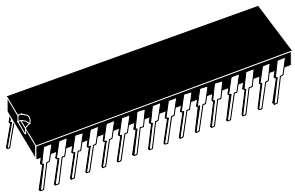
ST72251

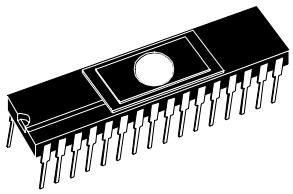
8-BIT MCU WITH 4 TO 8K ROM/OTP/EPROM, 256 BYTES RAM, ADC, WDG, SPI, I²C AND 2 TIMERS

DATASHEET

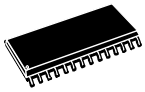
- User Program Memory (ROM/OTP/EPROM): 4 to 8K bytes
- Data RAM: 256 bytes, including 64 bytes of stack
- Master Reset and Power-On Reset
- Run, Wait, Slow and Halt modes
- 22 multifunctional bidirectional I/O lines:
  - 22 programmable interrupt inputs
  - 8 high sink outputs
  - 6 Analog alternate inputs
  - 16 Alternate Functions
  - EMI filtering
- Programmable watchdog (WDG)
- Two 16-bit Timers, each featuring:
  - 2 Input Captures
  - 2 Output Compares
  - External Clock input (on Timer A only)
  - PWM and Pulse Generator modes
- Synchronous Serial Peripheral Interface (SPI)
- Full I²C multiple Master/Slave interface
- 8-bit Analog-to-Digital converter (6 channels)
- 8-bit Data Manipulation
- 63 Basic Instructions
- 17 main Addressing Modes
- 8 x 8 Unsigned Multiply Instruction
- True Bit Manipulation
- Complete Development Support on PC/DOS-WINDOWS™ Real-Time Emulator
- Full Software Package on DOS/WINDOWS™ (C-Compiler, Cross-Assembler, Debugger)



PSDIP32



CSDIP32W



SO28

(See ordering information at the end of datasheet)

Device Summary

Features	ST72251G1	ST72251G2
Program Memory - bytes	4K	8K
RAM (stack) - bytes	256 (64)	
Peripherals	Watchdog, Timers, SPI, I²C, ADC	
Operating Supply	3 to 5.5 V	
CPU Frequency	8MHz max (16MHz oscillator) 4MHz max over 85°C	
Temperature Range	- 40°C to + 125°C	
Package	SO28 - SDIP32	

Rev. 1.9

Table of Contents

Table of Contents

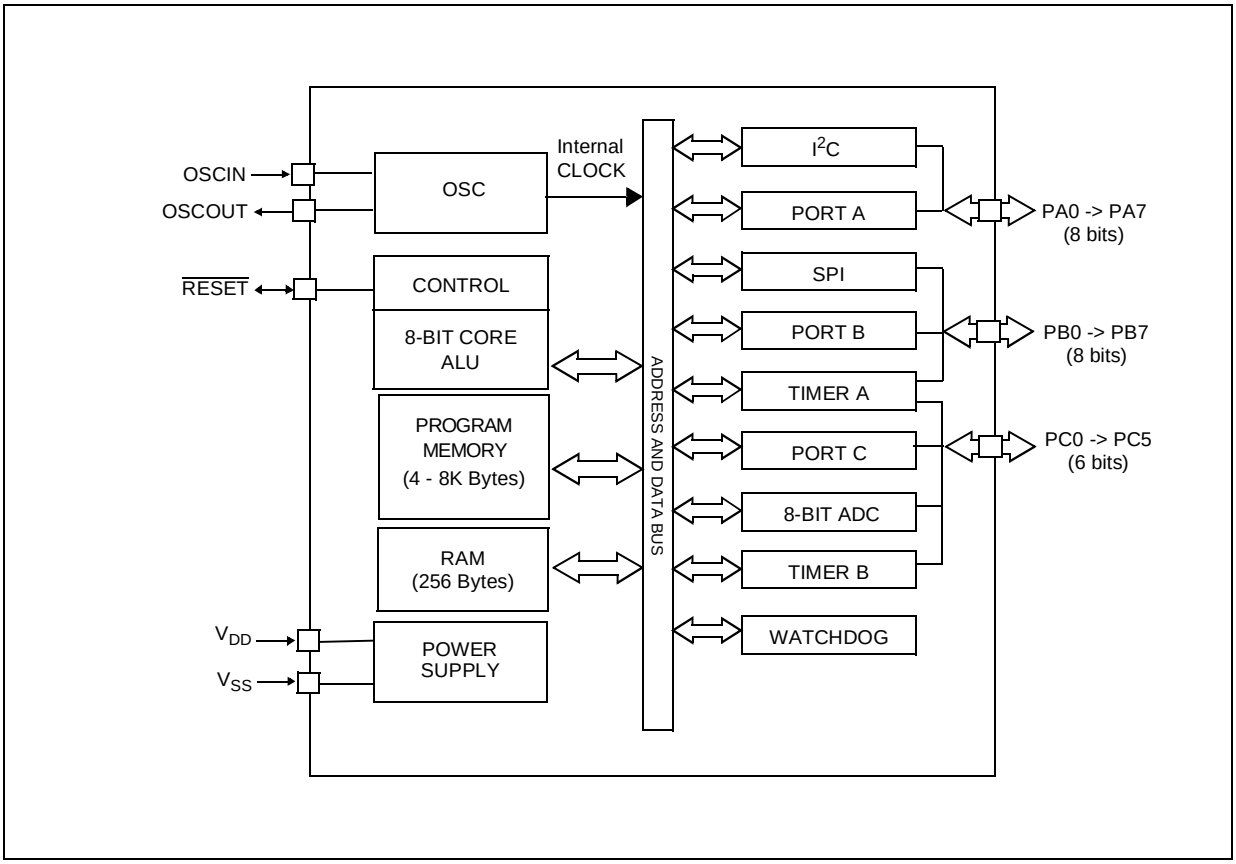
Table of Contents

1 GENERAL DESCRIPTION

1.1 INTRODUCTION

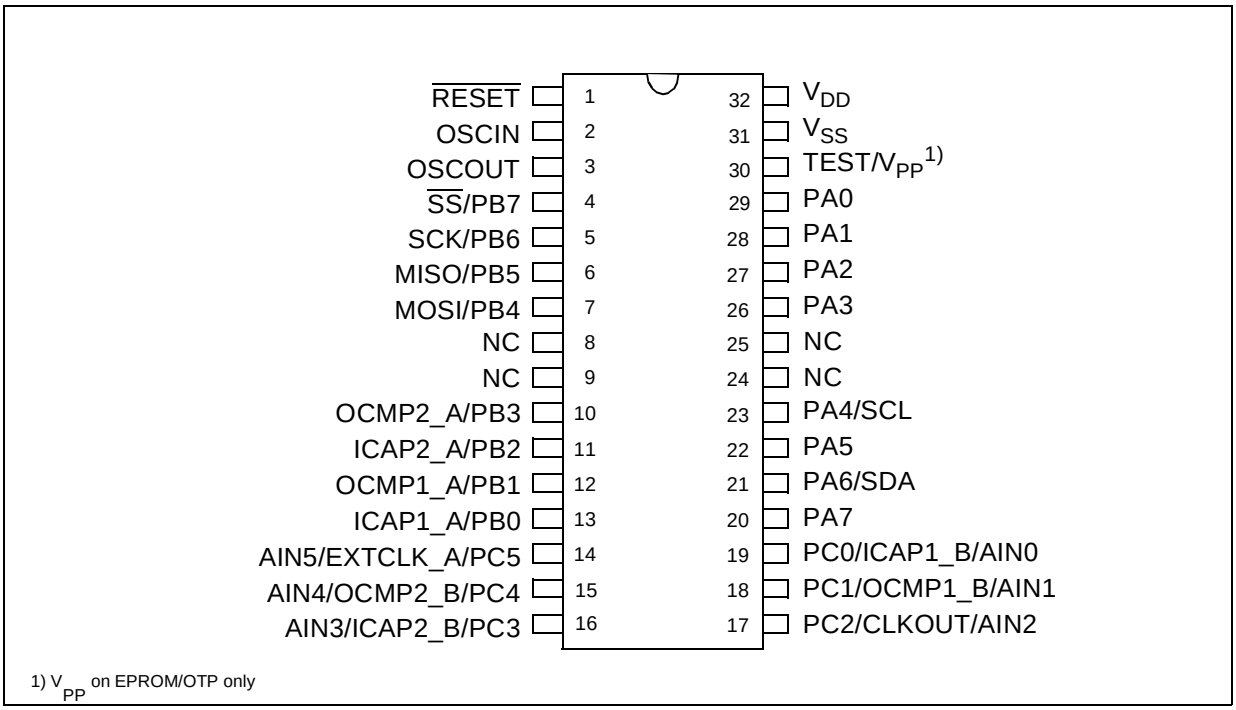
The ST72251 HCMOS Microcontroller Unit is a member of the ST7 family of Microcontrollers. The device is based on an industry-standard 8-bit core and features an enhanced instruction set. The device normally operates at a 16MHz oscillator frequency. Under software control, the ST72251 may be placed in either WAIT, SLOW 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 ST72251 features true bit manipulation, 8x8 unsigned multiplication and indirect addressing modes on the whole memory. The device includes an on-chip oscillator, CPU, program memory (ROM/OTP/EPROM versions), RAM, 22 I/O lines and the following on-chip peripherals: Analog-to-Digital converter (ADC) with 6 multiplexed analog inputs, industry standard synchronous SPI serial interface, I^2C multiple Master/ Slave interface, digital Watchdog, two independent 16-bit Timers, one featuring an External Clock Input, and both featuring Pulse Generator capabilities, 2 Input Captures and 2 Output Compares.

Figure 1. ST72251 Block Diagram



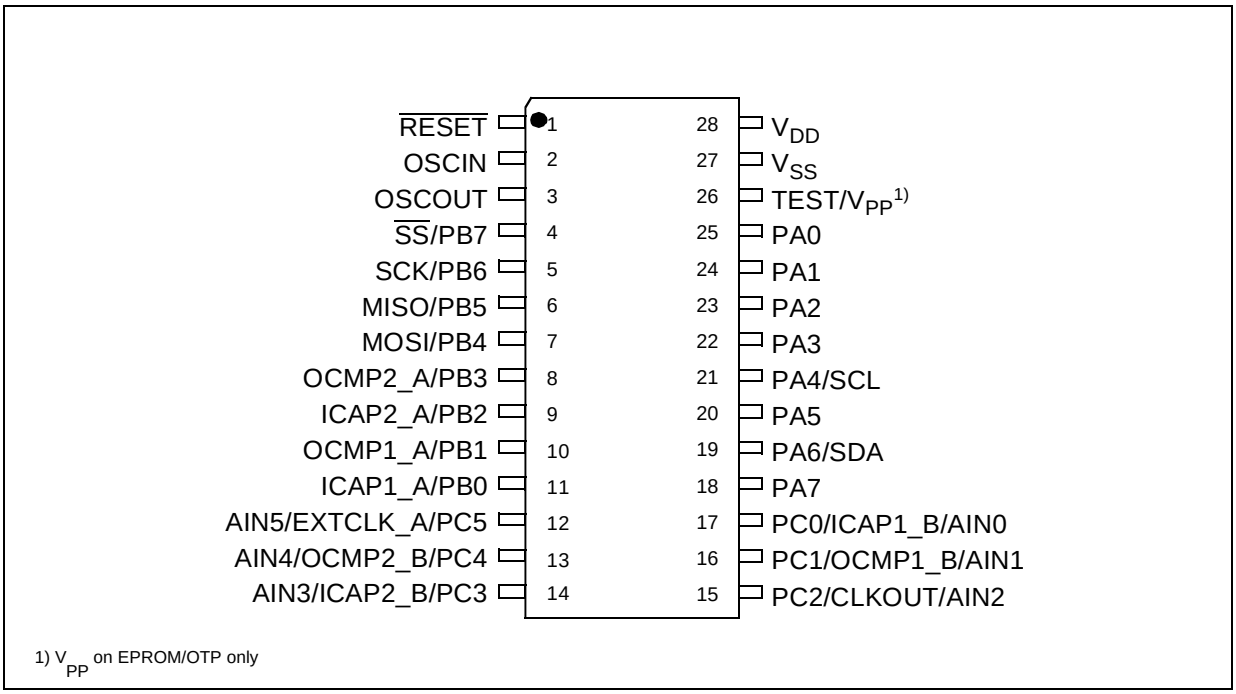
1.2 PIN DESCRIPTION

Figure 2. ST72251 Pinout (SDIP32)



1) V_{PP} on EPROM/OTP only

Figure 3. ST72251 Pinout (SO28)



1) V_{PP} on EPROM/OTP only

Table 1. ST72251 Pin Configuration

Pin n° SDIP32	Pin n° SO28	Pin Name	Type	Description	Remarks
1	1	$\overline{\text{RESET}}$	I/O	Bidirectional. Active low. Top priority non maskable interrupt.	
2	2	OSCIN	I	Input/Output Oscillator pin. These pins connect a parallel-resonant crystal, or an external source to the on-chip oscillator.	
3	3	OSCOUT	O		
4	4	PB7/$\overline{\text{SS}}$	I/O	Port B7 or SPI Slave Select (active low)	External Interrupt: EI1
5	5	PB6/SCK	I/O	Port B6 or SPI Serial Clock	External Interrupt: EI1
6	6	PB5/MISO	I/O	Port B5 or SPI Master In/ Slave Out Data	External Interrupt: EI1
7	7	PB4/MOSI	I/O	Port B4 or SPI Master Out / Slave In Data	External Interrupt: EI1
8		NC		Not Connected	
9		NC		Not Connected	
10	8	PB3/OCMP2_A	I/O	Port B3 or TimerA Output Compare 2	External Interrupt: EI1
11	9	PB2/ICAP2_A	I/O	Port B2 or TimerA Input Capture 2	External Interrupt: EI1
12	10	PB1/OCMP1_A	I/O	Port B1 or TimerA Output Compare 1	External Interrupt: EI1
13	11	PB0/ICAP1_A	I/O	Port B0 or TimerA Input Capture 1	External Interrupt: EI1
14	12	PC5/EXTCLK_A/AIN5	I/O	Port C5 or TimerA Input Clock or ADC Analog Input 5	External Interrupt: EI1
15	13	PC4/OCMP2_B/AIN4	I/O	Port C4 or TimerB Output Compare 2 or ADC Analog Input 4	External Interrupt: EI1
16	14	PC3/ICAP2_B/AIN3	I/O	Port C3 or TimerB Input Capture 2 or ADC Analog Input 3	External Interrupt: EI1
17	15	PC2/CLKOUT/AIN2	I/O	Port C2 or Internal Clock Frequency output or ADC Analog Input 2. Clockout is driven by the MCO bit of the miscellaneous register.	External Interrupt: EI1
18	16	PC1/OCMP1_B/AIN1	I/O	Port C1 or TimerB Output Compare 1 or ADC Analog Input 1	External Interrupt: EI1
19	17	PC0/ICAP1_B/AIN0	I/O	Port C0 or TimerB Input Capture 1 or ADC Analog Input 0	External Interrupt: EI1
20	18	PA7	I/O	Port A7, High Sink	External Interrupt: EI0
21	19	PA6/SDA	I/O	Port A6 or I^2C Data, High Sink	External Interrupt: EI0
22	20	PA5	I/O	Port A5, High Sink	External Interrupt: EI0
23	21	PA4/SCL	I/O	Port A4 or I^2C Clock, High Sink	External Interrupt: EI0
24		NC		Not Connected	
25		NC		Not Connected	
26	22	PA3	I/O	Port A3, High Sink	External Interrupt: EI0
27	23	PA2	I/O	Port A2, High Sink	External Interrupt: EI0
28	24	PA1	I/O	Port A1, High Sink	External Interrupt: EI0
29	25	PA0	I/O	Port A0, High Sink	External Interrupt: EI0
30	26	TEST/V_{PP}	I/S	Test mode pin (should be tied low in user mode). In the EPROM programming mode, this pin acts as the programming voltage input V_{PP}.	
31	27	V_{SS}	S	Ground	
32	28	V_{DD}	S	Main power supply	

1.3 EXTERNAL CONNECTIONS

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

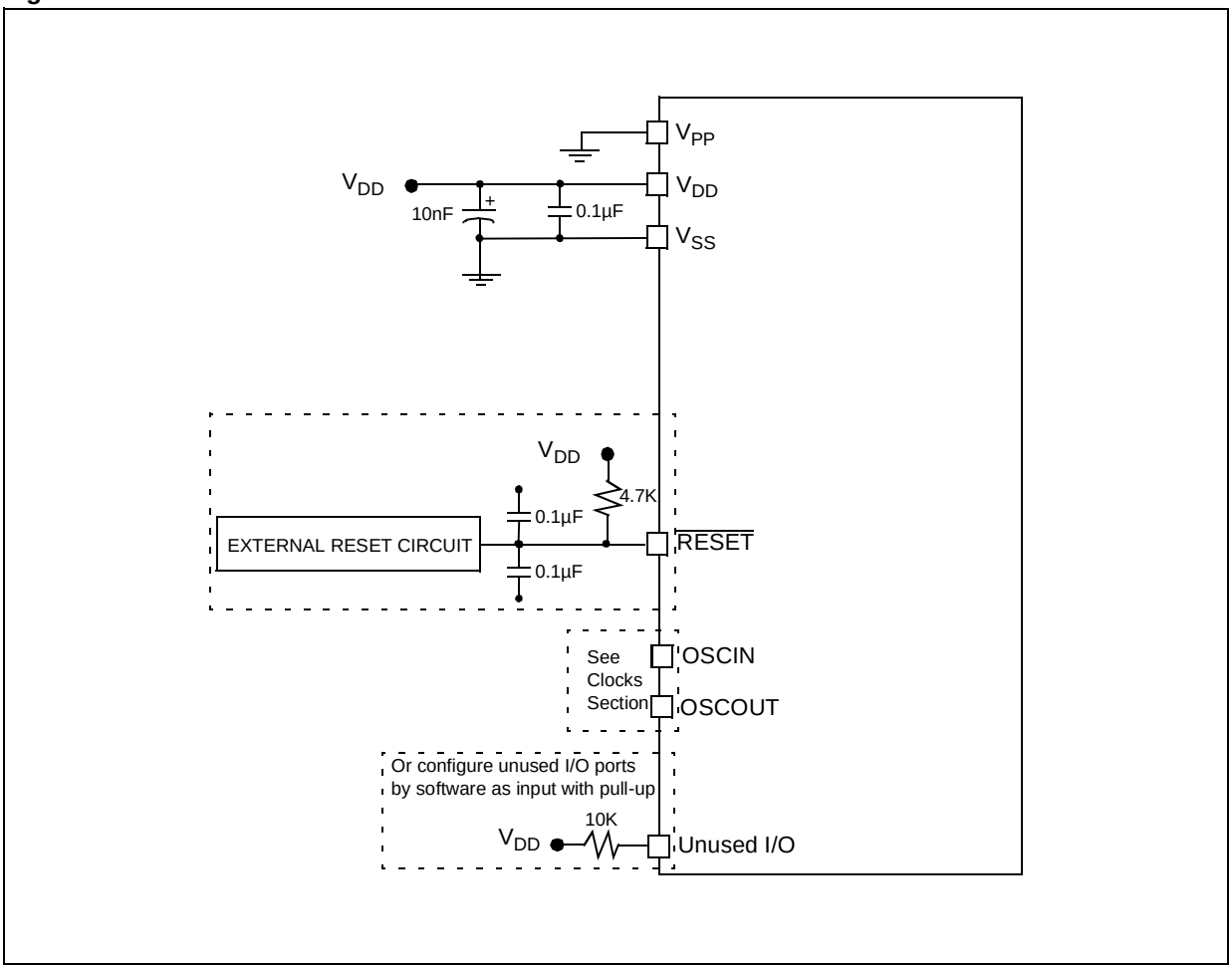
The V_{PP} pin is only used for programming OTP and EPROM devices and must be tied to ground in user mode.

The 10 nF and 0.1 µF 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 MEMORY MAP

Figure 5. Memory Map

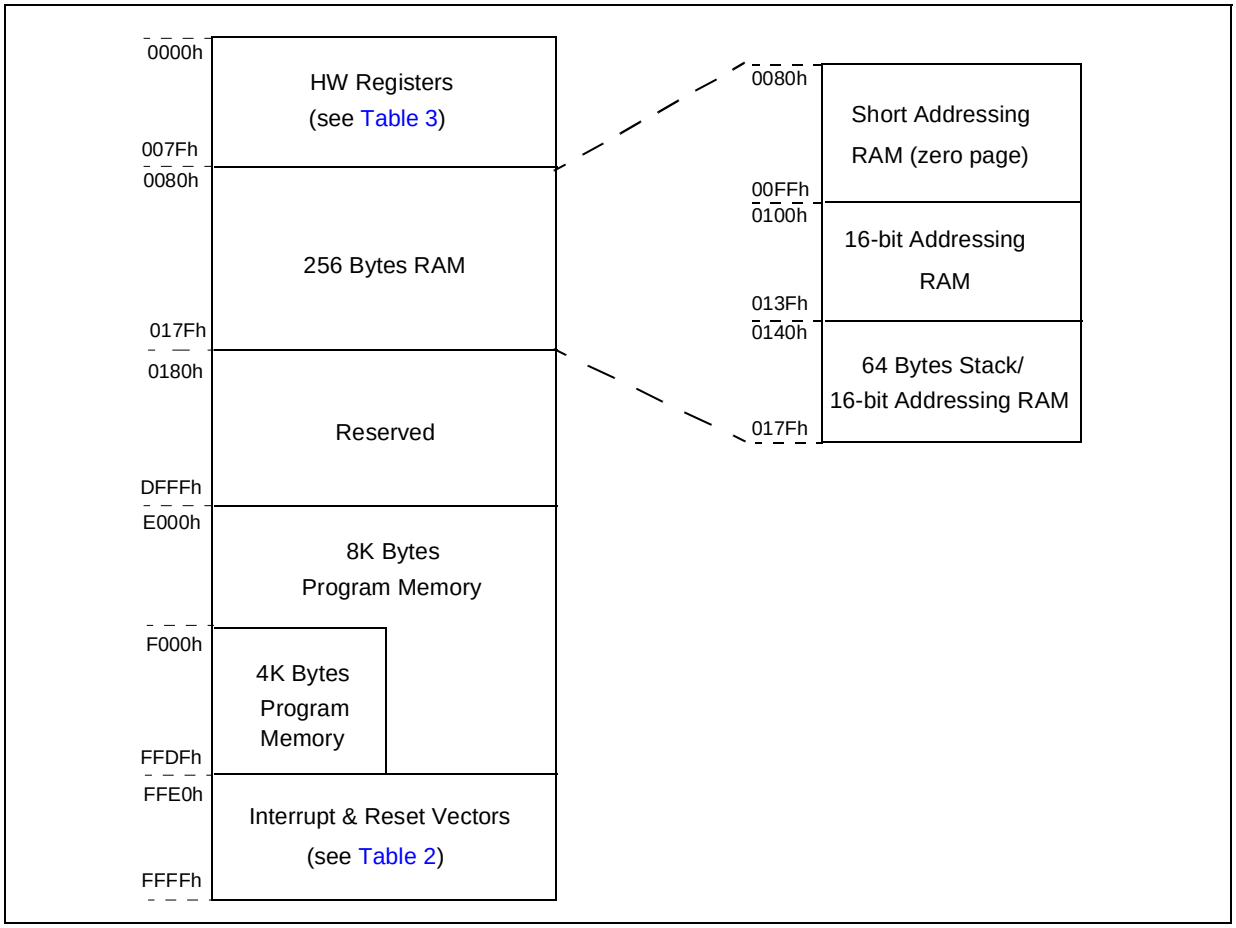


Table 2. Interrupt Vector Map

Vector Address	Description	Remarks
FFE0-FFE1h	Not Used	
FFE2-FFE3h	Not Used	
FFE4-FFE5h	I^2C Interrupt Vector	Internal Interrupt
FFE6-FFE7h	Not Used	
FFE8-FFE9h	Not Used	
FFEA-FFEBh	Not Used	
FFEC-FFEDh	Not Used	
FFEE-FFEFh	TIMER B Interrupt Vector	Internal Interrupt
FFF0-FFF1h	Not Used	
FFF2-FFF3h	TIMER A Interrupt Vector	Internal Interrupt
FFF4-FFF5h	SPI Interrupt Vector	Internal Interrupt
FFF6-FFF7h	Not Used	
FFF8-FFF9h	External Interrupt Vector EI1	External Interrupt
FFFA-FFFBh	External Interrupt Vector EI0	External Interrupt
FFFC-FFFDh	TRAP (software) Interrupt Vector	CPU Interrupt
FFFE-FFFFh	RESET Vector	

Table 3. Hardware Register Memory Map

Address	Block Name	Register Label	Register name	Reset Status	Remarks
0000h	Port C	PCDR	Data Register	00h	R/W
0001h		PCDDR	Data Direction Register	00h	R/W
0002h		PCOR	Option Register	00h	R/W
0003h	Reserved Area (1 Byte)				
0004h	Port B	PBDR	Data Register	00h	R/W
0005h		PBDDR	Data Direction Register	00h	R/W
0006h		PBOR	Option Register	00h	R/W
0007h	Reserved Area (1 Byte)				
0008h	Port A	PADR	Data Register	00h	R/W
0009h		PADDR	Data Direction Register	00h	R/W
000Ah		PAOR	Option Register	00h	R/W
000Bh to 001Fh	Reserved Area (21 Bytes)				
0020h		MISCR	Miscellaneous Register	00h	
0021h	SPI	SPIDR	Data I/O Register	xxh	R/W
0022h		SPICR	Control Register	0xh	R/W
0023h		SPISR	Status Register	00h	Read Only
0024h	WDG	WDGCR	Watchdog Control register	7Fh	R/W
0025h to 0027h	Reserved Area (3 Bytes)				
0028h	I^2C	I2CCR	Control Register	00h	R/W
0029h		I2CSR1	Status Register 1	00h	Read Only
002Ah		I2CSR2	Status Register 2	00h	Read Only
002Bh		I2CCCR	Clock Control Register	00h	R/W
002Ch		I2COAR1	Own Address Register 1	00h	R/W
002Dh		I2COAR2	Own Address Register 2	40h	R/W
002Eh		I2CDR	Data Register	00h	R/W
002Fh 0030h	Reserved Area (2 Bytes)				
0031h	Timer A	TACR2	Control Register2	00h	R/W
0032h		TACR1	Control Register1	00h	R/W
0033h		TASR	Status Register	00h	Read Only
0034h-0035h		TAIC1HR	Input Capture1 High Register	xxh	Read Only
		TAIC1LR	Input Capture1 Low Register	xxh	Read Only
0036h-0037h		TAOC1HR	Output Compare1 High Register	80h	R/W
		TAOC1LR	Output Compare1 Low Register	00h	R/W
0038h-0039h		TACHR	Counter High Register	FFh	Read Only
		TACLR	Counter Low Register	FCh	Read Only
003Ah-003Bh		TAACHR	Alternate Counter High Register	FFh	Read Only
		TAACLR	Alternate Counter Low Register	FCh	Read Only
003Ch-003Dh		TAIC2HR	Input Capture2 High Register	xxh	Read Only
		TAIC2LR	Input Capture2 Low Register	xxh	Read Only
003Eh-003Fh		TAOC2HR	Output Compare2 High Register	80h	R/W
		TAOC2LR	Output Compare2 Low Register	00h	R/W
0040h	Reserved Area (1 Byte)				

Address	Block Name	Register Label	Register name	Reset Status	Remarks
0041h		TBCR2	Control Register2	00h	R/W
0042h		TBCR1	Control Register1	00h	R/W
0043h		TBSR	Status Register	00h	Read Only
0044h-0045h		TBIC1HR	Input Capture1 High Register	xxh	Read Only
		TBIC1LR	Input Capture1 Low Register	xxh	Read Only
0046h-0047h		TBOC1HR	Output Compare1 High Register	80h	R/W
		TBOC1LR	Output Compare1 Low Register	00h	R/W
0048h-0049h	Timer B	TBCHR	Counter High Register	FFh	Read Only
		TBCLR	Counter Low Register	FCh	Read Only
004Ah-004Bh		TBACHR	Alternate Counter High Register	FFh	Read Only
		TBACLR	Alternate Counter Low Register	FCh	Read Only
004Ch-004Dh		TBIC2HR	Input Capture2 High Register	xxh	Read Only
		TBIC2LR	Input Capture2 Low Register	xxh	Read Only
004Eh-004Fh		TBOC2HR	Output Compare2 High Register	80h	R/W
		TBOC2LR	Output Compare2 Low Register	00h	R/W
0050h to 006Fh	Reserved Area (32 Bytes)				
0070h	ADC	ADCDR	Data Register	00h	Read Only
0071h		ADCCSR	Control/Status Register	00h	R/W
0072h to 007Fh	Reserved Area (14 Bytes)				

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 6 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)

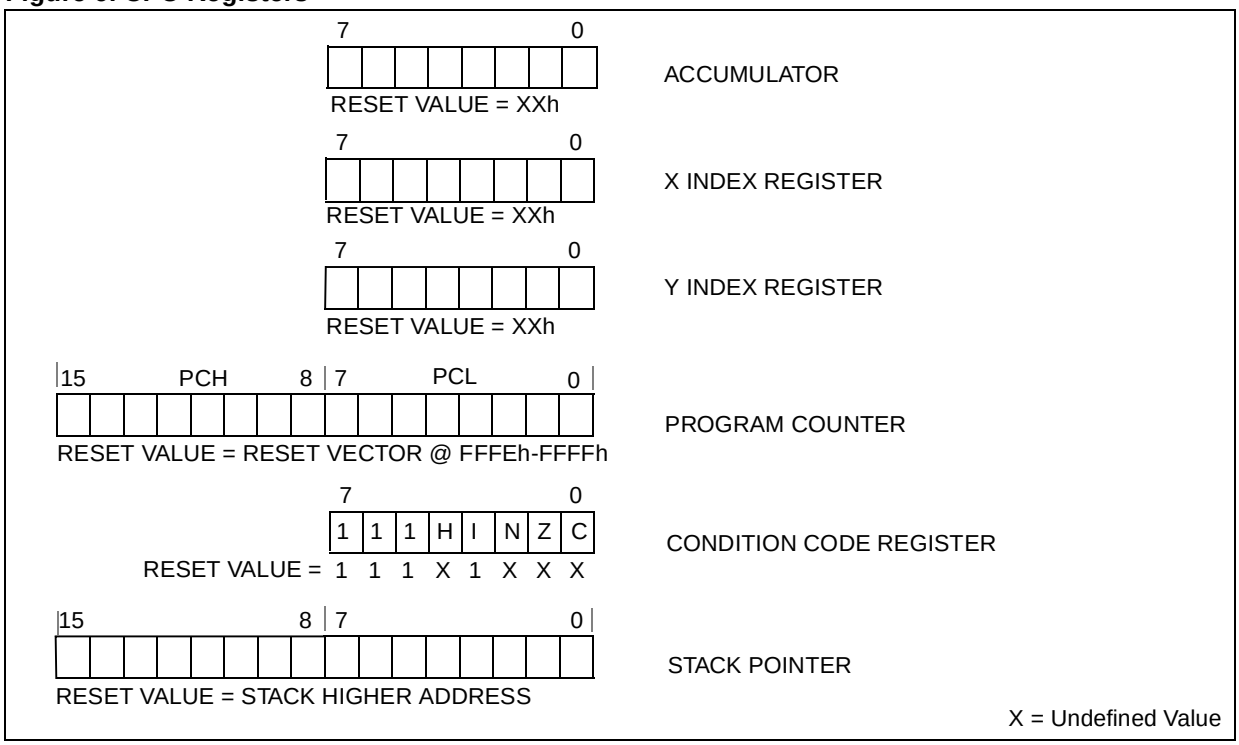
In indexed addressing modes, these 8-bit registers are used to create either effective addresses or temporary storage areas for data manipulation. (The Cross-Assembler generates a precede instruction (PRE) to indicate that the following instruction refers to the Y register.)

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^{th} 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.

CENTRAL PROCESSING UNIT (Cont'd)

Stack Pointer (SP)

Read/Write

Reset Value: 01 7Fh

15							8
0	0	0	0	0	0	0	1

7							0
0	1	SP5	SP4	SP3	SP2	SP1	SP0

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 (the 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.

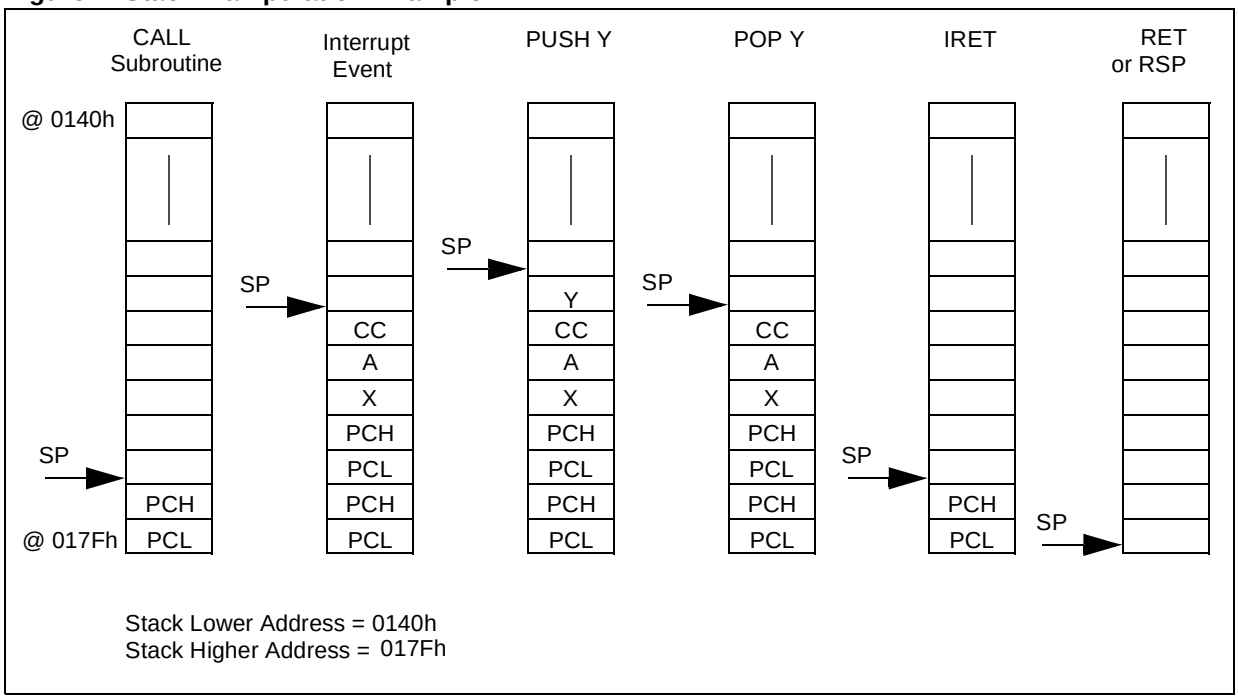
The stack is used to save the return address during a subroutine call and the CPU context during an interrupt. The user may also directly manipulate the stack by means of the PUSH and POP instructions. In the case of an interrupt, the PCL is stored at the first location pointed to by the SP. Then the 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, RESET, INTERRUPTS & POWER SAVING MODES

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}).

The external Oscillator clock is first divided by 2, and a division factor of 32 can be applied if Slow Mode is selected by setting the SMS bit in the Miscellaneous Register. This reduces the frequency of the f_{CPU}; the clock signal is also routed to the on-chip peripherals.

The internal oscillator is designed to operate with an AT-cut parallel resonant quartz crystal resonator in the frequency range specified for f_{osc}. The circuit shown in Figure 9 is recommended when using a crystal, and Table 4 lists the recommended capacitance and feedback resistance values. 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.

Use of an external CMOS oscillator is recommended when crystals outside the specified frequency ranges are to be used.

Table 4. .Recommended Values for 16 MHz Crystal Resonator (C_0 < 7pF)

R_{SMAX}	40 Ω	60 Ω	150 Ω
C_{OSCIN}	56pF	47pF	22pF
C_{OSCOUT}	56pF	47pF	22pF

C_0: parasitic shunt capacitance of the quartz crystal.
R_{SMAX}: equivalent serial resistor of the crystal (uper limit, see crystal specification).

C_{OSCOUT}, C_{OSCIN}: maximum total capacitance on OSCIN and OSCOUT, including the external capacitance plus the parasitic capacitance of the board and the device.

Figure 8. External Clock Source Connections

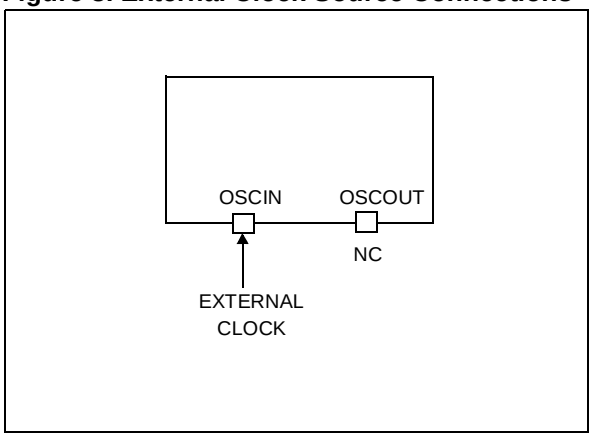


Figure 9. Crystal/Ceramic Resonator

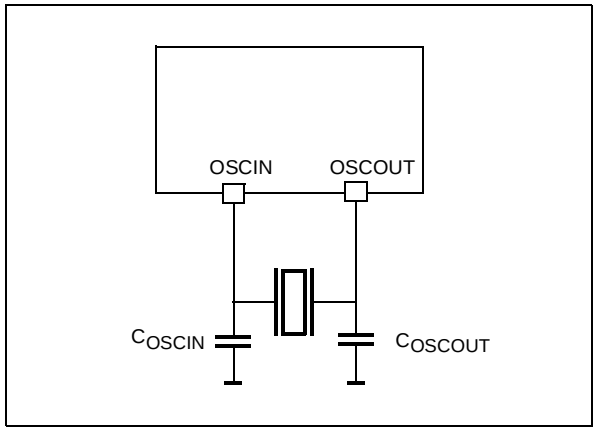
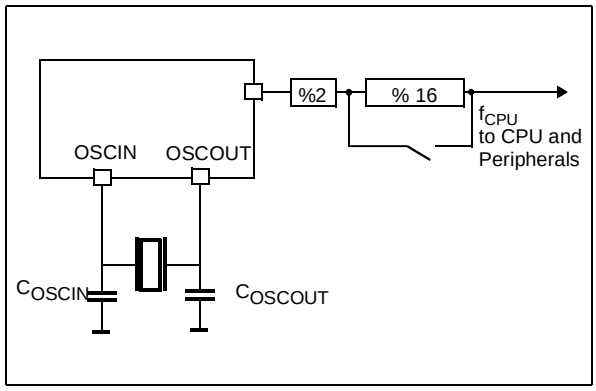


Figure 10. Clock Prescaler Block Diagram



3.2 RESET

3.2.1 Introduction

There are three sources of Reset:

– $\overline{\text{RESET}}$ pin (external source)

– Power-On Reset (Internal source)

– WATCHDOG (Internal Source)

The Reset Service Routine vector is located at address FFFEh-FFFFh.

3.2.2 External Reset

The $\overline{\text{RESET}}$ pin is both an input and an open-drain output with integrated pull-up resistor. When one of the internal Reset sources is active, the Reset pin is driven low, for a duration of $t_{RESET,}$ to reset the whole application.

3.2.3 Reset Operation

The duration of the Reset state is a minimum of 4096 internal CPU Clock cycles. During the Reset state, all I/Os take their reset value.

A Reset signal originating from an external source must have a duration of at least t_{PULSE} in order to be recognised. This detection is asynchronous and therefore the MCU can enter Reset state even in Halt mode.

At the end of the Reset cycle, the MCU may be held in the Reset state by an External Reset signal. The $\overline{\text{RESET}}$ pin may thus be used to ensure V_{DD} has risen to a point where the MCU can operate correctly before the user program is run. Following a Reset event, or after exiting Halt mode, a 4096 CPU Clock cycle delay period is initiated in order to allow the oscillator to stabilise and to ensure that recovery has taken place from the Reset state.

In the high state, the $\overline{\text{RESET}}$ pin is connected internally to a pull-up resistor (R_{ON}). This resistor can be pulled low by external circuitry to reset the device.

The $\overline{\text{RESET}}$ pin is an asynchronous signal which plays a major role in EMS performance. In a noisy environment, it is recommended to use the external connections shown in Figure 4.

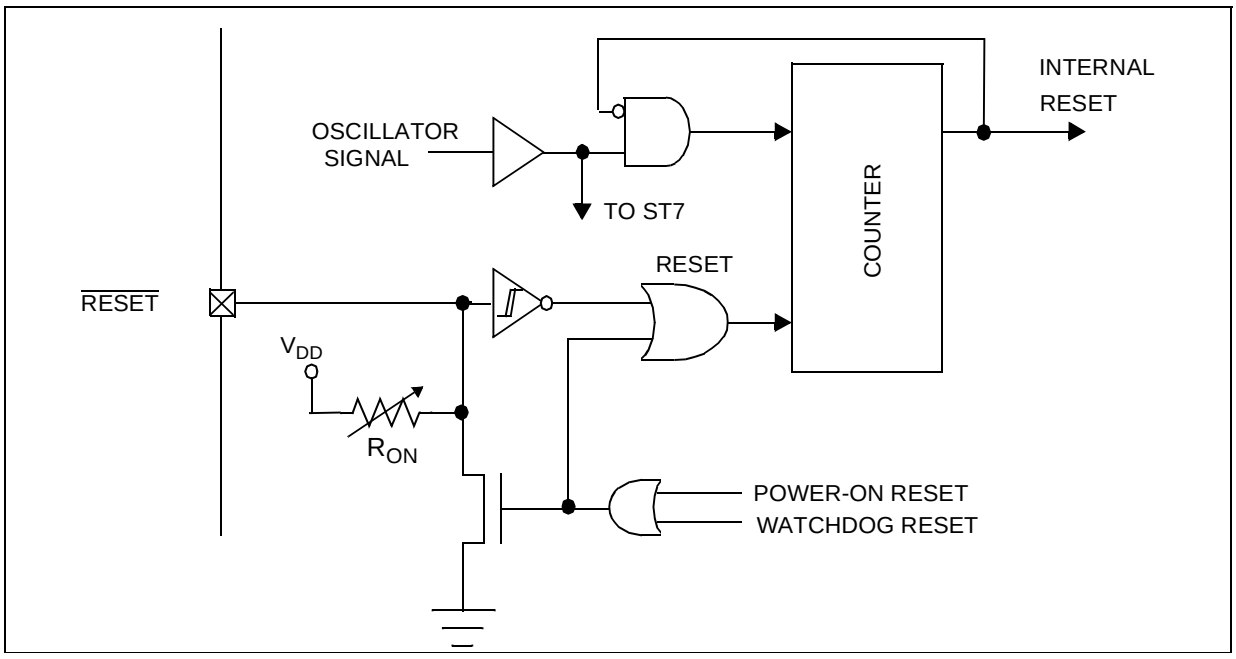
3.2.4 Power-on Reset

This circuit detects the ramping up of V_{DD}, and generates a pulse that is used to reset the application (at approximately V_{DD}= 2V).

Power-On Reset is designed exclusively to cope with power-up conditions, and should not be used in order to attempt to detect a drop in the power supply voltage.

Caution: *to re-initialize the Power-On Reset, the power supply must fall below approximately 0.8V (Vtn), prior to rising above 2V. If this condition is not respected, on subsequent power-up the Reset pulse may not be generated. An external Reset pulse may be required to correctly reactivate the circuit.*

Figure 11. Reset Block Diagram



4 INTERRUPTS

The ST7 core may be interrupted by one of two different methods: maskable hardware interrupts as listed in the Interrupt Mapping Table and a non-maskable software interrupt (TRAP). The Interrupt processing flowchart is shown in Figure 12.
The maskable interrupts must be enabled by clearing the I bit in order to be serviced. However, disabled interrupts may be latched and processed when they are enabled (see external interrupts subsection).

Note: After reset, all interrupts are disabled.

When an interrupt has to be serviced:

– Normal processing is suspended at the end of the current instruction execution.

– The PC, X, A and CC registers are saved onto the stack.

– The I bit of the CC register is set to prevent additional interrupts.

– The PC is then loaded with the interrupt vector of the interrupt to service and the first instruction of the interrupt service routine is fetched (refer to the Interrupt Mapping Table for vector addresses).

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

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

Priority Management

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

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

Interrupts and Low Power Mode

All interrupts allow the processor to leave the WAIT low power mode. Only external and specifically mentioned interrupts allow the processor to leave the HALT low power mode (refer to the “Exit from HALT“ column in the Interrupt Mapping Table).

4.1 NON MASKABLE SOFTWARE INTERRUPT

This interrupt is entered when the TRAP instruction is executed regardless of the state of the I bit. It will be serviced according to the flowchart on Figure 12.

4.2 EXTERNAL INTERRUPTS

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

The external interrupt polarity is selected through the miscellaneous register or interrupt register (if available).

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

If several input pins, connected to the same interrupt vector, are configured as interrupts, their signals are logically ANDed and inverted before entering the edge/level detection block.

Caution: The type of sensitivity defined in the Miscellaneous or Interrupt register (if available) applies to the ei source. In case of an ANDed source (as described on the I/O ports section), a low level on an I/O pin configured as input with interrupt, masks the interrupt request even in case of rising-edge sensitivity.

4.3 PERIPHERAL INTERRUPTS

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

– The I bit of the CC register is cleared.

– The corresponding enable bit is set in the control register.

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

Clearing an interrupt request is done by:

– Writing “0” to the corresponding bit in the status register or

– Access to the status register while the flag is set followed by a read or write of an associated register.

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

INTERRUPTS (Cont'd)

Figure 12. Interrupt Processing Flowchart

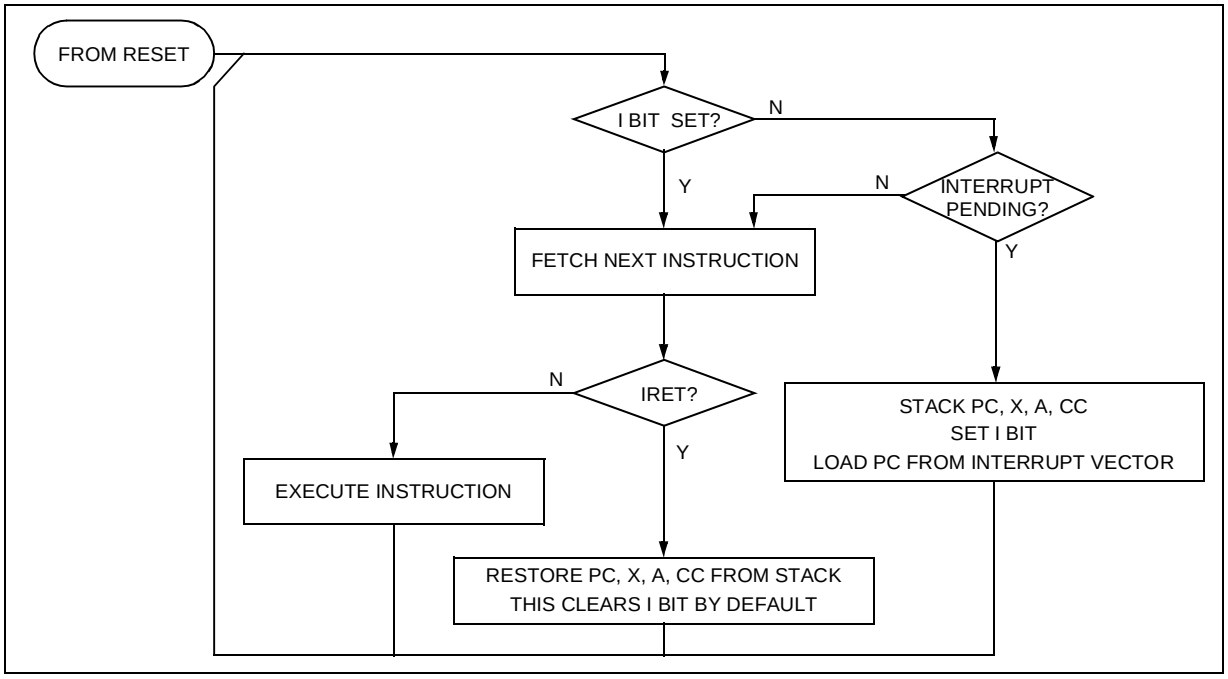


Table 5. Interrupt Mapping

Source Block	Description	Register Label	Flag	Exit from HALT	Vector Address	Priority Order
RESET	Reset	N/A	N/A	yes	FFFEh-FFFFh	Highest Priority
TRAP	Software	N/A	N/A	no	FFFCh-FFFDh	
EI0	External Interrupt PA0:PA7	N/A	N/A	yes	FFFAh-FFFBh	
EI1	External Interrupt PB0:PB7, PC0:PC5	N/A	N/A	yes	FFF8h-FFF9h	
Not Used					FFF6h-FFF7h	
SPI	Transfer Complete	SPISR	SPIF	no	FFF4h-FFF5h	
	Mode Fault		MODF			
TIMER A	Input Capture 1	TASR	ICF1_A	no	FFF2h-FFF3h	
	Output Compare 1		OCF1_A			
	Input Capture 2		ICF2_A			
	Output Compare 2		OCF2_A			
	Timer Overflow		TOF_A			
Not Used					FFF0h-FFF1h	
TIMER B	Input Capture 1	TBSR	ICF1_B	no	FFEEh-FFEFh	
	Output Compare 1		OCF1_B			
	Input Capture 2		ICF2_B			
	Output Compare 2		OCF2_B			
	Timer Overflow		TOF_B			
Not Used					FFECh-FFEDh	
					FFEAh-FFEBh	
					FFE8h-FFE9h	
					FFE6h-FFE7h	
I2C	I2C Peripheral Interrupts	I2CSR1 I2CSR2	**	no	FFE4h-FFE5h	
Not Used					FFE2h-FFE3h	Lowest Priority
					FFE0h-FFE1h	

** Many flags can cause an interrupt, see peripheral interrupt status register description.

4.4 POWER SAVING MODES

4.4.1 Introduction

There are three Power Saving modes. Slow Mode is selected by setting the relevant bits in the Miscellaneous register. Wait and Halt modes may be entered using the WFI and HALT instructions.

4.4.2 Slow Mode

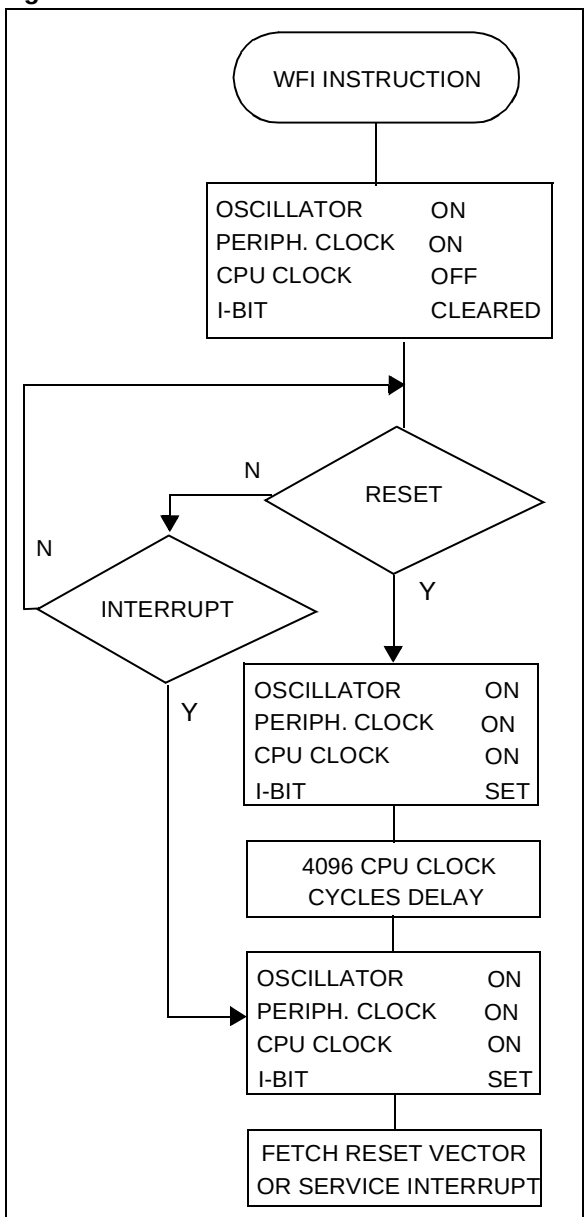
In Slow mode, the oscillator frequency can be divided by a value defined in the Miscellaneous Register. The CPU and peripherals are clocked at this lower frequency. Slow mode is used to reduce power consumption, and enables the user to adapt clock frequency to available supply voltage.

4.4.3 Wait Mode

Wait mode places the MCU in a low power consumption mode by stopping the CPU. All peripherals remain active. During Wait mode, the I bit (CC Register) is cleared, so as to enable all interrupts. All other registers and memory remain unchanged. The MCU will remain 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 13 below.

Figure 13. WAIT Flow Chart



Note: Before servicing an interrupt, the CC register is pushed on the stack. The I-Bit is set during the interrupt routine and cleared when the CC register is popped.

POWER SAVING MODES (Cont'd)

4.4.4 Halt Mode

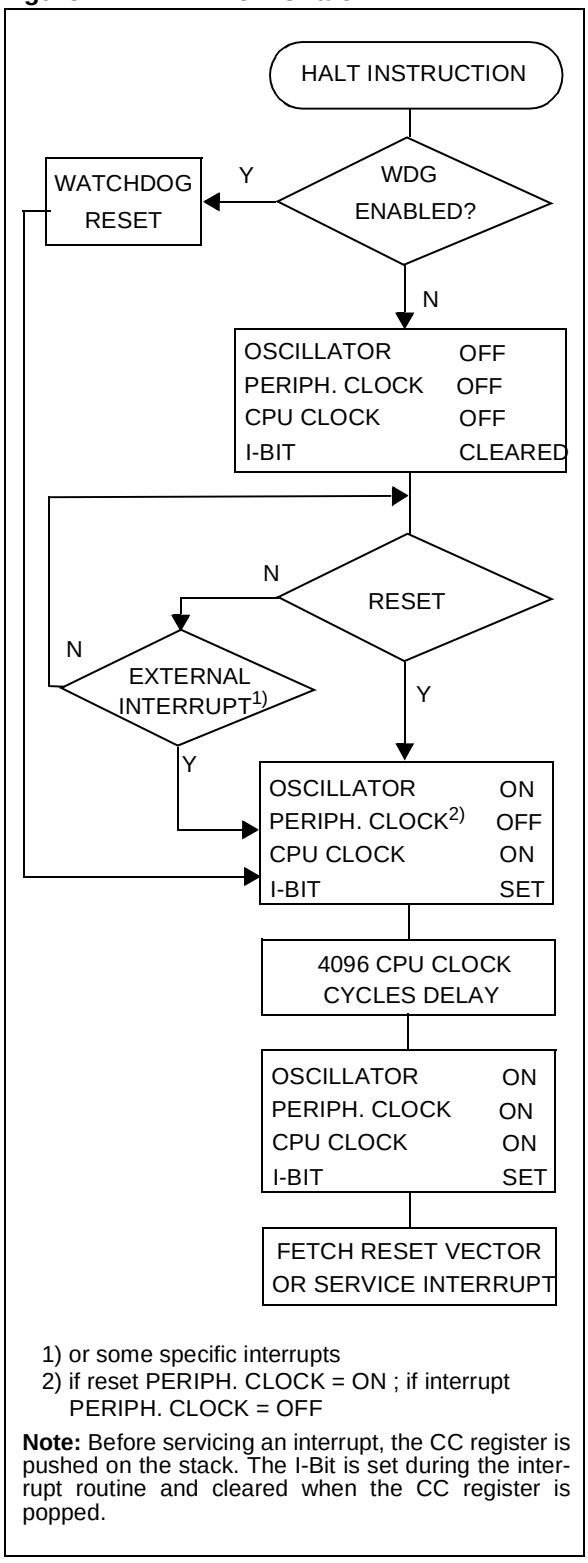
The Halt mode is the MCU lowest power consumption mode. The Halt mode is entered by executing the HALT instruction. The internal oscillator is then turned off, causing all internal processing to be stopped, including the operation of the on-chip peripherals. The Halt mode cannot be used when the watchdog is enabled, if the HALT instruction is executed while the watchdog system is enabled, a watchdog reset is generated thus resetting the entire MCU.

When entering Halt mode, the I bit in the CC Register is cleared so as to enable External Interrupts. If an interrupt occurs, the CPU becomes active.

The MCU can exit the Halt mode upon reception of an interrupt or a reset. Refer to the Interrupt Mapping Table. 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 14. HALT Flow Chart



1) or some specific interrupts
2) if reset PERIPH. CLOCK = ON ; if interrupt PERIPH. CLOCK = OFF

Note: Before servicing an interrupt, the CC register is pushed on the stack. The I-Bit is set during the interrupt routine and cleared when the CC register is popped.

4.5 MISCELLANEOUS REGISTER

The Miscellaneous register allows to select the SLOW operating mode, the polarity of external interrupt requests and to output the internal clock.

Register Address: 0020h — Read/Write

Reset Value: 0000 0000 (00h)

7							0
PEI3	PEI2	MCO	PEI1	PEI0	-	-	SMS

Bit 7:6 = **PEI[3:2]** *External Interrupt EI1 Polarity Option*.

These bits are set and cleared by software. They determine which event on EI1 causes the external interrupt according to Table 6.

Refer to the Pin Description Table at the beginning of this document for the list of pins connected to EI1.

Table 6. EI1 External Interrupt Polarity Options

MODE	PEI3	PEI2
Falling edge and low level (Reset state)	0	0
Falling edge only	1	0
Rising edge only	0	1
Rising and falling edge	1	1

Note: Any modification of one of these two bits resets the interrupt request related to this interrupt vector.

Bit 5 = **MCO** *Main Clock Out*
This bit is set and cleared by software. When set, it enables the output of the Internal Clock on the PC2 I/O port.

0 - PC2 is a general purpose I/O port.
1 - MCO alternate function (f_{CPU} is output on PC2 pin).

Bit 4:3 = **PEI[1:0]** *External Interrupt EI0 Polarity Option*.

These bits are set and cleared by software. They determine which event on EI0 causes the external interrupt according to Table 7.

Refer to the Pin Description Table at the beginning of this document for the list of pins connected to EI0.

Table 7. EI0 External Interrupt Polarity Options

MODE	PEI1	PEI0
Falling edge and low level (Reset state)	0	0
Falling edge only	1	0
Rising edge only	0	1
Rising and falling edge	1	1

Note:

Any modification of one of these two bits resets the interrupt request related to this interrupt vector.

Bit 1:2 = Unused, always read at 0.

Warning: Software must write 1 to these bits for compatibility with future products.

Bit 0 = **SMS** *Slow Mode Select*

This bit is set and cleared by software.
0- Normal mode - f_{CPU} = Oscillator frequency / 2 (Reset state)

1- Slow mode - f_{CPU} = Oscillator frequency /32

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)

and some of them to an optional register:

– Option Register (OR)

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

The following description takes into account the OR register, for specific ports which do not provide this register refer to the I/O Port Implementation Section 5.1.3. The generic I/O block diagram is shown on Figure 16.

5.1.2.1 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.

Different input modes can be selected by software through the OR register.

Notes:

1. All the inputs are triggered by a Schmitt trigger.

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 Interrupt request to the CPU. The interrupt polarity is given independently according to the description mentioned in the Miscellaneous register or in the interrupt register (where available).

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 several input pins are configured as inputs to the same interrupt vector, their signals are logically ANDed before entering the edge/level detection block. For this reason if one of the interrupt pins is tied low, it masks the other ones.

5.1.2.2 Output Mode

The pin is configured in output mode by setting the corresponding DDR register bit.

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.

5.1.2.3 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)

5.1.2.4 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.

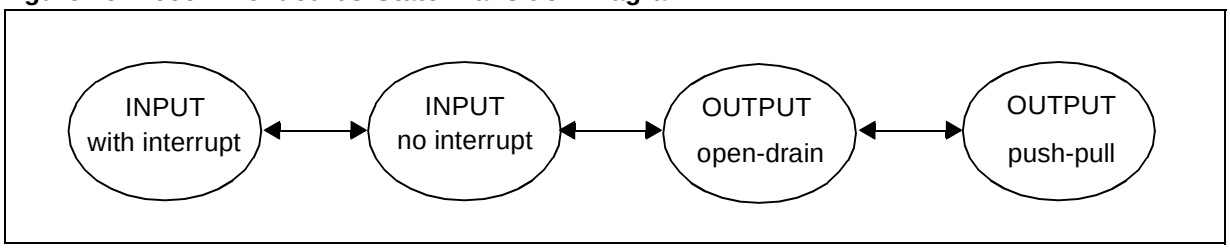
It is recommended not to change the voltage level or loading on any port pin while conversion is in progress. Furthermore it is recommended not to have clocking pins located close to a selected analog pin.

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 and OR registers and specific feature of the I/O port such as ADC Input (see Figure 16) or true open drain. Switching these I/O ports from one state to another should be done in a sequence that prevents unwanted side effects. Recommended safe transitions are illustrated in Figure 15. Other transitions are potentially risky and should be avoided, since they are likely to present unwanted side-effects such as spurious interrupt generation.

Figure 15. Recommended I/O State Transition Diagram



I/O PORTS (Cont'd)

Figure 16. I/O Block Diagram

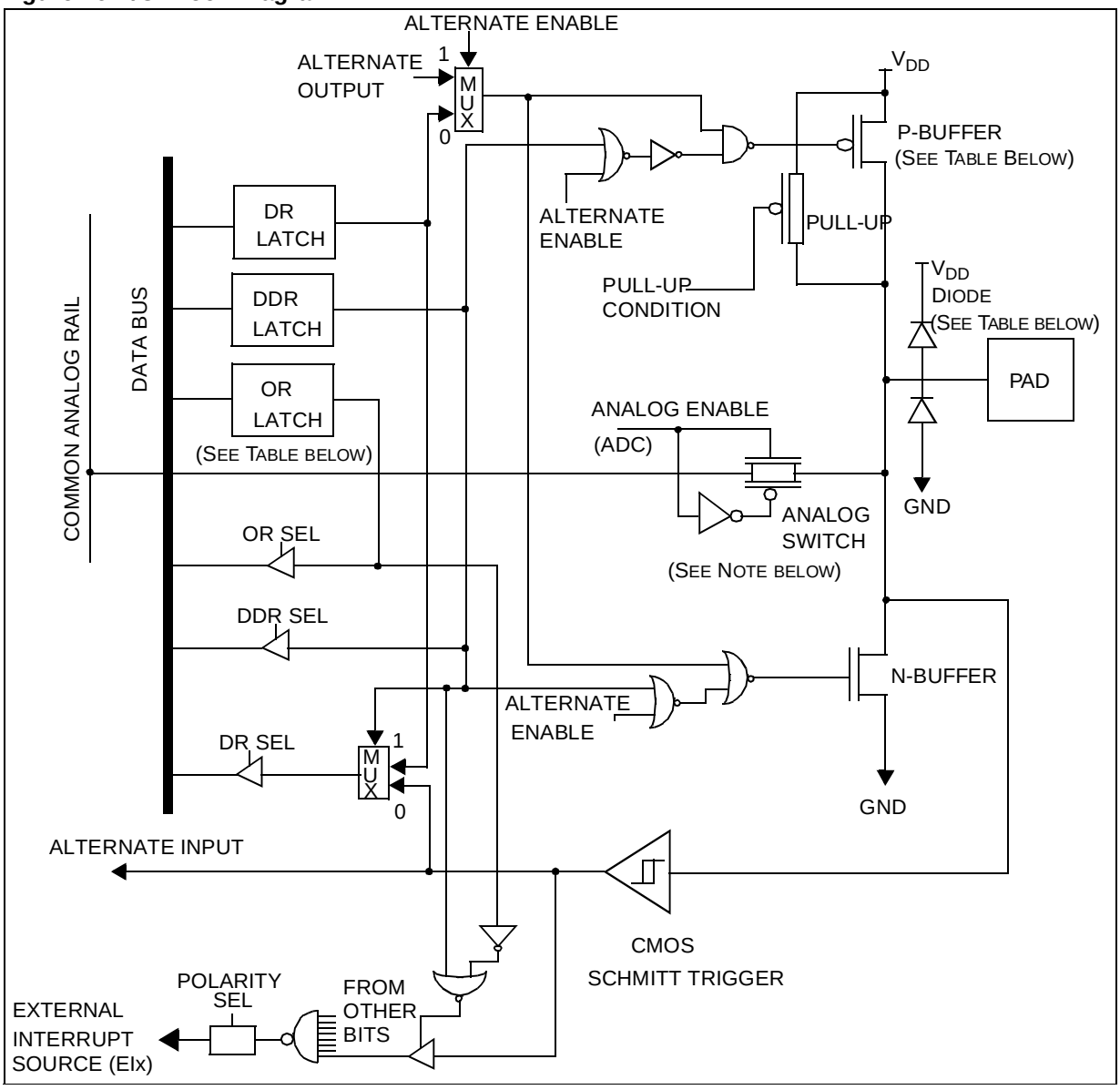


Table 8. Port Mode Configuration

Configuration Mode	Pull-up	P-buffer	V_{DD} Diode
Floating	0	0	1
Pull-up	1	0	1
Push-pull	0	1	1
True Open Drain	not present	not present	not present in OTP and EPROM devices
Open Drain (logic level)	0	0	1

Legend:
0 - present, not activated
1 - present and activated

Notes:
– No OR Register on some ports (see register map).
– ADC Switch on ports with analog alternate functions.

Table 9. Port Configuration

Port	Pin Name	Input (DDR = 0)		Output (DDR = 1)	
		OR = 0	OR = 1	OR = 0	OR = 1
Port A	PA0:PA7	Floating*	Floating with Interrupt	True Open Drain, High Sink Capability	Reserved
Port B	PB0:PB7	Floating*	Pull-up with Interrupt	Open Drain (Logic level)	Push-pull
Port C	PC0:PC5	Floating*	Pull-up with Interrupt	Open Drain (Logic level)	Push-pull

* Reset State

I/O PORTS (Cont'd)

5.1.4 Register Description

5.1.4.1 Data registers

Port A Data Register (PADR)

Port B Data Register (PBDR)

Port C Data Register (PCDR)

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

7							0
D7	D6	D5	D4	D3	D2	D1	D0

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).

5.1.4.2 Data direction registers

Port A Data Direction Register (PADDR)

Port B Data Direction Register (PBDDR)

Port C Data Direction Register (PCDDR)

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

7							0
DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0

Bit 7:0 = DD7-DD0 *Data Direction Register 8 bits.*

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

0: Input mode
1: Output mode

5.1.4.3 Option registers

Port A Option Register (PAOR)

Port B Option Register (PBOR)

Port C Option Register (PCOR)

Read/Write
Reset Value: 0000 0000 (00h) (no interrupt)

7							0
O7	O6	O5	O4	O3	O2	O1	O0

Bit 7:0 = O7-O0 *Option Register 8 bits.*

For specific I/O pins, this register is not implemented. In this case the DDR register is enough to select the I/O pin configuration.

The OR register allow to distinguish: in input mode if the interrupt capability or the floating configuration is selected, in output mode if the push-pull or open drain configuration is selected.

Each bit is set and cleared by software.

Input mode:

0: floating input
1: input interrupt with or without pull-up

Output mode (only for PB0:PB7, PC0:PC5):

0: output open drain (with P-Buffer inactivated)
1: output push-pull

Output mode (only for PA0:PA7):

0: output open drain

1: reserved

I/O PORTS (Cont'd)

Table 10. I/O Port Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0000h	**PCDR** Reset Value	D7 0	D6 0	D5 0	D4 0	D37 0	D2 0	D1 0	D0 0
0001h	**PCDDR** Reset Value	DD7 0	DD6 0	DD5 0	DD4 0	DD3 0	DD2 0	DD1 0	DD0 0
0002h	**PCOR** Reset Value	O7 0	O6 0	O5 0	O4 0	O3 0	O2 0	O1 0	O0 0
0004h	**PBDR** Reset Value	D7 0	D6 0	D5 0	D4 0	D37 0	D2 0	D1 0	D0 0
0005h	**PBDDR** Reset Value	DD7 0	DD6 0	DD5 0	DD4 0	DD3 0	DD2 0	DD1 0	DD0 0
0006h	**PBOR** Reset Value	O7 0	O6 0	O5 0	O4 0	O3 0	O2 0	O1 0	O0 0
0008h	**PADR** Reset Value	D7 0	D6 0	D5 0	D4 0	D37 0	D2 0	D1 0	D0 0
0009h	**PADDR** Reset Value	DD7 0	DD6 0	DD5 0	DD4 0	DD3 0	DD2 0	DD1 0	DD0 0
000Ah	**PAOR** Reset Value	O7 0	O6 0	O5 0	O4 0	O3 0	O2 0	O1 0	O0 0

5.2 WATCHDOG TIMER (WDG)

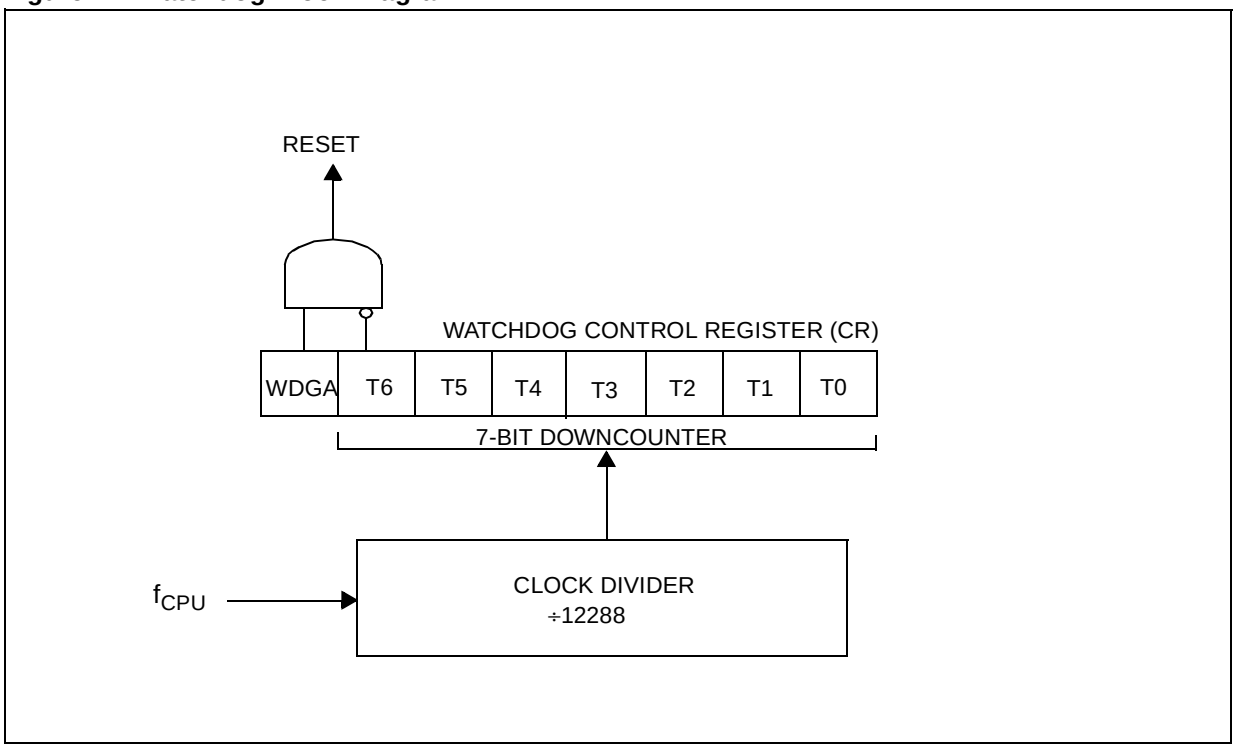
5.2.1 Introduction

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.

5.2.2 Main Features

- Programmable timer (64 increments of 12288 CPU cycles)
- Programmable reset
- Reset (if watchdog activated) after a HALT instruction or when the T6 bit reaches zero

Figure 17. Watchdog Block Diagram



WATCHDOG TIMER (Cont'd)

5.2.3 Functional Description

The counter value stored in the CR register (bits T6:T0), is decremented every 12,288 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 11):

– 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 11. Watchdog Timing (f_{CPU} = 8 MHz)

	CR Register initial value	WDG timeout period (ms)
Max	FFh	98.304
Min	C0h	1.536

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).

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

5.2.4 Low Power Modes

Mode	Description
WAIT	No effect on Watchdog.
HALT	Immediate reset generation as soon as the HALT instruction is executed if the Watchdog is activated (WDGA bit is set).

5.2.5 Interrupts

None.

5.2.6 Register Description

CONTROL REGISTER (CR)

Read/Write

Reset Value: 0111 1111 (7Fh)

7							0
WDGA	T6	T5	T4	T3	T2	T1	T0

Bit 7 = **WDGA** *Activation bit.*
This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.
0: Watchdog disabled
1: Watchdog enabled

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).

Table 12. Watchdog Timer Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0024h	**WDGCR**	WDGA	T6	T5	T4	T3	T2	T1	T0
	Reset Value	0	1	1	1	1	1	1	1

5.3 16-BIT TIMER

5.3.1 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).

5.3.2 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 18.

***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'.

5.3.3 Functional Description

5.3.3.1 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).

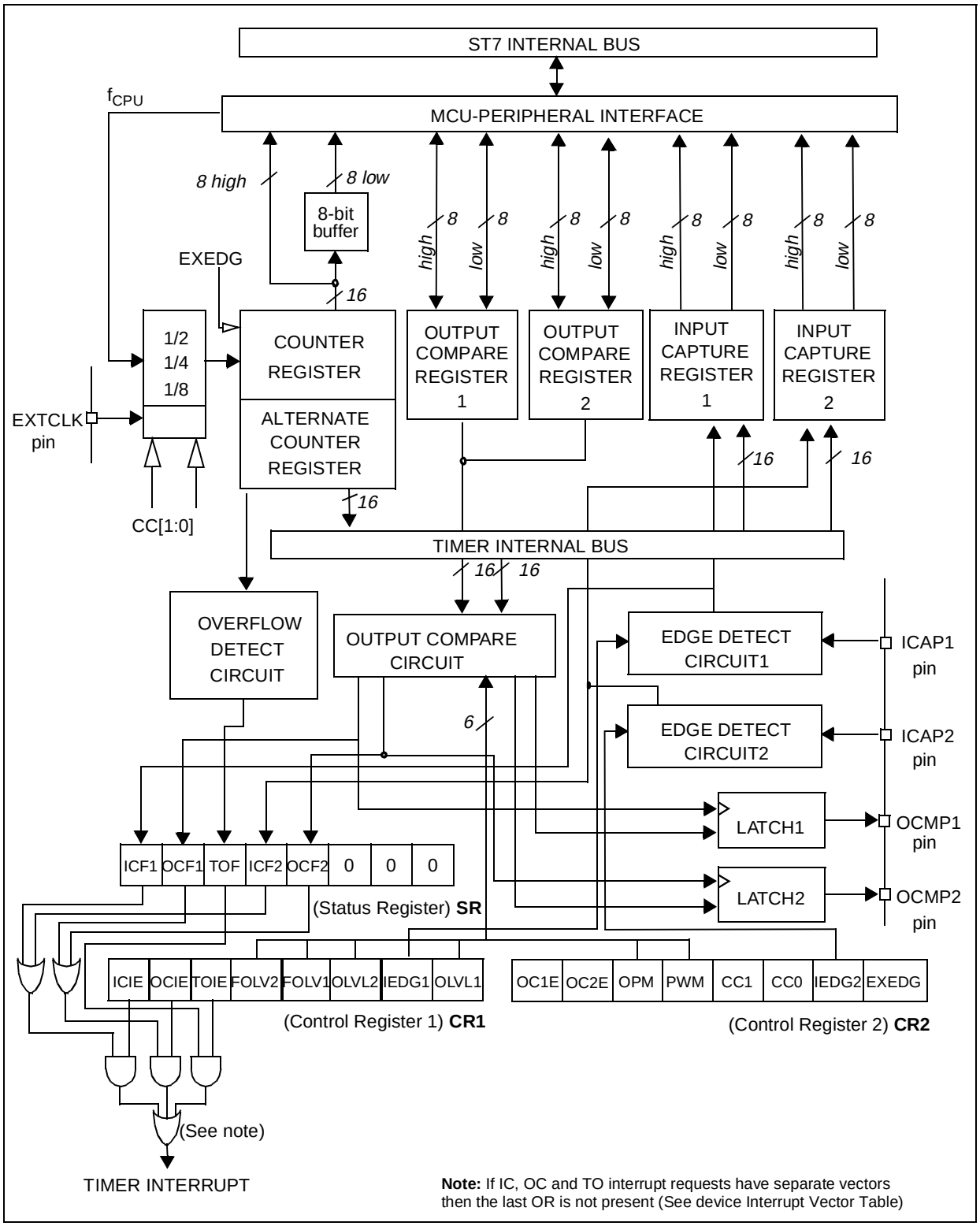
These two read-only 16-bit registers contain the same value but with the difference that reading the ACLR register does not clear the TOF bit (Timer overflow flag), located in the Status register (SR). (See note at the end of paragraph titled 16-bit read sequence).

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 13. The value in the counter register repeats every 131072, 262144 or 524288 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.

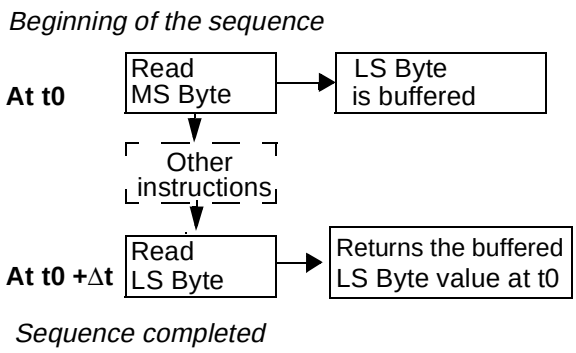
16-BIT TIMER (Cont'd)

Figure 18. Timer Block Diagram



16-BIT TIMER (Cont'd)

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



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).

5.3.3.2 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 19. Counter Timing Diagram, internal clock divided by 2

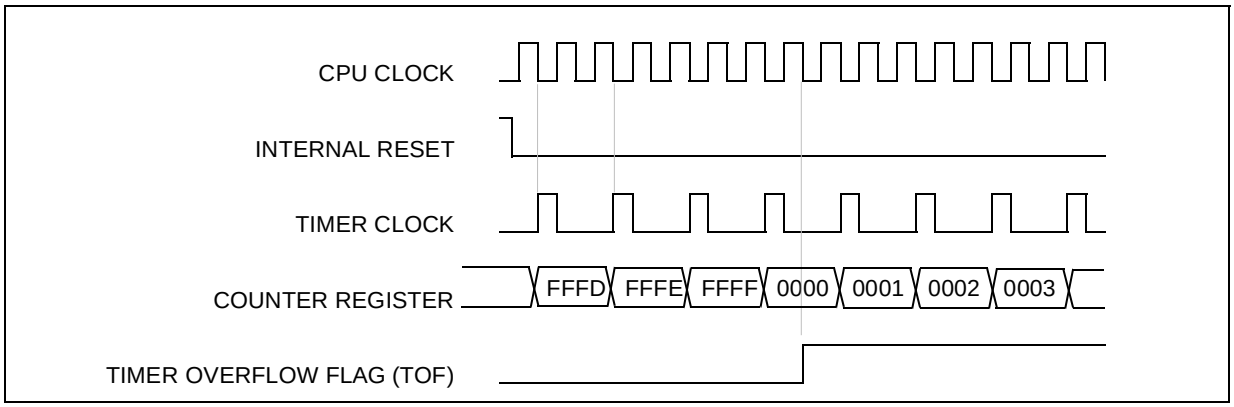


Figure 20. Counter Timing Diagram, internal clock divided by 4

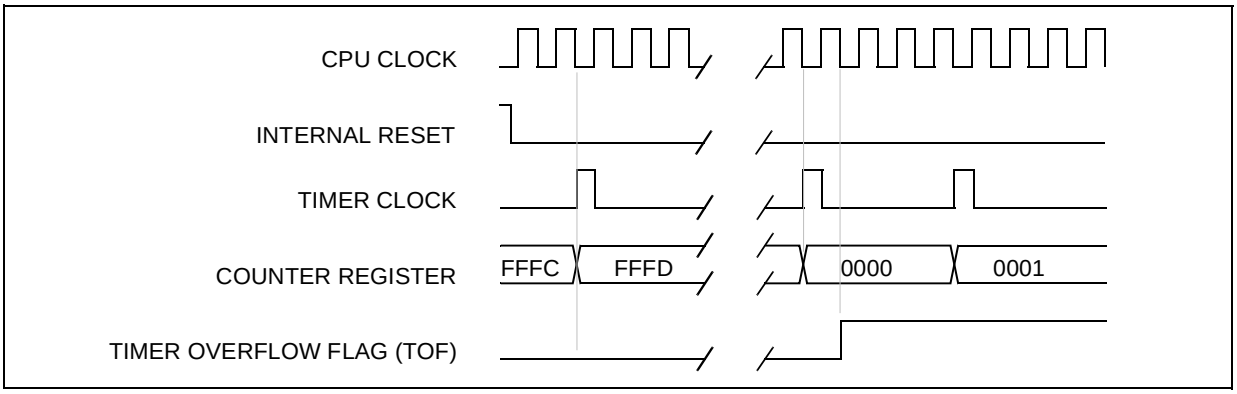
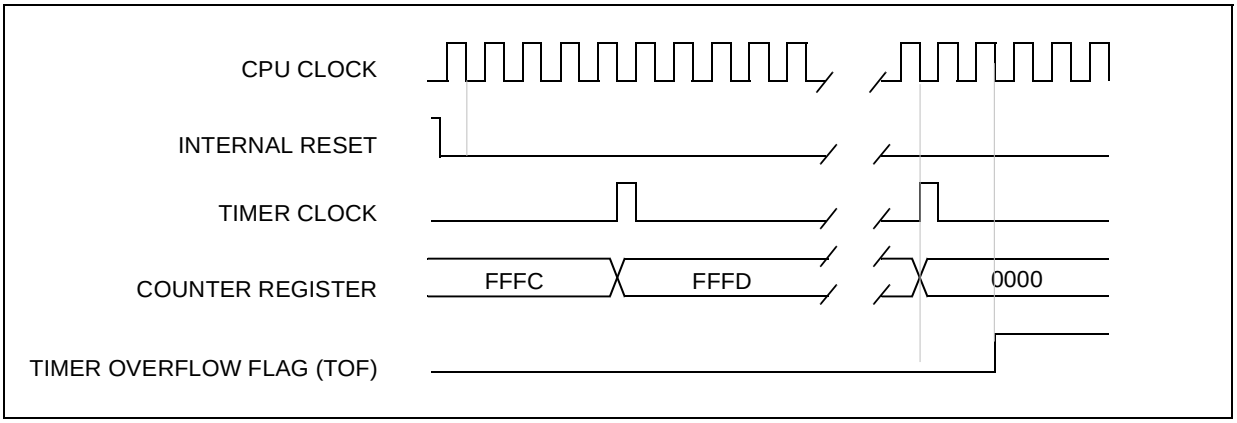


Figure 21. Counter Timing Diagram, internal clock divided by 8



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)

5.3.3.3 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).

	MS Byte	LS Byte
ICiR	IC*i*HR	IC*i*LR

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

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

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

Procedure:

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

– Select the timer clock (CC[1:0]) (see Table 13).

– 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 ICAP1pin must be configured as a floating input).

When an input capture occurs:

– The ICF*i* bit is set.

– The IC*i*R register contains the value of the free running counter on the active transition on the ICAP*i* pin (see Figure 23).

– 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 IC*i*LR register.

Notes:

1. After reading the IC*i*HR register, the transfer of input capture data is inhibited and ICF*i* will never be set until the IC*i*LR register is also read.
2. The IC*i*R 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 IC*i*HR (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 22. Input Capture Block Diagram

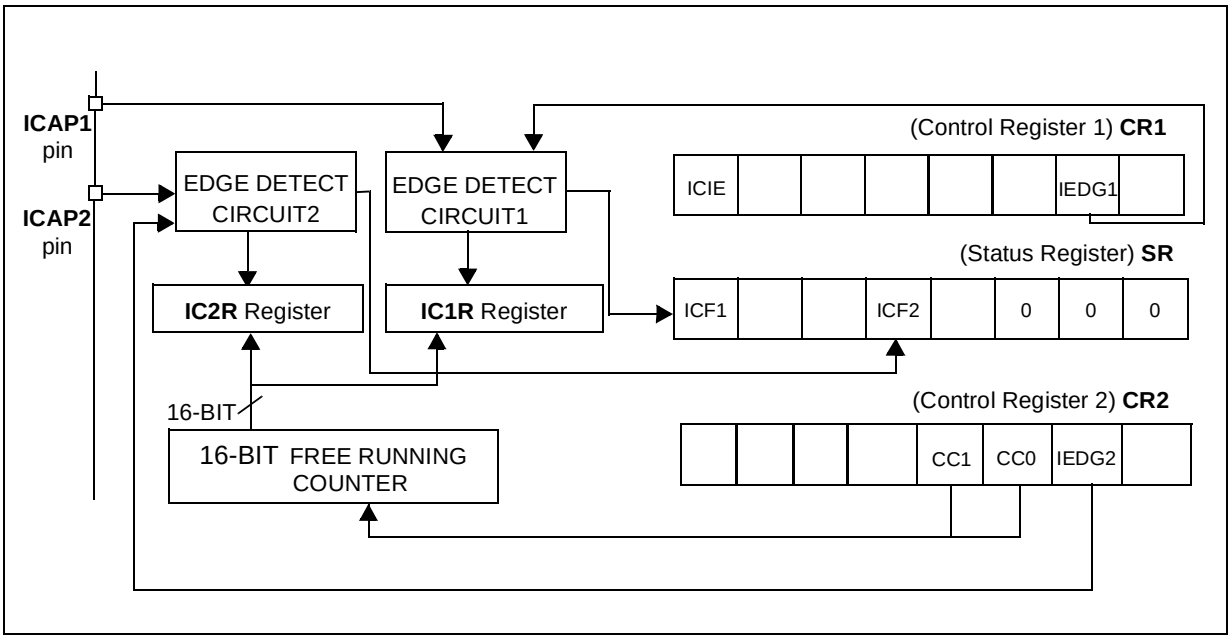
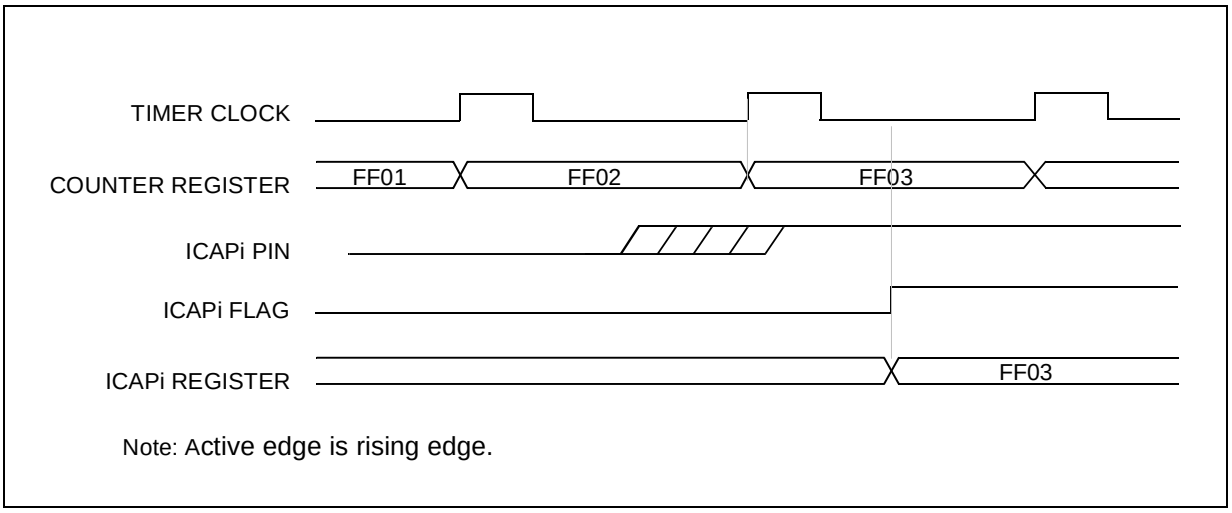


Figure 23. Input Capture Timing Diagram



Note: Active edge is rising edge.

16-BIT TIMER (Cont'd)

5.3.3.4 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	OC*i*HR	OC*i*LR

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

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

Procedure:

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

- Set the OC*i*E bit if an output is needed then the OCMP*i* pin is dedicated to the output compare *i* signal.
- Select the timer clock (CC[1:0]) (see Table 13).

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 OCRi 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 CR1 register and the I bit is cleared in the CC register (CC).

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

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

Where:

Δt = Output compare period (in seconds)

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

PRESC = Timer prescaler factor (2, 4 or 8 depending on CC[1:0] bits, see Table 13)

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

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

Where:

Δt = Output compare period (in seconds)

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

Clearing the output compare interrupt request (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*i*LR register.

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

- Write to the OC*i*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*i*LR register (enables the output compare function and clears the OCF*i* bit).

16-BIT TIMER (Cont'd)

Notes:

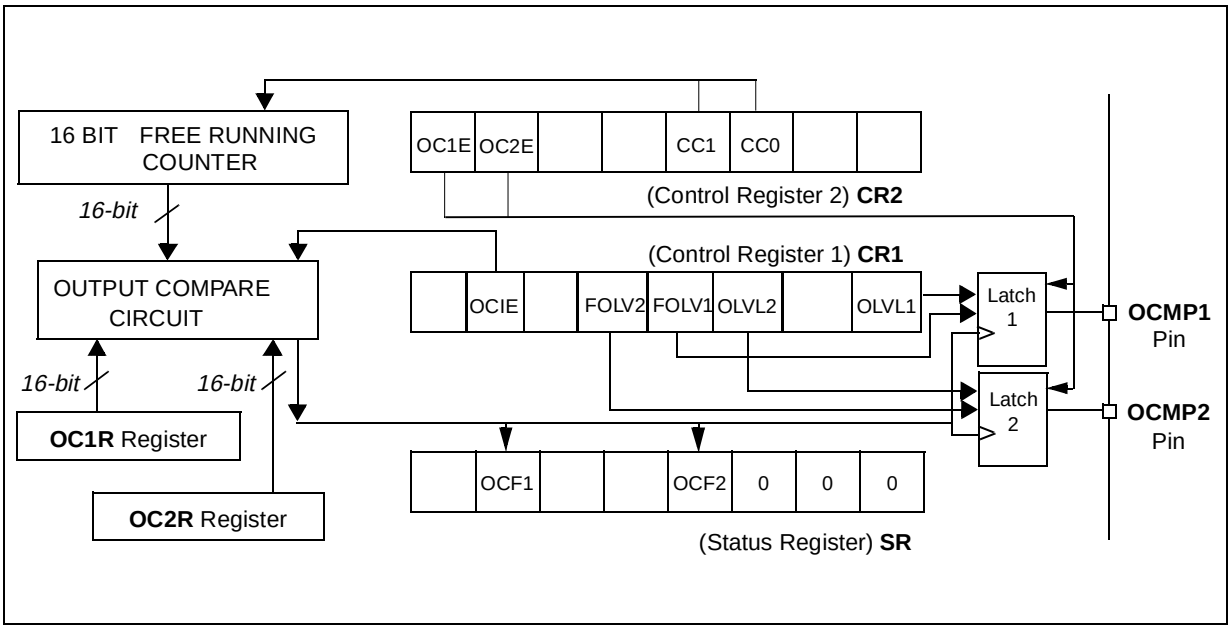
1. After a processor write cycle to the OC*i*HR register, the output compare function is inhibited until the OC*i*LR register is also written.
2. If the OC*i*E bit is not set, the OCMP*i* pin is a general I/O port and the OLVL*i* bit will not appear when a match is found but an interrupt could be generated if the OCIE bit is set.
3. When the timer clock is $f_{CPU}/2$, OCF*i* and OCMP*i* are set while the counter value equals the OC*i*R register value (see Figure 25, on page 39). 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, OCF*i* and OCMP*i* are set while the counter value equals the OC*i*R register value plus 1 (see Figure 26, on page 39).
4. The output compare functions can be used both for generating external events on the OCMP*i* pins even if the input capture mode is also used.
5. The value in the 16-bit OC*i*R register and the OLV*i* bit should be changed after each successful comparison in order to control an output waveform or establish a new elapsed timeout.

Forced Compare Output capability

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

FOLVL*i* bits have no effect in either One-Pulse mode or PWM mode.

Figure 24. Output Compare Block Diagram



16-BIT TIMER (Cont'd)

Figure 25. Output Compare Timing Diagram, $f_{TIMER} = f_{CPU}/2$

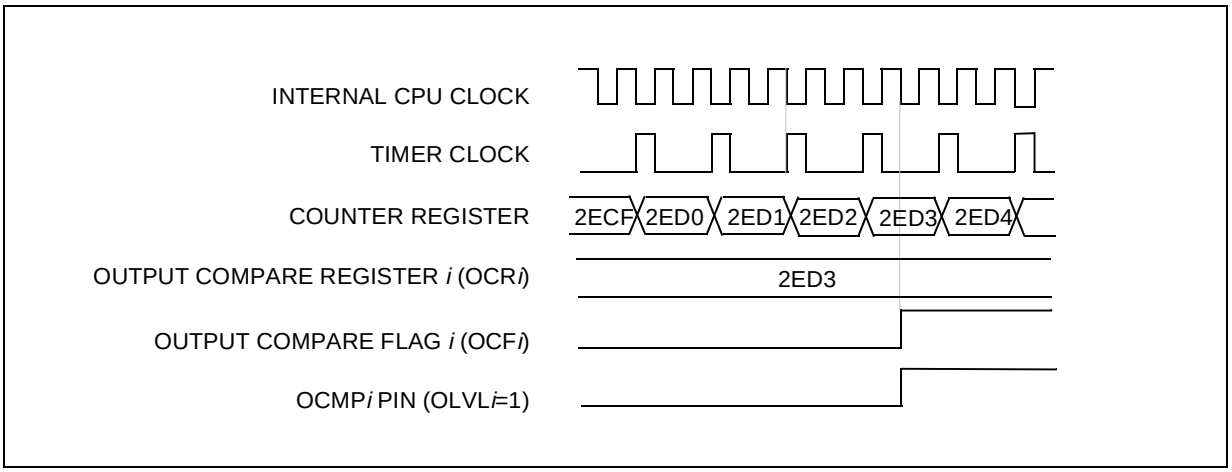
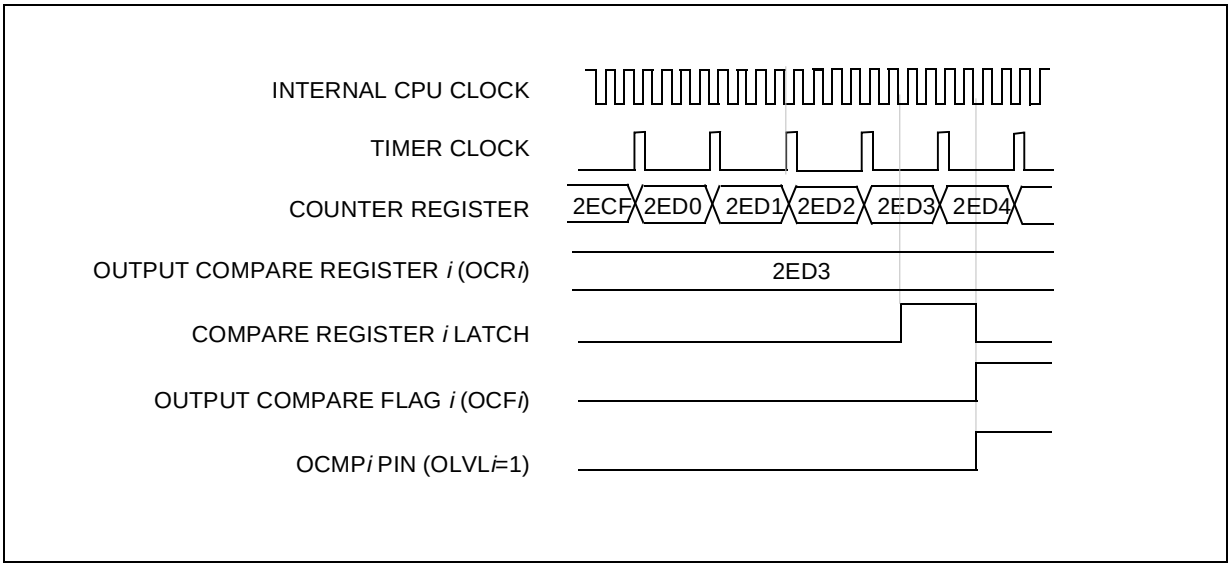


Figure 26. Output Compare Timing Diagram, $f_{TIMER} = f_{CPU}/4$



16-BIT TIMER (Cont'd)

5.3.3.5 One Pulse Mode

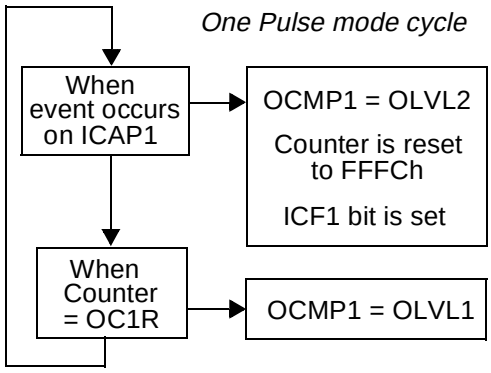
One Pulse mode enables the generation of a pulse when an external event occurs. This mode is selected via the OPM bit in the CR2 register.

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 13).



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:

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

Where:

t = Pulse period (in seconds)

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

PRESC = Timer prescaler factor (2, 4 or 8 depending on the CC[1:0] bits, see Table 13)

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

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

Where:

t = Pulse period (in seconds)

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

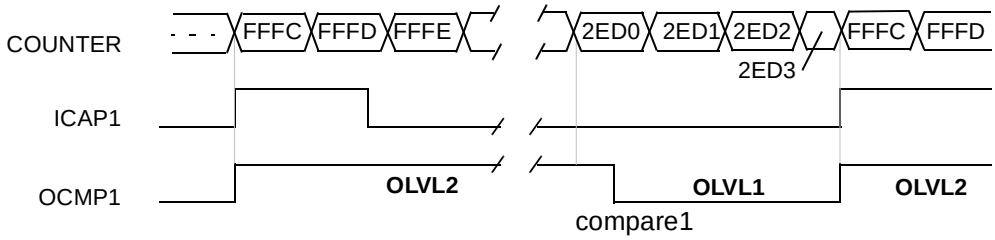
When the value of the counter is equal to the value of the contents of the OC1R register, the OLVL1 bit is output on the OCMP1 pin (see Figure 27).

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 generates interrupt if ICIE is set.
5. When One Pulse mode is used OC1R is dedicated to this mode. Nevertheless OC2R and OCF2 can be used to indicate that a period of time has elapsed but cannot generate an output waveform because the OLVL2 level is dedicated to One Pulse mode.

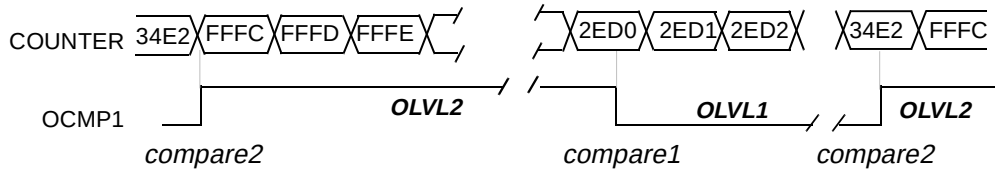
16-BIT TIMER (Cont'd)

Figure 27. One Pulse Mode Timing Example



Note: IEDG1=1, OC1R=2ED0h, OLVL1=0, OLVL2=1

Figure 28. Pulse Width Modulation Mode Timing Example



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

16-BIT TIMER (Cont'd)

5.3.3.6 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.

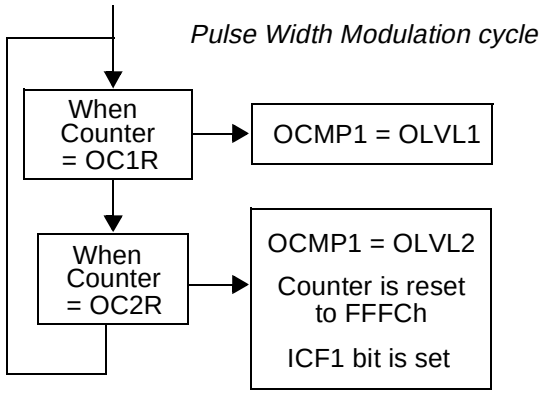
Procedure

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 13).

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

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

Where:

t = Signal or pulse period (in seconds)

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

PRESC = Timer prescaler factor (2, 4 or 8 depending on CC[1:0] bits, see Table 13)

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

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

Where:

t = Signal or pulse period (in seconds)

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

The Output Compare 2 event causes the counter to be initialized to FFFCh (See Figure 28)

Notes:

1. After a write instruction to the OC*i*HR register, the output compare function is inhibited until the OC*i*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)

5.3.4 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* 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* bit is set, and the counter value present when exiting from HALT mode is captured into the IC*i*R register.

5.3.5 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).

5.3.6 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[1)]	No	Partially [2)]
PWM Mode	No	Not Recommended[3)]	No	No

1) See note 4 in Section 5.3.3.5 One Pulse Mode

2) See note 5 in Section 5.3.3.5 One Pulse Mode

3) See note 4 in Section 5.3.3.6 Pulse Width Modulation Mode

16-BIT TIMER (Cont'd)

5.3.7 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 13. Clock Control Bits

Timer Clock	CC1	CC0
f_{CPU} / 4	0	0
f_{CPU} / 2	0	1
f_{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.

7							0
MSB							LSB

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.

7							0
MSB							LSB

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.

7							0
MSB							LSB

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.

7							0
MSB							LSB

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.

7							0
MSB							LSB

ALTERNATE COUNTER LOW REGISTER (ACLR)

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.

7							0
MSB							LSB

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).

7							0
MSB							LSB

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).

7							0
MSB							LSB

16-BIT TIMER (Cont'd)

Table 14. 16-Bit Timer Register Map and Reset Values

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
TimerA: 32	**CR1**	ICIE	OCIE	TOIE	FOLV2	FOLV1	OLVL2	IEDG1	OLVL1
TimerB: 42	Reset Value	0	0	0	0	0	0	0	0
TimerA: 31	**CR2**	OC1E	OC2E	OPM	PWM	CC1	CC0	IEDG2	EXEDG
TimerB: 41	Reset Value	0	0	0	0	0	0	0	0
TimerA: 33	**SR**	ICF1	OCF1	TOF	ICF2	OCF2	-	-	-
TimerB: 43	Reset Value	0	0	0	0	0	0	0	0
TimerA: 34	**IC1HR**	MSB	-	-	-	-	-	-	LSB
TimerB: 44	Reset Value	-							-
TimerA: 35	**IC1LR**	MSB	-	-	-	-	-	-	LSB
TimerB: 45	Reset Value	-							-
TimerA: 36	**OC1HR**	MSB	-	-	-	-	-	-	LSB
TimerB: 46	Reset Value	1	0	0	0	0	0	0	0
TimerA: 37	**OC1LR**	MSB	-	-	-	-	-	-	LSB
TimerB: 47	Reset Value	0	0	0	0	0	0	0	0
TimerA: 3E	**OC2HR**	MSB	-	-	-	-	-	-	LSB
TimerB: 4E	Reset Value	1	0	0	0	0	0	0	0
TimerA: 3F	**OC2LR**	MSB	-	-	-	-	-	-	LSB
TimerB: 4F	Reset Value	0	0	0	0	0	0	0	0
TimerA: 38	**CHR**	MSB							LSB
TimerB: 48	Reset Value	1	1	1	1	1	1	1	1
TimerA: 39	**CLR**	MSB							LSB
TimerB: 49	Reset Value	1	1	1	1	1	1	0	0
TimerA: 3A	**ACHR**	MSB							LSB
TimerB: 4A	Reset Value	1	1	1	1	1	1	1	1
TimerA: 3B	**ACLR**	MSB							LSB
TimerB: 4B	Reset Value	1	1	1	1	1	1	0	0
TimerA: 3C	**IC2HR**	MSB	-	-	-	-	-	-	LSB
TimerB: 4C	Reset Value	-							-
TimerA: 3D	**IC2LR**	MSB	-	-	-	-	-	-	LSB
TimerB: 4D	Reset Value	-							-

5.4 I²C BUS INTERFACE (I2C)

5.4.1 Introduction

The I²C Bus Interface serves as an interface between the microcontroller and the serial I²C bus. It provides both multimaster and slave functions, and controls all I²C bus-specific sequencing, protocol, arbitration and timing. It supports fast I²C mode (400kHz).

5.4.2 Main Features

- Parallel-bus/I²C protocol converter
- Multi-master capability
- 7-bit/10-bit Addressing
- Transmitter/Receiver flag
- End-of-byte transmission flag
- Transfer problem detection

I²C Master Features:

- Clock generation
- I²C bus busy flag
- Arbitration Lost Flag
- End of byte transmission flag
- Transmitter/Receiver Flag
- Start bit detection flag
- Start and Stop generation

I²C Slave Features:

- Stop bit detection
- I²C bus busy flag
- Detection of misplaced start or stop condition
- Programmable I²C Address detection
- Transfer problem detection
- End-of-byte transmission flag
- Transmitter/Receiver flag

5.4.3 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 I²C bus by a data pin (SDAI) and by a clock pin (SCLI). It can be connected both with a standard I²C bus and a Fast I²C bus. This selection is made by software.

Mode Selection

The interface can operate in the four following modes:

– Slave transmitter/receiver

– Master transmitter/receiver

By default, it operates in slave mode.

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

Communication Flow

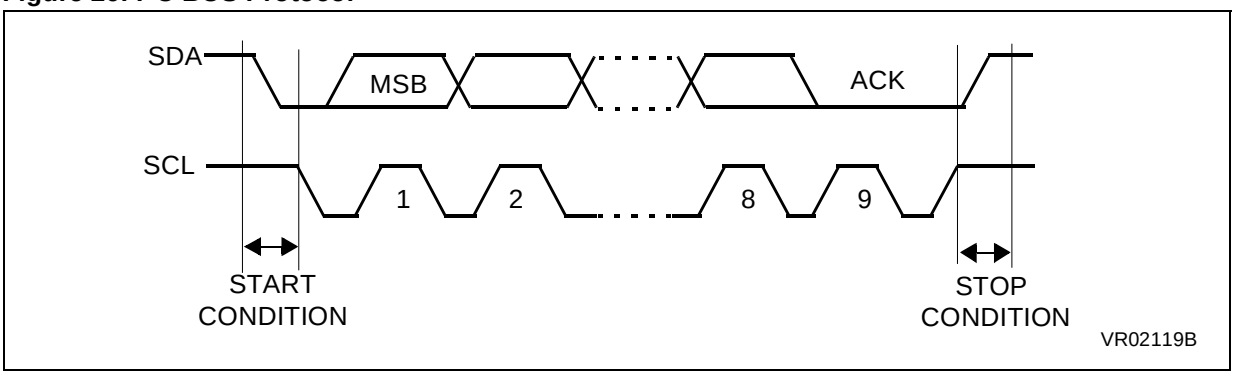
In Master mode, it initiates a data transfer and generates the clock signal. A serial data transfer always begins with a start condition and ends with a stop condition. Both start and stop conditions are generated in master mode by software.

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

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

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to Figure 29.

Figure 29. I²C BUS Protocol



I²C BUS INTERFACE (Cont'd)

Acknowledge may be enabled and disabled by software.

The I²C interface address and/or general call address can be selected by software.

The speed of the I²C interface may be selected between Standard (0-100KHz) and Fast I²C (100-400KHz).

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.

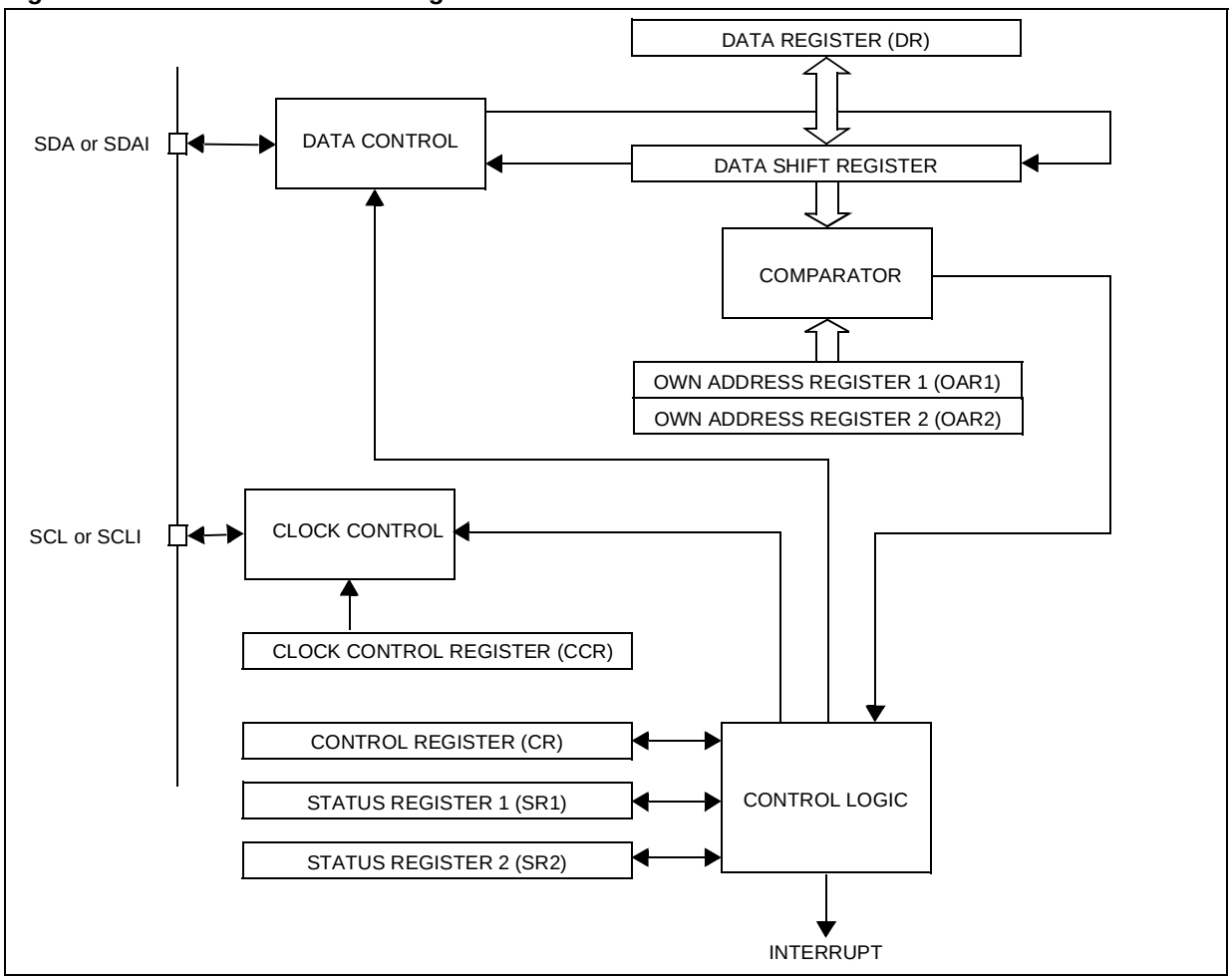
Receiver mode: the interface holds the clock line low after reception to wait for the microcontroller to read the byte in the Data Register.

The SCL frequency (F_{scl}) is controlled by a programmable clock divider which depends on the I²C bus mode.

When the I²C cell is enabled, the SDA and SCL ports must be configured as floating inputs. In this case, the value of the external pull-up resistor used depends on the application.

When the I²C cell is disabled, the SDA and SCL ports revert to being standard I/O port pins.

Figure 30. I²C Interface Block Diagram



I²C BUS INTERFACE (Cont'd)

5.4.4 Functional Description

Refer to the CR, SR1 and SR2 registers in Section 5.4.7. for the bit definitions.

By default the I²C interface operates in Slave mode (M/SL bit is cleared) except when it initiates a transmit or receive sequence.

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

5.4.4.1 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).

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

Header matched (10-bit mode only): the interface generates an acknowledge pulse if the ACK bit is set.

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

Address matched: the interface generates in sequence:

– Acknowledge pulse if the ACK bit is set.

– EVF and ADSL bits are set with an interrupt if the ITE bit is set.

Then the interface waits for a read of the SR1 register, **holding the SCL line low** (see Figure 31 Transfer sequencing EV1).
Next, in 7-bit mode read the DR register to determine from the least significant bit (Data Direction Bit) if the slave must enter Receiver or Transmitter mode.

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

Slave Receiver

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

– Acknowledge pulse if the ACK bit is set

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

Then the interface waits for a read of the SR1 register followed by a read of the DR register, **holding the SCL line low** (see Figure 31 Transfer sequencing EV2).

Slave Transmitter

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

The slave waits for a read of the SR1 register followed by a write in the DR register, **holding the SCL line low** (see Figure 31 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 31 Transfer sequencing EV4).

Error Cases

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

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

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

I²C BUS INTERFACE (Cont'd)

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.

5.4.4.2 Master Mode

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

Start condition

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

Once the Start condition is sent:

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

Then the master waits for a read of the SR1 register followed by a write in the DR register with the Slave address, **holding the SCL line low** (see Figure 31 Transfer sequencing EV5).

Slave address transmission

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

In 7-bit addressing mode, one address byte is sent.

In 10-bit addressing mode, sending the first byte including the header sequence causes the following event:

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

Then 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 31 Transfer sequencing EV9).

Then the second address byte is sent by the interface.

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 31 Transfer sequencing EV6).

Next the master must enter Receiver or Transmitter mode.

Note: In 10-bit addressing mode, to switch the master to Receiver mode, software must generate a repeated Start condition and resend the header sequence with the least significant bit set (11110xx1).

Master Receiver

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

– Acknowledge pulse if 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 31 Transfer sequencing EV7).

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

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

I²C BUS INTERFACE (Cont'd)

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 31 Transfer sequencing EV8).

When the acknowledge bit is received, the interface sets:

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

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

Error Cases

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

– **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.

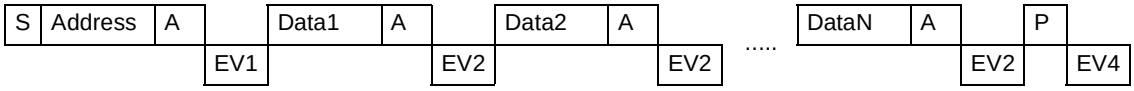
– **ARLO:** Detection of an arbitration lost condition. In this case the ARLO bit is set by hardware (with an interrupt if the ITE bit is set and the interface goes automatically back to slave mode (the M/SL bit is cleared).

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.

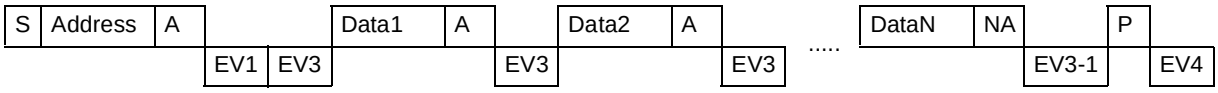
I²C BUS INTERFACE (Cont'd)

Figure 31. Transfer Sequencing

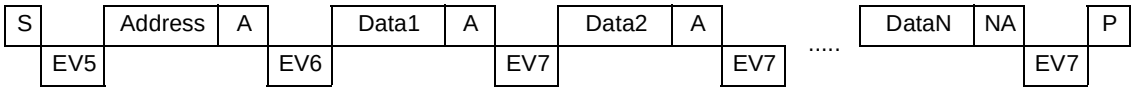
7-bit Slave receiver:



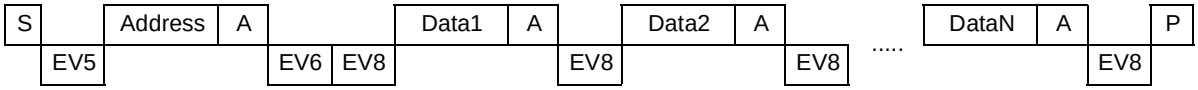
7-bit Slave transmitter:



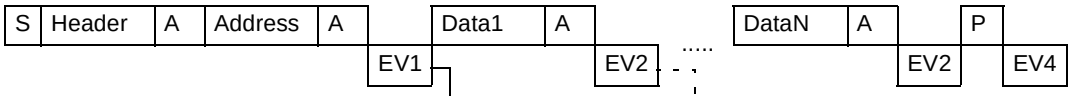
7-bit Master receiver:



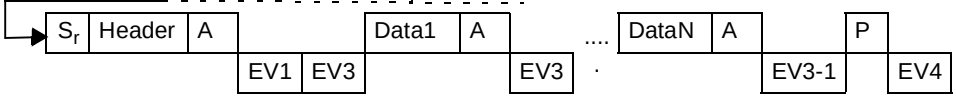
7-bit Master transmitter:



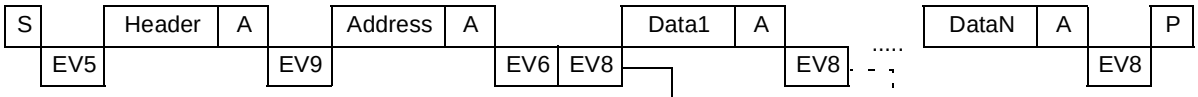
10-bit Slave receiver:



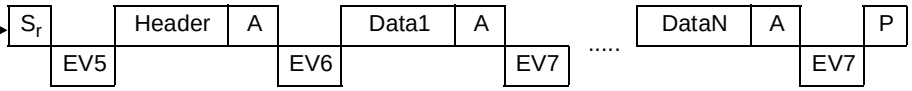
10-bit Slave transmitter:



10-bit Master transmitter



10-bit Master receiver:



Legend: S=Start, S_r = Repeated Start, P=Stop, A=Acknowledge, NA=Non-acknowledge, EVx=Event (with interrupt if ITE=1)

EV1: EVF=1, ADSL=1, cleared by reading SR1 register.

EV2: EVF=1, BTF=1, cleared by reading SR1 register followed by reading DR register.

EV3: EVF=1, BTF=1, cleared by reading SR1 register followed by writing DR register.

EV3-1: EVF=1, AF=1, BTF=1; AF is cleared by reading SR1 register. BTF is cleared by releasing the lines (STOP=1, STOP=0) or by writing DR register (DR=FFh). **Note:** If lines are released by STOP=1, STOP=0, the subsequent EV4 is not seen.

EV4: EVF=1, STOPF=1, cleared by reading SR2 register.

EV5: EVF=1, SB=1, cleared by reading SR1 register followed by writing DR register.

EV6: EVF=1, cleared by reading SR1 register followed by writing CR register (for example PE=1).

EV7: EVF=1, BTF=1, cleared by reading SR1 register followed by reading DR register.

EV8: EVF=1, BTF=1, cleared by reading SR1 register followed by writing DR register.

EV9: EVF=1, ADD10=1, cleared by reading SR1 register followed by writing DR register.

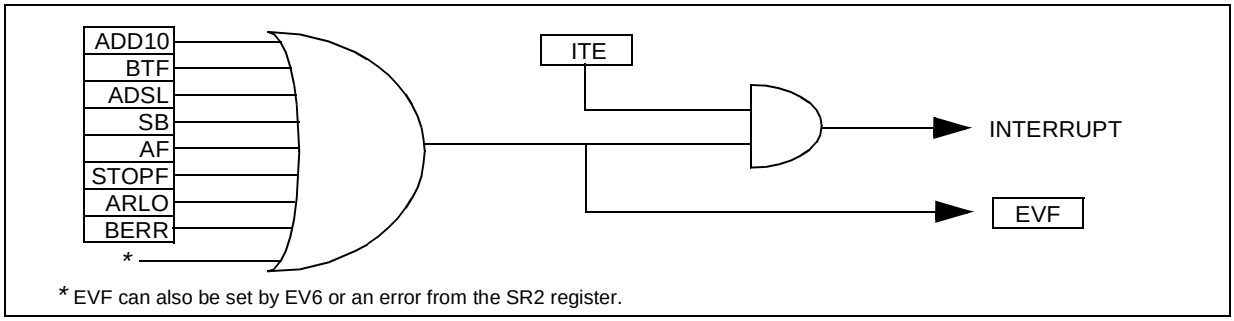
I²C BUS INTERFACE (Cont'd)

5.4.5 Low Power Modes

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

5.4.6 Interrupts

Figure 32. Event Flags and Interrupt Generation



* EVF can also be set by EV6 or an error from the SR2 register.

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
10-bit Address Sent Event (Master mode)	ADD10	ITE	Yes	No
End of Byte Transfer Event	BTF		Yes	No
Address Matched Event (Slave mode)	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

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

I²C BUS INTERFACE (Cont'd)

5.4.7 Register Description

I²C CONTROL REGISTER (CR)
Read / Write
Reset Value: 0000 0000 (00h)

7							0
0	0	PE	ENGC	START	ACK	STOP	ITE

Bit 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 I²C 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 32 for the relationship between the events and the interrupt.
SCL is held low when the ADD10, SB, BTF or ADSL flags or an EV6 event (See Figure 31) is detected.

I²C BUS INTERFACE (Cont'd)

I²C STATUS REGISTER 1 (SR1)

Read Only

Reset Value: 0000 0000 (00h)

7							0
EVF	ADD10	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 31. It is also cleared by hardware when the interface is disabled (PE=0).
0: No event
1: One of the following events has occurred:

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

Bit 6 = **ADD10** *10-bit addressing in Master mode.*
This bit is set by hardware when the master has sent the first byte in 10-bit address mode. It is cleared by software reading SR2 register followed by a write in the DR register of the second address byte. It is also cleared by hardware when the peripheral is disabled (PE=0).

0: No ADD10 event occurred.
1: Master has sent first address byte (header)

Bit 5 = **TRA** *Transmitter/Receiver.*
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 in progress on the bus. This information 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 31). 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

I²C BUS INTERFACE (Cont'd)

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

I²C STATUS REGISTER 2 (SR2)
Read Only
Reset Value: 0000 0000 (00h)

7							0
0	0	0	AF	STOPF	ARLO	BERR	GCAL

Bit 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

I²C BUS INTERFACE (Cont'd)

I²C 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 I²C mode.*
This bit is set and cleared by software. It is not cleared when the interface is disabled (PE=0).
0: Standard I²C mode
1: Fast I²C mode

Bit 6:0 = **CC6-CC0** *7-bit clock divider.*
These bits select the speed of the bus (F_{SCL}) depending on the I²C mode. They are not cleared when the interface is disabled (PE=0).

– Standard mode (FM/SM=0): F_{SCL} <= 100kHz

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

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

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

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

I²C DATA REGISTER (DR)

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

7							0
D7	D6	D5	D4	D3	D2	D1	D0

Bit 7:0 = **D7-D0** *8-bit Data Register.*
These bits contain the byte to be received or transmitted on the bus.

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

– Receiver mode: the first data byte is received automatically in the DR register using the least significant bit of the address.
Then, the following data bytes are received one by one after reading the DR register.

I²C BUS INTERFACE (Cont'd)

I²C OWN ADDRESS REGISTER (OAR1)

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

7							0
ADD7	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0

7-bit Addressing Mode

Bit 7:1 = **ADD7-ADD1** *Interface address*.
These bits define the I²C bus address of the interface. They are not cleared when the interface is disabled (PE=0).

Bit 0 = **ADD0** *Address direction bit.*
This bit is don't care, the interface acknowledges either 0 or 1. It is not cleared when the interface is disabled (PE=0).

Note: Address 01h is always ignored.

10-bit Addressing Mode

Bit 7:0 = **ADD7-ADD0** *Interface address*.
These are the least significant bits of the I²C bus address of the interface. They are not cleared when the interface is disabled (PE=0).

I²C OWN ADDRESS REGISTER (OAR2)

Read / Write
Reset Value: 0100 0000 (40h)

7							0
FR1	FR0	0	0	0	ADD9	ADD8	0

Bit 7:6 = **FR1-FR0** *Frequency bits.*

These bits are set by software only when the interface is disabled (PE=0). To configure the interface to I²C specifed delays select the value corresponding to the microcontroller frequency F_{CPU}.

F_{CPU} Range (MHz)	FR1	FR0
2.5 - 6	0	0
6 -10	0	1
10 - 14	1	0
14 - 24	1	1

Bit 5:3 = Reserved

Bit 2:1 = **ADD9-ADD8** *Interface address*.
These are the most significant bits of the I²C bus address of the interface (10-bit mode only). They are not cleared when the interface is disabled (PE=0).

Bit 0 = Reserved.

Table 15. I²C Register Map

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
28	**CR**			PE	ENGC	START	ACK	STOP	ITE
29	**SR1**	EVF		TRA	BUSY	BTF	ADSL	M/SL	SB
2A	**SR2**				AF	STOPF	ARLO	BERR	GCAL
2B	**CCR**	FM/SM	CC6 .. CC0						
2C	**OAR1**	ADD7 .. ADD0							
2D	**OAR2**	FR1	FR0				ADD9	ADD8	
2E	**DR**	DR7 .. DR0							

I²C INTERFACE (Cont'd)

5.4.8 Application Considerations

5.4.8.1 Programming Considerations

The interface can be used in two modes:

– Interrupt

– Polling

Caution: Care should be taken when polling error events as the asynchronous setting of error bits can result in error events being missed.

5.4.8.2 Application Example

The software routines given below describe a possible software driver to control the I²C interface connected to an external EEPROM without communication error management.

The interface configuration is Master mode.

This polling software does not use the EVF flag and can be easily adapted to manage interrupts keeping the same strategy.

```
/*************** (c) 1997  SGS-Thomson Microelectronics ******************/
/*                                                                       */
/* THE SOFTWARE INCLUDED IN THIS DOCUMENT IS FOR GUIDANCE ONLY. SGS-THOMSON */
/*    SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL    */
/* DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM USE OF THIS SOFTWARE.   */
/*                                                                       */
/*************************************************************************/


/*---------------------------- Basic Routines ----------------------------*/

void I2Cm_Start (void)               /* Generates I2C-Bus Start Condition.*/
{
  SetBit(I2C_CR,START);                        /* Generate start condition.*/
  while (!ValBit(I2C_SR1,SB)); /* Wait for the Start bit generation (“EV5”).*/
}

void I2Cm_Stop (void)                 /* Generates I2C-Bus Stop Condition.*/
{
  SetBit(I2C_CR,STOP);                         /* Generate stop condition.*/
}

void I2Cm_Ack (void)                   /* Activate acknowledge generation.*/
{
 SetBit(I2C_CR,ACK);      /* Acknowledge after the next received data bytes.*/
}

void I2Cm_nAck (void)               /* Disactivate acknowledge generation.*/
{
  ClrBit(I2C_CR,ACK); /* Non acknowledge after the next received data bytes.*/
}
```

I²C INTERFACE (Cont'd)

```
/*------------------------ Initialization Routine ------------------------*/

void I2Cm_Init (void)
{
  I2C_CR = 0x00;             /* Force reset status of the control register.*/
  I2C_CCR = 0X12;                    /* Set the I2C-bus speed to 0-100KHz.*/
  asm TNZ I2C_DR;            /* Touch registers to remove pending interrupt.*/
  asm TNZ I2C_SR1;
  asm TNZ I2C_SR2;
  I2C_CR = 0x24;         /* PE=1, ACK=1: switch on the peripheral interface.*/
  I2C_CR = 0x24;          /* Write twice: switch on periph then set config.*/
  I2Cm_Start();                              /* Start condition generation.*/
  I2Cm_Stop();                               /* Stop  condition generation.*/
}


/*------------------------ Communication Routines ------------------------*/

void I2Cm_SetAddr (char i2c_addr)
{
  I2Cm_Start();                             /* Generates a start condition.*/
  I2C_DR = i2c_addr;                    /* Write address to be transmitted.*/
}

void I2Cm_TxData (char i2c_data)                  /* Transmits a data byte.*/
{
  do {
    if (I2C_SR2)  while(1);   /* Communication error detected: infinite loop.*/
     SetBit(I2C_CR,PE);         /* Touch the control register to pass "EV6".*/
  } while (!ValBit(I2C_SR1,BTF));                    /* Wait for BTF ("EV8").*/
  I2C_DR = i2c_data;                   /* Write data byte to be transmitted.*/
}
char I2Cm_RxData (char last) /* Return received data byte (last one flagged).*/
{
  do {
    if (I2C_SR2)  while(1);   /* Communication error detected: infinite loop.*/
    SetBit(I2C_CR,PE);          /* Touch the control register to pass "EV6".*/
  } while (!ValBit(I2C_SR1,BTF));                    /* Wait for BTF ("EV7").*/
  if (last) I2Cm_Stop();  /* End of communication: stop condition generation.*/
  return(I2C_DR);                        /* Read/return data byte received.*/
}
```

I²C INTERFACE (Cont'd)

```
/*---------------------- EEPROM Communication Routines -------------------*/

void I2Cm_Tx (char *buff_add, char sub_add, char nb, char dest_add)
{     /* Transmit data buffer to EEPROM with least significant bytes first.*/
  I2Cm_SetAddr(dest_add);             /* Slave address selection on I2C bus.*/
  I2Cm_TxData(sub_add);                              /* Sub address selection.*/
  for (;nb > 0; nb--)   /* Loop to send all selected data from output buffer.*/
    I2Cm_TxData(*(buff_add+nb-1));     /* Next output buffer data byte sent.*/
  I2Cm_Stop();           /* End of communication: stop condition generation.*/
}

void I2Cm_Rx (char *buff_add, char sub_add, char nb, char dest_add)
{    /* Receive data buffer from EEPROM with least significant bytes first.*/
  I2Cm_SetAddr(dest_add);             /* Slave address selection on I2C bus.*/
  I2Cm_TxData(sub_add);                              /* Sub address selection.*/
  I2Cm_SetAddr(dest_add|0x01);      /* Slave address selection in read mode.*/
  do                   /* Loop to send all selected data from output buffer.*/
  {
    nb--;
    if (nb==0)                                        /* Last byte to receive.*/
    {
      I2Cm_nAck();                  /* Non acknowledge after last reception.*/
      *(buff_add+nb) = I2Cm_RxData(1);          /* Last data byte reception.*/
    }
    else
      *(buff_add+nb) = I2Cm_RxData(0);          /* Next data byte reception.*/
  } while (nb > 0);         /* Loop to receive the selected number of bytes.*/
  I2Cm_Ack();                                 /* Acknowledge after reception.*/
}

/*** (c) 1997  SGS-Thomson Microelectronics *************** END OF EXAMPLE ***/
```

5.5 SERIAL PERIPHERAL INTERFACE (SPI)

5.5.1 Introduction

The Serial Peripheral Interface (SPI) allows full-duplex, synchronous, serial communication with external devices. An SPI system may consist of a master and one or more slaves or a system in which devices may be either masters or slaves.

The SPI is normally used for communication between the microcontroller and external peripherals or another microcontroller.

Refer to the Pin Description chapter for the device-specific pin-out.

5.5.2 Main Features

- Full duplex, three-wire synchronous transfers
- Master or slave operation
- Four master mode frequencies
- Maximum slave mode frequency = fCPU/2.
- Four programmable master bit rates
- Programmable clock polarity and phase
- End of transfer interrupt flag
- Write collision flag protection
- Master mode fault protection capability.

5.5.3 General description

The SPI is connected to external devices through 4 alternate pins:

- MISO: Master In Slave Out pin
- MOSI: Master Out Slave In pin
- SCK: Serial Clock pin
- $\overline{SS}$: Slave select pin

A basic example of interconnections between a single master and a single slave is illustrated on Figure 33.

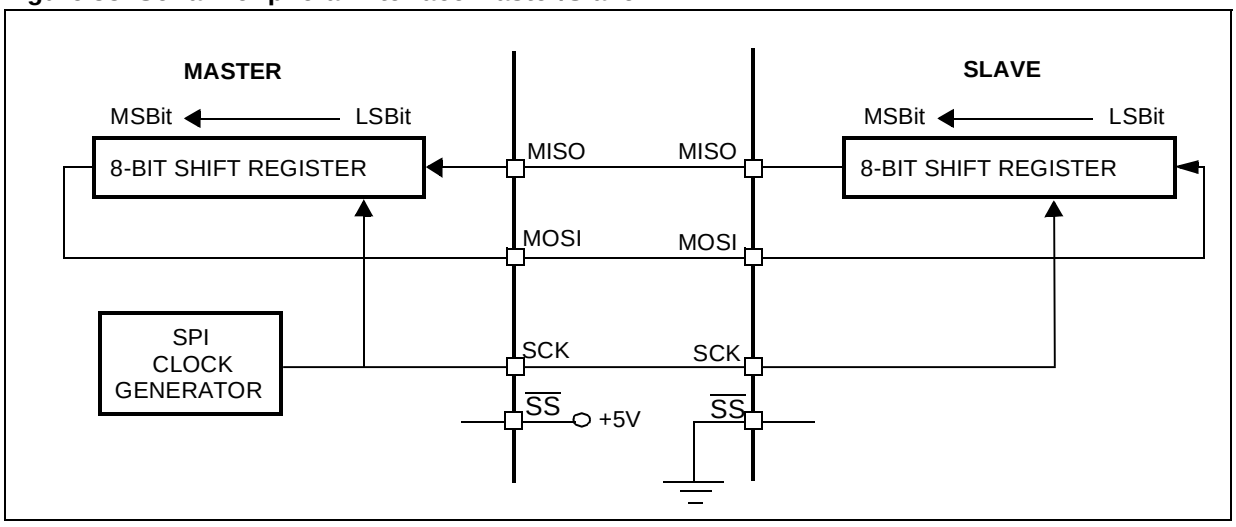
The MOSI pins are connected together as are MISO pins. In this way data is transferred serially between master and slave (most significant bit first).

When the master device transmits data to a slave device via MOSI pin, the slave device responds by sending data to the master device via the MISO pin. This implies full duplex transmission with both data out and data in synchronized with the same clock signal (which is provided by the master device via the SCK pin).

Thus, the byte transmitted is replaced by the byte received and eliminates the need for separate transmit-empty and receiver-full bits. A status flag is used to indicate that the I/O operation is complete.

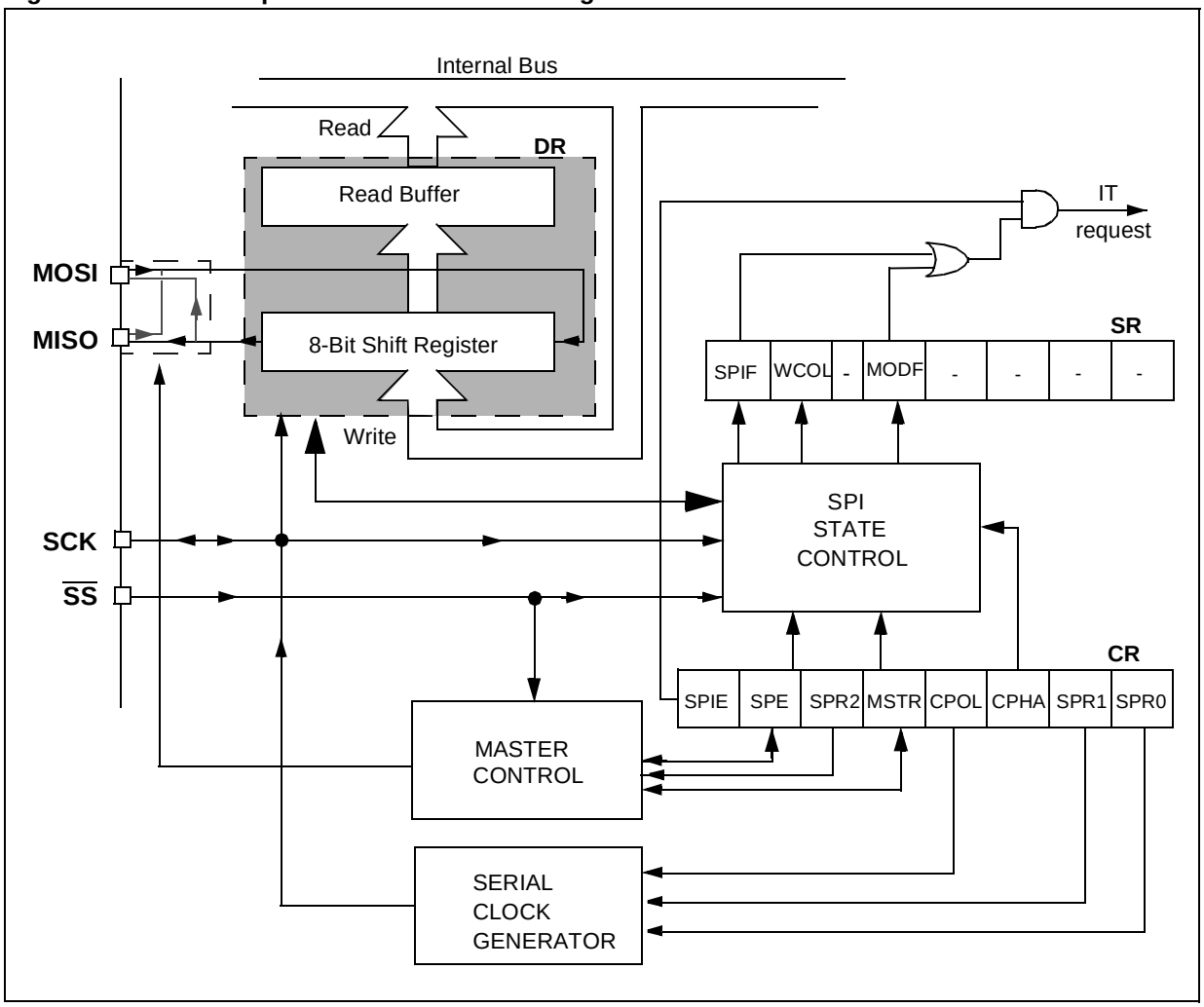
Four possible data/clock timing relationships may be chosen (see Figure 36) but master and slave must be programmed with the same timing mode.

Figure 33. Serial Peripheral Interface Master/Slave



SERIAL PERIPHERAL INTERFACE (Cont'd)

Figure 34. Serial Peripheral Interface Block Diagram



SERIAL PERIPHERAL INTERFACE (Cont'd)

5.5.4 Functional Description

Figure 33 shows the serial peripheral interface (SPI) block diagram.

This interface contains 3 dedicated registers:

- A Control Register (CR)
- A Status Register (SR)
- A Data Register (DR)

Refer to the CR, SR and DR registers in Section 5.5.7for the bit definitions.

5.5.4.1 Master Configuration

In a master configuration, the serial clock is generated on the SCK pin.

Procedure

- Select the SPR0 & SPR1 bits to define the serial clock baud rate (see CR register).
- Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock (see Figure 36).
- The $\overline{SS}$ pin must be connected to a high level signal during the complete byte transmit sequence.
- The MSTR and SPE bits must be set (they remain set only if the $\overline{SS}$ pin is connected to a high level signal).

In this configuration the MOSI pin is a data output and to the MISO pin is a data input.

Transmit sequence

The transmit sequence begins when a byte is written the DR register.

The data byte is parallel loaded into the 8-bit shift register (from the internal bus) during a write cycle and then shifted out serially to the MOSI pin most significant bit first.

When data transfer is complete:

- The SPIF bit is set by hardware
- An interrupt is generated if the SPIE bit is set and the I bit in the CCR register is cleared.

During the last clock cycle the SPIF bit is set, a copy of the data byte received in the shift register is moved to a buffer. When the DR register is read, the SPI peripheral returns this buffered value.

Clearing the SPIF bit is performed by the following software sequence:

1. An access to the SR register while the SPIF bit is set
2. A read to the DR register.

Note: While the SPIF bit is set, all writes to the DR register are inhibited until the SR register is read.

SERIAL PERIPHERAL INTERFACE (Cont'd)

5.5.4.2 Slave Configuration

In slave configuration, the serial clock is received on the SCK pin from the master device.

The value of the SPR0 & SPR1 bits is not used for the data transfer.

Procedure

- For correct data transfer, the slave device must be in the same timing mode as the master device (CPOL and CPHA bits). See Figure 36.
- The $\overline{SS}$ pin must be connected to a low level signal during the complete byte transmit sequence.
- Clear the MSTR bit and set the SPE bit to assign the pins to alternate function.

In this configuration the MOSI pin is a data input and the MISO pin is a data output.

Transmit Sequence

The data byte is parallel loaded into the 8-bit shift register (from the internal bus) during a write cycle and then shifted out serially to the MISO pin most significant bit first.

The transmit sequence begins when the slave device receives the clock signal and the most significant bit of the data on its MOSI pin.

When data transfer is complete:

- The SPIF bit is set by hardware
- An interrupt is generated if SPIE bit is set and I bit in CCR register is cleared.

During the last clock cycle the SPIF bit is set, a copy of the data byte received in the shift register is moved to a buffer. When the DR register is read, the SPI peripheral returns this buffered value.

Clearing the SPIF bit is performed by the following software sequence:

1. An access to the SR register while the SPIF bit is set.
2. A read to the DR register.

Notes: While the SPIF bit is set, all writes to the DR register are inhibited until the SR register is read.

The SPIF bit can be cleared during a second transmission; however, it must be cleared before the second SPIF bit in order to prevent an overrun condition (see Section 5.5.4.6).

Depending on the CPHA bit, the $\overline{SS}$ pin has to be set to write to the DR register between each data byte transfer to avoid a write collision (see Section 5.5.4.4).

SERIAL PERIPHERAL INTERFACE (Cont'd)

5.5.4.3 Data Transfer Format

During an SPI transfer, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). The serial clock is used to synchronize the data transfer during a sequence of eight clock pulses.

The $\overline{SS}$ pin allows individual selection of a slave device; the other slave devices that are not selected do not interfere with the SPI transfer.

Clock Phase and Clock Polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits.

The CPOL (clock polarity) bit controls the steady state value of the clock when no data is being transferred. This bit affects both master and slave modes.

The combination between the CPOL and CPHA (clock phase) bits selects the data capture clock edge.

Figure 36, shows an SPI transfer with the four combinations of the CPHA and CPOL bits. The diagram may be interpreted as a master or slave timing diagram where the SCK pin, the MISO pin, the MOSI pin are directly connected between the master and the slave device.

The $\overline{SS}$ pin is the slave device select input and can be driven by the master device.

The master device applies data to its MOSI pin-clock edge before the capture clock edge.

CPHA bit is set

The second edge on the SCK pin (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set) is the MSBit capture strobe. Data is latched on the occurrence of the second clock transition.

No write collision should occur even if the $\overline{SS}$ pin stays low during a transfer of several bytes (see Figure 35).

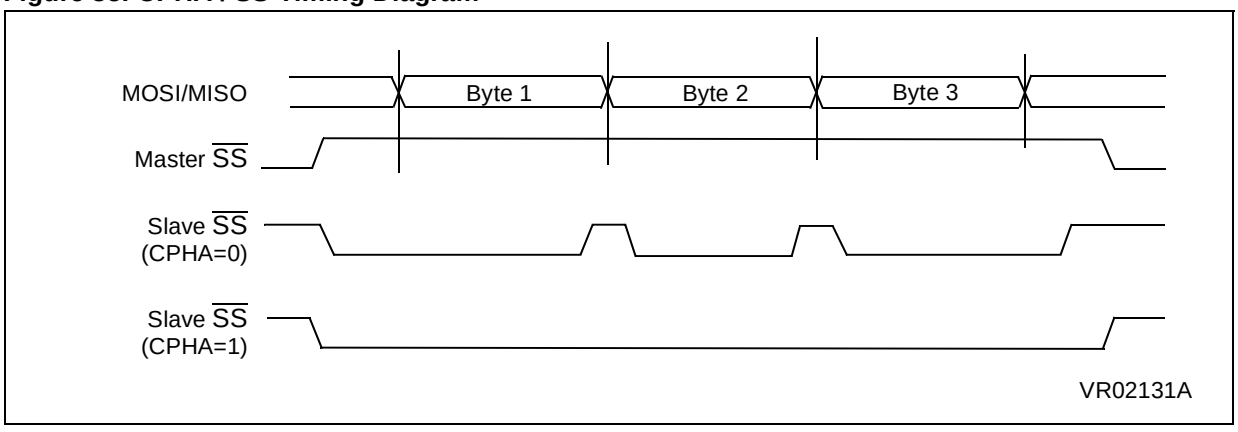
CPHA bit is reset

The first edge on the SCK pin (falling edge if CPOL bit is set, rising edge if CPOL bit is reset) is the MSBit capture strobe. Data is latched on the occurrence of the first clock transition.

The $\overline{SS}$ pin must be toggled high and low between each byte transmitted (see Figure 35).

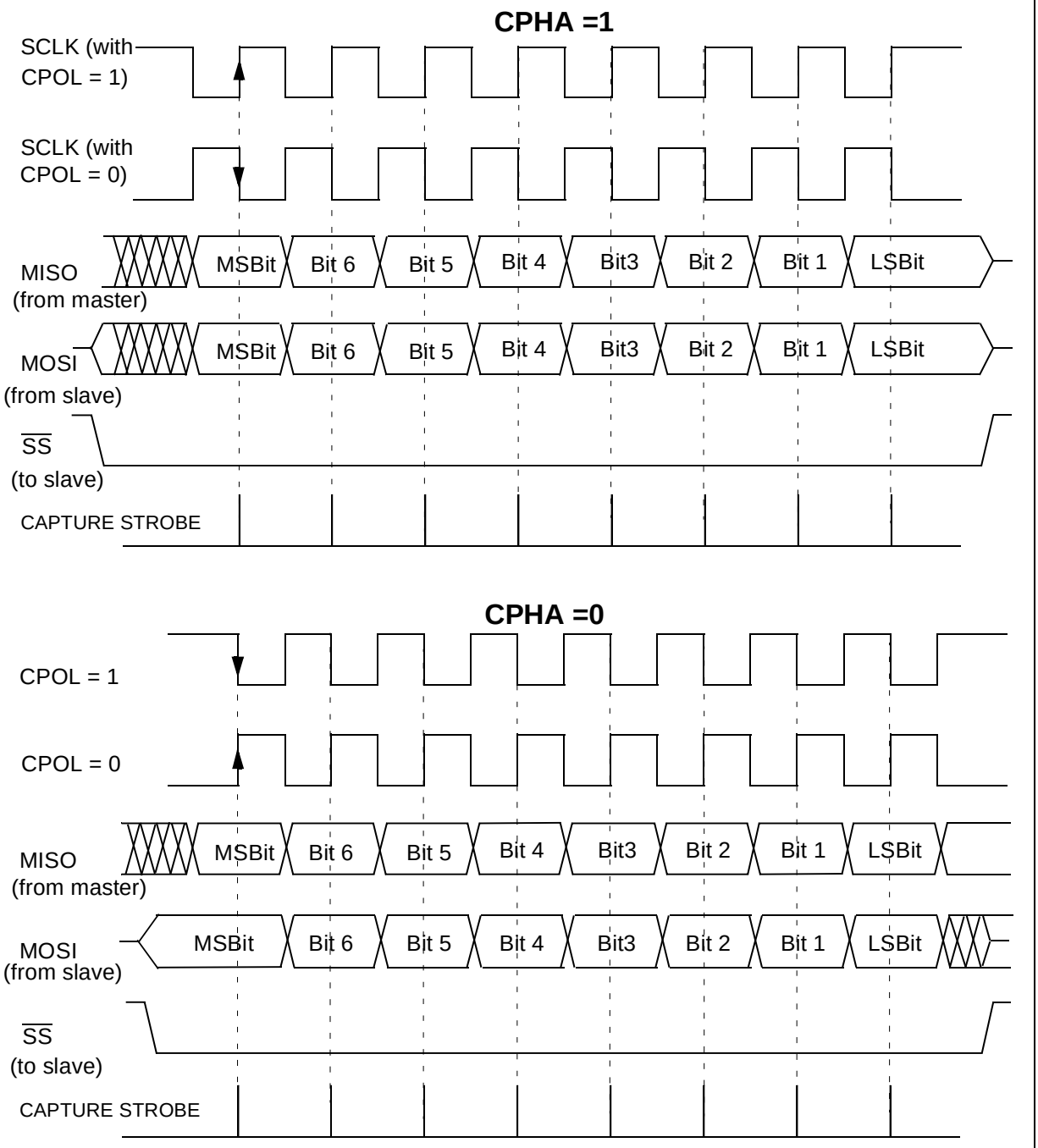
To protect the transmission from a write collision a low value on the $\overline{SS}$ pin of a slave device freezes the data in its DR register and does not allow it to be altered. Therefore the $\overline{SS}$ pin must be high to write a new data byte in the DR without producing a write collision.

Figure 35. CPHA / $\overline{SS}$ Timing Diagram



SERIAL PERIPHERAL INTERFACE (Cont'd)

Figure 36. Data Clock Timing Diagram



Note: This figure should not be used as a replacement for parametric information. Refer to the Electrical Characteristics chapter.

VR02131B

SERIAL PERIPHERAL INTERFACE (Cont'd)

5.5.4.4 Write Collision Error

A write collision occurs when the software tries to write to the DR register while a data transfer is taking place with an external device. When this happens, the transfer continues uninterrupted; and the software write will be unsuccessful.

Write collisions can occur both in master and slave mode.

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

In Slave mode

When the CPHA bit is set:

The slave device will receive a clock (SCK) edge prior to the latch of the first data transfer. This first clock edge will freeze the data in the slave device DR register and output the MSBit on to the external MISO pin of the slave device.

The $\overline{SS}$ pin low state enables the slave device but the output of the MSBit onto the MISO pin does not take place until the first data transfer clock edge.

When the CPHA bit is reset:

Data is latched on the occurrence of the first clock transition. The slave device does not have any way of knowing when that transition will occur; therefore, the slave device collision occurs when software attempts to write the DR register after its $\overline{SS}$ pin has been pulled low.

For this reason, the $\overline{SS}$ pin must be high, between each data byte transfer, to allow the CPU to write in the DR register without generating a write collision.

In Master mode

Collision in the master device is defined as a write of the DR register while the internal serial clock (SCK) is in the process of transfer.

The $\overline{SS}$ pin signal must be always high on the master device.

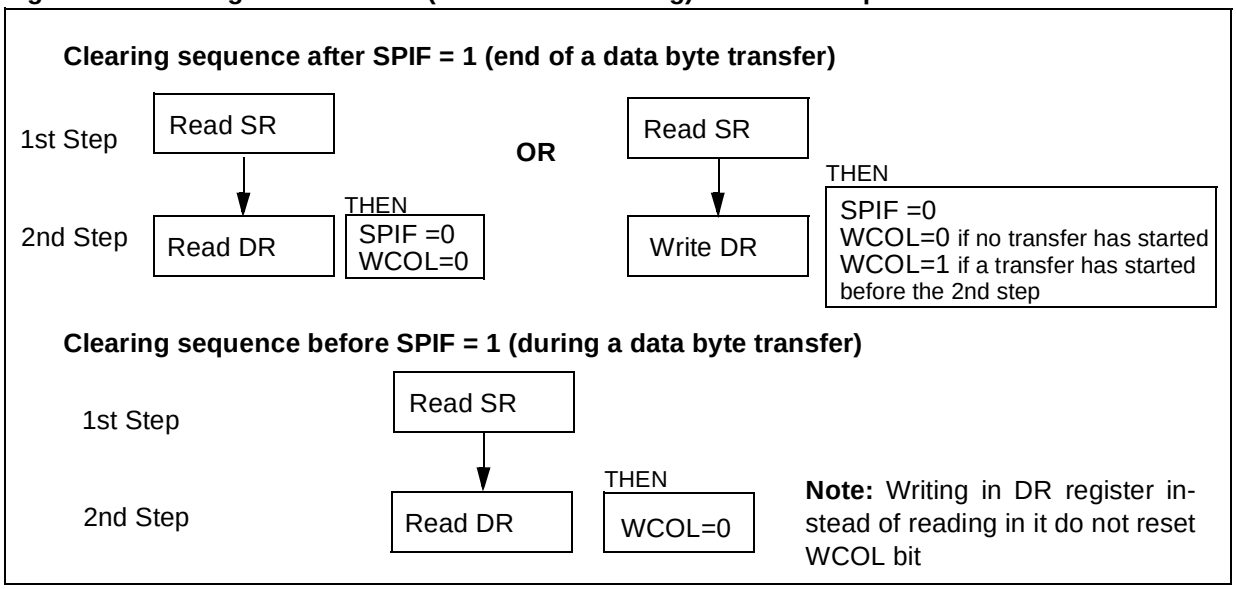
WCOL bit

The WCOL bit in the SR register is set if a write collision occurs.

No SPI interrupt is generated when the WCOL bit is set (the WCOL bit is a status flag only).

Clearing the WCOL bit is done through a software sequence (see Figure 37).

Figure 37. Clearing the WCOL bit (Write Collision Flag) Software Sequence



SERIAL PERIPHERAL INTERFACE (Cont'd)

5.5.4.5 Master Mode Fault

Master mode fault occurs when the master device has its $\overline{SS}$ pin pulled low, then the MODF bit is set.

Master mode fault affects the SPI peripheral in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the SPIE bit is set.
- The SPE bit is reset. This blocks all output from the device and disables the SPI peripheral.
- The MSTR bit is reset, thus forcing the device into slave mode.

Clearing the MODF bit is done through a software sequence:

1. A read or write access to the SR register while the MODF bit is set.
2. A write to the CR register.

Notes: To avoid any multiple slave conflicts in the case of a system comprising several MCUs, the $\overline{SS}$ pin must be pulled high during the clearing sequence of the MODF bit. The SPE and MSTR bits may be restored to their original state during or after this clearing sequence.

Hardware does not allow the user to set the SPE and MSTR bits while the MODF bit is set except in the MODF bit clearing sequence.

In a slave device the MODF bit can not be set, but in a multi master configuration the device can be in slave mode with this MODF bit set.

The MODF bit indicates that there might have been a multi-master conflict for system control and allows a proper exit from system operation to a reset or default system state using an interrupt routine.

5.5.4.6 Overrun Condition

An overrun condition occurs when the master device has sent several data bytes and the slave device has not cleared the SPIF bit issuing from the previous data byte transmitted.

In this case, the receiver buffer contains the byte sent after the SPIF bit was last cleared. A read to the DR register returns this byte. All other bytes are lost.

This condition is not detected by the SPI peripheral.

SERIAL PERIPHERAL INTERFACE (Cont'd)

5.5.4.7 Single Master and Multimaster Configurations

There are two types of SPI systems:

– Single Master System

– Multimaster System

Single Master System

A typical single master system may be configured, using an MCU as the master and four MCUs as slaves (see Figure 38).

The master device selects the individual slave devices by using four pins of a parallel port to control the four $\overline{SS}$ pins of the slave devices.

The $\overline{SS}$ pins are pulled high during reset since the master device ports will be forced to be inputs at that time, thus disabling the slave devices.

Note: To prevent a bus conflict on the MISO line the master allows only one active slave device during a transmission.

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

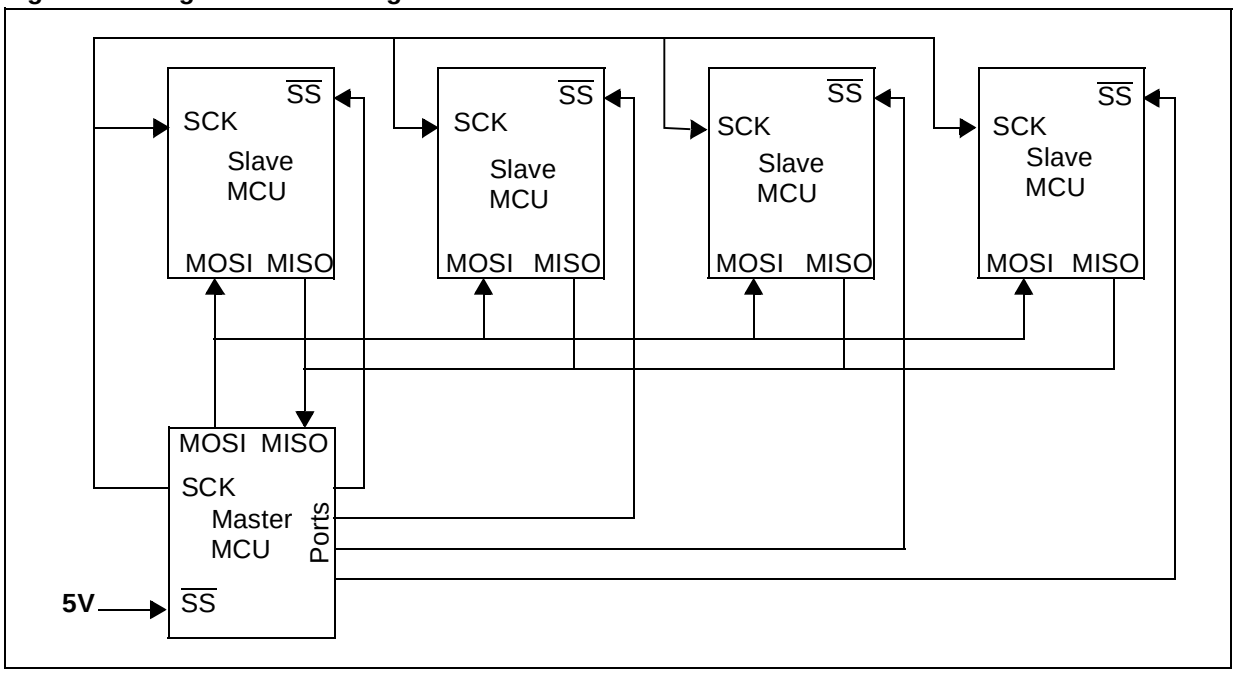
Other transmission security methods can use ports for handshake lines or data bytes with command fields.

Multi-master System

A multi-master system may also be configured by the user. Transfer of master control could be implemented using a handshake method through the I/O ports or by an exchange of code messages through the serial peripheral interface system.

The multi-master system is principally handled by the MSTR bit in the CR register and the MODF bit in the SR register.

Figure 38. Single Master Configuration



SERIAL PERIPHERAL INTERFACE (Cont'd)

5.5.5 Low Power Modes

Mode	Description
WAIT	No effect on SPI. SPI interrupt events cause the device to exit from WAIT mode.
HALT	SPI registers are frozen. In HALT mode, the SPI is inactive. SPI operation resumes when the MCU is woken up by an interrupt with "exit from HALT mode" capability.

5.5.6 Interrupts

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

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

SERIAL PERIPHERAL INTERFACE (Cont'd)

5.5.7 Register Description

CONTROL REGISTER (CR)

Read/Write

Reset Value: 0000xxxx (0xh)

7							0
SPIE	SPE	SPR2	MSTR	CPOL	CPHA	SPR1	SPR0

Bit 7 = **SPIE** *Serial peripheral interrupt enable.*
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SPI interrupt is generated whenever SPIF=1 or MODF=1 in the SR register

Bit 6 = **SPE** *Serial peripheral output enable.*
This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$=0 (see Section 5.5.4.5 Master Mode Fault).
0: I/O port connected to pins
1: SPI alternate functions connected to pins

The SPE bit is cleared by reset, so the SPI peripheral is not initially connected to the external pins.

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

this bit is set and cleared by software and it is cleared by reset. It is used with the SPR[1:0] bits to set the baud rate. Refer to Table 16.
0: Divider by 2 enabled
1: Divider by 2 disabled

Bit 4 = **MSTR** *Master.*
This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$=0 (see Section 5.5.4.5 Master Mode Fault).
0: Slave mode is selected
1: Master mode is selected, the function of the SCK pin changes from an input to an output and the functions of the MISO and MOSI pins are reversed.

Bit 3 = **CPOL** *Clock polarity.*
This bit is set and cleared by software. This bit determines the steady state of the serial Clock. The CPOL bit affects both the master and slave modes.
0: The steady state is a low value at the SCK pin.
1: The steady state is a high value at the SCK pin.

Bit 2 = **CPHA** *Clock phase.*
This bit is set and cleared by software.
0: The first clock transition is the first data capture edge.
1: The second clock transition is the first capture edge.

Bit 1:0 = **SPR[1:0]** *Serial peripheral rate.*
These bits are set and cleared by software.Used with the SPR2 bit, they select one of six baud rates to be used as the serial clock when the device is a master.

These 2 bits have no effect in slave mode.

Table 16. Serial Peripheral Baud Rate

Serial Clock	SPR2	SPR1	SPR0
$f_{CPU}/4$	1	0	0
$f_{CPU}/8$	0	0	0
$f_{CPU}/16$	0	0	1
$f_{CPU}/32$	1	1	0
$f_{CPU}/64$	0	1	0
$f_{CPU}/128$	0	1	1

SERIAL PERIPHERAL INTERFACE (Cont'd)

STATUS REGISTER (SR)

Read Only

Reset Value: 0000 0000 (00h)

7							0
SPIF	WCOL	-	MODF	-	-	-	-

Bit 7 = **SPIF** *Serial Peripheral data transfer flag.*
This bit is set by hardware when a transfer has been completed. An interrupt is generated if SPIE=1 in the CR register. It is cleared by a software sequence (an access to the SR register followed by a read or write to the DR register).
0: Data transfer is in progress or has been approved by a clearing sequence.
1: Data transfer between the device and an external device has been completed.

Note: While the SPIF bit is set, all writes to the DR register are inhibited.

Bit 6 = **WCOL** *Write Collision status.*

This bit is set by hardware when a write to the DR register is done during a transmit sequence. It is cleared by a software sequence (see Figure 37).
0: No write collision occurred
1: A write collision has been detected

Bit 5 = Unused.

Bit 4 = **MODF** *Mode Fault flag.*
This bit is set by hardware when the $\overline{SS}$ pin is pulled low in master mode (see Section 5.5.4.5 Master Mode Fault). An SPI interrupt can be generated if SPIE=1 in the CR register. This bit is cleared by a software sequence (An access to the SR register while MODF=1 followed by a write to the CR register).
0: No master mode fault detected
1: A fault in master mode has been detected

Bits 3-0 = Unused.

DATA I/O REGISTER (DR)

Read/Write

Reset Value: Undefined

7							0
D7	D6	D5	D4	D3	D2	D1	D0

The DR register is used to transmit and receive data on the serial bus. In the master device only a write to this register will initiate transmission/reception of another byte.

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

Warning:

A write to the DR register places data directly into the shift register for transmission.

A write to the the DR register returns the value located in the buffer and not the contents of the shift register (See Figure 34).

SERIAL PERIPHERAL INTERFACE (Cont'd)

Table 17. SPI Register Map and Reset Values

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
21	**DR**	D7	D6	D5	D4	D3	D2	D1	D0
	Reset Value	x	x	x	x	x	x	x	x
22	**CR**	SPIE	SPE	SPR2	MSTR	CPOL	CPHA	SPR1	SPR0
	Reset Value	0	0	0	0	x	x	x	x
23	**SR**	SPIF	WCOL	-	MODF	-	-	-	-
	Reset Value	0	0	0	0	0	0	0	0

5.6 8-BIT A/D CONVERTER (ADC)

5.6.1 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.

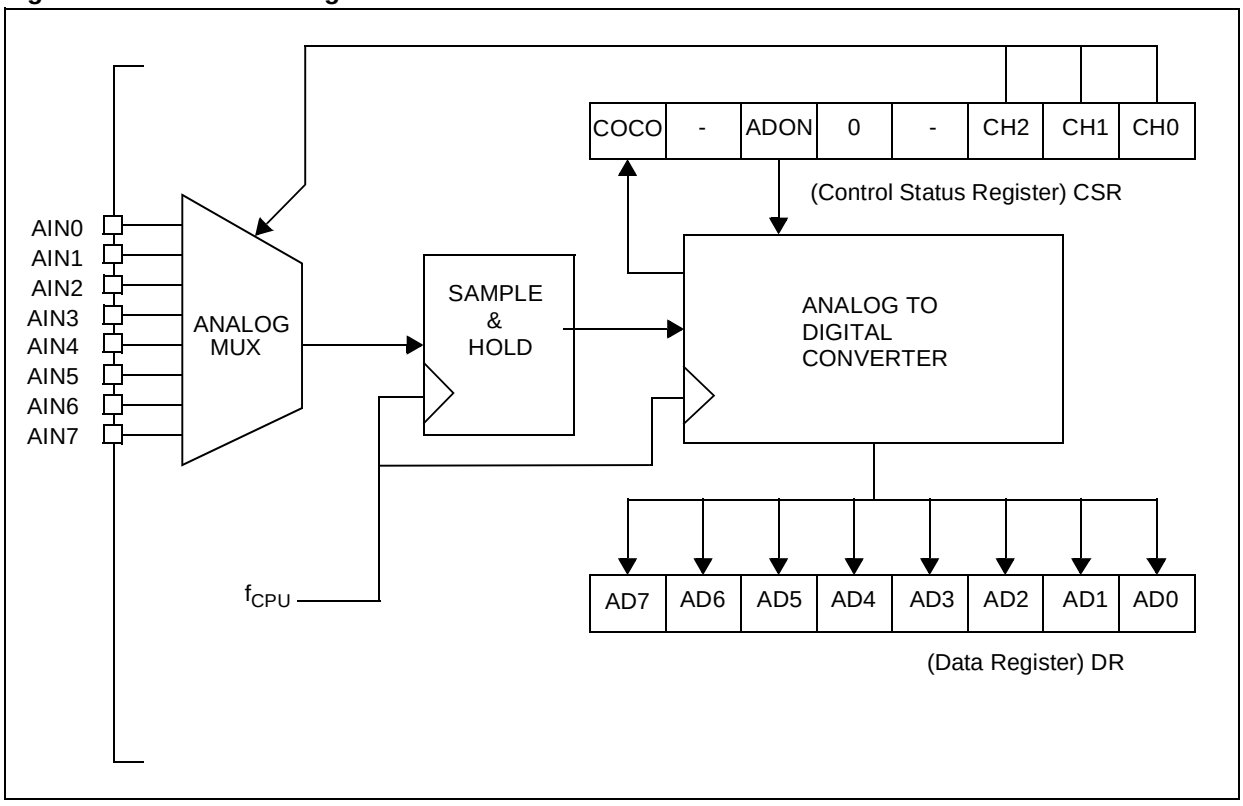
The result of the conversion is stored in a 8-bit Data Register. The A/D converter is controlled through a Control/Status Register.

5.6.2 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 39.

Figure 39. ADC Block Diagram



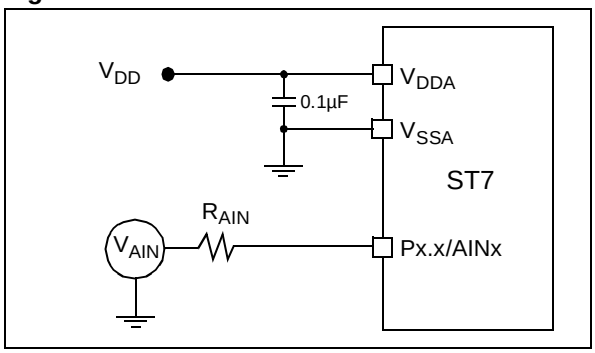
8-BIT A/D CONVERTER (ADC) (Cont'd)

5.6.3 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 40. 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 alloted 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 18.

– Set the ADON bit. Then the A/D converter is enabled after a stabilization time (typically 30 µs). 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.

5.6.4 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.

5.6.5 Interrupts

None.

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

5.6.6 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 µs 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 18. 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.

Table 19. ADC Register Map

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
70 Reset Value	**DR**	AD7 0	AD6 0	AD5 0	AD4 0	AD3 0	AD2 0	AD1 0	AD0 0
71 Reset Value	**CSR**	COCO 0	- 0	ADON 0	0 0	- 0	CH2 0	CH1 0	CH0 0

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 20. 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[1)]			+ 1
Relative	Indirect		jrne [$10]	PC-128/PC+127[1)]	00..FF	byte	+ 2
Bit	Direct		bset $10,#7	00..FF			+ 1
Bit	Indirect		bset [$10],#7	00..FF	00..FF	byte	+ 2
Bit	Direct	Relative	btjt $10,#7,skip	00..FF			+ 2
Bit	Indirect	Relative	btjt [$10],#7,skip	00..FF	00..FF	byte	+ 3

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 21. 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 submodes:

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				
Condition Code Flag modification	SIM	RIM	SCF	RCF				

Using a pre-byte

The instructions are described with one to four bytes.

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

The whole instruction becomes:

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

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

- PDY 90 Replace an X based instruction using immediate, direct, indexed, or inherent addressing mode by a Y one.
- PIX 92 Replace an instruction using direct, direct bit, or direct relative addressing mode to an instruction using the corresponding indirect addressing mode.
  It also changes an instruction using X indexed addressing mode to an instruction using indirect X indexed addressing mode.
- PIY 91 Replace an instruction using X indirect indexed addressing mode by a Y one.

INSTRUCTION GROUPS (Cont'd)

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

INSTRUCTION GROUPS (Cont'd)

Mnemo	Description	Function/Example	Dst	Src	H	I	N	Z	C
JRULE	Jump if (C + Z = 1)	Unsigned <=							
LD	Load	dst <= src	reg, M	M, reg			N	Z	
MUL	Multiply	X,A = X * A	A, X, Y	X, Y, A	0				0
NEG	Negate (2's compl)	neg $10	reg, M				N	Z	C
NOP	No Operation								
OR	OR operation	A = A + M	A	M			N	Z	
POP	Pop from the Stack	pop reg	reg	M					
		pop CC	CC	M	H	I	N	Z	C
PUSH	Push onto the Stack	push Y	M	reg, CC					
RCF	Reset carry flag	C = 0							0
RET	Subroutine Return								
RIM	Enable Interrupts	I = 0				0			
RLC	Rotate left true C	C <= Dst <= C	reg, M				N	Z	C
RRC	Rotate right true C	C => Dst => C	reg, M				N	Z	C
RSP	Reset Stack Pointer	S = Max allowed							
SBC	Subtract with Carry	A = A - M - C	A	M			N	Z	C
SCF	Set carry flag	C = 1							1
SIM	Disable Interrupts	I = 1				1			
SLA	Shift left Arithmetic	C <= Dst <= 0	reg, M				N	Z	C
SLL	Shift left Logic	C <= Dst <= 0	reg, M				N	Z	C
SRL	Shift right Logic	0 => Dst => C	reg, M				0	Z	C
SRA	Shift right Arithmetic	Dst7 => Dst => C	reg, M				N	Z	C
SUB	Subtraction	A = A - M	A	M			N	Z	C
SWAP	SWAP nibbles	Dst[7..4] <=> Dst[3..0]	reg, M				N	Z	
TNZ	Test for Neg & Zero	tnz lbl1					N	Z	
TRAP	S/W trap	S/W interrupt				1			
WFI	Wait for Interrupt					0			
XOR	Exclusive OR	A = A XOR M	A	M			N	Z	

7 ELECTRICAL CHARACTERISTICS

7.1 ABSOLUTE MAXIMUM RATINGS

This product contains devices to protect the inputs against damage due to high static voltages, however it is advisable to take normal precaution to avoid application of any voltage higher than the specified maximum rated voltages.

For proper operation it is recommended that V_I and V_O be higher than V_{SS} and lower than V_{DD}. Reliability is enhanced if unused inputs are connected to an appropriate logic voltage level (V_{DD} or V_{SS}).

Power Considerations.The average chip-junction temperature, T_J, in Celsius can be obtained from:

T_J= TA + PD x RthJA

Where:T_A = Ambient Temperature.

RthJA = Package thermal resistance (junction-to ambient).

P_D = P_{INT} + P_{PORT}.

P_{INT} = I_{DD} x V_{DD} (chip internal power).

P_{PORT} =Port power dissipation determined by the user)

Symbol	Parameter	Value	Unit
V_{DD}	Supply Voltage	-0.3 to 6.0	V
V_I	Input Voltage	V_{SS} - 0.3 to V_{DD} + 0.3	V
V_{AI}	Analog Input Voltage (A/D Converter)	V_{SS} - 0.3 to V_{DD} + 0.3	V
V_O	Output Voltage	V_{SS} - 0.3 to V_{DD} + 0.3	V
IV_{DD}	Total Current into V_{DD} (source)	80	mA
IV_{SS}	Total Current out of V_{SS} (sink)	80	mA
T_J	Junction Temperature	150	°C
T_{STG}	Storage Temperature	-60 to 150	°C

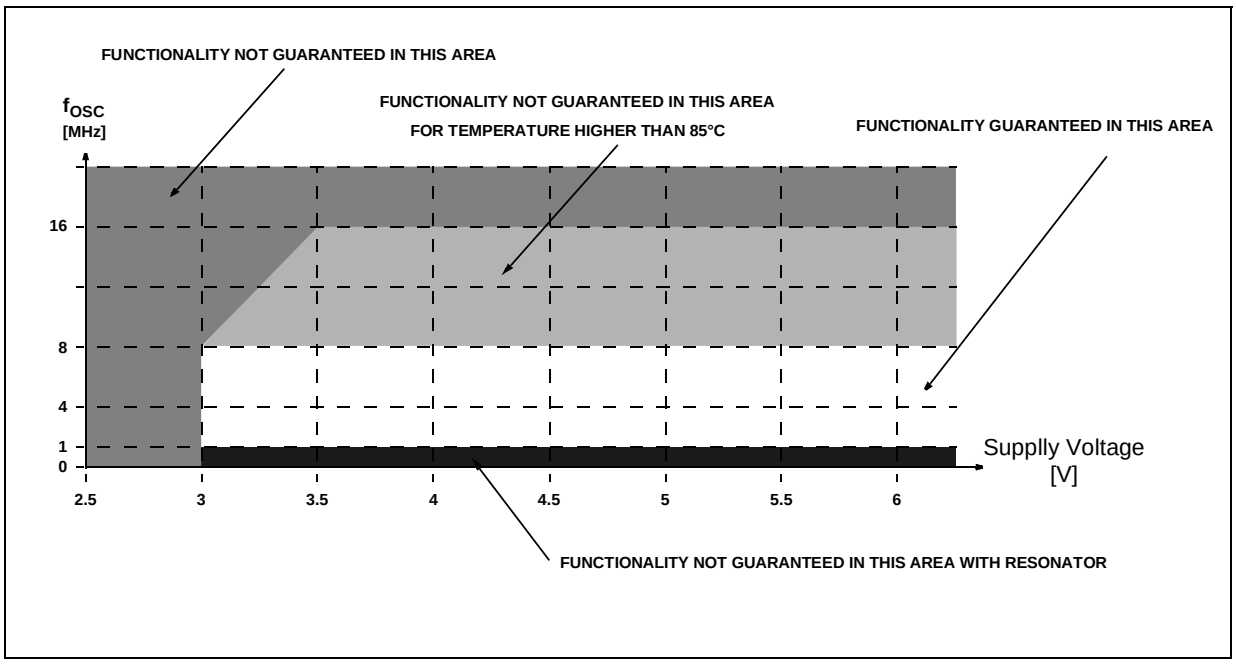
Note: Stresses above those listed as “absolute maximum ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

7.2 RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
T_A	Operating Temperature	1 Suffix Version	0		70	°C
		6 Suffix Version	-40		85	°C
		3 Suffix Version	-40		125	°C
V_{DD}	Operating Supply Voltage	f_{OSC} = 16 MHz (1 & 6 Suffix) f_{OSC} = 8 MHz	3.5 3.0		5.5 5.5	V
f_{OSC}	Oscillator Frequency	V_{DD} = 3.0V V_{DD} = 3.5V (1 & 6 Suffix)	0 [1] 0 [1]		8 16	MHz

Note 1: A/D operation and Oscillator start-up are not guaranteed below 1MHz.

Figure 41. Maximum Operating Frequency (Fmax) Versus Supply Voltage (V_{DD})



7.3 DC ELECTRICAL CHARACTERISTICS

(T_A = -40°C to +125°C and V_{DD} = 5V unless otherwise specified)

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
V_{IL}	Input Low Level Voltage All Input pins	3V < V_{DD} < 5.5V			V_{DD} x 0.3	V
V_{IH}	Input High Level Voltage All Input pins	3V < V_{DD} < 5.5V	V_{DD} x 0.7			V
V_{HYS}	Hysteresis Voltage [1] All Input pins			400		mV
V_{OL}	Low Level Output Voltage All Output pins	I_{OL} = +10µA I_{OL} = + 2mA			0.1 0.4	V
	Low Level Output Voltage High Sink I/O pins	I_{OL} = +10µA I_{OL} = +10mA I_{OL} = + 15mA I_{OL} = + 20mA, T_A < 85°C			0.1 1.5 3.0 3.0	V
V_{OH}	High Level Output Voltage All Output pins	I_{OH} = - 10µA I_{OH} = - 2mA	4.9 4.2			V
I_{IL} I_{IH}	Input Leakage Current All Input pins but RESET [4]	V_{IN} = V_{SS} (No Pull-up configured) V_{IN} = V_{DD}		0.1	1.0	µA
I_{IH}	Input Leakage Current RESET pin	V_{IN} = V_{DD}		0.1	1.0	µA
R_{PU}	I/O Weak Pull-up R_{PU}	V_{IN} < V_{IL}		100		kΩ
I_{DD}	Supply Current in RUN Mode [2]	f_{OSC} = 4 MHz, f_{CPU} = 2 MHz f_{OSC} = 8 MHz, f_{CPU} = 4 MHz f_{OSC} = 16 MHz, f_{CPU} = 8 MHz		3 5.5 10	6 11 20	mA
	Supply Current in SLOW Mode [2]	f_{OSC} = 4 MHz, f_{CPU}= 125 kHz f_{OSC} = 8 MHz, f_{CPU}= 250 kHz f_{OSC} = 16 MHz, f_{CPU}= 500 kHz		1.5 2.5 4	3 5 8	mA
	Supply Current in WAIT Mode [3]	f_{OSC} = 4MHz, f_{CPU} = 2MHz f_{OSC} = 8MHz, f_{CPU} = 4 MHz f_{OSC} = 16MHz, f_{CPU} = 8 MHz		2 3.5 6	4 7 12	mA
	Supply Current in WAIT-MINI-MUM Mode[5]	f_{OSC} = 4 MHz, f_{CPU}= 125 kHz f_{OSC} = 8 MHz, f_{CPU}= 250 kHz f_{OSC} = 16 MHz, f_{CPU}= 500 kHz		0.8 1 1.6	1.5 2 3.5	mA
	Supply Current in HALT Mode (ROM version)	I_{LOAD} = 0mA, T_A<85°C I_{LOAD} = 0mA, 85°C<T_A<125°C		1 5	10 20	µA
	Supply Current in HALT Mode (OTP version)	I_{LOAD} = 0mA, T_A<25°C I_{LOAD} = 0mA, 25°C<T_A<125°C		1	10 50	µA

Notes:

1. Hysteresis voltage between switching levels. Based on characterisation results, not tested.
2. CPU running with memory access, no DC load or activity on I/O's; clock input (OSCIN) driven by external square wave.
3. No DC load or activity on I/O's; clock input (OSCIN) driven by external square wave.
4. Except OSCIN and OSCOUT
5. WAIT Mode with SLOW Mode selected. Based on characterisation results, not tested.

7.4 RESET CHARACTERISTICS

(T_A=-40...+125°C and V_{DD}=5V±10% unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ [1]	Max	Unit
R_{ON}	Reset Weak Pull-up R_{ON}	$V_{IN} > V_{IH}$ $V_{IN} < V_{IL}$	20 60	40 120	80 240	kΩ
t_{RESET}	Pulse duration generated by watch-dog and POR reset			1		µs
t_{PULSE}	Minimum pulse duration to be applied on external $\overline{RESET}$ pin		10 [1]			ns

Note:

1) These values given only as design guidelines and are not tested.

7.5 OSCILLATOR CHARACTERISTICS

(T_A = -40°C to +125°C unless otherwise specified)

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
g_m	Oscillator transconductance		2		9	mA/V
f_{OSC}	Crystal frequency		1		16	MHz
t_{START}	Osc. start up time	V_{DD} = 5V±10%			50	ms

7.6 A/D CONVERTER CHARACTERISTICS

(T_A = -40°C to +125°C and V_{DD} = 5V±10% unless otherwise specified)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
T_{SAMPLE}	Sample Duration			31.5		$1/f_{CPU}$
Res	ADC Resolution	f_{CPU}=8MHz V_{DD}=V_{DDA}=5V		8		bit
DLE	Differential Linearity Error*			±0.6	±1	
ILE	Integral Linearity Error*				±2	
V_{AIN}	Analog Input Voltage		V_{SSA}		V_{DDA}	V
I_{ADC}	Supply current rise during A/D conversion	f_{CPU}=8MHz V_{DD}=V_{DDA}=5V		1		mA
t_{STAB}	Stabilization time after ADC enable			30		µs
t_{CONV}	Conversion Time			8 64		µs $1/f_{CPU}$
R_{AIN}	Resistance of analog sources (V_{AIN})	f_{CPU}=8MHz, T=25°C, V_{DD}=V_{DDA}=5V			15	KΩ
C_{HOLD}	Hold Capacitance				22	pF
R_{SS}	Resistance of sampling switch and internal trace				2	KΩ

***Note**: *ADC Accuracy vs. Negative Injection Current*:

For I_{inj}=0.8mA, the typical leakage induced inside the die is 1.6µA and the effect on the ADC accuracy is a loss of 1 LSB by 10KΩ increase of the external analog source impedance.
These measurement results and recommendations take worst case injection conditions into account:
- negative injection
- injection to an Input with analog capability, adjacent to the enabled Analog Input
- at 5V V_{DD} supply, and worst case temperature.

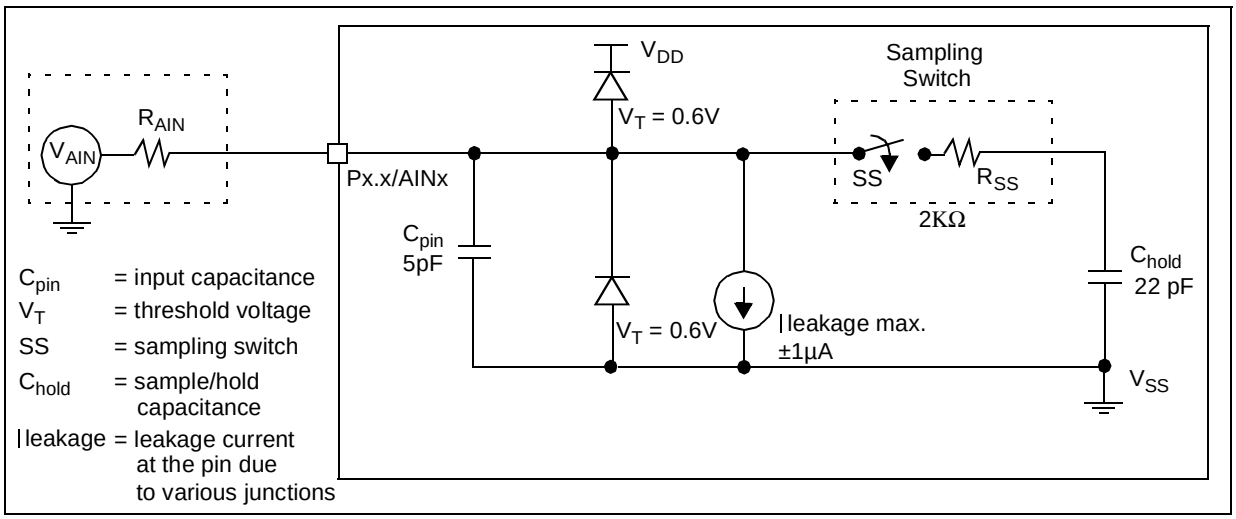
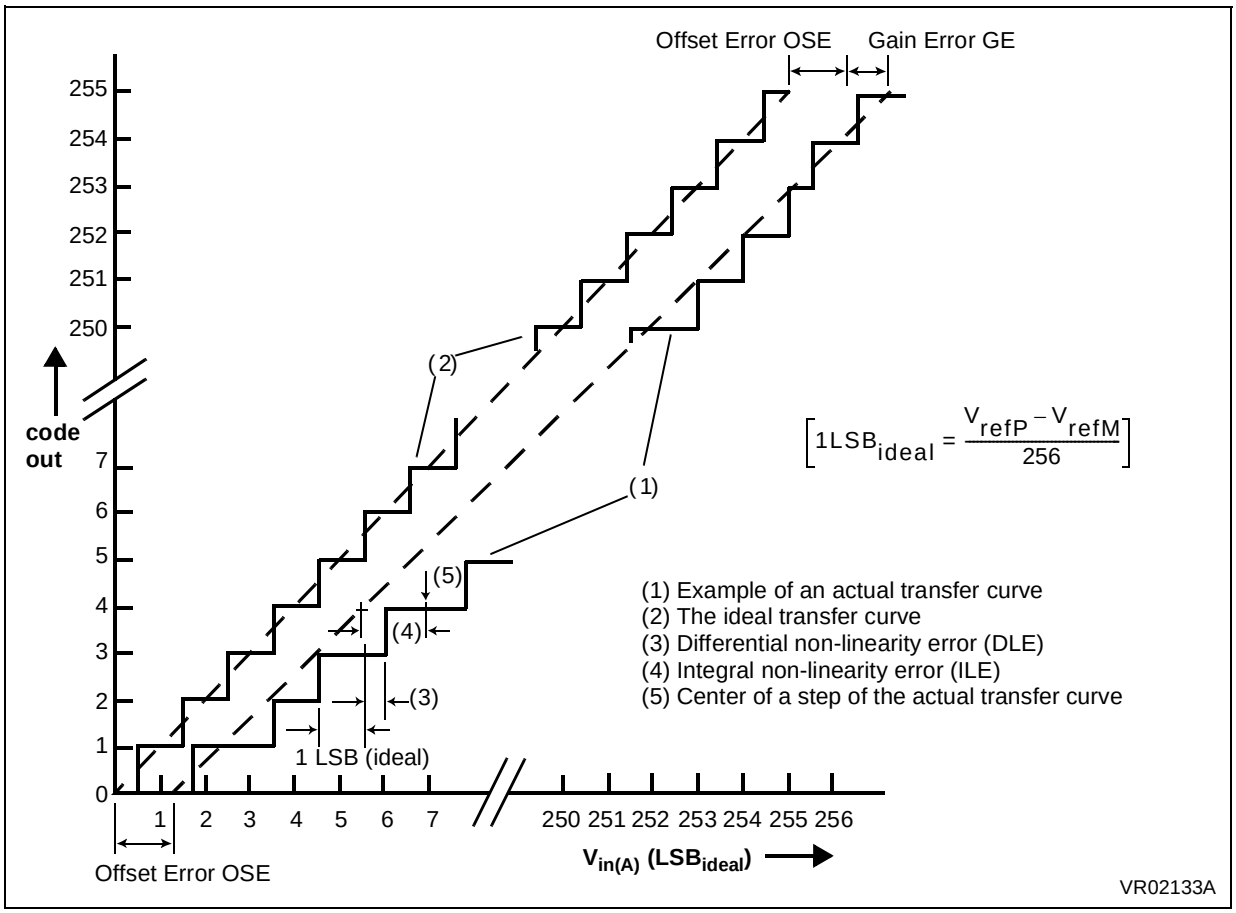


Figure 42. ADC conversion characteristics

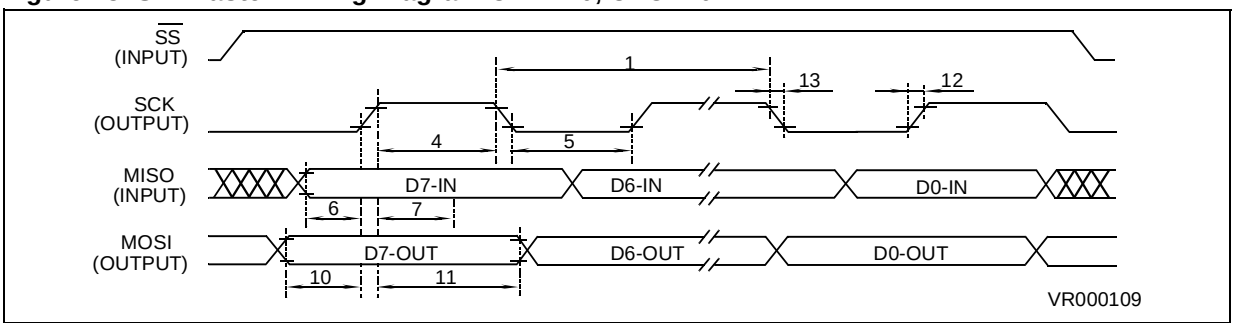


7.7 SPI CHARACTERISTICS

Serial Peripheral Interface						
Ref.	**Symbol**	**Parameter**	**Condition**	**Value**		**Unit**
				Min.	**Max.**	
	f_{SPI}	SPI frequency	Master Slave	1/128 dc	1/4 1/2	f_{CPU}
1	t_{SPI}	SPI clock period	Master Slave	4 2		t_{CPU}
2	t_{Lead}	Enable lead time	Slave	120		ns
3	t_{Lag}	Enable lag time	Slave	120		ns
4	t_{SPI_H}	Clock (SCK) high time	Master Slave	100 90		ns
5	t_{SPI_L}	Clock (SCK) low time	Master Slave	100 90		ns
6	t_{SU}	Data set-up time	Master Slave	100 100		ns
7	t_H	Data hold time (inputs)	Master Slave	100 100		ns
8	t_A	Access time (time to data active from high impedance state)	Slave	0	120	ns
9	t_{Dis}	Disable time (hold time to high impedance state)			240	ns
10	t_V	Data valid	Master (before capture edge) Slave (after enable edge)	0.25	 120	t_{CPU} ns
11	t_{Hold}	Data hold time (outputs)	Master (before capture edge) Slave (after enable edge)	0.25 0		t_{CPU} ns
12	t_{Rise}	Rise time (20% V_{DD} to 70% V_{DD}, C_L = 200pF)	Outputs: SCK,MOSI,MISO Inputs: SCK,MOSI,MISO,$\overline{SS}$		100 100	ns µs
13	t_{Fall}	Fall time (70% V_{DD} to 20% V_{DD}, C_L = 200pF)	Outputs: SCK,MOSI,MISO Inputs: SCK,MOSI,MISO,$\overline{SS}$		100 100	ns µs

Measurement points are V_{OL}, V_{OH}, V_{IL} and V_{IH} in the SPI Timing Diagram

Figure 43. SPI Master Timing Diagram CPHA=0, CPOL=0



SPI CHARACTERISTICS (Cont'd)

Measurement points are V_{OL}, V_{OH}, V_{IL} and V_{IH} in the SPI Timing Diagram

Figure 44. SPI Master Timing Diagram CPHA=0, CPOL=1

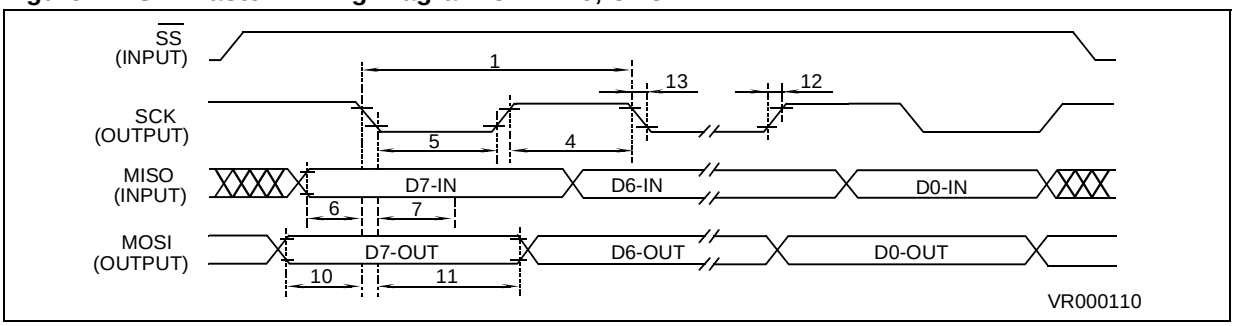


Figure 45. SPI Master Timing Diagram CPHA=1, CPOL=0

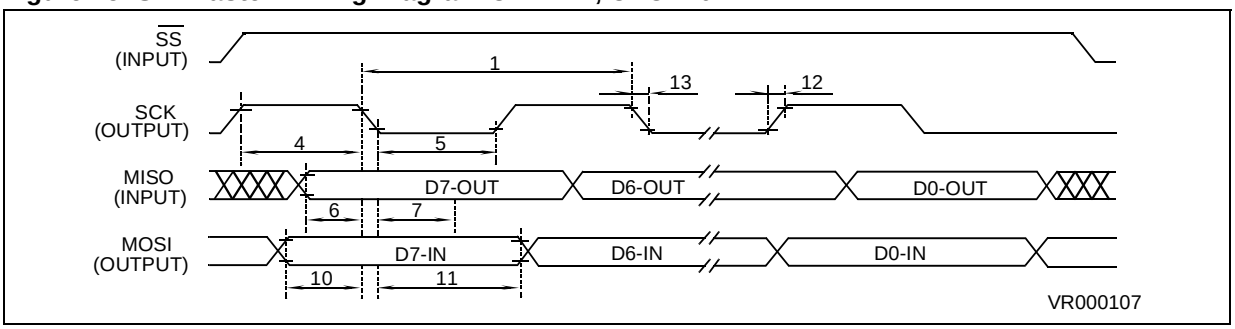
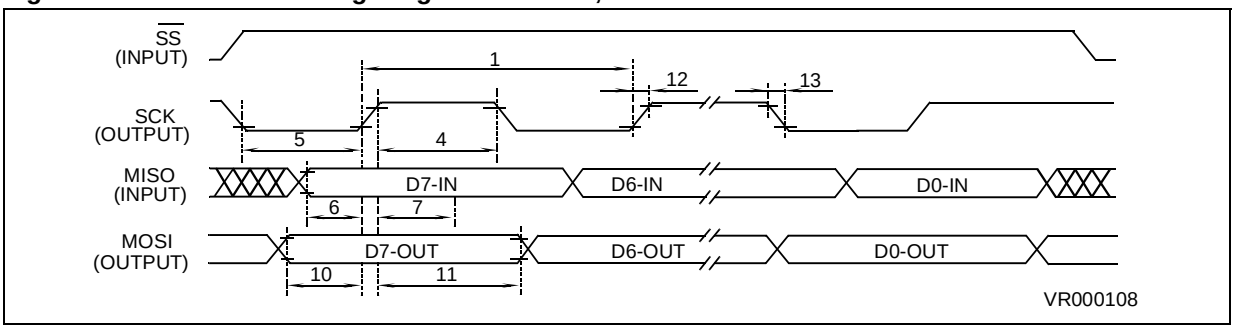


Figure 46. SPI Master Timing Diagram CPHA=1, CPOL=1



SPI CHARACTERISTICS (Cont'd)

Measurement points are V_{OL}, V_{OH}, V_{IL} and V_{IH} in the SPI Timing Diagram

Figure 47. SPI Slave Timing Diagram CPHA=0, CPOL=0

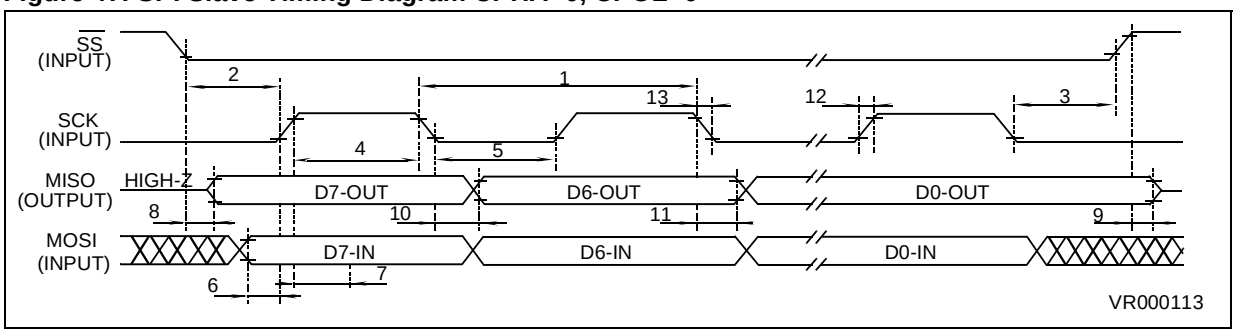


Figure 48. SPI Slave Timing Diagram CPHA=0, CPOL=1

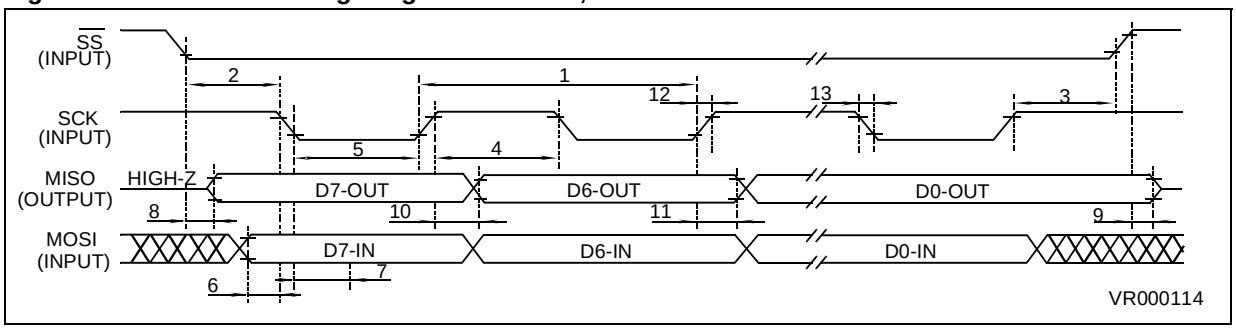


Figure 49. SPI Slave Timing Diagram CPHA=1, CPOL=0

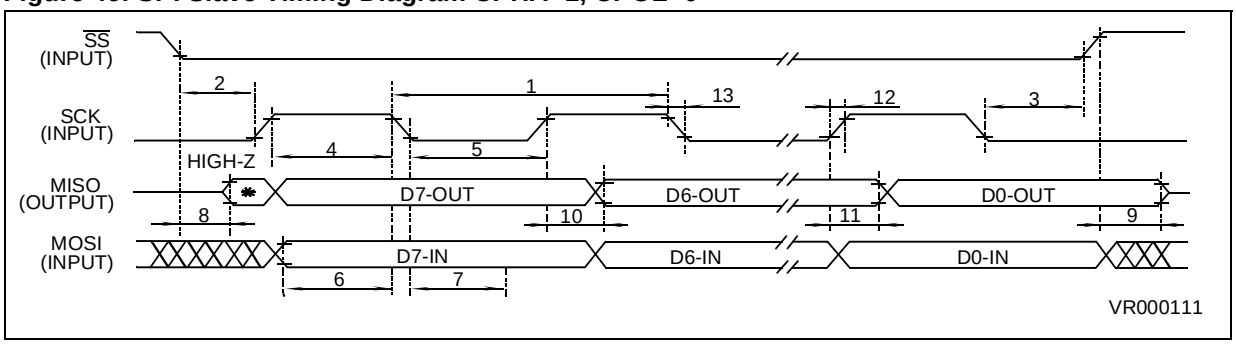
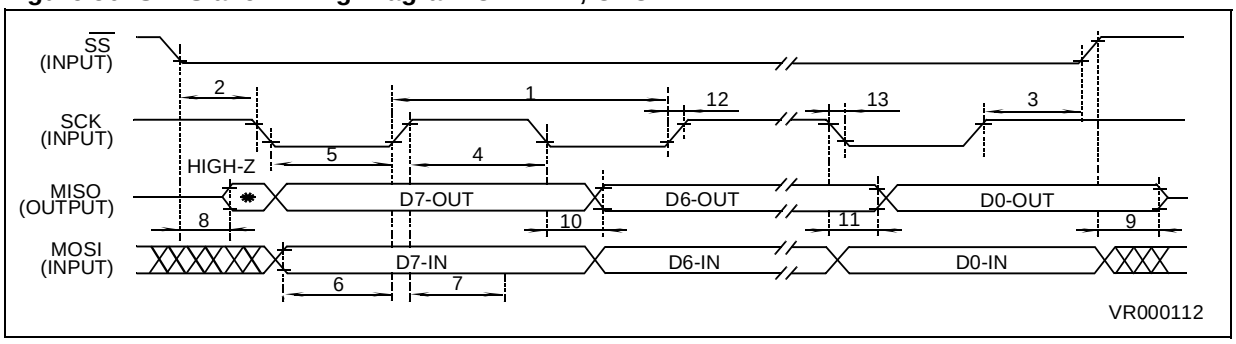


Figure 50. SPI Slave Timing Diagram CPHA=1, CPOL=1



7.8 I2C CHARACTERISTICS

I2C-Bus Electrical specifications						
Symbol	**Parameter**	**Standard mode I2C**		**Fast mode I2C**		**Unit**
		Min	**Max**	**Min**	**Max**	
V_{IL}	Low level input voltage: fixed input levels V_{DD}-related input levels	 -0.5 -0.5	 1.5 0.3 V_{DD}	 -0.5 -0.5	 1.5 0.3 V_{DD}	V
V_{IH}	High level input voltage: fixed input levels V_{DD}-related input levels	 3.0 0.7 V_{DD}	 V_{DD}+0.5 V_{DD}+0.5	 3.0 0.7 V_{DD}	 V_{DD}+0.5 V_{DD}+0.5	V
V_{HYS}	Hysteresis of Schmitt trigger inputs fixed input levels V_{DD}-related input levels	 na na	 na na	 0.2 0,05 V_{DD}		V
T_{SP}	Pulse width of spikes which must be suppressed by the input filter	na	na	0 ns	50 ns	ns
 V_{OL1} V_{OL2}	Low level output voltage (open drain and open collector) at 3 mA sink current at 6 mA sink current	 0 na	 0.4 na	 0 0	 0.4 0.6	V
T_{OF}	Output fall time from VIH min to VIL max with a bus capacitor from 10 pF to 400 pF with up to 3 mA sink current at VOL1 with up to 6 mA sink current at VOL2	 na	 250 na	 20+0.1Cb 20+0.1Cb	 250 250	ns
I	Input current each I/O pin with an input voltage between 0.4V and 0.9 V_{DD} max	- 10	10	-10	10	µA
C	Capacitor for each I/O pin		10		10	pF

Note: Cb = total capacitance of one bus line in pF.

I2C-Bus Timings						
Symbol	**Parameter**	**Standard I2C**		**Fast I2C**		**Unit**
		Min	**Max**	**Min**	**Max**	
T_{BUF}	Bus free time between a STOP and START condition	4.7		1.3		ms
$T_{HD:STA}$	Hold time START condition. After this period, the first clock pulse is generated	4.0		0.6		µs
T_{LOW}	LOW period of the SCL clock	4.7		1.3		µs
T_{HIGH}	HIGH period of the SCL clock	4.0		0.6		µs
$T_{SU:STA}$	Set-up time for a repeated START condition	4.7		0.6		µs
$T_{HD:DAT}$	Data hold time	0 (1)		0 (1)	0.9 (2)	ns
$T_{SU:DAT}$	Data set-up time	250		100		ns
T_R	Rise time of both SDA and SCL signals		1000	20+0.1Cb	300	ns
TF	Fall time of both SDA and SCL signals		300	20+0.1Cb	300	ns
$T_{SU:STO}$	Set-up time for STOP condition	4.0		0.6		ns
Cb	Capacitive load for each bus line		400		400	pF

Note 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
Note 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

8 GENERAL INFORMATION

8.1 EPROM ERASURE

EPROM version 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.

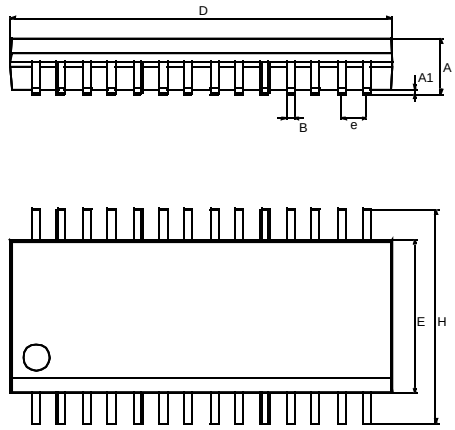
It is recommended that the EPROM devices be kept out of direct sunlight, since the UV content of sunlight can be sufficient to cause functional failure. Extended exposure to room level fluorescent lighting may also cause erasure.

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^2 is required to erase the device. It will be erased in 15 to 20 minutes if such a UV lamp with a 12mW/cm^2 power rating is placed 1 inch from the device window without any interposed filters.

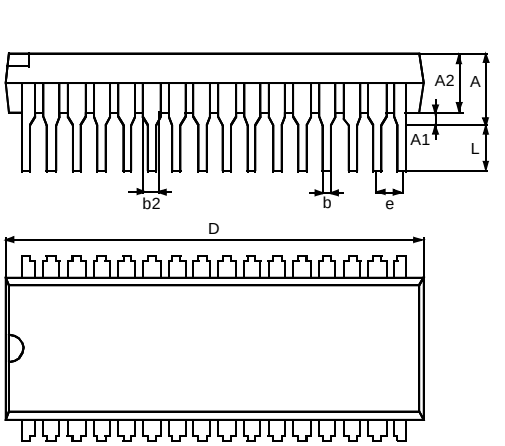
8.2 PACKAGE MECHANICAL DATA

Figure 51. 28-Pin Plastic Small Outline Package, 300-mil Width



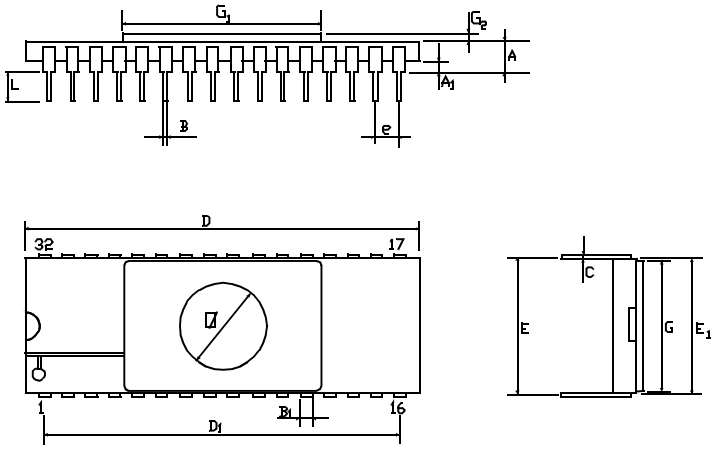
Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A	2.35		2.65	0.093		0.104
A1	0.10		0.30	0.004		0.012
B	0.33		0.51	0.013		0.020
C	0.23		0.32	0.009		0.013
D	17.70		18.10	0.697		0.713
E	7.40		7.60	0.291		0.299
e		1.27			0.050	
H	10.00		10.65	0.394		0.419
h	0.25		0.75	0.010		0.030
α	0°		8°	0°		8°
L	0.40		1.27	0.016		0.050
	Number of Pins					
N	28					

Figure 52. 32-Pin Plastic Dual In-Line Package, Shrink 400-mil Width



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A	3.56	3.76	5.08	0.140	0.148	0.200
A1	0.51			0.020		
A2	3.05	3.56	4.57	0.120	0.140	0.180
b	0.36	0.46	0.58	0.014	0.018	0.023
b1	0.76	1.02	1.40	0.030	0.040	0.055
C	0.20	0.25	0.36	0.008	0.010	0.014
D	27.43		28.45	1.080	1.100	1.120
E	9.91	10.41	11.05	0.390	0.410	0.435
E1	7.62	8.89	9.40	0.300	0.350	0.370
e		1.78			0.070	
eA		10.16			0.400	
eB			12.70			0.500
eC			1.40			0.055
L	2.54	3.05	3.81	0.100	0.120	0.150
	Number of Pins					
N	32					

Figure 53. 32-Pin Shrink Ceramic Dual In-Line Package



CDIP32SW

Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A			3.63			0.143
A1	0.38			0.015		
B	0.36	0.46	0.58	0.014	0.018	0.023
B1	0.64	0.89	1.14	0.025	0.035	0.045
C	0.20	0.25	0.36	0.008	0.010	0.014
D	29.41	29.97	30.53	1.158	1.180	1.202
D1		26.67			1.050	
E		10.16			0.400	
E1	9.45	9.91	10.36	0.372	0.390	0.408
e		1.78			0.070	
G		9.40			0.370	
G1		14.73			0.580	
G2		1.12			0.044	
L		3.30			0.130	
Ø		7.37			0.290	
	Number of Pins					
N	32					

8.3 ORDERING INFORMATION

Each device is available for production in user programmable version (OTP) as well as in factory coded version (ROM). OTP devices are shipped to customer with a default blank content FFh, while ROM factory coded parts contain the code sent by customer. There is one common EPROM version for debugging and prototyping which features the maximum memory size and peripherals of the family. Care must be taken to only use resources available on the target device.

8.3.1 Transfer Of Customer Code

Customer code is made up of the ROM contents and the list of the selected options (if any). 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 selected options are communicated to STMicroelectronics using the correctly completed OPTION LIST appended.

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

Figure 54. ROM Factory Coded Device Types

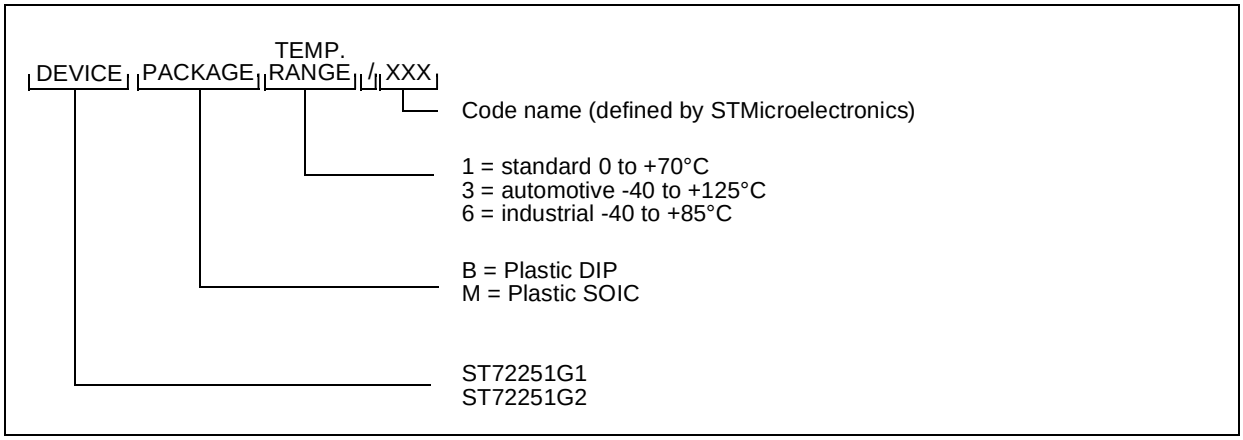
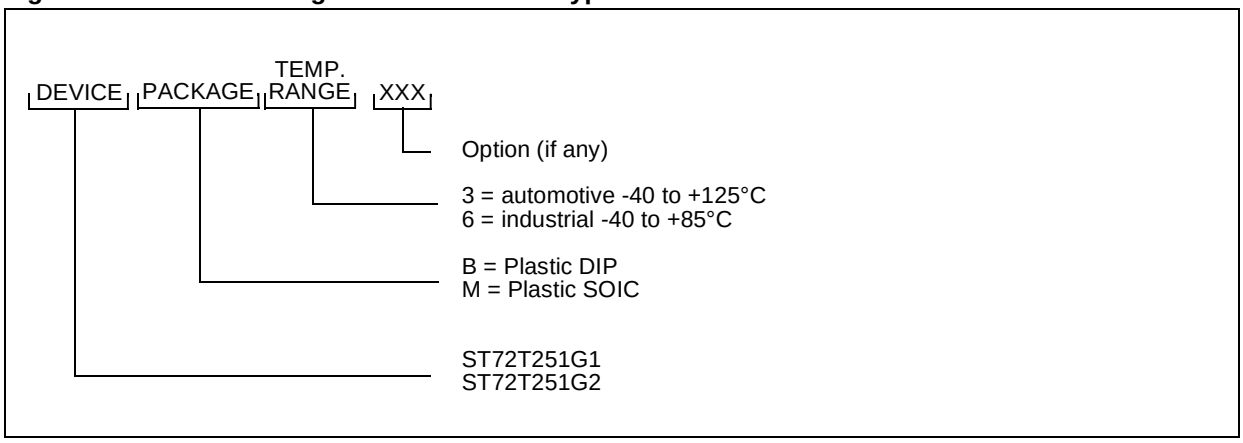


Figure 55. OTP User Programmable Device Types



Note: The ST72E251G2D0 (CERDIP 25 °C) is used as the EPROM version for the above devices.

ST72251 MICROCONTROLLER OPTION LIST

Customer . . . . . .
Address . . . . . .
. . . . . .
Contact . . . . . .
Phone No . . . . . .
Reference . . . . . .

STMicroelectronics references

Device:	[ ] ST72251		
Package:	[ ] Dual in Line Plastic	[ ] Small Outline Plastic with conditionning:	
		[ ] Standard (Stick)	
		[ ] Tape & Reel	
Temperature Range:	[ ] 0°C to + 70°C	[ ] - 40°C to + 85°C	[ ] - 40°C to + 125°C
Special Marking:	[ ] No	[ ] Yes "_ _ _ _ _ _ _ _ _ _ "	

Authorized characters are letters, digits, '.', '-', '/' and spaces only.

Maximum character count:	SDIP32:	10
	SO28:	8

Comments :
Supply Operating Range in the application:
Oscillator Fequency in the application:
Notes . . . . . .
Signature . . . . . .
Date . . . . . .

9 SUMMARY OF CHANGES

Change Description (Rev. 1.5 to 1.6)	Page
Added new External Connections section	8
Removed RP external resistor .	15
Changed ORed to ANDed in External interrupts paragraph, to read “If several input pins, connected to the same interrupt vector, are configured as interrupts, their signals are logically ANDed before entering the edge/level detection block”.	17 and 23
Added note ”Any modification of one of these two bits resets the interrupt request related to this interrupt vector.”	22
Added clamping diodes to I/O pin figure and table	25
Added sections on low power modes and interrupts to peripheral descriptions	30,42,54,72,77
Changed 16-bit timer Chapter	31 to 47
Added details to description of FOLV1 and FOLV2 bits	43
Added ADC recommended external connections	77
Added Reset characteristics section	88
Added figure to ADC electrical characteristics section	89
Change Description (Rev. 1.6 to 1.7)	
SPR2 bit reinstated in SPI chapter	63 to 75
Change Description (Rev. 1.7 to 1.8) of 31 May 2001	
SPI frequency changed from $f_{CPU}/2$ to $f_{CPU}/4$ in Table 16	74
Change Description (Rev. 1.8 to 1.9) of 7 June 2001	
Changed section 7.3 on page 88 (IDD value for OTP versions).	88