	Min	Max	Unit
$V_{DD}$	Supply voltage <sup>1)</sup>	$f_{CPU} = 4$ MHz ; USB not guaranteed	3.00	4.00	V
		$f_{CPU} = 8$ MHz ; USB not guaranteed	$V_{IT+}$	4.00	V
		$f_{CPU} = 8$ MHz or 4 MHz USB guaranteed	4.0	5.25	
		$f_{CPU} = 8$ MHz or 4 MHz USB not guaranteed	5.25	5.50	
$f_{OSC}$	External clock frequency		12	24	MHz

**Note 1:** USB 1.1 specifies that the power supply must be between 4.00 and 5.25 Volts. The USB cell is therefore guaranteed only in that range.

## 7.4 POWER CONSUMPTION

( $T_A = 0$  to  $+70^\circ\text{C}$  unless otherwise specified)

GENERAL						
Symbol	Parameter	Conditions	Min	Typ.	Max	Unit
$V_{DD}$	Operating Supply Voltage	RUN & WAIT mode $f_{OSC} = 24$ MHz $f_{CPU} = 8$ MHz	4	5	5.5	V
$V_{DDA}$	Analog Reference Voltage		4	5	5.5	V
$I_{DD}$	CPU RUN mode (see Note 1)	I/O in input mode $f_{CPU} = 8$ MHz, $T_A = 20^\circ\text{C}$ (For $V_{DD}$ : see Note 5)		14	20	mA
	CPU WAIT mode (See Note 2)			8	12	mA
	CPU HALT mode (see Note 3)				100	$\mu\text{A}$
	USB Suspend mode (see Note 4)			350	450	$\mu\text{A}$

**Note 1:** All peripherals running.

**Note 2:** Oscillator, 16-bit Timer (free running counter) and watchdog running.  
All others peripherals (including EPROM/RAM memories) disabled.

**Note 3:** CPU in HALT mode, USB Transceiver disabled, Low Voltage Reset function enabled.

**Note 4:** Low voltage reset function enabled.  
CPU in HALT mode.

USB in suspend mode. External pull-up (1.5Kohms to USBVCC) and pull-down (15Kohms to  $V_{SSA}$ ) connected on drivers.

**Note 5:**  $V_{DD} = 5.5$  V except in USB Suspend mode where  $V_{DD} = 5.25$  V





## 7.5 I/O PORT CHARACTERISTICS

( $T_A = 0$  to  $+70^\circ\text{C}$  unless otherwise specified)

STANDARD I/O PORT PINS						
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{OL}$	Output Low Level Voltage Port A1, Port A2 (High Current open drain)	$I_{OL} = -25\text{mA}$ $V_{DD}=5\text{V}$	-	-	1.5	V
	Output Low Level Voltage Port A0, Port A(3:7), Port C(0:2), Push Pull	$I_{OL} = -1.6\text{mA}$ $V_{DD}=5\text{V}$	-	-	0.4	V
	Output Low Level Voltage Port B (0:7), Push Pull	$I_{OL} = -10\text{mA}$ $V_{DD}=5\text{V}$	-	-	1.3	V
$V_{OH}$	Output High Level Voltage Port A0, Port A(3:7), Port C(0:2) Push Pull	$I_{OH} = 1.6\text{mA}$	$V_{DD}-0.8$	-	-	V
$V_{OH}$	Output High Level Voltage Port B (0:7) Push Pull	$I_{OH} = 10\text{mA}$	$V_{DD}-1.3$	-	-	V
$V_{IH}$	Input High Level Voltage PA(0:7), PB(0:7), PC(0:2), $\overline{\text{RESET}}$	Leading Edge	$0.7 \times V_{DD}$		$V_{DD}$	V
$V_{IL}$	Input Low Voltage PA(0-7), PB(0-7), PC(0-2), $\overline{\text{RESET}}$	Trailing Edge	$V_{SS}$		$0.3 \times V_{DD}$	V
$R_{PU}$	Pull-up resistor	$V_{DD} = 5\text{V}$	80	100	120	k $\Omega$
CIO	I/O Pin Capacitance <sup>1)</sup>			5		pF
$t_{f(I/O)\text{out}}$	Output High to Low Level Fall Time All I/O ports	CL=50pF Between 10% and 90%		25 <sup>2)</sup>		ns
$t_{r(I/O)\text{out}}$	Output Low to High Level Rise Time I/O ports in Push Pull mode			25 <sup>2)</sup>		ns
$t_{r(I/O)\text{out}}$	External Interrupt pulse time <sup>1)</sup>		1			$t_{\text{CPU}}$

All voltages are referred to  $V_{SS}$  unless otherwise specified.

**Note 1:** Guaranteed by design, not tested in production.

**Note 2:** Data based on characterization results, not tested in production.

## 7.6 LOW VOLTAGE DETECTOR (LVD) CHARACTERISTICS

LOW VOLTAGE RESET Electrical Specifications						
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{IT+}$	Low Voltage Reset Threshold $V_{DD}$ rising	$V_{DD}$ Max. Variation 50mV/ $\mu$ s	3.6	3.75	4.0	V
$V_{IT-}$	Low Voltage Reset Threshold $V_{DD}$ falling	$V_{DD}$ Max. Variation 50mV/ $\mu$ s	3.2	3.5	3.7	V
$V_{hys}$	Hysteresis ( $V_{IT+} - V_{IT-}$ )		200	250		mV

## 7.7 CONTROL TIMING CHARACTERISTICS

(Operating conditions  $T_A = 0$  to  $+70^\circ\text{C}$  unless otherwise specified)

CONTROL TIMINGS						
Symbol	Parameter	Conditions	Value			Unit
			Min	Typ.	Max	
$f_{OSC}$	Oscillator Frequency				24	MHz
$f_{CPU}$	Operating Frequency				8	MHz
$t_{RL}$	External RESET Input pulse Width		1.5			$t_{CPU}$
$t_{PORL}$	Internal Power Reset Duration		4096			$t_{CPU}$
$T_{DOGL}$	Watchdog & Low Voltage Reset Output Pulse Width		200			ns
$t_{DOG}$	Watchdog Time-out	$f_{cpu} = 8\text{MHz}$	49152 6		3145728 384	$t_{CPU}$ ms
$t_{OXOV}$	Crystal Oscillator Start-up Time				50	ms
$t_{DDR}$	Power up rise time	from $V_{DD} = 0$ to 4V			100	ms

**Note 1:** The minimum period  $t_{LIL}$  should not be less than the number of cycle times it takes to execute the interrupt service routine plus 21 cycles.

## 7.8 COMMUNICATION INTERFACE CHARACTERISTICS

The values given in the specifications of dedicated functions are generally not applicable for chips. Therefore, only the limits listed below are valid for the product.  $T = 0 \dots +70^{\circ}\text{C}$ ,  $V_{\text{DD}} - V_{\text{SS}} = 5 \text{ V}$  unless otherwise specified.

### 7.8.1 USB - Universal Bus Interface

(Operating conditions  $T_{\text{A}} = 0$  to  $+70^{\circ}\text{C}$ ,  $V_{\text{DD}} = 4.0$  to  $5.25\text{V}$  unless otherwise specified)

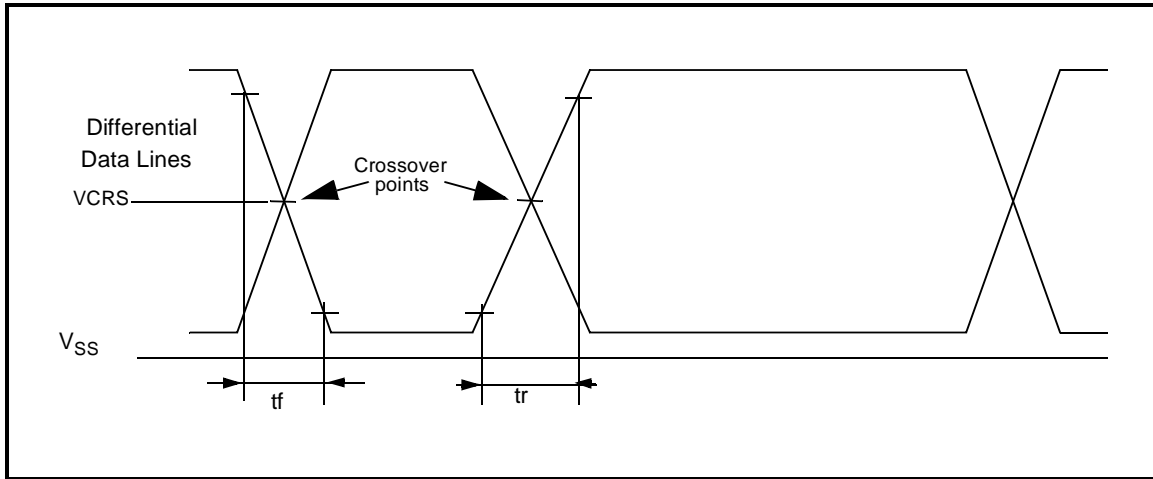
USB DC Electrical Characteristics					
Parameter	Symbol	Conditions	Min.	Max.	Unit
Input Levels:					
Differential Input Sensitivity	VDI	I(D+, D-)	0.2		V
Differential Common Mode Range	VCM	Includes VDI range	0.8	2.5	V
Single Ended Receiver Threshold	VSE		0.8	2.0	V
Output Levels					
Static Output Low	VOL	RL of 1.5K ohms to 3.6v		0.3	V
Static Output High	VOH	RL of 15K ohm to $V_{\text{SS}}$	2.8	3.6	V
USBVCC: voltage level	USBV	$V_{\text{DD}}=5\text{v}$	3.00	3.60	V

**Note 1:** RL is the load connected on the USB drivers.

**Note 2:** All the voltages are measured from the local ground potential.

## COMMUNICATION INTERFACE CHARACTERISTICS (Cont'd)

Figure 44. USB: Data signal Rise and fall time



USB: Low speed electrical characteristics					
Parameter	Symbol	Conditions	Min	Max	Unit
Driver characteristics:					
Rise time	tr	Note 1, CL=50 pF	75		ns
		Note 1, CL=600 pF		300	ns
Fall Time	tf	Note 1, CL=50 pF	75		ns
		Note 1, CL=600 pF		300	ns
Rise/ Fall Time matching	trfm	tr/tf	80	120	%
Output signal Crossover Voltage	VCRS		1.3	2.0	V

Note1: Measured from 10% to 90% of the data signal. For more detailed informations, please refer to Chapter 7 (Electrical) of the USB specification (version 1.1).

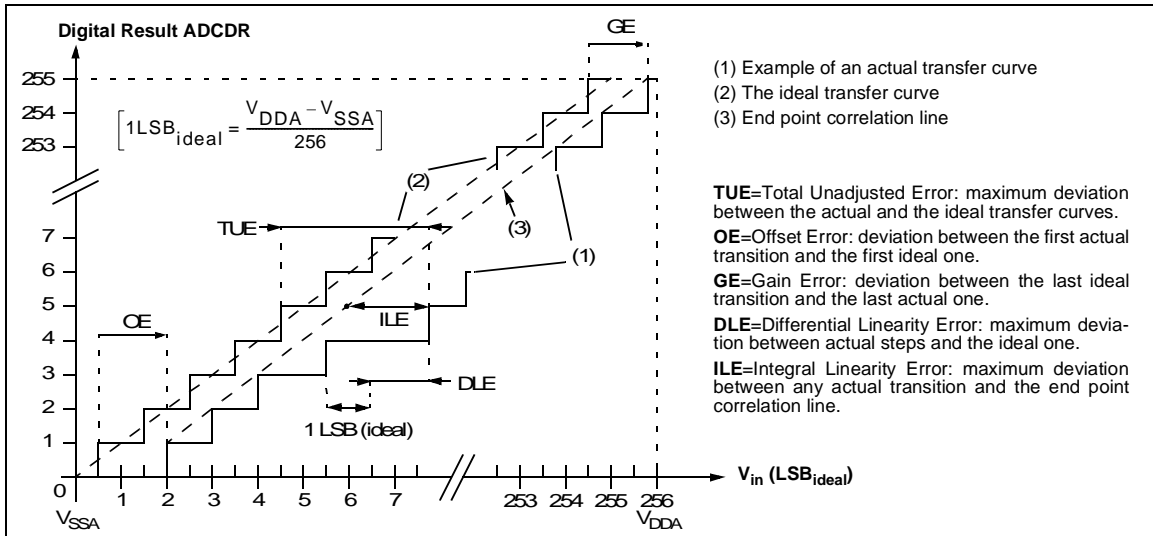
**COMMUNICATION INTERFACE CHARACTERISTICS (Cont'd)****7.8.2 I<sup>2</sup>C - Inter IC Control Interface**

I <sup>2</sup> C/DDC-Bus Timings						
Parameter	Standard I <sup>2</sup> C		Fast I <sup>2</sup> C		Symbol	Unit
	Min	Max	Min	Max		
Bus free time between a STOP and START condition	4.7		1.3		T <sub>BUF</sub>	ms
Hold time START condition. After this period, the first clock pulse is generated	4.0		0.6		T <sub>HD:STA</sub>	μs
LOW period of the SCL clock	4.7		1.3		T <sub>LOW</sub>	μs
HIGH period of the SCL clock	4.0		0.6		T <sub>HIGH</sub>	μs
Set-up time for a repeated START condition	4.7		0.6		T <sub>SU:STA</sub>	μs
Data hold time	0 (1)		0 (1)	0.9(2)	T <sub>HD:DAT</sub>	ns
Data set-up time	250		100		T <sub>SU:DAT</sub>	ns
Rise time of both SDA and SCL signals		1000	20+0.1Cb	300	T <sub>R</sub>	ns
Fall time of both SDA and SCL signals		300	20+0.1Cb	300	TF	ns
Set-up time for STOP condition	4.0		0.6		T <sub>SU:STO</sub>	ns
Capacitive load for each bus line		400		400	Cb	pF

- 1) The device must internally provide a hold time of at least 300 ns for the SDA signal in order to bridge the undefined region of the falling edge of SCL
- 2) The maximum hold time of the START condition has only to be met if the interface does not stretch the low period of SCL signal

Cb = total capacitance of one bus line in pF

7.9 8-BIT ADC CHARACTERISTICS



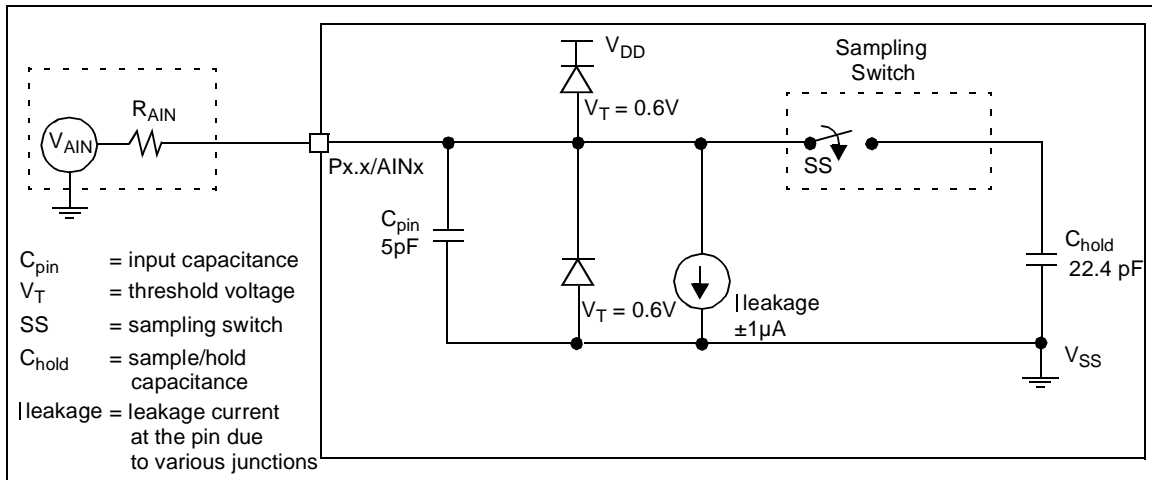
ADC Analog to Digital Converter (8-bit)						
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
TUE	Total unadjusted error*	$f_{\text{ADC}}=f_{\text{CPU}}=4\text{MHz}$ $V_{\text{DD}}=V_{\text{DDA}}=5\text{V}$			2	LSB
OE	Offset error*		-1		1	
GE	Gain Error*		-2		2	
DLE	Differential linearity error*				1	
ILE	Integral linearity error*				2	
$V_{\text{AIN}}$	Conversion range voltage		$V_{\text{SSA}}$		$V_{\text{DDA}}$	V
$I_{\text{ADC}}$	A/D conversion supply current			1		mA
$t_{\text{STAB}}$	Stabilization time after enable ADC				30	$\mu\text{s}$
$t_{\text{LOAD}}$	Sample capacitor loading time	$f_{\text{ADC}}=f_{\text{CPU}}=4\text{MHz}$ $V_{\text{DD}}=V_{\text{DDA}}=5\text{V}$		8		$\mu\text{s}$
					32	
$t_{\text{CONV}}$	Hold conversion time			8		$\mu\text{s}$
					32	$1/f_{\text{ADC}}$
$R_{\text{AIN}}$	External input resistor				20	$\text{K}\Omega$
$R_{\text{ADC}}$	Internal input resistor				18	$\text{K}\Omega$
$C_{\text{SAMPLE}}$	Sample capacitor				22	pF

**\*Note: ADC Accuracy vs. Negative Injection Current.**  
 For  $I_{\text{inj}}=-0.8\text{mA}$ , the typical leakage induced inside the die is  $1.6\mu\text{A}$  and the effect on the ADC accuracy is a loss of 1 LSB by  $10\text{K}\Omega$  increase of the external analog source impedance.

These measurements results and recommendations are done in the worst condition of injection:

- negative injection
- injection to an Input with analog capability ,adjacent to the enabled Analog Input
- at  $5\text{V } V_{\text{DD}}$  supply, and worse temperature case.

## 8-BIT ADC CHARACTERISTICS (Cont'd)



## 8 PACKAGE CHARACTERISTICS

### 8.1 PACKAGE MECHANICAL DATA

Figure 45. 34-Pin Shrink Plastic Small Outline Package, 300-mil Width

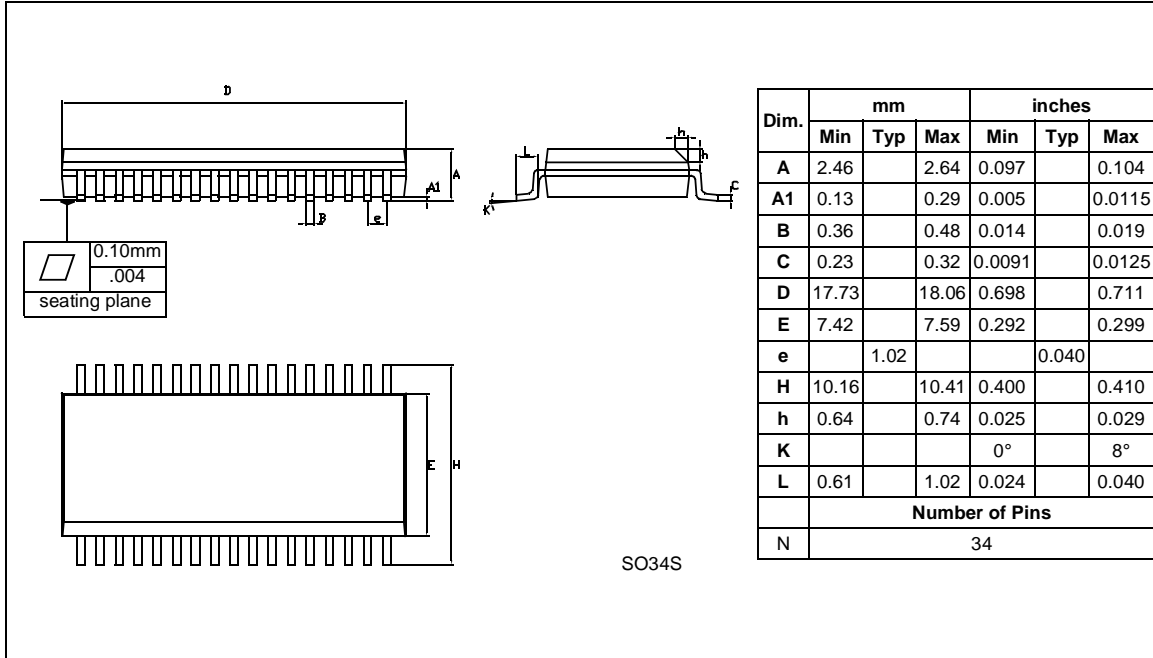


Figure 46. 32-Pin Shrink Plastic Dual in Line Package, 400-mil Width

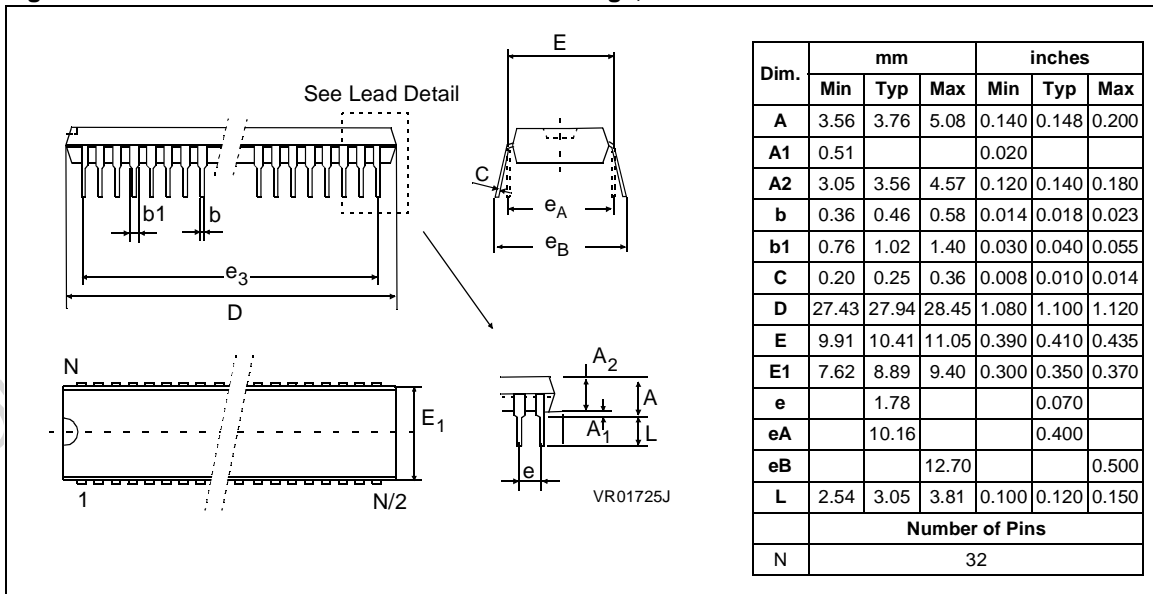
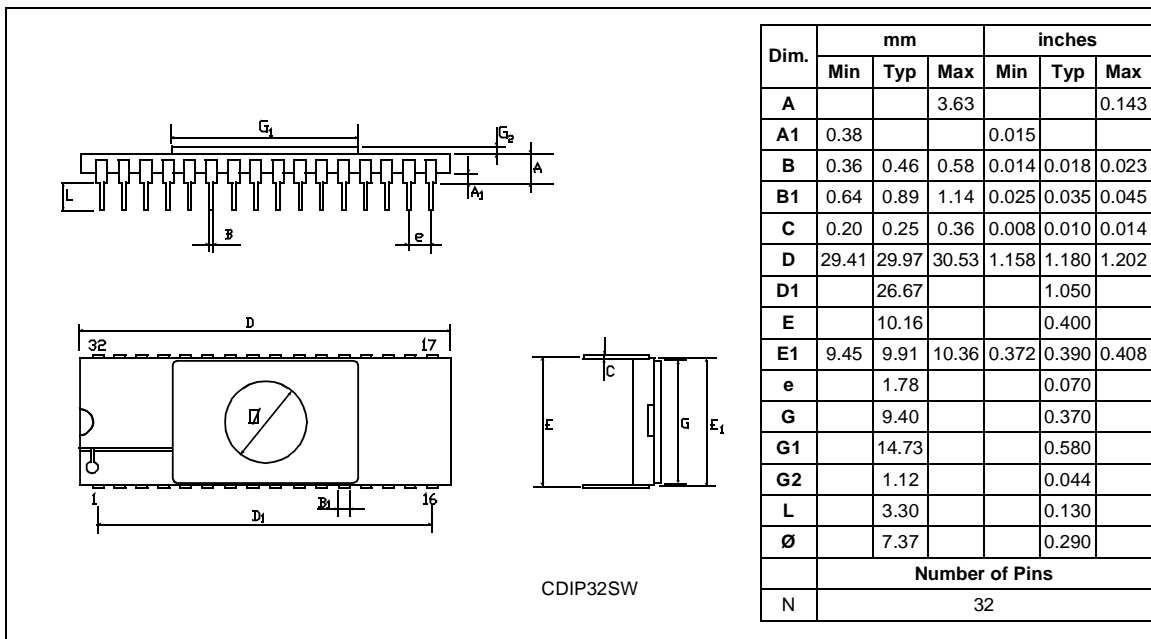




Figure 47. 32-Pin Shrink Ceramic Dual In-Line Package



## 9 DEVICE CONFIGURATION AND ORDERING INFORMATION

The following section deals with the procedure for transfer of customer codes to STMicroelectronics.

### 9.1 DEVICE ORDERING INFORMATION AND TRANSFER OF CUSTOMER CODE

Customer code is made up of the ROM contents. The ROM contents are to be sent on diskette, or by electronic means, with the hexadecimal file in .S19 format generated by the development tool. All unused bytes must be set to FFh.

The customer code should be communicated to STMicroelectronics with the correctly completed OPTION LIST appended.

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

Figure 48. Sales Type Coding Rules

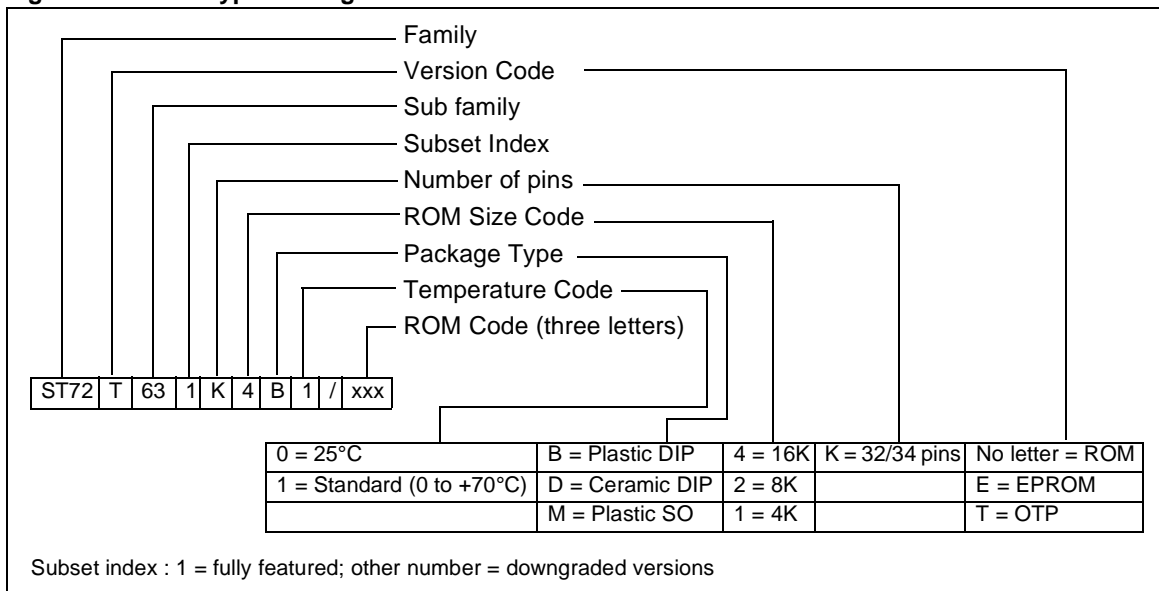


Table 26. Ordering Information

Sales Type <sup>1)</sup>	Program Memory (bytes)	RAM (bytes)	Package
ST72E631K4D0	16K EPROM	512	CSDIP32
ST72631K4M1/xxx	16K ROM		SO34
ST72T631K4M1	16K OTP		PSDIP32
ST72631K4B1/xxx	16K ROM		
ST72T631K4B1	16K OTP	256	SO34
ST72632K2M1/xxx	8K ROM		PSDIP32
ST72T632K2M1	8K OTP		
ST72632K2B1/xxx	8K ROM		
ST72T632K2B1	8K OTP	256	SO34
ST72633K1M1/xxx	4K ROM		PSDIP32
ST72T633K1M1	4K OTP		
ST72633K1B1/xxx	4K ROM		
ST72T633K1B1	4K OTP		

Note 1. /xxx stands for the ROM code name assigned by STMicroelectronics.

Table 27. Development Tools

Development Tool	Sales Type	Remarks
Real time emulator	ST7263-EMU2	
EPROM Programming Board	ST72E63-EPB/EU	220V Power Supply
	ST72E63-EPB/US	110V Power Supply

## ST7263X MICROCONTROLLER OPTION LIST

Customer: .....

Address: .....

.....

Contact: .....

Phone No: .....

Reference : .....

## STMicroelectronics references:

Device:  ST72631K4  
 ST72632K2  
 ST72633K1

Package:  Dual in Line Plastic  
 Small Outline Plastic  
Specify conditioning  
 Standard (stick)  
 Tape & Reel  
 Die form  
Specify conditioning  
 Inked unscribed wafers  
 Inked and scribed wafers

Special Marking:  No  Yes "\_\_\_\_\_"

For marking, one line is possible with maximum 13 characters.  
Authorized characters are letters, digits, '.', '-', '/' and spaces only.

We have checked the ROM code verification file returned to us by STMicroelectronics. It conforms exactly with the ROM code file originally supplied. We therefore authorize STMicroelectronics to proceed with device manufacture.

Signature .....

Date .....

## 9.2 ST7 APPLICATION NOTES

IDENTIFICATION	DESCRIPTION
<b>PROGRAMMING AND TOOLS</b>	
AN985	EXECUTING CODE IN ST7 RAM
AN986	USING THE ST7 INDIRECT ADDRESSING MODE
AN987	ST7 IN-CIRCUIT PROGRAMMING
AN988	STARTING WITH ST7 ASSEMBLY TOOL CHAIN
AN989	STARTING WITH ST7 HIWARE C
AN1039	ST7 MATH UTILITY ROUTINES
AN1064	WRITING OPTIMIZED HIWARE C LANGUAGE FOR ST7
AN1106	TRANSLATING ASSEMBLY CODE FROM HC05 TO ST7
<b>EXAMPLE DRIVERS</b>	
AN969	ST7 SCI COMMUNICATION BETWEEN THE ST7 AND A PC
AN970	ST7 SPI COMMUNICATION BETWEEN THE ST7 AND E <sup>2</sup> PROM
AN971	ST7 I <sup>2</sup> C COMMUNICATION BETWEEN THE ST7 AND E <sup>2</sup> PROM
AN972	ST7 SOFTWARE SPI MASTER COMMUNICATION
AN973	SCI SOFTWARE COMMUNICATION WITH A PC USING ST72251 16-BIT TIMER
AN974	REAL TIME CLOCK WITH THE ST7 TIMER OUTPUT COMPARE
AN976	DRIVING A BUZZER USING THE ST7 PWM FUNCTION
AN979	DRIVING AN ANALOG KEYBOARD WITH THE ST7 ADC
AN980	ST7 KEYPAD DECODING TECHNIQUES, IMPLEMENTING WAKE-UP ON KEYSTROKE
AN1017	USING THE ST7 USB MICROCONTROLLER
AN1041	USING ST7 PWM SIGNAL TO GENERATE ANALOG OUTPUT (SINUSOID)
AN1042	ST7 ROUTINE FOR I <sup>2</sup> C SLAVE MODE MANAGEMENT
AN1044	MULTIPLE INTERRUPT SOURCES MANAGEMENT FOR ST7 MCUS
AN1045	ST7 SOFTWARE IMPLEMENTATION OF I <sup>2</sup> C BUS MASTER
AN1046	ST7 UART EMULATION SOFTWARE
AN1047	MANAGING RECEPTION ERRORS WITH THE ST7 SCI PERIPHERAL
AN1048	ST7 SOFTWARE LCD DRIVER
AN1078	ST7 TIMER PWM DUTY CYCLE SWITCH FOR TRUE 0% or 100% DUTY CYCLE
AN1082	DESCRIPTION OF THE ST72141 MOTOR CONTROL
AN1083	ST72141 BLDC MOTOR CONTROL SOFTWARE AND FLOWCHART EXAMPLE
AN1129	PWM MANAGEMENT FOR BLDC MOTOR DRIVES USING THE ST72141
AN1130	BRUSHLESS DC MOTOR DRIVE WITH 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
AN1182	USING THE ST7 USB LOW-SPEED FIRMWARE
<b>PRODUCT OPTIMIZATION</b>	
AN982	USING CERAMIC RESONATORS WITH THE ST7
AN1014	HOW TO MINIMIZE THE ST7 POWER CONSUMPTION
AN1070	ST7 CHECKSUM SELF-CHECKING CAPABILITY
AN1179	PROGRAMMING ST7 FLASH MICROCONTROLLERS IN REMOTE ISP
<b>PRODUCT EVALUATION</b>	
AN910	ST7 AND ST9 PERFORMANCE BENCHMARKING
AN990	ST7 BENEFITS VERSUS INDUSTRY STANDARD
AN1086	ST7 / ST10U435 CAN-do SOLUTIONS FOR CAR MULTIPLEXING
AN1150	BENCHMARK ST72 VS PC16
AN1151	PERFORMANCE COMPARISON BETWEEN ST72254 & PC16F8

## 9.3 TO GET MORE INFORMATION

To get the latest information on this product please use the ST web server: <http://mcu.st.com/>

## 10 SUMMARY OF CHANGES


Description of the changes between the current release of the specification and the previous one.

Revision	Main changes	Date
1.8	Changed status of the document (datasheet instead of preliminary data). Added Section 9.2 and section 9.3 on page 108.	August 00

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 the express written approval of STMicroelectronics.

The ST logo is a registered trademark of STMicroelectronics

©2001 STMicroelectronics - All Rights Reserved.

 Purchase of I<sup>2</sup>C Components by STMicroelectronics conveys a license under the Philips I<sup>2</sup>C Patent. Rights to use these components in an I<sup>2</sup>C system is granted provided that the system conforms to the I<sup>2</sup>C Standard Specification as defined by Philips.

STMicroelectronics Group of Companies

Australia - Brazil - China - Finland - France - Germany - Hong Kong - India - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain  
Sweden - Switzerland - United Kingdom - U.S.A.

<http://www.st.com>