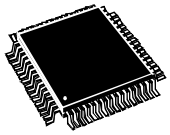
ST7265x

LOW-POWER, FULL-SPEED USB 8-BIT MCU WITH 32K FLASH, 5K RAM, FLASH CARD I/F, TIMER, PWM, ADC, I^2C, SPI

PRELIMINARY DATA

- **Memories**
  - Up to 32K of ROM or High Density Flash (HDFlash) program memory with read/write protection
  - For HDFlash devices, In-Application Programming (IAP) via USB and In-Circuit programming (ICP)
  - Up to 5 Kbytes of RAM with up to 256 bytes stack
- **Clock, Reset and Supply Management**
  - PLL for generating 48 MHz USB clock using a 12 MHz crystal
  - Low Voltage Reset (except on E suffix devices)
  - Dual supply management: analog voltage detector on the USB power line to enable smart power switching from USB power to battery (on E suffix devices).
  - Programmable Internal Voltage Regulator for Memory cards (2.8V to 3.5V) supplying:
    Flash Card I/O lines (voltage shifting)
    Up to 50 mA for Flash card supply
  - Clock-out capability
- **47 programmable I/O lines**
  - 15 high sink I/Os (8mA @0.6V / 20mA@1.3V)
  - 5 true open drain outputs
  - 24 lines programmable as interrupt inputs
- **USB (Universal Serial Bus) Interface**
  - with DMA for full speed bulk applications compliant with USB 12 Mbs specification (version 2.0 compliant)
  - On-Chip 3.3V USB voltage regulator and transceivers with software power-down
  - 5 USB endpoints:
    1 control endpoint
    2 IN endpoints supporting interrupt and bulk
    2 OUT endpoints supporting interrupt and bulk
  - Hardware conversion between USB bulk packets and 512-byte blocks



TQFP64 10x10

- **Mass Storage Interface**
  - DTC (Data Transfer Coprocessor): Universal Serial/Parallel communications interface, with software plug-ins for current and future protocol standards:
    Compact Flash
    Multimedia Card
    Secure Digital Card
    SmartMediaCard
    Sony Memory Stick
    NAND Flash
    ATA Peripherals
- **2 Timers**
  - Configurable Watchdog for system reliability
  - 16-bit Timer with 2 output compare functions.
- **2 Communications Interfaces**
  - SPI synchronous serial interface
  - I^2C Single Master Interface up to 400 KHz
- **D/A and A/D Peripherals**
  - PWM/BRM Generator (with 2 10-bit PWM/BRM outputs)
  - 8-bit A/D Converter (ADC) with 8 channels
- **Instruction Set**
  - 8-bit data manipulation
  - 63 basic instructions
  - 17 main addressing modes
  - 8 x 8 unsigned multiply instruction
  - True bit manipulation
- **Development Tools**
  - Full hardware/software development package

Device Summary

Features	ST72651	ST72F651	ST72652	ST72F652
Program memory	32K ROM	32K FLASH	16K ROM	16K FLASH
User RAM (stack) - bytes	5K (256)		512 (256)	
Peripherals	USB, DTC, Timer, ADC, SPI, I^2C, PWM, WDT		USB, DTC, WDT	
Operating Supply	Dual 2.7V to 5.5V or 4.0V to 5.5V (for USB)		Single 4.0V to 5.5V	
Package	TQFP64 (10 x10)			
Operating Temperature	0°C to +70°C			

Table of Contents

Table of Contents

1 INTRODUCTION

The ST7265x MCU supports volume data exchange with a host (computer or kiosk) via a full speed USB interface. The MCU is capable of handling various transfer protocols, with a particular emphasis on mass storage applications.

ST7265x is compliant with the USB Mass Storage Class specifications, and supports related protocols such as BOT (Bulk Only Transfer) and CBI (Control, Bulk, Interrupt).

It is based on the ST7 standard 8-bit core, with specific peripherals for managing USB full speed data transfer between the host and most types of FLASH media card:

– A full speed USB interface with Serial Interface Engine, and on-chip 3.3V regulator and transceivers.

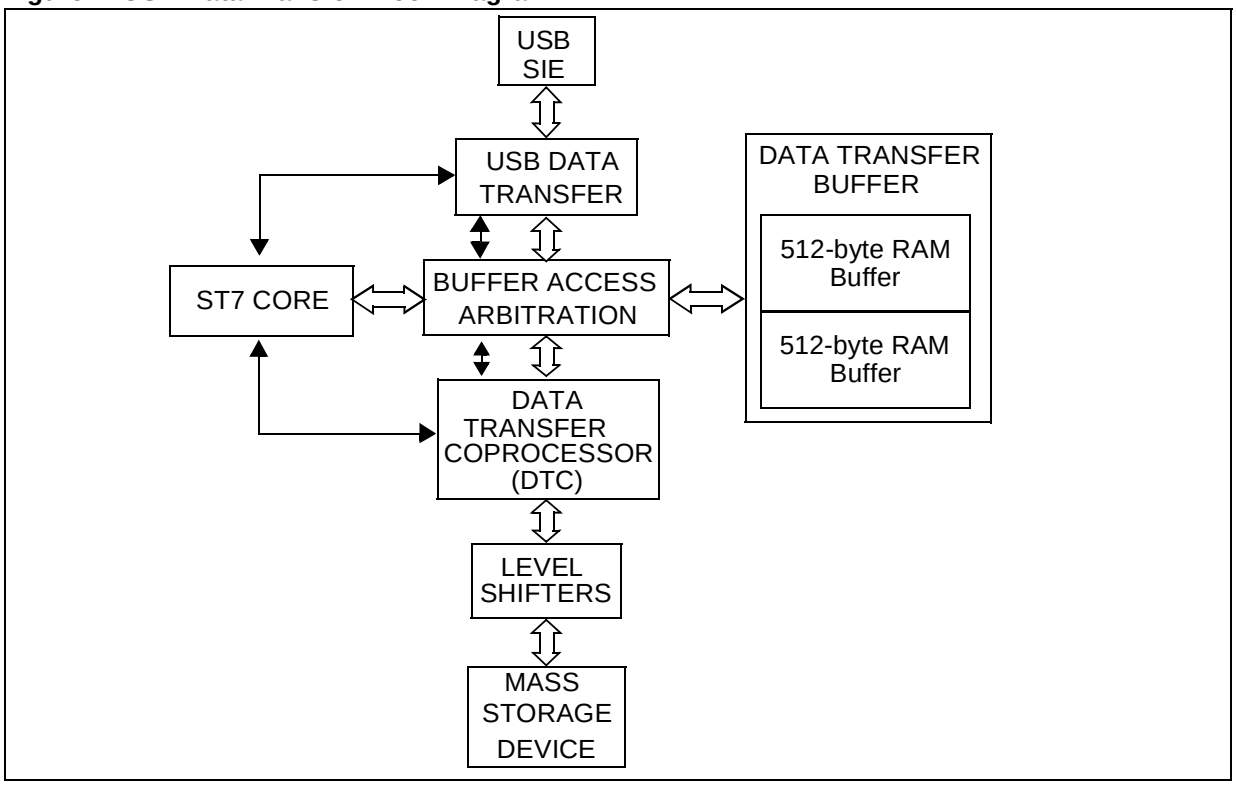
– A dedicated 24 MHz Data Buffer Manager state machine for handling 512-byte data blocks (this size corresponds to a sector both on computers and FLASH media cards).

– A Data Transfer Coprocessor (DTC), able to handle fast data transfer with external devices. This DTC also computes the CRC or ECC required to handle Mass storage media.

– An Arbitration block gives the ST7 core priority over the USB and DTC when accessing the Data Buffer. In USB mode, the USB interface is serviced before the DTC.

– A FLASH Supply Block able to provide programmable supply voltage and I/O electrical levels to the FLASH media.

Figure 1. USB Data Transfer Block Diagram



INTRODUCTION (Cont'd)

In addition to the peripherals for USB full speed data transfer, the ST7265x includes all the necessary features for stand-alone applications with FLASH mass storage.

– Low voltage reset ensuring proper power-on or power-off of the device (not on all products)

– Digital Watchdog

– 16-bit Timer with 2 output compare functions (not on all products - see device summary).

– Two 10-bit PWM outputs (not on all products - see device summary)

– Serial Peripheral interface (not on all products - see device summary)

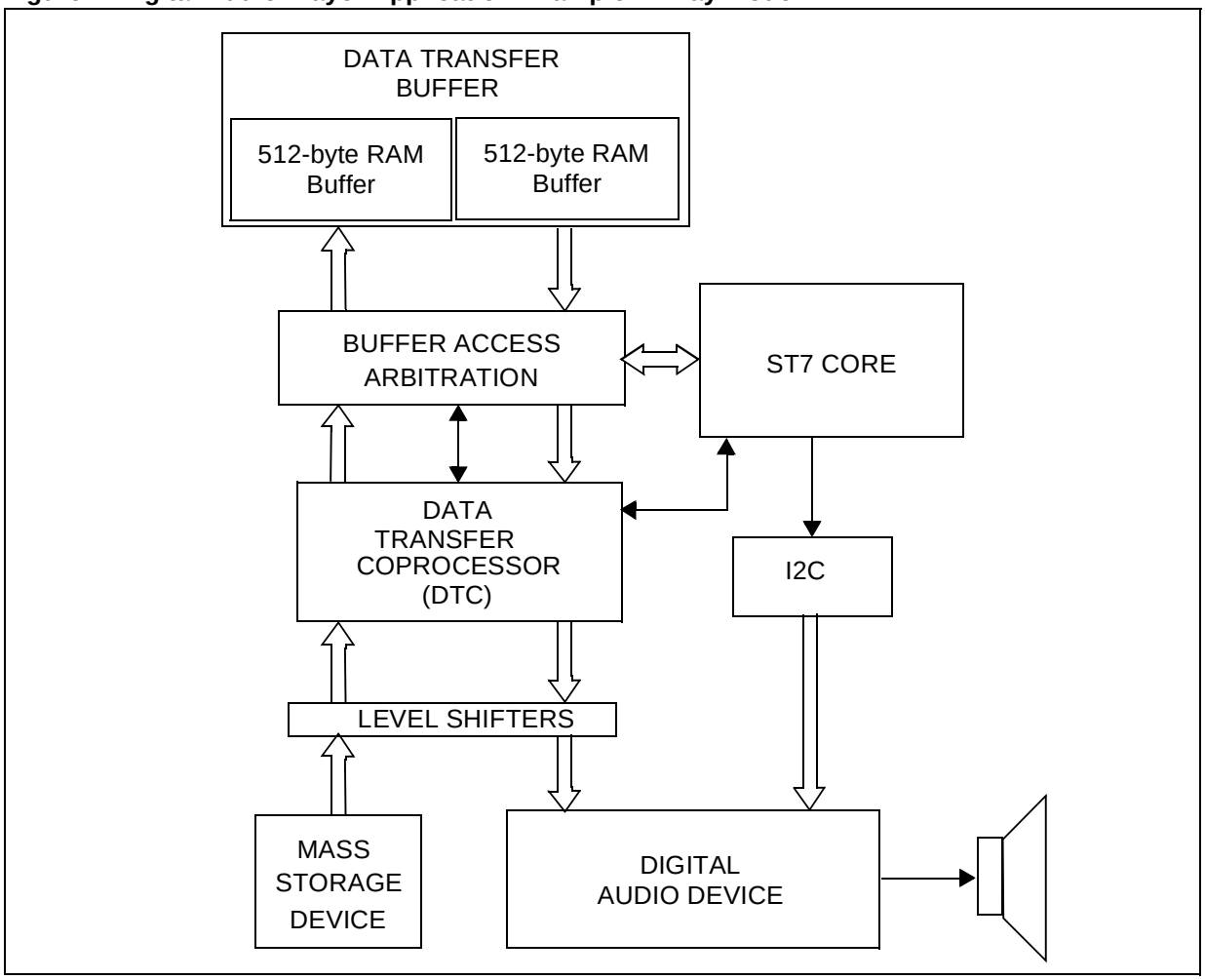
– Fast I^2C Single Master interface (not on all products - see device summary)

– 8-bit Analog-to-Digital converter (ADC) with 8 multiplexed analog inputs (not on all products - see device summary)

The ST72F65x are the Flash versions of the ST7265x in a TQFP64 package.

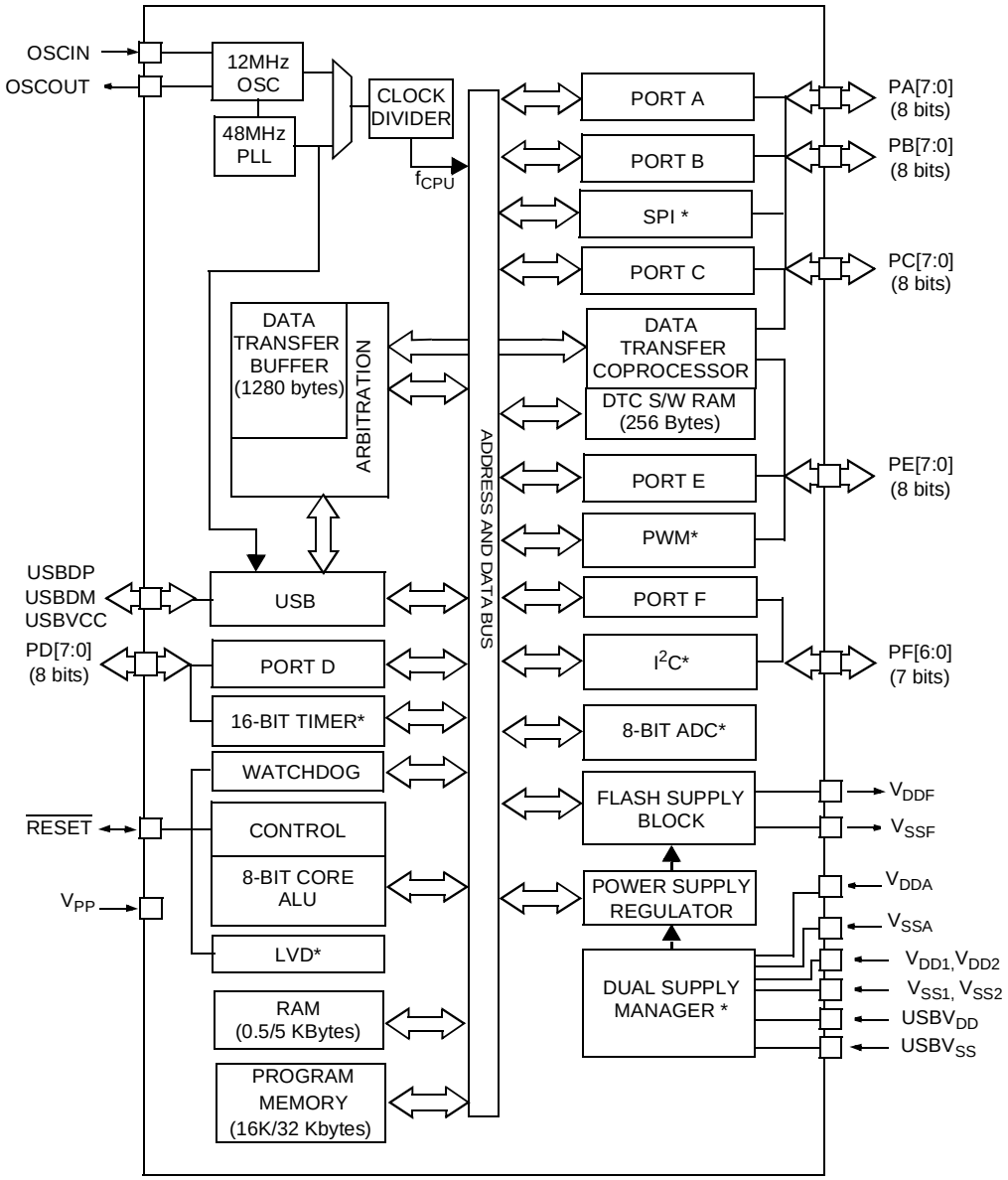
The ST7265x are the ROM versions in a TQFP64 package.

Figure 2. Digital Audio Player Application Example in Play Mode



INTRODUCTION (Cont'd)

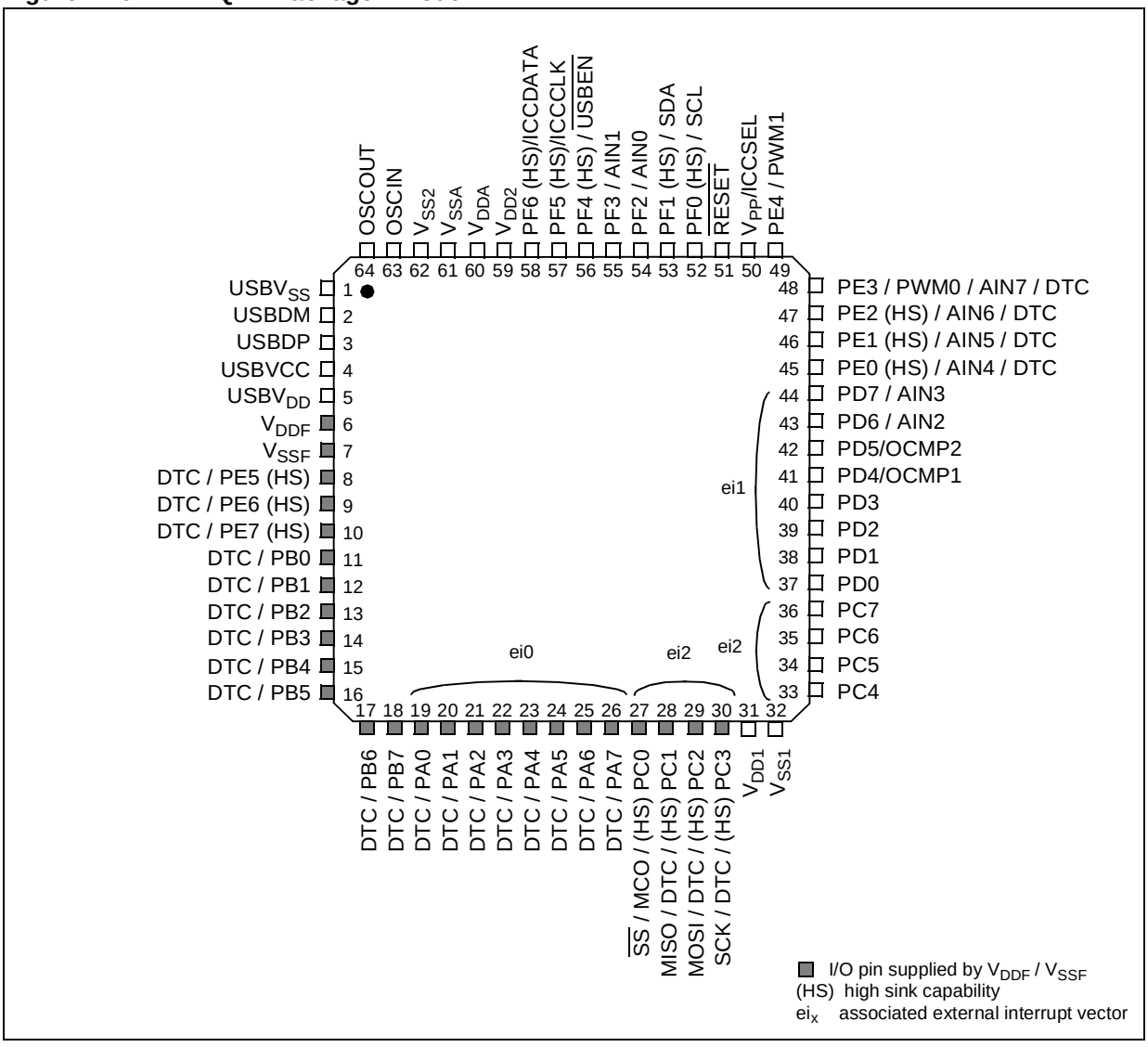
Figure 3. ST7265x Block Diagram



* not on all products (refer to Table 1: Device Summary)

2 PIN DESCRIPTION

Figure 4. 64-Pin TQFP Package Pinout



PIN DESCRIPTION (Cont'd)

Legend / Abbreviations:

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

V_{DDF} powered: I/O powered by the alternate supply rail, supplied by V_{DDF} and V_{SSF}.

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

Output level: HS = High Sink (on N-buffer only)

Port and control configuration:

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

– Output: OD = open drain, T = true open drain, PP = push-pull, OP = pull-up enabled by option byte.

Refer to "Port B (without Option Register)" on page 50 for more details on the software configuration of the I/O ports.

The RESET configuration of each pin is shown in bold.

Table 1. Device Pin Description

Pin	Pin Name	Type	V_{DDF} Powered	Level		Port / Control					Main Function (after reset)	Alternate Function
				Input	Output	Input			Output			
TQFP64						float	wpu	int	OD	PP		
1	$USBV_{SS}$	S									USB Digital ground	
2	USBDM	I/O									USB bidirectional data (data -)	
3	USBDP	I/O									USB bidirectional data (data +)	
4	USBVCC	O									USB power supply, output by the on-chip USB 3.3V linear regulator. **Note:** An external decoupling capacitor (typ. 100nF, min 47nF) must be connected between this pin and $USBV_{SS}$.	
5	$USBV_{DD}$	S									USB Power supply voltage (4V - 5.5V) **Note:** External decoupling capacitors (typ. 4.7µF+100nF, min 2.2µF+100nFmust be connected between this pin and $USBV_{SS}$.	
6	V_{DDF}	S	X								Power Line for alternate supply rail. Can be used as input (with external supply) or output (when using the on-chip voltage regulator). **Note:** An external decoupling capacitor (min. 20nF) must be connected to this pin to stabilize the regulator.	
7	V_{SSF}	S	X								Ground Line for alternate supply rail. Can be used as input (with external supply) or output (when using the on-chip voltage regulator)	
8	PE5/DTC	I/O	X	C_T	HS	X[2]			X[2]	X	Port E5	DTC I/O with serial capability (MMC_CMD)
9	PE6/DTC	I/O	X	C_T	HS	X			X	X	Port E6	DTC I/O with serial capability (MMC_DAT)
10	PE7/DTC	I/O	X	C_T	HS	X			X	X	Port E7	DTC I/O with serial capability (MMC_CLK)
11	PB0/DTC	I/O	X	CT		X				X	Port B0	DTC
12	PB1/DTC	I/O	X	CT		X				X	Port B1	DTC
13	PB2/DTC	I/O	X	CT		X				X	Port B2	DTC
14	PB3/DTC	I/O	X	CT		X				X	Port B3	DTC

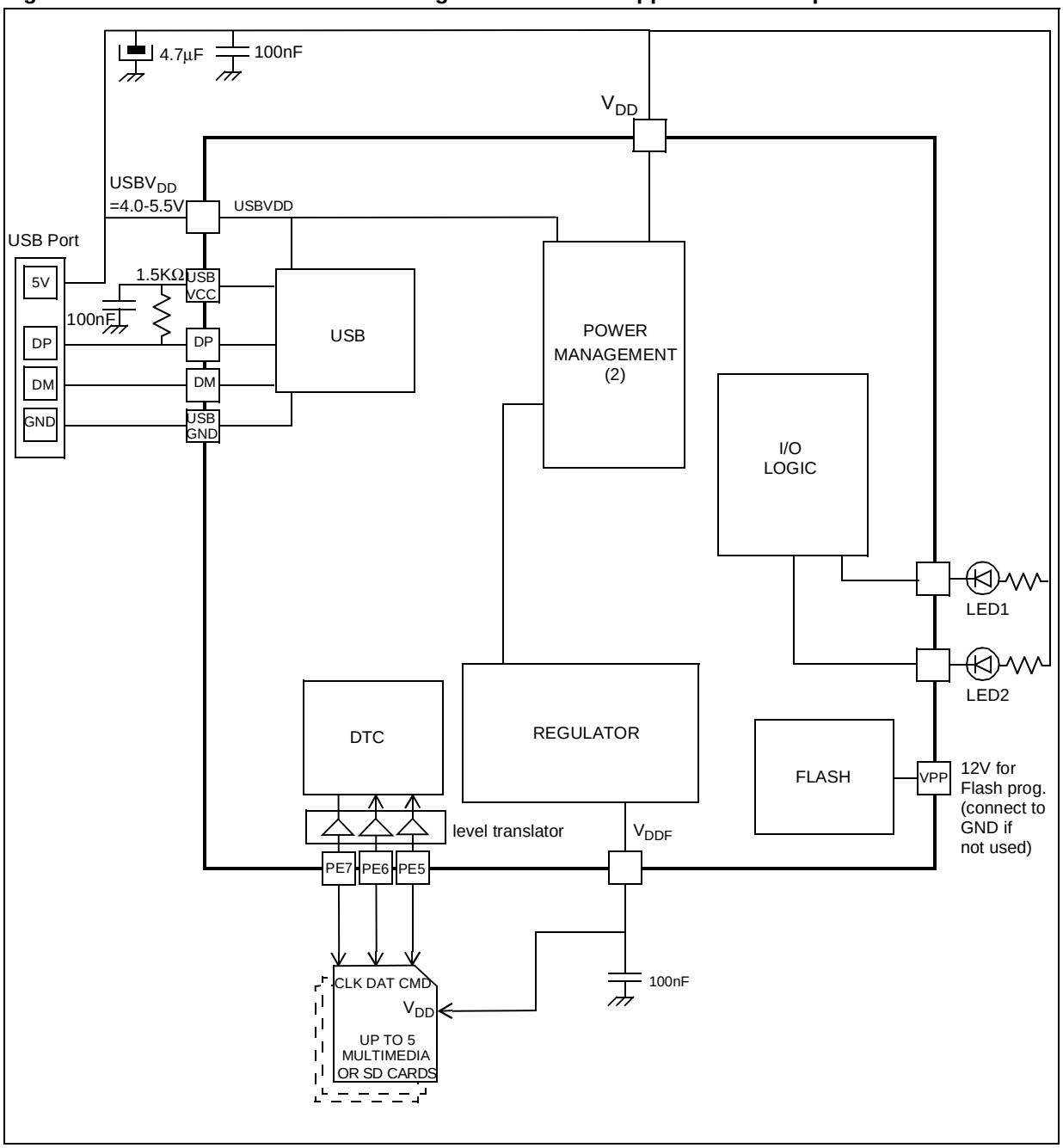
Pin	Pin Name	Type	V_{DDF} Powered	Level		Port / Control					Main Function (after reset)	Alternate Function
				Input	Output	Input			Output			
TQFP64						float	wpu	int	OD	PP		
15	PB4/DTC	I/O	X	CT		X				X	Port B4	DTC
16	PB5/DTC	I/O	X	CT		X				X	Port B5	DTC
17	PB6/DTC	I/O	X	CT		X				X	Port B6	DTC
18	PB7/DTC	I/O	X	CT		X				X	Port B7	DTC
19	PA0/DTC	I/O	X	CT		X		ei0	X	X	Port A0	DTC
20	PA1/DTC	I/O	X	CT		X			X	X	Port A1	DTC
21	PA2/DTC	I/O	X	CT		X			X	X	Port A2	DTC
22	PA3/DTC	I/O	X	CT		X			X	X	Port A3	DTC
23	PA4/DTC	I/O	X	CT		X			X	X	Port A4	DTC
24	PA5/DTC	I/O	X	CT		X			X	X	Port A5	DTC
25	PA6/DTC	I/O	X	CT		X			X	X	Port A6	DTC
26	PA7/DTC	I/O	X	CT		X			X	X	Port A7	DTC
27	PC0/MCO/$\overline{SS}$	I/O	X	CT	HS	X		ei2		X	Port C0	Main Clock Output / SPI Slave Select[1]
28	PC1/DTC/MIS0	I/O	X	C_T	HS	X				X	Port C1	DTC I/O with serial capability (DATARQ) / SPI Master In Slave Out[1]
29	PC2/DTC/MOSI	I/O	X	C_T	HS	X				X	Port C2	DTC I/O with serial capability (SDAT) / SPI Master Out Slave In[1]
30	PC3/DTC/SCK	I/O	X	C_T	HS	X				X	Port C3	DTC I/O with serial capability (SCLK) / SPI Serial Clock[1]
31	V_{DD1}	S									Power supply voltage (2.7V - 5.5V)	
32	V_{SS1}	S									Digital ground	
33	PC4/DTC	I/O		C_T		X		ei2		X	Port C4	DTC
34	PC5/DTC	I/O		C_T		X				X	Port C5	DTC
35	PC6/DTC	I/O		C_T		X				X	Port C6	DTC
36	PC7/DTC	I/O		C_T		X				X	Port C7	DTC
37	PD0	I/O		CT		X		ei1	X	X	Port D0	
38	PD1	I/O		CT		X			X	X	Port D1	
39	PD2	I/O		CT		X			X	X	Port D2	
40	PD3	I/O		CT		X			X	X	Port D3	
41	PD4/OCMP1	I/O		CT		X			X	X	Port D4	Timer Output Compare 1[1]
42	PD5/OCMP2	I/O		CT		X			X	X	Port D5	Timer Output Compare 2[1]
43	PD6/AIN2	I/O		CT		X			X	X	Port D6	Analog Input 2[1]
44	PD7/AIN3	I/O		CT		X			X	X	Port D7	Analog Input 3[1]

Pin	Pin Name	Type	V_{DDF} Powered	Level		Port / Control					Main Function (after reset)	Alternate Function
				Input	Output	Input			Output			
TQFP64						float	wpu	int	OD	PP		
45	PE0/DTC/AIN4	I/O		CT	HS	X			X	X	Port E0	Analog Input 4[1]/ DTC
46	PE1/DTC/AIN5	I/O		C_T	HS	X			X	X	Port E1	Analog Input 5[1]/ DTC
47	PE2/DTC/AIN6	I/O		C_T	HS	X			X	X	Port E2	Analog Input 6[1]/ DTC
48	PE3/AIN7/DTC/ PWM0	I/O		C_T		X			X	X	Port E3	Analog Input 7[1]/ DTC / PWM Output 0[1]
49	PE4/PWM1	I/O		C_T		X			X	X	Port E4	PWM Output 1[1]
50	V_{PP} /ICCSEL	S									Flash programming voltage. Must be held low in normal operating mode.	
51	$\overline{\text{RESET}}$	I/O					X		X		Bidirectional. This active low signal forces the initialization of the MCU. This event is the top priority non maskable interrupt. This pin is switched low when the Watchdog has triggered or V_{DD} is low. It can be used to reset external peripherals.	
52	PF0 / SCL	I/O		C_T	HS	X			T		Port F0	I^2C Serial Clock[1]
53	PF1 / SDA	I/O		C_T	HS	X			T		Port F1	I^2C Serial Data[1]
54	PF2 / AIN0	I/O		C_T		X				X	Port F2	Analog Input 0[1]
55	PF3 / AIN1	I/O		C_T		X				X	Port F3	Analog Input 1[1]
56	PF4 / $\overline{\text{USBEN}}$	I/O		C_T	HS	X			T		Port F4	USB Power Management USB Enable (alternate function selected by option bit)
57	PF5 / ICCCLK	I/O		C_T	HS	X			T		Port F5	ICC Clock Output
58	PF6 / ICCDATA	I/O		C_T	HS	X			T		Port F6	ICC Data Input
59	V_{DD2}	S									Main Power supply voltage (2.7V - 5.5V on devices without LVD, otherwise 4V - 5.5V).	
60	V_{DDA}	S									Analog supply voltage	
61	V_{SSA}	S									Analog ground	
62	V_{SS2}	S									Digital ground	
63	OSCIN	I									Input/Output Oscillator pins. These pins connect a 12 MHz parallel-resonant crystal, or an external source to the on-chip oscillator.	
64	OSCOUT	O										

[1] If the peripheral is present on the device (see Device Summary on page 1)

[2] A weak pull-up can be enabled on PE5 input and open drain output by configuring the PEOR register and depending on the PE5PU bit in the option byte.

Figure 5. Multimedia Card Or Secure Digital Card Writer Application Example



MultiMedia Card Pin	CMD	DAT	CLK
ST72F65 pin	PE5	PE6	PE7
ST7 / DTC (1)	DTC	DTC	DTC

(1) This line shows if the ST72F65 pin is controlled by the ST7 core or by the DTC.

(2) As this is a single power supply application, the $\overline{USBEN}$ function in not needed. Thus PF4/$\overline{USBEN}$ pin can be used as a normal I/O by configuring it as such by the option byte.

Figure 6. Smartmedia Card Writer Or Flash Drive Application Example

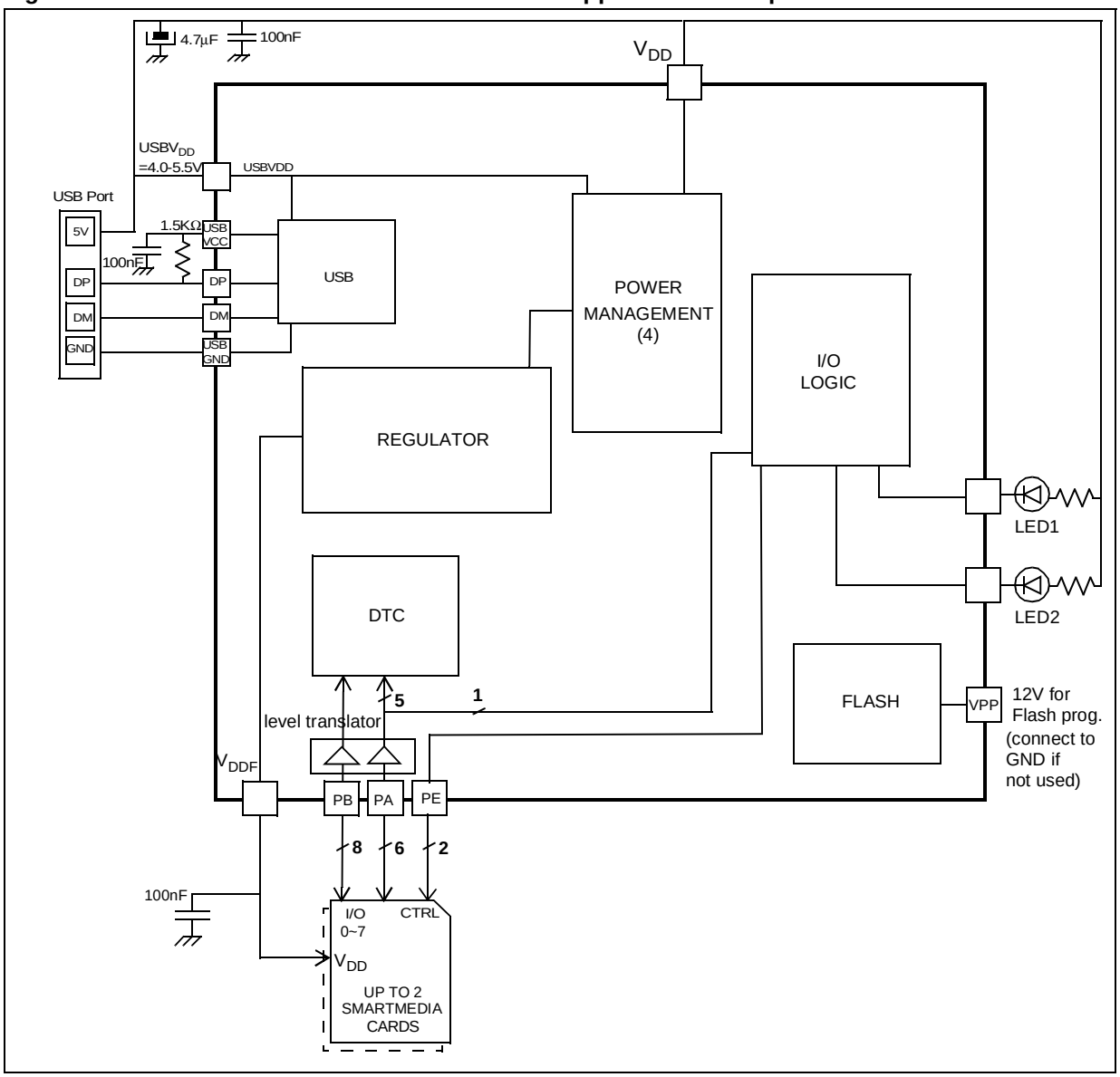


Table 2. SmartMedia Interface Pin Assignment

SmartMedia Pin	I/O0~7	CLE	WE	ALE	RE	R/B	WP(2)	CE1(2)	CE2(2)(3)
ST72F65 pin	PB0-7	PA0	PA1	PA2	PA3	PA4	PA7	PE1	PE0
ST7 / DTC (1)	DTC	DTC	DTC	DTC	DTC	DTC	ST7	ST7	ST7

(1): This line shows if the ST72F65 pin is controlled by the ST7 core or the DTC.

(2): These lines are not controlled by the DTC but by the user software running on the ST7 core. The ST72F65 pin choice is at customer discretion. The pins shown here are only shown as an example.

(3): When a single card is to be handled, PA7 is free for other functions. When 2 Smartmedia are to be handled, pins from both cards should be tied together (i.e. CLE1 with CLE2...) except for the CE pins. CE pin from card 1 should be connected to PA6 and CE pin from card 2 should be connect to PA7. Selection of the operating card is done by ST7 software.

(4) As this is a single power supply application, the $\overline{USBEN}$ function in not needed. Thus PF4/$\overline{USBEN}$ pin can be used as a normal I/O by configuring it as such by the option byte.

Figure 7. Compact Flash Card Writer Application Example

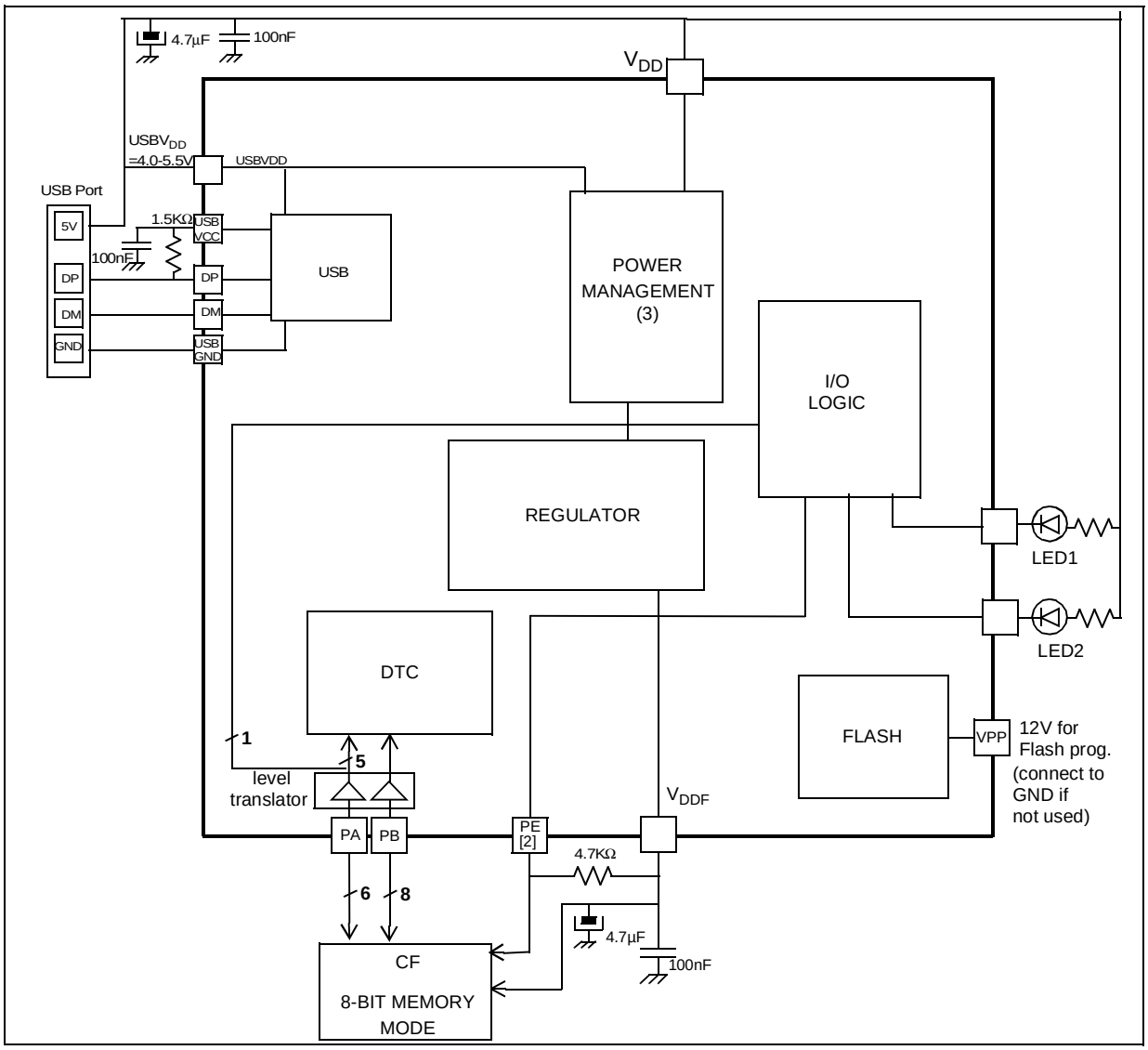


Table 3. Compact Flash Card Writer Pin Assignment

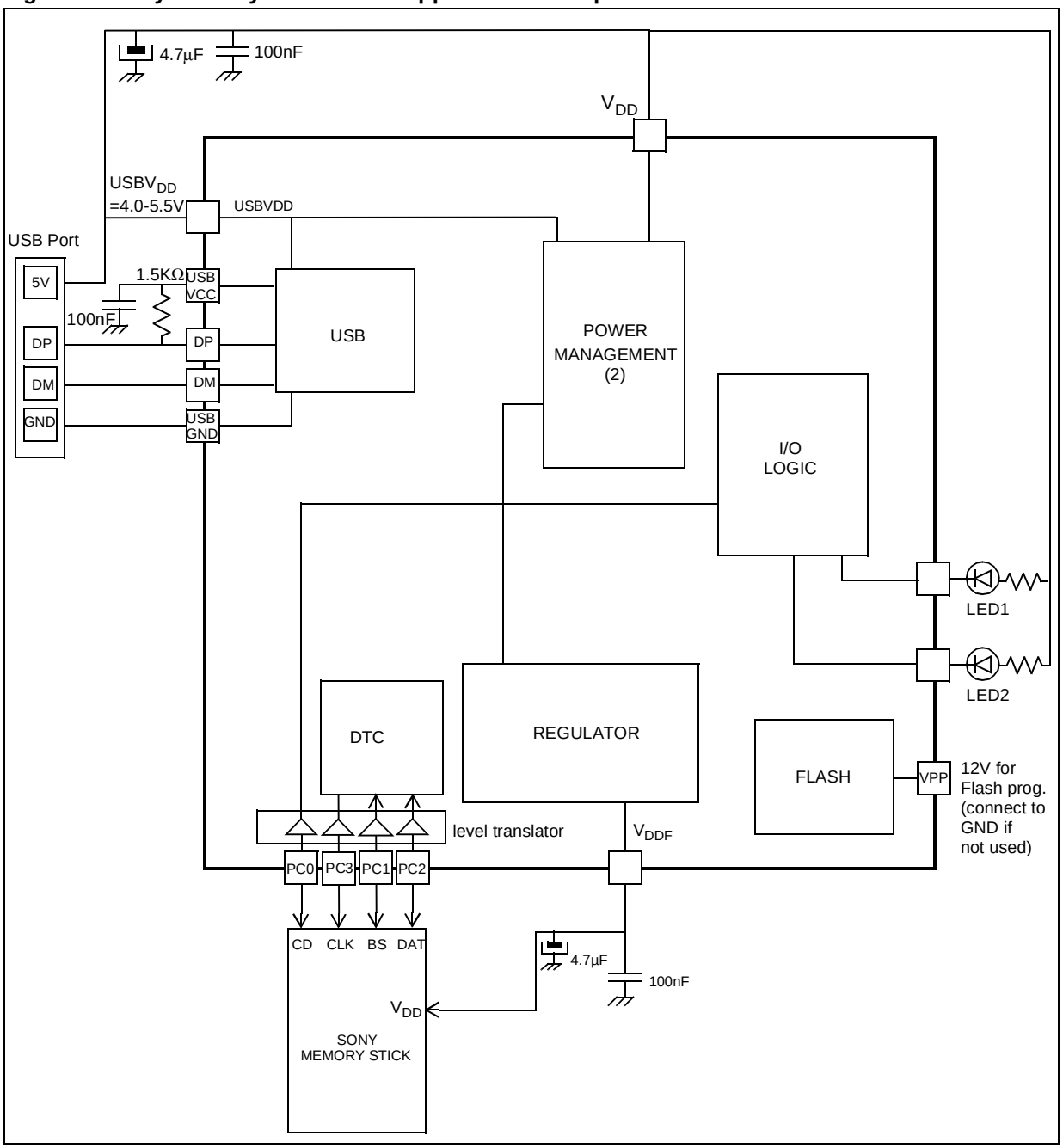
Compact Flash Card Pin	D0-7	D8-15	$\overline{VS1}$, $\overline{VS2}$, $\overline{WAIT}$, $\overline{CS1}$, $\overline{INPACK}$, $\overline{BVD1}$, $\overline{BVD2}$	$\overline{IORD}$, $\overline{IOWR}$, $\overline{REG}$, $\overline{CE2}$, V_{CC}	CSEL, RESET, GND, A3-10	A0-2	$\overline{CE1}$	$\overline{RE}$	$\overline{WE}$	$\overline{CD1}$	CD2, RDY/BSY, WP
ST72F65 pin	PB0-7	NC	NC	V_{DDF}	V_{SSF}	PA0-2	PE2 +pull-up 4.7kΩ	PA3	PA5	PA6 +pull-up 100kΩ	NC
ST7 / DTC (1)	DTC	-	-	Power	Power	DTC	ST7	DTC	DTC	ST7	-

(1) This line shows if the ST72F65 pin is controlled by the ST7 core or by the DTC.

(2) These lines are not controlled by the DTC but by the user software running on the ST7 core. The choice of ST72F65 pin is at the customer's discretion. The pins shown here are given only as an example.

(3) As this is a single power supply application, the $\overline{USBEN}$ function in not needed. Thus PF4/$\overline{USBEN}$ pin can be used as a normal I/O by configuring it as such by the option byte.

Figure 8. Sony Memory Stick Writer Application Example



MultiMedia Card Pin	CMD	DAT	CLK
ST72F65 pin	PE5	PE6	PE7
ST7 / DTC (1)	DTC	DTC	DTC

(1) This line shows if the ST72F65 pin is controlled by the ST7 core or by the DTC.

(2) As this is a single power supply application, the $\overline{USBEN}$ function in not needed. Thus PF4/USBEN pin can be used as a normal I/O by configuring it as such by the option byte.

3 REGISTER & MEMORY MAP

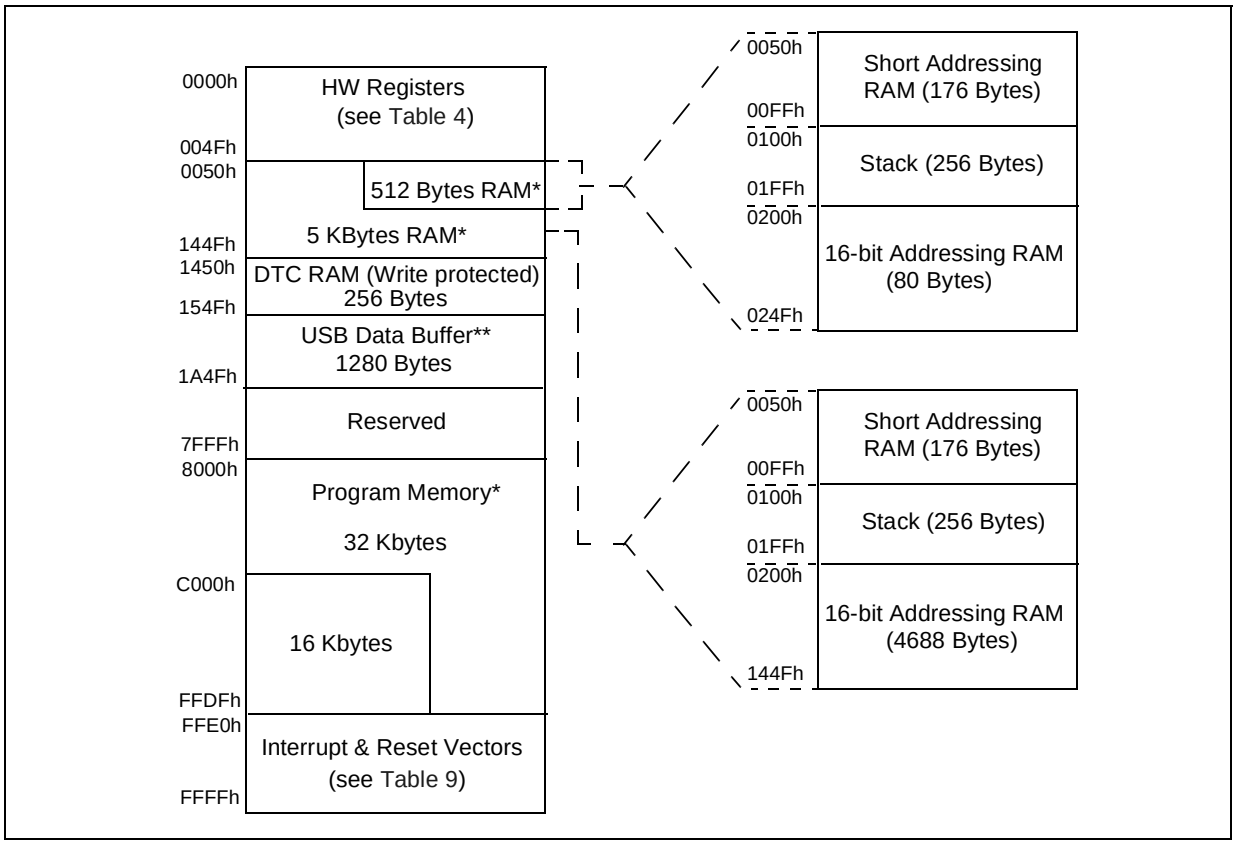
As shown in Figure 9, the MCU is capable of addressing 64 Kbytes of memories and I/O registers.

The available memory locations consist of 80 bytes of register locations, up to 5 Kbytes of RAM and up to 32 Kbytes of user program memory. The RAM space includes up to 256 bytes for the stack from 0100h to 01FFh.

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

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

Figure 9. Memory Map



* Program memory and RAM sizes are product dependent (see Table –)

** The ST7 core is not able to read or write in the USB data buffer if the ST7265x is running at 6Mz in standalone mode.

Table 4. Hardware Register Memory Map

Address	Block	Register Label	Register name	Reset Status	Remarks
0000h		PADR	Port A Data Register	00h	R/W
0001h		PADDR	Port A Data Direction Register	00h	R/W
0002h		PAOR	Port A Option Register	00h	R/W
0003h		PBDR	Port B Data Register	00h	R/W
0004h		PBDDR	Port B Data Direction Register	00h	R/W
0005h	Reserved Area (1 byte)				
0006h		PCDR	Port C Data Register	00h	R/W
0007h		PCDDR	Port C Data Direction Register	00h	R/W
0008h		PCOR	Port C Option Register	00h	R/W
0009h		PDDR	Port D Data Register	00h	R/W
000Ah		PDDDR	Port D Data Direction Register	00h	R/W
000Bh		PDOR	Port D Option Register	00h	R/W
000Ch		PEDR	Port E Data Register	00h	R/W
000Dh		PEDDR	Port E Data Direction Register	00h	R/W
000Eh		PEOR	Port E Option Register	00h	R/W
000Fh		PFDR	Port F Data Register	00h	R/W
0010h		PFDDR	Port F Data Direction Register	00h	R/W
0011h	Reserved Area (1 byte)				
0012h	ADC[1]	ADCDR	ADC Data Register	00h	Read only
0013h		ADCCSR	ADC Control Status Register	00h	R/W
0014h	WDG	WDGCR	Watchdog Control Register	7Fh	R/W
0015h to 0017h	Reserved Area (3 bytes)				
0018h	DSM	PCR	Power Control Register	00h	R/W
0019h	SPI	SPIDR	SPI Data I/O Register	xxh	R/W
001Ah		SPICR	SPI Control Register	0xh	R/W
001Bh		SPICSR	SPI Control/Status Register	00h	R/W
001Ch	DTC	DTCCR	DTC Control Register	00h	R/W
001Dh		DTCSR	DTC Status Register	00h	R/W
001Eh		Reserved			
001Fh		DTCPR	DTC Pointer Register	00h	R/W

Address	Block	Register Label	Register name	Reset Status	Remarks
0020h	TIM	TCR1	Timer Control Register 1	00h	R/W
0021h		TCR2	Timer Control Register 2	00h	R/W
0022h		TSR	Timer Status Register	00h	Read Only
0023h		CHR	Timer Counter High Register	FFh	Read Only
0024h		CLR	Timer Counter Low Register	FCh	Read Only
0025h		ACHR	Timer Alternate Counter High Register	FFh	Read Only
0026h		ACLR	Timer Alternate Counter Low Register	FCh	Read Only
0027h		OC1HR	Timer Output Compare 1 High Register	80h	R/W
0028h		OC1LR	Timer Output Compare 1 Low Register	00h	R/W
0029h		OC2HR	Timer Output Compare 2 High Register	80h	R/W
002Ah		OC2LR	Timer Output Compare 2 Low Register	00h	R/W
002Bh	Flash		Flash Control Status Register	00h	R/W
002Ch	ITC	ITSPR0	Interrupt Software Priority Register 0	FFh	R/W
002Dh		ITSPR1	Interrupt Software Priority Register 1	FFh	R/W
002Eh		ITSPR2	Interrupt Software Priority Register 2	FFh	R/W
002Fh		ITSPR3	Interrupt Software Priority Register 3	FFh	R/W
0030h	USB	USBISTR	USB Interrupt Status Register	00h	R/W
0031h		USBIMR	USB Interrupt Mask Register	00h	R/W
0032h		USBCTLR	USB Control Register	06h	R/W
0033h		DADDR	Device Address Register	00h	R/W
0034h		USBSR	USB Status Register	00h	R/W
0035h		EP0R	Endpoint 0 Register	00h	R/W
0036h		CNT0RXR	EP 0 Reception Counter Register	00h	R/W
0037h		CNT0TXR	EP 0 Transmission Counter Register	00h	R/W
0038h		EP1RXR	Endpoint 1 Register	00h	R/W
0039h		CNT1RXR	EP 1 Reception Counter Register	00h	R/W
003Ah		EP1TXR	Endpoint 1 Register	00h	R/W
003Bh		CNT1TXR	EP 1 Transmission Counter Register	00h	R/W
003Ch		EP2RXR	Endpoint 2 Register	00h	R/W
003Dh		CNT2RXR	EP 2 Reception Counter Register	00h	R/W
003Eh		EP2TXR	Endpoint 2 Register	00h	R/W
003Fh		CNT2TXR	EP 2 Transmission Counter Register	00h	R/W
0040h	I^2C [1]	I2CCR	I^2C Control Register	00h	R/W
0041h		I2CSR1	I^2C Status Register 1	00h	Read only
0042h		I2CSR2	I^2C Status Register 2	00h	Read only
0043h		I2CCCR	I^2C Clock Control Register	00h	R/W
0044h		Not used			
0045h		Not used			
0046h		I2CDR	I^2C Data Register	00h	R/W
0047h	USB	BUFCSR	Buffer Control/Status Register	00h	R/W
0048h	Reserved Area (1 Byte)				
0049h		MISCR1	Miscellaneous Register 1	00h	R/W
004Ah		MISCR2	Miscellaneous Register 2	00h	R/W
004Bh	Reserved Area (1 Byte)				

Address	Block	Register Label	Register name	Reset Status	Remarks
004Ch		MISCR3	Miscellaneous Register 3	00h	R/W
004Dh	PWM[1]	PWM0	10-bit PWM/BRM registers	80h	R/W
004Eh		BRM10		00h	R/W
004Fh		PWM1		80h	R/W

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

4 FLASH PROGRAM MEMORY

4.1 Introduction

The ST7 dual voltage High Density Flash (HDFlash) is a non-volatile memory that can be electrically erased as a single block or by individual sectors and programmed on a Byte-by-Byte basis using an external V_{PP} supply.

The HDFlash devices can be programmed and erased off-board (plugged in a programming tool) or on-board using ICP (In-Circuit Programming) or IAP (In-Application Programming).

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

4.2 Main Features

- Three Flash programming modes:
  - Insertion in a programming tool. In this mode, all sectors including option bytes can be programmed or erased.
  - ICP (In-Circuit Programming). In this mode, all sectors including option bytes can be programmed or erased without removing the device from the application board.
  - IAP (In-Application Programming) In this mode, all sectors except Sector 0, can be programmed or erased without removing the device from the application board and while the application is running.
- ICT (In-Circuit Testing) for downloading and executing user application test patterns in RAM
- Read-out protection against piracy
- Register Access Security System (RASS) to prevent accidental programming or erasing

4.3 Structure

The Flash memory is organised in sectors and can be used for both code and data storage.

Depending on the overall Flash memory size in the microcontroller device, there are up to three user sectors (see Table 5). Each of these sectors can be erased independently to avoid unnecessary erasing of the whole Flash memory when only a partial erasing is required.

The first two sectors have a fixed size of 4 Kbytes (see Figure 10). They are mapped in the upper part of the ST7 addressing space so the reset and interrupt vectors are located in Sector 0 (F000h-FFFFh).

Table 5. Sectors available in Flash devices

Flash Memory Size (bytes)	Available Sectors
4K	Sector 0
8K	Sectors 0,1
> 8K	Sectors 0,1, 2

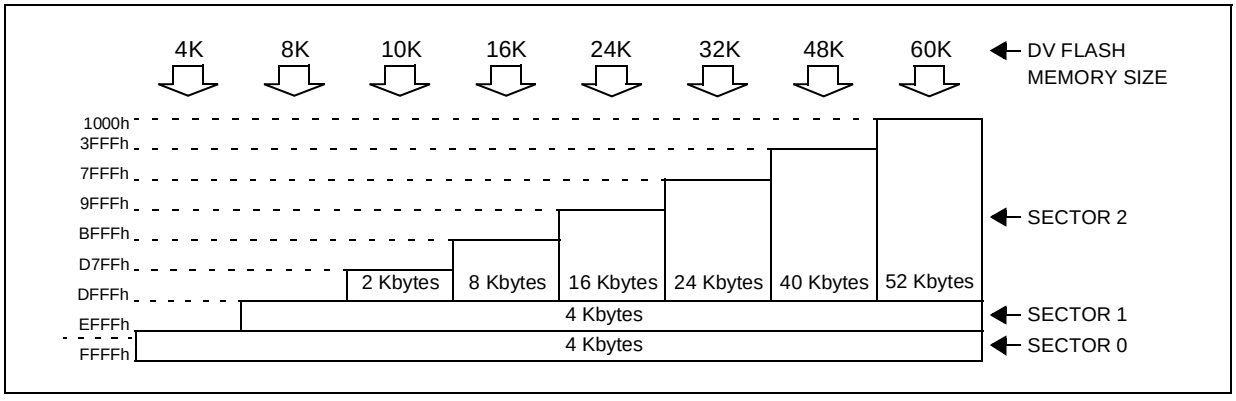
4.4 Program Memory Read-out Protection

The read-out protection is enabled through an option bit.

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

Refer to the Option Byte description for more details.

Figure 10. Memory Map and Sector Address



FLASH PROGRAM MEMORY (Cont'd)

4.5 ICP (In-Circuit Programming)

To perform ICP the microcontroller must be switched to ICC (In-Circuit Communication) mode by an external controller or programming tool.

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

When using an STMicroelectronics or third-party programming tool that supports ICP and the specific microcontroller device, the user needs only to implement the ICP hardware interface on the application board (see Figure 11). For more details on the pin locations, refer to the device pinout description.

ICP needs six pins to be connected to the programming tool. These pins are:

- $\overline{\text{RESET}}$: device reset
- V_{SS}: device power supply ground
- ICCCLK: ICC output serial clock pin
- ICCDATA: ICC input serial data pin
- ICCSEL/V_{PP}: programming voltage
- V_{DD}: application board power supply

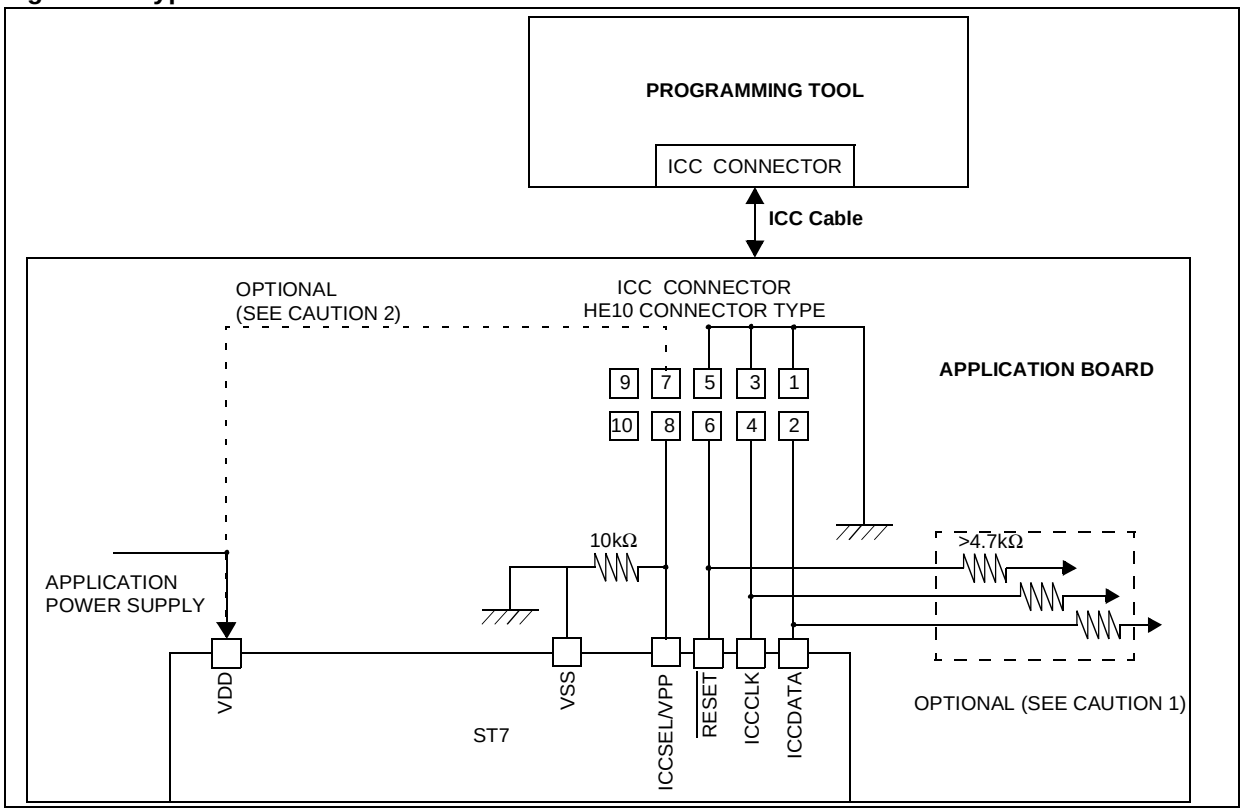
CAUTIONS:

1. If $\overline{\text{RESET}}$, ICCCLK or ICCDATA pins are used for other purposes in the application, a serial resistor has to be implemented to avoid a conflict in case one of the other devices forces the signal level. If these pins are used as outputs in the application, the serial resistors are not necessary. As soon as the external controller is plugged to the board, even if an ICC session is not in progress, the ICCCLK and ICCDATA pins are not available for the application.

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

Note: To develop a custom programming tool, refer to the ST7 Flash Programming and ICC Reference Manual which gives full details on the ICC protocol hardware and software.

Figure 11. Typical ICP Interface



FLASH PROGRAM MEMORY (Cont'd)

4.6 IAP (In-Application Programming)

This mode uses a BootLoader program previously stored in Sector 0 by the user (in ICP mode or by plugging the device in a programming tool).

This mode is fully controlled by user software. This allows it to be adapted to the user application, (user-defined strategy for entering programming mode, choice of communications protocol used to fetch the data to be stored, etc.). For example, it is possible to download code from the SPI, SCI, USB or CAN interface and program it in the Flash. IAP mode can be used to program any of the Flash sectors except Sector 0, which is write/erase protected to allow recovery in case errors occur during the programming operation.

4.6.1 Register Description

FLASH CONTROL/STATUS REGISTER (FCSR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	0	0

This register is reserved for use by Programming Tool software. It controls the Flash programming and erasing operations. For details on customizing Flash programming methods and In-Circuit Testing, refer to the ST7 Flash Programming and ICC Reference Manual.

5 CENTRAL PROCESSING UNIT

5.1 INTRODUCTION

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

5.2 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

5.3 CPU REGISTERS

The 6 CPU registers shown in Figure 12 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)

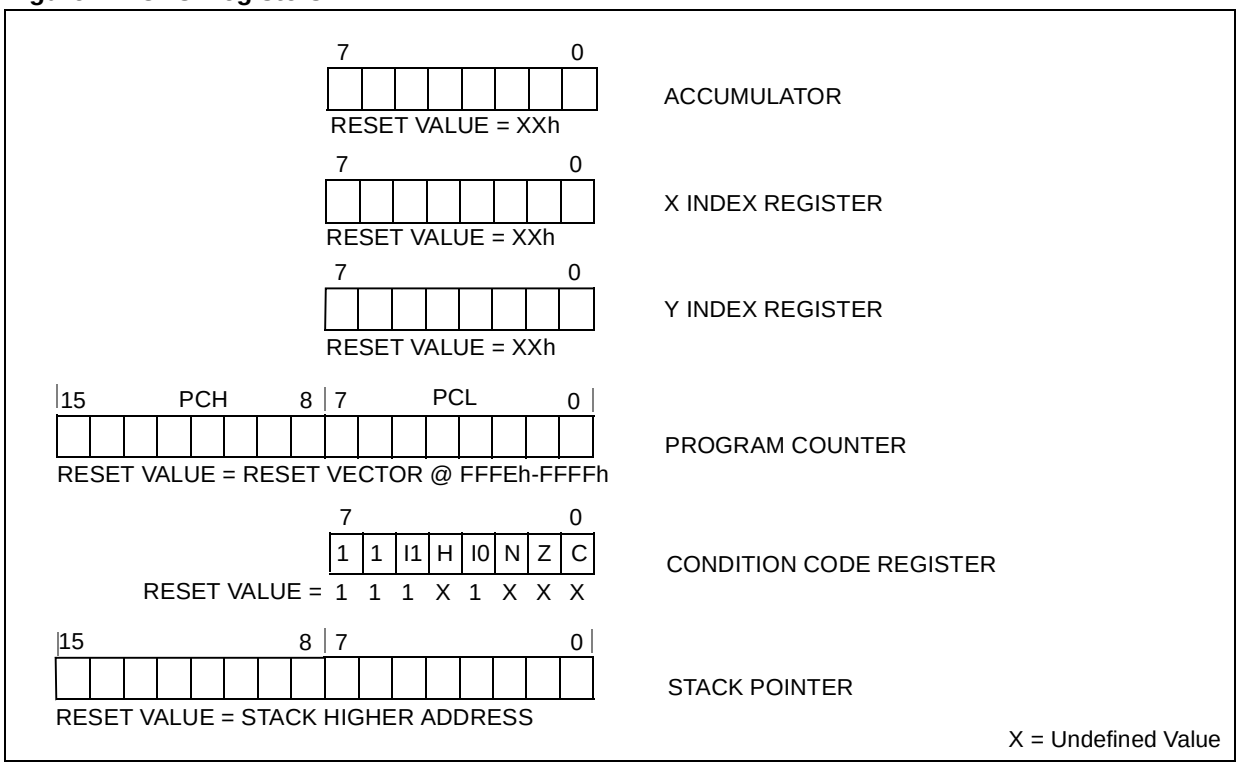
These 8-bit registers are used to create effective addresses or as temporary storage areas for data manipulation. (The Cross-Assembler generates a precede instruction (PRE) to indicate that the following instruction refers to the Y register.)

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 12. CPU Registers



CENTRAL PROCESSING UNIT (Cont'd)

Condition Code Register (CC)

Read/Write

Reset Value: 111x1xxx

7							0
1	1	I1	H	I0	N	Z	C

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

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

Arithmetic Management Bits

Bit 4 = **H** *Half carry*.

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

0: No half carry has occurred.
1: A half carry has occurred.

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

Bit 2 = **N** *Negative*.

This bit is set and cleared by hardware. It is representative of the result sign of the last arithmetic, logical or data manipulation. It's a copy of the result 7^{th} bit.
0: The result of the last operation is positive or null.
1: The result of the last operation is negative (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: 01 FFh

15							8
0	0	0	0	0	0	0	1

7							0
SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0

The Stack Pointer is a 16-bit register which is always pointing to the next free location in the stack. It is then decremented after data has been pushed onto the stack and incremented before data is popped from the stack (see Figure 13).

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

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

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

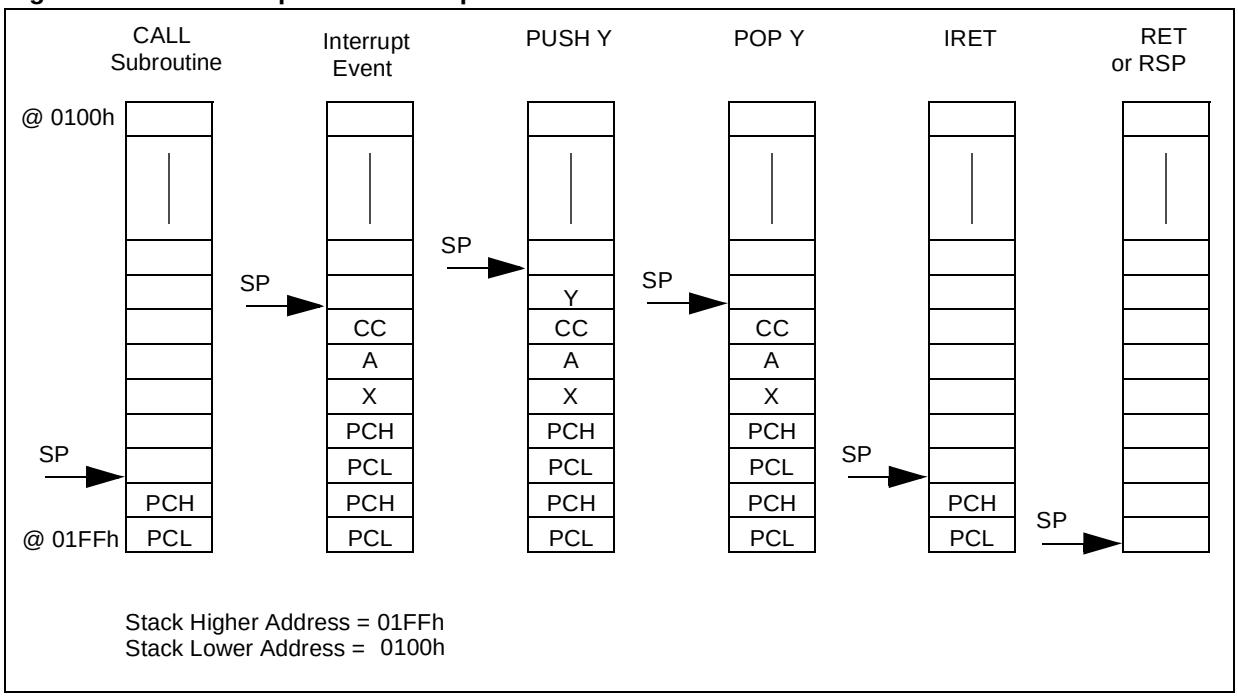
The stack is used to save the return address during a subroutine call and the CPU context during an interrupt. The user may also directly manipulate the stack by means of the PUSH and POP instructions. In the case of an interrupt, the PCL is stored at the first location pointed to by the SP. Then the other registers are stored in the next locations as shown in Figure 13.

– 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 13. Stack Manipulation Example



6 SUPPLY, RESET AND CLOCK MANAGEMENT

6.1 CLOCK SYSTEM

6.1.1 General Description

The MCU accepts either a 12 MHz crystal or an external clock signal to drive the internal oscillator. The internal clock (f_{CPU}) is derived from the internal oscillator frequency (f_{OSC}), which is 12 Mhz in Stand-alone mode and 48Mhz in USB mode.

The internal clock (f_{CPU}) is software selectable using the CP[1:0] and CPEN bits in the MISCR1 register.

In USBV_{DD} power supply mode, the PLL is active, generating a 48MHz clock to the USB. In this mode, f_{CPU} can be configured to be up to 8 MHz. In V_{DD} mode the PLL and the USB clock are disabled, and the maximum frequency of f_{CPU} is 6 MHz.

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

The internal oscillator is designed to operate with an AT-cut parallel resonant quartz in the frequency range specified for f_{osc}. The circuit shown in Figure 15 is recommended when using a crystal, and Table 6 lists the recommended capacitance. The crystal and associated components should be mounted as close as possible to the input pins in order to minimize output distortion and start-up stabilisation time.

Table 6. Recommended Values for 12-MHz Crystal Resonator

R_{SMAX}	20 Ω	25 Ω	70 Ω
C_{OSCIN}	56pF	47pF	22pF
C_{OSCOUT}	56pF	47pF	22pF

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

6.1.2 External Clock

An external clock may be applied to the OSCIN input with the OSCOUT pin not connected, as shown on Figure 14. The t_{OXOV} specifications does not apply when using an external clock input. The equivalent specification of the external clock source should be used instead of t_{OXOV} (see Section 6.5 CONTROL TIMING).

Figure 14. External Clock Source Connections

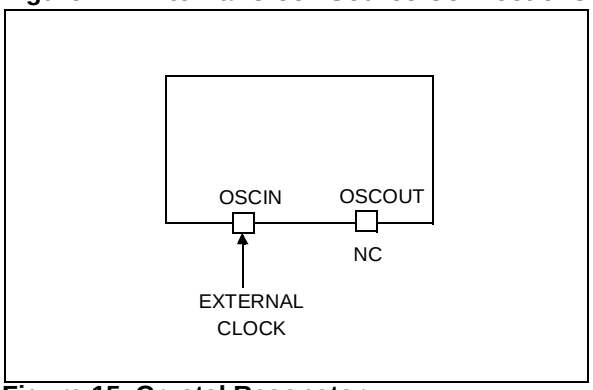
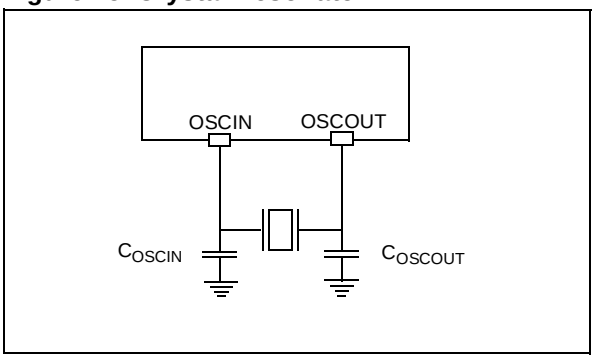


Figure 15. Crystal Resonator



6.2 RESET SEQUENCE MANAGER (RSM)

6.2.1 Introduction

The reset sequence manager includes three RESET sources as shown in Figure 6.2.2:

- External $\overline{\text{RESET}}$ source pulse
- Internal LVD RESET (Low Voltage Detection)
- Internal WATCHDOG RESET

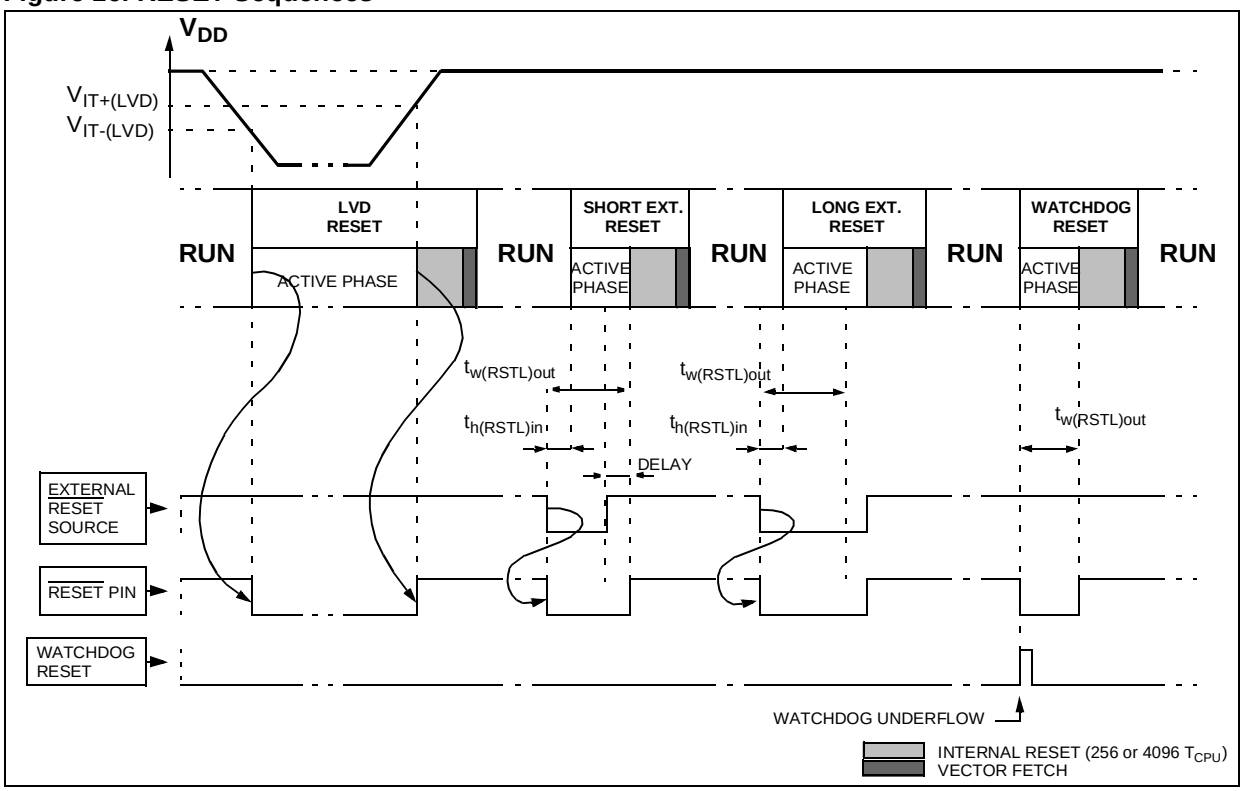
These sources act on the $\overline{\text{RESET}}$ pin and it is always kept low during the delay phase.

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

The basic RESET sequence consists of 3 phases as shown in Figure 16:

- Active Phase depending on the RESET source
- Min 512 CPU clock cycle delay (see Figure 18 and Figure 19
- RESET vector fetch

Figure 16. RESET Sequences



RESET SEQUENCE MANAGER (Cont'd)

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

The $\overline{\text{RESET}}$ pin is both an input and an open-drain output with integrated R_{ON} weak pull-up resistor. This pull-up has no fixed value but varies in accordance with the input voltage. It can be pulled low by external circuitry to reset the device. See electrical characteristics section for more details.

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

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.

If the external $\overline{\text{RESET}}$ pulse is shorter than $t_{w(RSTL)out}$ (see short ext. Reset in Figure 16), the signal on the $\overline{\text{RESET}}$ pin will be stretched. Otherwise the delay will not be applied (see long ext. Reset in Figure 16).

Starting from the external RESET pulse recognition, the device $\overline{\text{RESET}}$ pin acts as an output that is pulled low during at least $t_{w(RSTL)out}$.

6.2.3 Internal Low Voltage Detection RESET

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

- Power-On RESET
- Voltage Drop RESET

The device $\overline{\text{RESET}}$ pin acts as an output that is pulled low when $V_{DD}<V_{IT+}$ (rising edge) or $V_{DD}<V_{IT-}$ (falling edge) as shown in Figure 16.

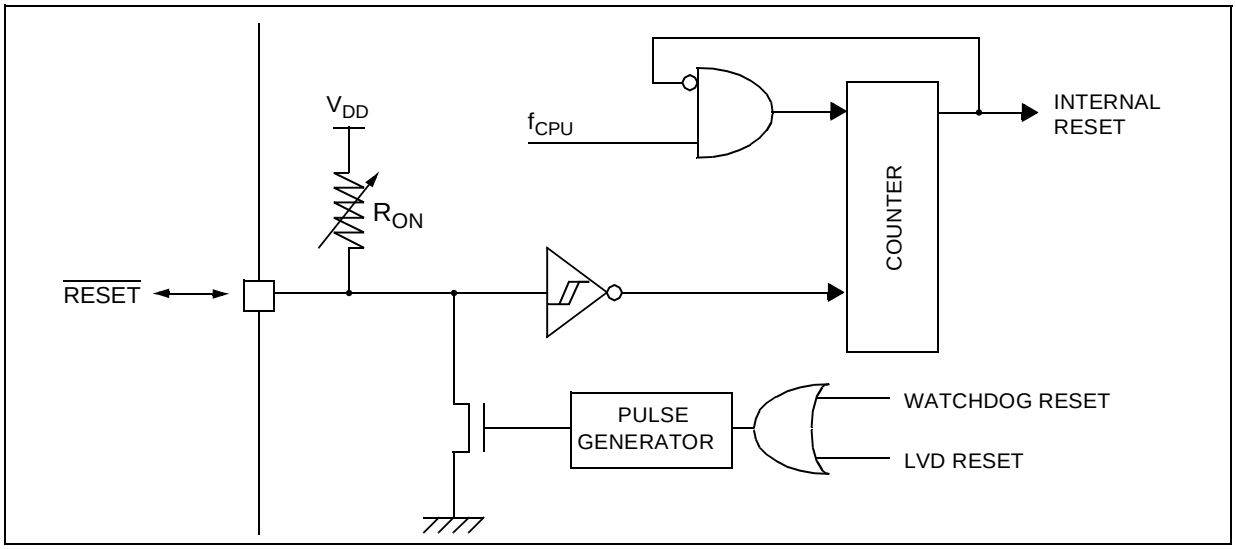
The LVD filters spikes on V_{DD} shorter than $t_{g(VDD)}$ to avoid parasitic resets.

6.2.4 Internal Watchdog RESET

The RESET sequence generated by a internal Watchdog counter overflow is shown in Figure 16.

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

Figure 17. Reset Block Diagram



RESET SEQUENCE MANAGER (Cont'd)

In stand-alone mode, the 512 CPU clock cycle delay allows the oscillator to stabilize and ensures that recovery has taken place from the Reset state.

In USB mode the delay is 256 clock cycles counted from when the PLL LOCK signal goes high.

The RESET vector fetch phase duration is 2 clock cycles.

Figure 18. Reset Delay in Stand-alone Mode

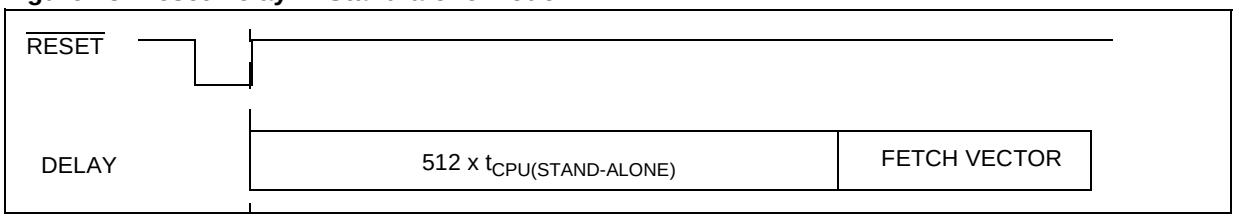
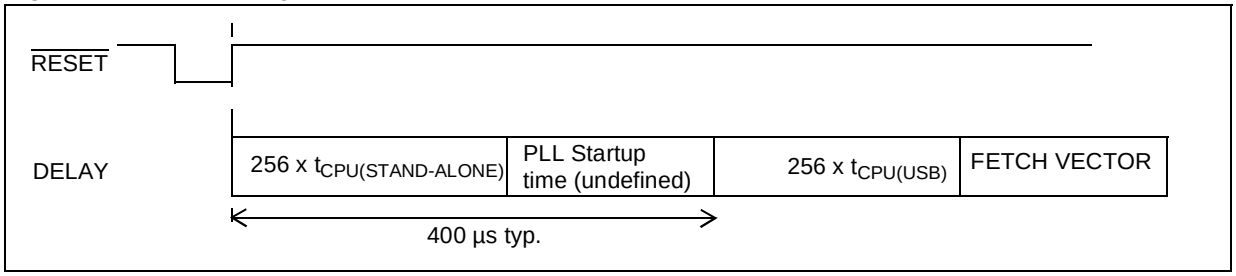


Figure 19. Reset Delay in USB Mode



Note: For a description of Stand-alone mode and USB mode refer to Section 6.4.

6.3 LOW VOLTAGE DETECTOR (LVD)

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

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

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

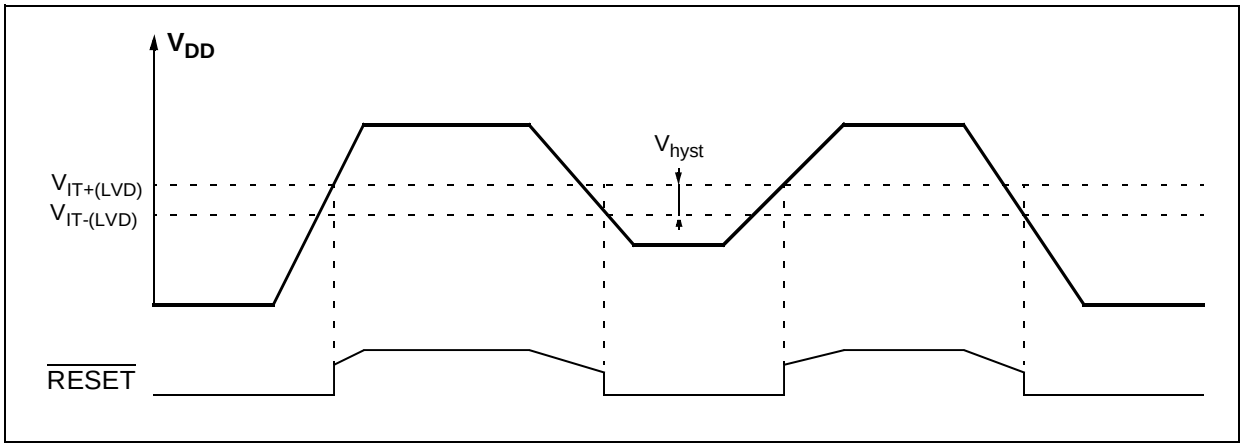
– V_{IT+} when V_{DD} is rising

– V_{IT-} when V_{DD} is falling

The LVD function is illustrated in Figure 20.

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

Figure 20. Low Voltage Detector vs Reset



6.4 POWER SUPPLY MANAGEMENT

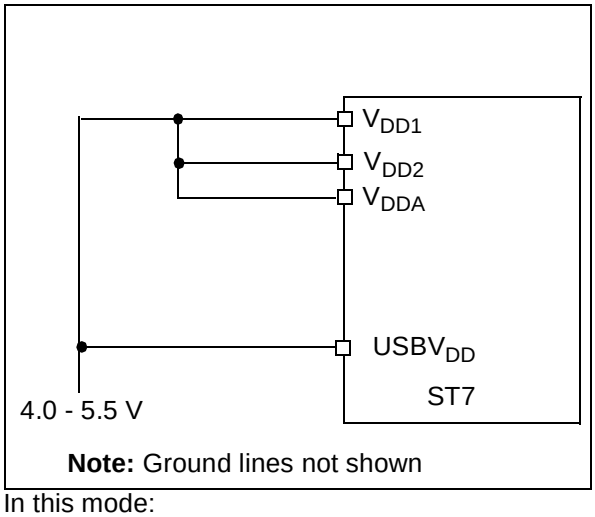
6.4.1 Single Power Supply Management

In applications operating only when connected to the USB (Flash writers, Backup systems), the microcontroller must operate from a single power supply (i.e. USB bus power supply or the local power source in the case of self-powered devices). Devices with LVD (no E suffix) or without LVD (E suffix) can support this configuration.

In order to enable the Single Power Supply Management, the PLGIE bit in the PCR register should kept cleared by software (reset default value).

In this case, pin V_{DD} and $USBV_{DD}$ of the microcontroller must be connected together and supplied by a 4.0 to 5.5V voltage supply, either from the USB cable or from the local power source. See Figure 21.

Figure 21. Single Power Supply Mode



Note: Ground lines not shown

In this mode:

– The PLL is running at 48 MHz

– The on-chip USB interface is enabled

– The core can run at up to 8MHz internal frequency

– The microcontroller can be either USB bus powered or supplied by the local power source (self powered)

– The $\overline{USBEN}$ function is not used. The PF4 pin can be configured to work as a normal I/O by programming the Option Byte.

6.4.2 Dual Power Supply Management

In case of a device that can be used both when powered by the USB or from a battery (Digital Audio Player, Digital Camera, PDA), the microcontroller can operate in two power supply modes: *Stand-alone* Mode and *USB* Mode. This configuration is only available on devices without LVD (E suffix). Devices with LVD are kept under reset when the power supply drops below the LVD threshold voltage and thus Stand-Alone mode can not be entered.

In order to enable Dual Power Supply Management:

– the $\overline{USBEN}$ pin function must be selected by programming the option byte.

– the user software must set the PLGIE bit in the PCR register in the initialization routine.

Stand-Alone Mode

This mode is to be used when no USB communication is needed. The microcontroller in this mode can run at very low voltage, making the design of low power / battery supplied systems easy. In this mode:

– The USB cable is unplugged (no voltage input on $USBV_{DD}$ pin)

– The PLL is off

– The on-chip USB interface is disabled

– The core can run at up to 6 MHz internal frequency

– $\overline{USBEN}$ is kept floating by H/W.

– The microcontroller is supplied through the V_{DD} pin

USB Mode

When connected to the USB, the microcontroller can run at full speed, still saving battery power by using USB power or self power source. To go into USB mode, a voltage from 4.0V to 5.5V must be provided to the $USBV_{DD}$ pin. In this mode:

– The USB cable is plugged in

– $USBV_{DD}$ pin is supplied by a 4.0 to 5.5V supply voltage, either from the USB cable or from the self powering source

– The PLL is running at 48 MHz

– The on-chip USB interface is enabled

– The core can run at up to 8 MHz internal frequency

– $\overline{USBEN}$ is set to output low level by hardware. This signal can be used to control an external transistor (USB SWITCH) to change the power supply configuration (see Figure 22).

– The microcontroller can be USB bus powered

POWER SUPPLY MANAGEMENT (Cont'd)

6.4.2.1 Switching from Stand-Alone Mode to USB Mode

In Stand-Alone Mode, when the user plugs in the USB cable, 4V min. is input to $USBV_{DD}$. The on-chip power Supply Manager generates an internal interrupt when $USBV_{DD}$ reaches $USBV_{IT+}$ (if the PLGIE bit in the PCR register is set). The user program then can finish the current processing, and MUST generate a software RESET afterwards.

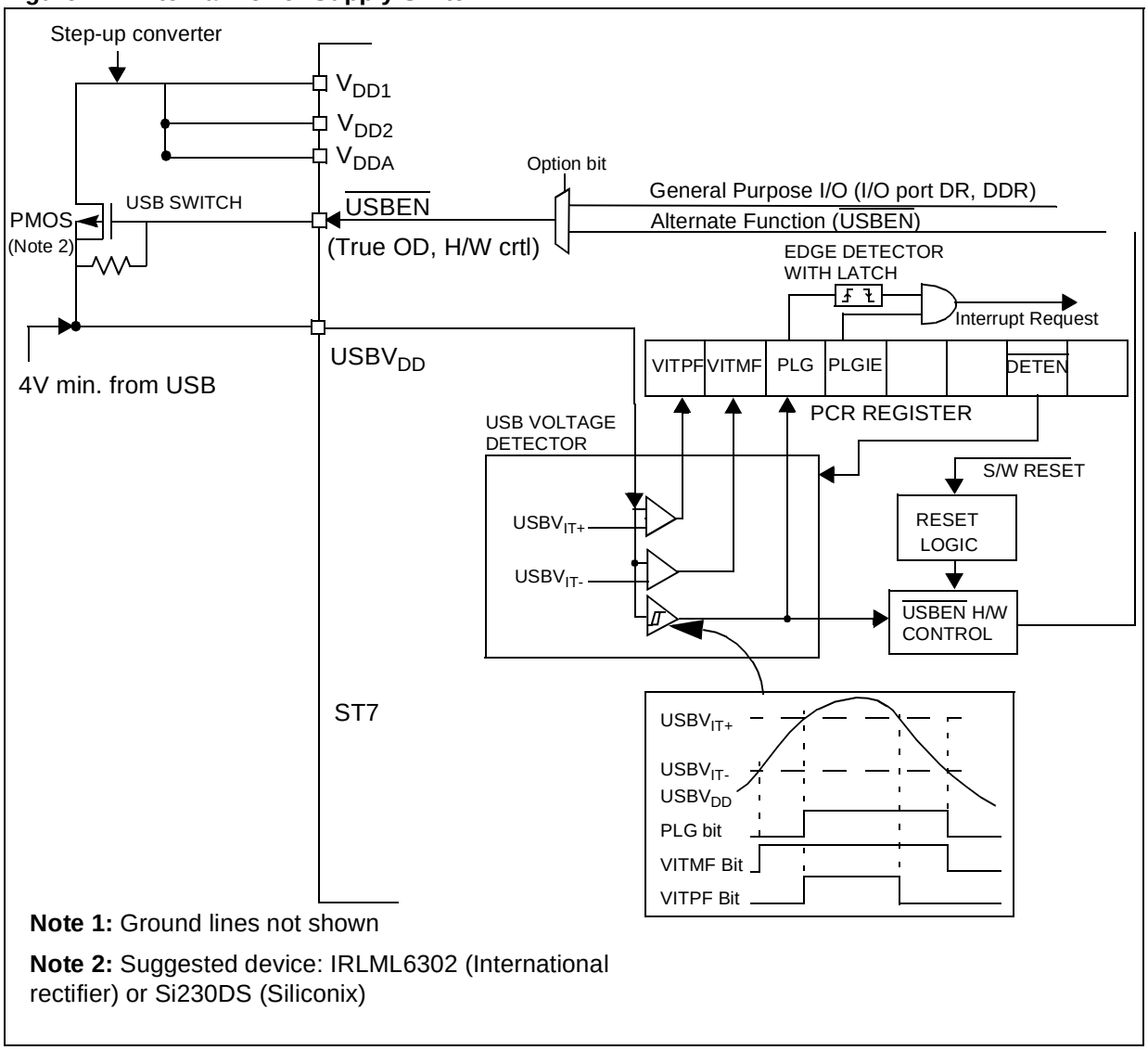
This puts the microcontroller into reset state and all I/O ports go into input high impedance mode.

During and after this (software induced) reset phase, the USBEN pin is set to output low level by hardware. This causes the USB SWITCH to be turned ON. Consequently, V_{DD} pin is powered by $USBV_{DD}$ supply. See Figure 22.

Once in USB mode, no power is drawn from the step-up converter output.

For more details, refer to Figure 23.

Figure 22. External Power Supply Switch



Note 1: Ground lines not shown

Note 2: Suggested device: IRLML6302 (International rectifier) or Si230DS (Siliconix)

POWER SUPPLY MANAGEMENT (Cont'd)

6.4.2.2 Switching from USB Mode to Stand-Alone Mode

In USB Mode, when the user unplugs the USB cable, the voltage level drops on the $USBV_{DD}$ line. The on-chip Power Supply Manager generates a PLG interrupt when $USBV_{DD}$ reaches $USBV_{IT-}$. The user program then can finish the current processing, and MUST generate a software RESET.

Caution: Care should be taken as during this period the microcontroller clock is provided from the PLL output. Functionality in this mode is not guaranteed for voltages below V_{PLLmin}.

Software must ensure that the software $\overline{\text{RESET}}$ is generated before V_{DD}. drops below V_{PLLmin}. Failing to do this will cause the clock circuitry to stop, freezing the microcontroller operations.

Once the user program has executed the software reset, the microcontroller goes into reset state and all I/O ports go into floating input mode.

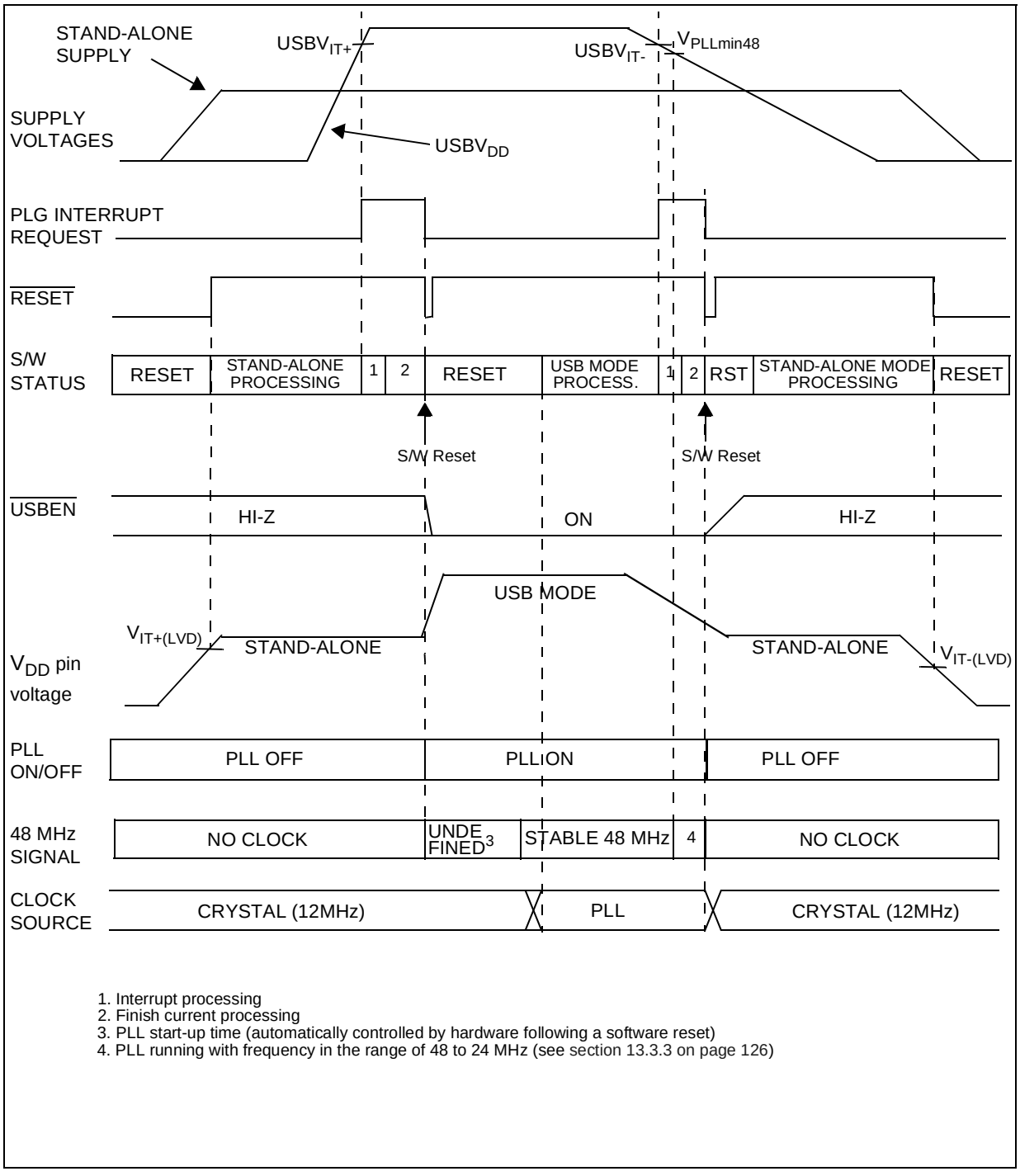
During and after this (software induced) reset phase, the USBEN pin is put in high impedance by hardware. It causes the USB SWITCH to be turned OFF, so $USBV_{DD}$ is disconnected from V_{DD}. The PLL is automatically stopped and the internal frequency is provided by a division of the crystal frequency. Refer to Figure 23.

The microcontroller is still powered by the residual $USBV_{DD}$ voltage (higher than step-up converter set output level). This V_{DD} voltage decreases during the reset phase until it reaches the step-up converter set output voltage. At that time, step-up converter resumes operation, and powers the application.

Caution: In order to avoid applying excessive voltage to the Storage Media, a minimum delay must be ensured during (and after if needed) the reset phase, prior to switching ON the external STORAGE switch. See Figure 24 and Figure 25.

POWER SUPPLY MANAGEMENT (Cont'd)

Figure 23. Power Supply Management: Dual Power Supply



1. Interrupt processing
2. Finish current processing
3. PLL start-up time (automatically controlled by hardware following a software reset)
4. PLL running with frequency in the range of 48 to 24 MHz (see section 13.3.3 on page 126)

POWER SUPPLY MANAGEMENT (Cont'd)

6.4.3 Storage Media Interface I/Os

The microcontroller is able to drive Storage Media through an interface operating at a different voltage from the rest of the circuit.

This is achieved by powering the Storage Media interface I/O circuitry through a specific supply rail connected to V_{DDF} pin. The V_{DDF} pin can be used either as an input or output.

If the on-chip voltage regulator is off, power to the interface I/Os should be provided externally to the V_{DDF} pin. This should be the case when in Stand-Alone Mode, or in USB mode when the current required to power the Storage Media is above the current capacity of the on-chip regulator.

If the on-chip voltage regulator is on, it powers the interface I/Os, and V_{DDF} pin can supply the Storage Media. This is recommended in USB Mode, when the current required to power the Storage Media is within the capacity of the on-chip regulator.

Application Example:

Stand-Alone Mode

– The Storage Media interface supply is powered by V_{DD} enabled by an external switch (see Figure 24) which connects V_{DD} to V_{DDF}. This switch can be driven by any True Open Drain I/O pin and controlled by user software.

– The on-chip voltage regulator must be disabled to avoid any conflict and to decrease consumption (reset the REGEN bit in the PCR register).

USB Mode

– In this case the core of the microcontroller is running from the USB bus power or the self power supply. V_{DD} and $USBV_{DD}$ pins are supplied with a voltage from 4.0 to 5.5V.

– The Storage Media Interface can be powered through the on-chip regulator (providing power to the I/O pins and output on pin V_{DDF}) if the current requirement is within the output capacity of the on chip regulator.

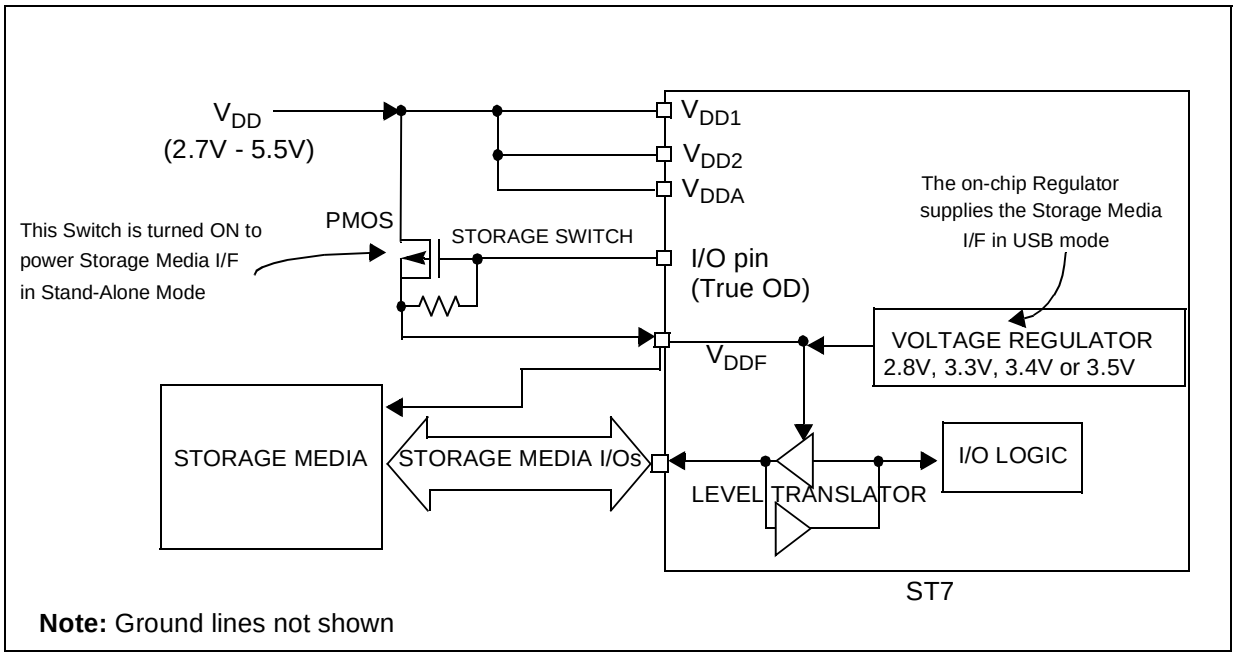
– The regulator output voltage can be programmed to 2.8V, 3.3V, 3.4V or 3.5 Volts, depending on the Storage Media specifications. (see VSET[1:0] bits in PCR register description)

– Should the current requirement for the Storage Media be higher than the current capacity of the on chip regulator, an external regulator should be used (See Figure 25). Thus the on-chip voltage regulator must be disabled to avoid any conflict (reset the REGEN bit in the PCR register).

Caution: The user should ensure that V_{DD} does not exceed the maximum rating specified for the Storage Media V_{DDF} max when switching STORAGE switch on.

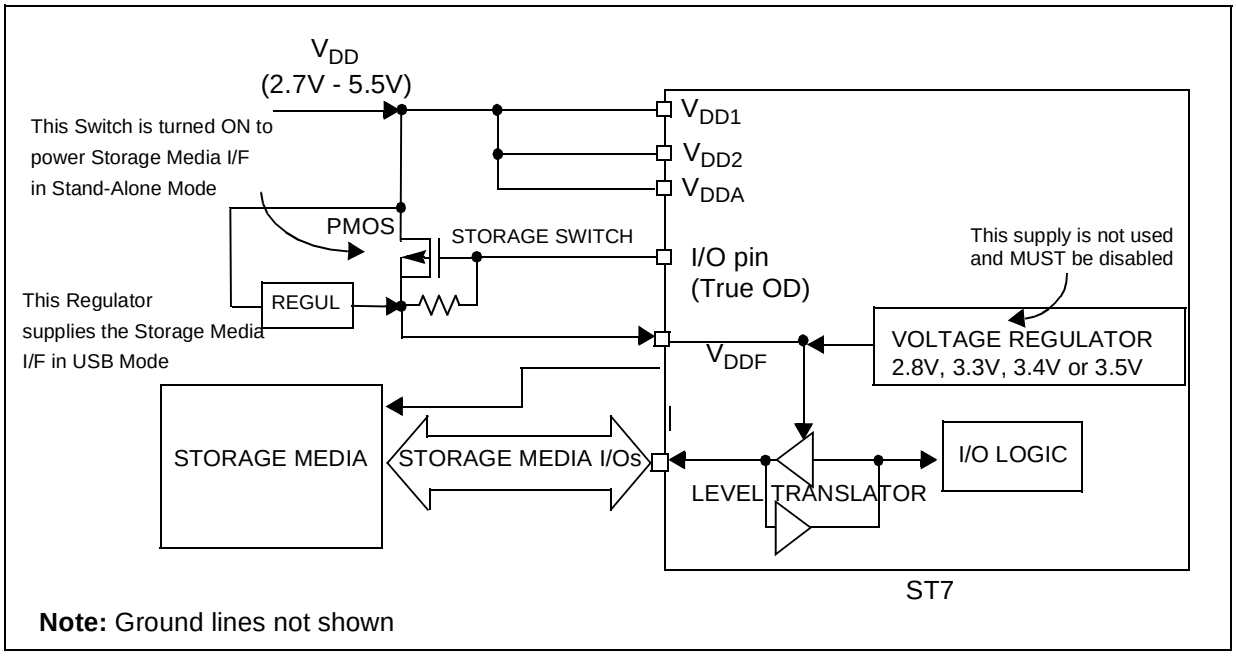
POWER SUPPLY MANAGEMENT (Cont'd)

Figure 24. Storage Media Interface Supply Switch (for low current Media)



Note: Ground lines not shown

Figure 25. Storage Media Interface Supply Switch (for high current Media)



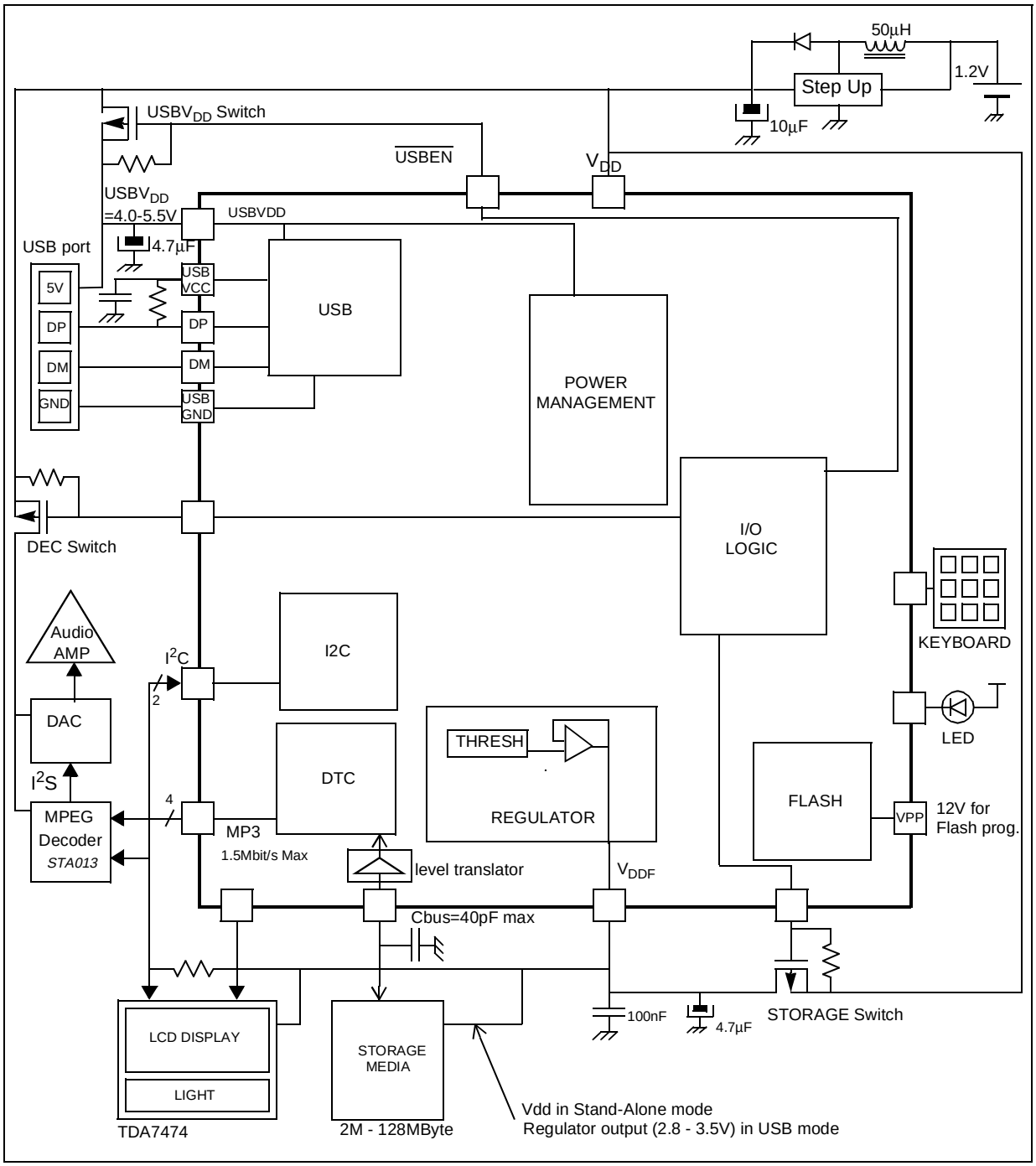
Note: Ground lines not shown

POWER SUPPLY MANAGEMENT (Cont'd)

6.4.4 Power Management Application Example

In the example shown in Figure 26, the V_{DD} supply is provided by a step up. In this case the step up must be capable of tolerating voltages up to 5.5V on its Vout pin.

Figure 26. Dual Power Supply Application Example (low current Storage Media)



POWER SUPPLY MANAGEMENT (Cont'd)

6.4.5 Register Description

POWER CONTROL REGISTER (PCR)

Reset Value: 0000 0000 (00h)

7							0
ITPF	ITMF	PLG	PLGIE	VSET1	VSET0	$\overline{DET}$ $\overline{EN}$	REGEN

Bit 7 = **ITPF** *Voltage Input Threshold Plus Flag*
This bit is set by hardware when $USBV_{DD}$ rises over $USBV_{IT+}$ and cleared by hardware when $USBV_{DD}$ drops below $USBV_{IT+}$.
0: $USBV_{DD} < USBV_{IT+}$
1:$USBV_{DD} > USBV_{IT+}$

Bit 6 = **ITMF** *Voltage Input Threshold Minus Flag*
This bit is set by hardware when $USBV_{DD}$ rises over $USBV_{IT-}$ and cleared by hardware when $USBV_{DD}$ drops below $USBV_{IT-}$.
0: $USBV_{DD} < USBV_{IT-}$
1:$USBV_{DD} > USBV_{IT-}$

Bit 5 =**PLG** *USB Plug/Unplug detection.*
This bit is set by hardware when it detects that the USB cable has been plugged in. It is cleared by hardware when the USB cable is unplugged. (Detection happens when $USBV_{DD}$ rises over $USBV_{IT+}$ or when $USBV_{DD}$ drops below $USBV_{IT-}$). If the PLGIE bit is set, the rising edge of the PLG bit also generates an interrupt request.
0: USB cable unplugged
1: USB cable plugged in

Bit 4 = **PLGIE** *USB Plug/Unplug Interrupt Enable.*
This bit is set and cleared by software.
0: Single supply mode: PLG interrupt disabled.
1: Dual supply mode: PLG interrupt enabled (generates an interrupt on the rising edge of PLG).

Bit 3:2 = **VSET[1:0]** *Voltage Regulator Output Voltage.*

These bits are set and cleared by software to select the output voltage of the on-chip voltage regulator (for the V_{DDF} output).

VSET1	VSET0	Voltage output of the regulator
0	0	3.5V
0	1	3.4V
1	0	3.3V
1	1	2.8V

Bit 1 = $\overline{\textbf{DETEN}}$ *USB Voltage Detector Enable.*
This bit is set and cleared by software. It is used to power-off the USB voltage detector in Stand-alone mode.
0: The USB voltage detector is enabled.
1: The USB voltage detector disabled (ITPF, ITMF and PLG bits are forced high)

Bit 0 = **REGEN** *Voltage Regulator Enable.*
This bit is set and cleared by software.
0: The regulator is completely shutdown and no current is drawn from the power supply by the voltage reference.
1: The on-chip voltage regulator is powered-on.

7 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
  - 3 non maskable events: TLI, 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 7). The processing flow is shown in Figure 27.

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).

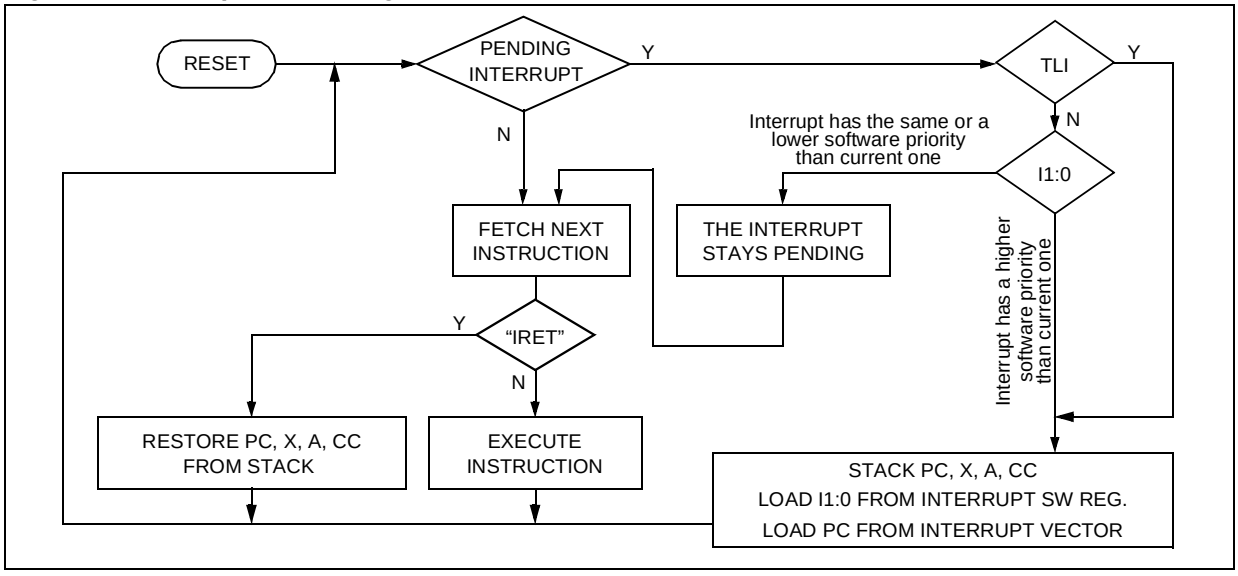
The interrupt service routine should end with the IRET instruction which causes the contents of the saved registers to be recovered from the stack.

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 7. 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 27. Interrupt Processing Flowchart



INTERRUPTS (Cont'd)

Servicing Pending Interrupts

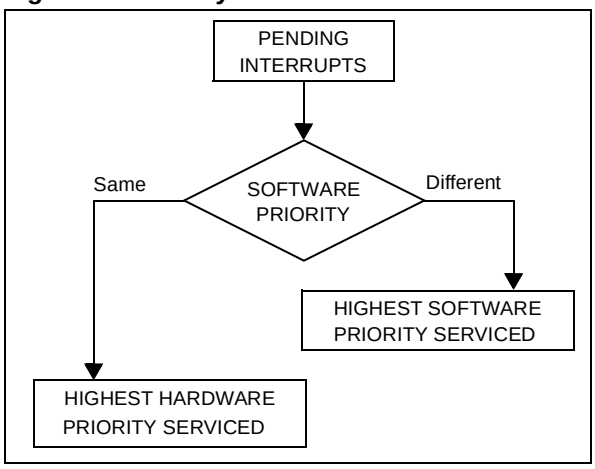
As several interrupts can be pending at the same time, the interrupt to be taken into account is determined by the following two-step process:

– the highest software priority interrupt is serviced,

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

Figure 28 describes this decision process.

Figure 28. Priority Decision Process



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

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, TRAP and TLI 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, TLI, 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 27). 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.

■ TLI (Top Level Hardware Interrupt)

This hardware interrupt occurs when a specific edge is detected on the dedicated TLI pin.
Caution: A TRAP instruction must not be used in a TLI service routine.

■ 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 27 as a TLI.
Caution: TRAP can be interrupted by a TLI.

■ 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 NANDed.

■ Peripheral Interrupts

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

INTERRUPTS (Cont'd)

7.3 INTERRUPTS AND LOW POWER MODES

All interrupts allow the processor to exit the WAIT low power mode. On the contrary, only external and other specified interrupts allow the processor to exit from the HALT modes (see column "Exit from HALT" in "Interrupt Mapping" table). When several pending interrupts are present while exiting HALT mode, the first one serviced can only be an interrupt with exit from HALT mode capability and it is selected through the same decision process shown in Figure 28.

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 29 and Figure 30 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 30. The interrupt hardware priority is given in this order from the lowest to the highest: MAIN, IT4, IT3, IT2, IT1, IT0, TLI. The software priority is given for each interrupt.

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

Figure 29. Concurrent Interrupt Management

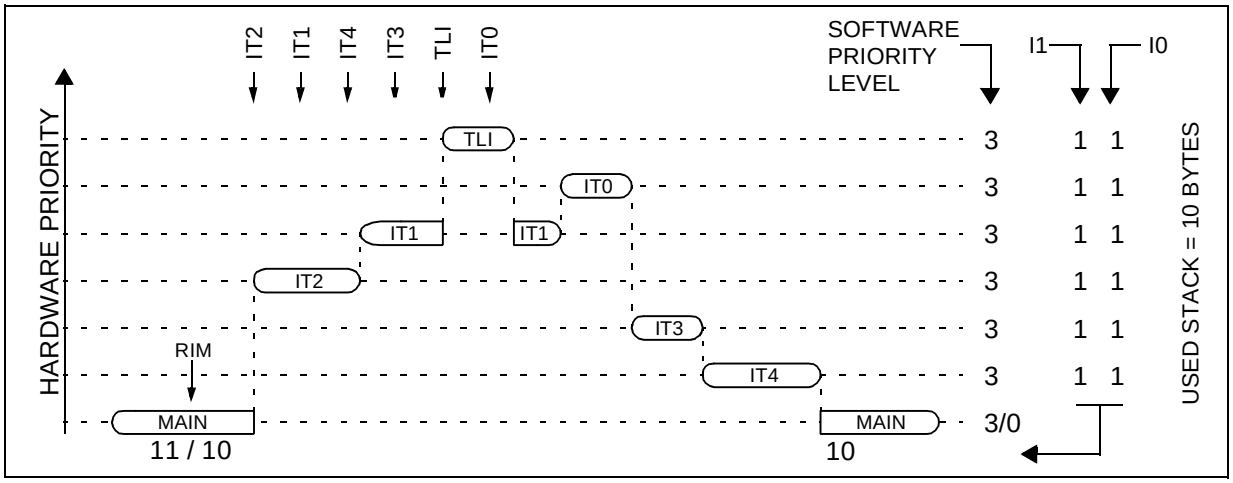
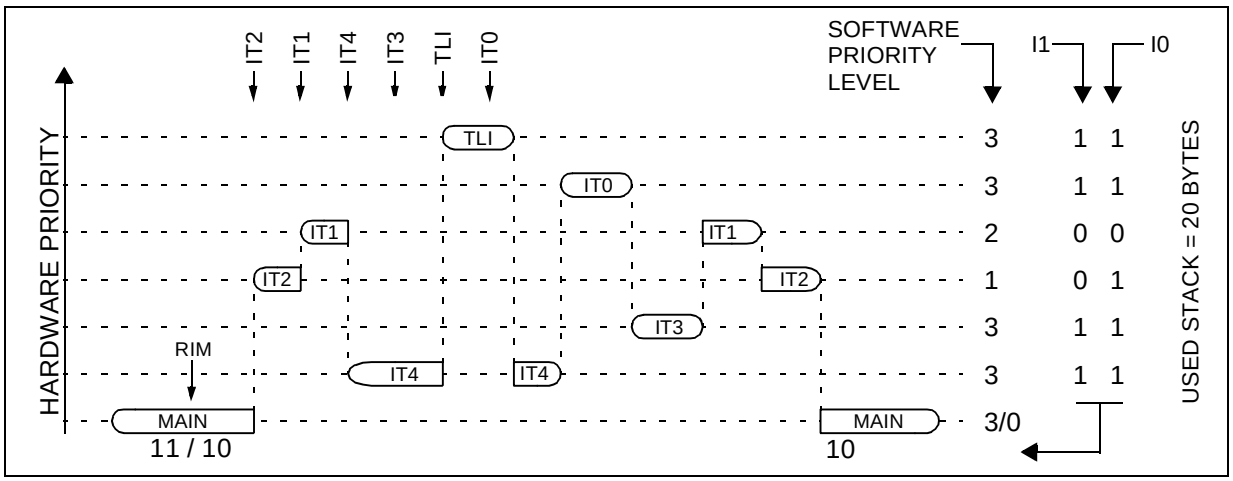


Figure 30. Nested Interrupt Management



INTERRUPTS (Cont'd)

7.5 INTERRUPT REGISTER DESCRIPTION

CPU CC REGISTER INTERRUPT BITS

Read/Write

Reset Value: 111x 1010 (xAh)

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

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

These two bits indicate the current interrupt software priority.

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

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

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

***Note**: TLI, 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 correspondance 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, TRAP and TLI vectors have no software priorities. When one is serviced, the I1 and I0 bits of the CC register are both set.

***Note**: Bits in the ISPRx registers which correspond to the TLI can be read and written but they are not significant in the interrupt process management.

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

INTERRUPTS (Cont'd)

Table 8. 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.

Table 9. Interrupt Mapping

N°	Source Block	Description	Register Label	Priority Order	Exit from HALT	Address Vector
	RESET	Reset	N/A	Highest Priority	yes	FFFEh-FFFFh
	TRAP	Software Interrupt			no	FFFCh-FFFDh
0	ICP	Flash Start Programming NMI Interrupt			yes	FFFAh-FFFBh
1	PLG	Power Management USB Plug/Unplug	PCR		yes	FFF8h-FFF9h
2	EI0	External Interrupt Port A	N/A		yes	FFF6h-FFF7h
3	DTC	DTC Peripheral Interrupt	DTCSR		no	FFF4h-FFF5h
4	USB	USB Peripheral Interrupt	USBISTR		no	FFF2h-FFF3h
5	ESUSP	USB End Suspend Interrupt	USBISTR		yes	FFF0h-FFF1h
6	EI1	External Interrupt Port D	N/A		yes	FFEEh-FFEFh
7	I^2C	I^2C Interrupt	I2CSRx		no	FFECh-FFEDh
8	TIM	Timer interrupt	TSR		no	FFEAh-FFEBh
9	EI2	External Interrupt Port C	N/A	Lowest Priority	yes	FFE8h-FFE9h
10	SPI	SPI interrupt	SPICSR		yes	FFE6h-FFE7h

INTERRUPTS (Cont'd)

Table 10. Nested Interrupts Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
002Ch		DTC		EI0		PLG		ISP	
	ISPR0	I1_3	I0_3	I1_2	I0_2	I1_1	I0_1		
	Reset Value	1	1	1	1	1	1	1	1
002Dh		I^2C		EI1		ESUSP		USB	
	ISPR1	I1_7	I0_7	I1_6	I0_6	I1_5	I0_5	I1_4	I0_4
	Reset Value	1	1	1	1	1	1	1	1
002Eh		Not used		SPI		EI2		TIM	
	ISPR2	I1_11	I0_11	I1_10	I0_10	I1_9	I0_9	I1_8	I0_8
	Reset Value	1	1	1	1	1	1	1	1
002Fh						Not used		Not used	
	ISPR3					I1_13	I0_13	I1_12	I0_12
	Reset Value	1	1	1	1	1	1	1	1

8 POWER SAVING MODES

8.1 INTRODUCTION

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

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

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

The user can also switch off any unused on-chip peripherals individually by programming the MISCR2 register.

8.2 WAIT MODE

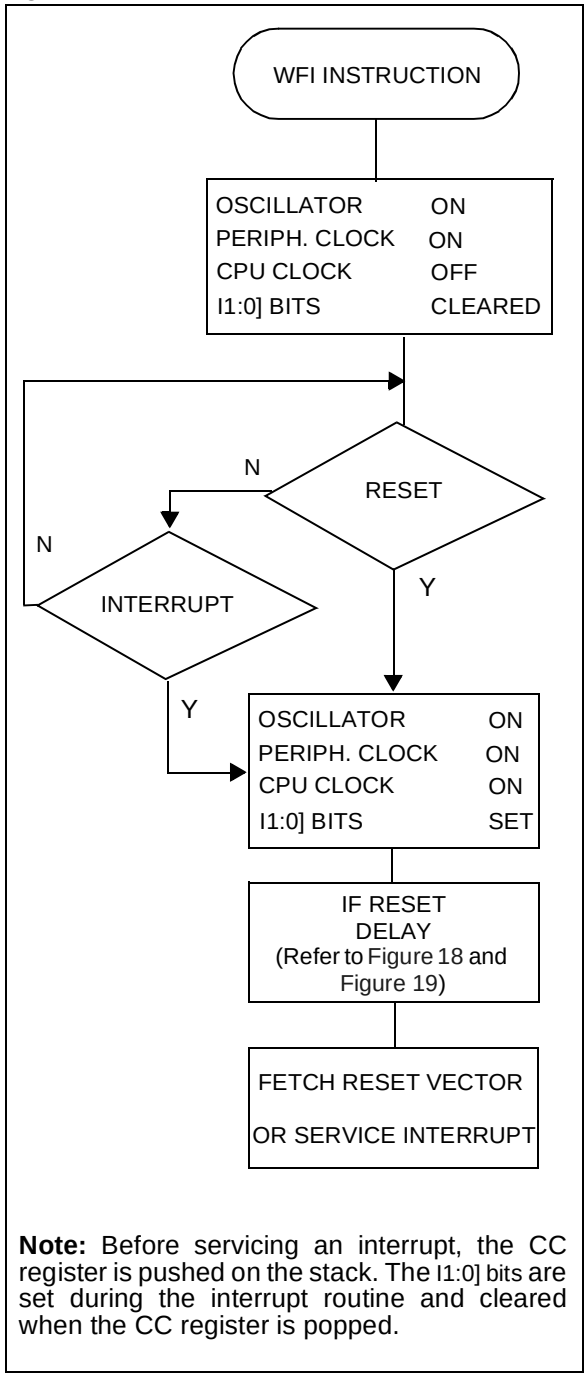
WAIT mode places the MCU in a low power consumption mode by stopping the CPU.

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

All peripherals remain active. During WAIT mode, the I1:0] bits in the CC register are forced to 0, to enable all interrupts. All other registers and memory remain unchanged. The MCU remains in WAIT mode until an interrupt or Reset occurs, whereupon the Program Counter branches to the starting address of the interrupt or Reset service routine. The MCU will remain in WAIT mode until a Reset or an Interrupt occurs, causing it to wake up.

Refer to Figure 31.

Figure 31. WAIT Mode Flow Chart



Note: Before servicing an interrupt, the CC register is pushed on the stack. The I1:0] bits are set during the interrupt routine and cleared when the CC register is popped.

POWER SAVING MODES (Cont'd)

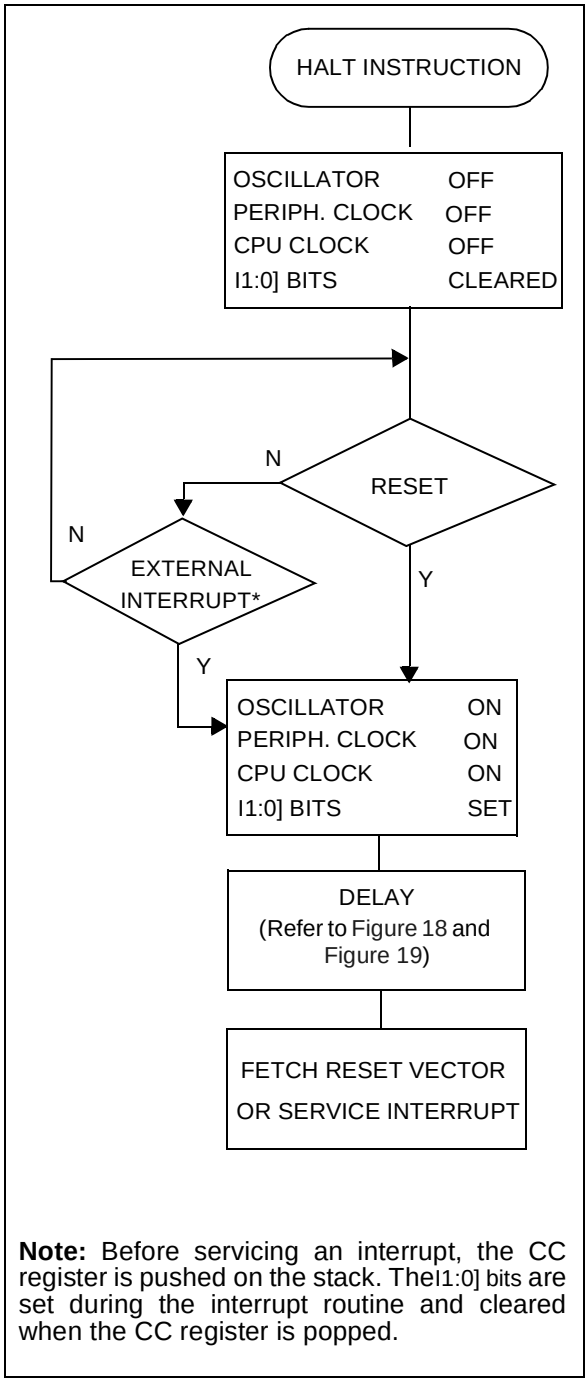
8.3 HALT MODE

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

When entering HALT mode, the I1:0] bits in the Condition Code Register are cleared. Thus, any of the external interrupts (ITi or USB end suspend mode), are allowed and if an interrupt occurs, the CPU clock becomes active.

The MCU can exit HALT mode on reception of either an external interrupt on ITi, an end suspend mode interrupt coming from USB peripheral, or a reset. The oscillator is then turned on and a stabilization time is provided before releasing CPU operation. The stabilization time is 512 CPU clock cycles.
After the start up delay, the CPU continues operation by servicing the interrupt which wakes it up or by fetching the reset vector if a reset wakes it up.

Figure 32. HALT Mode Flow Chart



Note: Before servicing an interrupt, the CC register is pushed on the stack. The I1:0] bits are set during the interrupt routine and cleared when the CC register is popped.

9 I/O PORTS

9.1 INTRODUCTION

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

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

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

9.2 FUNCTIONAL DESCRIPTION

Each port has 2 main registers:

– Data Register (DR)

– Data Direction Register (DDR)

and one optional register:

– Option Register (OR)

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

The following description takes into account the OR register, (for specific ports which do not provide this register refer to the I/O Port Implementation section). The generic I/O block diagram is shown in Figure 33

9.2.1 Input Modes

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

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

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

Notes:
1. Writing the DR register modifies the latch value but does not affect the pin status.
2. When switching from input to output mode, the DR register has to be written first to drive the correct level on the pin as soon as the port is configured as an output.

External interrupt function

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

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

Each external interrupt vector is linked to a dedicated group of I/O port pins (see pinout description and interrupt section). If several input pins are selected simultaneously as interrupt source, these are logically NANDed and inverted. For this reason if one of the interrupt pins is tied low, it masks the other ones.

In case of a floating input with interrupt configuration, special care must be taken when changing the configuration (see Figure 34).

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

9.2.2 Output Modes

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

DR register value and output pin status:

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

The output configuration is selected by setting the corresponding DDR register bit. In this case, writing the DR register applies this digital value to the I/O pin through the latch. Reading the DR register returns the digital value present on the external I/O pin. Consequently even in output mode a value written to an open drain port may differ from the value read from the port. For example, if software writes a ‘1’ in the latch, this value will be applied to the pin, but the pin may stay at ‘0’ depending on the state of the external circuitry. For this reason, bit manipulation even using instructions like BRES and BSET must not be used on open drain ports as they work by reading a byte, changing a bit and writing back a byte. A workaround for applications requiring bit manipulation on Open Drain I/Os is given in the note below Table 13.

9.2.3 Alternate Functions

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

I/O PORTS (Cont'd)

When the signal is coming from an on-chip peripheral, the I/O pin is automatically configured in output mode (push-pull or open drain according to the peripheral).

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

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

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

Analog alternate function

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

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

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

I/O PORTS (Cont'd)

Figure 33. I/O Port General Block Diagram

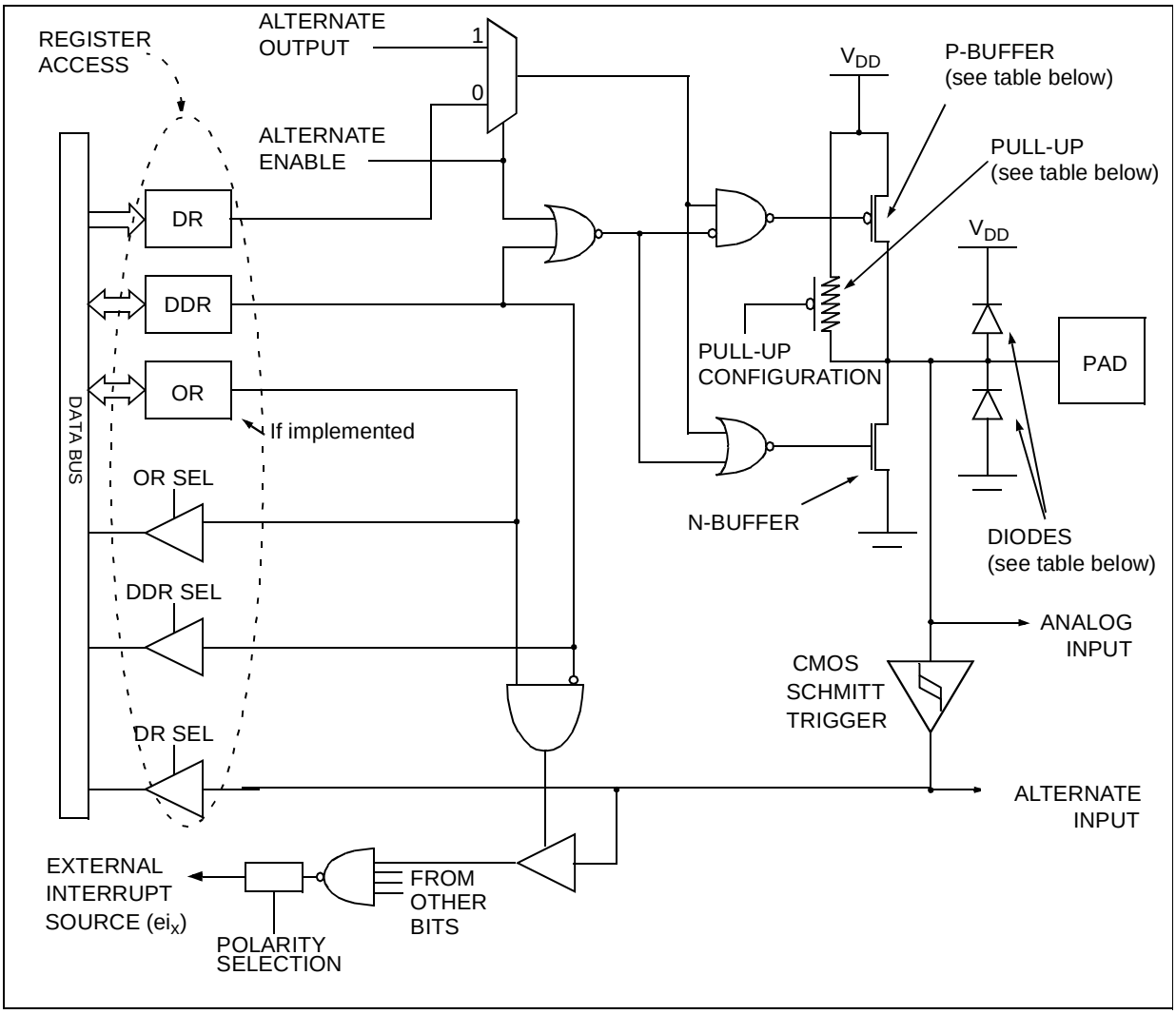


Table 11. I/O Port Mode Options

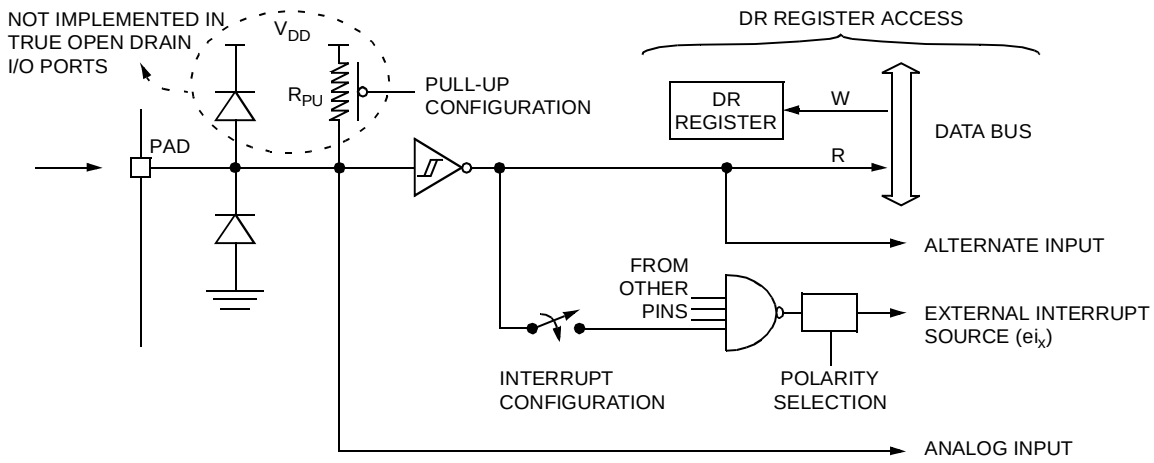
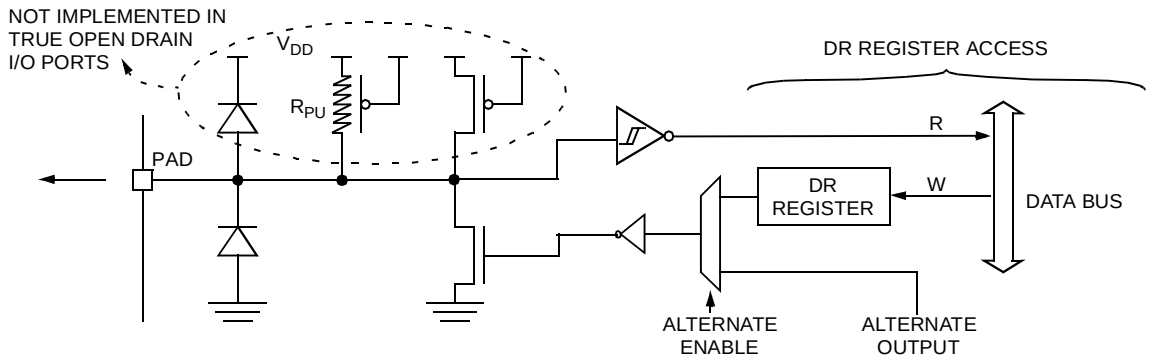
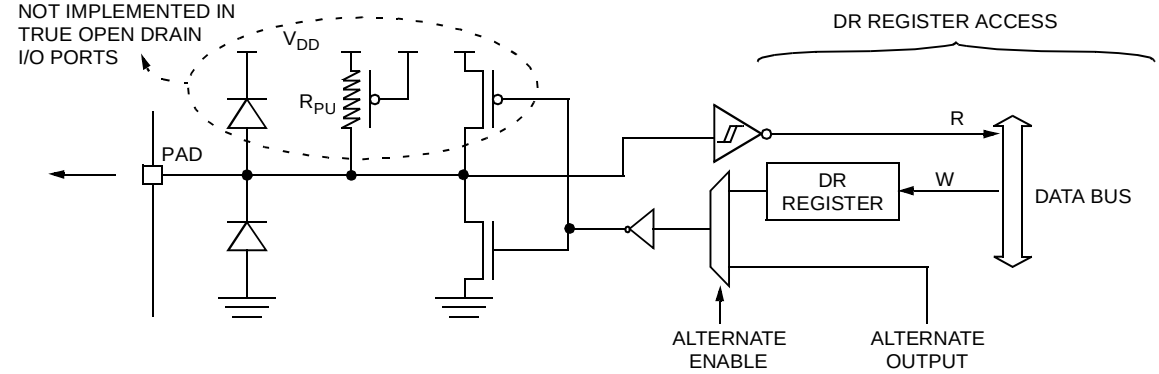
Configuration Mode		Pull-Up	P-Buffer	Diodes	
				to V_{DD}	to V_{SS}
Input	Floating with/without Interrupt	Off	Off	On	On
	Pull-up with/without Interrupt	On			
Output	Push-pull	Off	On		
	Open Drain (logic level)		Off		
	True Open Drain	NI	NI	NI (see note)	

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

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

I/O PORTS (Cont'd)

Table 12. I/O Port Configurations

	Hardware Configuration
INPUT [1]	
OPEN-DRAIN OUTPUT [2]	
PUSH-PULL OUTPUT [2]	

Notes:

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

I/O PORTS (Cont'd)

9.2.4 Bit manipulation on Open Drain Outputs

As mentioned in Section 9.2.2, software should avoid using bit manipulation instructions on the DR register in open drain output mode, but must always access it using byte instructions. If bit manipulation is needed, one solution is to use a copy of the DR register in RAM, change the bits (using BRES or BCLR instructions for example) and copy the whole byte into the DR register each time the value has to be output on a port. This way, no bit manipulation is performed on the DR register but each bit of the DR register can be controlled separately.

A second solution is to reverse the roles of the DDR and DR registers, and use the DDR register instead of the DR register to control the output value as follows:

– Write '1' into the OR register bits corresponding to the I/O ports to be used as open drain outputs (Push-Pull configuration).

– Write '0' into the corresponding DR register bits.

– Use the DDR register bits to control the value on the I/O port (see Table 13).

Table 13. Special DDR Control for OD ports

DR	OR	DDR	Effect on open-drain I/O port
0	1	1	A value of 0 is output
0	1	0	The port is floating (or has a value of 1 if an external pull-up is used)

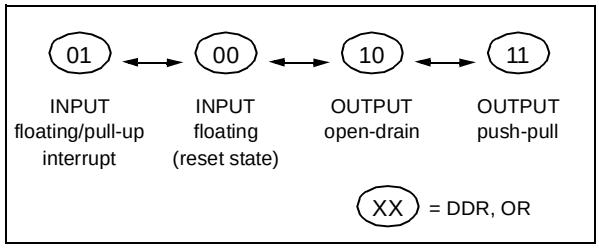
This is equivalent to using the open drain configuration (OR=0) and using the DR register to control the output value. This way each bit of DDR register can be modified separately without the risk of corrupting the other bits of the port.

9.3 I/O PORT IMPLEMENTATION

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

Switching these I/O ports from one state to another should be done in a sequence that prevents unwanted side effects. Recommended safe transitions are illustrated in Figure 34 Other transitions are potentially risky and should be avoided, since they are likely to present unwanted side-effects such as spurious interrupt generation.

Figure 34. Interrupt I/O Port State Transitions



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

Port B (without Option Register)

PB[7:0]

MODE	DDR
floating input	0
push-pull output	1

I/O PORTS (Cont'd)

Table 14. Port Configuration (with Option Register)

Port	Pin name	Input		Output		
		OR = 0	OR = 1	OR = 0	OR = 1	High-Sink
Port A	PA7:0	floating	floating with interrupt	open drain	push-pull	No
Port C	PC7:4	floating	floating with interrupt	push-pull		No
	PC3:0	floating	floating with interrupt	push-pull		Yes
Port D	PD7:0	floating	floating with interrupt	open drain	push-pull	No
Port E	PE7:6	floating		open drain	push-pull	Yes
	PE5	floating	with pull-up, if selected by option byte see Section 15.1)	open drain (with pull-up, if selected by option byte see Section 15.1)	push-pull	Yes
	PE4:3	floating		open drain	push-pull	No
	PE2:0	floating		open drain	push-pull	Yes
Port F	PF6:4	floating		True open drain		Yes
	PF3:2	floating		push-pull		No
	PF1:0	floating		True open drain		Yes

I/O PORTS (Cont'd)

9.4 Register Description

DATA REGISTER (DR)

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

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 returns either the DR register latch content (pin configured as output) or the digital value applied to the I/O pin (pin configured as input).

DATA DIRECTION REGISTER (DDR)

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

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, 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 (ports A, C and D). For port E configuration, refer to Table 14.

Output mode:
0: Output open drain (with P-Buffer deactivated)
1: Output push-pull

I/O PORTS (Cont'd)

Table 15. I/O Port Register Map and Reset Values

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

10 MISCELLANEOUS REGISTERS

MISCELLANEOUS REGISTER 1 (MISCR1)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
IS11	IS10	MCO	IS21	IS20	CP1	CP0	CPEN

Bits 7:6 = **IS1[1:0]** *ei0 Interrupt sensitivity*
Interrupt sensitivity, defined using the IS1[1:0] bits, is applied to the ei0 interrupts (Port A):

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

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

Bit 5 = **MCO** *Main clock out selection*
This bit enables the MCO alternate function on the I/O port. It is set and cleared by software.
0: MCO alternate function disabled (I/O pin free for general-purpose I/O)
1: MCO alternate function enabled (f_{CPU} output on I/O port)

Bits 4:3 = **IS2[1:0]** *ei1 Interrupt sensitivity*
Interrupt sensitivity, defined using the IS2[1:0] bits, is applied to the ei1 external interrupts (Port D):

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

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

Bits 2:1 = **CP[1:0]** *CPU clock prescaler*
These bits select the CPU clock prescaler which is applied in the different slow modes. Their action is conditioned by the setting of the CPEN bit. These two bits are set and cleared by software

Operating Mode	f_{CPU}	CP1	CP0	CPEN
Stand-alone mode (f_{OSC} = 12 MHz)	3 MHz	x	x	0
	6 MHz*	0	0	1
	1.5 MHz	1	0	1
	750 KHz	0	1	1
	375 KHz	1	1	1
USB mode (48 MHz PLL)	6 MHz	x	x	0
	8 MHz	0	0	1
	2 MHz	1	0	1
	1 MHz	0	1	1
	250 KHz	1	1	1

Caution:
– The ST7 core is not able to read or write in the USB data buffer if the ST7265x is configured at 6 MHz in standalone mode.
– In USB mode, with $f_{CPU} \leq 2$ MHz, if the ST7 core accesses the USB data buffer, this may prevent the USB interface from accessing the buffer, resulting in a USB buffer overrun error. This is because an access to memory lasts one cycle and the USB has to send/receive at a fixed baud rate.

Bit 0 = **CPEN** *Clock Prescaler Enable*
This bit is set and cleared by software. It is used with the CP[1:0] bits to configure the internal clock frequency.
0: Default f_{CPU} used (3 or 6 MHz)
1: f_{CPU} determined by CP[1:0] bits

MISCELLANEOUS REGISTERS (Cont'd)

MISCELLANEOUS REGISTER 2 (MISCR2)

Reset Value: 0000 0000 (00h)

7							0
0	0	0	P4	P3	P2	P1	P0

Bits 7:5 = Reserved.

Bits 4:0 = **P[4:0]** *Power Management Bits*
These bits are set and cleared by software. They can be used to switch the on-chip peripherals of the microcontroller ON or OFF. The registers are not changed by switching the peripheral OFF and then ON (contents are frozen while OFF).
0: Peripheral ON (running)
1: Peripheral OFF

Bit	Peripheral
P0	PWM
P1	Timer
P2	I2C
P3	USB
P4	DTC

MISCELLANEOUS REGISTER 3 (MISCR3)

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

7							0
WDG HALT	0	0	0	IS31	IS30	PWM 1	PWM 0

Bit 7 = **WDGHALT** *Watchdog and HALT Mode*
This bit is set and cleared by software. It determines if a RESET is generated when entering Halt mode while the Watchdog is active (WDGA bit =1 in the WDGCR register).

In either case, the Watchdog will not reset the MCU if a HALT instruction is executed while the USB is in Suspend mode.
0: If the Watchdog is active, it will reset the MCU if a HALT instruction is executed (unless the USB is in Suspend mode)
1: When a HALT instruction is executed, the MCU will enter Halt mode (without generating a reset) even if the Watchdog is active.

Bits 6:4 = Reserved, forced by hardware to 0.

Bits 3:2= IS3[1:0] *ei2 Interrupt sensitivity*
Interrupt sensitivity, defined using the IS3[1:0] bits, is applied to the ei2 interrupts (Port C):

IS31	IS30	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 must be written only when I1 and I0 of the CC register are both set to 1 (level 3).

Bit 1 = **PWM1** *PWM1 Output Control*
0: PWM1 Output alternate function disabled (I/O pin free for general purpose I/O).
1: PWM1 Output alternate function enabled

Bit 0 = **PWM0** *PWM0 Output Control*
0: Output alternate function disabled (I/O pin free for general purpose I/O).
1: PWM0 Output alternate function enabled

11 ON-CHIP PERIPHERALS

11.1 WATCHDOG TIMER (WDG)

11.1.1 Introduction

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

11.1.2 Main Features

- Programmable timer (64 increments of 65536 CPU cycles)
- Programmable reset
- Reset (if watchdog activated) when the T6 bit reaches zero
- Hardware Watchdog selectable by option byte

11.1.3 Functional Description

The counter value stored in the CR register (bits T[6:0]), is decremented every 65,536 machine cycles, and the length of the timeout period can be programmed by the user in 64 increments.

If the watchdog is activated (the WDGA bit is set) and when the 7-bit timer (bits T[6:0]) rolls over from 40h to 3Fh (T6 becomes cleared), it initiates a reset cycle pulling low the reset pin for typically 500ns.

The application program must write in the CR register at regular intervals during normal operation to prevent an MCU reset. The value to be stored in the CR register must be between FFh and C0h (see Table 16):

– The WDGA bit is set (watchdog enabled)

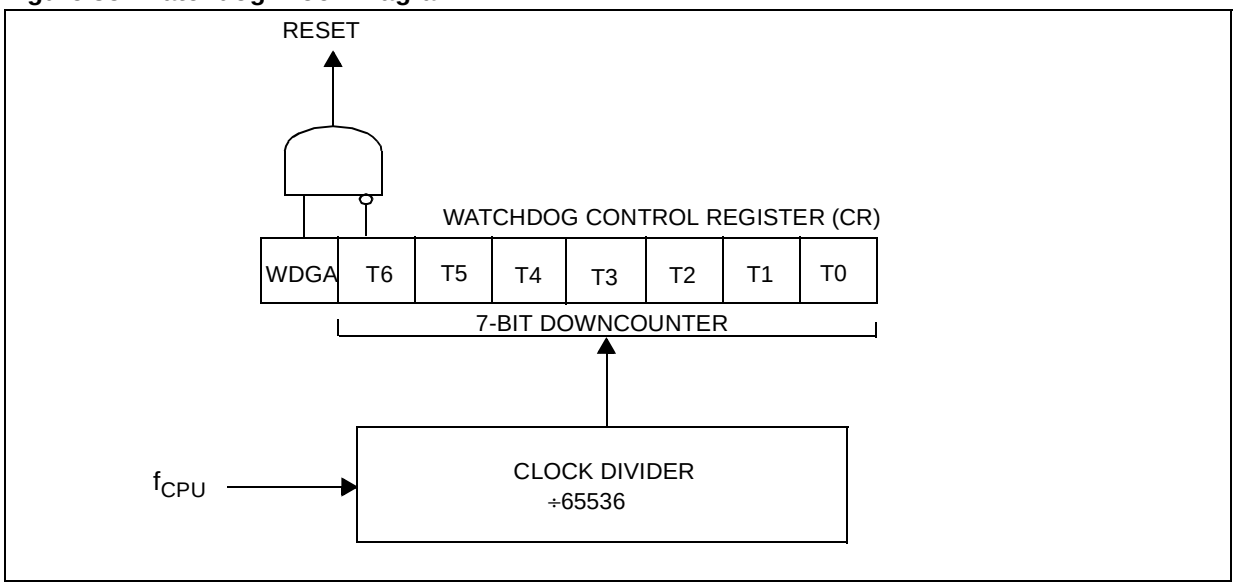
– The T6 bit is set to prevent generating an immediate reset

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

Table 16.Watchdog Timing (f_{CPU} = 8 MHz)

	CR Register initial value	WDG timeout period (ms)
Max	FFh	524.288
Min	C0h	8.192

Figure 35. Watchdog Block Diagram



WATCHDOG TIMER (Cont'd)

11.1.4 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).

11.1.5 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.

11.1.6 Low Power Modes

WAIT Instruction

No effect on Watchdog.

HALT Instruction

If the WDGHALT bit in the MISCR3 register is set, Halt mode can be used when the watchdog is enabled. When the oscillator is stopped, the WDG stops counting and is no longer able to generate a reset until the microcontroller receives an external interrupt or a reset.

If an external interrupt is received, the WDG restarts counting after 514 CPU clocks. In the case of the Software Watchdog option, if a reset is generated, the WDG is disabled (reset state).

Note: In USB mode, and in Suspend mode, a reset is not generated by entering Halt mode

Recommendations

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

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

– When using an external interrupt to wake up the microcontroller, reinitialize the corresponding I/O as Input 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.

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

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

11.1.7 Interrupts

None.

11.1.8 Register Description

CONTROL REGISTER (CR)

Read/Write

Reset Value: 0111 1111 (7Fh)

7							0
WDGA	T6	T5	T4	T3	T2	T1	T0

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

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

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

Table 17. Watchdog Timer Register Map and Reset Values

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

11.2 DATA TRANSFER COPROCESSOR (DTC)

11.2.1 Introduction

The Data Transfer Coprocessor is a Universal Serial/Parallel Communications Interface. By means of software plug-ins provided by STMicroelectronics, the user can configure the ST7 to handle a wide range of protocols and physical interfaces such as:

– 8 or 16-bit IDE mode Compact Flash
– Multimedia Card (MMC protocol)
– SmartMediaCard
– Secure Digital Card

Support for different devices or future protocol standards does not require changing the microcontroller hardware, but only installing a different software plug-in.

Once the plug-in (up to 256 bytes) stored in the ROM or FLASH memory of the ST7 device is loaded in the DTC RAM, and that the DTC operation is started, the I/O ports mapped to the DTC assume specific alternate functions.

Main Features

- Full-Speed data transfer from USB to I/O ports without ST7 core intervention
- Protocol-independency
- Support for serial and parallel devices
- Maskable Interrupts

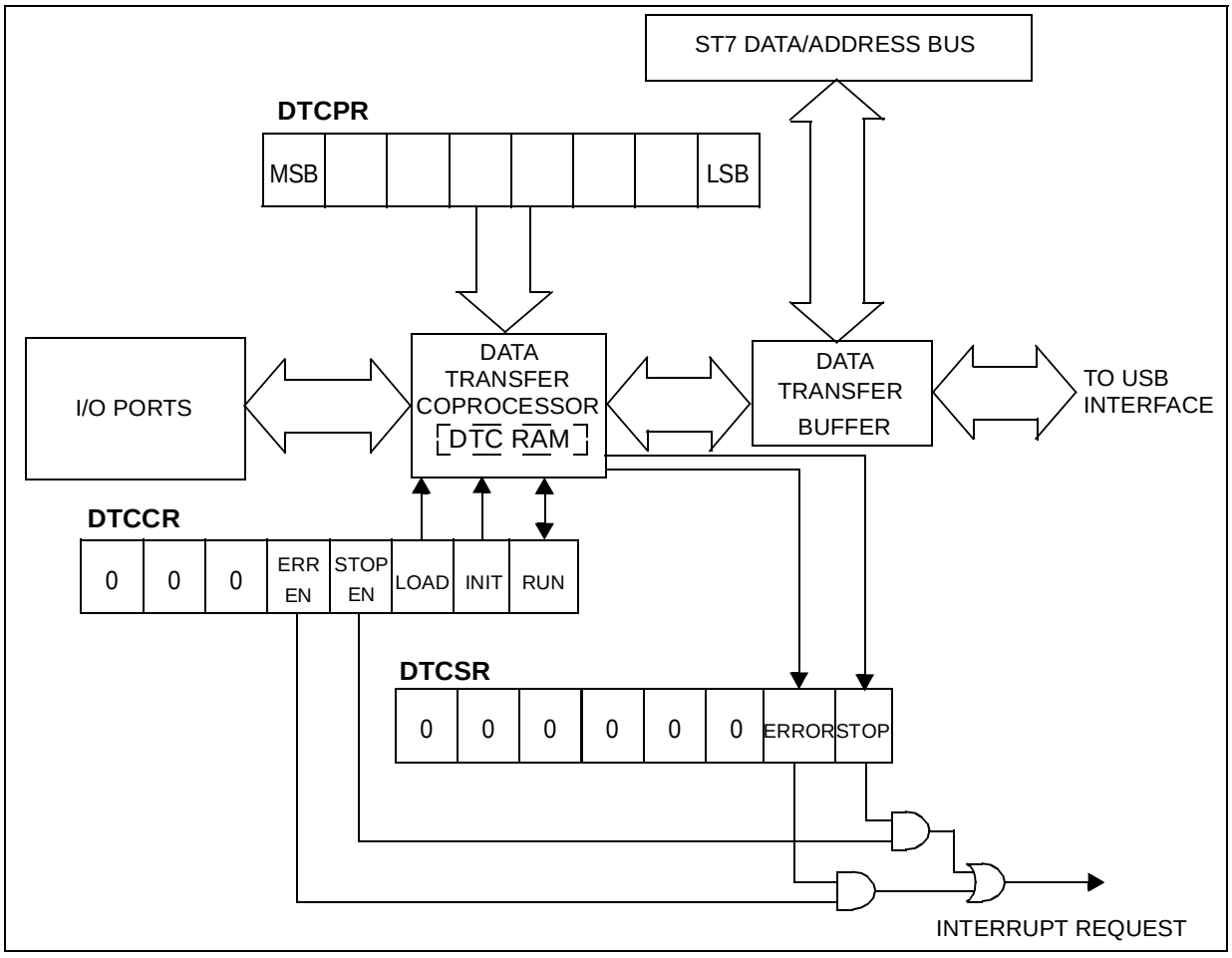
11.2.2 Functional Description

The block diagram is shown in Figure 36. The main function of the DTC is to quickly transfer data between :

- USB and ST7 I/O ports
- in between ST7 I/O ports

The protocol used to read or write from the I/O port is defined by the S/W plug-in in the DTC RAM.

Figure 36. DTC Block Diagram



Data Transfer Coprocessor (Cont'd)

When the USB interface is used, data transfer is typically controlled by a host computer.

The ST7 core can also read from and write to the data buffer of the DTC. Typically, the ST7 controls the application when the USB not used (autonomous mode). The buffer can potentially be accessed by any one of three requestors, the ST7, the DTC and the USB. Mastership of the buffer is not time limited. While a master is accessing the buffer, other requests will not be acknowleged until the buffer is freed by the master. If several requests are pending, when the buffer is free it is granted to the source with the highest priority in the daisy-chain (fixed by hardware), first the ST7, secondly the USB and finally the DTC.

Note: Any access by the ST7 to the buffer requires more cycles than either a DTC or USB access. For performance reasons, when the USB interface is exchanging data with the DTC, ST7 accesses should be avoided if possible.

11.2.3 Loading the Protocol Software

The DTC must first be initialized by loading the protocol-specific software plug-in (provided by STMicroelectronics) into the DTC RAM. To do this:

1. Stop the DTC by clearing the RUN bit in the DTCCR register
2. Remove the write protection by setting the LOAD bit in the DTCCR register
3. Load the (null-terminated) software plug-in in the DTC RAM.
4. Restore the write protection by clearing the LOAD bit in the DTCCR register

The DTC is then ready for operation.

11.2.4 Executing the Protocol Functions

To execute any of the software plug-in functions follow the procedure below:

1. Clear the RUN bit to stop the DTC
2. Select the function by writing its address in the DTCPR register (refer to the separate document for address information).
3. Set the INIT bit in the DTCCR register to copy the DTCPR pointer to the DTC.
4. Clear the INIT bit to return to idle state.
5. Set the RUN bit to start the DTC.

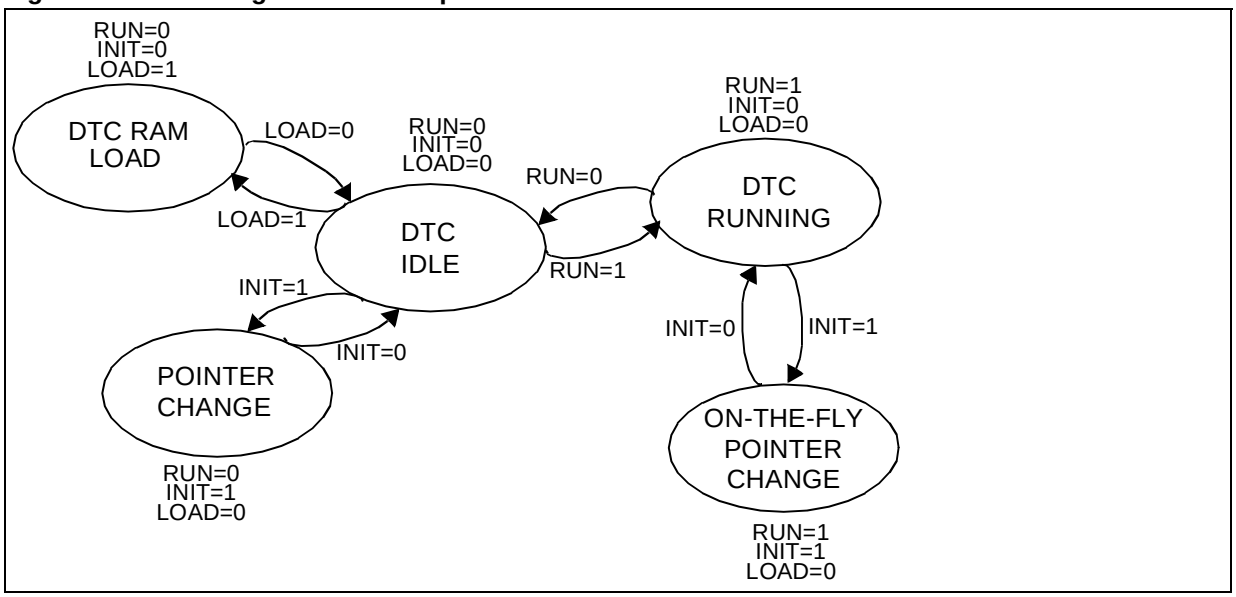
11.2.5 Changing the DTCPR pointer on the fly

As shown in Figure 37, the pointer can be changed by writing INIT=1 while the DTC is running (RUN=1), however if the DTC is executing an internal interrupt routine, there will be a delay until interrupt handling is completed.

11.2.6 Low Power Modes

Mode	Description
WAIT	No effect on DTC
HALT	DTC halted.

Figure 37. State Diagram of DTC Operations



Data Transfer Coprocessor (Cont'd)

11.2.7 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
Error	ERROR	ERREN	Yes	No
Stop	STOP	STOPEN	Yes	No

Note: The DTC interrupt events are connected to the same interrupt vector (see Interrupts chapter).

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

11.2.8 Register Description

DTC CONTROL REGISTER (DTCCR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
0	0	0	ERR EN	STOP EN	LOAD	INIT	RUN

Bit 7:5 = Reserved. Must be left at reset value.

Bit 4 = **ERREN** *Error Interrupt Enable*

This bit is set and cleared by software.
0: Error interrupt disabled
1: Error interrupt enabled

Bit 3 = **STOPEN** *Stop Interrupt Enable*

This bit is set and cleared by software.
0: Stop interrupt disabled
1: Stop interrupt enabled

Bit 2 = **LOAD** *Load Enable*

This bit is set and cleared by software. It can only be set while RUN=0.
0: Write access to DTC RAM disabled
1: Write access DTC RAM enabled

Bit 1 = **INIT** *Initialization*

This bit is set and cleared by software.
0: Do not copy DTCPR to DTC
1: Copy the DTCPR pointer to DTC

Bit 0 = **RUN** *START/STOP Control*

This bit is set and cleared by software. It can only be set while LOAD=0. It is also cleared by hardware when STOP=1
0: Stop DTC
1: Start DTC

DTC STATUS REGISTER (DTCSR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	ERROR	STOP

Bit 7:2 = Reserved. Forced by hardware to 0.

Bit 1 = **ERROR** *Error Flag*

This bit is set by hardware and cleared by software reading this register.
0: No Error event occurred
1: Error event occurred (DTC is running)

Bit 0 = **STOP** *Stop Flag*

This bit is set by hardware and cleared by software reading this register.
0: No Stop event occurred
1: Stop event occurred (DTC terminated execution at the current intruction)

DTC POINTER REGISTER (DTCPR)
Write Only
Reset Value: 0000 0000 (00h)

7							0
MSB							LSB

Bit 7:0 = **PC[7:0]** *Pointer Register.*

This register is written by software. It gives the address of an entry point in the protocol software that has previously been loaded in the DTC RAM.

Note: To start executing the function, after writing this address, set the INIT bit.

11.2.8.1 Data Transfer Coprocessor (Cont'd)

Table 18. DTC Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
1C	DTCCR	0 0	0 0	0 0	ERREN 0	STOPEN 0	LOAD 0	INIT 0	RUN 0
1D	DTCSR	0 0	0 0	0 0	0 0	0 0	0 0	ERROR 0	STOP 0
1F	DTCPR	MSB 0	0	0	0	0	0	0	LSB 0

11.3 USB INTERFACE (USB)

11.3.1 Introduction

The USB Interface implements a full-speed function interface between the USB and the ST7 microcontroller. It is a highly integrated circuit which includes the transceiver, 3.3 voltage regulator, SIE and USB Data Buffer interface. No external components are needed apart from the external pull-up on USBDP for full speed recognition by the USB host.

11.3.2 Main Features

- USB Specification Version 2.0 Compliant
- Supports Full-Speed USB Protocol
- Five Endpoints (including default endpoint)
- CRC generation/checking, NRZI encoding/ decoding and bit-stuffing
- USB Suspend/Resume operations
- Special Data transfer mode with USB Data Buffer Memory (2 x 512 bytes for upload or download) to DTC
- On-Chip 3.3V Regulator
- On-Chip USB Transceiver

11.3.3 Functional Description

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

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

Serial Interface Engine

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

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

Endpoints

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

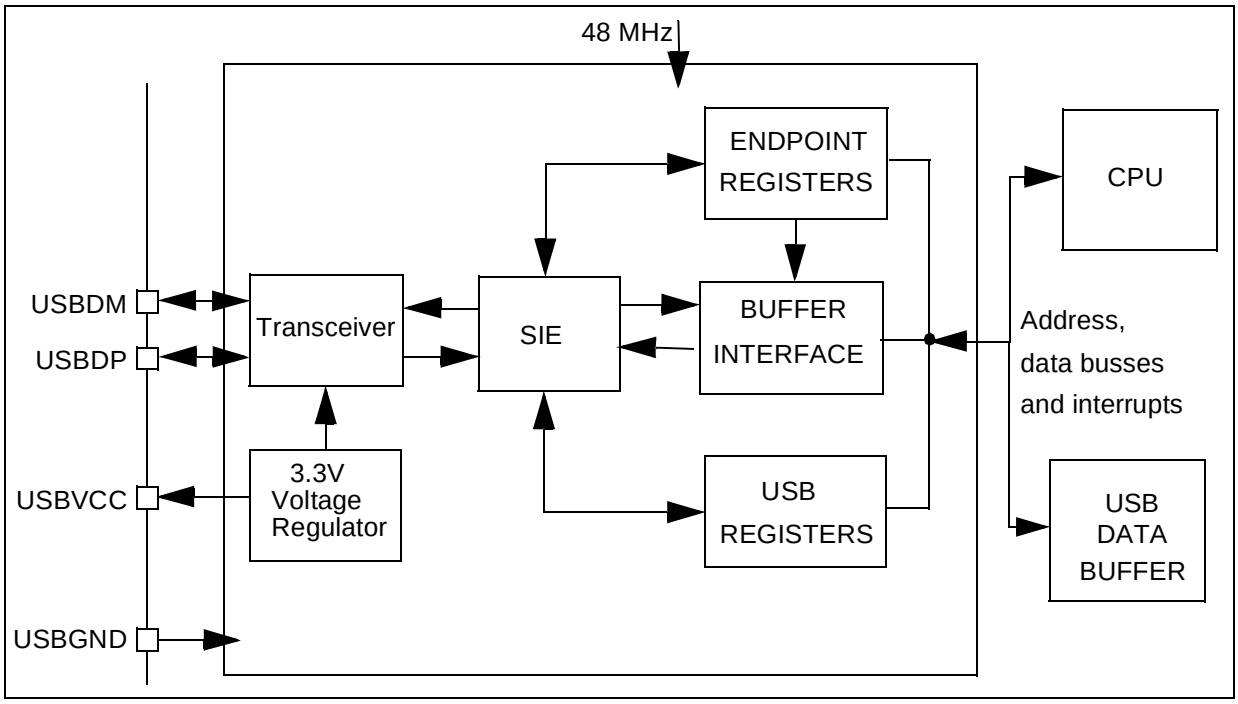
Data Transfer to/from USB Data Buffer Memory

When a token for a valid Endpoint is recognized by the USB interface, the related data transfer takes place to/from the USB data buffer. In normal configuration (MOD[1:0] bits=00 in the CTLR register), at the end of the transaction, an interrupt is generated.

Interrupts

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

Figure 38. USB Block Diagram



USB INTERFACE (Cont'd)

USB Endpoint RAM Buffers

There are five bidirectional Endpoints including one control Endpoint 0. Endpoint 1 and Endpoint 2 are counted as 4 bulk or interrupt Endpoints (two IN and two OUT).

Endpoint 0 and Endpoint 1 are both 2 x 16 bytes in size. Endpoint 2 is 2 x 64 bytes in size and can be configured to physically target different USB Data Buffer areas depending on the MOD[1:0] bits in the CTLR register (see Figure 39, Figure 40 and Figure 41).

The USB Data Buffer operates as a double buffer; while one 512-byte block is being read/written by the DTC, the USB interface reads/writes the other 512-byte block.

The management of the data transfer is performed in upload and download mode (2 x 512 byte buffers for Endpoint 2) by the USB Data Buffer Manager.

Figure 39. Endpoint 2 Normal Mode selected by (MOD[1:0] Bits = 00h)

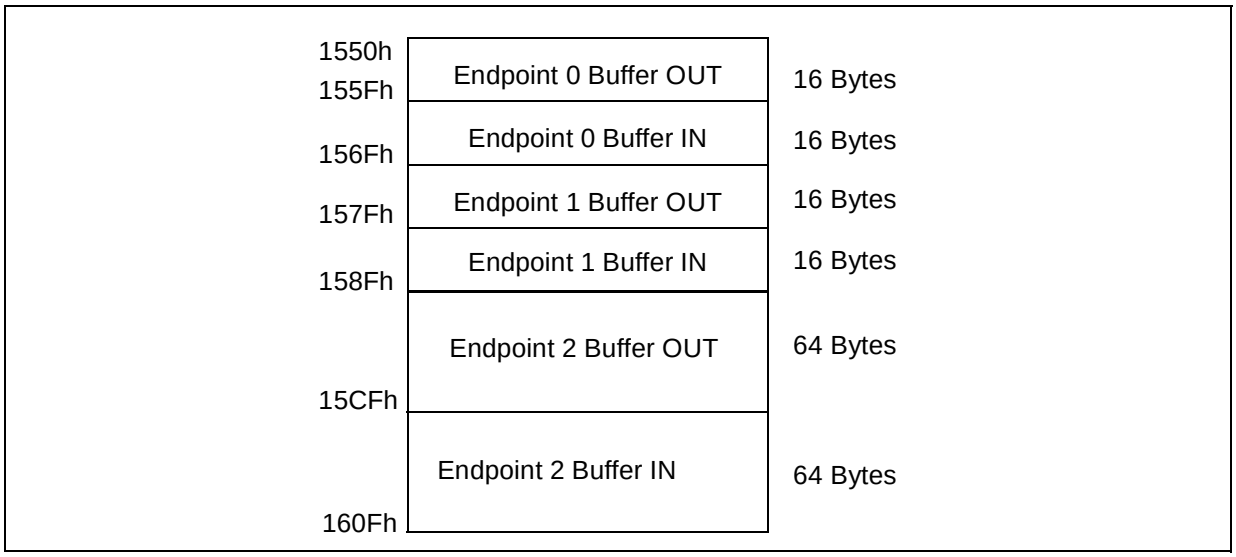
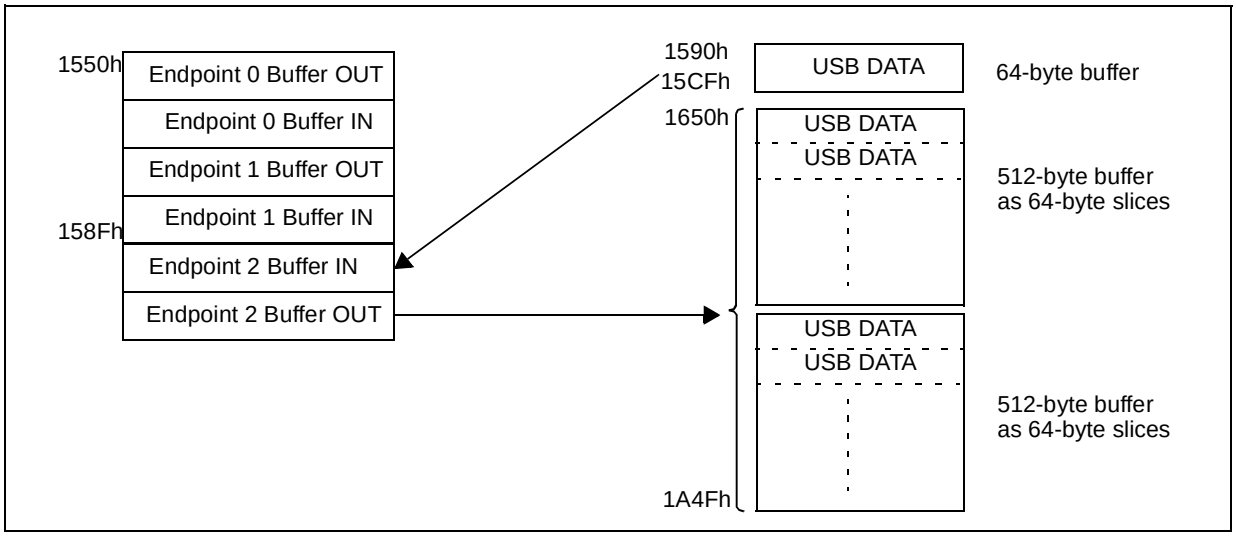
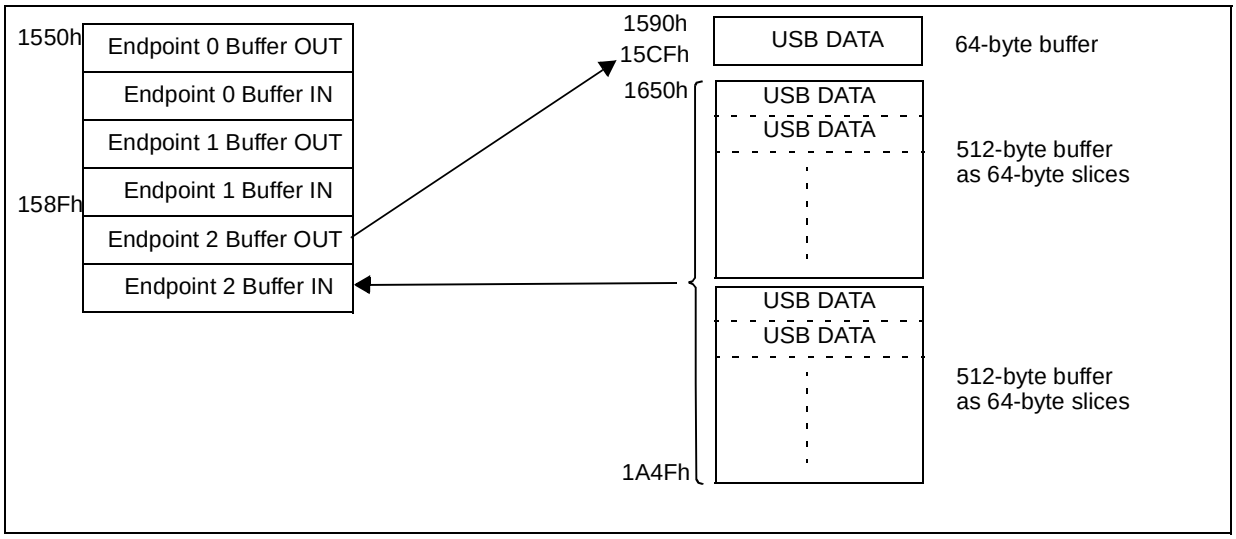


Figure 40. Endpoint 2 Download Mode selected by MOD[1:0] Bits = 10b



USB INTERFACE (Cont'd)

Figure 41. Endpoint 2 Upload Mode selected by MOD[1:0] Bits = 01b



USB INTERFACE (Cont'd)

11.3.4 USB Data Buffer Manager

The USB Data Buffer Manager performs the data transfer between the USB interface and the two 512 Bytes RAM areas used for Endpoint 2 in both Upload and Download modes. It also controls the status of Endpoint 2, by setting the endpoint as NAK when the current buffer is not yet available for either transmission (Upload) or reception (Download).

It is based on a stand-alone hardware state-machine that runs in parallel to the ST7 processing flow. However, at any time, the ST7 software can initialize the USB Data Buffer Manager state-machine in order to synchronize operations by writing a '1' to the CLR bit in the BUFCSR register.

Dedicated buffer status flags are defined to synchronize the USB Data Buffer Manager with the Data Transfer Coprocessor (DTC). These flags are used by the software plug-ins provided by STMicroelectronics) running on the DTC.

11.3.4.1 Data Transfer Modes

In USB normal mode (MOD[1:0]=00b), the maximum memory size of Endpoint 2 is 64 bytes, and therefore reception of 512 bytes packets requires ST7 software intervention every 64 bytes. This means that after a CTR interrupt the hardware puts the Endpoint 2 status bits for the current direction (transmit or receive) in NAK status. The ST7 software must then write the status bits to VALID when it is ready to transmit or receive new data.

On the contrary, in Upload or Download mode, the physical address of Endpoint 2 is automatically incremented every 64 bytes until a 512-byte buffer is full.

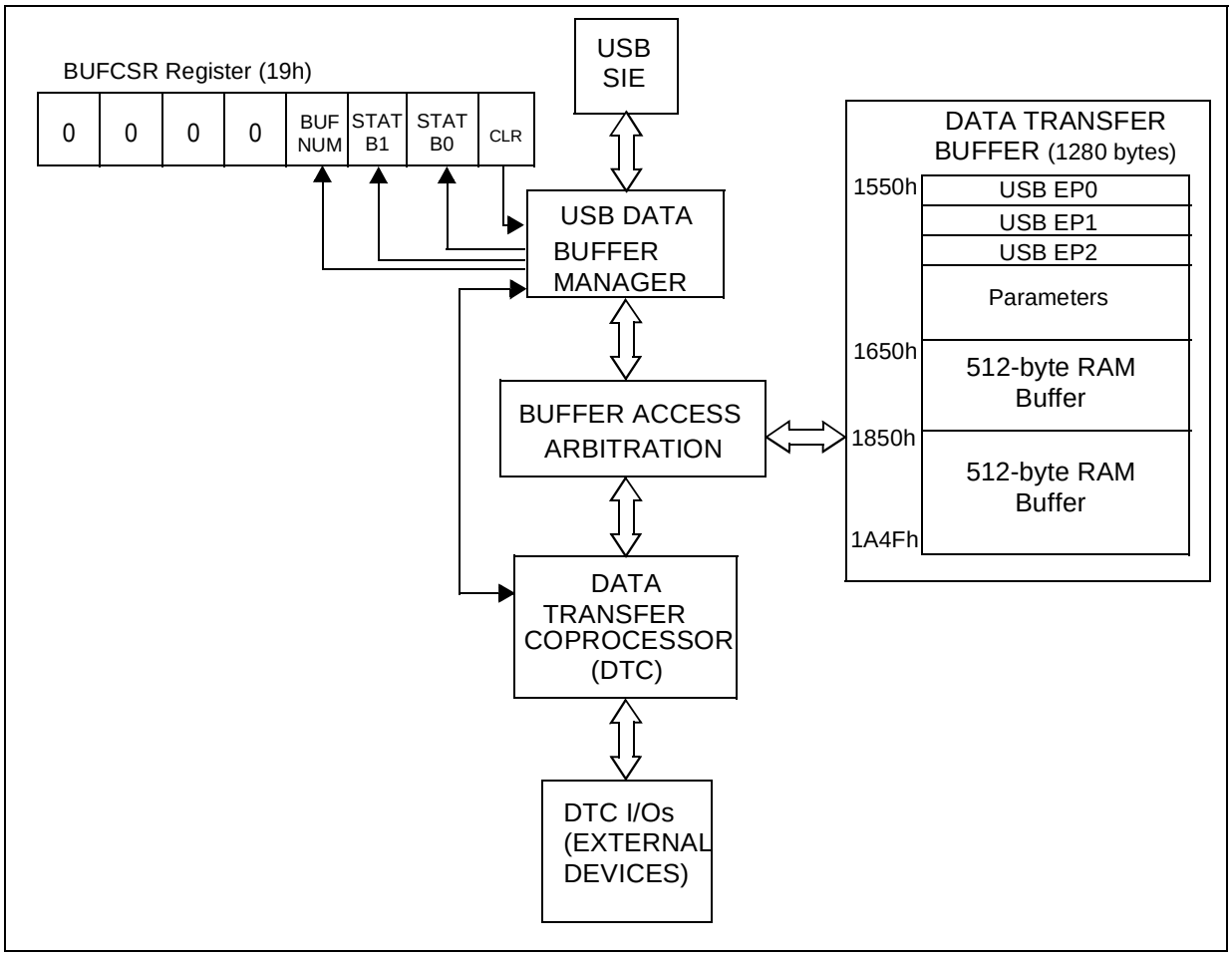
Toggling between the two buffers is automatically managed as soon as 512 bytes have been transmitted to USB (Upload mode) or received from USB (Download), if the next buffer is available: Otherwise, the endpoint is set to invalid until a buffer has been released by the DTC.

11.3.4.2 Switching back to Normal Mode

The USB interface is reset by hardware in Normal mode on reception of a packet with a length below the maximum packet size. In this case, the few bytes are received into one of the two 512-byte buffers and the ST7 must process by software the data received. For this purpose, the information indicating which 512-byte buffer was last addressed is given to the ST7 by the USB Data Buffer Manager (BUFNUM bit in the BUFCSR register), and the number of received bytes is obtained by reading the USB interface registers. With these two items of information, the ST7 can determine what kind of data has been received, and what action has to be taken.

USB INTERFACE (Cont'd)

Figure 42. Overview of USB, DTC and ST7 Interconnections



USB INTERFACE (Cont'd)

11.3.5 Register Description

BUFFER CONTROL/STATUS REGISTER (BUFCSR)

Read Only (except bit 0, read/write)

Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	BUF-NUM	STAT B1	STAT B0	CLR

Bits 7:4 = Reserved, forced by hardware to 0.

Bit 3 = **BUFNUM** *Current USB Buffer Number*
This bit is set and cleared by hardware. When data are received by Endpoint 2 in normal mode (refer to the description of the MOD[1:0] bits in the EP2RXR register) it indicates which buffer contains the data.
0: Current buffer is Buffer 0
1: Current buffer is Buffer 1

Bits 2:1 = **STATB[1:0]** *Buffer Status Bits*
These bits are set and cleared by hardware. When data are transmitted or received by Endpoint 2 in upload or download mode (refer to the description of the MOD[1:0] bits in the EP2RXR register) the STATB[1:0] bits indicate the status as follows:

Meaning		STATBn Value
Upload Mode	Buffer n not full (USB waiting to read Buffer n)	0
	Buffer n full (USB can upload this buffer)	1
Download Mode	Buffer n empty (Can be written to by USB)	0
	Buffer n not empty (USB waiting to write to this buffer)	1

Bit 0 = **CLR** *Clear Buffer Status*
This bit is written by software to clear the BUFNUM and STATB[1:0] bits (it also resets the packet counter of the Buffer Manager state machine). It can be used to re-initialize the upload/download flow (refer to the description of the MOD[1:0] bits in the EP2RXR register).
0: No effect
1: Clear BUFNUM and STATB[1:0] bits

INTERRUPT STATUS REGISTER (ISTR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
CTR	0	SOVR	ERROR	SUSP	ESUSP	RESET	SOF

These bits cannot be set by software. When an interrupt occurs these bits are set by hardware. Software must read them to determine the interrupt type and clear them after servicing.
Note: The CTR bit (which is an OR of all the endpoint CTR flags) cannot be cleared directly, only by clearing the CTR flags in the Endpoint registers.

Bit 7 = **CTR** *Correct Transfer*.
This bit is set by hardware when a correct transfer operation is performed. This bit is an OR of all CTR flags (CTR0 in the EP0R register and CTR_RX and CTR_TX in the EPnR registers). By looking in the USBSR register, the type of transfer can be determined from the PID[1:0] bits for Endpoint 0. For the other Endpoints, the Endpoint number on which the transfer was made is identified by the EP[1:0] bits and the type of transfer by the IN/OUT bit.
0: No Correct Transfer detected
1: Correct Transfer detected

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

Bit 6 = Reserved, forced by hardware to 0.

Bit 5 = **SOVR** Setup Overrun.
This bit is set by hardware when a correct Setup transfer operation is performed while the software is servicing an interrupt which occurred on the same Endpoint (CTR0 bit in the EP0R register is still set when SETUP correct transfer occurs).
0: No SETUP overrun detected
1: SETUP overrun detected

When this event occurs, the USBSR register is not updated because the only source of the SOVR event is the SETUP token reception on the Control Endpoint (EP0).

USB INTERFACE (Cont'd)

Bit 4 = **ERR** *Error.*
This bit is set by hardware whenever one of the errors listed below has occurred:
0: No error detected
1: Timeout, CRC, bit stuffing, nonstandard framing or buffer overrun error detected

Note: Refer to the ERR[2:0] bits in the USBSR register to determine the error type.

Bit 3 = **SUSP** *Suspend mode request.*
This bit is set by hardware when a constant idle state is present on the bus line for more than 3 ms, indicating a suspend mode request from the USB.

The suspend request check is active immediately after each USB reset event and is disabled by hardware when suspend mode is forced (FSUSP bit in the CTLR register) until the end of resume sequence.

Bit 2 = **ESUSP** *End Suspend mode.*
This bit is set by hardware when, during suspend mode, activity is detected that wakes the USB interface up from suspend mode.

This interrupt is serviced by a specific vector, in order to wake up the ST7 from HALT mode.
0: No End Suspend detected
1: End Suspend detected

Bit 1 = **RESET** *USB reset.*
This bit is set by hardware when the USB reset sequence is detected on the bus.
0: No USB reset signal detected
1: USB reset signal detected

Note: The DADDR, EP0R, EP1RXR, EP1TXR and EP2RXR, EP2TXR registers are reset by a USB reset.

Bit 0 = **SOF** *Start of frame.*
This bit is set by hardware when a SOF token is received on the USB.
0: No SOF received
1: SOF received

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

INTERRUPT MASK REGISTER (IMR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
CTRM	0	SOVRM	ERRM	SUSPM	ESUSPM	RESETM	SOFM

These bits are mask bits for all the interrupt condition bits included in the ISTR register. Whenever one of the IMR bits is set, if the corresponding ISTR bit is set, and the I- bit in the CC register is cleared, an interrupt request is generated. For an explanation of each bit, please refer to the description of the ISTR register.

CONTROL REGISTER (CTLR)

Read/Write

Reset value: 0000 0110 (06h)

7							0
RSM	USB_RST	0	0	RESUME	PDWN	FSUSP	FRES

Bit 7 = **RSM** *Resume Detected*
This bit shows when a resume sequence has started on the USB port, requesting the USB interface to wake-up from suspend state. It can be used to determine the cause of an ESUSP event.
0: No resume sequence detected on USB
1: Resume sequence detected on USB

Bit 6 = **USB_RST** *USB Reset detected.*
This bit shows that a reset sequence has started on the USB. It can be used to determine the cause of an ESUSP event (Reset sequence).
0: No reset sequence detected on USB
1: Reset sequence detected on USB

Bits 5:4 Reserved, forced by hardware to 0.

Bit 3 = **RESUME** *Resume.*
This bit is set by software to wake-up the Host when the ST7 is in suspend mode.
0: Resume signal not forced
1: Resume signal forced on the USB bus.

Software should clear this bit after the appropriate delay.

USB INTERFACE (Cont'd)

Bit 2 = **PDWN** *Power down.*
This bit is set by software to turn off the 3.3V on-chip voltage regulator that supplies the external pull-up resistor and the transceiver.
0: Voltage regulator on
1: Voltage regulator off

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

Bit 1 = **FSUSP** *Force suspend mode.*
This bit is set by software to enter Suspend mode. The ST7 should also be put in Halt mode to reduce power consumption.
0: Suspend mode inactive
1: Suspend mode active

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

Bit 0 = **FRES** *Force reset.*
This bit is set by software to force a reset of the USB interface, just as if a RESET sequence came from the USB.
0: Reset not forced
1: USB interface reset forced.

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

DEVICE ADDRESS REGISTER (DADDR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
0	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0

Bit 7 Reserved, forced by hardware to 0.

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

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

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

USB STATUS REGISTER (USBSR)

Read only

Reset Value: 0000 0000 (00h)

7							0
PID1	PID0	IN/ OUT	EP1	EP0	ERR2	ERR1	ERR0

Bits 7:6 = **PID[1:0]** *Token PID bits 1 & 0 for Endpoint 0 Control.*
USB token PIDs are encoded in four bits. PID[1:0] correspond to the most significant bits of the PID field of the last token PID received by Endpoint 0.
Note: The least significant PID bits have a fixed value of 01.
When a CTR interrupt occurs on Endpoint 0 (see register ISTR) the software should read the PID[1:0] bits to retrieve the PID name of the token received.
The USB specification defines PID bits as:

PID1	PID0	PID Name
0	0	OUT
1	0	IN
1	1	SETUP

Bit 5 = **IN/OUT** *Last transaction direction for Endpoint 1 or 2.*

This bit is set by hardware when a CTR interrupt occurs on Endpoint 1 or Endpoint 2.
0: OUT transaction
1: IN transaction

Bits 4:3 = **EP[1:0]** *Endpoint number.*
These bits identify the endpoint which required attention.
00 = Endpoint 0
01 = Endpoint 1
10 = Endpoint 2

USB INTERFACE (Cont'd)

Bits 2:0 = **ERR[2:0]** *Error type*.

These bits identify the type of error which occurred:

ERR2	ERR1	ERR0	Meaning
0	0	0	No error
0	0	1	Bitstuffing error
0	1	0	CRC error
0	1	1	EOP error (unexpected end of packet or SE0 not followed by J-state)
1	0	0	PID error (PID encoding error, unexpected or unknown PID)
1	0	1	Memory over / underrun (memory controller has not answered in time to a memory data request)
1	1	1	Other error (wrong packet, timeout error)

Note: These bits are set by hardware when an error interrupt occurs and are reset automatically when the error bit (ISTR bit 4) is cleared by software.

ENDPOINT 0 REGISTER (EP0R)

Read/Write

Reset value: 0000 0000 (00h)

7							0
CTR0	DTOG_TX	STAT_TX1	STAT_TX0	0	DTOG_RX	STAT_RX1	STAT_RX0

This register is used for controlling Endpoint 0. Bits 6:4 and bits 2:0 are also reset by a USB reset, either received from the USB or forced through the FRES bit in CTLR.

Bit 7 = **CTR0** *Correct Transfer*.
This bit is set by hardware when a correct transfer operation is performed on Endpoint 0. This bit must be cleared after the corresponding interrupt has been serviced.
0: No CTR on Endpoint 0
1: Correct transfer on Endpoint 0

Bit 6 = **DTOG_TX** *Data Toggle, for transmission transfers*.
It contains the required value of the toggle bit (0=DATA0, 1=DATA1) for the next transmitted data packet. This bit is set by hardware on reception of a SETUP PID. DTOG_TX toggles only when the transmitter has received the ACK signal from the USB host. DTOG_TX and also DTOG_RX are normally updated by hardware, on receipt of a relevant PID. They can be also written by the user, both for testing purposes and to force a specific (DATA0 or DATA1) token.

Bits 5:4 = **STAT_TX [1:0]** *Status bits, for transmission transfers*.
These bits contain the information about the endpoint status, as listed below:

Table 19. Transmission Status Encoding

STAT_TX1	STAT_TX0	Meaning
0	0	**DISABLED:** no function can be executed on this endpoint and messages related to this endpoint are ignored.
0	1	**STALL**: the endpoint is stalled and all transmission requests result in a STALL handshake.
1	0	**NAK**: the endpoint is NAKed and all transmission requests result in a NAK handshake.
1	1	**VALID**: this endpoint is enabled (if an address match occurs, the USB interface handles the transaction).

These bits are written by software. Hardware sets the STAT_TX and STAT_RX bits to NAK when a correct transfer has occurred (CTR=1) addressed to this endpoint; this allows software to prepare the next set of data to be transmitted.

Bit 3 = Reserved, forced by hardware to 0.

Bit 2 = **DTOG_RX** *Data Toggle, for reception transfers*.
It contains the expected value of the toggle bit (0=DATA0, 1=DATA1) for the next data packet. This bit is cleared by hardware in the first stage (Setup Stage) of a control transfer (SETUP transactions start always with DATA0 PID). The receiver toggles DTOG_RX only if it receives a correct data packet and the packet's data PID matches the receiver sequence bit.

USB INTERFACE (Cont'd)

Bits 1:0 = **STAT_RX [1:0]** *Status bits, for reception transfers*.
These bits contain the information about the endpoint status, as listed below:

Table 20. Reception Status Encoding

STAT_RX1	STAT_RX0	Meaning
0	0	**DISABLED:** no function can be executed on this endpoint and messages related to this endpoint are ignored.
0	1	**STALL:** the endpoint is stalled and all reception requests result in a STALL handshake.
1	0	**NAK**: the endpoint is NAKed and all reception requests result in a NAK handshake.
1	1	**VALID**: this endpoint is enabled (if an address match occurs, the USB interface handles the transaction).

These bits are written by software. Hardware sets the STAT_RX and STAT_TX bits to NAK when a correct transfer has occurred (CTR=1) addressed to this endpoint, so the software has the time to examine the received data before acknowledging a new transaction.

Notes:

If a SETUP is received while the status is other than DISABLED, it is acknowledged and the two directional status bits are set to NAK by hardware.

When a STALL is answered by the USB device, the two directional status bits are set to STALL by hardware.

ENDPOINT 1 RECEPTION REGISTER (EP1RXR)

Read/Write

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	CTR_RX	DTOG_RX	STAT_RX1	STAT_RX0

This register is used for controlling Endpoint 1 reception. Bits 2:0 are also reset by a USB reset, either received from the USB or forced through the FRES bit in the CTLR register.

Bits 7:4 Reserved, forced by hardware to 0.

Bit 3 = **CTR_RX** *Correct Reception Transfer*.
This bit is set by hardware when a correct transfer operation is performed in reception. This bit must be cleared after the corresponding interrupt has been serviced.

Bit 2 = **DTOG_RX** *Data Toggle, for reception transfers*.
It contains the expected value of the toggle bit (0=DATA0, 1=DATA1) for the next data packet. The receiver toggles DTOG_RX only if it receives a correct data packet and the packet's data PID matches the receiver sequence bit.

Bits 1:0 = **STAT_RX [1:0]** *Status bits, for reception transfers*.
These bits contain the information about the endpoint status, as listed below:

Table 21. Reception Status Encoding:

STAT_RX1	STAT_RX0	Meaning
0	0	**DISABLED:** reception transfers cannot be executed.
0	1	**STALL:** the endpoint is stalled and all reception requests result in a STALL handshake.
1	0	**NAK**: the endpoint is naked and all reception requests result in a NAK handshake.
1	1	**VALID**: this endpoint is enabled for reception.

These bits are written by software, but hardware sets the STAT_RX bits to NAK when a correct transfer has occurred (CTR=1) addressed to this endpoint, so the software has the time to examine the received data before acknowledging a new transaction.

USB INTERFACE (Cont'd)

ENDPOINT 1 TRANSMISSION REGISTER (EP1TXR)

Read/Write

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	CTR_TX	DTOG_TX	STAT_TX1	STAT_TX0

This register is used for controlling Endpoint 1 transmission. Bits 2:0 are also reset by a USB reset, either received from the USB or forced through the FRES bit in the CTLR register.

Bit 3 = **CTR_TX** *Correct Transmission Transfer*.
This bit is set by hardware when a correct transfer operation is performed in transmission. This bit must be cleared after the corresponding interrupt has been serviced.
0: No CTR in transmission on Endpoint 1
1: Correct transfer in transmission on Endpoint 1

Bit 2 = **DTOG_TX** *Data Toggle, for transmission transfers*.
This bit contains the required value of the toggle bit (0=DATA0, 1=DATA1) for the next data packet. DTOG_TX toggles only when the transmitter has received the ACK signal from the USB host. DTOG_TX and DTOG_RX are normally updated by hardware, at the receipt of a relevant PID. They can be also written by the user, both for testing purposes and to force a specific (DATA0 or DATA1) token.

Bits 1:0 = **STAT_TX [1:0]** *Status bits, for transmission transfers*.
These bits contain the information about the endpoint status, which is listed below

Table 22. Transmission Status Encoding

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

These bits are written by software, but hardware sets the STAT_TX bits to NAK when a correct transfer has occurred (CTR=1) addressed to this endpoint. This allows software to prepare the next set of data to be transmitted.

ENDPOINT 2 RECEPTION REGISTER (EP2RXR)

Read/Write

Reset value: 0000 0000 (00h)

7							0
MOD1	MOD0	0	0	CTR_RX	DTOG_RX	STAT_RX1	STAT_RX0

This register is used for controlling endpoint 2 reception. Bits 2:0 are also reset by a USB reset, either received from the USB or forced through the FRES bit in the CTLR register.

Bits 7:6 = **MOD[1:0]** *Endpoint 2 mode*.
These bits are set and cleared by software. They select the Endpoint 2 mode (See Figure 40 and Figure 41).

MOD1	MOD0	Mode
0	0	Normal mode: Endpoint 2 is managed by user software
0	1	Upload mode to USB data buffer: Bulk mode IN under hardware control from DTC[1]
1	0	Download mode from USB data buffer: Bulk mode OUT under hardware control to DTC[2].

Notes:

1. Before selecting Download mode, software must write the maximum packet size value (for instance 64) in the CNT2RXR register and write the STAT_RX bits in the EP2RXR register to VALID.

2. Before selecting Upload mode, software must write the maximum packet size value (for instance 64) in the CNT2TXR register and write the STAT_TX bits in the EP2TXR register to NAK.

USB INTERFACE (Cont'd)

Download Mode

IN transactions are managed the same way as in normal mode (by software with the help of CTR interrupt) but OUT transactions are managed by hardware. This means that no CTR interrupt is generated at the end of an OUT transaction and the STAT_RX bits are set to valid by hardware when the buffer is ready to receive new data. This allows the 512-byte buffer to be written without software intervention.

If the USB interface receives a packet which has a length lower than the maximum packet size (written in the CNT2RXR register, see Note below), the USB interface switches back to normal mode and generates a CTR interrupt and the STAT_RX bits of the EP2R register are set to NAK by hardware as in normal mode.

Upload Mode

OUT transactions are managed in the same way as normal mode and IN transactions are managed by hardware in the same way as OUT transactions in download mode.

Bits 5:4 Reserved, forced by hardware to 0.

Bit 3 = **CTR_RX** *Reception Correct Transfer*.
This bit is set by hardware when a correct transfer operation is performed in reception. This bit must be cleared after that the corresponding interrupt has been serviced.

Bit 2 = **DTOG_RX** *Data Toggle, for reception transfers*.
It contains the expected value of the toggle bit (0=DATA0, 1=DATA1) for the next data packet. USB INTERFACE (Cont'd)

The receiver toggles DTOG_RX only if it receives a correct data packet and the packet's data PID matches the receiver sequence bit.

Bits 1:0 = **STAT_RX [1:0]** *Status bits, for reception transfers*.
These bits contain the information about the endpoint status, which is listed below:

Table 23. Reception Status Encoding

STAT_RX1	STAT_RX0	Meaning
0	0	**DISABLED:** reception transfers cannot be executed.
0	1	**STALL:** the endpoint is stalled and all reception requests result in a STALL handshake.
1	0	**NAK**: the endpoint is naked and all reception requests result in a NAK handshake.
1	1	**VALID**: this endpoint is enabled for reception.

These bits are written by software, but hardware sets the STAT_RX bits to NAK when a correct transfer has occurred (CTR=1) addressed to this endpoint, so the software has the time to examine the received data before acknowledging a new transaction.

Note: These bits are write protected in download mode (if MOD[1:0] =10b in the EP2RXR register)

ENDPOINT 2 TRANSMISSION REGISTER (EP2TXR)

Read/Write

Reset value: 0000 0000 (00h)

7							0
0	0	0	0	CTR_TX	DTOG_TX	STAT_TX1	STAT_TX0

This register is used for controlling Endpoint 2 transmission. Bits 2:0 are also reset by a USB reset, either received from the USB or forced through the FRES bit in the CTLR register.

Bit 3 = **CTR_TX** *Transmission Transfer Correct*.
This bit is set by hardware when a correct transfer operation is performed in transmission. This bit must be cleared after the corresponding interrupt has been serviced.
0: No CTR in transmission on Endpoint 2
1: Correct transfer in transmission on Endpoint 2

USB INTERFACE (Cont'd)

Bit 2= **DTOG_TX** *Data Toggle, for transmission transfers*.
This bit contains the required value of the toggle bit (0=DATA0, 1=DATA1) for the next data packet. DTOG_TX and DTOG_RX are normally updated by hardware, on receipt of a relevant PID. They can be also written by the user, both for testing purposes and to force a specific (DATA0 or DATA1) token.

Bits 1:0 = **STAT_TX [1:0]** *Status bits, for transmission transfers*.
These bits contain the information about the endpoint status, which is listed below

Table 24. Transmission Status Encoding

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

These bits are written by software, but hardware sets the STAT_TX bits to NAK when a correct transfer (CTR=1) addressed to this endpoint has occurred. This allows software to prepare the next set of data to be transmitted.

Note: These bits are write protected in upload mode (MOD[1:0] =01b in the EP2RXR register)

RECEPTION COUNTER REGISTER (CNT0RXR, CNT1RXR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
0	0	0	CNT4	CNT3	CNT2	CNT1	CNT0

This register contains the allocated buffer size for endpoint 0 or 1 reception, setting the maximum number of bytes the related endpoint can receive with the next OUT (or SETUP for Endpoint 0) transaction. At the end of a reception, the value of this register is the max size decremented by the number of bytes received (to determine the number of bytes received, the software must subtract the content of this register from the allocated buffer size).

RECEPTION COUNTER REGISTER (CNT2RXR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
0	CNT6	CNT5	CNT4	CNT3	CNT2	CNT	CNT0

This register contains the allocated buffer size for endpoint 2 reception, setting the maximum number of bytes the related endpoint can receive with the next OUT transaction. At the end of a reception, the value of this register is the maximum size decremented by the number of bytes received (to determine the number of bytes received, the software must subtract the content of this register from the allocated buffer size).

TRANSMISSION COUNTER REGISTER (CNT0TXR, CNT1TXR)

Read/Write

Reset Value 0000 0000 (00h)

7							0
0	0	0	CNT4	CNT3	CNT2	CNT1	CNT0

This register contains the number of bytes to be transmitted by Endpoint 0 or 1 at the next IN token addressed to it.

TRANSMISSION COUNTER REGISTER (CNT2TXR)

Read/Write

Reset Value 0000 0000 (00h)

7							0
0	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0

This register contains the number of bytes to be transmitted by Endpoint 2 at the next IN token addressed to it.

Table 25. USB Register Map and Reset values

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
47	BUFCSR Reset Value	0 0	0 0	0 0	0 0	BUFNUM 0	BUF1ST 0	BUF0ST 0	RESETST 0
30	USBISTR Reset Value	CTR 0	0 0	SOVR 0	ERR 0	SUSP 0	ESUSP 0	RESET 0	SOF 0
31	USBIMR Reset Value	CTRM 0	0 0	SOVRM 0	ERRM 0	SUSPM 0	ESUSPM 0	RESETM 0	SOFM 0
32	USBCTLR Reset Value	RSM 0	USB_RST 0	0	0	RESUME 0	PDWN 1	FSUSP 1	FRES 0
33	DADDR Reset Value	0	ADD6 0	ADD5 0	ADD4 0	ADD3 0	ADD2 0	ADD1 0	ADD0 0
34	USBSR Reset Value	PID1 0	PID0 0	IN /OUT 0	EP1 0	EP0 0	ERR2 0	ERR1 0	ERR0 0
35	EP0R Reset Value	CTR0 0	DTOG_TX 0	STAT_TX1 0	STAT_TX0 0	0 0	DTOG_RX 0	STAT_RX1 0	STAT_RX0 0
36	CNT0RXR Reset Value	0 0	0 0	0 0	CNT4 0	CNT3 0	CNT2 0	CNT1 0	CNT0 0
37	CNT0TXR Reset Value	0 0	0 0	0 0	CNT4 0	CNT3 0	CNT2 0	CNT1 0	CNT0 0
38	EP1RXR Reset Value	0	0	0	0	CTR_RX 0	DTOG_RX 0	STAT_RX1 0	STAT_RX0 0
39	CNT1RXR Reset Value	0 0	0 0	0 0	CNT4 0	CNT3 0	CNT2 0	CNT1 0	CNT0 0
3A	EP1TXR Reset Value	0	0	0	0	CTR_TX 0	DTOG_TX 0	STAT_TX1 0	STAT_TX0 0
3B	CNT1TXR Reset Value	0 0	0 0	0 0	CNT4 0	CNT3 0	CNT2 0	CNT1 0	CNT0 0
3C	EP2RXR Reset Value	MOD1 0	MOD0 0	0	0	CTR_RX 0	DTOG_RX 0	STAT_RX1 0	STAT_RX0 0
3D	CNT2RXR Reset Value	0 0	CNT6 0	CNT5 0	CNT4 0	CNT3 0	CNT2 0	CNT1 0	CNT0 0
3E	EP2TXR Reset Value	0	0	0	0	CTR_TX 0	DTOG_TX 0	STAT_TX1 0	STAT_TX0 0
3F	CNT2TXR Reset Value	0 0	CNT6 0	CNT5 0	CNT4 0	CNT3 0	CNT2 0	CNT1 0	CNT0 0

11.4 16-BIT TIMER

11.4.1 Introduction

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

11.4.2 Main Features

- Programmable prescaler: f_{CPU} divided by 2, 4 or 8.
- Overflow status flag and maskable interrupt
- Output compare functions with
  - 2 dedicated 16-bit registers
  - 2 dedicated programmable signals
  - 2 dedicated status flags
  - 1 dedicated maskable interrupt
- 2 alternate functions on I/O ports (OCMP1, OCMP2)

The Block Diagram is shown in Figure 43.

11.4.3 Functional Description

11.4.3.1 Counter

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

Counter Register (CR):

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

Alternate Counter Register (ACR)

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

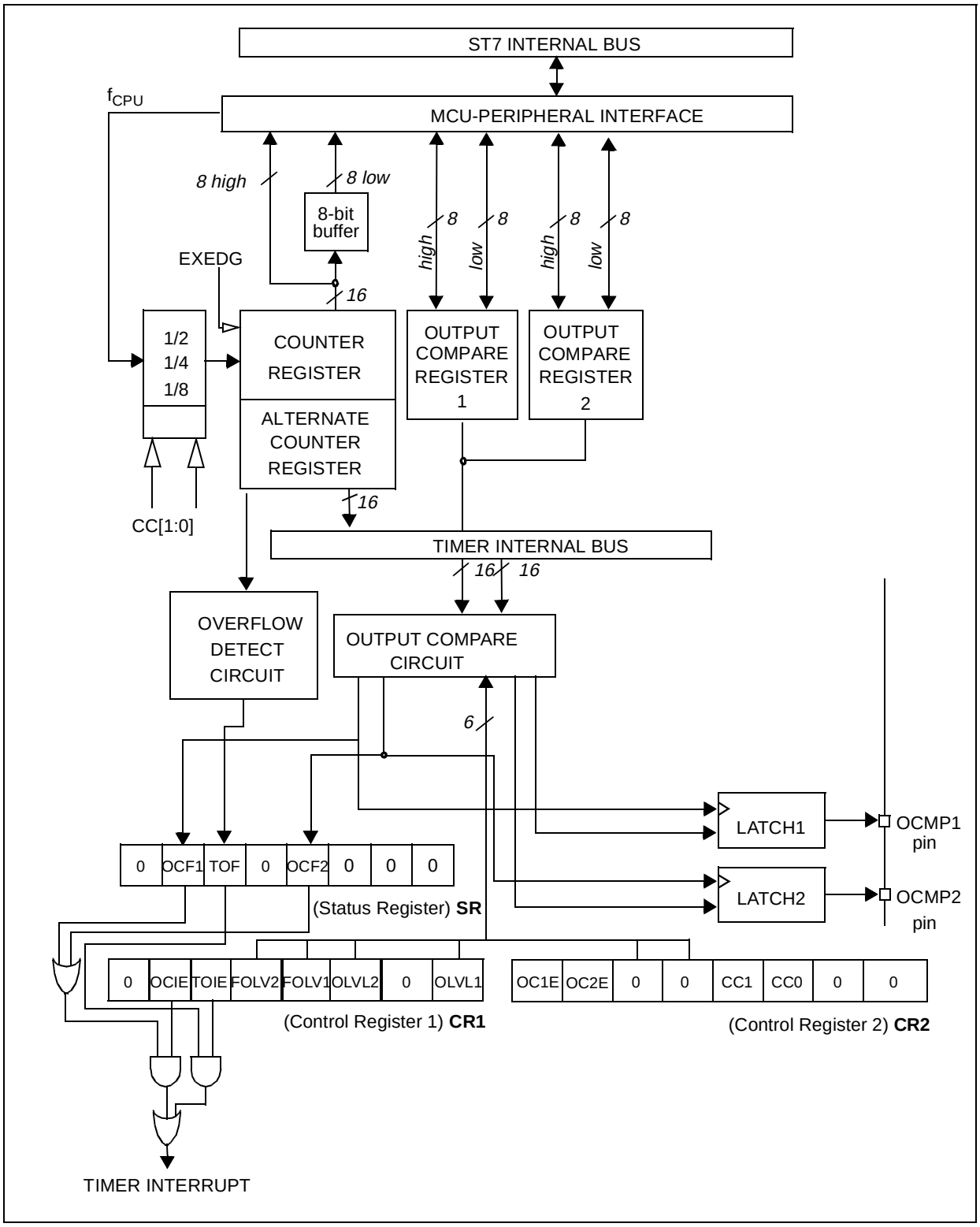
These two read-only 16-bit registers contain the same value but with the difference that reading the ACLR register does not clear the TOF bit (Timer overflow flag), located in the Status register, (SR), (see note at the end of paragraph titled 16-bit read sequence).

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

The timer clock depends on the clock control bits of the CR2 register, as illustrated in Table 26 Clock Control Bits. The value in the counter register repeats every 131.072, 262.144 or 524.288 CPU clock cycles depending on the CC[1:0] bits.
The timer frequency can be $f_{CPU}/2$, $f_{CPU}/4$, $f_{CPU}/8$ or an external frequency.

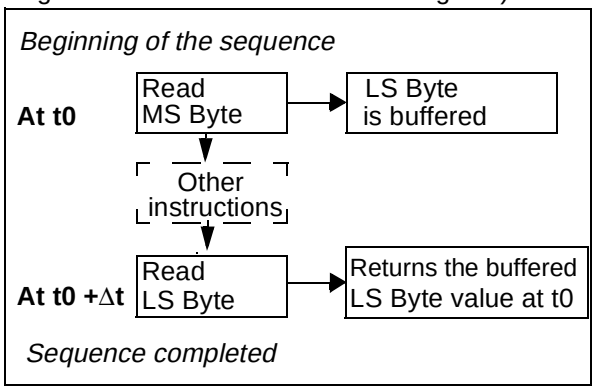
16-BIT TIMER (Cont'd)

Figure 43. Timer Block Diagram



16-BIT TIMER (Cont'd)

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



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

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

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

Whatever the timer mode used 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).

16-BIT TIMER (Cont'd)

Figure 44. Counter Timing Diagram, internal clock divided by 2

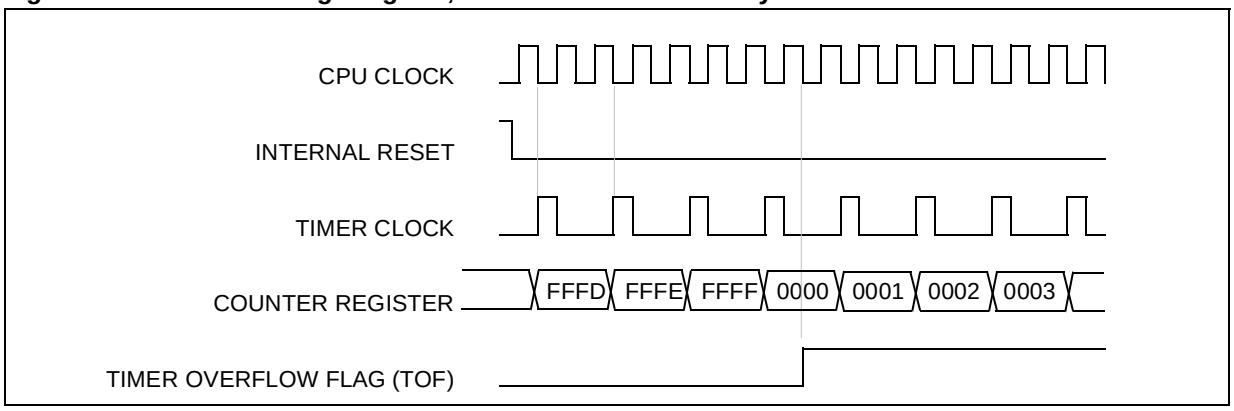


Figure 45. Counter Timing Diagram, internal clock divided by 4

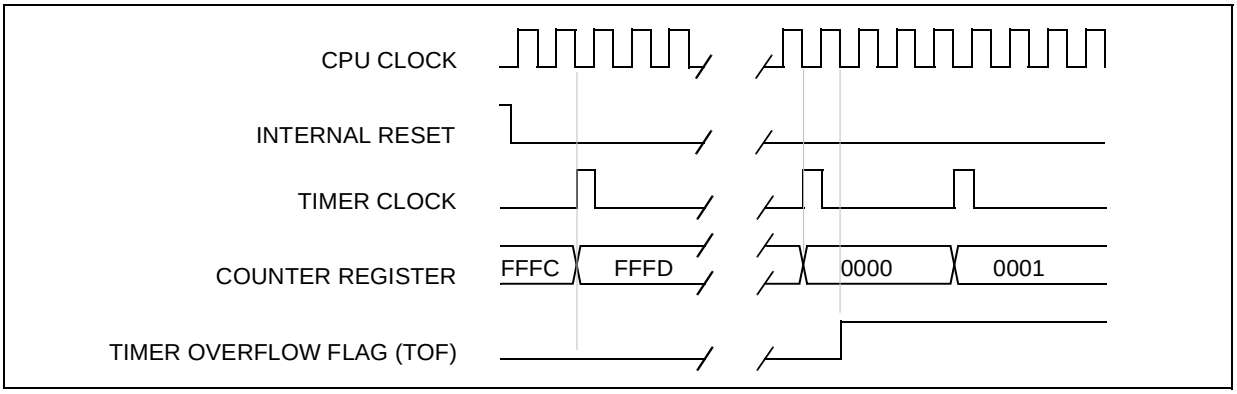
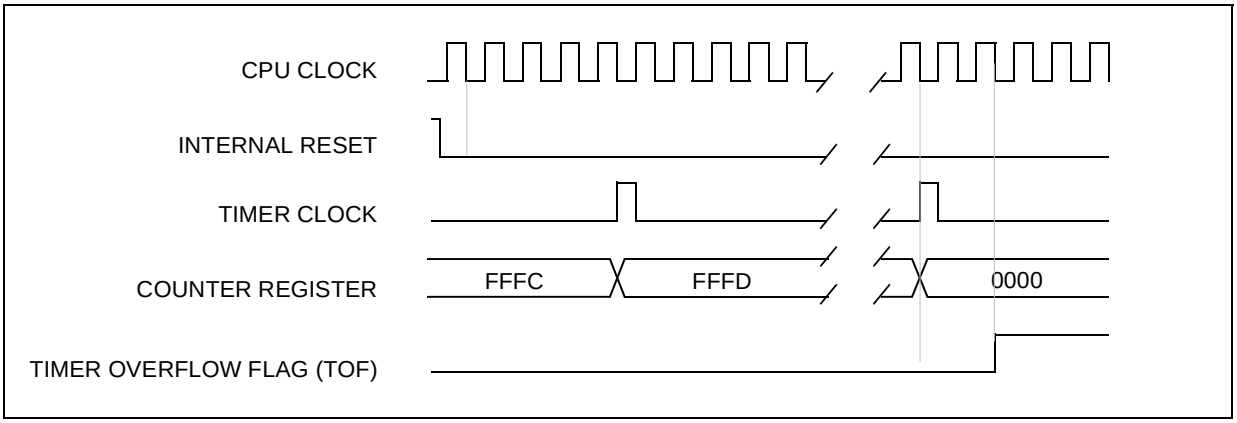


Figure 46. Counter Timing Diagram, internal clock divided by 8



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)

11.4.3.2 Output Compare

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

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

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

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

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

	MS Byte	LS Byte
OC*i*R	OC*i*HR	OC*i*LR

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

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

Procedure:

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

- Set the OC*i*E bit if an output is needed then the OCMP*i* pin is dedicated to the output compare *i* signal.
- Select the timer clock (CC[1:0]) (see Table 26 Clock Control Bits).

And select the following in the CR1 register:

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

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

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

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

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

Where:

Δt = Output compare period (in seconds)

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

PRESC = Timer prescaler factor (2, 4 or 8 depending on CC[1:0] bits, see Table 26 Clock Control Bits)

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

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

Where:

Δt = Output compare period (in seconds)

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

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

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

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

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

16-BIT TIMER (Cont'd)

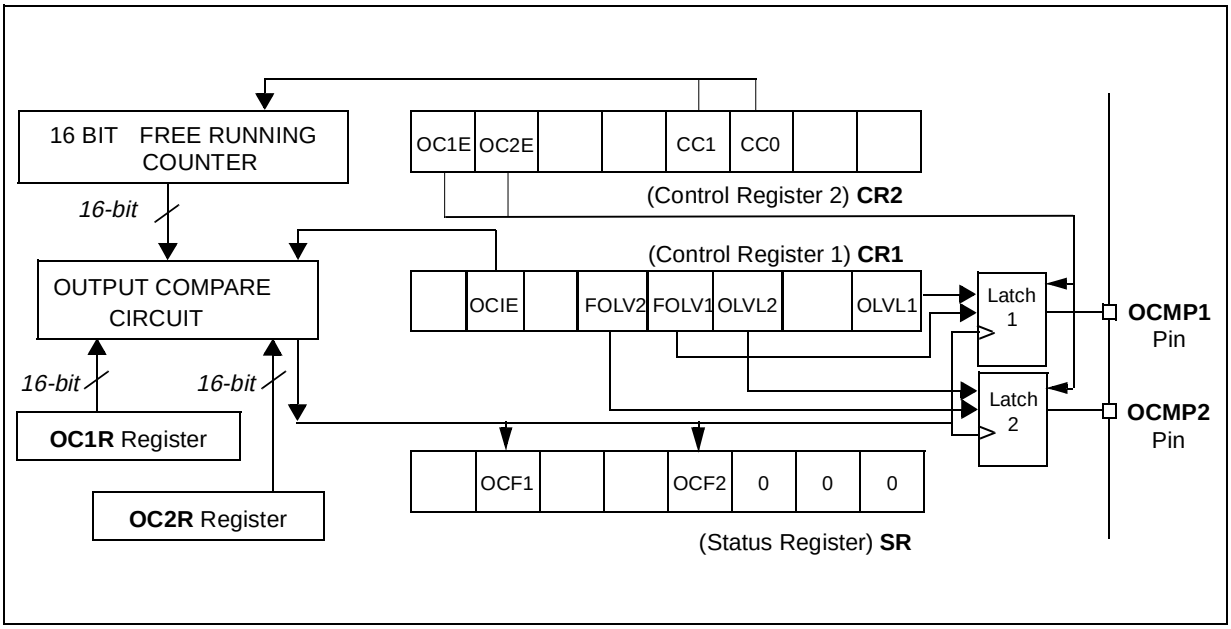
Notes:

1. After a processor write cycle to the OC*i*HR register, the output compare function is inhibited until the OC*i*LR register is also written.
2. If the OC*i*E bit is not set, the OCMP*i* pin is a general I/O port and the OLVL*i* bit will not appear when a match is found but an interrupt could be generated if the OCIE bit is set.
3. When the timer clock is $f_{CPU}/2$, OCF*i* and OCMP*i* are set while the counter value equals the OC*i*R register value (see Figure 48 on page 82).
   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 on page 82).
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 timeout period.

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.

Figure 47. Output Compare Block Diagram



16-BIT TIMER (Cont'd)

Figure 48. Output Compare Timing Diagram, $f_{TIMER} = f_{CPU}/2$

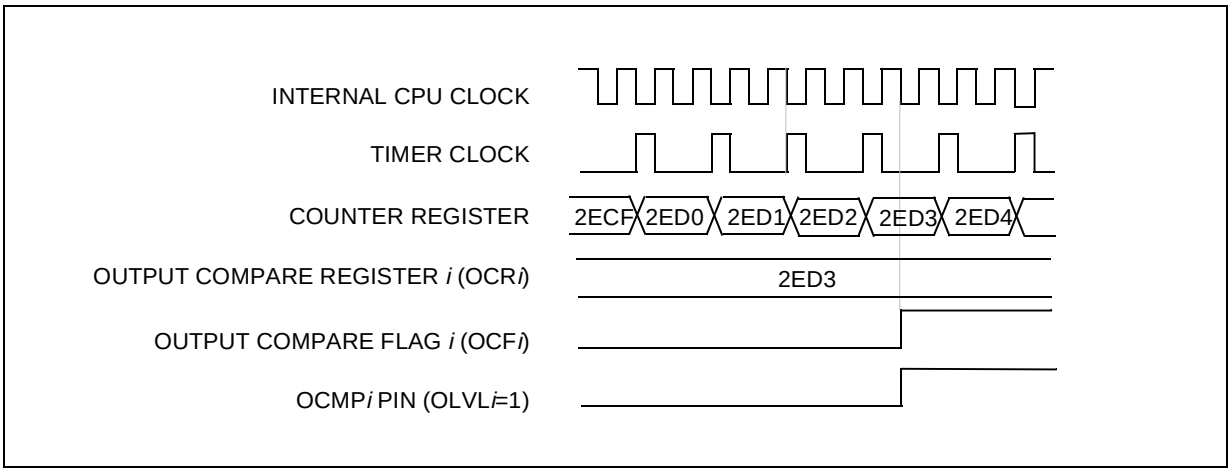
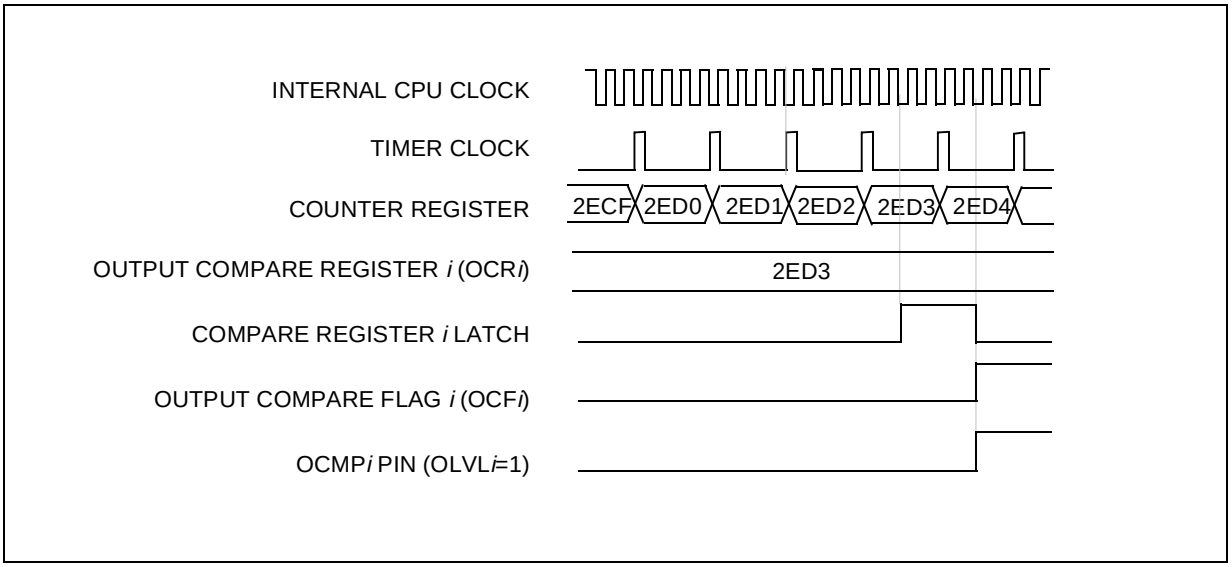


Figure 49. Output Compare Timing Diagram, $f_{TIMER} = f_{CPU}/4$



16-BIT TIMER (Cont'd)

11.4.4 Low Power Modes

Mode	Description
WAIT	No effect on 16-bit Timer. Timer interrupts cause the device to exit from WAIT mode.
HALT	16-bit Timer registers are frozen. In HALT mode, the counter stops counting until Halt mode is exited. Counting resumes from the previous count when the MCU is woken up by an interrupt with "exit from HALT mode" capability or from the counter reset value when the MCU is woken up by a RESET.

11.4.5 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
Output Compare 1 event	OCF1	OCIE	Yes	No
Output Compare 2 event	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).

16-BIT TIMER (Cont'd)

11.4.6 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
0	OCIE	TOIE	FOLV2	FOLV1	OLVL2	0	OLVL1

Bit 7 = Reserved, forced by hardware to 0.

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 = Reserved, forced by hardware to 0.

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	0	0	CC1	CC0	0	0

Bit 7 = **OC1E** *Output Compare 1 Pin Enable.*
This bit is used only to output the signal from the timer on the OCMP1 pin (OLV1 in Output Compare mode, both OLV1 and OLV2 in PWM and one-pulse mode). Whatever the value of the OC1E bit, the internal Output Compare 1 function of the timer remains active.
0: OCMP1 pin alternate function disabled (I/O pin free for general-purpose I/O).
1: OCMP1 pin alternate function enabled.

Bit 6 = **OC2E** *Output Compare 2 Pin Enable.*
This bit is used only to output the signal from the timer on the OCMP2 pin (OLV2 in Output Compare mode). Whatever the value of the OC2E bit, the internal Output Compare 2 function of the timer remains active.
0: OCMP2 pin alternate function disabled (I/O pin free for general-purpose I/O).
1: OCMP2 pin alternate function enabled.

Bits 5:4 = Reserved, forced by hardware to 0.

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

The timer clock mode depends on these bits:

Table 26. Clock Control Bits

Timer Clock	CC1	CC0
f_{CPU} / 4	0	0
f_{CPU} / 2	0	1
f_{CPU} / 8	1	0
Reserved	1	1

Bits 1:0 = Reserved, forced by hardware to 0.

STATUS REGISTER (SR)

Read Only

Reset Value: 0000 0000 (00h)

The three least significant bits are not used.

7							0
0	OCF1	TOF	0	OCF2	0	0	0

Bit 7 = Reserved, forced by hardware to 0.

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 = Reserved, forced by hardware to 0.

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.

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

16-BIT TIMER (Cont'd)

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

OUTPUT COMPARE 2 HIGH REGISTER (OC2HR)

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

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

7							0
MSB							LSB

OUTPUT COMPARE 2 LOW REGISTER (OC2LR)

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

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

7							0
MSB							LSB

COUNTER HIGH REGISTER (CHR)

Read Only
Reset Value: 1111 1111 (FFh)

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

7							0
MSB							LSB

COUNTER LOW REGISTER (CLR)

Read Only
Reset Value: 1111 1100 (FCh)

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

7							0
MSB							LSB

ALTERNATE COUNTER HIGH REGISTER (ACHR)

Read Only
Reset Value: 1111 1111 (FFh)

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

7							0
MSB							LSB

ALTERNATE COUNTER LOW REGISTER (ACLR)

Read Only
Reset Value: 1111 1100 (FCh)

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

7							0
MSB							LSB

16-BIT TIMER (Cont'd)

Table 27. Timer Register Map

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
20	TCR1	0	OCIE	TOIE	FOLV2	FOLV1	OLVL2	0	OLVL1
21	TCR2	OC1E	OC2E	0	0	CC1	CC0	0	0
22	TSR	0	OCF1	TOF	0	OCF2	0	0	0
23	CHR	MSB							LSB
24	CLR	MSB							LSB
25	ACHR	MSB							LSB
26	ACLR	MSB							LSB
27	OC1HR	MSB							LSB
28	OC1LR	MSB							LSB
29	OC2HR	MSB							LSB
2A	OC2LR	MSB							LSB

11.5 PWM/BRM GENERATOR (DAC)

11.5.1 Introduction

This PWM/BRM peripheral includes a 6-bit Pulse Width Modulator (PWM) and a 4-bit Binary Rate Multiplier (BRM) Generator. It allows the digital to analog conversion (DAC) when used with external filtering.

Note: The number of PWM and BRM channels available depends on the device. Refer to the device pin description and register map.

11.5.2 Main Features

- Fixed frequency: $f_{CPU}/64$
- Resolution: T_{CPU}
- Steps of $V_{DD}/2^{10}$ (5mV if V_{DD}=5V)

11.5.3 Functional Description

The 10 bits of the 10-bit PWM/BRM are distributed as 6 PWM bits and 4 BRM bits. The generator consists of a 10-bit counter (common for all channels), a comparator and the PWM/BRM generation logic.

PWM Generation

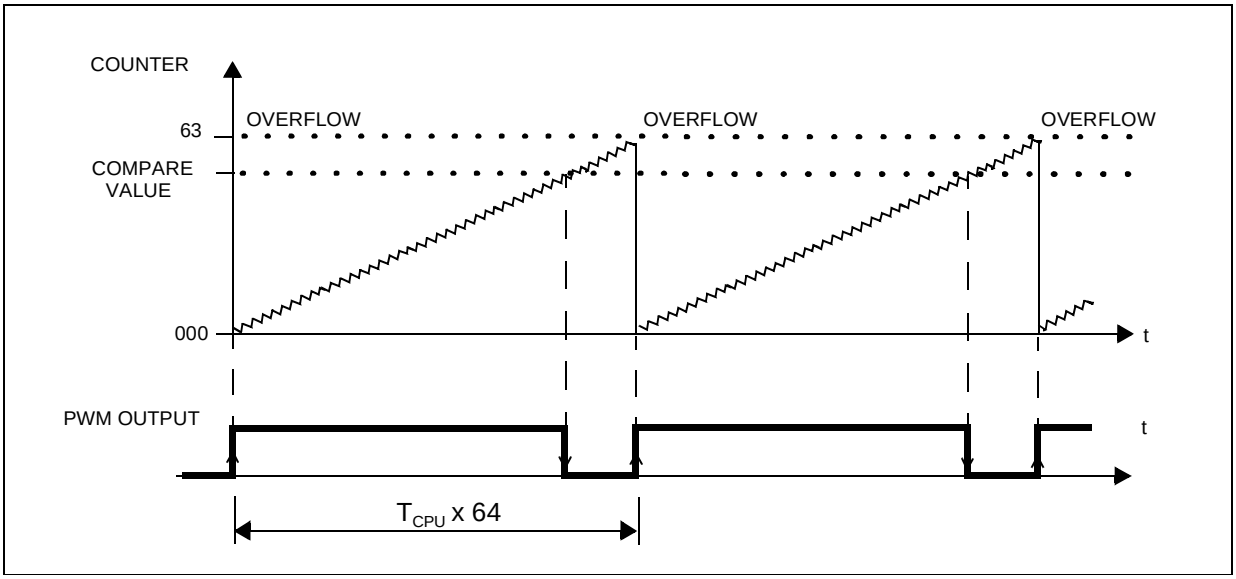
The counter increments continuously, clocked at internal CPU clock. Whenever the 6 least significant bits of the counter (defined as the PWM counter) overflow, the output level for all active channels is set.

The state of the PWM counter is continuously compared to the PWM binary weight for each channel, as defined in the relevant PWM register, and when a match occurs the output level for that channel is reset.

This Pulse Width modulated signal must be filtered, using an external RC network placed as close as possible to the associated pin. This provides an analog voltage proportional to the average charge passed to the external capacitor. Thus for a higher mark/space ratio (high time much greater than low time) the average output voltage is higher. The external components of the RC network should be selected for the filtering level required for control of the system variable.

Each output may individually have its polarity inverted by software, and can also be used as a logical output.

Figure 50. PWM Generation



PWM/BRM GENERATOR (Cont'd)

PWM/BRM Outputs

The PWM/BRM outputs are assigned to dedicated pins.
The PWM/BRM outputs can be connected to an RC filter (see Figure 51 for an example).
The RC filter time must be higher than T_{CPU}x64.

Figure 51. Typical PWM Output Filter

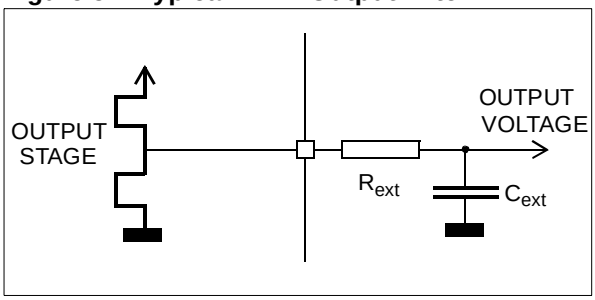


Table 28. 6-Bit PWM Ripple After Filtering

Cext (µF)	V RIPPLE (mV)
0.128	78
1.28	7.8
12.8	0.78

With RC filter (R=1KΩ),

f_{CPU} = 8 MHz

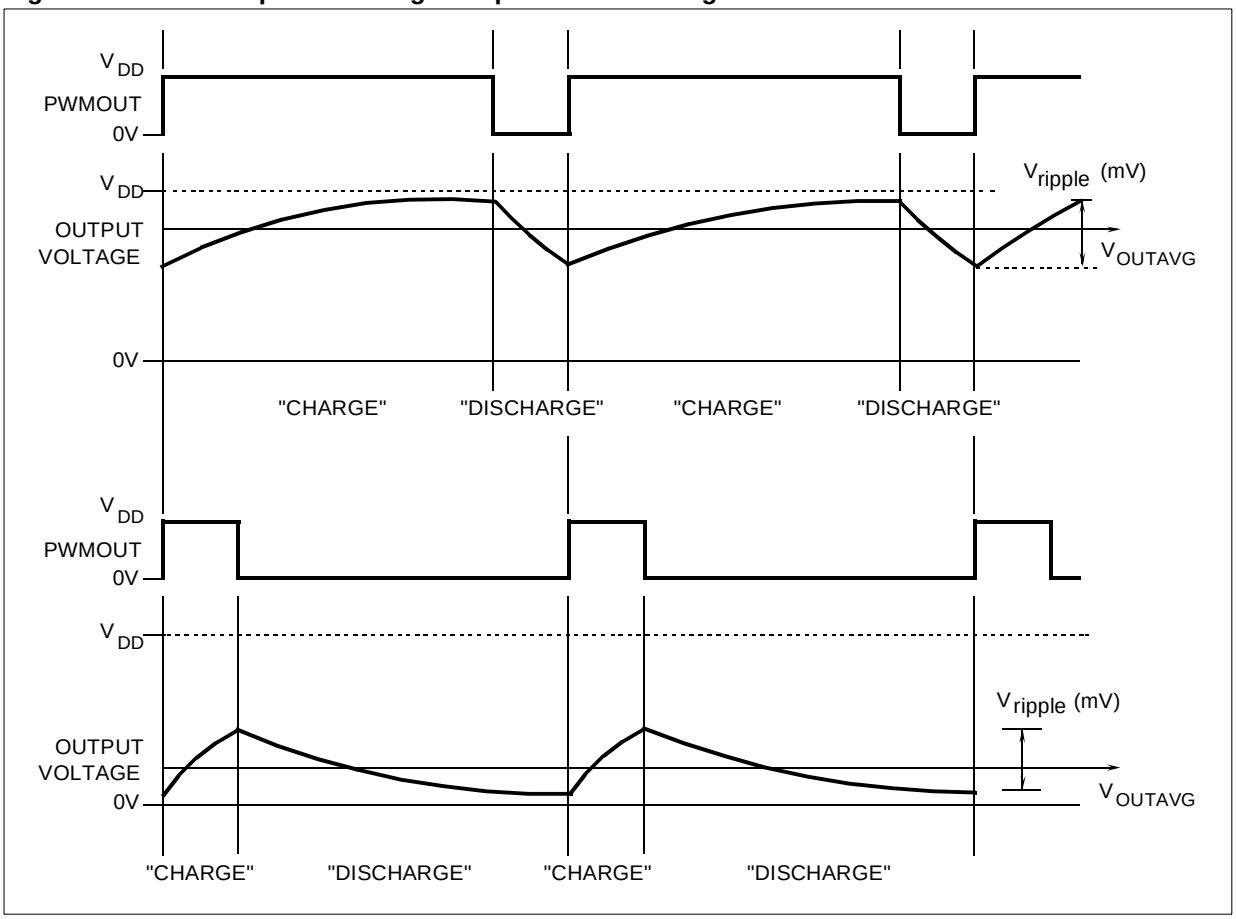
V_{DD} = 5V

PWM Duty Cycle 50%

R=R_{ext}

Note: after a reset these pins are tied low by default and are not in a high impedance state.

Figure 52. PWM Simplified Voltage Output After Filtering



PWM/BRM GENERATOR (Cont'd)

BRM Generation

The BRM bits allow the addition of a pulse to widen a standard PWM pulse for specific PWM cycles. This has the effect of "fine-tuning" the PWM Duty cycle (without modifying the base duty cycle), thus, with the external filtering, providing additional fine voltage steps.

The incremental pulses (with duration of T_{CPU}) are added to the beginning of the original PWM pulse. The PWM intervals which are added to are specified in the 4-bit BRM register and are encoded as shown in the following table. The BRM values shown may be combined together to provide a summation of the incremental pulse intervals specified.

The pulse increment corresponds to the PWM resolution.

For example,if

– Data 18h is written to the PWM register

– Data 06h (00000110b) is written to the BRM register

– with a 8MHz internal clock (125ns resolution)

Then 3.0 µs-long pulse will be output at 8 µs intervals, except for cycles numbered 2,4,6,10,12,14, where the pulse is broadened to 3.125 µs.

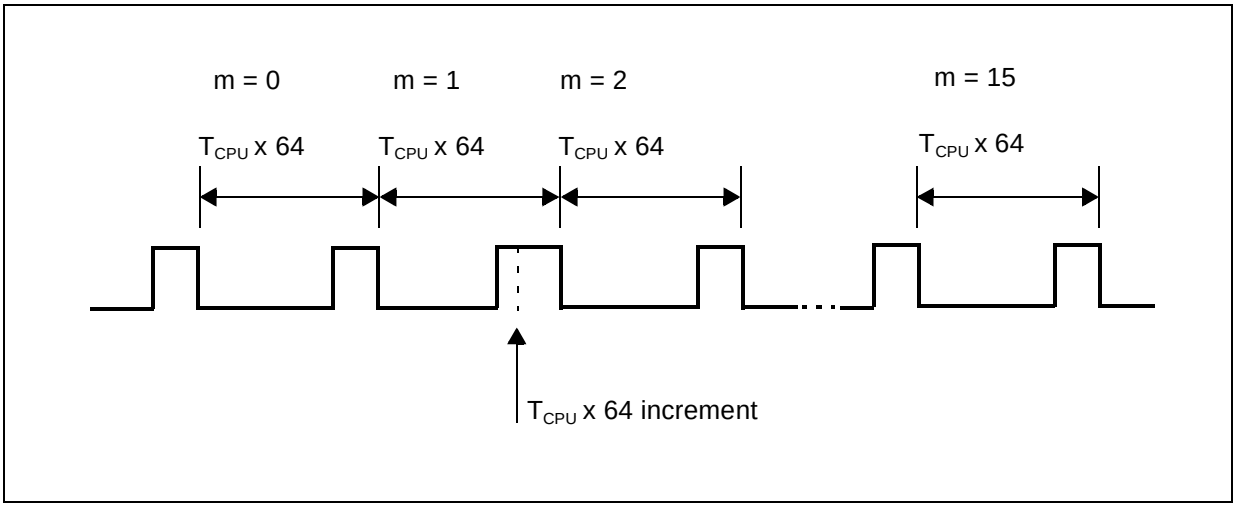
Note. If 00h is written to both PWM and BRM registers, the generator output will remain at "0". Conversely, if both registers hold data 3Fh and 0Fh, respectively, the output will remain at "1" for all intervals 1 to 15, but it will return to zero at interval 0 for an amount of time corresponding to the PWM resolution (T_{CPU}).

An output can be set to a continuous "1" level by clearing the PWM and BRM values and setting POL = "1" (inverted polarity) in the PWM register. This allows a PWM/BRM channel to be used as an additional I/O pin if the DAC function is not required.

Table 29. Bit BRM Added Pulse Intervals (Interval #0 not selected).

BRM 4 - Bit Data	Incremental Pulse Intervals
0000	none
0001	i = 8
0010	i = 4,12
0100	i = 2,6,10,14
1000	i = 1,3,5,7,9,11,13,15

Figure 53. BRM pulse addition (PWM > 0)



PWM/BRM GENERATOR (Cont'd)

Figure 54. Simplified Filtered Voltage Output Schematic with BRM Added

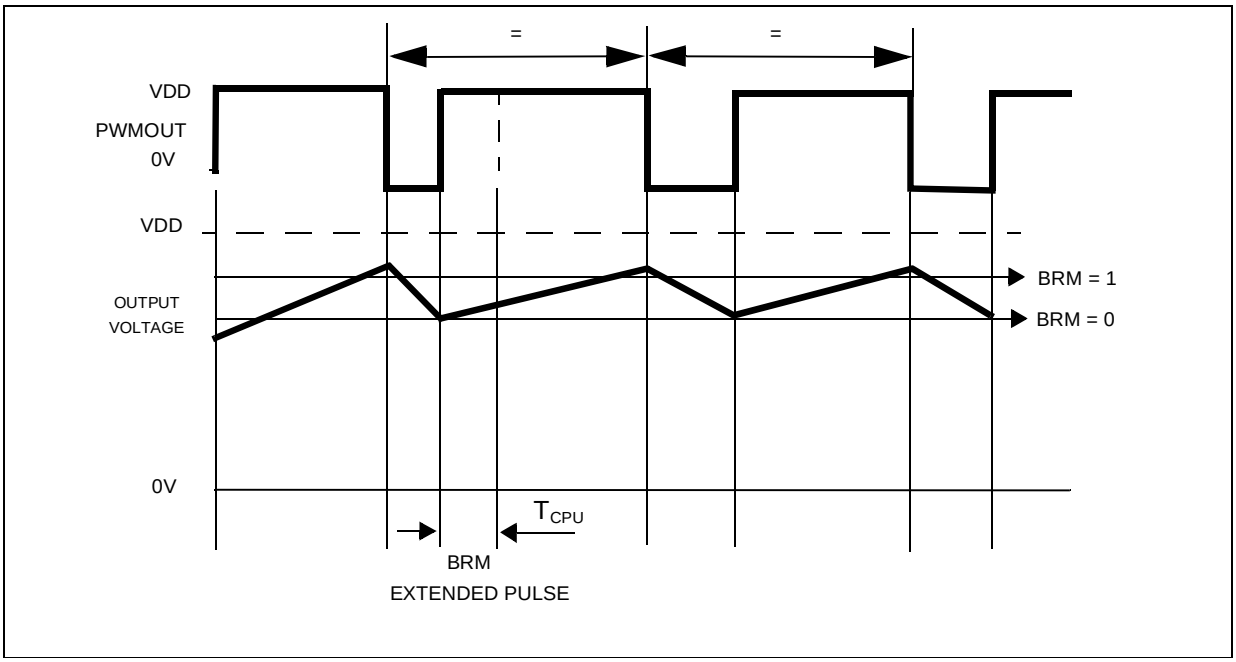
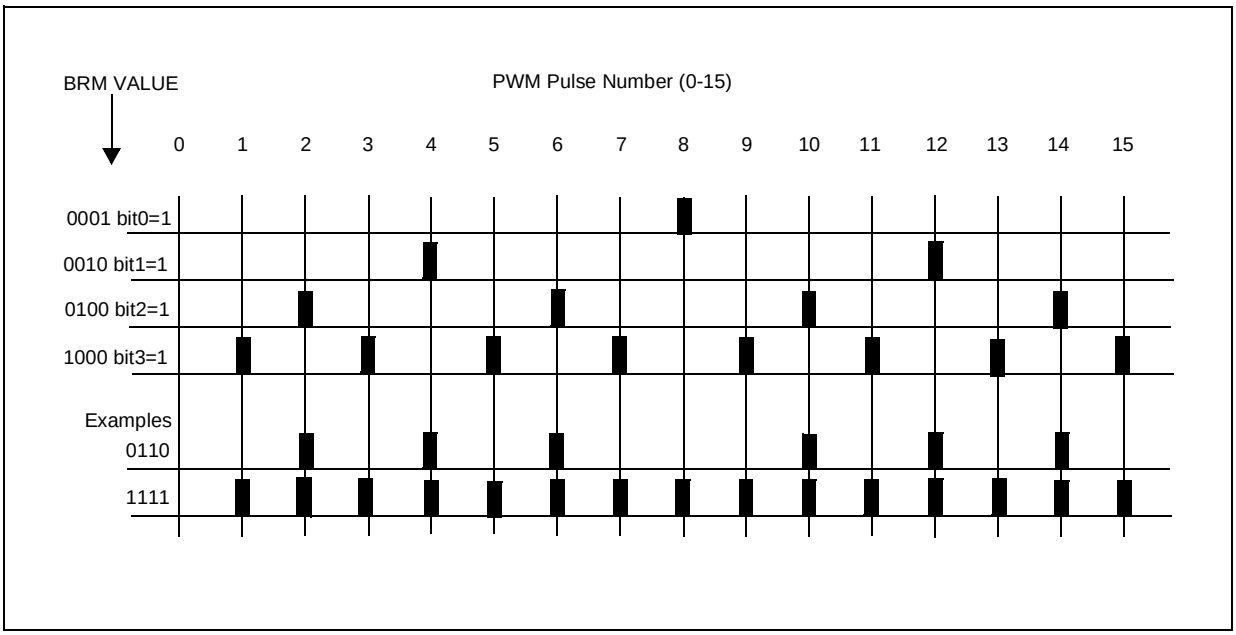
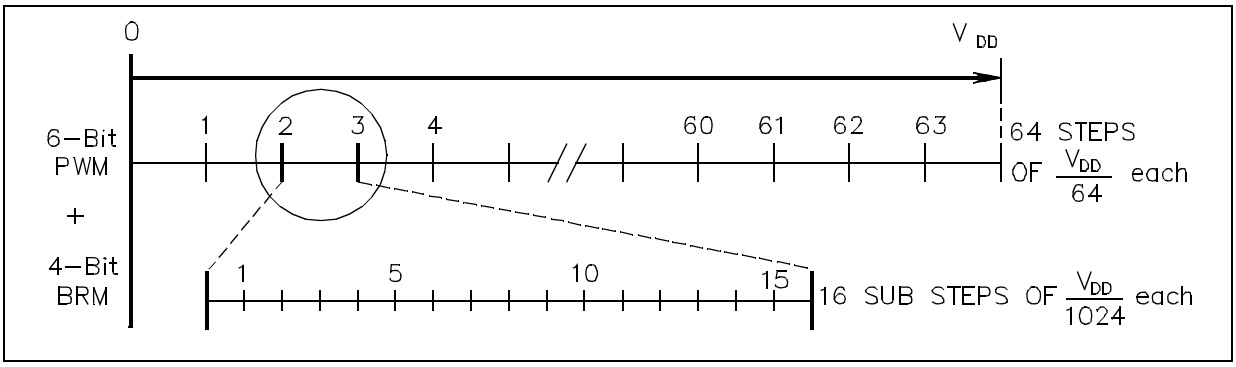


Figure 55. Graphical Representation of 4-Bit BRM Added Pulse Positions



PWM/BRM GENERATOR (Cont'd)

Figure 56. Precision for PWM/BRM Tuning for VOUTEFF (After filtering)



11.5.4 Register Description

On a channel basis, the 10 bits are separated into two data registers:

Note: The number of PWM and BRM channels available depends on the device. Refer to the device pin description and register map.

PULSE BINARY WEIGHT REGISTERS (PWMi)

Read / Write
Reset Value 1000 0000 (80h)

7							0
1	POL	P5	P4	P3	P2	P1	P0

Bit 7 = Reserved (Forced by hardware to "1")

Bit 6 = **POL** *Polarity Bit for channel i.*
0: The channel *i* outputs a "1" level during the binary pulse and a "0" level after.
1: The channel *i* outputs a "0" level during the binary pulse and a "1" level after.

Bit 5:0 = **P[5:0]** *PWM Pulse Binary Weight for channel i.*

This register contains the binary value of the pulse.

BRM REGISTERS

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

7							0
B7	B6	B5	B4	B3	B2	B1	B0

These registers define the intervals where an incremental pulse is added to the beginning of the original PWM pulse. Two BRM channel values share the same register.

Bit 7:4 = **B[7:4]** *BRM Bits (channel i+1).*

Bit 3:0 = **B[3:0]** *BRM Bits (channel i)*

Note: From the programmer's point of view, the PWM and BRM registers can be regarded as being combined to give one data value.

For example :

1	POL	P	P	P	P	P	P	+	B	B	B	B

Effective (with external RC filtering) DAC value

1	POL	P	P	P	P	P	P	B	B	B	B

PULSE WIDTH MODULATION (Cont'd)

Table 30. PWM Register Map and Reset Values

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
4D	**PWM0**	1	POL	P5	P4	P3	P2	P1	P0
	Reset Value	1	0	0	0	0	0	0	0
4E	**BRM10**	B7	B6	B5	B4	B3	B2	B1	B0
	Reset Value	0	0	0	0	0	0	0	0
4F	**PWM1**	1	POL	P5	P4	P3	P2	P1	P0
	Reset Value	1	0	0	0	0	0	0	0

11.6 SERIAL PERIPHERAL INTERFACE (SPI)

11.6.1 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 however the SPI interface can not be a master in a multi-master system.

11.6.2 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}/2 max.)
- f_{CPU}/2 max. slave mode frequency
- $\overline{SS}$ Management by software or hardware
- Programmable clock polarity and phase
- End of transfer interrupt flag
- Write collision, Master Mode Fault and Overrun flags

11.6.3 General Description

Figure 57 shows the serial peripheral interface (SPI) block diagram. There are 3 registers:

– SPI Control Register (SPICR)

– SPI Control/Status Register (SPICSR)

– SPI Data Register (SPIDR)

The SPI is connected to external devices through 3 pins:

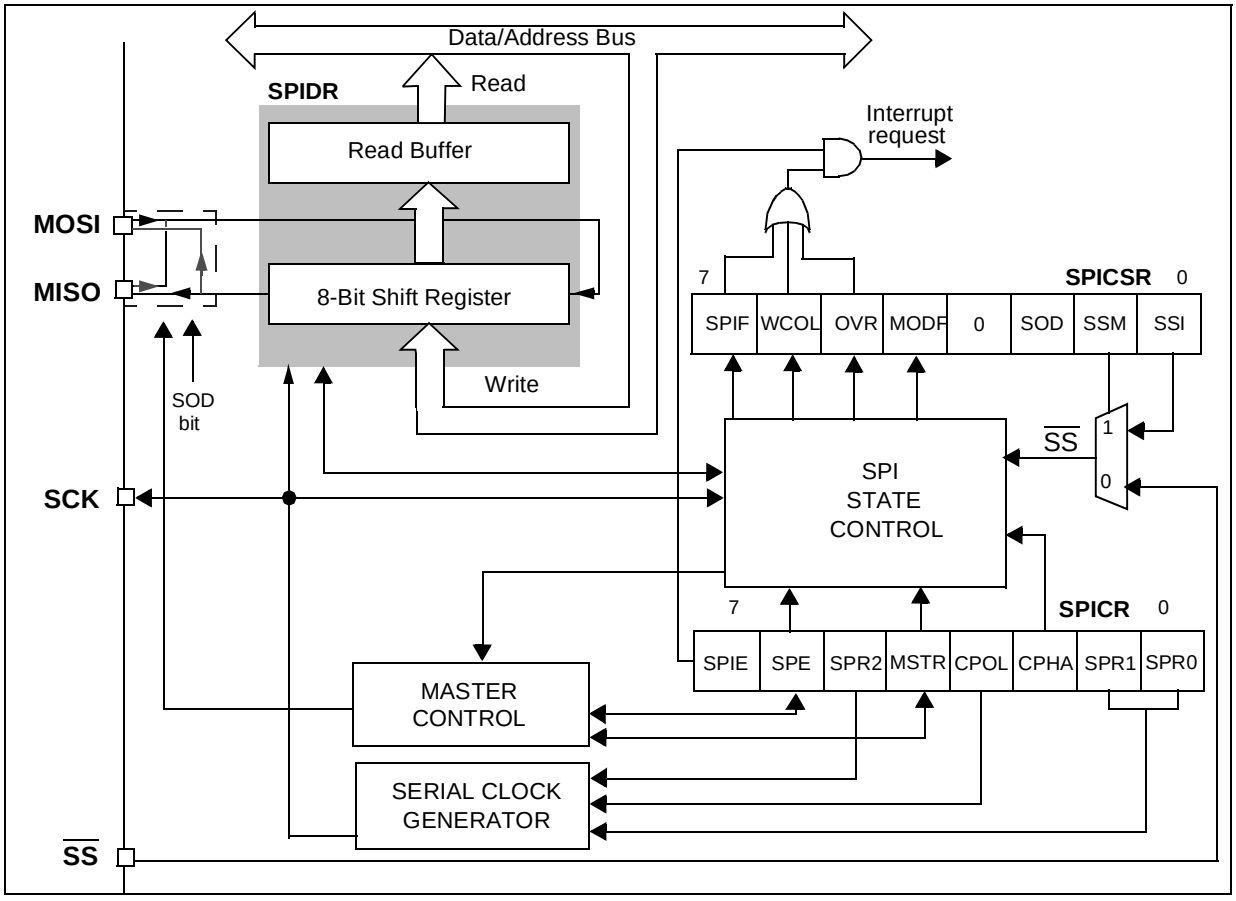
– MISO: Master In / Slave Out data

– MOSI: Master Out / Slave In data

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

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

Figure 57. Serial Peripheral Interface Block Diagram



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.6.3.1 Functional Description

A basic example of interconnections between a single master and a single slave is illustrated in Figure 58.

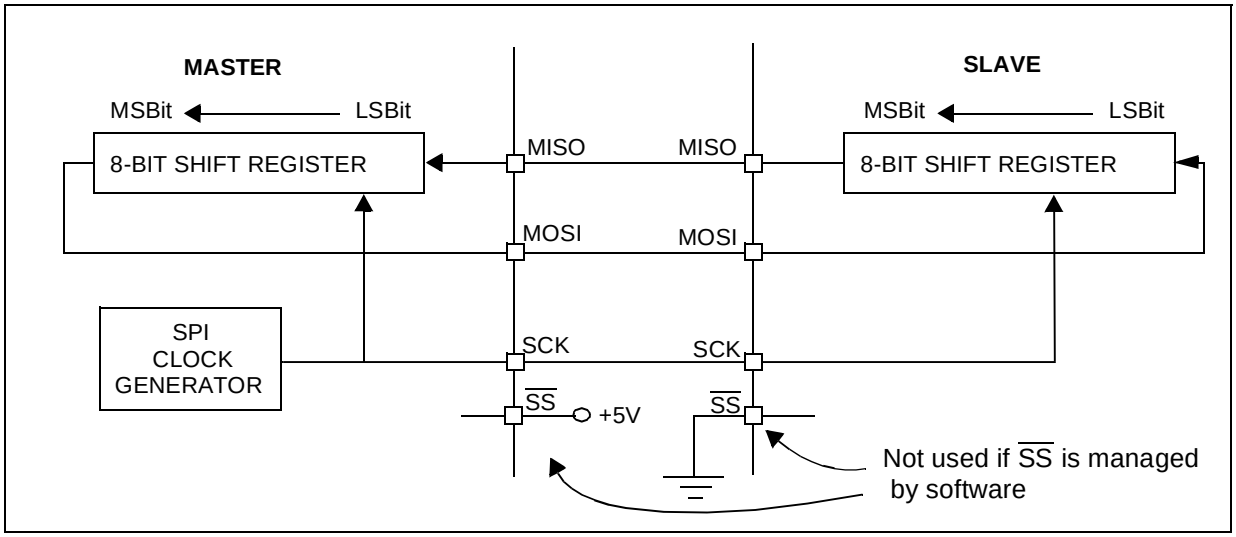
The MOSI pins are connected together and the MISO pins are connected together. In this way data is transferred serially between master and slave (most significant bit first).

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

Figure 58. Single Master/ Single Slave Application



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.6.3.2 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 60)

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 59):

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 SS pin either can be tied to V_{SS}, or made free for standard I/O by managing the 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 SS is not pulled high, a Write Collision error will occur when the slave writes to the shift register (see Section 11.6.5.3).

Figure 59. Generic $\overline{SS}$ Timing Diagram

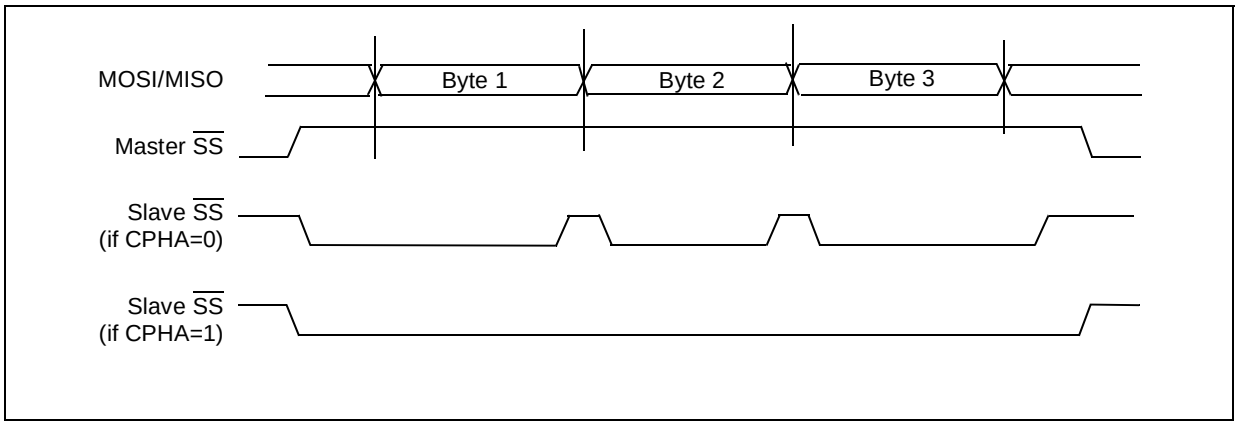
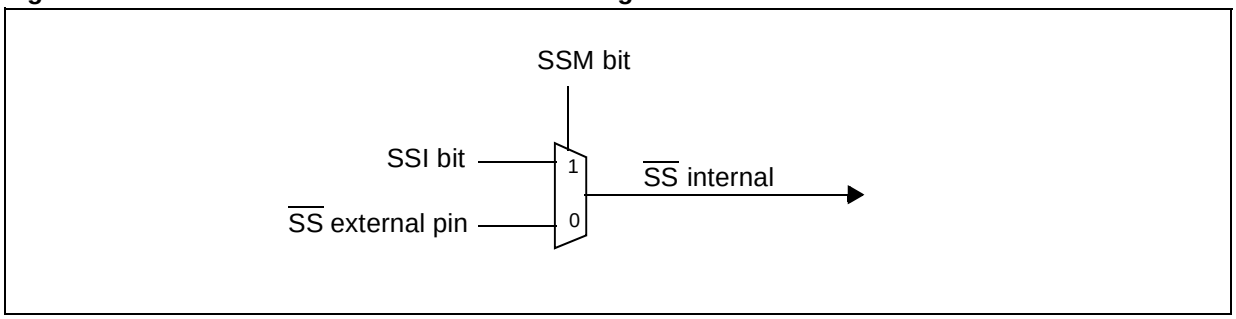


Figure 60. Hardware/Software Slave Select Management



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.6.3.3 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 two steps in order (if the SPICSR register is not written first, the SPICR register setting may be not taken into account):

1. Write to the SPICSR 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 61 shows the four possible configurations.
     Note: The slave must have the same CPOL and CPHA settings as the master.
   - Either set the SSM bit and set the SSI bit or clear the SSM bit and tie the SS pin high for the complete byte transmit sequence.
2. Write to the SPICR register:
   - Set the MSTR and SPE bits
     Note: MSTR and SPE bits remain set only if SS is high).

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

11.6.3.4 Master Mode Transmit Sequence

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

When data transfer is complete:

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

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

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

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

11.6.3.5 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 61).
     Note: The slave must have the same CPOL and CPHA settings as the master.
   - Manage the $\overline{SS}$ pin as described in Section 11.6.3.2 and Figure 59. If CPHA=1 $\overline{SS}$ must be held low continuously. If CPHA=0 $\overline{SS}$ must be held low during byte transmission and pulled up between each byte to let the slave write in the shift register.
2. Write to the SPICR register to clear the MSTR bit and set the SPE bit to enable the SPI I/O functions.

11.6.3.6 Slave Mode Transmit Sequence

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

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

When data transfer is complete:

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

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

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

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

The SPIF bit can be cleared during a second transmission; however, it must be cleared before the second SPIF bit in order to prevent an Overrun condition (see Section 11.6.5.2).

SERIAL PERIPHERAL INTERFACE (Cont'd)

11.6.4 Clock Phase and Clock Polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits (See Figure 61).

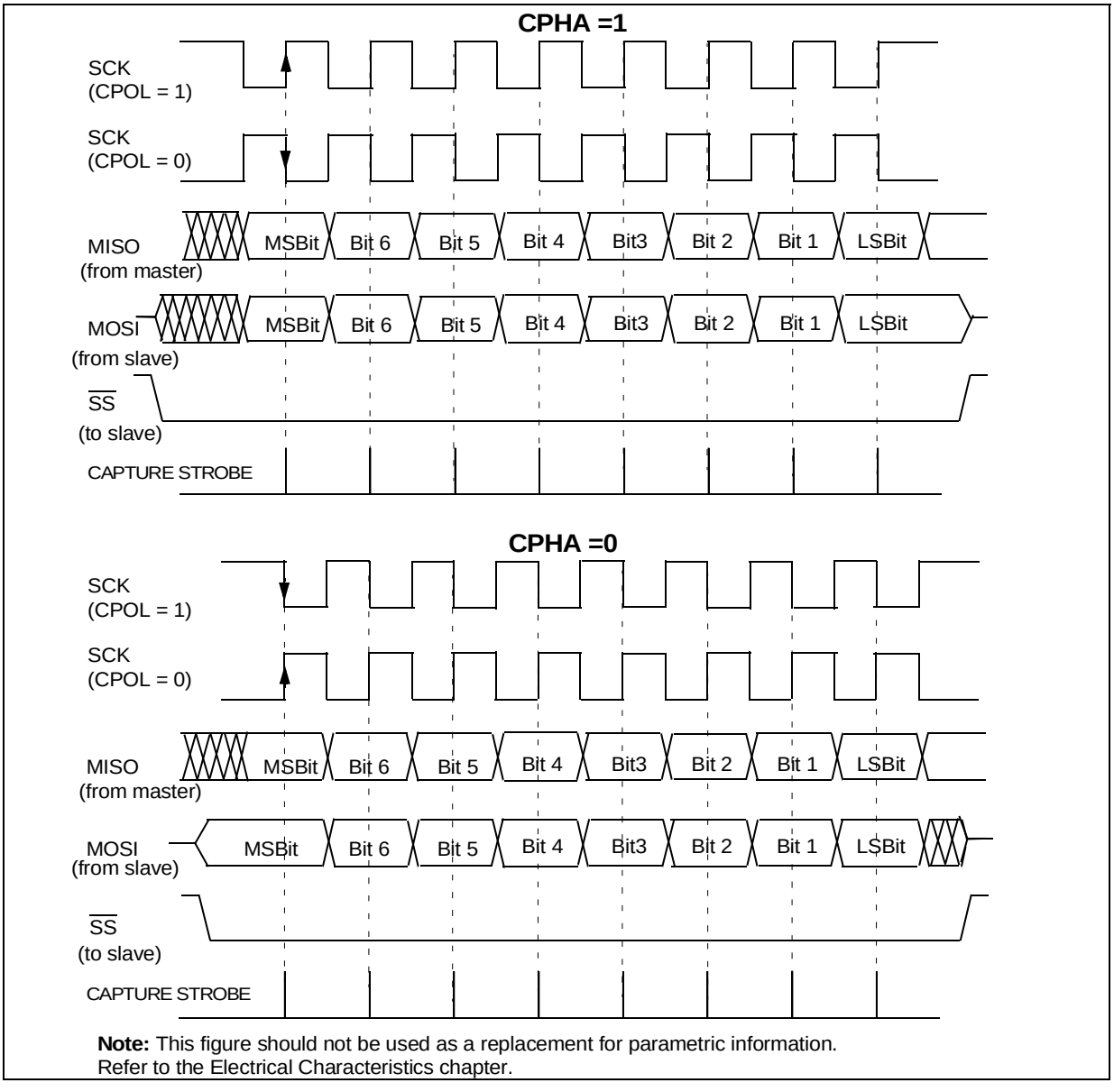
Note: The idle state of SCK must correspond to the polarity selected in the SPICSR register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).

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

Figure 61, 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 61. Data Clock Timing Diagram



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

SERIAL PERIPHERAL INTERFACE (Cont'd)

11.6.5 Error Flags

11.6.5.1 Master Mode Fault (MODF)

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

When a Master mode fault occurs:

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

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

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

Clearing the MODF bit is done through a software sequence:

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

2. A write to the SPICR register.

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

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

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

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

11.6.5.2 Overrun Condition (OVR)

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

When an Overrun occurs:

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

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

The OVR bit is cleared by reading the SPICSR register.

11.6.5.3 Write Collision Error (WCOL)

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

Write collisions can occur both in master and slave mode. See also Section 11.6.3.2 Slave Select Management.

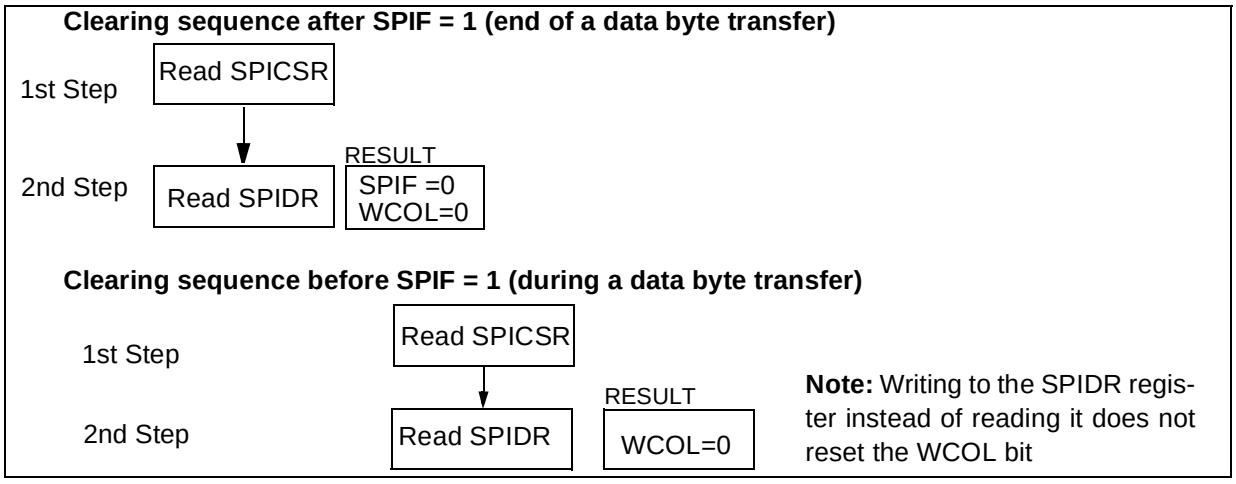
Note: a "read collision" will never occur since the received data byte is placed in a buffer in which access is always synchronous with the MCU operation.

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 62).

Figure 62. Clearing the WCOL bit (Write Collision Flag) Software Sequence



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.6.5.4 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 an MCU as the master and four MCUs as slaves (see Figure 63).

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.

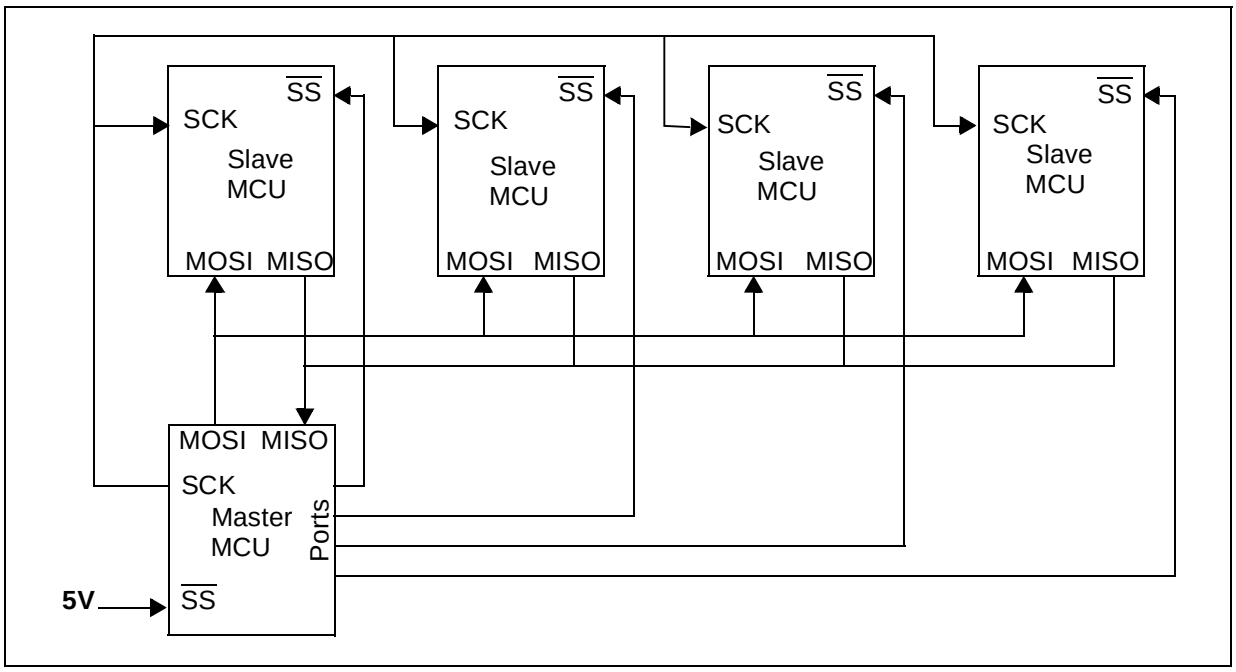
The $\overline{SS}$ pins are pulled high during reset since the master device ports will be forced to be inputs at that time, thus disabling the slave devices.

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.

Figure 63. Single Master / Multiple Slave Configuration



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.6.6 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 MCU is woken up by an interrupt with "exit from HALT mode" capability. The data received is subsequently read from the SPIDR register when the software is running (interrupt vector fetching). If several data are received before the wake-up event, then an overrun error is generated. This error can be detected after the fetch of the interrupt routine that woke up the device.

11.6.6.1 Using the SPI to wakeup the MCU from Halt mode

In slave configuration, the SPI is able to wakeup the ST7 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 ST7 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 ST7 enters Halt mode. So if Slave selection is configured as external (see Section 11.6.3.2), make sure the master drives a low level on the $\overline{SS}$ pin when the slave enters Halt mode.

11.6.7 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

SERIAL PERIPHERAL INTERFACE (Cont'd)

11.6.8 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 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, SS=0 (see Section 11.6.5.1 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 31 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, SS=0 (see Section 11.6.5.1 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 31. SPI Master mode SCK Frequency

Serial Clock	SPR2	SPR1	SPR0
$f_{CPU}/2$	1	0	0
$f_{CPU}/4$	0	0	0
$f_{CPU}/8$	0	0	1
$f_{CPU}/16$	1	1	0
$f_{CPU}/32$	0	1	0
$f_{CPU}/64$	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 62).
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 11.6.5.2). An interrupt is generated if SPIE = 1 in SPICSR register. The OVR bit is cleared by software reading the SPICSR register.
0: No overrun error
1: Overrun error detected

Bit 4 = **MODF** *Mode Fault flag (Read only).*
This bit is set by hardware when the $\overline{SS}$ pin is pulled low in master mode (see Section 11.6.5.1 Master Mode Fault (MODF)). An SPI interrupt can be generated if SPIE=1 in the SPICSR register. This bit is cleared by a software sequence (An access to the SPICSR register while MODF=1 followed by a write to the SPICR register).
0: No master mode fault detected
1: A fault in master mode has been detected

Bit 3 = Reserved, must be kept cleared.

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

Bit 1 = **SSM** $\overline{SS}$ *Management.*
This bit is set and cleared by software. When set, it disables the alternate function of the SPI $\overline{SS}$ pin and uses the SSI bit value instead. See Section 11.6.3.2 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 57).

SERIAL PERIPHERAL INTERFACE (Cont'd)

Table 32. SPI Register Map and Reset Values

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

11.7 I²C SINGLE MASTER BUS INTERFACE (I2C)

11.7.1 Introduction

The I^2C Bus Interface serves as an interface between the microcontroller and the serial I^2C bus. It provides single master functions, and controls all I^2C bus-specific sequencing, protocol and timing. It supports fast I²C mode (400kHz).

11.7.2 Main Features

- Parallel bus/I^2C protocol converter
- Interrupt generation
- Standard I^2C mode/Fast I^2C mode
- 7-bit Addressing

■ **I^2C single Master Mode**

- I^2C bus busy flag
- End of byte transmission flag
- Transmitter/Receiver flag
- Clock generation

11.7.3 General Description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa, using either an interrupt or polled handshake. The interrupts are enabled or disabled by software. The interface is connected to the I^2C bus by a data pin (SDAI) and by a clock pin (SCLI). It can be connected both with a standard I^2C bus and a Fast I^2C bus. This selection is made by software.

Mode Selection

The interface can operate in the two following modes:

– Master transmitter/receiver

By default, it is idle.

The interface automatically switches from idle to master after it generates a START condition and from master to idle after it generates a STOP condition.

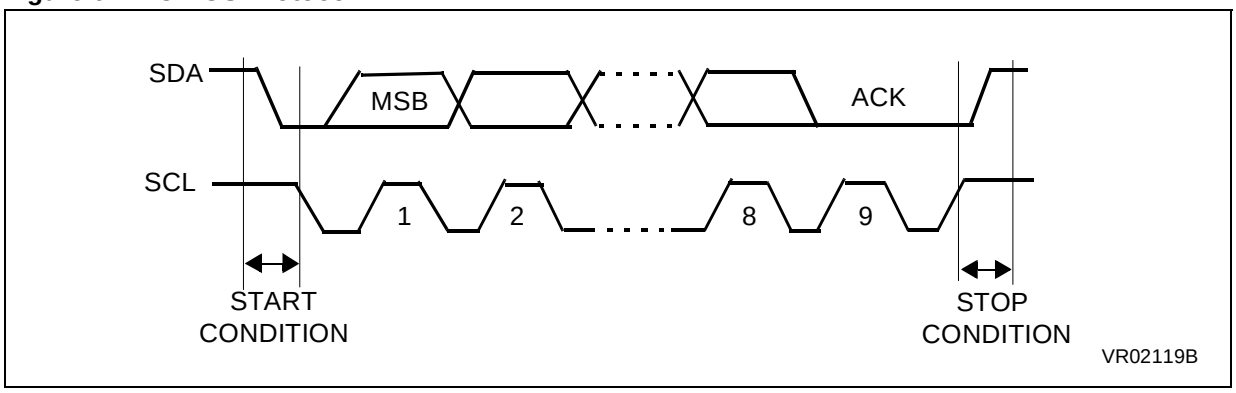
Communication Flow

The interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a start condition and ends with a stop condition. Both start and stop conditions are generated by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte following the start condition is the address byte.

A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to Figure 64.

Figure 64. I^2C BUS Protocol



I²C SINGLE MASTER BUS INTERFACE (Cont'd)

Acknowledge may be enabled and disabled by software.

The speed of the I^2C interface may be selected between Standard (0-100KHz) and Fast I^2C (100-400KHz).

SDA/SCL Line Control

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

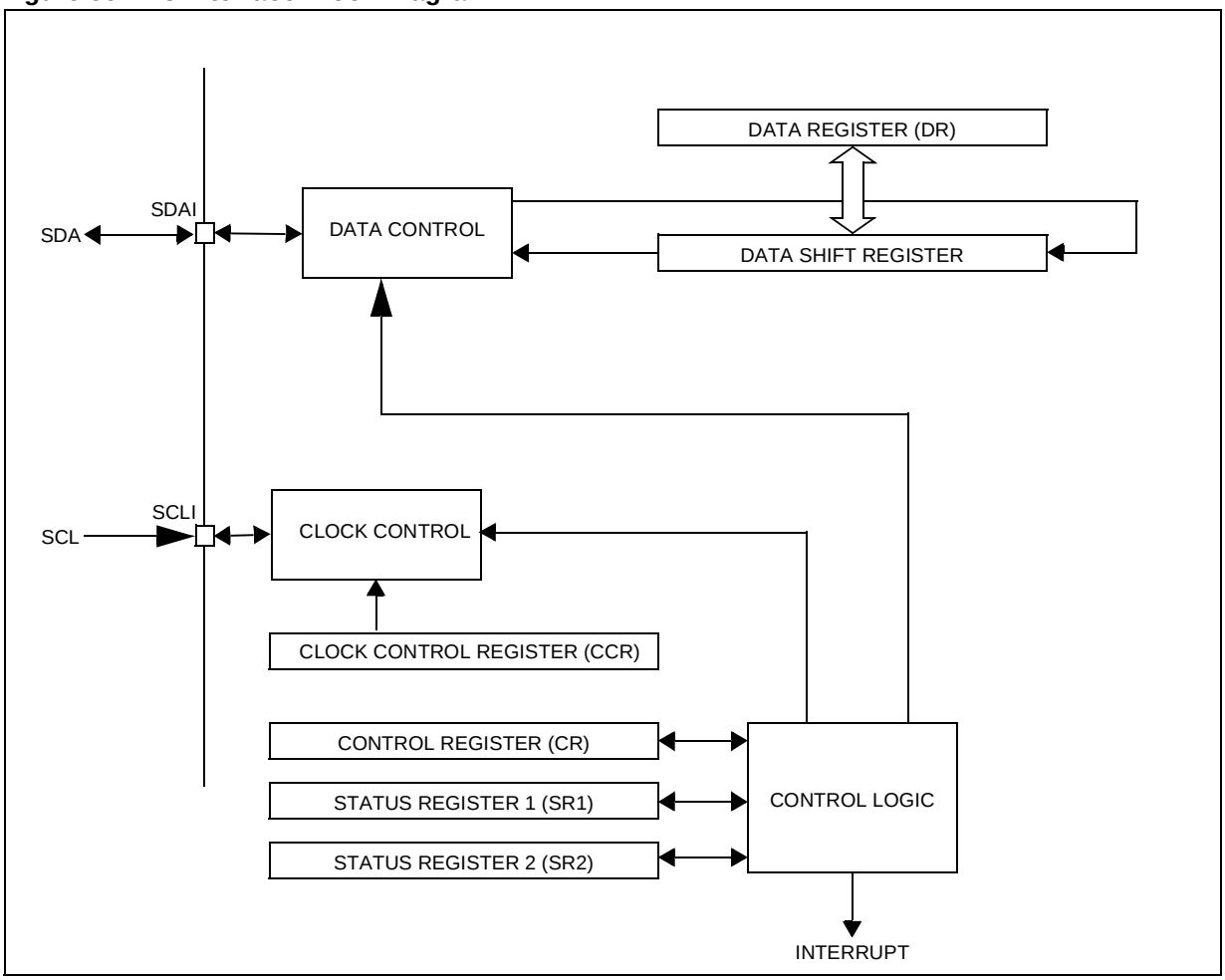
Receiver mode: the interface holds the clock line low after reception to wait for the microcontroller to read the byte in the Data Register.

The SCL frequency (F_{scl}) is controlled by a programmable clock divider which depends on the I^2C bus mode.

When the I2C cell is enabled, the SDA and SCL ports must be configured as floating open-drain output or floating input. In this case, the value of the external pull-up resistance used depends on the application.

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

Figure 65. I^2C Interface Block Diagram



I²C SINGLE MASTER BUS INTERFACE (Cont'd)

11.7.4 Functional Description (Master Mode)

Refer to the CR, SR1 and SR2 registers in Section 11.7.5. for the bit definitions.

By default the I^2C interface operates in idle mode (M/IDL bit is cleared) except when it initiates a transmit or receive sequence.

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

Start condition and Transmit Slave address

Setting the START bit causes the interface to switch to Master mode (M/IDL bit set) and generates a Start condition.

Once the Start condition is sent:

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

Then the master waits for a read of the SR1 register followed by a write in the DR register with the Slave address byte, **holding the SCL line low** (see Figure 66 Transfer sequencing EV1).

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

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

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

Then the master waits for a read of the SR1 register followed by a write in the CR register (for example set PE bit), **holding the SCL line low** (see Figure 66 Transfer sequencing EV2).

Next the master must enter Receiver or Transmitter mode.

Master Receiver

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

– Acknowledge pulse if if the ACK bit is set

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

Then the interface waits for a read of the SR1 register followed by a read of the DR register, **holding the SCL line low** (see Figure 66 Transfer sequencing EV3).

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

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

Master Transmitter

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

The master waits for a read of the SR1 register followed by a write in the DR register, **holding the SCL line low** (see Figure 66 Transfer sequencing EV4).

When the acknowledge bit is received, the interface sets:

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

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

Error Case

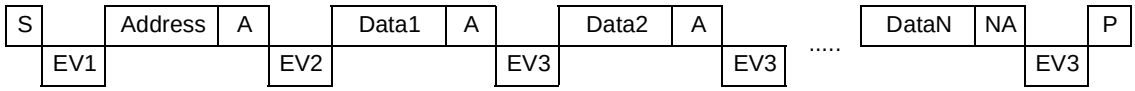
– **AF**: Detection of a non-acknowledge bit. In this case, the EVF and AF bits are set by hardware with an interrupt if the ITE bit is set. To resume, set the START or STOP bit.

Note: The SCL line is not held low.

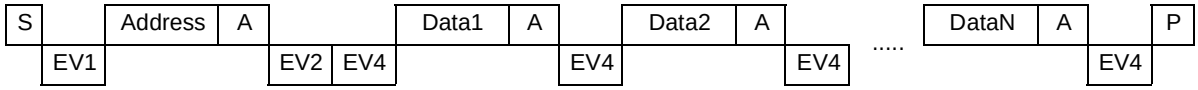
I²C SINGLE MASTER BUS INTERFACE (Cont'd)

Figure 66. Transfer Sequencing

Master receiver:



Master transmitter:



Legend:

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

EVx=Event (with interrupt if ITE=1)

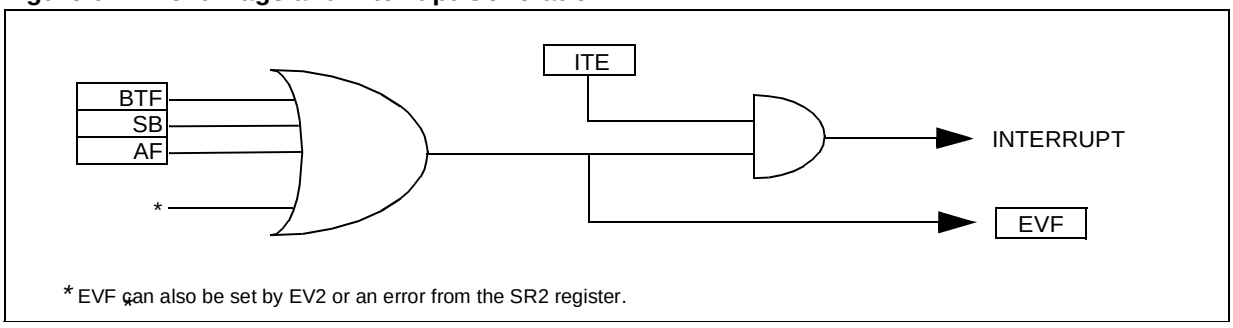
EV1: EVF=1, SB=1, cleared by reading SR1 register followed by writing DR register.

EV2: EVF=1, cleared by reading SR1 register followed by writing CR register (for example PE=1).

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

EV4: EVF=1, BTF=1, cleared by reading SR1 register followed by writing DR register.

Figure 67. Event Flags and Interrupt Generation



I²C SINGLE MASTER BUS INTERFACE (Cont'd)

11.7.5 Register Description

I²C CONTROL REGISTER (CR)

Read / Write

Reset Value: 0000 0000 (00h)

7							0
0	0	PE	0	START	ACK	STOP	ITE

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

Bit 5 = **PE** *Peripheral enable.*
This bit is set and cleared by software.
0: Peripheral disabled
1: Master capability
Notes:
– When PE=0, all the bits of the CR register and the SR register except the Stop bit are reset. All outputs are released while PE=0
– When PE=1, the corresponding I/O pins are selected by hardware as alternate functions.
– To enable the I^2C interface, write the CR register **TWICE** with PE=1 as the first write only activates the interface (only PE is set).

Bit 4 = Reserved. Forced to 0 by hardware.

Bit 3 = **START** *Generation of a Start condition*.
This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE=0) or when the Start condition is sent (with interrupt generation if ITE=1).
– In master mode:
0: No start generation
1: Repeated start generation
– In idle mode:
0: No start generation
1: Start generation when the bus is free

Bit 2 = **ACK** *Acknowledge enable.*
This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE=0).
0: No acknowledge returned
1: Acknowledge returned after a data byte is received

Bit 1 = **STOP** *Generation of a Stop condition*.
This bit is set and cleared by software. It is also cleared by hardware when the interface is disabled (PE=0) or when the Stop condition is sent.
– In Master mode only:
0: No stop generation
1: Stop generation after the current byte transfer or after the current Start condition is sent.

Bit 0 = **ITE** *Interrupt enable.*
This bit is set and cleared by software and cleared by hardware when the interface is disabled (PE=0).
0: Interrupts disabled
1: Interrupts enabled

Refer to Figure 67 for the relationship between the events and the interrupt.
SCL is held low when the SB or BTF flags or an EV2 event (See Figure 66) is detected.

I²C SINGLE MASTER BUS INTERFACE (Cont'd)

I²C STATUS REGISTER 1 (SR1)

Read Only

Reset Value: 0000 0000 (00h)

7							0
EVF	0	TRA	0	BTF	0	M/IDL	SB

Bit 7 = **EVF** *Event flag.*
This bit is set by hardware as soon as an event occurs. It is cleared by software reading SR2 register in case of error event or as described in Figure 66. It is also cleared by hardware when the interface is disabled (PE=0).
0: No event
1: One of the following events has occurred:

- BTF=1 (Byte received or transmitted)
- SB=1 (Start condition generated)
- AF=1 (No acknowledge received after byte transmission if ACK=1)
- Address byte successfully transmitted.

Bit 6 = Reserved. Forced to 0 by hardware.

Bit 5 = **TRA** *Transmitter/Receiver.*
When BTF is set, TRA=1 if a data byte has been transmitted. It is cleared automatically when BTF is cleared. It is also cleared by hardware when the interface is disabled (PE=0).
0: Data byte received (if BTF=1)
1: Data byte transmitted

Bit 4 = Reserved. Forced to 0 by hardware.

Bit 3 = **BTF** *Byte transfer finished.*
This bit is set by hardware as soon as a byte is correctly received or transmitted with interrupt generation if ITE=1. It is cleared by software reading SR1 register followed by a read or write of DR register. It is also cleared by hardware when the interface is disabled (PE=0).

- Following a byte transmission, this bit is set after reception of the acknowledge clock pulse. In case an address byte is sent, this bit is set only after the EV2 event (See Figure 66). BTF is cleared by reading SR1 register followed by writing the next byte in DR register.
- Following a byte reception, this bit is set after transmission of the acknowledge clock pulse if ACK=1. BTF is cleared by reading SR1 register followed by reading the byte from DR register.

The SCL line is held low while BTF=1.

0: Byte transfer not done
1: Byte transfer succeeded

Bit 2 = Reserved. Forced to 0 by hardware.

Bit 1 = **M/IDL** *Master/Idle.*
This bit is set by hardware as soon as the interface is in Master mode (writing START=1). It is cleared by hardware after generating a Stop condition on the bus. It is also cleared when the interface is disabled (PE=0).
0: Idle mode
1: Master mode

Bit 0 = **SB** *Start bit generated.*
This bit is set by hardware as soon as the Start condition is generated (following a write START=1). An interrupt is generated if ITE=1. It is cleared by software reading SR1 register followed by writing the address byte in DR register. It is also cleared by hardware when the interface is disabled (PE=0).
0: No Start condition
1: Start condition generated

I²C SINGLE MASTER BUS INTERFACE (Cont'd)

I²C STATUS REGISTER 2 (SR2)

Read Only

Reset Value: 0000 0000 (00h)

7							0
0	0	0	AF	0	0	0	0

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

Bit 4 = **AF** *Acknowledge failure*.
This bit is set by hardware when no acknowledge is returned. An interrupt is generated if ITE=1. It is cleared by software reading SR2 register or by hardware when the interface is disabled (PE=0).

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

0: No acknowledge failure
1: Acknowledge failure

Bit 3:0 = Reserved. Forced to 0 by hardware.

I²C SINGLE MASTER BUS INTERFACE (Cont'd)

I²C CLOCK CONTROL REGISTER (CCR)

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

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

Bit 7 = **FM/SM** *Fast/Standard I²C mode.*
This bit is set and cleared by software. It is not cleared when the interface is disabled (PE=0).
0: Standard I²C mode
1: Fast I²C mode

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

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

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

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

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

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

I²C DATA REGISTER (DR)

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

7							0
D7	D6	D5	D4	D3	D2	D1	D0

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

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

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

Table 33. I²C Register Map

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
40	CR			PE		START	ACK	STOP	ITE
	Reset Value	0	0	0	0	0	0	0	0
41	SR1	EVF		TRA	BUSY*	BTF		M/IDL	SB
	Reset Value	0	0	0	0	0	0	0	0
42	SR2				AF			BERR	
	Reset Value	0	0	0	0	0	0	0	0
43	CCR	FM/SM	CC6	CC5	CC4	CC3	CC2	CC1	CC0
	Reset Value	0	0	0	0	0	0	0	0
46	DR	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0
	Reset Value	0	0	0	0	0	0	0	0

*Not present. Forced to 0 by hardware

11.8 8-BIT A/D CONVERTER (ADC)

11.8.1 Introduction

The on-chip Analog to Digital Converter (ADC) peripheral is a 8-bit, successive approximation converter with internal sample and hold circuitry. This peripheral has up to 16 multiplexed analog input channels (refer to device pin out description) that allow the peripheral to convert the analog voltage levels from up to 16 different sources.

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

11.8.2 Main Features

- 8-bit conversion
- Up to 16 channels with multiplexed input
- Linear successive approximation
- Data register (DR) which contains the results
- Conversion complete status flag
- On/off bit (to reduce consumption)

The block diagram is shown in Figure 68.

11.8.3 Functional Description

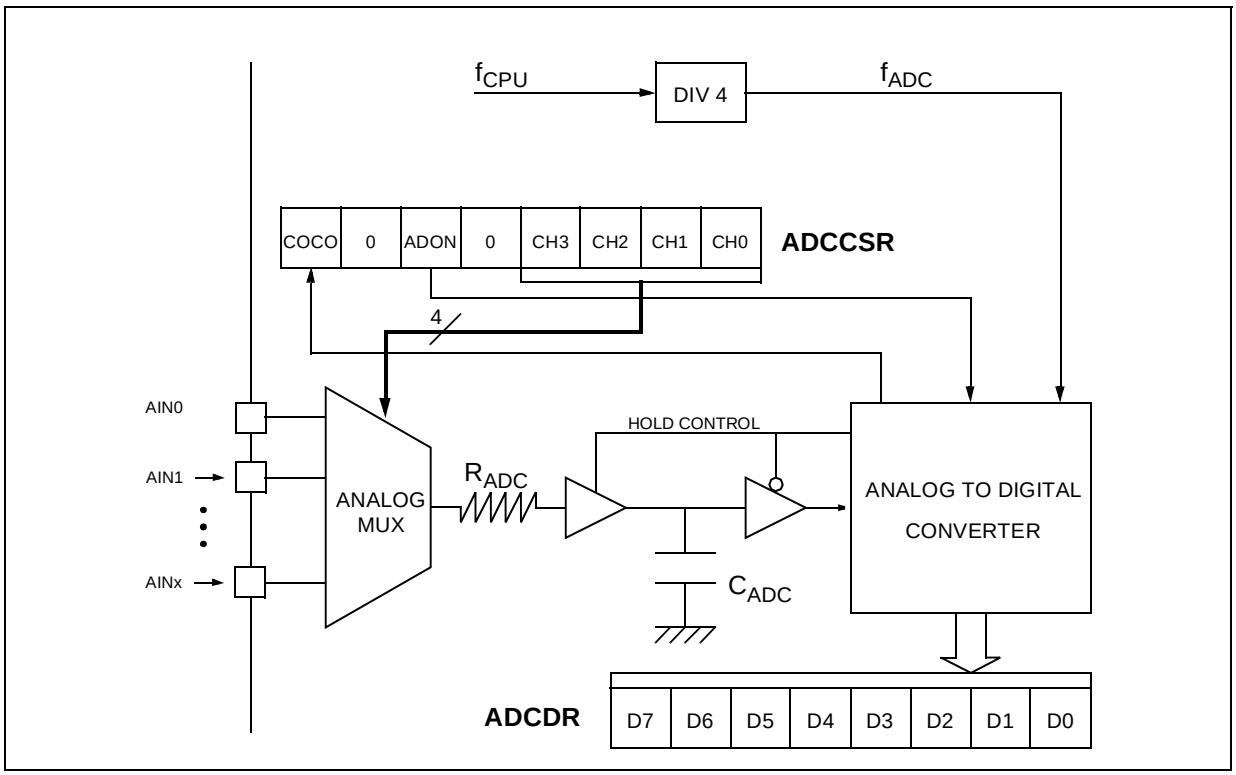
11.8.3.1 Analog Power Supply

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

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

See electrical characteristics section for more details.

Figure 68. ADC Block Diagram



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

11.8.3.2 Digital A/D Conversion Result

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

If the input voltage (V_{AIN}) is greater than or equal to V_{DDA} (high-level voltage reference) then the conversion result in the DR register is FFh (full scale) without overflow indication.

If input voltage (V_{AIN}) is lower than or equal to V_{SSA} (low-level voltage reference) then the conversion result in the DR register is 00h.

The A/D converter is linear and the digital result of the conversion is stored in the ADCDR register. The accuracy of the conversion is described in the parametric section.

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

11.8.3.3 A/D Conversion Phases

The A/D conversion is based on two conversion phases as shown in Figure 69:

- Sample capacitor loading [duration: t_{LOAD}]
  During this phase, the V_{AIN} input voltage to be measured is loaded into the C_{ADC} sample capacitor.
- A/D conversion [duration: t_{CONV}]
  During this phase, the A/D conversion is computed (8 successive approximation cycles) and the C_{ADC} sample capacitor is disconnected from the analog input pin to get the optimum analog to digital conversion accuracy.

While the ADC is on, these two phases are continuously repeated.

At the end of each conversion, the sample capacitor is kept loaded with the previous measurement load. The advantage of this behaviour is that it minimizes the current consumption on the analog pin in case of single input channel measurement.

11.8.3.4 Software Procedure

Refer to the control/status register (CSR) and data register (DR) in Section 11.8.6 for the bit definitions and to Figure 69 for the timings.

ADC Configuration

The total duration of the A/D conversion is 12 ADC clock periods ($1/f_{ADC}=4/f_{CPU}$).

The analog input ports must be configured as input, no pull-up, no interrupt. Refer to the «I/O ports» chapter. Using these pins as analog inputs does not affect the ability of the port to be read as a logic input.

In the CSR register:

– Select the CH[3:0] bits to assign the analog channel to be converted.

ADC Conversion

In the CSR register:

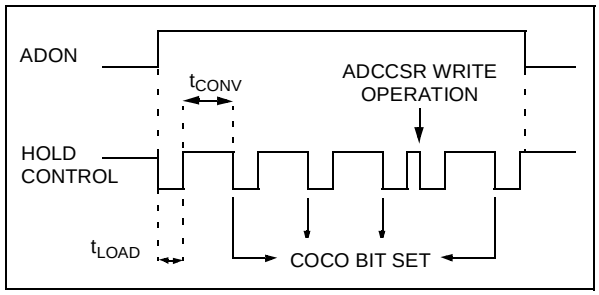
– Set the ADON bit to enable the A/D converter and to start the first conversion. From this time on, the ADC performs a continuous conversion of the selected channel.

When a conversion is complete

– The COCO bit is set by hardware.
– No interrupt is generated.
– The result is in the DR register and remains valid until the next conversion has ended.

A write to the CSR register (with ADON set) aborts the current conversion, resets the COCO bit and starts a new conversion.

Figure 69. ADC Conversion Timings



11.8.4 Low Power Modes

Mode	Description
WAIT	No effect on A/D Converter
HALT	A/D Converter disabled. After wakeup from Halt mode, the A/D Converter requires a stabilisation time before accurate conversions can be performed.

Note: The A/D converter may be disabled by resetting the ADON bit. This feature allows reduced power consumption when no conversion is needed and between single shot conversions.

11.8.5 Interrupts

None

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

11.8.6 Register Description

CONTROL/STATUS REGISTER (CSR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
COCO	0	ADON	0	CH3	CH2	CH1	CH0

Bit 7 = **COCO** *Conversion Complete*
This bit is set by hardware. It is cleared by software reading the result in the DR register or writing to the CSR register.
0: Conversion is not complete
1: Conversion can be read from the DR register

Bit 6 = **Reserved.** *must always be cleared.*

Bit 5 = **ADON** *A/D Converter On*
This bit is set and cleared by software.
0: A/D converter is switched off
1: A/D converter is switched on

Bit 4 = **Reserved.** *must always be cleared.*

Bits 3:0 = **CH[3:0]** *Channel Selection*
These bits are set and cleared by software. They select the analog input to convert.

Channel Pin*	CH3	CH2	CH1	CH0
AIN0	0	0	0	0
AIN1	0	0	0	1
AIN2	0	0	1	0
AIN3	0	0	1	1
AIN4	0	1	0	0
AIN5	0	1	0	1
AIN6	0	1	1	0
AIN7	0	1	1	1
AIN8	1	0	0	0
AIN9	1	0	0	1
AIN10	1	0	1	0
AIN11	1	0	1	1
AIN12	1	1	0	0
AIN13	1	1	0	1
AIN14	1	1	1	0
AIN15	1	1	1	1

***Note**: The number of pins AND the channel selection varies according to the device. Refer to the device pinout.

DATA REGISTER (DR)

Read Only

Reset Value: 0000 0000 (00h)

7							0
D7	D6	D5	D4	D3	D2	D1	D0

Bits 7:0 = **D[7:0]** *Analog Converted Value*
This register contains the converted analog value in the range 00h to FFh.

Note: Reading this register reset the COCO flag.

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

Table 34. ADC Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0012h	**ADCDR**	D7	D6	D5	D4	D3	D2	D1	D0
	Reset Value	0	0	0	0	0	0	0	0
0013h	**ADCCSR**	COCO		ADON		CH3	CH2	CH1	CH0
	Reset Value	0	0	0	0	0	0	0	0

12 INSTRUCTION SET

12.1 ST7 ADDRESSING MODES

The ST7 Core features 17 different addressing modes which can be classified in 7 main groups:

Addressing Mode	Example
Inherent	nop
Immediate	ld A,#$55
Direct	ld A,$55
Indexed	ld A,($55,X)
Indirect	ld A,([$55],X)
Relative	jrne loop
Bit operation	bset byte,#5

The ST7 Instruction set is designed to minimize the number of bytes required per instruction: To do so, most of the addressing modes may be subdivided in two sub-modes called long and short:

– Long addressing mode is more powerful because it can use the full 64 Kbyte address space, however it uses more bytes and more CPU cycles.

– Short addressing mode is less powerful because it can generally only access page zero (0000h - 00FFh range), but the instruction size is more compact, and faster. All memory to memory instructions use short addressing modes only (CLR, CPL, NEG, BSET, BRES, BTJT, BTJF, INC, DEC, RLC, RRC, SLL, SRL, SRA, SWAP)

The ST7 Assembler optimizes the use of long and short addressing modes.

Table 35. ST7 Addressing Mode Overview

Mode			Syntax	Destination	Pointer Address (Hex.)	Pointer Size (Hex.)	Length (Bytes)
Inherent			nop				+ 0
Immediate			ld A,#$55				+ 1
Short	Direct		ld A,$10	00..FF			+ 1
Long	Direct		ld A,$1000	0000..FFFF			+ 2
No Offset	Direct	Indexed	ld A,(X)	00..FF			+ 0
Short	Direct	Indexed	ld A,($10,X)	00..1FE			+ 1
Long	Direct	Indexed	ld A,($1000,X)	0000..FFFF			+ 2
Short	Indirect		ld A,[$10]	00..FF	00..FF	byte	+ 2
Long	Indirect		ld A,[$10.w]	0000..FFFF	00..FF	word	+ 2
Short	Indirect	Indexed	ld A,([$10],X)	00..1FE	00..FF	byte	+ 2
Long	Indirect	Indexed	ld A,([$10.w],X)	0000..FFFF	00..FF	word	+ 2
Relative	Direct		jrne loop	PC+/-127			+ 1
Relative	Indirect		jrne [$10]	PC+/-127	00..FF	byte	+ 2
Bit	Direct		bset $10,#7	00..FF			+ 1
Bit	Indirect		bset [$10],#7	00..FF	00..FF	byte	+ 2
Bit	Direct	Relative	btjt $10,#7,skip	00..FF			+ 2
Bit	Indirect	Relative	btjt [$10],#7,skip	00..FF	00..FF	byte	+ 3

INSTRUCTION SET OVERVIEW (Cont'd)

12.1.1 Inherent

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

Inherent Instruction	Function
NOP	No operation
TRAP	S/W Interrupt
WFI	Wait For Interrupt (Low Power Mode)
HALT	Halt Oscillator (Lowest Power Mode)
RET	Sub-routine Return
IRET	Interrupt Sub-routine Return
SIM	Set Interrupt Mask (level 3)
RIM	Reset Interrupt Mask (level 0)
SCF	Set Carry Flag
RCF	Reset Carry Flag
RSP	Reset Stack Pointer
LD	Load
CLR	Clear
PUSH/POP	Push/Pop to/from the stack
INC/DEC	Increment/Decrement
TNZ	Test Negative or Zero
CPL, NEG	1 or 2 Complement
MUL	Byte Multiplication
SLL, SRL, SRA, RLC, RRC	Shift and Rotate Operations
SWAP	Swap Nibbles

12.1.2 Immediate

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

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

12.1.3 Direct

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

The direct addressing mode consists of two submodes:

Direct (short)

The address is a byte, thus requires only one byte after the opcode, but only allows 00 - FF addressing space.

Direct (long)

The address is a word, thus allowing 64 Kbyte addressing space, but requires 2 bytes after the opcode.

12.1.4 Indexed (No Offset, Short, Long)

In this mode, the operand is referenced by its memory address, which is defined by the unsigned addition of an index register (X or Y) with an offset.

The indirect addressing mode consists of three sub-modes:

Indexed (No Offset)

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

Indexed (Short)

The offset is a byte, thus requires only one byte after the opcode and allows 00 - 1FE addressing space.

Indexed (long)

The offset is a word, thus allowing 64 Kbyte addressing space and requires 2 bytes after the opcode.

12.1.5 Indirect (Short, Long)

The required data byte to do the operation is found by its memory address, located in memory (pointer).

The pointer address follows the opcode. The indirect addressing mode consists of two sub-modes:

Indirect (short)

The pointer address is a byte, the pointer size is a byte, thus allowing 00 - FF addressing space, and requires 1 byte after the opcode.

Indirect (long)

The pointer address is a byte, the pointer size is a word, thus allowing 64 Kbyte addressing space, and requires 1 byte after the opcode.

INSTRUCTION SET OVERVIEW (Cont'd)

12.1.6 Indirect Indexed (Short, Long)

This is a combination of indirect and short indexed addressing modes. The operand is referenced by its memory address, which is defined by the unsigned addition of an index register value (X or Y) with a pointer value located in memory. The pointer address follows the opcode.

The indirect indexed addressing mode consists of two sub-modes:

Indirect Indexed (Short)

The pointer address is a byte, the pointer size is a byte, thus allowing 00 - 1FE addressing space, and requires 1 byte after the opcode.

Indirect Indexed (Long)

The pointer address is a byte, the pointer size is a word, thus allowing 64 Kbyte addressing space, and requires 1 byte after the opcode.

Table 36. Instructions Supporting Direct, Indexed, Indirect and Indirect Indexed Addressing Modes

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

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

12.1.7 Relative mode (Direct, Indirect)

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

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

The relative addressing mode consists of two sub-modes:

Relative (Direct)

The offset is following the opcode.

Relative (Indirect)

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

INSTRUCTION SET OVERVIEW (Cont'd)

12.2 INSTRUCTION GROUPS

The ST7 family devices use an Instruction Set consisting of 63 instructions. The instructions may be subdivided into 13 main groups as illustrated in the following table:

Load and Transfer	LD	CLR						
Stack operation	PUSH	POP	RSP					
Increment/Decrement	INC	DEC						
Compare and Tests	CP	TNZ	BCP					
Logical operations	AND	OR	XOR	CPL	NEG			
Bit Operation	BSET	BRES						
Conditional Bit Test and Branch	BTJT	BTJF						
Arithmetic operations	ADC	ADD	SUB	SBC	MUL			
Shift and Rotates	SLL	SRL	SRA	RLC	RRC	SWAP	SLA	
Unconditional Jump or Call	JRA	JRT	JRF	JP	CALL	CALLR	NOP	RET
Conditional Branch	JRxx							
Interruption management	TRAP	WFI	HALT	IRET				
Condition Code Flag modification	SIM	RIM	SCF	RCF				

Using a pre-byte

The instructions are described with one to four opcodes.

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

The whole instruction becomes:

PC-2 End of previous instruction

PC-1 Prebyte

PC opcode

PC+1 Additional word (0 to 2) according to the number of bytes required to compute the effective address

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

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

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

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

INSTRUCTION SET OVERVIEW (Cont'd)

Mnemo	Description	Function/Example	Dst	Src	I1	H	I0	N	Z	C
ADC	Add with Carry	A = A + M + C	A	M		H		N	Z	C
ADD	Addition	A = A + M	A	M		H		N	Z	C
AND	Logical And	A = A . M	A	M				N	Z	
BCP	Bit compare A, Memory	tst (A . M)	A	M				N	Z	
BRES	Bit Reset	bres Byte, #3	M							
BSET	Bit Set	bset Byte, #3	M							
BTJF	Jump if bit is false (0)	btjf Byte, #3, Jmp1	M							C
BTJT	Jump if bit is true (1)	btjt Byte, #3, Jmp1	M							C
CALL	Call subroutine									
CALLR	Call subroutine relative									
CLR	Clear		reg, M					0	1	
CP	Arithmetic Compare	tst(Reg - M)	reg	M				N	Z	C
CPL	One Complement	A = FFH-A	reg, M					N	Z	1
DEC	Decrement	dec Y	reg, M					N	Z	
HALT	Halt				1		0			
IRET	Interrupt routine return	Pop CC, A, X, PC			I1	H	I0	N	Z	C
INC	Increment	inc X	reg, M					N	Z	
JP	Absolute Jump	jp [TBL.w]								
JRA	Jump relative always									
JRT	Jump relative									
JRF	Never jump	jrf *								
JRIH	Jump if Port B INT pin = 1	(no Port B Interrupts)								
JRIL	Jump if Port B INT pin = 0	(Port B interrupt)								
JRH	Jump if H = 1	H = 1 ?								
JRNH	Jump if H = 0	H = 0 ?								
JRM	Jump if I1:0 = 11	I1:0 = 11 ?								
JRNM	Jump if I1:0 <> 11	I1:0 <> 11 ?								
JRMI	Jump if N = 1 (minus)	N = 1 ?								
JRPL	Jump if N = 0 (plus)	N = 0 ?								
JREQ	Jump if Z = 1 (equal)	Z = 1 ?								
JRNE	Jump if Z = 0 (not equal)	Z = 0 ?								
JRC	Jump if C = 1	C = 1 ?								
JRNC	Jump if C = 0	C = 0 ?								
JRULT	Jump if C = 1	Unsigned <								
JRUGE	Jump if C = 0	Jmp if unsigned >=								
JRUGT	Jump if (C + Z = 0)	Unsigned >								

INSTRUCTION SET OVERVIEW (Cont'd)

Mnemo	Description	Function/Example	Dst	Src	I1	H	I0	N	Z	C
JRULE	Jump if (C + Z = 1)	Unsigned <=								
LD	Load	dst <= src	reg, M	M, reg				N	Z	
MUL	Multiply	X,A = X * A	A, X, Y	X, Y, A		0				0
NEG	Negate (2's compl)	neg $10	reg, M					N	Z	C
NOP	No Operation									
OR	OR operation	A = A + M	A	M				N	Z	
POP	Pop from the Stack	pop reg	reg	M						
		pop CC	CC	M	I1	H	I0	N	Z	C
PUSH	Push onto the Stack	push Y	M	reg, CC						
RCF	Reset carry flag	C = 0								0
RET	Subroutine Return									
RIM	Enable Interrupts	I1:0 = 10 (level 0)			1		0			
RLC	Rotate left true C	C <= A <= C	reg, M					N	Z	C
RRC	Rotate right true C	C => A => C	reg, M					N	Z	C
RSP	Reset Stack Pointer	S = Max allowed								
SBC	Substract with Carry	A = A - M - C	A	M				N	Z	C
SCF	Set carry flag	C = 1								1
SIM	Disable Interrupts	I1:0 = 11 (level 3)			1		1			
SLA	Shift left Arithmetic	C <= A <= 0	reg, M					N	Z	C
SLL	Shift left Logic	C <= A <= 0	reg, M					N	Z	C
SRL	Shift right Logic	0 => A => C	reg, M					0	Z	C
SRA	Shift right Arithmetic	A7 => A => C	reg, M					N	Z	C
SUB	Substraction	A = A - M	A	M				N	Z	C
SWAP	SWAP nibbles	A7-A4 <=> A3-A0	reg, M					N	Z	
TNZ	Test for Neg & Zero	tnz lbl1						N	Z	
TRAP	S/W trap	S/W interrupt			1		1			
WFI	Wait for Interrupt				1		0			
XOR	Exclusive OR	A = A XOR M	A	M				N	Z	

13 ELECTRICAL CHARACTERISTICS

13.1 PARAMETER CONDITIONS

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

13.1.1 Minimum and Maximum Values

Unless otherwise specified the minimum and maximum values are guaranteed in the worst conditions of ambient temperature, supply voltage and frequencies by tests in production on 100% of the devices with an ambient temperature at T_A=25°C and T_A=$T_A$max (given by the selected temperature range).

Data based on characterization results, design simulation and/or technology characteristics are indicated in the table footnotes and are not tested in production. Based on characterization, the minimum and maximum values refer to sample tests and represent the mean value plus or minus three times the standard deviation (mean±3Σ).

13.1.2 Typical Values

Unless otherwise specified, typical data are based on T_A=25°C, V_{DD}=5V (for the 4.5V≤V_{DD}≤5.5V voltage range) and V_{DD}=3.3V (for the 3V≤V_{DD}≤4V voltage range). They are given only as design guidelines and are not tested.

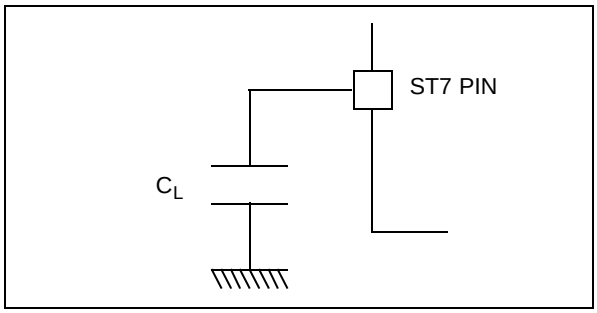
13.1.3 Typical Curves

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

13.1.4 Loading Capacitor

The loading conditions used for pin parameter measurement is shown in Figure 70.

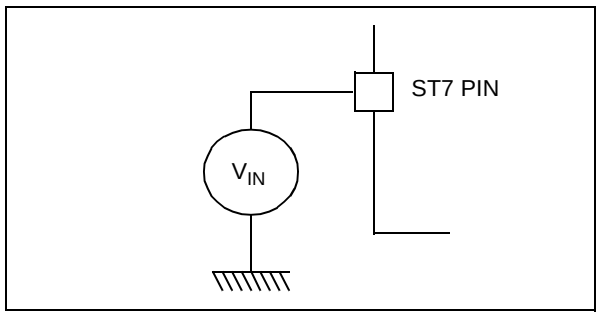
Figure 70. Pin Loading Conditions



13.1.5 Pin input Voltage

The input voltage measurement on a pin of the device is described in Figure 71.

Figure 71. Pin input Voltage



13.2 ABSOLUTE MAXIMUM RATINGS

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

13.2.1 Voltage Characteristics

Symbol	Ratings	Maximum value	Unit
V_{DD} - V_{SS}	Supply voltage	6.0	V
V_{IN} [1) & 2)]	Input voltage on any pin	V_{SS}-0.3 to V_{DD}+0.3	
$V_{ESD(HBM)}$	Electro-static discharge voltage (Human Body Model)	1500	

13.2.2 Current Characteristics

Symbol	Ratings	Maximum value	Unit
I_{VDD}	Total current into V_{DD} power lines (source) [3)]	100	mA
I_{VSS}	Total current out of V_{SS} ground lines (sink) [3)]	80	
I_{IO}	Output current sunk by any standard I/O and control pin	25	
	Output current sunk by any high sink I/O pin	50	
	Output current source by any I/Os and control pin	- 25	
$I_{INJ(PIN)}$ [2) & 4)]	Injected current on V_{PP} pin	± 5	
	Injected current on $\overline{\text{RESET}}$ pin	± 5	
	Injected current on OSC1 and OSC2 pins	± 5	
	Injected current on any other pin [5) & 6)]	± 5	
$\Sigma I_{INJ(PIN)}$ [2)]	Total injected current (sum of all I/O and control pins) [5)]	± 20	

13.2.3 Thermal Characteristics

Symbol	Ratings	Value	Unit
T_{STG}	Storage temperature range	-65 to +150	°C
T_J	Maximum junction temperature	TBD	°C

Notes:

1. Directly connecting the $\overline{\text{RESET}}$ and I/O pins to V_{DD} or V_{SS} could damage the device if an unintentional internal reset is generated or an unexpected change of the I/O configuration occurs (for example, due to a corrupted program counter). To guarantee safe operation, this connection has to be done through a pull-up or pull-down resistor (typical: 4.7kΩ for $\overline{\text{RESET}}$, 10kΩ for I/Os). Unused I/O pins must be tied in the same way to V_{DD} or V_{SS} according to their reset configuration.

2. 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 $V_{IN}>V_{DD}$ while a negative injection is induced by $V_{IN}<V_{SS}$.

3. All power (V_{DD}) and ground (V_{SS}) lines must always be connected to the external supply.

4. Negative injection disturbs the analog performance of the device. In particular, it induces leakage currents throughout the device including the analog inputs. To avoid undesirable effects on the analog functions, care must be taken:
- Analog input pins must have a negative injection less than 0.8 mA (assuming that the impedance of the analog voltage is lower than the specified limits)
- Pure digital pins must have a negative injection less than 1.6mA. In addition, it is recommended to inject the current as far as possible from the analog input pins.

5. When several inputs are submitted to a current injection, the maximum $\Sigma I_{INJ(PIN)}$ is the absolute sum of the positive and negative injected currents (instantaneous values). These results are based on characterisation with $\Sigma I_{INJ(PIN)}$ maximum current injection on four I/O port pins of the device.

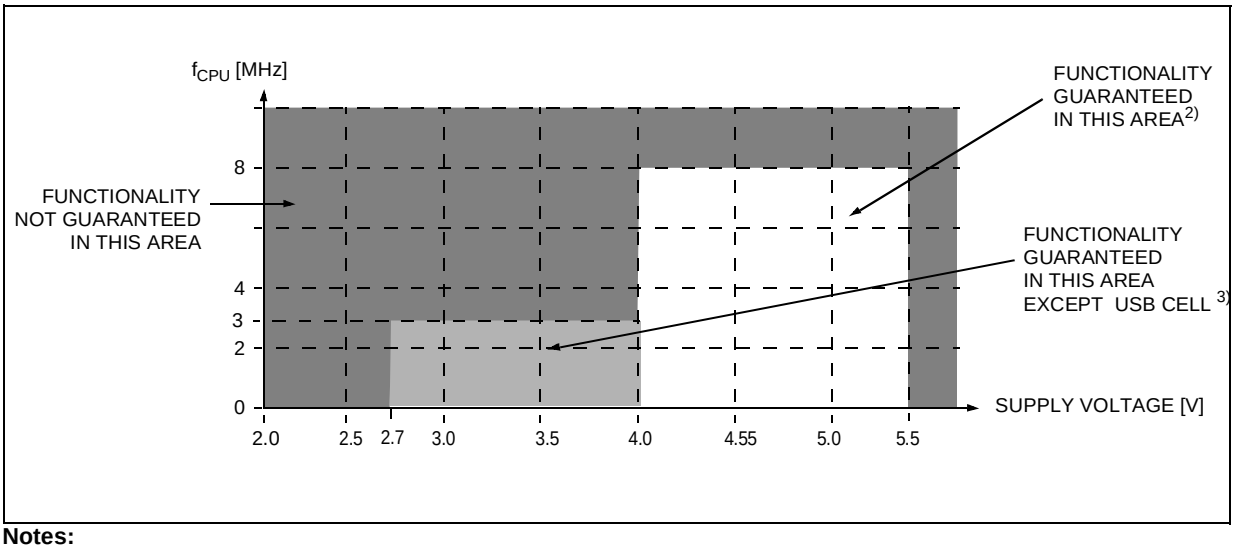
6. True open drain I/O port pins do not accept positive injection.

13.3 OPERATING CONDITIONS

13.3.1 General Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
V_{DD}	Supply voltage with USB peripheral enabled	see Figure 72	4.0	5.5	V
	Supply voltage with USB peripheral disabled and LVD off	see Figure 72	2.7	5.5	V
f_{OSC}	External clock frequency		12	12	MHz
T_A	Ambient temperature range		0	70	°C

Figure 72. f_{OSC} Maximum Operating Frequency Versus V_{DD} Supply Voltage 1)



Notes:

A/D operation not guaranteed below 1MHz.

1. Operating conditions with T_A=0 to +70°C.
2. This mode is supported by all devices.
3. This mode is only supported by ST72(F)651AR6T1E or ST72(F)651R6T1E devices (without LVD)

OPERATING CONDITIONS (Cont'd)

13.3.2 Operating Conditions with Low Voltage Detector (LVD)

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

Symbol	Parameter	Conditions	Min	Typ [1]	Max	Unit
V_{IT+}	Reset release threshold (V_{DD} rise)		2.9	3.5	3.8	V
V_{IT-}	Reset generation threshold (V_{DD} fall)		2.6	3.1	3.5	
V_{hys}	LVD voltage threshold hysteresis	V_{IT+}-V_{IT-}	150	300		mV
f_{CUTOFF}	LVD filter cut-off frequency [2]	Not detected by the LVD		10		MHz.
Vt_{POR}	V_{DD} rise time [3]		0.3		10	ms

Notes:

1. LVD typical data are based on T_A=25°C. They are given only as design guidelines and are not tested.
2. Not tested, guaranteed by construction.
3. The V_{DD} rise time condition is needed to insure a correct device power-on and LVD reset. Not tested in production.

13.3.3 Power Supply Manager Characteristics

Subject to general operating conditions for V_{DD}, f_{OSC}, and T_A. Not guaranteed on LVD devices (without E suffix).

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$USBV_{IT+}$	Reset release threshold (V_{DD} rise)		3.50	3.80	4.00	V
$USBV_{IT-}$	Reset generation threshold (V_{DD} fall)		3.30	3.65	3.80	
$USBV_{hys}$	USB voltage threshold hysteresis	$USBV_{IT+}$-$USBV_{IT-}$	100	200	300	mV
$V_{PLLmin48}$	Minimum voltage required for stable 48MHz PLL operation (PLL locked)		3.7 [1]			V
$V_{PLLmin40}$	Minimum voltage required for 40MHz PLL operation (PLL un-locked)		3.4 [1]			V
$V_{PLLmin24}$	Minimum voltage required for 24MHz PLL operation (PLL un-locked)		3.0 [1]			V

1. Not tested, guaranteed by construction.

13.3.4 Storage Device Supply Characteristics

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

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DDF}	Voltage output for external storage device (I_{load} max = 50mA)	USB Mode: VSET[1:0]=11	2.5	2.8	3.2	V
		10	2.9	3.3	3.6	
		01	3.0	3.4	3.8	
		00	3.1	3.5	3.9	

Note: In Stand-alone mode V_{DDF} must be connected to V_{DD}

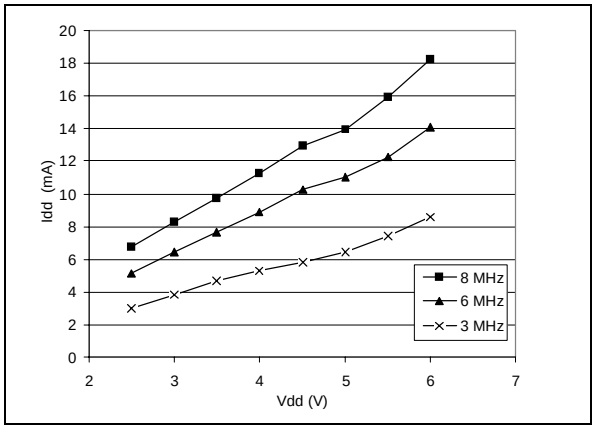
13.4 SUPPLY CURRENT CHARACTERISTICS

The following current consumption specified for the ST7 functional operating modes over temperature range does not take into account the clock source current consumption. To get the total device consumption, the two current values must be added (except for HALT mode for which the clock is stopped).

13.4.1 RUN Mode

Symbol	Parameter	Conditions		Typ [1]	Max [2]	Unit
I_{DD}	Supply current in RUN mode [3] (see Figure 73)	$4.0V \le V_{DD} \le 5.5V$	f_{CPU}=8MHz	14	20	mA
	Supply current in RUN mode [3] (see Figure 73)	$2.7V \le V_{DD} \le 4.0V$	f_{CPU}=3MHz	4	8	

Figure 73. Typical I_{DD} in RUN vs. f_{CPU}



Notes:

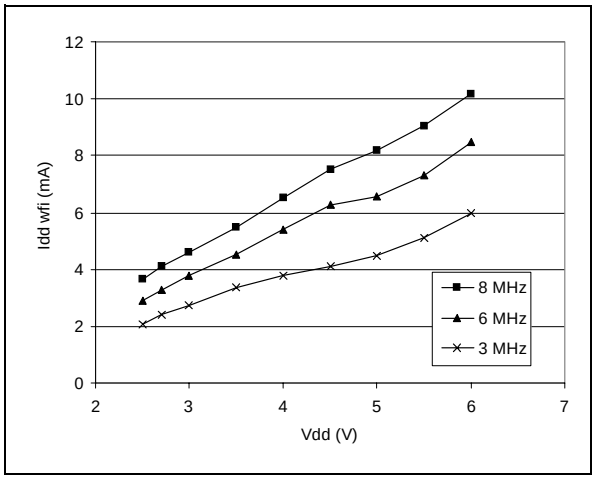
1. Typical data are based on T_A=25°C, V_{DD}=5V ($4.0V \le V_{DD} \le 5.5V$ range) and V_{DD}=3.3V ($2.7V \le V_{DD} \le 4.0V$ range).
2. Data based on characterization results, tested in production at V_{DD} =5.5V. and f_{CPU} = 8MHz
3. CPU running with memory access, all I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (OSC1) driven by external square wave, LVD disabled.

SUPPLY CURRENT CHARACTERISTICS (Cont'd)

13.4.2 WAIT Mode

Symbol	Parameter	Conditions		Typ [1]	Max [2]	Unit
I_{WFI}	Supply current in WAIT mode [3] (see Figure 74)	$4.0V \le V_{DD} \le 5.5V$	f_{CPU}=8MHz	8	11	mA
	Supply current in WAIT mode [3] (see Figure 74)	$2.7V \le V_{DD} \le 4.0V$	f_{CPU}=3MHz	3	6	

Figure 74. Typical I_{DD} in WAIT vs. f_{CPU}



Notes:

1. Typical data are based on T_A=25°C, V_{DD}=5V ($4V \le V_{DD} \le 5.5V$ range) and V_{DD}=3.3V ($2.7V \le V_{DD} \le 4.0V$ range).
2. Data based on characterization results, tested in production at V_{DD} = 5.5V and f_{CPU} = 8MHz.
3. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (OSC1) driven by external square wave, LVD disabled.

SUPPLY CURRENT CHARACTERISTICS (Cont'd)

13.4.3 HALT Mode

Symbol	Parameter	Conditions		Typ [1]	Max	Unit
I_{HALT}	Supply current in HALT mode [2]	LVD OFF	V_{DD}=5.5V	3	TBD	µA
			V_{DD}=3.0V	1	TBD	
		LVD ON	V_{DD}=5.5V	110	TBD	
			V_{DD}=3.0V	60	TBD	

Notes:

1. Typical data are based on T_A=25°C.
2. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load).

13.4.4 SUSPEND Mode

Symbol	Parameter	Conditions		Typ [1]	Max [3]	Unit
I_{SUSP}	Supply current in SUSPEND mode [2]	LVD OFF	V_{DD}=4-5.25V	150	230	µA
		LVD ON	V_{DD}=4-5.25V	230	300	

Notes:

1. Typical data are based on T_A=25°C.
2. External pull-up (1.5kΩ connected to $USBV_{CC}$) and pull-down (15kΩ connected to $USBV_{SS}$) current not included.
3. T_A=25°C

13.5 CLOCK AND TIMING CHARACTERISTICS

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

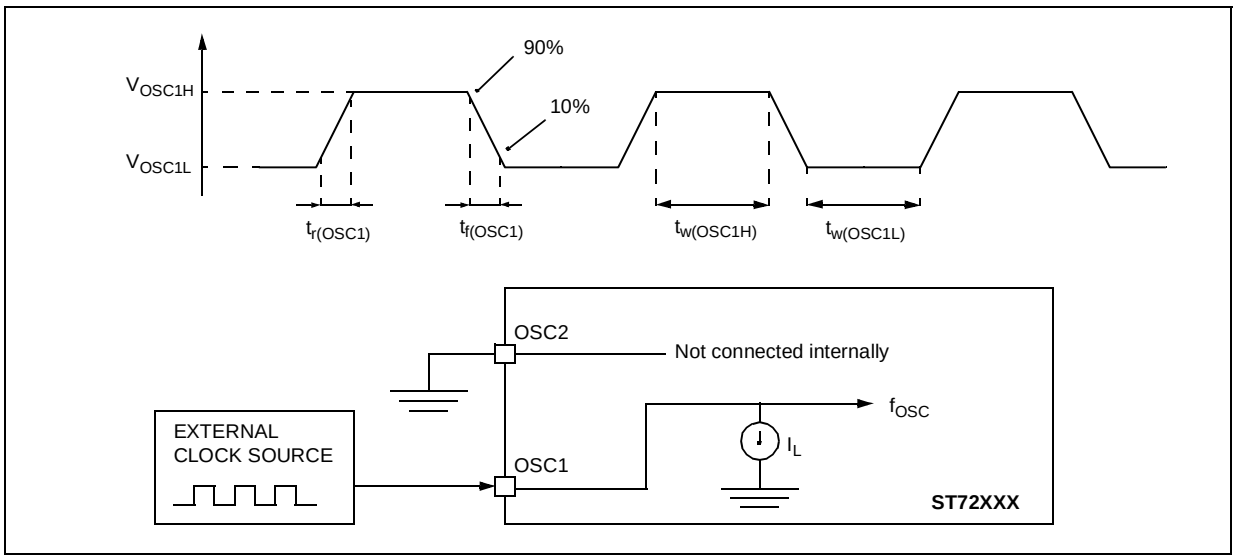
13.5.1 General Timings

Symbol	Parameter	Conditions	Min	Typ [1]	Max	Unit
$t_{c(INST)}$	Instruction cycle time		2	4	12	t_{CPU}
		f_{CPU}=8MHz	250	500	1500	ns
$t_{v(IT)}$	Interrupt reaction time [2] $t_{v(IT)} = \Delta t_{c(INST)} + 10$		10		22	t_{CPU}
		f_{CPU}=8MHz	1.25		2.75	µs

13.5.2 External Clock Source

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{OSC1H}	OSC1 input pin high level voltage		$0.7xV_{DD}$		V_{DD}	V
V_{OSC1L}	OSC1 input pin low level voltage		V_{SS}		$0.3xV_{DD}$	
$t_{w(OSC1H)}$ $t_{w(OSC1L)}$	OSC1 high or low time [3]		15			ns
$t_{r(OSC1)}$ $t_{f(OSC1)}$	OSC1 rise or fall time [3]				15	
I_L	OSCx Input leakage current	$V_{SS} \le V_{IN} \le V_{DD}$			±1	µA

Figure 75. Typical Application with an External Clock Source



Notes:

1. Data based on typical application software. Not tested in production.
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.
3. Data based on design simulation and/or technology characteristics, not tested in production.

13.6 MEMORY CHARACTERISTICS

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

13.6.1 RAM and Hardware Registers

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{RM}	Data retention mode [1]	HALT mode (or RESET)	2			V

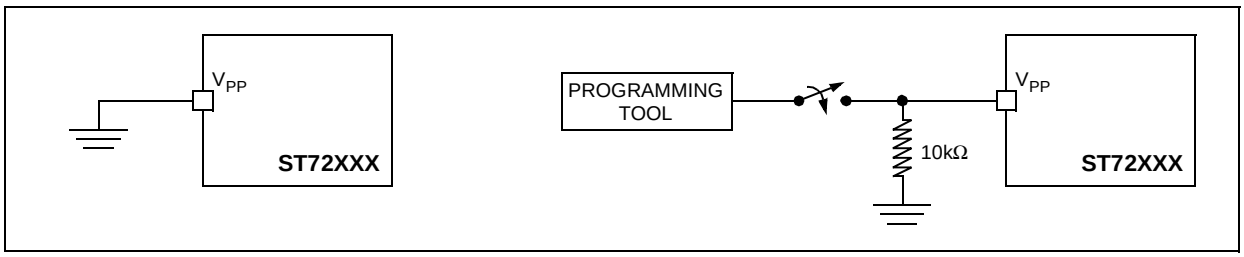
13.6.2 FLASH Memory

Operating Conditions: f_{CPU} = 8 MHz.

DUAL VOLTAGE FLASH MEMORY						
Symbol	**Parameter**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
f_{CPU}	Operating Frequency	Read mode			8	MHz
		Write / Erase mode, T_A=25°C			8	
V_{PP}	Programming Voltage	4.0V <= V_{DD} <= 5.5V	11.4		12.6	V
I_{PP}	V_{PP} Current	Write / Erase			30 [1]	mA
t_{PROG}	Byte Programming Time	T_A=25°C		100	500 [1]	µs
t_{ERASE}	Sector Erasing Time			2	10 [1]	sec
	Device Erasing Time			5	10 [1]	
t_{VPP}	Internal V_{PP} Stabilization Time			10		µs
t_{RET}	Data Retention	$T_A \leq$ 55°C	20			years
N_{RW}	Write Erase Cycles	T_A=25°C	100			cycles

Note 1: Guaranteed by Design.

Figure 76. Two typical Applications with V_{PP} Pin[1]



Note 1: When the ICP mode is not required by the application, V_{PP} pin must be tied to V_{SS}.

13.7 EMC CHARACTERISTICS

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

13.7.1 Functional EMS

(Electro Magnetic Susceptibility)

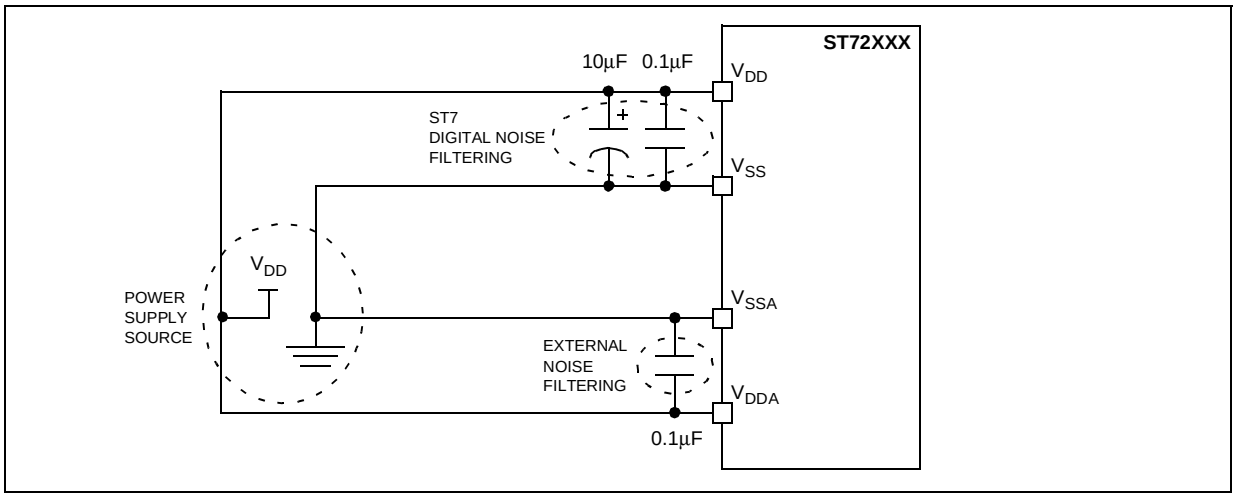
Based on a simple running application on the product (toggling 2 LEDs through I/O ports), the product is stressed by two electro magnetic events until a failure occurs (indicated by the LEDs).

- **ESD**: Electro-Static Discharge (positive and negative) is applied on all pins of the device until a functional disturbance occurs. This test conforms with the IEC 1000-4-2 standard.
- **FTB**: Burst of Fast Transient voltage (positive and negative) is applied to V_{DD} and V_{SS} through a 100pF capacitor, until a functional disturbance occurs. This test conforms with the IEC 1000-4-4 standard.

A device reset allows normal operations to be resumed.

Symbol	Parameter	Conditions	Neg [1]	Pos [1]	Unit
V_{FESD}	Voltage limits to be applied on any I/O pin to induce a functional disturbance	V_{DD}=5V, T_A=+25°C, f_{OSC}=8MHz conforms to IEC 1000-4-2	-1	1	kV
V_{FFTB}	Fast transient voltage burst limits to be applied through 100pF on V_{DD} and V_{DDA} pins to induce a functional disturbance	V_{DD}=5V, T_A=+25°C, f_{OSC}=8MHz conforms to IEC 1000-4-4	-4	4	

Figure 77. EMC Recommended star network power supply connection [2]



Notes:

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

2. The suggested 10µF and 0.1µF decoupling capacitors on the power supply lines are proposed as a good price vs. EMC performance tradeoff. They have to be put as close as possible to the device power supply pins. Other EMC recommendations are given in other sections (I/Os, RESET, OSCx pin characteristics).

EMC CHARACTERISTICS (Cont'd)

13.7.2 Absolute 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 AN1181 ST7 application note.

13.7.2.1 Electro-Static Discharge (ESD)

Electro-Static Discharges (3 positive then 3 negative pulses 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 of the device (3 parts*(n+1) supply pin). One model is simulated: Human Body Model. This test conforms to the JESD22-A114A standard. See Figure 78 and the following test sequence.

Human Body Model Test Sequence

– C_L is loaded through S1 by the HV pulse generator.

– S1 switches position from generator to R.

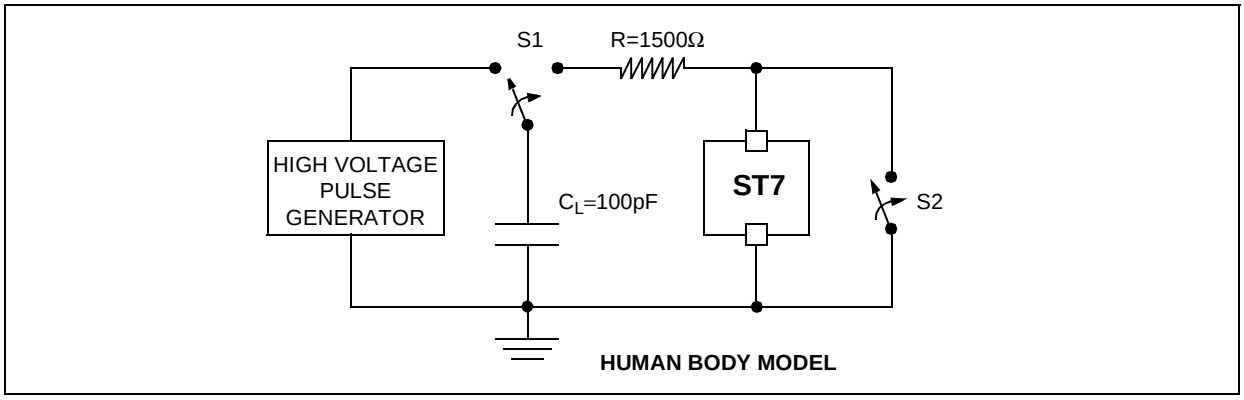
– A discharge from C_L through R (body resistance) to the ST7 occurs.

– S2 must be closed 10 to 100ms after the pulse delivery period to ensure the ST7 is not left in charge state. S2 must be opened at least 10ms prior to the delivery of the next pulse.

Absolute Maximum Ratings

Symbol	Ratings	Conditions	Maximum value [1]	Unit
$V_{ESD(HBM)}$	Electro-static discharge voltage (Human Body Model)	T_A=+25°C	1500	V

Figure 78. Typical equivalent ESD Circuit



Notes:

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

EMC CHARACTERISTICS (Cont'd)

13.7.2.2 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), a current injection (applied to each input, output and configurable I/O pin) and a power supply switch sequence are performed on each sample. This test conforms to the EIA/JESD 78 IC latch-up standard. For more details, refer to the AN1181 ST7 application note.
- **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 and is described in Figure 79. For more details, refer to the AN1181 ST7 application note.

13.7.2.3 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:

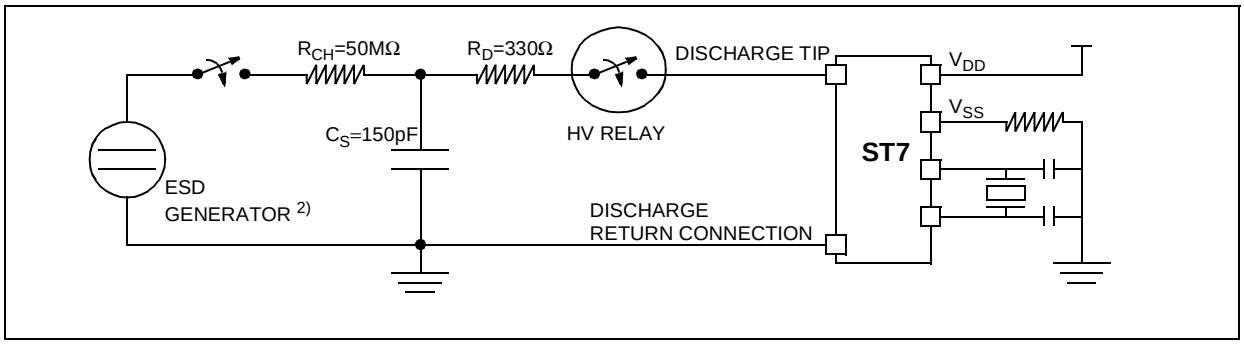
Most of the common failures (unexpected reset and program counter corruption) can be reproduced by manually forcing a low state on the RESET pin or the Oscillator pins for 1 second.

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).

Electrical Sensitivities

Symbol	Parameter	Conditions	Class [1]
LU	Static latch-up class	T_A=+25°C T_A=+85°C T_A=+125°C	A A A
DLU	Dynamic latch-up class	V_{DD}=5.5V, f_{OSC}=4MHz, T_A=+25°C	A

Figure 79. Simplified Diagram of the ESD Generator for DLU



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).

2. Schaffner NSG435 with a pointed test finger.

EMC CHARACTERISTICS (Cont'd)

13.7.3 ESD Pin Protection Strategy

To protect an integrated circuit against Electro-Static Discharge the stress must be controlled to prevent degradation or destruction of the circuit elements. The stress generally affects the circuit elements which are connected to the pads but can also affect the internal devices when the supply pads receive the stress. The elements to be protected must not receive excessive current, voltage or heating within their structure.

An ESD network combines the different input and output ESD protections. This network works, by allowing safe discharge paths for the pins subjected to ESD stress. Two critical ESD stress cases are presented in Figure 80 and Figure 81 for standard pins and in Figure 82 and Figure 83 for true open drain pins.

Standard Pin Protection

To protect the output structure the following elements are added:

– A diode to V_{DD} (3a) and a diode from V_{SS} (3b)
– A protection device between V_{DD} and V_{SS} (4)

To protect the input structure the following elements are added:

– A resistor in series with the pad (1)
– A diode to V_{DD} (2a) and a diode from V_{SS} (2b)
– A protection device between V_{DD} and V_{SS} (4)

Figure 80. Positive Stress on a Standard Pad vs. V_{SS}

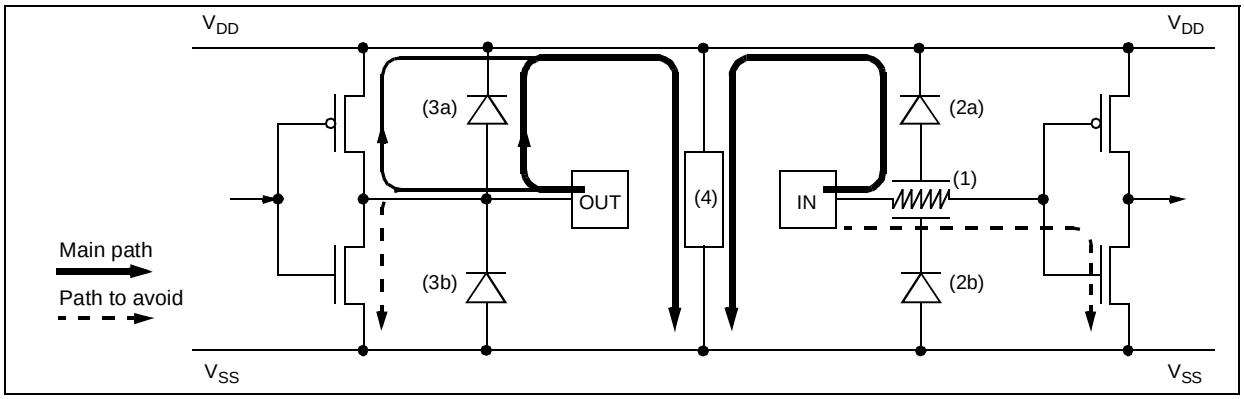
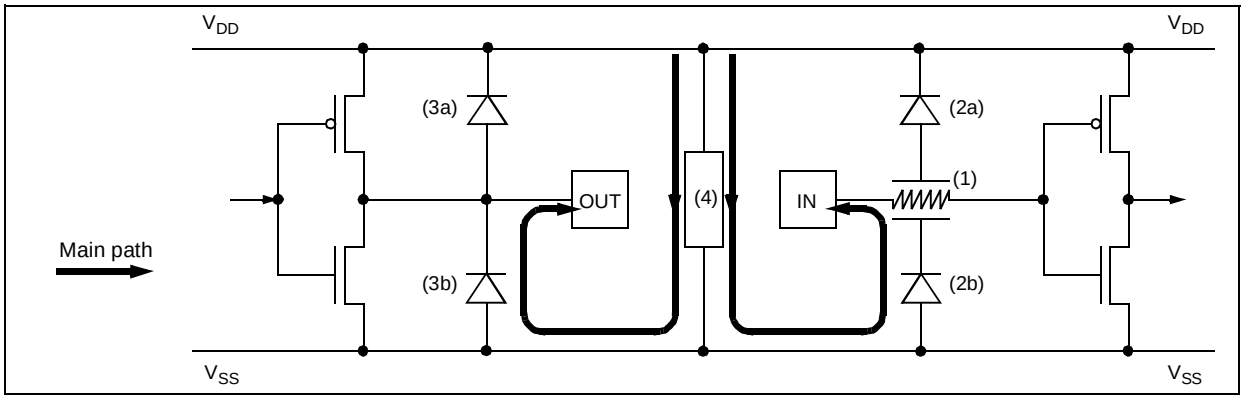


Figure 81. Negative Stress on a Standard Pad vs. V_{DD}



EMC CHARACTERISTICS (Cont'd)

True Open Drain Pin Protection

The centralized protection (4) is not involved in the discharge of the ESD stresses applied to true open drain pads due to the fact that a P-Buffer and diode to V_{DD} are not implemented. An additional local protection between the pad and V_{SS} (5a & 5b) is implemented to completely absorb the positive ESD discharge.

Multisupply Configuration

When several types of ground (V_{SS}, V_{SSA}, ...) and power supply (V_{DD}, V_{DDA}, ...) are available for any reason (better noise immunity...), the structure shown in Figure 84 is implemented to protect the device against ESD.

Figure 82. Positive Stress on a True Open Drain Pad vs. V_{SS}

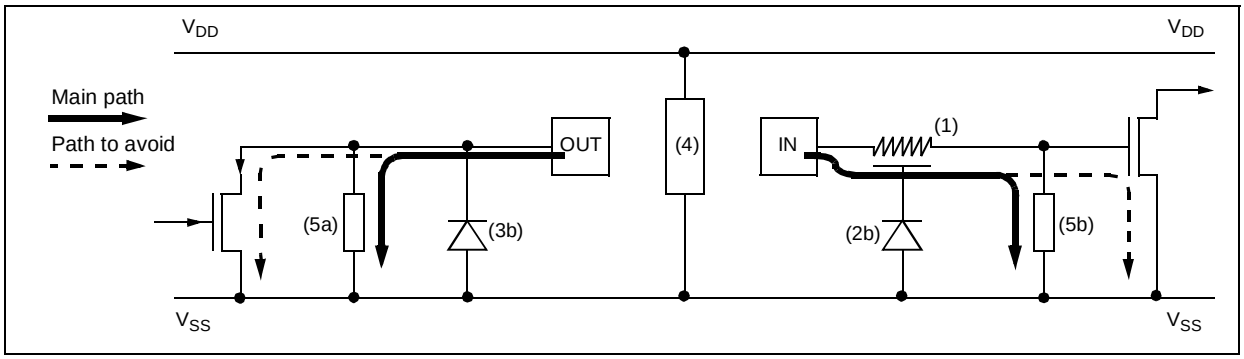


Figure 83. Negative Stress on a True Open Drain Pad vs. V_{DD}

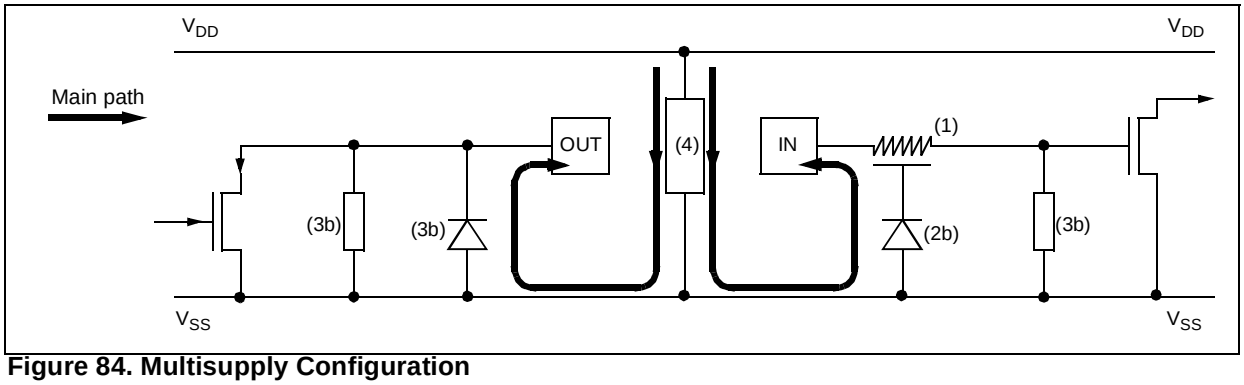
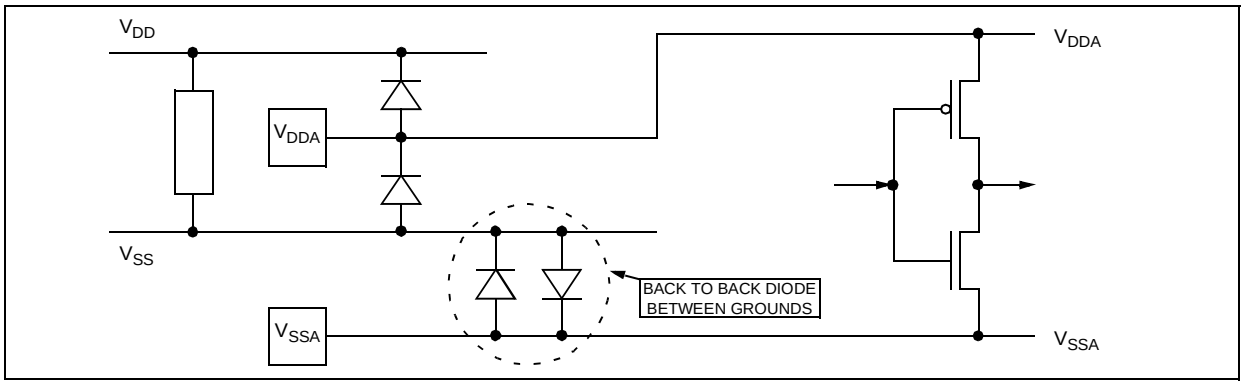


Figure 84. Multisupply Configuration



13.8 I/O PORT PIN CHARACTERISTICS

13.8.1 General Characteristics

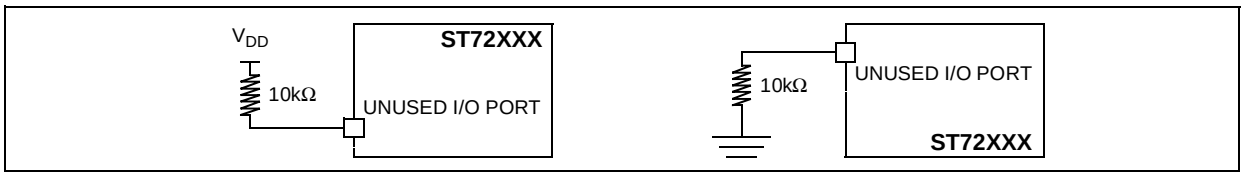
Subject to general operating conditions for V_{DD}, f_{OSC}, and T_A unless otherwise specified.

Symbol	Parameter	Conditions		Min	Typ [1]	Max	Unit
V_{IL}	Input low level voltage	V_{DD} = 5.0V		V_{SS}		$0.3xV_{DD}$	V
V_{IH}	Input high level voltage	V_{DD} = 5.0V		$0.7xV_{DD}$		V_{DD}	
V_{hys}	Schmitt trigger voltage hysteresis [3]				400		mV
I_L	Input leakage current	$V_{SS} \le V_{IN} \le V_{DD}$				±1	µA
I_S	Static current consumption [4]	Floating input mode				200	
R_{PU}	Weak pull-up equivalent resistor [5]	$V_{IN}=V_{SS}$	V_{DD}=5V	70	100	130	kΩ
			V_{DD}=3V	130	200	260	
C_{IO}	I/O pin capacitance [6]				5		pF
$t_{f(IO)out}$	Output high to low level fall time [6]	C_L=50pF Between 10% and 90%			25		ns
$t_{r(IO)out}$	Output low to high level rise time [6]				25		
$t_{w(IT)in}$	External interrupt pulse time [7]			1			t_{CPU}

.Notes:

1. Unless otherwise specified, typical data are based on T_A=25°C and V_{DD}=5V.
2. Data based on characterization results, not tested in production.
3. Hysteresis voltage between Schmitt trigger switching levels. Based on characterization results, not tested.
4. Configuration not recommended, all unused pins must be kept at a fixed voltage: using the output mode of the I/O for example or an external pull-up or pull-down resistor (see Figure). Data based on design simulation and/or technology characteristics, not tested in production.
5. The R_{PU} pull-up equivalent resistor is based on a resistive transistor (corresponding I_{PU} current characteristics described in Figure 87). This data is based on characterization results, tested in production at V_{DD}=5V.
6. Data based on characterization results, not tested in production.
7. 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 85Two typical Applications with unused I/O Pin



. I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 86. V_{IL} and V_{IH} vs. V_{DD} with $V_{IN}=V_{SS}$

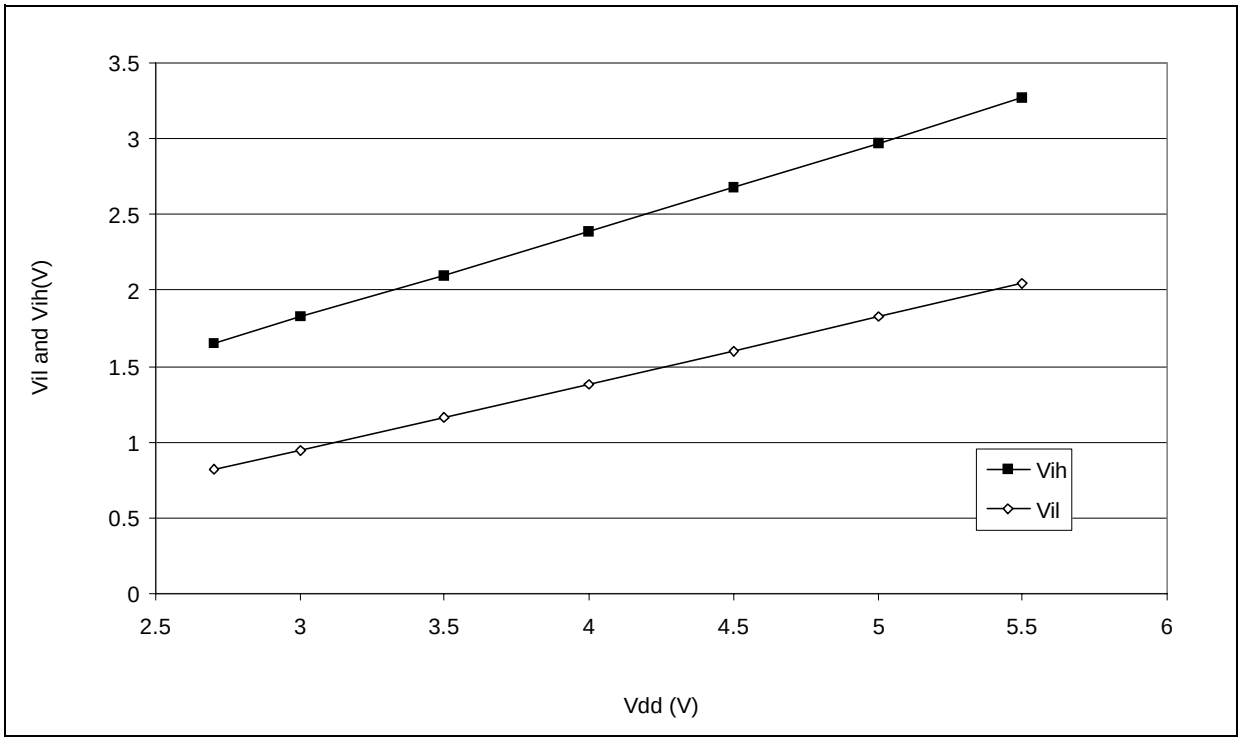


Figure 87. Typical I_{PU} vs. V_{DD} with $V_{IN}=V_{SS}$

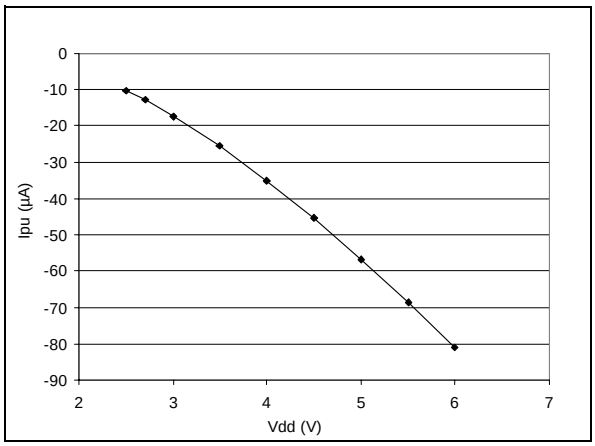
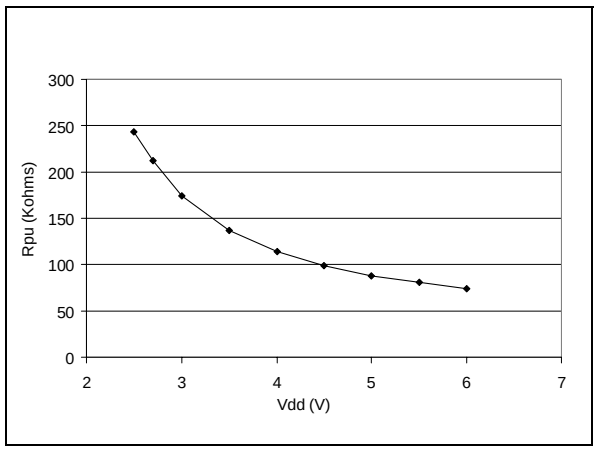


Figure 88. Typical R_{PU} vs. V_{DD} with $V_{IN}=V_{SS}$



I/O PORT PIN CHARACTERISTICS (Cont'd)

13.8.2 Output Driving Current

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

Symbol	Parameter	Conditions		Min	Max	Unit
V_{OL} 1)	Output low level voltage for a standard I/O pin when 8 pins are sunk at same time (see Figure 89 and Figure 92)	V_{DD}=5V	I_{IO}=+5mA		1.2	V
			I_{IO}=+2mA		0.5	
	Output low level voltage for a high sink I/O pin when 4 pins are sunk at same time (see Figure 91 and Figure 93)		I_{IO}=+20mA		1.3	
			I_{IO}=+8mA		0.6	
V_{OH} 2)	Output high level voltage for an I/O pin when 8 pins are sourced at same time (see Figure 90 and Figure 94)		I_{IO}=-5mA	V_{DD}-1.4		
			I_{IO}=-2mA	V_{DD}-0.7		

Figure 89. Typical V_{OL} at V_{DD}=5V (standard)

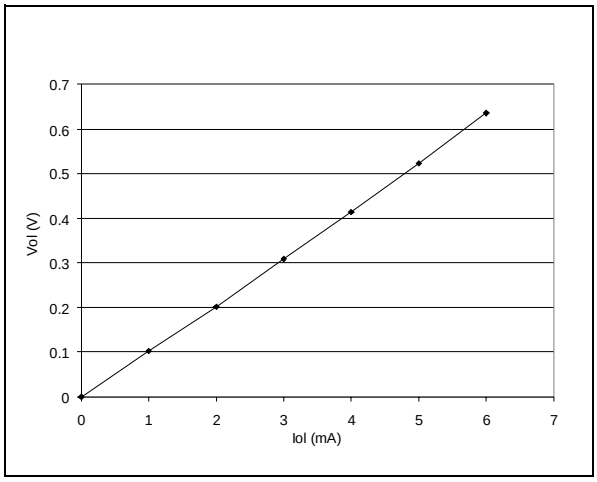


Figure 91. Typical V_{OL} at V_{DD}=5V (high-sink)

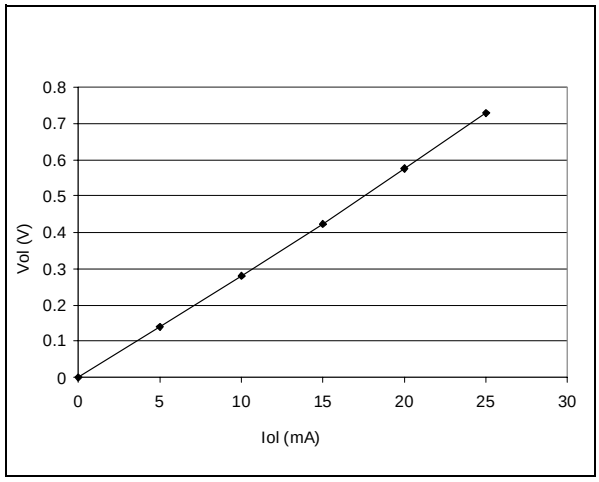
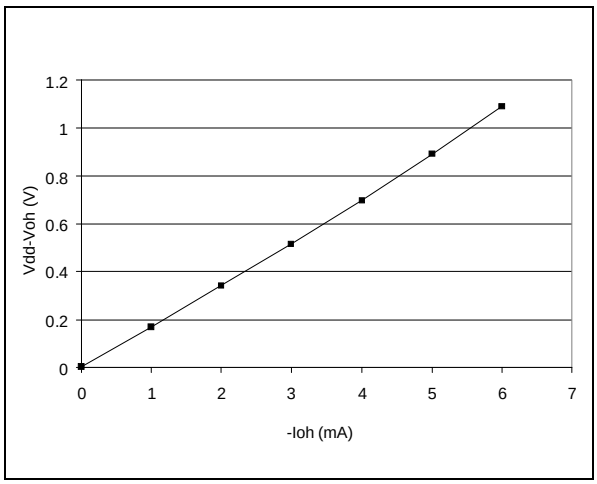


Figure 90Typical V_{DD}-V_{OH} at V_{DD}=5V



Notes:

1. The I_{IO} current sunk must always respect the absolute maximum rating specified in Section 13.2.2 and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VSS}.

2. The I_{IO} current sourced must always respect the absolute maximum rating specified in Section 13.2.2 and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VDD}. True open drain I/O pins does not have V_{OH}.

I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 92.Typical V_{OL} vs. V_{DD} (standard I/Os)

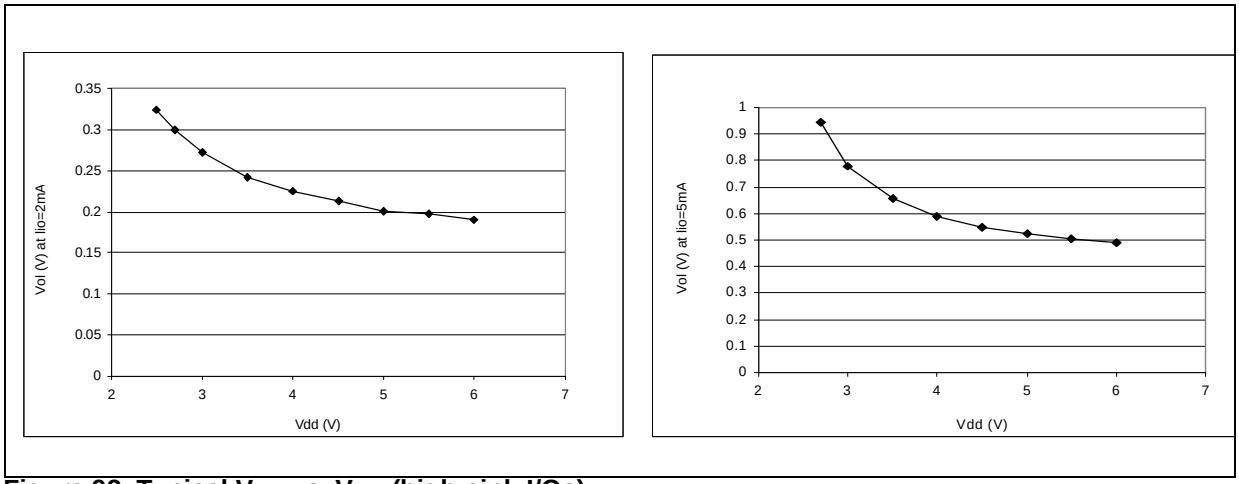


Figure 93. Typical V_{OL} vs. V_{DD} (high-sink I/Os)

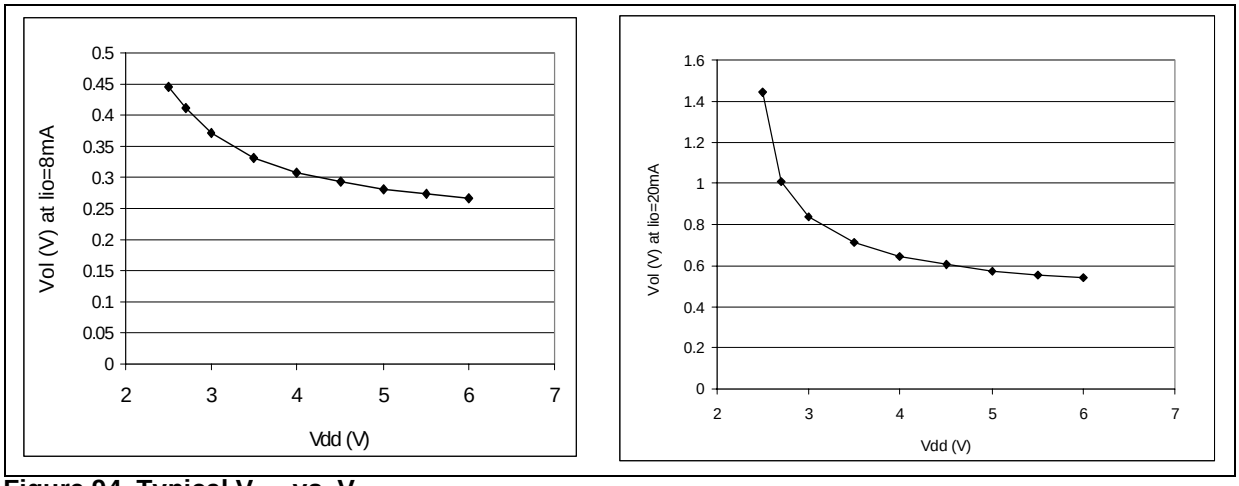
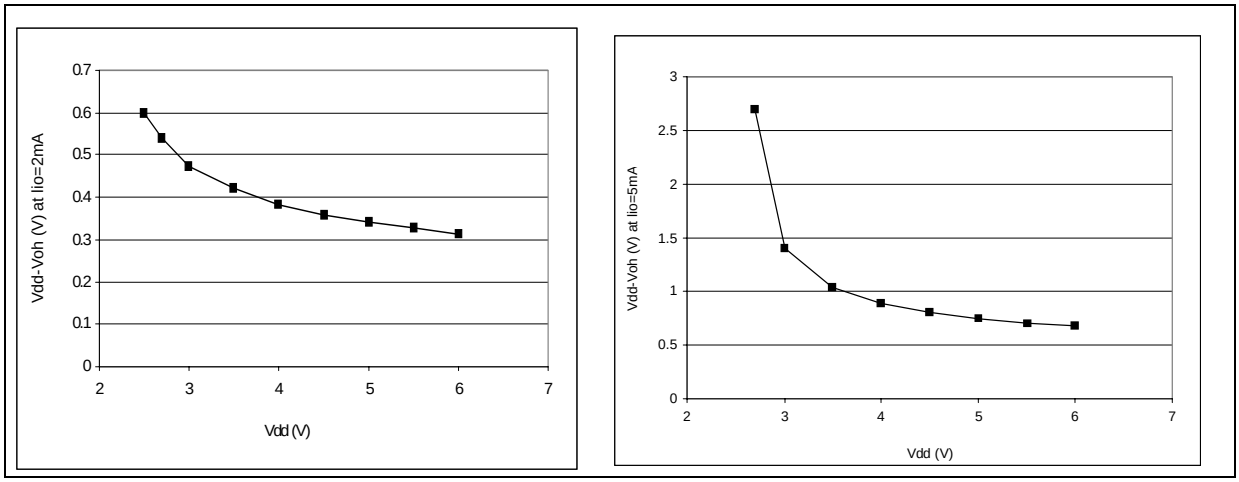


Figure 94. Typical V_{OH} vs. V_{DD}



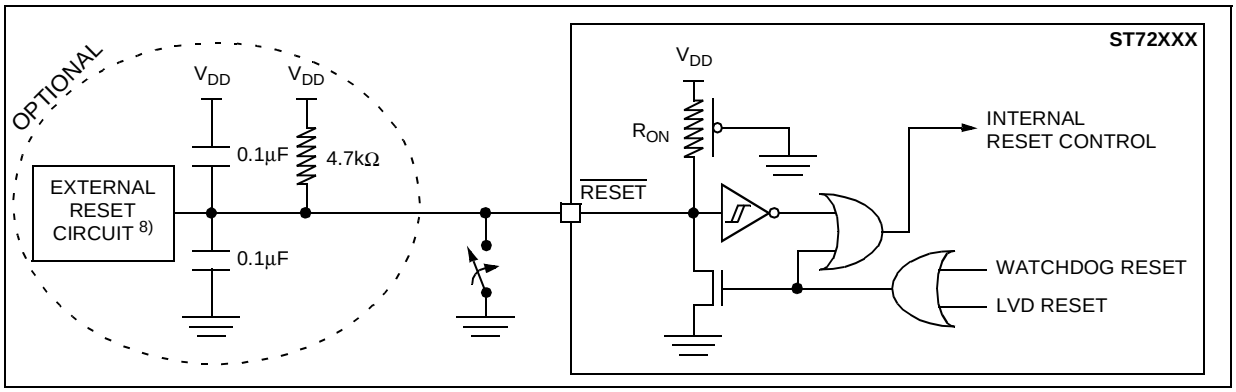
13.9 CONTROL PIN CHARACTERISTICS

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

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

Symbol	Parameter	Conditions		Min	Typ [1]	Max	Unit
V_{IL}	Input low level voltage [2]	V_{DD}=5V		V_{SS}		0.3xV_{DD}	V
V_{IH}	Input high level voltage [2]	V_{DD}=5V		0.7xV_{DD}		V_{DD}	
V_{hys}	Schmitt trigger voltage hysteresis [3]				400		mV
V_{OL}	Output low level voltage [4]	V_{DD}=5V	I_{IO}=+5mA		0.68	0.95	V
			I_{IO}=+2mA		0.28	0.45	
R_{ON}	Weak pull-up equivalent resistor [5]	V_{IN}=V_{SS}	V_{DD}=5V	70	100	130	kΩ
			V_{DD}=3.3V	130	200	260	
$t_{w(RSTL)out}$	Generated reset pulse duration	External pin or internal reset sources			4		$1/f_{SFOSC}$
$t_{h(RSTL)in}$	External reset pulse hold time [6]			20			µs
$t_{g(RSTL)in}$	Filtered glitch duration [7]					100	ns

Figure 95. Typical Application with $\overline{\text{RESET}}$ pin



Notes:

1. Unless otherwise specified, typical data are based on T_A=25°C and V_{DD}=5V.
2. Data based on characterization results, not tested in production.
3. Hysteresis voltage between Schmitt trigger switching levels. Based on characterization results, not tested.
4. The I_{IO} current sunk must always respect the absolute maximum rating specified in Section 13.2.2 and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VSS}. Not tested in production.
5. The R_{ON} pull-up equivalent resistor is based on a resistive transistor (corresponding I_{ON} current characteristics described in Figure 95). This data is based on characterization results, not tested in production.
6. All short pulse applied on $\overline{\text{RESET}}$ pin with a duration below $t_{h(RSTL)in}$ can be ignored.
7. The reset network protects the device against parasitic resets, especially in a noisy environment.
8. The output of the external reset circuit must have an open-drain output to drive the ST7 reset pad. Otherwise the device can be damaged when the ST7 generates an internal reset (LVD or watchdog).

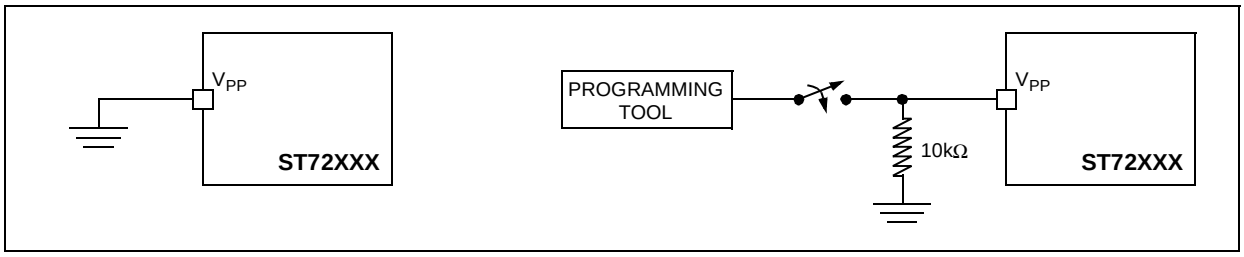
CONTROL PIN CHARACTERISTICS (Cont'd)

13.9.2 V_{PP} Pin

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

Symbol	Parameter	Conditions	Min	Max	Unit
V_{IL}	Input low level voltage [1]		V_{SS}	0.2	V
V_{IH}	Input high level voltage [1]		V_{DD}-0.1	12.6	
I_L	Input leakage current	$V_{IN}=V_{SS}$		±1	µA

Figure 96. Two typical Applications with V_{PP} Pin [2]



Notes:

1. Data based on design simulation and/or technology characteristics, not tested in production.
2. When the ICP mode is not required by the application, V_{PP} pin must be tied to V_{SS}.

13.10 TIMER PERIPHERAL CHARACTERISTICS

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

Refer to I/O port characteristics for more details on the input/output alternate function characteristics (output compare, input capture, external clock, PWM output...).

13.10.1 Watchdog Timer

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{w(WDG)}$	Watchdog time-out duration		65,536		4,194,304	t_{CPU}
		f_{CPU}=8MHz	8.192		524.288	ms

13.10.2 PWM Generator

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
T	Repetition rate	T_{CPU} =125ns	-	125	-	KHz
Res	Resolution	T_{CPU} =125ns	-	125	-	ns
s	Output step	V_{DD}=5V	-	5	-	mV

13.11 COMMUNICATION INTERFACE CHARACTERISTICS

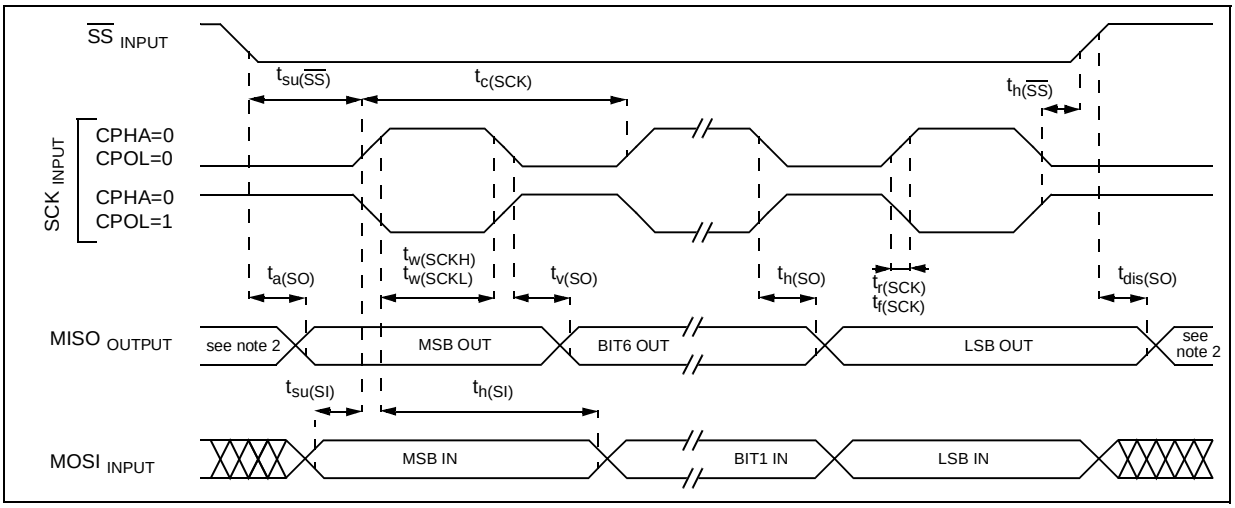
13.11.1 SPI - Serial Peripheral Interface

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

Refer to I/O port characteristics for more details on the input/output alternate function characteristics (SS, SCK, MOSI, MISO).

Symbol	Parameter	Conditions	Min	Max	Unit
f_{SCK} $1/t_{c(SCK)}$	SPI clock frequency	Master f_{CPU}=8MHz	$f_{CPU}/128$ 0.0625	$f_{CPU}/4$ 2	MHz
		Slave f_{CPU}=8MHz	0	$f_{CPU}/2$ 4	
$t_{r(SCK)}$ $t_{f(SCK)}$	SPI clock rise and fall time		see I/O port pin description		
$t_{su(\overline{SS})}$	$\overline{SS}$ setup time	Slave	120		ns
$t_{h(\overline{SS})}$	$\overline{SS}$ hold time	Slave	120		
$t_{w(SCKH)}$ $t_{w(SCKL)}$	SCK high and low time	Master Slave	100 90		
$t_{su(MI)}$ $t_{su(SI)}$	Data input setup time	Master Slave	100 100		
$t_{h(MI)}$ $t_{h(SI)}$	Data input hold time	Master Slave	100 100		
$t_{a(SO)}$	Data output access time	Slave	0	120	
$t_{dis(SO)}$	Data output disable time	Slave		240	
$t_{v(SO)}$	Data output valid time	Slave (after enable edge)		120	
$t_{h(SO)}$	Data output hold time		0		
$t_{v(MO)}$	Data output valid time	Master (before capture edge)	0.25		t_{CPU}
$t_{h(MO)}$	Data output hold time		0.25		

Figure 97. SPI Slave Timing Diagram with CPHA=0 3)



Notes:

1. Data based on design simulation and/or characterisation results, not tested in production.
2. When no communication is on-going the data output line of the SPI (MOSI in master mode, MISO in slave mode) has its alternate function capability released. In this case, the pin status depends on the I/O port configuration.
3. Measurement points are done at CMOS levels: $0.3xV_{DD}$ and $0.7xV_{DD}$.

COMMUNICATION INTERFACE CHARACTERISTICS (Cont'd)

Figure 98. SPI Slave Timing Diagram with CPHA=1[1)]

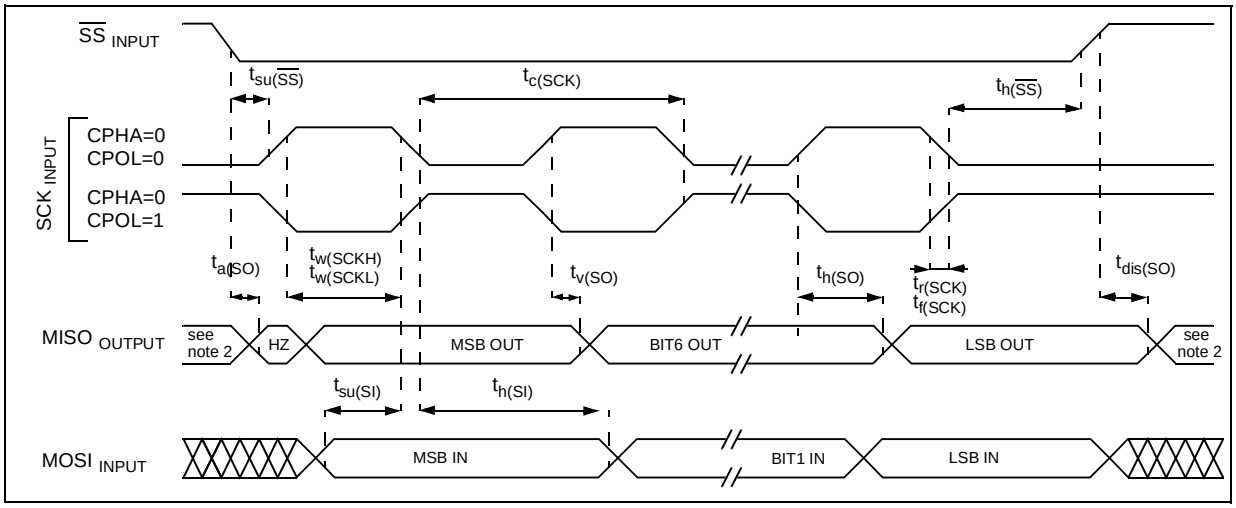
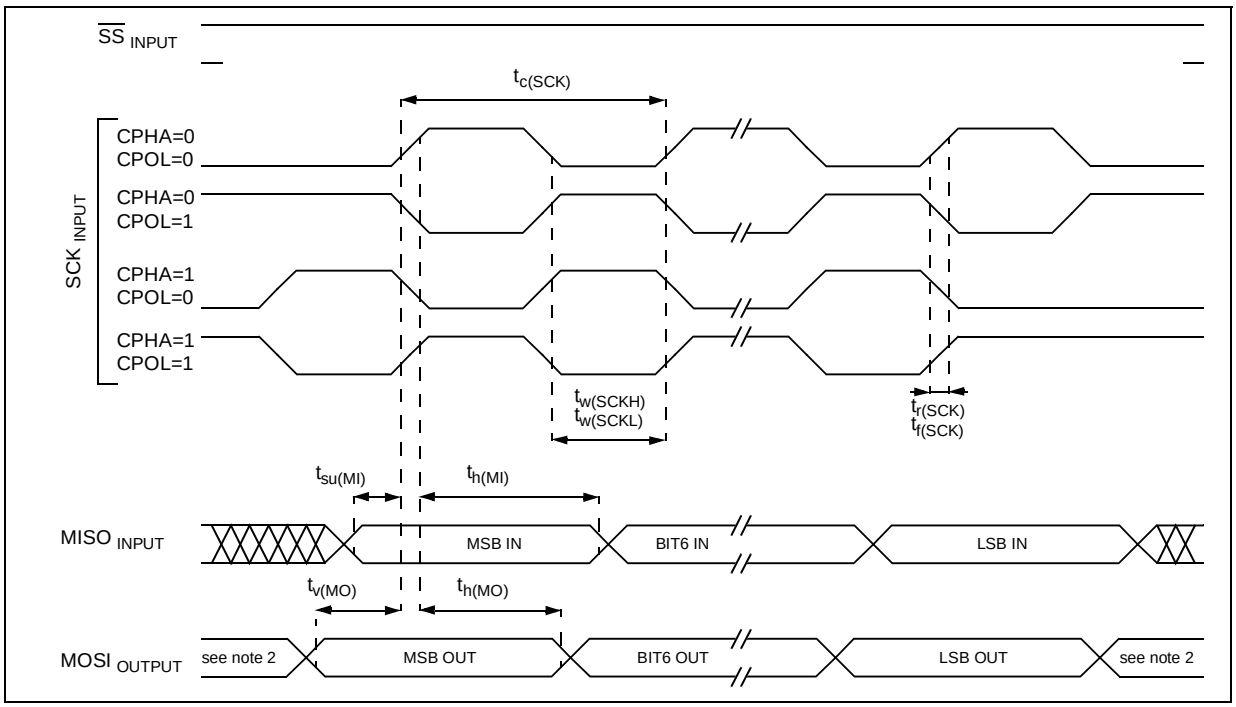


Figure 99. SPI Master Timing Diagram [1)]



Notes:

1. Measurement points are done at CMOS levels: $0.3xV_{DD}$ and $0.7xV_{DD}$.

2. When no communication is on-going the data output line of the SPI (MOSI in master mode, MISO in slave mode) has its alternate function capability released. In this case, the pin status depends of the I/O port configuration.

COMMUNICATION INTERFACE CHARACTERISTICS (Cont'd)

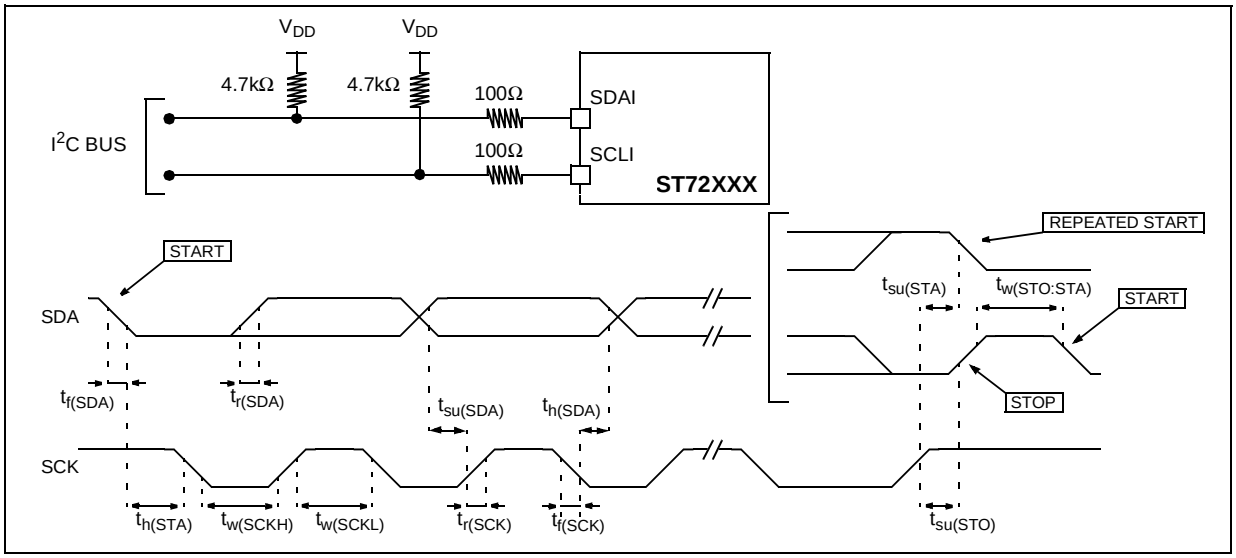
13.11.2 I²C - Inter IC Control Interface

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

Refer to I/O port characteristics for more details on the input/output alternate function characteristics (SDAI and SCLI). The ST7 I²C interface meets the requirements of the Standard I²C communication protocol described in the following table.

Symbol	Parameter	Standard mode I²C		Fast mode I²C		Unit
		Min 1)	Max 1)	Min 1)	Max 1)	
$t_{w(SCLL)}$	SCL clock low time	4.7		1.3		µs
$t_{w(SCLH)}$	SCL clock high time	4.0		0.6		
$t_{su(SDA)}$	SDA setup time	250		100		ns
$t_{h(SDA)}$	SDA data hold time	0 3)		0 2)	900 3)	
$t_{r(SDA)}$ $t_{r(SCL)}$	SDA and SCL rise time		1000	$20+0.1C_b$	300	
$t_{f(SDA)}$ $t_{f(SCL)}$	SDA and SCL fall time		300	$20+0.1C_b$	300	
$t_{h(STA)}$	START condition hold time	4.0		0.6		µs
$t_{su(STA)}$	Repeated START condition setup time	4.7		0.6		
$t_{su(STO)}$	STOP condition setup time	4.0		0.6		ns
$t_{w(STO:STA)}$	STOP to START condition time (bus free)	4.7		1.3		ms
C_b	Capacitive load for each bus line		400		400	pF

Figure 100. Typical Application with I²C Bus and Timing Diagram 4)



Notes:

1. Data based on standard I²C protocol requirement, not tested in production.
2. The device must internally provide a hold time of at least 300ns for the SDA signal in order to bridge the undefined region of the falling edge of SCL.
3. The maximum hold time of the START condition has only to be met if the interface does not stretch the low period of SCL signal.
4. Measurement points are done at CMOS levels: $0.3xV_{DD}$ and $0.7xV_{DD}$.

COMMUNICATION INTERFACE CHARACTERISTICS (Cont'd)

13.11.3 I^2C - Inter IC Control Interface

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

1) The device must internally provide a hold time of at least 300 ns for the SDA signal in order to bridge the undefined region of the falling edge of SCL

2) The maximum hold time of the START condition has only to be met if the interface does not stretch the low period of SCL signal

Cb = total capacitance of one bus line in pF

COMMUNICATION INTERFACE CHARACTERISTICS (Cont'd)

13.11.4 USB - Universal Bus Interface

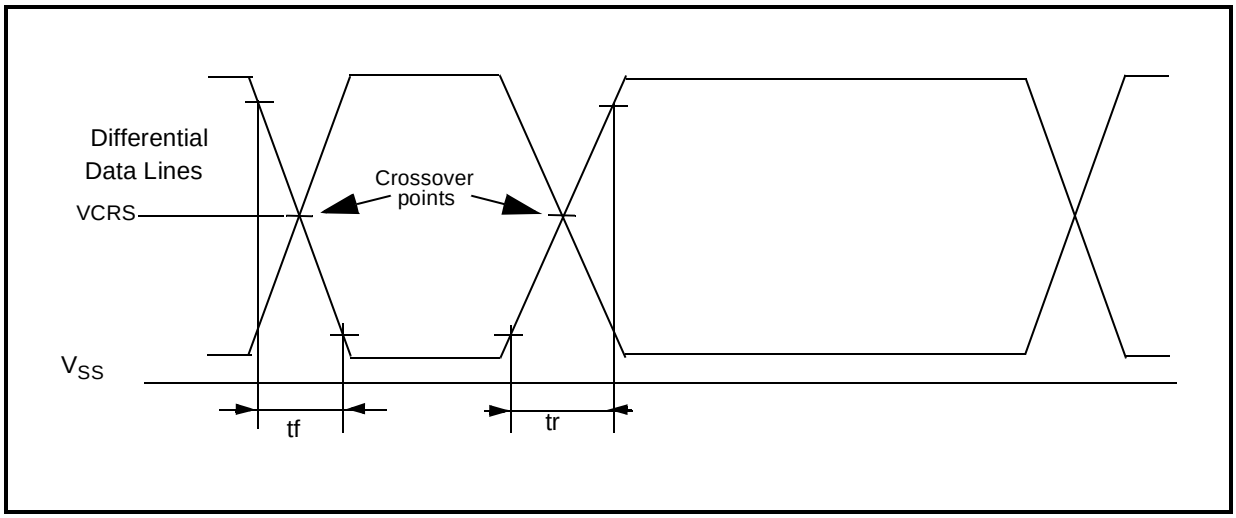
USB DC Electrical Characteristics					
Parameter	**Symbol**	**Conditions**	**Min.** [2]	**Max.** [2]	**Unit**
Input Levels:					
Differential Input Sensitivity	VDI	I(D+, D-)	0.2		V
Differential Common Mode Range	VCM	Includes VDI range	0.8	2.5	V
Single Ended Receiver Threshold	VSE		1.3	2.0	V
Output Levels					
Static Output Low	VOL	RL of 1.5K ohms to 3.6V [1]		0.3	V
Static Output High	VOH	RL of 15K ohm to V_{SS} [1]	2.8	3.6	V
$USBV_{CC}$: voltage level [3]	USBV	V_{DD}=4.0V - 5.5V I_{LOAD} Max = 3mA	3.00	3.60	V

Note 1: RL is the load connected on the USB drivers.

Note 2: All the voltages are measured from the local ground potential.

Note 3: An external decoupling capacitor (typical 100nF, min 47nF) must be connected between this pin and $USBV_{SS}$.

Figure 101. USB: Data Signal Rise and Fall Time



USB: Full speed electrical characteristics					
Parameter	**Symbol**	**Conditions**	**Min**	**Max**	**Unit**
Driver characteristics:					
Rise time	tr	Note 1,CL=50 pF	4	20	ns
Fall Time	tf	Note 1, CL=50 pF	4	20	ns
Rise/ Fall Time matching	trfm	tr/tf	90	110	%
Output signal Crossover Voltage	VCRS		1.3	2.0	V

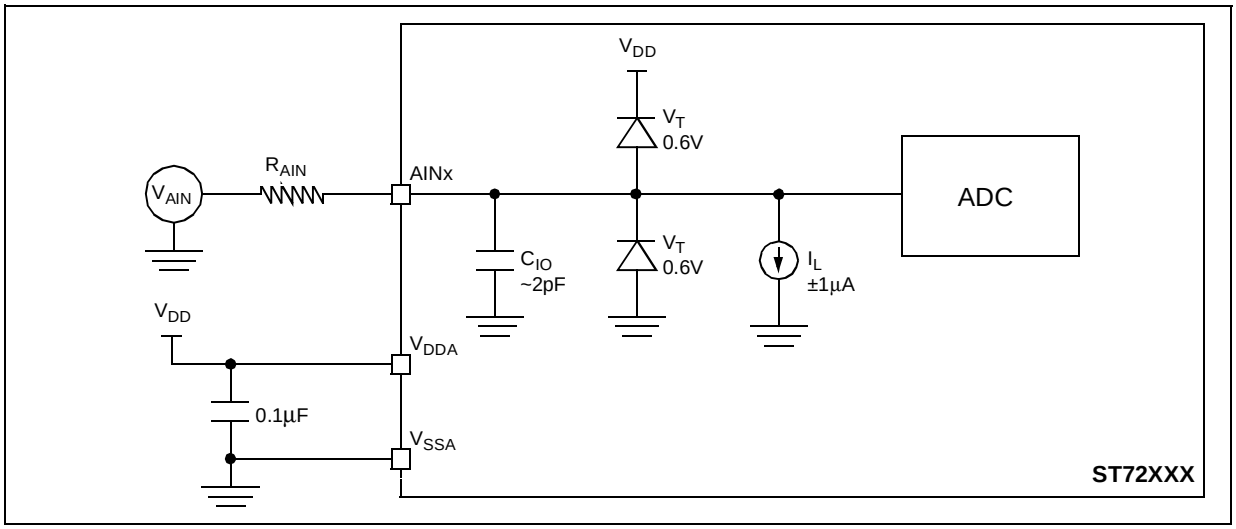
Note1: Measured from 10% to 90% of the data signal. For more detailed informations, please refer to Chapter 7 (Electrical) of the USB specification (version 1.1).

13.12 8-BIT ADC CHARACTERISTICS

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

Symbol	Parameter	Conditions	Min	Typ [1]	Max	Unit
f_{ADC}	ADC clock frequency				4	MHz
V_{AIN}	Conversion range voltage [2]		V_{SSA}		V_{DDA}	V
R_{AIN}	External input resistor				10 [3]	kΩ
C_{ADC}	Internal sample and hold capacitor			6		pF
t_{STAB}	Stabilization time after ADC enable	f_{CPU}=8MHz, f_{ADC}=2MHz	0 [4]			µs
t_{ADC}	Conversion time (Sample+Hold)		6			µs
	- Sample capacitor loading time		4			$1/f_{ADC}$
	- Hold conversion time		8			$1/f_{ADC}$

Figure 102. Typical Application with ADC



Notes:

1. Unless otherwise specified, typical data are based on T_A=25°C and V_{DD}-V_{SS}=5V. They are given only as design guidelines and are not tested.

2. When V_{DDA} and V_{SSA} pins are not available on the pinout, the ADC refer to V_{DD} and V_{SS} .

3. Any added external serial resistor will downgrade the ADC accuracy (especially for resistance greater than 10kΩ). Data based on characterization results, not tested in production.

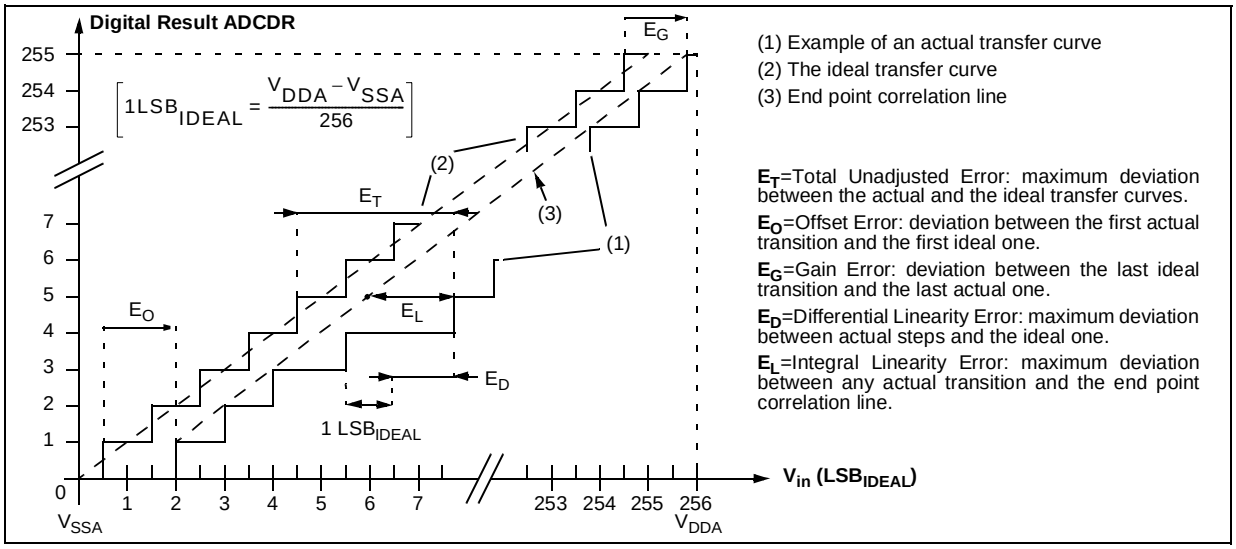
4. The stabilization time of the AD converter is masked by the first t_{LOAD}. The first conversion after the enable is then always valid.

8-BIT ADC CHARACTERISTICS (Cont'd)

ADC Accuracy

Symbol	Parameter	Conditions ⇒ ⇓	V_{DD}=5.5V, 2) f_{CPU}=1MHz		V_{DD}=5.0V, 3) f_{CPU}=8MHz 1)		V_{DD}=3.3V, 3) f_{CPU}=8MHz 1)	
			Min	Max	Min	Max	Min	Max
E_T	Total Unadjusted Error 1)			2.5		2.5		2.5
E_0	Offset Error		-0.5	1.5	-1.0	1.5	-1.0	1.5
E_G	Gain Error 1)		-2.0	0	-2.0	0	-2.0	0
$\|E_D\|$	Differential linearity error 1)			1.5		1.5		1.5
$\|E_L\|$	Integral linearity error 1)			2.5		2.5		3.0

Figure 103. ADC Accuracy Characteristics



Notes:

1. ADC Accuracy vs. Negative Injection Current:
For I_{INJ-}=0.8mA, the typical leakage induced inside the die is 1.6µA and the effect on the ADC accuracy is a loss of 1 LSB for each 10KΩ increase of the external analog source impedance. This effect on the ADC accuracy has been observed under worst-case conditions for injection:
- negative injection
- injection to an Input with analog capability, adjacent to the enabled Analog Input
- at 5V V_{DD} supply, and worst case temperature.

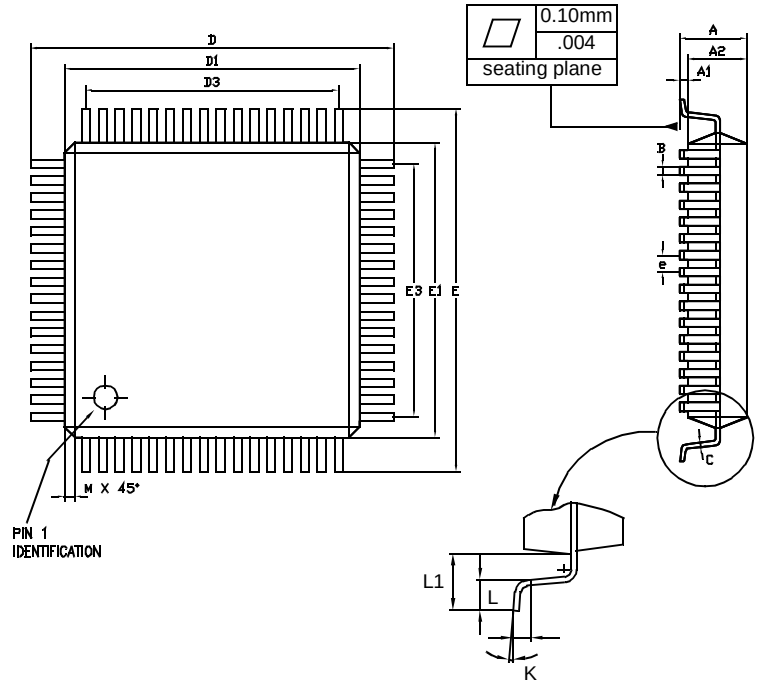
2. Data based on characterization results with T_A=25°C.

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

14 PACKAGE CHARACTERISTICS

14.1 PACKAGE MECHANICAL DATA

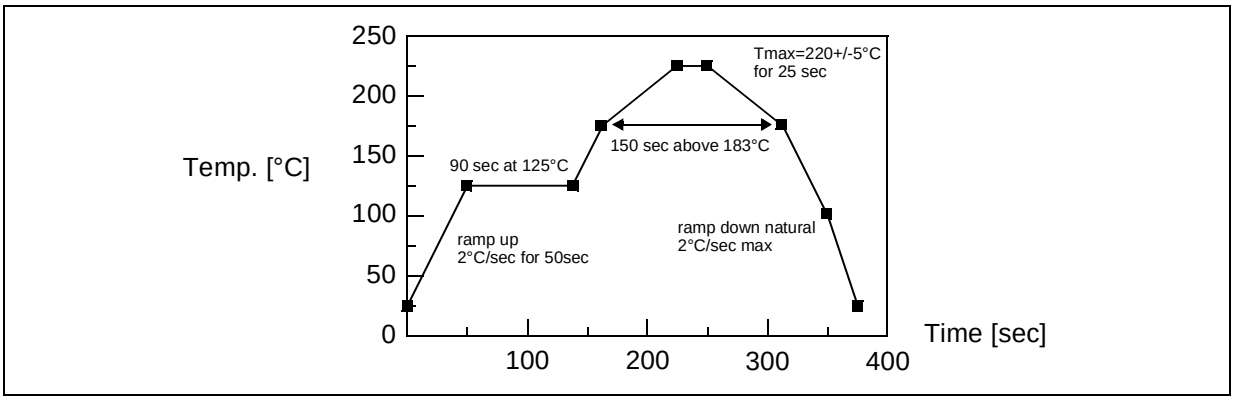
Figure 104. 64-Pin 10 x 10 Thin Quad Flat Package



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		12.00			0.472	
D1		10.00			0.394	
E		12.00			0.472	
E1		10.00			0.394	
e		0.50			0.020	
K	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		ND	16	NE	16

PACKAGE MECHANICAL DATA (Cont'd)

Figure 105. Recommended Reflow Oven Profile (MID JEDEC)



15 DEVICE CONFIGURATION AND ORDERING INFORMATION

Each device is available for production in user programmable versions (FLASH) as well as in factory coded versions (ROM). FLASH devices are shipped to customers with a default content (FFh), while ROM factory coded parts contain the code supplied by the customer. This implies that FLASH devices have to be configured by the customer using the Option Bytes while the ROM devices are factory-configured.

15.1 OPTION BYTE

The option byte allows the hardware configuration of the microcontroller to be selected.
The option byte has no address in the memory map and can be accessed only in programming mode (for example using a standard ST7 programming tool). The default content of the FLASH is fixed to FFh. This means that all the options have "1" as their default value.
In masked ROM devices, the option byte is fixed in hardware by the ROM code (see option list)

7							0
-	PE5 PU	PS MOD 1	PS MOD 0	-	WDG SW	USB EN	FMP_ R

OPT7 = Reserved. Must be kept at 1.

OPT6 = **PE5PU** *I/O Port PE5 Pull-up Option*
This option bit determines if a pull-up is connected on Port E5.

0: Pull up present on PE5
1: No pull-up on PE5

When PE5PU=00:

– For input, software can enable or disable the pull-up by programming PEOR.5 and PEDDR.5=0.

– For output, the pull-up is enabled when Open Drain is selected by programming PEOR.5= and PEDDR.5=1.

Refer to the following table.

Configuration	PE5PU OPTION	PEOR.5	PEDDR.5
Input floating	0	0	0
Output Open Drain with Pull-up		0	1
Input with pull-up		1	0
Output push pull		1	1
Input floating	1	0	0
Output Open Drain		0	1
Input floating		1	0
Output push pull		1	1

OPT5:4 = **PSMOD[1:0]** *Power Supply Mode*
These option bits configure the power supply mode.

Mode	OPT5	OPT4
Stand-alone mode forced	0	0
Dual Supply (normal) Mode	x	1
USB mode forced	1	0

OPT3 = Reserved. Must be kept at 1.

OPT2= **WDG SW** *Hardware or software watchdog*
This option bit selects the watchdog type.
0: Hardware (watchdog always enabled)
1: Software (watchdog to be enabled by software)

OPT1 = **USBEN**
0: USBEN alternate function disabled. Port F4 is free for general purpose I/O
1: USBEN alternate function enabled on Port F4 (function controlled by hardware)

OPT0= **FMP_R** *Flash memory read-out protection*
This option indicates if the user flash memory is protected against read-out piracy. This protection is based on read and a write protection of the memory in test modes and IAP. Erasing the option bytes when the FMP_R option is selected induce the whole user memory erasing first.
0: Read-out protection enabled
1: Read-out protection disabled

15.2 DEVICE ORDERING INFORMATION AND TRANSFER OF CUSTOMER CODE

Customer code is made up of the ROM contents. The ROM contents are to be sent on diskette, or by electronic means, with the hexadecimal file in .S19 format generated by the development tool. All unused bytes must be set to FFh.

The customer code should be communicated to STMicroelectronics with the correctly completed OPTION LIST appended.

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

Figure 106. Sales Type Coding Rules [2)]

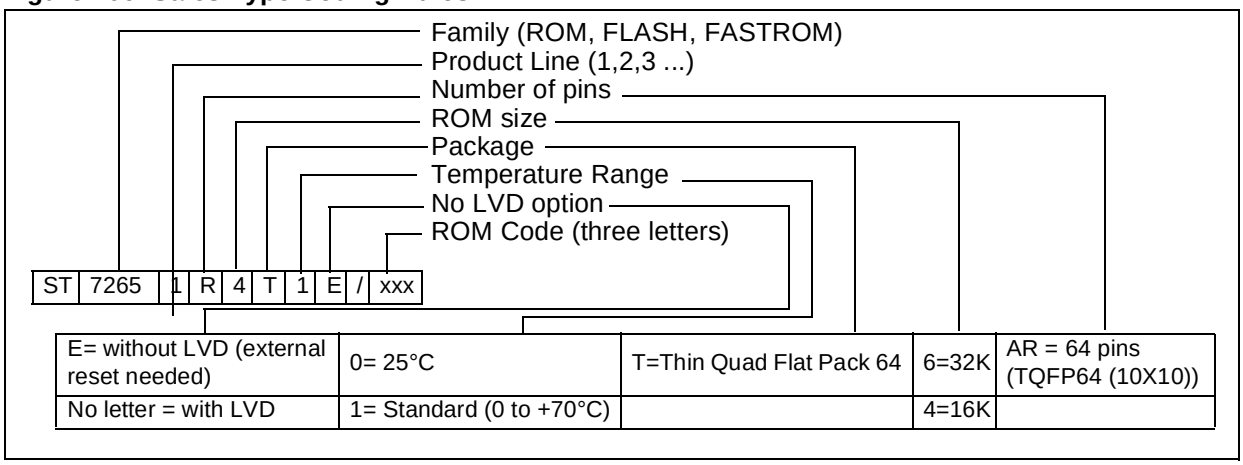


Table 37. Ordering Information

Sales Type [1) 2)]	Program Memory (bytes)	User RAM (bytes)	Package	Operating Voltage
ST72F651AR6T1	32K FLASH	5K	TQFP 64 (10X10)	4.0V-5.5V
ST72F652AR4T1	16K FLASH	512		
ST72651AR6T1/xxx	32K ROM	5K		
ST72652AR4T1/xxx	16K ROM	512		
ST72P651AR6T1/xxx	32K FASTROM	5K		
ST72P652AR4T1/xxx	16K FASTROM	512		
ST72F651AR6T1E	32K FLASH	5K		2.7V-5.5V
ST72F652AR4T1E	16K FLASH	512		
ST72651AR6T1E/xxx	32K ROM	5K		
ST72652AR4T1E/xxx	16K ROM	512		
ST72P651AR6T1E/xxx	32K FASTROM	5K		
ST72P652AR4T1E/xxx	16K FASTROM	512		

Note 1. /xxx stands for the ROM or FASTROM code name assigned by STMicroelectronics

Note 2. Devices with E Suffix have no embedded LVD

ST7265x MICROCONTROLLER OPTION LIST

Customer . . . . . .
Address . . . . . .
. . . . . .
Contact . . . . . .
Phone No . . . . . .
Reference/ROM or FASTROM Code*: . . . . . .
*ROM or FASTROM code name is assigned by STMicroelectronics.
ROM or 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:	16K (without low voltage feature)	32K
TQFP64:	[ ] ST72652AR4T1	[ ] ST72651AR6T1
FASTROM DEVICE:	16K (without low voltage feature)	32K
TQFP64:	[ ] ST72P652AR4T1	[ ] ST72P651AR6T1

Conditioning (check only one option):

[ ] Tray [ ] Tape & Reel

LVD option: [ ] Yes [ ] No

Marking: [ ] Standard marking
[ ] Special marking (ROM only):
TQFP64 (10 char. max): _ _ _ _ _ _ _ _ _ _

Authorized characters are letters, digits, '.', '-', '/' and spaces only.
Please consult your local STMicroelectronics sales office for other marking details if required.

Pull-up on PE5: [ ] Disabled [ ] Enabled

Power Supply mode: [ ] Stand-alone mode
[ ] Dual supply mode
[ ] USB mode

Watchdog Selection: [ ] Software Activation [ ] Hardware Activation

USBEN alternate function: [ ] Disabled [ ] Enabled

Readout Protection: [ ] Disabled [ ] Enabled

Software Development: [ ] STMicroelectronics [ ] Customer [ ] External laboratory

Comments: . . . . . .
. . . . . .
Supply Operating Range in the application: . . . . . .
Notes . . . . . .
. . . . . .
Date . . . . . .

Signature . . . . . .

15.3 DEVELOPMENT TOOLS

STmicroelectronics offers a range of hardware and software development tools for the ST7 microcontroller family. Full details of tools available for the ST7 from third party manufacturers can be obtain from the STMicroelectronics Internet site:
➟ http//mcu.st.com.

Tools from these manufacturers include C compliers, emulators and gang programmers.

STMicroelectronics Tools

Three types of development tool are offered by ST, all of them connect to a PC via a parallel (LPT) port: see Table 38 for more details.

Table 38. STMicroelectronics Tool Features

	In-Circuit Emulation	Programming Capability[1)]	Sales Type	Remarks
ST7 FLASH HDS2 Emulator [3)]	Yes, powerful emulation features including trace/ logic analyzer	No	ST7MDTU5-EMU2B	
ST7 Programming Board [3)]	No	Yes	ST7MDTU5-EPB2/EU ST7MDTU5-EPB2/US	220V 110V
Gang Programmer			See 3rd Party	TQFP64 package
C Hiware Compiler			ST7-HICROSS	for PC
Hiware Debugger			ST7-HIWAVE	for PC

Note:
1. In-Application Programming (IAP) and In-Circuit programming for Flash devices.
2. These products come with a CD ROM which contains the following software:
– ST7 Assembly toolchain
– STVD7 and WGDB7 powerful Source Level Debugger for Win 3.1, Win 95 and NT
– C compiler demo versions
– ST Realizer for Win 3.1 and Win 95
– Windows Programming Tools for Win 3.1, Win 95 and NT
3. DIP packages only.

15.4 ST7 APPLICATION NOTES

IDENTIFICATION	DESCRIPTION
EXAMPLE DRIVERS	
AN 969	SCI COMMUNICATION BETWEEN ST7 AND PC
AN 970	SPI COMMUNICATION BETWEEN ST7 AND EEPROM
AN 971	I²C COMMUNICATING BETWEEN ST7 AND M24CXX EEPROM
AN 972	ST7 SOFTWARE SPI MASTER COMMUNICATION
AN 973	SCI SOFTWARE COMMUNICATION WITH A PC USING ST72251 16-BIT TIMER
AN 974	REAL TIME CLOCK WITH ST7 TIMER OUTPUT COMPARE
AN 976	DRIVING A BUZZER THROUGH ST7 TIMER PWM FUNCTION
AN 979	DRIVING AN ANALOG KEYBOARD WITH THE ST7 ADC
AN 980	ST7 KEYPAD DECODING TECHNIQUES, IMPLEMENTING WAKE-UP ON KEYSTROKE
AN1017	USING THE ST7 UNIVERSAL SERIAL BUS MICROCONTROLLER
AN1041	USING ST7 PWM SIGNAL TO GENERATE ANALOG OUTPUT (SINUSOID)
AN1042	ST7 ROUTINE FOR I²C SLAVE MODE MANAGEMENT
AN1044	MULTIPLE INTERRUPT SOURCES MANAGEMENT FOR ST7 MCUS
AN1045	ST7 S/W IMPLEMENTATION OF I²C BUS MASTER
AN1046	UART EMULATION SOFTWARE
AN1047	MANAGING RECEPTION ERRORS WITH THE ST7 SCI PERIPHERALS
AN1048	ST7 SOFTWARE LCD DRIVER
AN1078	PWM DUTY CYCLE SWITCH IMPLEMENTING TRUE 0% & 100% DUTY CYCLE
AN1082	DESCRIPTION OF THE ST72141 MOTOR CONTROL PERIPHERAL REGISTERS
AN1083	ST72141 BLDC MOTOR CONTROL SOFTWARE AND FLOWCHART EXAMPLE
AN1105	ST7 PCAN PERIPHERAL DRIVER
AN1129	PERMANENT MAGNET DC MOTOR DRIVE.
AN1130	AN INTRODUCTION TO SENSORLESS BRUSHLESS DC MOTOR DRIVE APPLICATIONS WITH THE ST72141
AN1148	USING THE ST7263 FOR DESIGNING A USB MOUSE
AN1149	HANDLING SUSPEND MODE ON A USB MOUSE
AN1180	USING THE ST7263 KIT TO IMPLEMENT A USB GAME PAD
AN1276	BLDC MOTOR START ROUTINE FOR THE ST72141 MICROCONTROLLER
AN1321	USING THE ST72141 MOTOR CONTROL MCU IN SENSOR MODE
AN1325	USING THE ST7 USB LOW-SPEED FIRMWARE V4.X
AN1445	USING THE ST7 SPI TO EMULATE A 16-BIT SLAVE
AN1475	DEVELOPING AN ST7265X MASS STORAGE APPLICATION
AN1504	STARTING A PWM SIGNAL DIRECTLY AT HIGH LEVEL USING THE ST7 16-BIT TIMER
PRODUCT EVALUATION	
AN 910	PERFORMANCE BENCHMARKING
AN 990	ST7 BENEFITS VERSUS INDUSTRY STANDARD
AN1077	OVERVIEW OF ENHANCED CAN CONTROLLERS FOR ST7 AND ST9 MCUS
AN1086	U435 CAN-DO SOLUTIONS FOR CAR MULTIPLEXING
AN1150	BENCHMARK ST72 VS PC16
AN1151	PERFORMANCE COMPARISON BETWEEN ST72254 & PC16F876
AN1278	LIN (LOCAL INTERCONNECT NETWORK) SOLUTIONS
PRODUCT MIGRATION	
AN1131	MIGRATING APPLICATIONS FROM ST72511/311/214/124 TO ST72521/321/324
AN1322	MIGRATING AN APPLICATION FROM ST7263 REV.B TO ST7263B
AN1365	GUIDELINES FOR MIGRATING ST72C254 APPLICATION TO ST72F264
PRODUCT OPTIMIZATION	

IDENTIFICATION	DESCRIPTION
AN 982	USING ST7 WITH CERAMIC RENATOR
AN1014	HOW TO MINIMIZE THE ST7 POWER CONSUMPTION
AN1015	SOFTWARE TECHNIQUES FOR IMPROVING MICROCONTROLLER EMC PERFORMANCE
AN1040	MONITORING THE VBUS SIGNAL FOR USB SELF-POWERED DEVICES
AN1070	ST7 CHECKSUM SELF-CHECKING CAPABILITY
AN1324	CALIBRATING THE RC OSCILLATOR OF THE ST7FLITE0 MCU USING THE MAINS
AN1477	EMULATED DATA EEPROM WITH XFLASH MEMORY
AN1502	EMULATED DATA EEPROM WITH ST7 HDFLASH MEMORY
AN1529	EXTENDING THE CURRENT & VOLTAGE CAPABILITY ON THE ST7265 VDDF SUPPLY
AN1530	ACCURATE TIMEBASE FOR LOW-COST ST7 APPLICATIONS WITH INTERNAL RC OSCILLATOR
PROGRAMMING AND TOOLS	
AN 978	KEY FEATURES OF THE STVD7 ST7 VISUAL DEBUG PACKAGE
AN 983	KEY FEATURES OF THE COSMIC ST7 C-COMPILER PACKAGE
AN 985	EXECUTING CODE IN ST7 RAM
AN 986	USING THE INDIRECT ADDRESSING MODE WITH ST7
AN 987	ST7 SERIAL TEST CONTROLLER PROGRAMMING
AN 988	STARTING WITH ST7 ASSEMBLY TOOL CHAIN
AN 989	GETTING STARTED WITH THE ST7 HIWARE C TOOLCHAIN
AN1039	ST7 MATH UTILITY ROUTINES
AN1064	WRITING OPTIMIZED HIWARE C LANGUAGE FOR ST7
AN1071	HALF DUPLEX USB-TO-SERIAL BRIDGE USING THE ST72611 USB MICROCONTROLLER
AN1106	TRANSLATING ASSEMBLY CODE FROM HC05 TO ST7
AN1179	PROGRAMMING ST7 FLASH MICROCONTROLLERS IN REMOTE ISP MODE (IN-SITU PROGRAMMING)
AN1446	USING THE ST72521 EMULATOR TO DEBUG A ST72324 TARGET APPLICATION
AN1478	PORTING AN ST7 PANTA PROJECT TO CODEWARRIOR IDE
AN1527	DEVELOPING A USB SMARTCARD READER WITH ST7SCR
AN1575	ON-BOARD PROGRAMMING METHODS FOR XFLASH AND HDFLASH ST7 MCUS

16 SUMMARY OF CHANGES

Description of the changes between the current release of the specification and the previous one.

Revision	Main changes	Date
2.2	Removed references to TQFP64 14x14 package Added section 13.7.2 on page 133 Changed Table 37 on page 154 Updated option list (page 155) Modified description of effect of USB reset on USBEPnR bits in Section 11.3.5 Added note to Table 33, “I2C Register Map,” on page 112 “Busy bit not used” Updated SPI Section 11.6 Multimaster mode removed.	Nov 02

Notes: