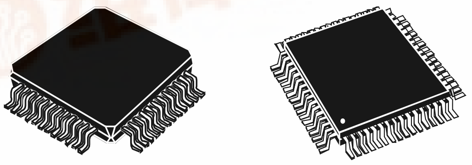
ST7267

USB 2.0 HIGH SPEED MASS STORAGE MICROCONTROLLER

- **USB 2.0 Interface compliant with Mass Storage Device Class**
  – Integrated USB 2.0 PHY
  – Supports USB High Speed and Full Speed
  – 1 control endpoint with two 64-byte buffers
  – 1 IN 64-byte bulk / interrupt endpoint
  – 1 OUT 64-byte bulk / interrupt endpoint
  – 1 IN 512-byte double buffer bulk endpoint
  – 1 OUT 512-byte double buffer bulk endpoint
  – Suspend and Resume operations
- **Mass Storage Controller Interface (MSCI)**
  – 16-bit RISC Core
  – Supports all types of NAND Flash devices
  – Reed-Solomon Encoder/Decoder for MLC NAND Flash support: on-the-fly correction (4 bytes of a 512-byte block)
- **Memories**
  – 54K of ROM
  – 4 Kbytes of RAM with up to 256 bytes stack
  – 2 Kbytes of MSCI CODE RAM
  – 5 Kbytes of dual-ported RAM
- **Embedded 8-bit ST7 MCU**
- **Supply Management**
  – 3.3V operation
  – Integrated 3.3V-1.8V voltage regulator
- **Clock Management**
  – Integrated PLL for generating core and USB 2.0 clock sources using an external 12 MHz crystal



- **Interrupt Management**
  – 11 Interrupt vectors plus TRAP and RESET
  – 40 I/Os interrupt source mapped on 5 vectors
  – Nested interrupt management
- **I/O Ports**
  – Up to 40 general purpose I/O port pins
  – Two 5V tolerant I/Os
- **Communication Interface**
  – 1 SPI Synchronous serial interface
- **Timers**
  – Configurable Watchdog for system reliability
  – 16-bit timer
  – Time Base Unit
- **TQFP48 7x7 and TQFP64 10x10 lead-free packages**
- **Development Support**
  – Complete reference design including BOM and gerber files
  – Supports Windows ME, Windows 2K, Windows XP. Drivers available for Windows SE
  – Complete application package available to design a USB 2.0 Flash Drive application

Features	ST7267C8	ST7267R8
Program memory	54K	
User RAM (stack) - bytes	4K (256)	
Peripherals	WDG, TBU, Timer, SPI, MSCI	
USB interface	USB 2.0	
# of NAND devices supported	4	
Operating Supply	2.7V to 3.6V	
Operating Temperature	0°C to +70°C	
Packages	TQFP48 7x7	TQFP64 10x10

Rev. 2

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

1 INTRODUCTION

The ST7267 is a USB 2.0 highspeed Mass Storage microcontroller. The USB 2.0 highspeed interface including PHY and function supports USB 2.0 Mass Storage Device Class.

The Mass Storage Controller Interface (MSCI) features a 16-bit RISC ALU core combined with the Reed-Solomon Encoder/Decoder on-the-fly correction on 512 data byte blocks provides a flexible, high transfer rate solution for interfacing all types of NAND Flash memory devices.

The internal 60 MHz PLL driven by the 12 MHz oscillator is used to generate the 480 MHz frequency for the USB 2.0 PHY.

The ST7 CPU runs the application program from the internal ROM and RAM. USB data and patch code are stored in internal RAM.

The I/O ports provide functions for EEPROM connection, LEDs and write protect switch control.

The internal 3.3V to 1.8V voltage regulator provides the 1.8V supply voltage to the digital part of the circuit.

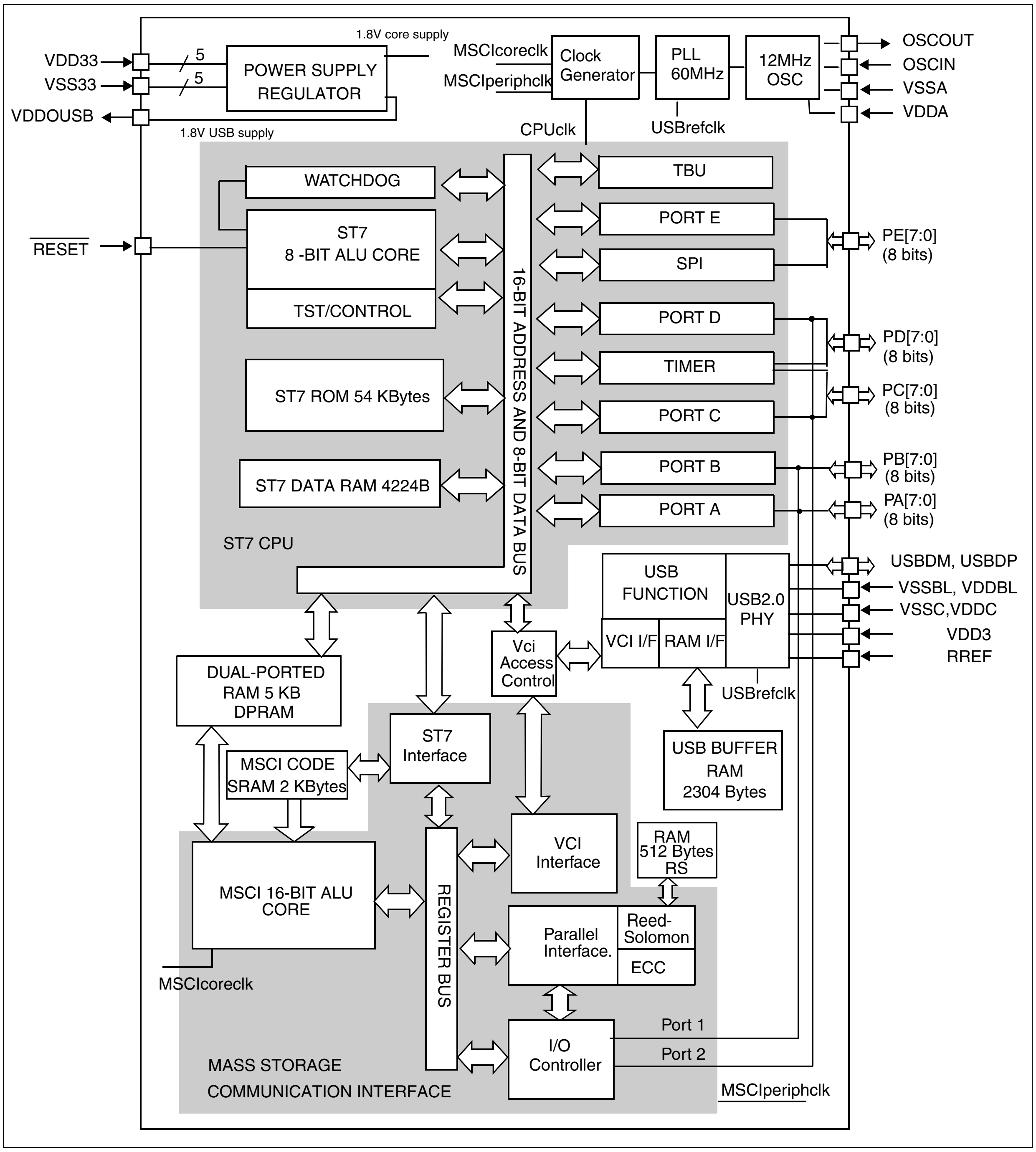
1.1 Related Documentation

For details on the programming model of the ST7 CPU and the MSCI, please refer to the following manuals:

- Mass Storage Controller Interface MSCI 16-bit Core Programming Manual
- ST7 Programming Manual

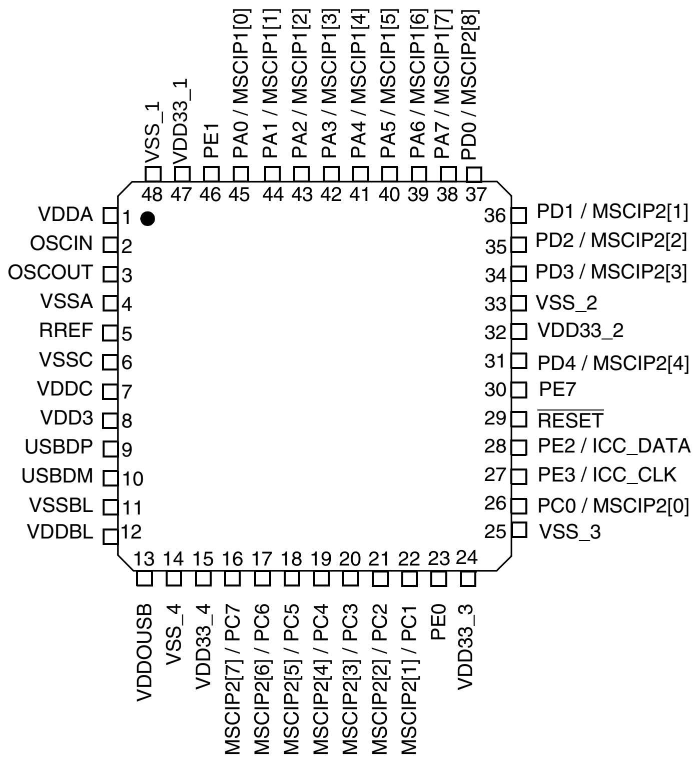
INTRODUCTION (Cont'd)

Figure 1. Device Block Diagram



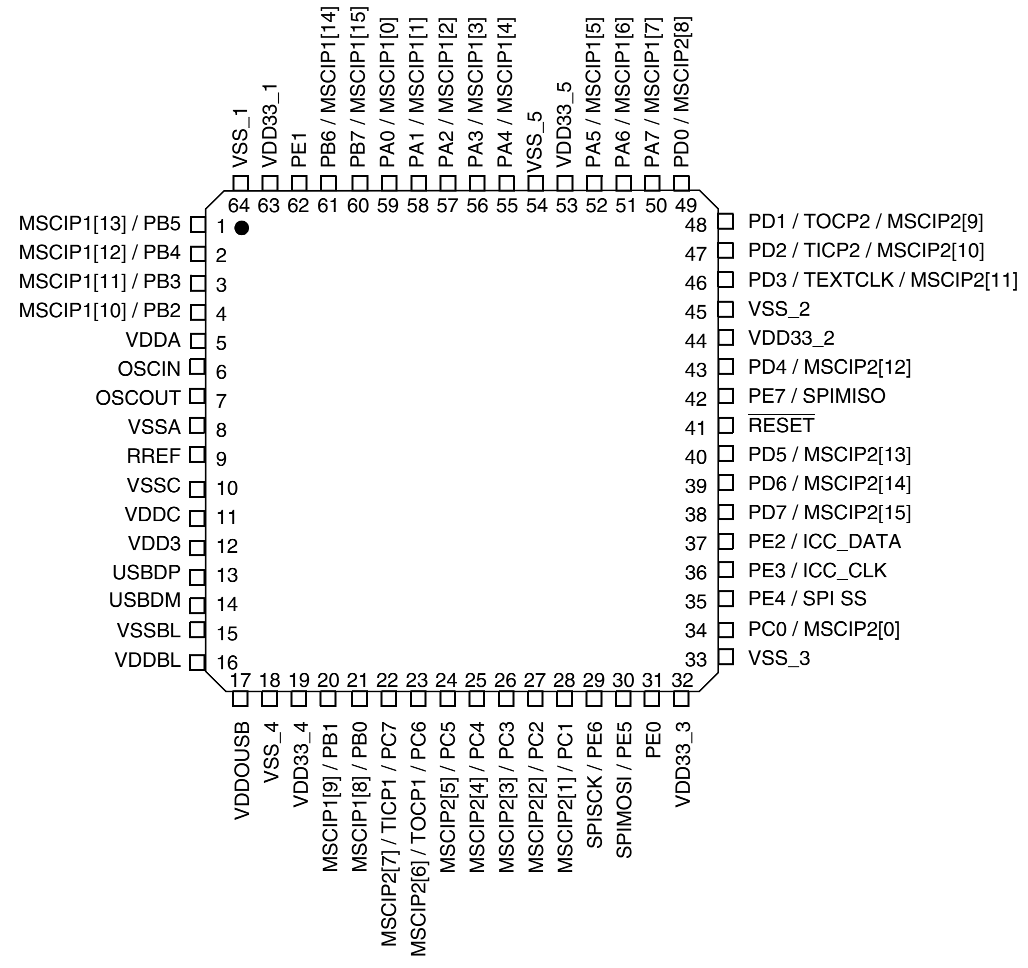
2 PIN DESCRIPTION

Figure 2. 48-Pin TQFP Package Pinout



PIN DESCRIPTION (Cont'd)

Figure 3. 64-Pin TQFP Package Pinout



PIN DESCRIPTION (Cont'd)

Legend / Abbreviations for tables 2 thru 6:

Type:	I = input, O = output, S = supply
Input level:	A = Dedicated analog input
In/Output level:r	C_T = CMOS $0.3V_{DD}/0.7V_{DD}$ with input trigger
	T_T= TTL 0.8V / 2V with Schmitt trigger
Output level:	D8 = 8mA drive
	D4 = 4mA drive
	D2 = 2mA drive

Port and control configuration:

– Input:float = floating, wpu = weak pull-up, wpd = weak pull-down, int = interrupt

– Output: OD = pseudo open drain, PP = push-pull

Table 1. Power Supply

Pin		Pin Name	Type	Description
TQFP48	TQFP64			
48	64	VSS_1	S	Ground
47	63	VDD33_1	S	I/Os and Regulator supply voltage
33	45	VSS_2	S	Ground
32	44	VDD33_2	S	I/Os and Regulator supply voltage
25	33	VSS_3	S	Ground
24	32	VDD33_3	S	I/Os and Regulator supply voltage
14	18	VSS_4	S	Ground
15	19	VDD33_4	S	I/Os and Regulator supply voltage
	54	VSS_5	S	Ground
	53	VDD33_5	S	I/Os supply voltage
13	17	VDDOUSB	S	USB PHY, OSC and PLL power supply output (1.8V)

Table 2. Control & System

Pin		Pin Name	Type	Power	Description
TQFP48	TQFP64				
29	41	$\overline{\text{RESET}}$	I	3.3	Reset input with filter and pull-up

PIN DESCRIPTION (Cont'd)

Table 3. USB 2.0 Interface

Pin		Pin Name	Type	Description
TQFP48	TQFP64			
12	16	VDDBL	S	Supply voltage for buffers and deserialisation ffs (1.8V)
11	15	VSSBL	S	Ground for buffers and deserialisation ffs (1.8V)
10	14	USBDM	I/O	USB DATA -
9	13	USBDP	I/O	USB DATA +
8	12	VDD3	S	Supply voltage for the FS compliance (3.3V)
7	11	VDDC	S	Supply voltage for DLL & xor tree (1.8V)
6	10	VSSC	S	Ground for DLL & xor tree (1.8V)
5	9	RREF	A	Ref. resistor for integrated impedances process adapt (11.5kohms 1% Pull Down)

Table 4. USB 2.0 and core Clock System

Pin		Pin Name	Type	Description
TQFP48	TFQP64			
4	8	VSSA	S	Ground for osc & PLL (1.8V)
3	7	OSCOUT	O	12MHz oscillator output
2	6	OSCIN	I	12MHz oscillator input
1	5	VDDA	S	Supply voltage for osc & PLL (1.8V)

PIN DESCRIPTION (Cont'd)

Table 5. General Purpose I/O Ports / Mass Storage I/Os

Pin		Pin Name	Type	5V tolerant	Level		Configuration					Main function (after reset)	Alternate function	
TQFP48	TQFP64				Input	Outputs	Input			Output			ALT1	ALT2
							float	wpu	int	OD	PP			
45	59	PA0	I/O		T_T	D4	X	X	ei0	X	X	Port A0		MSCIP1[0]
44	58	PA1	I/O		T_T	D4	X	X		X	X	Port A1		MSCIP1[1]
43	57	PA2	I/O		T_T	D4	X	X		X	X	Port A2		MSCIP1[2]
42	56	PA3	I/O		T_T	D4	X	X		X	X	Port A3		MSCIP1[3]
41	55	PA4	I/O		T_T	D4	X	X		X	X	Port A4		MSCIP1[4]
40	52	PA5	I/O		T_T	D4	X	X		X	X	Port A5		MSCIP1[5]
39	51	PA6	I/O		T_T	D4	X	X		X	X	Port A6		MSCIP1[6]
38	50	PA7	I/O		T_T	D4	X	X		X	X	Port A7		MSCIP1[7]
	21	PB0 (2)	I/O		T_T	D4	X	X	ei1	X	X	Port B0		MSCIP1[8]
	20	PB1 (2)	I/O		T_T	D4	X	X		X	X	Port B1		MSCIP1[9]
	4	PB2 (2)	I/O		T_T	D4	X	X		X	X	Port B2		MSCIP1[10]
	3	PB3 (2)	I/O		T_T	D4	X	X		X	X	Port B3		MSCIP1[11]
	2	PB4 (2)	I/O		T_T	D4	X	X		X	X	Port B4		MSCIP1[12]
	1	PB5 (2)	I/O		T_T	D4	X	X		X	X	Port B5		MSCIP1[13]
	61	PB6 (2)	I/O		T_T	D4	X	X		X	X	Port B6		MSCIP1[14]
	60	PB7 (2)	I/O		T_T	D4	X	X		X	X	Port B7		MSCIP1[15]
26	34	PC0	I/O		T_T	D8	X	X	ei2	X	X	Port C0		MSCIP2[0]
22	28	PC1	I/O		T_T	D8	X	X		X	X	Port C1		MSCIP2[1]
21	27	PC2	I/O		T_T	D8	X	X		X	X	Port C2		MSCIP2[2]
20	26	PC3	I/O		T_T	D8	X	X		X	X	Port C3		MSCIP2[3]
19	25	PC4	I/O		T_T	D4	X	X		X	X	Port C4		MSCIP2[4]
18	24	PC5	I/O		T_T	D4	X	X		X	X	Port C5		MSCIP2[5]
17	23	PC6	I/O		T_T	D4	X	X		X	X	Port C6	Timer OCP1	MSCIP2[6]
16	22	PC7	I/O		T_T	D4	X	X		X	X	Port C7	Timer ICP1	MSCIP2[7]
37	49	PD0	I/O		T_T	D2	X	X	ei3	X	X	Port D0		MSCIP2[8]
36	48	PD1	I/O		T_T	D2	X	X		X	X	Port D1	Timer OCP2	MSCIP2[9]
35	47	PD2	I/O		T_T	D2	X	X		X	X	Port D2	Timer ICP2	MSCIP2[10]
34	46	PD3	I/O		T_T	D2	X	X		X	X	Port D3	Timer EXTCLK	MSCIP2[11]
31	43	PD4	I/O		T_T	D2	X	X		X	X	Port D4		MSCIP2[12]
	40	PD5 (2)	I/O		T_T	D2	X	X		X	X	Port D5		MSCIP2[13]
	39	PD6 (2)	I/O		T_T	D2	X	X		X	X	Port D6		MSCIP2[14]
	38	PD7 (2)	I/O		T_T	D2	X	X		X	X	Port D7		MSCIP2[15]

Pin		Pin Name	Type	5V tolerant	Level		Configuration					Main function (after reset)	Alternate function	
							Input			Output				
TQFP48	TQFP64				Input	Outputs	float	wpu	int	OD	PP		ALT1	ALT2
23	31	PE0	I/O		T_T	D8	X	X	ei4	X	X	Port E0		
46	62	PE1	I/O		T_T	D8	X	X		X	X	Port E1		
28	37	PE2	I/O		T_T	D8	X	X		X	X	Port E2	ICC_DATA	
27	36	PE3 (1)	I/O		T_T	D8	X	X		X	X	Port E3	ICC_CLK	
	35	PE4 (2)	I/O		T_T	D2	X	X		X	X	Port E4	SPI SS	
	30	PE5 (2)	I/O		T_T	D2	X	X		X	X	Port E5	SPI MOSI	
	29	PE6 (2)	I/O	X	T_T	D2	X	X		X	X	Port E6	SPI SCK	
30	42	PE7	I/O	X	T_T	D2	X	X		X	X	Port E7	SPI MISO	

Notes:

1. Caution: during normal operation this pin must be pulled-up, internally or externally. This is to avoid entering ICC mode unexpectedly during a reset. In the application, even if the pin is configured as output, any reset will put it back in pull-up

2. Ports unavailable in the 48-pin packages (PB7:0, PD7:5, PE6:4) are forced to input mode with internal pull-up activated to avoid possible floating I/O consumption.

PIN DESCRIPTION (Cont'd)

Figure 4. NAND Flash Drive Application Example (TQFP64: parallel access through up to 4 CEs)

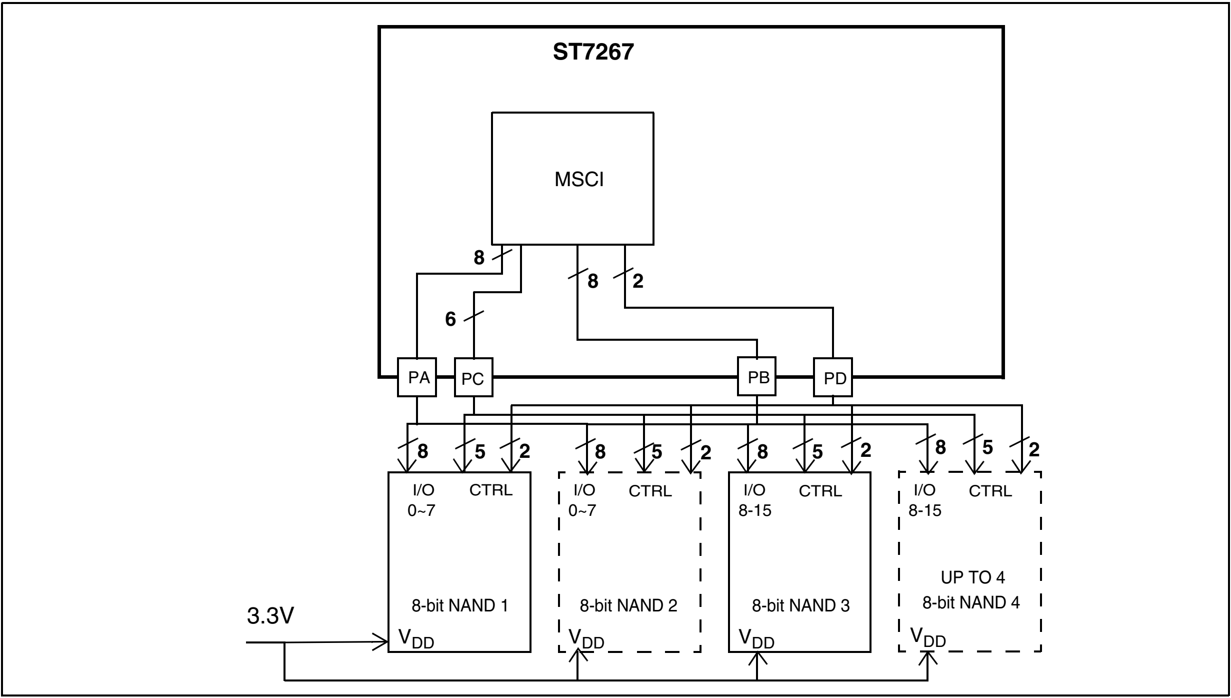


Table 6. NAND Interface Pin Assignment for different applications

NAND 16-bit parallel access through 4 CEs [(1)]	IO[0-7]	IO[8-15]	ALE	CLE	WE	RE	CE1	CE2	CE3	CE4	RnB	WP
ST7267 pins	PA0-7	PB0-7	PC0	PC1	PC2	PC3	PC4	PC5 [(2)]	PC6 [(2)]	PC7 [(2)]	PD0	PD1

Notes:

1. when 8-bit NANDs are connected, the TQFP48 package can be used
2. if only one NAND is used, these I/Os are free for other applications

PIN DESCRIPTION (Cont'd)

Figure 5. NAND Flash Drive Application Example (TQFP64: 16-bit parallel access on 8-bit NAND)

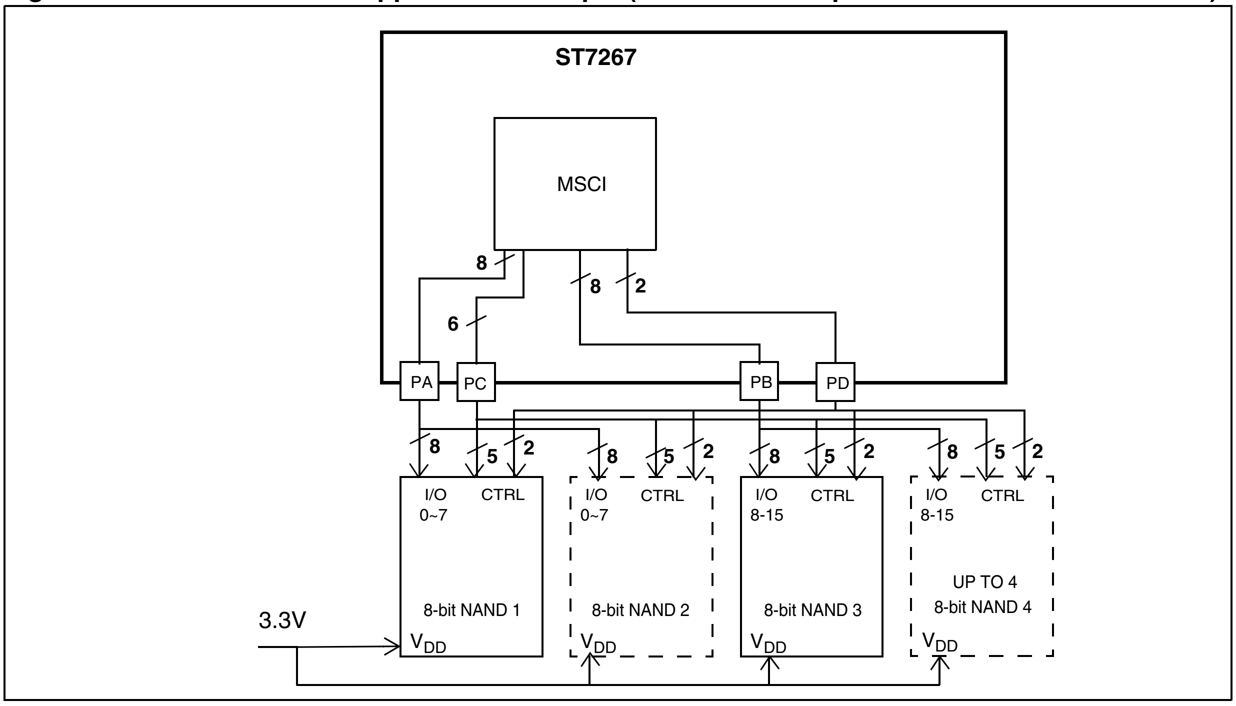


Table 7. NAND Interface Pin Assignment (2 or 4 8-bit NANDs)

NAND Pin	I/O0-7	ALE	CLE	WE	RE	CE1	I/O8-15	CE2	RnB	WP
ST7267 pin	PA0-7	PC0	PC1	PC2	PC3	PC4	PB0-7	PC5(1)	PD0	PD1

Note:
1. When two NANDs have to be handled, this I/O is free for other functions. Pins from different chips should be tied together (i.e. CLE1 with CLE2...).

3 ST7 REGISTER & MEMORY MAP

As shown in Figure 6, the ST7 core is capable of addressing 64K bytes of memories and I/O registers.

The available memory locations consist of:

- 80 bytes of register locations
- 4 Kbytes of ST7 DATA RAM (including up to 256 bytes for the stack from 0100h to 01FFh).
- 54 Kbytes of ST7 CODE ROM program memory
- The highest address bytes contain the user reset and interrupt vectors in ROM which are remapped in ST7 DATA RAM.

Two memory spaces are addressable by both the ST7 and the MSCI

- 2 Kbytes of MSCI CODE RAM
- 5 Kbytes of dual-ported RAM

3.1 Paged Memory Space

The MSCI CODE RAM and the DPRAM are mapped in the same address range on the ST7 bus (1100h to 24FFh).

During initialisation, the MSCI program code has to be loaded in the MSCI CODE RAM by the ST7

To do this, set the RAMLD bit in the MSCI Control Register (MCR). This can only be done while the MSCI is reset state. Refer to section 14.2.3 on page 116 for details.

When the MSCI code is loaded, clear the RAMLD bit to disable any further access to MSCI CODE RAM from the ST7 bus.

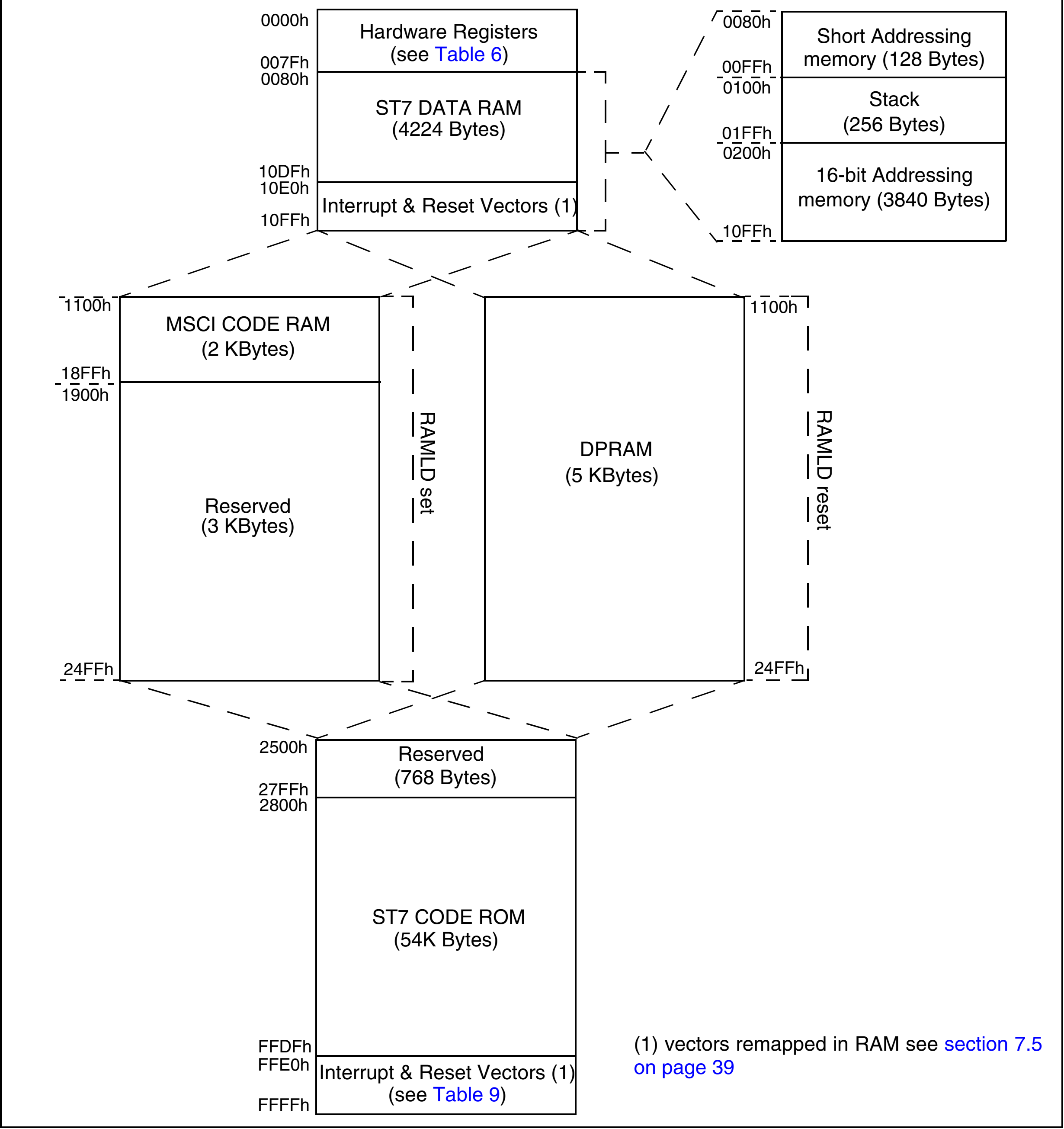
For a description of the MSCI Register and Memory Map, refer to section 12 on page 111.

3.2 Interrupt and Vector Remapping

For flexibility, the interrupt and reset vectors can be mapped in RAM. See section 7.5 on page 39.

IMPORTANT: Memory locations noted “Reserved” must never be accessed. Accessing a reserved area generates a hardware reset of the device.

Figure 6. ST7 Memory Map



(1) vectors remapped in RAM see section 7.5 on page 39

Table 8. Hardware Register Memory Map

@	Block	Register Label	Register name	Reset Status	Remarks
0000h	USBHS	PWRR	PoWeR management Register	20h	r/w
0001h		FADDR	Function ADDress Register	00h	r/w
0002h			Not used. Always return 00h		
0003h		ITINR	Interrupt EP0 and IN EP Register	00h	r
0004h			Not used. Always return 00h		
0005h		ITOUTR	Interrupt OUT EP Register	00h	r
0006h			Not used. Always return 00h		
0007h		ITINER	Interrupt IN Enable Register	07h	r/w
0008h			Not used. Always return 00h		
0009h		ITOUTER	Interrupt OUT Enable Register	06h	r/w
000Ah		ITUSBER	Interrupt USB Enable Register	06h	r/w
000Bh		ITUSBR	Interrupt USB Register	00h	r
000Ch		FRNBRM	FRame NumBer Register (MSB)	00h	r
000Dh		FRNBRL	FRame NumBer Register (LSB)	00h	r
000Eh		TSTMODE	TeST MODEs	00h	r/w
000Fh		INDEXR	INDEX Register	00h	r/w
0010h		INMAXPRM	IN EP n Max Pkt size Register (MSB)	00h	r/w
0011h		INMAXPRL	IN EP n Max Pkt size Register (LSB)	00h	r/w
0012h		INCSRM	IN EP n Control Status Register (MSB)	00h	r/w
0013h		INCSRL	Control Status Reg for EP0 or IN EP n (LSB)	00h	r/w
0014h		OUTMAXPRM	OUT EP n Max Pkt size Register (MSB)	00h	r/w
0015h		OUTMAXPRL	OUT EP n Max Pkt size Register (LSB)	00h	r/w
0016h		OUTCSRM	OUT EP n Control Status Register (MSB)	00h	r/w
0017h		OUTCSRL	OUT EP n Control Status Register (LSB)	00h	r/w
0018h		OUTCNTRM	OUT EP n Count Register (MSB)	00h	r
0019h		OUTCNTRL	OUT EP n Count Register (LSB)	00h	r
001Ah to 001Fh	Reserved (6 Bytes)				
0020h 0021h	USBHS	EP0DR	Endpoint 0 Data Register	xxh	r/w
0022h 0023h		EP1DR	Endpoint 1 Data Register	xxh	r/w
0024h 0025h		EP2DR	Endpoint 2 Data Register	xxh	r/w
0026h	ST7 I/O Ports	PADR	Port A Data Register	00h(1)	r/w
0027h		PADDR	Port A Data Direction Register	00h	r/w
0028h		PAOR	Port A Option Register	00h	r/w
0029h		PBDR	Port B Data Register	00h(1)	r/w
002Ah		PBDDR	Port B Data Direction Register	00h	r/w
002Bh		PBOR	Port B Option Register	00h	r/w

@	Block	Register Label	Register name	Reset Status	Remarks
002Ch	ST7 I/O Ports	PCDR	Port C Data Register	00h(1)	r/w
002Dh		PCDDR	Port C Data Direction Register	00h	r/w
002Eh		PCOR	Port C Option Register	00h	r/w
002Fh		PDDR	Port D Data Register	00h(1)	r/w
0030h		PDDDR	Port D Data Direction Register	00h	r/w
0031h		PDOR	Port D Option Register	00h	r/w
0032h		PEDR	Port E Data Register	00h(1)	r/w
0033h		PEDDR	Port E Data Direction Register	00h	r/w
0034h		PEOR	Port E Option Register	00h	r/w
0035h	WDG	WDGCR	Watchdog Control Register	7Fh	r/w
0036h					
0037h	ITC	ISPR0	Interrupt Software Priority Register 0	FFh	r/w
0038h		ISPR1	Interrupt Software Priority Register 1	FFh	r/w
0039h		ISPR2	Interrupt Software Priority Register 2	FFh	r/w
003Ah		ISPR3	Interrupt Software Priority Register 3	FFh	r/w
003Bh		EICR0	External Interrupt Control Register 0	00h	r/w
003Ch		EICR1	External Interrupt Control Register 1	00h	r/w
003Dh		PAEIENR	Port A External Interrupt Enable register	00h	r/w
003Eh		PBEIENR	Port B External Interrupt Enable register	00h	r/w
003Fh		PCEIENR	Port C External Interrupt Enable register	00h	r/w
0040h		PDEIENR	Port D External Interrupt Enable register	00h	r/w
0041h		PEEIENR	Port E External Interrupt Enable register	00h	r/w
0042h		PAEISR	Port A External Interrupt Status Register	00h	r/w
0043h		PBEISR	Port B External Interrupt Status Register	00h	r/w
0044h		PCEISR	Port C External Interrupt Status Register	00h	r/w
0045h		PDEISR	Port D External Interrupt Status Register	00h	r/w
0046h		PEEISR	Port E External Interrupt Status Register	00h	r/w
0047h	TIMER	TCR2	Timer Control Register 2	00h	r/w
0048h		TCR1	Timer Control Register 1	00h	r/w
0049h		TCSR	Timer Control/Status Register	xxh	r
004Ah		TIC1HR	Timer Input Capture 1 High Register	xxh	r
004Bh		TIC1LR	Timer Input Capture 1 Low Register	xxh	r
004Ch		TOC1HR	Timer Output Compare 1 High Register	80h	r/w
004Dh		TOC1LR	Timer Output Compare 1 Low Register	00h	r/w
004Eh		TCHR	Timer Counter High Register	FFh	r
004Fh		TCLR	Timer Counter Low Register	FCh	r
0050h		TACHR	Timer Alternate Counter High Register	FFh	r
0051h		TACLR	Timer Alternate Counter Low Register	FCh	r
0052h		TIC2HR	Timer Input Capture 2 High Register	xxh	r
0053h		TIC2LR	Timer Input Capture 2 Low Register	xxh	r
0054h		TOC2HR	Timer Output Compare 2 High Register	80h	r/w
0055h		TOC2LR	Timer Output Compare 2 Low Register	00h	r/w
0056h	MISC	MISCR1	Miscellaneous Register 1	00h	r/w

@	Block	Register Label	Register name	Reset Status	Remarks
0057h	Reserved (1 Byte)				
0058h 0059h	TBU	TBUCVR TBUCSR	TBU Counter Value Register TBU Control/Status Register	00h 00h	r/w r/w
005Ah	Reserved (1 Byte)				
005Bh 005Ch 005Dh 005Eh	CKGEN	CCMR CELSPCR CEHSPCR	CKGEN Control Mode Register CKGEN Enable of Low Speed Periph. CLK Reg. CKGEN Enable of High Speed Periph. CLK Reg.	00h 00h 00h	r/w r/w r/w
005Fh 0060h	EOS	EOSSR EOSCR	End Of Suspend Status Register End Of Suspend Control Register	00h 00h	r/w r/w
0061h 0062h 0063h	SPI	SPIDR SPICR SPICSR	SPI Data I/O register SPI Control Register SPI Control/Status Register	xxh 0xh 00h	r/w r/w r
0064h to 0068h	Reserved (5 Bytes)				
0069h 006Ah 006Bh 006Ch 006Dh 006Eh 006Fh	MSCI	MCR MSR MPCM MPCL MCRCH MCRCL	MSCI Control Register MSCI Status Register MSCI PC register (MSB) MSCI PC register (LSB) MSCI CRC (MSB) MSCI CRC (LSB)	01h 00h 00h 00h 00h 00h	r/w r/w r/w r/w r r
0070h to 0073h	Reserved (4 Bytes)				
0074h 0075h 0076h 0077h 0078h 0079h 007Ah 007Bh 007Ch	DM[(2)]	DMCR DMCSR DMBK1M DMBK1L DMBK2M DMBK2L DMCR2 DMCSR2 DMENFCT	Debug Module Control Register Debug Module Control / Status Register Debug Module BreaKpoint 1 register (MSB) Debug Module BreaKpoint 1 register (LSB) Debug Module BreaKpoint 2 register (MSB) Debug Module BreaKpoint 2 register (LSB) Debug Module Control Register 2 Debug Module Control / Status Register 2 Debug Module Enable Function register	00h 10h FFh FFh FFh FFh 00h 00h FFh	r/w r/w r/w r/w r/w r/w r/w r r/w

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. For a description of the registers of the Debug Module used for In-Circuit Debugging, see ICC reference manual.

Table 9. Interrupt Mapping

N°	Source Block	Description	Register Label	Priority Order	Exit from HALT	Address Vector
	RESET	Reset vector	N/A	Highest Priority	yes	FFFEh-FFFFh
	TRAP	Software Interrupt vector			no	FFFCh-FFFDh
0		NMI Interrupt				FFFAh-FFFBh
1	USB	USB Interrupt	see Note 1		no	FFF8h-FFF9h
2	EOS	USB End of Suspend Interrupt			yes	FFF6h-FFF7h
3	MSCI	MSCI interrupt	MSCI		no	FFF4h-FFF5h
4	EI0	External Interrupt Port A	N/A		yes	FFF2h-FFF3h
5	EI1	External Interrupt Port B	N/A		yes	FFF0h-FFF1h
6	EI2	External Interrupt Port C	N/A		yes	FFEEh-FFEFh
7	EI3	External Interrupt Port D	N/A		yes	FFECh-FFEDh
8	EI4	External Interrupt Port E	N/A		yes	FFEAh-FFEBh
9	SPI	SPI interrupt	SPICSR		yes	FFE8h-FFE9h
10	TIMER	Timer interrupt	T1SR	Lowest Priority	no	FFE6h-FFE7h
11	TBU	TimeBase Unit	TBUCSR		no	FFE4h-FFE5h

Note 1: please see USB chapter

4 ST7 CENTRAL PROCESSING UNIT

4.1 INTRODUCTION

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

4.2 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

4.3 CPU REGISTERS

The 6 CPU registers shown in Figure 7 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)

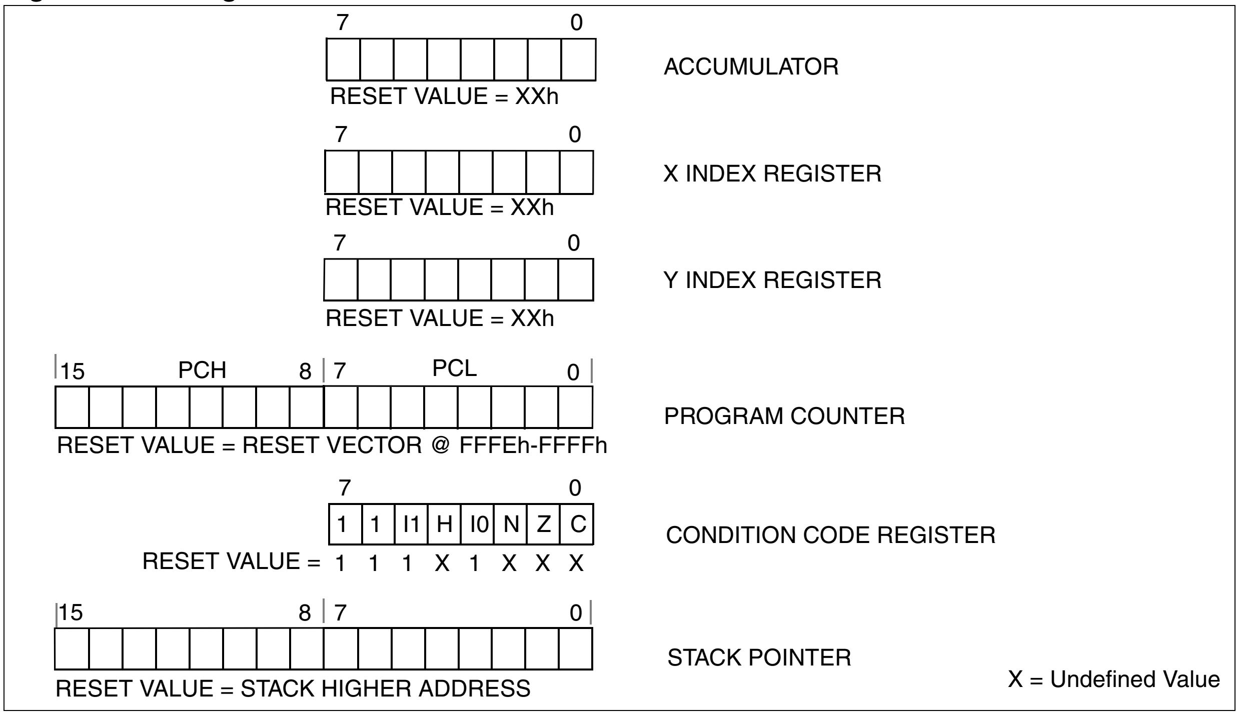
These 8-bit registers are used to create effective addresses or as temporary storage areas for data manipulation. (The Cross-Assembler generates a precede instruction (PRE) to indicate that the following instruction refers to the Y register.)

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 7. 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 7th bit.
0: The result of the last operation is positive or null.
1: The result of the last operation is negative
(i.e. the most significant bit is a logic 1).

This bit is accessed by the JRMI and JRPL instructions.

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

This bit is set and cleared by hardware. This bit indicates that the result of the last arithmetic, logical or data manipulation is zero.
0: The result of the last operation is different from zero.
1: The result of the last operation is zero.

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

Bit 0 = **C** *Carry/borrow.*

This bit is set and cleared by hardware and software. It indicates an overflow or an underflow has occurred during the last arithmetic operation.
0: No overflow or underflow has occurred.
1: An overflow or underflow has occurred.

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

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: 01FFh

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 8).

Since the stack is 256 bytes deep, the 8 most significant bits are forced by hardware. Following a CPU Reset, or after a Reset Stack Pointer instruction (RSP), the Stack Pointer contains its reset value (the SP6 to SP0 bits are set) which is the stack higher address.

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

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

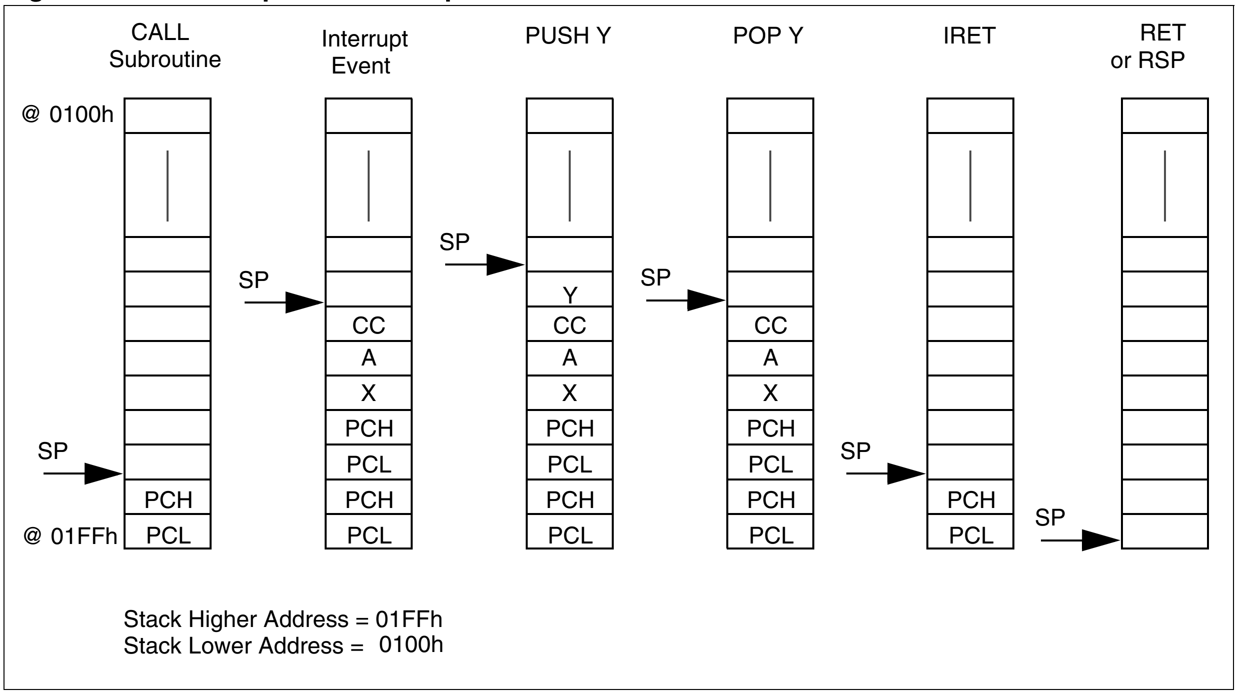
The stack is used to save the return address during a subroutine call and the CPU context during an interrupt. The user may also directly manipulate the stack by means of the PUSH and POP instructions. In the case of an interrupt, the PCL is stored at the first location pointed to by the SP. Then the other registers are stored in the next locations as shown in Figure 8.

– 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 8. Stack Manipulation Example



5 ST7 POWER SAVING MODES

5.1 INTRODUCTION

To give a large measure of flexibility to the application in terms of power consumption, three main power saving modes are implemented in the Device (see Figure 9):

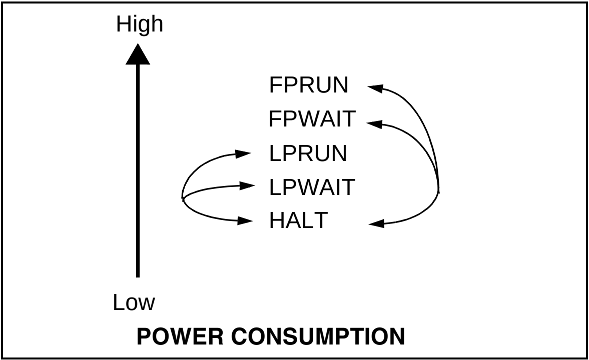
- Low Power Mode (PLL OFF)
- Wait
- Halt

After a RESET low power mode is selected by default. This mode drives the Device (CPU and embedded peripherals except USB) by means of a master clock which is based on the main oscillator frequency.

From this low power mode, different modes may be selected using specific CPU instruction.

Important note: Moreover, if the USB cell is not used, the UPO bit of the EOSCR register must be set to avoid any USB2 PHY consumption.

Figure 9. Power Saving Mode Transitions



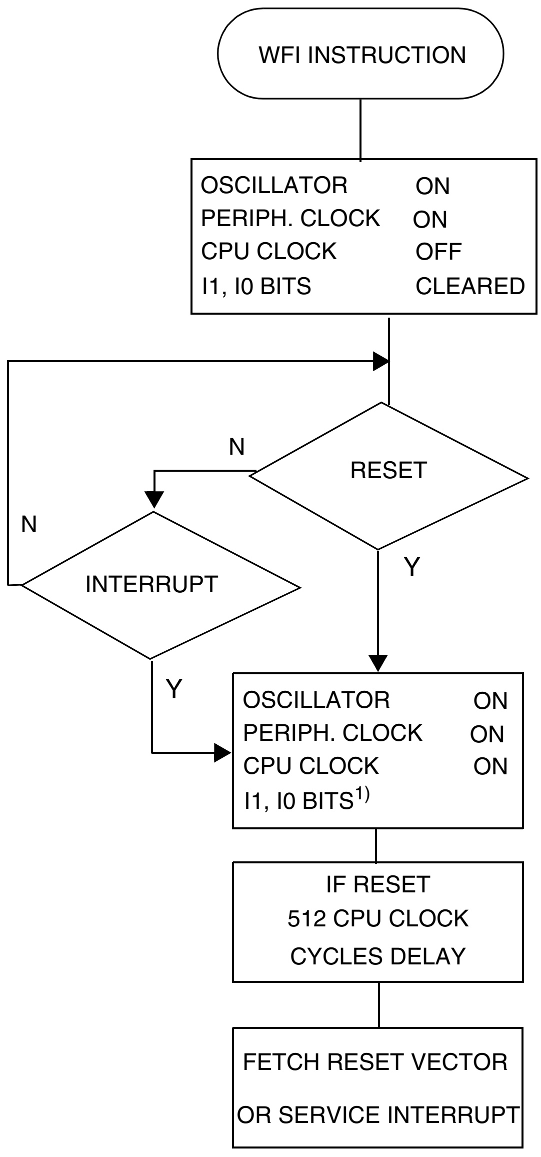
5.2 WAIT MODE

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

This power saving mode is selected by executing the "WFI" CPU instruction.

All peripherals remain active. During WAIT mode, the I bits in the CC register are forced to 0, enabling all interrupts. All other registers and memory remain unchanged. The Device remains in WAIT mode until an interrupt or reset occurs. If the event is an interrupt, the program counter immediately branches to the starting address of the interrupt or reset service routine. If the wake up event is a reset, before fetching the reset vector, there is a 512 CPU clock cycle delay to allow for stabilization.
Refer to Figure 10.

Figure 10. WAIT Mode Flow Chart



Note: 1) Before servicing an interrupt, the CC register is pushed on the stack. The I0 and I1 bit values for each interrupt are predefined by the user in the ISPRx register. During the interrupt routine these values are loaded into I0 and I1 bits and cleared when the CC register is popped.

ST7 POWER SAVING MODES (Cont'd)

5.3 HALT MODE

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

To further decrease the consumption (especially for the Suspend mode):

– The internal regulator must be put in powerdown mode by setting the REG_OFF bit of the CCMR register.

– The active slew rate compensation cell of the IOs must be stopped by setting the CPO bit of the EOSCR register.

Entering HALT mode clears the I bits in the CC register, enabling interrupts. If an interrupt is pending, the Device wakes up immediately. Not all interrupts will wake up the Device from HALT, only those listed in the Interrupt Mapping Table in the Interrupt section allow wake-up.

Specific interrupts such as an external interrupt or an USB end of suspend interrupt (as described in Table 16) or a reset wakes up the Device from HALT mode.

– If a reset is the wake-up event, the main oscillator is immediately turned on and a 512 CPU cycle delay is used to stabilize the oscillator. After the start up delay the device starts in Low power mode and the CPU resumes operation by fetching the reset vector.

– If an interrupt is the wake-up event, the main oscillator is immediately turned on and a 512 CPU cycle delay is used to stabilize the oscillator. After the start up delay, if the device was in low power mode before entering in halt mode the device starts in low power mode and the CPU resumes operation. But if the device was in full power mode before entering in halt, the operation are resumed only after the PLL lock

Refer to Figure 11 for more details.

5.3.1 HALT MODE RECOMMENDATIONS

– Make sure that an external event is available or that the USB end of suspend interrupt is enabled to wake up the Device from Halt mode.

– When using an external interrupt to wake up the Device, 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.

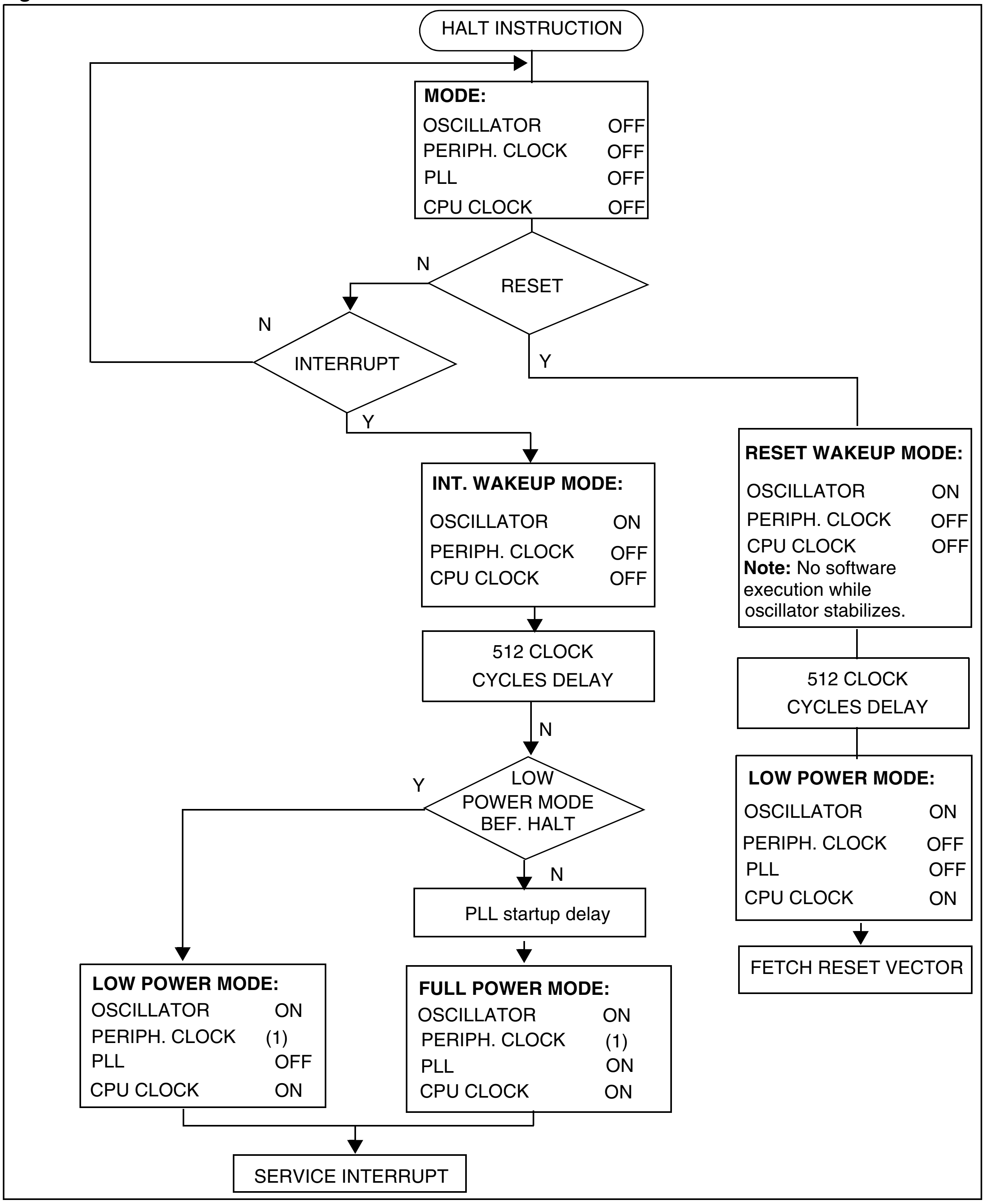
– For the same reason, reinitialize the level sensitiveness of each external interrupt as a precautionary measure.

– 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 I bits 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).

ST7 POWER SAVING MODES (Cont'd)

Figure 11. HALT Mode Flow Chart



1) periph. clock status is the one before the halt according to CER register.

6 ST7 I/O PORTS

6.1 INTRODUCTION

The I/O ports allow data transfer. An I/O port can contain up to 8 pins. Each pin can be programmed independently either as a digital input or digital output. In addition, specific pins may have several other functions. These functions can include external interrupt, alternate signal input/output for on-chip peripherals or analog input.

6.2 FUNCTIONAL DESCRIPTION

A Data Register (DR) and a Data Direction Register (DDR) are always associated with each port. The Option Register (OR), which allows input/output options, may or may not be implemented. The following description takes into account the OR register. Refer to the Port Configuration table for Device specific information.

An I/O pin is programmed using the corresponding bits in the DDR, DR and OR registers: bit x corresponding to pin x of the port.

Figure 12 shows the generic I/O block diagram.

6.2.1 Input Modes

Clearing the DDRx bit selects input mode. In this mode, reading its DR bit returns the digital value from that I/O pin.

If an OR bit is available, different input modes can be configured by software: floating or pull-up. Refer to I/O Port Implementation section for configuration.

Note: Writing to the DR modifies the latch value but does not change the state of the input pin.

External Interrupt Function

In input mode, external interrupts can be enabled by setting the corresponding bit in the PxEIENR register.

Falling or rising edge sensitivity is programmed independently for each interrupt vector. The External Interrupt Control Register (EICR) controls this sensitivity.

Several pins may be tied to one external interrupt vector. Refer to Pin Description to see which ports have external interrupts.

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

6.2.2 Output Modes

Setting the DDRx bit selects output mode. Writing to the DR bits applies a digital value to the I/O through the latch. Reading the DR bits returns the previously stored value.

If an OR bit is available, different output modes can be selected by software: push-pull or open-drain. Refer to I/O Port Implementation section for configuration.

Table 10. DR value and output pin status

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

Note: When switching from input to output mode, first set the DR bit to set the correct level to be applied on the pin, then write the DDR to configure the pin as an output.

6.2.3 Alternate Functions

Many I/Os of the Device have one or more alternate functions to output. This may include output signals from, or input signals to, on-chip peripherals. The Device Pin Description table describes which peripheral signals can be input/output to which ports.

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

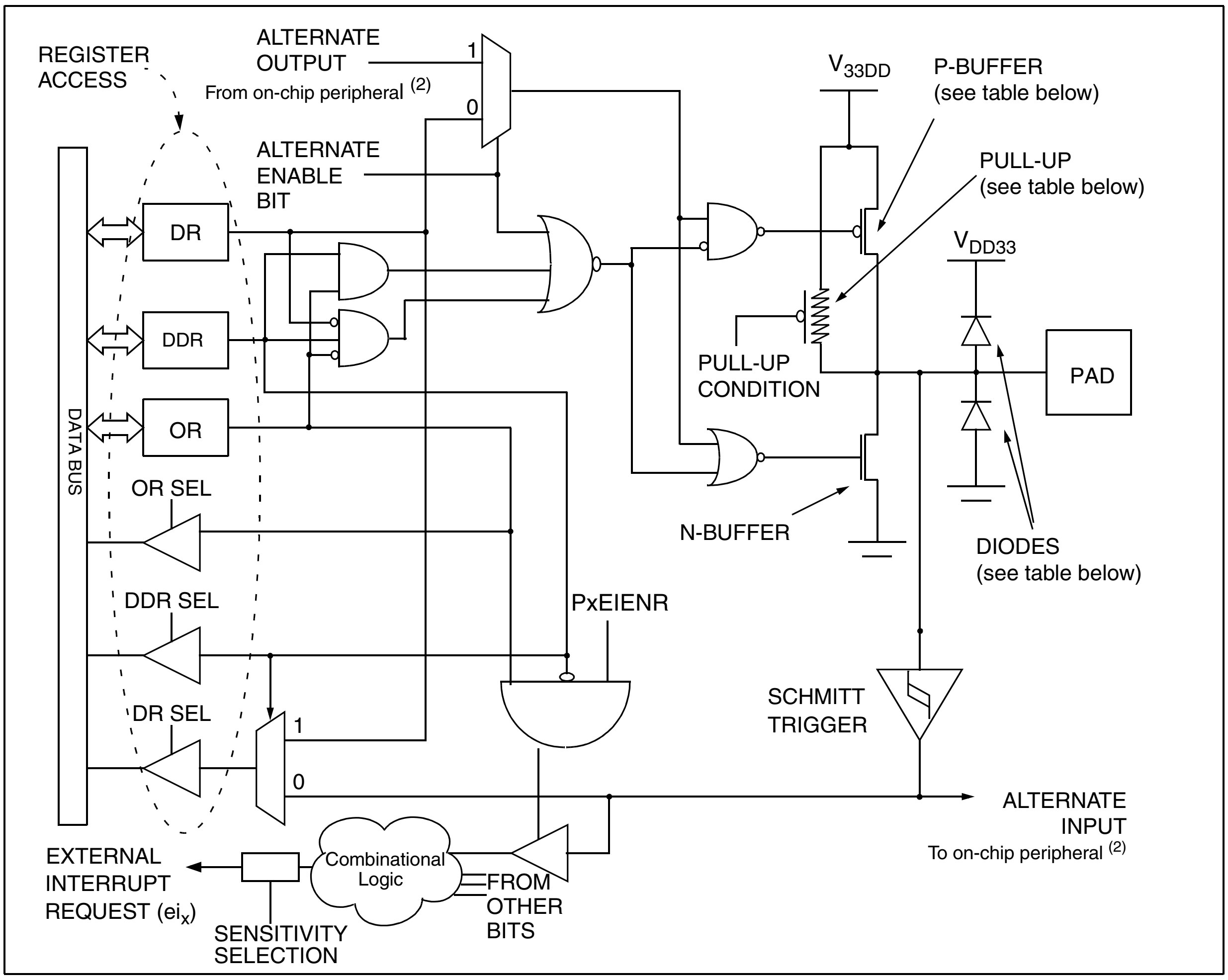
Configuring an I/O as floating enables alternate function input. It is not recommended to configure an I/O as pull-up as this will increase current consumption. Before using an I/O as an alternate input, configure it without interrupt. Otherwise spurious interrupts can occur.

Configure an I/O as input floating for an on-chip peripheral signal which can be input and output.

Caution:
I/Os which can be configured as both an analog and digital alternate function need special attention. The user must control the peripherals so that the signals do not arrive at the same time on the same pin. If an external clock is used, only the clock alternate function should be employed on that I/O pin and not the other alternate function.

ST7 I/O PORTS (Cont'd)

Figure 12. I/O Port General Block Diagram [(1)]



Notes:

1. Refer to the Port Configuration table for Device specific information.
2. MSCI can control Port A, B, C and D. See the MSCI I/0 Ports chapter.

Table 11. ST7 I/O Port Mode Options

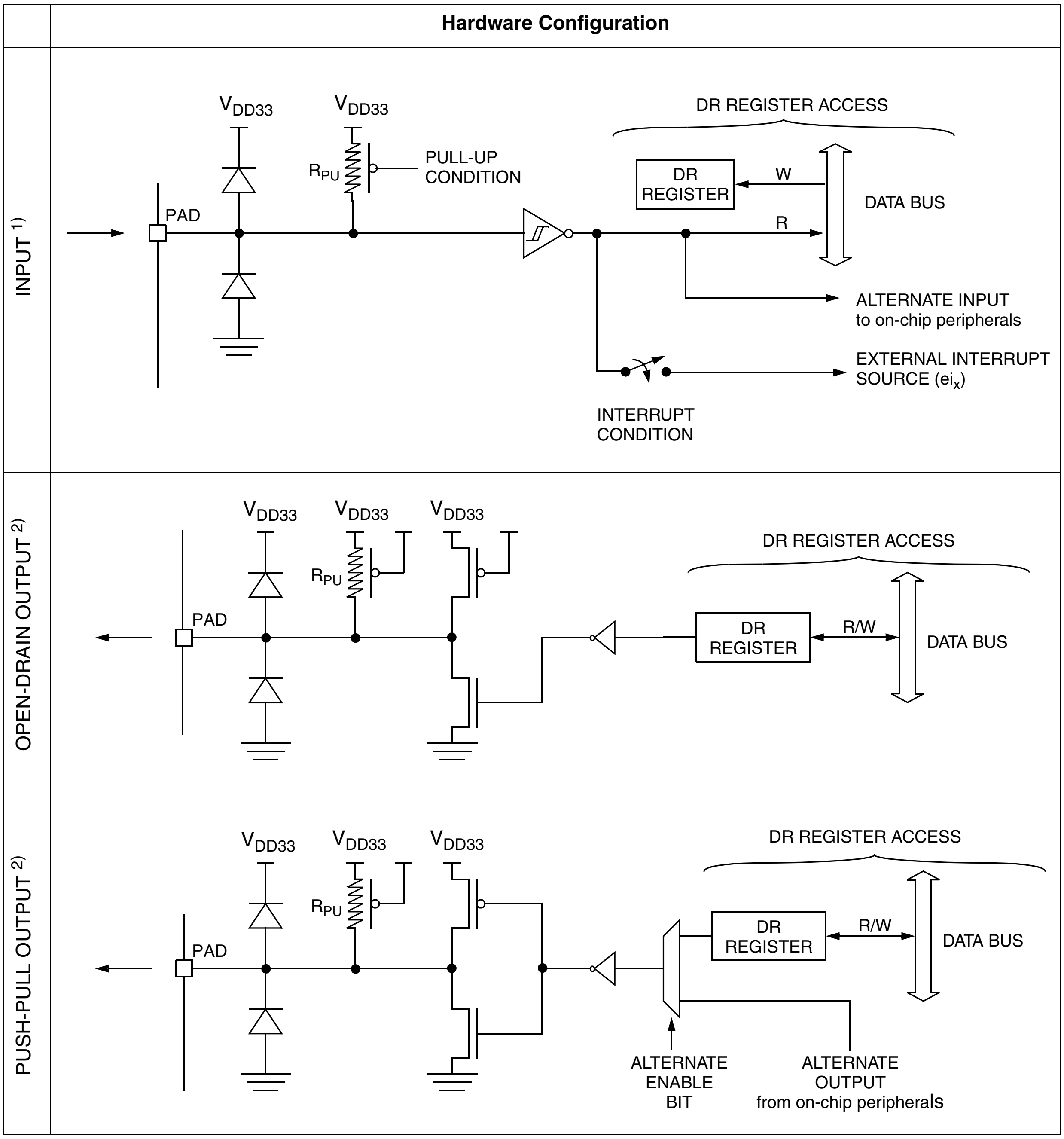
Configuration Mode		**Pull-Up**	**P-Buffer**	**Diodes**	
				to V_{33} OR V_{DD} V_{DD}	**to V_{SS}**
Input	Floating with/without Interrupt	Off	Off	On	On
	Pull-up with/without Interrupt	On			
Output	Push-pull	Off	On		
	Open Drain (logic level)		Off		
	True Open Drain	NI	NI	NI (see note)	

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

Note: The diode to $V_{33\ OR}$ V_{DD} V_{DD} is not implemented in the true open drain pads. A local protection between the pad and V_{OL} is implemented to protect the device against possible stress.

ST7 I/O PORTS (Cont'd)

Figure 13. Standard I/O Port Configurations



Notes:

1. When the I/O port is in input configuration and the associated alternate function is enabled as an output, reading the DR register will read the alternate function 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.

ST7 I/O PORTS (Cont'd)

6.3 ST7 I/O PORT IMPLEMENTATION

The hardware implementation on each I/O port depends on the settings in the DDR and OR registers and specific I/O port features such as ADC input or open drain.

Switching these I/O ports from one state to another should be done in a sequence that prevents unwanted side effects.

6.4 UNUSED I/O PINS

Unused I/O pins must be connected to fixed voltage levels. Refer to the Electrical Characteristics Section.

6.5 LOW POWER MODES

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

6.6 INTERRUPTS

The external interrupt event generates an interrupt if the corresponding configuration is selected with DDR and PxEIENR registers and if the I bit in the CC register is cleared (RIM instruction).

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

Table 12. ST7 I/O Port Configuration

Port 1)	Pin name	Input		Output	
		OR = 0	OR = 1	OR = 0	OR = 1
Port A	PA7:0	floating	pull-up	open drain	push-pull
Port B	PB7:0	floating	pull-up	open drain	push-pull
Port C	PC7:0	floating	pull-up	open drain	push-pull
Port D	PD7:0	floating	pull-up	open drain	push-pull
Port E	PE7:0	floating	pull-up	open drain	push-pull

Note:

1. Ports unavailable in the 48-pin packages (PB7:0, PD7:5, PE6:4) are forced to input mode with internal pull-up activated to avoid possible floating I/O consumption.

ST7 I/O PORTS (Cont'd)

6.7 Register Description

DATA REGISTER (DR)

Port x Data Register
PxDR with x = A, B, C, D or E.

Read/Write

Reset Value: 0000 0000 (00h)

7							0
D7	D6	D5	D4	D3	D2	D1	D0

Bits 7:0 = **D[7:0]** *Data register 8 bits.*

The DR register has a specific behaviour according to the selected input/output configuration. Writing the DR register is always taken into account even if the pin is configured as an input; this allows to always have the expected level on the pin when toggling to output mode. Reading the DR register always returns the digital value applied to the I/O pin (pin configured as input).

Note:

For this register, bits corresponding to I/O ports which are unavailable in the 48-pin package are read as 1.

DATA DIRECTION REGISTER (DDR)

Port x Data Direction Register
PxDDR with x = A, B, C, D or E.

Read/Write

Reset Value: 0000 0000 (00h)

7							0
DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0

Bits 7:0 = **DD[7:0]** *Data direction register 8 bits.*

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

0: Input mode
1: Output mode

OPTION REGISTER (OR)

Port x Option Register
PxOR with x = A,B, C, D, or E.

Read/Write

Reset Value: 0000 0000 (00h)

7							0
O7	O6	O5	O4	O3	O2	O1	O0

Bits 7:0 = **O[7:0]** *Option register 8 bits.*

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

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

Each bit is set and cleared by software.
Input mode:
0: Floating input
1: Floating input with interrupt.
Output mode:
0: Output open drain (with P-Buffer deactivated)
1: Output push-pull

ST7 I/O PORTS (Cont'd)

Table 13. ST7 I/O Port Register Map and Reset Values

Address (Hex.)	**Register Label** [1) 2)]	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reset Value of all I/O port registers		0	0	0	0	0	0	0	0
0026h	**PADR**	MSB							LSB
0027h	**PADDR**								
0028h	**PAOR**								
0029h	**PBDR**	MSB							LSB
002Ah	**PBDDR**								
002Bh	**PBOR**								
002Ch	**PCDR**	MSB							LSB
002Dh	**PCDDR**								
002Eh	**PCOR**								
002Fh	**PDDR**	MSB							LSB
0030h	**PDDDR**								
0031h	**PDOR**								
0032h	**PEDR**	MSB							LSB
0033h	**PEDDR**								
0034h	**PEOR**								

Notes:

1. PxDDR and PxOR bits corresponding to IOs which are unavailable in the 48-pin package are forced to 0. They are write protected.
2. PxDR bits corresponding to IOs which are unavailable in the 48-pin package are read as 1.

7 ST7 INTERRUPTS

7.1 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.

7.2 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 14). The processing flow is shown in Figure 14

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).

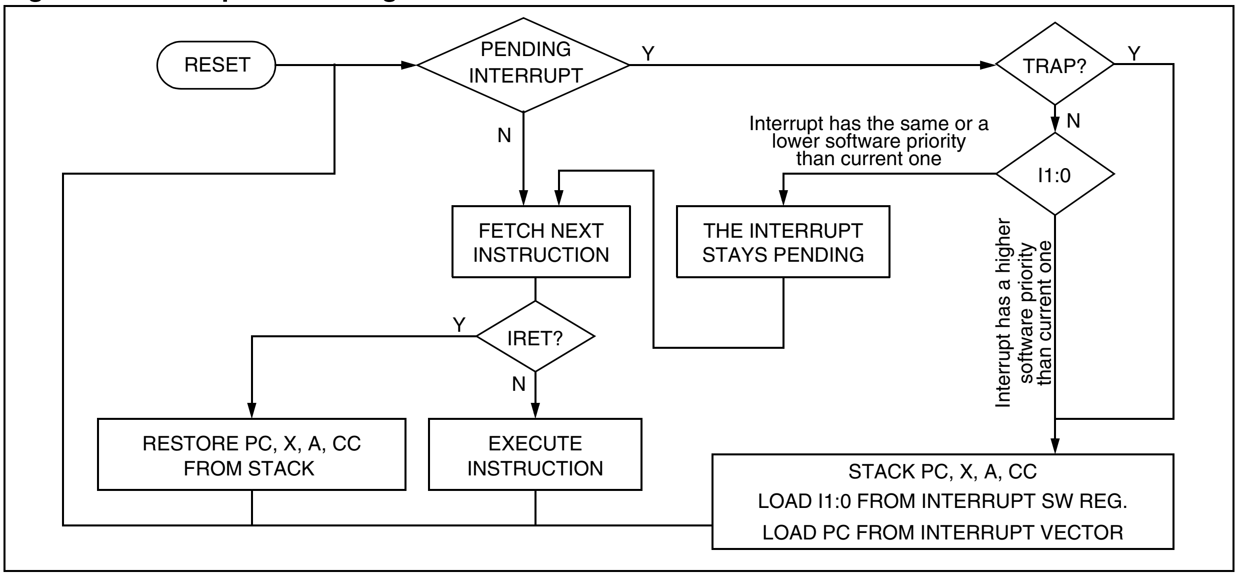
The interrupt service routine should end with the IRET instruction which causes the contents of the saved registers to be recovered from the stack.

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 14. 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 14. Interrupt Processing Flowchart



RESET
PENDING INTERRUPT
Y
N
TRAP?
Y
N
Interrupt has the same or a lower software priority than current one
I1:0
Interrupt has a higher software priority than current one
FETCH NEXT INSTRUCTION
THE INTERRUPT STAYS PENDING
IRET?
Y
N
RESTORE PC, X, A, CC FROM STACK
EXECUTE INSTRUCTION
STACK PC, X, A, CC
LOAD I1:0 FROM INTERRUPT SW REG.
LOAD PC FROM INTERRUPT VECTOR

ST7 INTERRUPTS (Cont'd)

Servicing Pending Interrupts

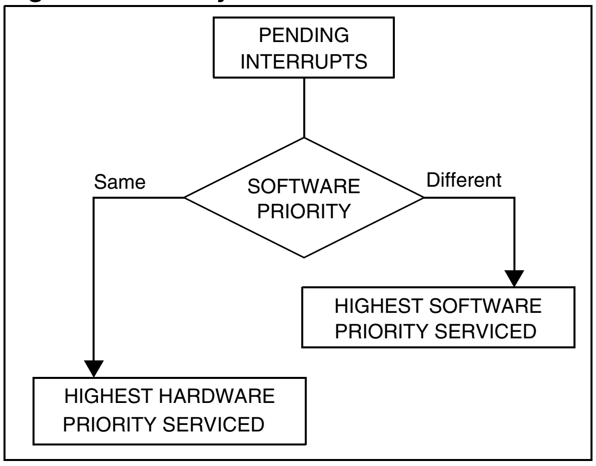
As several interrupts can be pending at the same time, the interrupt to be taken into account is determined by the following two-step process:

– 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 15 describes this decision process.

Figure 15. Priority Decision Process


PENDING INTERRUPTS
Same
SOFTWARE PRIORITY
Different
HIGHEST SOFTWARE PRIORITY SERVICED
HIGHEST HARDWARE PRIORITY SERVICED

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.

Note 1: The hardware priority is exclusive while the software one is not. This allows the previous process to succeed with only one interrupt.
Note 2: RESET and TRAP are non maskable and they 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 14). 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 14.

■ 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.

ST7 INTERRUPTS (Cont'd)

7.3 INTERRUPTS AND LOW POWER MODES

All interrupts allow the processor to exit from WAIT low power mode. On the contrary, only external and other specified interrupts allow the processor to exit from HALT mode (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 15.

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.

7.4 CONCURRENT & NESTED MANAGEMENT

The following Figure 16 and Figure 17 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 17. The interrupt hardware priority is given in this order from the lowest to the highest: MAIN, IT4, IT3, IT2, IT1, IT0. The software priority is given for each interrupt.

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

Figure 16. Concurrent Interrupt Management

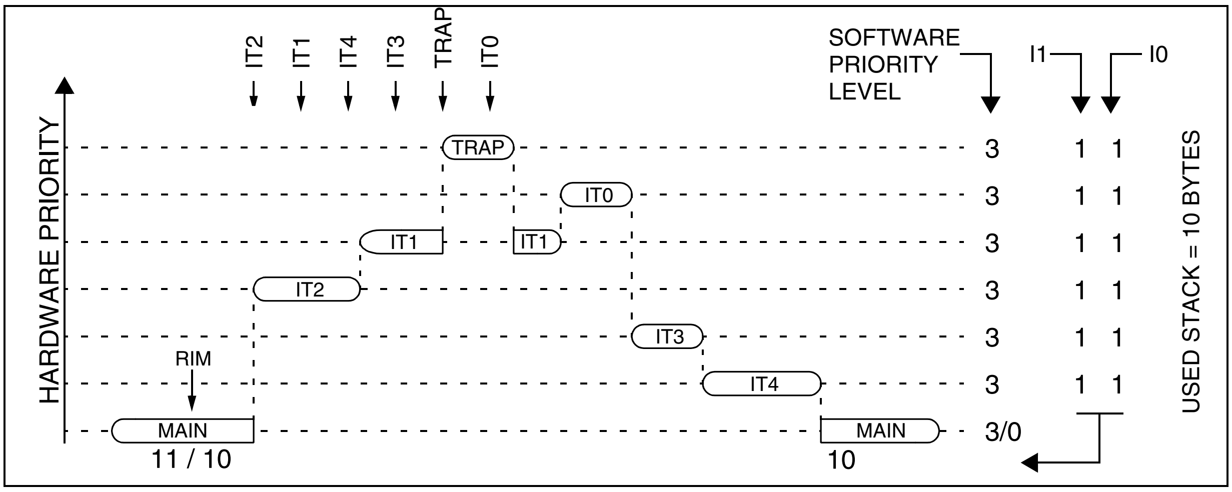
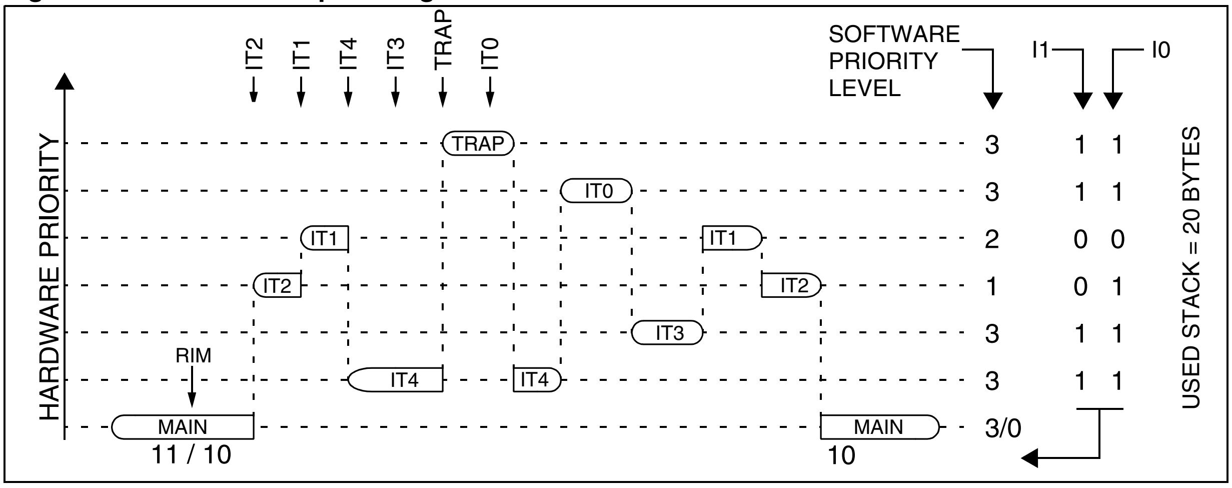


Figure 17. Nested Interrupt Management



ST7 INTERRUPTS (Cont'd)

Table 15. 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	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. In order not to lose the current software priority level, the RIM, SIM, HALT, WFI and POP CC instructions should never be used in an interrupt routine.

ST7 INTERRUPTS (Cont'd)

7.5 Interrupt Vector Table Management

For added flexibility, the ST7267 features two interrupt vector table modes. After reset, the interrupt vectors are located in ROM. The application can switch the vectors to RAM by executing the procedure given below.

Prior to switching the vectors to RAM, the RAM area must be initialised.

To switch the vectors from ROM to RAM:

1. Initialise the RAM area with the correct interrupt vectors
2. Set the USVR bit in the MISC1 register to enable the interrupt vector table in RAM

Table 16. Interrupt Mapping

N°	Source Block	Description	Register Label	Priority Order	Exit from HALT	Address Vector (2)
	RESET	Reset vector	N/A	Highest Priority	yes	FFFEh-FFFFh
	TRAP	Software Interrupt vector			no	FFFCh-FFFDh
0		Unused				FFFAh-FFFBh
1	USB	USB Interrupt	(1)		no	FFF8h-FFF9h
2	EOS	USB End of Suspend Interrupt	EOSSR		yes	FFF6h-FFF7h
3	MSCI	MSCI interrupt	MSCI		no	FFF4h-FFF5h
4	EI0	External Interrupt Port A	N/A		yes	FFF2h-FFF3h
5	EI1	External Interrupt Port B	N/A		yes	FFF0h-FFF1h
6	EI2	External Interrupt Port C	N/A		yes	FFEEh-FFEFh
7	EI3	External Interrupt Port D	N/A		yes	FFECh-FFEDh
8	EI4	External Interrupt Port E	N/A		yes	FFEAh-FFEBh
9	SPI	SPI interrupt	SPICSR		yes	FFE8h-FFE9h
10	TIMER	Timer interrupt	T1SR	Lowest Priority	no	FFE6h-FFE7h
11	TBU	TimeBase Unit	TBUCSR		no	FFE4h-FFE5h

Notes:

1. see USB chapter section 10.5 on page 93
2. This is the vector address in ROM. If the vector table is in RAM the address will be in the range 10FFh to 10E4h.

ST7 INTERRUPTS (Cont'd)

7.6 EXTERNAL INTERRUPTS

When an event occurs on an I/O port, this incoming signal is interpreted as an external interrupt. This signal can also be used to wake up the Device from HALT. There are several controlling factors for external interrupts:

– Priority (Hardware and Software)

– Enable/Disable control bits

– Sensitivity Control

– Status Flag

Up to 8 signals on 8 ports can share one external interrupt. For example, ei0 is shared on all 8 ports of Port A.

7.6.1 Software and Hardware Priorities

External interrupts have default priorities associated with them. They are as listed in the Interrupt Mapping table. These are the hardware priorities and are unchangeable.

Software priorities are user assigned by programming the appropriate bits in the Interrupt Software Priority register (ISPRx) for a given external interrupt. The whole external interrupt group will have the same priority. For example, ISPR1 bits[0:1] control the software priority for Port A's external interrupt, ei0.

These two types of priorities are important to manage because they function in the same manner as other interrupts for concurrent and nested modes.

7.6.2 Enable and Sensitivity Controls

At an external interrupt event, for the interrupt to be acknowledged, it must be enabled. There is a control bit for each external interrupt. They are found in the External Interrupt Enable Port x register (PxEINENR).

The external interrupt sensitivity is controlled by the ISxx bits in the EICRx registers (Figure 18). This control allows to have up to 4 fully independent external interrupt source sensitivities.

Each external interrupt source can be generated on four different events on the pin:

- Falling edge
- Rising edge
- Falling and rising edge
- Falling edge and low level

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).

7.6.3 Status Flag

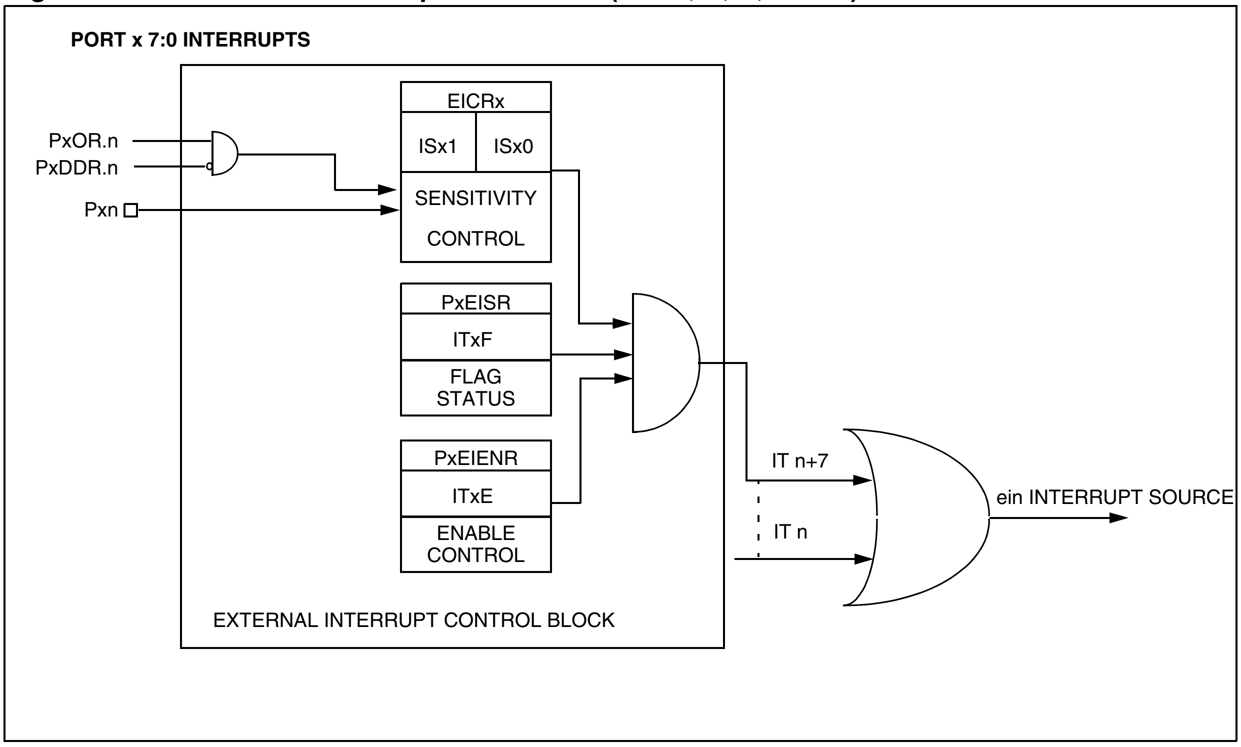
When an event occurs signalling that an external interrupt is requested, a flag is set by hardware. This flag informs the user which external interrupt has occurred. Each external interrupt has its own specific flag. They are found in the External Interrupt Port x register (PxEISR). If the corresponding external interrupt is enabled when this flag is set, the external interrupt is serviced.

If several interrupts are pending, the interrupts are serviced according to their priority (software and or hardware, according to which interrupt mode is being employed).

If there is an unwanted pending interrupt, it can be cleared by writing a different value in the ISx[1:0] in the EICRx registers.

ST7 INTERRUPTS (Cont'd)

Figure 18. Port x External Interrupt Control bits (x is A, B, C, D or E)



ST7 INTERRUPTS (Cont'd)

7.7 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 are non maskable sources and 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).

ST7 INTERRUPTS (Cont'd)

EXTERNAL INTERRUPT CONTROL REGISTER 0 (EICR0)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
ISD1	ISD0	ISC1	ISC0	ISB1	ISB0	ISA1	ISA0

Bit 7:6 =**ISD[1:0]** *Port D ei3 sensitivity IT[31-24]*
The interrupt sensitivity, defined using the ISD[1:0] bits, is applied to the ei3 external interrupts:

ISD1	ISD0	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).

Bits 5:4 = **ISC[1:0]** *Port C ei2 sensitivity IT[23-16]*
The interrupt sensitivity, defined using the ISC[1:0] bits, is applied to the ei2 external interrupts:

ISC1	ISC0	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).

Bits 3:2 = **ISB[1:0]** *Port B ei1 sensitivityIT[15-8]*
The interrupt sensitivity, defined using the ISB[1:0] bits, is applied to the ei1 external interrupts:

ISB1	ISB0	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).

Bits 1:0 = **ISA[1:0]** *Port A ei0 sensitivity IT[7-0]*
The interrupt sensitivity, defined using the ISA[1:0] bits, is applied to the ei0 external interrupts:

ISA1	ISA0	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).

EXTERNAL INTERRUPT CONTROL REGISTER 1 (EICR1)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	ISE1	ISE0

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

Bit 1:0 =**ISE[1:0]** *Port E ei4 sensitivity IT[39-32]*
The interrupt sensitivity, defined using the ISE[1:0] bits, is applied to the ei4 external interrupts:

ISE1	ISE0	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).

PORT A EXTERNAL INTERRUPT ENABLE REGISTER (PAEIENR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
IT7E	IT6E	IT5E	IT4E	IT3E	IT2E	IT1E	IT0E

ST7 INTERRUPTS (Cont'd)

Bits 7:0 = **ITxE** *Port A interrupt enable*

These bits are set and cleared by software.
0: ITx external interrupt disabled.
1: ITx external interrupt enabled.

PORT B EXTERNAL INTERRUPT ENABLE REGISTER (PBEIENR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
IT15E	IT14E	IT13E	IT12E	IT11E	IT10E	IT9E	IT8E

Bits 7:0 = **ITxE** *Port B interrupt enable*

These bits are set and cleared by software.
0: ITx external interrupt disabled.
1: ITx external interrupt enabled.

PORT C EXTERNAL INTERRUPT ENABLE REGISTER (PCEIENR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
IT23E	IT22E	IT21E	IT20E	IT19E	IT18E	IT17E	IT16E

These bits are set and cleared by software.

Bits 7:0 = **ITxE** *Port C interrupt enable*
0: ITx external interrupt disabled.
1: ITx external interrupt enabled.

PORT D EXTERNAL INTERRUPT ENABLE REGISTER (PDEIENR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
IT31E	IT30E	IT29E	IT28E	IT27E	IT26E	IT25E	IT24E

These bits are set and cleared by software.

Bits 7:0 = **ITxE** *Port D interrupt enable*
0: ITx external interrupt disabled.
1: ITx external interrupt enabled.

PORT E EXTERNAL INTERRUPT ENABLE REGISTER (PEEIENR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
IT39E	IT38E	IT37E	IT36E	IT35E	IT34E	IT33E	IT32E

These bits are set and cleared by software.

Bits 7:0 = **ITxE** *Port E interrupt enable*
0: ITx external interrupt disabled.
1: ITx external interrupt enabled.

PORT A EXTERNAL INTERRUPT STATUS REGISTER (PAEISR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
IT7F	IT6F	IT5F	IT4F	IT3F	IT2F	IT1F	IT0F

Bits 7:0 = **ITxF** *Port A interrupt flag*

These bits are set by hardware and cleared by software (by writing 0).
0: ITx external interrupt not requested.
1: ITx external interrupt requested.

PORT B EXTERNAL INTERRUPT STATUS REGISTER (PBEISR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
IT15F	IT14F	IT13F	IT12F	IT11F	IT10F	IT9F	IT8F

Bits 7:0 = **ITxF** *Port B interrupt flag*

These bits are set by hardware and cleared by software (by writing 0).
0: ITx external interrupt not requested.
1: ITx external interrupt requested.

ST7 INTERRUPTS (Cont'd)

PORT C EXTERNAL INTERRUPT STATUS REGISTER (PCEISR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
IT23F	IT22F	IT21F	IT20F	IT19F	IT18F	IT17F	IT16F

Bits 7:0 = **ITxF** *Port C interrupt flag*

These bits are set by hardware and cleared by software (by writing 0).
0: ITx external interrupt not requested.
1: ITx external interrupt requested.

PORT D EXTERNAL INTERRUPT STATUS REGISTER (PDEISR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
IT31F	IT30F	IT29F	IT28F	IT27F	IT26F	IT25F	IT24F

Bits 7:0 = **ITxF** *Port D interrupt flag*

These bits are set by hardware and cleared by software (by writing 0).
0: ITx external interrupt not requested.
1: ITx external interrupt requested.

PORT E EXTERNAL INTERRUPT STATUS REGISTER (PEEISR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
IT39F	IT38F	IT37F	IT36F	IT35F	IT34F	IT33F	IT32F

Bits 6:0 = **ITxF** *Port E interrupt flag*

These bits are set by hardware and cleared by software (by writing 0).
0: ITx external interrupt not requested.
1: ITx external interrupt requested.

DEBUG MODULE REGISTERS

DM CONTROL REGISTER (DMCR)

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

7							0
0	MTR	0	0	0	0	0	0

Bit 7 = Reserved, must be kept cleared.

Bit 6 = **MTR** *Monitor Control.*
This bit must be set to access all DM registers, if this bit is cleared all DM registers except MTR bit are write protected. This bit is set by software or by hardware at the beginning of ICC Monitor execution. It is cleared by hardware at the end of the ICC Monitor.

0: ICC Monitor program is not running
1: ICC Monitor program is running

Bit 5:0 = Reserved, must be kept cleared.

DM CONTROL REGISTER 2 (DMCR2)

Read/Write

Reset Value: 0000 0000 (00h)

Bit 7:1= Reserved, must be kept cleared

7							0
0	0	0	0	0	0	0	SVR

Bit 0 = **SVR** *Switch Interrupt Vectors to RAM*

This bit is set and cleared by software. It switches the interrupt vector table location to RAM
0: Interrupt vector table located in ROM
1: Interrupt vector table located in RAM

ST7 INTERRUPTS (Cont'd)

Table 17. Nested Interrupts Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0037h		MSCI		EOS		USB 2		TLI	
	ISPR0 Reset Value	I1_3 1	I0_3 1	I1_2 1	I0_2 1	I1_1 1	I0_1 1	1	1
0038h		EI3		EI2		EI1		EI0	
	ISPR1 Reset Value	I1_7 1	I0_7 1	I1_6 1	I0_6 1	I1_5 1	I0_5 1	I1_4 1	I0_4 1
0039h		TBU		TIMER		SPI		EI4	
	ISPR2 Reset Value	I1_11 1	I0_11 1	I1_10 1	I0_10 1	I1_9 1	I0_9 1	I1_8 1	I0_8 1
003Ah						Not used		Not used	
	ISPR3 Reset Value	1	1	1	1	I1_13 1	I0_13 1	I1_12 1	I0_12 1
003Bh	**EICR0** Reset Value	ISD1 0	ISD0 0	ISC1 0	ISC0 0	ISB1 0	ISB0 0	ISA1 0	ISA0 0
003Ch	**EICR1** Reset Value	0	0	0	0	0	0	ISE1 0	ISE0 0
003Dh	**PAEIENR** Reset Value	IT7E 0	IT6E 0	IT5E 0	IT4E 0	IT3E 0	IT2E 0	IT1E 0	IT0E 0
003Eh	**PBEIENR** Reset Value	IT15E 0	IT14E 0	IT13E0 0	IT12E 0	IT11E 0	IT10E 0	IT9E 0	IT8E 0
003Fh	**PCEIENR** Reset Value	IT23E 0	IT22E 0	IT21E 0	IT20E 0	IT19E 0	IT18E 0	IT17E 0	IT16E 0
0040h	**PDEIENR** Reset Value	IT31E 0	IT30E 0	IT29E 0	IT28E 0	IT27E 0	IT26E 0	IT25E 0	IT24E 0
0041h	**PEEIENR** Reset Value	IT39E 0	IT38E 0	IT37E 0	IT36E 0	IT35E 0	IT34E 0	IT33E 0	IT32E 0
0042h	**PAEISR** Reset Value	IT7F 0	IT6F 0	IT5F 0	IT4F 0	IT3F 0	IT2F 0	IT1F 0	IT0F 0
0043h	**PBEISR** Reset Value	IT15F 0	IT14F 0	IT13F 0	IT12F 0	IT11F 0	IT10F 0	IT9F 0	IT8F 0
0044h	**PCEISR** Reset Value	IT23F 0	IT22F 0	IT21F 0	IT20F 0	IT19F 0	IT18F 0	IT17F 0	IT16F 0
0045h	**PDEISR** Reset Value	IT31F 0	IT30F 0	IT29F 0	IT28F 0	IT27F 0	IT26F 0	IT25F 0	IT24F 0
0046h	**PEEISR** Reset Value	IT39F 0	IT38F 0	IT37F 0	IT36F 0	IT35F 0	IT34F 0	IT33F 0	IT32F 0
0074h	**DMCR** Reset Value	0	MTR 0	0	0	0	0	0	0
007Ah	**DMCR2** Reset Value	0	0	0	0	0	0	0	SVR 0

8 ST7 CLOCK, RESET AND SUPPLY MANAGEMENT

8.1 CLOCK SYSTEM

The main clock of the Device is generated by a 12 MHz crystal oscillator (main oscillator)

The associated hardware configurations are shown in Table 18. Refer to the electrical characteristics section for more details.

Crystal Oscillator

The internal oscillator is designed to operate with a 12MHz AT-cut parallel resonant quartz.

The crystal and associated components should be installed as close as possible to the input pins in order to minimize output distortion and start-up stabilization time.

Table 18. Device Clock Source

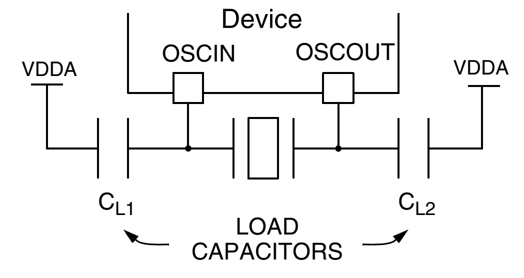
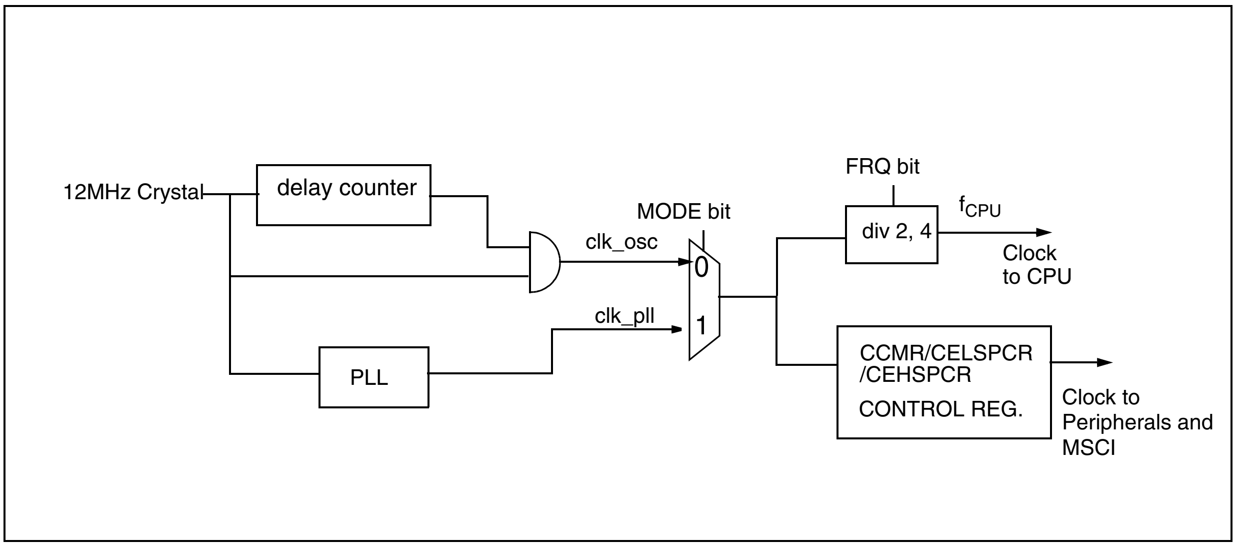
	Hardware Configuration
Crystal Oscillator	 Device OSCIN OSCOUT VDDA VDDA C_{L1} C_{L2} LOAD CAPACITORS

Figure 19. Clock Control Block Diagram



8.2 CLOCK MANAGEMENT

There are two types of run mode:

- Low power mode: the oscillator is the clock source (PLL is off). In this mode the USB clock domain is switched off (no 60 MHz clock is available).
- Full power mode for full operation with USB. The clock source is the PLL output (60 MHz).

After reset the device starts running in low power mode. To switch to full power mode set MODE bit of CCMR.

Control bits are also provided to enable or disable the clock to individual on-chip peripherals.

In additional the application software can put the Device in Wait, HALT.

8.2.1 Register Description

CLOCK CONTROL MODE REGISTER (CCMR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
RE-GOFF (1)	0	0	0	0	LOCK	MODE	FRQ

Bit 7 = **REGOFF** *Regulator mode in halt*
0: Regulator ON
1: Regulator OFF

Put the 3.3V to 1.8V regulator in power-down mode when the Halt instruction is executed. In this mode the 1.8V is provided but with low current sink capability (to maintain the RAM content and enable the wake-up). This bit has to be set before entering in halt mode.

Note 1: This bit is automatically cleared after the wake-up from halt mode.

Bit 6:3 = Reserved, must be kept cleared.

Bit 2 = **LOCK** *PLL lock (Read Only)*
This bit gives the PLL lock status.
0: PLL is not locked
1: PLL is locked

Bit 1 = **MODE** *Run mode*
This bit defines the device run mode.
0: Low power mode
1: Full power mode

Bit 0 = **FRQ** *CPU clock frequency*
This bit defines the CPU clock frequency.
0: Clock source frequency divided by 2.
1: Clock source frequency divided by 4.

Table 19. Clock frequency selection

State (1,2)	LOCK	MODE	FRQ	CPU clk	MSCI core clk	MSCI periph clk	USB clk
0	0	0	0	6 MHz	6 MHz	12 MHz	No clk
1	0	0	1	3 MHz	6 MHz	12 MHz	No clk
2	0	1	0	6 MHz	6 MHz	12 MHz	No clk
3	0	1	1	3 MHz	6 MHz	12 MHz	No clk
4	1	1	0	30 MHz	30 MHz	60 MHz	60 MHz
5	1	1	1	15 MHz	30 MHz	60 MHz	60 MHz

Notes:
1. state 2 and 3 are intermediate states waiting for PLL lock
2. state 4 cannot be used with the emulator.

CLOCK MANAGEMENT (Cont'd)

CLOCK ENABLE OF LOW SPEED PERIPHERALS CLK REGISTER (CELSPCR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
0	0	0	TBU	SPI	WDG	TIM	DM

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

Bit 6 = **TBU** *Timebase Unit*

This bit enables the clock of the Timebase Unit. It is set and cleared by software.
0: TBU clock disabled
1: TBU clock enabled

Bit 3 = **SPI** *SPI clock enable*
This bit enables the clock of the SPI. It is set and cleared by software.
0: SPI clock disabled
1: SPI clock enabled

Bit 2 = **WDG** *clock enable*
This bit enables the clock of the WDG. It is set and cleared by software.
0: WDG clock disabled
1: WDG clock enabled

Note: when WDGHWR option is activated the watchdog clock is always enabled.

Bit 1 = **TIM** *Timer clock enable*
This bit enables the clock of the timer. It is set and cleared by software.
0: Timer clock disabled
1: Timer clock enabled

Bit 0 = **DM** *DM clock enable*
This bit enables the clock of the Debug Module. It is set by software and cannot be reset.
0: DM clock disabled
1: DM clock enabled

CLOCK MANAGEMENT (Cont'd)

CLOCK ENABLE OF HIGH SPEED PERIPHERALS CLK REGISTER (CEHSPCR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	USB	DEC	ENC	MSCI

Bit 7:4 = Reserved, must be kept cleared.

Bit 3 = **USB** *USB clock enable*
This bit enables the clock of the USB. It is set and cleared by software.
0: USB clock disabled
1: USB clock enabled

Bit 2 = **DEC** *Decoder clock enable*
This bit enables the clock of the Reed-Solomon decoder. It is set and cleared by software.
0: Decoder clock disabled
1: Decoder clock enabled

Bit 1= **ENC** *Encoder clock enable*
This bit enables the clock of the Reed-Solomon encoder. It is set and cleared by software.
0: Encoder clock disabled
1: Encoder clock enabled

Bit 0 = **MSCI** *MSCI clock enable*
This bit enables the clock of the MSCI. It is set and cleared by software.
0: MSCI clock disabled
1: MSCI clock enabled

Table 20. Clock, Reset and Supply Control/Status Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
005Bh	**CCMR** Reset Value	REGOFF 0	0	0	0	0	LOCK 0	MODE 0	FRQ 0
005Ch	**CELSPCR** Reset Value	0	0	0	TBU 0	SPI 0	WDG 0	TIM 0	ICD 0
005Dh	**CEHSPCR** Reset Value	0	0	0	0	USB 0	DEC 0	ENC 0	MSCI 0

8.3 RESET SEQUENCE MANAGER (RSM)

8.3.1 Introduction

The reset sequence manager includes two RESET sources as shown in Figure 21:

- External $\overline{\text{RESET}}$ source pulse
- Internal WATCHDOG RESET
- Illegal Opcode reset

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

The basic RESET sequence consists of 3 phases as shown in Figure 20:

- Active Phase depending on the RESET source
- 512 CPU clock cycle delay
- RESET vector fetch

The 512 CPU clock cycle delay allows the oscillator to stabilise and ensures that recovery has taken place from the Reset state.

The RESET vector fetch phase duration is 2 clock cycles.

Figure 20. RESET Sequence Phases

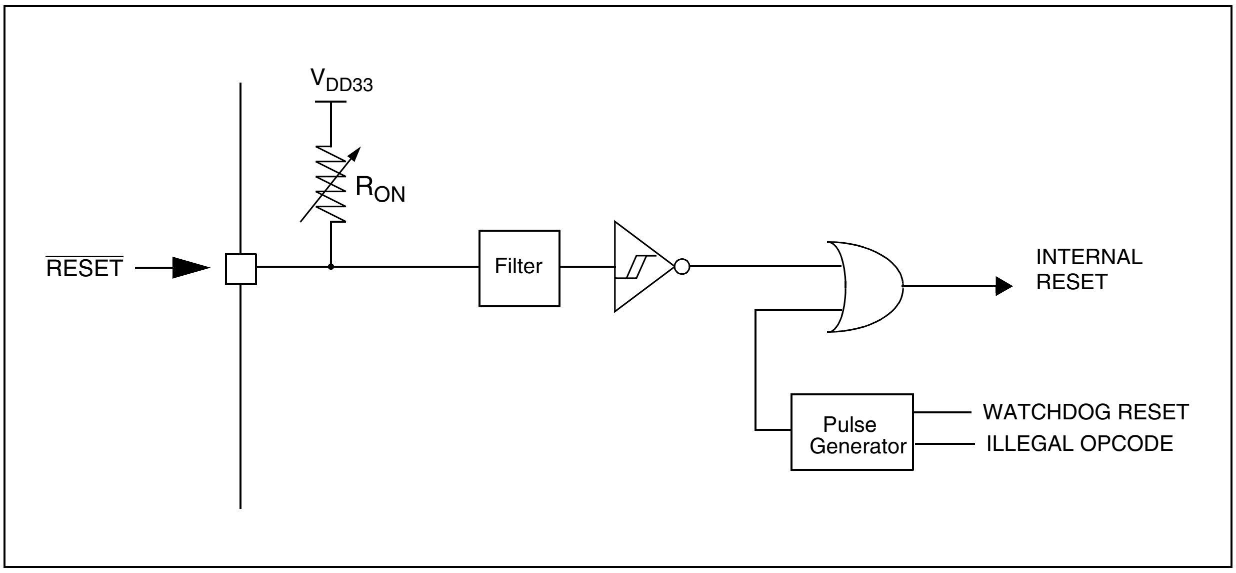
RESET		
Active Phase	INTERNAL RESET 512 CLOCK CYCLES	FETCH VECTOR

8.3.2 Asynchronous External $\overline{\text{RESET}}$ pin

The $\overline{\text{RESET}}$ pin is an input 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 Characteristic section for more details.

A RESET signal originating from an external source must have a duration of at least $t_{ew(RSTL)in}$ in order to be allow a correct internal start-up phase (see Figure 22). This detection is asynchronous and therefore the Device can enter reset state even in HALT mode.

Figure 21. Reset Block Diagram



RESET SEQUENCE MANAGER (Cont'd)

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

8.3.3 Internal Watchdog RESET

The RESET sequence generated by a internal Watchdog counter overflow is shown in Figure 22.

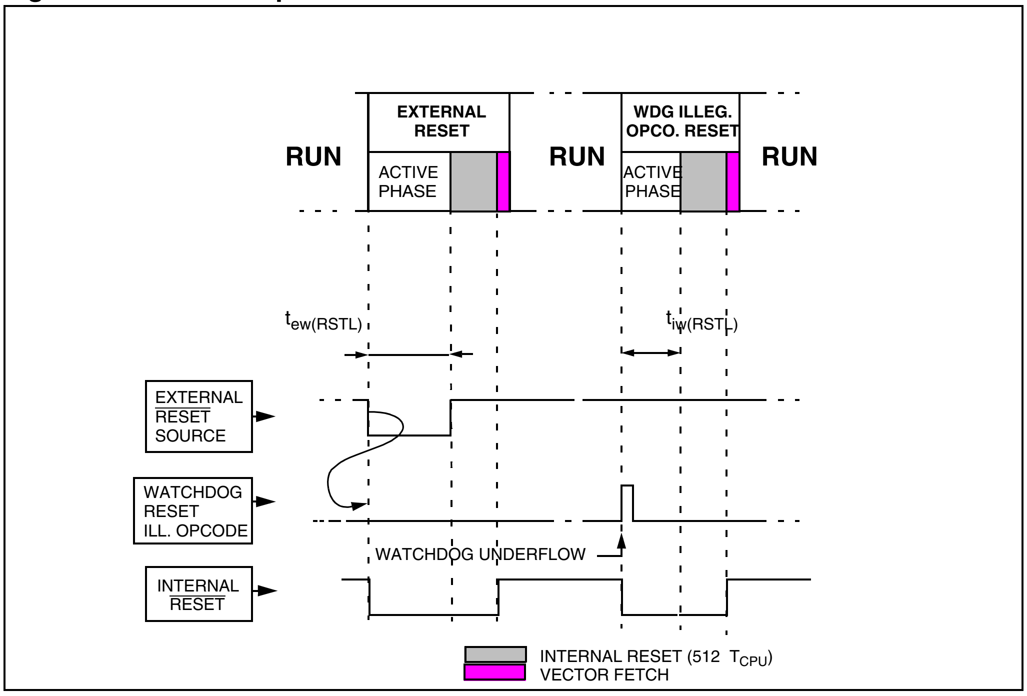
Starting from the Watchdog counter underflow, the Device is reset internally for at least $t_{iw(RSTL)}$.

8.3.4 Illegal Opcode reset

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

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

Figure 22. RESET Sequences


RUN
EXTERNAL RESET
ACTIVE PHASE
RUN
WDG ILLEG. OPCO. RESET
ACTIVE PHASE
RUN
$t_{ew(RSTL)}$
$t_{iw(RSTL)}$
EXTERNAL $\overline{\text{RESET}}$ SOURCE
WATCHDOG RESET ILL. OPCODE
WATCHDOG UNDERFLOW
INTERNAL $\overline{\text{RESET}}$
INTERNAL RESET (512 T_{CPU})
VECTOR FETCH

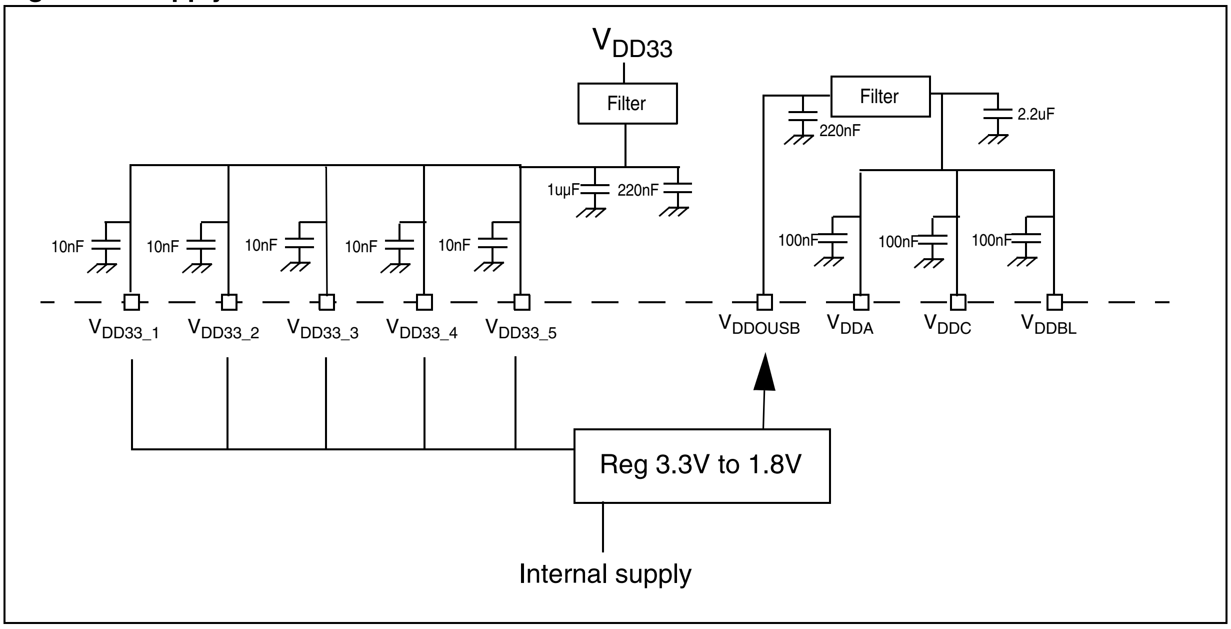
8.4 SUPPLY MANAGEMENT

The device operates with a single 3.3V supply power source.

The 3.3V supply is converted to 1.8V by an internal voltage regulator.

The VDDOUSB pad has to be connected to the VDDBL, VDDC and VDDA inputs of the chip through a filter.

Figure 23. Supply Interconnections



9 ST7 MISCELLANEOUS REGISTER

MISCELLANEOUS REGISTER 1 (MISCR1)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	USVR	0	0	ST7VG

Bits 7:4 = Reserved

Bit 3= **USVR** *User Switch Interrupt Vector in RAM*
This bit is set and cleared by software. It switches the interrupt vector table location to RAM
0: Interrupt vector table located in ROM
1: Interrupt vector table located in RAM

Bits 2:1 = Reserved

Bit 0= **ST7VG** *ST7 VAC Guard*
This bit is set and cleared by software. It gives the priority to the ST7 for the access to USB registers or buffers (when an operation is on-going by the MSCI it will be finished)
0: USB access priority given to MSCI
1: USB access priority given to ST7

Table 21. Miscellaneous Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0056h	**MISCR1** Reset Value	0	0	0	0	USVR 0	0	0	ST7VG 0

10 ST7 ON-CHIP PERIPHERALS

10.1 WATCHDOG TIMER (WDG)

10.1.1 Introduction

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

10.1.2 Main Features

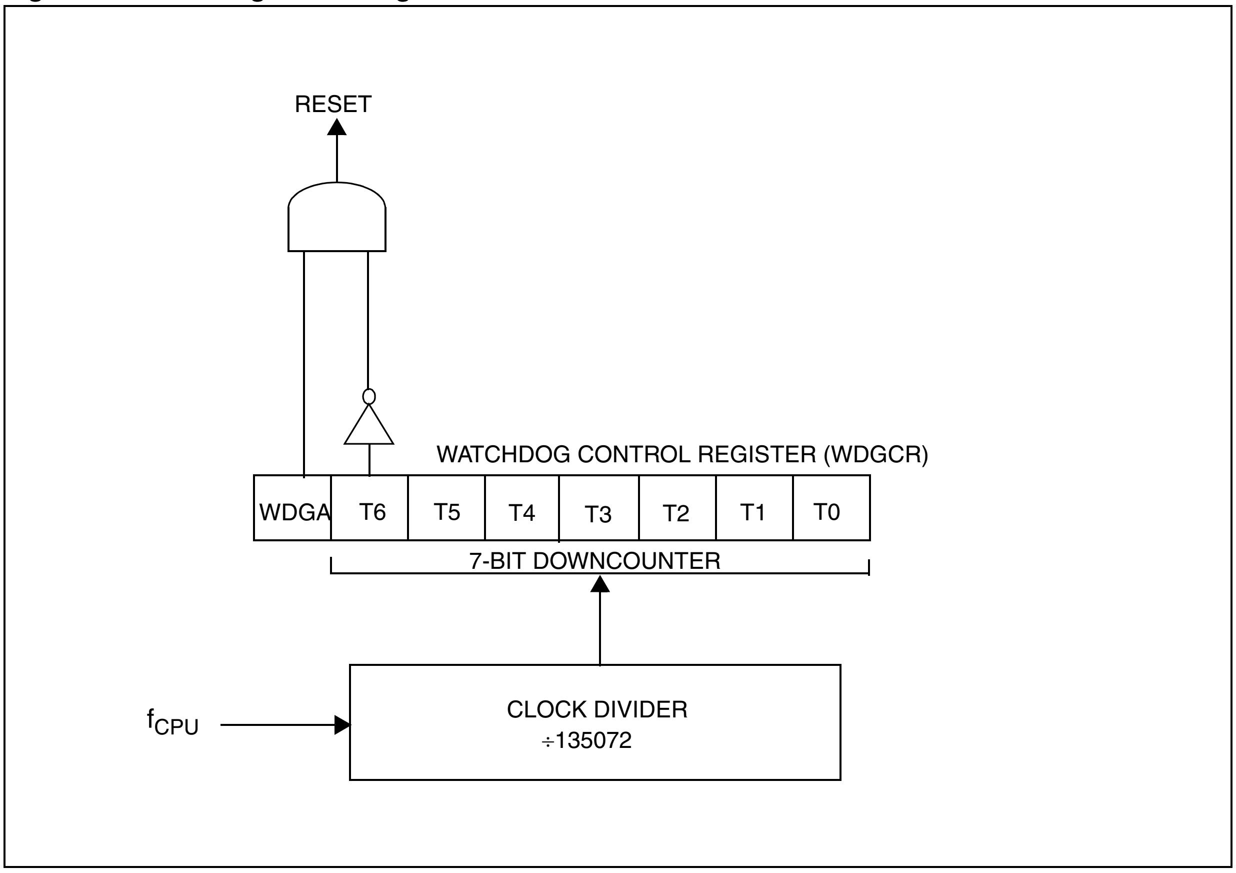
- Programmable free-running downcounter (64 increments of 131072 CPU cycles)
- Programmable reset
- Watchdog event (if the WDGA is set) when the T6 bit reaches zero
- Hardware Watchdog event selectable by option byte

10.1.3 Functional Description

The counter value stored in the WDGCR register (bits T[6:0]), is decremented every 131072 machine cycles, and the length of the timeout period can be programmed by the user in 64 increments.

If the watchdog is activated (the WDGA bit is set) and when the 7-bit timer (bits T[6:0]) rolls over from 40h to 3Fh (T6 becomes cleared), it generates a Watchdog reset.

Figure 24. Watchdog Block Diagram



WATCHDOG TIMER (Cont'd)

The application program must write in the WDGCR register at regular intervals during normal operation to prevent a Watchdog reset. This down-counter is free-running: it counts down even if the watchdog is disabled.

The value to be stored in the WDGCR register must be between FFh and C0h (see Table 22 .Watchdog Timing (fCPU = 15 MHz)):

– The WDGA bit is set (watchdog reset enabled)

– The T6 bit is set to prevent generating an immediate Watchdog reset

– The T[5:0] bits contain the number of increments which represents the time delay before the watchdog produces a reset.

Table 22.Watchdog Timing (f_{CPU} = 15 MHz)

	CR Register initial value	WDG timeout period (ms)
Max	FFh	559.2
Min	C0h	8.73

10.1.4 Generating a Software reset

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

10.1.5 Software Watchdog Option

If Software Watchdog is selected by option byte, the watchdog is disabled following a reset. Once activated it cannot be disabled, except by a reset.

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

10.1.6 Hardware Watchdog Option

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

10.1.7 Low Power Modes

Mode	Description
WAIT	No effect on Watchdog: the downcounter continues to decrement.
HALT	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 external interrupt is received, the Watchdog restarts counting after 512 CPU clocks. If a reset is generated, the Watchdog is disabled (reset state) unless Hardware Watchdog is selected by option byte.

WATCHDOG TIMER (Cont'd)

10.1.8 Using Halt Mode with the WDG

The following recommendations apply 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.

10.1.9 Interrupts

None

10.1.10 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** *Watchdog Reset Activation bit*.
This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog generates a reset when T6 reaches 0.
0: Watchdog Reset disabled
1: Watchdog Reset enabled

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

Bit 6:0 = **T[6:0]** *7-bit timer (MSB to LSB).*
These bits contain the decremented value. A watchdog event is produced when it rolls over from 40h to 3Fh (T6 becomes cleared).

Table 23. Watchdog Timer Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0035h	**WDGCR** Reset Value	WDGA 0	T6 1	T5 1	T4 1	T3 1	T2 1	T1 1	T0 1

10.2 TIMEBASE UNIT (TBU)

10.2.1 Introduction

The Timebase unit (TBU) can be used to generate periodic interrupts.

10.2.2 Main Features

- 8-bit upcounter
- Programmable prescaler
- Period between interrupts: max. 8.1ms (at 8 MHz f_{CPU})
- Maskable interrupt

10.2.3 Functional Description

The TBU operates as a free-running upcounter.

When the TCEN bit in the TBUCSR register is set by software, counting starts at the current value of the TBUCV register. The TBUCV register is incremented at the clock rate output from the prescaler selected by programming the PR[2:0] bits in the TBUCSR register.

When the counter rolls over from FFh to 00h, the OVF bit is set and an interrupt request is generated if ITE is set.

The user can write a value at any time in the TBUCV register.

10.2.4 Programming Example

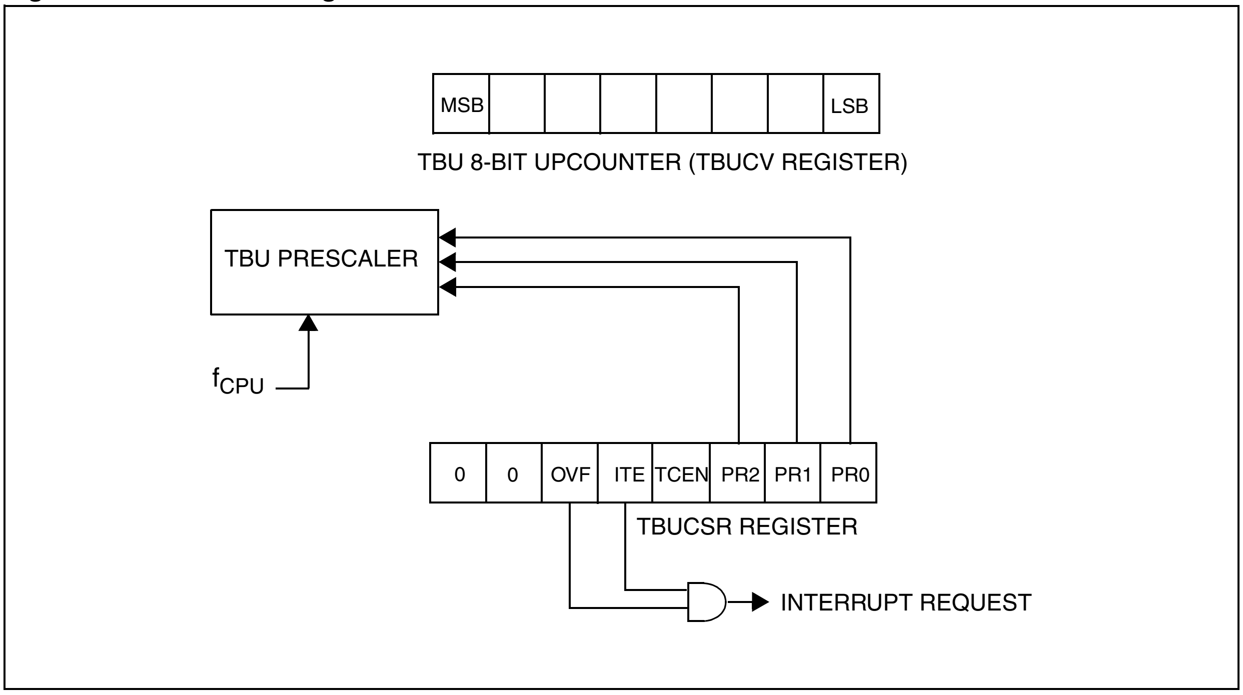
In this example, timer is required to generate an interrupt after a delay of 1 ms.

Assuming that f_{CPU} is 8 MHz and a prescaler division factor of 256 will be programmed using the PR[2:0] bits in the TBUCSR register, 1 ms = 32 TBU timer ticks.

In this case, the initial value to be loaded in the TBUCV must be (256-32) = 224 (E0h).

```
ld A, E0h
ld TBUCV, A   ; Initialize counter value
ld A 1Fh      ;
ld TBUCSR, A  ; Prescaler factor = 256,
              ; interrupt enable,
              ; TBU enable
```

Figure 25. TBU Block Diagram



TIMEBASE UNIT (Cont'd)

10.2.5 Low Power Modes

Mode	Description
WAIT	No effect on TBU
HALT	TBU halted.

10.2.6 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
Counter Over-flow Event	OVF	ITE	Yes	No

Note: The OVF interrupt event is connected to an interrupt vector (see Interrupts chapter).
It generates an interrupt if the ITE bit is set in the TBUCSR register and the I-bit in the CC register is reset (RIM instruction).

10.2.7 Register Description

TBU COUNTER VALUE REGISTER (TBUCV)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
CV7	CV6	CV5	CV4	CV3	CV2	CV1	CV0

Bit 7:0 = **CV[7:0]** *Counter Value*

This register contains the 8-bit counter value which can be read and written anytime by software. It is continuously incremented by hardware if TCEN=1.

TBU CONTROL/STATUS REGISTER (TBUCSR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
0	0	OVF	ITE	TCEN	PR2	PR1	PR0

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

Bit 5 = **OVF** *Overflow Flag*

This bit is set only by hardware, when the counter value rolls over from FFh to 00h. It is cleared by software reading the TBUCSR register. Writing to this bit does not change the bit value.
0: No overflow
1: Counter overflow

Bit 4 = **ITE** *Interrupt enabled.*

This bit is set and cleared by software.
0: Overflow interrupt disabled
1: Overflow interrupt enabled. An interrupt request is generated when OVF=1.

Bit 3 = **TCEN** *TBU Enable.*

This bit is set and cleared by software.
0: TBU counter is frozen and the prescaler is reset.
1: TBU counter and prescaler running.

Bit 2:0 = **PR[2:0]** *Prescaler Selection*

These bits are set and cleared by software to select the prescaling factor.

PR2	PR1	PR0	Prescaler Division Factor
0	0	0	2
0	0	1	4
0	1	0	8
0	1	1	16
1	0	0	32
1	0	1	64
1	1	0	128
1	1	1	256

TIMEBASE UNIT (Cont'd)

Table 24. TBU Register Map and Reset Values

Address (Hex.)	**Register Label**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0058h	**TBUCVR** Reset Value	CV7 0	CV6 0	CV5 0	CV4 0	CV3 0	CV2 0	CV1 0	CV0 0
0059h	**TBUCSR** Reset Value	0 0	0 0	OVF 0	ITE 0	TCEN 0	PR2 0	PR1 0	PR0 0

10.3 16-BIT TIMER

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

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

10.3.2 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
- 1 or 2 Output Compare functions each with:
  - 2 dedicated 16-bit registers
  - 2 dedicated programmable signals
  - 2 dedicated status flags
  - 1 dedicated maskable interrupt
- 1 or 2 Input Capture functions each 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 1.

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

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

10.3.3 Functional Description

10.3.3.1 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).

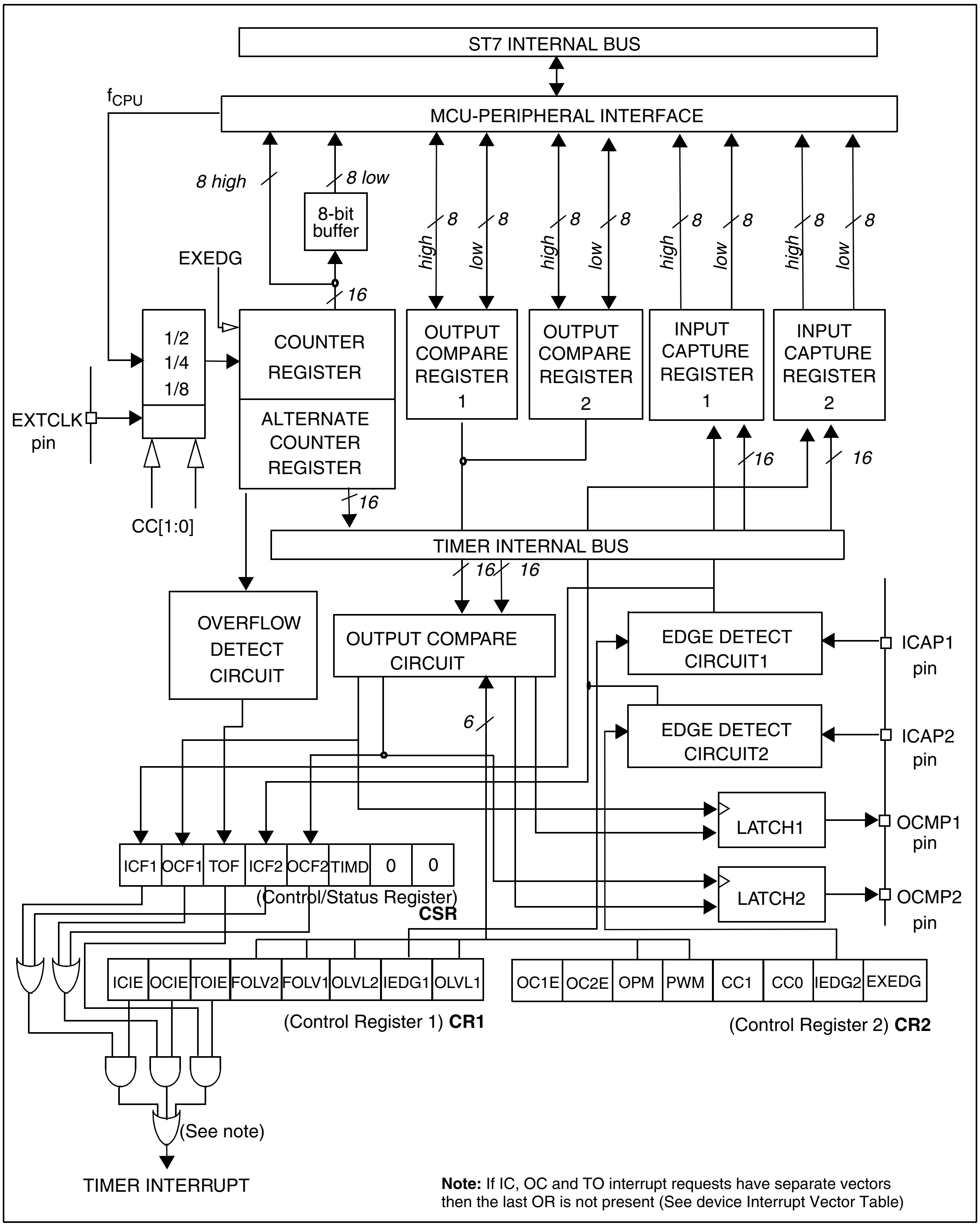
These two read-only 16-bit registers contain the same value but with the difference that reading the ACLR register does not clear the TOF bit (Timer overflow flag), located in the Status register, (SR), (see note at the end of paragraph titled 16-bit read sequence).

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 1. The value in the counter register repeats every 131072, 262144 or 524288 CPU clock cycles depending on the CC[1:0] bits.
The timer frequency can be $f_{CPU}/2$, $f_{CPU}/4$, $f_{CPU}/8$ or an external frequency.

16-BIT TIMER (Cont'd)

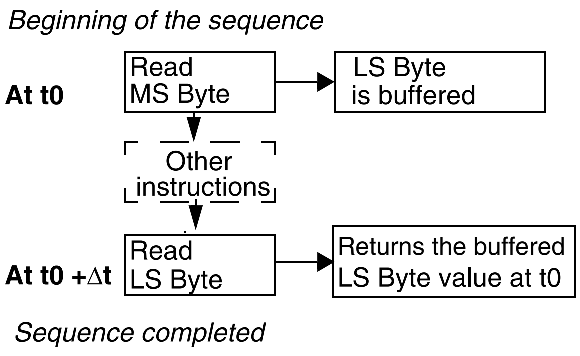
Figure 26. Timer Block Diagram



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).



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 (MCU awakened by an interrupt) or from the reset count (MCU awakened by a Reset).

10.3.3.2 External Clock

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

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

The counter is synchronized 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 27. Counter Timing Diagram, internal clock divided by 2

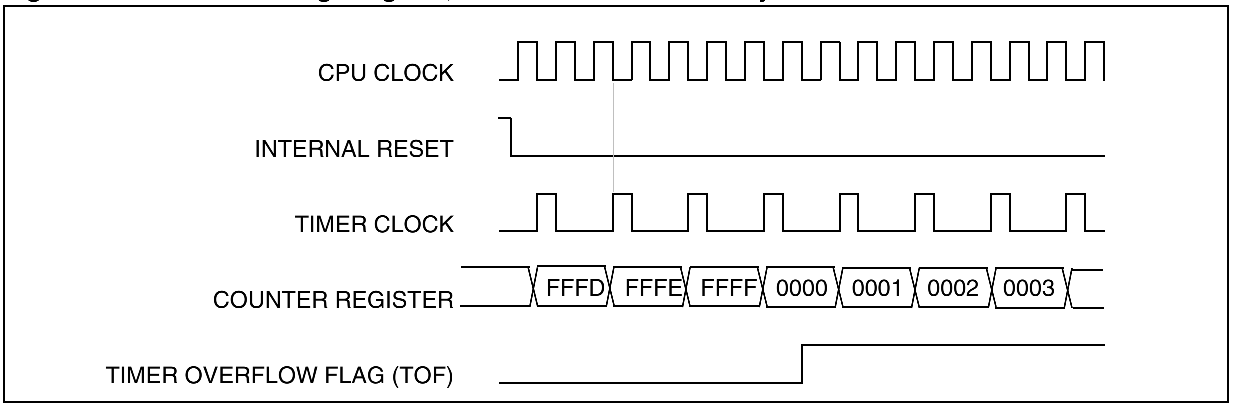


Figure 28. Counter Timing Diagram, internal clock divided by 4

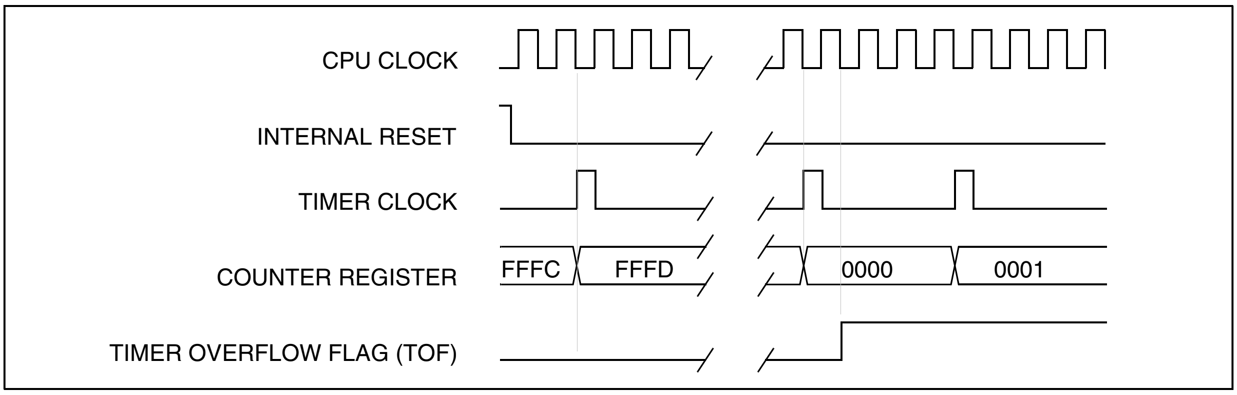
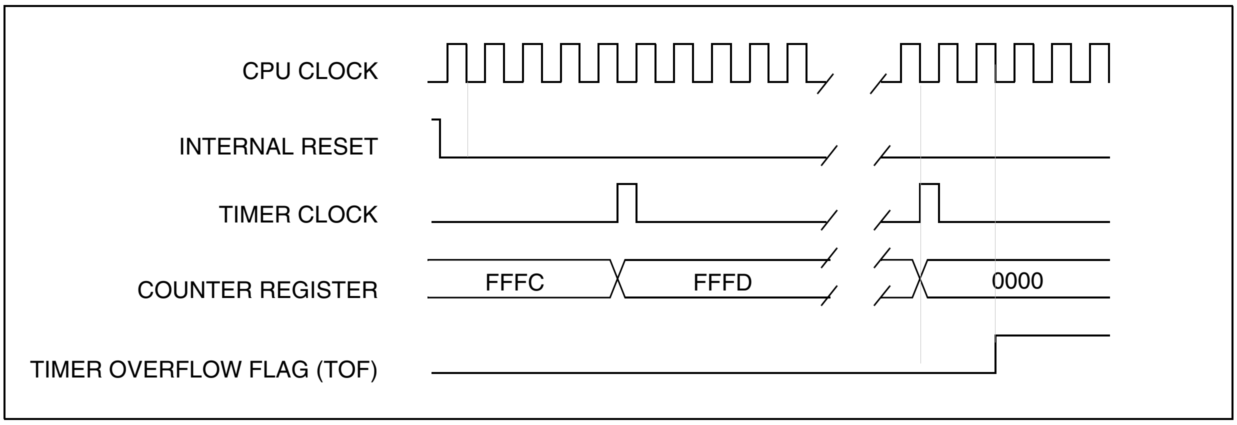


Figure 29. Counter Timing Diagram, internal clock divided by 8



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

16-BIT TIMER (Cont'd)

10.3.3.3 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 16-bit input capture registers (IC1R and IC2R) are used to latch the value of the free running counter after a transition is detected on 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 1).

– Select the edge of the active transition on the ICAP2 pin with the IEDG2 bit (the ICAP2 pin must be configured as floating input or input with pull-up without interrupt if this configuration is available).

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 or input with pull-up without interrupt if this configuration is available).

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 6).

– A timer interrupt is generated if the ICIE bit is set and the I bit is cleared in the CC register. Otherwise, the interrupt remains pending until both conditions become true.

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

1. Reading the SR register while the ICF*i* bit is set.
2. An access (read or write) to the 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 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 activates the input capture function.
   Moreover if one of the ICAP*i* pins is configured as an input and the second one as an output, an interrupt can be generated if the user toggles the output pin and if the ICIE bit is set.
   This can be avoided if the input capture function *i* is disabled by reading the IC*i*HR (see note 1).
6. The TOF bit can be used with interrupt generation in order to measure events that go beyond the timer range (FFFFh).

16-BIT TIMER (Cont'd)

Figure 30. Input Capture Block Diagram

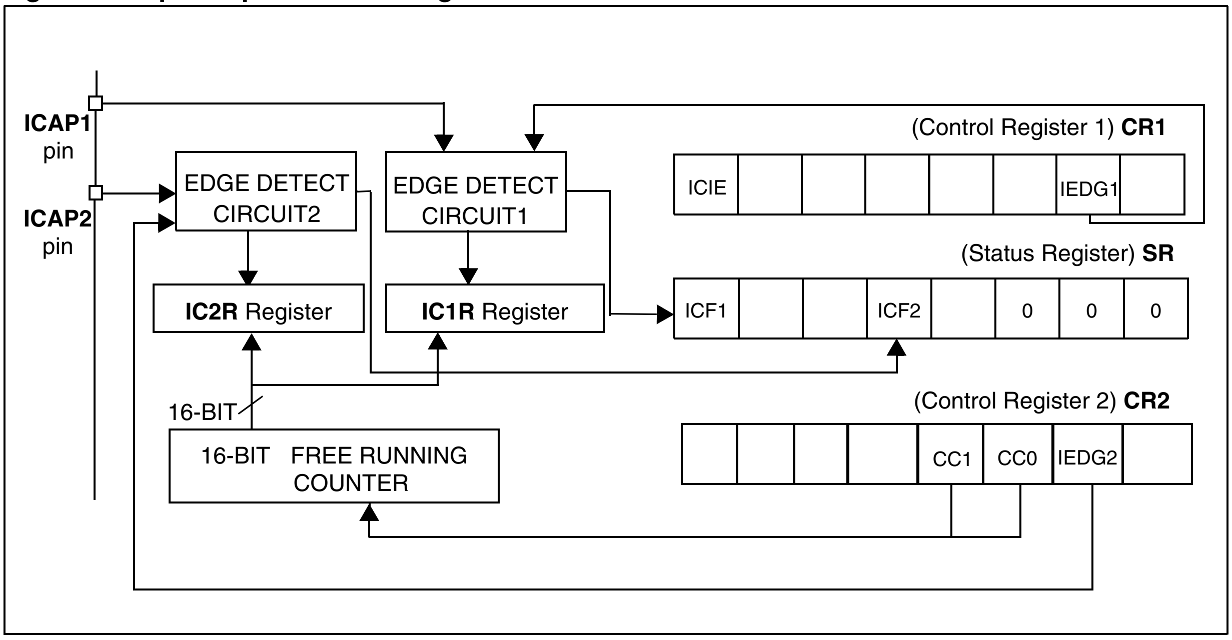
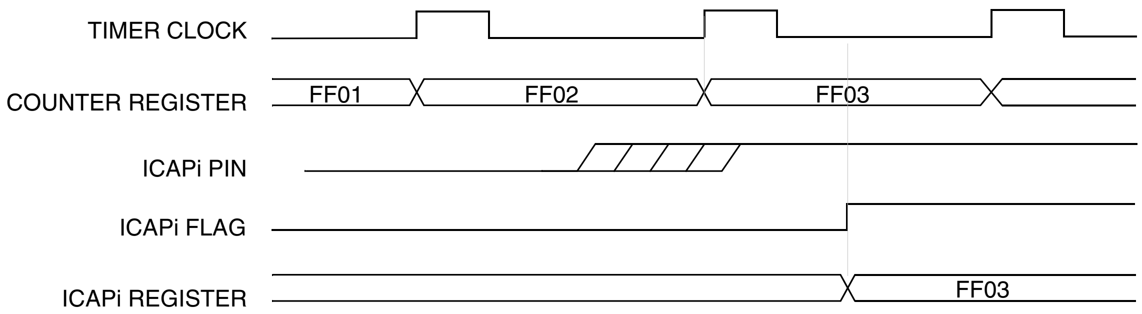


Figure 31. Input Capture Timing Diagram



Note: The rising edge is the active edge.

16-BIT TIMER (Cont'd)

10.3.3.4 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 OC*i*E 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 1).

And select the following in the CR1 register:

– Select the OLVL*i* bit to applied to the OCMP*i* pins after the match occurs.

– Set the OCIE bit to generate an interrupt if it is needed.

When a match is found between OCRi register and CR register:

– OCF*i* bit is set.

– The OCMP*i* pin takes OLVL*i* bit value (OCMP*i* pin latch is forced low during reset).

– A timer interrupt is generated if the OCIE bit is set in the CR1 register and the I bit is cleared in the CC register (CC).

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

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

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 1)

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:

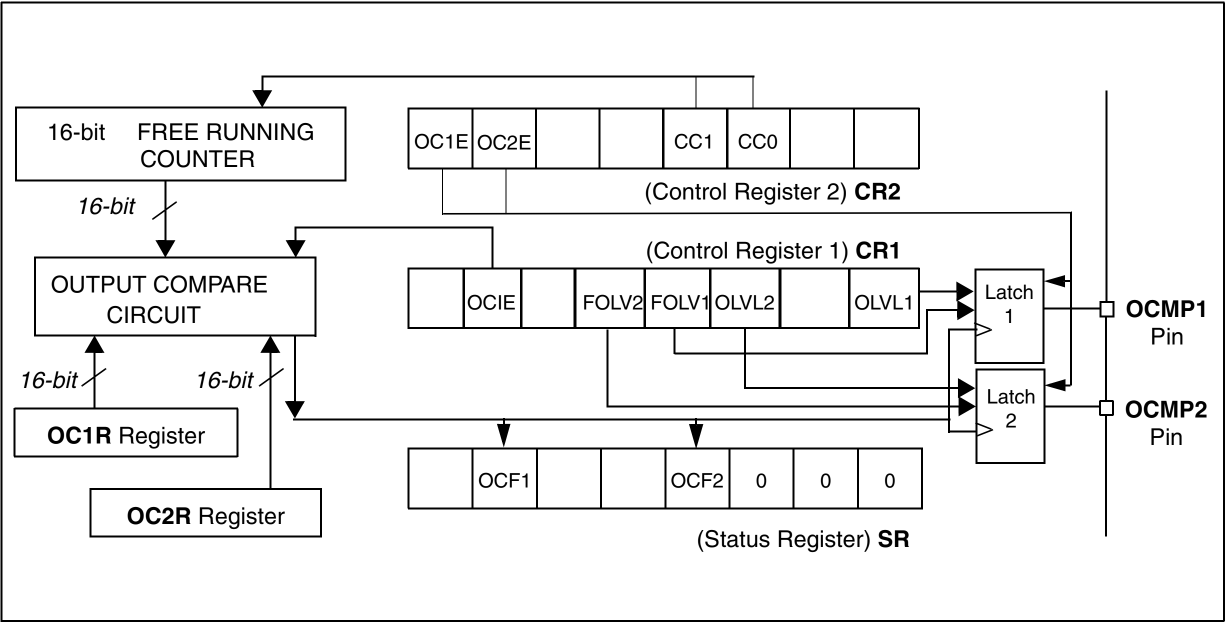
1. After a processor write cycle to the OC*i*HR register, the output compare function is inhibited until the OC*i*LR register is also written.
2. If the OC*i*E bit is not set, the OCMP*i* pin is a general I/O port and the OLVL*i* bit will not appear when a match is found but an interrupt could be generated 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 8). This behaviour is the same in OPM or PWM mode. When the timer clock is $f_{CPU}/4$, $f_{CPU}/8$ or in external clock mode, OCF*i* and OCMP*i* are set while the counter value equals the OC*i*R register value plus 1 (see Figure 9).
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.

The FOLVL*i* bits have no effect in both one pulse mode and PWM mode.

Figure 32. Output Compare Block Diagram



16-BIT TIMER (Cont'd)

Figure 33. Output Compare Timing Diagram, f_{TIMER} =f_{CPU}/2

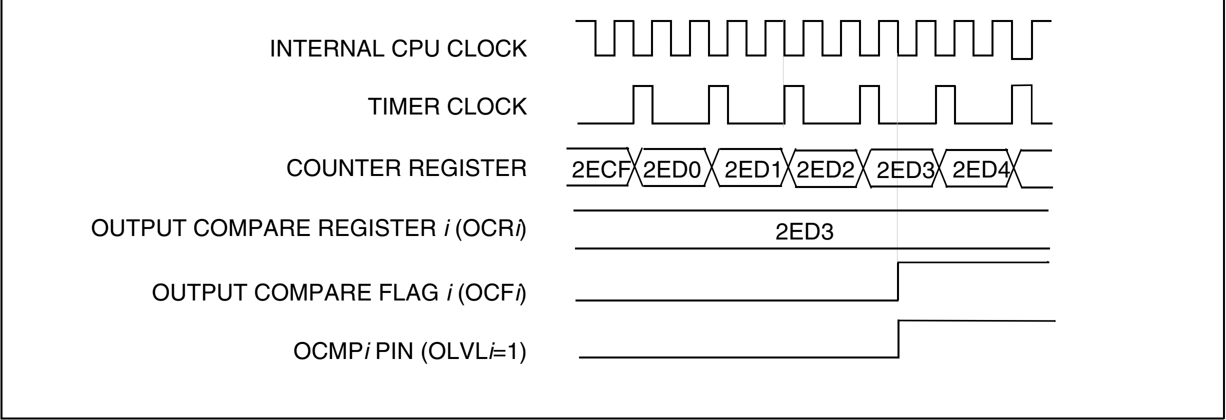
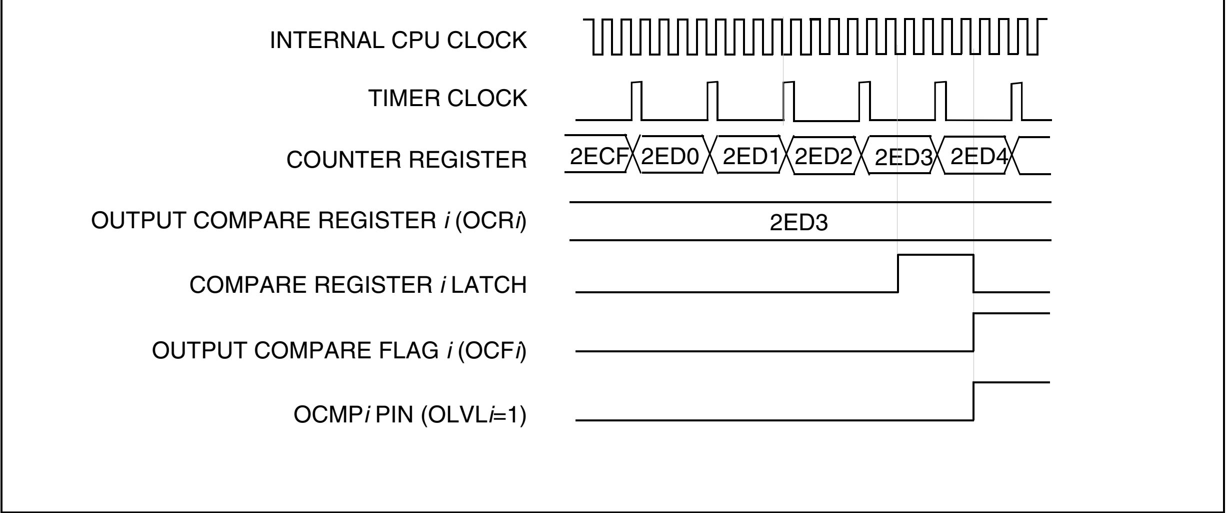


Figure 34. Output Compare Timing Diagram, f_{TIMER} =f_{CPU}/4



16-BIT TIMER (Cont'd)

10.3.3.5 One Pulse Mode

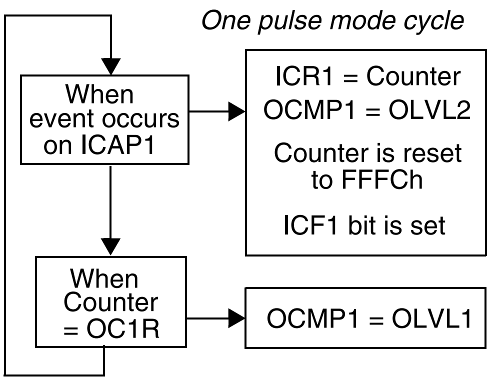
One Pulse mode enables the generation of a pulse when an external event occurs. This mode is selected via the OPM bit in the CR2 register.

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 1).



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

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

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

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

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

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

Where:

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

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)

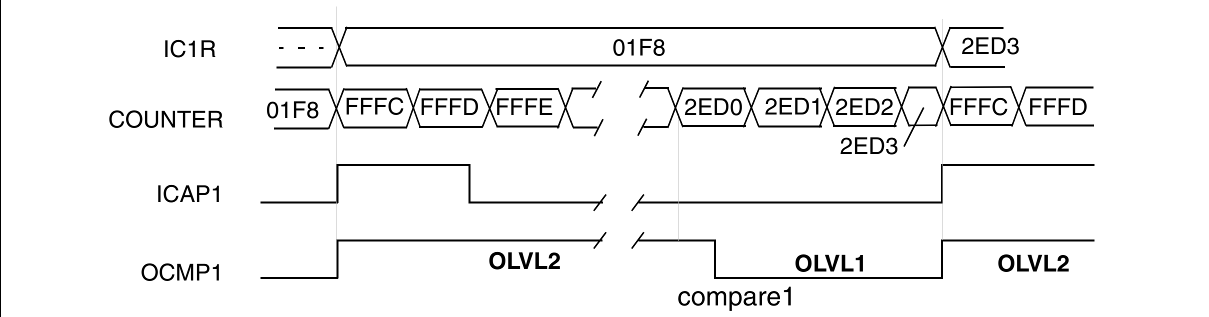
When the value of the counter is equal to the value of the contents of the OC1R register, the OLVL1 bit is output on the OCMP1 pin, (See Figure 10).

Notes:

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

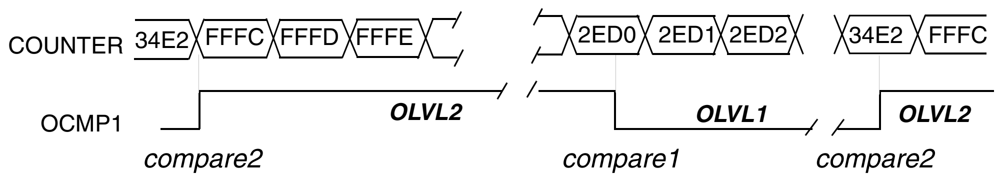
16-BIT TIMER (Cont'd)

Figure 35. One Pulse Mode Timing Example



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

Figure 36. Pulse Width Modulation Mode Timing Example with 2 Output Compare Functions



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

Note: On timers with only 1 Output Compare register, a fixed frequency PWM signal can be generated using the output compare and the counter overflow to define the pulse length.

16-BIT TIMER (Cont'd)

10.3.3.6 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.

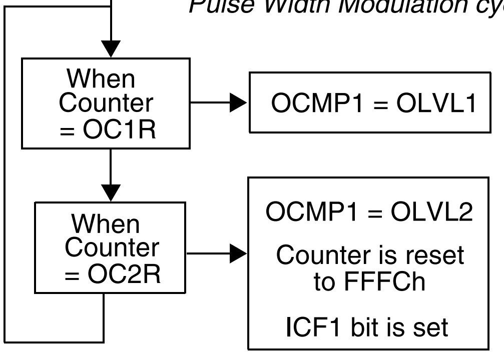
In PWM mode, double buffering is implemented on the output compare registers. Any new values written in the OC1R and OC2R registers are taken into account only at the end of the PWM period (OC2) to avoid spikes on the PWM output pin (OCMP1).

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 the OC1R register.
   – Using the OLVL2 bit, select the level to be applied to the OCMP1 pin after a successful comparison with the OC2R register.
4. Select the following in the CR2 register:
   – Set OC1E bit: the OCMP1 pin is then dedicated to the output compare 1 function.
   – Set the PWM bit.
   – Select the timer clock (CC[1:0]) (see Table 1).

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:

$$OC\textit{i}R \text{ Value} = \frac{t * f_{CPU}}{PRESC} - 5$$

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 1)

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

$$OC\textit{i}R = t * f_{EXT} - 5$$

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 11)

Notes:

1. After a write instruction to the OC*i*HR register, the output compare function is inhibited until the OC*i*LR register is also written.
2. The OCF1 and OCF2 bits cannot be set by hardware in PWM mode therefore the Output Compare interrupt is inhibited.
3. The ICF1 bit is set by hardware when the counter reaches the OC2R value and can produce a timer interrupt if the ICIE bit is set and the I bit is cleared.
4. In PWM mode the ICAP1 pin can not be used to perform input capture because it is disconnected 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.
5. When the Pulse Width Modulation (PWM) and One Pulse Mode (OPM) bits are both set, the PWM mode is the only active one.

16-BIT TIMER (Cont'd)

10.3.4 Low Power Modes

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

10.3.5 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).

10.3.6 Summary of Timer modes

MODES	TIMER 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 Recommended1)	No	Partially 2)
PWM Mode	No	Not Recommended3)	No	No

1) See note 4 in Section 0.1.3.5 One Pulse Mode

2) See note 5 in Section 0.1.3.5 One Pulse Mode

3) See note 4 in Section 0.1.3.6 Pulse Width Modulation Mode

16-BIT TIMER (Cont'd)

10.3.7 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 25. 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/Write (bits 7:3 read only)

Reset Value: xxxx x0xx (xxh)

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, or the counter reset, 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 26. 16-Bit Timer Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0048h	**TCR1** Reset Value	ICIE 0	OCIE 0	TOIE 0	FOLV2 0	FOLV1 0	OLVL2 0	IEDG1 0	OLVL1 0
0047h	**TCR2** Reset Value	OC1E 0	OC2E 0	OPM 0	PWM 0	CC1 0	CC0 0	IEDG2 0	EXEDG 0
0049h	**TCSR** Reset Value	ICF1 -	OCF1 -	TOF -	ICF2 -	OCF2 -	TIMD 0	- -	- -
004Ah	**TIC1HR** Reset Value	MSB -	-	-	-	-	-	-	LSB -
004Bh	**TIC1LR** Reset Value	MSB -	-	-	-	-	-	-	LSB -
004Ch	**TOC1HR** Reset Value	MSB 1	0	0	0	0	0	0	LSB 0
004Dh	**TOC1LR** Reset Value	MSB 0	0	0	0	0	0	0	LSB 0
004Eh	**TCHR** Reset Value	MSB 1	1	1	1	1	1	1	LSB 1
004Fh	**TCLR** Reset Value	MSB 1	1	1	1	1	1	0	LSB 0
0050h	**TACHR** Reset Value	MSB 1	1	1	1	1	1	1	LSB 1
0051h	**TACLR** Reset Value	MSB 1	1	1	1	1	1	0	LSB 0
0052h	**TIC2HR** Reset Value	MSB -	-	-	-	-	-	-	LSB -
0053h	**TIC2LR** Reset Value	MSB -	-	-	-	-	-	-	LSB -
0054h	**TOC2HR** Reset Value	MSB 1	0	0	0	0	0	0	LSB 0
0055h	**TOC2LR** Reset Value	MSB 0	0	0	0	0	0	0	LSB 0

10.4 SERIAL PERIPHERAL INTERFACE (SPI)

10.4.1 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.

10.4.2 Main Features

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

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

10.4.3 General Description

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

– SPI Control Register (SPICR)

– SPI Control/Status Register (SPICSR)

– SPI Data Register (SPIDR)

The SPI is connected to external devices through 4 pins:

– MISO: Master In / Slave Out data

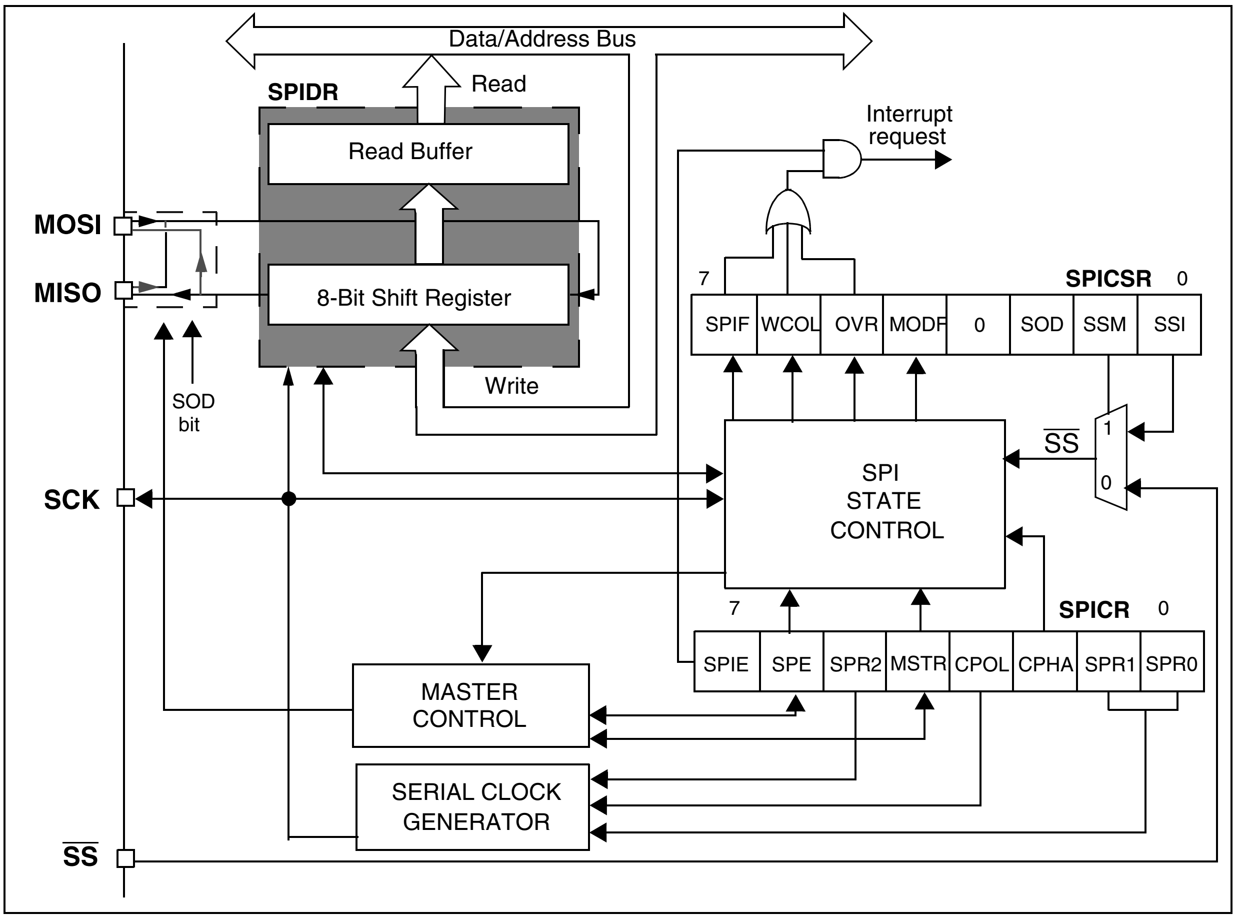
– MOSI: Master Out / Slave In data

– 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 (Cont'd)

Figure 37. Serial Peripheral Interface Block Diagram



SERIAL PERIPHERAL INTERFACE (Cont'd)

10.4.3.1 Functional Description

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

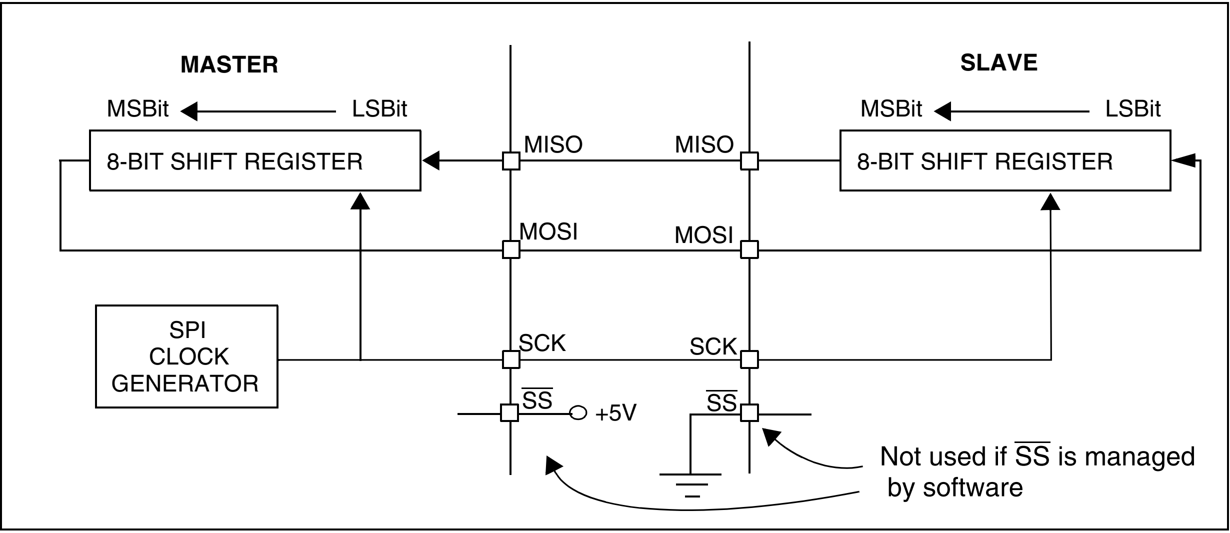
The MOSI pins are connected together and the MISO pins are connected together. In this way data is transferred serially between master and slave (most significant bit first).

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

Figure 38. Single Master/ Single Slave Application



SERIAL PERIPHERAL INTERFACE (Cont'd)

10.4.3.2 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 2nd clock edge):

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

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

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

Figure 39. Generic $\overline{SS}$ Timing Diagram

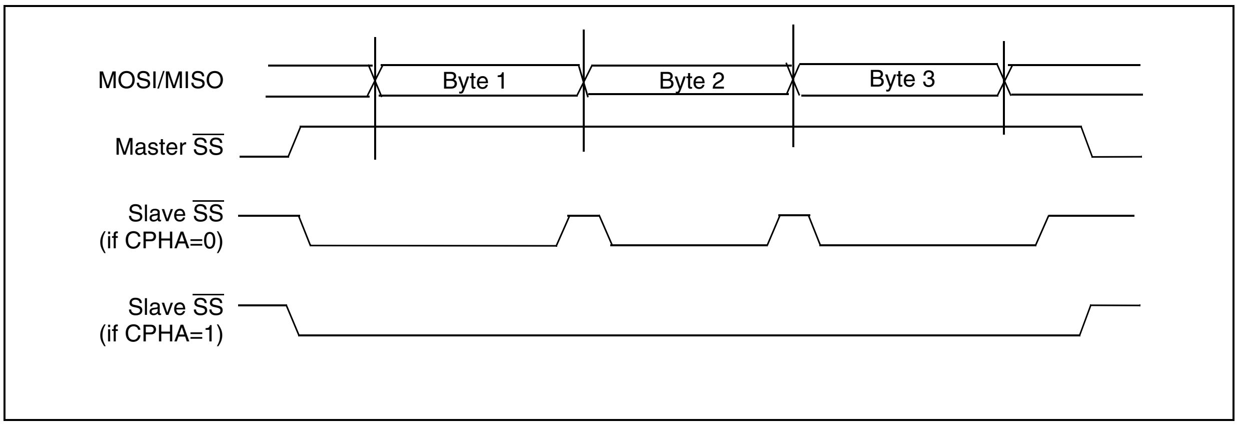
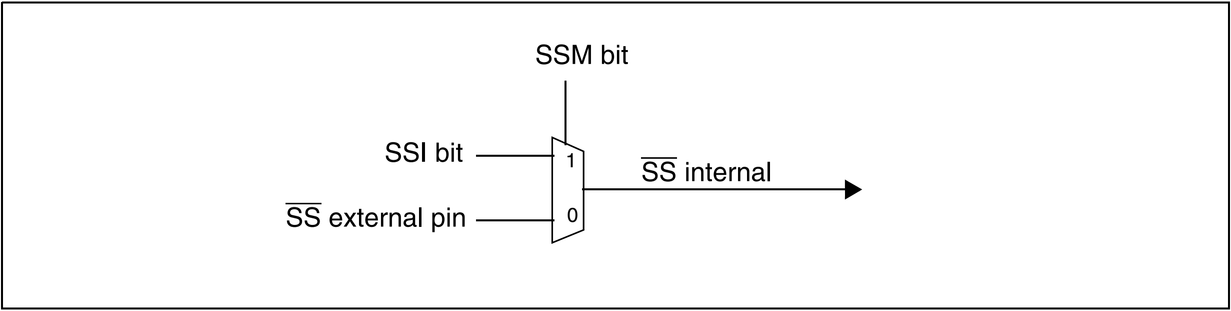

MOSI/MISO
Byte 1
Byte 2
Byte 3
Master $\overline{SS}$
Slave $\overline{SS}$ (if CPHA=0)
Slave $\overline{SS}$ (if CPHA=1)

Figure 40. Hardware/Software Slave Select Management



SERIAL PERIPHERAL INTERFACE (Cont'd)

10.4.3.3 Master Mode Operation

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

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

To operate the SPI in master mode, perform the following steps in order (if the SPICSR register is not written first, the SPICR register setting (MSTR bit) may be not taken into account):

1. Write to the SPICR register:
   – Select the clock frequency by configuring the SPR[2:0] bits.
   – Select the clock polarity and clock phase by configuring the CPOL and CPHA bits. Figure 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 $\overline{SS}$ pin high for the complete byte transmit sequence.
3. Write to the SPICR register:
   – Set the MSTR and SPE bits
   Note: MSTR and SPE bits remain set only if $\overline{SS}$ is high).

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

10.4.3.4 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.

10.4.3.5 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 $\overline{SS}$ pin as described in Section 0.1.3.2 and Figure 3. If CPHA=1 $\overline{SS}$ must be held low continuously. If CPHA=0 $\overline{SS}$ must be held low during byte transmission and pulled up between each byte to let the slave write in the shift register.
2. Write to the SPICR register to clear the MSTR bit and set the SPE bit to enable the SPI I/O functions.

10.4.3.6 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)

10.4.4 Clock Phase and Clock Polarity

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

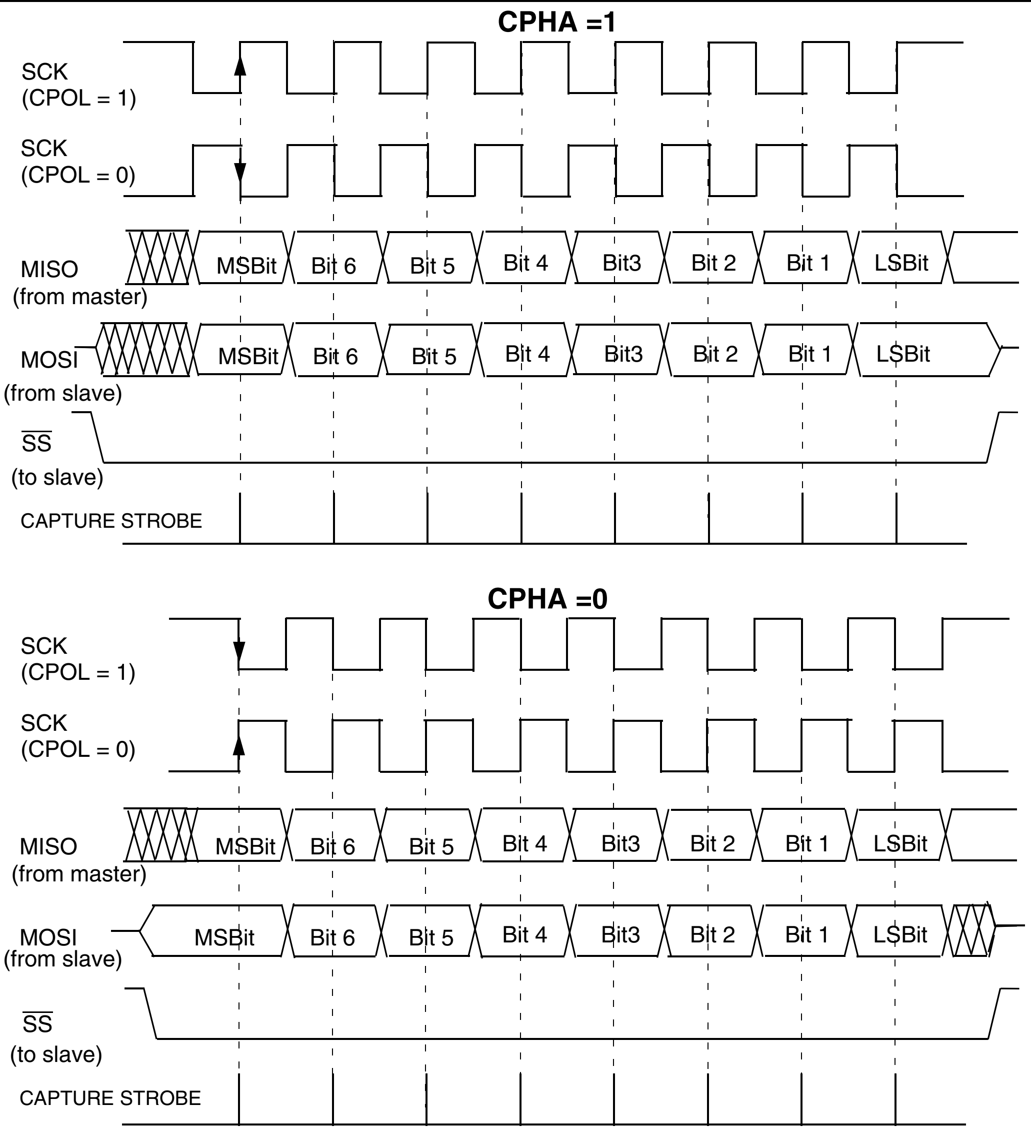
Note: The idle state of SCK must correspond to the polarity selected in the SPICSR register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).

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, 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 41. 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)

10.4.5 Error Flags

10.4.5.1 Master Mode Fault (MODF)

Master mode fault occurs when the master device has its $\overline{SS}$ pin pulled low.

When a Master mode fault occurs:

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

Clearing the MODF bit is done through a software sequence:

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

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

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

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

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

10.4.5.2 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.

10.4.5.3 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.

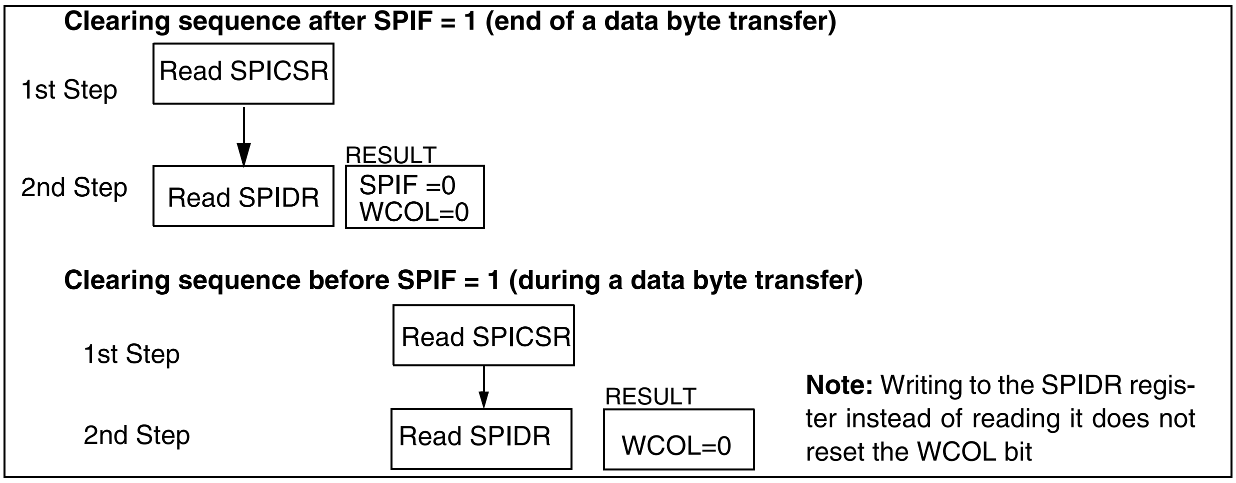
Note: a "read collision" will never occur since the received data byte is placed in a buffer in which access is always synchronous with the CPU operation.

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


Clearing sequence after SPIF = 1 (end of a data byte transfer)
1st Step
Read SPICSR
2nd Step
Read SPIDR
RESULT
SPIF =0
WCOL=0
Clearing sequence before SPIF = 1 (during a data byte transfer)
1st Step
Read SPICSR
2nd Step
Read SPIDR
RESULT
WCOL=0
Note: Writing to the SPIDR register instead of reading it does not reset the WCOL bit

SERIAL PERIPHERAL INTERFACE (Cont'd)

10.4.5.4 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.

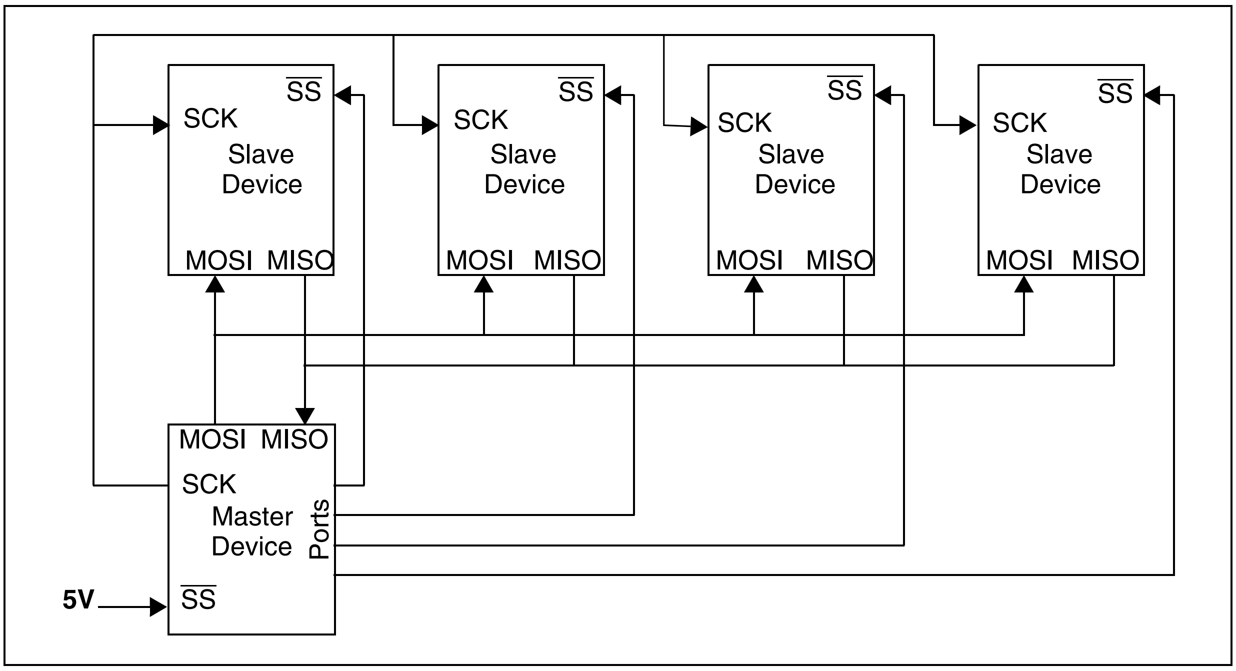
Other transmission security methods can use ports for handshake lines or data bytes with command fields.

Multi-Master System

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

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

Figure 43. Single Master / Multiple Slave Configuration



SERIAL PERIPHERAL INTERFACE (Cont'd)

10.4.6 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.

10.4.6.1 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.

10.4.7 Interrupts

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

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

SERIAL PERIPHERAL INTERFACE (Cont'd)

10.4.8 Register Description

CONTROL REGISTER (SPICR)

Read/Write

Reset Value: 0000 xxxx (0xh)

7							0
SPIE	SPE	SPR2	MSTR	CPOL	CPHA	SPR1	SPR0

Bit 7 = **SPIE** *Serial Peripheral Interrupt Enable.*
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SPI interrupt is generated whenever an End of Transfer event, Master Mode Fault or 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 27. SPI Master mode SCK Frequency

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

SERIAL PERIPHERAL INTERFACE (Cont'd)

CONTROL/STATUS REGISTER (SPICSR)

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

7							0
SPIF	WCOL	OVR	MODF	-	SOD	SSM	SSI

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

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

Bit 6 = **WCOL** *Write Collision status (Read only).*
This bit is set by hardware when a write to the SPIDR register is done during a transmit sequence. It is cleared by a software sequence (see Figure 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

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 28. SPI Register Map and Reset Values

Address (Hex.)	**Register Label**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0061h	**SPIDR** Reset Value	MSB x	x	x	x	x	x	x	LSB x
0062h	**SPICR** Reset Value	SPIE 0	SPE 0	SPR2 0	MSTR 0	CPOL x	CPHA x	SPR1 x	SPR0 x
0063h	**SPICSR** Reset Value	SPIF 0	WCOL 0	OVR 0	MODF 0	0	SOD 0	SSM 0	SSI 0

10.5 USB INTERFACE

10.5.1 Introduction

The USB Interface implements a high/full speed function interface between the USB and the ST7 microcontroller. It is a highly integrated circuit which includes the transceiver, USB controller and USB Data Buffer interface. No external components are needed apart from the external reference resistor.

This USB function is based on ST PHY and Mentor MUSBHSFC USB2.0 Function controller. So parts of this specification are based on Mentor Graphics design documentation and used by permission.

10.5.2 Main Features

- USB Specification Version 2.0 Compliant
- On-Chip USB PHY
- Supports High/Full Speed USB Protocol
- 1 control endpoint with two 64 byte buffers
- 1 IN bulk / interrupt endpoint with 64 byte buffers
- 1 OUT bulk / interrupt endpoint with 64 byte buffers
- 1 IN bulk endpoint with a double packet buffering capability (2*512 bytes)
- 1 OUT bulk endpoint with a double packet buffering capability (2*512 bytes)
- Specific data transfer mode between USB buffer and MSCI for high transfer rate (does not require ST7 intervention)
- USB Suspend/Resume operations

10.5.3 Functional Description

The block diagram in Figure 44, gives an overview of the USB interface hardware.

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

USB2.0 PHY

The USB2.0 PHY serialises or deserialises the USB data in order to send them in parallel (16-bit) to the USB packet encoding/decoding block.

Packet Encoding/Decoding CRC

This block Encodes/Decodes the packet to be sent/to be received through the UTMI interface.

It also performs frame formatting, including CRC generation and checking.

Endpoint Control

This block is composed of two controller state machines one for endpoint 0 and another for endpoints 1 and 2.

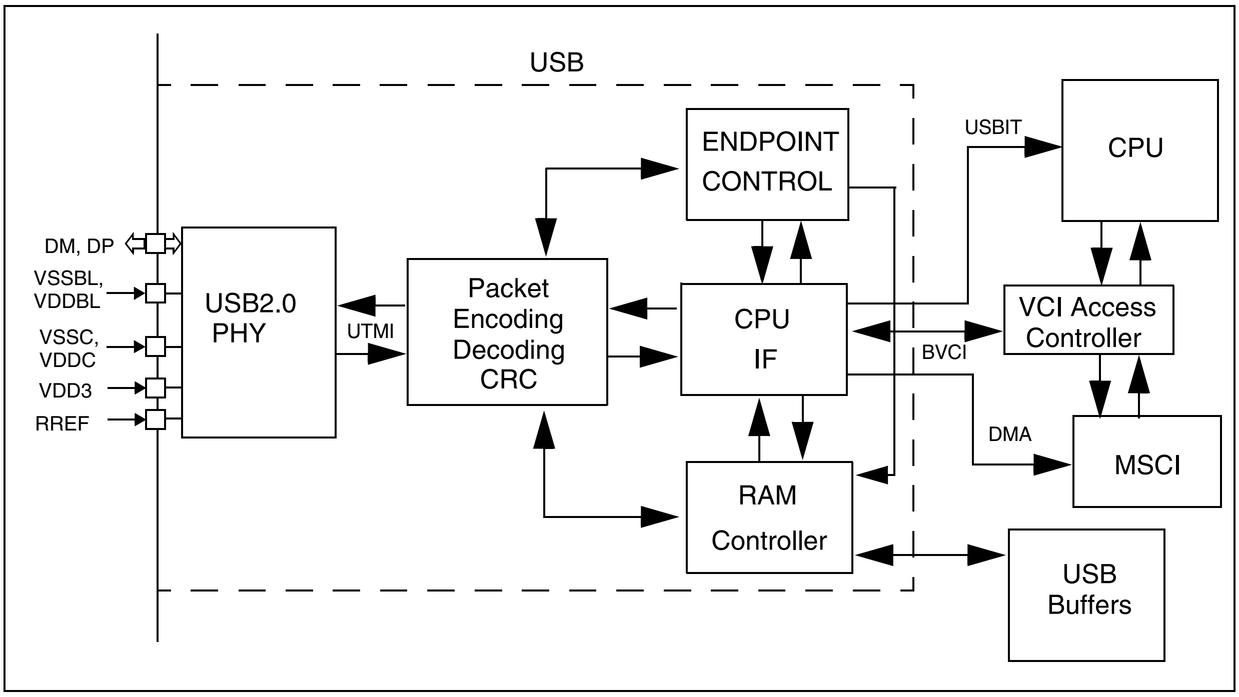
CPU interface

The CPU interface provides the access to the control and status registers and the USB buffers (FIFO) for each endpoint (through RAM controller block). It also generates an interrupt at the end of a reception / transmission or when suspend or resume is detected. The CPU interface is compatible with the VSIA standard BVCI (Basic Virtual Component Interface).

RAM controller

The RAM controller generates the SRAM control signals to access the endpoint FIFOs selected by the Endpoint Control block pointer information.

Figure 44. USB Block Diagram



10.5.4 USB2.0 PHY

The USB2.0 PHY serialises or deserialises the USB data in order to send it in parallel (16-bit) to the USB packet encoding/decoding block through a UTMI compliant interface.

The USB2.0 PHY requires only one external reference resistor (11.5kΩ) to be connected to RREF chip input for process compensation.

10.5.5 USB buffers

The USB buffers are as follows:

- two 64 byte buffers for control transfer on endpoint 0
- two 64 byte buffers, one for IN and one for OUT bulk / interrupt transfer on endpoint 1
- two double 512 byte buffers, one for IN (2*512) and one for OUT (2*512) bulk transfer on endpoint 2

This buffer is implemented by a 2304 byte SRAM. To access to these endpoint buffers three addresses are available, one for each endpoint. Writing to these addresses loads data into the IN buffer of the corresponding endpoint. Reading from these addresses unloads data from the OUT buffer of the corresponding endpoint.

USB INTERFACE (Cont'd)

10.5.6 Register Description

The registers are divided in three groups:

– common USB registers (function controller control and status registers)

– indexed registers (endpoint control and status registers)

– buffer register access

– end of suspend detection registers

10.5.6.1 Common USB registers

POWER REGISTER (PWRR)

Read/Write

Reset Value: 0010 0000 (20h)

7							0
0	SCON	HSE	HSM	RST	RSM	SUSM	ESUSM

Bit 7 = Reserved, forced by hardware to 0.

Bit 6 = **SCON** *Soft Connect/Disconnect*.
This bit is set by software to enabled the USB D+ / D- lines.
0: D+/D- tri-stated
1: D+/D- lines enabled

Bit 5 = **HSE** *HS Enable*.
This bit is set by software to negotiate for high speed mode when the device is reset by the hub.
0: Full Speed mode
1: High Speed mode

Bit 4 = **HSM** *HS Mode (Read Only)*.
This bit is set by hardware when the function has successfully negotiated the HS mode during the Reset phase.
0: FS mode
1: HS mode

Bit 3 = **RST** *Reset (Read Only).*
This bit is set by hardware while reset signalling is present on the bus.
0: No Reset on the bus
1: Reset on the bus

Bit 2 = **RSM** *Resume*.
This bit is set by software to generate Resume signalling to wake-up from suspend mode.
0: No Resume generated
1: Resume generated

Software should clear this bit after 10ms (and before 15ms) to end Resume signalling.

Bit 1 = **SUSM** *Suspend Mode (Read Only).*
This bit is set by hardware when suspend mode is entered. It is cleared when software reads the interrupt register or sets the RSM bit.
0: Suspend mode inactive
1: Suspend mode active

Bit 0 = **ESUSM** *Enable Suspend Mode.*
This bit is set by software to enable the SUSPENDM UTMI signal.
0: SUSPENDM signal not activated
1: SUSPENDM signal activated

When this bit is not set suspend mode will be detected but the SUSPENDM signal will remain high (active low).

USB INTERFACE (Cont'd)

FUNCTION ADDRESS (FADDR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
UPD	FAD6	FAD5	FAD4	FAD3	FAD2	FAD1	FAD0

Bit 7 = **UPD** *Update* (Read Only).
Set when FADDR is written. Cleared when the new address takes effect (at the end of the current transfer).

Bits 6:0 = **FAD[6:0]** *Function address.*
These bits are written by software with the function address provided by the SET_ADDRESS standard device request (see *Universal Serial Bus Specification* Revision 2.0, Chapter 9).

INTERRUPT IN REGISTER (ITINR)

Read only.

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	0	EP2I	EP1I	EP0

This register indicates which IN endpoint (1 or 2) interrupt is currently pending. It also indicates whether the Endpoint 0 interrupt is currently active

Bit 7:3 = Reserved.

Bit 2 = **EP2I** *Endpoint 2 IN flag*.
0: No IN interrupt on endpoint 2
1: IN interrupt on endpoint 2

Bit 1 = **EP1I** *Endpoint 1 IN flag*.
0: No IN interrupt on endpoint 1
1: IN interrupt on endpoint 1

Bit 0 = **EP0** *Endpoint 0 flag*.
0: No interrupt on endpoint 0
1: Interrupt on endpoint 0

Note: all pending interrupt flags are cleared when this register is read.

INTERRUPT OUT REGISTER (ITOUTR)

Read only.

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	0	EP2O	EP1O	0

This register indicates which OUT Endpoint interrupt is currently pending.

Bit 7:3 = Reserved.

Bit 2 = **EP2O** *Endpoint 2 OUT*.
0: No OUT interrupt on endpoint 2
1: OUT interrupt on endpoint 2

Bit 1 = **EP1O** *Endpoint 1 OUT*.
0: No OUT interrupt on endpoint 1
1: OUT interrupt on endpoint 1

Bit 0 = Reserved.

Note: all active interrupts are cleared when this register is read.

INTERRUPT IN ENABLE REGISTER (ITINER)

Read/Write.

Reset value: 0000 0111 (07h)

7							0
0	0	0	0	0	EP2IE	EP1IE	EP0E

This register indicates which IN Endpoint (1 or 2) interrupt is enabled. It also indicates whether the Endpoint 0 interrupt is enabled.

Bit 7:3 = Reserved.

Bit 2 = **EP2IE** *Endpoint 2 IN Enabled*.
0: IN interrupt on endpoint 2 disabled
1: IN interrupt on endpoint 2 enabled

Bit 1 = **EP1IE** *Endpoint 1 IN* Enabled.
0: IN interrupt on endpoint 1 disabled
1: IN interrupt on endpoint 1 enabled

Bit 0 = **EP0E** *Endpoint 0* Enabled.
0: interrupt on endpoint 0 disabled
1: Interrupt on endpoint 0 enabled

INTERRUPT OUT ENABLE REGISTER (ITOUTER)

Read/Write.

Reset value: 0000 0110 (06h)

7							0
0	0	0	0	0	EP2OE	EP1OE	0

This register indicates which OUT Endpoint interrupt is enabled.

Bit 7:3 = Reserved.

Bit 2 = **EP2OE** *Endpoint 2 OUT Enable*.
0: OUT interrupt on endpoint 2 disabled
1: OUT interrupt on endpoint 2 enabled

Bit 1 = **EP1OE** *Endpoint 1 OUT Enable*.
0: OUT interrupt on endpoint 1 disabled
1: OUT interrupt on endpoint 1 enabled

Bit 0 = Reserved.

INTERRUPT USB ENABLE REGISTER (ITUSBER)

Read/Write.

Reset value: 0000 0110 (06h)

7							0
0	0	0	0	SOFE	RSTE	RSME	SUSPE

These bits are written by software to enable / disable the USB interrupts.

Bit 7:4 = Reserved.

Bit 3= **SOFE** *Start Of Frame Enable*.
0: SOF interrupt disabled
1: SOF interrupt enabled

Bit 2 = **RSTE** *Reset Enable*.
0: Reset interrupt disabled
1: Reset interrupt enabled

Bit 1 = **RSME** *Resume Enable*.
0: Resume interrupt disabled
1: Resume interrupt enabled

Bit 0 = **SUSPE** *Suspend Enable*.
0: Suspend interrupt disabled
1: Suspend interrupt enabled

INTERRUPT USB REGISTER (ITUSBR)

Read only.

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	SOF	RST	RSM	SUSP

This register indicates which USB interrupts are currently pending.

Bit 7:4 = Reserved.

Bit 3= **SOF** *Start Of Frame Interrupt*.
0: No SOF interrupt
1: SOF interrupt pending

Bit 2 = **RST** *Reset Interrupt*.
0: No Reset interrupt
1: Reset interrupt pending

Bit 1 = **RSM** *Resume Interrupt*.
0: No Resume interrupt
1: Resume interrupt pending

Bit 0 = **SUSP** *Suspend Interrupt*.
0: No Suspend interrupt
1: Suspend interrupt pending

Note: all pending interrupts are cleared when this register is read.

USB INTERFACE (Cont'd)

FRAME NUMBER REGISTER MSB (FRNBRM)

Read Only.

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	0	FN10	FN9	FN8

This register indicates the MSB of the last frame number received.

Bit 7:3 = Reserved.

Bit 2:0 = **FN[10:8]** *Frame number MSB*.

FRAME NUMBER REGISTER LSB (FRNBRL)

Read Only.

Reset value: 0000 0000 (00h)

7							0
FN7	FN6	FN5	FN4	FN3	FN2	FN1	FN0

This register indicates the LSB of the last frame number received.

Bit 7:0 = **FN[7:0]** *Frame number LSB*.

TEST MODE REGISTER (TSTMODER)

Read/Write

Reset value: 0000 0000(00h)

7							0
0	0	FFS	FHS	TPAK	TK	TJ	TSE0N

This register is used to put the USB device into one of the four test modes described in the USB 2.0 specification. It is not used in normal operation.

Bit 7:6 = Reserved.

Bit 5 = **FFS** *Force Full-Speed*.
Software can set this bit to force the USB device into Full-speed mode when it receives a USB reset.

Bit4 = **FHS** *Force High-Speed*.
Software can set this bit to force the USB device into High-speed mode when it receives a USB reset.

Bit 3 = **TPAK** *Test Packet*.
Software can set this bit to enter the Test_Packet test mode. In this mode, the USB device– in high-speed mode – repetitively transmits on the bus a 53-byte test packet.

Note: The 53-byte test packet must be loaded into the Endpoint 0 FIFO before the test mode is entered.

Bit 2 = **TK** *Test K*.
Software can set this bit to enter the Test_K test mode. In this mode, the USB device– in high-speed mode – transmits a continuous K on the bus.

Bits 1 = **TJ** *Test J*.
Software can set this bit to enter the Test_J test mode. In this mode, the MUSBHSFC – in high-speed mode – transmits a continuous J on the bus.

Bits 0= **TSE0N** *Test SE0 NAK.*
Software can set this bit to enter the Test_SE0_NAK test mode. In this mode, the USB Device remains in high-speed mode and responds to any valid IN token with a NAK.

INDEX REGISTER (INDEXR)

Read/Write

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	IND1	IND0

This register is written by software to determine which endpoint control/status registers are addressed at 10h to 19h.

Bits [7:2] = Reserved.

Bit 1:0= **IND[1:0]** *Index*.

IND1	IND0	Meaning
0	0	Endpoint 0 addressing
0	1	Endpoint 1 addressing
1	0	Endpoint 2 addressing
1	1	-

Each IN endpoint and each OUT endpoint has its own set of control/status registers. Only one set of IN control/status and one set of OUT control/status registers appear in the memory map at any one time. Before accessing an endpoint's control/status registers, the endpoint number should be written to the Index register to ensure that the correct control/status registers appear in the memory map

10.5.6.2 Indexed registers

Note: The action of the following registers is undefined if the selected endpoint has not been configured.

IN MAX PACKET REGISTER MSB (INMAXPRM)

Read/Write

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	0	IMP10	IMP9	IMP8

This register defines the most significant byte of the maximum payload transmitted in a single transaction.

Bits 7:3 = Reserved.

Bit 2:0= **IMP[10:8]** *IN Max Packet*.

IN MAX PACKET REGISTER LSB (INMAXPRL)

Read/Write

Reset value: 0000 0000 (00h)

7							0
IMP7	IMP6	IMP5	IMP4	IMP3	IMP2	IMP1	IMP0

This register defines the lowest significant byte of the maximum payload transmitted in a single transaction.

Bits 7:0 = **IMP[7:0]** *IN Max Packet*.

INMAXPR are registers that define the maximum amount of data that can be transferred through the selected IN endpoint in a single frame / microframe (High-speed transfers). There is an INMAXPR register for each IN endpoint (except Endpoint 0).

The value written to the INMAXPRx registers should match the *wMaxPacketSize* field of the Standard Endpoint Descriptor for the associated endpoint (see *Universal Serial Bus Specification* Revision 2.0, Chapter 9). A mismatch could cause unexpected results.

The total amount of data represented by the value written to these registers (maximum payload × maximum number of transactions) must not exceed the FIFO size for the IN endpoint, and should not exceed half the FIFO size if double-buffering is required.

If these registers are changed after packets have been sent from the endpoint, the IN endpoint FIFO should be completely flushed (using the FLFI bit in INCSRL) after writing the new value to these registers.

IN CONTROL STATUS REGISTER MSB (INCSRM)

INCSRM is the MSB of a register that provides control and status bits for IN transactions through the currently-selected endpoint. There is an INCSRM register for each IN endpoint (not including Endpoint 0).

For endpoint 0:

This register is reserved and returns 00h.

For endpoint 1 and 2:

Read/Write

Reset value: 0000 0000 (00h)

7							0
ASET	o	0	DMAE	FDT	0	0	0

Bit 7 = **ASET** *Auto Set*.
If the CPU sets this bit, IPR will be automatically set when data of the maximum packet size (value in INMAXPR) is loaded into the IN FIFO. If a packet of less than the maximum packet size is loaded, IPR will have to be set manually.

Bit 6:5 = Reserved.

Bit 4 = **DMAE** *DMA Enable*
0: DMA request for the IN endpoint disabled

1: DMA request for the IN endpoint enabled.

Bit 3 = **FDT** *Force Data Toggle*

The CPU sets this bit to force the endpoint's IN data toggle to switch after each data packet is sent regardless of whether an ACK was received.
0: Data toggle not forced
1: Data toggle forced

Bits 2:0= Reserved.

IN CONTROL STATUS REGISTER LSB (INCSRL)

INCSRL is the LSB of a register that provides control and status bits for IN transactions through the currently-selected endpoint. There is an INCSRL register for each IN endpoint (not including Endpoint 0). For endpoint 0 this control status register is common to SETUP, IN or OUT transactions.

For endpoint 0 (CSR0):

CSR0 appears in the memory map when the Index register is set to 0. It is used for all control/status of Endpoint 0. For details of how to service device requests to Endpoint 0, see Section 12.4.10: 'Endpoint 0 Handling'.

Read/Write

Reset value: 0000 0000 (00h)

7							0
SSE	SOPR	SDST	SE	DE	STST	IPR	OPR

Bit 7 = **SSE** *Serviced Setup End*.
Software writes a 1 to this bit to clear the SE bit. SSE is cleared automatically.

Bit 6 = **SOPR** *Serviced OUT Packet Ready*.
Software writes a 1 to this bit to clear the OPR bit. SOPR is cleared automatically

Bit 5 = **SDST** *Send Stall.*
Software writes a 1 to this bit to terminate the current transaction. The STALL handshake will be transmitted and then this bit will be cleared automatically.

Bit 4 = **SE** *Setup End (Read Only)*
This bit will be set when a control transaction ends before the DE bit has been set. An interrupt will be generated and the FIFO flushed at this time. The bit is cleared by software writing a 1 to the SSE bit.

Bit 3 = **DE** *Data End*
Software sets this bit:
- when setting IPR bit for the last data packet.
- when clearing OPR bit after unloading the last data packet.
- when setting IPR bit for a zero length data packet.
It is cleared automatically.

Bit 2 = **STST** *Sent Stall*
This bit is set when a STALL handshake is transmitted. The CPU should clear this bit.

Bit 1 = **IPR** *In Packet Ready*
Software sets this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is generated when the bit is cleared.

Bit 0 = **OPR** *Out Packet Ready (Read Only)*
This bit is set when a data packet has been received. An interrupt is generated when this bit is set. Software clears this bit by setting the SOPR bit.

For endpoint 1 and 2:

Read/Write

Reset value: 0000 0000 (00h)

7							0
0	CDT	STST	SDST	FLFI	UNDR	FINE	IPR

Bit 7 = Reserved.

Bit 6 = **CDT** *Clear Data Toggle*.
Software writes a 1 to this bit to reset the endpoint IN data toggle to 0.

Bit 5 = **STST** *Sent Stall*
This bit is set when a STALL handshake is transmitted. The FIFO is flushed and the IPR bit is cleared. Software should clear this bit.

Bit 4 = **SDST** *Send Stall*
The CPU writes a 1 to this bit to issue a a STALL handshake to an IN token. The CPU clears this bit to terminate the stall condition.

Bit 3 = **FLFI** *Flush FIFO (Self clearing)*

Software writes a 1 to this bit to flush the next packet to be transmitted from the endpoint IN FIFO. The FIFO pointer is reset and the IPR bit is cleared.

Note: If the FIFO contains two packets, FlushFIFO will need to be set twice to completely clear the FIFO

Bit 2 = **UNDR** *Under Run*
This bit is set when a NAK is returned in response to an IN token. Software should clear this bit.

Bit 1 = **FINE** *FIFO not empty*
This bit is set when there is at least 1 packet in the IN FIFO.

Bit 0 = **IPR** *In Packet Ready*
Software sets this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. If the FIFO is double-buffered, it is also automatically cleared when there is space for a second packet in the FIFO. An interrupt is generated (if enabled) when this bit is cleared.

OUT MAX PACKET REGISTER MSB (OUTMAX-PRM)

Read/Write

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	0	OMP10	OMP9	OMP8

This register defines the most significant byte of the maximum payload transmitted in a single transaction.

Bits 7:3 = Reserved.

Bit 2:0= **OMP[10:8]** *OUT Max Packet*.

OUT MAX PACKET REGISTER LSB (OUTMAX-PRL)

Read/Write

Reset value: 0000 0000 (00h)

7							0
OMP7	OMP6	OMP5	OMP4	OMP3	OMP2	OMP1	OMP0

This register defines the least significant byte of the maximum payload transmitted in a single transaction.

Bits 7:0 = **OMP[7:0]** *OUT Max Packet*.

OUTMAXPR are registers that define the maximum amount of data that can be transferred through the selected OUT endpoint in a single frame / microframe (High-speed transfers). There is an OUTMAXP register for each OUT endpoint (except Endpoint 0).

The value written to the OUTMAXPRx registers should match the *wMaxPacketSize* field of the Standard Endpoint Descriptor for the associated endpoint (see *Universal Serial Bus Specification* Revision 2.0, Chapter 9). A mismatch could cause unexpected results.

The total amount of data represented by the value written to these registers (maximum payload × maximum number of transactions) must not exceed the FIFO size for the OUT endpoint, and should not exceed half the FIFO size if double-buffering is required.

OUT CONTROL STATUS REGISTER MSB (OUTCSRM)

OUTCSRM is the MSB of a register that provides control and status bits for OUT transactions through the currently-selected endpoint.

Read/Write

Reset value: 0000 0000 (00h)

7							0
ACLR	o	DMAE	DNY	DMAM	0	0	0

Bit 7 = **ACLR** *Auto Clear*.
If software sets this bit then the OPR bit will be au-

tomatically cleared when a packet of OUTMAXP bytes has been unloaded from the OUT FIFO. When packets of less than the maximum packet size are unloaded, OPR will have to be cleared manually.

Bit 6 = Reserved.

Bit 4 = **DMAE** DMA Enable
0: DMA request for the OUT endpoint disabled
1: DMA request for the OUT endpoint enabled.

Bit 4 = **DNY** *Disable Nyet*
Software sets this bit to disable the sending of NYET handshakes. When set, all successfully received OUT packets are ACK'd including at the point at which the FIFO becomes full.

Note: This bit only has effect in High-speed mode. In this mode it should be set for all Interrupt endpoints.

Bit 3 = **DMAM** *DMA Mode*
Two modes of operation are supported: In DMA Mode 0 a DMA request is generated for all received packets, together with an interrupt (if enabled); In DMA Mode 1 a DMA request (but no interrupt) is generated for OUT packets of size OUTMAXP bytes and an interrupt (but no DMA request) is generated for OUT packets of any other size. DMAM is set by software to select the DMA mode.
0: DMA Mode 0
1: DMA Mode 1

Bit 2:0= Reserved.

OUT CONTROL STATUS REGISTER LSB (OUTCSRL)

OUTCSRL is the LSB of a register that provides control and status bits for OUT transactions through the currently-selected endpoint.

Read/Write

Reset value: 0000 0000 (00h)

7							0
CDT	STST	SDST	FLFI	0	0	FIFU	OPR

Bit 7 = **CDT** *Clear Data Toggle*.
Software writes a 1 to this bit to reset the endpoint OUT data toggle to 0.

Bit 6 = **STST** *Sent Stall*
This bit is set when a STALL handshake is transmitted. Software should clear this bit.

Bit 5= **SDST** *Send Stall*
Software writes a 1 to this bit to terminate the current transaction. The STALL handshake will be transmitted and then this bit will be cleared automatically.

Bit 4= **FLFI** *Flush FIFO (Self clearing)*

Software writes a 1 to this bit to flush the next packet to be read from the endpoint OUT FIFO. **Note:** If the FIFO contains two packets, FlushFIFO will need to be set twice to completely clear the FIFO.

Bit 3:2 = Reserved.

Bit 1 = **FIFU** *FIFO full flag*
This bit is set when no more packets can be loaded into the OUT FIFO. FIFU is cleared by hardware.

Bit 0 = **OPR** *OUT Packet Ready flag*
This bit is set when a data packet has been received. Software should clear this bit when the packet has been unloaded from the OUT FIFO. An interrupt is generated (if enabled) when the bit is set.

OUT COUNT REGISTER MSB (OUTCNTRM)

Read Only

Reset value: 0000 0000 (00h)

7							0
0	0	0	OC12	OC11	OC10	OC9	OC8

OUTCNTRM is the MSB register that holds the number of received data bytes in the packet in the OUT FIFO.

Note: The value returns changes as the contents of the FIFO change and is only valid while OPR bit in the OUTCSRL register is set.

Bits 7:5 = Reserved.

Bit 4:0= **OC[12:8]** *OUT Count*.

OUT COUNT REGISTER LSB (OUTCNTRL)

Read/Write

Reset value: 0000 0000 (00h)

7							0
OC7	OC6	OC5	OC4	OC3	OC2	OC1	OC0

OUTCNTRL is the LSB register that holds the number of received data bytes in the packet in the OUT FIFO.

Notes:

- The value returned changes as the contents of the FIFO change and is only valid while OPR bit in the OUTCSRL register is set.
- For endpoint 0 and 1 only OUTCNTRL has to be read (OUTCNTRM is reserved and returns 00h).

Bits 7:0 = **OC[7:0]** *OUT Count*.

10.5.6.3 FIFO register addressing

This address range provides 3 addresses for CPU access to the FIFOs for each endpoint. Writing to these addresses loads data into the IN FIFO for the corresponding endpoint. Reading from these addresses unloads data from the OUT FIFO for the corresponding endpoint.

Address	R/W	FIFO accessed
21h	R	Endpoint 0 OUT / SETUP
	W	Endpoint 0 IN
23h	R	Endpoint 1 OUT
	W	Endpoint 1 IN
25h	R	Endpoint 2 OUT
	W	Endpoint 2 IN

10.5.6.4 End of suspend detection register

This register controls a specific end of suspend block that is able to wake-up the ST7 when the clocks are stopped.

EOS STATUS REGISTER (EOSSR)

Read / Write

Reset Value: 0000 0000 (00h)

7							0
LS1	LS0	0	0	0	0	0	EOS

Bit 7 = **LS1** *Line State 1 flag*.
This bit is read only by software
0: LS1 is at 0 (D-=0)
1: LS1 is at 1 (D-=1)

Bit 6 = **LS0** *Line State 0 flag*.
This bit is read only by software
0: LS0 is at 0 (D+=0)
1: LS0 is at 1 (D+=1)

Bit 5:1 = Reserved.

Bit 0 = **EOS** *End Of Suspend*.
This bit is set by hardware and cleared by software.
0: No EOS interrupt occurred
1: EOS interrupt occurred

Note:

A parasitic EOS bit set can occur at device start up so it is mandatory to clear this bit before enabling the interrupt (by setting EOSE).

EOS CONTROL REGISTER (EOSCR)

Read / Write

Reset Value: 0000 0000 (00h)

7							0
LSE	0	0	0	0	UPO	CPO	EOSE

Bit 7 = **LSE** *Line State Enable*.
This bit is set and cleared by software.
0: Line State flag disabled
1: Line State flag enabled.

Bit 6:3 = Reserved.

Bit 2 = **UPO** *USB2 Phy Off.*
This bit is set and cleared by software. When the USB part is not used the PHY can be stopped completely to remove useless consumption.
0: USB2 PHY enabled
1: USB2 PHY disabled.

Bit 1 = **CPO** *Compensation Cell Off.*
This bit is set and cleared by software. Set this bit to decrease the consumption (for instance before entering in suspend / halt mode) by removing the IOs active slew rate control PVT compensation.

0: Compensation cell enabled
1: Compensation cell disabled.

Bit 0 = **EOSE** *End Of Suspend Enabled*.
This bit is set and cleared by software to enable/disable the End Of Suspend interrupt generation.
0: EOS interrupt disabled
1: EOS interrupt enabled

Note: before enabling EOS interrupt (setting EOSE bit), it is advised to clear the EOS bit of EOSSR register to avoid any parasitic interrupt.

10.5.7 Programming consideration

This section describes how to control the USB device.

10.5.7.1 Soft Connect/Disconnect

The UTMI interface between the PHY and the USB controller can be switched between normal mode and non-driving mode by setting/clearing bit 6 of the Power register (PWRR).

When the Soft Connect/Disconnect bit is set to 1, the PHY is placed in its normal mode and the D+/D- lines of the USB bus are enabled. At the same time, the USB controller is placed in ‘Powered’ state, in which it will not respond to any USB signalling except a USB reset.

10.5.8 USB reset

When a reset condition is detected on the USB, the USB controller performs the following actions:

- Sets FAddr to 0.
- Sets Index to 0.
- Flushes all endpoint FIFOs.
- Clears all control/status registers.
- Enables all interrupts, except Suspend.
- Generates a Reset interrupt.

When the software receives a Reset interrupt, it should close any open pipes and wait for bus enumeration to begin.

10.5.9 Suspend /Resume

When the USB device sees no activity on the USB for 3 ms it will generate a Suspend interrupt. It is up to the software to decide what to disable when the USB is in Suspend mode.

Note: To decrease the consumption, refer to the Halt mode description chapter, section 5.3 on page 27.

Software may perform “Remote Wake-up” by setting the Resume bit in the Power register (bit 2). The USB device will then send Resume signalling on the USB to wake up the hub.

The USB may exit Suspend mode by sending Resume signalling on the bus, this detection could be performed differently if the clocks are active or not in the device.

10.5.9.1 Remote wake-up

If the USB device is in Suspend mode and the software wants to initiate a remote wake-up, it should write to the Power register to set the Resume bit (bit 2) to 1. Of course if the clocks are stopped, it will need an external event (ST7 external interrupt) to restart the clocks.

Software should leave this bit set for approximately 10 ms (minimum of 2 ms, a maximum of 15 ms) then reset it to 0. By this time the hub should have taken over driving Resume signalling on the USB.

Note: No Resume interrupt is generated when software initiates a remote wake-up.

10.5.9.2 Clock active during suspend

If the Enable Suspend Mode bit in the Power register (bit 0) is set when a Suspend interrupt is generated, the UTM will be put into Suspend mode by the SUSPENDM line. The USB controller will, however, remain active and therefore can detect when Resume signalling occurs on the USB. It will then bring the UTM out of Suspend mode and generate a Resume interrupt.

10.5.9.3 Clock inactive during suspend

When a Suspend interrupt is received, software may put the device in Halt mode by executing the ST7 halt instruction. In this case a specific block "end of Suspend Block" will track any change on UTM linestate signals. If a linestate toggles the EOS block generates an USB End Of Suspend Interrupt to wake-up the ST7 from Halt.

After the clocks are restored the USB device will detect the wake-up event (Resume or Reset).

10.5.10 Endpoint 0 handling

Endpoint 0 is the main control endpoint of the core. As such, the routines required to service Endpoint 0 are more complicated than those required to service other endpoints.

The software is required to handle all the Standard Device Requests that may be received via Endpoint 0. These are described in the *Universal Serial Bus Specification*, Revision 2.0, Chapter 9. The protocol for these device requests involves different numbers and types of transaction per transfer. To accommodate this, application software needs to take a state machine approach to command decoding and handling.

The Standard Device Requests can be divided into three categories: *Zero Data Requests* (in which all the information is included in the command), *Write Requests* (in which the command will be followed by additional data), and *Read Requests* (in which the device is required to send data back to the host).

This section looks at the sequence of events that the software must perform to process the different types of device request.

10.5.10.1 Endpoint 0 Service Routine

An Endpoint 0 interrupt is generated:

- When the core sets the OPR bit (CSR0) after a valid token has been received and data has been written to the FIFO.
- When the core clears the IPR bit (CSR0) after the packet of data in the FIFO has been successfully transmitted to the host.
- When the core sets the STST bit (CSR0) after a control transaction is ended due to a protocol violation.
- When the core sets the SE bit (CSR0) because a control transfer has ended before DE (CSR0) is set.

Whenever the Endpoint 0 service routine is entered, the software must first check to see if the current control transfer has been ended due to either a STALL condition or a premature end of control transfer. If the control transfer ends due to a STALL condition, the STST bit is set. If the control transfer ends due to a premature end of control transfer, the SE bit is set. In either case, the software should abort processing the current control transfer and set the state to IDLE.

10.5.10.2 Error Handling

A control transfer may be aborted due to a protocol error on the USB, the host prematurely ending the transfer, or if the function controller software wishes to abort the transfer (e.g. because it cannot process the command).

The USB controller will automatically detect protocol errors and send a STALL packet to the host under the following conditions:

1. Host sends more data during the OUT Data phase of a write request than was specified in the command. This condition is detected when the host sends an OUT token after the DE bit (CSR0) has been set.

2. Host requests more data during the IN Data phase of a read request than was specified in the command. This condition is detected when the host sends an IN token after the DE bit in the CSR0 register has been set.

3. Host sends more than MaxP data bytes in an OUT data packet.

4. Host sends a non-zero length DATA1 packet during the STATUS phase of a read request.

When the USB controller has sent the STALL packet, it sets the STST bit (CSR0) and generates an interrupt. When the software receives an Endpoint 0 interrupt with the STST bit set, it should abort the current transfer, clear the SentStall bit, and return to the IDLE state.

If the host prematurely ends a transfer by entering the STATUS phase before all the data for the request has been transferred, or by sending a new SETUP packet before completing the current transfer, then the SE bit (CSR0) will be set and an Endpoint 0 interrupt generated. When the software receives an Endpoint 0 interrupt with the SE bit set, it should abort the current transfer, set the SSE bit (CSR0), and return to the IDLE state. If the OPR bit (CSR0) is set this indicates that the host has sent another SETUP packet and the software should then process this command.

10.5.11 Bulk IN Endpoint

A Bulk IN endpoint is used to transfer non-periodic data from the function controller to the host.

Three optional features are available for use with a Bulk IN endpoint:

- **Double packet buffering**

If the value written to the INMAXPR register is less than, or equal to, half the size of the FIFO allocated to the endpoint, double packet buffering will be automatically enabled. When enabled, up to two packets can be stored in the FIFO awaiting transmission to the host.

- **DMA**

If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint is able to accept another packet in its FIFO. This feature is used to allow transfer to the MSCI without ST7 intervention in order to allow high speed transfer to/from the USB controller.

- **AutoSet**

When the AutoSet (INCSRM) feature is enabled, the IPR bit (INCSRL.bit0) will be automatically set when a packet of INMAXPR bytes has been loaded into the FIFO. This is particularly useful when DMA is used to load the FIFO as it avoids the need for any processor intervention when loading individual packets during a large Bulk transfer.

10.5.12 Bulk OUT Endpoint

A Bulk OUT endpoint is used to transfer non-periodic data from the host to the function controller.

Three optional features are available for use with a Bulk OUT endpoint:

■ Double packet buffering

If the value written to the OUTMAXP register is less than, or equal to, half the size of the FIFO allocated to the endpoint, double packet buffering will be automatically enabled. When enabled, up to two packets can be stored in the FIFO.

■ DMA

If DMA is enabled for the endpoint, a DMA request will be generated whenever the endpoint has a packet in its FIFO. This feature can be used to allow an external DMA controller to unload packets from the FIFO without processor intervention.

■ AutoClear

When the AutoClear feature is enabled, the OPR bit (OutCSR.D0) will be automatically cleared when a packet of OUTMAXP bytes has been unloaded from the FIFO. This is particularly useful when DMA is used to unload the FIFO as it avoids the need for any processor intervention when unloading individual packets during a large Bulk transfer.

10.5.13 Interrupt IN Endpoint

An Interrupt IN endpoint is used to transfer periodic data from the function controller to the host.

An Interrupt IN endpoint uses the same protocol as a Bulk IN endpoint and can be used the same way. However, though DMA can be used, it offers little benefit as Interrupt endpoints are usually expected to transfer all their data in a single packet.

Interrupt IN endpoints also support one feature that Bulk IN endpoints do not, in that they support continuous toggling of the data toggle bit. This feature is enabled by setting the FDT bit in the INCSRM register. When this bit is set to 1, the USB controller will consider the packet as having been successfully sent and toggle the data bit for the endpoint, regardless of whether an ACK was received from the host.

10.5.14 Interrupt OUT endpoint

An Interrupt OUT endpoint is used to transfer periodic data from the host to a function controller.

An Interrupt OUT endpoint uses almost the same protocol as a Bulk OUT endpoint and can be used the same way. The one difference is that Interrupt endpoints do not support PING flow control. This means that the USB controller should never respond with a NYET handshake, only ACK/NAK/STALL. To ensure this, the DNY bit in the OUTCSRM register should be set to 1 to disable the transmission of NYET handshakes in High-speed mode.

Though DMA can be used with an Interrupt OUT endpoint, it generally offers little benefit as Interrupt endpoints are usually expected to transfer all their data in a single packet.

10.5.15 Low Power modes

Mode	Description
WAIT	No effect on USB. USB interrupt events cause the device to exit from WAIT mode.
HALT	USB registers are frozen. In halt mode, the USB is inactive. USB operations resume when the MCU is woken up by an interrupt with “exit from halt capability” or by an event on the USB line in case of suspend. This event will generate an EOS interrupt which will wake-up from halt mode.

10.5.16 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Exit From Wait	Exit From Halt
Start Of Frame	SOF	SOFE	Yes	No
USB Reset	RST	RSTE	Yes	No
Resume	RSM	RSME	Yes	No
Suspend	SUSP	SUSPE	Yes	No
EP0	EP0	EP0E	Yes	No
EP1 IN	EP1I	EP1IE	Yes	No
EP2 IN	EP2I	EP2IE	Yes	No
EP1 OUT	EP1O	EP1OE	Yes	No
EP2 OUT	EP2O	EP2OE	Yes	No
USB End Of Suspend	EOS	EOSE	Yes	Yes

Note: The USB End Of Suspend interrupt event is connected to a single interrupt vector (EOS) with exit from halt capability (wake-up). The other interrupt events are connected to another interrupt vector: USB interrupt (USB).

They generate an interrupt if the corresponding enable control bit is set and the interrupt mask bits (I0, I1) in CC register are reset (RIM instruction).

Table 29. USB Register Map and Reset Values

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
0000h	PWRR Reset Value	0	SCON 0	HSE 1	HSM 0	RST 0	RSM 0	SUSM 0	ESUSM 0
0001h	FADDR Reset Value	UPD 0	FAD6 0	FAD5 0	FAD4 0	FAD3 0	FAD2 0	FAD1 0	FAD0 0
0003h	ITINR Reset Value	0	0	0	0	0	EP2I 0	EP1I 0	EP0 0
0005h	ITOUTR Reset Value	0	0	0	0	0	EP2O 0	EP1O 0	0
0007h	ITINER Reset Value	0	0	0	0	0	EP2IE 1	EP1IE 1	EP0E 1
0009h	ITOUTER Reset Value	0	0	0	0	0	EP2OE 1	EP1OE 1	0
000Ah	ITUSBER Reset Value	0	0	0	0	SOFE 0	RSTE 1	RSME 1	SUSPE 0
000Bh	ITUSBR Reset Value	0	0	0	0	SOF 0	RST 0	RSM 0	SUSP 0
000Ch	FRNBRM Reset Value	0	0	0	0	0	FN10 0	FN9 0	FN8 0
000Dh	FRNBRL Reset Value	FN7 0	FN6 0	FN5 0	FN4 0	FN3 0	FN2 0	FN1 0	FN0 0
000Eh	TSTMODER Reset Value	0	0	FFS 0	FHS 0	TPAK 0	TK 0	TJ 0	TSE0N 0
000Fh	INDEXR Reset Value	0	0	0	0	0	0	IND1 0	IND0 0
0010h	INMAXPRM Reset Value	0	0	0	0	0	IMP10 0	IMP9 0	IMP8 0
0011h	INMAXPRL Reset Value	IMP7 0	IMP6 0	IMP5 0	IMP4 0	IMP3 0	IMP2 0	IMP1 0	IMP0 0
0012h	INCSRM IND=0 Reset Value	0	0	0	0	0	0	0	0
	INCSRM IND=1 or 2 Reset Value	ASET 0	0	0	DMAE 0	FDT 0	0	0	0
0013h	INCSRL (CSR0) IND=0 Reset Value	SSE 0	SOPR 0	SDST 0	SE 0	DE 0	STST 0	IPR 0	OPR 0
	INCSRL IND=1 or 2 Reset Value	0	CDT 0	STST 0	SDST 0	FLFI 0	UNDR 0	FINE 0	IPR
0014h	OUTMAXPRM Reset Value	0	0	0	0	0	OMP10 0	OMP9 0	OMP8 0

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
0015h	OUTMAXPRL Reset Value	OMP7 0	OMP6 0	OMP5 0	OMP4 0	OMP3 0	OMP2 0	OMP1 0	OMP0 0
0016h	OUTCSRM Reset Value	ACLR 0	0	DMAE 0	DNY 0	DMAM 0	0	0	0
0017h	OUTCSRL Reset Value	CDT 0	STST 0	SDST 0	FLFI 0	0	0	FIFU 0	OPR 0
0018h	OUTCNTRM Reset Value	0	0	0	OC12 0	OC11 0	OC10 0	OC9 0	OC8 0
0019h	OUTCNTRL Reset Value	OC7 0	OC6 0	OC5 0	OC4 0	OC3 0	OC2 0	OC1 0	OC0 0
005Fh	EOSSR Reset Value	LS1 0	LS0 0	0	0	0	0	0	EOS 0
0060h	EOSCR Reset Value	LSE 0	0	0	0	0	0	0	EOSE 0

Notes:

1. The USB registers can also be accessed by the MSCI through the VCI interface (see section 16 on page 126).

10.5.17 IMPORTANT NOTES

EP1/EP2 configuration

With the ST7267, it is not possible to deactivate EP1 or EP2. If not used, it must be configured in STALL condition.

11 MASS STORAGE COMMUNICATION INTERFACE (MSCI)

11.1 INTRODUCTION

The MSCI is a complete system designed to handle various communication protocols.

The MSCI is built around a 16-bit RISC core capable of addressing up to 4K bytes of program memory, up to 8K bytes of data memory and a maximum of 32 internal registers

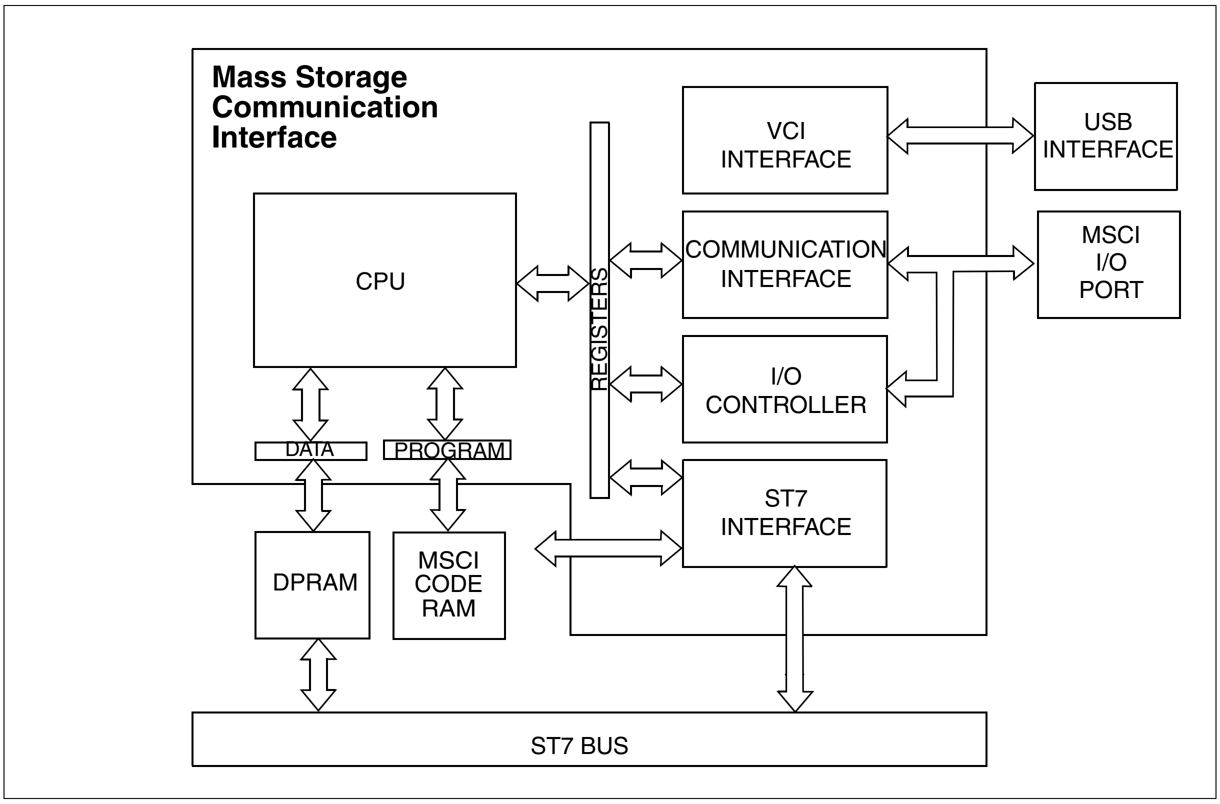
The MSCI system provides three communication interfaces:

- VCI
- Generic Parallel Interface
- Standard I/O Controller

All the MSCI system is controlled by the ST7 through a dedicated interface which manages:

- program memory upload
- MSCI soft reset
- MSCI core PC forcing

Figure 45. MSCI Block Diagram



12 MSCI REGISTER & MEMORY MAP

As shown in Figure 46, the MSCI is able to address 1KWords of program memory, 2.5KWords of data memory and 32 registers.

.

The detail of the MSCI register map is given in Table 30.

For a description of the ST7 memory map refer to section 3 on page 17

Figure 46. MSCI Memory Map

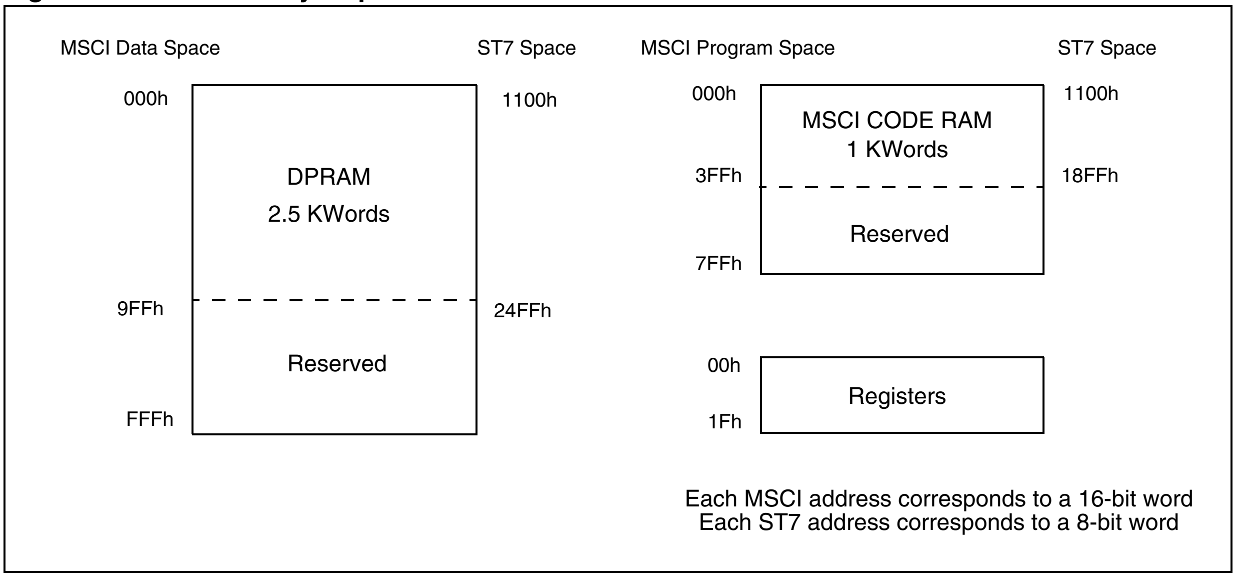


Table 30. MSCI Hardware Register Map

Address	Block	Register Label	Register Name	Reset Status	Remarks
00h 01h 02h 03h	MSCI CPU	R0 R1 R2 R3	General Purpose CPU Register 0 General Purpose CPU Register 1 General Purpose CPU Register 2 General Purpose CPU Register 3	0000h 0000h 0000h 0000h	R/W R/W R/W R/W
04h 05h 06h 07h	CPU DP	DP0 DP1 DP2 DP3	Data memory Pointer 0 Data memory Pointer 1 Data memory Pointer 2 Data memory Pointer 3	000h 000h 000h 000h	R/W R/W R/W R/W
08h 09h 0Ah	Port 1	P1DRO P1DRI P1DDR	Port 1 Data Register Output Port 1 Data Register Input Port 1 Data Direction Register	0000h 0000h 0000h	R/W R R/W
0Bh 0Ch 0Dh	Port 2	P2DRO P2DRI P2DDR	Port 2 Data Register Output Port 2 Data Register Input Port 2 Data Direction Register	0000h 0000h 0000h	R/W R R/W
0Eh 0Fh 10h 11h	VCI	VCR VSR VFDR VTAR	VCI Control Register VCI Status Register VCI FIFO Data Register VCI Target Address Register	02F0h 0001h 0000h 0000h	R/W R/W R/W R/W

Address	Block	Register Label	Register Name	Reset Status	Remarks
12h	Parallel Interface	PNDR	Parallel interface Number of Data Register	0000h	R/W
13h		PFDR	Parallel interface FIFO Data Register	0000h	R/W
14h		PCR1	Parallel interface Control Register 1	007Fh	R/W
15h		PCR2	Parallel interface Control Register 2	0000h	R/W
16h		PSR	Parallel interface Status Register	0009h	R
17h		ELP1	ECC Line Parity 1	FFFFh	R
18h		ECP1	ECC Column Parity 1	00FFh	R
19h		ELP2	ECC Line Parity 2	FFFFh	R
1Ah		ECP2	ECC Column Parity 2	00FFh	R
1Bh		RCSR	Reed Solomon control status register	4000h	R/W
1Ch		RDFR	Reed Solomon decoder FIFO register	0000h	R/W
1Dh		REFR	Reed Solomon Encoder FIFO register	0000h	R
1Eh 1Fh			Reserved	0000h 0000h	R R

13 MSCI CENTRAL PROCESSING UNIT

13.1 Introduction

The CPU has a full 16-bit architecture. Ten internal registers allow efficient 16-bit data manipulation. The CPU is able to execute 39 basic instructions. It features 6 main addressing modes and can address 8 internal registers.

13.2 Main Features

- Enable executing 39 basic instructions
- 6 main addressing modes
- Four 16-bit general purpose registers
- Four 12-bit data pointers
- One 11-bit program counter
- One 5-bit status register

13.3 CPU Registers

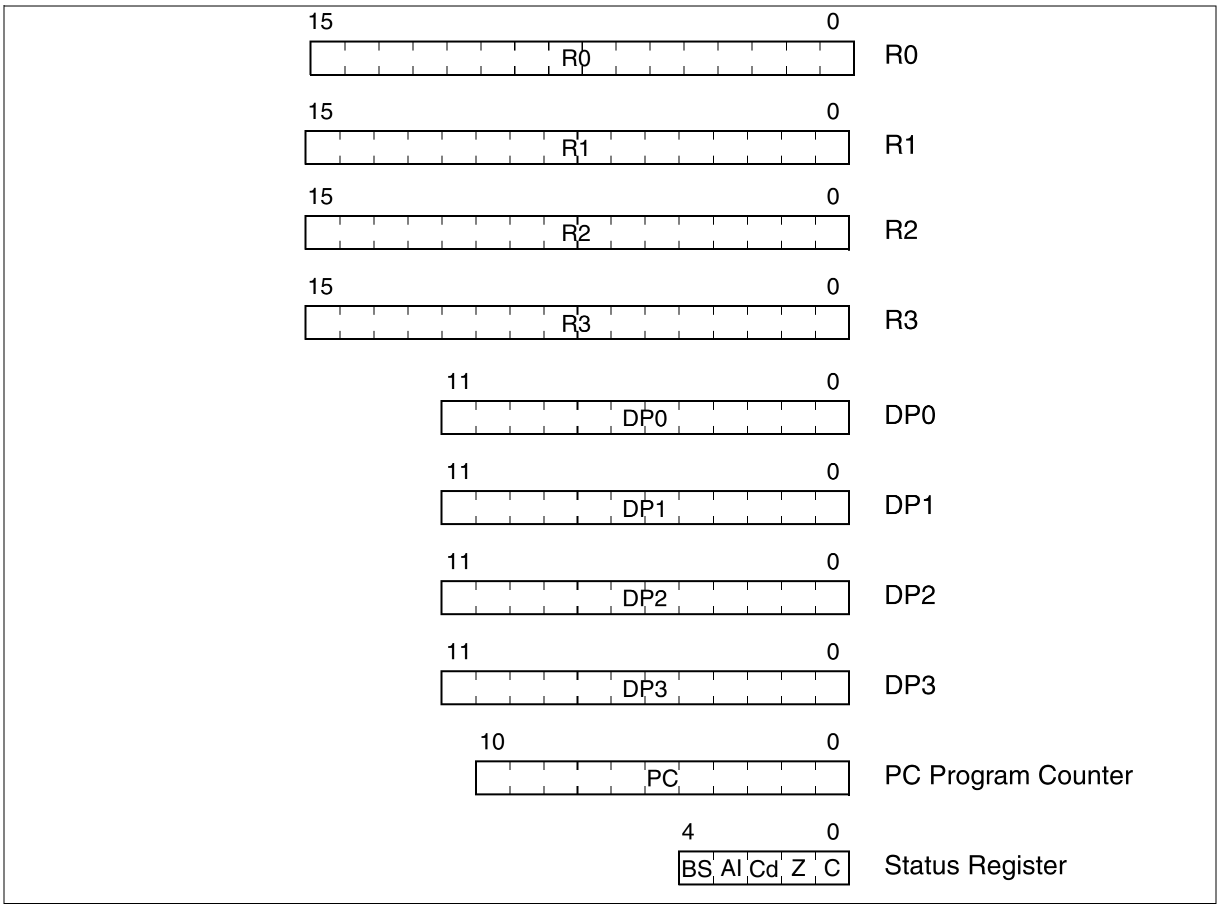
The 10 CPU registers are shown in the programming model in Figure 47. Following a CALL instruction, R0, R1, DP0, PC and Status are saved. They are restored following a RET instruction.

GP Registers(R0, R1, R2, R3). These 16-bit general purpose registers are used to hold operands and the results of the arithmetic and logic calculations as well as data manipulations.

Data Pointer Registers (DP0, DP1, DP2, DP3). These 12-bit registers are used to handle data memory addressing. As the result, the MSCI core can access up to 8Kbyte of data memory.

Program Counter (PC). The program counter is a 11-bit register used to store the address of the next instruction to be executed by the CPU. It is automatically refreshed after each processed instruction. As a result, the MSCI core can access up to 4Kbyte of program memory.

Figure 47. MSCI CPU Registers



MSCI CENTRAL PROCESSING UNIT (Cont'd)

STATUS REGISTER (STATUS)
Read / Write
Reset Value: 0000 0000 0001 1000 (0018h)

15															0
-	-	-	-	-	-	-	-	-	-	-	BS	AI	Cond	Z	C

The 5-bit Status register contains:
- two configuration flags handling special Wait on Bit State (WBS) instruction, and post-incremented/ decremented Data RAM Load management.
- three status flags Condition, Zero and Carry.

Bit 15:5 = *Reserved*.

Configuration flags

Bit 4 = **BS** *Bit Set*.
When set to 1, this bit indicates that the Wait Bit State (WBS) instruction will wait until the selected bit is set. Otherwise, it will wait until the selected bit is reset.
This bit is set and cleared by software using WO-Set and WORst instructions.
0: WBS instruction will wait until the polled bit is reset
1: WBS instruction will wait until the polled bit is set

Bit 3 = **AI** *Auto Increment*.
When set to 1, this bit indicates that the CPU is in post-increment mode. If reset, the CPU is in post-decrement mode (See LD instructions in the MSCI CPU Programming Manual).
This bit is set and cleared by software using AutoINC and AutoDEC instructions.
0: Auto-decrement mode
1: Auto-Increment mode

Status flags

Bit 2 = **Cond** *Condition*.
When set to 1, this bit indicates that the result of the last comparison is true.
This bit is set and cleared by hardware or software using SECond and CLCond instructions.
0: Last comparison result is false
1: Last comparison result is true

Bit 1 = **Z** *Zero*.
When set to 1, this bit indicates that the result of the last ALU operation is zero.
This bit can be set and cleared by hardware or software using SEZ and CLZ instructions.
0: Last ALU operation result is not zero
1: Last ALU operation result is zero

Bit 0 = **C** *Carry*.
When set to 1, this bit indicates that a carry borrow out of the ALU occurred during the last arithmetic operation. This bit is also affected during, shift instructions. See ADD, ADDC, SUB, SUBC instructions.
This bit can be set and cleared by hardware or software using SEC and CLC instructions.
0: No carry
1: Carry borrow out

14 MSCI ST7 INTERFACE

14.1 Introduction

The MSCI ST7 interface provides three main system control functions:

- Controlling the MSCI from the ST7
- MSCI Interrupt Requests to ST7
- MSCI Code RAM Upload

The MSCI ST7 Interface block diagram is shown in Figure 48.

14.2 Functional Description

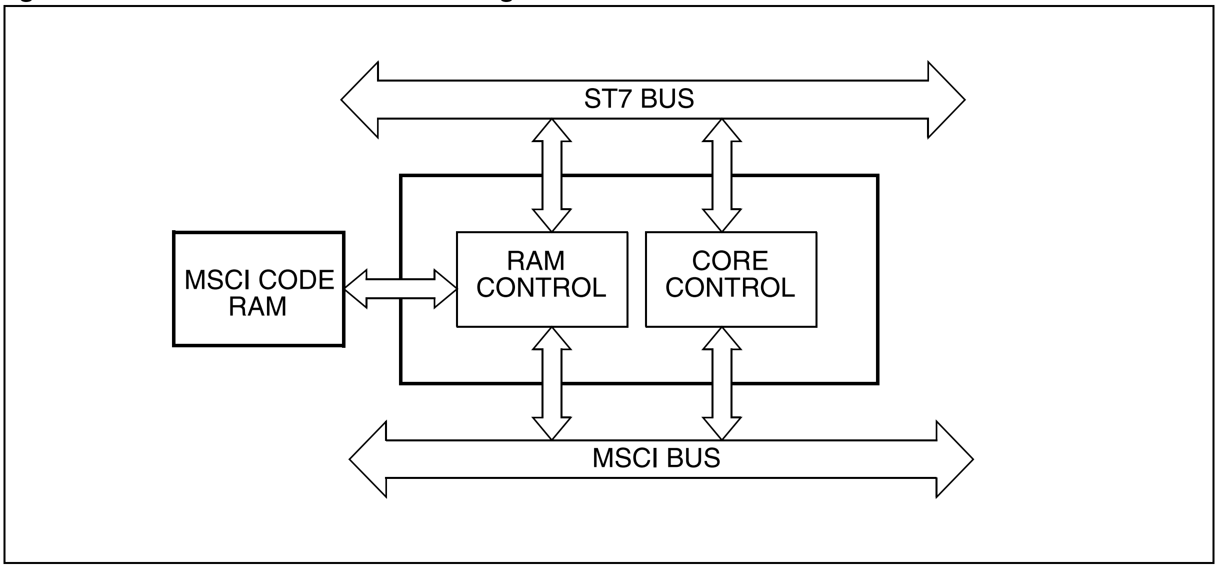
The MSCI ST7 Interface provides three main functions between ST7 core and MSCI system.

14.2.1 ST7 Control of the MSCI

The ST7 can control the MSCI through the MCR (MSCI Control Register). It allows the ST7 to:

- Reset the whole MSCI system by setting the SFTR bit.
- Force the MSCI core program counter to the specified values in the MPCL and MPCM (MSCI Program Counter LSB and MSCI Program Counter MSB) registers by setting the PCR bit.
- Continue the program executed by the MSCI after it has been self-stopped with an internal STOP instruction by writing a'1' in the GO bit.

Figure 48. MSCI ST7 Interface Block Diagram



MSCI ST7 INTERFACE (Cont'd)

14.2.2 Interrupt generation from MSCI to ST7

Each time the MSCI core executes a STOP instruction it stops itself and the STOP bit in the MSR (MSCI Status Register) is set.

An interrupt is generated if the STPIE bit in the MCR (MSCI Control Register) is set.

This feature allows the MSCI core to interrupt the ST7 each time it completes a function ended by a STOP instruction.

The ITR MSCI CPU instruction sets the ITR bit in the MSR register without stopping the MSCI cpu.

An interrupt is generated if the STPIE bit in the MCR (MSCI Control Register) is set.

14.2.3 Program RAM upload

The MSCI ST7 interface provides read and write access to the program memory of the MSCI core. The RAM is mapped in the memory array of the ST7 core like any other memory but access to this memory is protected depending on the state of the MSCI system.

To access the memory with the ST7 the MSCI system must be in PC or Soft reset, i.e the PCR or SFTR bits of the MCR (MSCI Control Register) are set.

In this state, the RAMLD bit of the MCR (MSCI Control Register) can be set to switch the Data RAM access from the MSCI to the ST7.

When the RAMLD bit in the MCR (MSCI Control Register) is reset, the ST7 cannot access the MSCI program memory:

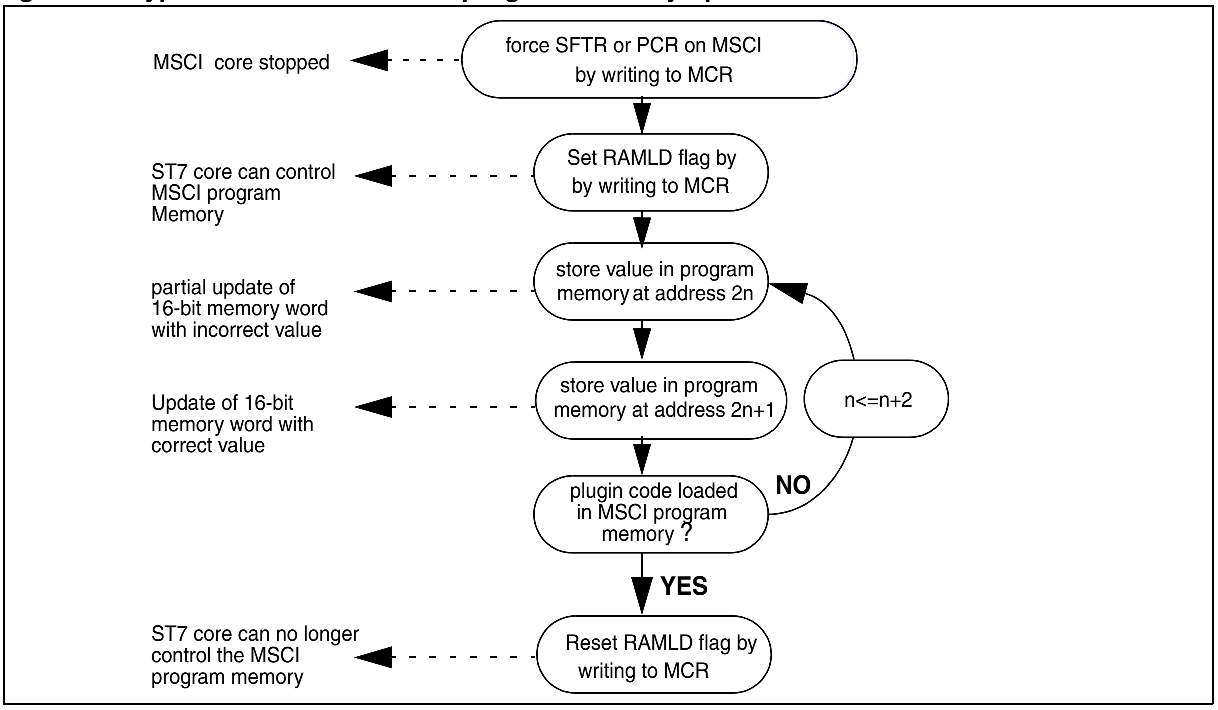
– a write access by the ST7 to the MSCI program memory has no effect.

– a read access by the ST7 to the MSCI program memory returns $00 value.

When the RAMLD bit in the MCR (MSCI Control Register) is set, the ST7 can read or write from/ into this memory. Reading can be performed randomly at any address of the RAM.

Writing into this memory can only be performed by writing pairs of bytes starting with the even address byte followed by the next higher byte.

Figure 49. Typical ST7 flow for MSCI program memory update


MSCI core stopped
force SFTR or PCR on MSCI by writing to MCR
ST7 core can control MSCI program Memory
Set RAMLD flag by by writing to MCR
partial update of 16-bit memory word with incorrect value
store value in program memory at address 2n
Update of 16-bit memory word with correct value
store value in program memory at address 2n+1
n<=n+2
plugin code loaded in MSCI program memory ?
NO
YES
ST7 core can no longer control the MSCI program memory
Reset RAMLD flag by writing to MCR

MSCI ST7 INTERFACE (Cont'd)

14.2.4 ST7 Write Access to MSCI Code RAM

The ST7 can only write in MSCI RAM when the RAMLD bit of the MCR (MSCI Control Register) is set.
To set this flag, the MSCI core must be stopped either by a PC reset or a soft reset.

Due to 8/16-Bit conversion, writing in the MSCI program memory must be done in a specific sequence:

- At each even access the 8-bit value is stored in a temporary register (but the 16-bit RAM word is updated with an incorrect temporary value).
- At each odd access the 16-bit value (made of the 8-bit value stored previously in the temporary register for the LSB and of the 8-bit of the even access for the MSB) is written in the memory.

For this reason an odd number of bytes cannot be written in this memory and the bytes must always be written in ascending order (from a low address to higher addresses)

14.2.4.1 ST7 Read Access to MSCI Code RAM

Reading from the ram is possible only when the RAMLD bit of the MCR (MSCI Control Register) is set.
To set this flag, the MSCI core must be stopped either by a PC reset or a soft reset.

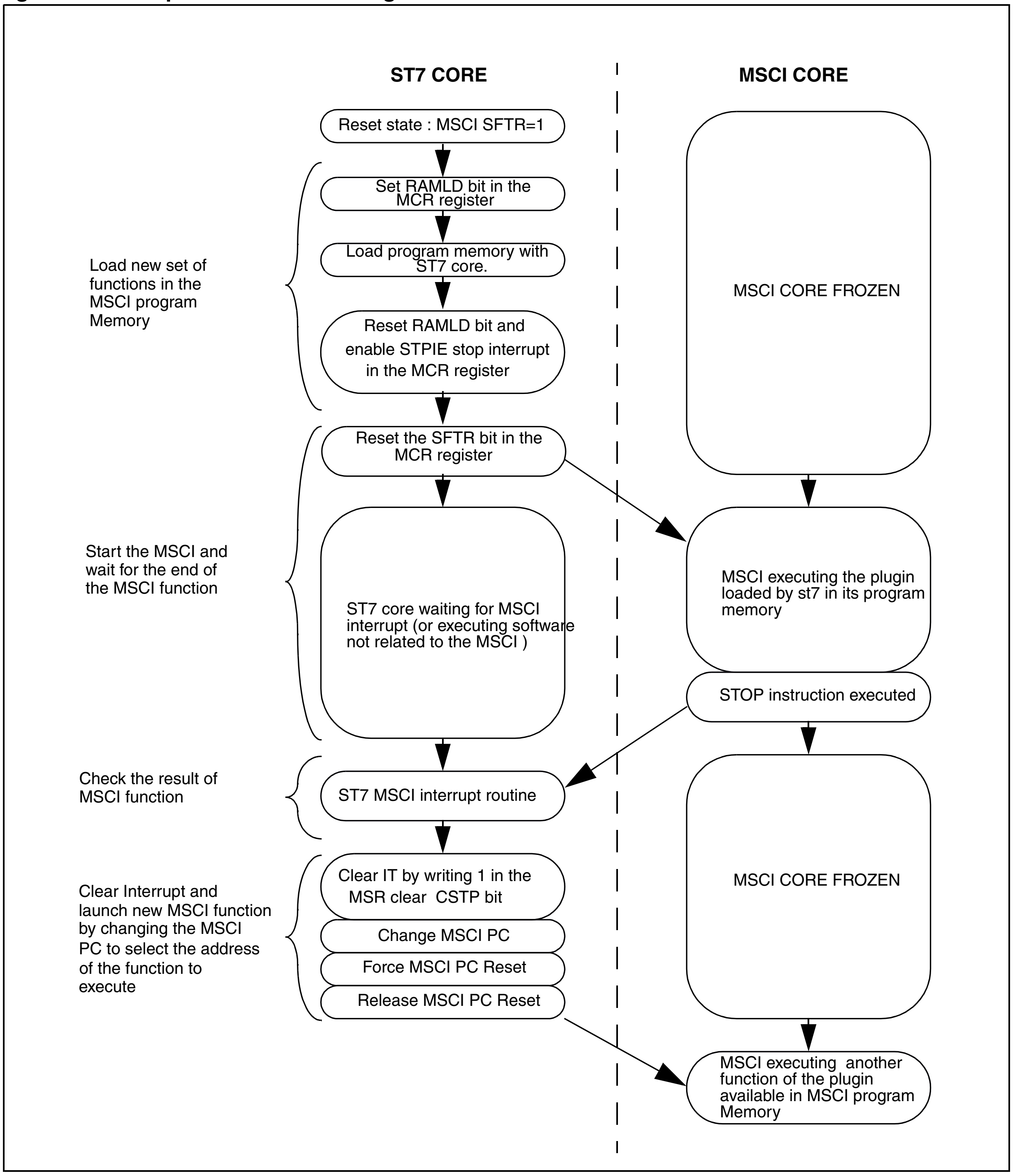
Read access by the st7 core can be performed randomly at any address of the RAM.

MSCI ST7 INTERFACE (Cont'd)

14.2.5 Example Control Flow

Figure 50 gives an example of an application flow showing the ST7 controlling the MSCI.

Figure 50. Example of ST7 Controlling MSCI



ST7 CORE
MSCI CORE
Reset state : MSCI SFTR=1
Set RAMLD bit in the MCR register
Load program memory with ST7 core.
Reset RAMLD bit and enable STPIE stop interrupt in the MCR register
Load new set of functions in the MSCI program Memory
Reset the SFTR bit in the MCR register
ST7 core waiting for MSCI interrupt (or executing software not related to the MSCI)
Start the MSCI and wait for the end of the MSCI function
ST7 MSCI interrupt routine
Check the result of MSCI function
Clear IT by writing 1 in the MSR clear CSTP bit
Change MSCI PC
Force MSCI PC Reset
Release MSCI PC Reset
Clear Interrupt and launch new MSCI function by changing the MSCI PC to select the address of the function to execute
MSCI CORE FROZEN
MSCI executing the plugin loaded by st7 in its program memory
STOP instruction executed
MSCI CORE FROZEN
MSCI executing another function of the plugin available in MSCI program Memory

MSCI ST7 INTERFACE (Cont'd)

14.2.6 ST7 Register Description

ALL THESE REGISTERS ARE IN THE ST7 ME-MEMORY MAP AND CANNOT BE ACCESSED BY THE MSCI CORE

MSCI CONTROL REGISTER (MCR)
Read / Write
Reset Value: 0000 0001 (01h)

7							0
-	-	ITRIE	STPIE	GO	RAMLD	PCR	SFTR

Bit 7:6 = Reserved

Bit 5 = **ITRIE** *ITR Interrupt Enable*.
This bit is set and cleared by ST7 software
0: ITR interrupt disabled
1: ITR interrupt enabled

Bit 4 = **STPIE** *Stop Interrupt Enable*.
This bit is set and cleared by ST7 software
0: STOP interrupt disabled
1: STOP interrupt enabled

Bit 3 = **GO** *Go*.
This bit is set by ST7 software and cleared by hardware. It is always read as '0'. It generates a pulse to launch the MSCI after it has been self-stopped by an internal STOP instruction.
0: No effect
1: Generate a starting pulse

Bit 2 = **RAMLD** *RAM Load*.
This bit is set and cleared by ST7 software. The MSCI must be under PC or Soft reset before starting a read or write sequence. It can be written only when PCR or SFTR is set (or when MSCI is stopped by emulator in emulation mode)
See Section 14.2.4 and Section 14.2.4.1
0: RAM access from ST7 disabled
1: RAM access from ST7 enabled

Bit 1 = **PCR** *Program Counter Reset*.
This bit is set and cleared by ST7 software. It can be written only when RAMLD is cleared.
0: MSCI program counter reset not forced
1: MSCI program counter forced to MPCM & MPCL value.

Bit 0 = **SFTR** *Soft Reset*.
This bit is set and cleared by ST7 software. It can be written only when RAMLD is cleared.
0: MSCI system reset not forced
1: MSCI system under reset.

MSCI STATUS REGISTER (MSR)
Read / Write
Reset Value: 0000 0000 (00h)

7							0
-	-	ITR	STP	-	-	CITR	CSTP

Bit 7:6 = Reserved

Bit 5 = **ITR** *MSCI CORE ITR flag*.
This bit is set by MSCI software and cleared by a PC reset or a soft reset or by writing a '1' in the CITR bit in ST7 software.
0: MSCI ITR flag not set
1: MSCI ITR flag set by an internal ITR instruction.

Bit 4 = **STP** *MSCI CORE Stop* flag.
This bit is set by MSCI software and cleared by a PC reset or a soft reset or by writing a '1' in the CSTP bit in ST7 software
0: MSCI stop flag not set
1: MSCI stop flag set by an internal STOP instruction

Bit 3:2 = Reserved

Bit 1 = **CITR** *Clear ITR flag*.
This bit is set by ST7 software to clear the MSCI ITR flag and interrupt and reset by hardware. It is always read as '0'.
0: No effect
1: Clears MSCI ITR flag and interrupt if pending

Bit 0 = **CSTP** *Clear Stop flag*.
This bit is set by ST7 software to clear the MSCI STOP flag and interrupt and reset by hardware. It is always read as '0'.
0: No effect
1: Clears MSCI STOP flag and interrupt if pending

Note:

To set the RAMLD bit when PCR=0 and SFTR=0, two write accesses to MCR register are needed. The first access must set either the SFTR bit or the PCR bit to enable write access to RAMLD bit. The second write access can set the RAMLD bit. Except in emulation mode when MSCI is stopped by emulator.

To clear the PCR bit or the SFTR bit when RAMLD bit is set, two write accesses to MCR register are needed. The first access to reset the RAMLD bit. The second write access to clear PCR bit or SFTR bit. This must be done also in emulation mode.

MSCI ST7 INTERFACE (Cont'd)

MSCI PROGRAM COUNTER MSB (MPCM)
Read / Write
Reset Value: 0000 0000 (00h)

7							0
-	-	-	-	-	PC10	PC9	PC8

Bit 7:3 = Reserved

Bit 2:0 = **PC[10:8]** *Program Counter (MSB).*
MSCI core Program Counter MSB. These bits are set and cleared by ST7 software and are loaded in the MSCI core program counter at each PC reset or Soft reset.

MSCI PROGRAM COUNTER LSB (MPCL)
Read / Write
Reset Value: 0000 0000 (00h)

7							0
PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

Bit 7:0 = **PC[7:0]** *Program Counter (LSB).*
MSCI core Program Counter LSB. These bits are set and cleared by ST7 software and are loaded in the MSCI core program counter at each PC reset or Soft reset.

MSCI CRC MSB (MCRCM)
Read
Reset Value: 0000 0000 (00h)

7							0
CRC15	CRC14	CRC13	CRC12	CRC11	CRC10	CRC9	CRC8

Bit 7:0 = **CRC[15:8]** *CRC (MSB).*
MSCI cyclic redundancy code. This code provides a signature of the MSCI code execution (read only). It is generated by MSCI core. It is reset by SOFT Reset.

MSCI CRC MSB (MCRCL)
Read
Reset Value: 0000 0000 (00h)

7							0
CRC7	CRC6	CRC5	CRC4	CRC3	CRC2	CRC1	CRC0

Bit 7:0 = **CRC[7:0]** *CRC (LSB).*
MSCI cyclic redundancy code. This code provides a signature of the MSCI code execution (read only). It is generated by MSCI core. It is reset by SOFT Reset.

MSCI ST7 INTERFACE (Cont'd)

14.2.7 Low Power modes

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

14.2.8 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Clear Interrupt Bit	Exit From Wait	Exit From Halt
MSCI STOP	STP	STPIE	CITR	Yes	No
MSCI ITR	ITR	ITRIE	CSTP	Yes	No

Note: Both stop and ITR interrupts are connected to the same interrupt vector. They generate an interrupt if the corresponding enable control bit is set and the interrupt mask bits (I0, I1) in CC register are reset (RIM instruction)

Table 31. MSCI ST7 Interface User Register Map and Reset Values

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
0069h	MCR Reset Value	0	0	ITRIE 0	STPIE 0	GO 0	RAMLD 0	PCR 0	SFTR 1
006Ah	MSR Reset Value	0	0	ITR 0	STP 0	0	0	CITR 0	CSTP 0
006Bh	MPCM Reset Value	0	0	0	0	0	PC10 0	PC9 0	PC8 0
006Ch	MPCL Reset Value	PC7 0	PC6 0	PC5 0	PC4 0	PC3 0	PC2 0	PC1 0	PC0 0
006Ch	MCRCH Reset Value	CRC15 0	CRC14 0	CRC13 0	CRC12 0	CRC11 0	CRC10 0	CRC9 0	CRC8 0
006Dh	MCRCL Reset Value	CRC7 0	CRC6 0	CRC5 0	CRC4 0	CRC3 0	CRC2 0	CRC1 0	CRC0 0

15 MSCI I/O CONTROLLER

15.1 Introduction

The MSCI I/O ports can be configured in different functional modes:
- data transfer through digital inputs and outputs

and for specific pins:
- alternate input/output signals for the parallel interface.

Each of the two I/O ports contains 16 pins. Each pin can be programmed independently as digital input or digital output.

15.2 Functional Description

Each port has 3 main registers:

- Data Register Out (DRO)
- Data Register Input (DRI)
- Data Direction Register (DDR)

Each I/O pin may be programmed using the corresponding register bits in the DDR register: bit X corresponding to pin X of the port. The same correspondence is used for the DRO and DRI registers.
The DRO and DDR registers can be read and written by the MSCI core. The DRI register is an image corresponding to the logic level on the I/OO pin and can be read by the MSCI core (read only register).

The MSCI I/O control block diagram is shown in Figure 51.

15.2.1 Input mode

The input configuration is selected by clearing the corresponding DDR register bit. In this case, reading the DRI register returns the digital value applied to the external I/O pin.

In this mode writing the DRO register has no effect on the pad. However the values are written in the DRO register and if the DDR register is set to output mode, the value on the port will be the value written previously in the DRO register.

15.2.2 Output mode

The output configuration is selected by setting the corresponding DDR register bit. In this case, writing the DRO register applies this digital value to the I/O pin through the latch.

In this mode reading the DRI register returns the digital value applied to the external I/O pin.

15.2.3 Alternate functions

When an on-chip peripheral is configured to use a pin in output mode, the alternate function is automatically selected. This alternate function takes priority over the standard I/O programming.

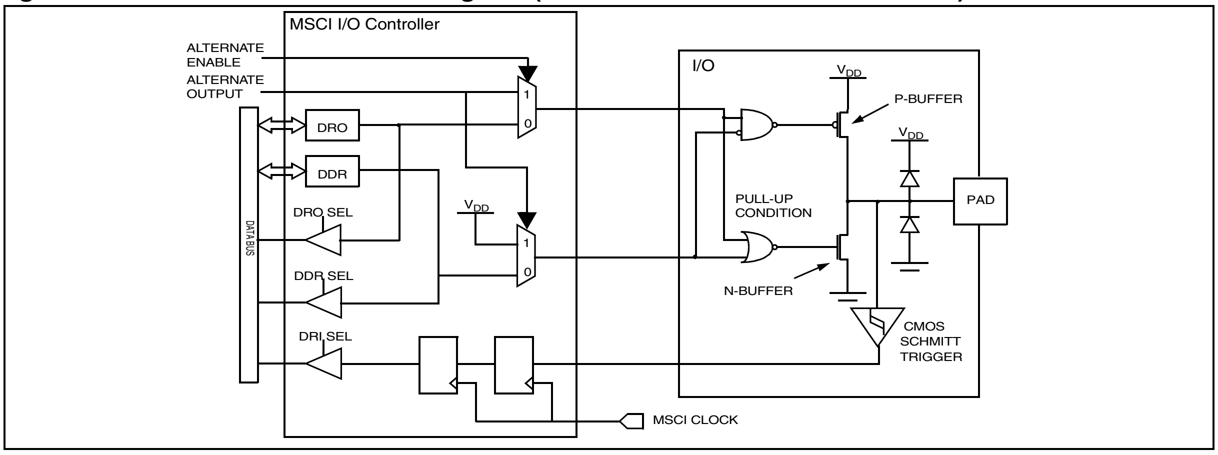
When the signal is coming from the parallel interface, the I/O pin is automatically configured in output mode when needed. There are two cases depending of the type of signal:

- Control signals are configured using the CS bit in the PCR2 register.

- Data signals are configured when the DIR bit in the PCR1 register is set in output mode and data transmission is on going.

When the signal is going to the parallel interface (input mode), the I/O pin has to be configured in input mode by the standard I/O programming to avoid conflicts.

Figure 51. I/O Port General Block Diagram (When I/Os are dedicated to MSCI)



MSCI I/O PORTS (Cont'd)

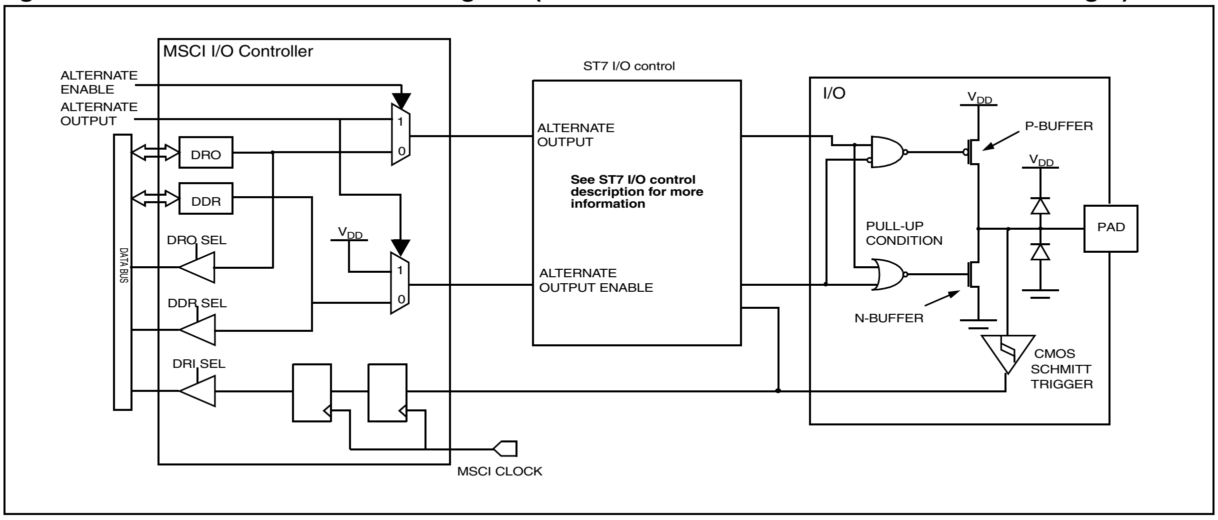
15.3 I/O sharing between ST7 and MSCI

If MSCI I/Os are shared between the MSCI I/O controller and the ST7 I/O controller, the MSCI outputs are connected to alternate function of the ST7 I/O logic and the MSCI output enable signals are connected to the alternate enable inputs of the ST7 I/O logic.

When ST7 configures a shared I/O in Input floating mode, the MSCI I/O logic can directly control this I/O.

When ST7 configures an I/O in output mode, the MSCI can control this port in output mode by setting the corresponding DDR bit because it has the priority on I/O control. Consequently, the value on the port is forced by the corresponding bit of the MSCI DRO register (of port 1 or port 2). However the MSCI can't force this I/O to be in input mode.

Figure 52. I/O Port General Block Diagram (when MSCI I/Os are shared with ST7I/ O Logic)



MSCI DDR	ST7 DDR	EFFECT ON PORT
0	0	Port in input mode. Can be read by both MSCI through DRI register and ST7 through DR register.
0	1	Port in output mode, value forced by ST7 DR register.
1	0	Port in output mode, value forced by MSCI DRO register.
1	1	Port in output mode, value forced by MSCI DRO register.

MSCI I/O PORTS (Cont'd)

15.3.1 Register Description

DATA REGISTER OUTPUT (DRO)
Read / Write
Reset Value: 0000 0000 0000 0000 (0000h)

15							8	7							0
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Bit 15:0 = **D[15:0]** *Data Output*.
Data output on the I/Os when they are configured in output mode.

DATA REGISTER INPUT (DRI)
Read
Reset Value: xxxx xxxx xxxx xxxx (xxxxh)

The reset value depends on the value forced externally on the pins)

15							8	7							0
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Bit 15:0 = **D[15:0]** *Data Input*.
Input of the I/Os regardless of whether they are configured in input or output mode. The value read is not directly the value on the PAD but the value sampled twice to avoid metastability problems. For this reason, the value read from the DRI register is the value that was present on the PAD 2 MSCI clock cycles before the current time.

DATA DIRECTION REGISTER OUTPUT (DDR)
Read / Write
Reset Value: 0000 0000 0000 0000 (0000h)

15							8	7							0
Dir15	Dir14	Dir13	Dir12	Dir11	Dir10	Dir9	Dir8	Dir7	Dir6	Dir5	Dir4	Dir3	Dir2	Dir1	Dir0

Bit 15:0 = **Dir[15:0]** *Direction*.
Define the I/O configuration: input mode or output mode.
0: Input mode
1: Output mode

16 MSCI VCI INTERFACE

16.1 Introduction

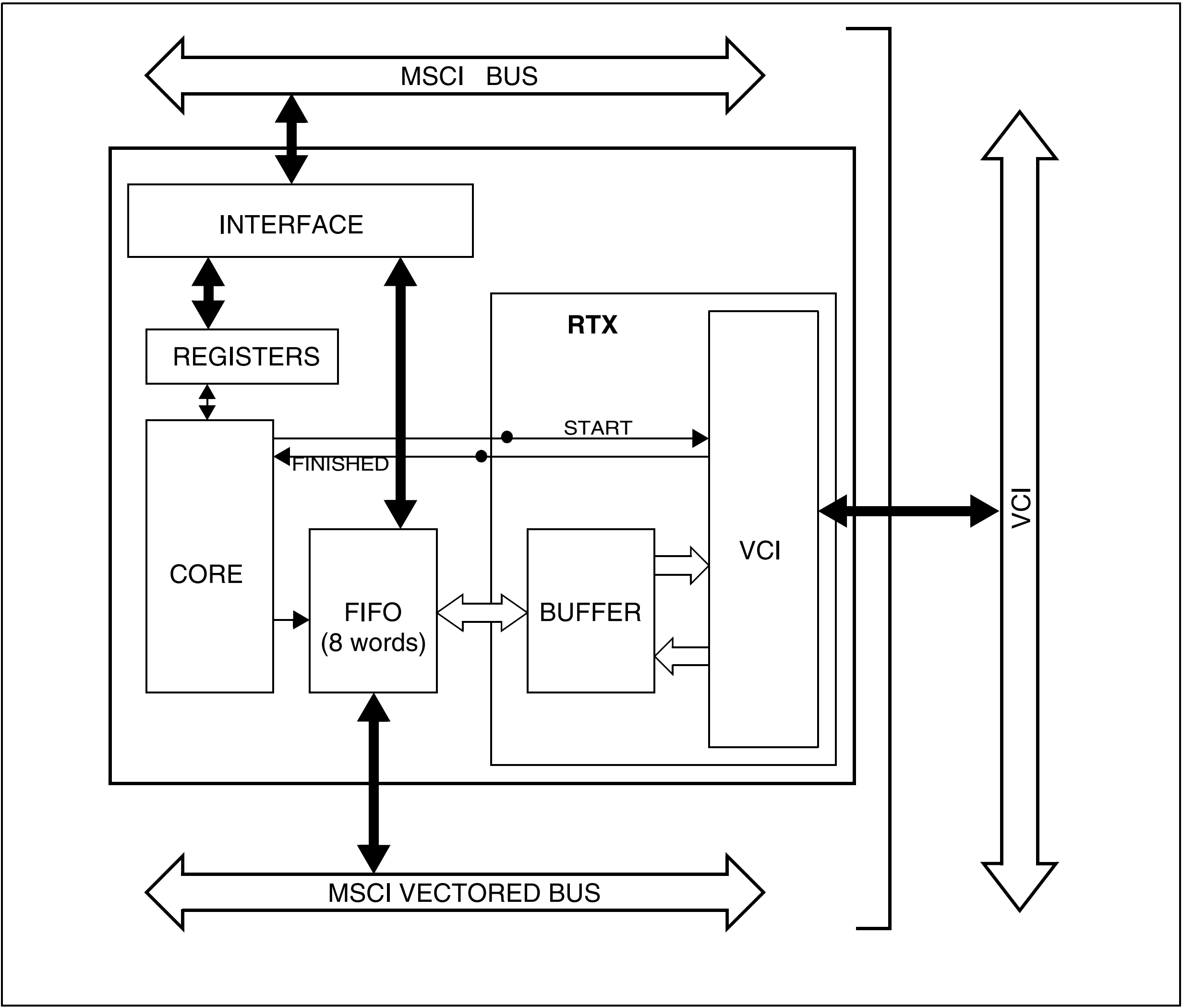
The MSCI VCI Interface handles communications between the MSCI and a VCI Target.

The protocol used is based on the B-VCI specification. The MSCI always acts as initiator

16.2 Main Features

- Based on the B-VCI specification
- 4 Registers
  - Control register
  - Status register
  - Target address register
  - 8-word Data FIFO (for transmission and reception)

Figure 53. MSCI VCI Interface Block Diagram



MSCI VCI INTERFACE (Cont'd)

16.3 Functional Description

The VCI Interface is designed to handle the Basic VCI protocol. Figure 54 shows a state diagram of the logic described below, following the device from state to state.

Configuration

When the VCI Interface is disabled (bit VCIEN in the VCR (VCI Control Register) is reset), the VCI Interface can be configured.

The configuration consist of:

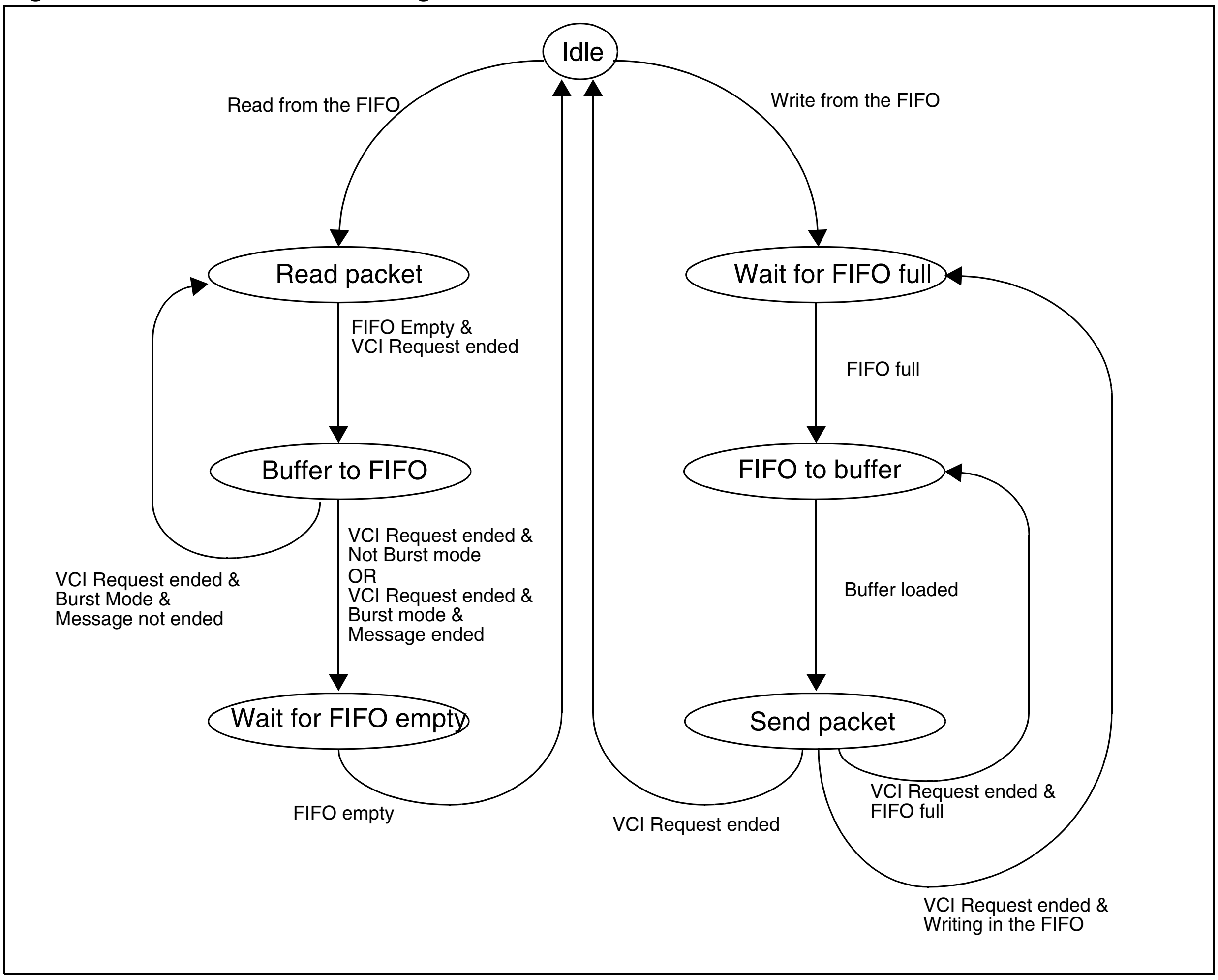
- write to the NP bits of the VCR to define the number of 8-Word packets for reading in burst mode if the VCI interface is configured in burst mode.
- defining through the bit 8BM of the VCR if the VCI sends and receive 8-bit or 16-bit data when not configured in burst mode.
- selecting with bit BM of the VCR if the message is one word long (normal mode) or is composed of n packets of m words.

Idle state

Once the interface is configured, it can be enabled by setting the bit VCIEN of the VCR.

The VCI Interface state machine is in idle state, waiting for the MSCI CPU to read or write to the FIFO to start the communication.

Figure 54. VCI Interface State Diagram


Idle
Read from the FIFO
Write from the FIFO
Read packet
FIFO Empty & VCI Request ended
Buffer to FIFO
VCI Request ended & Burst Mode & Message not ended
VCI Request ended & Not Burst mode OR VCI Request ended & Burst mode & Message ended
Wait for FIFO empty
FIFO empty
Wait for FIFO full
FIFO full
FIFO to buffer
Buffer loaded
Send packet
VCI Request ended
VCI Request ended & FIFO full
VCI Request ended & Writing in the FIFO

MSCI VCI INTERFACE (Cont'd)

Read / Write operation in normal mode

Normal mode means that the BM bit of the VCR is reset: burst mode is not enabled.

To read a single word or byte message, once the MSCI VCI Interface enabled and in idle state, the MSCI CPU just has to read the word in the VFDR (VCI FIFO Data Register).The MSCI CPU is stopped until the data is available from the target address.

The target address is given by the VTAR (VCI Target Address Register).

To write a single word or byte message, once the MSCI VCI Interface in idle state, the MSCI CPU just has to write the word to transmit in the VFDR (VCI FIFO Data Register) if the FIFO is empty. The word is automatically transmitted to the VCI Target.

Thus, from a software point of view, the FIFO register acts like any other register in normal mode for both Read and Write operations.

Note: after a write operation, the user has to wait for the end of the communication by polling the CP bit of the VSR before stopping the VCI interface with the VCIEN bit of the VCR. If not, communication may be cut leading to an incorrect VCI message transmission.

Read operation in burst mode

A complete message of N 8-Word packets can be read from the register pointed to by VTAR (VCI Target Address Register). The message size, N, is defined by NP bits in the VCR (VCI Control Register).
When the MSCI VCI Interface is on and in idle state, the MSCI CPU reads the words in the VFDR (VCI FIFO Data Register). Once the FIFO is empty, it is reloaded with the content of the VCI buffer which contains the next packet. If the packet is not available, the CPU is stopped until it arrives.
After the last word of the last packet has been read, the LWR bit of the VSR (VCI Status Register) is set.

Write operation in burst mode

A 8-Word packet can be written to the register pointed to by VTAR (VCI Target Address Register).
When the MSCI VCI Interface is on and in idle state, the MSCI CPU writes the word to transmit in the VFDR (VCI FIFO Data Register). Once the FIFO is full (8 words have been written), its content is automatically transmitted to the VCI buffer which send on the VCI bus. The FIFO becomes empty and the bit FE of the VCR (VCI Control Register) is set, and thus, the CPU loads the next 8 words in the FIFO.
Because of the FIFO structure, loads (LD instruction) must not be used with immediate value.

Note: after a write operation, the user has to wait for the end of the communication by polling the CP bit of the VSR before stopping the VCI interface with the VCIEN bit of the VCR. If not, communication may be cut leading to an incorrect VCI message transmission.

Both read or write can be performed through:

- the regular MSCI bus: as in normal mode, the FIFO register acts as a regular register.
- the MSCI vectored bus: the complete FIFO is read/write in one CPU cycle to/from an other FIFO in the MSCI system. This is done using the LDv instruction.

16.4 Error Management

The MSCI VCI Interface does not perform any error management.

MSCI VCI INTERFACE (Cont'd)

16.4.1 MSCI VCI Interface Registers

VCI CONTROL REGISTER (VCR)
Read / Write
Reset Value: 0000 0010 1111 0000 (02F0h)

15			-			8	7				0
-	-	-	-	-	-	EP<1:0>	NP<3:0>	-	8BM	BM	VCIEN

Bit 15:8 = *Reserved.*

Bit 9:8 = **EP[1:0]** *End Point Index Bits.*
Index register for selecting the USB endpoint status and control registers.

Bit 7:4 = **NP** *Number of Packets.*
Defines the number of 8-Word packets in a message from 2 to 32. Refer to Table 32.
This bit is set and cleared by software. It can only be set when the VCI interface is disabled.

Table 32. Number of Packets vs. NP value

NP value	No. of Pkts
0h	2
1h	4
2h	6
3h	8
4h	10
5h	12
6h	14
7h	16
8h	18
9h	20
Ah	22
Bh	24
Ch	26
Dh	28
Eh	30
Fh	32

Bit 3 = *Reserved*

Bit 2 = **8BM** *8-Bit Mode.*
Select if the VCI communication is 8-bit or 16-bit through the value of the VCI BE lines.
This bit is set and cleared by software. It can only be set when the VCI interface is disabled.
0: 16-bit mode: VCI BE = "11"
1: 8-bit mode: VCI BE = "01"

Bit 1 = **BM** *Burst Mode.*
This bit is set and cleared by software. It can only be set when the VCI interface is disabled.
0: Word
1: Burst

Bit 0 = **VCIEN** *VCI EN.*
This bit is set and cleared by software.
To prevent spurious communication breaks, the user must reset this bit when no communication is in progress i.e the CP bit of the VSR register is '0'.
0: VCI Interface is disabled
1: VCI Interface is enabled

MSCI VCI INTERFACE (Cont'd)

VCI STATUS REGISTER (VSR)
Read / Write
Reset Value: 0000 0000 0000 0001 (0001h)

15							8	7							0
-	-	-	-	-	-	-	-	-	-	UDRO	UDRI	LWR	CP	FF	FE

Bit 15:6 = *Reserved*.

Bit 5 = **UDRO** *USB DMA Request Out*.
This bit is set and cleared by hardware by the USB cell.
This bit is set when the USB OUT endpoint assigned to the MSCI is empty and waiting for data
0: No request
1: USB DMA request for OUT endpoint

Bit 4 = **UDRI** *USB DMA Request In*.
This bit is set and cleared by hardware by the USB cell.
This bit is set when the MSCI dedicated USB IN endpoint is full and waiting to be flushed:
0: No request
1: USB DMA request for IN endpoint

Bit 3 = **LWR** *Last Word Read*.
This bit is set by hardware when the last word of the last packet of message has been read in the FIFO and cleared by software by writing '1'.
0: No message / message not buffered
1: Message buffered

Bit 2 = **CP** *Communication in Progress*.
This bit is set and cleared by hardware when a communication is in progress.
Because of internal VCI interface pipelining, the user must check that the CP bit is '0' before turning off the VCI interface.
For write operations in burst mode, CP is set after the first word has been written in the FIFO.
0: No Communication
1: Communication in Progress

Bit 1 = **FF** *FIFO Full*.
This bit is set and cleared by hardware when the FIFO is full.
0: FIFO not full
1: FIFO full

Bit 0 = **FE** *FIFO Empty*.
This bit is set and cleared by hardware when the FIFO is empty.
0: FIFO not empty
1: FIFO empty

MSCI VCI INTERFACE (Cont'd)

VCI TARGET ADDRESS REGISTER (VTAR)
Read / Write
Reset Value: 0000 0000 0000 0000 (0000h)

15							8	7							0
-	-	-	-	-	-	-	-	-	-	AD5	AD4	AD3	AD2	AD1	AD0

Bit 15:6 = *Reserved*.

Bit 5:0 = **TA[5:0]** *Target Address*.
Target address.
Can be read or written by software. It can only be written when no communication is active.

VCI FIFO DATA REGISTER (VFDR)
Read / Write
Reset Value: 0000 0000 0000 0000 (0000h)

15							8	7							0
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Bit 15:0 = **FD[15:0]** *FIFO Data*.
FIFO Data.
Can be read or written by hardware and software.

MSCI VCI INTERFACE (Cont'd)

16.4.2 MSCI VCI Interface software example

16.4.2.1 Burst mode / scalar mode

The program below gives an example of MSCI control flow between VCI and the data RAM in burst mode:

```
        CLR     DP1             ; Clear data pointer
CONF_R  LDl     VTAR, #$20      ; Target : USB output FIFO
        LDl     VCR, #$03       ; VCI On in burst mode
        AUTOINC                 ; Set Autoinc mode

LOOPR   CALL    RDBK            ; Read 8-word block
        CPBS    VSR, #3         ; Last block ?
        JPNCond LOOPR           ; If not loop

        CLR     VCR             ; Turn off the VCI

CONF_W  LDl     VTAR, #$21      ; Target : USB input FIFO
        LDl     VCR, #$03       ; VCI On in burst mode
        CLR     DP1             ; Clear Data Pointer
        AUTOINC                 ; Set Autoinc mode
        WOSet                   ; Wait On Set mode

LOOPW   WBS     VSR, #0         ; Wait for FIFO empty
        CALL    WRBK            ; Write 8-word block
        CPBS    DP1, #8         ; DP1 = $100 ?
        JPNCond LOOPW           ; If not, continue

        WORst                   ; Wait on Reset mode
        WBS     VSR, #2         ; Wait for the end of communication
        CLR     VCR             ; Turn off the VCI interface
        STOP                    ; End of routine

RDBK    LD      [DP1]+, VFDR
        LD      [DP1]+, VFDR
        LD      [DP1]+, VFDR
        LD      [DP1]+, VFDR
        LD      [DP1]+, VFDR
        LD      [DP1]+, VFDR
        LD      [DP1]+, VFDR
        LD      [DP1]+, VFDR
        RET

WRBK    LD      VFDR, [DP1]+
        LD      VFDR, [DP1]+
        LD      VFDR, [DP1]+
        LD      VFDR, [DP1]+
        LD      VFDR, [DP1]+
        LD      VFDR, [DP1]+
        LD      VFDR, [DP1]+
        LD      VFDR, [DP1]+
        RET
```

MSCI VCI INTERFACE (Cont'd)

16.4.2.2 Burst mode / vectored mode

The program below gives an example of MSCI control flow between VCI and another interface with a FIFO called PFR through the vectored bus in burst mode:

Transfer from VCI to parallel interface in vectored mode

```
INIT_R  LDl     PCR2,#$29       ; init for parallel itf
        LDl     PCR1,#$2A       ; INPUT 20 MHz for NAND
        LDh     PCR1,#$00       ; 8-bit lsb input
        LDl     PNDR,#$00
        LDh     PNDR,#$02       ; 512 bytes to send

CONF_R  LDl     VTAR, #$20      ; Target : USB output FIFO
        LDl     VCR, #$03       ; VCI On in burst mode
        WOSet                   ; Wait On Set mode
LOOPR   WBS     PSR, #0         ; Wait parallel itf fifo empty
        LDv     PFDR,VFDR       ; Read 8-word block
        CPBS    VSR, #3         ; Last block ?
        JPNCond LOOPR           ; If not loop

        WBS     PSR, #2         ; Wait for the end of communication
        CLR     VCR             ; Turn off the VCI
```

Transfer from parallel interface to VCI in vectored mode

```
INIT_W  LDl     PCR2,#$28       ; init for parallel itf
        LDl     PCR1,#$b2       ; 15 MHz for NAND output
        LDh     PCR1,#$02       ; RE and 16-bit
        LDl     PNDR,#$00
        LDh     PNDR,#$02       ; 512 bytes to read

CONF_W  LDl     VTAR, #$21      ; Target : USB input FIFO
        LDl     VCR, #$03       ; VCI On in burst mode
        WOSet                   ; Wait On Set mode

LOOPW   WBS     VSR, #0         ; Wait for VCI itf FIFO empty
        LDv     VFDR, PFDR      ; Write 8-word block
        CPBS    PSR, #3         ; Last block?
        JPNCond LOOPW           ; If not, continue

        WORst                   ; Wait on Reset mode
        WBS     VSR, #2         ; Wait for the end of communication
        CLR     VCR             ; Turn off the VCI interface
```

MSCI VCI INTERFACE (Cont'd)

16.5 USB register addressing

The MSCI VCI interface enables the access to the USB registers and FIFO. As described in the MSCI VCI interface the USB registers can be accessed either in 8-bit or 16-bit word.

Due to the fact that the USB is designed using a little-endian structure and ST7 a big endian structure the USB address for the MSCI is different from the one described in USB chapter (ST7 addressing).

The USB register mapping is described in table 39 for a access in 8-bit mode and table 40 for an access in 16-bit mode (to an even address)

Table 33 describes the byte accessed when odd address is accessed in 16-bit mode.

Table 33. Byte accessed in 16-bit mode

Address	Byte accessed	
0	1	0
1	2	1
2	3	2

Important note: Before accessing an indexed register (endpoint status and control registers, address 10h to 18h) the endpoint number to be accessed have to be written in EP bits of VCR register (bit 9:8). The default endpoint addressed (reset value) is endpoint 2.

Due to the fact that the MSCI is a 16-bit core it is naturally recommended to access the buffer in 16-bit mode (normal or burst mode).

Table 34. USB Register Map with Reset Values seen from MSCI through VCI interface (8-bit access)

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
00h	FADDR Reset Value	UPD 0	FAD6 0	FAD5 0	FAD4 0	FAD3 0	FAD2 0	FAD1 0	FAD0 0
01h	PWRR Reset Value	0	SCON 0	HSE 1	HSM 0	RST 0	RSM 0	SUSM 0	ESUSM 0
02h	ITINR Reset Value	0	0	0	0	0	EP2I 0	EP1I 0	EP0 0
04h	ITOUTR Reset Value	0	0	0	0	0	EP2O 0	EP1O 0	0
06h	ITINER Reset Value	0	0	0	0	0	EP2IE 1	EP1IE 1	EP0E 1
08h	ITOUTER Reset Value	0	0	0	0	0	EP2OE 1	EP1OE 1	0
0Ah	ITUSBR Reset Value	0	0	0	0	SOF 0	RST 0	RSM 0	SUSP 0
0Bh	ITUSBER Reset Value	0	0	0	0	SOFE 0	RSTE 1	RSME 1	SUSPE 0
0Ch	FRNBRL Reset Value	FN7 0	FN6 0	FN5 0	FN4 0	FN3 0	FN2 0	FN1 0	FN0 0
0Dh	FRNBRM Reset Value	0	0	0	0	0	FN10 0	FN9 0	FN8 0
0Eh	RESERVED								
0Fh	RESERVED								
10h	INMAXPRL Reset Value	IMP7 0	IMP6 0	IMP5 0	IMP4 0	IMP3 0	IMP2 0	IMP1 0	IMP0 0
11h	INMAXPRM Reset Value	0	0	0	0	0	IMP10 0	IMP9 0	IMP8 0
12h	INCSRL (CSR0) IND=0 Reset Value	SSE 0	SOPR 0	SDST 0	SE 0	DE 0	STST 0	IPR 0	OPR 0
	INCSRL IND=1 or 2 Reset Value	0	CDT 0	STST 0	SDST 0	FLFI 0	UNDR 0	FINE 0	IPR
13h	INCSRM IND=0 Reset Value	0	0	0	0	0	0	0	0
	INCSRM IND=1 or 2 Reset Value	ASET 0	0	0	DMAE 0	FDT 0	0	0	0
14h	OUTMAXPRL Reset Value	OMP7 0	OMP6 0	OMP5 0	OMP4 0	OMP3 0	OMP2 0	OMP1 0	OMP0 0

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
15h	OUTMAXPRM Reset Value	0	0	0	0	0	OMP10 0	OMP9 0	OMP8 0
16h	OUTCSRL Reset Value	CDT 0	STST 0	SDST 0	FLFI 0	0	0	FIFU 0	OPR 0
17h	OUTCSRM Reset Value	ACLR 0	0	DMAE 0	DNY 0	DMAM 0	0	0	0
18h	OUTCNTRL Reset Value	OC7 0	OC6 0	OC5 0	OC4 0	OC3 0	OC2 0	OC1 0	OC0 0
19h	OUTCNTRM Reset Value	0	0	0	OC12 0	OC11 0	OC10 0	OC9 0	OC8 0

Table 35. USB Register Map seen from MSCI through VCI interface (16-bit access)

Addr. (Hex.)	Register Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00h	PWRR / FADDR	0	SCON	HSE	HSM	RST	RSM	SUSM	ES-USM	UPD	FAD6	FAD5	FAD4	FAD3	FAD2	FAD1	FAD0
02h	ITINR	0	0	0	0	0	0	0	0	0	0	0	0	0	EP2I	EP1I	EP0
04h	ITOUTR	0	0	0	0	0	0	0	0	0	0	0	0	0	EP2O	EP1O	0
06h	ITINER	0	0	0	0	0	0	0	0	0	0	0	0	0	EP2IE	EP1IE	EP0E
08h	ITOUTER	0	0	0	0	0	0	0	0	0	0	0	0	0	EP2OE	EP1OE	0
0Ah	ITUSBER / ITUSBR	0	0	0	0	SOFE	RSTE	RSME	SUSPE	0	0	0	0	SOF	RST	RSM	SUSP
0Ch	FRNBR	0	0	0	0	0	0	FN10	FN8	FN7	FN6	FN5	FN4	FN3	FN2	FN1	FN0
0Eh	RESERVED																
10h	INMAXPR	0	0	0	0	0	IMP10	IMP9	IMP8	IMP7	IMP6	IMP5	IMP4	IMP3	IMP2	IMP1	IMP0
12h	INCSR (CSR0) IND=0									SSE	SOPR	SDST	SE	DE	STST	IPR	OPR
	INCSR IND=1 or 2	ASET	0	0	DMAE	FDT	0	0	0	0	CDT	STST	SDST	FLFI	UNDR	FINE	IPR
14h	OUTMAX-PR	0	0	0	0	0	OMP10	OMP9	OMP8	OMP7	OMP6	OMP5	OMP4	OMP3	OMP2	OMP1	OMP0
16h	OUTCSR	ACLR	0	DMAE	DNY	DMAM	0	0	0	CDT	STST	SDST	FLFI	0	0	FIFU	OPR
18h	OUTCNTR	0	0	0	OC12	OC11	OC10	OC9	OC8	OC7	OC6	OC5	OC4	OC3	OC2	OC1	OC0
20h	Endpoint 0 OUT FIFO (Read)	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
	Endpoint 0 IN FIFO (Write)	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
22h	Endpoint 1 OUT FIFO (Read)	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
	Endpoint 1 IN FIFO (Write)	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Addr. (Hex.)	Register Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
24h	Endpoint 2 OUT FIFO (Read)	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
	Endpoint 2 IN FIFO (Write)	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

17 MSCI PARALLEL INTERFACE

17.1 INTRODUCTION

The MSCI Parallel Interface is a hardware block controlled by the MSCI core. It provides fast parallel communication in master mode with the following capabilities:

- 8 or 16-bit data width.
- 8 control lines to output the configurable control signal.
- High speed continuous data flow can be obtained with internal double buffering.
- 22-bit ECC Error Code correction generator for 1 bit correction in 256 byte packet.
- Hardware Reed Solomon encoder and decoder correcting 4 bytes in a 512-byte packet.

17.2 FUNCTIONAL DESCRIPTION

The parallel interface can be used in input or output.

Three data width configurations can be used:

- 8-bit data on first half of the 16-bit data port
- 8-bit data on second half of the 16-bit data port
- 16-bit data.

In output mode the parallel interface automatically sends data on data I/O ports (8-bit or 16-bit) and generates Write Enable or clock signals on dedicated I/Os. Data is output at beginning of cycle.

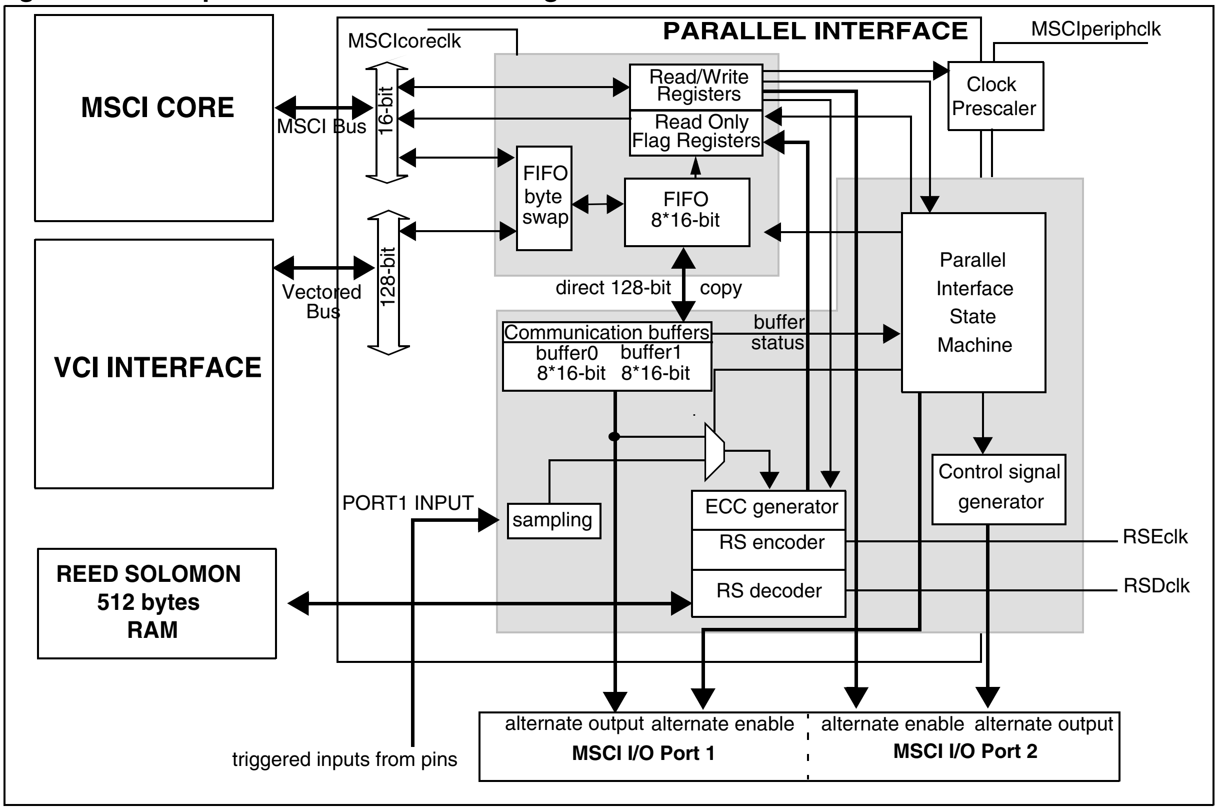
In input mode the parallel interface automatically reads data from data I/O ports (8-bit or 16-bit) and generates Read Enable or clock signals on dedicated I/Os. Data can be sampled either at the end of each cycle or on the edge of the control signal.

The shape of the control signals, the clock frequency, the control signals output ports can be controlled with dedicated configuration registers.

note: All the control signals are generated by only one generator and have the same shape.

Two Error Code Corrections algorithms are available to ensure data reliability. One ECC generator compliant with Smart Media Card specification (1-bit correction in 256-byte packets) and one Reed Solomon algorithm (4-byte correction in 512-byte packets) with full hardware encoding and decoding.

Figure 55. MSCI parallel Interface Block Diagram



MSCI PARALLEL INTERFACE (Cont'd)

17.2.1 FIFO management

17.2.1.1 Input mode

In input mode the FIFO is filled by the parallel interface and emptied by MSCI software when all FIFO words are read by the MSCI core. When the FIFO is empty and a communication buffer is full, the buffer is copied into the FIFO. The FF flag is set. The MSCI software can read all the words received by the FIFO by reading the PFDR register. Each time a read access is performed, the words are shifted out of the FIFO. First read access returns the first word received, second read access returns the second word received and so on... When the MSCI software reads the 8th word, the FIFO status is set to "empty". If more data is to be read, then the FIFO will be filled again as soon as the communication buffer is ready. (buffer0/ buffer1 selection is round robin).

The position of the bytes in the word can be reversed when reading the FIFO if the FIFO Swap Byte bit is set (bit 1 of the PCR2 register).

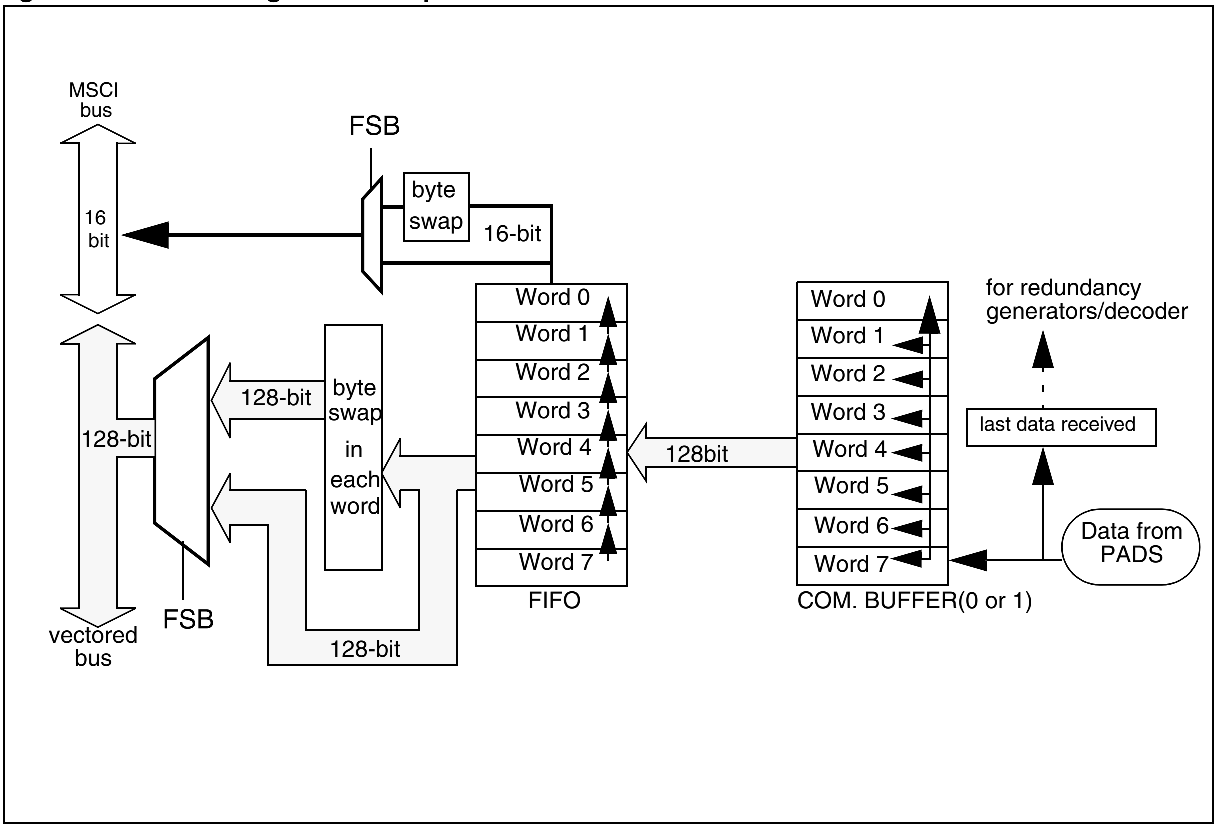
If a read operation is performed on the PFDR when the parallel communication interface is configured in input mode and when the FIFO is not full, the MSCI core is frozen until data is available from the FIFO (this also works with LDV instruction).

If the number of words to receive is not a multiple of 8 (size of the FIFO), the FIFO will be set to full when the last words of the communication are ready in the communication buffer. The program can read all the FIFO words to clear the FIFO (FIFO status reset to empty) or read only the necessary words and reset the FIFO status by writing '1' in the bit 13 of the PCR2 register.

The FIFO status must not be cleared when communication is on going, the MSCI program must first check that the EOC flag is set (bit 2 of the PSR register).

Note: The last data received (8-bit or 16-bit) is also saved internally and used by the ECC generator when the parallel interface is configured in input mode (or by the Reed Solomon encoder/decoder when enabled)

Figure 56. FIFO management in input mode



MSCI PARALLEL INTERFACE (Cont'd)

17.2.1.2 Output mode

In output mode, the FIFO is filled by the software and emptied by the parallel interface: each time a write access is performed by the MSCI software, the words are stored in the FIFO at the next location. When the MSCI software writes the 8th word, the FIFO status is set to "full". When the FIFO is full and one of the double communication buffer is empty, the FIFO is copied to the free buffer. The FE flag is set after this copy is done. If the communication is not over the FIFO can be filled again by the software and so on... The flag "Last Byte into FIFO" (bit 3 of the PSR register) is set when the last expected word is written in the FIFO.

The position of the bytes in the word can be reversed when writing into the FIFO if the FIFO Swap Byte bit is set (bit 1 of the PCR2 register).

If a write operation is performed on the PFDR when the parallel communication interface is configured in output mode and when the FIFO is full, the word is not stored in the FIFO.

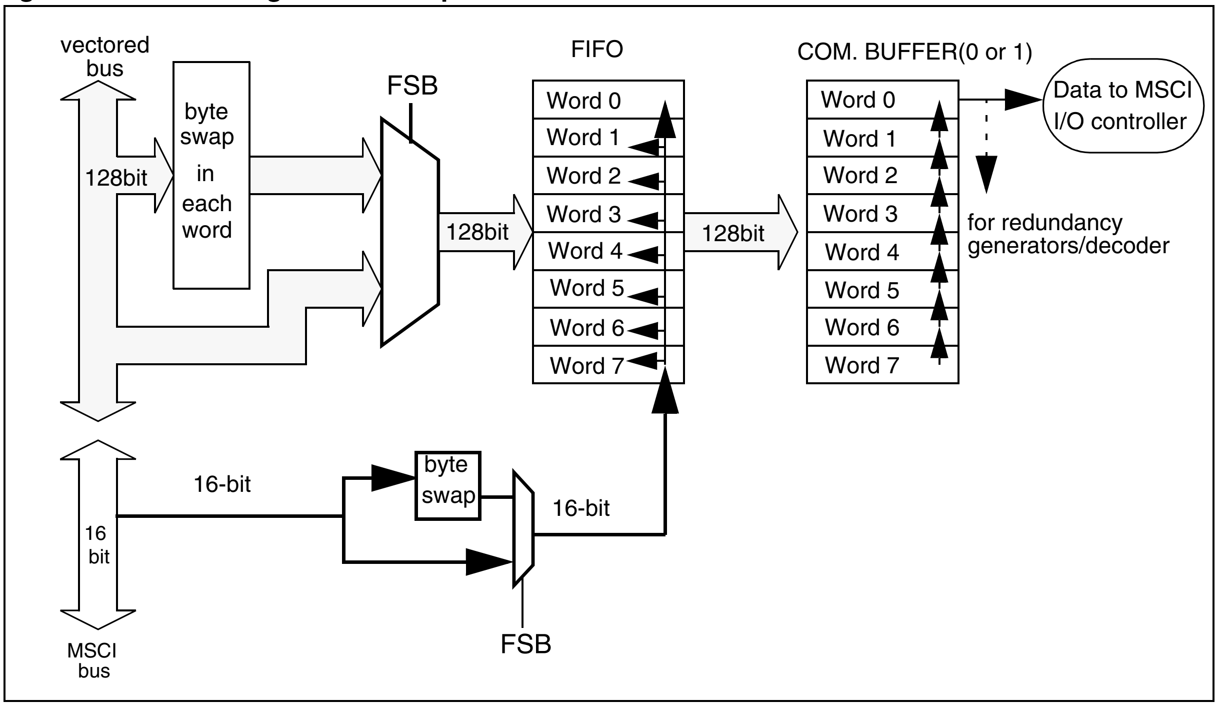
A full FIFO write with the LDV instruction must not be performed if the FIFO is not empty. **The program must wait for FIFO empty before writing the FIFO with a LDV instruction otherwise data may be lost**.

If the number of words to send is not a multiple of 8 (size of the FIFO), the FIFO will be copied in the communication buffer as soon as a buffer is empty and the last byte is written in the FIFO. The FIFO flag full is not set but if other words are written in the FIFO while the flag "Last Byte into FIFO" is set, the words are not stored in the FIFO. No other word can be written in the FIFO until a new start pulse is generated by writing 1 in the bit 15 of the PCR1 register. (the Last Byte into FIFO fag is cleared by the start pulse).

In output mode, each data is output at the beginning of the cycle and stays up till the end of one cycle.

Note: In output mode the ECC generator (and the Reed Solomon encoder/decoder if enabled) receive the value sent to I/Os.

Figure 57. FIFO management in output mode



MSCI PARALLEL INTERFACE (Cont'd)

17.3 CONFIGURING THE CONTROL LINES

Eight lines are available to provide automatic control signals output toward external communication devices. Only one control signal generator is used to create one control signal that can be output on eight dedicated I/Os. One output enable signal is available for each output (bits 4 to 11 of the PCR2 register). If an enable bit is set, the corresponding I/O is forced to output mode and controlled by the control signal generator. If the enable signal is reset, the corresponding I/O can be controlled by the MSCI I/O registers.

17.3.1 Control Signal Enable bits

When enabling some control signals by setting the corresponding CSE bits of the PCR2 register, unexpected pulse can be observed on I/O when CLDV value if the port was previously in input mode or if the port level was not the value chosen for CLDV.

To avoid an unexpected pulse on the control I/Os when switching from MSCI I/O controller to parallel interface control, the following MSCI software sequences must be used:

If control I/Os were previously in input mode:

a) Configure CLDV bit of the PCR2 register with all control lines disabled.

b) Load the MSCI I/O Controller DRO bits corresponding to the control I/O used with the CLDV value.

c) Select output mode with MSCI I/O controller for the control line I/O by setting the corresponding bits of the DDR register.

d) Set the control signal enable bits in the PCR2 register.

If control I/Os were already configured in output mode by MCI I/O logic:

a) Configure CLDV bit of the PCR2 register with all control lines disabled.

b) Load the MSCI I/O controller DRO bits corresponding to the control I/O used with the CLDV value.

c) Set the control signals enable bits in the PCR2 register.

17.3.2 Control Signal Parameters

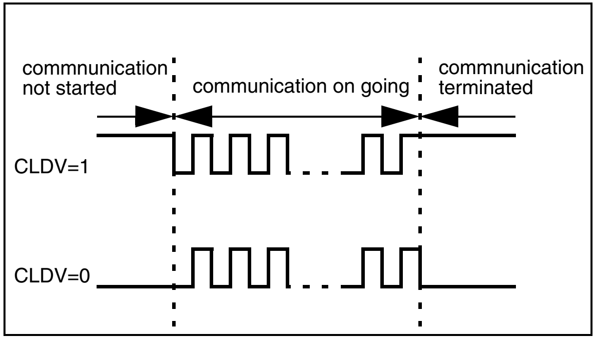
When a communication is started, a pulse is output for each data sent to (or read from) the data I/Os.

The shape of this pulse can be configured with the PCR1 and PCR2 registers. The parameters described in this chapter must be configured to obtain the desired control signal.

17.3.2.1 Control Line Default Value (CLDV)

CLDV (bit 0 in the PCR2 register) The *Control Line Default Value* is the value that is output on enabled control I/Os when communication is not running.

Figure 58. Effect of CLDV Parameter



MSCI PARALLEL INTERFACE (Cont'd)

17.3.2.2 Frequency Prescaler (F)

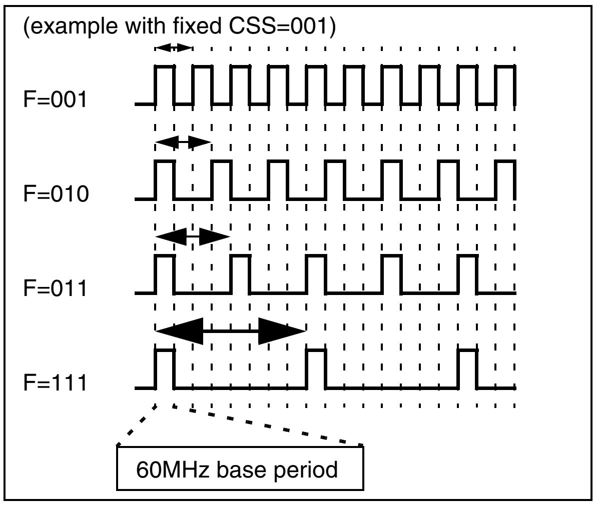
F[2:0] (bits 6:4 of the PCR1 register) *Frequency prescaler* is used to select the duration of the control signal pulse (and the frequency of the data output/input). The output frequency Fo is:

$$Fo = \frac{60}{(F+1)}MHz$$

The period of the control signal is (F+1) times higher than the base period of 60 MHz

Note: the value F[2:0]=000 is not allowed. Maximum frequency is obtained for F=001 => 30MHz

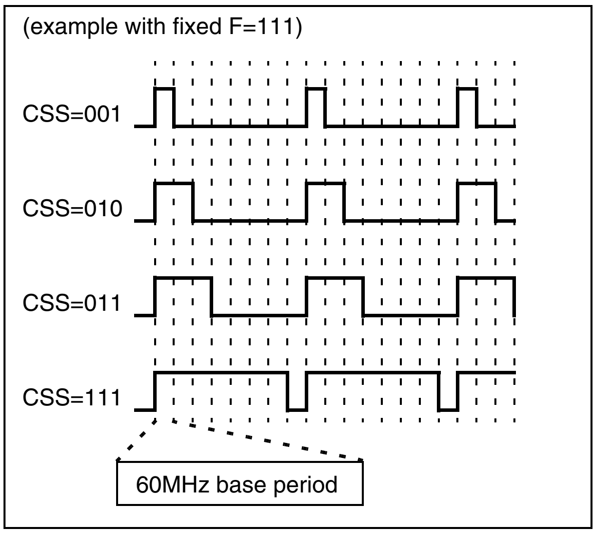
17.3.2.3 Effect of F Parameter



17.3.2.4 Control Signal Shape (CSS)

CSS[2:0] (bits 2:0 of the PCR1 register) The *Control Signal Shape* bits are used to select the duty cycle of the pulse signal. They represent the position of the edge relative to the beginning of the cycle. If the value selected is 001 the control signal edge (rising or falling) will occur 1 period of the 60MHz base frequency after the beginning of the clock cycle. If the value selected is 010, the control signal edge (rising or falling) will occur 2 periods of the base frequency after the beginning of the clock cycle and so on... For this reason, this value must be lower than the value selected for the frequency selection or equal to this value otherwise, the control signal will remain stable during the whole clock cycle. For the same reason, this value must also be greater than zero.

Figure 59. Effect of CSS Parameter

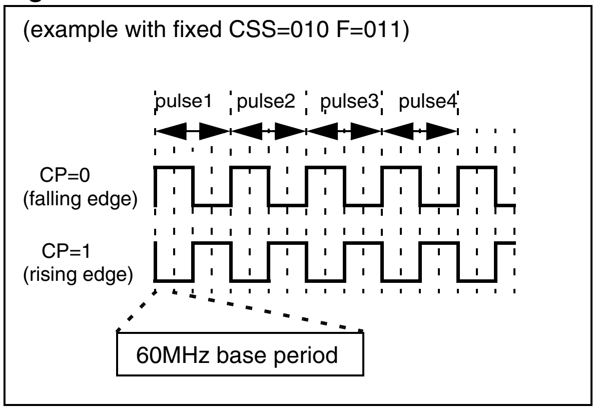


MSCI PARALLEL INTERFACE (Cont'd)

17.3.2.5 Clock Polarity (CP)

CP (bit 3 of the PCR1 register) *Clock Polarity*. This bit is used to select whether the pulse will be a rising or a falling edge (0=falling edge, 1=rising edge). If CP=0 the control signal value will be 1 at the beginning of the cycle and will fall to 0 during the cycle according to the value chosen for CSS parameter. If CP=1 the control signal value will be 0 at the beginning of the cycle and will rise to 1 during the cycle according to the value chosen for CSS parameter

Figure 60. Effect of CP Parameter



Important note: The timings shown in the figures are those obtained while communication is continuous (no parallel interface double buffer underrun or overrun). However this can be performed only if the data transfer to/from the FIFO is faster than the communication data transfer rate:

– In output mode, if the FIFO can't be filled as fast as the double buffer is sent to I/Os, the communication is stopped each time the double buffer is empty. The control signal is stretched until one buffer is full (no additional pulse generated).

– In input mode, if the FIFO is not read by MSCI software as fast as the data is read from I/Os, the communication is stopped each time the double buffer is full. The control is stretched to inactive state until one buffer is empty.

  – If "read on edge" mode is selected the data sampling position will not be modified.

  – **In "read at end of cycle" mode, the sampling will be performed at the end of the stretched cycle** (external data is assumed to be maintained until the end of the stretched cycle)

MSCI PARALLEL INTERFACE (Cont'd)

17.4 MSCI PARALLEL INTERFACE CONFIGURATION EXAMPLES

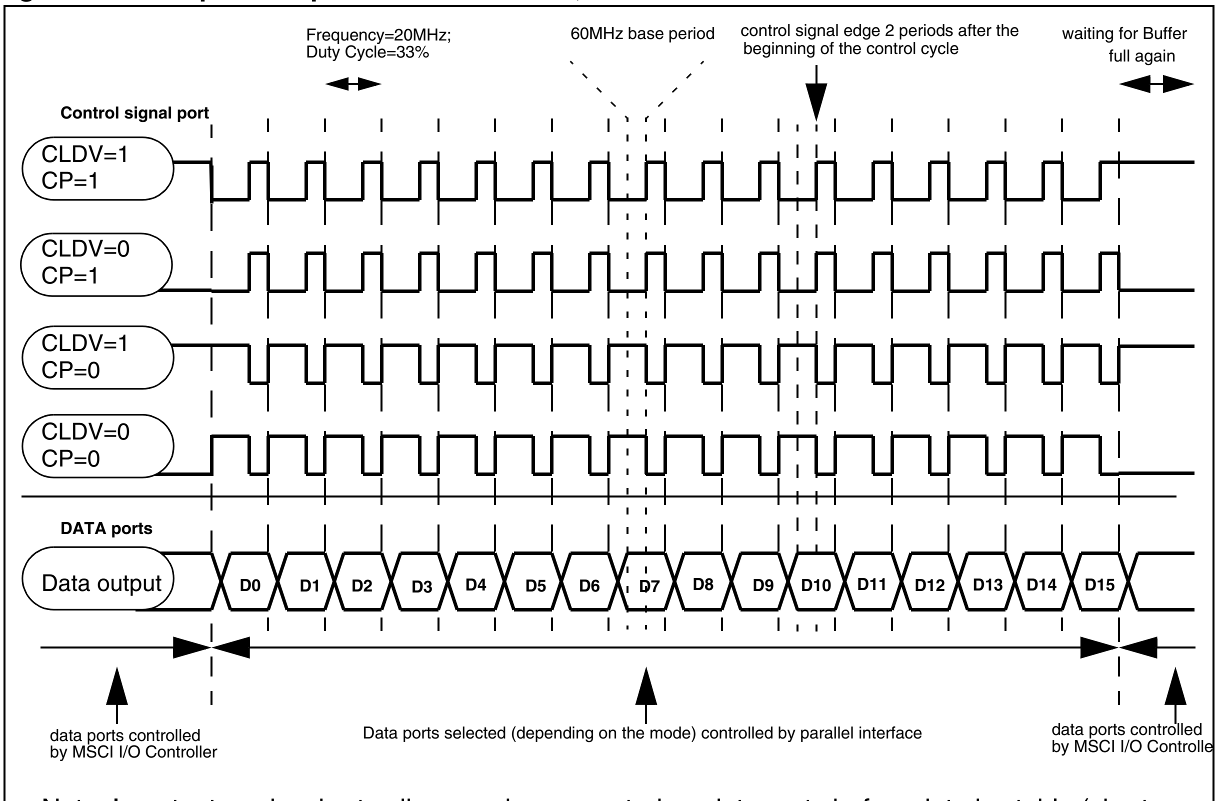
17.4.1 Examples for output mode

Once the parallel interface is configured and started, the communication begins as soon as one communication buffer is full. If the double buffer is filled by the MSCI software (through the FIFO) faster than the data is sent to the I/Os, the communication will be continuous during the complete packet. If the buffer becomes empty during a communication, inactive states will be inserted to wait until the buffer is ready. During these inactive states, the data port is not driven by the parallel interface, the control lines are frozen at the CLDV level (Control Lines Default Value).

In output mode, data is always output at the beginning of the control signal cycle whatever the control signal is. The control signal must be configured in order to match targeted device protocol.

The Control Line Default Value (CLDV) parameter is specific and must be configured before others control signals parameter and before enabling any Control Signal output with the CSE bits of the PCR2 register.

Figure 61. Examples: output mode with F=010; CSS=010



Frequency=20MHz; Duty Cycle=33%
60MHz base period
control signal edge 2 periods after the beginning of the control cycle
waiting for Buffer full again
Control signal port
CLDV=1 CP=1
CLDV=0 CP=1
CLDV=1 CP=0
CLDV=0 CP=0
DATA ports
Data output
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
data ports controlled by MSCI I/O Controller
Data ports selected (depending on the mode) controlled by parallel interface
data ports controlled by MSCI I/O Controlle

Note: In output mode, short spikes can be generated on data ports before data is stable (shorter. than the data output delay).
Only the Control line signals are guaranteed to be spike free signals (mandatory for synchronous communication).

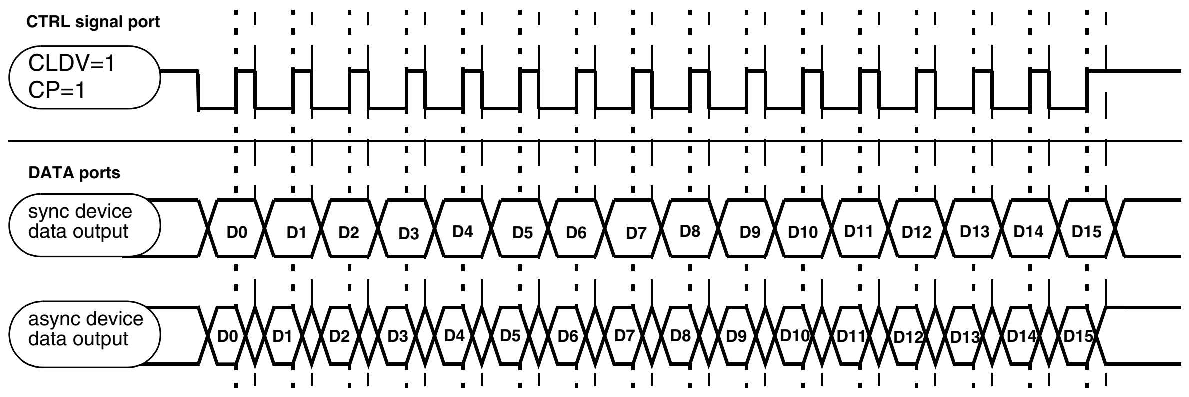
MSCI PARALLEL INTERFACE (Cont'd)

17.4.2 Examples for input mode

Once the parallel interface configured and started, the communication begins. If the double buffer is read by the MSCI software (through the FIFO) faster than the data is received from the I/Os, the communication will be continuous during the complete packet. If the two buffers become full during a communication, inactive states will be inserted to wait until the buffer is ready. During these inactive states, the control lines are frozen at the CLDV (Control Lines Default Value) level.

The data ports used are not forced to input mode by the parallel interface. They must be configured in input mode by the MSCI I/O controller (and by the ST7 I/O Controller) to let an external device drive them.

Figure 62. Examples: input mode with F=010; CSS=010; CP=1; CLDV=1


CTRL signal port
CLDV=1
CP=1
DATA ports
sync device data output
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15
async device data output
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15

If the device is synchronous and outputs data after the falling edge, the ROE can be reset to sample at the end of the control signal cycle if setup time can be guaranteed. If setup time can not be guaranteed, then the ROE bit must be set (data output time of the external device must be shorter than the low level time on control line)

- - - - -	data sampling if ROE=1
— —	data sampling if ROE=0

If the device is asynchronous and data is output only when control signal is low, the ROE bit must be set to sample data on the rising edge of the control signal. (data output time of the external device must be shorter than the low level time on control line)

Note: If CP=0 configuration is selected (falling edge generated) only the control signal generation will be modified. This means that data will be sampled on the falling edge of the control signal if ROE=1 or at the end of the control signal if ROE=0.

MSCI PARALLEL INTERFACE (Cont'd)

17.5 CASE OF NON CONTINUOUS DATA FLOW

17.5.1 Double buffer underflow in output mode

If the MSCI software is not able to send data fast enough to the FIFO while the MSCI parallel interface still has to send data, the communication stops until FIFO is filled again and copied into the communication buffer.

The control signal is forced to the CLDV state and no data is driven on the port by the parallel interface (data I/O state is controlled by the MSCI I/O Controller when communication is frozen).

see Figure 63

17.5.2 Double buffer overflow in input mode

If the MSCI software is not able to read data fast enough from the FIFO while the MSCI parallel interface still has to store data, the communication stops until a communication buffer is free again.

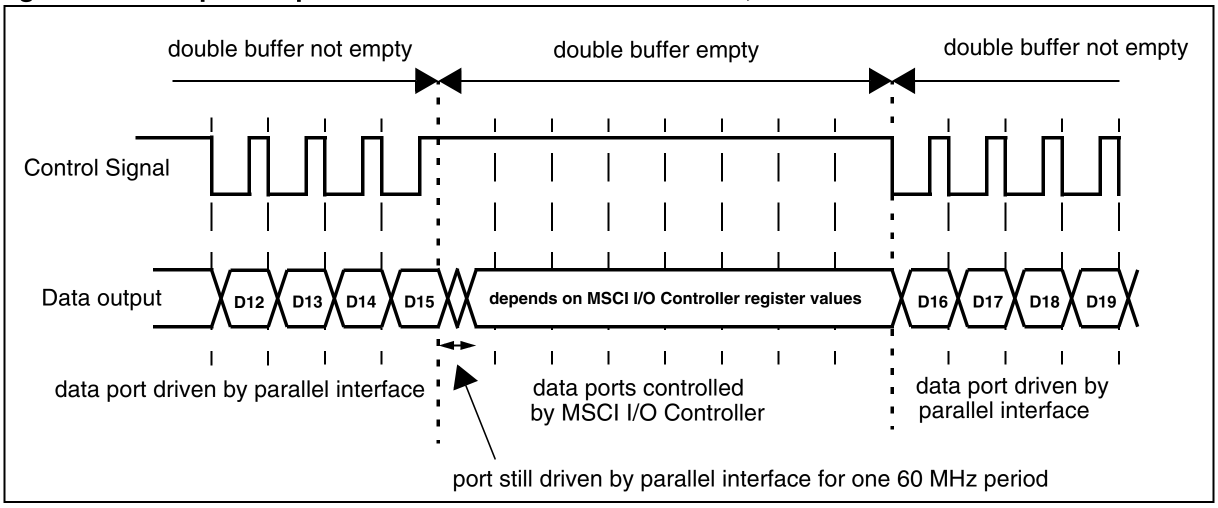
If the option ROE=1 is selected, the control signal is forced to the CLDV state and no data is driven on the port by the parallel interface (data I/O state depends on the MSCI I/O Controller).

If the option ROE=0, the last value of the control signal is kept on the port until a buffer is available. (last data is read at the end of the stretched cycle)

In both cases, no additional pulse is generated.

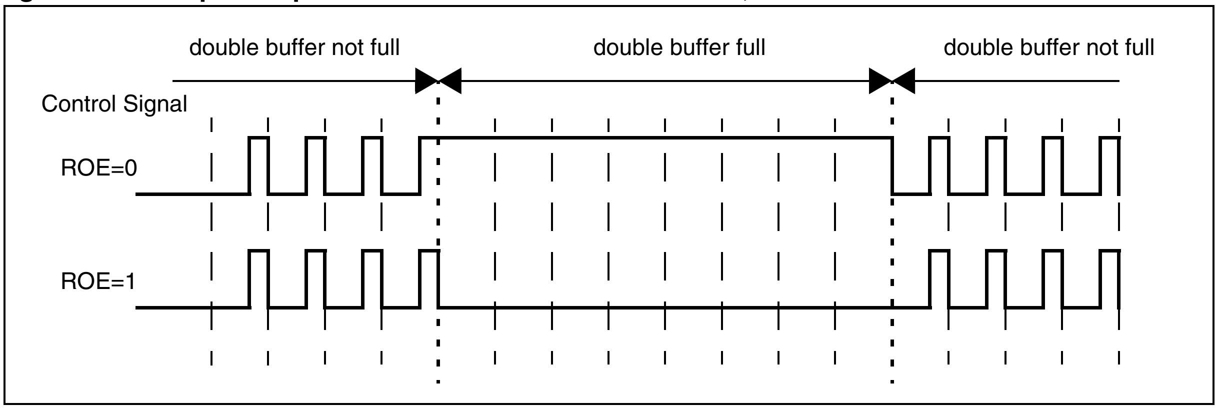
see Figure 64

Figure 63. Example: output mode with CP=1 CLDV=1 F=010; CSS=010



double buffer not empty
double buffer empty
double buffer not empty
Control Signal
Data output
D12
D13
D14
D15
depends on MSCI I/O Controller register values
D16
D17
D18
D19
data port driven by parallel interface
data ports controlled by MSCI I/O Controller
data port driven by parallel interface
port still driven by parallel interface for one 60 MHz period

Figure 64. Examples: input mode with CP=1 CLDV=0 F=010; CSS=010



MSCI PARALLEL INTERFACE (Cont'd)

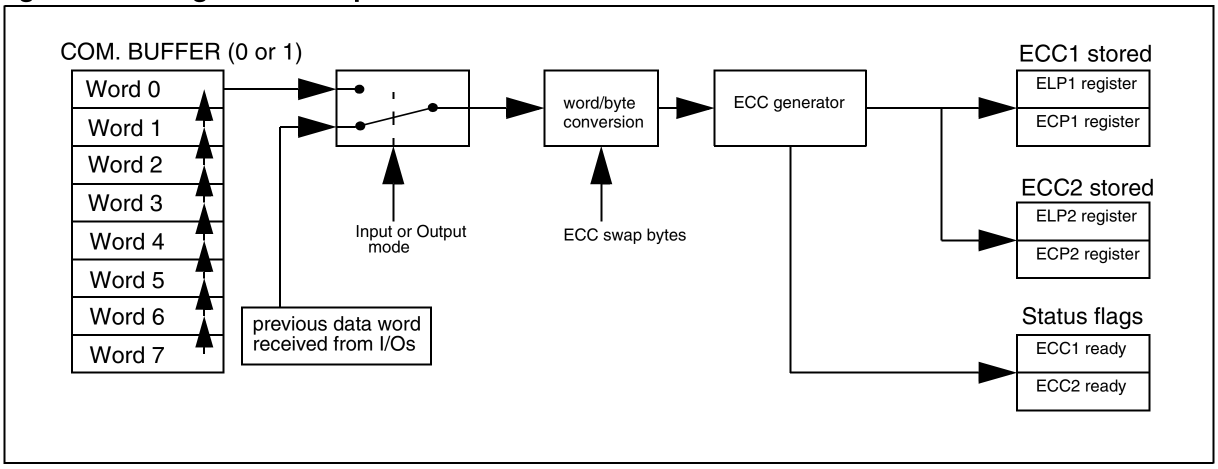
17.6 ECC GENERATOR

The ECC generator is a hardware system that computes Error Code Correction parity bits from data sent or received by the parallel interface. It is compliant with the Smart Media Card specification. It allows correction of one bit in each 256-byte data packet. It is designed to work with 512-byte packets: two ECC codes are stored in internal registers and can be read by the MSCI core after the 512-byte packet has been sent or received by the parallel interface.

Each Generated code is made of 16 line parity bits and 6 column parity bits. This 22bit ECC is generated every 256 bytes of data. If the parallel interface is used in 16-bit data mode, the data word is split into two bytes that are sent to the ECC generator. The order of the bytes in the word for the ECC generator can be selected with the bit 2 of the PCR2 register (*ECC Swap Bytes).* For each 512-byte packet, two ECC results are automatically stored by the ECC generator without interrupting the communication. The ECC generated for the first 256 bytes is stored in the ELP1 and ECP1 registers, the ECC generated for the next 256 bytes is stored in the ELP2 and ECP2 registers. If more than 2*256 bytes are sent/received, only the two first ECCs are stored. Two flags are available in the PSR register to indicate whether ECC1 and ECC2 are available for reading or not.

The ECC generator and the ECC line and column parity registers are reset when a new start is performed on the parallel interface.

Figure 65. ECC generator implementation



MSCI PARALLEL INTERFACE (Cont'd)

17.7 REED SOLOMON ENCODER

The Reed Solomon encoder is a hardware system that computes error code correction parity symbols from data sent (or received) by the parallel interface. It allows the correction of 4 bytes in each 512-byte packet.

The RS encoder works with 10 bit symbols. The 2 most significant bits of the RS encoder input are forced to 0 by hardware.

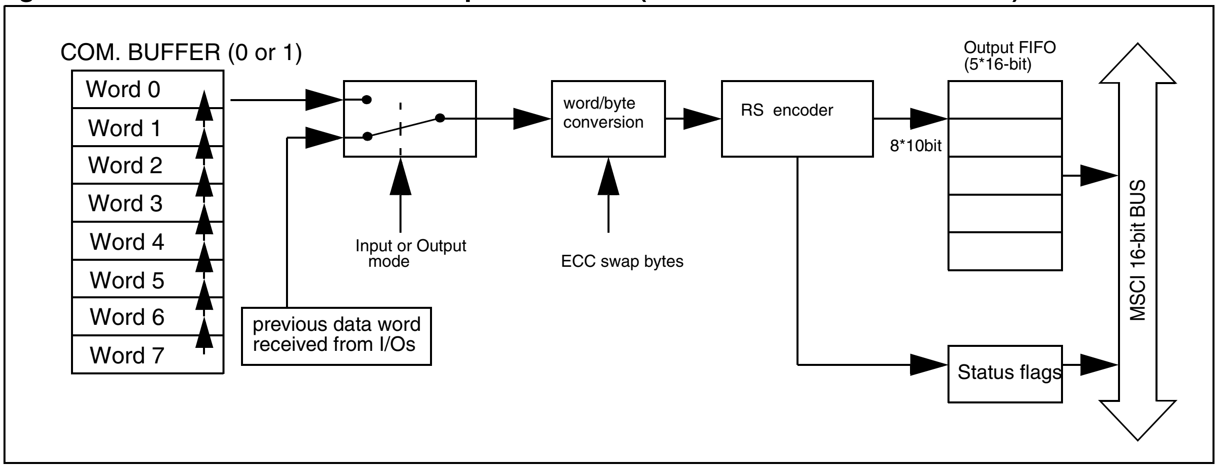
The generated code is made of eight 10-bit symbols that can be read as five 16-bit words by the MSCI core after the "parity symbols ready" flag is set. If the parallel interface is used in 16-bit data mode, the data word is split into two bytes that are sent to the ECC generator. The order of the bytes in the word for the ECC generator can be selected with bit 2 of the PCR2 register (*ECC Swap Bytes)*

The redundant code in the output FIFO is replaced each time a new set of parity symbols is available.

The output FIFO status can not be reset by MSCI software, all five 16-bit words must be read in order to make the FIFO point to the first word again.

Reading this register when the FIFO is not ready does NOT freeze the MSCI CPU.

Figure 66. Reed Solomon encoder implementation (When RS encoder enable = 1)



MSCI PARALLEL INTERFACE (Cont'd)

17.8 REED SOLOMON DECODER

The Reed Solomon decoder is a hardware system that can detect and correct 4 erroneous bytes in a 512-byte data packet received (or sent) by the parallel interface. It must receive packets made of 520 10-bit symbols (512 data symbols followed by 8 parity symbols). See Figure 67

The data symbols are provided to the decoder by the parallel interface when the decoder is enabled. They are converted to 10 bit by adding 2 most significant bits and forcing them to 0. The parity symbols must be sent to the decoder through a dedicated input FIFO. This FIFO makes the conversion from 16-bit words to 10-bit words for the parity symbols.

After receiving a complete packet of 520 symbols, the Reed Solomon decoder automatically starts its algorithm. After a few cycles it is able to indicate whether the packet is corrupted with the "errors" and "errors_valid" flags. If the packet contains errors, the correction algorithm is automatically started. After approximately 20 MSCI cycles (depends on the error) the Reed Solomon is ready to output 512-byte corrected data packed. Data output of the decoder can be read through a dedicated output FIFO. See Figure 68

The Input FIFO can be filled at any time by the MSCI core. When the RS decoder has received all the data bytes, the MSCI program must send the redundant symbols to the decoder. To do this the MSCI software must write 1 in the "Feed Decoder" bit of the RSCSR register. This sends the content of the input FIFO in 10-bit format to the decoder.

The redundancy words must be sent in the same order and with the same byte ordering as they were read from the encoder output!

The output FIFO is automatically filled by the decoder when the errors are corrected (this only happens if an error was detected). The MSCI software can recover the 512-byte data packet from this output FIFO by series of eight 16-bit words.

Note: the same register is used to access input FIFO and output FIFO of the RS decoder. Writing into this register store data in the input FIFO, reading this register returns data from the output FIFO.

Figure 67. Reed Solomon 520-symbol frame.

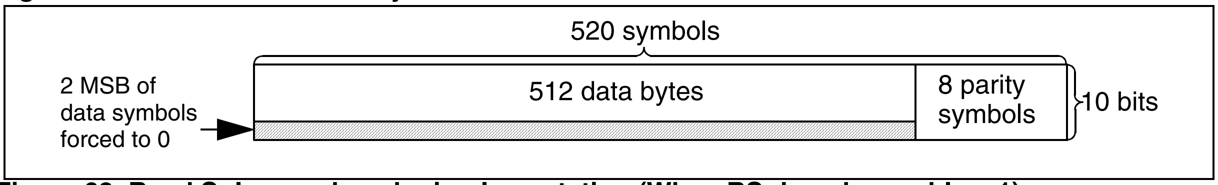
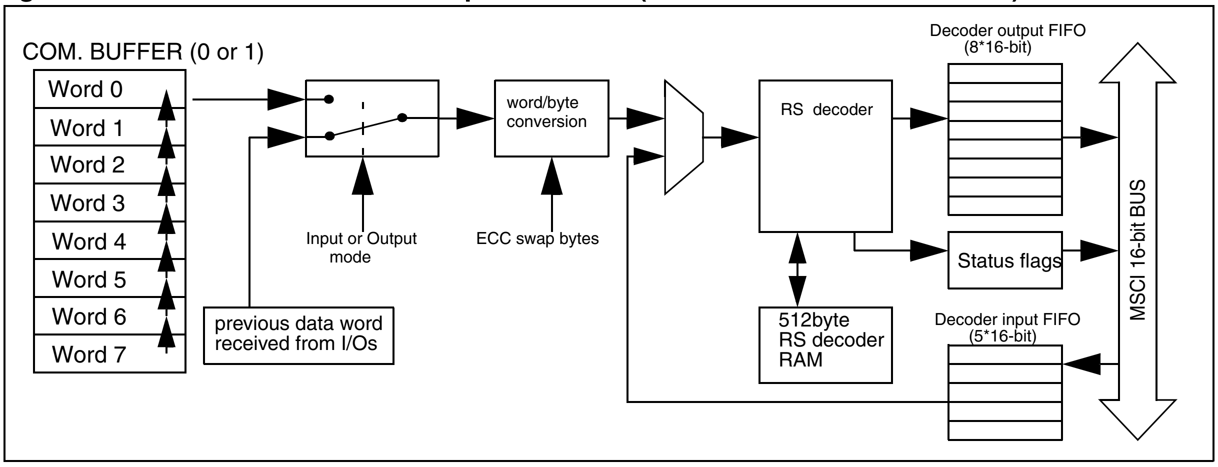
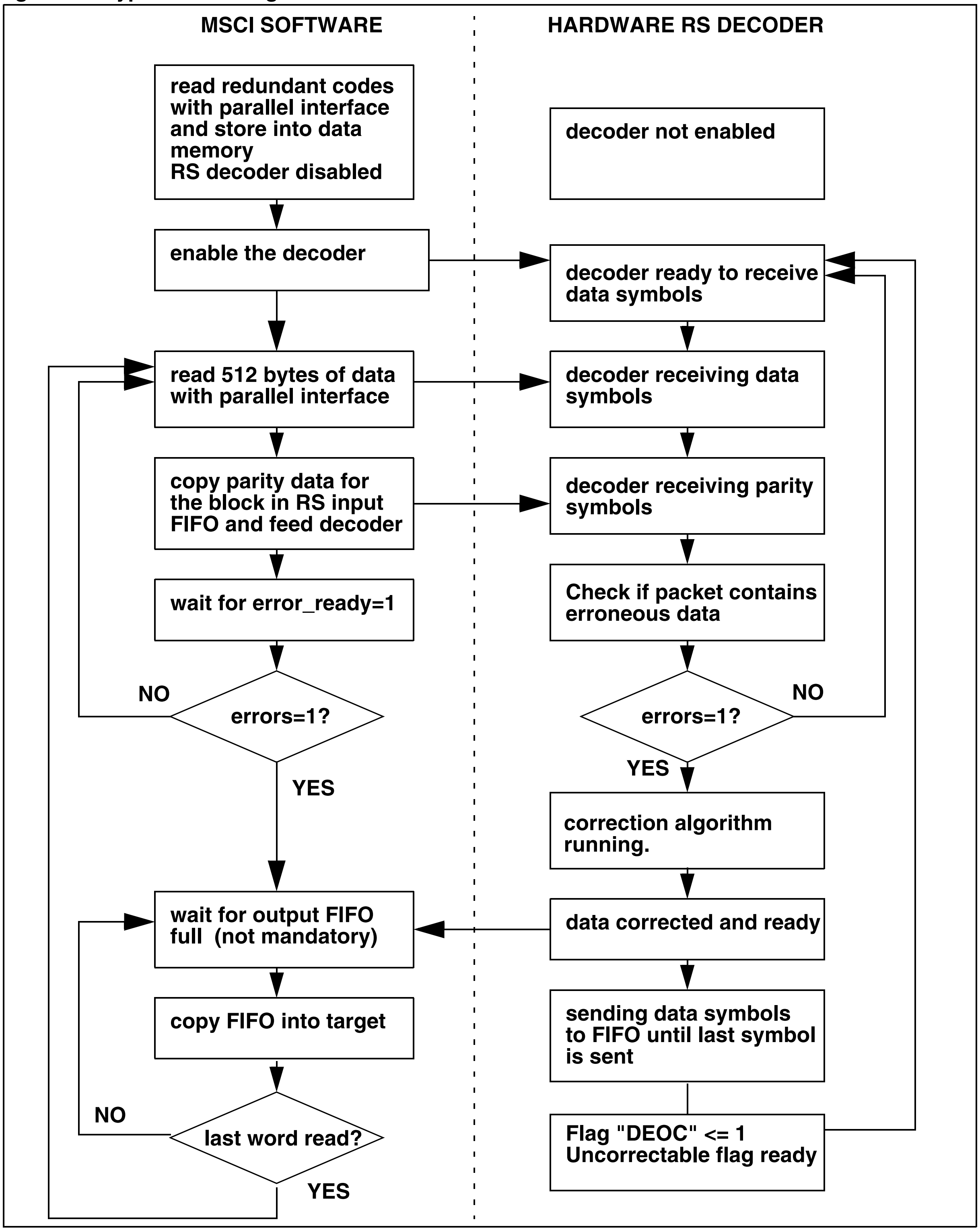


Figure 68. Reed Solomon decoder implementation (When RS decoder enable = 1)



MSCI PARALLEL INTERFACE (Cont'd)

Figure 69. Typical decoding flow with Reed Solomon decoder


MSCI SOFTWARE
HARDWARE RS DECODER
read redundant codes with parallel interface and store into data memory RS decoder disabled
decoder not enabled
enable the decoder
decoder ready to receive data symbols
read 512 bytes of data with parallel interface
decoder receiving data symbols
copy parity data for the block in RS input FIFO and feed decoder
decoder receiving parity symbols
wait for error_ready=1
Check if packet contains erroneous data
NO
errors=1?
YES
NO
errors=1?
YES
correction algorithm running.
wait for output FIFO full (not mandatory)
data corrected and ready
copy FIFO into target
sending data symbols to FIFO until last symbol is sent
NO
last word read?
YES
Flag "DEOC" <= 1 Uncorrectable flag ready

MSCI PARALLEL INTERFACE (Cont'd)

17.9 MSCI SOFTWARE EXAMPLES

17.9.1 Loop for data send

This loop can be used even if the number of bytes to send is not multiple of 16 (size of the FIFO). The extra words written in the FIFO are ignored.

```
    ;--------------------------------
    ; CONFIGURE SEND
    ;--------------------------------
    LDl PCR2,#$19     ;
    LDl PCR1,#$AA     ; 20 MHz for NAND
    LDh PCR1,#$01     ; RE and 8-bit on msb
    LDl PNDR,#$00
    LDh PNDR,#$02     ; 512 bytes to send
    BSET PCR1,#15     ; start // transfer
    CLR DP0           ; CLEAR DP0 pointer
    CALL SEND_DATA

STOP

SEND_DATA
    WOSet
LOOP_SEND
    WBS PSR,#0              ;wait for FIFO empty
    LD PFDR,[DP0]+          ;.Fill FIFO
    LD PFDR,[DP0]+          ;..Fill FIFO
    LD PFDR,[DP0]+          ;...Fill FIFO
    LD PFDR,[DP0]+          ;....Fill FIFO
    LD PFDR,[DP0]+          ;.....Fill FIFO
    LD PFDR,[DP0]+          ;......Fill FIFO
    LD PFDR,[DP0]+          ;.......Fill FIFO
    LD PFDR,[DP0]+          ;........Fill FIFO
    CPBS PSR,#3             ;last bytes stored in FIFO?
    JPNCond  LOOP_SEND      ;if no loop
    WBS PSR,#2              ;else wait untill End Of Transmit...
    RET                     ;and return to main program
```

Note: FIFO can also be filled using vectorial mode. Please refer to VCI interface software example section.

MSCI PARALLEL INTERFACE (Cont'd)

17.9.2 Loop for data read

This loop can be used only if the number of bytes to receive is not multiple of 16 (size of the FIFO) otherwise, extra bytes would be read that do not belong to the communication frame. In input mode, the FIFO is set to full when the last bytes are received.

```
    ;--------------------------------
    ; CONFIGURE READ
    ;--------------------------------
    LDl PCR2,#$29    ; CS1
    LDl PCR1,#$2A    ; INPUT
    LDh PCR1,#$01    ; 8-bit msb input
    LDl PNDR,#$00
    LDh PNDR,#$02    ; 512 bytes to read
    BSET PCR1,#15    ; start // transfer
    CLR DP0          ; CLEAR DP0 pointer
    CALL READ_DATA
    STOP

READ_DATA

     WOSet
LOOP_READ
    ;WBS PSR,#1             ;wait for FIFO full (not mandatory because if fifo
                             ;is empty MSCI core is freezed )
    LD [DP0]+,PFDR           ;........Read FIFO
    LD [DP0]+,PFDR           ;.......Read FIFO
    LD [DP0]+,PFDR           ;......Read FIFO
    LD [DP0]+,PFDR           ;.....Read FIFO
    LD [DP0]+,PFDR           ;....Read FIFO
    LD [DP0]+,PFDR           ;...Read FIFO
    LD [DP0]+,PFDR           ;..Read FIFO
    LD [DP0]+,PFDR           ;.Read FIFO
    CPBS PSR,#3              ;last bytes read from FIFO ???
    JPNCond  LOOP_READ       ;if no loop
    WBS PSR,#2               ;else wait untill rnd Of reception...
    RET                      ;and return to main program
```

Note: FIFO can also be transferred into VCI FIFO using vectored mode. Please refer to VCI interface software example section.

MSCI PARALLEL INTERFACE (Cont'd)

17.10 REGISTER DESCRIPTION

PARALLEL INTERFACE NUMBER OF DATA REGISTER (PNDR)

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

15							8	7							0
-	-	-	-	-	-	N9	N8	N7	N6	N5	N4	N3	N2	N1	N0

Bit 15:10 = *Reserved*

Bit 9:0 = **N[9:0]** *Number of Data Register*.
Number of bytes to send in output mode or to read in input mode. **Must be strictly positive** (value 00h is not allowed). In 16-bit mode, this number must be even. If the number of bytes to send/receive is odd in 16-bit mode, the last incomplete word is not sent/read.

PARALLEL INTERFACE FIFO DATA REGISTER (PFDR)

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

15							8	7							0
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Bit 15:0 = **D[15:0]** *FIFO Data*.
When the parallel interface is configured in output mode (PCR1[7]=1), writing to this register adds one word to the FIFO.
Writing into the FIFO when it is already full has no effect. **Write access in the PFDR register can only be performed in 16-bit mode. Immediate 8-bit write through LDL or LDM or BSET/BRES must not be used on this register.**

When the parallel interface is configured in input mode PCR1[7]=0, reading this register returns one word from the FIFO.
If FIFO is empty when reading this register, the parallel interface sends a freeze signal to the MSCI core until the FIFO is filled.

Refer to the FIFO management Section 17.2.1 **for more information.**

MSCI PARALLEL INTERFACE (Cont'd)

PARALLEL INTERFACE CONFIGURATION REGISTER 1 (PCR1)

Read / Write

Reset Value: 0000 0000 0111 1111 (007Fh)

15							8	7							0
START	-	-	-	-	-	DM1	DM0	DIR	F2	F1	F0	CP	CSS2	CSS1	CSS0

Bit 15 = **START** *Start communication.(write only)*
This bit is set by software and is always read as 0. Setting in this bit generates a start pulse that initiates the data transfer. In input mode communication starts immediately (FIFO must be empty). In output mode communication starts as soon as the FIFO is full and copied into the buffer.

Bit 14:10 = *Reserved.*

Bit 9:8 = **DM[1:0]** *Data Mode.*
These bits are set and cleared by software.
When 8-bit output mode is used, data bytes are sent to port P1[7:0] or P1[15:8] depending on whether the LSB or MSB mode is selected. In 16-bit mode, data is sent to P1[15:0]. When 8-bit mode is used, the most significant byte is sent or read first. The FSB bit in the PCR2 register can be used to reverse byte order when writing into the FIFO or reading from the FIFO (equivalent to sending or reading least significant byte first).
00: 8-bit mode on LSB only
01: 8-bit mode on MSB only
10: 16-bit mode.

Bit 7 = **DIR** *Direction.*
This bit is set and cleared by software.
In output mode the data ports used are forced to output mode by the parallel interface when data is to be output. In input mode the data ports can be controlled by the MSCI I/O controller registers. Data ports must be left in input mode so they can be driven by external device.
0: Input mode
1: Output mode

Bit 6:4 = **F[2:0]** *Frequency prescaler.*
These bits are set and cleared by software. They select the prescaler factor for the control signal (applied to the 60 MHz clock)

F[2]	F[1]	F[0]	Control Signal Frequency
0	0	0	Not allowed
0	0	1	30MHz
0	1	0	20MHz
0	1	1	15MHz
1	0	0	12MHz
1	0	1	10MHz
1	1	0	8.5MHz
1	1	1	7.5MHz

Bit 3 = **CP** *Clock Polarity.*
This bit is set and cleared by software.
0: Falling edge clock pulse
1: Rising edge clock pulse

Bit 2:0 = **CSS[2:0]** *Control Signal Shape.*
These bits are set and cleared by software.
They are used to select the clock edge position in the clock cycle by steps of 16.66ns and thus, control the clock duty cycle. CSS[2:0] must be lower than F[2:0] or equal to F (Frequency selection) and greater than 0.

For more detailed description of the control signal generation refer to chapter "CONFIGURATION OF THE CONTROL LINE SIGNALS"

MSCI PARALLEL INTERFACE (Cont'd)

PARALLEL INTERFACE CONFIGURATION REGISTER 2 (PCR2)

Read / Write

Reset Value: 0000 0000 0000 0000 (0000h)

15							8	7							0
-	ED	RF	PID	CSE7	CSE6	CSE5	CSE4	CSE3	CSE2	CSE1	CSE0	ROE	ESB	FSB	CLDV

Bit 15 = *Reserved.*

Bit 14 = **ED** *ECC Disable.*
This bit is set and cleared by software.
It disables the ECC generator and forces the ECC line and column parities to reset value.
0: ECC generator enabled.
1: ECC generator disabled.

Bit 13 = **RF** *Reset FIFO.(write only)*
This bit is set by software and is always read as 0. Writing '1' in this bit resets the FIFO to empty. Note: this Reset FIFO command does not modify the value of the number of bytes written in the FIFO. The LBF byte will still rise after writing the expected number of bytes in the FIFO even if some were deleted by a FIFO reset. This bit must be used only to force the FIFO to empty if the FIFO is left at a non empty state at the end of a transfer.

Bit 12 = **PID** *Parallel Interface Disable.*
This bit is set and cleared by software.
Disabling the Parallel interface stops the communication and the double buffer but does not reset the configuration registers.
Note: setting the PID bit does not release the control signal output enable and does not reset the FIFO.
0: Parallel interface is enabled
1: Parallel interface is disabled

Bit 11:4 = **CSE[7:0]** *Control Signal Enable.*
These bits are set and cleared by software.
0: Control line output disabled
(not driven by control signal generator)
1: Control line output enabled
(driven by control signal generator)

Bit 3 = **ROE** *Read On Edge.*
This bit is set and cleared by software.
This bit is only used in input mode to define when the input data has to be read. Read on edge configuration must be selected for external devices using Read enable signals. Read at end of cycle configuration must be chosen for synchronous external devices with data maintained on data port.
0: Read at the end of each cycle.
1: Read on the active edge of the control signal.

Bit 2 = **ESB** *ECC Swap Bytes.*
This bit is set and cleared by software.
It selects the order of the bytes in the word when sent to the ECC generator in 16-bit mode.
0: Do not swap bytes (most significant byte first)
1: Swap bytes (least significant byte first)

Bit 1 = **FSB** *FIFO Swap Bytes.*
This bit is set and cleared by software.
It selects the order of the bytes in the word when reading/writing from/to the FIFO. It affects both standard 16-bit access to FIFO and direct FIFO copy with LDV instruction.
0: Do not swap bytes.
1: Swap bytes.

Bit 0 = **CLDV** *Control Lines Default Value.*
This bit is set and cleared by software.
It selects the default value that is forced on the active control lines when no pulse is generated and when communication is over or not started. When the default value is changed, it will be effective on the enabled control ports one 60 MHz clock cycle after. To avoid an unexpected pulse on the control signal, it is mandatory to change the CLDV value only when all control lines are disabled.
0: Control line default value = 0
1: Control line default value = 1

Note: Configuration registers PNDR, PCR1 and PCR2 must not be modified when a communication is on-going.

MSCI PARALLEL INTERFACE (Cont'd)

PARALLEL INTERFACE STATUS REGISTER (PSR)

Read

Reset Value: 0000 0000 0000 1101 (000Dh)

15							8	7							0
0	0	0	0	0	0	0	0	0	RDPE	E2R	E1R	LBF	EOC	FF	FE

Bit 15:7 = *Reserved*

Bit 6 = **RDPE** *RS Decoder Protocol Error.*
This bit is set by hardware when data are sent to RS Decoder when it is not ready. MSCI must receive a soft reset to restart the RS decoder in a correct state. (Can be used for debug)
0: No RS protocol error
1: RS protocol error detected

Bit 5 = **E2R** *ECC2 ready.*
This bit is set by hardware when the ECC2 line parity and column parity are updated and reset by hardware when a new start is generated by writing '1' in bit 15 of the PCR1 register.
0: ECC2 not ready
1: ECC2 ready

Bit 4 = **E1R** *ECC1 ready.*
This bit is set by hardware when the ECC1 line parity and column parity are updated and cleared by hardware when a new start is generated by writing a 1 in bit 15 of the PCR1 register.
0: ECC1 not ready
1: ECC1 ready

Bit 3 = **LBF** *Last Byte of FIFO.*
In output mode, this bit is set by hardware when the total number of bytes written in the FIFO is equal to the number of bytes to be sent. In input mode it is set by hardware when the total number of bytes read from the FIFO is equal to the expected number of bytes. In both modes, it is reset by hardware when a new start pulse is generated by writing a 1 in bit 15 of the PCR1 register.
0: Last byte of the FIFO not read/written
1: Last byte of the FIFO read/written

Bit 2 = **EOC** *End Of Communication.*
This bit is set by hardware when the parallel interface communication is over and reset by software when a new start is generated by writing '1' in bit 15 of the PCR1 register.
0: Communication is on going.
1: Communication is over.

Bit 1 = **FF** *FIFO full.*
This bit is set and cleared by hardware.
0: FIFO not full
1: FIFO full

Bit 0 = **FE** *FIFO empty.*
This bit is set and cleared by hardware.
0: FIFO not empty
1: FIFO empty

Note: The E2R and E1R flags are not reset immediately by the start but 4 MSCI clock cycles after.

ECC1 LINE PARITY (ELP1)

Read

Reset Value: 1111 1111 1111 1111 (FFFFh)

15							8	7							0
LP15	LP14	LP13	LP12	LP11	LP10	LP9	LP8	LP7	LP6	LP5	LP4	LP3	LP2	LP1	LP0

Bit 15:0 = **LP[15:0]** *Line Parity.*
ECC1 Line parity bits These bits are set by hardware and reset when a new start is generated by writing '1' in bit 15 of the PCR1 register.

MSCI PARALLEL INTERFACE (Cont'd)

ECC1 COLUMN PARITY (ECP1)
Read
Reset Value: 0000 0000 1111 1111 (00FFh)

15							8	7							0
0	0	0	0	0	0	0	0	CP5	CP4	CP3	CP2	CP1	CP0	1	1

Bit 15:8 = *Reserved.*

Bit 7:2 = **CP[6:0]** *ECC1 Column parity.*
ECC1 column parity bits These bits are set by hardware and reset when a new start is generated by writing a 1 in bit 15 of the PCR1 register.

Bit 1:0 = *Reserved.*

ECC2 LINE PARITY (ELP2)
Read
Reset Value: 1111 1111 1111 1111 (FFFFh)

15							8	7							0
LP15	LP14	LP13	LP12	LP11	LP10	LP9	LP8	LP7	LP6	LP5	LP4	LP3	LP2	LP1	LP0

Bit 15:0 = **LP[15:0]** *Line Parity.*
ECC2 line parity bits These bits are set by hardware and reset when a new start is generated by writing a 1 in bit 15 of the PCR1 register.

ECC2 COLUMN PARITY (ECP2)
Read
Reset Value: 0000 0000 1111 1111 (00FFh)

15							8	7							0
0	0	0	0	0	0	0	0	CP5	CP4	CP3	CP2	CP1	CP0	1	1

Bit 15:8 = *Reserved.*

Bit 7:2 = **CP[6:0]** *ECC1 Column parity.*
ECC1 column parity bits These bits are set by hardware and reset when a new start is generated by writing a 1 in bit 15 of the PCR1 register.

Bit 1:0 = *Reserved.*

MSCI PARALLEL INTERFACE (Cont'd)

REED SOLOMON CONTROL STATUS REGISTER (RCSR)
Read / Write
Reset Value: 0100 0000 0000 0000 (4000h).

15							8	7							0
DOFF	DOFE	DLWR	DRR	DEFV	DEF	DEOC	DUE	DNE[2]	DNE[1]	DNE[0]	DE	EPR	FD	EE	DIFF

Bit 15 = **DOFF** *Decoder Output FIFO Full.*
This bit is set by hardware when the decoder output FIFO is full and reset by hardware when MSCI software reads data from this FIFO through the RDFDR register.
0: RS Decoder output FIFO not full.
1: RS Decoder output FIFO full.

Bit 14 = **DOFE** *Decoder Output FIFO Empty.*
This bit is set by hardware when the 8th word is read by MSCI software from the decoder output FIFO through the RDFDR register. It is reset by hardware when FIFO is full again.
0: RS Decoder output FIFO not empty.
1: RS Decoder output FIFO empty.

Bit 13 = **DLWR** *Decoder Last Word Read.*
This bit is set by hardware when the last word is read from the decoder output FIFO. It is reset by hardware when a word that is not the last of a 512-byte packet is read from the decoder output FIFO.
0: RS Decoder last word not read.
1: RS Decoder last word read.

Bit 12 = **DRR** *Decoder Ready to Receive.*
This bit is set and reset by hardware. It indicates whether the decoder is ready to be used or not. Data must not be sent to the decoder if DRR=0. refer to Section 17.8 for more information.
0: RS Decoder not ready to receive.
1: RS Decoder ready to receive.

Bit 11 = **DEFV** *Decoder Error Flag Valid.*
This bit is set by hardware when the error flag is valid. This happens a few MSCI cycles after the decoder received a complete packet. It is reset by hardware when the first symbol of the next data packet is received by the decoder.
0: RS Decoder error flag is valid.
1: RS Decoder error flag is not valid.

Bit 10 = **DEF** *Decoder Error Flag.*
This bit is set by hardware when an error is detected in the current data packet. This flag is significant only when DEFV bit is set. It is reset by hardware when the first symbol of the next data packet is received by the decoder.
0: No error detected by RS decoder.
1: Error(s) detected by RS decoder.

Bit 9 = **DEOC** *Decoder End Of Correction.*
This bit is set by hardware when the correction algorithm is finished and when all words of the 512byte data packet are read from the decoder output. It is reset by hardware when the first symbol of the next data packet is received by the decoder.
0: RS Decoder correction not finished.
1: RS Decoder correction finished.

Bit 8 = **DUE** *Decoder Uncorrectable Error.*
This bit is set by hardware when the decoder detects an error that can not be recovered. This flag is significant only when DEOC=1 (after full decoding process including the reading of the complete decoded data packet from the decoder output FIFO. It is reset by hardware when the first symbol of the next data packet is received by the decoder.
0: No uncorrectable error detected by RS decoder.
1: Uncorrectable error(s) detected by RS decoder.

Bit [7:5] = **DNE[2:0]** *Decoder Number of Errors.*
These bits are set by hardware by the decoder to give the number of errors detected. This flag is significant only when DEOC=1 and if the DUE flag is not set. It is reset by hardware when the first symbol of the next data packet is received by the decoder.

DNE[2]	**DNE[1]**	**DNE[0]**	**Number of errors**
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	N/A
1	1	0	N/A
1	1	1	N/A

MSCI PARALLEL INTERFACE (Cont'd)

Bit 4 = **DE** *Decoder Enable.*
This bit is set and reset by software to enable or disable the RS decoder. If decoder is enabled, data sent or received by parallel interface are also sent to the decoder input. If DE is reset, data transferred with parallel interface are not taken into account by the RS decoder.
0: RS Decoder disabled
1: RS Decoder enabled

Bit 3 = **EPR** *Encoder Parity Ready.*
This bit is set by hardware when parity bits are ready to be read from the encoder output FIFO. It is reset by hardware when the first data of a new data packet is received by the encoder.
0: RS Encoder parity is not ready
1: RS Encoder parity is ready

Bit 2 = **FD** *Feed Decoder. (write only)*
Write 1 in this bit to send the content of the decoder input FIFO to the decoder. This must be done to provide parity symbols to the decoder in 10-bit format (eight 10-bit symbols are automatically sent from the decoder input FIFO to the decoder cell). The Decoder enable bit must be set before writing 1 in bit FD to have the parity symbols correctly taken into account by the RS decoder.
0: No effect
1: Send parity symbols to decoder now.

Bit 1 = **EE** *Encoder Enable.*
This bit is set and reset by software to enable or disable the RS encoder. When encoder is enabled, data sent or received by the parallel interface are also sent to the encoder input. If EE is reset, data transferred through the parallel interface are not taken into account by the RS encoder.
0: RS Encoder disabled.
1: RS Encoder enabled.

Bit 0 = **DIFF** *Decoder Input FIFO Full.*
This bit is set by hardware when the decoder input FIFO is full and reset by hardware when the content of the FIFO is sent to the Decoder.
0: Decoder input FIFO not full.
1: Decoder input FIFO full.

REED SOLOMON DECODER FIFO REGISTER (RDFR)
Read / Write
Reset Value: 0000 0000 0000 0000 (0000h).

15							8	7							0
DFD15	DFD14	DFD13	DFD12	DFD11	DFD10	DFD9	DFD8	DFD7	DFD6	DFD5	DFD4	DFD3	DFD2	DFD1	DFD0

Bit [15:0] = **DFD** *Decoder FIFO Data.*
Writing into this register adds a word into the RS decoder input FIFO (used to send parity symbols in 10-bit format to the decoder)

Reading this register returns a 16-bit word from the RS decoder Output FIFO. If RS decoder Output FIFO is empty when read, the MSCI core is frozen until the FIFO is ready to be read.

REED SOLOMON ENCODER FIFO REGISTER (REFR)
Read
Reset Value: 0000 0000 0000 0000 (0000h).

15							8	7							0
EFD15	EFD14	EFD13	EFD12	EFD11	EFD10	EFD9	EFD8	EFD7	EFD6	EFD5	EFD4	EFD3	EFD2	EFD1	EFD0

Bit [15:0] = **EFD** *Encoder FIFO Data.*

Reading this register returns a 16-bit word from the RS encoder Output FIFO. This FIFO contains 5 16-bit words that must all be read in order to let the FIFO pointer pointing on the first word for next data packet. Reading this register when FIFO is not ready does NOT freeze the MSCI CPU.

18 ELECTRICAL CHARACTERISTICS

18.1 PARAMETER CONDITIONS

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

18.1.1 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).

18.1.2 Typical values

Unless otherwise specified, typical data are based on T_A=25°C, VDD33=3.3V. They are given only as design guidelines and are not tested.

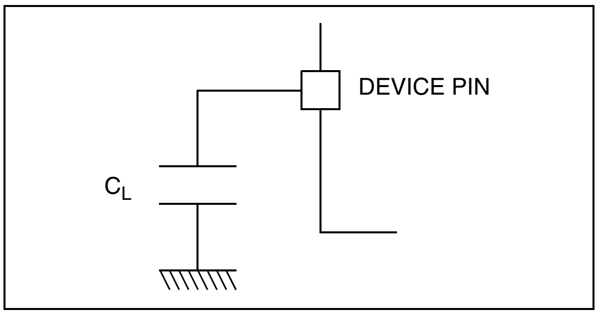
18.1.3 Typical curves

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

18.1.4 Loading capacitor

The loading conditions used for pin parameter measurement are shown in Figure 70.

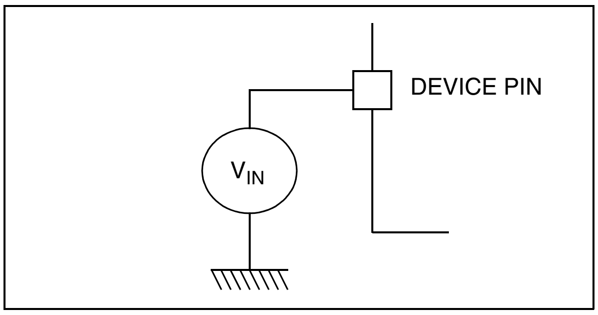
Figure 70. Pin loading conditions



18.1.5 Pin input voltage

The input voltage measurement on a pin of the device is described in Figure 71.

Figure 71. Pin input voltage



18.2 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.

18.2.1 Voltage Characteristics

Symbol	Ratings	Maximum value	Unit
V_{DD33} - V_{SS}	Supply voltage	4.0	V
V_{IN} 1) & 2)	Input voltage on any other pin	V_{SS}-0.3 to V_{DD33}+0.3	V
$V_{ESD(HBM)}$	Electro-static discharge voltage (Human Body Model)	see section 18.7.3 on page 168	
$V_{ESD(MM)}$	Electro-static discharge voltage (Machine Model)		

18.2.2 Current Characteristics

Symbol	Ratings	Maximum value	Unit
I_{VDD33}	Total current into V_{DD33} power lines (source) 3)	200	mA
I_{VSS}	Total current out of V_{SS} ground lines (sink) 3)	200	
I_{IO} (4)	Output current sunk by any I/O D2 type	25	
	Output current sunk by any I/O D4 type	35	
	Output current sunk by any I/O D8 type	50	
	Output current source by any I/Os and control pin	-25	

18.2.3 Thermal Characteristics

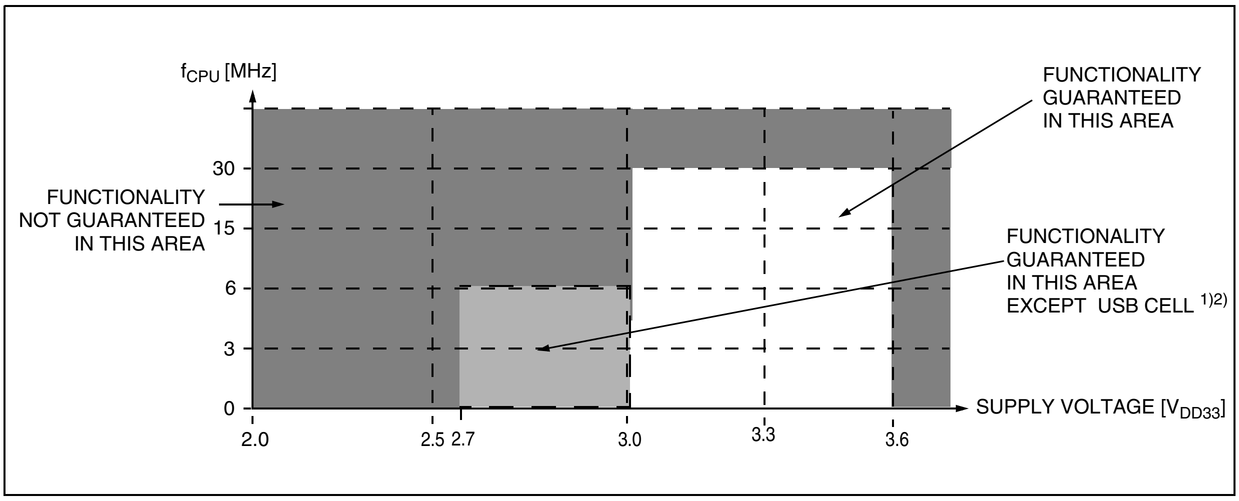
Symbol	Ratings	Value	Unit
T_{STG}	Storage temperature range	-65 to +150	°C
T_{JMAX}	Maximum junction temperature 7)	120	°C

Notes:
1. Directly connecting the $\overline{\text{RESET}}$ and I/O pins to V_{DD33} 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 $\overline{\text{RESET}}$, 10kΩ for I/Os). For the same reason, unused I/O pins must not be directly tied to V_{DD33} or V_{SS}.
2. When the current limitation is not possible, the V_{IN} absolute maximum rating must be respected, otherwise refer to $I_{INJ(PIN)}$ specification. A positive injection is induced by $V_{IN}>V_{DD33}$ while a negative injection is induced by $V_{IN}<V_{SS}$.
3. All power supply (V_{DD33}) and ground (V_{SS}) lines must always be connected to the external supply.
4. Refer to table 6, to know the output drive capability of each I/Os

18.3 OPERATING CONDITIONS

18.3.1 General Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
f_{CPU}	Internal clock frequency		0	30	MHz
V_{DD33}	Power Supply		2.7	3.6	V
T_A	Ambient temperature range		0	70	°C



Notes:

1. USB2 PHY does not function under 3V.
2. Supported by low voltage devices, ST7267C8T1L and ST7267R8T1L

18.4 SUPPLY CURRENT CHARACTERISTICS

The following current consumption specified for the Device 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).

18.4.1 RUN and WAIT Modes

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
I_{DD}	Supply current in RUN mode (see Figure 72)	PLL ON	f_{OSC}=12MHz, f_{CPU}=30MHz	15	25	35	mA
		PLL ON	f_{OSC}=12MHz, f_{CPU}=15MHz	10 [1]	20	30 [1]	
		PLL OFF	f_{OSC}=12MHz, f_{CPU}=6MHz	8 [1]	14	22 [1]	
		PLL OFF	f_{OSC}=12MHz, f_{CPU}=3MHz	7 [1]	12	21 [1]	
	Supply current in WAIT mode (see Figure 73)	PLL ON	f_{OSC}=12MHz, f_{CPU}=6MHz	10 [1]	20	30 [1]	
		PLL OFF	f_{OSC}=12MHz, f_{CPU}=3MHz	6	10	20	

Note:

1. Not tested in production, guaranteed by characterization.

Figure 72. Typical I_{DD} in RUN vs. f_{CPU}

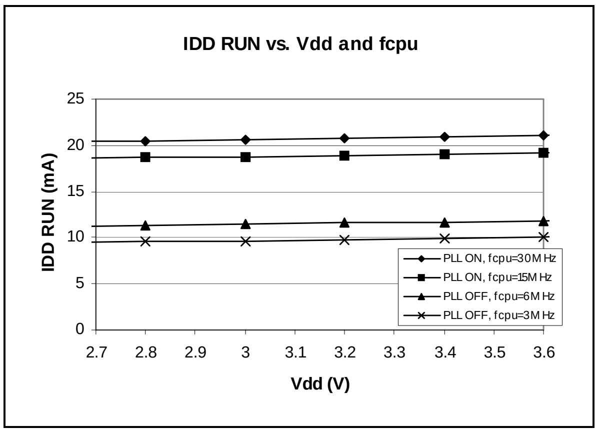
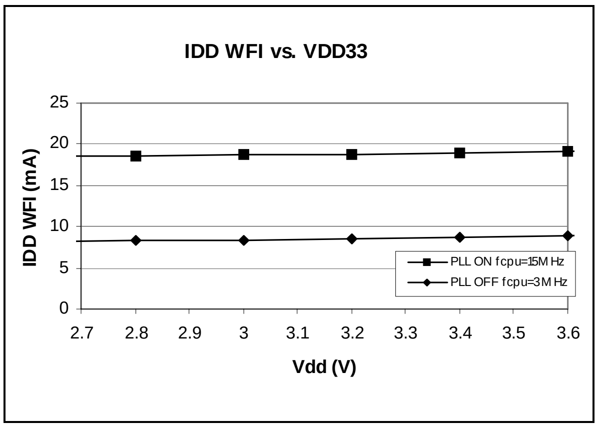


Figure 73. Typical I_{DD} in WAIT vs. f_{CPU}



18.4.2 HALT Modes

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{DD}	Supply current in HALT mode	Regulator and PHY ON	2 [1)]	4	7 [1)]	mA
		Powerdown mode [2)]	60	120	400	µA

Notes:

1. Not tested in production, guaranteed by characterization.
2. In order to reach this value, the software must force the regulator and the PHY into powerdown mode and the IOs compensation cell off.

18.4.3 Supply and Clock Managers

The previous current consumption specified for the Device 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).

Symbol	Parameter	Conditions	Typ [1)]	Max [2)]	Unit
$I_{DD(CK)}$	Supply current of crystal oscillator [3)]		1000	2000	µA

Notes:

1. Typical data are based on T_A=25°C and f_{CPU}=12MHz.
2. Data based on characterization results, not tested in production.
3. Data based on characterization results done with the external components specified in Section 18.5.2, not tested in production.

18.5 CLOCK AND TIMING CHARACTERISTICS

Subject to general operating conditions for V_{DD33}, f_{OSC}, and T_A.

18.5.1 General Timings

Symbol	Parameter	Conditions	Min	Typ [1)]	Max	Unit
$t_{c(INST)}$	Instruction cycle time		2	3	12	t_{CPU}
		f_{CPU}=15MHz	133	200	800	ns
$t_{v(IT)}$	Interrupt reaction time [2)] $t_{v(IT)} = \Delta t_{c(INST)} + 10$		10		22	t_{CPU}
		f_{CPU}=12MHz	0.666		1.466	μs

Notes:

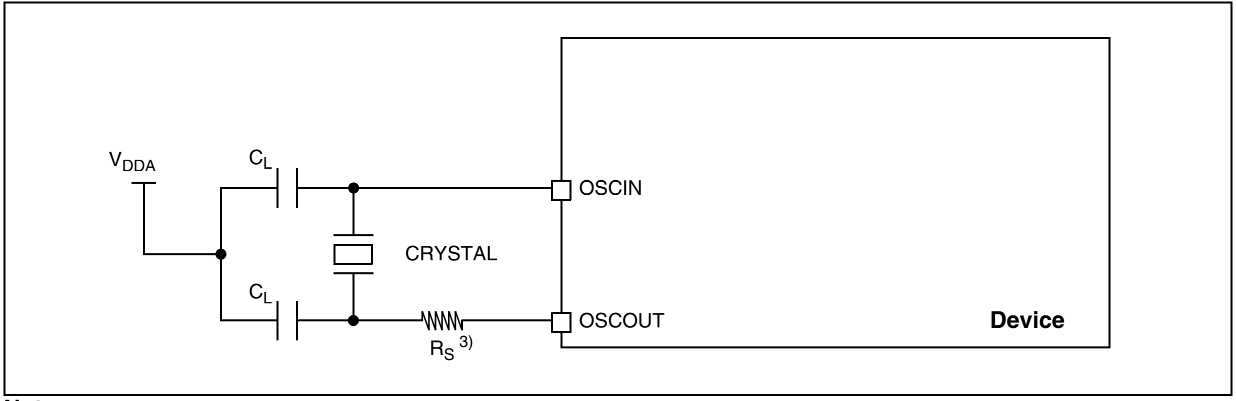
1. Data based on typical application software.
2. Time measured between interrupt event and interrupt vector fetch. $\Delta t_{c(INST)}$ is the number of t_{CPU} cycles needed to finish the current instruction execution.

18.5.2 Crystal Oscillator

The Device internal clock is supplied from a crystal oscillator. All the information given in this paragraph are based on characterization results with specified typical external components. In the application 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 manufacturer for more details (frequency, package, accuracy...).

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{OSC}	Oscillator frequency [1)]			12		MHz
CK_{ACC}	Total crystal oscillator accuracy	abs. value + temp + aging			+/-60	ppm
α_{OSC}	Crystal oscillator duty cycle [2)]		45	50	55	%

Figure 74. Typical Application with a Crystal



Notes:

1. The oscillator selection can be optimized in terms of supply current using an high quality crystal with small R_S value. Refer to the crystal manufacturer characteristics for more details.
2. The crystal oscillator duty cycle has to be adjusted through the two C_L capacitors. Refer to the crystal manufacturer for more details.
3. Depending on the crystal power dissipation, a serial resistor R_S may be added. Refer to the crystal manufacturer for more details.

Table 36. Typical C_L and R_S Values by Crystal

Supplier	Typical Crystal	C_L (pF)	R_S (Ω)
NDK	AT51 or AT41	16	560

18.6 MEMORY CHARACTERISTICS

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

18.6.1 RAM and Hardware Registers

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{RM}	Data retention mode 1)	HALT mode (or RESET)	1.4			V

Note:

1. Minimum V_{18_DIG} 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.

18.7 EMC CHARACTERISTICS

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

18.7.1 Functional EMS (Electro Magnetic Susceptibility)

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

- **ESD**: Electro-Static Discharge (positive and negative) is applied on all pins of the device until a functional disturbance occurs. This test conforms with the IEC 1000-4-2 standard.
- **FTB**: A Burst of Fast Transient voltage (positive and negative) is applied to V_{DD33} and V_{SS33} 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.

18.7.1.1 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	For TQFP64 (10x10), V_{DD33}=3.3V, T_A=+25°C, f_{OSC}=12MHz conforms to IEC 1000-4-2	4B
V_{FFTB}	Fast transient voltage burst limits to be applied through 100pF on V_{DD33} and V_{SS33} pins to induce a functional disturbance	For TQFP64 (10x10), V_{DD33}=3.3V, T_A=+25°C, f_{OSC}=12MHz conforms to IEC 1000-4-4	4A

18.7.2 Electro Magnetic Interference (EMI)

Based on a simple application running on the product (toggling 2 LEDs through the I/O ports), the product is monitored in terms of emission. This emission test is in line with the norm SAE J 1752/3 which specifies the board and the loading of each pin.

Symbol	Parameter	Conditions	Monitored Frequency Band	Max vs. [f_{OSC}/f_{CPU}]		Unit
				12/15MHz	12/30MHz	
S_{EMI}	Peak level	V_{DD33}=3.3V, T_A=+25°C, TQFP64 10x10 package conforming to SAE J 1752/3 **Note:** Refer to Application Note AN1709 for data on other package types.	0.1MHz to 30MHz	16	20	dBµV
			30MHz to 130MHz	21	25	dBµV
			130MHz to 1GHz	27	25	dBµV
			SAE EMI Level	4	4	-

Notes:

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

EMC CHARACTERISTICS (Cont'd)

18.7.3 Absolute Maximum Ratings (Electrical Sensitivity)

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

18.7.3.1 Electro-Static Discharge (ESD)

Electro-Static Discharges (a positive then a negative pulse separated by 1 second) are applied to the pins of each sample according to each pin combination. The sample size depends on the number of supply pins in the device (3 parts*(n+1) supply pin). This test conforms to the JESD22-A114A/A115A standard.

Absolute Maximum Ratings

Symbol	Ratings	Conditions	Maximum value [1)]	Unit
$V_{ESD(HBM)}$	Electro-static discharge voltage (Human Body Model)	T_A=+25°C	2000	V

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

18.7.3.2 Static and Dynamic Latch-Up

- **LU**: 3 complementary static tests are required on 10 parts to assess the latch-up performance. A supply overvoltage (applied to each power supply pin) and a current injection (applied to each input, output and configurable I/O pin) are performed on each sample. This test conforms to the EIA/JESD 78 IC latch-up standard. For more details, refer to the application note AN1181.
- **DLU**: Electro-Static Discharges (one positive then one negative test) are applied to each pin of 3 samples when the micro is running to assess the latch-up performance in dynamic mode. Power supplies are set to the typical values, the oscillator is connected as near as possible to the pins of the micro and the component is put in reset mode. This test conforms to the IEC1000-4-2 and SAEJ1752/3 standards. For more details, refer to the application note AN1181.

Electrical Sensitivities

Symbol	Parameter	Conditions	Class [1)]
LU	Static latch-up class	T_A=+25°C	A
DLU	Dynamic latch-up class	V_{DD33}=3.3V, f_{OSC}=12MHz, T_A=+25°C	A

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

18.8 I/O PORT PIN CHARACTERISTICS

18.8.1 General Characteristics

Subject to general operating conditions for VDD33, f_{OSC}, and T_A unless otherwise specified.

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{IL}	Input low level voltage	TTL ports				$0.16xV_{DD33}$	V
V_{IH}	Input high level voltage			$0.85xV_{DD33}$			
V_{hys}	Schmitt trigger voltage hysteresis 1)			400			mV
I_L	Input leakage current	$V_{SS} \leq V_{IN} \leq$VDD33, standard I/Os				1	μA
I_{L5V}	5V tolerant input leakage current	$V_{SS} \leq V_{IN} \leq$VDD33				10	
		V_{IN}=5V, 25°C			30		
R_{PU}	Weak pull-up equivalent resistor 2)	$V_{IN}=V_{SS}$	VDD33=3.3 V	32	50	75	kΩ
$C_{(IOD2)}$	I/O pin capacitance 2mA drive				1.5		pF
$C_{(IOD2)}$	I/O pin capacitance 2mA drive, 5V tolerant				1.7		pF
$C_{(IOD4)}$	I/O pin capacitance 4mA drive				1.9		pF
$C_{(IOD8)}$	I/O pin capacitance 8mA drive				2.7		pF
$t_{f(IOD2)out}$	Output high to low level fall time 3)	C_L=50pF Between 10% and 90%				10	ns
$t_{r(IOD2)out}$	Output low to high level rise time 3)					10	
$t_{f(IOD4)out}$	Output high to low level fall time 3)					6	ns
$t_{r(IOD4)out}$	Output low to high level rise time 3)					6	
$t_{f(IOD8)out}$	Output high to low level fall time 3)					3.5	
$t_{r(IOD8)out}$	Output low to high level rise time 3)					3.5	
$t_{w(IT)in}$	External interrupt pulse time 4)			1			t_{CPU}

Notes:

1. Hysteresis voltage between Schmitt trigger switching levels. Based on characterization results, not tested in production.

2. The R_{PU} pull-up equivalent resistor is based on a resistive transistor. This data is based on characterization results, tested in production at V_{DD33} max.

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

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 75. Typical V_{IL} and V_{IH} standard I/Os

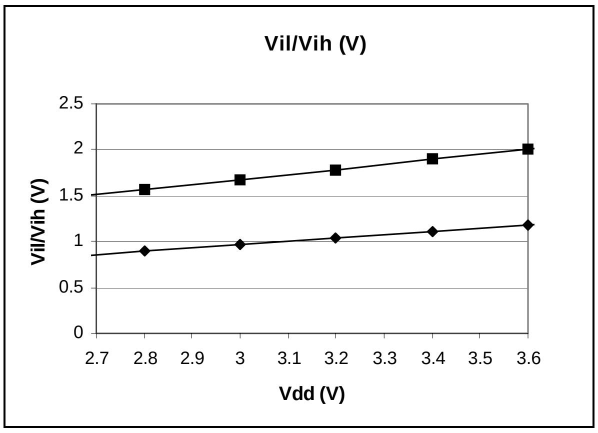
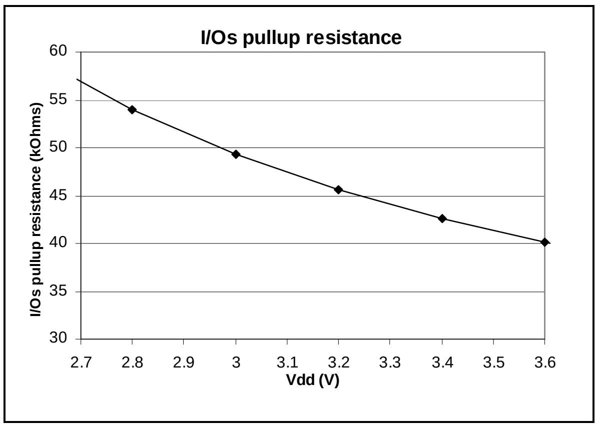
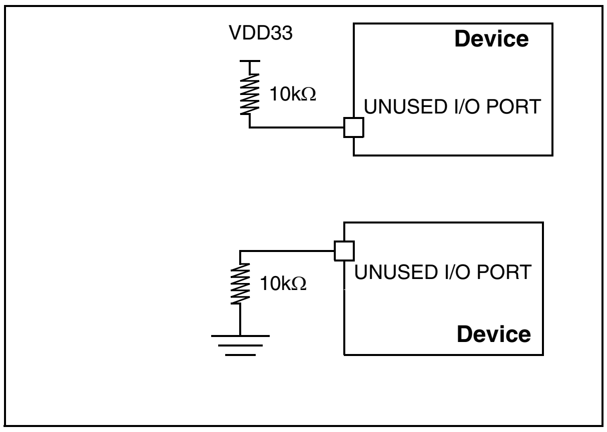


Figure 76. Typical R_{PU} vs. V_{DD33} with V_{IN}=V_{SS}



I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 77. Two typical Applications with unused I/O Pin



I/O PORT PIN CHARACTERISTICS (Cont'd)

18.8.2 Output Driving Current

Subject to general operating conditions for V_{DD33}, f_{OSC}, and T_A unless otherwise specified.

Symbol	Parameter	Conditions		Min	Max	Unit
V_{OL} 1)	Output low level voltage for a D2 I/O pin when 8 pins are sunk at same time (see Figure 78)	V_{DD33}=3.3V	I_{IO}=2mA		300	mV
	Output low level voltage for a D4 I/O pin when 8 pins are sunk at same time (see Figure 79)		I_{IO}=4mA		400	
	Output low level voltage for a D8 I/O pin when 8 pins are sunk at same time (see Figure 80)		I_{IO}=8mA		500	
V_{DD33}-V_{OH} 2)	Output high level voltage for a D2 I/O pin when 8 pins are sourced at same time (see and Figure 81)		I_{IO}=2mA		600	mV
	Output high level voltage for a D4 I/O pin when 8 pins are sourced at same time (see Figure 82)		I_{IO}=4mA		600	
	Output high level voltage for a D8 I/O pin when 8 pins are sourced at same time (see Figure 83)		I_{IO}=8mA		600	

Notes:

1. The I_{IO} current sunk must always respect the absolute maximum rating specified in Section 18.2.2 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 18.2.2 and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VDD33}. True open drain I/O pins does not have V_{OH}.

Figure 78. Typical V_{OL} at V_{DD33}=3.3V (I/O D2)

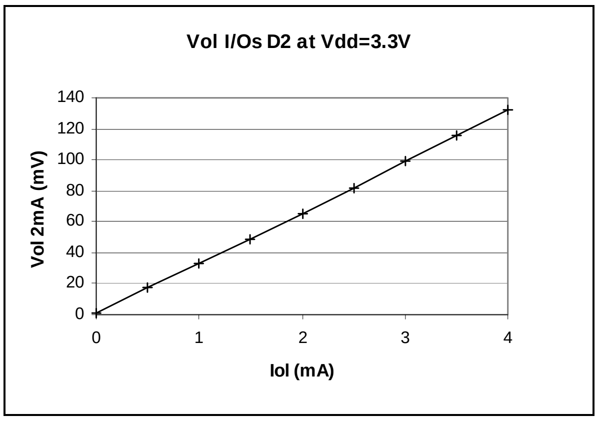


Figure 79. Typical V_{OL} at V_{DD33}=3.3V (I/O D4)

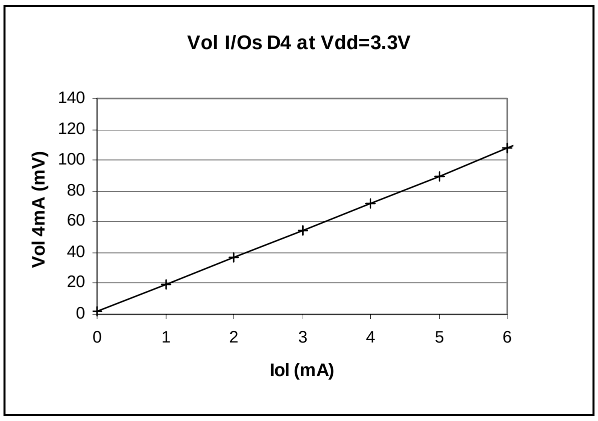


Figure 80. Typical V_{OL} at V_{DD33}=3.3V (I/O D8)

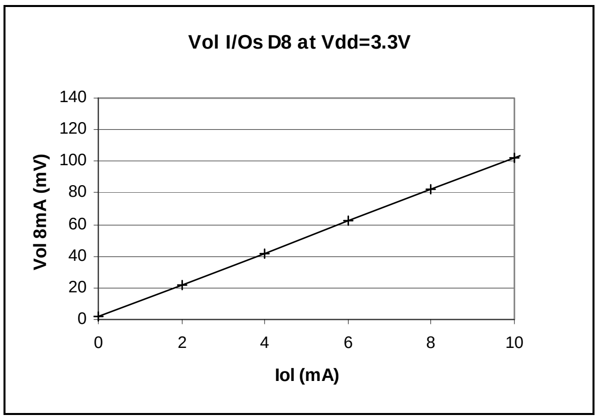


Figure 81. Typical V_{DD33}-V_{OH} vs. V_{DD33} (IO D2)

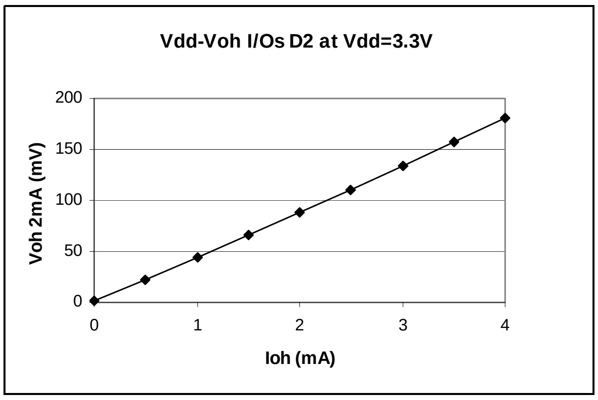


Figure 82. Typical V_{DD33}-V_{OH} vs. V_{DD33} (IO D4)

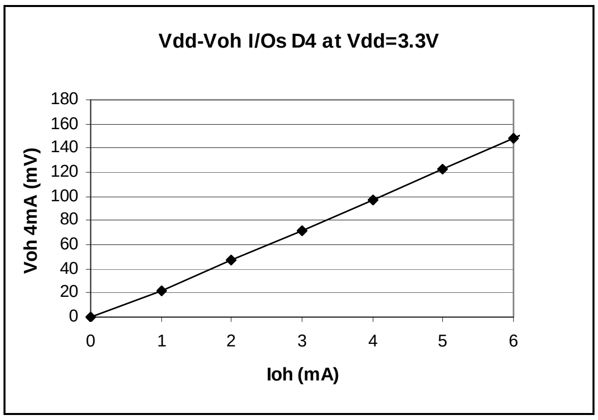
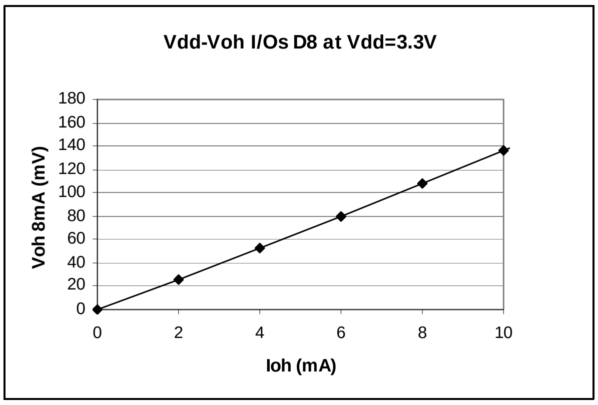


Figure 83. Typical V_{DD33}-V_{OH} vs. V_{DD33} (IO D8)



18.9 CONTROL PIN CHARACTERISTICS

18.9.1 Asynchronous $\overline{\text{RESET}}$ Pin

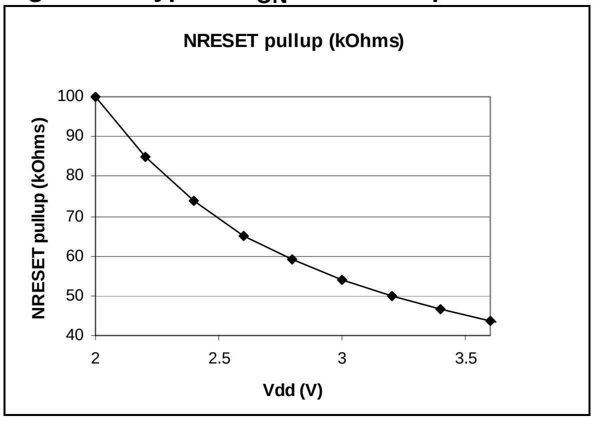
T_A = 0 to +55°C unless otherwise specified

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IL}	Input low level voltage [1]				$0.16xV_{DD33}$	V
V_{IH}	Input high level voltage		$0.85xV_{DD33}$			
V_{hys}	Schmitt trigger voltage hysteresis[1]			450		mV
R_{ON}	Pull-up equivalent resistor	V_{DD33}=3.3V	20	40	80	kΩ
		V_{DD33}=2V		100		
$t_{eh(RSTL)}$	External reset pulse hold time [2]		2.5			µs
$t_{g(RSTL)}$	Filtered glitch duration [3]			200		ns
$t_{ew(RSTL)}$	External reset pulse duration [4]		500			µs
$t_{iw(RSTL)}$	Internal reset pulse duration			2		Tcpu

Notes:

1. The user must ensure that the level on the $\overline{\text{RESET}}$ pin can go below the V_{IL} max. level specified in Section 18.9.1. Otherwise the reset will not be taken into account internally.
2. To guarantee the reset of the Device, a minimum pulse has to be applied to the $\overline{\text{RESET}}$ pin. All short pulses applied on $\overline{\text{RESET}}$ pin with a duration below $t_{eh(RSTL)}$ can be ignored. Not tested in production, guaranted by design.
3. The reset network protects the device against parasitic resets.
4. The user must ensure that external reset duration respect this timing to guarantee a correct start-up of the internal regulator at power-up. Not tested in production, guaranted by design.

Figure 84. Typical R_{ON} on $\overline{\text{RESET}}$ pin



18.10 TIMER PERIPHERAL CHARACTERISTICS

Subject to general operating conditions for V_{DD33}, f_{OSC}, and T_A unless otherwise specified.

Refer to I/O port characteristics for more details on the input/output alternate function characteristics (output compare, input capture, external clock, PWM output...).

18.10.1 Watchdog Timer

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{w(WDG)}$	Watchdog time-out duration		101 372		6 487 808	t_{CPU}
		f_{CPU} = 3MHz	33.79		2162.58	ms
		f_{CPU} = 6MHz	16.896		1081.32	
		f_{CPU} = 15MHz	6.758		432.54	
		f_{CPU} = 30MHz	3.379		216.24	

18.10.2 Time Base Unit Timer

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{w(TBU)}$	TBU time-out duration	Standalone mode	2		262144	t_{CPU}
		f_{CPU} = 3MHz	0.666		87380	us
		f_{CPU} = 6MHz	0.333		43691	
		f_{CPU} = 15MHz	0.133		17477	
		f_{CPU} = 30MHz	0.066		8737	

18.10.3 16-Bit Timer

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{w(ICAP)in}$	Input capture pulse time		1			t_{CPU}
$t_{res(PWM)}$	PWM resolution time		2			t_{CPU}
		f_{CPU} = 3MHz	666.67			ns
		f_{CPU} = 6MHz	333.33			
		f_{CPU} = 15MHz	133.33			
		f_{CPU} = 30MHz	66.67			
f_{EXT}	Timer external clock frequency		0		$f_{CPU}/4$	MHz
f_{PWM}	PWM repetition rate		0		$f_{CPU}/4$	MHz
Res_{PWM}	PWM resolution				16	bit

18.11 OTHER COMMUNICATION INTERFACE CHARACTERISTICS

18.11.1 MSCI Parallel Interface

Figure 85. Timing diagrams for input mode (with max load on CTRL signal=50pf)

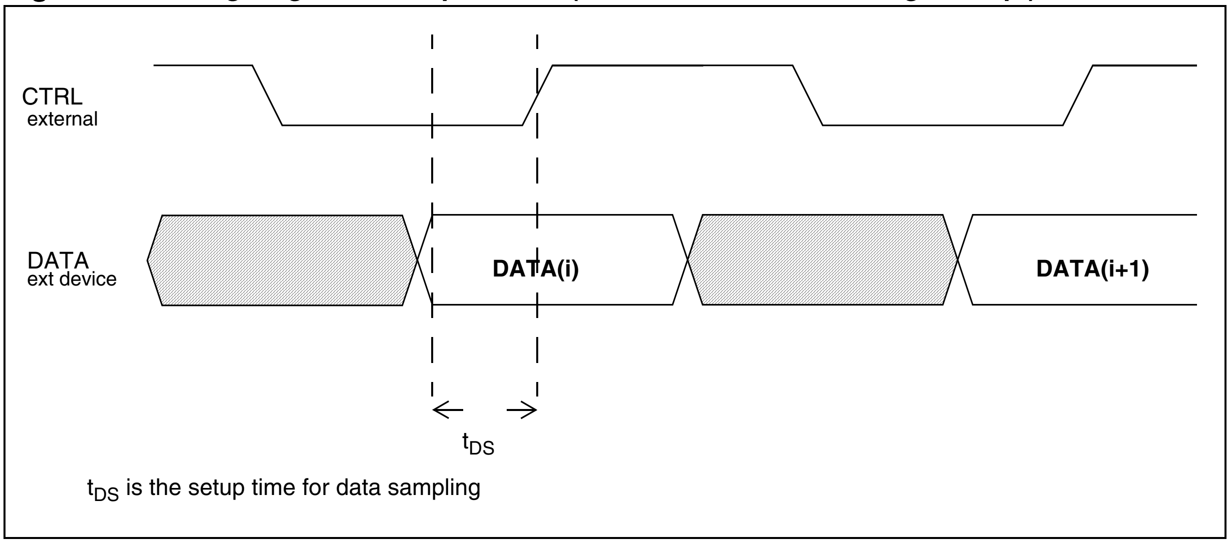


Figure 86. Timing diagrams for output mode (with max CTRL signal=50pf, DATA)

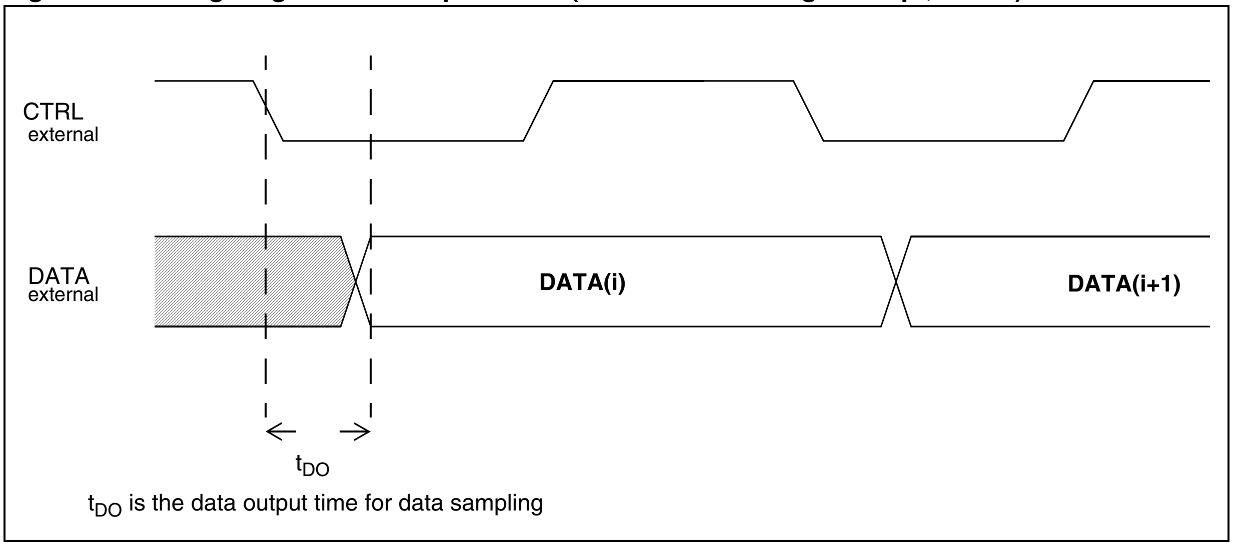


Table 37. MSCI Parallel Interface: DC Characteristics

MSCI DC Electrical Characteristics						
Parameter	**Symbol**	**Conditions**	**Min.**	**Typ [1]**	**Max.**	**Unit**
Data Setup Time	t_{DS}			11		ns
Data Output time	t_{DO}			6		ns
CTRL line capacitance	Cctrl			50		pF
Data line capacitance	Cdata			50		pF

Notes:
1. Data based on design simulation and not tested in production.

18.11.2 USB (Universal Bus Interface)

Table 38. USB Interface: DC Characteristics

USB DC Electrical Characteristics						
Symbol	**Parameter**	**Conditions**	**Min.**	**Typ.**	**Max.**	**Unit**
$I_{DDsuspend}$	Suspend current	V_{DD33}=3.3V, regulator and PHY ON	0.5 [1]	1.5	6 [1]	mA
		V_{DD33}=3.3V, Powerdown mode, 25°C [2]	60	90	190	uA
R_{PU}	Pull-up resistor [1]			1.5		kΩ
Full Speed Mode						
V_{TERM}	Termination Voltage		0.8		2.0	V
VOH	High Level Output Voltage		2.8		3.6	V
VOL	Low Level Output Voltage				0.8	V
V_{CRS}	Crossover Voltage		1.3		2.0	V
High Speed Mode						
V_{HSOH}	HS Data Signalling High			400		mV
V_{HSOL}	HS Data Signalling Low			5		mV

Notes:
1. Not tested in production, guaranteed by characterization.
2. In order to reach this value, the software must force the regulator into powerdown mode and the IOs compensation cell off.

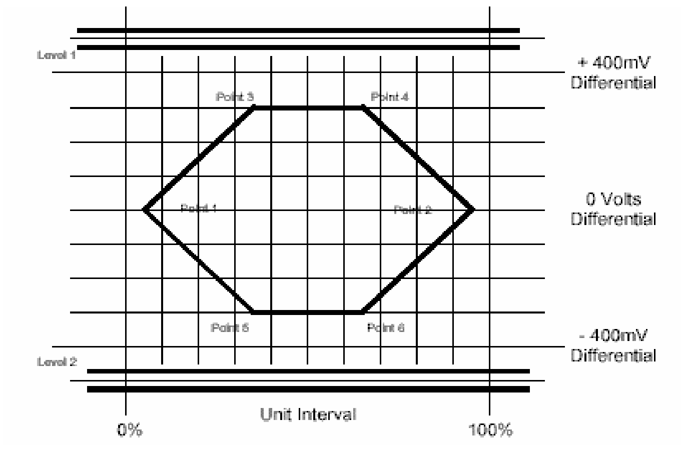
Table 39. USB Interface: Timing

USB DC Electrical Characteristics					
Symbol	**Parameter**	**Conditions**	**Min.**	**Max.**	**Unit**
Full Speed Mode					
T_{FR}	Rise Time	C_L=50pF	4	20	ns
T_{FF}	Fall Time	C_L=50pF	4	20	ns
High Speed Mode					
T_{HSR}	Rise Time			500 1)	ps
T_{HSF}	Fall Time			500 1)	ps
T_{HSDRAT}	HS Data Rate		479.76	480.24	Mb/s

Notes:
1. Not tested in production, guaranteed by characterization.

Table 40. USB High Speed Transmit Waveform requirements

	Voltage Level (DP - DN)	Time
Unit Interval (UI)	-	2.082 to 2.084 ns
Level 1	475 mV	-
Level 2	-475 mV	-
Point 1	0V	5% UI
Point 2	0V	95% UI
Point 3	300 mV	35% UI
Point 4	300 mV	65% UI
Point 5	-300 mV	35% UI
Point 6	-300 mV	65% UI



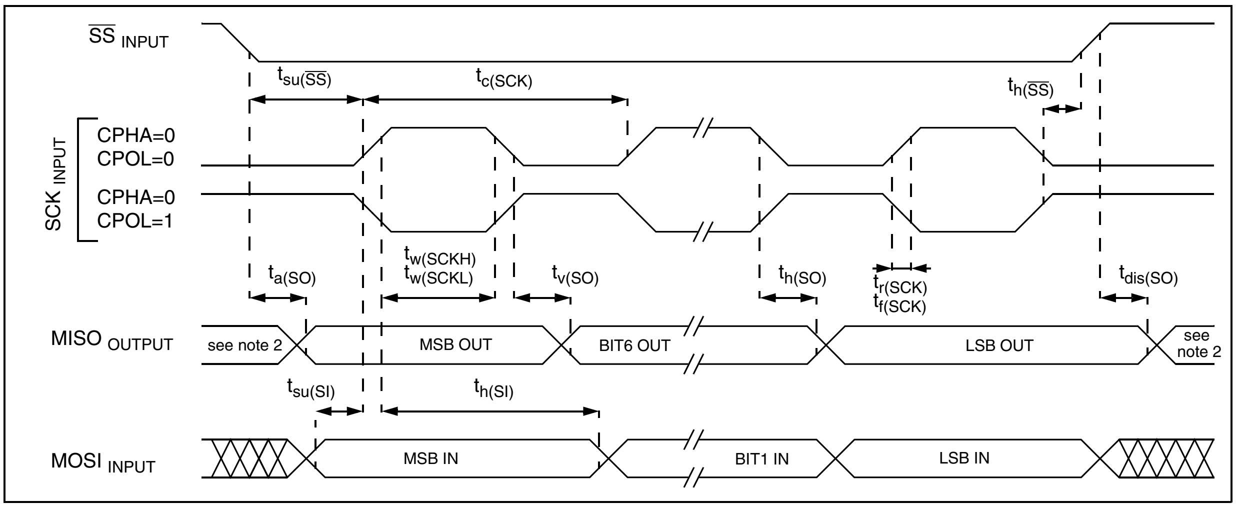
18.11.3 SPI - Serial Peripheral Interface

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

Refer to I/O port characteristics for more details on the input/output alternate function characteristics ($\overline{SS}$, SCK, MOSI, MISO).

Symbol	Parameter	Conditions	Min	Max	Unit
f_{SCK} $1/t_{c(SCK)}$	SPI clock frequency	Master f_{CPU}=6MHz	$f_{CPU}/64$ 0.0937	$f_{CPU}/2$ 3	MHz
		Slave f_{CPU}=6MHz	0	$f_{CPU}/2$ 3	
$t_{r(SCK)}$ $t_{f(SCK)}$	SPI clock rise and fall time		see I/O port pin description		
$t_{su(\overline{SS})}$	$\overline{SS}$ setup time	Slave	160		ns
$t_{h(\overline{SS})}$	$\overline{SS}$ hold time	Slave	160		
$t_{w(SCKH)}$ $t_{w(SCKL)}$	SCK high and low time	Master Slave	130 120		
$t_{su(MI)}$ $t_{su(SI)}$	Data input setup time	Master Slave	130 130		
$t_{h(MI)}$ $t_{h(SI)}$	Data input hold time	Master Slave	130 130		
$t_{a(SO)}$	Data output access time	Slave	0	160	
$t_{dis(SO)}$	Data output disable time	Slave		320	
$t_{v(SO)}$	Data output valid time	Slave (after enable edge)		160	
$t_{h(SO)}$	Data output hold time		0		
$t_{v(MO)}$	Data output valid time	Master (before capture edge)	0.25		t_{CPU}
$t_{h(MO)}$	Data output hold time		0.25		

Figure 87. SPI Slave Timing Diagram with CPHA=0 3)



Notes:
1. Data based on design simulation, not tested in production.
2. When no communication is on-going the data output line of the SPI (MOSI in master mode, MISO in slave mode) has its alternate function capability released. In this case, the pin status depends on the I/O port configuration.
3. Measurement points are done at CMOS levels: $0.3xV_{DD33}$ and $0.7xV_{DD33}$.

COMMUNICATION INTERFACE CHARACTERISTICS (Cont'd)

Figure 88. SPI Slave Timing Diagram with CPHA=1[1)]

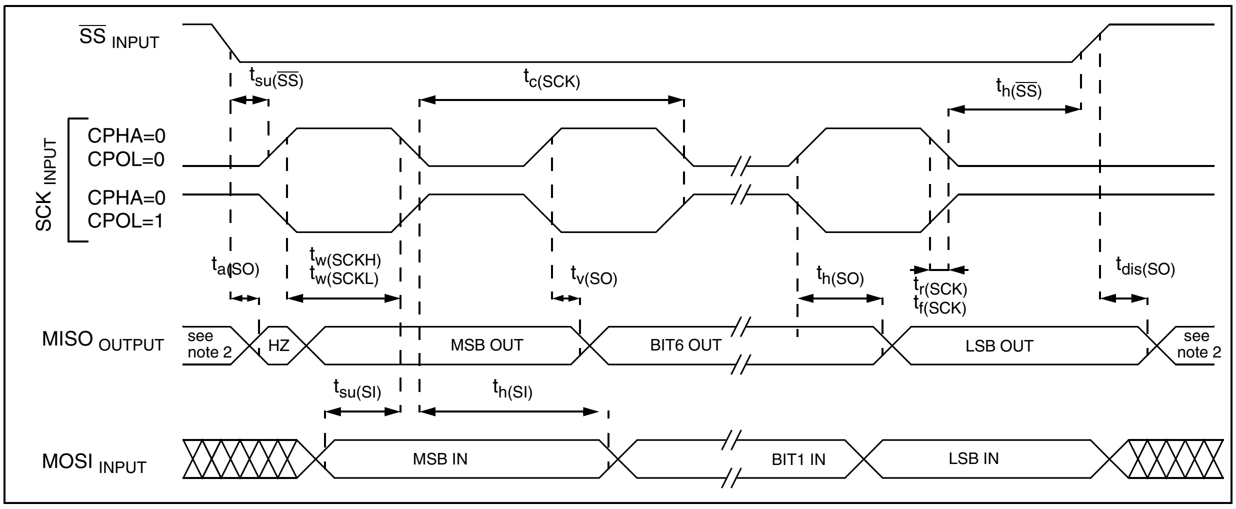
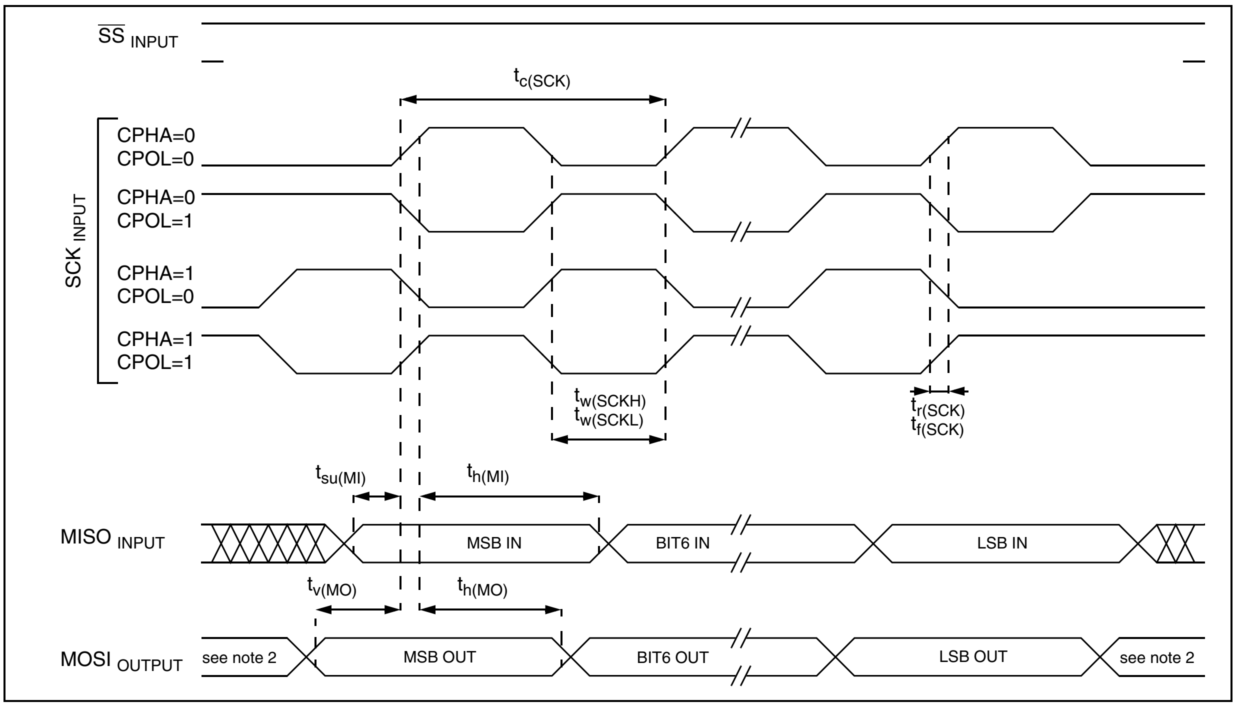


Figure 89. SPI Master Timing Diagram [1)]



Notes:

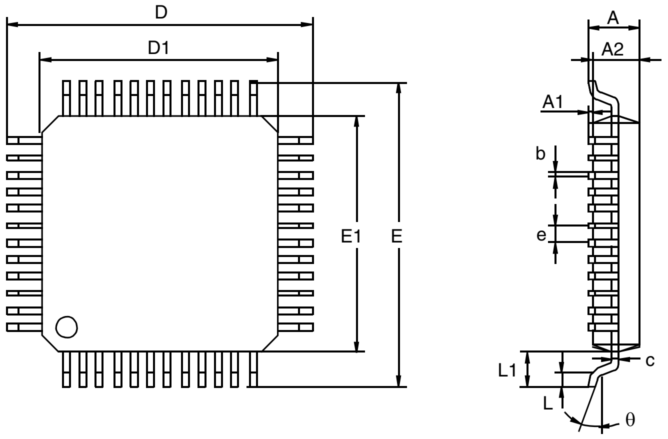
1. Measurement points are done at CMOS levels: $0.3xV_{DD33}$ and $0.7xV_{DD33}$.
2. When no communication is on-going the data output line of the SPI (MOSI in master mode, MISO in slave mode) has its alternate function capability released. In this case, the pin status depends of the I/O port configuration.

19 PACKAGE CHARACTERISTICS

19.1 PACKAGE MECHANICAL DATA

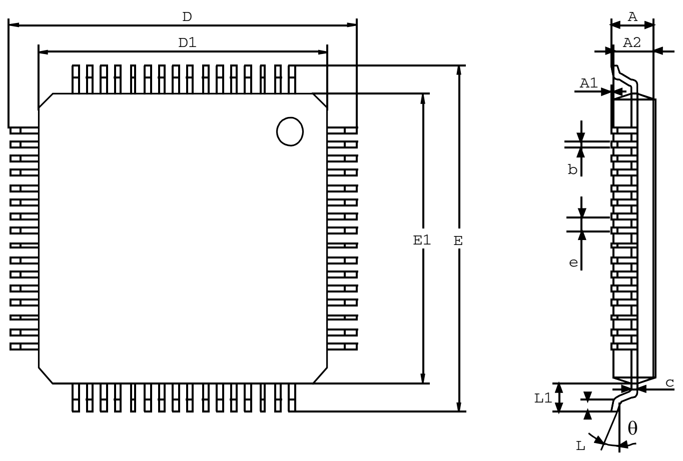
In order to meet environmental requirements, ST offers these devices in ECOPACK® packages. These packages have a Lead-free second level interconnect . The category of second level interconnect is marked on the package and on the inner box label, in compliance with JEDEC Standard JESD97. The maximum ratings 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, with specific Application Notes covering the main technical aspects related to lead-free conversion (AN2033, AN2034, AN2035 and AN2036).

Figure 90. 48-Pin Thin Quad Flat Package



Dim.	mm			inches		
	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					

Figure 91. 64-Pin Thin Quad Flat Package (10 x10)



Dim.	mm			inches		
	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		12.00			0.472	
D1		10.00			0.394	
E		12.00			0.472	
E1		10.00			0.394	
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	64					

19.2 THERMAL CHARACTERISTICS

Symbol	Ratings	Value	Unit
R_{thJA}	Package thermal resistance (junction to ambient) TQFP48 TQFP64	 70 55	°C/W
P_D	Power dissipation [1)]	400	mW

Notes:

1. The power dissipation is obtained from the formula $P_D=P_{INT}+P_{PORT}$ where P_{INT} is the Device internal power ($I_{DD}xV_{DD}$) and P_{PORT} is the port power dissipation determined by the user.

20 DEVICE CONFIGURATION AND ORDERING INFORMATION

20.1 OPTION BYTE

The option byte allows the hardware configuration of the microcontroller to be selected. In masked ROM devices, the option bytes are fixed in hardware by the ROM code (see option list).

7							0
-	-	-	-	-	-	WDGH-WR	-

OPT1 = WDGHWR *Hardware Watchdog Reset*
This option permanently enables the watchdog reset.

0: Hardware Watchdog is disabled (watchdog to be enabled by software).

1: Hardware Watchdog is activated (watchdog always enabled)

.

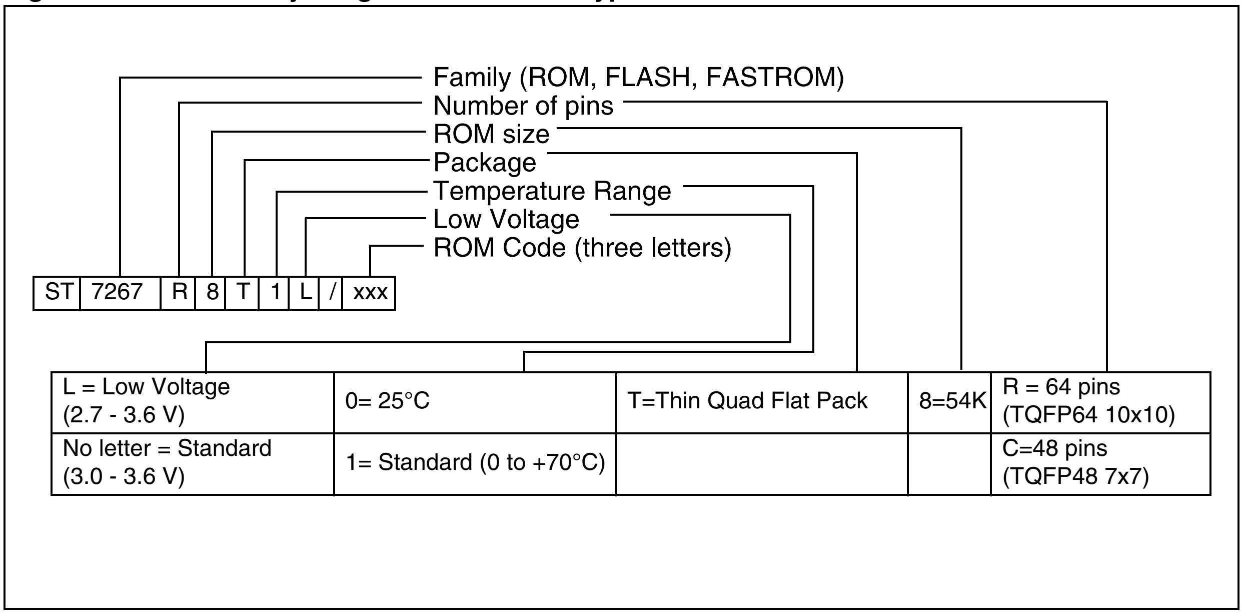
21 DEVICE ORDERING INFORMATION AND TRANSFER OF CUSTOMER CODE

Customer code is made up of the ROM contents and the list of the selected options (if any). The ROM contents are to be sent on diskette, or by electronic means, with the S19 hexadecimal file generated by the development tool. All unused bytes must be set to FFh.

The selected options are communicated to STMicroelectronics using the correctly completed OPTION LIST appended.

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

Figure 92. ROM Factory-Programmed Device Types



L = Low Voltage (2.7 - 3.6 V)	0= 25°C	T=Thin Quad Flat Pack	8=54K	R = 64 pins (TQFP64 10x10)
No letter = Standard (3.0 - 3.6 V)	1= Standard (0 to +70°C)			C=48 pins (TQFP48 7x7)

DEVICE OPTION LIST

ST7267

Customer: . . . .
Address: . . . .
. . . .
Contact: . . . .
Phone No: . . . .
Reference: . . . .
FASTROM code must be sent in .S19 format. .Hex extension cannot be processed.

STMicroelectronics references:

Device Type/Memory Size/Package (check only one option):

ROM DEVICE:	STANDARD VOLTAGE	LOW VOLTAGE
TQFP48:	[ ] ST7267C8T1	[ ] ST7267C8T1L
TQFP64:	[ ] ST7267R8T1	[ ] ST7267R8T1L

Conditioning (check only one option):

Packaged Product	Die Product (dice tested at 25°C only)
[ ] Tape & Reel	[ ] Sawn wafer on sticky foil
[ ] Tray	

Special Marking: [ ] No [ ] Yes "_ _ _ _ _ _ _ _ _ _ _ _ _"
Authorized characters are letters, digits, '.', '-', '/' and spaces only.
For marking, two lines are possible with a maximum of 8 characters per line.

Watchdog Reset: WDGHWR [ ] Software activation [ ] Hardware activation

Date . . . .
Signature . . . .

22 REVISION HISTORY

Table 41. Revision History

Date	Revision	Description of Changes
4-Oct-2005	1	First Release
14-Nov-2005	2	Changed status of the document: “Preliminary Data” removed

Notes: