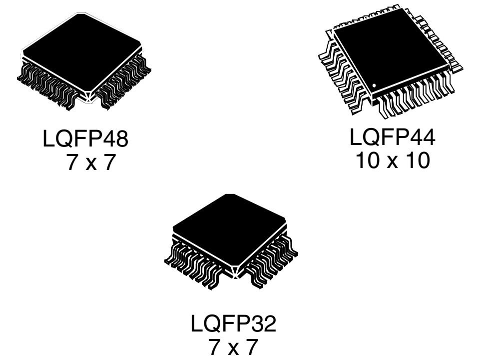
ST72340, ST72344, ST72345

8-BIT MCU WITH UP TO 16K FLASH MEMORY, 10-BIT ADC, TWO 16-BIT TIMERS, TWO I²C, SPI, SCI



- **Memories**
  - up to 16 Kbytes Program memory: Single voltage extended Flash (XFlash) with read-out and write protection, In-Circuit and In-Application Programming (ICP and IAP). 10K write/erase cycles guaranteed, data retention: 20 years at 55°C.
  - up to 1 Kbyte RAM
  - 256 bytes data EEPROM with read-out protection. 300K write/erase cycles guaranteed, data retention: 20 years at 55°C.
- **Clock, Reset and Supply Management**
  - Power On / Power Off safe reset with 3 programmable threshold levels (LVD)
  - Auxiliary Voltage Detector (AVD)
  - Clock sources: crystal/ceramic resonator oscillators, high-accuracy internal RC oscillator or external clock
  - PLL for 4x or 8x frequency multiplication
  - 5 Power Saving Modes: Slow, Wait, Halt, Auto-Wakeup from Halt and Active Halt
  - Clock output capability (f_{CPU})
- **Interrupt Management**
  - Nested interrupt controller
  - 10 interrupt vectors plus TRAP and RESET
  - 9 external interrupt lines on 4 vectors
- **Up to 34 I/O Ports**
  - up to 34 multifunctional bidirectional I/O lines
  - up to 12 high sink outputs (10 on 32-pin devices)
- **4 Timers**
  - Configurable window watchdog timer
  - Realtime base
  - 16-bit timer A with: 1 input capture, 1 output compares, external clock input, PWM and Pulse generator modes
  - 16-bit timer B with: 2 input captures, 2 output compares, PWM and Pulse generator modes
- **3 Communication Interfaces**
  - I²C Multi Master / Slave
  - I²C Slave 3 Addresses No Stretch with DMA access and Byte Pair Coherency on I²C Read
  - SCI asynchronous serial interface (LIN compatible)
  - SPI synchronous serial interface
- **1 Analog peripheral**
  - 10-bit ADC with 12 input channels (8 on 32-pin devices)
- **Instruction Set**
  - 8-bit data manipulation
  - 63 basic instructions with illegal opcode detection
  - 17 main addressing modes
  - 8 x 8 unsigned multiply instruction
- **Development tools**
  - Full hardware/software development package
  - On-Chip Debug Module

Device Summary

Features	ST72F340		ST72F344		ST72F345
Program memory - bytes	8K	16K	8K	16K	16K
RAM (stack) - bytes	512 (256)	1K (256)	512 (256)	1K (256)	1K (256)
EEPROM data - bytes	256	256	256	256	256
Common peripherals	Window Watchdog, 2 16-bit Timers, SCI, SPI, I2CMMS				
Other peripherals	-		10-bit ADC		I2C3SNS, 10-bit ADC
Int high-accuracy 1MHz RC	Not present		Present		Present
CPU Frequency	8MHz @ 3.3V to 5.5V, 4MHz @ 2.7V to 5.5V				
Temperature Range	-40°C to +85 °C				
Package	LQFP32 7x7, LQFP44 10x10				LQFP48 7x7

Rev. 2

Table of Contents

Please pay special attention to the Section "KNOWN LIMITATIONS" on page 187

1 INTRODUCTION

The ST7234x devices are members of the ST7 microcontroller family. All devices are based on a common industry-standard 8-bit core, featuring an enhanced instruction set.

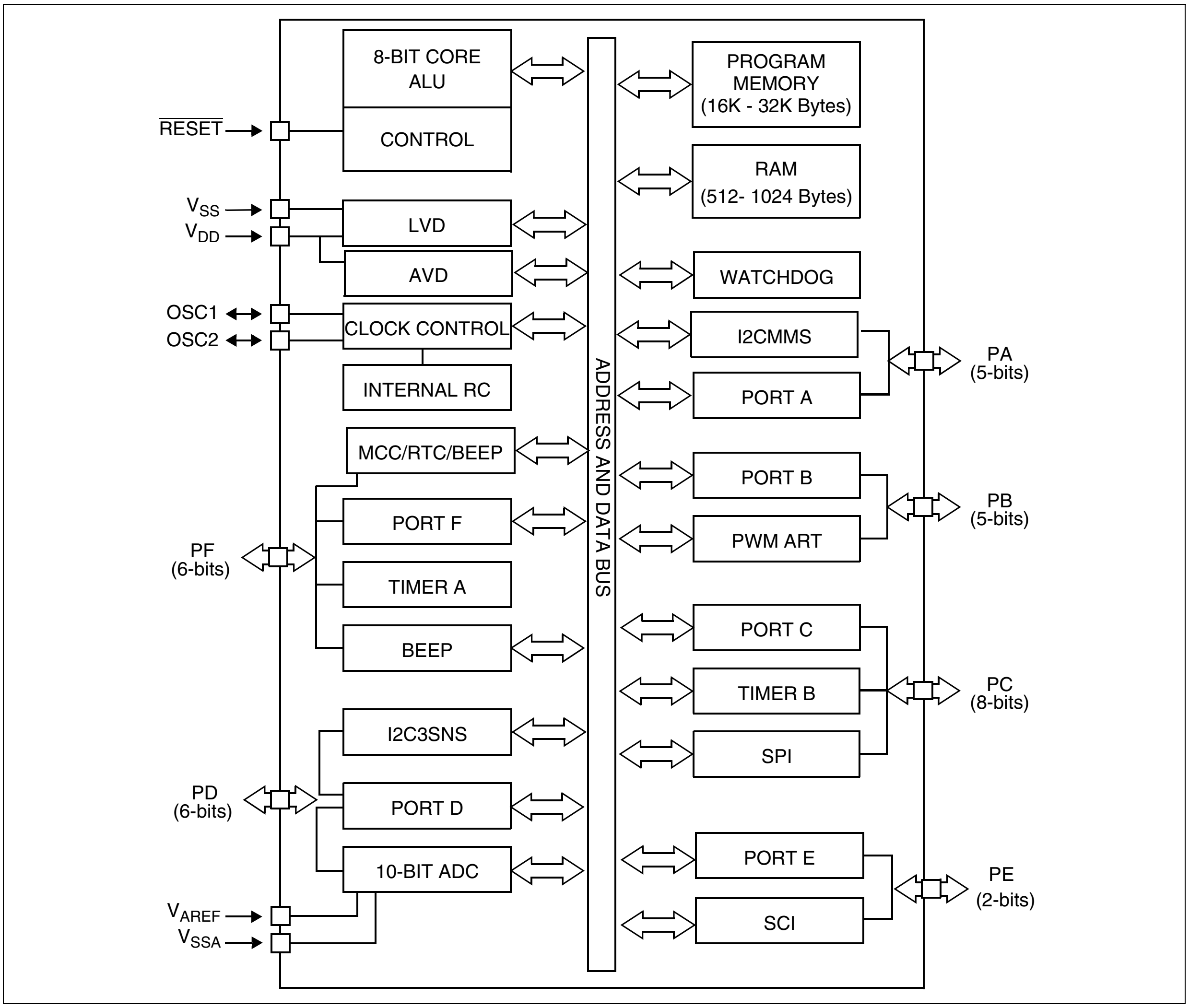
They feature single-voltage FLASH memory with byte-by-byte In-Circuit Programming (ICP) and In-Application Programming (IAP) capabilities.

Under software control, all devices can be placed in WAIT, SLOW, Auto-Wakeup from Halt, Active-HALT or HALT mode, reducing power consumption when the application is in idle or stand-by state.

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

The devices feature an on-chip Debug Module (DM) to support in-circuit debugging (ICD). For a description of the DM registers, refer to the ST7 ICC Protocol Reference Manual.

Figure 1. General Block Diagram



2 PIN DESCRIPTION

Figure 2. LQFP32 Package Pinout

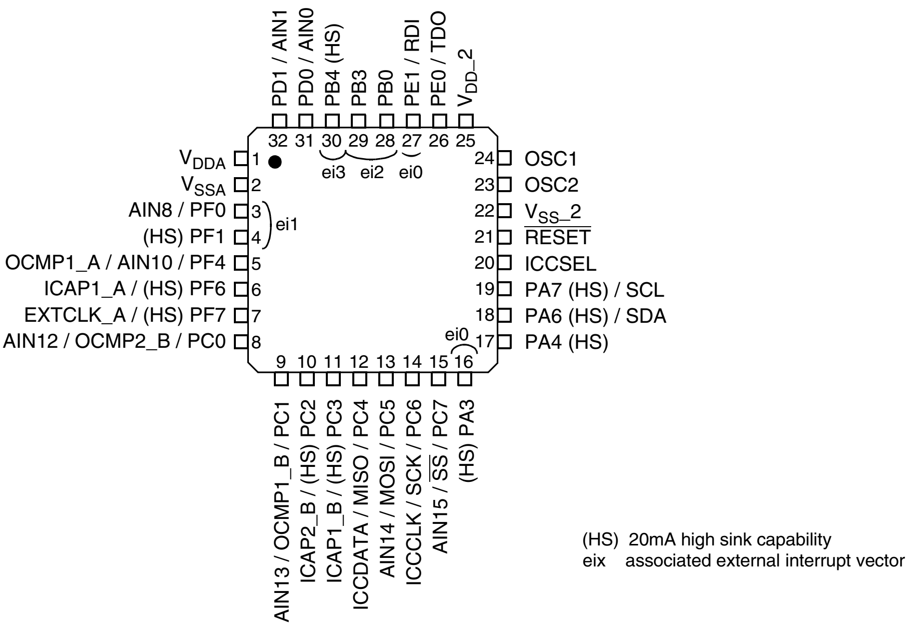
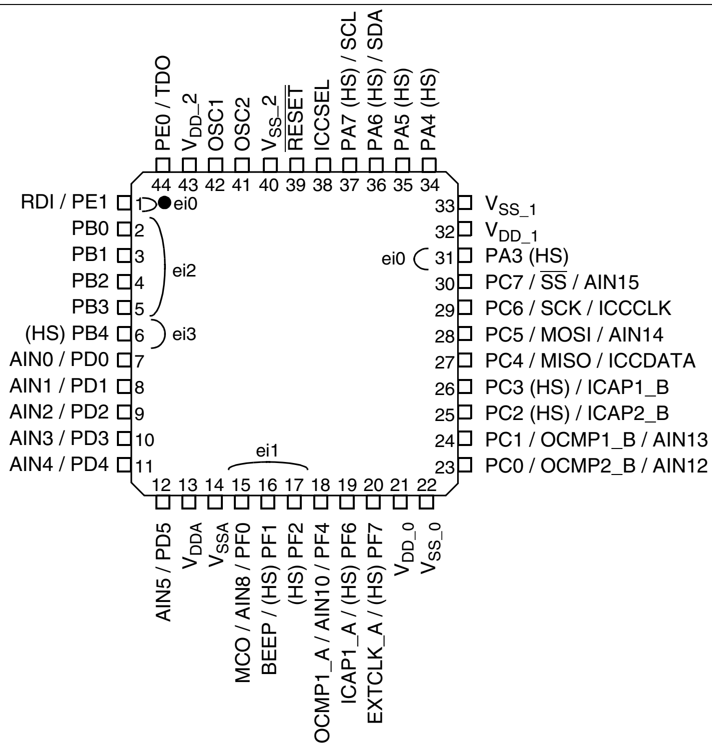
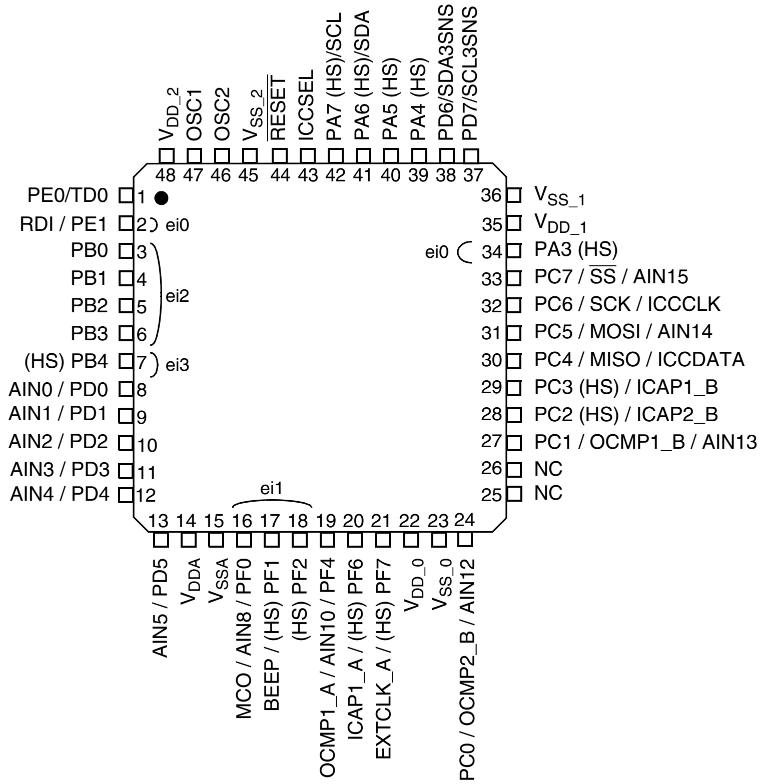


Figure 3. LQFP44 Package Pinout



PIN DESCRIPTION (Cont'd)

Figure 4. LQFP48 Package Pinout



PIN DESCRIPTION (Cont'd)

For external pin connection guidelines, refer to See "ELECTRICAL CHARACTERISTICS" on page 152.

Legend / Abbreviations for Table 1:

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

Input level: A = Dedicated analog input

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

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

Port and control configuration:

- Input: float = floating, wpu = weak pull-up, int = interrupt [1)], ana = analog
- Output: OD = open drain [2)], PP = push-pull

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

On the chip, each I/O port may have up to 8 pads. Pads that are not bonded to external pins are set in input pull-up configuration after reset through the option byte Package selection. The configuration of these pads must be kept at reset state to avoid added current consumption.

Table 1. Device Pin Description

Pin n°			Pin Name	Type	Level		Port						Main function (after reset)	Alternate Function	
LQFP32	LQFP44	LQFP48			Input	Output	Input				Output				
							float	wpu	int	ana	OD	PP			
1	13	14	V_{DDA}	S									Analog Supply Voltage		
2	14	15	V_{SSA}	S									Analog Ground Voltage		
3	15	16	PF0/MCO/ AIN8	I/O	C_T		**X**	ei1		X	X	X	Port F0	Main clock out ($f_{OSC}/2$)	ADC Analog Input 8
4	16	17	PF1 (HS)/ BEEP	I/O	C_T	HS	**X**	ei1			X	X	Port F1	Beep signal output	
	17	18	PF2 (HS)	I/O	C_T	HS	**X**		ei1		X	X	Port F2		
5	18	19	PF4/ OCMP1_A/ AIN10	I/O	C_T		**X**	X		X	X	X	Port F4	Timer A Output Compare 1	ADC Analog Input 10
6	19	20	PF6 (HS)/ ICAP1_A	I/O	C_T	HS	**X**	X			X	X	Port F6	Timer A Input Capture 1	
7	20	21	PF7 (HS)/ EXTCLK_A	I/O	C_T	HS	**X**	X			X	X	Port F7	Timer A External Clock Source	
-	21	22	V_{DD_0}	S									Digital Main Supply Voltage		
-	22	23	V_{SS_0}	S									Digital Ground Voltage		
8	23	24	PC0/ OCMP2_B/ AIN12	I/O	C_T		**X**	X		X	X	X	Port C0	Timer B Output Compare 2	ADC Analog Input 12
9	24	27	PC1/ OCMP1_B/ AIN13	I/O	C_T		**X**	X		X	X	X	Port C1	Timer B Output Compare 1	ADC Analog Input 13
10	25	28	PC2 (HS)/ ICAP2_B	I/O	C_T	HS	**X**	X			X	X	Port C2	Timer B Input Capture 2	

Pin n°			Pin Name	Type	Level		Port						Main function (after reset)	Alternate Function	
LQFP32	LQFP44	LQFP48			Input	Output	Input				Output				
							float	wpu	int	ana	OD	PP			
11	26	29	PC3 (HS)/ ICAP1_B	I/O	C_T	HS	**X**	X			X	X	Port C3	Timer B Input Capture 1	
12	27	30	PC4/MISO/ ICCDATA[3)]	I/O	C_T		**X**	X			X	X	Port C4	SPI Master In / Slave Out Data	ICC Data Input
13	28	31	PC5/MOSI/ AIN14	I/O	C_T		**X**	X		X	X	X	Port C5	SPI Master Out / Slave In Data	ADC Analog Input 14
14	29	32	PC6/SCK/ ICCCLK[3)]	I/O	C_T		**X**	X			X	X	Port C6	SPI Serial Clock	ICC Clock Output
15	30	33	PC7/$\overline{SS}$/AIN15	I/O	C_T		**X**	X		X	X	X	Port C7	SPI Slave Select (active low)	ADC Analog Input 15
16	31	34	PA3 (HS)	I/O	C_T	HS	**X**		ei0		X	X	Port A3		
-	32	35	V_{DD_1}	S									Digital Main Supply Voltage		
-	33	36	V_{SS_1}	S									Digital Ground Voltage		
-	-	37	PD7/ SCL3SNS	I/O	C_T	HS	**X**				T		Port D7	I2C3SNS Serial Clock	
-	-	38	PD6/ SDA3SNS	I/O	C_T	HS	**X**				T		Port D6	I2C3SNS Serial Data	
17	34	39	PA4 (HS)	I/O	C_T	HS	**X**	X			X	X	Port A4		
	35	40	PA5 (HS)	I/O	C_T	HS	**X**	X			X	X	Port A5		
18	36	41	PA6 (HS)/SDA	I/O	C_T	HS	**X**				T		Port A6	I2C Serial Data	
19	37	42	PA7 (HS)/SCL	I/O	C_T	HS	**X**				T		Port A7	I2C Serial Clock	
20	38	43	ICCSEL	I									ICC Mode selection		
21	39	44	$\overline{RESET}$	I/O	C_T								Top priority non maskable interrupt.		
22	40	45	V_{SS_2}	S									Digital Ground Voltage		
23	41	46	OSC2	O									Resonator oscillator inverter output		
24	42	47	OSC1	I									External clock input or Resonator oscillator inverter input		
25	43	48	V_{DD_2}	S									Digital Main Supply Voltage		
26	44	1	PE0/TDO	I/O	C_T		**X**	X			X	X	Port E0	SCI Transmit Data Out	
27	1	2	PE1/RDI	I/O	C_T		**X**		ei0		X	X	Port E1	SCI Receive Data In	
28	2	3	PB0	I/O	C_T		**X**	ei2			X	X	Port B0		
-	3	4	PB1	I/O	C_T		**X**	ei2			X	X	Port B1		
-	4	5	PB2	I/O	C_T		**X**	ei2			X	X	Port B2		
29	5	6	PB3	I/O	C_T		**X**		ei2		X	X	Port B3		
30	6	7	PB4 (HS)	I/O	C_T	HS	**X**	ei3			X	X	Port B4		
31	7	8	PD0/AIN0	I/O	C_T		**X**	X		X	X	X	Port D0	ADC Analog Input 0	
32	8	9	PD1/AIN1	I/O	C_T		**X**	X		X	X	X	Port D1	ADC Analog Input 1	
-	9	10	PD2/AIN2	I/O	C_T		**X**	X		X	X	X	Port D2	ADC Analog Input 2	
-	10	11	PD3/AIN3	I/O	C_T		**X**	X		X	X	X	Port D3	ADC Analog Input 3	
-	11	12	PD4/AIN4	I/O	C_T		**X**	X		X	X	X	Port D4	ADC Analog Input 4	
	12	13	PD5/AIN5	I/O	C_T		**X**	X		X	X	X	Port D5	ADC Analog Input 5	

Notes:

1. In the interrupt input column, “eiX” defines the associated external interrupt vector. If the weak pull-up column (wpu) is merged with the interrupt column (int), then the I/O configuration is pull-up interrupt input, else the configuration is floating interrupt input.

2. In the open drain output column, “T” defines a true open drain I/O (P-Buffer and protection diode to V_{DD} are not implemented).

3. On the BGA package, ICCDATA and ICCCLK are bonded on pins E3 and A4 respectively. They are not implemented as alternate functions on PC4 and PC6.

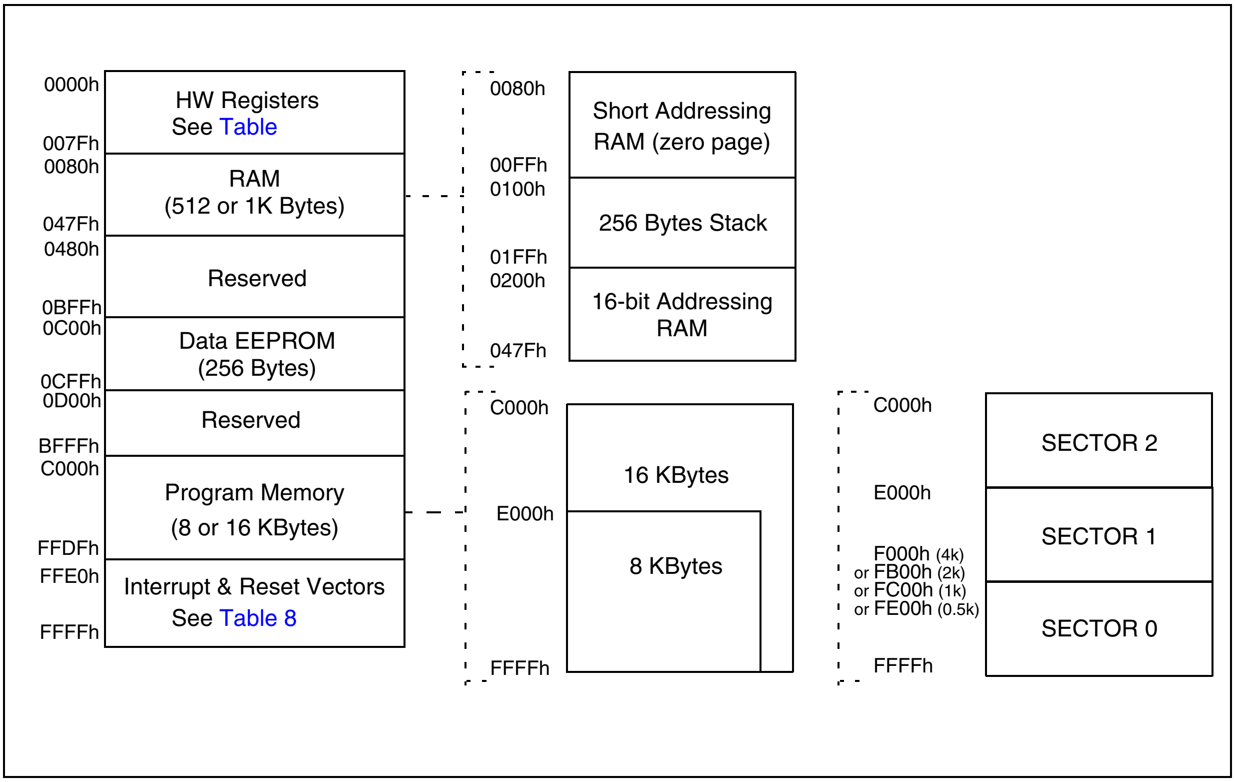
3 REGISTER & MEMORY MAP

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

The available memory locations consist of 128 bytes of register locations, up to 1 Kbytes of RAM, 256 bytes of Data EEPROM and up to 16 Kbytes of user program memory. The RAM space includes up to 256 bytes for the stack from 0100h to 01FFh.

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

Figure 5. Memory Map



REGISTER AND MEMORY MAP (Cont'd)

Table 2. Hardware Register Map

Address	Block	Register Label	Register Name	Reset Status	Remarks
0000h 0001h 0002h	Port A[2]	PADR PADDR PAOR	Port A Data Register Port A Data Direction Register Port A Option Register	00h[1] 00h 00h	R/W R/W R/W
0003h 0004h 0005h	Port B[2]	PBDR PBDDR PBOR	Port B Data Register Port B Data Direction Register Port B Option Register	00h [1] 00h 00h	R/W R/W R/W
0006h 0007h 0008h	Port C[2]	PCDR PCDDR PCOR	Port C Data Register Port C Data Direction Register Port C Option Register	00h[1] 00h 00h	R/W R/W R/W
0009h 000Ah 000Bh	Port D[2]	PDADR PDDDR PDOR	Port D Data Register Port D Data Direction Register Port D Option Register	00h[1] 00h 00h	R/W R/W R/W
000Ch 000Dh 000Eh	Port E[2]	PEDR PEDDR PEOR	Port E Data Register Port E Data Direction Register Port E Option Register	00h[1] 00h 00h	R/W R/W R/W
000Fh 0010h 0011h	Port F[2]	PFDR PFDDR PFOR	Port F Data Register Port F Data Direction Register Port F Option Register	00h[1] 00h 00h	R/W R/W R/W
0012h to 0016h	Reserved area (5 bytes)				
0017h 0018h	RC	RCCRH RCCRL	RC oscillator Control Register High RC oscillator Control Register Low	FFh 03h	R/W R/W
0019h	Reserved area (1 byte)				
001Ah to 001Fh	DM[3]	Reserved area (6 bytes)			
00020h	EEPROM	EECSR	Data EEPROM Control/Status Register	00h	R/W
0021h 0022h 0023h	SPI	SPIDR SPICR SPICSR	SPI Data I/O Register SPI Control Register SPI Control Status Register	xxh 0xh 00h	R/W R/W R/W
0024h 0025h 0026h 0027h	ITC	ISPR0 ISPR1 ISPR2 ISPR3	Interrupt Software Priority Register 0 Interrupt Software Priority Register 1 Interrupt Software Priority Register 2 Interrupt Software Priority Register 3	FFh FFh FFh FFh	R/W R/W R/W R/W
0028h		EICR	External Interrupt Control Register	00h	R/W
00029h	FLASH	FCSR	Flash Control/Status Register	00h	R/W
002Ah	WWDG	WDGCR	Watchdog Control Register	7Fh	R/W
002Bh	SI	SICSR	System Integrity Control/Status Register	000x 000xb	R/W

Address	Block	Register Label	Register Name	Reset Status	Remarks
002Ch	MCC	MCCSR	Main Clock Control/Status Register	00h	R/W
002Dh		MCCBCR	MCC Beep Control Register	00h	R/W
002Eh	AWU	AWUCSR	AWU Control/Status Register	00h	R/W
002Fh		AWUPR	AWU Prescaler Register	FFh	R/W
0030h	WWDG	WDGWR	Window Watchdog Control Register	7Fh	R/W
0031h	TIMER A	TACR2	Timer A Control Register 2	00h	R/W
0032h		TACR1	Timer A Control Register 1	00h	R/W
0033h		TACSR	Timer A Control/Status Register	xxh	R/W
0034h		TAIC1HR	Timer A Input Capture 1 High Register	xxh	Read Only
0035h		TAIC1LR	Timer A Input Capture 1 Low Register	xxh	Read Only
0036h		TAOC1HR	Timer A Output Compare 1 High Register	80h	R/W
0037h		TAOC1LR	Timer A Output Compare 1 Low Register	00h	R/W
0038h		TACHR	Timer A Counter High Register	FFh	Read Only
0039h		TACLR	Timer A Counter Low Register	FCh	Read Only
003Ah		TAACHR	Timer A Alternate Counter High Register	FFh	Read Only
003Bh		TAACLR	Timer A Alternate Counter Low Register	FCh	Read Only
003Ch		TAIC2HR	Timer A Input Capture 2 High Register	xxh	Read Only
003Dh		TAIC2LR	Timer A Input Capture 2 Low Register	xxh	Read Only
003Eh		TAOC2HR	Timer A Output Compare 2 High Register	80h	R/W
003Fh		TAOC2LR	Timer A Output Compare 2 Low Register	00h	R/W
0040h	Reserved Area (1 Byte)				
0041h	TIMER B	TBCR2	Timer B Control Register 2	00h	R/W
0042h		TBCR1	Timer B Control Register 1	00h	R/W
0043h		TBCSR	Timer B Control/Status Register	xxh	R/W
0044h		TBIC1HR	Timer B Input Capture 1 High Register	xxh	Read Only
0045h		TBIC1LR	Timer B Input Capture 1 Low Register	xxh	Read Only
0046h		TBOC1HR	Timer B Output Compare 1 High Register	80h	R/W
0047h		TBOC1LR	Timer B Output Compare 1 Low Register	00h	R/W
0048h		TBCHR	Timer B Counter High Register	FFh	Read Only
0049h		TBCLR	Timer B Counter Low Register	FCh	Read Only
004Ah		TBACHR	Timer B Alternate Counter High Register	FFh	Read Only
004Bh		TBACLR	Timer B Alternate Counter Low Register	FCh	Read Only
004Ch		TBIC2HR	Timer B Input Capture 2 High Register	xxh	Read Only
004Dh		TBIC2LR	Timer B Input Capture 2 Low Register	xxh	Read Only
004Eh		TBOC2HR	Timer B Output Compare 2 High Register	80h	R/W
004Fh		TBOC2LR	Timer B Output Compare 2 Low Register	00h	R/W
0050h	SCI	SCISR	SCI Status Register	C0h	Read Only
0051h		SCIDR	SCI Data Register	xxh	R/W
0052h		SCIBRR	SCI Baud Rate Register	00h	R/W
0053h		SCICR1	SCI Control Register 1	x000 0000b	R/W
0054h		SCICR2	SCI Control Register 2	00h	R/W
0055h			Reserved area	--	
0056h		SCIERPR	SCI Extended Receive Prescaler Register	00h	R/W
0057h		SCIETPR	SCI Extended Transmit Prescaler Register	00h	R/W

Address	Block	Register Label	Register Name	Reset Status	Remarks
0058h 0059h 005Ah 005Bh 005Ch 005Dh 005Eh	I^2C	I2CCR I2CSR1 I2CSR2 I2CCCR I2COAR1 I2COAR2 I2CDR	I^2C Control Register I^2C Status Register 1 I^2C Status Register 2 I^2C Clock Control Register I^2C Own Address Register 1 I^2C Own Address Register2 I^2C Data Register	00h 00h 00h 00h 00h 40h 00h	R/W Read Only Read Only R/W R/W R/W R/W
005Fh	Reserved area (1 byte)				
0060h 0061h 0062h 0063h 0064h 0065h 0066h 0067h 0068h 0069h	I^2C3SNS	I2C3SCR1 I2C3SCR2 I2C3SSR I2C3SBCR I2C3SSAR1 I2C3SCAR1 I2C3SSAR2 I2C3SCAR2 I2C3SSAR3 I2C3SCAR3	I^2C3SNS Control Register 1 I^2C3SNS Control Register 2 I^2C3SNS Status Register I^2C3SNS Byte Count Register I^2C3SNS Slave Address 1 Register I^2C3SNS Current Address 1 Register I^2C3SNS Slave Address 2 Register I^2C3SNS Current Address 2 Register I^2C3SNS Slave Address 3 Register I^2C3SNS Current Address 3 Register	00h 00h 00h 00h 00h 00h 00h 00h 00h 00h	R/W R/W Read Only Read Only R/W R/W R/W R/W R/W R/W
0070h 0071h 0072h	ADC	ADCCSR ADCDRH ADCDRL	A/D Control Status Register A/D Data Register High A/D Data Low Register	00h xxh 0000 00xxb	R/W Read Only Read Only
0073h to 007Fh	Reserved area (13 bytes)				

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

Notes:

1. The contents of the I/O port DR registers are readable only in output configuration. In input configuration, the values of the I/O pins are returned instead of the DR register contents.

2. The bits associated with unavailable pins must always keep their reset value.

3. For a description of the Debug Module registers, see ST7 ICC protocol reference manual.

4 FLASH PROGRAM MEMORY

4.1 Introduction

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

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

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

4.2 Main Features

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

4.3 PROGRAMMING MODES

The ST7 can be programmed in three different ways:

– Insertion in a programming tool. In this mode, FLASH sectors 0 and 1, option byte row and data EEPROM (if present) can be programmed or erased.

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

– In-Application Programming. In this mode, sector 1 and data EEPROM (if present) can be programmed or erased without removing the device from the application board and while the application is running.

4.3.1 In-Circuit Programming (ICP)

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

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

– Download ICP Driver code in RAM from the ICCDATA pin

– Execute ICP Driver code in RAM to program the FLASH memory

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

4.3.2 In Application Programming (IAP)

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

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

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

FLASH PROGRAM MEMORY (Cont'd)

4.4 ICC interface

ICP needs a minimum of 4 and up to 7 pins to be connected to the programming tool. These pins are:

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

Notes:

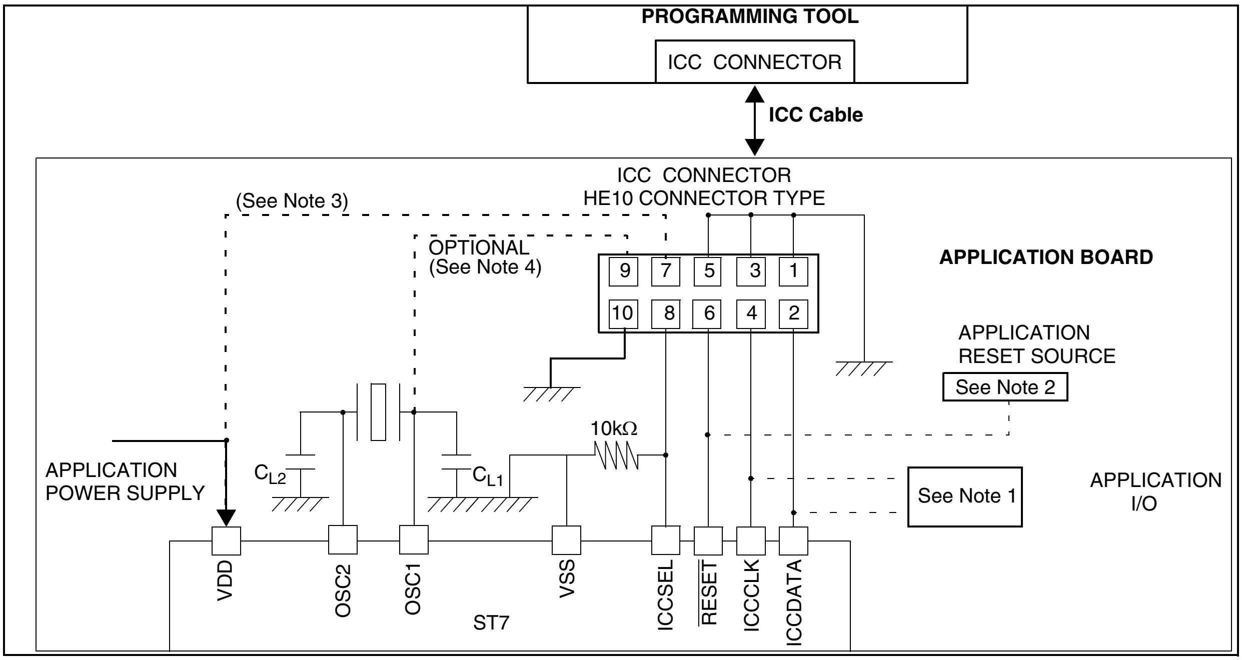
1. If the ICCCLK or ICCDATA pins are only used as outputs in the application, no signal isolation is necessary. As soon as the Programming Tool is plugged to the board, even if an ICC session is not in progress, the ICCCLK and ICCDATA pins are not available for the application. If they are used as inputs by the application, isolation such as a serial resistor has to be implemented in case another device forces the signal. Refer to the Programming Tool documentation for recommended resistor values.

2. During the ICP session, the programming tool must control the $\overline{\text{RESET}}$ pin. This can lead to conflicts between the programming tool and the application reset circuit if it drives more than 5mA at high level (push pull output or pull-up resistor<1K). A schottky diode can be used to isolate the application RESET circuit in this case. When using a classical RC network with R>1K or a reset management IC with open drain output and pull-up resistor>1K, no additional components are needed. In all cases the user must ensure that no external reset is generated by the application during the ICC session.

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

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

Figure 6. Typical ICC Interface



FLASH PROGRAM MEMORY (Cont'd)

4.5 Memory Protection

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

4.5.1 Read out Protection

Readout protection, when selected provides a protection against program memory content extraction and against write access to Flash memory. Even if no protection can be considered as totally unbreakable, the feature provides a very high level of protection for a general purpose microcontroller. Both program and data E^2 memory are protected.

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

Read-out protection selection depends on the device type:

– In Flash devices it is enabled and removed through the FMP_R bit in the option byte.

– In ROM devices it is enabled by mask option specified in the Option List.

4.5.2 Flash Write/Erase Protection

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

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

Write/erase protection is enabled through the FMP_W bit in the option byte.

4.6 Register Description

FLASH CONTROL/STATUS REGISTER (FCSR)
Read/Write
Reset Value: 000 0000 (00h)
1st RASS Key: 0101 0110 (56h)
2nd RASS Key: 1010 1110 (AEh)

7							0
0	0	0	0	0	OPT	LAT	PGM

Note: This register is reserved for programming using ICP, IAP or other programming methods. It controls the XFlash programming and erasing operations. For details on XFlash programming, refer to the ST7 Flash Programming Reference Manual.

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

5 DATA EEPROM

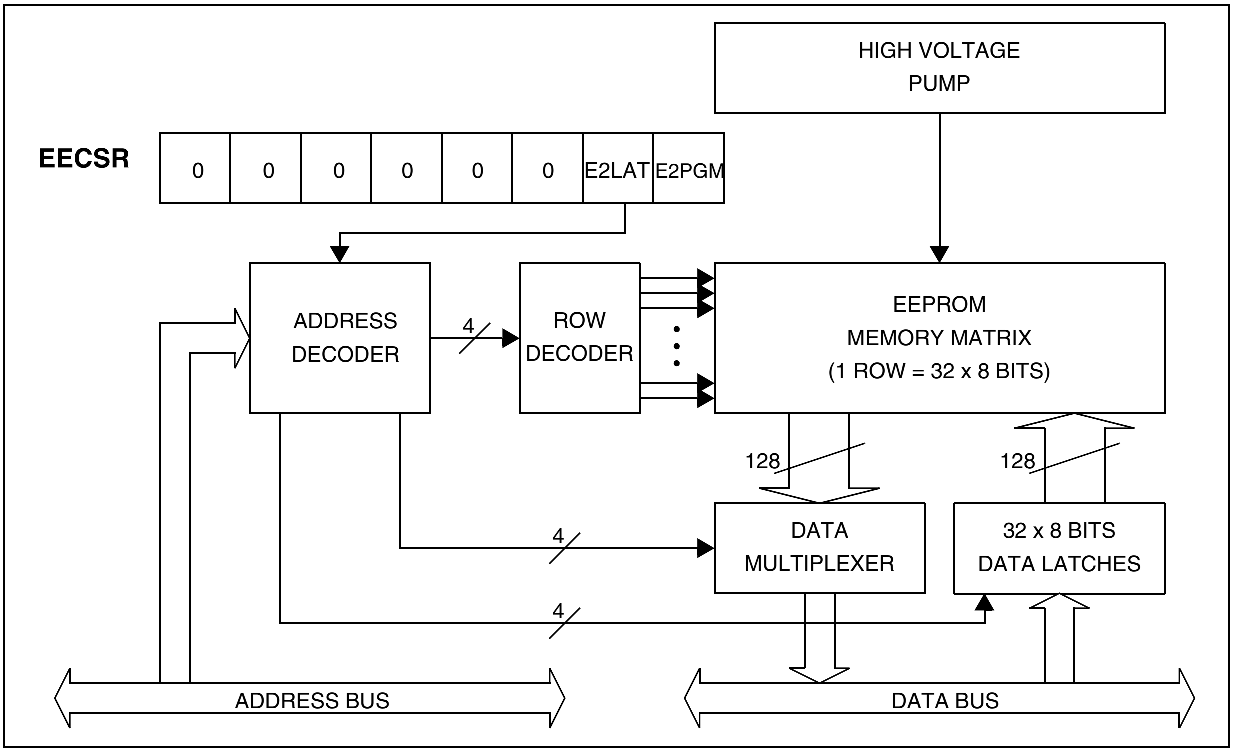
5.1 INTRODUCTION

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

5.2 MAIN FEATURES

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

Figure 7. EEPROM Block Diagram



DATA EEPROM (Cont'd)

5.3 MEMORY ACCESS

The Data EEPROM memory read/write access modes are controlled by the E2LAT bit of the EEPROM Control/Status register (EECSR). The flowchart in Figure 8 describes these different memory access modes.

Read Operation (E2LAT = 0)

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

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

Write Operation (E2LAT = 1)

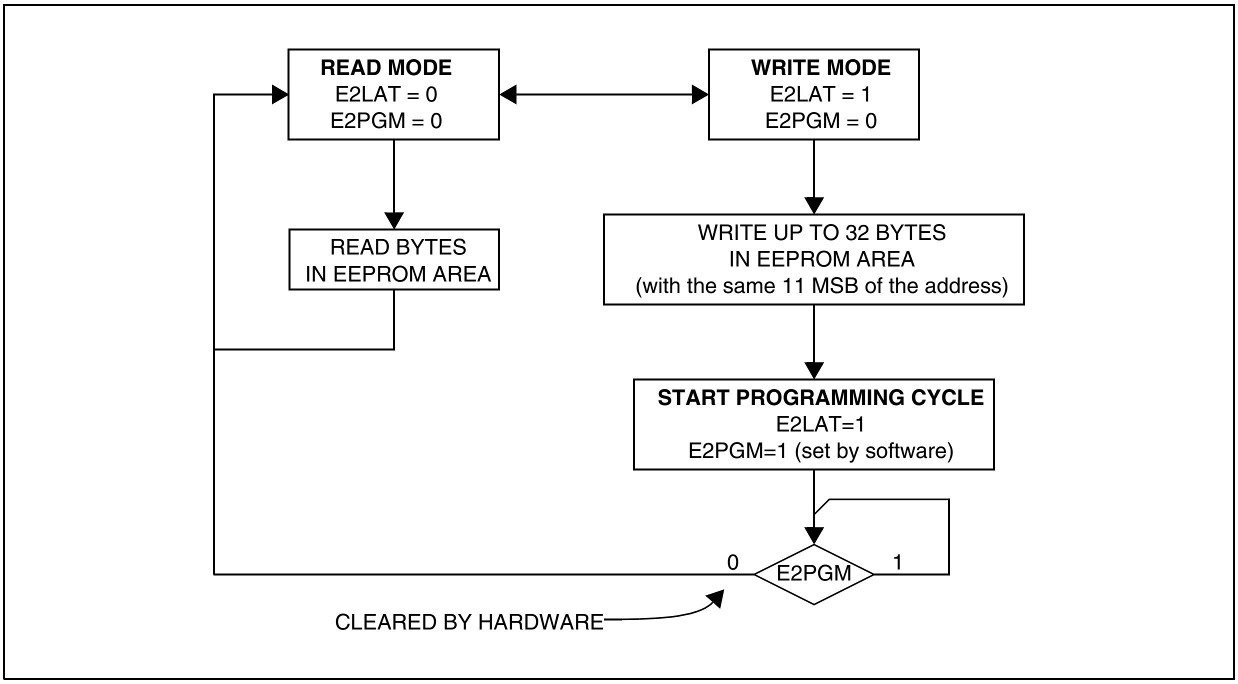
To access the write mode, the E2LAT bit has to be set by software (the E2PGM bit remains cleared). When a write access to the EEPROM area occurs, the value is latched inside the 32 data latches according to its address.

When E2PGM bit is set by the software, all the previous bytes written in the data latches (up to 32) are programmed in the EEPROM cells. The effective high address (row) is determined by the last EEPROM write sequence. To avoid wrong programming, the user must take care that all the bytes written between two programming sequences have the same high address: only the five Least Significant Bits of the address can change.

The programming cycle is fully completed when the E2PGM bit is cleared.

Note: Care should be taken during the programming cycle. Writing to the same memory location will over-program the memory (logical AND between the two write access data result) because the data latches are only cleared at the end of the programming cycle and by the falling edge of the E2LAT bit.
It is not possible to read the latched data.
This note is illustrated by the Figure 10.

Figure 8. Data EEPROM Programming Flowchart

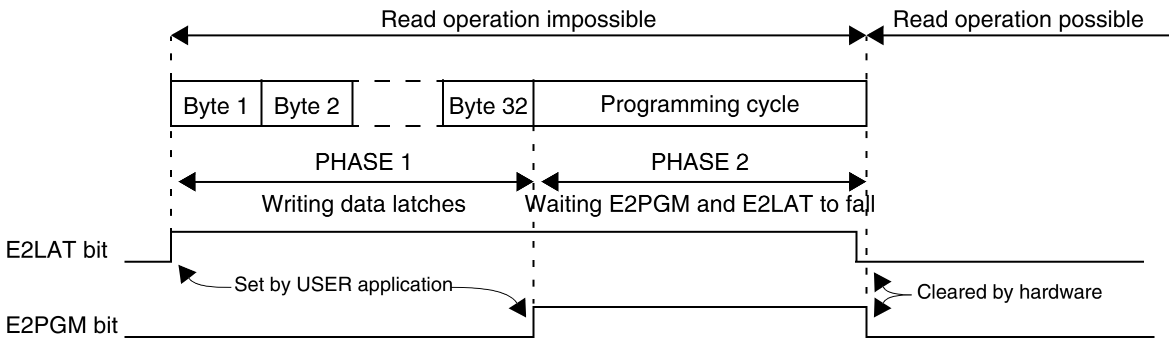


DATA EEPROM (Cont'd)

Figure 9. Data E²PROM Write Operation

ROW DEFINITION

⇓ Row / Byte ⇒	0	1	2	3	...	30	31	Physical Address
0								00h...1Fh
1								20h...3Fh
...								
N								Nx20h...Nx20h+1Fh



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

DATA EEPROM (Cont'd)

5.4 POWER SAVING MODES

Wait mode

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

Active Halt mode

Refer to Wait mode.

Halt mode

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

5.5 ACCESS ERROR HANDLING

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

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

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

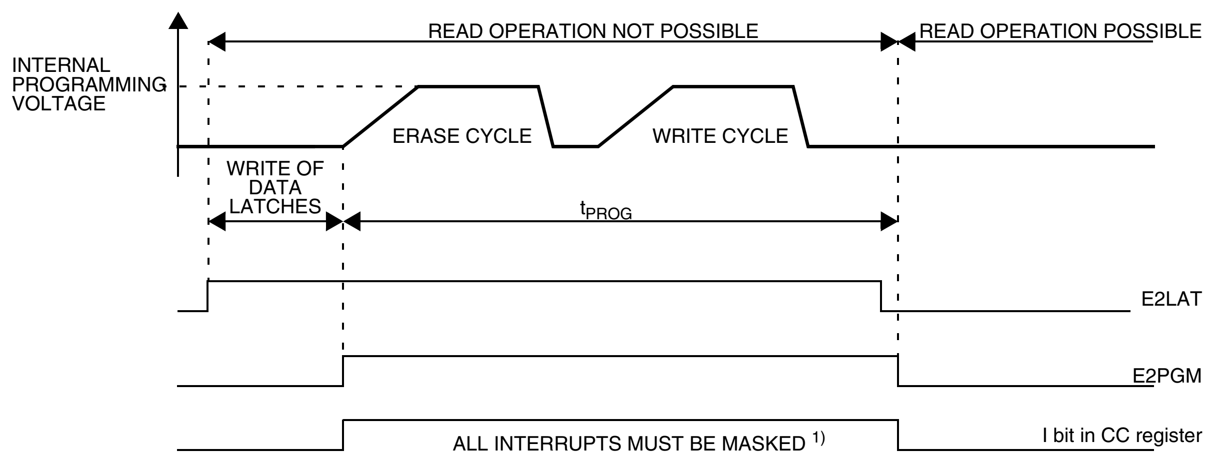
5.6 DATA EEPROM READ-OUT PROTECTION

The read-out protection is enabled through an option bit (see option byte section).

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

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

Figure 10. Data EEPROM Programming Cycle



Note 1: refer to "Programming of EEPROM data" on page 189

DATA EEPROM (Cont'd)

5.7 REGISTER DESCRIPTION

EEPROM CONTROL/STATUS REGISTER (EEC-SR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	E2LAT	E2PGM

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

Bit 1 = **E2LAT** *Latch Access Transfer*
This bit is set by software. It is cleared by hardware at the end of the programming cycle. It can only be cleared by software if the E2PGM bit is cleared.
0: Read mode
1: Write mode

Bit 0 = **E2PGM** *Programming control and status*
This bit is set by software to begin the programming cycle. At the end of the programming cycle, this bit is cleared by hardware.
0: Programming finished or not yet started
1: Programming cycle is in progress

Note: if the E2PGM bit is cleared during the programming cycle, the memory data is not guaranteed

DATA EEPROM (Cont'd)

Table 3. DATA EEPROM Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0020h	**EECSR** Reset Value	0	0	0	0	0	0	E2LAT 0	E2PGM 0

6 CENTRAL PROCESSING UNIT

6.1 INTRODUCTION

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

6.2 MAIN FEATURES

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

6.3 CPU REGISTERS

The six CPU registers shown in Figure 1 are not present in the memory mapping and are accessed by specific instructions.

Accumulator (A)

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

Index Registers (X and Y)

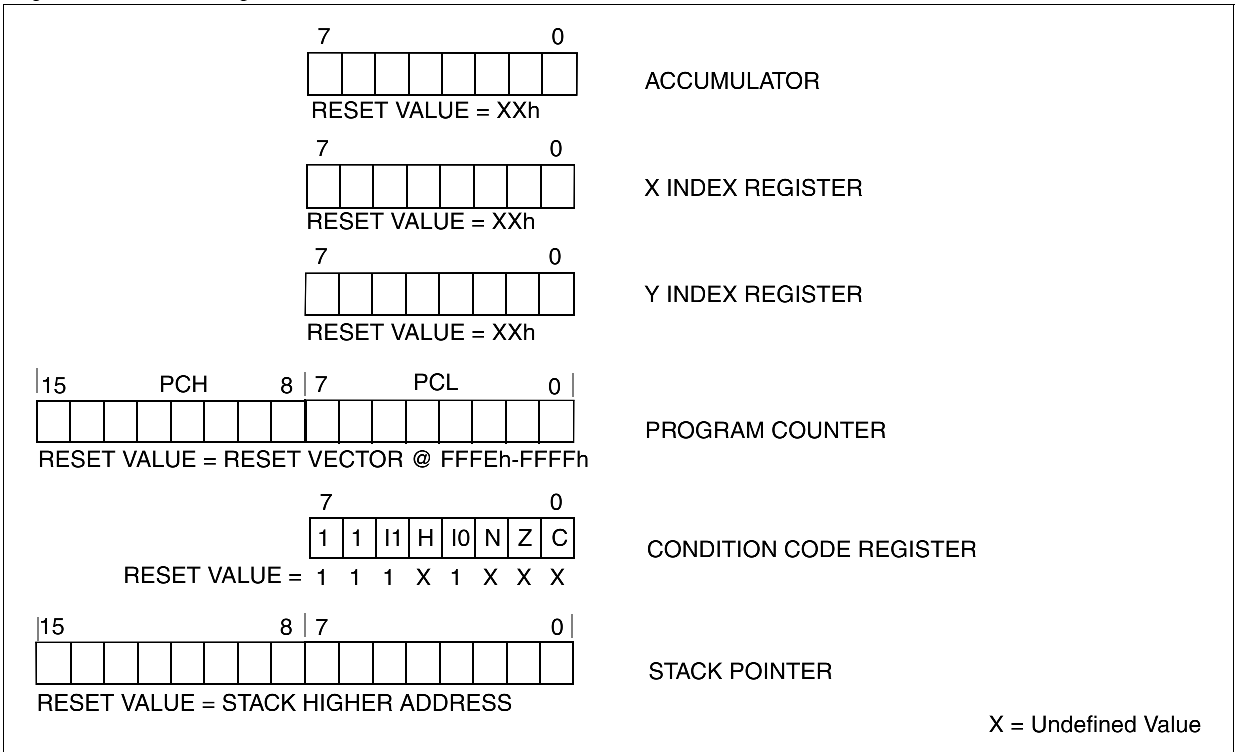
These 8-bit registers are used to create effective addresses or as temporary storage areas for data manipulation. (The Cross-Assembler generates a precede instruction (PRE) to indicate that the following instruction refers to the Y register.)

The Y register is not affected by the interrupt automatic procedures.

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 11. CPU Registers



CENTRAL PROCESSING UNIT (Cont'd)

Condition Code Register (CC)

Read/Write

Reset Value: 111x1xxx

7							0
1	1	I1	H	I0	N	Z	C

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

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

Arithmetic Management Bits

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 instructions. 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 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's a copy of the result 7^{th} bit.
0: The result of the last operation is positive or null.
1: The result of the last operation is negative (that is, the most significant bit is a logic 1).

This bit is accessed by the JRMI and JRPL instructions.

Bit 1 = **Z** *Zero*.

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.

Interrupt Management Bits

Bit 5,3 = **I1, I0** *Interrupt*

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

Interrupt Software Priority	I1	I0
Level 0 (main)	1	0
Level 1	0	1
Level 2	0	0
Level 3 (= interrupt disable)	1	1

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

See the interrupt management chapter for more details.

CENTRAL PROCESSING UNIT (Cont'd)

Stack Pointer (SP)

Read/Write

Reset Value: 01 FFh

15							8
0	0	0	0	0	0	0	1

7							0
SP7	SP6	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 12).

Since the stack is 256 bytes deep, the 8 most significant bits are forced by hardware. Following an MCU Reset, or after a Reset Stack Pointer instruction (RSP), the Stack Pointer contains its reset value (the SP7 to SP0 bits are set) which is the stack higher address.

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

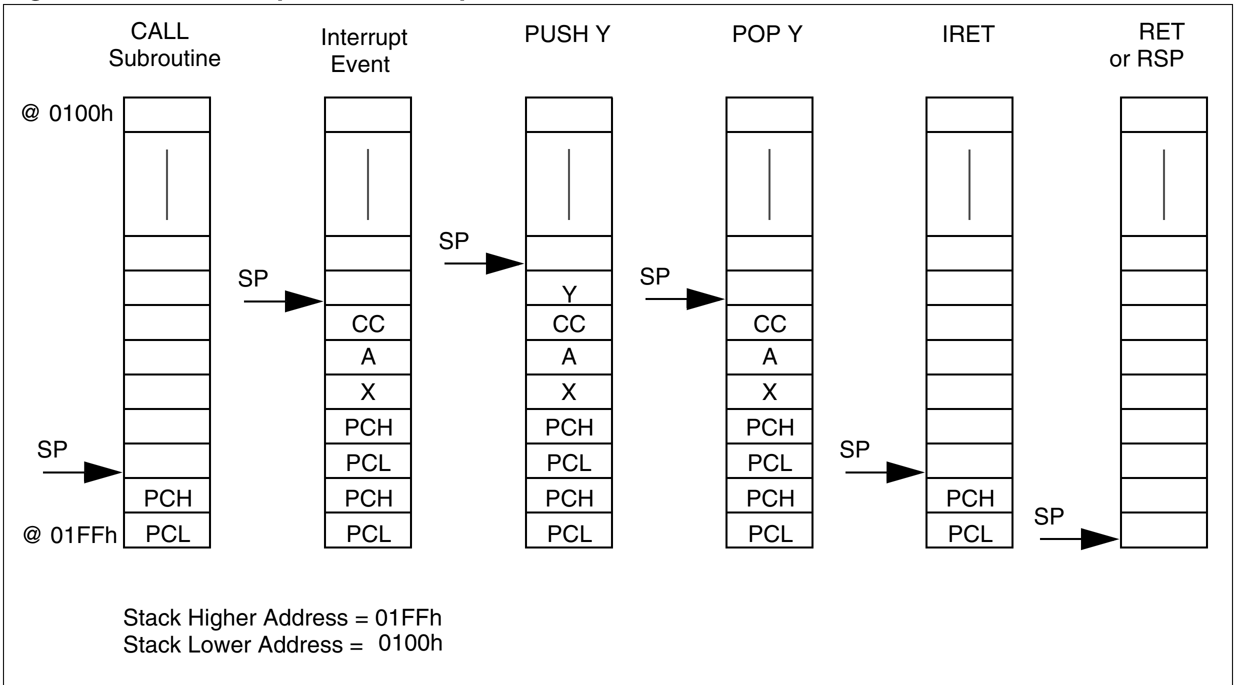
The stack is used to save the return address during a subroutine call and the CPU context during an interrupt. The user may also directly manipulate the stack by means of the PUSH and POP instructions. In the case of an interrupt, the PCL is stored at the first location pointed to by the SP. Then the other registers are stored in the next locations as shown in Figure 12.

– 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 12. Stack Manipulation Example



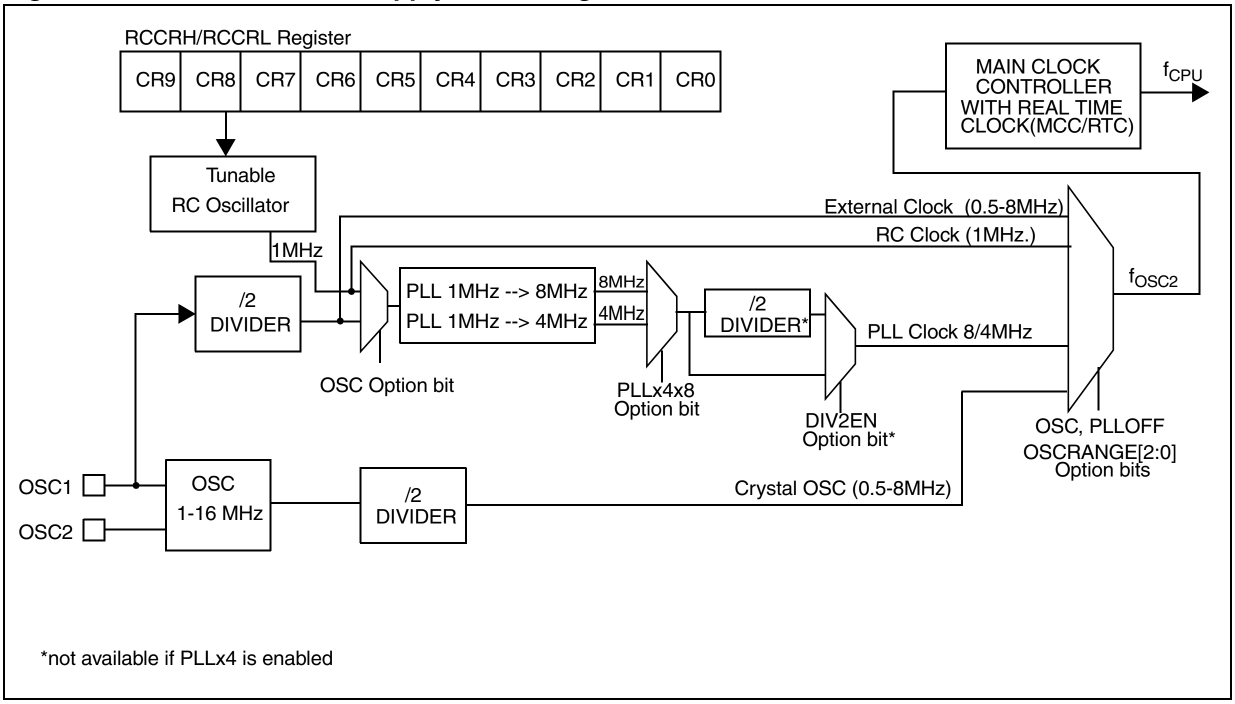
7 SUPPLY, RESET AND CLOCK MANAGEMENT

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

Main features

- Clock Management
  - 1 MHz high-accuracy internal RC oscillator (enabled by option byte)
  - 1 to 16 MHz External crystal/ceramic resonator (enabled by option byte)
  - External Clock Input (enabled by option byte)
  - PLL for multiplying the frequency by 8 or 4 (enabled by option byte)
- Reset Sequence Manager (RSM)
- System Integrity Management (SI)
  - Main supply Low voltage detection (LVD) with reset generation (enabled by option byte)
  - Auxiliary Voltage Detector (AVD) with interrupt capability for monitoring the main supply (enabled by option byte)

Figure 13. Clock, Reset and Supply Block Diagram



7.1 PHASE LOCKED LOOP

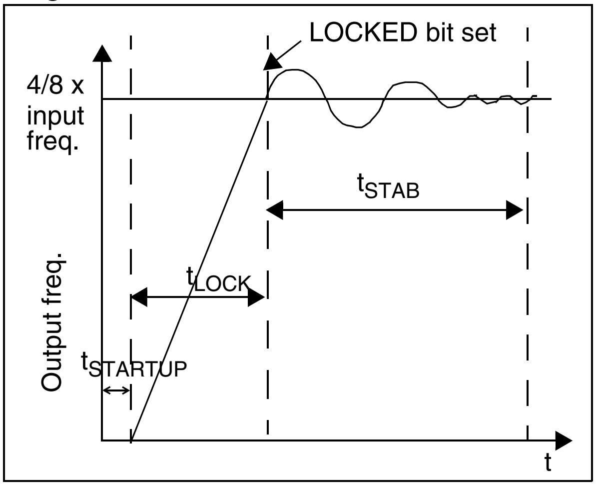
The PLL can be used to multiply a 1MHz frequency from the RC oscillator or the external clock by 4 or 8 to obtain f_{OSC} of 4 or 8 MHz. The PLL is enabled and the multiplication factor of 4 or 8 is selected by 3 option bits. Refer to Table 4 for the PLL configuration depending on the required frequency and the application voltage. Refer to Section 15.1 for the option byte description.

Table 4. PLL Configurations

Target Ratio	V_{DD}	PLL Ratio	DIV2
x4[1)]	2.7V - 3.65V	x4	OFF
x4	3.3V - 5.5V	x8	ON
x8		x8	OFF

1) For a target ratio of x4 between 3.3V - 3.65V, this is the recommended configuration.

Figure 14. PLL Output Frequency Timing Diagram



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

When the PLL output signal reaches the operating frequency, the LOCKED bit in the SICSCR register is set. Full PLL accuracy (ACC_{PLL}) is reached after a stabilization time of t_{STAB} (see Figure 14)

Refer to Section 7.5.4 on page 35 for a description of the LOCKED bit in the SICSR register.

Caution: The PLL is not recommended for applications where timing accuracy is required.

7.2 MULTI-OSCILLATOR (MO)

The main clock of the ST7 can be generated by three different source types coming from the multi-oscillator block:

- an external source
- 4 crystal or ceramic resonator oscillators
- an internal high-accuracy RC oscillator

Each oscillator is optimized for a given frequency range in terms of consumption and is selectable through the option byte. The associated hardware configurations are shown in Table 5. Refer to the electrical characteristics section for more details.

Caution: The OSC1 and/or OSC2 pins must not be left unconnected. For the purposes of Failure Mode and Effect Analysis, it should be noted that if the OSC1 and/or OSC2 pins are left unconnected, the ST7 main oscillator may start and, in this configuration, could generate an f_{OSC} clock frequency in excess of the allowed maximum (>16MHz.), putting the ST7 in an unsafe/undefined state. The product behaviour must therefore be considered undefined when the OSC pins are left unconnected.

External Clock Source

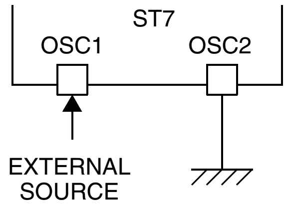
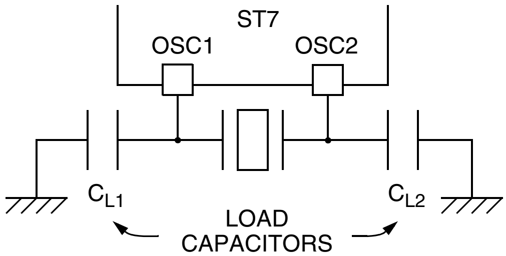
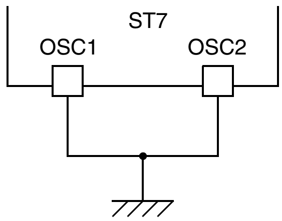
In this external clock mode, a clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC1 pin while the OSC2 pin is tied to ground.

Crystal/Ceramic Oscillators

This family of oscillators has the advantage of producing a very accurate rate on the main clock of the ST7. The selection within a list of 4 oscillators with different frequency ranges has to be done by option byte in order to reduce consumption (refer to Section 15.1 on page 181 for more details on the frequency ranges). In this mode of the multi-oscillator, the resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and start-up stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

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

Table 5. ST7 Clock Sources

	Hardware Configuration
External Clock	
Crystal/Ceramic Resonators	
Internal RC Oscillator	

MULTI-OSCILLATOR (Cont'd)

Internal RC Oscillator

The device contains a high-precision internal RC oscillator. It must be calibrated to obtain the frequency required in the application. This is done by software writing a calibration value in the RCCRH and RCCRL Registers.

Whenever the microcontroller is reset, the RCCR returns to its default value (FF 03h), i.e. each time the device is reset, the calibration value must be loaded in the RCCRH and RCCRL registers. Predefined calibration values are stored in XFLASH for 3 and 5V V_{DD} supply voltages at 25°C, as shown in the following table.

RCCR	Conditions	Address
RCCR0	V_{DD}=5V T_A=25°C f_{RC}=1MHz	BEE0, BEE1
RCCR1	V_{DD}=3V T_A=25°C f_{RC}=1MHz	BEE4, BEE5

Note:

– To improve clock stability, it is recommended to place a decoupling capacitor between the V_{DD} and V_{SS} pins.

– These two 10-bit values are systematically programmed by ST, including on FASTROM devices. Consequently, customers intending to use FASTROM service must not use these addresses.

– RCCR0 and RCCR1 calibration values will be erased if the read-out protection bit is reset after it has been set. See "Memory Protection" on page 17.

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

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

7.3 REGISTER DESCRIPTION

RC CONTROL REGISTER (RCCRH)

Read / Write

Reset Value: 1111 1111 (FFh)

7							0
CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2

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

RC CONTROL REGISTER (RCCRL)

Read / Write

Reset Value: 0000 0011 (03h)

7							0
0	0	0	0	0	0	CR1	CR0

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

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

This 10-bit value must be written immediately after reset to adjust the RC oscillator frequency in order to obtain the specified accuracy. The application can store the correct value for each voltage range in EEPROM and write it to this register at start-up.

0000h = maximum available frequency

03FFh = lowest available frequency

Note: To tune the oscillator, write a series of different values in the register until the correct frequency is reached. The fastest method is to use a dichotomy starting with 200h.

7.4 RESET SEQUENCE MANAGER (RSM)

7.4.1 Introduction

The reset sequence manager includes three RESET sources as shown in Figure 16:

- External RESET source pulse
- Internal LVD RESET (Low Voltage Detection)
- Internal WATCHDOG RESET

Note: A reset can also be triggered following the detection of an illegal opcode or prebyte code. Refer to Section 12.2.1 on page 149 for further details.

These sources act on the RESET pin and it is always kept low during the delay phase.

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

The basic RESET sequence consists of 3 phases as shown in Figure 15:

- Active Phase depending on the RESET source
- 256 or 4096 CPU clock cycle delay (selected by option byte)
- RESET vector fetch

The 256 or 4096 CPU clock cycle delay allows the oscillator to stabilise and ensures that recovery has taken place from the Reset state. The shorter or longer clock cycle delay should be selected by option byte to correspond to the stabilization time of the external oscillator used in the application (see Section 15.1 on page 181).

The RESET vector fetch phase duration is 2 clock cycles.

Figure 15. RESET Sequence Phases

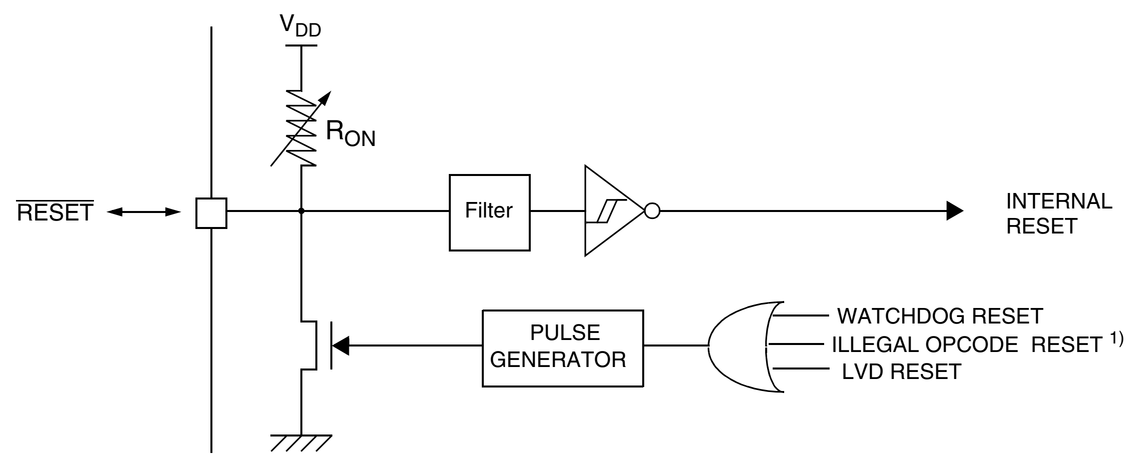
RESET		
Active Phase	INTERNAL RESET 256 or 4096 CLOCK CYCLES	FETCH VECTOR

7.4.2 Asynchronous External RESET pin

The RESET pin is both an input and an open-drain output with integrated R_{ON} weak pull-up resistor. This pull-up has no fixed value but varies in accordance with the input voltage. It can be pulled low by external circuitry to reset the device. See "ELECTRICAL CHARACTERISTICS" on page 152 for more details.

A RESET signal originating from an external source must have a duration of at least $t_{h(RSTL)in}$ in order to be recognized (see Figure 17). This detection is asynchronous and therefore the MCU can enter reset state even in HALT mode.

Figure 16. Reset Block Diagram



Note 1: See "Illegal Opcode Reset" on page 149. for more details on illegal opcode reset conditions.

RESET SEQUENCE MANAGER (Cont'd)

The RESET pin is an asynchronous signal which plays a major role in EMS performance. In a noisy environment, it is recommended to follow the guidelines mentioned in the electrical characteristics section.

If the external RESET pulse is shorter than $t_{w(RSTL)out}$ (see short ext. Reset in Figure 17), the signal on the RESET pin may be stretched. Otherwise the delay will not be applied (see long ext. Reset in Figure 17). Starting from the external RESET pulse recognition, the device RESET pin acts as an output that is pulled low during at least $t_{w(RSTL)out}$.

7.4.3 External Power-On RESET

If the LVD is disabled by option byte, to start up the microcontroller correctly, the user must ensure by means of an external reset circuit that the reset signal is held low until V_{DD} is over the minimum level specified for the selected f_{OSC} frequency. (see "OPERATING CONDITIONS" on page 154)

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

7.4.4 Internal Low Voltage Detector (LVD) RESET

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

- Power-On RESET
- Voltage Drop RESET

The device RESET pin acts as an output that is pulled low when $V_{DD}<V_{IT+}$ (rising edge) or $V_{DD}<V_{IT-}$ (falling edge) as shown in Figure 17.

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

Note:

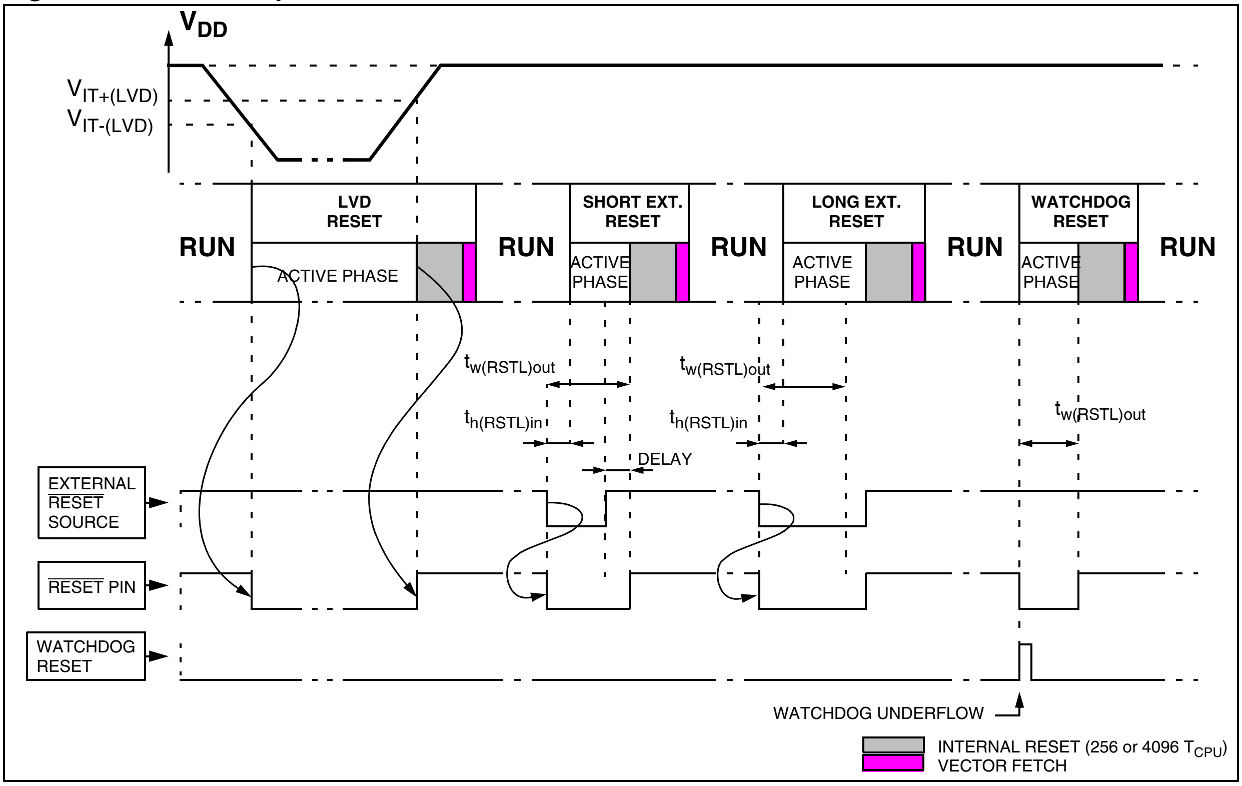
It is recommended to make sure that the V_{DD} supply voltage rises monotonously when the device is exiting from Reset, to ensure the application functions properly.

7.4.5 Internal Watchdog RESET

The RESET sequence generated by a internal Watchdog counter overflow is shown in Figure 17.

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

Figure 17. RESET Sequences



7.5 SYSTEM INTEGRITY MANAGEMENT (SI)

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

Note: A reset can also be triggered following the detection of an illegal opcode or prebyte code. Refer to Section 12.2.1 on page 149 for further details.

7.5.1 Low Voltage Detector (LVD)

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

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

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

– V_{IT+} when V_{DD} is rising
– V_{IT-} when V_{DD} is falling

The LVD function is illustrated in Figure 18.

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

Note: LVD Threshold Configuration

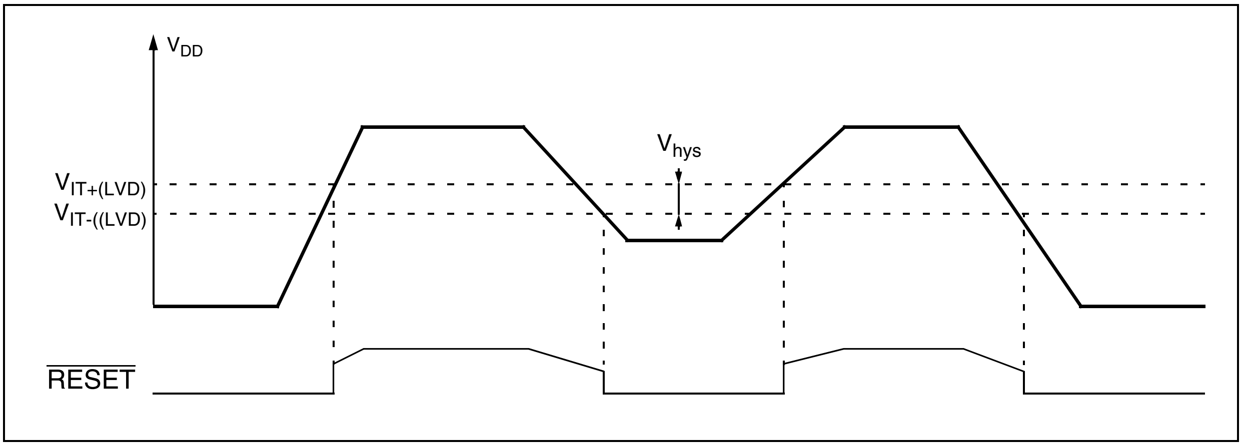
The voltage threshold can be configured by option byte to be low, medium or high. The configuration should be chosen depending on the f_{OSC} and V_{DD} parameters in the application. When correctly configured, the LVD ensures safe power-on and power-off conditions for the microcontroller without using any external components.

To determine which LVD thresholds to use:

– Define the minimum operating voltage for the application $V_{APP(min)}$
– Refer to the Electrical Characteristics section to get the minimum operating voltage for the MCU at the application frequency $V_{DD(min)}$.
– Select the LVD threshold that ensures that the internal RESET is released at $V_{APP(min)}$ and activated at $V_{DD(MCUmin)}$

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

Figure 18. Low Voltage Detector vs Reset



SYSTEM INTEGRITY MANAGEMENT (Cont'd)

7.5.2 Auxiliary Voltage Detector (AVD)

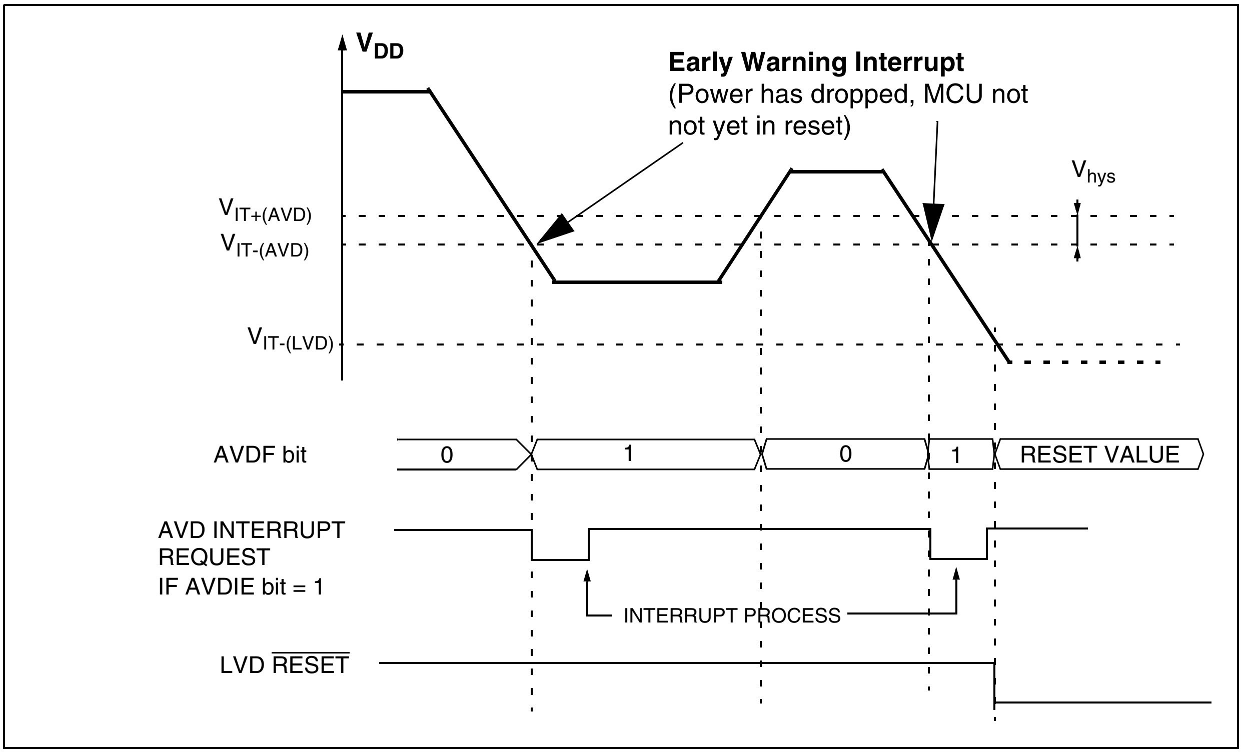
The AVD is used to provide the application with an early warning of a drop in voltage. If enabled, an interrupt can be generated allowing software to shut down safely before the LVD resets the microcontroller. See Figure 19.

The AVD function is active only if the LVD is enabled through the option byte (see Section 15.1 on page 181). The activation level of the AVD is fixed at around 0.5 mV above the selected LVD threshold.

In the case of a drop in voltage below $V_{IT-(PVD)}$, the AVDF flag is set and an interrupt request is issued.

If V_{DD} rises above the $V_{IT+(AVD)}$ threshold voltage the AVDF bit is cleared automatically by hardware. No interrupt is generated, and therefore software should poll the AVDF bit to detect when the voltage has risen, and resume normal processing.

Figure 19. Using the AVD to Monitor V_{DD}



SYSTEM INTEGRITY MANAGEMENT (Cont'd)

7.5.3 Low Power Modes

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

7.5.3.1 Interrupts

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

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

7.5.4 Register Description

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

Read/Write

Reset Value: 000x 000x (xxh)

7							0
0	PDVD IE	AVD F	LVD RF	LOC KED	0	0	WDG RF

Bit 7 = Reserved, must be kept cleared.

Bit 6 = **AVDIE** *Voltage Detector interrupt enable*
This bit is set and cleared by software. It enables an interrupt to be generated when the AVDF flag goes from 0 to 1. The pending interrupt information is automatically cleared when software enters the AVD interrupt routine.
0: PDVD interrupt disabled
1: PDVD interrupt enabled

Bit 5 = **AVDF** *Voltage Detector flag*
This read-only bit is set and cleared by hardware. If the AVDIE bit is set, an interrupt request is generated when the AVDF bit goes from 0 to 1. Refer to Figure 19 and to Section 7.5.2 for additional details.
0: V_{DD} over $V_{IT+(AVD)}$ threshold
1: V_{DD} under $V_{IT-(AVD)}$ threshold

Bit 4 = **LVDRF** *LVD reset flag*
This bit indicates that the last Reset was generated by the LVD block. It is set by hardware (LVD reset) and cleared by software (writing zero). See WDGRF flag description for more details. When the LVD is disabled by OPTION BYTE, the LVDRF bit value is undefined.

Bit 3 = **LOCKED** *PLL Locked Flag*
This bit is set and cleared by hardware. It is set automatically when the PLL reaches its operating frequency.
0: PLL not locked
1: PLL locked

Bits 2:1 = Reserved, must be kept cleared.

Bit 0 = **WDGRF** *Watchdog reset flag*
This bit indicates that the last Reset was generated by the Watchdog peripheral. It is set by hardware (watchdog reset) and cleared by software (writing zero) or an LVD Reset (to ensure a stable cleared state of the WDGRF flag when CPU starts).
Combined with the LVDRF flag information, the flag description is given by the following table.

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

Application notes

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

CAUTION: When the LVD is not activated with the associated option byte, the WDGRF flag can not be used in the application.

8 INTERRUPTS

8.1 INTRODUCTION

The ST7 enhanced interrupt management provides the following features:

- Hardware interrupts
- Software interrupt (TRAP)
- Nested or concurrent interrupt management with flexible interrupt priority and level management:
  - Up to 4 software programmable nesting levels
  - Up to 16 interrupt vectors fixed by hardware
  - 2 non maskable events: RESET, TRAP

This interrupt management is based on:

– Bit 5 and bit 3 of the CPU CC register (I1:0),

– Interrupt software priority registers (ISPRx),

– Fixed interrupt vector addresses located at the high addresses of the memory map (FFE0h to FFFFh) sorted by hardware priority order.

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

8.2 MASKING AND PROCESSING FLOW

The interrupt masking is managed by the I1 and I0 bits of the CC register and the ISPRx registers which give the interrupt software priority level of each interrupt vector (see Table 6). The processing flow is shown in Figure 20

When an interrupt request has to be serviced:

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

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

– I1 and I0 bits of CC register are set according to the corresponding values in the ISPRx registers of the serviced interrupt vector.

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

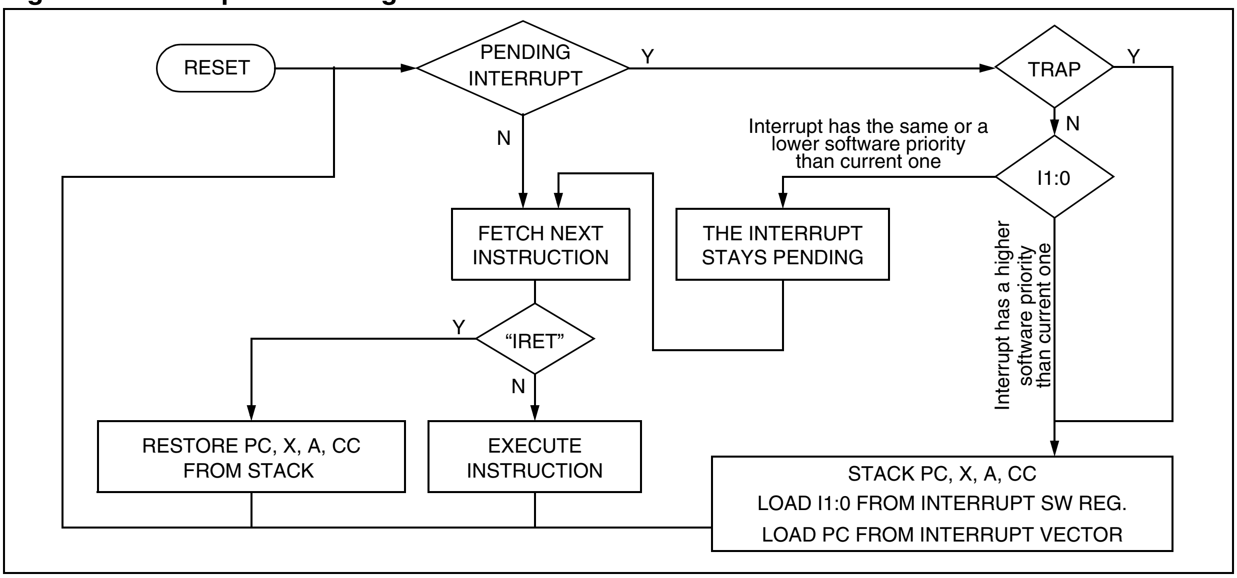
The interrupt service routine should end with the IRET instruction which causes the contents of the saved registers to be recovered from the stack.

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

Table 6. Interrupt Software Priority Levels

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

Figure 20. Interrupt Processing Flowchart



INTERRUPTS (Cont'd)

Servicing Pending Interrupts

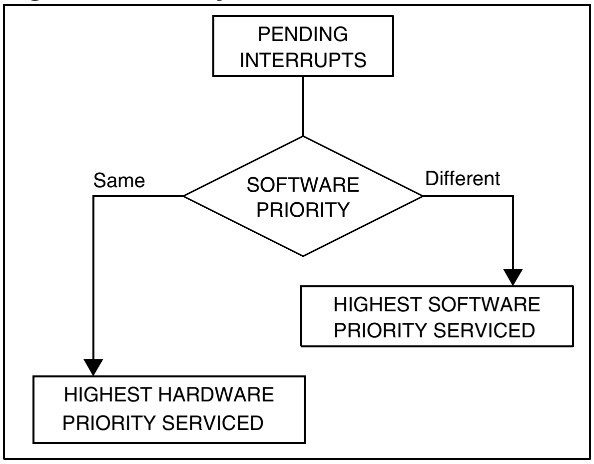
As several interrupts can be pending at the same time, the interrupt to be taken into account is determined by the following two-step process:

– the highest software priority interrupt is serviced,

– if several interrupts have the same software priority then the interrupt with the highest hardware priority is serviced first.

Figure 21 describes this decision process.

Figure 21. Priority Decision Process



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

Notes:

1. The hardware priority is exclusive while the software one is not. This allows the previous process to succeed with only one interrupt.
2. TLI, RESET and TRAP can be considered as having the highest software priority in the decision process.

Different Interrupt Vector Sources

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

Non-Maskable Sources

These sources are processed regardless of the state of the I1 and I0 bits of the CC register (see Figure 20). After stacking the PC, X, A and CC registers (except for RESET), the corresponding vector is loaded in the PC register and the I1 and I0 bits of the CC are set to disable interrupts (level 3). These sources allow the processor to exit HALT mode.

■ TRAP (Non Maskable Software Interrupt)

This software interrupt is serviced when the TRAP instruction is executed. It will be serviced according to the flowchart in Figure 20.

■ RESET

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

Maskable Sources

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

■ External Interrupts

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

■ Peripheral Interrupts

Usually the peripheral interrupts cause the MCU to exit from HALT mode except those mentioned in the "Interrupt Mapping" table. A peripheral interrupt occurs when a specific flag is set in the peripheral status registers and if the corresponding enable bit is set in the peripheral control register.
The general sequence for clearing an interrupt is based on an access to the status register followed by a read or write to an associated register.
Note: The clearing sequence resets the internal latch. A pending interrupt (i.e. waiting for being serviced) will therefore be lost if the clear sequence is executed.

INTERRUPTS (Cont'd)

8.3 INTERRUPTS AND LOW POWER MODES

All interrupts allow the processor to exit the WAIT low power mode. On the contrary, only external and other specified interrupts allow the processor to exit from the HALT modes (see column "Exit from HALT" in "Interrupt Mapping" table). When several pending interrupts are present while exiting HALT mode, the first one serviced can only be an interrupt with exit from HALT mode capability and it is selected through the same decision process shown in Figure 21.

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

8.4 CONCURRENT & NESTED MANAGEMENT

The following Figure 22 and Figure 23 show two different interrupt management modes. The first is called concurrent mode and does not allow an interrupt to be interrupted, unlike the nested mode in Figure 23. The interrupt hardware priority is given in this order from the lowest to the highest: MAIN, IT4, IT3, IT2, IT1, IT0, TLI. The software priority is given for each interrupt.

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

Figure 22. Concurrent Interrupt Management

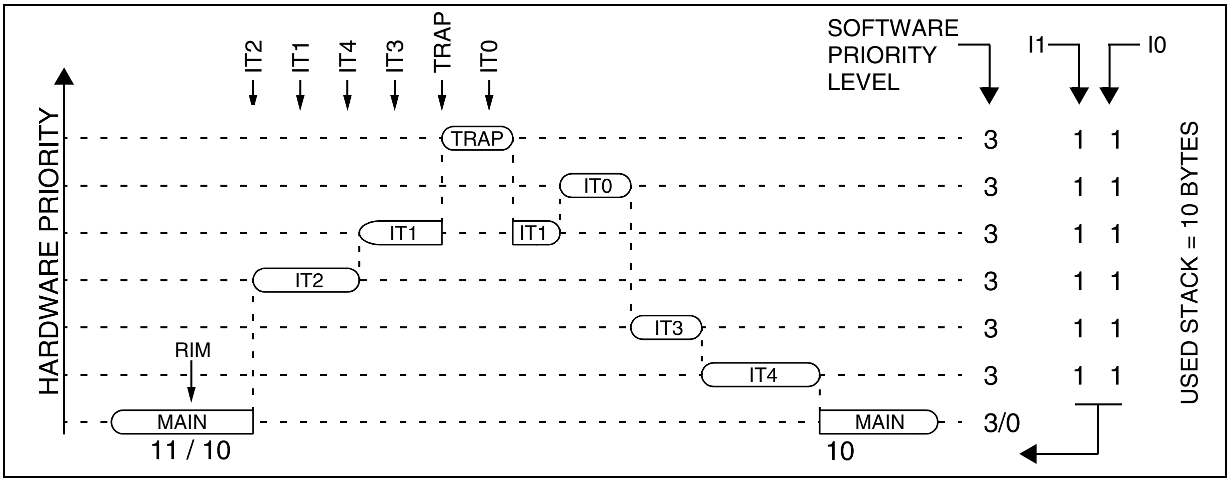
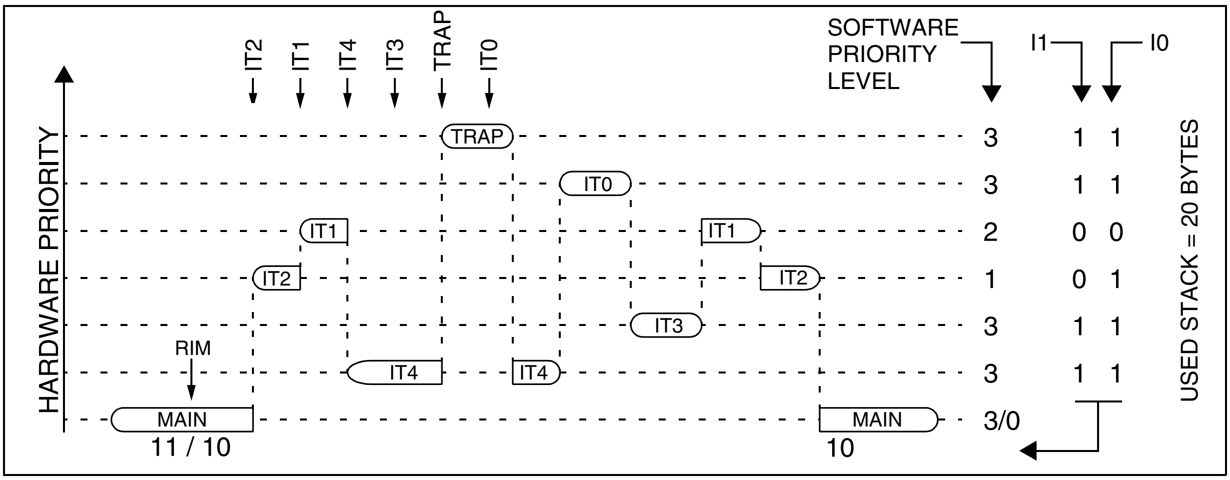


Figure 23. Nested Interrupt Management



INTERRUPTS (Cont'd)

8.5 INTERRUPT REGISTER DESCRIPTION

CPU CC REGISTER INTERRUPT BITS

Read/Write

Reset Value: 111x 1010 (xAh)

7							0
1	1	**I1**	H	**I0**	N	Z	C

Bit 5, 3 = **I1, I0** *Software Interrupt Priority*

These two bits indicate the current interrupt software priority.

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

These two bits are set/cleared by hardware when entering in interrupt. The loaded value is given by the corresponding bits in the interrupt software priority registers (ISPRx).

They can be also set/cleared by software with the RIM, SIM, HALT, WFI, IRET and PUSH/POP instructions (see "Interrupt Dedicated Instruction Set" table).

***Note**: TRAP and RESET events can interrupt a level 3 program.

INTERRUPT SOFTWARE PRIORITY REGISTERS (ISPRX)

Read/Write (bit 7:4 of **ISPR3** are read only)

Reset Value: 1111 1111 (FFh)

	7							0
ISPR0	I1_3	I0_3	I1_2	I0_2	I1_1	I0_1	I1_0	I0_0
ISPR1	I1_7	I0_7	I1_6	I0_6	I1_5	I0_5	I1_4	I0_4
ISPR2	I1_11	I0_11	I1_10	I0_10	I1_9	I0_9	I1_8	I0_8
ISPR3	1	1	1	1	I1_13	I0_13	I1_12	I0_12

These four registers contain the interrupt software priority of each interrupt vector.

– Each interrupt vector (except RESET and TRAP) has corresponding bits in these registers where its own software priority is stored. This correspondence is shown in the following table.

Vector address	ISPRx bits
FFFBh-FFFAh	I1_0 and I0_0 bits*
FFF9h-FFF8h	I1_1 and I0_1 bits
...	...
FFE1h-FFE0h	I1_13 and I0_13 bits

– Each I1_x and I0_x bit value in the ISPRx registers has the same meaning as the I1 and I0 bits in the CC register.

– Level 0 can not be written (I1_x=1, I0_x=0). In this case, the previously stored value is kept. (example: previous=CFh, write=64h, result=44h)

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

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

INTERRUPTS (Cont'd)

Table 7. Dedicated Interrupt Instruction Set

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

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

Table 8. Interrupt Mapping

N°	Source Block	Description	Register Label	Priority Order	Exit from HALT[1]	Address Vector
	RESET	Reset	N/A	Highest Priority	yes	FFFEh-FFFFh
	TRAP/ICD	Software or ICD Interrupt			no	FFFCh-FFFDh
0	AWU	Auto Wake Up Interrupt	AWUCSR		yes	FFFAh-FFFBh
1	MCC/RTC	RTC Time base interrupt	MCCSR		yes	FFF8h-FFF9h
2	ei0	External Interrupt Port PA3, PE1	N/A		yes	FFF6h-FFF7h
3	ei1	External Interrupt Port PF2:0	N/A		yes	FFF4h-FFF5h
4	ei2	External Interrupt Port PB3:0	N/A		yes	FFF2h-FFF3h
5	ei3	External Interrupt Port PB4	N/A		yes	FFF0h-FFF1h
6	I2C3SNS	I2C3SNS Address 3 Interrupt	I2C3SSR		no	FFEEh-FFEFh
7	I2C3SNS	I2C3SNS Address 1 & 2 Interrupt			no	FFECh-FFEDh
8	SPI	SPI Peripheral Interrupts	SPISR		yes[2]	FFEAh-FFEBh
9	TIMER A	TIMER A Peripheral Interrupts	TASR		no	FFE8h-FFE9h
10	TIMER B	TIMER B Peripheral Interrupts	TBSR		no	FFE6h-FFE7h
11	SCI	SCI Peripheral Interrupt	SCISR		no	FFE4h-FFE5h
12	AVD	Auxiliary Voltage Detector Interrupt	SICSR		no	FFE2h-FFE3h
13	I^{2}C	I^{2}C Peripheral Interrupt	I2CSRx	Lowest Priority	no	FFE0h-FFE1h

Notes:

1. Valid for HALT and ACTIVE-HALT modes except for the MCC/RTC interrupt source which exits from ACTIVE-HALT mode only and AWU interrupt which exits from AWUFH mode only.

2. Exit from HALT possible when SPI is in slave mode.

INTERRUPTS (Cont'd)

8.6 EXTERNAL INTERRUPTS

8.6.1 I/O Port Interrupt Sensitivity

The external interrupt sensitivity is controlled by the IPA, IPB and ISxx bits of the EICR register (Figure 24). This control allows to have up to 4 fully independent external interrupt source sensitivities.

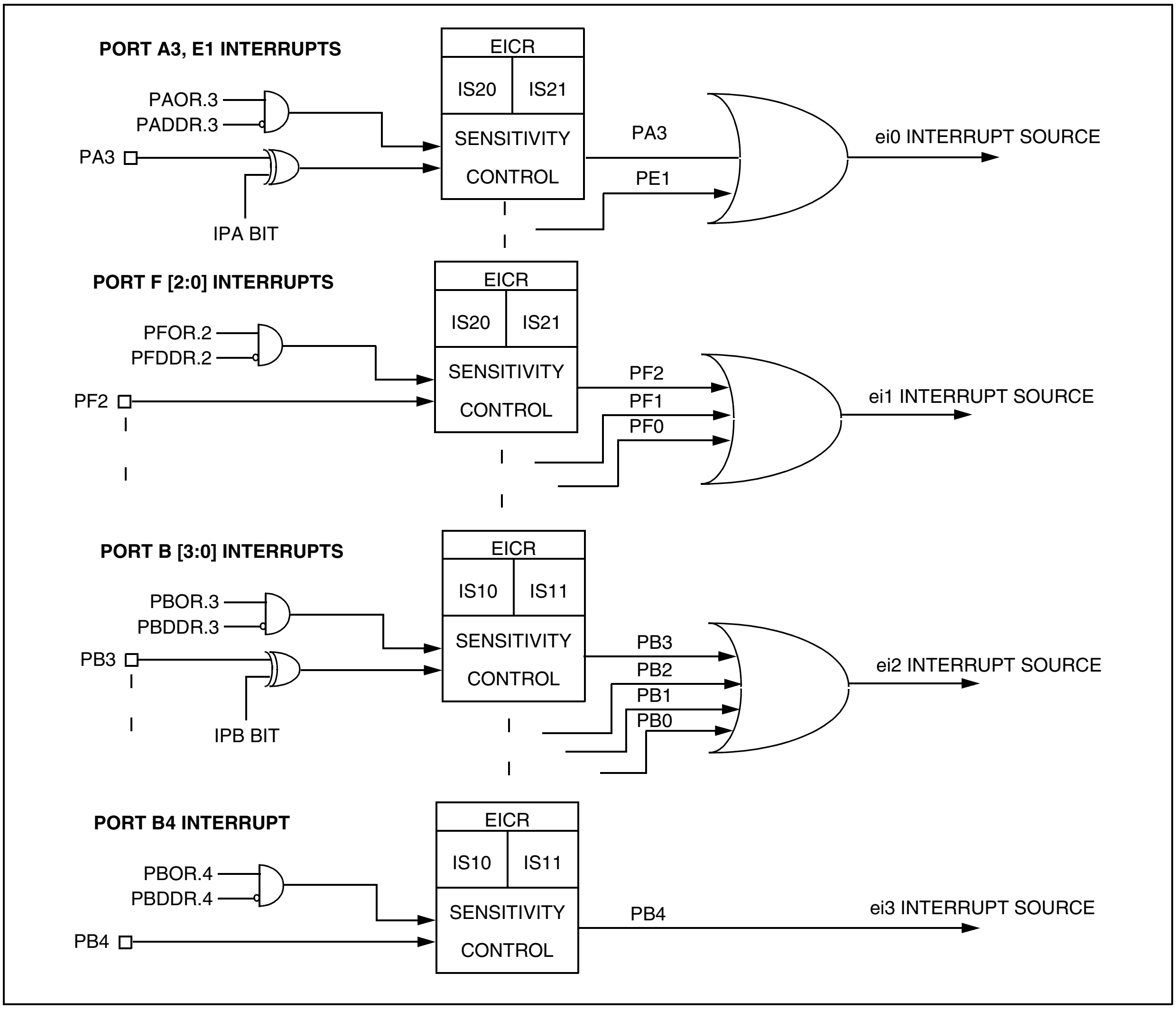
Each external interrupt source can be generated on four (or five) different events on the pin:

- Falling edge
- Rising edge
- Falling and rising edge
- Falling edge and low level
- Rising edge and high level (only for ei0 and ei2)

To guarantee correct functionality, the sensitivity bits in the EICR register can be modified only when the I1 and I0 bits of the CC register are both set to 1 (level 3). This means that interrupts must be disabled before changing sensitivity.

The pending interrupts are cleared by writing a different value in the ISx[1:0], IPA or IPB bits of the EICR.

Figure 24. External Interrupt Control bits



INTERRUPTS (Cont'd)

8.7 EXTERNAL INTERRUPT CONTROL REGISTER (EICR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
IS11	IS10	IPB	IS21	IS20	IPA	0	0

Bit 7:6 = **IS1[1:0]** *ei2 and ei3 sensitivity*
The interrupt sensitivity, defined using the IS1[1:0] bits, is applied to the following external interrupts:
- ei2 (port B3..0)

IS11	IS10	External Interrupt Sensitivity	
		IPB bit =0	**IPB bit =1**
0	0	Falling edge & low level	Rising edge & high level
0	1	Rising edge only	Falling edge only
1	0	Falling edge only	Rising edge only
1	1	Rising and falling edge	

- ei3 (port B4)

IS11	IS10	External Interrupt Sensitivity
0	0	Falling edge & low level
0	1	Rising edge only
1	0	Falling edge only
1	1	Rising and falling edge

These 2 bits can be written only when I1 and I0 of the CC register are both set to 1 (level 3).

Bit 5 = **IPB** *Interrupt polarity for port B*
This bit is used to invert the sensitivity of the port B [3:0] external interrupts. It can be set and cleared by software only when I1 and I0 of the CC register are both set to 1 (level 3).
0: No sensitivity inversion
1: Sensitivity inversion

Bit 4:3 = **IS2[1:0]** *ei0 and ei1 sensitivity*
The interrupt sensitivity, defined using the IS2[1:0] bits, is applied to the following external interrupts:

- ei0 (port A3, port E1)

IS21	IS20	External Interrupt Sensitivity	
		IPA bit =0	**IPA bit =1**
0	0	Falling edge & low level	Rising edge & high level
0	1	Rising edge only	Falling edge only
1	0	Falling edge only	Rising edge only
1	1	Rising and falling edge	

- ei1 (port F2..0)

IS21	IS20	External Interrupt Sensitivity
0	0	Falling edge & low level
0	1	Rising edge only
1	0	Falling edge only
1	1	Rising and falling edge

These 2 bits can be written only when I1 and I0 of the CC register are both set to 1 (level 3).

Bit 2 = **IPA** *Interrupt polarity for ports A3 and E1*
This bit is used to invert the sensitivity of the port A3 and E1 external interrupts. It can be set and cleared by software only when I1 and I0 of the CC register are both set to 1 (level 3).
0: No sensitivity inversion
1: Sensitivity inversion

Bits 1:0 = Reserved, must always be kept cleared.

INTERRUPTS (Cont'd)

Table 9. Nested Interrupts Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0024h		ei1		ei0		MCC + SI		AWU	
	ISPR0	I1_3	I0_3	I1_2	I0_2	I1_1	I0_1	I1_0	I0_0
	Reset Value	1	1	1	1	1	1	1	1
0025h		I2C3SNS		I2C3SNS		ei3		ei2	
	ISPR1	I1_7	I0_7	I1_6	I0_6	I1_5	I0_5	I1_4	I0_4
	Reset Value	1	1	1	1	1	1	1	1
0026h		SCI		TIMER B		TIMER A		SPI	
	ISPR2	I1_11	I0_11	I1_10	I0_10	I1_9	I0_9	I1_8	I0_8
	Reset Value	1	1	1	1	1	1	1	1
0027h						I2C		AVD	
	ISPR3					I1_13	I0_13	I1_12	I0_12
	Reset Value	1	1	1	1	1	1	1	1
0028h	**EICR**	IS11	IS10	IPB	IS21	IS20	IPA		
	Reset Value	0	0	0	0	0	0	0	0

9 POWER SAVING MODES

9.1 INTRODUCTION

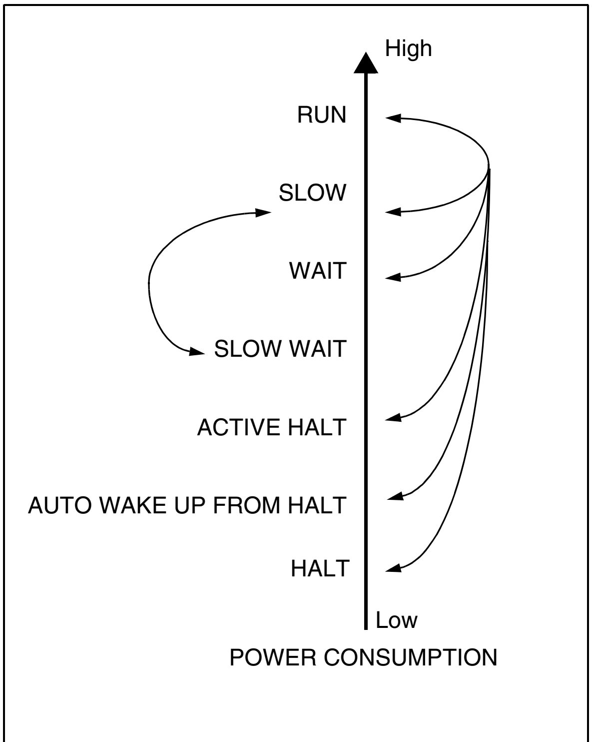
To give a large measure of flexibility to the application in terms of power consumption, five main power saving modes are implemented in the ST7 (see Figure 25):

- Slow
- Wait (and Slow-Wait)
- Active Halt
- Auto Wake up From Halt (AWUFH)
- Halt

After a RESET the normal operating mode is selected by default (RUN mode). This mode drives the device (CPU and embedded peripherals) by means of a master clock which is based on the main oscillator frequency divided or multiplied by 2 (f_{OSC2}).

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

Figure 25. Power Saving Mode Transitions



9.2 SLOW MODE

This mode has two targets:

– To reduce power consumption by decreasing the internal clock in the device,

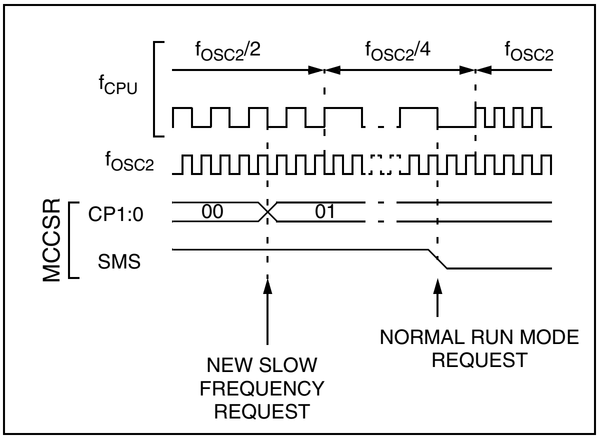
– To adapt the internal clock frequency (f_{CPU}) to the available supply voltage.

SLOW mode is controlled by three bits in the MCCSR register: the SMS bit which enables or disables Slow mode and two CPx bits which select the internal slow frequency (f_{CPU}).

In this mode, the master clock frequency (f_{OSC2}) can be divided by 2, 4, 8 or 16. The CPU and peripherals are clocked at this lower frequency (f_{CPU}).

Note: SLOW-WAIT mode is activated by entering WAIT mode while the device is in SLOW mode.

Figure 26. SLOW Mode Clock Transitions



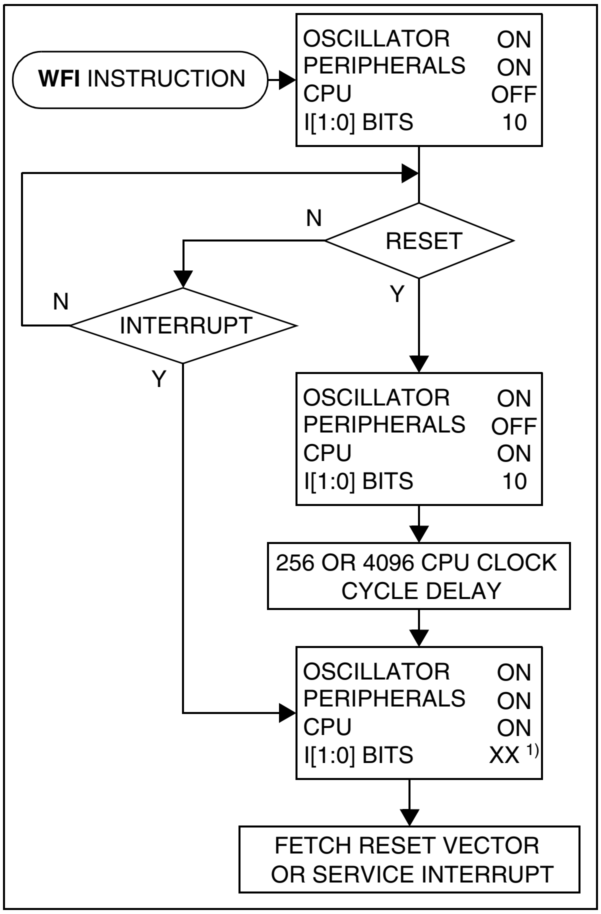
POWER SAVING MODES (Cont'd)

9.3 WAIT MODE

WAIT mode places the MCU in a low power consumption mode by stopping the CPU.
This power saving mode is selected by calling the 'WFI' instruction.
All peripherals remain active. During WAIT mode, the I[1:0] bits of the CC register are forced to '10', to enable all interrupts. All other registers and memory remain unchanged. The MCU remains in WAIT mode until an interrupt or RESET occurs, whereupon the Program Counter branches to the starting address of the interrupt or Reset service routine.
The MCU will remain in WAIT mode until a Reset or an Interrupt occurs, causing it to wake up.

Refer to Figure 27.

Figure 27. WAIT Mode Flow-chart



Note:

1. Before servicing an interrupt, the CC register is pushed on the stack. The I[1:0] bits of the CC register are set to the current software priority level of the interrupt routine and recovered when the CC register is popped.

POWER SAVING MODES (Cont'd)

9.4 HALT MODE

The HALT mode is the lowest power consumption mode of the MCU. It is entered by executing the 'HALT' instruction when the OIE bit of the Main Clock Controller Status register (MCCSR) is cleared (see Section 11.2 on page 65 for more details on the MCCSR register) and when the AWUEN bit in the AWUCSR register is cleared.

The MCU can exit HALT mode on reception of either a specific interrupt (see Table 8, "Interrupt Mapping," on page 40) or a RESET. When exiting HALT mode by means of a RESET or an interrupt, the oscillator is immediately turned on and the 256 or 4096 CPU cycle delay is used to stabilize the oscillator. After the start up delay, the CPU resumes operation by servicing the interrupt or by fetching the reset vector which woke it up (see Figure 29).

When entering HALT mode, the I[1:0] bits in the CC register are forced to '10b'to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes up immediately.

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

The compatibility of Watchdog operation with HALT mode is configured by the "WDGHALT" option bit of the option byte. The HALT instruction when executed while the Watchdog system is enabled, can generate a Watchdog RESET (see Section 11.1 on page 58 for more details).

Figure 28. HALT Timing Overview

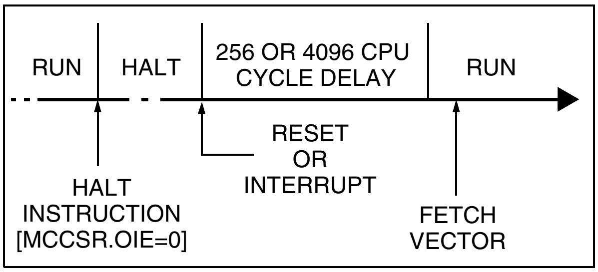
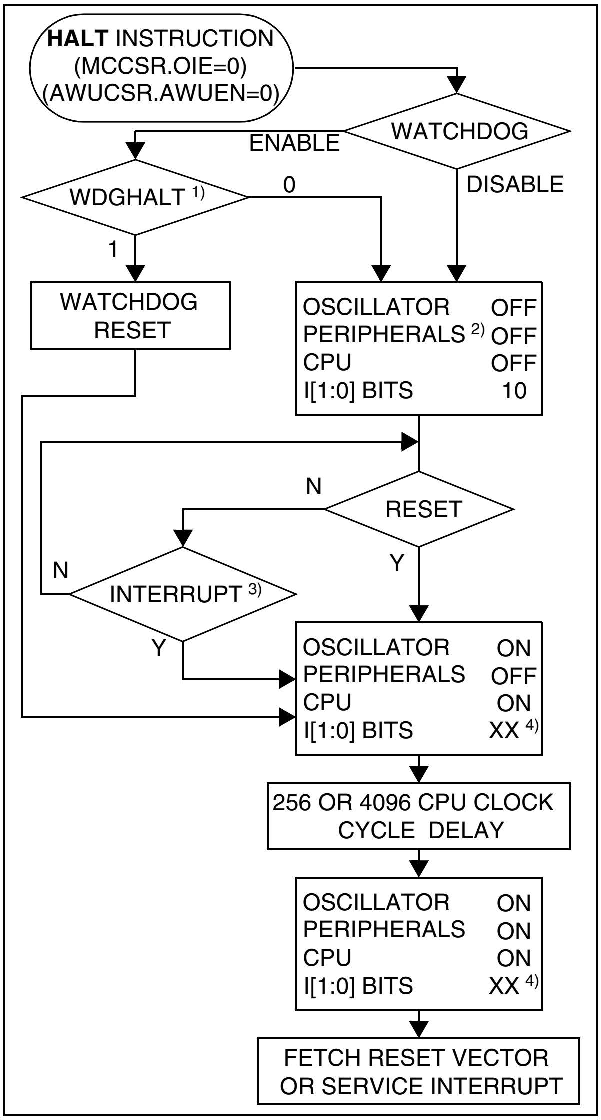


Figure 29. HALT Mode Flow-chart



Notes:

1. WDGHALT is an option bit. See option byte section for more details.

2. Peripheral clocked with an external clock source can still be active.

3. Only some specific interrupts can exit the MCU from HALT mode (such as external interrupt). Refer to Table 8, "Interrupt Mapping," on page 40 for more details.

4. Before servicing an interrupt, the CC register is pushed on the stack. The I[1:0] bits of the CC register are set to the current software priority level of the interrupt routine and recovered when the CC register is popped.

POWER SAVING MODES (Cont'd)

Halt Mode Recommendations

– Make sure that an external event is available to wake up the microcontroller from Halt mode.

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

– For the same reason, reinitialize the level sensitiveness of each external interrupt as a precautionary measure.

– The opcode for the HALT instruction is 0x8E. To avoid an unexpected HALT instruction due to a program counter failure, it is advised to clear all occurrences of the data value 0x8E from memory. For example, avoid defining a constant in ROM with the value 0x8E.

– As the HALT instruction clears the interrupt mask in the CC register to allow interrupts, the user may choose to clear all pending interrupt bits before executing the HALT instruction. This avoids entering other peripheral interrupt routines after executing the external interrupt routine corresponding to the wake-up event (reset or external interrupt).

9.5 ACTIVE-HALT MODE

ACTIVE-HALT mode is the lowest power consumption mode of the MCU with a real time clock available. It is entered by executing the 'HALT' instruction when MCC/RTC interrupt enable flag (OIE bit in MCCSR register) is set and when the AWUEN bit in the AWUCSR register is cleared (See "Register Description" on page 51.)

MCCSR OIE bit	Power Saving Mode entered when HALT instruction is executed
0	HALT mode
1	ACTIVE-HALT mode

The MCU can exit ACTIVE-HALT mode on reception of the RTC interrupt and some specific interrupts (see Table 8, "Interrupt Mapping," on page 40) or a RESET. When exiting ACTIVE-HALT mode by means of a RESET a 4096 or 256 CPU cycle delay occurs (depending on the option byte). After the start up delay, the CPU resumes operation by servicing the interrupt or by fetching the reset vector which woke it up (see Figure 31).

When entering ACTIVE-HALT mode, the I[1:0] bits in the CC register are cleared to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes up immediately.

In ACTIVE-HALT mode, only the main oscillator and its associated counter (MCC/RTC) are running to keep a wake-up time base. All other peripherals are not clocked except those which get their clock supply from another clock generator (such as external or auxiliary oscillator).

The safeguard against staying locked in ACTIVE-HALT mode is provided by the oscillator interrupt.

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

POWER SAVING MODES (Cont'd)

Figure 30. ACTIVE-HALT Timing Overview

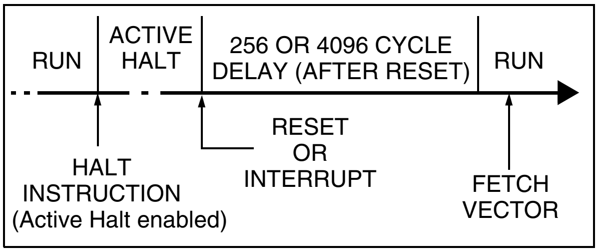
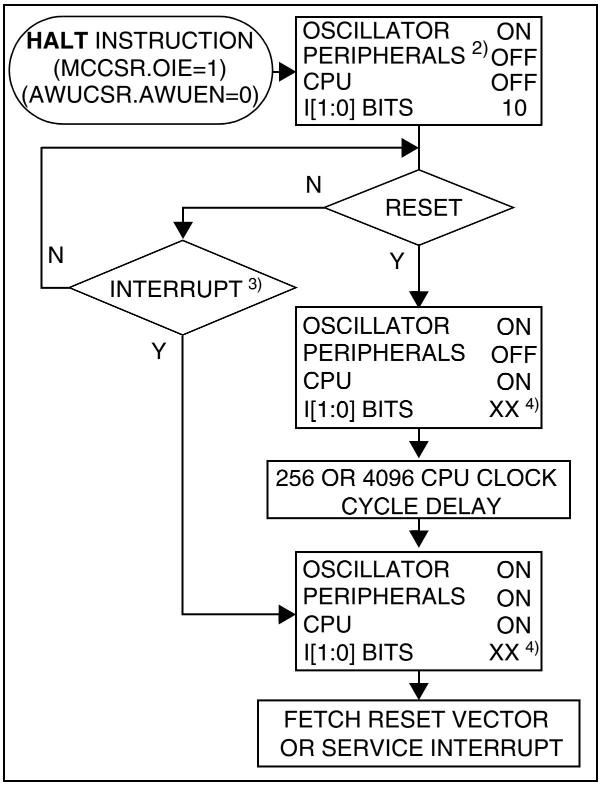


Figure 31. ACTIVE-HALT Mode Flow-chart



Notes:

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

2. Peripheral clocked with an external clock source can still be active.

3. Only the RTC interrupt and some specific interrupts can exit the MCU from ACTIVE-HALT mode (such as external interrupt). Refer to Table 8, "Interrupt Mapping," on page 40 for more details.

4. Before servicing an interrupt, the CC register is pushed on the stack. The I[1:0] bits in the CC register are set to the current software priority level of the interrupt routine and restored when the CC register is popped.

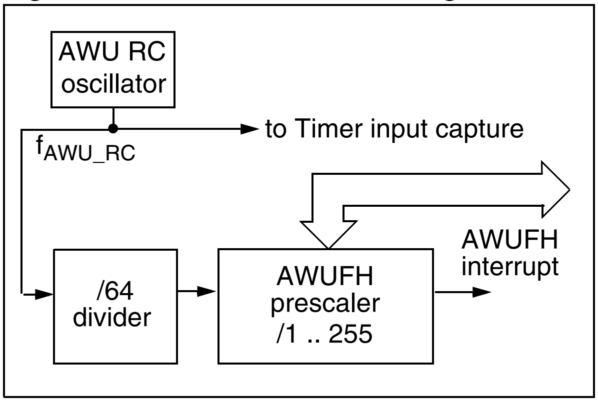
POWER SAVING MODES (Cont'd)

9.6 AUTO WAKE UP FROM HALT MODE

Auto Wake Up From Halt (AWUFH) mode is similar to Halt mode with the addition of an internal RC oscillator for wake-up. Compared to ACTIVE-HALT mode, AWUFH has lower power consumption because the main clock is not kept running, but there is no accurate realtime clock available.

It is entered by executing the HALT instruction when the AWUEN bit in the AWUCSR register has been set and the OIE bit in the MCCSR register is cleared (see Section 11.2 on page 65 for more details).

Figure 32. AWUFH Mode Block Diagram



As soon as HALT mode is entered, and if the AWUEN bit has been set in the AWUCSR register, the AWU RC oscillator provides a clock signal (f_{AWU_RC}). Its frequency is divided by a fixed divider and a programmable prescaler controlled by the AWUPR register. The output of this prescaler provides the delay time. When the delay has elapsed the AWUF flag is set by hardware and an interrupt wakes-up the MCU from Halt mode. At the same time the main oscillator is immediately turned on and a 256 or 4096 cycle delay is used to stabilize it. After this start-up delay, the CPU resumes operation by servicing the AWUFH interrupt. The AWU flag and its associated interrupt are cleared by software reading the AWUCSR register.

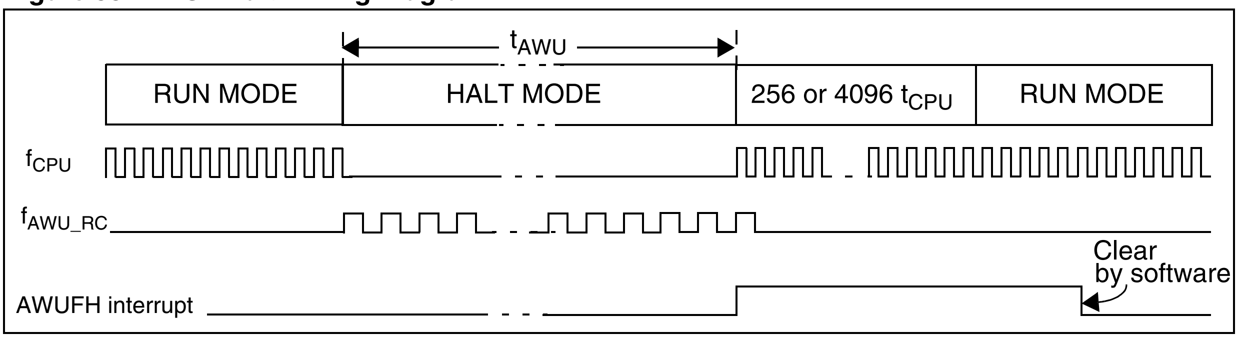
To compensate for any frequency dispersion of the AWU RC oscillator, it can be calibrated by measuring the clock frequency f_{AWU_RC} and then calculating the right prescaler value. Measurement mode is enabled by setting the AWUM bit in the AWUCSR register in Run mode. This connects internally f_{AWU_RC} to the ICAP2 input of the 16-bit timer A, allowing the f_{AWU_RC} to be measured using the main oscillator clock as a reference timebase.

Similarities with Halt mode

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

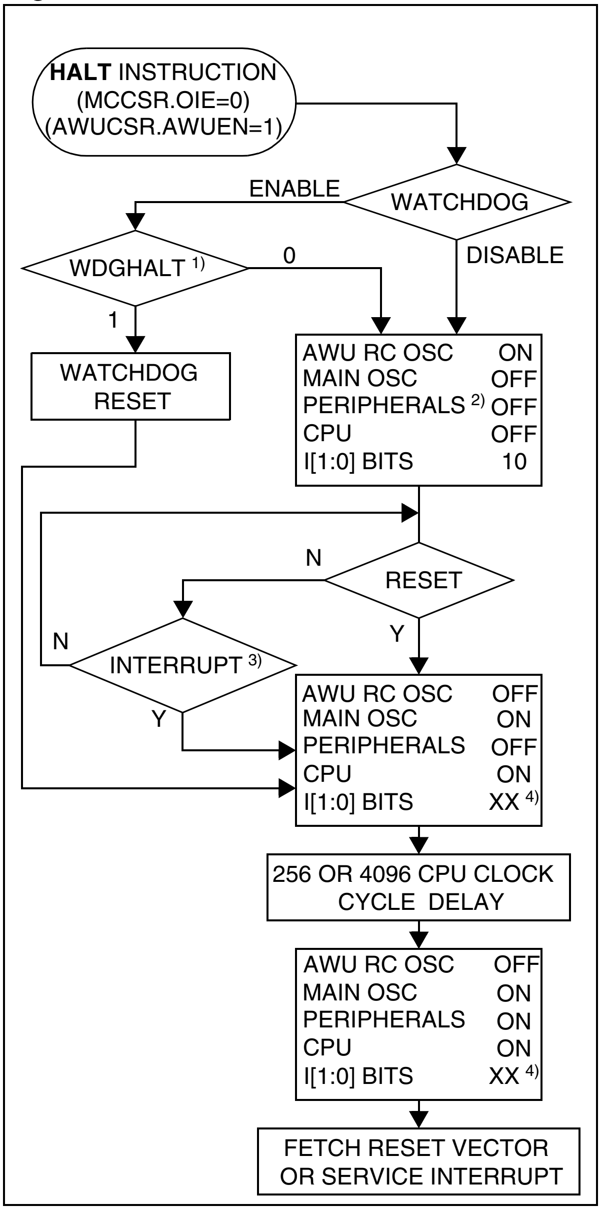
- The MCU can exit AWUFH mode by means of any interrupt with exit from Halt capability or a reset (see Section 9.4 "HALT MODE" on page 46).
- When entering AWUFH mode, the I[1:0] bits in the CC register are forced to 10b to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes up immediately.
- In AWUFH mode, the main oscillator is turned off causing all internal processing to be stopped, including the operation of the on-chip peripherals. None of the peripherals are clocked except those which get their clock supply from another clock generator (such as an external or auxiliary oscillator like the AWU oscillator).
- The compatibility of Watchdog operation with AWUFH mode is configured by the WDGHALT option bit in the option byte. Depending on this setting, the HALT instruction when executed while the Watchdog system is enabled, can generate a Watchdog RESET.

Figure 33. AWUF Halt Timing Diagram



POWER SAVING MODES (Cont'd)

Figure 34. AWUFH Mode Flow-chart



Notes:

1. WDGHALT is an option bit. See option byte section for more details.

2. Peripheral clocked with an external clock source can still be active.

3. Only an AWUFH interrupt and some specific interrupts can exit the MCU from HALT mode (such as external interrupt). Refer to Table 8, "Interrupt Mapping," on page 40 for more details.

4. Before servicing an interrupt, the CC register is pushed on the stack. The I[1:0] bits of the CC register are set to the current software priority level of the interrupt routine and recovered when the CC register is popped.

POWER SAVING MODES (Cont'd)

9.6.0.1 Register Description

AWUFH CONTROL/STATUS REGISTER (AWUCSR)
Read/Write (except bit 2 read only)
Reset Value: 0000 0000 (00h)

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

Bits 7:3 = Reserved.

Bit 2= **AWUF** *Auto Wake Up Flag*
This bit is set by hardware when the AWU module generates an interrupt and cleared by software on reading AWUCSR.
0: No AWU interrupt occurred
1: AWU interrupt occurred

Bit 1= **AWUM** *Auto Wake Up Measurement*
This bit enables the AWU RC oscillator and connects internally its output to the ICAP2 input of 16-bit timer A. This allows the timer to be used to measure the AWU RC oscillator dispersion and then compensate this dispersion by providing the right value in the AWUPR register.
0: Measurement disabled
1: Measurement enabled

Bit 0 = **AWUEN** *Auto Wake Up From Halt Enabled*
This bit enables the Auto Wake Up From Halt feature: once HALT mode is entered, the AWUFH wakes up the microcontroller after a time delay defined by the AWU prescaler value. It is set and cleared by software.
0: AWUFH (Auto Wake Up From Halt) mode disabled
1: AWUFH (Auto Wake Up From Halt) mode enabled

AWUFH PRESCALER REGISTER (AWUPR)
Read/Write
Reset Value: 1111 1111 (FFh)

7							0
AWU PR7	AWU PR6	AWU PR5	AWU PR4	AWU PR3	AWU PR2	AWU PR1	AWU PR0

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

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

In AWU mode, the period that the MCU stays in Halt Mode (t_{AWU} in Figure 33) is defined by

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

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

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

Table 10. AWU Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
002Eh	**AWUCSR** Reset Value	0	0	0	0	0	AWUF 0	AWUM 0	AWUEN 0
002Fh	**AWUPR** Reset Value	AWUPR7 1	AWUPR6 1	AWUPR5 1	AWUPR4 1	AWUPR3 1	AWUPR2 1	AWUPR1 1	AWUPR0 1

10 I/O PORTS

10.1 INTRODUCTION

The I/O ports offer different functional modes:
– transfer of data through digital inputs and outputs

and for specific pins:
– external interrupt generation
– alternate signal input/output for the on-chip peripherals.

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

10.2 FUNCTIONAL DESCRIPTION

Each port has two main registers:

– Data Register (DR)

– Data Direction Register (DDR)

and one optional register:

– Option Register (OR)

Each I/O pin may be programmed using the corresponding register bits in the 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). The generic I/O block diagram is shown in Figure 1

10.2.1 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. Writing the DR register modifies the latch value but does not affect the pin status.
2. When switching from input to output mode, the DR register has to be written first to drive the correct level on the pin as soon as the port is configured as an output.
3. Do not use read/modify/write instructions (BSET or BRES) to modify the DR register as this might corrupt the DR content for I/Os configured as input.

External interrupt function

When an I/O is configured as Input with Interrupt, an event on this I/O can generate an external interrupt request to the CPU.

Each pin can independently generate an interrupt request. The interrupt sensitivity is independently programmable using the sensitivity bits in the EICR register.

Each external interrupt vector is linked to a dedicated group of I/O port pins (see pinout description and interrupt section). If several input pins are selected simultaneously as interrupt sources, these are first detected according to the sensitivity bits in the EICR register and then logically ORed.

The external interrupts are hardware interrupts, which means that the request latch (not accessible directly by the application) is automatically cleared when the corresponding interrupt vector is fetched. To clear an unwanted pending interrupt by software, the sensitivity bits in the EICR register must be modified.

10.2.2 Output Modes

The output configuration is selected by setting the corresponding DDR register bit. In this case, writing 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.

Two different output modes can be selected by software through the OR register: Output push-pull and open-drain.

DR register value and output pin status:

DR	Push-pull	Open-drain
0	V_{SS}	Vss
1	V_{DD}	Floating

10.2.3 Alternate Functions

When an on-chip peripheral is configured to use a pin, the alternate function is automatically selected. This alternate function takes priority over the 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 must be configured in input mode. In this case, the pin state is also digitally readable by addressing the DR register.

Note: Input pull-up configuration can cause unexpected value at the input of the alternate peripheral input. When an on-chip peripheral use a pin as input and output, this pin has to be configured in input floating mode.

I/O PORTS (Cont'd)

Figure 35. I/O Port General Block Diagram

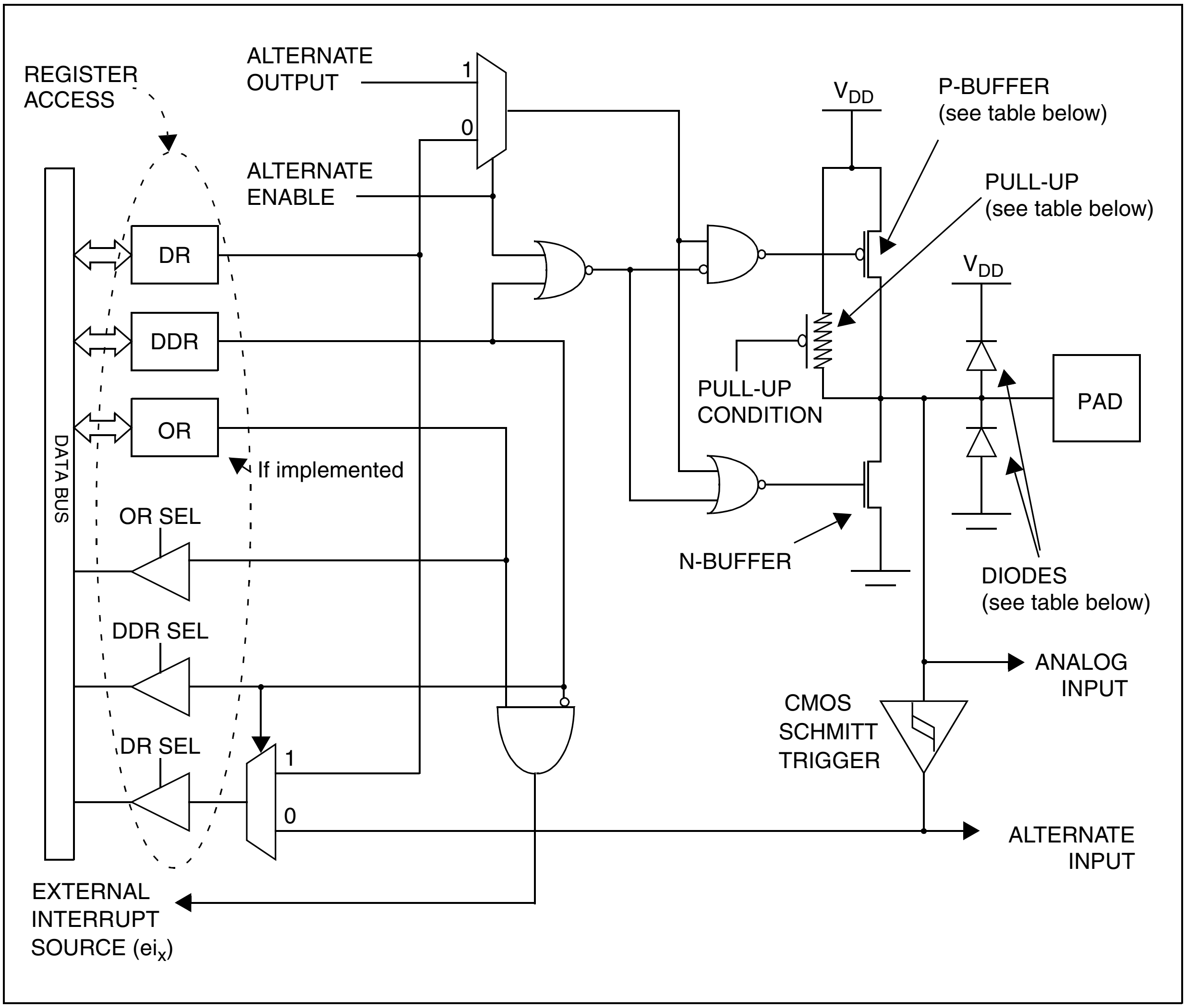


Table 11. I/O Port Mode Options

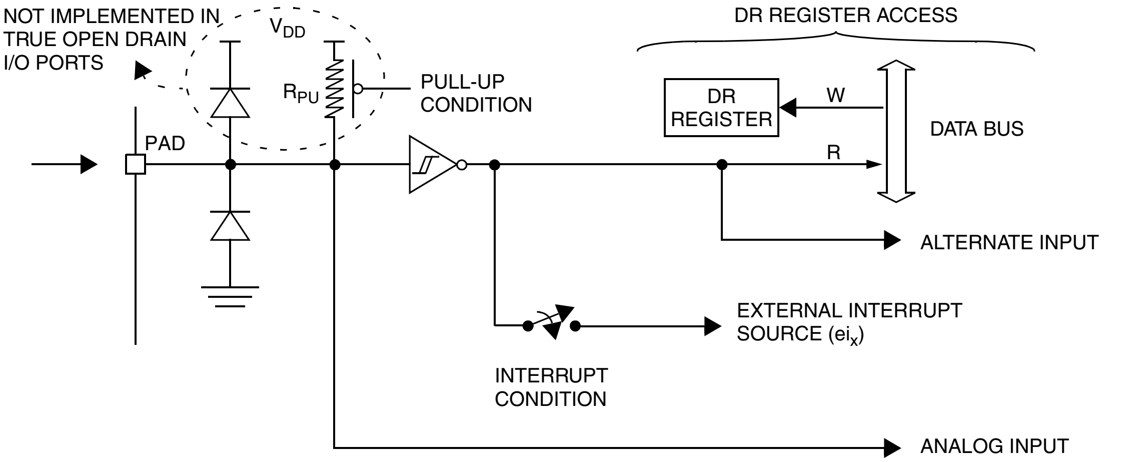
Configuration Mode		Pull-Up	P-Buffer	Diodes	
				to V_{DD}	to V_{SS}
Input	Floating with/without Interrupt	Off	Off	On	On
	Pull-up with/without Interrupt	On			
Output	Push-pull	Off	On		
	Open Drain (logic level)		Off		
	True Open Drain	NI	NI	NI (see note)	

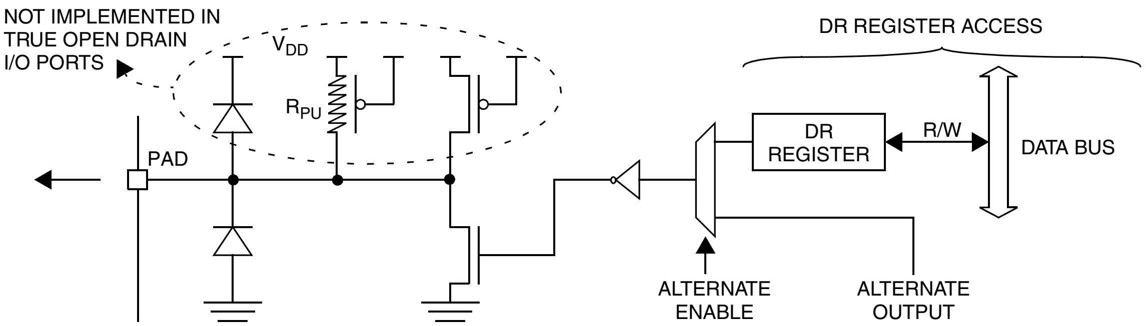
Legend: NI - not implemented
Off - implemented not activated
On - implemented and activated

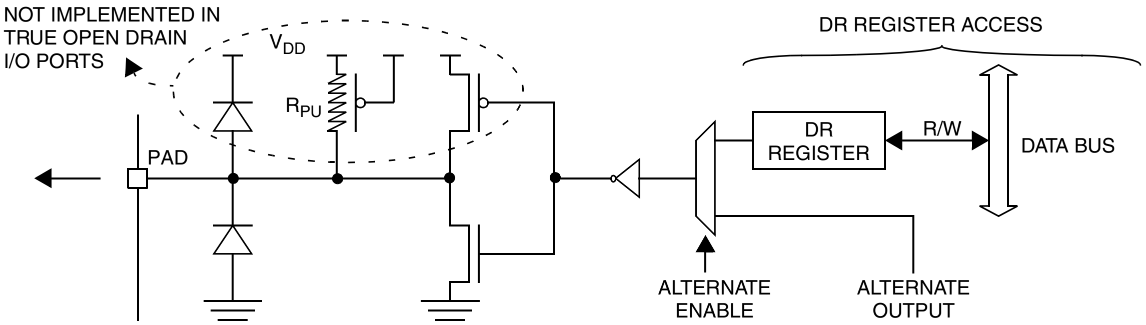
Note: The diode to V_{DD} is not implemented in the true open drain pads. A local protection between the pad and V_{SS} is implemented to protect the device against positive stress.

I/O PORTS (Cont'd)

Table 12. I/O Port Configurations

<table>
<tr><th></th><th>Hardware Configuration</th></tr>
<tr><td>INPUT 1)</td><td>


</td></tr>
<tr><td>OPEN-DRAIN OUTPUT 2)</td><td>


</td></tr>
<tr><td>PUSH-PULL OUTPUT 2)</td><td>


</td></tr>
</table>

Notes:

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

I/O PORTS (Cont'd)

CAUTION: 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.

Analog alternate function

When the pin is used as an ADC input, the I/O must be configured as floating input. 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.

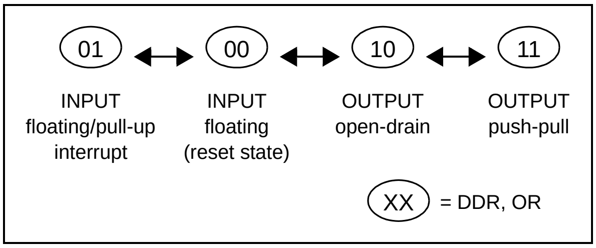
WARNING: The analog input voltage level must be within the limits stated in the absolute maximum ratings.

10.3 I/O PORT IMPLEMENTATION

The hardware implementation on each I/O port depends on the settings in the DDR and OR registers and specific feature of the I/O port such as ADC Input 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 2 on page 4. Other transitions are potentially risky and should be avoided, since they are likely to present unwanted side-effects such as spurious interrupt generation.

Figure 36. Interrupt I/O Port State Transitions



10.4 LOW POWER MODES

Mode	Description
WAIT	No effect on I/O ports. External interrupts cause the device to exit from WAIT mode.
HALT	No effect on I/O ports. External interrupts cause the device to exit from HALT mode.

10.5 INTERRUPTS

The external interrupt event generates an interrupt if the corresponding configuration is selected with DDR and OR registers and the interrupt mask in the CC register is not active (RIM instruction).

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

I/O PORTS (Cont'd)

10.5.1 I/O port implementation

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

Standard ports

PA5:4, PC7:0, PD5:0, PE0, PF7:6, 4

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

Interrupt ports

PB4, PB2:0, PF1:0 (with pull-up)

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

PA3, PE1, PB3, PF2 (without pull-up)

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

True open drain ports

PA7:6 , **PD7:6**

MODE	DDR
floating input	0
open drain (high sink ports)	1

Table 13. Port configuration

Port	Pin name	Input		Output	
		OR = 0	OR = 1	OR = 0	OR = 1
Port A	PA7:6	floating		true open-drain	
	PA5:4	floating	pull-up	open drain	push-pull
	PA3	floating	floating interrupt	open drain	push-pull
Port B	PB3	floating	floating interrupt	open drain	push-pull
	PB4, PB2:0	floating	pull-up interrupt	open drain	push-pull
Port C	PC7:0	floating	pull-up	open drain	push-pull
Port D	PD7:6	floating		true open-drain	
	PD5:0	floating	pull-up	open drain	push-pull
Port E	PE1	floating	floating interrupt	open drain	push-pull
	PE0	floating	pull-up	open drain	push-pull
Port F	PF7:6, 4	floating	pull-up	open drain	push-pull
	PF2	floating	floating interrupt	open drain	push-pull
	PF1:0	floating	pull-up interrupt	open drain	push-pull

CAUTION: In small packages, an internal pull-up is applied permanently to the non-bonded I/O pins. So they have to be kept in input floating configuration to avoid unwanted consumption.

I/O PORTS (Cont'd)

Table 14. I/O port register map and reset values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
Reset Value of all I/O port registers		0	0	0	0	0	0	0	0
0000h	**PADR**	MSB							LSB
0001h	**PADDR**								
0002h	**PAOR**								
0003h	**PBDR**	MSB							LSB
0004h	**PBDDR**								
0005h	**PBOR**								
0006h	**PCDR**	MSB							LSB
0007h	**PCDDR**								
0008h	**PCOR**								
0009h	**PDDR**	MSB							LSB
000Ah	**PDDDR**								
000Bh	**PDOR**								
000Ch	**PEDR**	MSB							LSB
000Dh	**PEDDR**								
000Eh	**PEOR**								
000Fh	**PFDR**	MSB							LSB
0010h	**PFDDR**								
0011h	**PFOR**								

11 ON-CHIP PERIPHERALS

11.1 WINDOW WATCHDOG (WWDG)

11.1.1 Introduction

The Window Watchdog 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 contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

11.1.2 Main Features

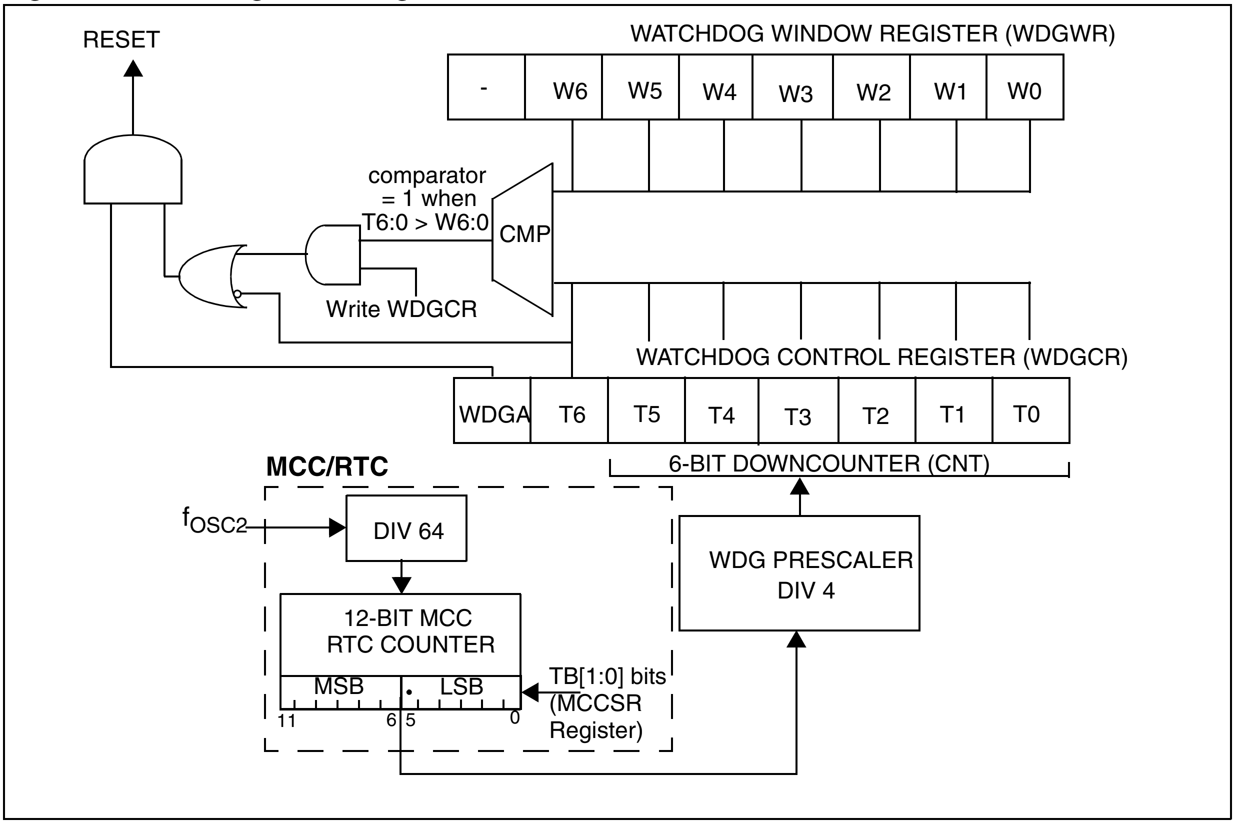
- Programmable free-running downcounter
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes less than 40h
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window (see Figure 4)
- Hardware/Software Watchdog activation (selectable by option byte)
- Optional reset on HALT instruction (configurable by option byte)

11.1.3 Functional Description

The counter value stored in the WDGCR register (bits T[6:0]), is decremented every 16384 f_{OSC2} cycles (approx.), 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 downcounter (T[6:0] bits) rolls over from 40h to 3Fh (T6 becomes cleared), it initiates a reset cycle pulling low the reset pin for typically 30µs. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

Figure 37. Watchdog Block Diagram



WINDOW WATCHDOG (Cont'd)

The application program must write in the WDGCR register at regular intervals during normal operation to prevent an MCU reset. This operation must occur only when the counter value is lower than the window register value. The value to be stored in the WDGCR register must be between FFh and C0h (see Figure 2):

– Enabling the watchdog:
  When Software Watchdog is selected (by option byte), the watchdog is disabled after a reset. It is enabled by setting the WDGA bit in the WDGCR register, then it cannot be disabled again except by a reset.

  When Hardware Watchdog is selected (by option byte), the watchdog is always active and the WDGA bit is not used.

– Controlling the downcounter:
  This downcounter is free-running: It counts down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.
  The T[5:0] bits contain the number of increments which represents the time delay before the watchdog produces a reset (see Figure 2. Approximate Timeout Duration). The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WDGCR register (see Figure 3).

  The window register (WDGWR) contains the high limit of the window: To prevent a reset, the downcounter must be reloaded when its value is lower than the window register value and greater than 3Fh. Figure 4 describes the window watchdog process.

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

– Watchdog Reset on Halt option
  If the watchdog is activated and the watchdog reset on halt option is selected, then the HALT instruction will generate a Reset.

11.1.4 Using Halt Mode with the WDG

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

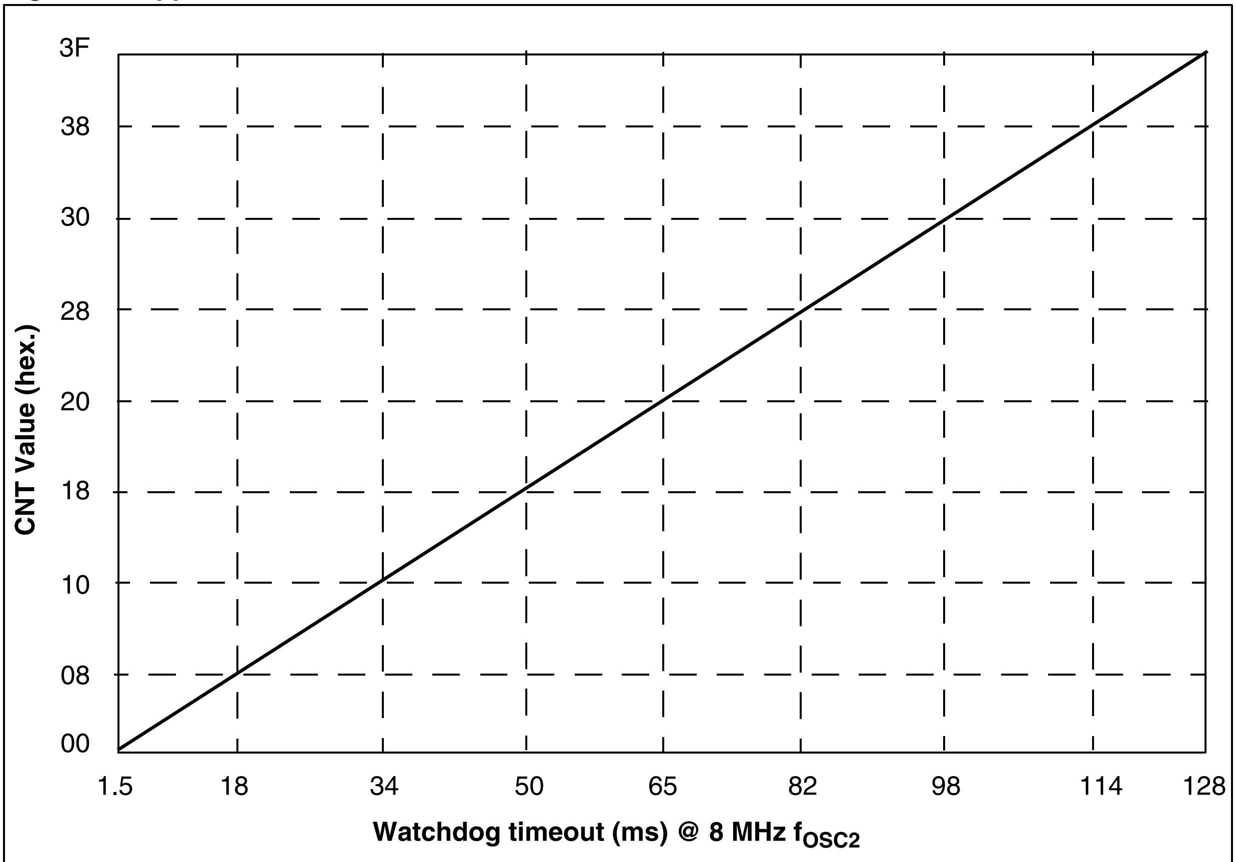
WINDOW WATCHDOG (Cont'd)

11.1.5 How to Program the Watchdog Timeout

Figure 2 shows the linear relationship between the 6-bit value to be loaded in the Watchdog Counter (CNT) and the resulting timeout duration in milliseconds. This can be used for a quick calculation without taking the timing variations into account. If more precision is needed, use the formulae in Figure 3.

Caution: When writing to the WDGCR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 38. Approximate Timeout Duration



WINDOW WATCHDOG (Cont'd)

Figure 39. Exact Timeout Duration (t_{min} and t_{max})

WHERE:

t_{min0} = (LSB + 128) x 64 x t_{OSC2}

t_{max0} = 16384 x t_{OSC2}

t_{OSC2} = 125ns if f_{OSC2} = 8 MHz

CNT = Value of T[5:0] bits in the WDGCR register (6 bits)

MSB and LSB are values from the table below depending on the timebase selected by the TB[1:0] bits in the MCCSR register

TB1 Bit (MCCSR Reg.)	TB0 Bit (MCCSR Reg.)	Selected MCCSR Timebase	MSB	LSB
0	0	2ms	4	59
0	1	4ms	8	53
1	0	10ms	20	35
1	1	25ms	49	54

To calculate the minimum Watchdog Timeout (t_{min}):

IF $CNT < \left[\frac{MSB}{4}\right]$ **THEN** $t_{min} = t_{min0} + 16384 \times CNT \times t_{osc2}$

ELSE $t_{min} = t_{min0} + \left[16384 \times \left(CNT - \left[\frac{4CNT}{MSB}\right]\right) + (192 + LSB) \times 64 \times \left[\frac{4CNT}{MSB}\right]\right] \times t_{osc2}$

To calculate the maximum Watchdog Timeout (t_{max}):

IF $CNT \le \left[\frac{MSB}{4}\right]$ **THEN** $t_{max} = t_{max0} + 16384 \times CNT \times t_{osc2}$

ELSE $t_{max} = t_{max0} + \left[16384 \times \left(CNT - \left[\frac{4CNT}{MSB}\right]\right) + (192 + LSB) \times 64 \times \left[\frac{4CNT}{MSB}\right]\right] \times t_{osc2}$

Note: In the above formulae, division results must be rounded down to the next integer value.

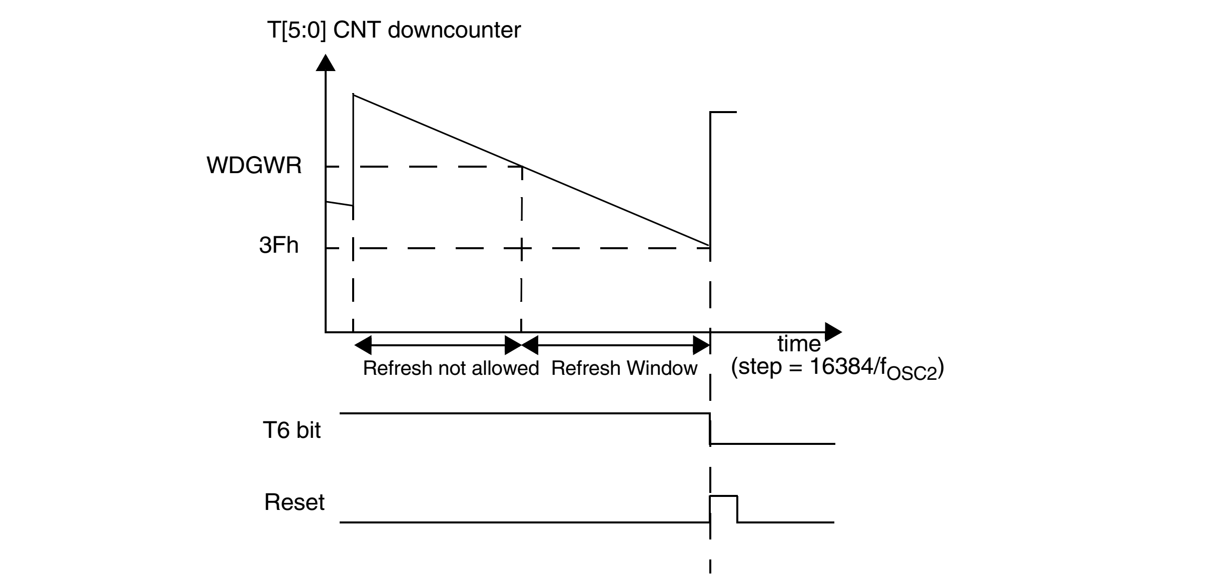
Example:

With 2ms timeout selected in MCCSR register

Value of T[5:0] Bits in WDGCR Register (Hex.)	Min. Watchdog Timeout (ms) t_{min}	Max. Watchdog Timeout (ms) t_{max}
00	1.496	2.048
3F	128	128.552

WINDOW WATCHDOG (Cont'd)

Figure 40. Window Watchdog Timing Diagram



11.1.6 Low Power Modes

<table>
<tr><th>Mode</th><th colspan="3">Description</th></tr>
<tr><td>SLOW</td><td colspan="3">No effect on Watchdog: The downcounter continues to decrement at normal speed.</td></tr>
<tr><td>WAIT</td><td colspan="3">No effect on Watchdog: The downcounter continues to decrement.</td></tr>
<tr><td rowspan="3">HALT</td><td>OIE bit in MCCSR register</td><td>WDGHALT bit in Option Byte</td><td></td></tr>
<tr><td>0</td><td>0</td><td>No Watchdog reset is generated. The MCU enters Halt mode. The Watchdog counter is decremented once and then stops counting and is no longer able to generate a watchdog reset until the MCU receives an external interrupt or a reset.
If an interrupt is received (refer to interrupt table mapping to see interrupts which can occur in halt mode), the Watchdog restarts counting after 256 or 4096 CPU clocks. If a reset is generated, the Watchdog is disabled (reset state) unless Hardware Watchdog is selected by option byte. For application recommendations see Section 0.1.8 below.</td></tr>
<tr><td>0</td><td>1</td><td>A reset is generated instead of entering halt mode.</td></tr>
<tr><td>ACTIVE HALT</td><td>1</td><td>x</td><td>No reset is generated. The MCU enters Active Halt mode. The Watchdog counter is not decremented. It stop counting. When the MCU receives an oscillator interrupt or external interrupt, the Watchdog restarts counting immediately. When the MCU receives a reset the Watchdog restarts counting after 256 or 4096 CPU clocks.</td></tr>
</table>

11.1.7 Hardware Watchdog Option

If Hardware Watchdog is selected by option byte, the watchdog is always active and the WDGA bit in the WDGCR is not used. Refer to the Option Byte description.

11.1.8 Using Halt Mode with the WDG (WDGHALT option)

The following recommendation applies if Halt mode is used when the watchdog is enabled.

– Before executing the HALT instruction, refresh the WDG counter, to avoid an unexpected WDG reset immediately after waking up the microcontroller.

WINDOW WATCHDOG (Cont'd)

11.1.9 Interrupts

None.

11.1.10 Register Description

CONTROL REGISTER (WDGCR)

Read/Write

Reset Value: 0111 1111 (7Fh)

7							0
WDGA	T6	T5	T4	T3	T2	T1	T0

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

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

0: Watchdog disabled

1: Watchdog enabled

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

Bits 6:0 = **T[6:0]** *7-bit counter (MSB to LSB)*.
These bits contain the value of the watchdog counter. It is decremented every 16384 f_{OSC2} cycles (approx.). A reset is produced when it rolls over from 40h to 3Fh (T6 becomes cleared).

WINDOW REGISTER (WDGWR)

Read/Write

Reset Value: 0111 1111 (7Fh)

7							0
-	W6	W5	W4	W3	W2	W1	W0

Bit 7 = Reserved

Bits 6:0 = **W[6:0]** *7-bit window value*
These bits contain the window value to be compared to the downcounter.

WINDOW WATCHDOG(Cont'd)

Table 15. Watchdog Timer Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
2A	**WDGCR** Reset Value	WDGA 0	T6 1	T5 1	T4 1	T3 1	T2 1	T1 1	T0 1
30	**WDGWR** Reset Value	- 0	W6 1	W5 1	W4 1	W3 1	W2 1	W1 1	W0 1

11.2 MAIN CLOCK CONTROLLER WITH REAL TIME CLOCK AND BEEPER (MCC/RTC)

The Main Clock Controller consists of three different functions:

- a programmable CPU clock prescaler
- a clock-out signal to supply external devices
- a real time clock timer with interrupt capability

Each function can be used independently and simultaneously.

11.2.1 Programmable CPU Clock Prescaler

The programmable CPU clock prescaler supplies the clock for the ST7 CPU and its internal peripherals. It manages SLOW power saving mode (See Section 9.2 "SLOW MODE" on page 44 for more details).

The prescaler selects the f_{CPU} main clock frequency and is controlled by three bits in the MCCSR register: CP[1:0] and SMS.

11.2.2 Clock-out Capability

The clock-out capability is an alternate function of an I/O port pin that outputs a f_{OSC2} clock to drive external devices. It is controlled by the MCO bit in the MCCSR register.

CAUTION: When selected, the clock out pin suspends the clock during ACTIVE-HALT mode.

11.2.3 Real Time Clock Timer (RTC)

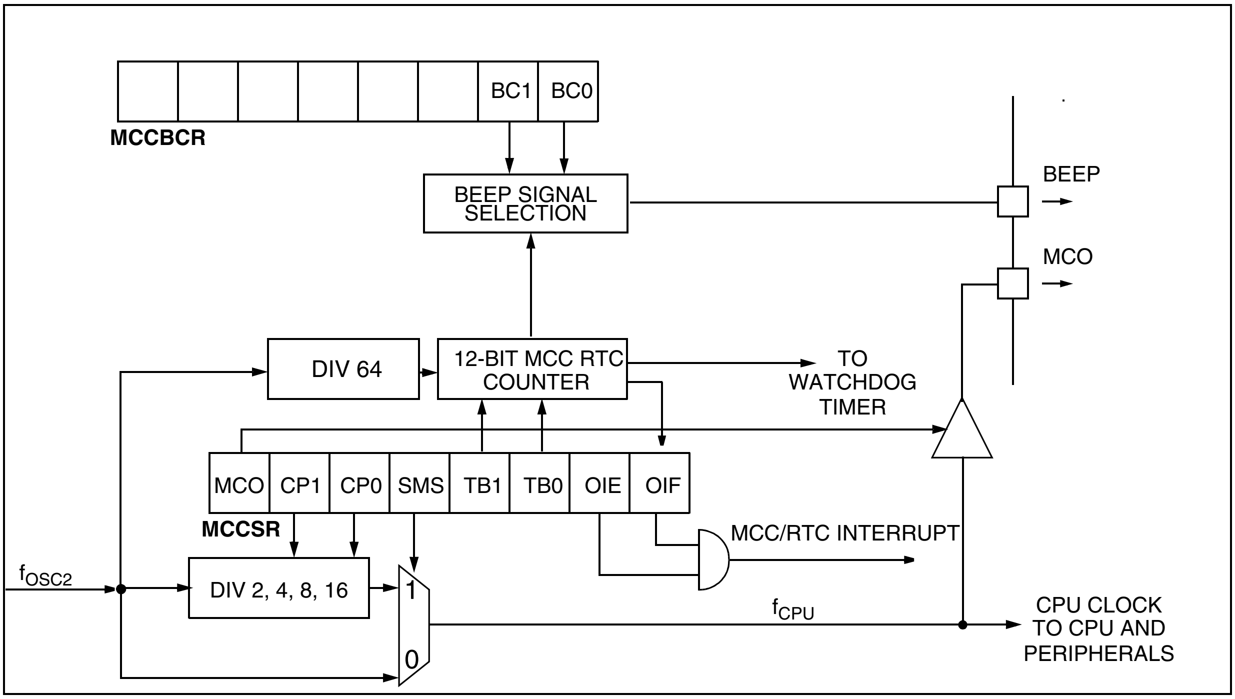
The counter of the real time clock timer allows an interrupt to be generated based on an accurate real time clock. Four different time bases depending directly on f_{OSC2} are available. The whole functionality is controlled by four bits of the MCCSR register: TB[1:0], OIE and OIF.

When the RTC interrupt is enabled (OIE bit set), the ST7 enters ACTIVE-HALT mode when the HALT instruction is executed. See Section 9.5 "ACTIVE-HALT MODE" on page 47 for more details.

11.2.4 Beeper

The beep function is controlled by the MCCBCR register. It can output three selectable frequencies on the BEEP pin (I/O port alternate function).

Figure 41. Main Clock Controller (MCC/RTC) Block Diagram



MAIN CLOCK CONTROLLER WITH REAL TIME CLOCK (Cont'd)

11.2.5 Low Power Modes

Mode	Description
WAIT	No effect on MCC/RTC peripheral. MCC/RTC interrupt cause the device to exit from WAIT mode.
ACTIVE-HALT	No effect on MCC/RTC counter (OIE bit is set), the registers are frozen. MCC/RTC interrupt cause the device to exit from ACTIVE-HALT mode.
HALT	MCC/RTC counter and registers are frozen. MCC/RTC operation resumes when the MCU is woken up by an interrupt with "exit from HALT" capability.

11.2.6 Interrupts

The MCC/RTC interrupt event generates an interrupt if the OIE bit of the MCCSR register is set and the interrupt mask in the CC register is not active (RIM instruction).

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
Time base overflow event	OIF	OIE	Yes	No [1)]

Note:
The MCC/RTC interrupt wakes up the MCU from ACTIVE-HALT mode, not from HALT mode.

11.2.7 Register Description

MCC CONTROL/STATUS REGISTER (MCCSR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
MCO	CP1	CP0	SMS	TB1	TB0	OIE	OIF

Bit 7 = **MCO** *Main clock out selection*
This bit enables the MCO alternate function on the PF0 I/O port. It is set and cleared by software.
0: MCO alternate function disabled (I/O pin free for general-purpose I/O)
1: MCO alternate function enabled (f_{CPU} on I/O port)

Note: To reduce power consumption, the MCO function is not active in ACTIVE-HALT mode.

Bits 6:5 = **CP[1:0]** *CPU clock prescaler*
These bits select the CPU clock prescaler which is applied in the different slow modes. Their action is conditioned by the setting of the SMS bit. These two bits are set and cleared by software

f_{CPU} in SLOW mode	CP1	CP0
f_{OSC2} / 2	0	0
f_{OSC2} / 4	0	1
f_{OSC2} / 8	1	0
f_{OSC2} / 16	1	1

Bit 4 = **SMS** *Slow mode select*
This bit is set and cleared by software.
0: Normal mode. $f_{CPU} = f_{OSC2}$
1: Slow mode. f_{CPU} is given by CP1, CP0
See Section 9.2 "SLOW MODE" on page 44 and Section 11.1 "WINDOW WATCHDOG (WWDG)" on page 58 for more details.

Bits 3:2 = **TB[1:0]** *Time base control*

These bits select the programmable divider time base. They are set and cleared by software.

Counter Prescaler	Time Base		TB1	TB0
	f_{OSC2} =4MHz	f_{OSC2}=8MHz		
16000	4ms	2ms	0	0
32000	8ms	4ms	0	1
80000	20ms	10ms	1	0
200000	50ms	25ms	1	1

A modification of the time base is taken into account at the end of the current period (previously set) to avoid an unwanted time shift. This allows to use this time base as a real time clock.

Bit 1 = **OIE** *Oscillator interrupt enable*
This bit set and cleared by software.
0: Oscillator interrupt disabled
1: Oscillator interrupt enabled
This interrupt can be used to exit from ACTIVE-HALT mode.
When this bit is set, calling the ST7 software HALT instruction enters the ACTIVE-HALT power saving mode.

MAIN CLOCK CONTROLLER WITH REAL TIME CLOCK (Cont'd)

Bit 0 = **OIF** *Oscillator interrupt flag*
This bit is set by hardware and cleared by software reading the MCCSR register. It indicates when set that the main oscillator has reached the selected elapsed time (TB1:0).
0: Timeout not reached
1: Timeout reached

CAUTION: The BRES and BSET instructions must not be used on the MCCSR register to avoid unintentionally clearing the OIF bit.

MCC BEEP CONTROL REGISTER (MCCBCR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	BC1	BC0

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

Bits 1:0 = **BC[1:0]** *Beep control*
These 2 bits select the PF1 pin beep capability.

BC1	**BC0**	**Beep mode with f_{OSC2}=8MHz**	
0	0	Off	
0	1	~2-KHz	Output Beep signal ~50% duty cycle
1	0	~1-KHz	
1	1	~500-Hz	

The beep output signal is available in ACTIVE-HALT mode but has to be disabled to reduce the consumption.

Table 16. Main Clock Controller Register Map and Reset Values

Address (Hex.)	**Register Label**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
002Bh	**SICSR** Reset Value	 0	AVDIE 0	AVDF 0	LVDRF x	LOCKED 0	 0	 0	WDGRF x
002Ch	**MCCSR** Reset Value	MCO 0	CP1 0	CP0 0	SMS 0	TB1 0	TB0 0	OIE 0	OIF 0
002Dh	**MCCBCR** Reset Value	 0	 0	 0	 0	 0	 0	BC1 0	BC0 0

11.3 16-BIT TIMER

11.3.1 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 pulse length measurement of up to two input signals (*input capture*) or generation of up to two output waveforms (*output compare* and *PWM*).

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

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

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

11.3.2 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
- Reduced Power Mode
- 5 alternate functions on I/O ports (ICAP1, ICAP2, OCMP1, OCMP2, EXTCLK)*

The Block Diagram is shown in Figure 42.

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

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

11.3.3 Functional Description

11.3.3.1 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).

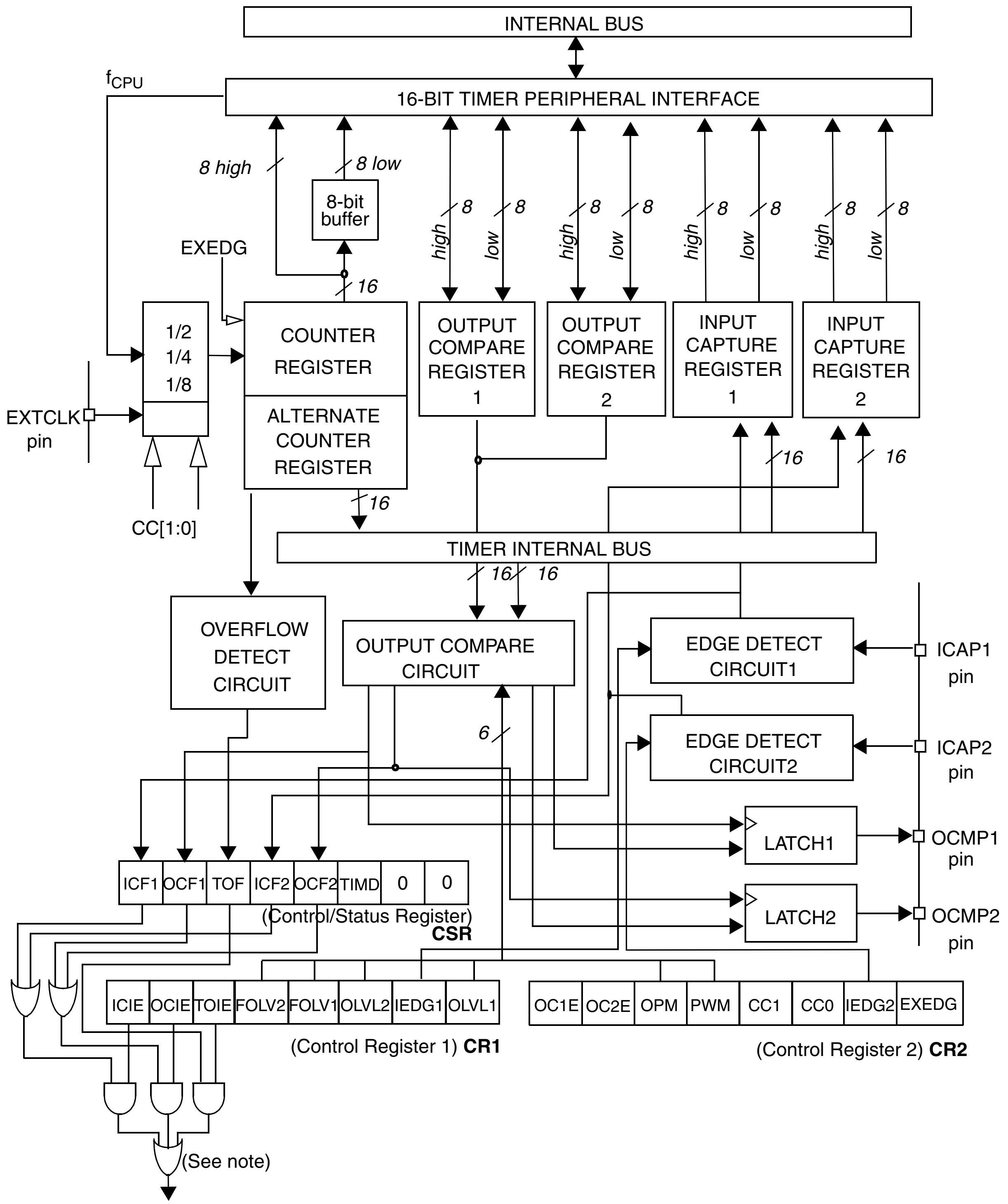
These two read-only 16-bit registers contain the same value but with the difference that reading the ACLR register does not clear the TOF bit (Timer overflow flag), located in the Status register, (SR), (see note at the end of paragraph titled 16-bit read sequence).

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 17 Clock Control Bits. The value in the counter register repeats every 131 072, 262 144 or 524 288 CPU clock cycles depending on the CC[1:0] bits.
The timer frequency can be $f_{CPU}/2$, $f_{CPU}/4$, $f_{CPU}/8$ or an external frequency.

16-BIT TIMER (Cont'd)

Figure 42. Timer Block Diagram

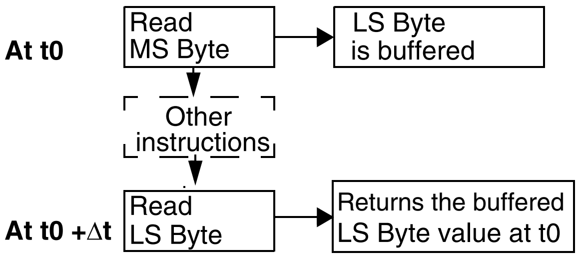


Note: If IC, OC and TO interrupt requests have separate vectors then the last OR is not present (See Device Interrupt Vector Table)

16-BIT TIMER (Cont'd)

16-bit read sequence: (from either the Counter Register or the Alternate Counter Register).

Beginning of the sequence



Sequence completed

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

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

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

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

– The TOF bit of the SR register is set.

– A timer interrupt is generated if:

  – TOIE bit of the CR1 register is set and

  – I bit of the CC register is cleared.

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

Clearing the overflow interrupt request is done in two steps:

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

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

The timer is not affected by WAIT mode.

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

11.3.3.2 External Clock

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

The status of the EXEDG bit in the CR2 register determines the type of level transition on the external clock pin EXTCLK that 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 43. Counter Timing Diagram, internal clock divided by 2

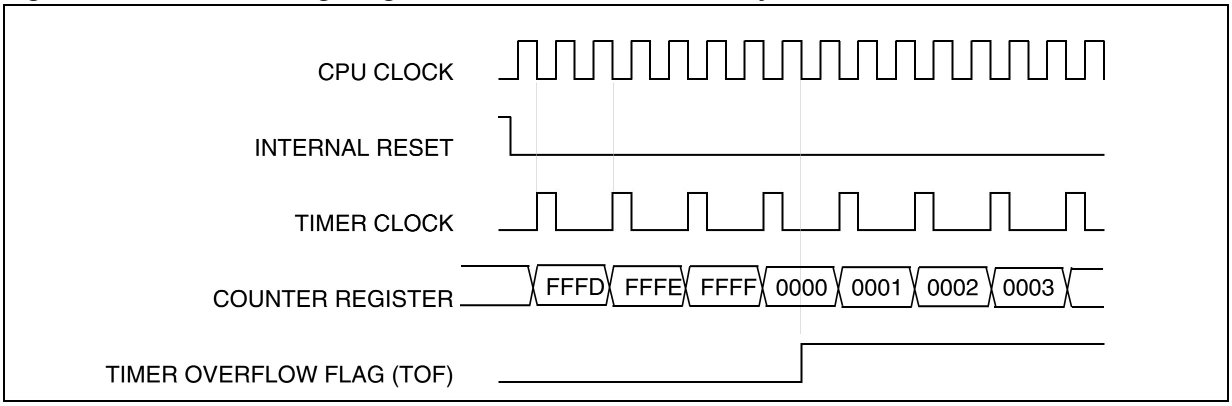


Figure 44. Counter Timing Diagram, internal clock divided by 4

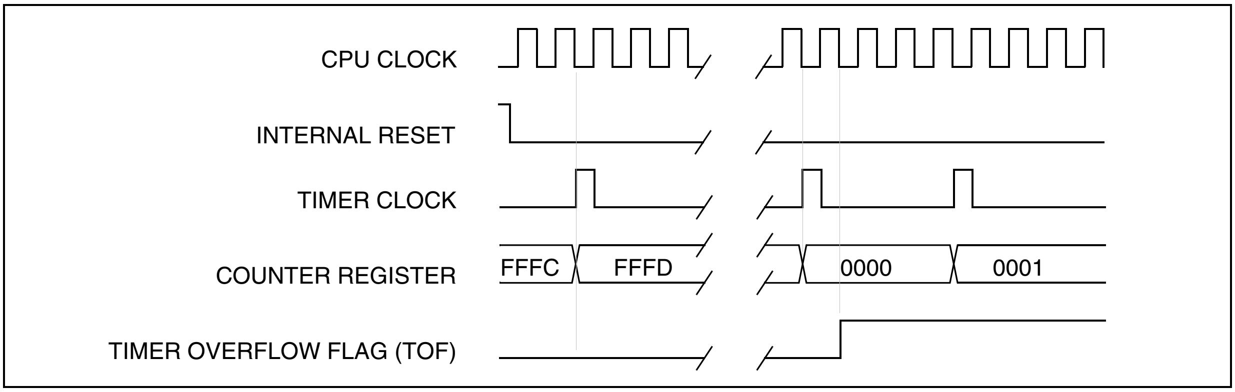
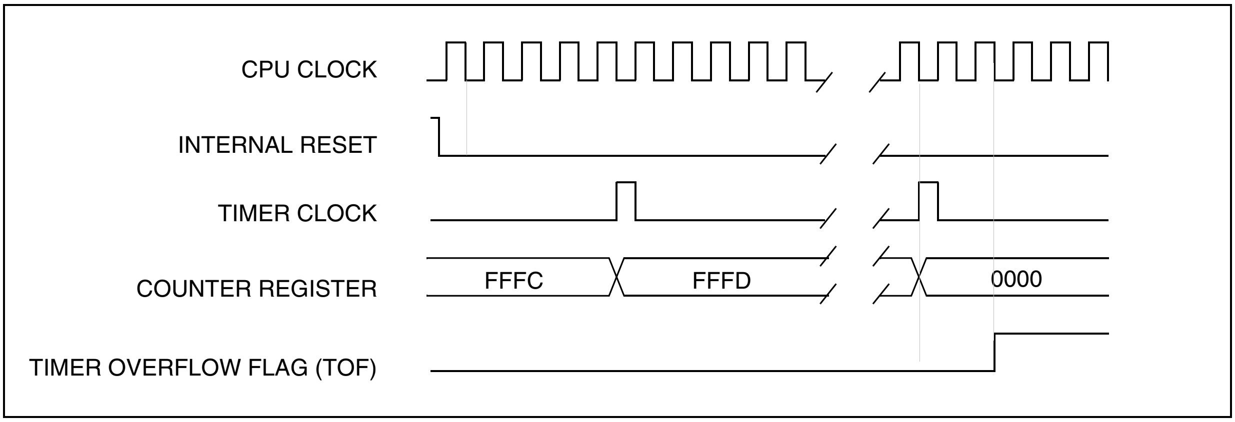


Figure 45. Counter Timing Diagram, internal clock divided by 8



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

16-BIT TIMER (Cont'd)

11.3.3.3 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 detected by the ICAP*i* pin (see figure 5).

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

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

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

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

Procedure:

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

– Select the timer clock (CC[1:0]) (see Table 17 Clock Control Bits).

– Select the edge of the active transition on the ICAP2 pin with the IEDG2 bit (the ICAP2 pin must be configured as 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 floating input).

When an input capture occurs:

– 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 47).

– 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, 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 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 toggle the output pin and if the ICIE bit is set.
   This can be avoided if the input capture function *i* is disabled by reading the IC*i*HR (see note 1).
6. The TOF bit can be used with interrupt in order to measure event that go beyond the timer range (FFFFh).

16-BIT TIMER (Cont'd)

Figure 46. Input Capture Block Diagram

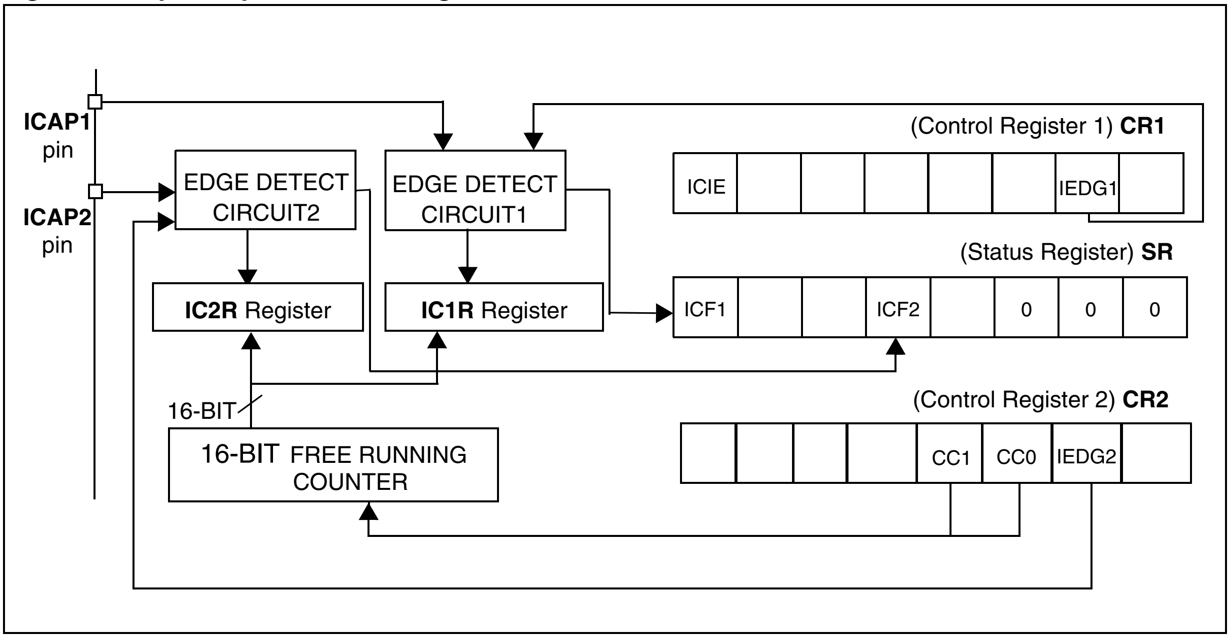
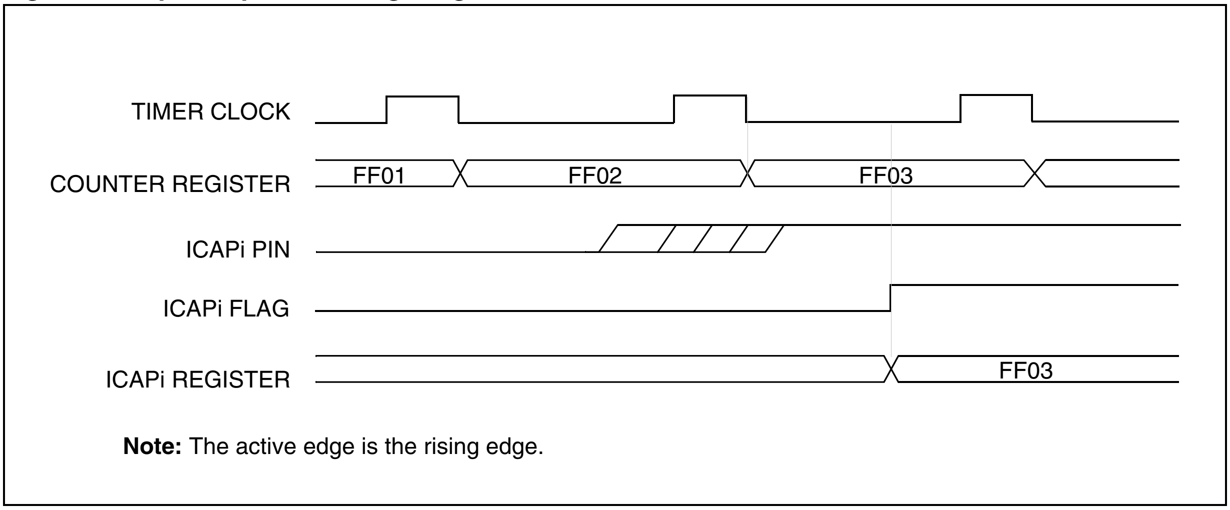


Figure 47. Input Capture Timing Diagram



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

16-BIT TIMER (Cont'd)

11.3.3.4 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 17 Clock Control Bits).

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 CR2 register and the I bit is cleared in the CC register (CC).

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

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

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 17 Clock Control Bits)

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

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

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:

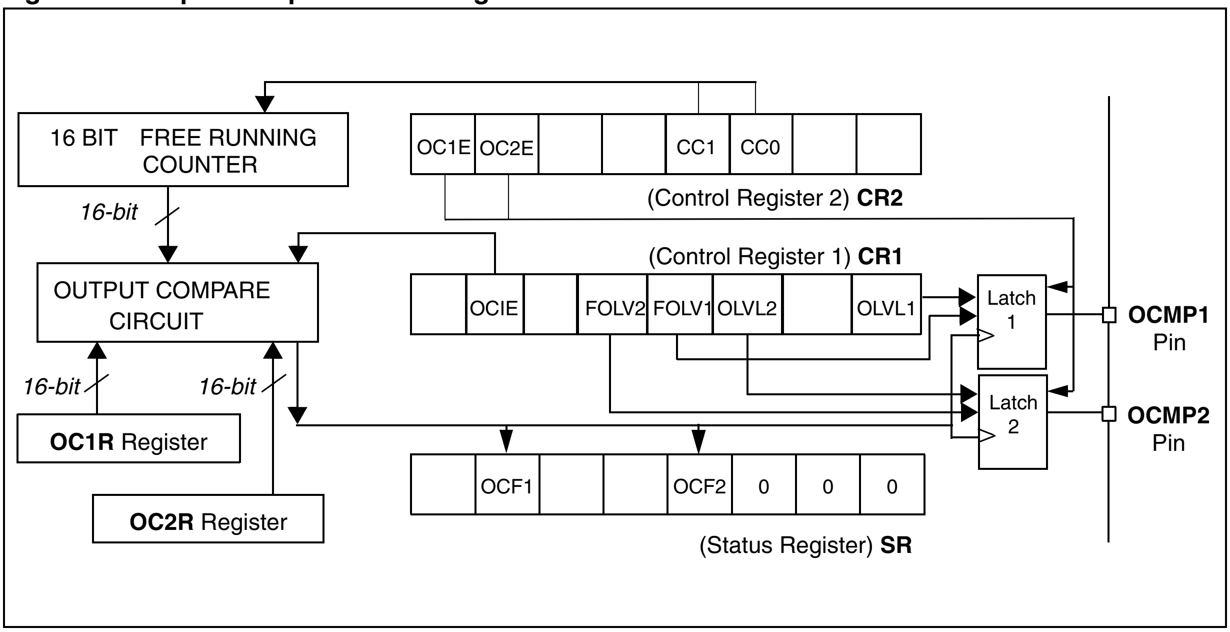
1. After a processor write cycle to the OC*i*HR register, the output compare function is inhibited until the OC*i*LR register is also written.
2. If the OC*i*E bit is not set, the OCMP*i* pin is a general I/O port and the OLVL*i* bit will not appear when a match is found but an interrupt could be generated 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 49 on page 78). 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 50 on page 78).
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 both one pulse mode and PWM mode.

Figure 48. Output Compare Block Diagram



16-BIT TIMER (Cont'd)

Figure 49. Output Compare Timing Diagram, $f_{TIMER} = f_{CPU}/2$

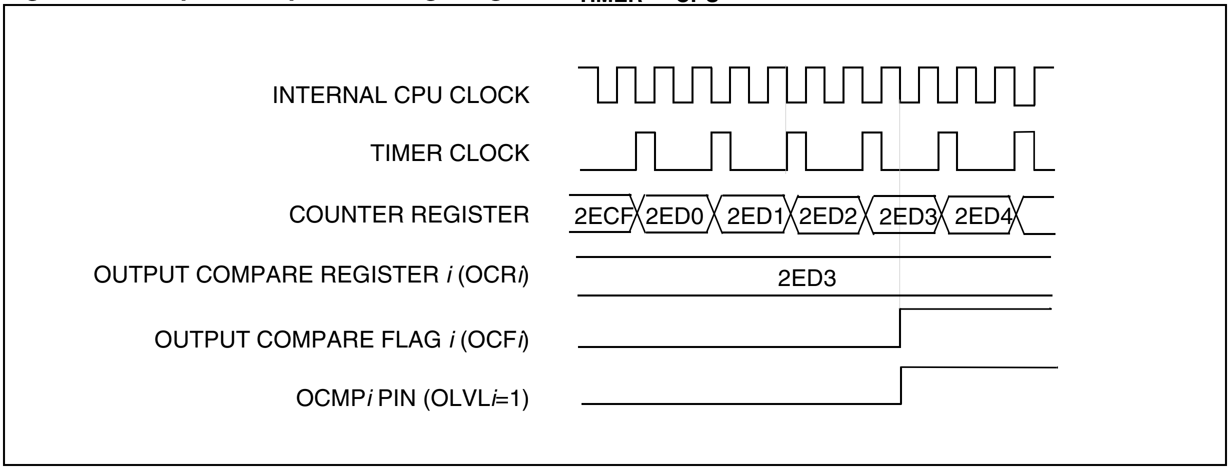
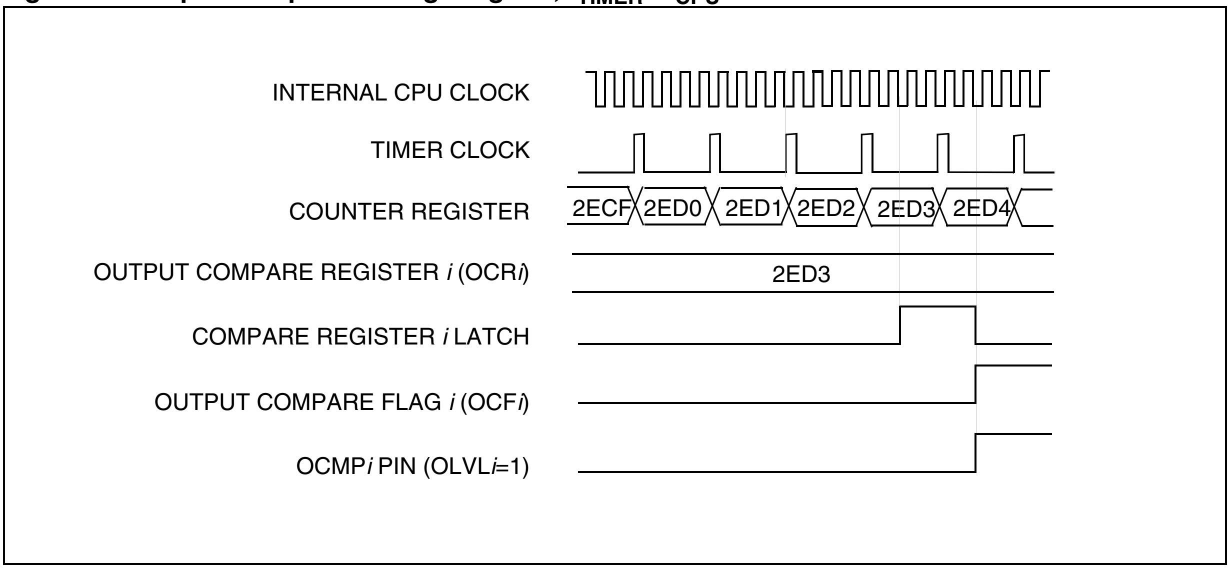


Figure 50. Output Compare Timing Diagram, $f_{TIMER} = f_{CPU}/4$



16-BIT TIMER (Cont'd)

11.3.3.5 One Pulse Mode

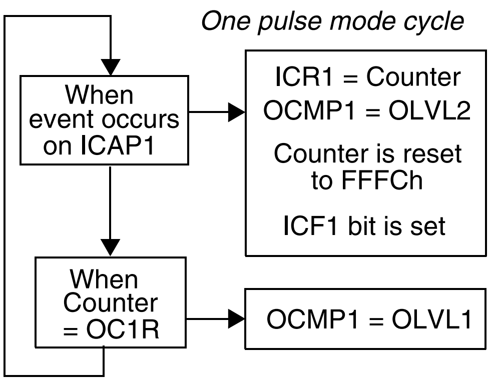
One Pulse mode enables the generation of a pulse when an external event occurs. This mode is selected via the OPM bit in the CR2 register.

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 17 Clock Control Bits).



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

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

Clearing the Input Capture interrupt request (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 17 Clock Control Bits)

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)

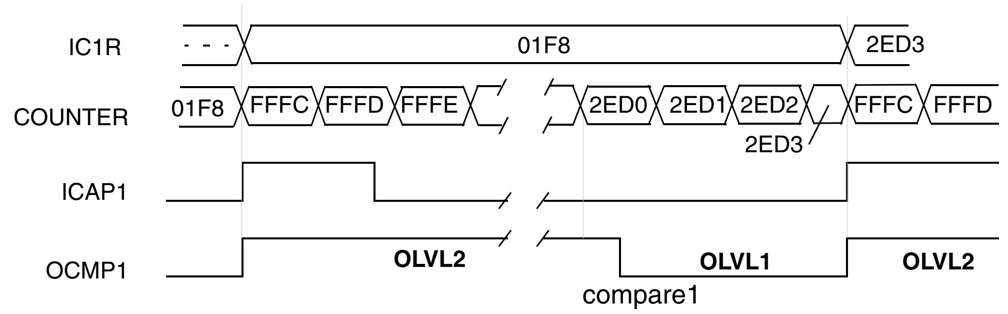
When the value of the counter is equal to the value of the contents of the OC1R register, the OLVL1 bit is output on the OCMP1 pin, (See Figure 51).

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 a period of time has been elapsed but cannot generate an output waveform because the level OLVL2 is dedicated to the one pulse mode.

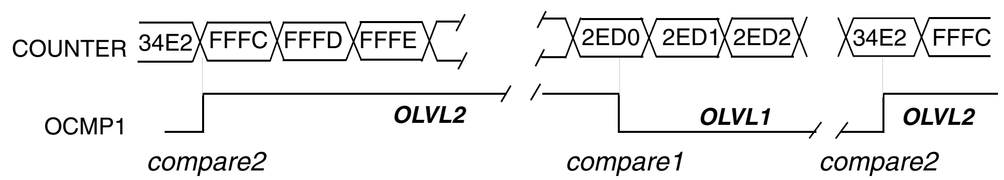
16-BIT TIMER (Cont'd)

Figure 51. One Pulse Mode Timing Example



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

Figure 52. Pulse Width Modulation Mode Timing Example



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

16-BIT TIMER (Cont'd)

11.3.3.6 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.

Pulse Width Modulation mode uses the complete Output Compare 1 function plus the OC2R register, and so this functionality can not be used when PWM mode is activated.

In PWM mode, double buffering is implemented on the output compare registers. Any new values written in the OC1R and OC2R registers are loaded in their respective shadow registers (double buffer) only at the end of the PWM period (OC2) to avoid spikes on the PWM output pin (OCMP1). The shadow registers contain the reference values for comparison in PWM "double buffering" mode.

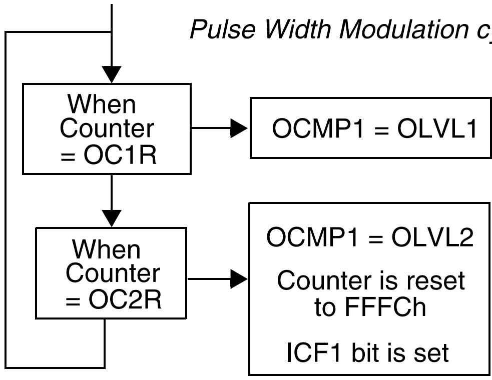
Note: There is a locking mechanism for transferring the OCiR value to the buffer. After a write to the OCiHR register, transfer of the new compare value to the buffer is inhibited until OCiLR is also written.

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

Procedure

To use pulse width modulation mode:

1. Load the OC2R register with the value corresponding to the period of the signal using the formula in the opposite column.
2. Load the OC1R register with the value corresponding to the period of the pulse if (OLVL1=0 and OLVL2=1) using 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 17 Clock Control Bits).



Pulse Width Modulation cycle

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 17 Clock Control Bits)

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 52)

Notes:

1. The OCF1 and OCF2 bits cannot be set by hardware in PWM mode therefore the Output Compare interrupt is inhibited.
2. The ICF1 bit is set by hardware when the counter reaches the OC2R value and can produce a timer interrupt if the ICIE bit is set and the I bit is cleared.

16-BIT TIMER (Cont'd)

3. In PWM mode the ICAP1 pin can not be used to perform input capture because it is disconnected to the timer. The ICAP2 pin can be used to perform input capture (ICF2 can be set and IC2R can be loaded) but the user must take care that the counter is reset each period and ICF1 can also generates interrupt if ICIE is set.
4. When the Pulse Width Modulation (PWM) and One Pulse Mode (OPM) bits are both set, the PWM mode is the only active one.

11.3.4 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 Device is woken up by an interrupt with "exit from HALT mode" capability or from the counter reset value when the Device is woken up by a RESET. If an input capture event occurs on the ICAP*i* pin, the input capture detection circuitry is armed. Consequently, when the Device is woken up by an interrupt with "exit from HALT mode" capability, the ICF*i* bit is set, and the counter value present when exiting from HALT mode is captured into the IC*i*R register.

11.3.5 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).

11.3.6 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 11.3.3.5 "One Pulse Mode" on page 79

2) See note 5 in Section 11.3.3.5 "One Pulse Mode" on page 79

3) See note 4 in Section 11.3.3.6 "Pulse Width Modulation Mode" on page 81

16-BIT TIMER (Cont'd)

11.3.7 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 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 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.

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

The timer clock mode depends on these bits:

Table 17. 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)

CONTROL/STATUS REGISTER (CSR)

Read Only

Reset Value: 0000 0000 (00h)

The three least significant bits are not used.

7							0
ICF1	OCF1	TOF	ICF2	OCF2	TIMD	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 has matched the content of the OC1R register. To clear this bit, first read the SR register, then read or write the low byte of the OC1R (OC1LR) register.

Bit 5 = **TOF** *Timer Overflow Flag.*
0: No timer overflow (reset value).
1: The free running counter rolled over from FFFFh to 0000h. To clear this bit, first read the SR register, then read or write the low byte of the CR (CLR) register.

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

Bit 4 = **ICF2** *Input Capture Flag 2.*
0: No input capture (reset value).
1: An input capture has occurred on the ICAP2 pin. To clear this bit, first read the SR register, then read or write the low byte of the IC2R (IC2LR) register.

Bit 3 = **OCF2** *Output Compare Flag 2.*
0: No match (reset value).
1: The content of the free running counter has matched the content of the OC2R register. To clear this bit, first read the SR register, then read or write the low byte of the OC2R (OC2LR) register.

Bit 2 = **TIMD** *Timer disable.*
This bit is set and cleared by software. When set, it freezes the timer prescaler and counter and disabled the output functions (OCMP1 and OCMP2 pins) to reduce power consumption. Access to the timer registers is still available, allowing the timer configuration to be changed while it is disabled.
0: Timer enabled
1: Timer prescaler, counter and outputs disabled

Bits 1:0 = Reserved, must be kept cleared.

16-BIT TIMER (Cont'd)

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 CSR 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 CSR register does not clear the TOF bit in the CSR 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 18. 16-Bit Timer Register Map and Reset Values

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

ON-CHIP PERIPHERALS (cont'd)

11.4 SERIAL PERIPHERAL INTERFACE (SPI)

11.4.1 Introduction

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

11.4.2 Main Features

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

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

11.4.3 General Description

Figure 1 on page 3 shows the serial peripheral interface (SPI) block diagram. There are three registers:

– SPI Control Register (SPICR)

– SPI Control/Status Register (SPICSR)

– SPI Data Register (SPIDR)

The SPI is connected to external devices through four pins:

– MISO: Master In / Slave Out data

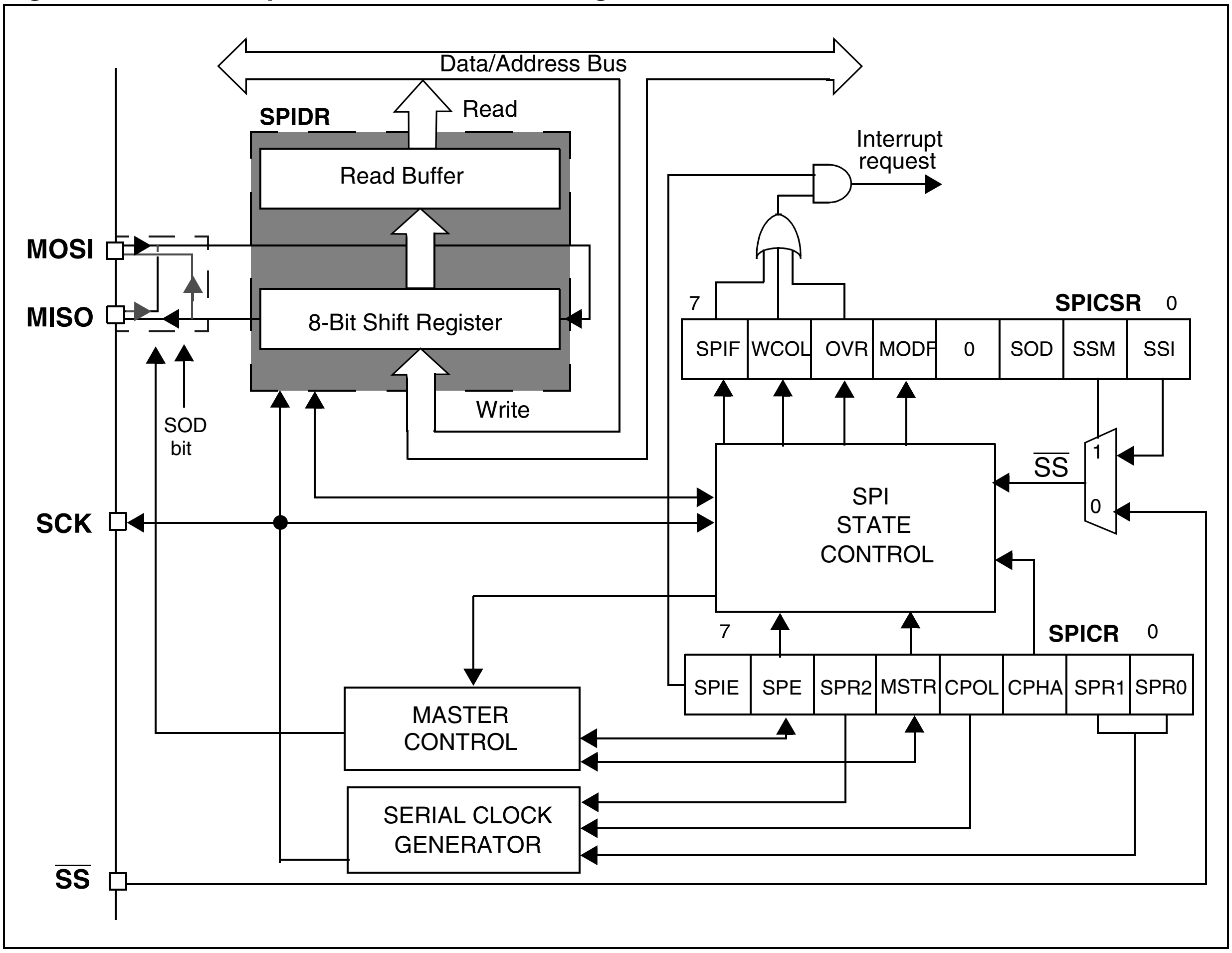
– MOSI: Master Out / Slave In data

– SCK: Serial Clock out by SPI masters and input by SPI slaves

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

SERIAL PERIPHERAL INTERFACE (SPI) (cont'd)

Figure 53. Serial Peripheral Interface Block Diagram



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.4.3.1 Functional Description

A basic example of interconnections between a single master and a single slave is illustrated in Figure 2.

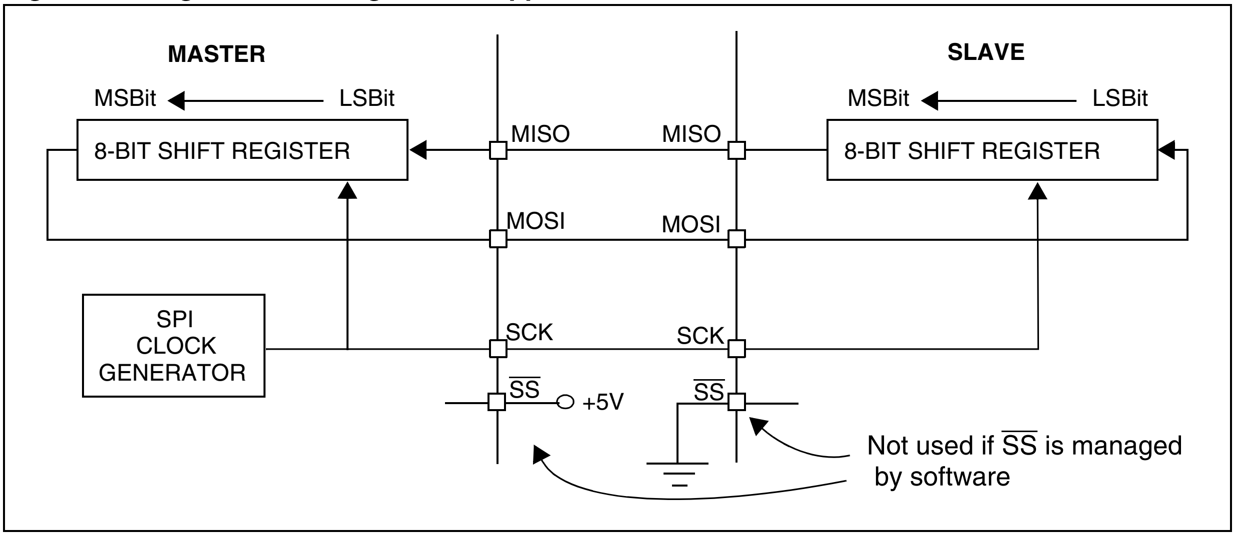
The MOSI pins are connected together and the MISO pins are connected together. In this way data is transferred serially between master and slave (most significant bit first).

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

To use a single data line, the MISO and MOSI pins must be connected at each node (in this case only simplex communication is possible).

Four possible data/clock timing relationships may be chosen (see Figure 5 on page 7) but master and slave must be programmed with the same timing mode.

Figure 54. Single Master/ Single Slave Application



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.4.3.2 Slave Select Management

As an alternative to using the $\overline{SS}$ pin to control the Slave Select signal, the application can choose to manage the Slave Select signal by software. This is configured by the SSM bit in the SPICSR register (see Figure 4).

In software management, the external $\overline{SS}$ pin is free for other application uses and the internal $\overline{SS}$ signal level is driven by writing to the SSI bit in the SPICSR register.

In Master mode:

– $\overline{SS}$ internal must be held high continuously

In Slave Mode:

There are two cases depending on the data/clock timing relationship (see Figure 3):

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

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

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

– $\overline{SS}$ internal must be held low during byte transmission and pulled high between each byte to allow the slave to write to the shift register. If $\overline{SS}$ is not pulled high, a Write Collision error will occur when the slave writes to the shift register (see Section 0.1.5.3).

Figure 55. Generic $\overline{SS}$ Timing Diagram

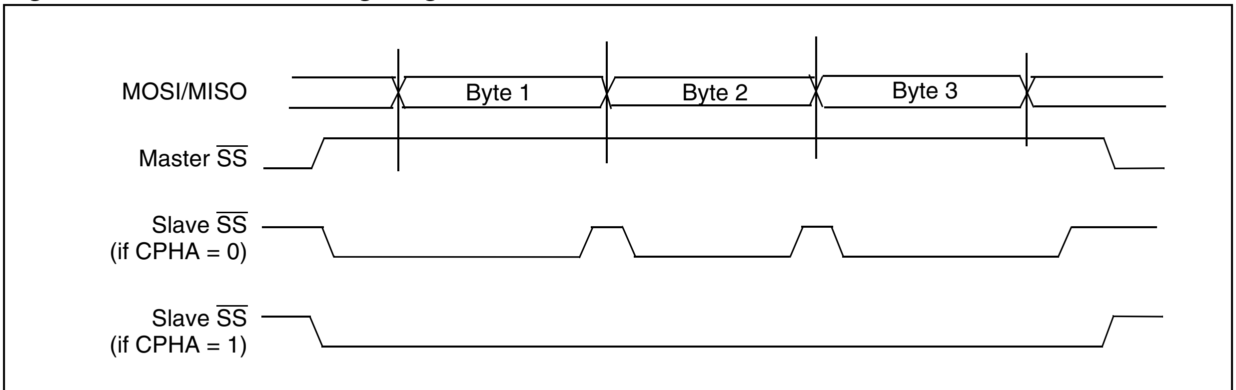
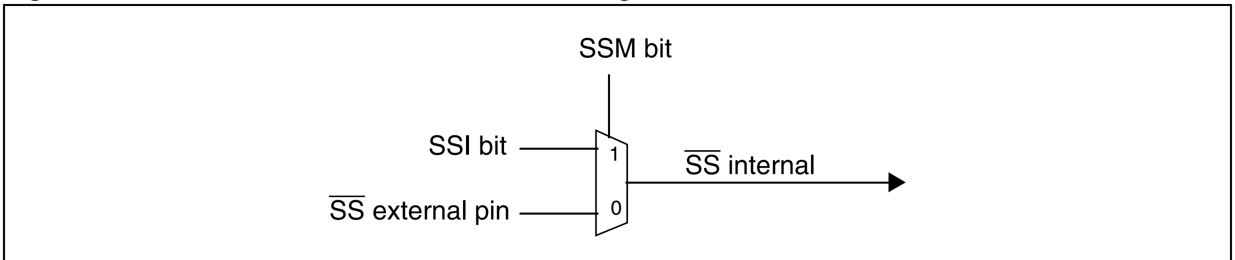


Figure 56. Hardware/Software Slave Select Management



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.4.3.3 Master Mode Operation

In master mode, the serial clock is output on the SCK pin. The clock frequency, polarity and phase are configured by software (refer to the description of the SPICSR register).

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

How to operate the SPI in master mode

To operate the SPI in master mode, perform the following steps in order:

1. Write to the SPICR register:
   – Select the clock frequency by configuring the SPR[2:0] bits.
   – Select the clock polarity and clock phase by configuring the CPOL and CPHA bits. Figure 5 shows the four possible configurations.
   Note: The slave must have the same CPOL and CPHA settings as the master.
2. Write to the SPICSR register:
   – Either set the SSM bit and set the SSI bit or clear the SSM bit and tie the SS pin high for the complete byte transmit sequence.
3. Write to the SPICR register:
   – Set the MSTR and SPE bits
   Note: MSTR and SPE bits remain set only if SS is high).

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

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

11.4.3.4 Master Mode Transmit Sequence

When software writes to the SPIDR register, the data byte is loaded into the 8-bit shift register and then shifted out serially to the MOSI pin most significant bit first.

When data transfer is complete:

– The SPIF bit is set by hardware.

– An interrupt request is generated if the SPIE bit is set and the interrupt mask in the CCR register is cleared.

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

1. An access to the SPICSR register while the SPIF bit is set
2. A read to the SPIDR register

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

11.4.3.5 Slave Mode Operation

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

To operate the SPI in slave mode:

1. Write to the SPICSR register to perform the following actions:
   – Select the clock polarity and clock phase by configuring the CPOL and CPHA bits (see Figure 5).
   Note: The slave must have the same CPOL and CPHA settings as the master.
   – Manage the SS pin as described in Section 0.1.3.2 and Figure 3. If CPHA = 1 SS must be held low continuously. If CPHA = 0 SS must be held low during byte transmission and pulled up between each byte to let the slave write in the shift register.
2. Write to the SPICR register to clear the MSTR bit and set the SPE bit to enable the SPI I/O functions.

11.4.3.6 Slave Mode Transmit Sequence

When software writes to the SPIDR register, the data byte is loaded into the 8-bit shift register and then shifted out serially to the MISO pin most significant bit first.

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

When data transfer is complete:

– The SPIF bit is set by hardware.

– An interrupt request is generated if SPIE bit is set and interrupt mask in the CCR register is cleared.

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

1. An access to the SPICSR register while the SPIF bit is set
2. A write or a read to the SPIDR register

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

The SPIF bit can be cleared during a second transmission; however, it must be cleared before the second SPIF bit in order to prevent an Overrun condition (see Section 0.1.5.2).

SERIAL PERIPHERAL INTERFACE (Cont'd)

11.4.4 Clock Phase and Clock Polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits (See Figure 5).

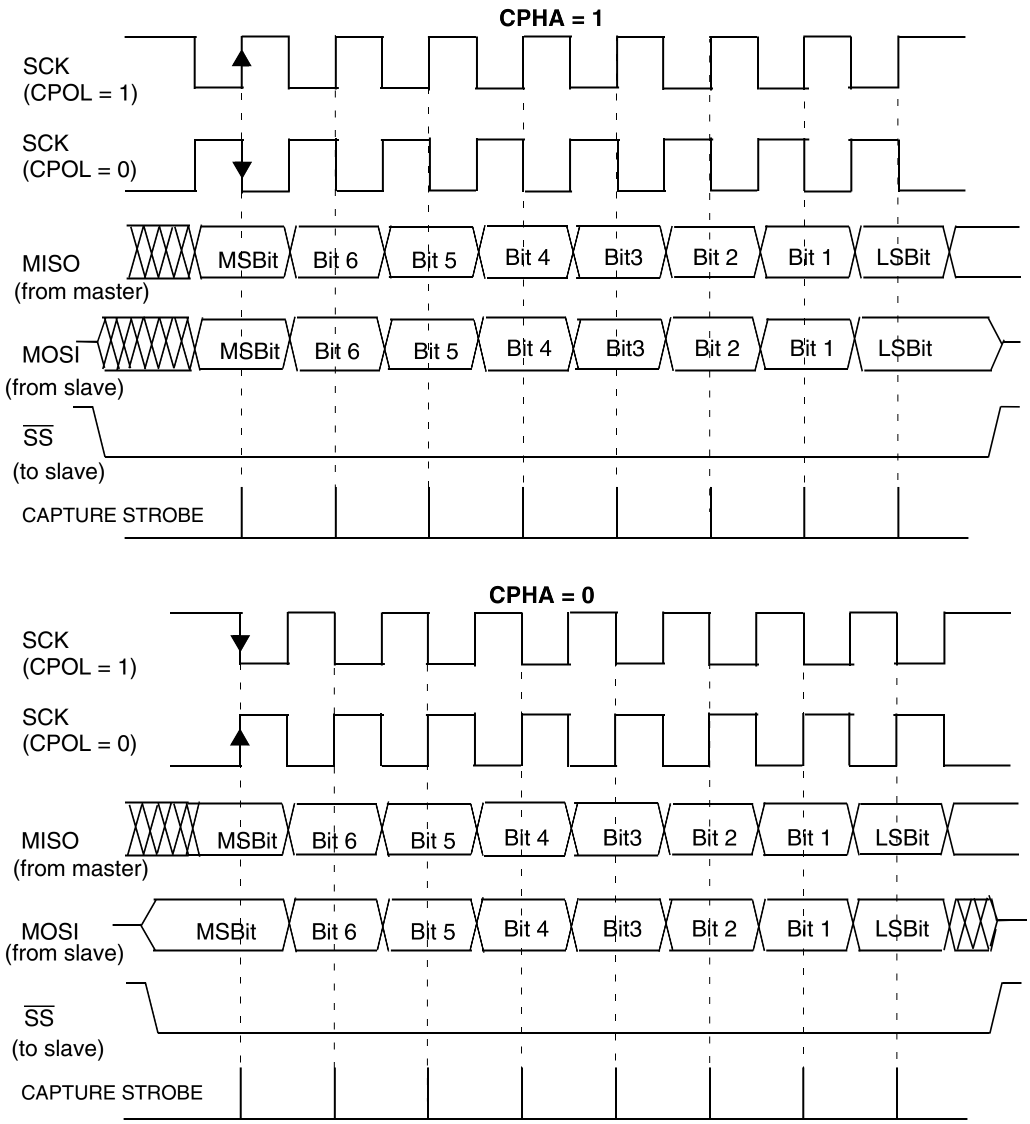
Note: The idle state of SCK must correspond to the polarity selected in the SPICSR register (by pulling up SCK if CPOL = 1 or pulling down SCK if CPOL = 0).

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

Figure 5 shows an SPI transfer with the four combinations of the CPHA and CPOL bits. The diagram may be interpreted as a master or slave timing diagram where the SCK pin, the MISO pin and the MOSI pin are directly connected between the master and the slave device.

Note: If CPOL is changed at the communication byte boundaries, the SPI must be disabled by resetting the SPE bit.

Figure 57. Data Clock Timing Diagram



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

SERIAL PERIPHERAL INTERFACE (Cont'd)

11.4.5 Error Flags

11.4.5.1 Master Mode Fault (MODF)

Master mode fault occurs when the master device's $\overline{SS}$ pin is pulled low.

When a Master mode fault occurs:

– The MODF bit is set and an SPI interrupt request is generated if the SPIE bit is set.

– The SPE bit is reset. This blocks all output from the device and disables the SPI peripheral.

– The MSTR bit is reset, thus forcing the device into slave mode.

Clearing the MODF bit is done through a software sequence:

1. A read access to the SPICSR register while the MODF bit is set.
2. A write to the SPICR register.

Notes: To avoid any conflicts in an application with multiple slaves, the $\overline{SS}$ pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits may be restored to their original state during or after this clearing sequence.

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

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

The MODF bit indicates that there might have been a multimaster conflict and allows software to handle this using an interrupt routine and either perform a reset or return to an application default state.

11.4.5.2 Overrun Condition (OVR)

An overrun condition occurs when the master device has sent a data byte and the slave device has not cleared the SPIF bit issued from the previously transmitted byte.

When an Overrun occurs:

– The OVR bit is set and an interrupt request is generated if the SPIE bit is set.

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

The OVR bit is cleared by reading the SPICSR register.

11.4.5.3 Write Collision Error (WCOL)

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

Write collisions can occur both in master and slave mode. See also Section 0.1.3.2 Slave Select Management.

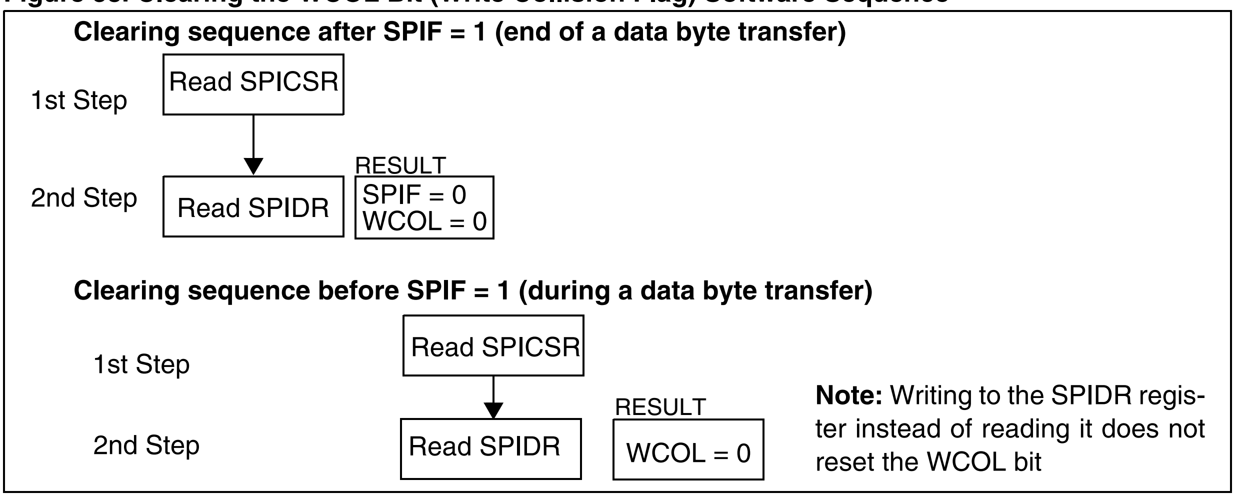
Note: A "read collision" will never occur since the received data byte is placed in a buffer in which access is always synchronous with the CPU operation.

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

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

Clearing the WCOL bit is done through a software sequence (see Figure 6).

Figure 58. Clearing the WCOL Bit (Write Collision Flag) Software Sequence



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.4.5.4 Single Master and Multimaster Configurations

There are two types of SPI systems:

– Single Master System

– Multimaster System

Single Master System

A typical single master system may be configured using a device as the master and four devices as slaves (see Figure 7).

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

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

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

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

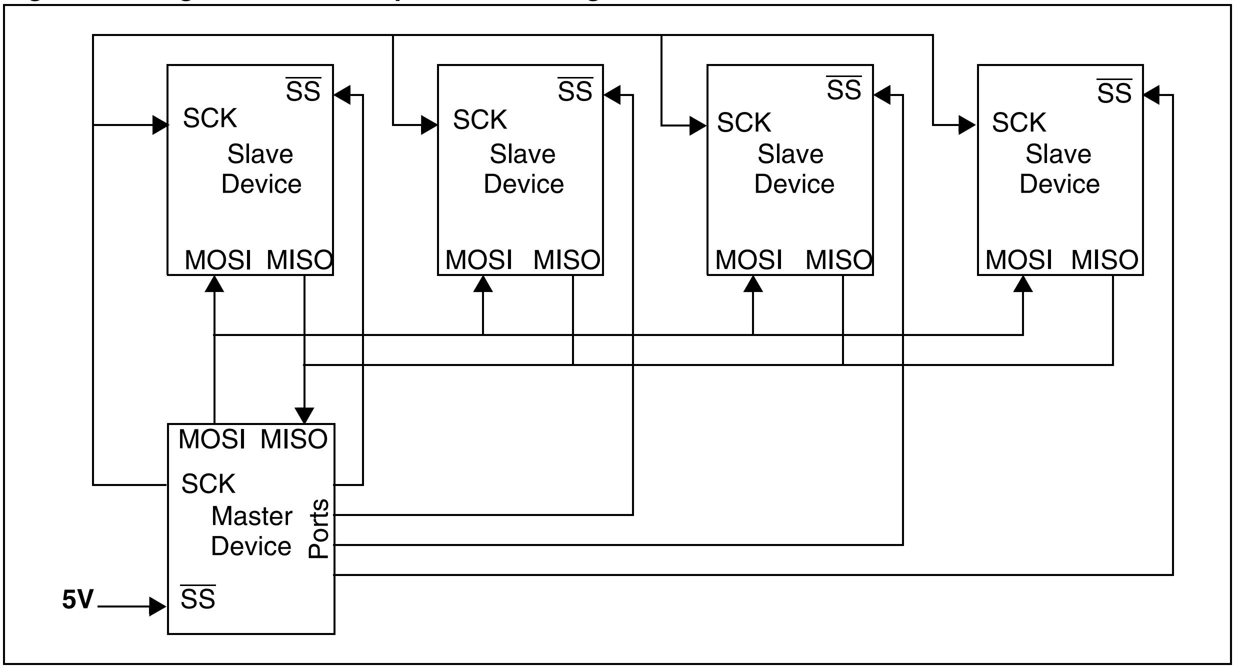
Other transmission security methods can use ports for handshake lines or data bytes with command fields.

Multimaster System

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

The multimaster system is principally handled by the MSTR bit in the SPICR register and the MODF bit in the SPICSR register.

Figure 59. Single Master / Multiple Slave Configuration



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.4.6 Low Power Modes

Mode	Description
WAIT	No effect on SPI. SPI interrupt events cause the device to exit from WAIT mode.
HALT	SPI registers are frozen. In HALT mode, the SPI is inactive. SPI operation resumes when the device is woken up by an interrupt with "exit from HALT mode" capability. The data received is subsequently read from the SPIDR register when the software is running (interrupt vector fetching). If several data are received before the wake-up event, then an overrun error is generated. This error can be detected after the fetch of the interrupt routine that woke up the Device.

11.4.6.1 Using the SPI to wake up the device from Halt mode

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

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

Caution: The SPI can wake up the device from HALT mode only if the Slave Select signal (external $\overline{SS}$ pin or the SSI bit in the SPICSR register) is low when the device enters HALT mode. So, if Slave selection is configured as external (see Section 0.1.3.2), make sure the master drives a low level on the $\overline{SS}$ pin when the slave enters HALT mode.

11.4.7 Interrupts

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

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

11.4.8 Register Description

SPI CONTROL REGISTER (SPICR)

Read/Write

Reset Value: 0000 xxxx (0xh)

7							0
SPIE	SPE	SPR2	MSTR	CPOL	CPHA	SPR1	SPR0

Bit 7 = **SPIE** *Serial Peripheral Interrupt Enable*
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SPI interrupt is generated whenever an End of Transfer event, Master Mode Fault or Overrun error occurs (SPIF = 1, MODF = 1 or OVR = 1 in the SPICSR register)

Bit 6 = **SPE** *Serial Peripheral Output Enable*
This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$ = 0 (see Section 0.1.5.1 Master Mode Fault (MODF)). The SPE bit is cleared by reset, so the SPI peripheral is not initially connected to the external pins.
0: I/O pins free for general purpose I/O
1: SPI I/O pin alternate functions enabled

Bit 5 = **SPR2** *Divider Enable*
This bit is set and cleared by software and is cleared by reset. It is used with the SPR[1:0] bits to set the baud rate. Refer to Table 1 SPI Master Mode SCK Frequency.
0: Divider by 2 enabled
1: Divider by 2 disabled

Note: This bit has no effect in slave mode.

Bit 4 = **MSTR** *Master Mode*
This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$ = 0 (see Section 0.1.5.1 Master Mode Fault (MODF)).
0: Slave mode
1: Master mode. The function of the SCK pin changes from an input to an output and the functions of the MISO and MOSI pins are reversed.

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

Note: If CPOL is changed at the communication byte boundaries, the SPI must be disabled by resetting the SPE bit.

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

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

Bits 1:0 = **SPR[1:0]** *Serial Clock Frequency*
These bits are set and cleared by software. Used with the SPR2 bit, they select the baud rate of the SPI serial clock SCK output by the SPI in master mode.

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

Table 19. SPI Master Mode SCK Frequency

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

SERIAL PERIPHERAL INTERFACE (Cont'd)

SPI CONTROL/STATUS REGISTER (SPICSR)

Read/Write (some bits Read Only)
Reset Value: 0000 0000 (00h)

7							0
SPIF	WCOL	OVR	MODF	-	SOD	SSM	SSI

Bit 7 = **SPIF** *Serial Peripheral Data Transfer Flag (Read only)*
This bit is set by hardware when a transfer has been completed. An interrupt is generated if SPIE = 1 in the SPICR register. It is cleared by a software sequence (an access to the SPICSR register followed by a write or a read to the SPIDR register).
0: Data transfer is in progress or the flag has been cleared.
1: Data transfer between the device and an external device has been completed.

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

Bit 6 = **WCOL** *Write Collision status (Read only)*
This bit is set by hardware when a write to the SPIDR register is done during a transmit sequence. It is cleared by a software sequence (see Figure 6).
0: No write collision occurred
1: A write collision has been detected

Bit 5 = **OVR** S*PI Overrun error (Read only)*
This bit is set by hardware when the byte currently being received in the shift register is ready to be transferred into the SPIDR register while SPIF = 1 (See Section 0.1.5.2). An interrupt is generated if SPIE = 1 in the SPICR register. The OVR bit is cleared by software reading the SPICSR register.
0: No overrun error
1: Overrun error detected

Bit 4 = **MODF** *Mode Fault flag (Read only)*
This bit is set by hardware when the $\overline{SS}$ pin is pulled low in master mode (see Section 0.1.5.1 Master Mode Fault (MODF)). An SPI interrupt can be generated if SPIE = 1 in the SPICR register. This bit is cleared by a software sequence (An access to the SPICSR register while MODF = 1 followed by a write to the SPICR register).
0: No master mode fault detected
1: A fault in master mode has been detected

Bit 3 = Reserved, must be kept cleared.

Bit 2 = **SOD** *SPI Output Disable*
This bit is set and cleared by software. When set, it disables the alternate function of the SPI output (MOSI in master mode / MISO in slave mode)
0: SPI output enabled (if SPE = 1)
1: SPI output disabled

Bit 1 = **SSM** *$\overline{SS}$ Management*
This bit is set and cleared by software. When set, it disables the alternate function of the SPI $\overline{SS}$ pin and uses the SSI bit value instead. See Section 0.1.3.2 Slave Select Management.
0: Hardware management ($\overline{SS}$ managed by external pin)
1: Software management (internal $\overline{SS}$ signal controlled by SSI bit. External $\overline{SS}$ pin free for general-purpose I/O)

Bit 0 = **SSI** *$\overline{SS}$ Internal Mode*
This bit is set and cleared by software. It acts as a 'chip select' by controlling the level of the $\overline{SS}$ slave select signal when the SSM bit is set.
0: Slave selected
1: Slave deselected

SPI DATA I/O REGISTER (SPIDR)

Read/Write
Reset Value: Undefined

7							0
D7	D6	D5	D4	D3	D2	D1	D0

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

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

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

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

A read to the SPIDR register returns the value located in the buffer and not the content of the shift register (see Figure 1).

SERIAL PERIPHERAL INTERFACE (Cont'd)

Table 20. SPI Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0021h	**SPIDR** Reset Value	MSB x	 x	 x	 x	 x	 x	 x	LSB x
0022h	**SPICR** Reset Value	SPIE 0	SPE 0	SPR2 0	MSTR 0	CPOL x	CPHA x	SPR1 x	SPR0 x
0023h	**SPICSR** Reset Value	SPIF 0	WCOL 0	OR 0	MODF 0	 0	SOD 0	SSM 0	SSI 0

11.5 SCI SERIAL COMMUNICATION INTERFACE

11.5.1 Introduction

The Serial Communications Interface (SCI) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. The SCI offers a very wide range of baud rates using two baud rate generator systems.

11.5.2 Main Features

- Full duplex, asynchronous communications
- NRZ standard format (Mark/Space)
- Dual baud rate generator systems
- Independently programmable transmit and receive baud rates up to 500K baud
- Programmable data word length (8 or 9 bits)
- Receive buffer full, Transmit buffer empty and End of Transmission flags
- 2 receiver wake-up modes:
  – Address bit (MSB)
  – Idle line
- Muting function for multiprocessor configurations
- Separate enable bits for Transmitter and Receiver
- 4 error detection flags:
  – Overrun error
  – Noise error
  – Frame error
  – Parity error
- 5 interrupt sources with flags:
  – Transmit data register empty
  – Transmission complete
  – Receive data register full
  – Idle line received
  – Overrun error detected
- Parity control:
  – Transmits parity bit
  – Checks parity of received data byte
- Reduced power consumption mode

11.5.3 General Description

The interface is externally connected to another device by three pins (see Figure 1). Any SCI bidirectional communication requires a minimum of two pins: Receive Data In (RDI) and Transmit Data Out (TDO):

– SCLK: Transmitter clock output. This pin outputs the transmitter data clock for synchronous transmission (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). This can be used to control peripherals that have shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable.

– TDO: Transmit Data Output. When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TDO pin is at high level.

– RDI: Receive Data Input is the serial data input. Oversampling techniques are used for data recovery by discriminating between valid incoming data and noise.

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

– An Idle Line prior to transmission or reception

– A start bit

– A data word (8 or 9 bits) least significant bit first

– A Stop bit indicating that the frame is complete.

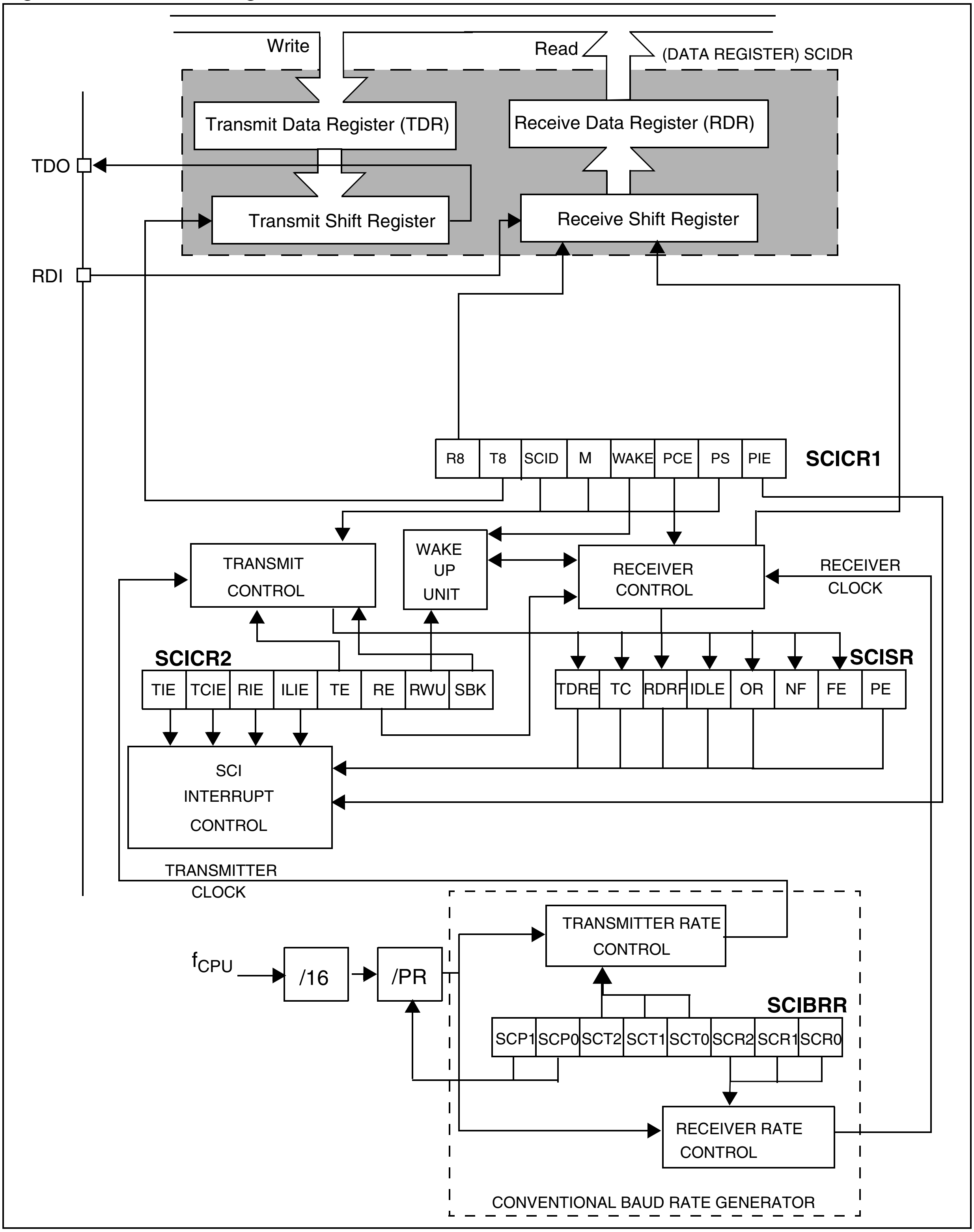
This interface uses two types of baud rate generator:

– A conventional type for commonly-used baud rates,

– An extended type with a prescaler offering a very wide range of baud rates even with non-standard oscillator frequencies.

SCI SERIAL COMMUNICATION INTERFACE (Cont'd)

Figure 60. SCI Block Diagram



SCI SERIAL COMMUNICATION INTERFACE (Cont'd)

11.5.4 Functional Description

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

– 2 control registers (SCICR1 and SCICR2)

– A status register (SCISR)

– A baud rate register (SCIBRR)

– An extended prescaler receiver register (SCIERPR)

– An extended prescaler transmitter register (SCIETPR)

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

11.5.4.1 Serial Data Format

Word length may be selected as being either 8 or 9 bits by programming the M bit in the SCICR1 register (see Figure 2).

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

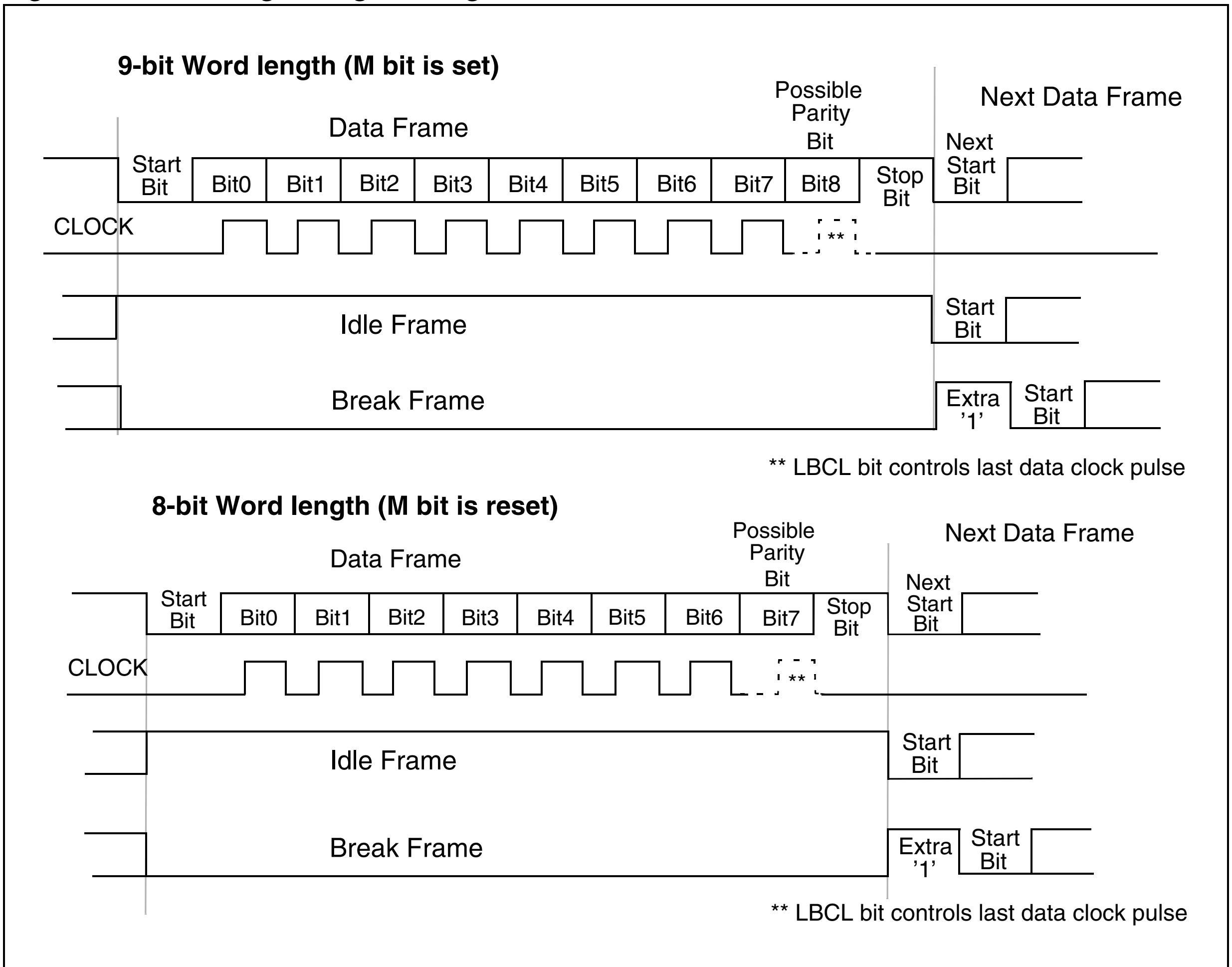
The TDO pin is in high state during the stop bit.

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

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

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

Figure 61. Word Length Programming



SCI SERIAL COMMUNICATION INTERFACE (Cont'd)

11.5.4.2 Transmitter

The transmitter can send data words of either 8 or 9 bits depending on the M bit status. When the M bit is set, word length is 9 bits and the 9th bit (the MSB) has to be stored in the T8 bit in the SCICR1 register.

When the transmit enable bit (TE) is set, the data in the transmit shift register is output on the TDO pin.

Character Transmission

During an SCI transmission, data shifts out least significant bit first on the TDO pin. In this mode, the SCIDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see Figure 2).

Procedure

– Select the M bit to define the word length.

– Select the desired baud rate using the SCIBRR and the SCIETPR registers.

– Set the TE bit to send an idle frame as first transmission.

– Access the SCISR register and write the data to send in the SCIDR register (this sequence clears the TDRE bit). Repeat this sequence for each data to be transmitted.

Clearing the TDRE bit is always performed by the following software sequence:
1. An access to the SCISR register
2. A write to the SCIDR register

The TDRE bit is set by hardware and it indicates:

– The TDR register is empty.

– The data transfer is beginning.

– The next data can be written in the SCIDR register without overwriting the previous data.

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

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

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

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

Clearing the TC bit is performed by the following software sequence:
1. An access to the SCISR register
2. A write to the SCIDR register

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

Break Characters

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

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

Idle Characters

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

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

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

SCI SERIAL COMMUNICATION INTERFACE (Cont'd)

11.5.4.3 Receiver

The SCI can receive data words of either 8 or 9 bits. When the M bit is set, word length is 9 bits and the MSB is stored in the R8 bit in the SCICR1 register.

Character reception

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

Procedure

– Select the M bit to define the word length.

– Select the desired baud rate using the SCIBRR and the SCIERPR registers.

– Set the RE bit, this enables the receiver which begins searching for a start bit.

When a character is received:

– The RDRF bit is set. It indicates that the content of the shift register is transferred to the RDR.

– An interrupt is generated if the RIE bit is set and the I bit is cleared in the CCR register.

– The error flags can be set if a frame error, noise or an overrun error has been detected during reception.

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

1. An access to the SCISR register
2. A read to the SCIDR register.

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

Break Character

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

Idle Character

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

Overrun Error

An overrun error occurs when a character is received when RDRF has not been reset. Data cannot be transferred from the shift register to the RDR register until the RDRF bit is cleared.

When a overrun error occurs:

– The OR bit is set.

– The RDR content is not lost.

– The shift register is overwritten.

– An interrupt is generated if the RIE bit is set and the I bit is cleared in the CCR register.

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

Noise Error

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

Normal data bits are considered valid if three consecutive samples (8th, 9th, 10th) have the same bit value, otherwise the NF flag is set. In the case of start bit detection, the NF flag is set on the basis of an algorithm combining both valid edge detection and three samples (8th, 9th, 10th). Therefore, to prevent the NF flag getting set during start bit reception, there should be a valid edge detection as well as three valid samples.

When noise is detected in a frame:

– The NF flag is set at the rising edge of the RDRF bit.

– Data is transferred from the Shift register to the SCIDR register.

– No interrupt is generated. However this bit rises at the same time as the RDRF bit which itself generates an interrupt.

The NF flag is reset by a SCISR register read operation followed by a SCIDR register read operation.

During reception, if a false start bit is detected (e.g. 8th, 9th, 10th samples are 011,101,110), the frame is discarded and the receiving sequence is not started for this frame. There is no RDRF bit set for this frame and the NF flag is set internally (not accessible to the user). This NF flag is accessible along with the RDRF bit when a next valid frame is received.

Note: If the application Start Bit is not long enough to match the above requirements, then the NF Flag may get set due to the short Start Bit. In this case, the NF flag may be ignored by the application software when the first valid byte is received.

See also Section 0.1.4.10 .

Framing Error

A framing error is detected when:

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

– A break is received.

When the framing error is detected:

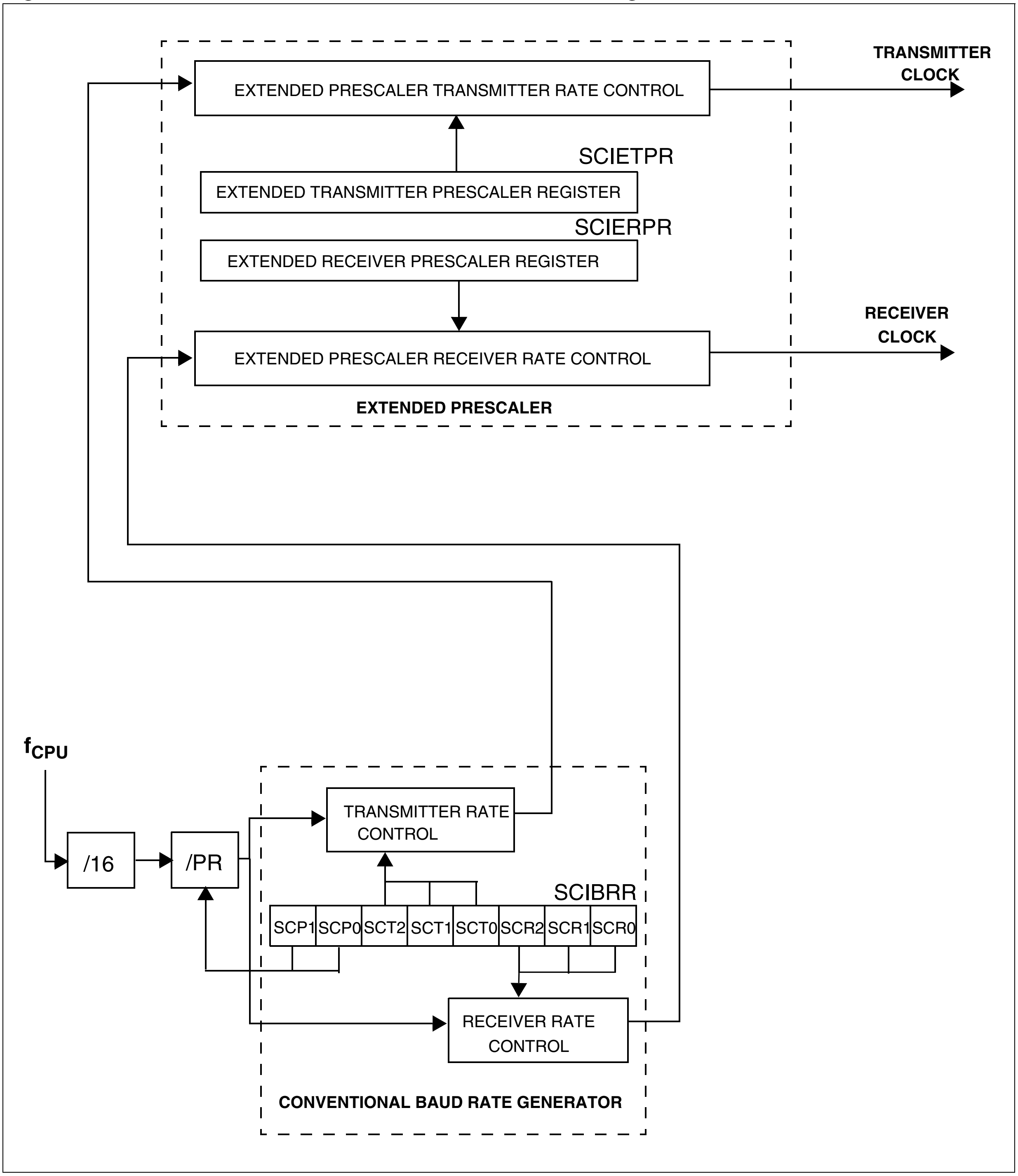
– the FE bit is set by hardware

– Data is transferred from the Shift register to the SCIDR register.

– No interrupt is generated. However this bit rises at the same time as the RDRF bit which itself generates an interrupt.

The FE bit is reset by a SCISR register read operation followed by a SCIDR register read operation.

Figure 62. SCI Baud Rate and Extended Prescaler Block Diagram



SCI SERIAL COMMUNICATION INTERFACE (Cont'd)

11.5.4.4 Conventional Baud Rate Generation

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

:

$$Tx = \frac{f_{CPU}}{(16*PR)*TR} \qquad Rx = \frac{f_{CPU}}{(16*PR)*RR}$$

with:

PR = 1, 3, 4 or 13 (see SCP[1:0] bits)

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

(see SCT[2:0] bits)

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

(see SCR[2:0] bits)

All these bits are in the SCIBRR register.

Example: If f_{CPU} is 8 MHz (normal mode) and if PR = 13 and TR = RR = 1, the transmit and receive baud rates are 38400 baud.

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

11.5.4.5 Extended Baud Rate Generation

The extended prescaler option gives a very fine tuning on the baud rate, using a 255 value prescaler, whereas the conventional Baud Rate Generator retains industry standard software compatibility.

The extended baud rate generator block diagram is shown in Figure 3.

The output clock rate sent to the transmitter or to the receiver is the output from the 16 divider divided by a factor ranging from 1 to 255 set in the SCIERPR or the SCIETPR register.

Note: The extended prescaler is activated by setting the SCIETPR or SCIERPR register to a value other than zero. The baud rates are calculated as follows:

$$Tx = \frac{f_{CPU}}{16*ETPR*(PR*TR)} \qquad Rx = \frac{f_{CPU}}{16*ERPR*(PR*RR)}$$

with:

ETPR = 1, ..., 255 (see SCIETPR register)

ERPR = 1, ..., 255 (see SCIERPR register)

11.5.4.6 Receiver Muting and Wake-up Feature

In multiprocessor configurations it is often desirable that only the intended message recipient should actively receive the full message contents, thus reducing redundant SCI service overhead for all non addressed receivers.

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

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

None of the reception status bits can be set.

All the receive interrupts are inhibited.

A muted receiver can be woken up in one of the following two ways:

– by Idle Line detection if the WAKE bit is reset,

– by Address Mark detection if the WAKE bit is set.

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

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

SCI SERIAL COMMUNICATION INTERFACE (Cont'd)

11.5.4.7 Parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the SCICR1 register. Depending on the frame length defined by the M bit, the possible SCI frame formats are as listed in Table 1.

Table 21. Frame Formats

M bit	PCE bit	SCI frame
0	0	I SB I 8 bit data I STB I
	1	I SB I 7-bit data I PB I STB I
1	0	I SB I 9-bit data I STB I
	1	I SB I 8-bit data PB I STB I

Legend:

SB: Start Bit
STB: Stop Bit
PB: Parity Bit

Note: In case of wake up by an address mark, the MSB bit of the data is taken into account and not the parity bit

Even parity: The parity bit is calculated to obtain an even number of "1s" inside the frame made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

Example: data = 00110101; 4 bits set => parity bit is 0 if even parity is selected (PS bit = 0).

Odd parity: The parity bit is calculated to obtain an odd number of "1s" inside the frame made of the 7 or 8 LSB bits (depending on whether M is equal to 0 or 1) and the parity bit.

Example: data = 00110101; 4 bits set => parity bit is 1 if odd parity is selected (PS bit = 1).

Transmission mode: If the PCE bit is set then the MSB bit of the data written in the data register is not transmitted but is changed by the parity bit.

Reception mode: If the PCE bit is set then the interface checks if the received data byte has an even number of "1s" if even parity is selected (PS = 0) or an odd number of "1s" if odd parity is selected (PS = 1). If the parity check fails, the PE flag is set in the SCISR register and an interrupt is generated if PIE is set in the SCICR1 register.

11.5.4.8 SCI Clock Tolerance

During reception, each bit is sampled 16 times. The majority of the 8th, 9th and 10th samples is considered as the bit value. For a valid bit detection, all the three samples should have the same value otherwise the noise flag (NF) is set. For example: if the 8th, 9th and 10th samples are 0, 1 and 1 respectively, then the bit value is "1", but the Noise Flag bit is set because the three samples values are not the same.

Consequently, the bit length must be long enough so that the 8th, 9th and 10th samples have the desired bit value. This means the clock frequency should not vary more than 6/16 (37.5%) within one bit. The sampling clock is resynchronized at each start bit, so that when receiving 10 bits (one start bit, 1 data byte, 1 stop bit), the clock deviation must not exceed 3.75%.

Note: The internal sampling clock of the microcontroller samples the pin value on every falling edge. Therefore, the internal sampling clock and the time the application expects the sampling to take place may be out of sync. For example: If the baud rate is 15.625 kbaud (bit length is 64µs), then the 8th, 9th and 10th samples will be at 28µs, 32µs and 36µs respectively (the first sample starting ideally at 0µs). But if the falling edge of the internal clock occurs just before the pin value changes, the samples would then be out of sync by ~4us. This means the entire bit length must be at least 40µs (36µs for the 10th sample + 4µs for synchronization with the internal sampling clock).

SCI SERIAL COMMUNICATION INTERFACE (Cont'd)

11.5.4.9 Clock Deviation Causes

The causes which contribute to the total deviation are:

– D_{TRA}: Deviation due to transmitter error (Local oscillator error of the transmitter or the transmitter is transmitting at a different baud rate).

– D_{QUANT}: Error due to the baud rate quantization of the receiver.

– D_{REC}: Deviation of the local oscillator of the receiver: This deviation can occur during the reception of one complete SCI message assuming that the deviation has been compensated at the beginning of the message.

– D_{TCL}: Deviation due to the transmission line (generally due to the transceivers)

All the deviations of the system should be added and compared to the SCI clock tolerance:

$D_{TRA} + D_{QUANT} + D_{REC} + D_{TCL} < 3.75\%$

11.5.4.10 Noise Error Causes

See also description of Noise error in Section 0.1.4.3 .

Start bit

The noise flag (NF) is set during start bit reception if one of the following conditions occurs:

1. A valid falling edge is not detected. A falling edge is considered to be valid if the three consecutive samples before the falling edge occurs are detected as '1' and, after the falling edge occurs, during the sampling of the 16 samples, if one of the samples numbered 3, 5 or 7 is detected as a "1".
2. During sampling of the 16 samples, if one of the samples numbered 8, 9 or 10 is detected as a "1".

Therefore, a valid Start Bit must satisfy both the above conditions to prevent the Noise Flag getting set.

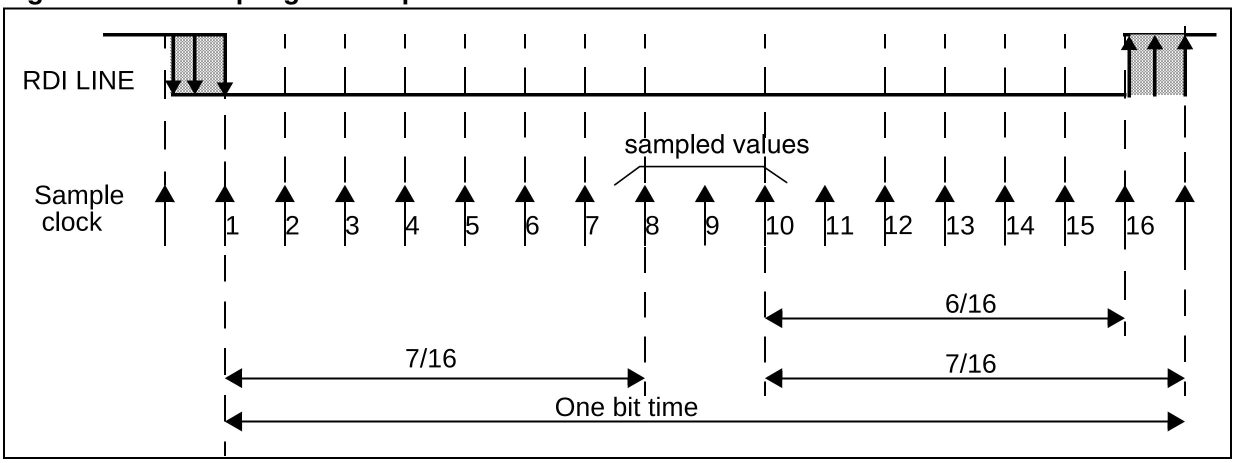
Data Bits

The noise flag (NF) is set during normal data bit reception if the following condition occurs:

– During the sampling of 16 samples, if all three samples numbered 8, 9 and 10 are not the same. The majority of the 8th, 9th and 10th samples is considered as the bit value.

Therefore, a valid Data Bit must have samples 8, 9 and 10 at the same value to prevent the Noise Flag getting set.

Figure 63. Bit Sampling in Reception Mode



SCI SERIAL COMMUNICATION INTERFACE (Cont'd)

11.5.5 Low Power Modes

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

11.5.6 Interrupts

The SCI interrupt events are connected to the same interrupt vector.

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).

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

SCI SERIAL COMMUNICATION INTERFACE (Cont'd)

11.5.7 Register Description

STATUS REGISTER (SCISR)

Read Only
Reset Value: 1100 0000 (C0h)

7							0
TDRE	TC	RDRF	IDLE	OR	NF	FE	PE

Bit 7 = **TDRE** *Transmit data register empty.*
This bit is set by hardware when the content of the TDR register has been transferred into the shift register. An interrupt is generated if the TIE bit = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a write to the SCIDR register).
0: Data is not transferred to the shift register
1: Data is transferred to the shift register

Note: Data is not transferred to the shift register until the TDRE bit is cleared.

Bit 6 = **TC** *Transmission complete.*
This bit is set by hardware when transmission of a frame containing Data is complete. An interrupt is generated if TCIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a write to the SCIDR register).
0: Transmission is not complete
1: Transmission is complete

Note: TC is not set after the transmission of a Preamble or a Break.

Bit 5 = **RDRF** *Received data ready flag.*
This bit is set by hardware when the content of the RDR register has been transferred to the SCIDR register. An interrupt is generated if RIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).
0: Data is not received
1: Received data is ready to be read

Bit 4 = **IDLE** *Idle line detect.*
This bit is set by hardware when an Idle Line is detected. An interrupt is generated if the ILIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).
0: No Idle Line is detected
1: Idle Line is detected

Note: The IDLE bit is not set again until the RDRF bit has been set itself (that is, a new idle line occurs).

Bit 3 = **OR** *Overrun error.*
This bit is set by hardware when the word currently being received in the shift register is ready to be transferred into the RDR register while RDRF = 1. An interrupt is generated if RIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).
0: No Overrun error
1: Overrun error is detected

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

Bit 2 = **NF** *Noise flag.*
This bit is set by hardware when noise is detected on a received frame. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).
0: No noise is detected
1: Noise is detected

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

Bit 1 = **FE** *Framing error.*
This bit is set by hardware when a desynchronization, excessive noise or a break character is detected. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).
0: No Framing error is detected
1: Framing error or break character is detected

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

Bit 0 = **PE** *Parity error.*
This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by a software sequence (a read to the status register followed by an access to the SCIDR data register). An interrupt is generated if PIE = 1 in the SCICR1 register.
0: No parity error
1: Parity error

SCI SERIAL COMMUNICATION INTERFACE (Cont'd)

CONTROL REGISTER 1 (SCICR1)

Read/Write

Reset Value: x000 0000 (x0h)

7							0
R8	T8	SCID	M	WAKE	PCE	PS	PIE

Bit 7 = **R8** *Receive data bit 8.*
This bit is used to store the 9th bit of the received word when M = 1.

Bit 6 = **T8** *Transmit data bit 8.*
This bit is used to store the 9th bit of the transmitted word when M = 1.

Bit 5 = **SCID** *Disabled for low power consumption*
When this bit is set the SCI prescalers and outputs are stopped and the end of the current byte transfer in order to reduce power consumption.This bit is set and cleared by software.
0: SCI enabled
1: SCI prescaler and outputs disabled

Bit 4 = **M** *Word length.*
This bit determines the word length. It is set or cleared by software.
0: 1 Start bit, 8 Data bits, 1 Stop bit
1: 1 Start bit, 9 Data bits, 1 Stop bit

Note: The M bit must not be modified during a data transfer (both transmission and reception).

Bit 3 = **WAKE** *Wake-Up method.*
This bit determines the SCI Wake-Up method, it is set or cleared by software.
0: Idle Line
1: Address Mark

Bit 2 = **PCE** *Parity control enable.*
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).
0: Parity control disabled
1: Parity control enabled

Bit 1 = **PS** *Parity selection.*
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity

Bit 0 = **PIE** *Parity interrupt enable.*
This bit enables the interrupt capability of the hardware parity control when a parity error is detected (PE bit set). It is set and cleared by software.
0: Parity error interrupt disabled
1: Parity error interrupt enabled

SCI SERIAL COMMUNICATION INTERFACE (Cont'd)

CONTROL REGISTER 2 (SCICR2)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK

Bit 7 = **TIE** *Transmitter interrupt enable*.
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SCI interrupt is generated whenever TDRE = 1 in the SCISR register

Bit 6 = **TCIE** *Transmission complete interrupt enable*
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SCI interrupt is generated whenever TC = 1 in the SCISR register

Bit 5 = **RIE** *Receiver interrupt enable*.
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SCI interrupt is generated whenever OR = 1 or RDRF = 1 in the SCISR register

Bit 4 = **ILIE** *Idle line interrupt enable.*
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SCI interrupt is generated whenever IDLE = 1 in the SCISR register.

Bit 3 = **TE** *Transmitter enable.*
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled

Notes:

– During transmission, a "0" pulse on the TE bit ("0" followed by "1") sends a preamble (idle line) after the current word.

– When TE is set there is a 1 bit-time delay before the transmission starts.

Bit 2 = **RE** *Receiver enable.*
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 = **RWU** *Receiver wake-up.*
This bit determines if the SCI is in mute mode or not. It is set and cleared by software and can be cleared by hardware when a wake-up sequence is recognized.
0: Receiver in active mode
1: Receiver in mute mode

Notes:

– Before selecting Mute mode (by setting the RWU bit) the SCI must first receive a data byte, otherwise it cannot function in Mute mode with wake-up by Idle line detection.

– In Address Mark Detection Wake-Up configuration (WAKE bit = 1) the RWU bit cannot be modified by software while the RDRF bit is set.

Bit 0 = **SBK** *Send break.*
This bit set is used to send break characters. It is set and cleared by software.
0: No break character is transmitted
1: Break characters are transmitted

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

SCI SERIAL COMMUNICATION INTERFACE (Cont'd)

DATA REGISTER (SCIDR)

Read/Write

Reset Value: Undefined

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

7							0
DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0

The Data register performs a double function (read and write) since it is composed of two registers, one for transmission (TDR) and one for reception (RDR).
The TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 1).
The RDR register provides the parallel interface between the input shift register and the internal bus (see Figure 1).

BAUD RATE REGISTER (SCIBRR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
SCP1	SCP0	SCT2	SCT1	SCT0	SCR2	SCR1	SCR0

Bits 7:6 = **SCP[1:0]** *First SCI Prescaler*
These 2 prescaling bits allow several standard clock division ranges:

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

Bits 5:3 = **SCT[2:0]** *SCI Transmitter rate divisor*
These 3 bits, in conjunction with the SCP1 and SCP0 bits define the total division applied to the bus clock to yield the transmit rate clock in conventional Baud Rate Generator mode.

TR dividing factor	SCT2	SCT1	SCT0
1	0	0	0
2			1
4		1	0
8			1
16	1	0	0
32			1
64		1	0
128			1

Note: This TR factor is used only when the ETPR fine tuning factor is equal to 00h; otherwise, TR is replaced by the (TR*ETPR) dividing factor.

Bits 2:0 = **SCR[2:0]** *SCI Receiver rate divisor.*
These 3 bits, in conjunction with the SCP1 and SCP0 bits define the total division applied to the bus clock to yield the receive rate clock in conventional Baud Rate Generator mode.

RR dividing factor	SCR2	SCR1	SCR0
1	0	0	0
2			1
4		1	0
8			1
16	1	0	0
32			1
64		1	0
128			1

Note: This RR factor is used only when the ERPR fine tuning factor is equal to 00h; otherwise, RR is replaced by the (RR*ERPR) dividing factor.

SCI SERIAL COMMUNICATION INTERFACE (Cont'd)

EXTENDED RECEIVE PRESCALER DIVISION REGISTER (SCIERPR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
ERPR 7	ERPR 6	ERPR 5	ERPR 4	ERPR 3	ERPR 2	ERPR 1	ERPR 0

Bits 7:0 = **ERPR[7:0]** *8-bit Extended Receive Prescaler Register.*

The extended Baud Rate Generator is activated when a value other than 00h is stored in this register. The clock frequency from the 16 divider (see Figure 3) is divided by the binary factor set in the SCIERPR register (in the range 1 to 255).

The extended baud rate generator is not active after a reset.

EXTENDED TRANSMIT PRESCALER DIVISION REGISTER (SCIETPR)

Read/Write

Reset Value:0000 0000 (00h)

7							0
ETPR 7	ETPR 6	ETPR 5	ETPR 4	ETPR 3	ETPR 2	ETPR 1	ETPR 0

Bits 7:0 = **ETPR[7:0]** *8-bit Extended Transmit Prescaler Register.*

The extended Baud Rate Generator is activated when a value other than 00h is stored in this register. The clock frequency from the 16 divider (see Figure 3) is divided by the binary factor set in the SCIETPR register (in the range 1 to 255).

The extended baud rate generator is not active after a reset.

Table 22. Baud Rate Selection

Symbol	Parameter	Conditions			Standard	Baud Rate	Unit
		f_{CPU}	Accuracy vs. Standard	Prescaler			
f_{Tx} f_{Rx}	Communication frequency	8 MHz	~0.16%	Conventional Mode			Hz
				TR (or RR) = 128, PR = 13	300	~300.48	
				TR (or RR) = 32, PR = 13	1200	~1201.92	
				TR (or RR) = 16, PR =13	2400	~2403.84	
				TR (or RR) = 8, PR = 13	4800	~4807.69	
				TR (or RR) = 4, PR = 13	9600	~9615.38	
				TR (or RR) = 16, PR = 3	10400	~10416.67	
				TR (or RR) = 2, PR = 13	19200	~19230.77	
				TR (or RR) = 1, PR = 13	38400	~38461.54	
			~0.79%	Extended Mode ETPR (or ERPR) = 35, TR (or RR) = 1, PR = 1	14400	~14285.71	

SERIAL COMMUNICATION INTERFACE (Cont'd)

Table 23. SCI Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0050h	**SCISR** Reset Value	TDRE 1	TC 1	RDRF 0	IDLE 0	OR 0	NF 0	FE 0	PE 0
0051h	**SCIDR** Reset Value	MSB x	x	x	x	x	x	x	LSB x
0052h	**SCIBRR** Reset Value	SCP1 0	SCP0 0	SCT2 0	SCT1 0	SCT0 0	SCR2 0	SCR1 0	SCR0 0
0053h	**SCICR1** Reset Value	R8 x	T8 0	SCID 0	M 0	WAKE 0	PCE 0	PS 0	PIE 0
0054h	**SCICR2** Reset Value	TIE 0	TCIE 0	RIE 0	ILIE 0	TE 0	RE 0	RWU 0	SBK 0
0056h	**SCIERPR** Reset Value	MSB 0	0	0	0	0	0	0	LSB 0
0057h	**SCIPETPR** Reset Value	MSB 0	0	0	0	0	0	0	LSB 0

11.6 I²C BUS INTERFACE (I2C)

11.6.1 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).

11.6.2 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

11.6.3 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

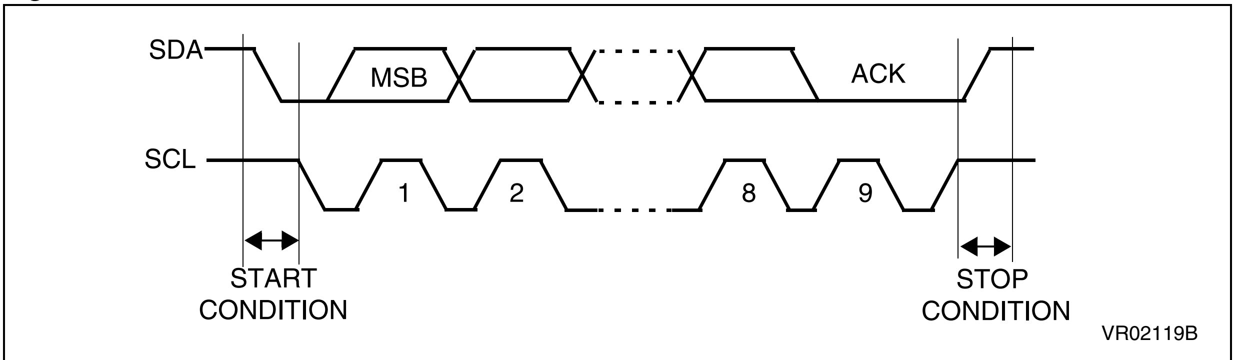
In Master mode, it initiates a data transfer and generates the clock signal. A serial data transfer always begins with a start condition and ends with a stop condition. Both start and stop conditions are generated in master mode by software.

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 64.

Figure 64. 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 (up to 100KHz) and Fast I²C (up to 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.

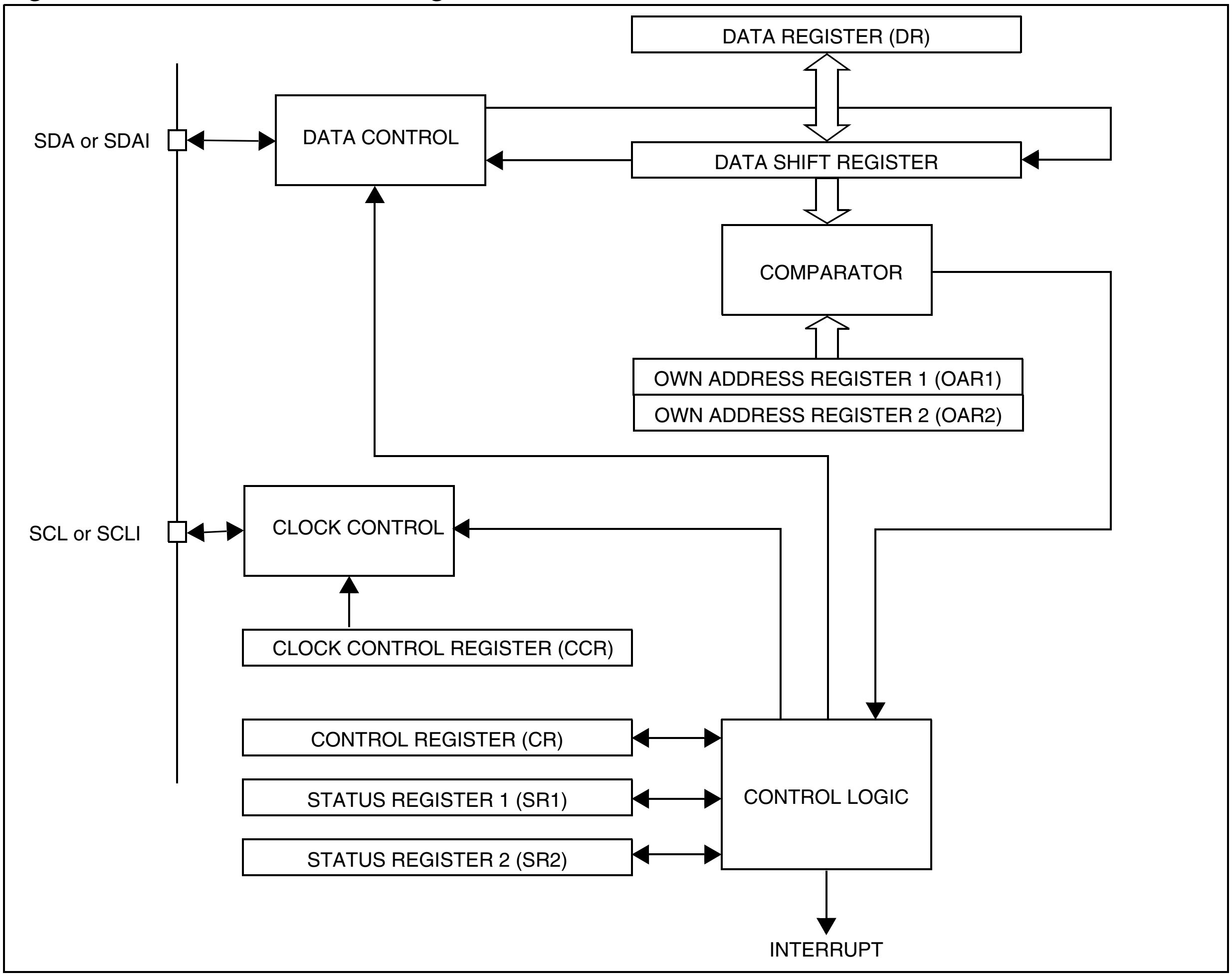
Receiver mode: the interface holds the clock line low after reception to wait for the microcontroller to read the byte in the Data Register.

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 65. I²C Interface Block Diagram



I²C BUS INTERFACE (Cont'd)

11.6.4 Functional Description

Refer to the CR, SR1 and SR2 registers in Section 11.6.7. 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.

11.6.4.1 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 66 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 66 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 66 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 66 Transfer sequencing EV4).

Error Cases

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

– **AF**: Detection of a non-acknowledge bit. In this case, the EVF and AF bits are set with an interrupt if the ITE bit is set.
The AF bit is cleared by reading the I2CSR2 register. However, if read before the completion of the transmission, the AF flag will be set again, thus possibly generating a new interrupt. Software must ensure either that the SCL line is back at 0 before reading the SR2 register, or be able

to correctly handle a second interrupt during the 9th pulse of a transmitted byte.

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

How to release the SDA / SCL lines

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

SMBus Compatibility

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

11.6.4.2 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 66 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 66 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 66 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 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 66 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 66 Transfer sequencing EV8).

When the acknowledge bit is received, the interface sets:

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

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

Error Cases

– **BERR**: Detection of a Stop or a Start condition during a byte transfer. In this case, the EVF and BERR bits are set by hardware with an interrupt if ITE is set.
  Note that BERR will not be set if an error is detected during the first pulse of each 9-bit transaction:
  Single Master Mode
  If a Start or Stop is issued during the first pulse of a 9-bit transaction, the BERR flag will not be set and transfer will continue however the BUSY flag will be reset. To work around this, slave devices should issue a NACK when they receive a misplaced Start or Stop. The reception of a NACK or BUSY by the master in the middle of communication gives the possibility to reinitiate transmission.
  Multimaster Mode
  Normally the BERR bit would be set whenever unauthorized transmission takes place while transfer is already in progress. However, an issue will arise if an external master generates an unauthorized Start or Stop while the I²C master is on the first pulse pulse of a 9-bit transaction. It is possible to work around this by polling the BUSY bit during I²C master mode transmission. The resetting of the BUSY bit can then be handled in a similar manner as the BERR flag being set.

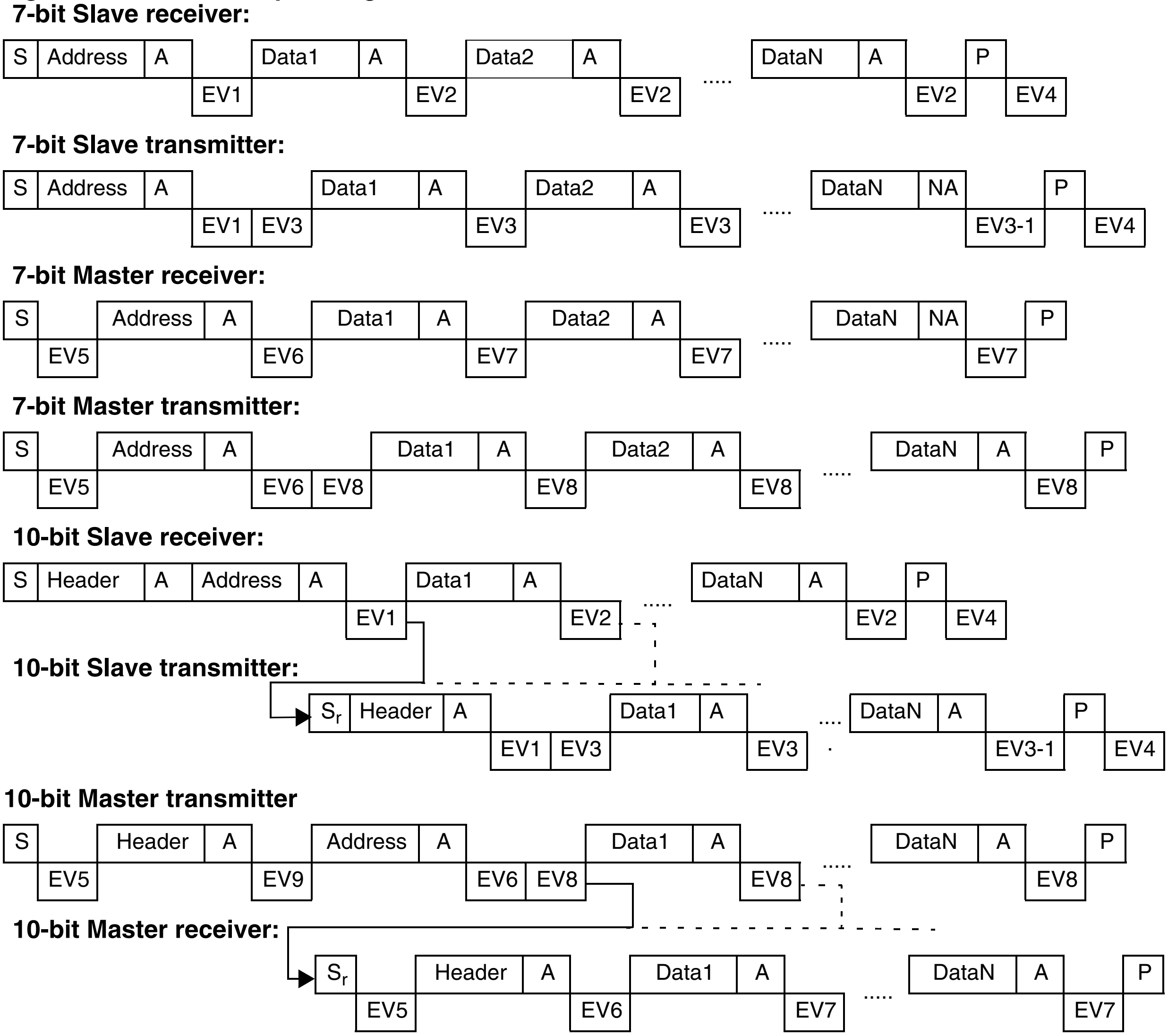
– **AF**: Detection of a non-acknowledge bit. In this case, the EVF and AF bits are set by hardware with an interrupt if the ITE bit is set. To resume, set the Start or Stop bit.
  The AF bit is cleared by reading the I2CSR2 register. However, if read before the completion of the transmission, the AF flag will be set again, thus possibly generating a new interrupt. Software must ensure either that the SCL line is back at 0 before reading the SR2 register, or be able to correctly handle a second interrupt during the 9th pulse of a transmitted byte.

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

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

I²C BUS INTERFACE (Cont'd)

Figure 66. Transfer Sequencing



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.

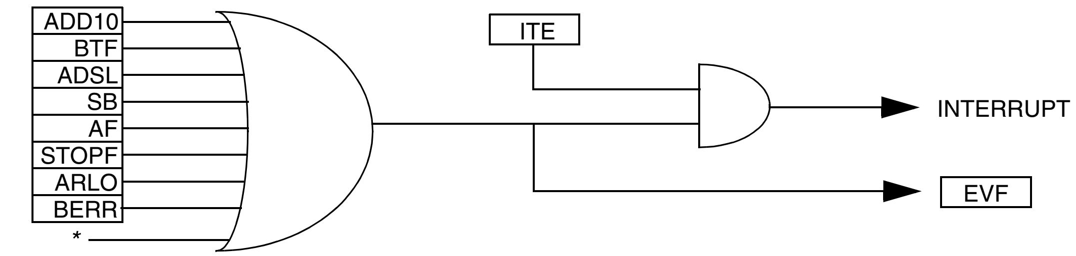
I²C BUS INTERFACE (Cont'd)

11.6.5 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.

11.6.6 Interrupts

Figure 67. 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)	ADSL		Yes	No
Start Bit Generation Event (Master mode)	SB		Yes	No
Acknowledge Failure Event	AF		Yes	No
Stop Detection Event (Slave mode)	STOPF		Yes	No
Arbitration Lost Event (Multimaster configuration)	ARLO		Yes	No
Bus Error Event	BERR		Yes	No

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)

11.6.7 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

Note: In accordance with the I2C standard, when GCAL addressing is enabled, an I2C slave can only receive data. It will not transmit data to the master.

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 67 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 66) 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 66. 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. The BUSY flag of the I2CSR1 register is cleared if a Bus Error occurs.
0: No communication on the bus
1: Communication ongoing on the bus

Bit 3 = **BTF** *Byte transfer finished.*
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 66). 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

I^2C 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

I^2C 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 but by other flags (SB or BTF) that are set at the same time.

0: No acknowledge failure
1: Acknowledge failure

Bit 3 = **STOPF** *Stop detection (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
Note:
– In a Multimaster environment, when the interface is configured in Master Receive mode it does not perform arbitration during the reception of the Acknowledge Bit. Mishandling of the ARLO bit from the I2CSR2 register may occur when a second master simultaneously requests the same data from the same slave and the I^2C master does not acknowledge the data. The ARLO bit is then left at 0 instead of being set.

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
Note:
– If a Bus Error occurs, a Stop or a repeated Start condition should be generated by the Master to re-synchronize communication, get the transmission acknowledged and the bus released for further communication

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

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

I^2C BUS INTERFACE (Cont'd)

I^2C CLOCK CONTROL REGISTER (CCR)

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

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

Bit 7 = **FM/SM** *Fast/Standard I^2C mode.*
This bit is set and cleared by software. It is not cleared when the interface is disabled (PE=0).
0: Standard I^2C mode
1: Fast I^2C mode

Bit 6:0 = **CC[6:0]** *7-bit clock divider.*
These bits select the speed of the bus (F_{SCL}) depending on the I^2C mode. They are not cleared when the interface is disabled (PE=0).

Refer to the Electrical Characteristics section for the table of values.

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

I^2C DATA REGISTER (DR)

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

7							0
D7	D6	D5	D4	D3	D2	D1	D0

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

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

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

I^2C BUS INTERFACE (Cont'd)

I^2C 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 = **ADD[7:1]** *Interface address*.
These bits define the I^2C 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 = **ADD[7:0]** *Interface address*.
These are the least significant bits of the I^2C bus address of the interface. They are not cleared when the interface is disabled (PE=0).

I^2C OWN ADDRESS REGISTER (OAR2)

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

7							0
FR1	FR0	0	0	0	ADD9	ADD8	0

Bit 7:6 = **FR[1:0]** *Frequency bits.*

These bits are set by software only when the interface is disabled (PE=0). To configure the interface to I^2C specified delays select the value corresponding to the microcontroller frequency F_{CPU}.

f_{CPU}	FR1	FR0
< 6 MHz	0	0
6 to 8 MHz	0	1

Bit 5:3 = Reserved

Bit 2:1 = **ADD[9:8]** *Interface address*.
These are the most significant bits of the I^2C bus address of the interface (10-bit mode only). They are not cleared when the interface is disabled (PE=0).

Bit 0 = Reserved.

I²C BUS INTERFACE (Cont'd)

Table 24. I²C Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0058h	**I2CCR** Reset Value	 0	 0	PE 0	ENGC 0	START 0	ACK 0	STOP 0	ITE 0
0059h	**I2CSR1** Reset Value	EVF 0	ADD10 0	TRA 0	BUSY 0	BTF 0	ADSL 0	M/SL 0	SB 0
005Ah	**I2CSR2** Reset Value	 0	 0	 0	AF 0	STOPF 0	ARLO 0	BERR 0	GCAL 0
005Bh	**I2CCCR** Reset Value	FM/SM 0	CC6 0	CC5 0	CC4 0	CC3 0	CC2 0	CC1 0	CC0 0
005Ch	**I2COAR1** Reset Value	ADD7 0	ADD6 0	ADD5 0	ADD4 0	ADD3 0	ADD2 0	ADD1 0	ADD0 0
005Dh	**I2COAR2** Reset Value	FR1 0	FR0 1	 0	 0	 0	ADD9 0	ADD8 0	 0
005Eh	**I2CDR** Reset Value	MSB 0	 0	 0	 0	 0	 0	 0	LSB 0

11.7 I2C TRIPLE SLAVE INTERFACE WITH DMA (I2C3S)

11.7.1 Introduction

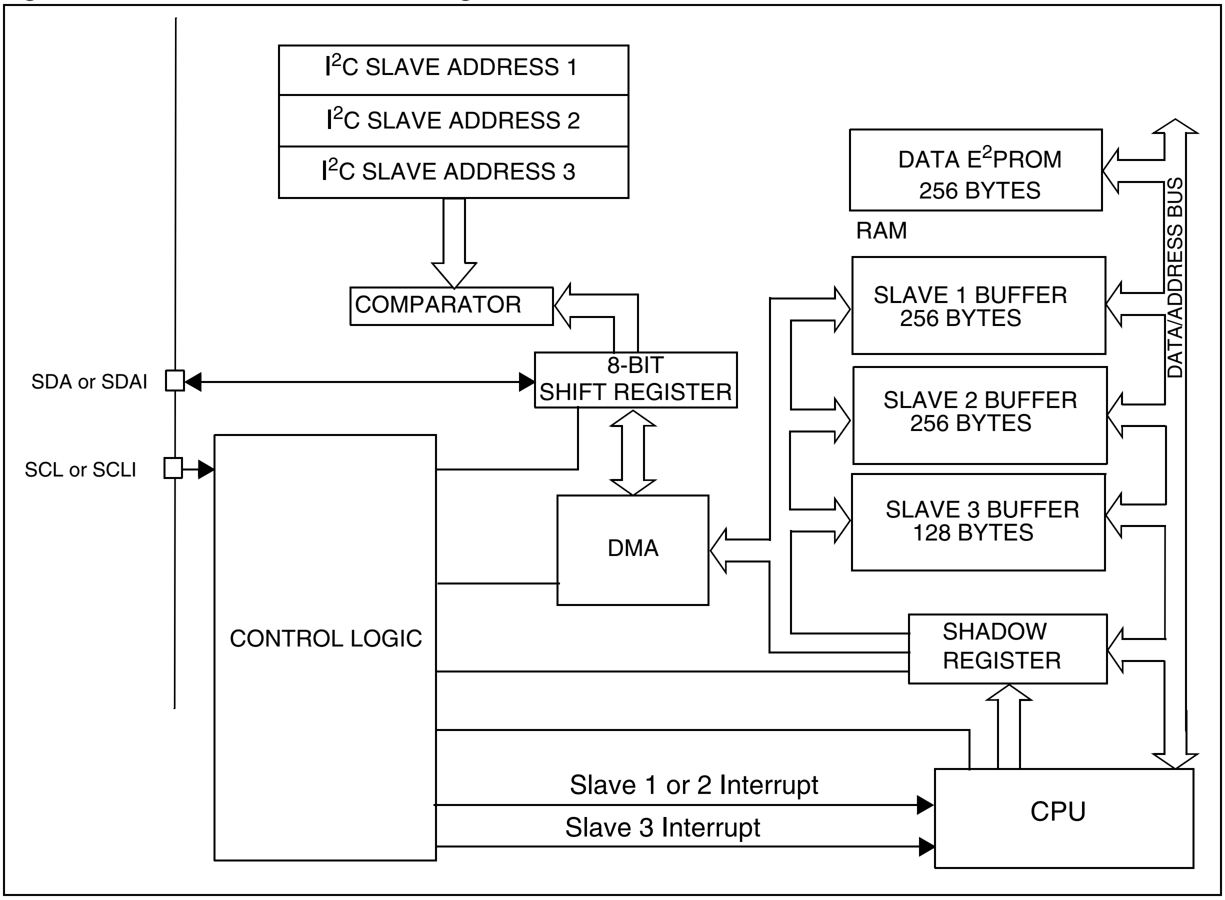
The I^2C3S interface provides three I2C slave functions, supporting both standard (up to 100kHz) and fast I^2C mode (100 to 400 kHz). Special features are provided for:

- Full-speed emulation of standard I^2C E^2PROMs
- Receiving commands to perform user-defined operations such as IAP

11.7.2 Main Features

- Three user configurable independent slave addresses can be individually enabled
- 2x 256 bytes and 1x 128 bytes buffers with fixed addresses in RAM
- 7-bit Addressing
- DMA transfer to/from I^2C bus and RAM
- Standard (transfers 256 bytes at up to 100 kHz)
- Fast Mode (transfers 256 bytes at up to 400 kHz)
- Transfer error detection and handling
- 3 interrupt flags per address for maximum flexibility
- Two interrupt request lines (one for Slaves 1 and 2, the other for Slave 3)
- Full emulation of standard I^2C EEPROMs:
  - Supports 5 read/write commands and combined format
  - No I^2C clock stretching
  - Programmable page size (8/16 bytes) or full buffer
  - Configurable write protection
- Data integrity and byte-pair coherency when reading 16-bit words from I^2C bus

Figure 68. I^2C3S Interface Block Diagram



I²C3S INTERFACE (Cont'd)

11.7.3 General Description

In addition to receiving and transmitting data, I2C3S converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The I2C3S is connected to the I²C bus by a data pin (SDA) and by a clock pin (SCL). It can be connected both with a standard I²C bus and a Fast I²C bus. The interface operates only in Slave mode as transmitter/receiver.

In order to fully emulate standard I²C EEPROM devices with highest transfer speed, the peripheral prevents I²C clock signal stretching and performs data transfer between the shift register and the RAM buffers using DMA.

11.7.3.1 Communication Flow

A serial data transfer normally begins with a start condition and ends with a stop condition. Both start and stop conditions are generated by an external master. Refer to Figure 64 for the standard protocol. The I2C3S is not a master and is not capable of generating a start/stop condition on the SDA line. The I2C3S is capable of recognising 3 slave addresses which are user programmable. The three I²C slave addresses can be individually enabled/disabled by software.

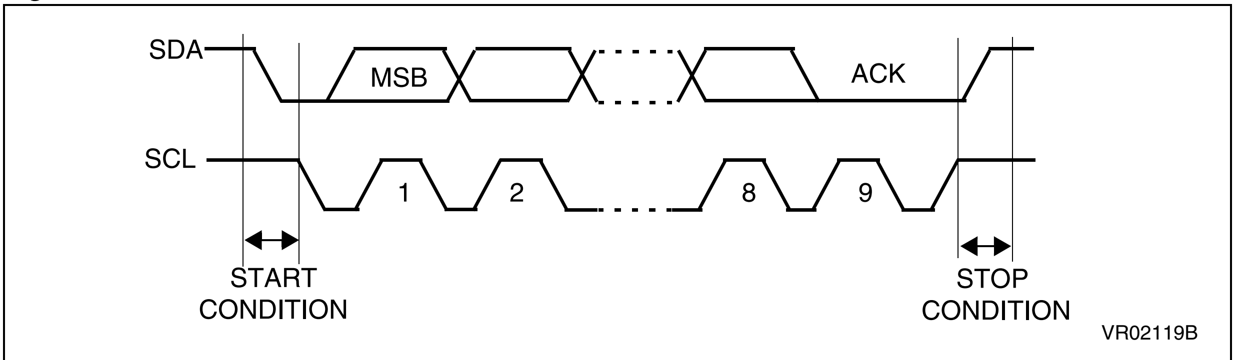
Since the I2C3S interface always acts as a slave it does not generate a clock. Data and addresses are transferred as 8-bit bytes, MSB first. The first byte following the start condition contains the slave address. A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter.

11.7.3.2 SDA/SCL Line Control

When the I2C3S interface 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 I2C3S interface is disabled, the SDA and SCL ports revert to being standard I/O port pins.

Figure 69. I²C BUS Protocol



I^2C3S INTERFACE (Cont'd)

11.7.4 Functional Description

The three slave addresses 1, 2 and 3 can be used as general purpose I^2C slaves. They also support all features of standard I^2C EEPROMs like the ST M24Cxx family and are able to fully emulate them.

Slaves 1 and 2 are mapped on the same interrupt vector. Slave 3 has a separate interrupt vector with higher priority.

The three slave addresses are defined by writing the 7 MSBs of the address in the I2C3SSAR1, I2C3SSAR2 and I2C3SSAR3 registers. The slaves are enabled by setting the enable bits in the same registers.

Each slave has its own RAM buffer at a fixed location in the ST7 RAM area.

– Slaves 1 and 2 have 256-byte buffers which can be individually protected from I^2C master write accesses.

– Slave 3 has a 128-byte RAM buffer without write protection feature.

All three slaves have individual read flags (RF) and write flags (WF) with maskable interrupts. These flags are set when the I^2C master has completed a read or write operation.

11.7.4.1 Paged operation

To allow emulation of Standard I^2C EEPROM devices, pages can be defined in the RAM buffer. The pages are configured using the PL[1:0] bits in the I2C3SCR1 register. 8/16-Byte page length has to be selected depending on the EEPROM device to emulate. The Full Page option is to be used when no paging of the RAM buffer is required. The configuration is common to the 3 slave addresses. The Full Page configuration corresponds to 256 bytes for address 1 and 2 and to 128 bytes for address 3.

Paging affects the handling of rollover when write operations are performed. In case the bottom of the page is reached, the write continues from the first address of the same page. Page length does not affect read operations: rollover is done on the whole RAM buffer whatever the configured page length.

The Byte count register is reset when it reaches 256 bytes, whatever the page length, for all slave addresses, including slave 3.

11.7.4.2 DMA

The I^2C slaves use a DMA controller to write/read data to/from their RAM buffer.

A DMA request is issued to the DMA controller on reception of a byte or just before transmission of a byte.

When a byte is written by DMA in RAM, the CPU is stalled for max. 2 cycles. When several bytes are transferred from the I2C bus to RAM, the DMA releases between each byte and the CPU resumes processing until the DMA writes the next byte.

11.7.4.3 RAM Buffer Write Protection

By setting the WP1/WP2 bits in the I2C3SCR2 register it is possible to protect the RAM buffer of Slaves 1/2 respectively against write access from the master.

If a write operation is attempted, the slave address is acknowledged, the current address register is overwritten, data is also acknowledged but it is not written to the RAM. Both the current address and byte count registers are incremented as in normal operation.

In case of write access to a write protected address, no interrupt is generated and the BusyW bit in the I2C3SCR2 register is not set.

Only write operations are disabled/enabled. Read operations are not affected.

11.7.4.4 Byte-pair coherency for I^2C Read operations

Byte-pair coherency allows the I^2C master to read a 16-bit word and ensures that it is not corrupted by a simultaneous CPU update. Two mechanisms are implemented, covering the two possible cases:

1. CPU updates a word in RAM after the first byte has been transferred to the I2C shift register from RAM. In this case, the first byte read from RAM would be the MSB of the old word and 2nd byte would be the LSB of the new word.

   To prevent this corruption, the I2C3S uses DMA to systematically read a 2-byte word when it receives a read command from the I^2C master. The MSB of the word should be at address 2n. Using DMA, the MSB is moved from RAM address 2n to the I2C shift register and the LSB from RAM address 2n+1 moved to a shadow register in the I2C3S peripheral. The CPU is stalled for a maximum of 2 cycles during word transfer.

   In case only one byte is read, the unused content of the shadow register will be automatically overwritten when a new read operation is performed.

   In case a second byte is read in the same I^2C message (no Stop or Restart condition) the content of the shadow register is transferred to the shift register and transmitted to the master.

I²C3S INTERFACE (Cont'd)

This process continues until a Stop or Restart condition occurs.

2. I2C3S attempts to read a word while the CPU is updating the RAM buffer. To prevent data corruption, the CPU must switch operation to Word mode prior to updating a word in the RAM buffer. Word mode is enabled by software using the B/W bit in the I2C3SCR2 register. In Word mode, when the CPU writes the MSB of a word to address 2n, it is stored in a shadow register rather than being actually written in RAM. When the CPU writes the second byte (the LSB) at address 2n+1, it is directly written in RAM. The next cycle after the write to address 2n+1, the MSB is automatically written from the shadow register to RAM address 2n. DMA is disabled for a 1 cycle while the CPU is writing a word.

   Word mode is disabled by hardware after the word update is performed. It must be enabled before each word update by CPU.

Use the following procedure when the ST7 writes a word in RAM:

1. Disable interrupts
2. Enable Word mode by setting the B/W and BusyW bits in the I2C3SCR2 register. BusyW bit is set to 1 when modifying any bits in Control Register 2. Writing a 1 to this bit does not actually modify BusyW but prevents accidental clearing of the bit.
3. Write Byte 1 in an even address in RAM. The byte is not actually written in RAM but in a shadow register. This address must be within the I2C RAM buffer of slave addresses 1, 2 or 3.
4. Write Byte 2 in the next higher address in RAM. This byte is actually written in RAM. During the next cycle, the shadow register content is written in the lower address. The DMA request is disabled during this cycle.
5. Byte mode resumes automatically after writing byte 2 and DMA is re-enabled.
6. Enable interrupts

Note: Word mode does not guarantee byte-pair coherency of words WRITTEN by the I2C master in RAM and read by the ST7. Byte pair coherency in this case must be handled by software.

Figure 70. 16-bit Word Write Operation Flowchart

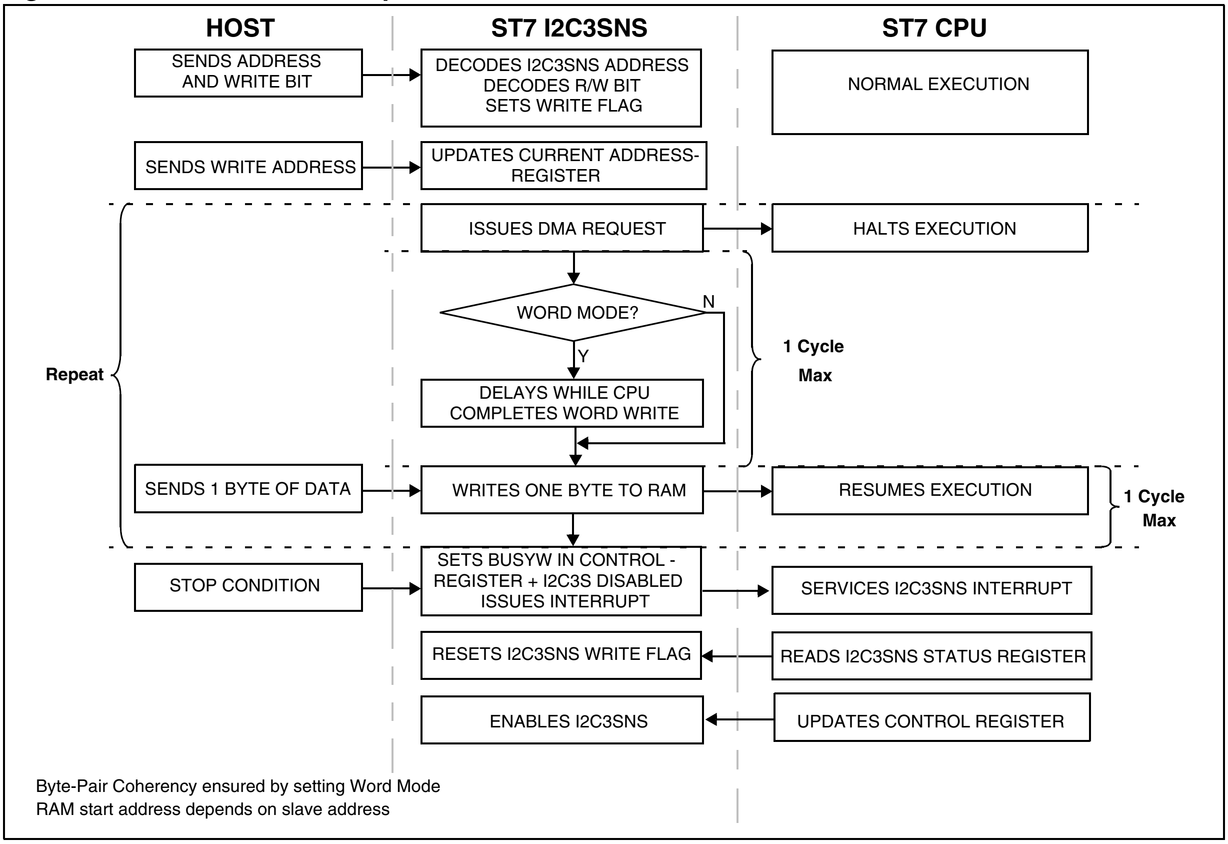
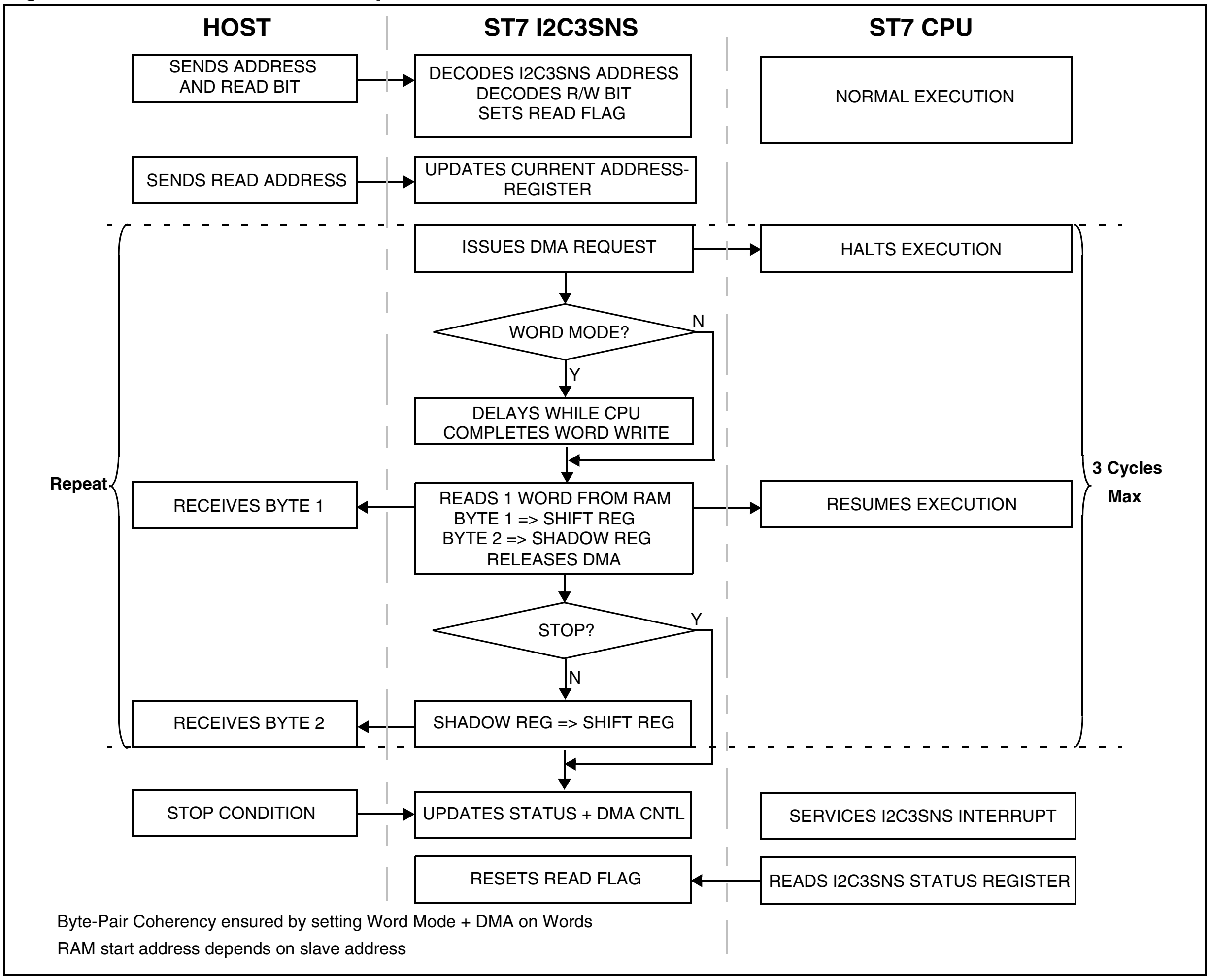


Figure 71. 16-bit Word Read Operation Flowchart



11.7.4.5 Application Note

Taking full advantage of its higher interrupt priority Slave 3 can be used to allow the addressing master to send data bytes as commands to the ST7. These commands can be decoded by the ST7 software to perform various operations such as programming the Data E2PROM via IAP (In-Application Programming).

Slave 3 writes the command byte and other data in the RAM and generates an interrupt. The ST7 then decodes the command and processes the data as decoded from the command byte. The ST7 also writes a status byte in the RAM which the addressing master can poll.

11.7.5 Address Handling

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 three addresses of the interface to decode which slave of the interface is being addressed.

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

Address matched: the interface generates in sequence the following:

– An Acknowledge pulse

– Depending on the LSB of the slave address sent by the master, slaves enter transmitter or receiver mode.

– Send an interrupt to the CPU after completion of the read/write operation after detecting the Stop/ Restart condition on the SDA line.

Notes:

– The Status Register has to be read to clear the event flag associated with the interrupt

– An interrupt will be generated only if the interrupt enable bit is set in the Control Register

– Slaves 1 and 2 have a common interrupt and the Slave 3 has a separate interrupt.

– At the end of write operation, I2C3S is temporarily disabled by hardware by setting BusyW bit in CR2. The byte count register, status register and current address register should be saved before resetting BusyW bit.

.

11.7.5.1 Slave Reception (Write operations)

Byte Write: The Slave address is followed by an 8-bit byte address. Upon receipt of this address an acknowledge is generated, address is moved into the current address register and the 8 bit data is clocked in. Once the data is shifted in, a DMA request is generated and the data is written in the RAM. The addressing device will terminate the write sequence with a stop condition. Refer to Figure 73

Page Write: A page write is initiated in similar way to a byte write, but the addressing device does not send a stop condition after the first data byte. The page length is programmed using bits 7:6 (PL[1:0]) in the Control Register1.

The current address register value is incremented by one every time a byte is written. When this address reaches the page boundary, the next byte will be written at the beginning of the same page. Refer to Figure 74.

11.7.5.2 Slave Transmission (Read Operations)

Current Address Read: The current address register maintains the last address accessed during the last read or write operation incremented by one.

During this operation the I2C slave reads the data pointed by the current address register. Refer to Figure 75.

Random Read: Random read requires a dummy byte write sequence to load in the byte address. The addressing device then generates restart condition and resends the device address similar to current address read with the read/write bit high. Refer to Figure 76. Some types of I2C masters perform a dummy write with a stop condition and then a current address read.

In either case, the slave generates a DMA request, sends an acknowledge and serially clocks out the data.

When the memory address limit is reached the current address will roll over and the random read will continue till the addressing master sends a stop condition.

Sequential Read: Sequential reads are initiated by either a current address read or a random address read. After the addressing master receives the data byte it responds with an acknowledge. As long as the slave receives an acknowledge it will continue to increment the current address register and clock out sequential data bytes.

When the memory address limit is reached the current address will roll over and the sequential read will continue till the addressing master sends a stop condition. Refer to Figure 78

11.7.5.3 Combined Format:

If a master wants to continue communication either with another slave or by changing the direction of transfer then the master would generate a restart and provide a different slave address or the same slave address with the R/W bit reversed. Refer to Figure 79**.**

I²C3S INTERFACE (Cont'd)

11.7.5.4 Rollover Handling

The RAM buffer of each slave is divided into pages whose length is defined according to PL1:0 bits in I2C3SCR1. Rollover takes place in these pages as described below.

In the case of Page Write, if the number of data bytes transmitted is more than the page length, the current address will roll over to the first byte of the current page and the previous data will be overwritten. This page size is configured using PL[1:0] bit in the I2C3SCR1 register.

In case of Sequential Read, if the current address register value reaches the memory address limit the address will roll over to the first address of the reserved area for the respective slave.

There is no status flag to indicate the roll over.

Note:

The reserved areas for slaves 1 and 2 have a limit of 256 bytes. The area for slave 3 is 128 bytes. The MSB of the address is hardwired, the addressing master therefore needs to send only an 8 bit address.

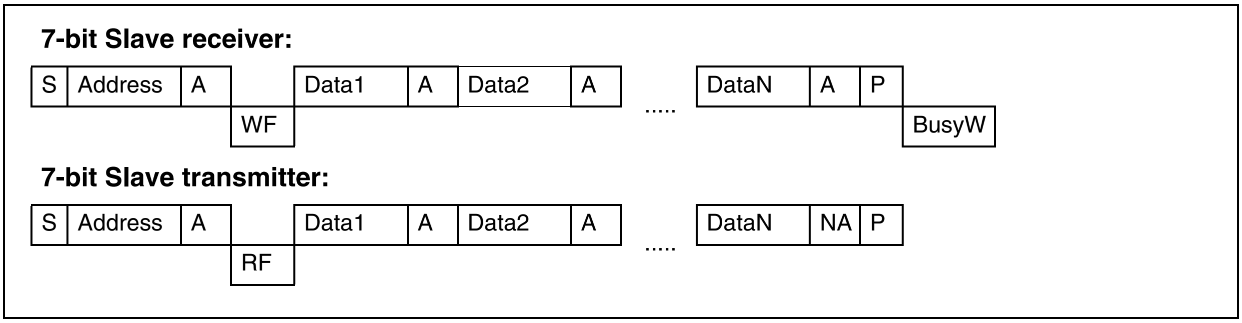
The page boundaries are defined based on page size configuration using PL[1:0] bit in the I2C3SCR1 register. If an 8-byte page size is selected, the upper 5 bits of the RAM address are fixed and the lower 3 bits are incremented. For example, if the page write starts at register address 0x0C, the write will follow the sequence 0x0C, 0x0D, 0x0E, 0x0F, 0x08, 0x09, 0x0A, 0x0B. If a 16-byte page size is selected, the upper 4 bits of the RAM address are fixed and the lower 4 bits are incremented. For example if the page write starts at register address 0x0C, the write will follow the sequence 0x0C, 0x0D, 0x0E, 0x0F, 0x00, 0x01, etc.

11.7.5.5 Error Conditions

– **BERR**: Detection of a Stop or a Start condition during a byte transfer. In this case, the BERR bit is set by hardware with an interrupt if ITER is set. During a stop condition, the interface discards the data, releases the lines and waits for another Start condition. However, a BERR on a Start condition will result in the interface discarding the data and waiting for the next slave address on the bus.

– **NACK**: Detection of a non-acknowledge bit not followed by a Stop condition. In this case, NACK bit is set by hardware with an interrupt if ITER is set.

Figure 72. Transfer Sequencing



Legend: S=Start, P=Stop, A=Acknowledge, NA=Non-acknowledge,
WF = WF event, WFx bit is set (with interrupt if ITWEx=1, after Stop or Restart conditions), cleared by reading the I2C3SSR register while no communication is ongoing.

RF = RF event, RFx is set (with interrupt if ITREx=1, after Stop or Restart conditions) , cleared by reading the I2C3SSR register while no communication is ongoing.

BusyW = BusyW flag in the I2C3CR2 register set, cleared by software writing 0.

Note: The I2C3S supports a repeated start (S_r) in place of a stop condition (P).

Figure 73. Byte Write

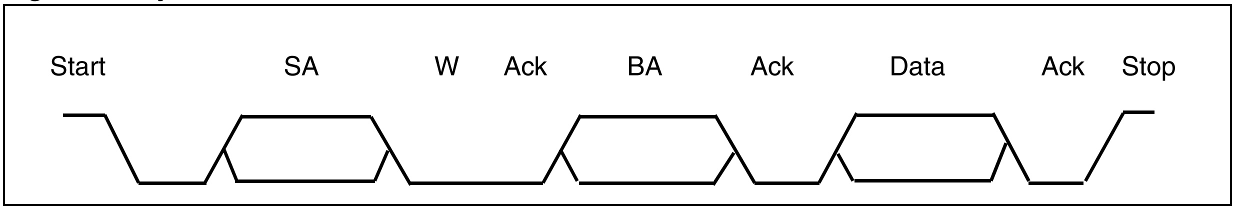


Figure 74. Page Write

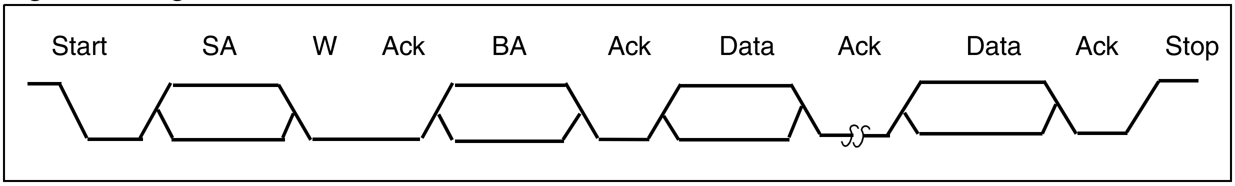


Figure 75. Current Address Read

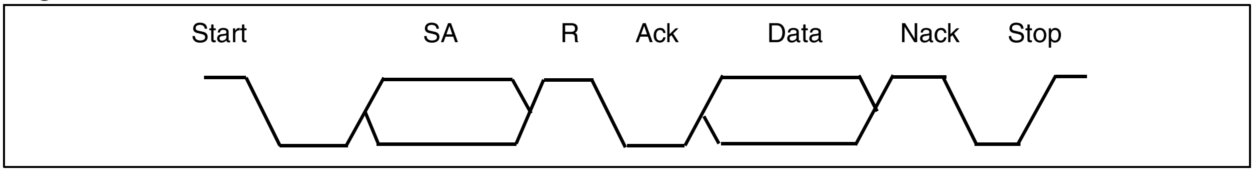


Figure 76. Random Read (Dummy write + restart + current address read)

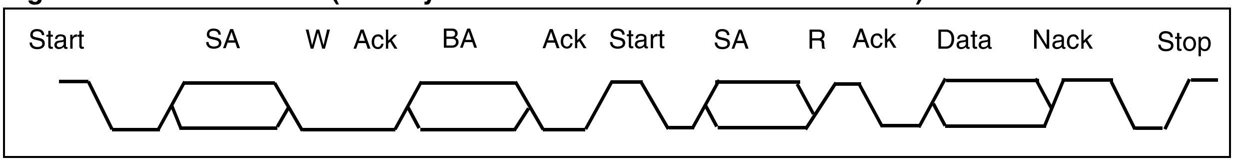


Figure 77. Random Read (Dummy write + stop + start + current address read)

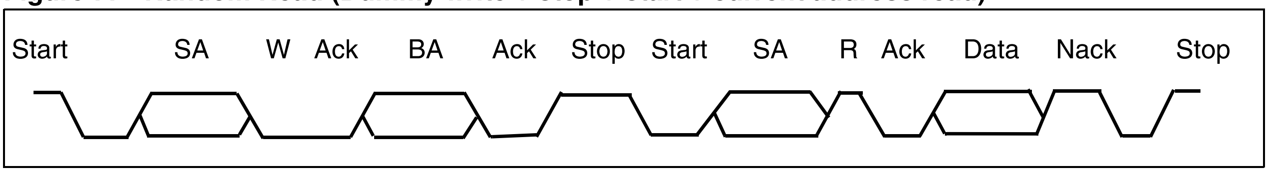


Figure 78. Sequential Read

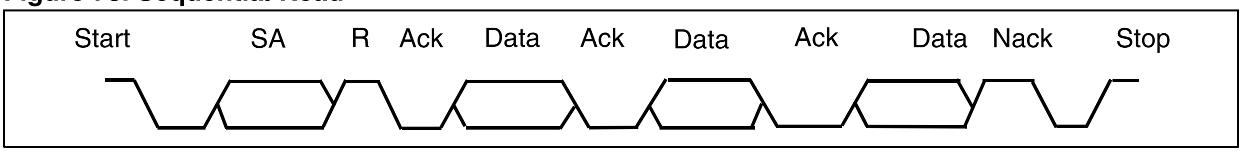
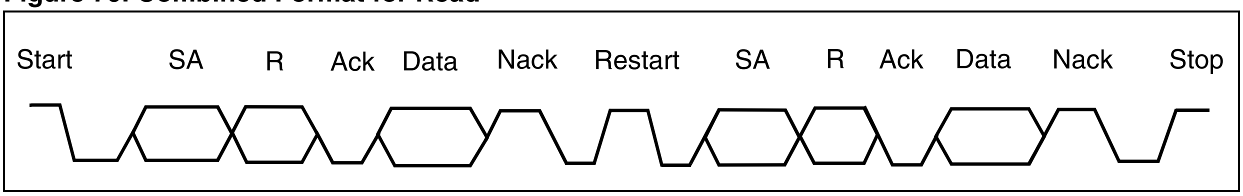


Figure 79. Combined Format for Read



Legend: SA - Slave Address W: Write

BA - Byte Address R: Read

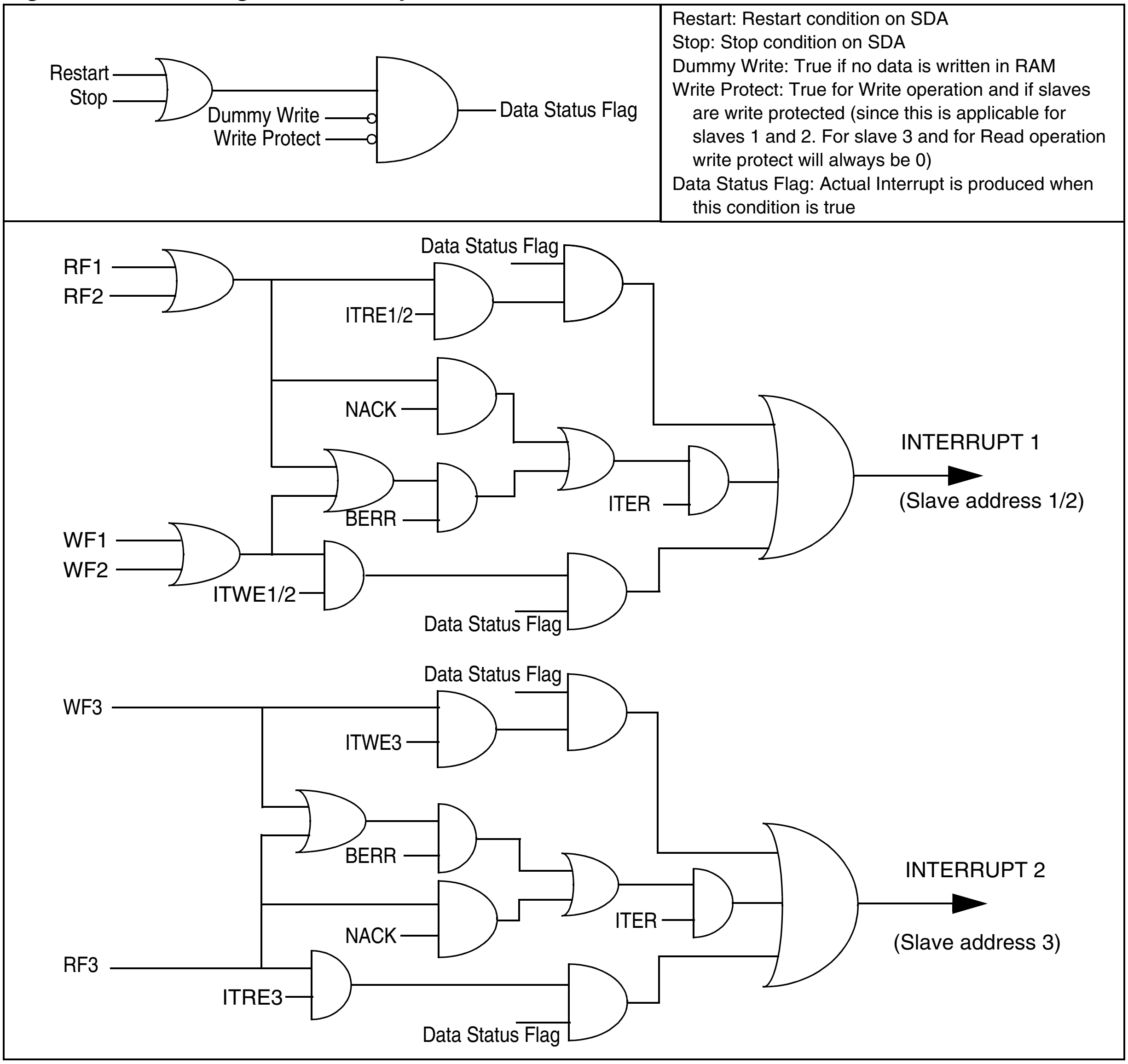
0.1.4I^2C3S INTERFACE (Cont'd)

11.7.6 Low Power Modes

Mode	Description
WAIT	No effect on I^2C interface. I2C interrupts causes the device to exit from WAIT mode.
HALT	I^2C registers are frozen. In HALT mode, the I^2C interface is inactive and does not acknowledge data on the bus. The I^2C interface resumes operation when the MCU is woken up by an interrupt with “exit from HALT mode” capability.
ACTIVE HALT	I^2C registers are frozen. In ACTIVE HALT mode, the I^2C interface is inactive and does not acknowledge data on the bus. The I^2C interface resumes operation when the MCU is woken up by an interrupt with “exit from ACTIVE HALT mode” capability.

11.7.7 Interrupt Generation

Figure 80. Event Flags and Interrupt Generation



Note: Read/Write interrupts are generated only after stop or restart conditions. Figure 80 shows the conditions for the generation of the two interrupts.

Interrupt Event	Flag	Enable Control Bit	Exit from Wait	Exit from Halt
Interrupt on write to Slave 1	WF1	ITWE1	Yes	No
Interrupt on write to Slave 2	WF2	ITWE1	Yes	No
Interrupt on write to Slave 3	WF3	ITWE2	Yes	No
Interrupt on Read from Slave 1, Slave 2 or Slave 3.	RF1- RF3	ITREx	Yes	No
Errors	BERR, NACK	ITER	Yes	No

11.7.8 Register Description

I²C 3S CONTROL REGISTER 1 (I2C3SCR1)
Read / Write
Reset Value: 0000 0000 (00h)

7							0
PL1	PL0	0	ITER	ITRE3	ITRE1/2	ITWE3	ITWE 1/2

Bits 7:6 = **PL1:0** *Page length configuration*
This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE=0).

PL1	PL0	Page length
0	0	8
0	1	16
1	0	Full Page (256 bytes for slave 1 & 2, 128 bytes for slave 3)
1	1	NA

Bit 5 = Reserved, must be kept at 0.

Bit 4 = **ITER** *BERR / NACK Interrupt enable*
This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE=0).
0: BERR / NACK interrupt disabled
1: BERR / NACK interrupt enabled

Note: In case of error, if ITER is enabled either interrupt 1 or 2 is generated depending on which slave flags the error (see Figure 80).

Bit 3= **ITRE3** *Interrupt enable on read from Slave 3*
This bit is set and cleared by software It is also cleared by hardware when interface is disabled (PE =0).
0: Interrupt on Read from Slave 3 disabled
1: Interrupt on Read from Slave 3 enabled

Bit 2 = **ITRE1/2** *Interrupt enable on read from Slave 1 or 2*
This bit is set and cleared by software It is also cleared by hardware when interface is disabled (PE =0)
0: Interrupt on Read from Slave 1 or 2 disabled
1: Interrupt on Read from Slave 1 or 2 enabled

Bit 1= **ITWE3** *Interrupt enable on write to Slave 3*
This bit is set and cleared by software. It is also cleared by hardware when interface is disabled.
0: Interrupt after write to Slave 3 disabled
1: Interrupt after write to Slave 3 enabled

Bit 0 = **ITWE1/2** *Interrupt enable on write to Slave 1 or 2*
This bit is set and cleared by software. It is also cleared by hardware when interface is disabled software. It is also cleared by hardware when when interface is disabled.
0: Interrupt after write to Slave 1 or 2 disabled
1: Interrupt after write to Slave 1 or 2 enabled

I2C CONTROL REGISTER 2 (I2C3SCR2)
Read / Write
Reset Value: 0000 0000 (00h)

7							0
0	0	0	WP2	WP1	PE	BusyW	B/W

Bits 7:5 = Reserved, must be kept at 0.

Bit 4= **WP2** *Write Protect enable for Slave 2*
This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE=0)
0: Write access to Slave 2 RAM buffer enabled
1: Write access to Slave 2 RAM buffer disabled

I²C3S INTERFACE (Cont'd)

Bit 3= **WP1** *Write Protect enable for Slave 1*
This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE=0).
0: Write access to Slave 1 RAM buffer enabled
1: Write access to Slave 1 RAM buffer disabled

Notes: (Applicable for both WP2/ WP1)

– Only write operations are disabled/enabled. Read operations are not affected.
– If a write operation is attempted, the slave address is acknowledged, the current address register is overwritten, data is also acknowledged but it is not written to the RAM.
– Both the current address and byte count registers are incremented as in normal operation.
– No interrupt generated if slave is write protected
– BusyW will not be set if slave is write protected

Bit 2= **PE** *Peripheral enable*
This bit is set and cleared by software.
0: Peripheral disabled
1: Slave capability enabled

Note: 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 1 = **BusyW** *Busy on Write to RAM Buffer*
This bit is set by hardware when a STOP/ RESTART is detected after a write operation. The I2C3S peripheral is temporarily disabled till this bit is reset. This bit is cleared by software. If this bit is not cleared before the next slave address reception, further communication will be non-acknowledged. This bit is set to 1 when modifying any bits in Control Register 2. Writing a 1 to this bit does not actually modify BusyW but prevents accidentally clearing of the bit.
0: No BusyW event occurred
1: A STOP/ RESTART is detected after a write operation

Bit 0 = **B/W** *Byte / Word Mode*
This control bit must be set by software before a word is updated in the RAM buffer and cleared by hardware after completion of the word update. In Word mode the CPU cannot be interrupted when it is modifying the LSB byte and MSB byte of the word. This mode is to ensure the coherency of data stored as words.
0: Byte mode
1: Word mode

Note: When word mode is enabled, all interrupts should be masked while the word is being written in RAM.

I²C3S STATUS REGISTER (I2C3SSR)

Read Only
Reset Value: 0000 0000 (00h)

7							0
NACK	BERR	WF3	WF2	WF1	RF3	RF2	RF1

Bit 7= **NACK** *Non Acknowledge not followed by Stop*

This bit is set by hardware when a non acknowledge returned by the master is not followed by a Stop or Restart condition. It is cleared by software reading the SR register or by hardware when the interface is disabled (PE=0).
0: No NACK error occurred
1: Non Acknowledge not followed by Stop

Bit 6 = **BERR** *Bus error*
This bit is set by hardware when the interface detects a misplaced Start or Stop condition. It is cleared by software reading SR 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 5 = **WF3** *Write operation to Slave 3*
This bit is set by hardware on reception of the direction bit in the I²C address byte for Slave 3. This bit is cleared when the status register is read when there is no communication ongoing or when the peripheral is disabled (PE = 0)
0: No write operation to Slave 3
1: Write operation performed to Slave 3

Bit 4 = **WF2** *Write operation to Slave 2*
This bit is set by hardware on reception of the direction bit in the I²C address byte for Slave 2. This bit is cleared when the status register is read when there is no communication ongoing or when the peripheral is disabled (PE = 0)
0: No write operation to Slave 2
1: Write operation performed to Slave 2

Bit 3 = **WF1** *Write operation to Slave 1*
This bit is set by hardware on reception of the direction bit in the I²C address byte for Slave 1. This bit is cleared by software when the status register is read when there is no communication ongoing

or by hardware when the peripheral is disabled (PE = 0).
0: No write operation to Slave 1
1: Write operation performed to Slave 1

I²C3S INTERFACE (Cont'd)

Bit 2 = **RF3** *Read operation from Slave 3*
This bit is set by hardware on reception of the direction bit in the I²C address byte for Slave 3. It is cleared by software reading the SR register when there is no communication ongoing. It is also cleared by hardware when the interface is disabled (PE=0).
0: No read operation from Slave 3
1: Read operation performed from Slave 3

Bit 1= **RF2** *Read operation from Slave 2*
This bit is set by hardware on reception of the direction bit in the I²C address byte for Slave 2. It is cleared by software reading the SR register when there is no communication ongoing. It is also cleared by hardware when the interface is disabled (PE=0).
0: No read operation from Slave 2
1: Read operation performed from Slave 2

Bit 0= **RF1** *Read operation from Slave 1*
This bit is set by hardware on reception of the direction bit in the I²C address byte for Slave 1. It is cleared by software reading SR register when there is no communication ongoing. It is also cleared by hardware when the interface is disabled (PE=0).
0: No read operation from Slave 1
1: Read operation performed from Slave 1

I²C BYTE COUNT REGISTER (I2C3SBCR)
Read only
Reset Value: 0000 0000 (00h)

7							0
NB7	NB6	NB5	NB4	NB3	NB2	NB1	NB0

Bits 7:0 = **NB [7:0]** *Byte Count Register*
This register keeps a count of the number of bytes received or transmitted through any of the three addresses. This byte count is reset after reception by a slave address of a new transfer and is incremented after each byte is transferred. This register is not limited by the full page length. It is also cleared by hardware when interface is disabled (PE =0).

I2C SLAVE 1 ADDRESS REGISTER (I2C3SSAR1)
Read / Write
Reset Value : 0000 0000 (00h)

7							0
ADDR 7	ADDR 6	ADDR 5	ADDR 4	ADDR 3	ADDR 2	ADDR 1	EN1

Bits 7:1 = **ADDR[7:1]** *Address of Slave 1*
This register contains the first 7 bits of Slave 1 address (excluding the LSB) and is user programmable. It is also cleared by hardware when interface is disabled (PE =0).

Bit 0= **EN1** *Enable bit for Slave Address 1*
This bit is used to enable/disable Slave Address 1. It is also cleared by hardware when interface is disabled (PE =0).
0: Slave Address 1 disabled
1: Slave Address 1 enabled

I2C SLAVE 2 ADDRESS REGISTER (I2C3SSAR2)
Read / Write
Reset Value: 0000 0000 (00h)

7							0
ADDR 7	ADDR 6	ADDR 5	ADDR 4	ADDR 3	ADDR 2	ADDR 1	EN2

Bits 7:1 = **ADDR[7:1]** *Address of Slave 2*.
This register contains the first 7 bits of Slave 2 address (excluding the LSB) and is user programmable. It is also cleared by hardware when interface is disabled (PE =0).

Bit 0= **EN2** *Enable bit for Slave Address 2*
This bit is used to enable/disable Slave Address 2. It is also cleared by hardware when interface is disabled (PE =0).
0: Slave Address 2 disabled
1: Slave Address 2 enabled

I^2C3S INTERFACE (Cont'd)

I2C SLAVE 3 ADDRESS REGISTER (I2C3SSAR3)

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

7							0
ADDR 7	ADDR 6	ADDR 5	ADDR 4	ADDR 3	ADDR 2	ADDR 1	EN3

Bit 7:1 = **ADDR[7:1]** *Address of Slave 3*
This register contains the first 7 bits of Slave 3 address (excluding the LSB) and is user programmable. It is also cleared by hardware when interface is disabled (PE =0).

Bit 0= **EN3** *Enable bit for Slave Address 3*
This bit is used to enable/disable Slave Address 3. It is also cleared by hardware when interface is disabled (PE =0).
0: Slave Address 3 disabled
1: Slave Address 3 enabled

I2C SLAVE 1 MEMORY CURRENT ADDRESS REGISTER (I2C3SCAR1)

Read only
Reset Value: 0000 0000 (00h)

7							0
CA7	CA6	CA5	CA4	CA3	CA2	CA1	CA0

Bit 7:0 = **CA[7:0]** *Current address of Slave 1 buffer*
This register contains the 8 bit offset of Slave Address 1 reserved area in RAM. It is also cleared by hardware when interface is disabled (PE =0).

I2C SLAVE 2 MEMORY CURRENT ADDRESS REGISTER (I2C3SCAR2)

Read only
Reset Value: 0000 0000 (00h)

7							0
CA7	CA6	CA5	CA4	CA3	CA2	CA1	CA0

Bit 7:0 = **CA[7:0]** *Current address of Slave 2 buffer*
This register contains the 8-bit offset of Slave Address 2 reserved area in RAM. It is also cleared by hardware when interface is disabled (PE =0).

I2C SLAVE 3 MEMORY CURRENT ADDRESS REGISTER (I2C3SCAR3)

Read only
Reset Value: 0000 0000 (00h)

7							0
CA7	CA6	CA5	CA4	CA3	CA2	CA1	CA0

Bit 6:0 = **CA[6:0]** *Current address of Slave 3 buffer*
This register contains the 8-bit offset of slave address 3 reserved area in RAM. It is also cleared by hardware when interface is disabled (PE =0).

Note: Slave address 3 can store only 128 bytes. For slave address 3, CA7 bit will remain 0. i.e. if the Byte Address sent is 0x80 then the Current Address register will hold the value 0x00 due to an overflow.

Table 25. I^2C3S Register Map

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
0060h	**I2C3SCR1**	PL1	PL0	0	ITER	ITRE3	ITRE1/2	ITWE3	ITWE1/2
0061h	**I2C3SCR2**	0	0	0	WP2	WP1	PE	BusyW	B/W
0062h	**I2C3SSR**	NACK	BERR	WF3	WF2	WF1	RF3	RF2	RF1
0063h	**I2C3SBCR**	NB7	NB6	NB5	NB4	NB3	NB2	NB1	NB1
0064h	**I2C3SSAR1**	ADDR7	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	EN1
0065h	**I2C3SCAR1**	CA 7 .. CA0							
0066h	**I2C3SSAR2**	ADDR7	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	EN2
0067h	**I2C3SCAR2**	CA 7 .. CA0							
0068h	**I2C3SSAR3**	ADDR7	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	EN3
0069h	**I2C3SCAR3**	CA 7 .. CA0							

11.8 10-BIT A/D CONVERTER (ADC)

11.8.1 Introduction

The on-chip Analog to Digital Converter (ADC) peripheral is a 10-bit, successive approximation converter with internal sample and hold circuitry. This peripheral has up to 16 multiplexed analog input channels (refer to device pin out description) that allow the peripheral to convert the analog voltage levels from up to 16 different sources.

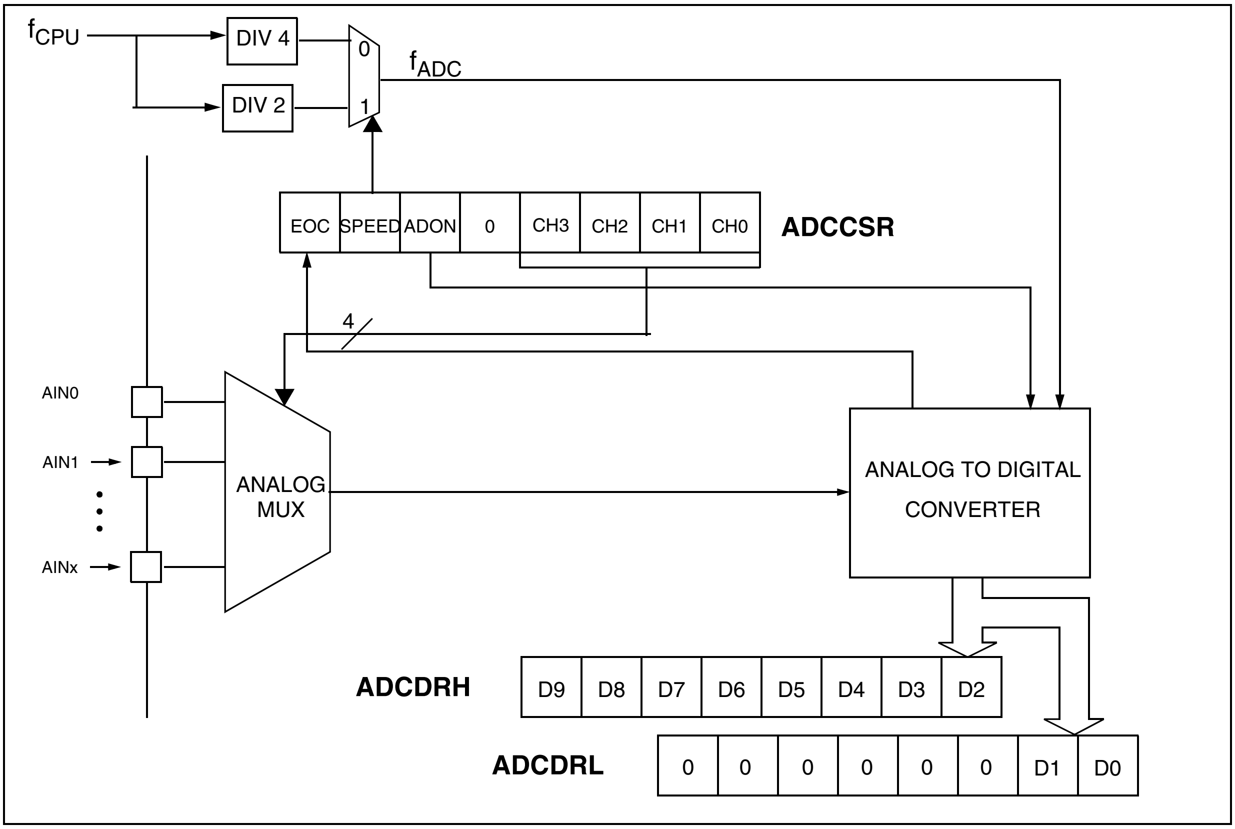
The result of the conversion is stored in a 10-bit Data Register. The A/D converter is controlled through a Control/Status Register.

11.8.2 Main Features

- 10-bit conversion
- Up to 16 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 81.

Figure 81. ADC Block Diagram



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

11.8.3 Functional Description

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

If the input voltage (V_{AIN}) is greater than V_{AREF} (high-level voltage reference) then the conversion result is FFh in the ADCDRH register and 03h in the ADCDRL register (without overflow indication).

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

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

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

11.8.3.1 A/D Converter Configuration

The analog input ports must be configured as input, no pull-up, no interrupt. Refer to the «I/O ports» chapter. Using these pins as analog inputs does not affect the ability of the port to be read as a logic input.

In the ADCCSR register:

– Select the CS[3:0] bits to assign the analog channel to convert.

11.8.3.2 Starting the Conversion

In the ADCCSR register:

– Set the ADON bit to enable the A/D converter and to start the conversion. From this time on, the ADC performs a continuous conversion of the selected channel.

When a conversion is complete:

– The EOC bit is set by hardware.

– The result is in the ADCDR registers.

A read to the ADCDRH resets the EOC bit.

To read the 10 bits, perform the following steps:

1. Poll the EOC bit
2. Read the ADCDRL register
3. Read the ADCDRH register. This clears EOC automatically.

Note: The data is not latched, so both the low and the high data register must be read before the next conversion is complete, so it is recommended to disable interrupts while reading the conversion result.

To read only 8 bits, perform the following steps:

1. Poll the EOC bit
2. Read the ADCDRH register. This clears EOC automatically.

11.8.3.3 Changing the conversion channel

The application can change channels during conversion. When software modifies the CH[3:0] bits in the ADCCSR register, the current conversion is stopped, the EOC bit is cleared, and the A/D converter starts converting the newly selected channel.

11.8.4 Low Power Modes

Note: The A/D converter may be disabled by resetting the ADON bit. This feature allows reduced power consumption when no conversion is needed.

Mode	Description
WAIT	No effect on A/D Converter
HALT	A/D Converter disabled. After wakeup from Halt mode, the A/D Converter requires a stabilization time t_{STAB} (see Electrical Characteristics) before accurate conversions can be performed.

11.8.5 Interrupts

None.

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

11.8.6 Register Description

CONTROL/STATUS REGISTER (ADCCSR)

Read/Write (Except bit 7 read only)

Reset Value: 0000 0000 (00h)

7							0
EOC	SPEED	ADON	0	CH3	CH2	CH1	CH0

Bit 7 = **EOC** *End of Conversion*
This bit is set by hardware. It is cleared by hardware when software reads the ADCDRH register or writes to any bit of the ADCCSR register.
0: Conversion is not complete
1: Conversion complete

Bit 6 = **SPEED** *ADC clock selection*
This bit is set and cleared by software.
0: $f_{ADC} = f_{CPU}/4$
1: $f_{ADC} = f_{CPU}/2$

Bit 5 = **ADON** *A/D Converter on*
This bit is set and cleared by software.
0: Disable ADC and stop conversion
1: Enable ADC and start conversion

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

Bits 3:0 = **CH[3:0]** *Channel Selection*
These bits are set and cleared by software. They select the analog input to convert.

Channel Pin*	CH3	CH2	CH1	CH0
AIN0	0	0	0	0
AIN1	0	0	0	1
AIN2	0	0	1	0
AIN3	0	0	1	1
AIN4	0	1	0	0
AIN5	0	1	0	1
Reserved	0	1	1	0
Reserved	0	1	1	1
AIN8	1	0	0	0
Reserved	1	0	0	1
AIN10	1	0	1	0
Reserved	1	0	1	1
AIN12	1	1	0	0
AIN13	1	1	0	1
AIN14	1	1	1	0
AIN15	1	1	1	1

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

DATA REGISTER (ADCDRH)

Read Only

Reset Value: 0000 0000 (00h)

7							0
D9	D8	D7	D6	D5	D4	D3	D2

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

DATA REGISTER (ADCDRL)

Read Only

Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	D1	D0

Bits7:2 = Reserved. Forced by hardware to 0.

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

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

Table 26. ADC Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0070h	**ADCCSR** Reset Value	EOC 0	SPEED 0	ADON 0	0	CH3 0	CH2 0	CH1 0	CH0 0
0071h	**ADCDRH** Reset Value	D9 0	D8 0	D7 0	D6 0	D5 0	D4 0	D3 0	D2 0
0072h	**ADCDRL** Reset Value	0	0	0	0	0	0	D1 0	D0 0

12 INSTRUCTION SET

12.1 ST7 ADDRESSING MODES

The ST7 Core features 17 different addressing modes which can be classified in seven 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 submodes called long and short:

– Long addressing mode is more powerful because it can use the full 64 Kbyte address space, however it uses more bytes and more CPU cycles.

– Short addressing mode is less powerful because it can generally only access page zero (0000h - 00FFh range), but the instruction size is more compact, and faster. All memory to memory instructions use short addressing modes only (CLR, CPL, NEG, BSET, BRES, BTJT, BTJF, INC, DEC, RLC, RRC, SLL, SRL, SRA, SWAP)

The ST7 Assembler optimizes the use of long and short addressing modes.

Table 27. 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)

12.1.1 Inherent

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

Inherent Instruction	Function
NOP	No operation
TRAP	S/W Interrupt
WFI	Wait For Interrupt (Low Power Mode)
HALT	Halt Oscillator (Lowest Power Mode)
RET	Subroutine Return
IRET	Interrupt Subroutine Return
SIM	Set Interrupt Mask
RIM	Reset Interrupt Mask
SCF	Set Carry Flag
RCF	Reset Carry Flag
RSP	Reset Stack Pointer
LD	Load
CLR	Clear
PUSH/POP	Push/Pop to/from the stack
INC/DEC	Increment/Decrement
TNZ	Test Negative or Zero
CPL, NEG	1 or 2 Complement
MUL	Byte Multiplication
SLL, SRL, SRA, RLC, RRC	Shift and Rotate Operations
SWAP	Swap Nibbles

12.1.2 Immediate

Immediate instructions have 2 bytes, the first byte contains the opcode, the second byte contains the operand value.

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

12.1.3 Direct

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

The direct addressing mode consists of two submodes:

Direct (Short)

The address is a byte, thus requires only 1 byte after the opcode, but only allows 00 - FF addressing space.

Direct (Long)

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

12.1.4 Indexed (No Offset, Short, Long)

In this mode, the operand is referenced by its memory address, which is defined by the unsigned addition of an index register (X or Y) with an offset.

The indirect addressing mode consists of three submodes:

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 1 byte after the opcode and allows 00 - 1FE addressing space.

Indexed (Long)

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

12.1.5 Indirect (Short, Long)

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

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

Indirect (Short)

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

Indirect (Long)

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

ST7 ADDRESSING MODES (Cont'd)

12.1.6 Indirect Indexed (Short, Long)

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

The indirect indexed addressing mode consists of two submodes:

Indirect Indexed (Short)

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

Indirect Indexed (Long)

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

Table 28. Instructions Supporting Direct, Indexed, Indirect and Indirect Indexed Addressing Modes

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

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

12.1.7 Relative Mode (Direct, Indirect)

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

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

The relative addressing mode consists of two submodes:

Relative (Direct)

The offset follows the opcode.

Relative (Indirect)

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

12.2 INSTRUCTION GROUPS

The ST7 family devices use an Instruction Set consisting of 63 instructions. The instructions may be subdivided into 13 main groups as illustrated in the following table:

Load and Transfer	LD	CLR						
Stack operation	PUSH	POP	RSP					
Increment/Decrement	INC	DEC						
Compare and Tests	CP	TNZ	BCP					
Logical operations	AND	OR	XOR	CPL	NEG			
Bit Operation	BSET	BRES						
Conditional Bit Test and Branch	BTJT	BTJF						
Arithmetic operations	ADC	ADD	SUB	SBC	MUL			
Shift and Rotates	SLL	SRL	SRA	RLC	RRC	SWAP	SLA	
Unconditional Jump or Call	JRA	JRT	JRF	JP	CALL	CALLR	NOP	RET
Conditional Branch	JRxx							
Interruption management	TRAP	WFI	HALT	IRET				
Condition Code Flag modification	SIM	RIM	SCF	RCF				

Using a prebyte

The instructions are described with 1 to 4 bytes.

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

The whole instruction becomes:

PC-2 End of previous instruction

PC-1 Prebyte

PC Opcode

PC+1 Additional word (0 to 2) according to the number of bytes required to compute the effective address

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

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

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

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

12.2.1 Illegal Opcode Reset

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

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

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	

13 ELECTRICAL CHARACTERISTICS

13.1 PARAMETER CONDITIONS

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

13.1.1 Minimum and Maximum values

Unless otherwise specified the minimum and maximum values are guaranteed in the worst conditions of ambient temperature, supply voltage and frequencies by tests in production on 100% of the devices with an ambient temperature at T_A=25°C and T_A=$T_A$max (given by the selected temperature range).

Data based on characterization results, design simulation and/or technology characteristics are indicated in the table footnotes and are not tested in production. Based on characterization, the minimum and maximum values refer to sample tests and represent the mean value plus or minus three times the standard deviation (mean±3Σ).

13.1.2 Typical values

Unless otherwise specified, typical data are based on T_A=25°C, V_{DD}=5V (for the 4.5V≤V_{DD}≤5.5V voltage range) and V_{DD}=3.3V (for the 3V≤V_{DD}≤3.6V voltage range). They are given only as design guidelines and are not tested.

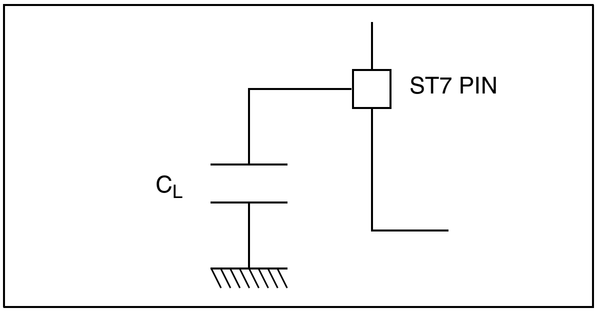
13.1.3 Typical curves

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

13.1.4 Loading capacitor

The loading conditions used for pin parameter measurement are shown in Figure 82.

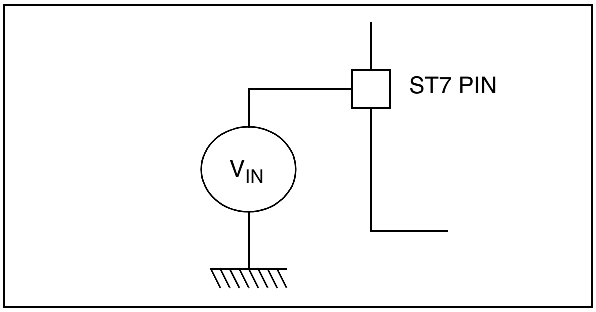
Figure 82. Pin loading conditions



13.1.5 Pin input voltage

The input voltage measurement on a pin of the device is described in Figure 83.

Figure 83. Pin input voltage



13.2 ABSOLUTE MAXIMUM RATINGS

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

13.2.1 Voltage Characteristics

Symbol	Ratings	Maximum value	Unit
V_{DD} - V_{SS}	Supply voltage	7.0	V
V_{IN}	Input voltage on any pin 1) & 2)	V_{SS}-0.3 to V_{DD}+0.3	
$V_{ESD(HBM)}$	Electrostatic discharge voltage (Human Body Model)	see Section 13.9.3 on page 165	

13.2.2 Current Characteristics

Symbol	Ratings	Maximum value	Unit
I_{VDD}	Total current into V_{DD} power lines (source) 3)	75	mA
I_{VSS}	Total current out of V_{SS} ground lines (sink) 3)	150	
I_{IO}	Output current sunk by any standard I/O and control pin	20	
	Output current sunk by any high sink I/O pin	40	
	Output current source by any I/Os and control pin	- 25	
$I_{INJ(PIN)}$ 2) & 4)	Injected current on ISPSEL pin	± 5	
	Injected current on $\overline{RESET}$ pin	± 5	
	Injected current on OSC1 and OSC2 pins	± 5	
	Injected current on PB0 pin 5)	+5	
	Injected current on any other pin 6)	± 5	
$\Sigma I_{INJ(PIN)}$ 2)	Total injected current (sum of all I/O and control pins) 6)	± 20	

13.2.3 Thermal Characteristics

Symbol	Ratings	Value	Unit
T_{STG}	Storage temperature range	-65 to +150	°C
T_J	Maximum junction temperature (see Table on page 180)		

Notes:

1. Directly connecting the $\overline{RESET}$ and I/O pins to V_{DD} or V_{SS} could damage the device if an unintentional internal reset is generated or an unexpected change of the I/O configuration occurs (for example, due to a corrupted program counter). To guarantee safe operation, this connection has to be done through a pull-up or pull-down resistor (typical: 4.7kΩ for $\overline{RESET}$, 10kΩ for I/Os). Unused I/O pins must be tied in the same way to V_{DD} or V_{SS} according to their reset configuration.

2. $I_{INJ(PIN)}$ must never be exceeded. This is implicitly insured if V_{IN} maximum is respected. If V_{IN} maximum cannot be respected, the injection current must be limited externally to the $I_{INJ(PIN)}$ value. A positive injection is induced by $V_{IN}>V_{DD}$ while a negative injection is induced by $V_{IN}<V_{SS}$. For true open-drain pads, there is no positive injection current, and the corresponding V_{IN} maximum must always be respected

3. All power (V_{DD}) and ground (V_{SS}) lines must always be connected to the external supply.

4. Negative injection disturbs the analog performance of the device. In particular, it induces leakage currents throughout the device including the analog inputs. To avoid undesirable effects on the analog functions, care must be taken:
- Analog input pins must have a negative injection less than 0.8 mA (assuming that the impedance of the analog voltage is lower than the specified limits)
- Pure digital pins must have a negative injection less than 1.6mA. In addition, it is recommended to inject the current as far as possible from the analog input pins.

5. No negative current injection allowed on PB0 pin.

6. When several inputs are submitted to a current injection, the maximum $\Sigma I_{INJ(PIN)}$ is the absolute sum of the positive and negative injected currents (instantaneous values). These results are based on characterisation with $\Sigma I_{INJ(PIN)}$ maximum current injection on four I/O port pins of the device.

13.3 OPERATING CONDITIONS

13.3.1 General Operating Conditions

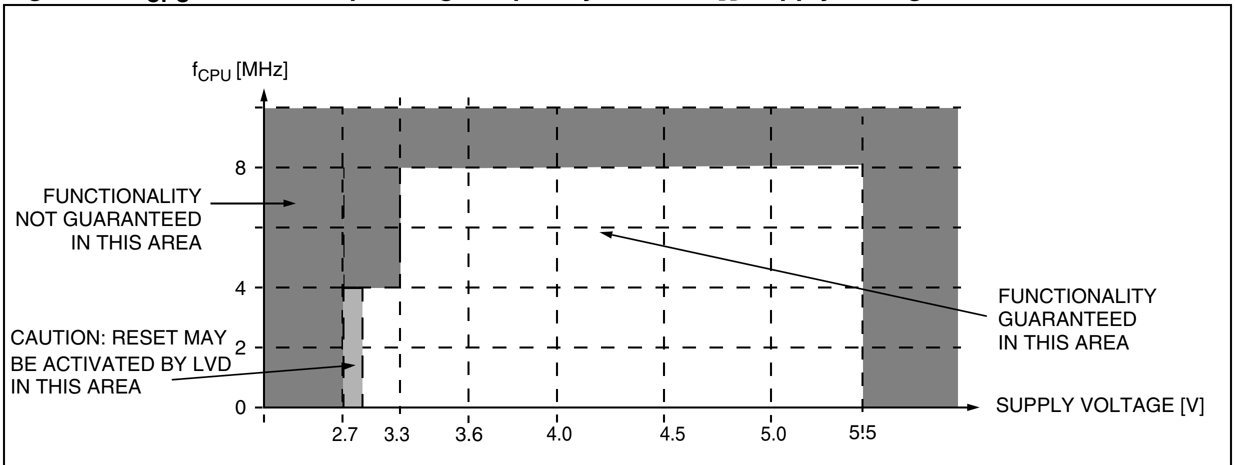
T_A = -40 to +85°C unless otherwise specified.

Symbol	Parameter	Conditions	Min	Max	Unit
V_{DD}	Supply voltage	f_{CPU} = 8 MHz. max.	3.3	5.5	V
		f_{CPU} = 4 MHz. max.	2.7	5.5	
f_{OSC}	External clock frequency	3.3V≤ V_{DD}≤5.5V	Up to 16		MHz
		2.7V≤V_{DD}<3.3V	Up to 8		

Note:

When the power supply is between 2.7 and 2.95V ($V_{IT+(LVD)}$ max), the device is either in the guaranteed functional area or in reset state, thus allowing deterministic application behaviour. However the LVD may generate a reset below 2.95V and the user should therefore not use the device below this level when the LVD is enabled.

Figure 84. f_{CPU} Maximum Operating Frequency Versus V_{DD} Supply Voltage



13.3.2 Low Voltage Detector (LVD) Thresholds

T_A = -40 to +85°C unless otherwise specified

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{IT+(LVD)}$	Reset release threshold (V_{DD} rise)	High Threshold Med. Threshold Low Threshold	3.85 3.24 2.60	4.20 3.56 2.88	4.61 3.90 3.14	V
$V_{IT-(LVD)}$	Reset generation threshold (V_{DD} fall)	High Threshold Med. Threshold Low Threshold	3.66 3.04 2.45	3.98 3.36 2.71	4.36 3.66 2.95	
$V_{hys(LVD)}$	LVD voltage threshold hysteresis	$V_{IT+(LVD)}$-$V_{IT-(LVD)}$		200		mV
Vt_{POR}	V_{DD} rise time rate		20 [1)]		100 [1)]	ms/V
$t_{g(VDD)}$	V_{DD} glitches filtered by LVD			150		ns

Note:

1. Not tested in production, guaranteed by design

13.3.3 Auxiliary Voltage Detector (AVD) Thresholds

T_A = -40 to +85°C unless otherwise specified

Symbol	Parameter	Conditions	Min [1)]	Typ	Max [1)]	Unit
$V_{IT+(AVD)}$	1=>0 AVDF flag toggle threshold (V_{DD} rise)	High Threshold Med. Threshold Low Threshold	4.15 3.64 3.00	4.50 3.96 3.28	4.91 4.30 3.54	V
$V_{IT-(AVD)}$	0=>1 AVDF flag toggle threshold (V_{DD} fall)	High Threshold Med. Threshold Low Threshold	3.96 3.44 2.85	4.28 3.76 3.11	4.66 4.06 3.35	
$V_{hys(AVD)}$	AVD voltage threshold hysteresis	$V_{IT+(AVD)}$-$V_{IT-(AVD)}$		200		mV
ΔV_{IT-}	Voltage drop between ADV flag set and LVD reset activated	$V_{IT-(AVD)}$-$V_{IT-(LVD)}$		450		mV

Note:

1. Not tested in production, guaranteed by characterization.

13.4 PLL CHARACTERISTICS

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{PLLIN}	PLL Input frequency [2]	V_{DD} = 2.7 to 3.65V PLL option x4 selected	0.95	1	1.05	MHz
		V_{DD} = 3.3 to 5.5V PLL option x8 selected	0.90	1	1.10	
$V_{DD(PLL)}$	PLL operating range	PLL option x4 selected [1]	2.7		3.65	V
		PLL option x8 selected	3.3		5.5	
$t_{w(JIT)}$	PLL jitter period	f_{RC} = 1MHz		8		kHz
JIT_{PLL}	PLL jitter ($\Delta f_{CPU}/f_{CPU}$)	VDD = 3.0V		3.0		%
		VDD = 5.0V		1.6		
$I_{DD(PLL)}$	PLL current consumption	T_A=25°C		600		µA

Note:

1. To obtain a x4 multiplication ratio in the range 3.3 to 5.5V, the DIV2EN option bit must enabled.

2. Guaranteed by design.

13.4.1 Internal RC Oscillator and PLL

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

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{DD(RC)}$	Internal RC Oscillator operating voltage	Refer to operating range of V_{DD} with T_A, Section 13.3.1 on page 154	2.7		5.5	V
$V_{DD(x4PLL)}$	x4 PLL operating voltage		2.7		5.5	
$V_{DD(x8PLL)}$	x8 PLL operating voltage		3.3		5.5	
$t_{STARTUP}$	PLL Startup time			60		PLL input clock (f_{PLL}) cycles

13.5 INTERNAL RC OSCILLATOR CHARACTERISTICS

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{RC}	Internal RC oscillator frequency [1]	RCCR = FF (reset value), T_A=25°C,V_{DD}=5V		625		kHz
		RCCR = RCCR0 [2], T_A=25°C,V_{DD}=5V		1000		
ACC_{RC}	Accuracy of Internal RC oscillator with RCCR=RCCR0 [2]	T_A=25°C,V_{DD}=5V	-1		+1	%
		T_A=25°C, V_{DD}=4.5 to 5.5V [3]	-1		+1	%
		T_A=25 to +85°C,V_{DD}=5V [3]	-3		+3	%
		T_A=25 to +85°C,V_{DD}=4.5 to 5.5V [3]	-3.5		+3.5	%
		T_A=-40 to +25°C,V_{DD}=4.5 to 5.5V [3]	-3		+7	%
$I_{DD(RC)}$	RC oscillator current consumption	T_A=25°C,V_{DD}=5V		600 [3]		µA
$t_{su(RC)}$	RC oscillator setup time	T_A=25°C,V_{DD}=5V			10 [2]	µs

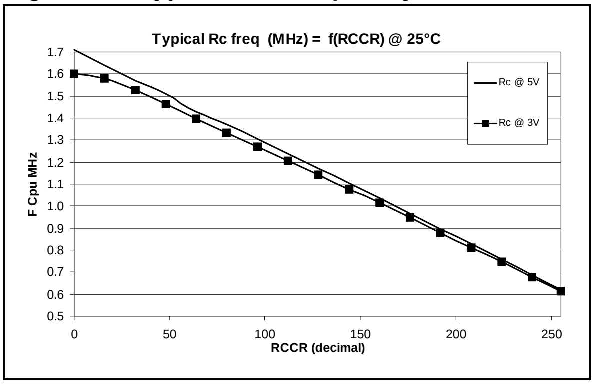
Notes:

1. If the RC oscillator clock is selected, to improve clock stability and frequency accuracy, it is recommended to place a decoupling capacitor, typically 100nF, between the V_{DD} and V_{SS} pins as close as possible to the ST7 device.

2. See "Internal RC Oscillator" on page 30

3. Expected results. Data based on characterization, not tested in production

Figure 85. Typical RC Frequency vs RCCR



13.6 SUPPLY CURRENT CHARACTERISTICS

The following current consumption specified for the ST7 functional operating modes over temperature range does not take into account the clock source current consumption. To get the total device consumption, the two current values must be added (except for HALT mode for which the clock is stopped).

13.6.1 Supply Current

T_A = -40 to +85°C unless otherwise specified

Symbol	Parameter	Conditions		Typ	Max	Unit
I_{DD}	Supply current in RUN mode	V_{DD}=5.5V	f_{CPU}=8MHz 1)	8.5	13	mA
	Supply current in WAIT mode		f_{CPU}=8MHz 2)	3.7	6	
	Supply current in SLOW mode		f_{CPU}=250kHz 3)	4.1	7	
	Supply current in SLOW WAIT mode		f_{CPU}=250kHz 4)	2.2	3.5	
	Supply current in HALT mode5)		-40°C≤T_A≤+85°C	1	10	µA
	Supply current in AWUFH mode 6)7)		T_A= +25°C	50	60	
	Supply current in Active Halt mode 6)7)		T_A= +25°C	500	700	

Notes:

1. CPU running with memory access, all I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (OSC1) driven by external square wave, LVD disabled.

2. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (OSC1) driven by external square wave, LVD disabled.

3. SLOW mode selected with f_{CPU} based on f_{OSC} divided by 32. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (OSC1) driven by external square wave, LVD disabled.

4. SLOW-WAIT mode selected with f_{CPU} based on f_{OSC} divided by 32. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (OSC1) driven by external square wave, LVD disabled.

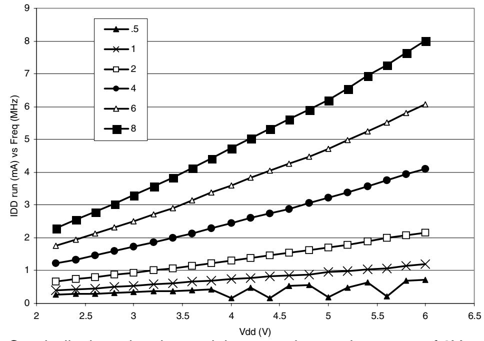
5. All I/O pins in output mode with a static value at V_{SS} (no load), LVD disabled. Data based on characterization results, tested in production at V_{DD} max and f_{CPU} max.

6. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load). Data tested in production at V_{DD} max. and f_{CPU} max.

7. This consumption refers to the Halt period only and not the associated run period which is software dependent.

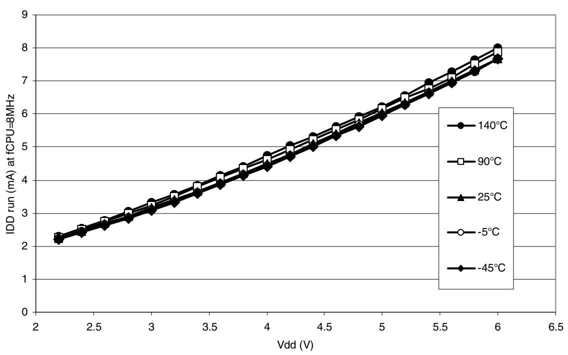
SUPPLY CURRENT CHARACTERISTICS

Figure 86. Typical I_{DD} in RUN vs. f_{CPU}



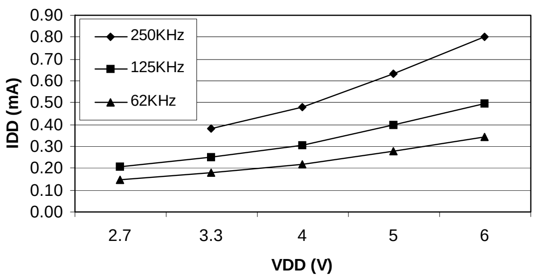
Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 87. Typical I_{DD} in RUN at f_{CPU} = 8MHz



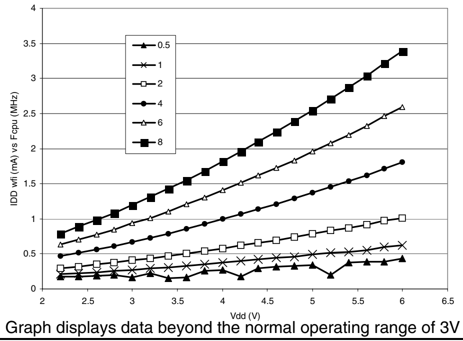
Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 88. Typical I_{DD} in SLOW vs. f_{CPU}



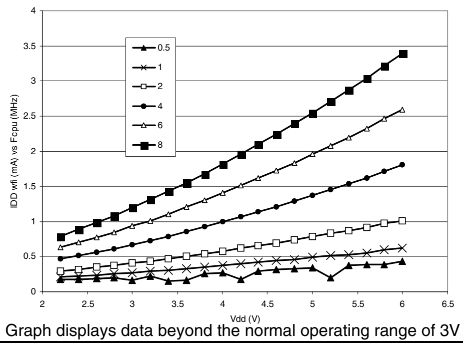
Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 89. Typical I_{DD} in WAIT vs. f_{CPU}



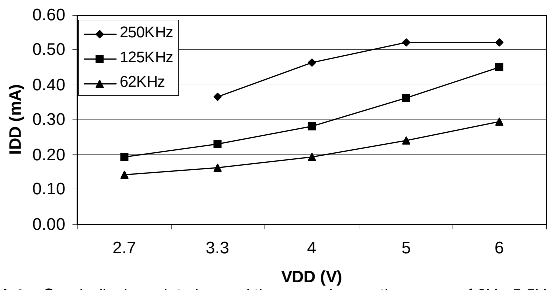
Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 90. Typical I_{DD} in WAIT at f_{CPU}= 8MHz



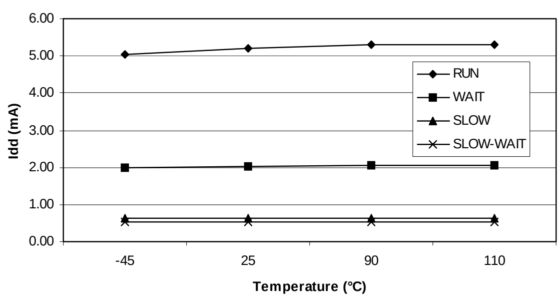
Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 91. Typical I_{DD} in SLOW-WAIT vs. f_{CPU}



Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 92. Typical I_{DD} vs. Temp. at V_{DD} = 5V and f_{CPU} = 8MHz



13.6.2 On-chip peripherals

Symbol	Parameter	Conditions		Typ	Unit
$I_{DD(16\text{-b timer})}$	16-bit Timer supply current [1]	f_{CPU}=4MHz	V_{DD}=3.0V	20	µA
		f_{CPU}=8MHz	V_{DD}=5.0V	100	
$I_{DD(SPI)}$	SPI supply current [2]	f_{CPU}=4MHz	V_{DD}=3.0V	250	
		f_{CPU}=8MHz	V_{DD}=5.0V	800	
$I_{DD(ADC)}$	ADC supply current when converting [3]	f_{ADC}=2MHz	V_{DD}=3.0V	300	
		f_{ADC}=4MHz	V_{DD}=5.0V	1000	
$I_{DD(I2C)}$	I2C supply current [4]	f_{CPU}=4MHz	V_{DD}=3.0V	100	
		f_{CPU}=8MHz	V_{DD}=5.0V	500	
$I_{DD(SCI)}$	SCI supply current [5]	f_{CPU}=4MHz	V_{DD}=3.0V	250	
		f_{CPU}=8MHz	V_{DD}=5.0V	800	

Notes:

1. Data based on a differential I_{DD} measurement between reset configuration (timer stopped) and a timer running in PWM mode at f_{cpu}=8MHz.

2. Data based on a differential I_{DD} measurement between reset configuration and a permanent SPI master communication (data sent equal to 55h).

3. Data based on a differential I_{DD} measurement between reset configuration and continuous A/D conversions.

4. Data based on a differential I_{DD} measurement between reset configuration (I2C disabled) and a permanent I2C master communication at 100kHz (data sent equal to 55h). This measurement include the pad toggling consumption (27kOhm external pull-up on clock and data lines).

5. Data based on a differential I_{DD} measurement between SCI low power state (SCID=1) and a permanent SCI data transmit sequence.

13.7 CLOCK AND TIMING CHARACTERISTICS

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

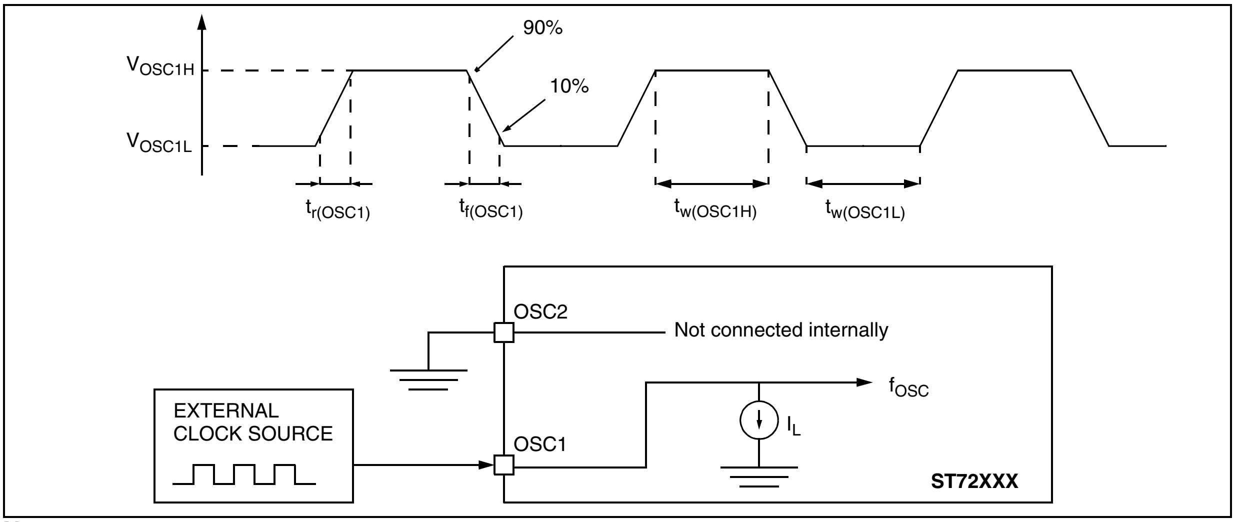
13.7.1 General Timings

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

13.7.2 External Clock Source

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{OSC1H}	OSC1 input pin high level voltage	see Figure 93	$0.7xV_{DD}$		V_{DD}	V
V_{OSC1L}	OSC1 input pin low level voltage		V_{SS}		$0.3xV_{DD}$	
$t_{w(OSC1H)}$ $t_{w(OSC1L)}$	OSC1 high or low time [4]		15			ns
$t_{r(OSC1)}$ $t_{f(OSC1)}$	OSC1 rise or fall time [4]				15	
I_L	OSCx Input leakage current	$V_{SS} \leq V_{IN} \leq V_{DD}$			±1	µA

Figure 93. Typical Application with an External Clock Source



Notes:

1. Guaranteed by Design. Not tested in production.

2. Data based on typical application software.

3. Time measured between interrupt event and interrupt vector fetch. $\Delta t_{c(INST)}$ is the number of t_{CPU} cycles needed to finish the current instruction execution.

4. Data based on design simulation and/or technology characteristics, not tested in production.

13.7.3 Auto Wakeup from Halt Oscillator (AWU)

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

CLOCK AND TIMING CHARACTERISTICS (Cont'd)

13.7.4 Crystal and Ceramic Resonator Oscillators

The ST7 internal clock can be supplied with four different Crystal/Ceramic resonator oscillators. All the information given in this paragraph is based on characterization results with specified typical external components. In the application, the resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and start-up stabilization time. Refer to the crystal/ceramic resonator manufacturer for more details (frequency, package, accuracy...).

Symbol	Parameter	Conditions	Min	Max	Unit
f_{OSC}	Oscillator Frequency [1]		1	16	MHz
R_F	Feedback resistor[2]		20	40	kΩ
C_{L1} C_{L2}	Recommended load capacitance versus equivalent serial resistance of the crystal or ceramic resonator (R_S)[3]	f_{OSC}= 1 to 2 MHz f_{OSC}= 2 to 4 MHz f_{OSC}= 4 to 8 MHz f_{OSC}= 8 to 16 MHz	20 20 15 15	60 50 35 35	pF

Symbol	Parameter	Conditions	Typ	Max	Unit
i_2	OSC2 driving current	V_{DD}=5V: f_{OSC}= 2MHz, C0 = 6pF, Cl1 = Cl2 = 68pF f_{OSC}= 4MHz, C0 = 6pF, Cl1 = Cl2 = 68pF f_{OSC}= 8MHz, C0 = 6pF, Cl1 = Cl2 = 40pF f_{OSC}= 16MHz, C0 = 7pF, Cl1 = Cl2 = 20pF	 426 425 456 660		µA

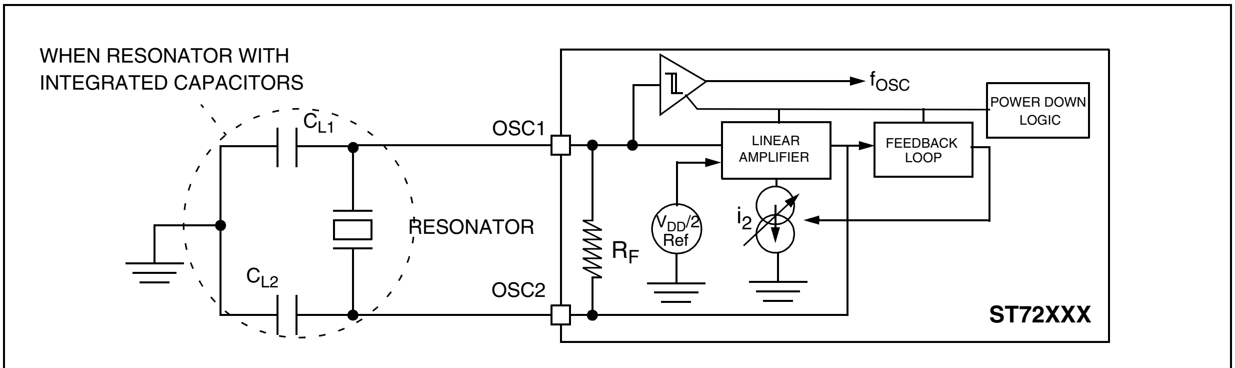
Notes:

1. The oscillator selection can be optimized in terms of supply current using an high quality resonator with small R_S value. Refer to crystal/ceramic resonator manufacturer for more details.

2. Data based on characterisation results, not tested in production. The relatively low value of the RF resistor, offers a good protection against issues resulting from use in a humid environment, due to the induced leakage and the bias condition change. However, it is recommended to take this point into account if the µC is used in tough humidity conditions.

3. For C_{L1} and C_{L2} it is recommended to use high-quality ceramic capacitors in the 5-pF to 25-pF range (typ.) designed for high-frequency applications and selected to match the requirements of the crystal or resonator. C_{L1} and C_{L2}, are usually the same size. The crystal manufacturer typically specifies a load capacitance which is the series combination of C_{L1} and C_{L2}. PCB and MCU pin capacitance must be included when sizing C_{L1} and C_{L2} (10 pF can be used as a rough estimate of the combined pin and board capacitance).

Figure 94. Typical Application with a Crystal or Ceramic Resonator



CLOCK AND TIMING CHARACTERISTICS (Cont'd)

Supplier	f_{OSC} (MHz)	Typical Ceramic Resonators[2)]
Murata	2	CSTCC2M00G56Z-R0
	4	SMD CSTCR4M00G53Z-R0
		Lead CSTLS4M00G53Z-R0
	8	SMD CSTCE8M00G52Z-R0
		Lead CSTLS4M0052Z-R0
	16	SMD CSTCE16M0V51Z-R0
		Lead CSTLS16M0X51Z-R0

Notes:

1. Resonator characteristics given by the ceramic resonator manufacturer.

2. SMD = [-R0: Plastic tape package (∅ =180mm), -B0: Bulk]
LEAD = [-A0: Flat pack package (Radial taping Ho= 18mm), -B0: Bulk]

For more information on these resonators, please consult www.murata.com

13.8 MEMORY CHARACTERISTICS

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

13.8.1 RAM and Hardware Registers

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{RM}	Data retention mode [1)]	HALT mode (or RESET)	1.6			V

13.8.2 FLASH Program Memory

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DD}	Operating voltage for Flash write/erase	Refer to operating range of V_{DD} with T_A, Section 13.3.1 on page 154	2.7		5.5	V
t_{prog}	Programming time for 1~32 bytes [2)]	T_A=–40 to +85°C		5	10	ms
t_{RET}	Data retention [4)]	T_A=+55°C[3)]	20			years
N_{RW}	Write erase cycles	T_A=+25°C	10K [7)]			cycles
I_{DD}	Supply current [6)]	Read / Write / Erase modes f_{CPU} = 8MHz, V_{DD} = 5.5V			2.6	mA
		No Read/No Write Mode			100	µA
		Power down mode / HALT		0	0.1	µA

13.8.3 EEPROM Data Memory

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DD}	Operating voltage for EEPROM write/erase	Refer to operating range of V_{DD} with T_A, Section 13.3.1 on page 154	2.7		5.5	V
t_{prog}	Programming time for 1~32 bytes	T_A=–40 to +85°C		5	10	ms
t_{ret}	Data retention [4)]	T_A=+55°C [3)]	20			years
N_{RW}	Write erase cycles	T_A=+25°C	300K [7)]			cycles

Notes:

1. Minimum V_{DD} supply voltage without losing data stored in RAM (in HALT mode or under RESET) or in hardware registers (only in HALT mode). Guaranteed by construction, not tested in production.

2. Up to 32 bytes can be programmed at a time.

3. The data retention time increases when the T_A decreases.

4. Data based on reliability test results and monitored in production.

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

6. Guaranteed by Design. Not tested in production.

7. Design target value pending full product characterization.

13.9 EMC CHARACTERISTICS

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

13.9.1 Functional EMS (Electro Magnetic Susceptibility)

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

- **ESD**: Electro-Static Discharge (positive and negative) is applied on all pins of the device until a functional disturbance occurs. This test conforms with the IEC 1000-4-2 standard.
- **FTB**: A Burst of Fast Transient voltage (positive and negative) is applied to V_{DD} and V_{SS} through a 100pF capacitor, until a functional disturbance occurs. This test conforms with the IEC 1000-4-4 standard.

A device reset allows normal operations to be resumed. The test results are given in the table below based on the EMS levels and classes defined in application note AN1709.

13.9.1.1 Designing hardened software to avoid noise problems

EMC characterization and optimization are performed at component level with a typical application environment and simplified MCU software. It should be noted that good EMC performance is highly dependent on the user application and the software in particular.

Therefore it is recommended that the user applies EMC software optimization and prequalification tests in relation with the EMC level requested for his application.

Software recommendations:

The software flowchart must include the management of runaway conditions such as:

– Corrupted program counter

– Unexpected reset

– Critical Data corruption (control registers...)

Prequalification trials:

Most of the common failures (unexpected reset and program counter corruption) can be reproduced by manually forcing a low state on the RESET pin or the Oscillator pins for 1 second.

To complete these trials, ESD stress can be applied directly on the device, over the range of specification values. When unexpected behaviour is detected, the software can be hardened to prevent unrecoverable errors occurring (see application note AN1015).

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

13.9.2 Electro Magnetic Interference (EMI)

Based on a simple application running on the product (toggling 2 LEDs through the I/O ports), the product is monitored in terms of emission. This emission test is in line with the norm SAE J 1752/3 which specifies the board and the loading of each pin.

Symbol	Parameter	Conditions	Monitored Frequency Band	Max vs. [f_{OSC}/f_{CPU}]		Unit
				8/4MHz	16/8MHz	
S_{EMI}	Peak level	V_{DD}=5V, T_A=+25°C, SO20 package, conforming to SAE J 1752/3	0.1MHz to 30MHz	TBD	TBD	dBµV
			30MHz to 130MHz	TBD	TBD	
			130MHz to 1GHz	TBD	TBD	
			SAE EMI Level	TBD	TBD	-

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

EMC CHARACTERISTICS (Cont'd)

13.9.3 Absolute Maximum Ratings (Electrical Sensitivity)

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

13.9.3.1 Electro-Static Discharge (ESD)

Electro-Static Discharges (a positive then a negative pulse separated by 1 second) are applied to the pins of each sample according to each pin combination. The sample size depends on the number of supply pins in the device (3 parts*(n+1) supply pin). Human Body Model can be simulated. This test conforms to the JESD22-A114A/A115A standard.

Absolute Maximum Ratings

Symbol	Ratings	Conditions	Maximum value 1)	Unit
$V_{ESD(HBM)}$	Electro-static discharge voltage (Human Body Model)	T_A=+25°C	>2000	V

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

13.9.3.2 Static and Dynamic Latch-Up

- **LU**: 3 complementary static tests are required on 6 parts to assess the latch-up performance. A supply overvoltage (applied to each power supply pin) and a current injection (applied to each input, output and configurable I/O pin) are performed on each sample. This test conforms to the EIA/JESD 78 IC latch-up standard. For more details, refer to the application note AN1181.
- **DLU**: Electro-Static Discharges (one positive then one negative test) are applied to each pin of 3 samples when the micro is running to assess the latch-up performance in dynamic mode. Power supplies are set to the typical values, the oscillator is connected as near as possible to the pins of the micro and the component is put in reset mode. This test conforms to the IEC1000-4-2 and SAEJ1752/3 standards. For more details, refer to the application note AN1181.

Electrical Sensitivities

Symbol	Parameter	Conditions	Class
LU	Static latch-up class	T_A=+25°C T_A=+85°C	A A
DLU	Dynamic latch-up class	V_{DD}=5.5V, f_{OSC}=4MHz, T_A=+25°C	A

13.10 I/O PORT PIN CHARACTERISTICS

13.10.1 General Characteristics

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

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

Notes:

1. Data based on validation/design results.

2. Configuration not recommended, all unused pins must be kept at a fixed voltage: using the output mode of the I/O for example or an external pull-up or pull-down resistor (see Figure 95). Static peak current value taken at a fixed V_{IN} value, based on design simulation and technology characteristics, not tested in production. This value depends on V_{DD} and temperature values.

3. The R_{PU} pull-up equivalent resistor is based on a resistive transistor.

4. To generate an external interrupt, a minimum pulse width has to be applied on an I/O port pin configured as an external interrupt source.

Figure 95. Two typical Applications with unused I/O Pin

Caution: During normal operation the ICCCLK pin must be pulled-up, internally or externally (external pull-up of 10k mandatory in noisy environment). This is to avoid entering ICC mode unexpectedly during a reset.

Note: I/O can be left unconnected if it is configured as output (0 or 1) by the software. This has the advantage of greater EMC robustness and lower cost.

I/O PORT PIN CHARACTERISTICS (Cont'd)

13.10.2 Output Driving Current

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

Symbol	Parameter	Conditions		Min	Max	Unit
V_{OL} 1)	Output low level voltage for a standard I/O pin when 8 pins are sunk at same time (see Figure 98)	V_{DD}=5V	I_{IO}=+5mA		1.0	V
			I_{IO}=+2mA		0.4	
	Output low level voltage for a high sink I/O pin when 4 pins are sunk at same time (see Figure 101)		I_{IO}=+20mA		1.3	
			I_{IO}=+8mA		0.75	
V_{OH} 2)	Output high level voltage for an I/O pin when 4 pins are sourced at same time (see Figure)		I_{IO}=-5mA	V_{DD}-1.5		
			I_{IO}=-2mA	V_{DD}-0.8		
V_{OL} 1)3)	Output low level voltage for a standard I/O pin when 8 pins are sunk at same time (see Figure 97)	V_{DD}=3.3V	I_{IO}=+2mA		0.7	
	Output low level voltage for a high sink I/O pin when 4 pins are sunk at same time		I_{IO}=+8mA		0.5	
V_{OH} 2)3)	Output high level voltage for an I/O pin when 4 pins are sourced at same time (Figure 108)		I_{IO}=-2mA	V_{DD}-0.8		
V_{OL} 1)3)	Output low level voltage for a standard I/O pin when 8 pins are sunk at same time (see Figure 99)	V_{DD}=2.7V	I_{IO}=+2mA		0.9	
	Output low level voltage for a high sink I/O pin when 4 pins are sunk at same time		I_{IO}=+8mA		0.6	
V_{OH} 2)3)	Output high level voltage for an I/O pin when 4 pins are sourced at same time (see ...)		I_{IO}=-2mA	V_{DD}-0.9		

Notes:

1. The I_{IO} current sunk must always respect the absolute maximum rating specified in Section 13.2.2 and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VSS}.

2. The I_{IO} current sourced must always respect the absolute maximum rating specified in Section 13.2.2 and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VDD}.

3. Not tested in production, based on characterization results.

I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 96. Typical V_{OL} at V_{DD}=2.4V (std I/Os)

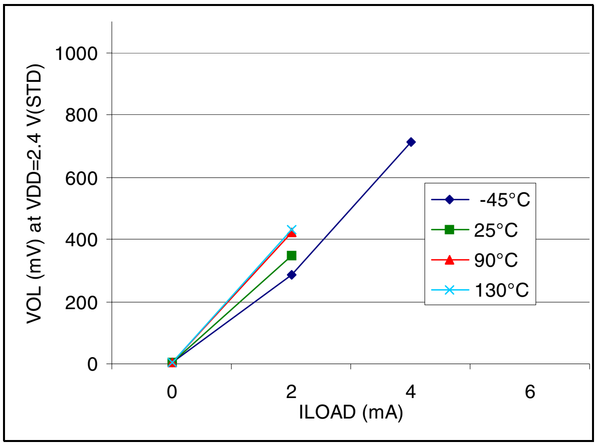


Figure 97. Typical V_{OL} at V_{DD}=3V (std I/Os)

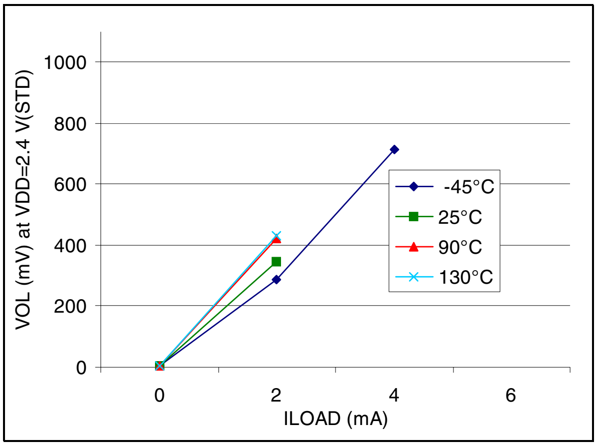


Figure 98. Typical V_{OL} at V_{DD}=5V (std I/Os)

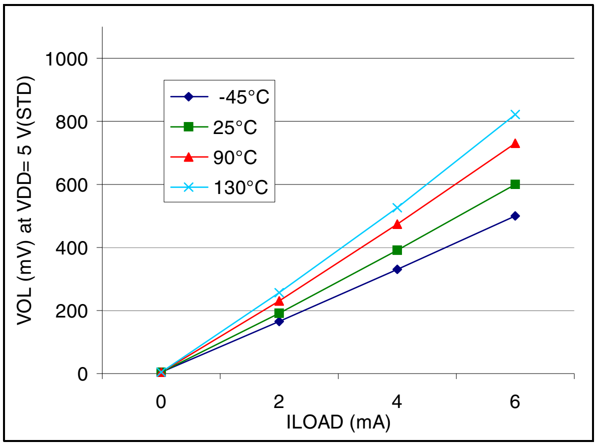


Figure 99. Typical V_{OL} at V_{DD}=2.4V (high-sink I/Os)

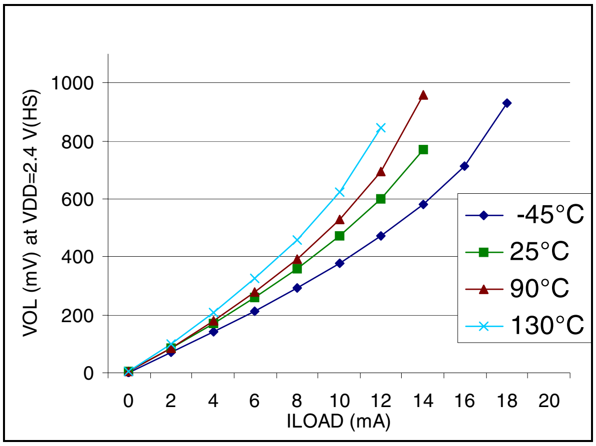


Figure 100. Typical V_{OL} at V_{DD}=3V (high-sink I/Os)

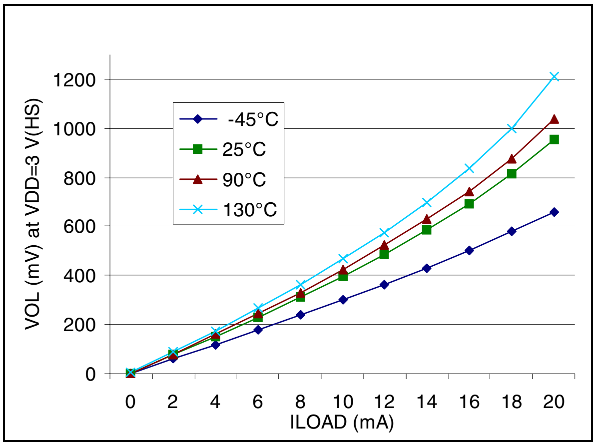
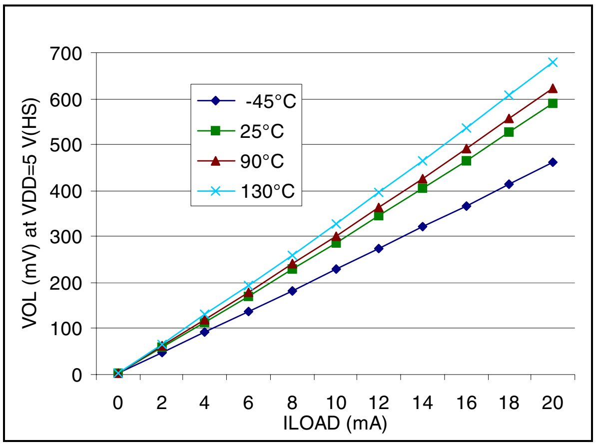


Figure 101. Typical V_{OL} at V_{DD}=5V (high-sink I/Os)



I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 102. Typical V_{OL} vs. V_{DD} (std I/Os, 2mA)

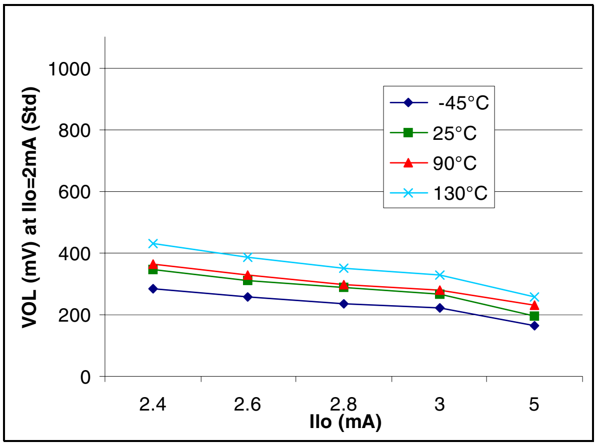


Figure 103. Typical V_{OL} vs. V_{DD} (std I/Os, 6mA)

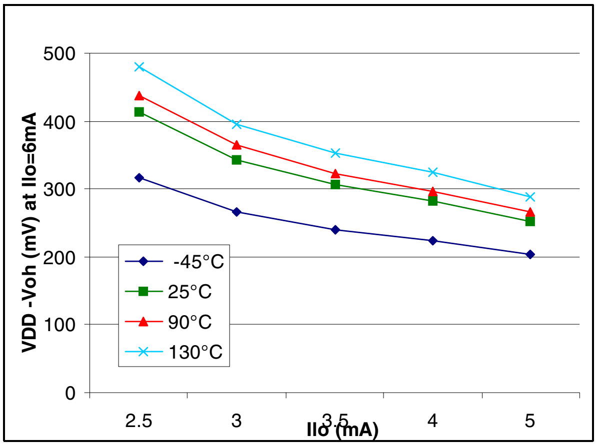


Figure 104. Typical V_{OL} vs. V_{DD} (HS I/Os, Iio=8mA)

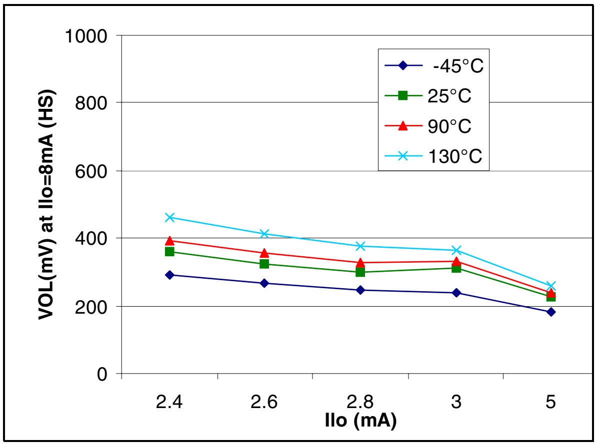


Figure 105. Typical V_{OL} vs. V_{DD} (HS I/Os, Iio=2mA)

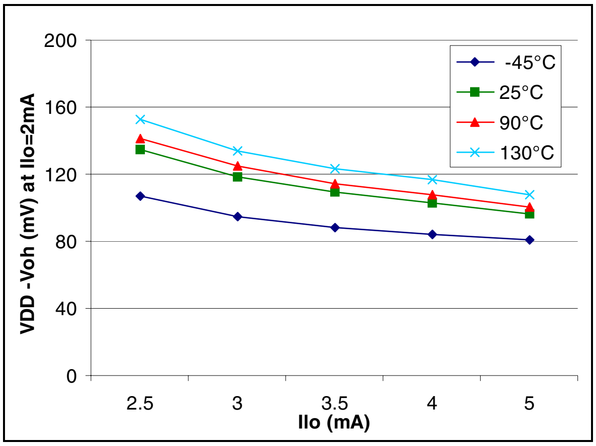


Figure 106. Typical V_{OL} vs. V_{DD} (HS I/Os, Iio=12mA)

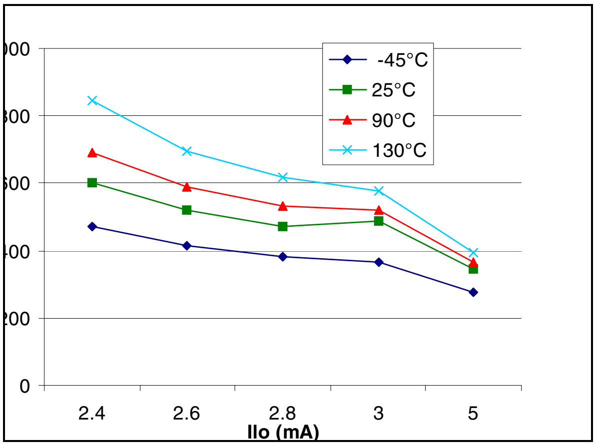


Figure 107. Typical V_{DD}-V_{OH} at V_{DD}=2.4V (std I/Os)

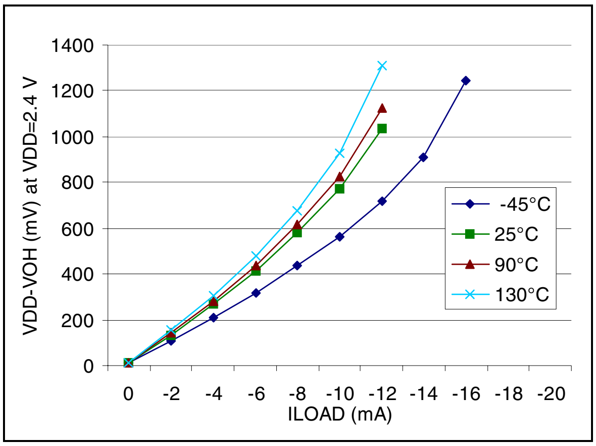
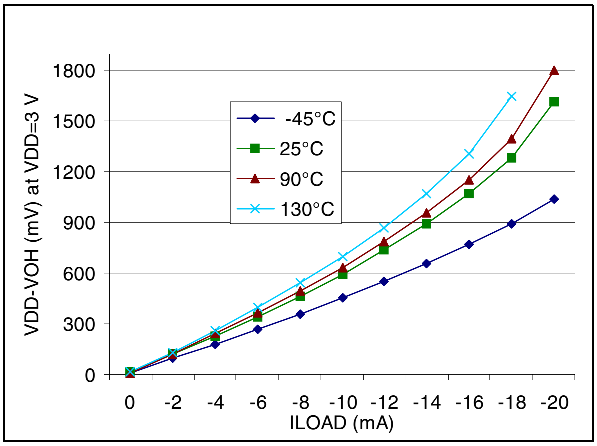


Figure 108. Typical V_{DD}-V_{OH} at V_{DD}=3V (std I/Os)



VDD-VOH (mV) at VDD=3 V
1800
1500
1200
900
600
300
0
-45°C
25°C
90°C
130°C
0 -2 -4 -6 -8 -10 -12 -14 -16 -18 -20
ILOAD (mA)

I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 109. Typical V_{DD}-V_{OH} at V_{DD}=4V (std)

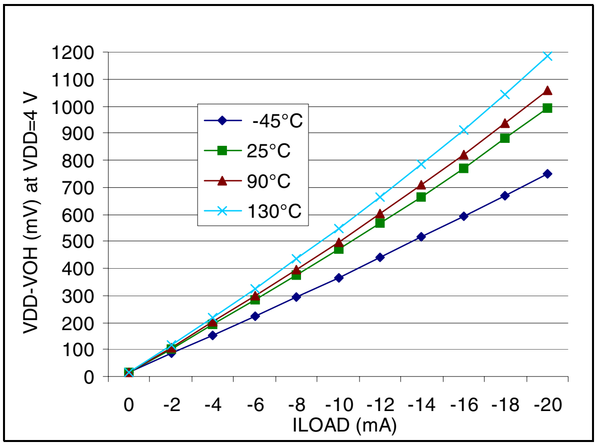


Figure 110. Typical V_{DD}-V_{OH} at V_{DD}=5V (std)

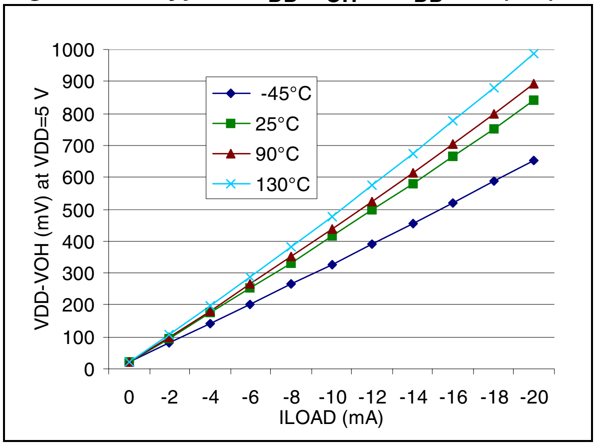
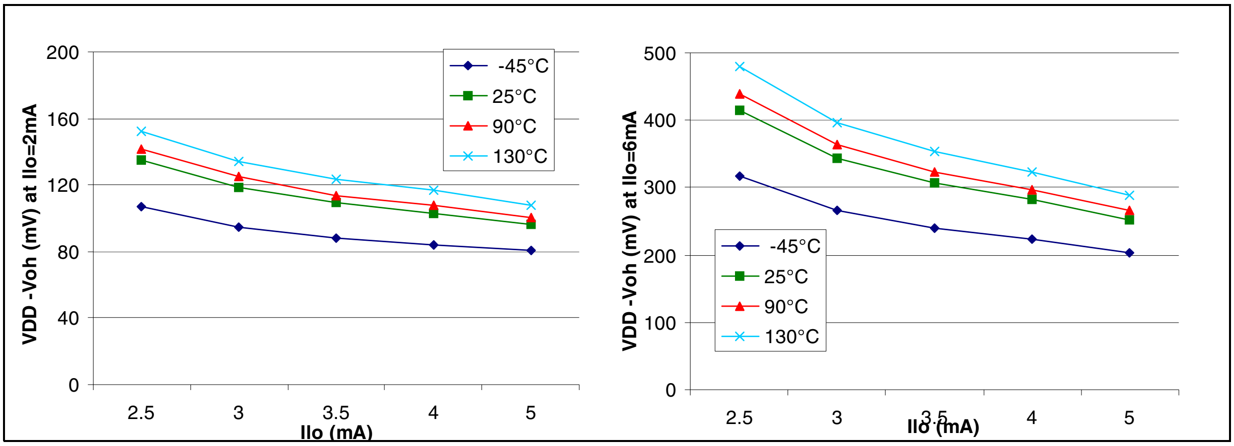


Figure 111. Typical V_{DD}-V_{OH} vs. V_{DD} (High Sink)



13.11 CONTROL PIN CHARACTERISTICS

13.11.1 Asynchronous $\overline{RESET}$ Pin

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

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{IL}	Input low level voltage			V_{ss} - 0.3		$0.3xV_{DD}$	V
V_{IH}	Input high level voltage			$0.7xV_{DD}$		V_{DD} + 0.3	
V_{hys}	Schmitt trigger voltage hysteresis 1)				2		V
V_{OL}	Output low level voltage 2)	V_{DD}=5V	I_{IO}=+5mA		0.5	1.0	V
			I_{IO}=+2mA		0.2	0.4	
R_{ON}	Pull-up equivalent resistor 3) 1)	V_{DD}=5V		20	40	80	kΩ
		V_{DD}=3V		40	70	120	
$t_{w(RSTL)out}$	Generated reset pulse duration	Internal reset sources			26		µs
$t_{h(RSTL)in}$	External reset pulse hold time 4)			20			µs
$t_{g(RSTL)in}$	Filtered glitch duration				200		ns

Notes:
1. Data based on characterization results, not tested in production.
2. The I_{IO} current sunk must always respect the absolute maximum rating specified in Section 13.2.2 and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VSS}.
3. The R_{ON} pull-up equivalent resistor is based on a resistive transistor. Specified for voltages on $\overline{RESET}$ pin between V_{ILmax} and V_{DD}
4. To guarantee the reset of the device, a minimum pulse has to be applied to the $\overline{RESET}$ pin. All short pulses applied on $\overline{RESET}$ pin with a duration below $t_{h(RSTL)in}$ can be ignored.

CONTROL PIN CHARACTERISTICS (Cont'd)

Figure 112. RESET pin protection when LVD is enabled.[1)2)3)4)]

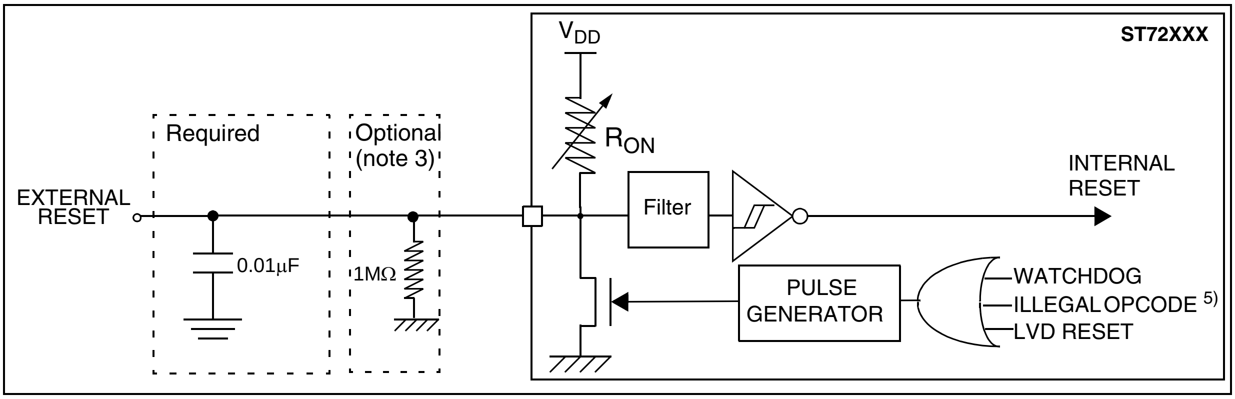
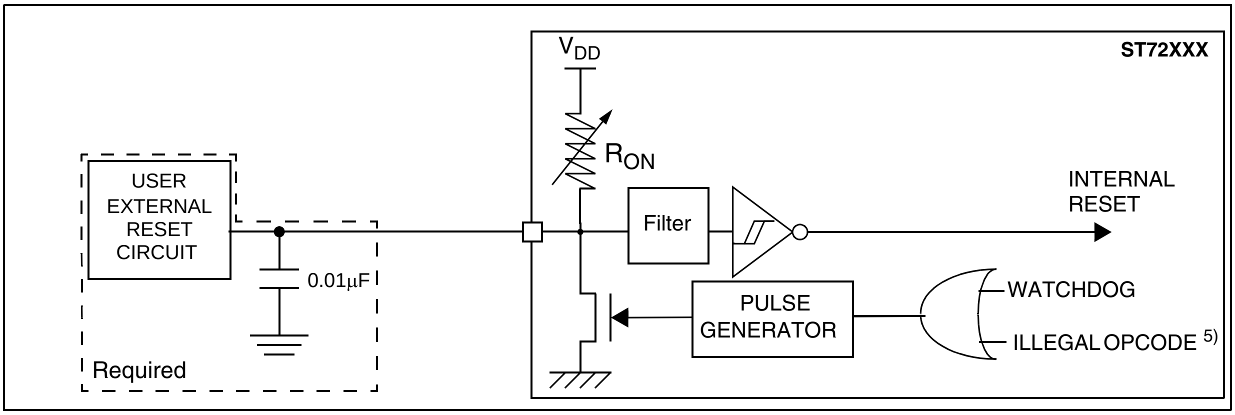


Figure 113. RESET pin protection when LVD is disabled.[1)]



Note 1:

– The reset network protects the device against parasitic resets.

– The output of the external reset circuit must have an open-drain output to drive the ST7 reset pad. Otherwise the device can be damaged when the ST7 generates an internal reset (LVD or watchdog).

– Whatever the reset source is (internal or external), the user must ensure that the level on the RESET pin can go below the V_{IL} max. level specified in Section 13.11.1 on page 172. Otherwise the reset will not be taken into account internally.

– Because the reset circuit is designed to allow the internal RESET to be output in the RESET pin, the user must ensure that the current sunk on the RESET pin is less than the absolute maximum value specified for $I_{INJ(RESET)}$ in Section 13.2.2 on page 153.

Note 2: When the LVD is enabled, it is recommended not to connect a pull-up resistor or capacitor. A 10nF pull-down capacitor is required to filter noise on the reset line.

Note 3: In case a capacitive power supply is used, it is recommended to connect a 1MΩ pull-down resistor to the RESET pin to discharge any residual voltage induced by the capacitive effect of the power supply (this will add 5µA to the power consumption of the MCU).

Note 4: Tips when using the LVD:

– 1. Check that all recommendations related to the reset circuit have been applied (see notes above)

– 2. Check that the power supply is properly decoupled (100nF + 10µF close to the MCU). Refer to AN1709 and AN2017. If this cannot be done, it is recommended to put a 100nF + 1MΩ pull-down on the RESET pin.

– 3. The capacitors connected on the RESET pin and also the power supply are key to avoid any start-up marginality. In most cases, steps 1 and 2 above are sufficient for a robust solution. Otherwise: replace 10nF pull-down on the RESET pin with a 5µF to 20µF capacitor."

Note 5: Please refer to "Illegal Opcode Reset" on page 149 for more details on illegal opcode reset conditions.

13.12 COMMUNICATION INTERFACE CHARACTERISTICS

13.12.1 I^2C and I²C3SNS Interfaces

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

Refer to I/O port characteristics for more details on the input/output alternate function characteristics (SDAI and SCLI). The ST7 I^2C and I2C3SNS interfaces meet the electrical and timing requirements of the Standard I^2C communication protocol.

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

Symbol	Parameter	Conditions	Min	Max	Unit
f_{SCL}	I²C SCL frequency	f_{CPU}=4 MHz to 8 MHz [1], V_{DD}= 2.7V to 5.5V		400	kHz
$f_{SCL3SNS}$	I²C3SNS SCL frequency			400	kHz

Note:

1. The I^2C and I2C3SNS interfaces will not function below the minimum clock speed of 4 MHz.

The following table gives the values to be written in the I2CCCR register to obtain the required I^2C SCL line frequency.

Table 29. SCL Frequency Table (Multimaster I^2C Interface)

f_{SCL}	I2CCCR Value							
	f_{CPU}=4 MHz.				f_{CPU}=8 MHz.			
	V_{DD} = 3.3 V		V_{DD} = 5 V		V_{DD} = 3.3 V		V_{DD} = 5 V	
	R_P=3.3kΩ	R_P=4.7kΩ	R_P=3.3kΩ	R_P=4.7kΩ	R_P=3.3kΩ	R_P=4.7kΩ	R_P=3.3kΩ	R_P=4.7kΩ
400	NA	NA	NA	NA	84h	84h	84h	84h
300	NA	NA	NA	NA	86h	86h	85h	87h
200	84h	84h	84h	84h	8Ah	8Ah	8Bh	8Ch
100	11h	10h	11h	11h	25h	24h	28h	28h
50	25h	24h	25h	26h	4Bh	4Ch	53h	54h
20	60h	5Fh	60h	62h	FFh	FFh	FFh	FFh

Legend:

R_P = External pull-up resistance

f_{SCL} = I^2C speed

NA = Not achievable

Note:

– For speeds around 200 kHz, achieved speed can have ±5% tolerance

– For other speed ranges, achieved speed can have ±2% tolerance

The above variations depend on the accuracy of the external components used.

13.13 10-BIT ADC CHARACTERISTICS

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

ADC Accuracy

Symbol	Parameter	Conditions [1)2)]	Typ	Max [3)]	Unit
$\|E_T\|$	Total unadjusted error	f_{CPU}=8 MHz, f_{ADC}=4 MHz R_{AIN}< 10kΩ, V_{DD}= 2.7V to 5.5V	4	8	LSB
$\|E_O\|$	Offset error		-1	-2	
$\|E_G\|$	Gain Error		-2	-4	
$\|E_D\|$	Differential linearity error		3	6	

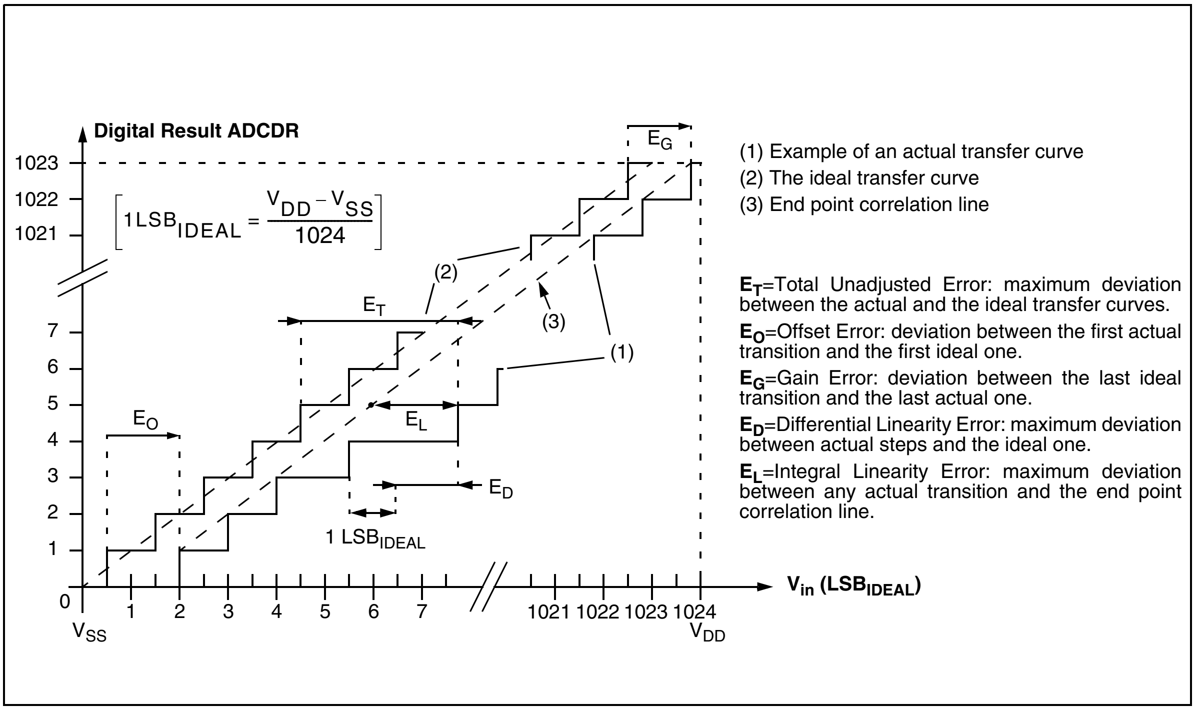
Note:

1. Data based on characterization results over the whole temperature range.

2. ADC accuracy vs negative injection current: Injecting negative current on any of the analog input pins may reduce the accuracy of the conversion being performed on another analog input.
The effect of negative injection current on robust pins is specified in Section 13.11 on page 172
Any positive injection current within the limits specified for $I_{INJ(PIN)}$ and $\Sigma I_{INJ(PIN)}$ in Section 13.10 does not affect the ADC accuracy.

3. Data based on characterization results, monitored in production to guarantee 99.73% within ± max value from -40°C to +125°C (± 3σ distribution limits).

Figure 114. ADC Accuracy Characteristics



ADC Characteristics

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

Symbol	Parameter	Conditions	Min	Typ [1]	Max	Unit
f_{ADC}	ADC clock frequency		0.4		4	MHz
V_{AIN}	Conversion voltage range [2]		V_{SSA}		V_{DDA}	V
R_{AIN}	External input resistor				10 [3]	kΩ
C_{ADC}	Internal sample and hold capacitor			6		pF
t_{STAB}	Stabilization time after ADC enable	f_{CPU}=8MHz, f_{ADC}=4MHz		0 [4]		µs
t_{ADC}	Conversion time (Sample+Hold)			3.5		
	- Sample capacitor loading time - Hold conversion time			4 10		$1/f_{ADC}$
I_{ADC}	Analog Part			1		mA
	Digital Part			0.2		

Notes:

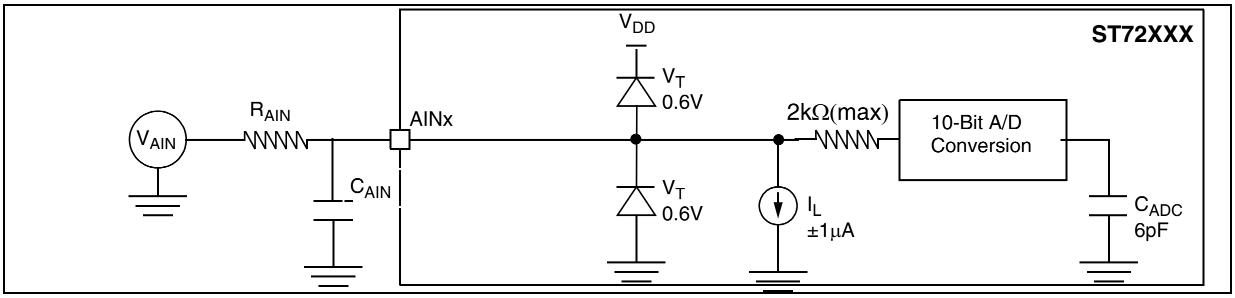
1. Unless otherwise specified, typical data are based on T_A=25°C and V_{DD}-V_{SS}=5V. They are given only as design guidelines and are not tested.

2. When V_{DDA} and V_{SSA} pins are not available on the pinout, the ADC refers to V_{DD} and V_{SS}.

3. Any added external serial resistor will downgrade the ADC accuracy (especially for resistance greater than 10kΩ). Data based on characterization results, not tested in production.

4. The stabilization time of the AD converter is masked by the first t_{LOAD}. The first conversion after the enable is then always valid.

Figure 115. Typical A/D Converter Application



Notes:

1. $C_{PARASITIC}$ represents the capacitance of the PCB (dependent on soldering and PCB layout quality) plus the pad capacitance (3pF). A high $C_{PARASITIC}$ value will downgrade conversion accuracy. To remedy this, f_{ADC} should be reduced.

2. This graph shows that depending on the input signal variation (f_{AIN}), C_{AIN} can be increased for stabilization time and decreased to allow the use of a larger serial resistor (R_{AIN}).

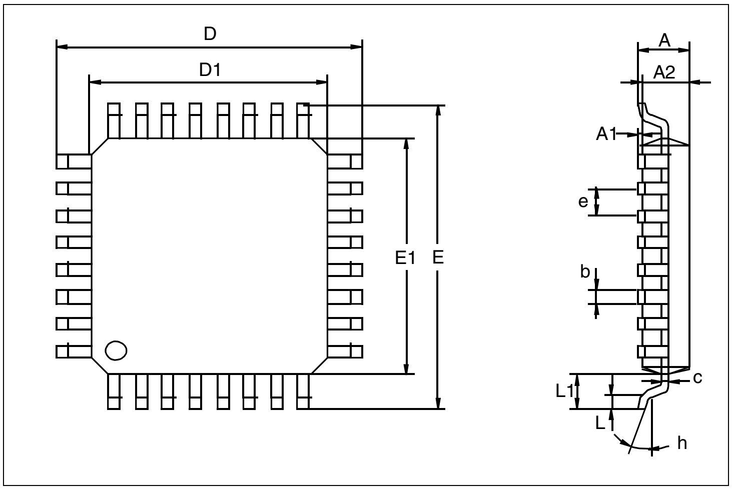
14 PACKAGE CHARACTERISTICS

In order to meet environmental requirements, ST offers these devices in ECOPACK® packages. These packages have a Lead-free second level interconnect. The category of second Level Interconnect is marked on the package and on the inner box label, in compliance with JEDEC Standard JESD97. The maximum ratings related to soldering conditions are also marked on the inner box label.

ECOPACK is an ST trademark. ECOPACK specifications are available at: www.st.com.

14.1 PACKAGE MECHANICAL DATA

Figure 116. 32-Pin Low Profile Quad Flat Package (7x7)

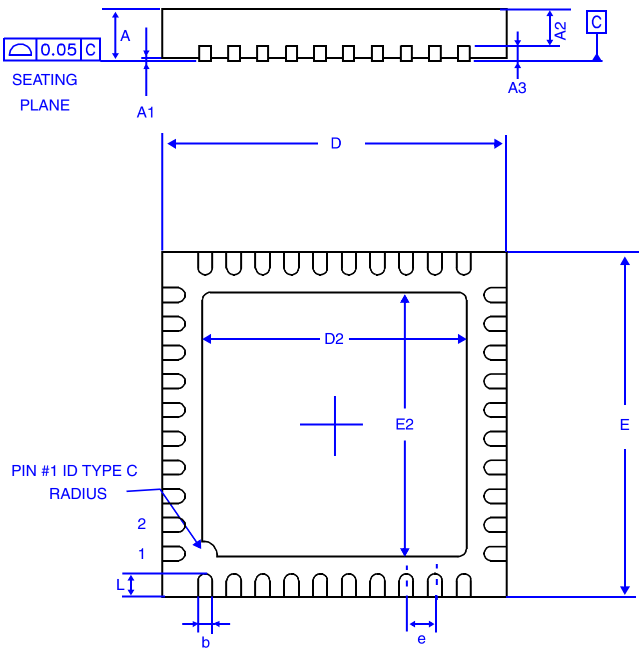


Dim.	mm			inches[1)]		
	Min	Typ	Max	Min	Typ	Max
A			1.60			0.063
A1	0.05		0.15	0.002		0.006
A2	1.35	1.40	1.45	0.053	0.055	0.057
b	0.30	0.37	0.45	0.012	0.015	0.018
C	0.09		0.20	0.004		0.008
D		9.00			0.354	
D1		7.00			0.276	
E		9.00			0.354	
E1		7.00			0.276	
e		0.80			0.031	
θ	0°	3.5°	7°	0°	3.5°	7°
L	0.45	0.60	0.75	0.018	0.024	0.030
L1		1.00			0.039	
	Number of Pins					
N	32					

Note 1. Values in inches are converted from mm and rounded to 3 decimal digits.

PACKAGE CHARACTERISTICS (Cont'd)

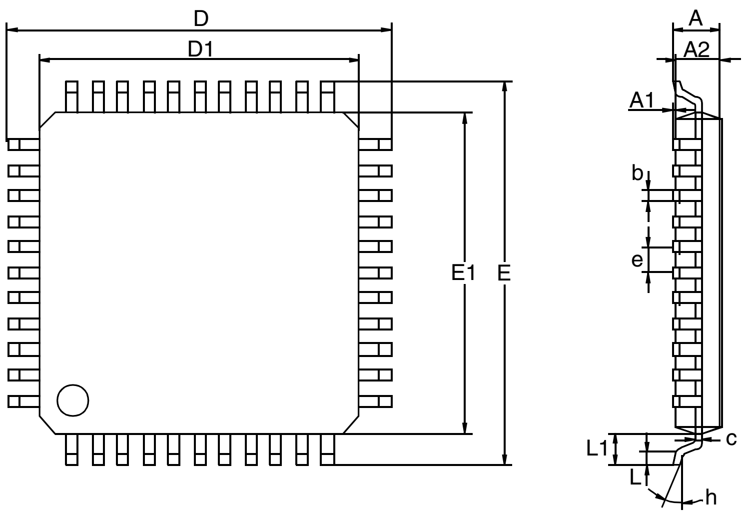
Figure 117. 40-Lead Very thin Fine pitch Quad Flat No-Lead Package



Dim.	mm			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A	0.80	0.90	1.00	0.031	0.035	0.039
A1		0.02	0.05		0.001	0.002
A2		0.65	1.00		0.026	0.039
A3		0.20			0.008	
b	0.18	0.25	0.30	0.007	0.010	0.012
D	5.85	6.00	6.15	0.230	0.236	0.242
D2	2.75	2.9	3.05	0.108	0.114	0.120
E	5.85	6	6.15	0.230	0.236	0.242
E2	2.75	2.9	3.05	0.108	0.114	0.120
e		0.50			0.020	
L	0.30	0.40	0.50	0.012	0.016	0.020
	Number of Pins					
N	40					

Note 1. Values in inches are converted from mm and rounded to 3 decimal digits.

Figure 118. 44-Pin Low Profile Quad Flat Package

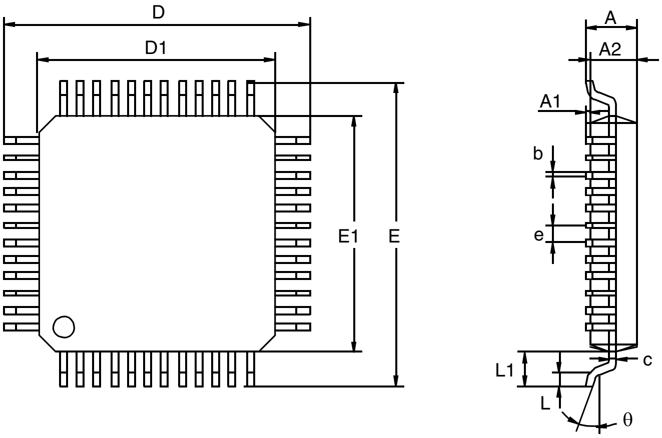


Dim.	mm			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A			1.60			0.063
A1	0.05		0.15	0.002		0.006
A2	1.35	1.40	1.45	0.053	0.055	0.057
b	0.30	0.37	0.45	0.012	0.015	0.018
C	0.09		0.20	0.004	0.000	0.008
D		12.00			0.472	
D1		10.00			0.394	
E		12.00			0.472	
E1		10.00			0.394	
e		0.80			0.031	
θ	0°	3.5°	7°	0°	3.5°	7°
L	0.45	0.60	0.75	0.018	0.024	0.030
L1		1.00			0.039	
	Number of Pins					
N	44					

Note 1. Values in inches are converted from mm and rounded to 3 decimal digits.

PACKAGE CHARACTERISTICS (Cont'd)

Figure 119. 48-Pin Low profile Quad Flat Package



Dim.	mm			inches[1)]		
	Min	Typ	Max	Min	Typ	Max
A			1.60			0.063
A1	0.05		0.15	0.002		0.006
A2	1.35	1.40	1.45	0.053	0.055	0.057
b	0.17	0.22	0.27	0.007	0.009	0.011
C	0.09		0.20	0.004		0.008
D		9.00			0.354	
D1		7.00			0.276	
E		9.00			0.354	
E1		7.00			0.276	
e		0.50			0.020	
θ	0°	3.5°	7°	0°	3.5°	7°
L	0.45	0.60	0.75	0.018	0.024	0.030
L1		1.00			0.039	
	Number of Pins					
N	48					

Note 1. Values in inches are converted from mm and rounded to 3 decimal digits.

PACKAGE CHARACTERISTICS (Cont'd)

Table 30. THERMAL CHARACTERISTICS

Symbol	Ratings		Value [3]	Unit
R_{thJA}	Package thermal resistance (junction to ambient)	LQFP32	60	°C/W
		LQFP44	54	
		LQFP48	73	
T_{Jmax}	Maximum junction temperature [1]		150	°C
P_{Dmax}	Power dissipation [2]	LQFP32	415	mW
		LQFP44	460	
		LQFP48	340	

Notes:

1. The maximum chip-junction temperature is based on technology characteristics.

2. The maximum power dissipation is obtained from the formula $P_D = (T_J - T_A) / R_{thJA}$.

The power dissipation of an application can be defined by the user with the formula: $P_D = P_{INT} + P_{PORT}$ where P_{INT} is the chip internal power (I_{DD} x V_{DD}) and P_{PORT} is the port power dissipation depending on the ports used in the application.

3. Values given for a 4-layer board. P_{Dmax} computed for T_A = 125°C.

15 DEVICE CONFIGURATION AND ORDERING INFORMATION

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

ST7P234x devices are Factory Advanced Service Technique ROM (FASTROM) versions: they are factory-programmed XFlash devices.

ST72F34x FLASH devices are shipped to customers with a default content (FFh). This implies that FLASH devices have to be configured by the customer using the Option Bytes.

15.1 OPTION BYTES

The four option bytes allow the hardware configuration of the microcontroller to be selected.

The option bytes can be accessed only in programming mode (for example using a standard ST7 programming tool).

OPTION BYTE 0

OPT7 = **WDG HALT** *Watchdog Reset on Halt*
This option bit determines if a RESET is generated when entering HALT mode while the Watchdog is active.
0: No Reset generation when entering Halt mode
1: Reset generation when entering Halt mode

OPT6 = **WDG SW** *Hardware or Software Watchdog*
This option bit selects the watchdog type.
0: Hardware (watchdog always enabled)
1: Software (watchdog to be enabled by software)

OPT5:4 = **LVD[1:0]** *Low voltage detection selection*
These option bits enable the LVD block with a selected threshold as shown in Table 31.

Table 31. LVD Threshold Configuration

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

OPT3:2 = **SEC[1:0]** *Sector 0 size definition*
These option bits indicate the size of sector 0 according to the following table.

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

OPT1 = **FMP_R** *Read-out protection*
Readout protection, when selected provides a protection against program memory content extraction and against write access to Flash memory. Erasing the option bytes when the FMP_R option is selected will cause the whole memory to be erased first and the device can be reprogrammed. Refer to the ST7 Flash Programming Reference Manual and Section 4.5 on page 17 for more details
0: Read-out protection off
1: Read-out protection on

OPT0 = **FMP_W** *FLASH write protection*
This option indicates if the FLASH program memory is write protected.
Warning: When this option is selected, the program memory (and the option bit itself) can never be erased or programmed again.
0: Write protection off
1: Write protection on

	OPTION BYTE 0								**OPTION BYTE 1**							
	7							0	7							0
	WDG HALT	WDG SW	LVD1	LVD0	SEC1	SEC0	FMP R	FMP W	RST C	OSCRANGE 2:0			OSC	DIV2 EN	PLL x4x8	PLL OFF
Default Value	1	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1

OPTION BYTES (Cont'd)

OPTION BYTE 1

OPT7 = **RSTC** *RESET clock cycle selection*
This option bit selects the number of CPU cycles inserted during the RESET phase and when exiting HALT mode. For resonator oscillators, it is advised to select 4096 due to the long crystal stabilization time.
0: Reset phase with 4096 CPU cycles
1: Reset phase with 256 CPU cycles

OPT6:4 = **OSCRANGE[2:0]** *Oscillator range*
When the internal RC oscillator is not selected (Option OSC=1), these option bits select the range of the resonator oscillator current source or the external clock source.

			OSCRANGE		
			2	**1**	**0**
Typ. frequency range with Resonator	LP	1~2MHz	0	0	0
	MP	2~4MHz	0	0	1
	MS	4~8MHz	0	1	0
	HS	8~16MHz	0	1	1
Reserved			1	0	0
External Clock			1	0	1
			1	1	0
			1	1	1

OPT3 = **OSC** *RC Oscillator selection*
0: RC oscillator on
1: RC oscillator off

OPT2 = **DIV2EN** *PLL Divide by 2 enable*

0: PLL division by 2 enabled
1: PLL division by 2 disabled

Note: DIV2EN must be kept disabled when PLLx4 is enabled

OPT1 = **PLLx4x8** *PLL Factor selection*
0: PLLx4
1: PLLx8

OPT0 = **PLLOFF** *PLL disable*
0: PLL enabled
1: PLL disabled (by-passed)

These option bits must be configured as described in Table 32 depending on the voltage range and the expected CPU frequency

Table 32. List of valid option combinations

Target Ratio	V_{DD}	Option Bits		
		DIV2 EN	**PLL OFF**	**PLL x4x8**
x4[1)]	2.7V - 3.65V	x	0	0
x4	3.3V - 5.5V	0	0	1
x8		1	0	1

Note:

1. For a target ratio of x4 between 3.3V - 3.65V, this is the recommended configuration.

OPTION BYTES (Cont'd)

OPTION BYTE 2

OPT7:0 = Reserved. Must be kept at 1.

OPTION BYTE 3

OPT7:6= **PKG1:0** *Package selection*
These option bits select the package.

Version	Selected Package	PKG 1	PKG 0
K	LQFP32	0	0
S	LQFP44	0	1
C	LQFP48	1	x

OPT5 = **I2C3S** *I2C3SNS selection*
0: I2C3SNS selected
1: I2C3SNS not selected

OPT4:0 = Reserved. Must be kept at 1.

<table>
<tr><td rowspan="2"></td><td colspan="8">OPTION BYTE 2
7 0</td><td colspan="8">OPTION BYTE 3
7 0</td></tr>
<tr><td colspan="8">Reserved</td><td>PKG1</td><td>PKG0</td><td>I2C3S</td><td colspan="5">Reserved</td></tr>
<tr><td>Default Value</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>x</td><td>x</td><td>x</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
</table>

15.2 DEVICE ORDERING INFORMATION

Table 33. Supported part numbers

Part Number	Peripherals	Program Memory (Bytes)	RAM (Bytes)	Data EEPROM (Bytes)	Temp. Range	Package
ST72F340K2T6	Common peripherals	8K FLASH	512	256	-40°C to 85°C	LQFP32
ST72F340S2T6						LQFP44
ST72F340K4T6		16K FLASH	1K			LQFP32
ST72F340S4T6						LQFP44
ST72F344K2T6	Common peripherals + 10-bit ADC, int high-accuracy 1MHz RC	8K FLASH	512			LQFP32
ST72F344S2T6						LQFP44
ST72F344K4T6		16K FLASH	1K			LQFP32
ST72F344S4T6						LQFP44
ST72F345C4T6	Common peripherals + I²C3SNS 10-bit ADC, int high-accuracy 1MHz RC	16K FLASH	1K			LQFP48

☐ Contact ST sales office for product availability

ST7234x FASTROM MICROCONTROLLER OPTION LIST
(Last update: October 2006)

Customer . . . .
Address . . . .
. . . .
Contact . . . .
Phone No . . . .
Reference FASTROM Code*: . . . .
*FASTROM code name is assigned by STMicroelectronics.
FASTROM code must be sent in .S19 format. .Hex extension cannot be processed.

Device Type/Memory Size/Package (check only one option):

FASTROM DEVICE:	8K	16K
LQFP32	[ ] ST72P344K4T	[ ] ST72P344K2T
LQFP44	[ ] ST72P344S4T	[ ] ST72P344S2T
LQFP48	[ ] ST72P345C4T	

Conditioning for LQFP (check only one option): [ ] Tape & Reel [ ] Tube

Version/ Temperature range (please refer to datasheet for specific sales conditions):

[ ] 0°C to +70°C [ ] -10°C to +85°C [ ] -40°C to +85°C

Special Marking: [ ] No [ ] Yes
Authorized characters are letters, digits, '.', '-', '/' and spaces only.
Maximum character count: LQFP32, LQFP48: 7 char. max "_ _ _ _ _ _ _ "
LQFP44: 10 char. max "_ _ _ _ _ _ _ _ _ _"

Clock Source Selection: [ ] External resonator or quartz
[ ] Internal RC Oscillator
[ ] External Clock:
[ ] LP: Low power (1 to 2 MHz)
[ ] MP: Medium power (2 to 4 MHz)
[ ] MS: Medium speed (4 to 8 MHz)
[ ] HS: High speed (8 to 16 MHz)

PLL: [ ] Disabled [ ] Enabled
[ ] PLL x 4 (*)
[ ] PLL x 8

DIV2: [ ] Disabled [ ] Enabled (*)

LVD Reset: [ ] Disabled [ ] Highest threshold
[ ] Medium threshold
[ ] Lowest threshold

Reset delay: [ ] 256 cycles [ ] 4096 cycles

Watchdog Selection: [ ] Software Activation [ ] Hardware Activation

Watchdog Reset on Halt: [ ] Disabled [ ] Enabled

Readout Protection: [ ] Disabled [ ] Enabled

FLASH Write Protection (**): [ ] Disabled [ ] Enabled
FLASH Sector 0 size (**): [ ] 0.5K [ ] 1K [ ]2K [ ] 4K
I2C3SNS (for ST72F345 only): [ ] Disabled [ ] Enabled

(*) DIV2 and PLLx4 cannot be enabled at the same time
(**) not available on first silicon version with waiver (contact ST local marketing)

Comments : . . . .
Supply Operating Range in the application: . . . .
Date: . . . .
Signature: . . . .

Please download the latest version of this option list from:
http://www.st.com/mcu > **downloads** > **ST7 microcontrollers** > **Option list**

15.3 DEVELOPMENT TOOLS

Development tools for the ST7 microcontrollers include a complete range of hardware systems and software tools from STMicroelectronics and third-party tool suppliers. The range of tools includes solutions to help you evaluate microcontroller peripherals, develop and debug your application, and program your microcontrollers.

15.3.1 Starter kits

ST offers complete, affordable **starter kits**. Starter kits are complete, affordable hardware/software tool packages that include features and samples to help you quickly start developing your application.

15.3.2 Development and debugging tools

Application development for ST7 is supported by fully optimizing **C Compilers** and the **ST7 Assembler-Linker** toolchain, which are all seamlessly integrated in the ST7 integrated development environments in order to facilitate the debugging and fine-tuning of your application. The Cosmic C Compiler is available in a free version that outputs up to 16KBytes of code.

The range of hardware tools includes full-featured **ST7-EMU3 series emulators** and the low-cost **RLink** in-circuit debugger/programmer. These tools are supported by the **ST7 Toolset** from STMicroelectronics, which includes the STVD7 integrated development environment (IDE) with high-level language debugger, editor, project manager and integrated programming interface.

15.3.3 Programming tools

During the development cycle, the **ST7-EMU3 series emulators** and the **RLink** provide in-circuit programming capability for programming the Flash microcontroller on your application board.

ST also provides a low-cost dedicated in-circuit programmer, the **ST7-STICK**, as well as **ST7 Socket Boards** which provide all the sockets required for programming any of the devices in a specific ST7 sub-family on a platform that can be used with any tool with in-circuit programming capability for ST7.

For production programming of ST7 devices, ST's third-party tool partners also provide a complete range of gang and automated programming solutions, which are ready to integrate into your production environment.

15.3.4 Order codes for ST72F34x development tools

Table 34. Development tool order codes

MCU	Starter kit	Emulator	Programming Tool	
			In-circuit debugger/ programmer	Dedicated programmer
ST72F340 ST72F344 ST72F345	ST72F34x-SK/RAIS 1)	ST7MDT40-EMU3	STX-RLINK 2) ST7-STICK 3)4)	ST7SB20J/xx 3)5) ST7SB40-QP48/xx 3)6)

Notes:

1. USB connection to PC
2. RLink with ST7 tool set
3. Add suffix /EU, /UK or /US for the power supply for your region
4. Parallel port connection to PC
5. Only available for LQFP32 and LQFP44 packages
6. Only available for LQFP48 package

For additional ordering codes for spare parts and accessories, refer to the online product selector at www.st.com/mcu.

16 KNOWN LIMITATIONS

16.1 External interrupt missed

To avoid any risk if generating a parasitic interrupt, the edge detector is automatically disabled for one clock cycle during an access to either DDR and OR. Any input signal edge during this period will not be detected and will not generate an interrupt.

This case can typically occur if the application refreshes the port configuration registers at intervals during runtime.

Workaround

The workaround is based on software checking the level on the interrupt pin before and after writing to the PxOR or PxDDR registers. If there is a level change (depending on the sensitivity programmed for this pin) the interrupt routine is invoked using the call instruction with three extra PUSH instructions before executing the interrupt routine (this is to make the call compatible with the IRET instruction at the end of the interrupt service routine).

But detection of the level change does not make sure that edge occurs during the critical 1 cycle duration and the interrupt has been missed. This may lead to occurrence of same interrupt twice (one hardware and another with software call).

To avoid this, a semaphore is set to '1' before checking the level change. The semaphore is changed to level '0' inside the interrupt routine. When a level change is detected, the semaphore status is checked and if it is '1' this means that the last interrupt has been missed. In this case, the interrupt routine is invoked with the call instruction.

There is another possible case i.e. if writing to PxOR or PxDDR is done with global interrupts disabled (interrupt mask bit set). In this case, the semaphore is changed to '1' when the level change is detected. Detecting a missed interrupt is done after the global interrupts are enabled (interrupt mask bit reset) and by checking the status of the semaphore. If it is '1' this means that the last interrupt was missed and the interrupt routine is invoked with the call instruction.

To implement the workaround, the following software sequence is to be followed for writing into the PxOR/PxDDR registers. The example is for for Port PF1 with falling edge interrupt sensitivity. The software sequence is given for both cases (global interrupt disabled/enabled).

Case 1: Writing to PxOR or PxDDR with Global Interrupts Enabled:

```
LD A,#01
LD sema,A      ; set the semaphore to '1'
LD A,PFDR
AND A,#02
LD  X,A        ; store the level before writing to PxOR/PxDDR
LD A,#$90
LD PFDDR,A     ; Write to PFDDR
LD A,#$ff
LD  PFOR,A     ; Write to PFOR
LD A,PFDR
AND A,#02
LD Y,A         ; store the level after writing to PxOR/PxDDR
LD A,X         ; check for falling edge
cp A,#02
jrne OUT
TNZ Y
jrne OUT
LD A,sema      ; check the semaphore status if edge is detected
CP A,#01
jrne OUT
call call_routine; call the interrupt routine
OUT:LD A,#00
LD sema,A
.call_routine  ; entry to call_routine
PUSH A
PUSH X
PUSH CC
.ext1_rt       ; entry to interrupt routine
LD A,#00
LD sema,A
IRET
```

Case 2: Writing to PxOR or PxDDR with Global Interrupts Disabled:

```
SIM          ; set the interrupt mask
LD A,PFDR
AND A,#$02
LD  X,A      ; store the level before writing to PxOR/PxDDR
LD A,#$90
```

```
LD PFDDR,A; Write into PFDDR
LD A,#$ff
LD PFOR,A          ; Write to PFOR
LD A,PFDR
AND A,#$02
LD Y,A       ; store the level after writing to PxOR/
PxDDR
LD A,X       ; check for falling edge
cp A,#$02
jrne OUT
TNZ Y
jrne OUT
LD A,#$01
LD sema,A  ; set the semaphore to '1' if edge is
detected
RIM            ; reset the interrupt mask
LD A,sema  ; check the semaphore status
CP A,#$01
jrne OUT
call call_routine; call the interrupt routine
RIM
OUT:         RIM
JP while_loop
.call_routine ; entry to call_routine
PUSH A
PUSH X
PUSH CC
.ext1_rt      ; entry to interrupt routine
LD A,#$00
LD sema,A
IRET
```

16.1.1 Unexpected Reset Fetch

If an interrupt request occurs while a "POP CC" instruction is executed, the interrupt controller does not recognise the source of the interrupt and, by default, passes the RESET vector address to the CPU.

Workaround

To solve this issue, a "POP CC" instruction must always be preceded by a "SIM" instruction.

16.2 Clearing active interrupts outside interrupt routine

When an active interrupt request occurs at the same time as the related flag is being cleared, an unwanted reset may occur.

Note: clearing the related interrupt mask will not generate an unwanted reset

Concurrent interrupt context

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

when:

– The interrupt flag is cleared within its own interrupt routine

– The interrupt flag is cleared within any interrupt routine

– The interrupt flag is cleared in any part of the code while this interrupt is disabled

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

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

Example:

```
SIM
reset interrupt flag
RIM
```

Nested interrupt context:

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

when:

– The interrupt flag is cleared within its own interrupt routine

– The interrupt flag is cleared within any interrupt routine with higher or identical priority level

– The interrupt flag is cleared in any part of the code while this interrupt is disabled

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

```
PUSH CC
SIM
reset interrupt flag
POP CC
```

16.3 16-bit Timer PWM Mode

In PWM mode, the first PWM pulse is missed after writing the value FFFCh in the OC1R register (OC1HR, OC1LR). It leads to either full or no PWM during a period, depending on the OLVL1 and OLVL2 settings.

16.4 SCI Wrong Break duration

Description

A single break character is sent by setting and resetting the SBK bit in the SCICR2 register. In some cases, the break character may have a longer duration than expected:

- 20 bits instead of 10 bits if M=0

- 22 bits instead of 11 bits if M=1

In the same way, as long as the SBK bit is set, break characters are sent to the TDO pin. This may lead to generate one break more than expected.

Occurrence

The occurrence of the problem is random and proportional to the baudrate. With a transmit frequency of 19200 baud (f_{CPU}=8MHz and SCIBRR=0xC9), the wrong break duration occurrence is around 1%.

Workaround

If this wrong duration is not compliant with the communication protocol in the application, software can request that an Idle line be generated before the break character. In this case, the break duration is always correct assuming the application is not doing anything between the idle and the break. This can be ensured by temporarily disabling interrupts.

The exact sequence is:

- Disable interrupts

- Reset and Set TE (IDLE request)

- Set and Reset SBK (Break Request)

- Re-enable interrupts

16.5 In-Application Programming

Not available on the first silicon revision currently in production (rev Z). This limitation will be corrected on the next silicon revision. Refer to Table 35 Silicon revision identification.

16.6 Programming of EEPROM data

Description

In user mode, when programming EEPROM data memory, the read access to the program memory between E000h and FFFFh can be corrupted.

Impact on application

The EEPROM programming routine must be located outside this program memory area.

Any access to the interrupt vector table can result in an unexpected code being executed, so the interrupts must be masked.

Workaround

The sequence to program the EEPROM data (refer to Section 5.3 on page 19) must be executed within C000h-DFFFh area or from the RAM. It is as follows:

set E2LAT bit

write up to 32 bytes in E2PROM area

SIM ; to disable the interrupts

set E2PGM bit

wait for E2PGM=0

RIM ; to enable the interrupts

return to the program memory

16.7 Flash Write/Erase Protection

Not available on the first silicon revision currently in production (rev Z). This limitation will be corrected on the next silicon revision. Refer to Table 35 Silicon revision identification.

Table 35. Silicon revision identification

Device	Status	Trace code marked on device	internal sales types on box label
ST72F344xxxx ST72F345xxxx	In Production	“xxxxxxxxZ”	72F344xxxx$x2 72F345xxxx$x2
	Under qualification	“xxxxxxxxX”	72F344xxxx$x4 72F345xxxx$x4

17 REVISION HISTORY

Date	Revision	Main changes
29-April-2006	1	First release on internet
23-Oct-2006	2	Removed references to BGA56 and QFN40 packages TQFP package naming changed to LQFP (Low-profile Quad Flat) Changed number of I/O ports on first page PDVD (Power Down Voltage Detector) replaced by AVD (Auxiliary Voltage Detector) Modified note 3 to Table 2 on page 12 Added PF4 to Figure 3 on page 6 and Figure 4 on page 7 "MEMORY ACCESS" on page 19 Modified Figure 8, Figure 9 on page 20 and Figure 10 on page 21 Changed RCCR table in Section 7.2 on page 29 (f_{RC}=1MHz) References to PDVDF, PDVDIE corrected to AVDF, AVDIE: Section 7.5.2 on page 34 Current characteristics Section 13.2.2 on page 153 updated General operating conditions table updated, Section 13.3.1 on page 154 Data updated in Section 13.3.2 on page 154, note replaced Table modified in Section 13.3.3 on page 155 Notes adjusted for table in Section 13.4 on page 156 Modified Section 13.5 on page 156 (for V_{DD}=5V) Table in Section 13.6.1 on page 157 modified Updated Section 13.6.2 on page 159 Added Section 13.7.2 and Figure 93 on page 160 Table in Section 13.8.2 on page 163 modified Absolute maximum ratings and electrical sensitivity table updated, Section 13.9.3 on page 165 Added note 1 to V_{IL} and V_{IH} in Section 13.10.1 on page 166 Table in Section 13.10.2 on page 167 modified (for V_{DD}= 3.3V and V_{DD}=2.7V) Modified graphs in Section 13.10.2 on page 167 $t_{g(RSTL)in}$ updated in Section 13.11 on page 172 Updated Table 29 on page 174 Updated Table 30 on page 180 Modified default values for option byte 2 and 3 on page 183 Added option list on page 185 Added "DEVELOPMENT TOOLS" on page 186 Added known limitations: "In-Application Programming" on page 189, "Programming of EEPROM data" on page 189, and "Flash Write/Erase Protection" on page 189 Modified Section 16.6 on page 189 Changed status of the document (datasheet instead of preliminary data)

Notes:

STMicroelectronics group of companies

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

www.st.com