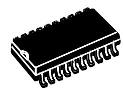
ST7L1

8-BIT MCU FOR AUTOMOTIVE WITH SINGLE VOLTAGE FLASH/ROM MEMORY, DATA EEPROM, ADC, 5 TIMERS, SPI

PRELIMINARY DATA

- **Memories**
  – 4 Kbytes single voltage extended Flash (XFlash) or ROM with read-out protection, In-Circuit programming and In-Application Programming (ICP and IAP), 10K write/erase cycles guaranteed, data retention 20 years at 55°C
  – 256 bytes RAM
  – 128 bytes data E2PROM with read-out protection, 300K write/erase cycles guaranteed, data retention 20 years at 55°C
- **Clock, Reset and Supply Management**
  – Enhanced reset system
  – Enhanced low voltage supervisor (LVD) for main supply and an auxiliary voltage detector (AVD) with interrupt capability for implementing safe power-down procedures
  – Clock sources: Internal 1% RC oscillator, crystal/ceramic resonator or external clock
  – Optional x4 or x8 PLL for 4 or 8 MHz internal clock (only x8 PLL available for ROM devices)
  – 5 power saving modes: Halt, Active Halt, Auto Wake-Up from Halt, Wait and Slow
- **I/O Ports**
  – Up to 17 multifunctional bidirectional I/O lines
  – 7 high sink outputs
- **5 Timers**
  – Configurable Watchdog timer
  – Two 8-bit Lite timers with prescaler, 1 realtime base and 1 input capture
  – Two 12-bit auto-reload timers with 4 PWM outputs, 1 input capture, 1 pulse and 4 output compare functions



SO20
300mil

- **Communication Interface**
  – SPI synchronous serial interface
- **Interrupt Management**
  – 12 interrupt vectors plus TRAP and RESET
  – 15 external interrupt lines (on 4 vectors)
- **A/D Converter**
  – 7 input channels
  – 10-bit precision
- **Instruction Set**
  – 8-bit data manipulation
  – 63 basic instructions with illegal opcode detection
  – 17 main addressing modes
  – 8 x 8 unsigned multiply instructions
- **Development Tools**
  – Full hardware/software development package
  – DM (Debug Module)

Device Summary

Features	ST7L15	ST7L19
Program memory - bytes	4K	
RAM (stack) - bytes	256 (128)	
Data EEPROM - bytes	-	128
Peripherals	Lite Timer with Watchdog, Autoreload Timer, SPI, 10-bit ADC	
Operating Supply	3V to 5.5V	
CPU Frequency	Up to 8 MHz (w/ext OSC up to 16 MHz and int 1 MHz RC 1%, PLLx8/4 MHz)	
Operating Temperature	Up to -40 to +85°C / -40 to +125°C	
Packages	SO20 300mil	

Rev 1

This is preliminary information on a new product now in development or undergoing evaluation. Details are subject to change without notice.

Table of Contents

Table of Contents

1 INTRODUCTION

The ST7L1 is a member of the ST7 microcontroller family suitable for automotive applications. All ST7 devices are based on a common industry-standard 8-bit core, featuring an enhanced instruction set.

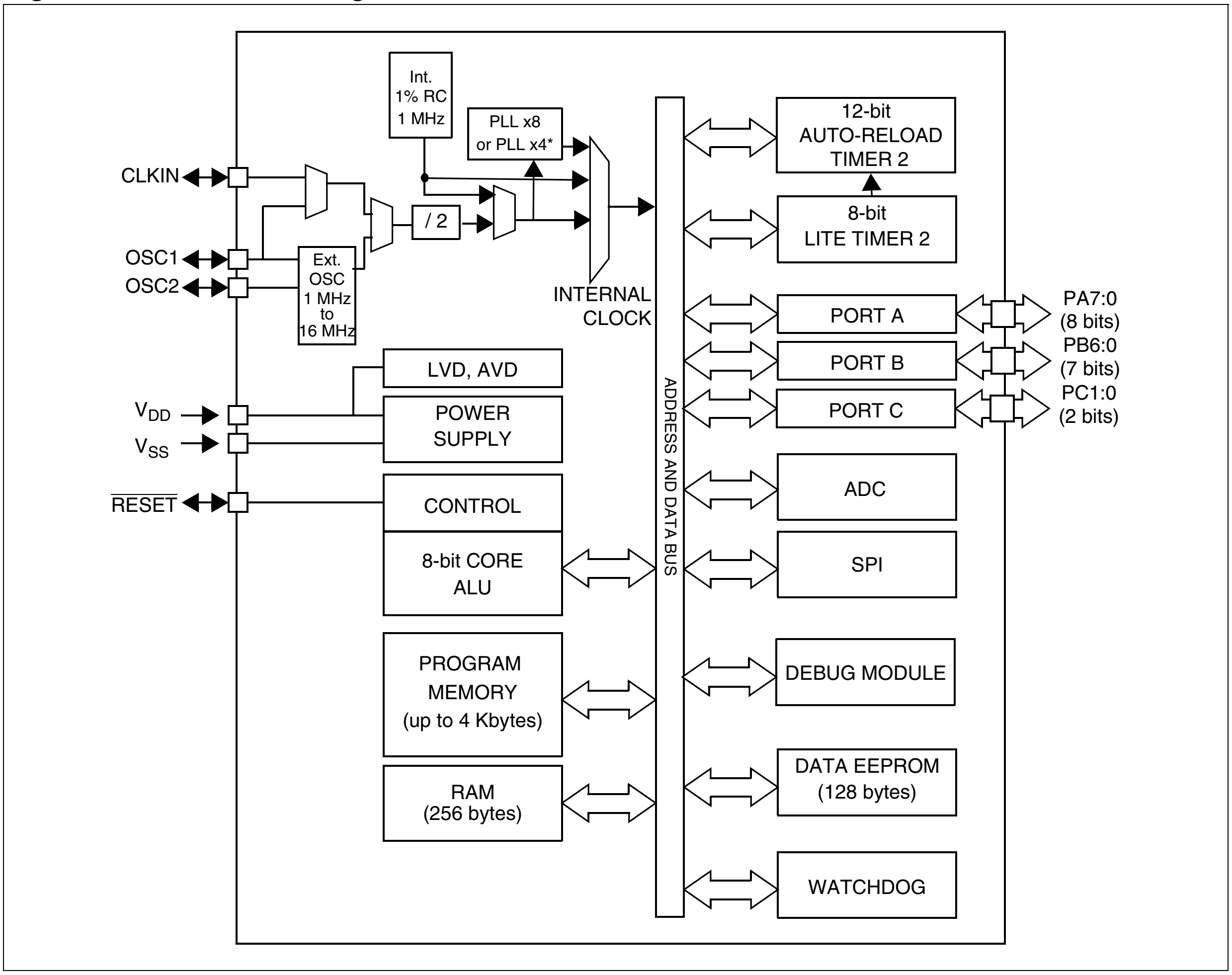
The ST7L1 features FLASH memory with byte-by-byte In-Circuit Programming (ICP) and In-Application Programming (IAP) capability.

Under software control, the ST7L1 device can be placed in WAIT, SLOW or HALT mode, reducing power consumption when the application is in idle or standby state.

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

For easy reference, all parametric data is located in section 13 on page 98. The ST7L1 features an on-chip Debug Module (DM) to support In-Circuit Debugging (ICD). For a description of the DM registers, refer to the *ST7 ICC Protocol Reference Manual*.

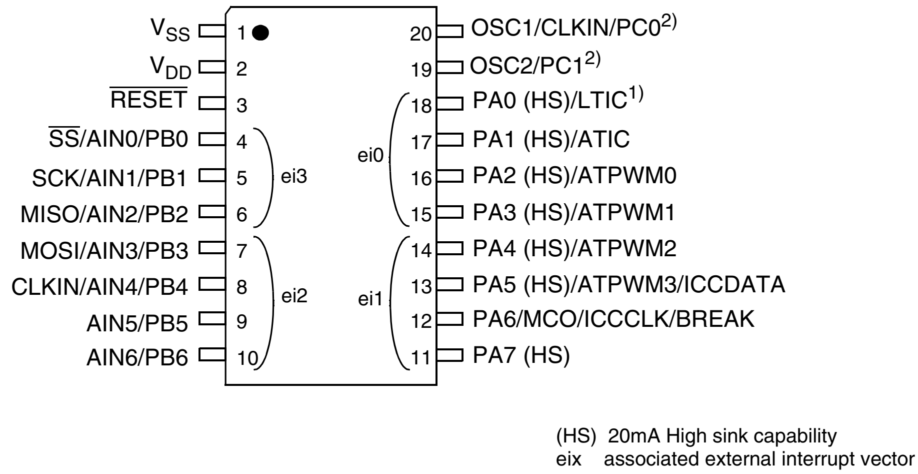
Figure 1. General Block Diagram



***Note**: Not available on ROM devices.

2 PIN DESCRIPTION

Figure 2. 20-Pin SO Package Pinout



Notes:
1. This pin cannot be configured as external interrupt in ROM devices.
2. OSC1 and OSC2 are not multiplexed in ROM devices and Port C is not present.

Legend / Abbreviations for Table 1:

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

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

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

Port and control configuration:
- Input: float = floating, wpu = weak pull-up, int = interrupt, ana = analog
- Output: OD = open drain, PP = push-pull

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

Table 1. Device Pin Description

Pin No.	Pin Name	Type	Level		Port / Control						Main Function (after reset)	Alternate Function
SO20			Input	Output	Input				Output			
					float	wpu	int	ana	OD	PP		
1	V_{SS}	S									Ground	
2	V_{DD}	S									Main power supply	
3	RESET	I/O	C_T			**X**			X		Top priority non maskable interrupt (active low)	

Pin No.	Pin Name	Type	Level		Port / Control						Main Function (after reset)	Alternate Function
SO20			Input	Output	Input				Output			
					float	wpu	int	ana	OD	PP		
4	PB0/AIN0/$\overline{SS}$	I/O	C_T		**X**	ei3		X	X	X	**Port B0**	ADC Analog Input 0 or SPI Slave Select (active low) **Caution:** No negative current injection allowed on this pin.
5	PB1/AIN1/SCK	I/O	C_T		**X**			X	X	X	**Port B1**	ADC Analog Input 1 or SPI Serial Clock
6	PB2/AIN2/MISO	I/O	C_T		**X**			X	X	X	**Port B2**	ADC Analog Input 2 or SPI Master In/ Slave Out Data
7	PB3/AIN3/MOSI	I/O	C_T		**X**	ei2		X	X	X	**Port B3**	ADC Analog Input 3 or SPI Master Out / Slave In Data
8[1)]	PB4/AIN4/CLKIN/ COMPIN-	I/O	C_T		**X**			X	X	X	**Port B4**	ADC Analog Input 4 or External clock input
9[1)]	PB5/AIN5	I/O	C_T		**X**			X	X	X	**Port B5**	ADC Analog Input 5
10[1)]	PB6/AIN6	I/O	C_T		**X**			X	X	X	**Port B6**	ADC Analog Input 6
11[1)]	PA7	I/O	C_T	HS	**X**	ei1			X	X	**Port A7**	
12	PA6 /MCO/ ICCCLK/BREAK	I/O	C_T		X	**ei1**			X	X	**Port A6**	Main Clock Output or In Circuit Communication Clock or External BREAK **Caution:** During normal operation this pin must be pulled- up, internally or externally (external pull-up of 10k mandatory in noisy environment). This is to avoid entering ICC mode unexpectedly during a reset. In the application, even if the pin is configured as output, any reset puts it back in input pull-up
13	PA5 /ICCDATA/ ATPWM3	I/O	C_T	HS	**X**	ei1			X	X	**Port A5**	In Circuit Communication Data or Auto-Reload Timer PWM3
14	PA4/ATPWM2	I/O	C_T	HS	**X**				X	X	**Port A4**	Auto-Reload Timer PWM2
15	PA3/ATPWM1	I/O	C_T	HS	**X**	ei0			X	X	**Port A3**	Auto-Reload Timer PWM1
16	PA2/ATPWM0	I/O	C_T	HS	**X**				X	X	**Port A2**	Auto-Reload Timer PWM0
17	PA1/ATIC	I/O	C_T	HS	**X**				X	X	**Port A1**	Auto-Reload Timer Input Capture
18[1)]	PA0/LTIC	I/O	C_T	HS	**X**				X	X	**Port A0**	Lite Timer Input Capture
19[2)]	OSC2/PC1	I/O			**X**					X	**Port C1**[3)]	Resonator oscillator inverter output
20[2)]	OSC1/CLKIN/PC0	I/O			**X**					X	**Port C0**[3)]	Resonator oscillator inverter input or External clock input

Notes:
1. This pin cannot be configured as external interrupt in ROM devices.
2. OSC1 and OSC2 are not multiplexed in ROM devices and Port C is not present.
3. PCOR not implemented but p-transistor always active in output mode (refer to Figure 30 on page 45)

3 REGISTER AND MEMORY MAP

As shown in Figure 3, the MCU can address 64 Kbytes of memories and I/O registers.

The available memory locations consist of 128 bytes of register locations, 256 bytes of RAM, 128 bytes of data EEPROM and up to 4 Kbytes of Flash program memory. The RAM space includes up to 128 bytes for the stack from 180h to 1FFh.

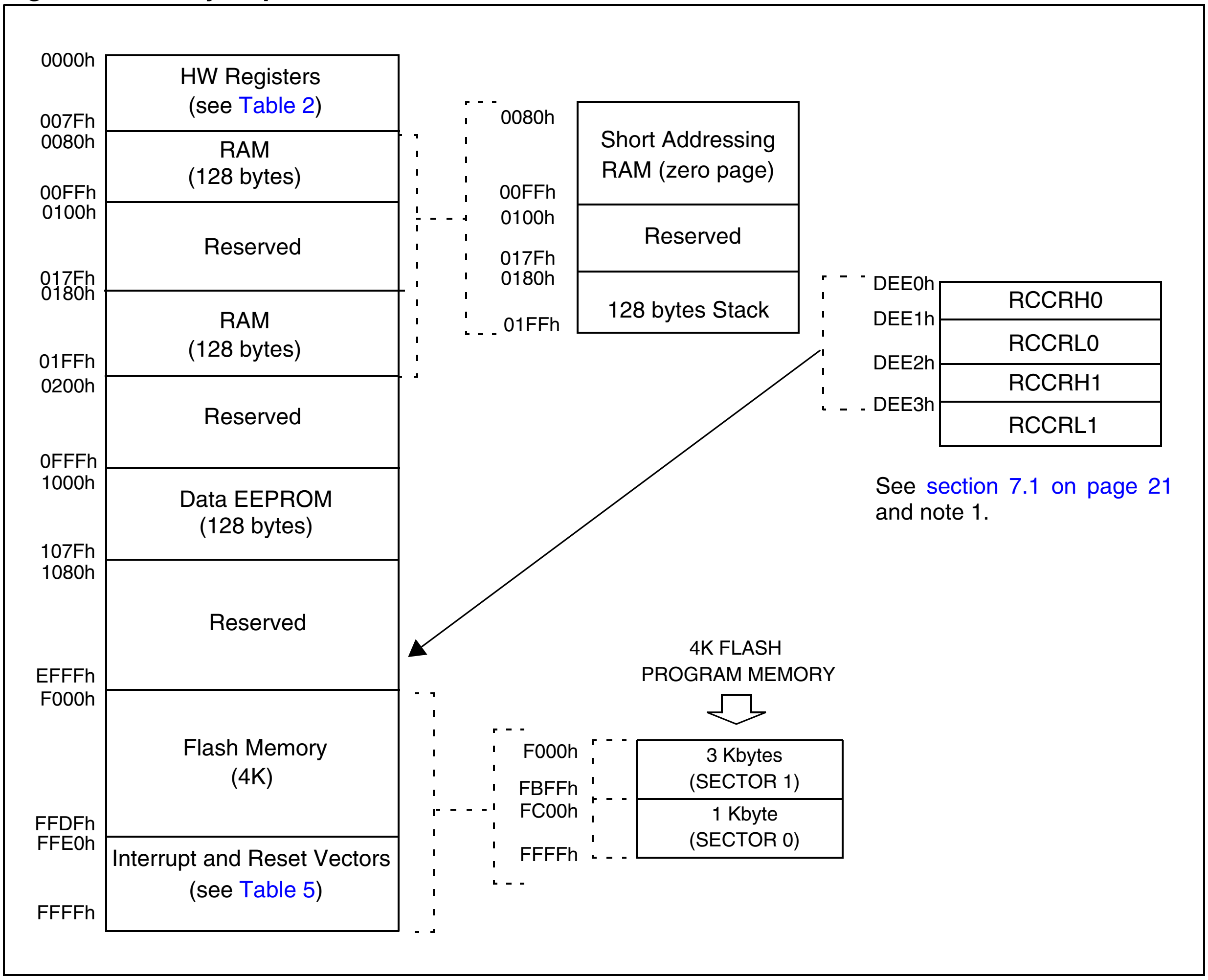
The highest address bytes contain the user reset and interrupt vectors.

The Flash memory contains two sectors (see Figure 3) mapped in the upper part of the ST7 addressing space so the reset and interrupt vectors are located in Sector 0 (F000h-FFFFh).

The size of Flash Sector 0 and other device options are configurable by Option byte (refer to section 15.1 on page 126).

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

Figure 3. Memory Map



Notes:

1. DEE0h, DEE1h, DEE2h and DEE3h addresses are located in a reserved area but are special bytes containing also the RC calibration values which are read-accessible only in user mode. If all the EEPROM data or Flash space (including the RC calibration values locations) has been erased (after the read out protection removal), then the RC calibration values can still be obtained through these four addresses.

REGISTER AND MEMORY MAP (Cont'd)

Table 2. Hardware Register Map

Address	Block	Register Label	Register Name	Reset Status	Remarks
0000h 0001h 0002h	Port A	PADR PADDR PAOR	Port A Data Register Port A Data Direction Register Port A Option Register	FFh[1)] 00h 40h	R/W R/W R/W
0003h 0004h 0005h	Port B	PBDR PBDDR PBOR	Port B Data Register Port B Data Direction Register Port B Option Register	FFh[1)] 00h 00h	R/W R/W R/W[2)]
0006h 0007h	Port C	PCDR PCDDR	Port C Data Register Port C Data Direction Register	0xh 00h	R/W R/W
0008h 0009h 000Ah 000Bh 000Ch	LITE TIMER 2	LTCSR2 LTARR LTCNTR LTCSR1 LTICR	Lite Timer Control/Status Register 2 Lite Timer Auto-reload Register Lite Timer Counter Register Lite Timer Control/Status Register 1 Lite Timer Input Capture Register	00h 00h 00h 0x00 0000b xxh	R/W R/W Read Only R/W Read Only
000Dh 000Eh 000Fh 0010h 0011h 0012h 0013h 0014h 0015h 0016h 0017h 0018h 0019h 001Ah 001Bh 001Ch 001Dh 001Eh 001Fh 0020h 0021h 0022h 0023h 0024h 0025h 0026h	AUTO-RELOAD TIMER 2	ATCSR CNTRH CNTRL ATRH ATRL PWMCR PWM0CSR PWM1CSR PWM2CSR PWM3CSR DCR0H DCR0L DCR1H DCR1L DCR2H DCR2L DCR3H DCR3L ATICRH ATICRL ATCSR2 BREAKCR ATR2H ATR2L DTGR BREAKEN	Timer Control/Status Register Counter Register High Counter Register Low Auto-Reload Register High Auto-Reload Register Low PWM Output Control Register PWM 0 Control/Status Register PWM 1 Control/Status Register PWM 2 Control/Status Register PWM 3 Control/Status Register PWM 0 Duty Cycle Register High PWM 0 Duty Cycle Register Low PWM 1 Duty Cycle Register High PWM 1 Duty Cycle Register Low PWM 2 Duty Cycle Register High PWM 2 Duty Cycle Register Low PWM 3 Duty Cycle Register High PWM 3 Duty Cycle Register Low Input Capture Register High Input Capture Register Low Timer Control/Status Register 2 Break Control Register Auto-Reload Register 2 High Auto-Reload Register 2 Low Dead Time Generation Register Break Enable Register	0x00 0000b 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 03h 00h 00h 00h 00h 03h	R/W Read Only Read Only R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W R/W Read Only Read Only R/W R/W R/W R/W R/W R/W
0027h to 002Dh	Reserved area (7 bytes)				
002Eh	WDG	WDGCR	Watchdog Control Register	7Fh	R/W
0002Fh	FLASH	FCSR	Flash Control/Status Register	00h	R/W
00030h	EEPROM	EECSR	Data EEPROM Control/Status Register	00h	R/W

Address	Block	Register Label	Register Name	Reset Status	Remarks
0031h 0032h 0033h	SPI	SPIDR SPICR SPICSR	SPI Data I/O Register SPI Control Register SPI Control Status Register	xxh 0xh 00h	R/W R/W R/W
0034h 0035h 0036h	ADC	ADCCSR ADCDRH ADCDRL	A/D Control Status Register A/D Data Register High Data Low Register	00h xxh 0xh	R/W Read Only R/W
0037h	ITC	EICR	External Interrupt Control Register	00h	R/W
0038h	MCC	MCCSR	Main Clock Control/Status Register	00h	R/W
0039h 003Ah	Clock and Reset	RCCR SICSR	RC oscillator Control Register System Integrity Control/Status Register	FFh 0110 0xx0b	R/W R/W
003Bh	PLL clock select	PLLTST	PLL test register	00h	R/W
003Ch	ITC	EISR	External Interrupt Selection Register	0Ch	R/W
003Dh to 0048h	Reserved area (12 bytes)				
0049h 004Ah	AWU	AWUPR AWUCSR	AWU Prescaler Register AWU Control/Status Register	FFh 00h	R/W R/W
004Bh 004Ch 004Dh 004Eh 004Fh 0050h 0051h	DM[3)]	DMCR DMSR DMBK1H DMBK1L DMBK2H DMBK2L DMCR2	DM Control Register DM Status Register DM Breakpoint Register 1 High DM Breakpoint Register 1 Low DM Breakpoint Register 2 High DM Breakpoint Register 2 Low DM Control Register 2	00h 00h 00h 00h 00h 00h 00h	R/W R/W R/W R/W R/W R/W R/W
0052h to 007Fh	Reserved area (46 bytes)				

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

Notes:

1. The contents of the I/O port DR registers are readable only in output configuration. In input configuration, the values of the I/O pins are returned instead of the DR register contents.
2. The bits associated with unavailable pins must always keep their reset value.
3. For a description of the Debug Module registers, see *ST7 ICC Protocol Reference Manual*.

4 FLASH PROGRAM MEMORY

4.1 INTRODUCTION

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

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

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

4.2 MAIN FEATURES

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

4.3 PROGRAMMING MODES

The ST7 can be programmed in three different ways:

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

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

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

4.3.1 In-Circuit Programming (ICP)

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

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

– Download ICP Driver code in RAM from the ICCDATA pin

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

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

4.3.2 In-Application Programming (IAP)

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

IAP mode is fully controlled by user software, allowing it to be adapted to the user application (such as a user-defined strategy for entering programming mode or a choice of communications protocol used to fetch the data to be stored). This mode can be used to program any memory areas except Sector 0, which is write/erase protected to allow recovery in case errors occur during the programming operation.

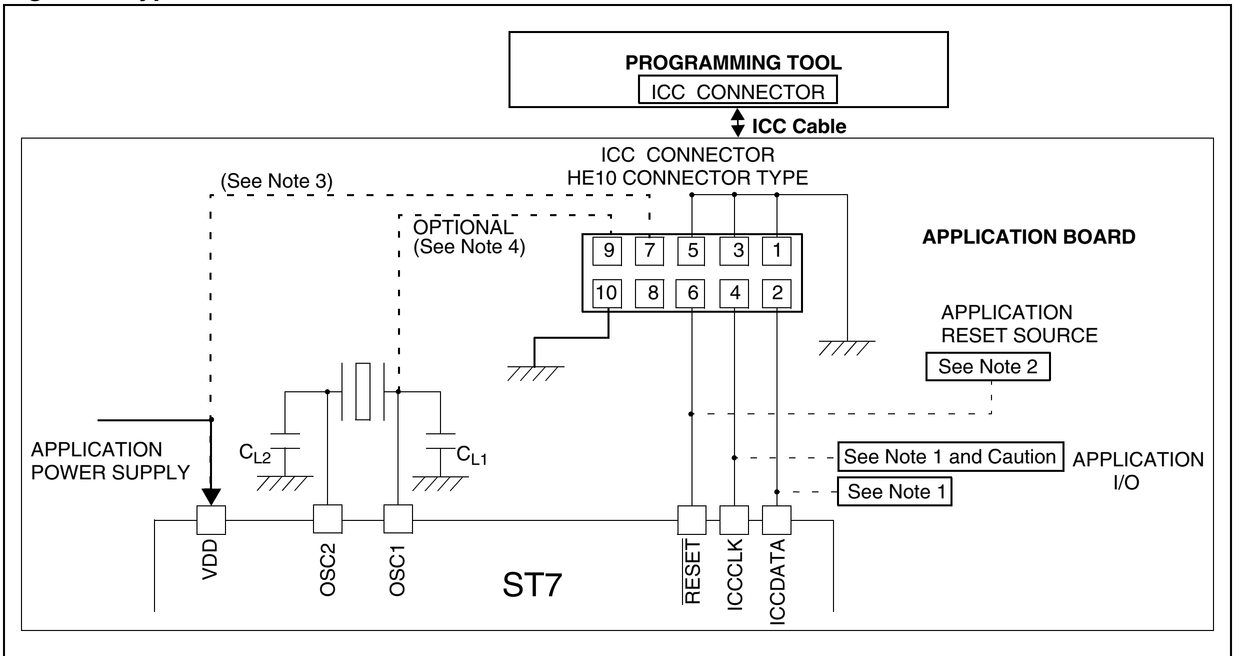
FLASH PROGRAM MEMORY (Cont'd)

4.4 ICC INTERFACE

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

– RESET: Device reset
– V_{SS}: Device power supply ground
– ICCCLK: ICC output serial clock pin
– ICCDATA: ICC input serial data pin
– OSC1: Main clock input for external source (not required on devices without OSC1/OSC2 pins)
– V_{DD}: Application board power supply (optional, see Note 3)

Figure 4. Typical ICC Interface



Notes:

1. If the ICCCLK or ICCDATA pins are only used as outputs in the application, no signal isolation is necessary. As soon as the Programming Tool is plugged to the board, even if an ICC session is not in progress, the ICCCLK and ICCDATA pins are not available for the application. If they are used as inputs by the application, isolation such as a serial resistor must be implemented in case another device forces the signal. Refer to the Programming Tool documentation for recommended resistor values.
2. During the ICP session, the programming tool must control the RESET pin. This can lead to conflicts between the programming tool and the application reset circuit if it drives more than 5mA at high level (push-pull output or pull-up resistor < 1K). A schottky diode can be used to isolate the application RESET circuit in this case. When using a classical RC network with R > 1K or a reset management IC with open drain output and pull-up resistor > 1K, no additional components are needed. In all cases the user must ensure that no external reset is generated by the application during the ICC session.
3. The use of pin 7 of the ICC connector depends on the Programming Tool architecture. This pin must be connected when using most ST Programming Tools (it is used to monitor the application power supply). Please refer to the *Programming Tool Manual*.
4. Pin 9 must be connected to the OSC1 pin of the ST7 when the clock is not available in the application or if the selected clock option is not programmed in the option byte. On ST7 devices with multi-oscillator capability, OSC2 must be grounded in this case.
5. In 38-pulse ICC mode, the internal RC oscillator is forced as a clock source, regardless of the selection in the option byte. For ST7L1 devices which do not support the internal RC oscillator, the “option byte disabled” mode must be used (35-pulse ICC mode entry, clock provided by the tool).

Caution: During normal operation the ICCCLK pin must be pulled up, internally or externally (external pull-up of 10k mandatory in noisy environment). This is to avoid entering ICC mode unexpectedly during a reset. In the application, even if the pin is configured as output, any reset puts it back in input pull-up.

FLASH PROGRAM MEMORY (Cont'd)

4.5 MEMORY PROTECTION

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

4.5.1 Read-out Protection

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

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

Read-out protection selection depends on the device type:

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

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

4.5.2 Flash Write/Erase Protection

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

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

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

4.6 RELATED DOCUMENTATION

For details on Flash programming and ICC protocol, refer to the *ST7 Flash Programming Reference Manual* and to the *ST7 ICC Protocol Reference Manual*.

4.7 REGISTER DESCRIPTION

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

7							0
0	0	0	0	0	OPT	LAT	PGM

Note: This register is reserved for programming using ICP, IAP or other programming methods. It controls the XFlash programming and erasing operations.

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

5 DATA EEPROM

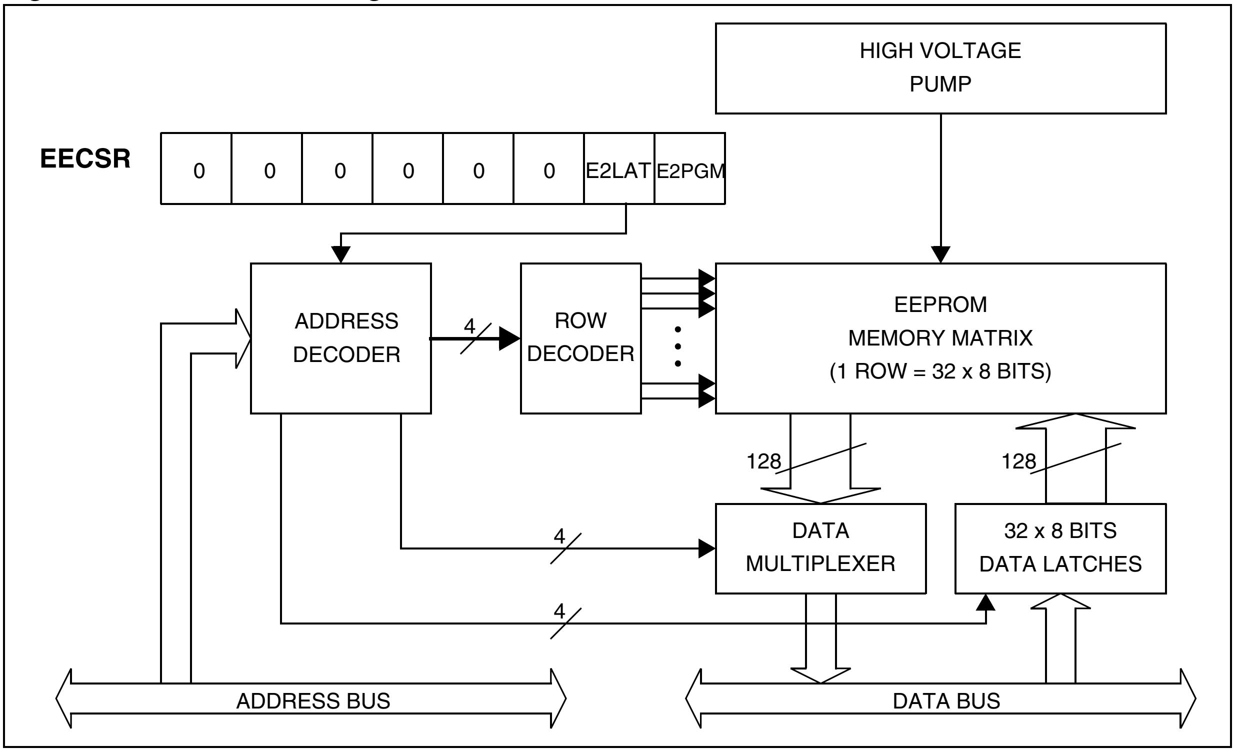
5.1 INTRODUCTION

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

5.2 MAIN FEATURES

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

Figure 5. EEPROM Block Diagram



DATA EEPROM (Cont'd)

5.3 MEMORY ACCESS

The Data EEPROM memory read/write access modes are controlled by the E2LAT bit of the EEPROM Control/Status register (EECSR). The flowchart in Figure 6 describes these different memory access modes.

Read Operation (E2LAT = 0)

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

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

Write Operation (E2LAT = 1)

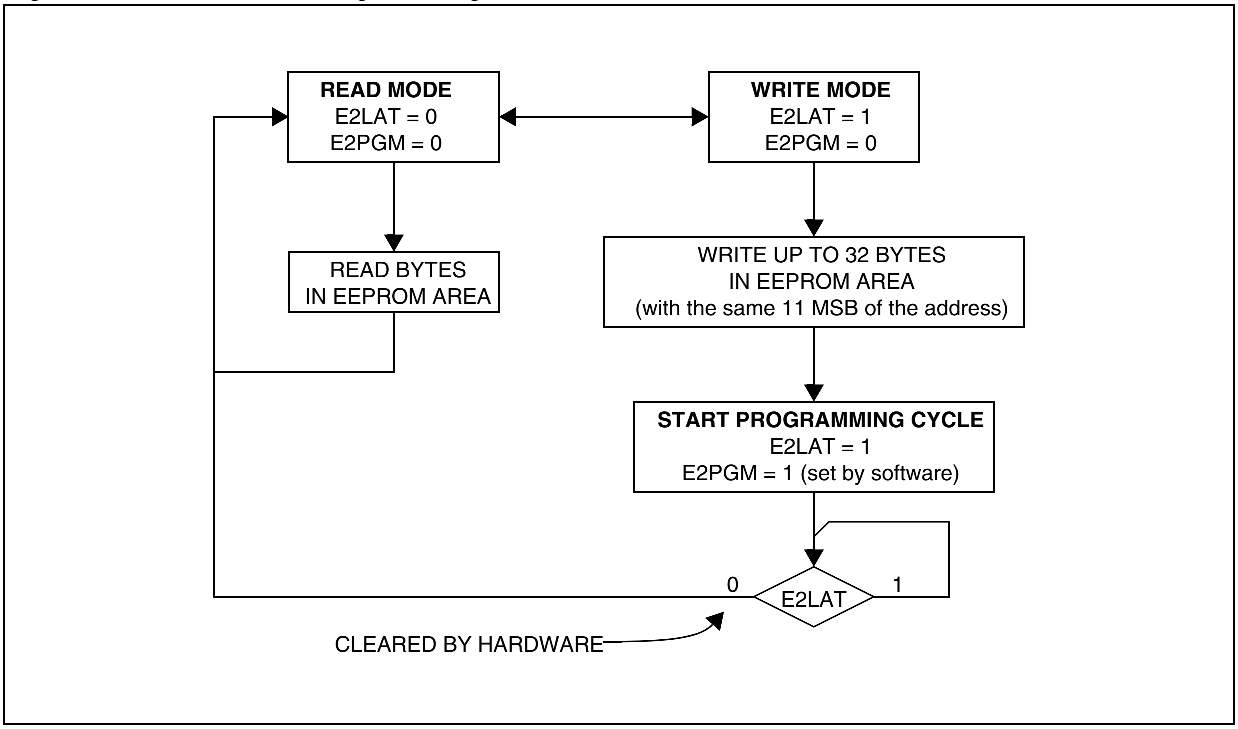
To access the write mode, the E2LAT bit must be set by software (the E2PGM bit remains cleared). When a write access to the EEPROM area occurs, the value is latched inside the 32 data latches according to its address.

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

At the end of the programming cycle, the PGM and LAT bits are cleared simultaneously.

Note: Care should be taken during the programming cycle. Writing to the same memory location over-programs the memory (logical AND between the two write access data results) because the data latches are only cleared at the end of the programming cycle and by the falling edge of the E2LAT bit.
It is not possible to read the latched data.
This note is illustrated by the Figure 8 on page 16.

Figure 6. Data EEPROM Programming Flowchart

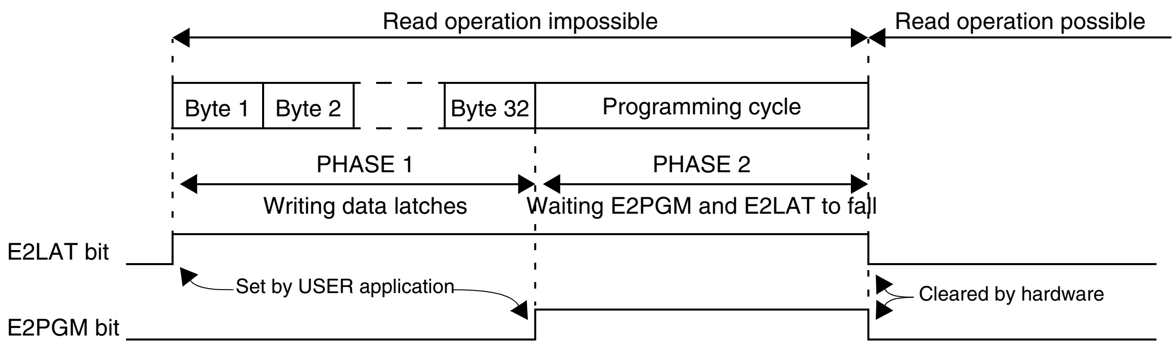


DATA EEPROM (Cont'd)

Figure 7. Data EEPROM Write Operation

ROW DEFINITION

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



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

DATA EEPROM (Cont'd)

5.4 POWER SAVING MODES

Wait mode

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

Active Halt mode

Refer to WAIT mode.

Halt mode

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

5.5 ACCESS ERROR HANDLING

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

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

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

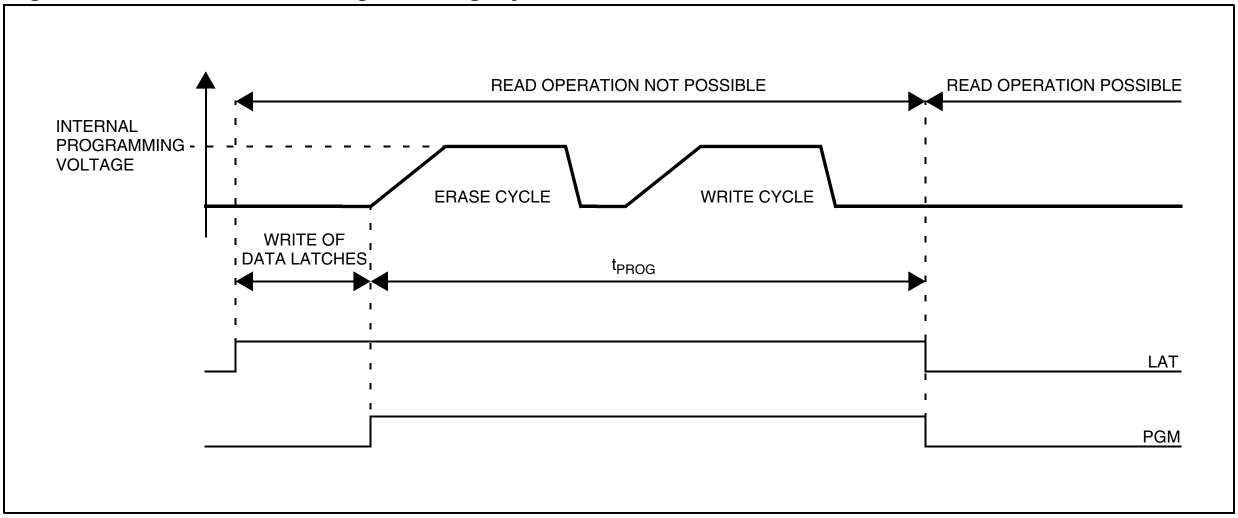
5.6 DATA EEPROM READ-OUT PROTECTION

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

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

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

Figure 8. Data EEPROM Programming Cycle



DATA EEPROM (Cont'd)

5.7 REGISTER DESCRIPTION

EEPROM CONTROL/STATUS REGISTER (EECSR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	E2LAT	E2PGM

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

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

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

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

Table 3. DATA EEPROM Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0030h	**EECSR** Reset Value	0	0	0	0	0	0	E2LAT 0	E2PGM 0

6 CENTRAL PROCESSING UNIT

6.1 INTRODUCTION

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

6.2 MAIN FEATURES

- 63 basic instructions
- Fast 8-bit by 8-bit multiply
- 17 main addressing modes
- Two 8-bit index registers
- 16-bit stack pointer
- Low power modes
- Maskable hardware interrupts
- Non-maskable software interrupt

6.3 CPU REGISTERS

The six CPU registers shown in Figure 9 are not present in the memory mapping and are accessed by specific instructions.

Accumulator (A)

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

Index Registers (X and Y)

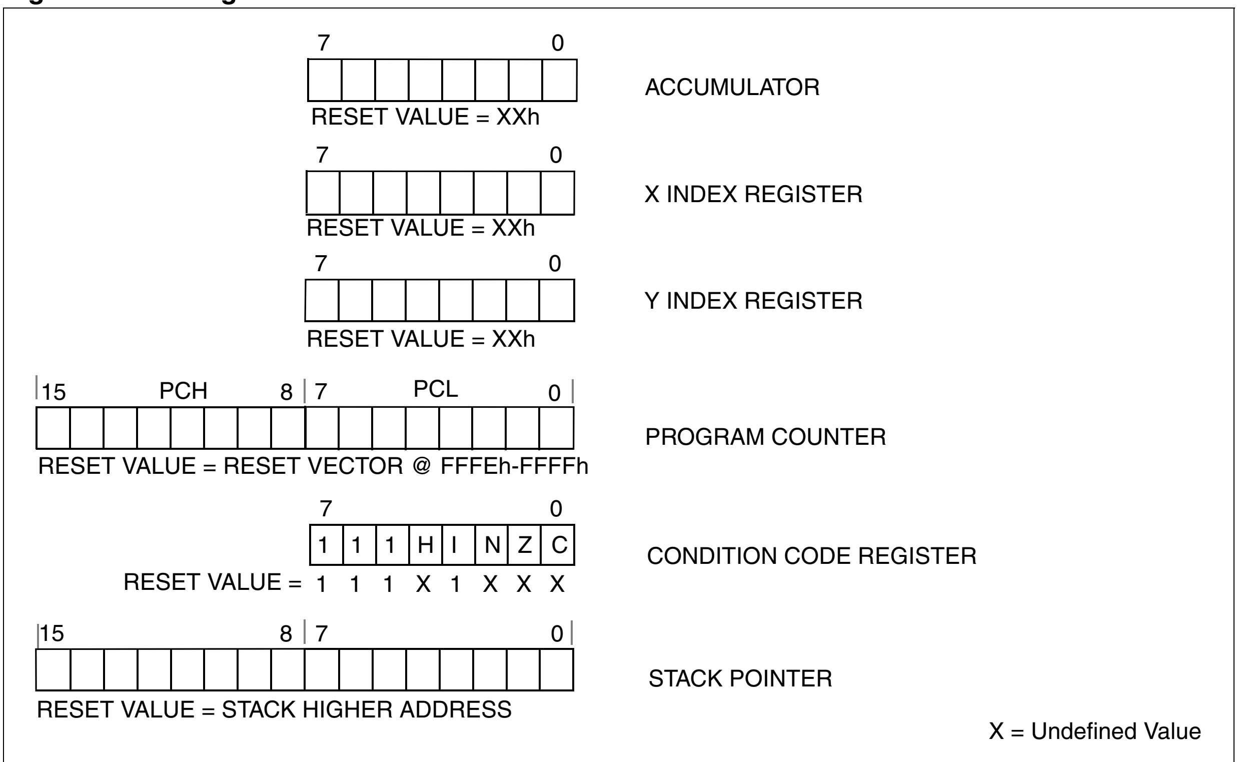
In indexed addressing modes, these 8-bit registers are used to create either effective addresses or temporary storage areas for data manipulation. (The Cross-Assembler generates a precede instruction (PRE) to indicate that the following instruction refers to the Y register.)

The Y register is not affected by the interrupt automatic procedures (not pushed to and popped from the stack).

Program Counter (PC)

The program counter is a 16-bit register containing the address of the next instruction to be executed by the CPU. It is made of two 8-bit registers PCL (Program Counter Low which is the LSB) and PCH (Program Counter High which is the MSB).

Figure 9. CPU Registers



CPU REGISTERS (Cont'd)

CONDITION CODE REGISTER (CC)

Read/Write

Reset Value: 111x1xxx

7							0
1	1	1	H	I	N	Z	C

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

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

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

This bit is set by hardware when a carry occurs between bits 3 and 4 of the ALU during an ADD or ADC instruction. It is reset by hardware during the same instructions.
0: No half carry has occurred.
1: A half carry has occurred.

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

Bit 3 = **I** *Interrupt mask*.

This bit is set by hardware when entering in interrupt or by software to disable all interrupts except the TRAP software interrupt. This bit is cleared by software.
0: Interrupts are enabled.
1: Interrupts are disabled.

This bit is controlled by the RIM, SIM and IRET instructions and is tested by the JRM and JRNM instructions.

Note: Interrupts requested while I is set are latched and can be processed when I is cleared. By default an interrupt routine is not interruptible because the I bit is set by hardware at the start of the routine and reset by the IRET instruction at the end of the routine. If the I bit is cleared by software in the interrupt routine, pending interrupts are serviced regardless of the priority level of the current interrupt routine.

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

This bit is set and cleared by hardware. It is representative of the result sign of the last arithmetic, logical or data manipulation. It is a copy of the 7th bit of the result.
0: The result of the last operation is positive or null.
1: The result of the last operation is negative (that is, the most significant bit is a logic 1).

This bit is accessed by the JRMI and JRPL instructions.

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

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

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

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

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

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

CPU REGISTERS (Cont'd)

STACK POINTER (SP)

Read/Write

Reset Value: 01FFh

15							8
0	0	0	0	0	0	0	1
7							0
1	SP6	SP5	SP4	SP3	SP2	SP1	SP0

The Stack Pointer is a 16-bit register which always points 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 10).

Since the stack is 128 bytes deep, the 9 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 SP6 to SP0 bits are set) which is the stack higher address.

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

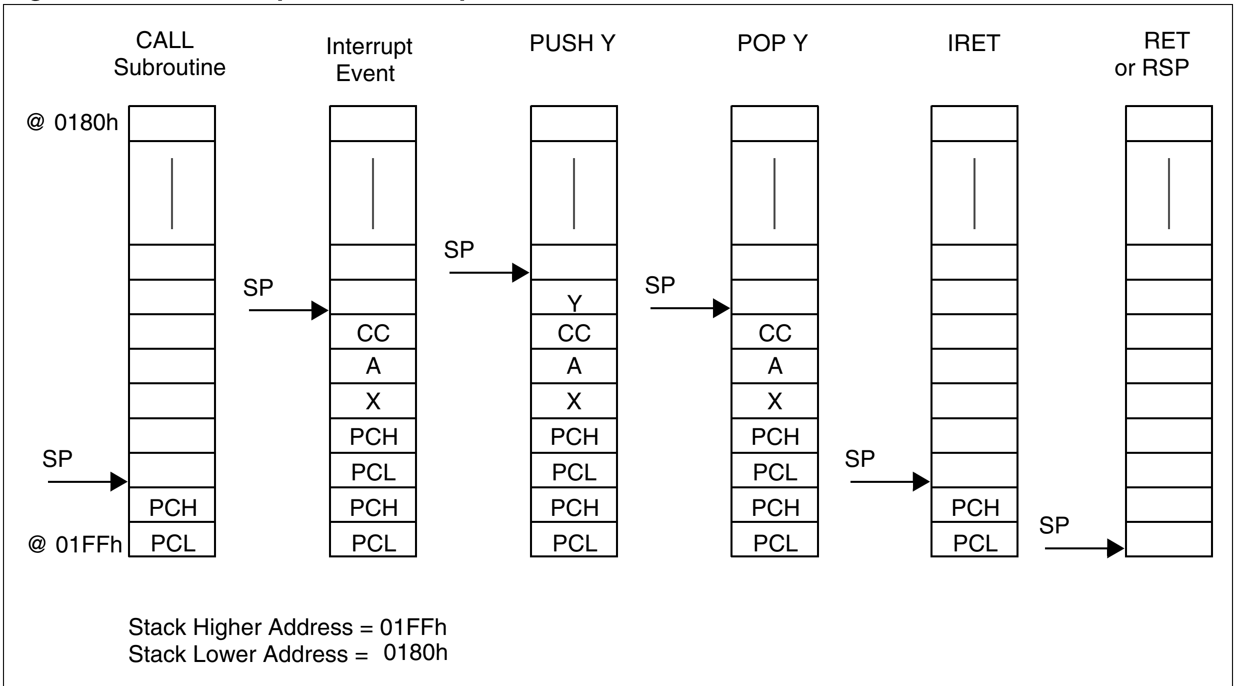
The stack is used to save the return address during a subroutine call and the CPU context during an interrupt. The user may also directly manipulate the stack by means of the PUSH and POP instructions. In the case of an interrupt, the PCL is stored at the first location pointed to by the SP. Then the other registers are stored in the next locations as shown in Figure 10.

– 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 occupies five locations in the stack area.

Figure 10. Stack Manipulation Example



7 SUPPLY, RESET AND CLOCK MANAGEMENT

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

Main features

- Clock Management
  - 1 MHz internal RC oscillator (enabled by option byte
  - 1 to 16 MHz External crystal/ceramic resonator (selected by option byte)
  - External Clock Input (enabled by option byte)
  - PLL for multiplying the frequency by 8 or 4 (enabled by option byte). Only multiplying by 8 is available for ROM devices.
- Reset Sequence Manager (RSM)
- System Integrity Management (SI)
  - main supply Low Voltage Detection (LVD) with reset generation (enabled by option byte)
  - Auxiliary Voltage Detector (AVD) with interrupt capability for monitoring the main supply (enabled by option byte)

7.1 INTERNAL RC OSCILLATOR ADJUSTMENT

The device contains an internal RC oscillator with an accuracy of 1% for a given device, temperature and voltage range (4.5V to 5.5V). It must be calibrated to obtain the frequency required in the application. This is done by the software writing a 10-bit calibration value in the RCCR (RC Control Register) and in the bits 6:5 in the SICSR (SI Control Status Register).

Whenever the microcontroller is reset, the RCCR returns to its default value (FFh), that is, each time the device is reset, the calibration value must be loaded in the RCCR. Predefined calibration values are stored in EEPROM for 3.3V and 5V V_{DD} supply voltages at 25°C, as shown in the following table.

RCCR	Conditions	ST7L1 Address
RCCRH0	V_{DD} = 5V T_A = 25°C f_{RC} = 1 MHz	DEE0h[1)] (CR[9:2])
RCCRL0		DEE1h[1)] (CR[1:0])
RCCRH1	V_{DD} = 3.3V T_A = 25°C f_{RC} = 1 MHz	DEE2h[1)] (CR[9:2])
RCCRL1		DEE3h[1)] (CR[1:0])

Note:

1. DEE0h, DEE1h, DEE2h, and DEE3h addresses are located in a reserved area but are special bytes containing also the RC calibration values which are read-accessible only in user mode. If all the EEPROM data or Flash space (including the RC calibration value locations) has been erased (after the read-out protection removal), then the RC calibration values can still be obtained through these four addresses.
For compatibility reasons with the SICSR register, CR[1:0] bits are stored in the fifth and sixth position of the DEE1 and DEE3 addresses.

Notes:

– In 38-pulse ICC mode, the internal RC oscillator is forced as a clock source, regardless of the selection in the option byte. For ST7L1 devices which do not support the internal RC oscillator, the “option byte disabled” mode must be used (35-pulse ICC mode entry, clock provided by the tool).

– For more information on the frequency and accuracy of the RC oscillator see “ELECTRICAL CHARACTERISTICS” on page 98.

– To improve clock stability and frequency accuracy, it is recommended to place a decoupling capacitor, typically 100nF, between the V_{DD} and V_{SS} pins as close as possible to the ST7 device.

– These bytes are systematically programmed by ST, including on FASTROM devices.

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

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

SUPPLY, RESET AND CLOCK MANAGEMENT (Cont'd)

7.2 PHASE LOCKED LOOP

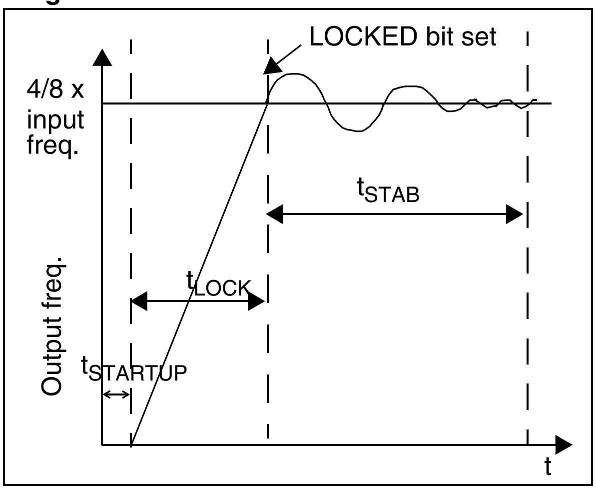
The PLL can be used to multiply a 1 MHz frequency from the RC oscillator or the external clock by 4 or 8 to obtain f_{OSC} of 4 or 8 MHz. The PLL is enabled and the multiplication factor of 4 or 8 is selected by 2 option bits:

– The x4 PLL is intended for operation with V_{DD} in the 3V to 3.6V range (available only on FLASH devices)

– The x8 PLL is intended for operation with V_{DD} in the 3.6V to 5.5V range[1)]

Refer to Section 15.1 for the option byte description.
If the PLL is disabled and the RC oscillator is enabled, then f_{OSC} = 1 MHz.
If both the RC oscillator and the PLL are disabled, f_{OSC} is driven by the external clock.

Figure 11. PLL Output Frequency Timing Diagram



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

When the PLL output signal reaches the operating frequency, the LOCKED bit in the SICSCR register is set. Full PLL accuracy (ACC_{PLL}) is reached after a stabilization time of t_{STAB} (see Figure 11 and section 13.3.4 on page 105)

Refer to section 7.6.4 on page 30 for a description of the LOCKED bit in the SICSR register.

Note:

1. It is possible to obtain f_{OSC} = 4 MHz in the 3.3V to 5.5V range with internal RC and PLL enabled by selecting 1 MHz RC and x8 PLL and setting the PLLdiv2 bit in the PLLTST register (see section 7.6.4 on page 30).

7.3 REGISTER DESCRIPTION

MAIN CLOCK CONTROL/STATUS REGISTER (MCCSR)
Read / Write
Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	MCO	SMS

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

Bit 1 = **MCO** *Main Clock Out enable*
This bit is read/write by software and cleared by hardware after a reset. This bit enables the MCO output clock.
0: MCO clock disabled, I/O port free for general purpose I/O.
1: MCO clock enabled.

Bit 0 = **SMS** *Slow Mode select*
This bit is read/write by software and cleared by hardware after a reset. This bit selects the input clock f_{OSC} or $f_{OSC}/32$.
0: Normal mode ($f_{CPU} = f_{OSC}$)
1: Slow mode ($f_{CPU} = f_{OSC}/32$)

RC CONTROL REGISTER (RCCR)
Read / Write
Reset Value: 1111 1111 (FFh)

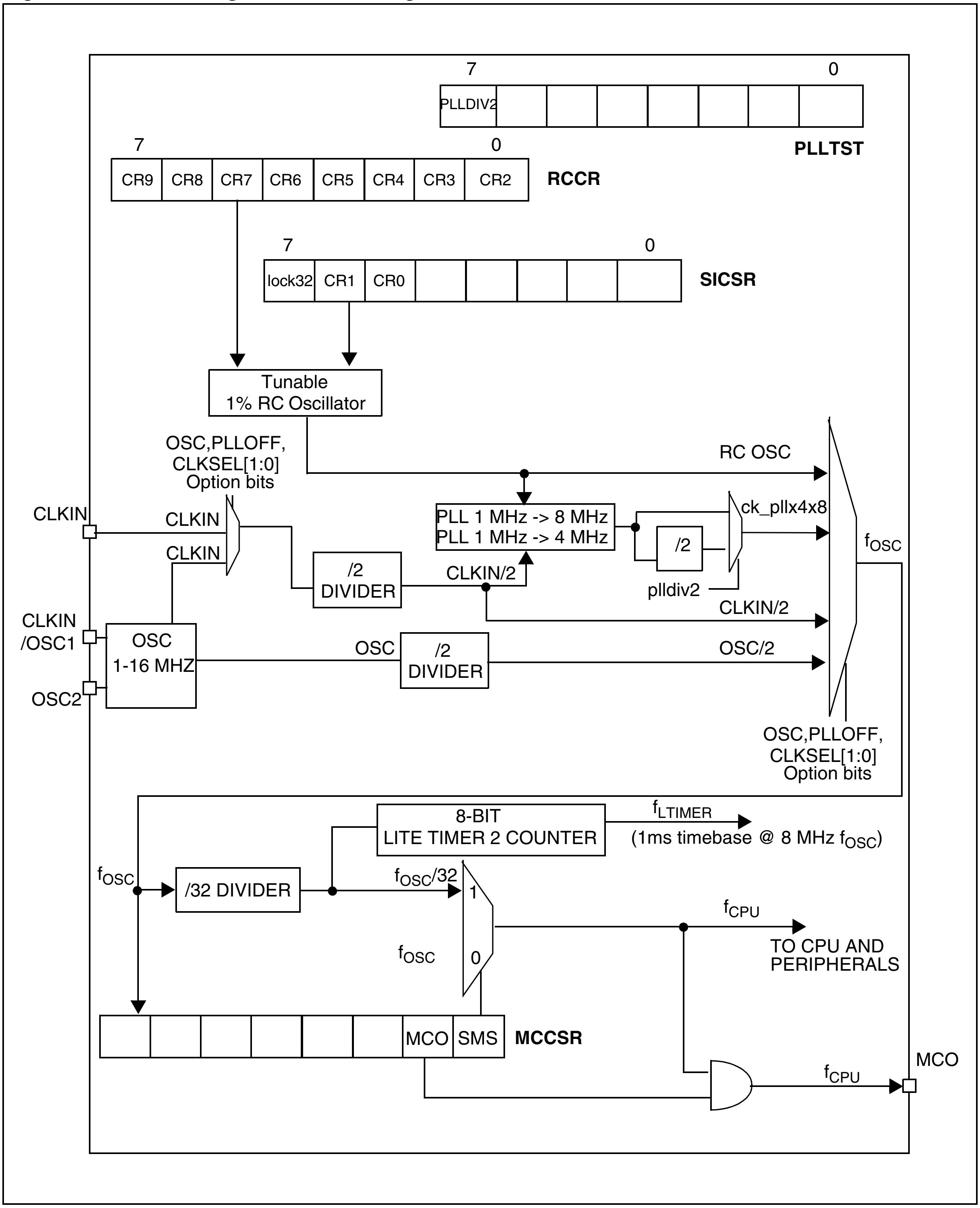
7							0
CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2

Bits 7:0 = **CR[9:2]** *RC Oscillator Frequency Adjustment Bits*
These bits must be written immediately after reset to adjust the RC oscillator frequency and to obtain an accuracy of 1%. The application can store the correct value for each voltage range in EEPROM and write it to this register at start-up.
00h = maximum available frequency
FFh = lowest available frequency
These bits are used with the CR[1:0] bits in the SICSR register. Refer to section 7.6.4 on page 30.

Note: To tune the oscillator, write a series of different values in the register until the correct frequency is reached. The fastest method is to use a dichotomy starting with 80h.

SUPPLY, RESET AND CLOCK MANAGEMENT (Cont'd)

Figure 12. Clock Management Block Diagram



SUPPLY, RESET AND CLOCK MANAGEMENT (Cont'd)

7.4 MULTI-OSCILLATOR (MO)

The main clock of the ST7 can be generated by four different source types coming from the multi-oscillator block (1 to 16 MHz):

- An external source
- 5 different configurations for crystal or ceramic resonator oscillators
- An internal high frequency RC oscillator

Each oscillator is optimized for a given frequency range in terms of consumption and is selectable through the option byte. The associated hardware configurations are shown in Table 4. Refer to Section 13 ELECTRICAL CHARACTERISTICS for more details.

External Clock Source

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

Note: When the Multi-Oscillator is not used, PB4 is selected by default as the external clock.

Crystal/Ceramic Oscillators

In this mode, with a self-controlled gain feature, an oscillator of any frequency from 1 to 16 MHz can be placed on OSC1 and OSC2 pins. This family of oscillators has the advantage of producing a very accurate rate on the main clock of the ST7. In this mode of the multi-oscillator, the resonator and the load capacitors must be placed as close as possible to the oscillator pins to minimize output distortion and start-up stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.

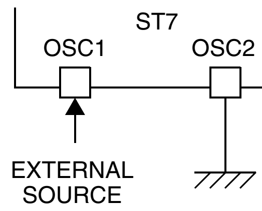
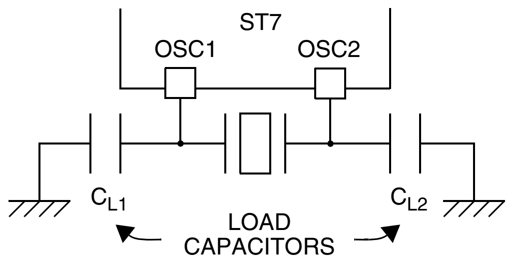
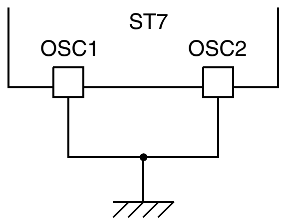
These oscillators are not stopped during the RESET phase to avoid losing time in the oscillator start-up phase.

Internal RC Oscillator

In this mode, the tunable 1% RC oscillator is the main clock source. The two oscillator pins must be tied to ground if dedicated to oscillator use, otherwise they are general purpose I/O.

The calibration is done through the RCCR[7:0] and SICSR[6:5] registers.

Table 4. ST7 Clock Sources

<table>
<tr><th></th><th>Hardware Configuration</th></tr>
<tr><td>External Clock</td><td>
</td></tr>
<tr><td>Crystal/Ceramic Resonators</td><td>
</td></tr>
<tr><td>Internal RC Oscillator</td><td>
</td></tr>
</table>

SUPPLY, RESET AND CLOCK MANAGEMENT (Cont'd)

7.5 RESET SEQUENCE MANAGER (RSM)

7.5.1 Introduction

The reset sequence manager includes three RESET sources as shown in Figure 14:

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

Note: A reset can also be triggered following the detection of an illegal opcode or prebyte code. Refer to section 12.2.1 on page 95 for further details.

These sources act on the RESET pin which 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 three phases as shown in Figure 13:

- Active Phase depending on the RESET source
- 256 or 4096 CPU clock cycle delay (see table below)
- RESET vector fetch

The 256 or 4096 CPU clock cycle delay allows the oscillator to stabilize and ensures that recovery has taken place from the RESET state. The shorter or longer clock cycle delay is automatically selected depending on the clock source chosen by option byte:

Clock Source	CPU Clock Cycle Delay
Internal RC Oscillator	256
External clock (connected to CLKIN pin)	256
External Crystal/Ceramic Oscillator (connected to OSC1/OSC2 pins)	4096

The RESET vector fetch phase duration is two clock cycles.

If the PLL is enabled by option byte, it outputs the clock after an additional delay of $t_{STARTUP}$ (see Figure 11).

Figure 13. RESET Sequence Phases

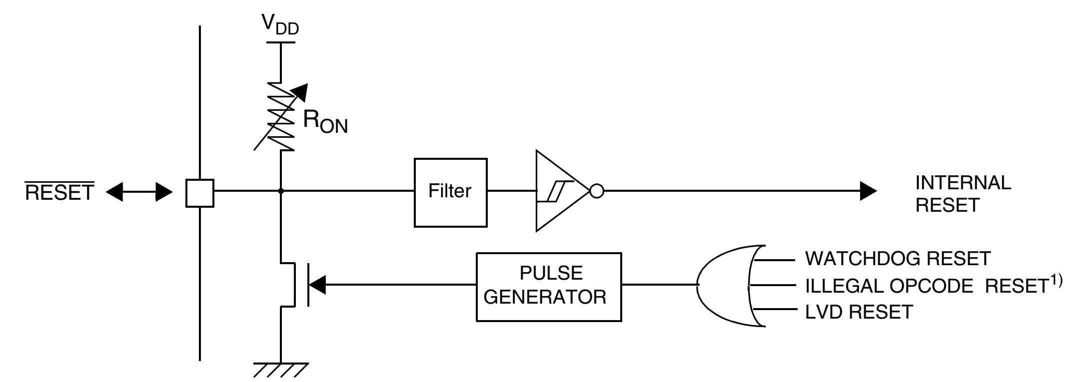
RESET		
Active Phase	INTERNAL RESET 256 or 4096 CLOCK CYCLES	FETCH VECTOR

7.5.2 Asynchronous External RESET Pin

The RESET pin is both an input and an open-drain output with integrated R_{ON} weak pull-up resistor. This pull-up has no fixed value but varies in accordance with the input voltage. It can be pulled low by external circuitry to reset the device. See Section 13 ELECTRICAL CHARACTERISTICS for more details.

A RESET signal originating from an external source must have a duration of at least $t_{h(RSTL)in}$ in order to be recognized (see Figure 15). This detection is asynchronous and therefore the MCU can enter the RESET state even in HALT mode.

Figure 14. Reset Block Diagram



Note 1: See "Illegal Opcode Reset" on page 95 for more details on illegal opcode reset conditions.

SUPPLY, RESET AND CLOCK MANAGEMENT (Cont'd)

The RESET pin is an asynchronous signal which plays a major role in EMS performance. In a noisy environment, it is recommended to follow the guidelines mentioned in Section 13 ELECTRICAL CHARACTERISTICS.

7.5.3 External Power-On RESET

If the LVD is disabled by the option byte, to start up the microcontroller correctly, the user must use an external reset circuit to ensure that the reset signal is held low until V_{DD} is over the minimum level specified for the selected f_{OSC} frequency.

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

7.5.4 Internal Low Voltage Detector (LVD) RESET

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

- Power-On RESET
- Voltage Drop RESET

The device RESET pin acts as an output that is pulled low when $V_{DD} < V_{IT+}$ (rising edge) or $V_{DD} < V_{IT-}$ (falling edge) as shown in Figure 15.

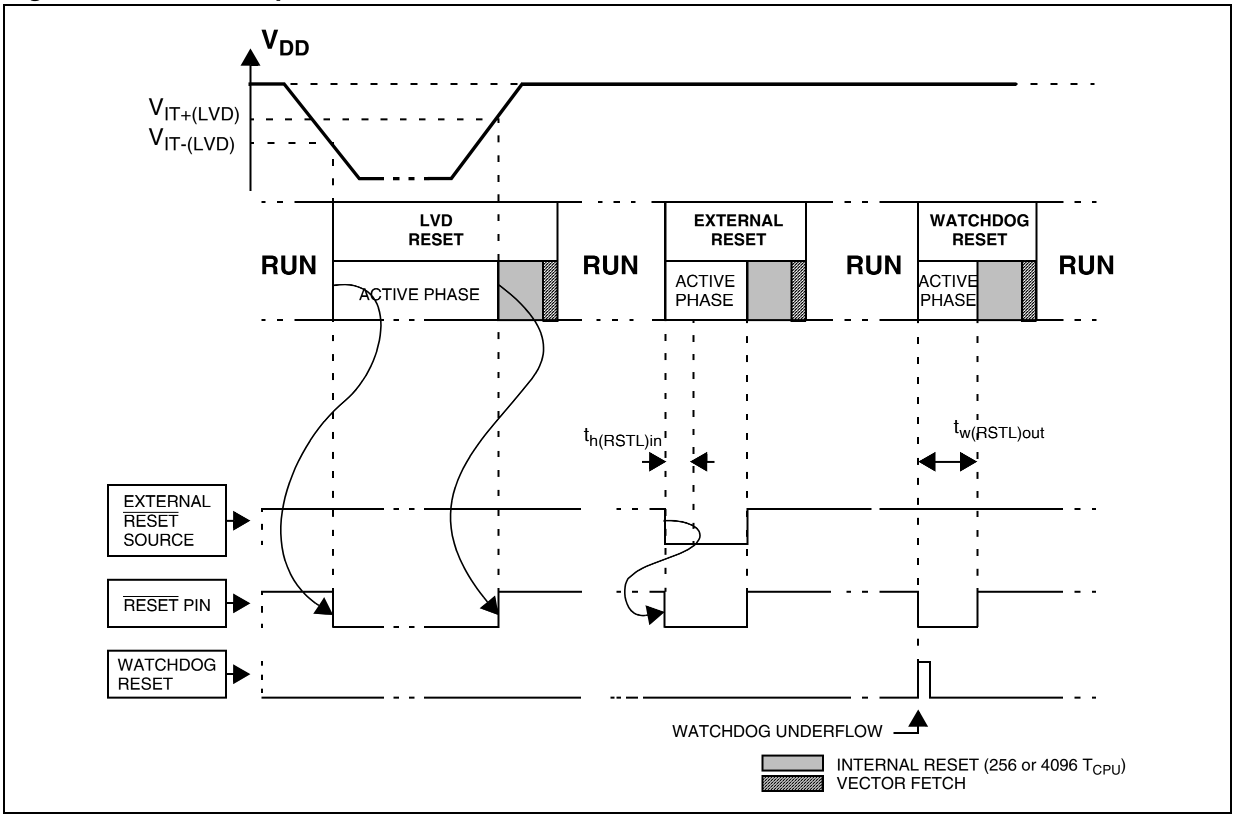
The LVD filters spikes on V_{DD} larger than $t_{g(VDD)}$ to avoid parasitic resets.

7.5.5 Internal Watchdog RESET

The RESET sequence generated by an internal Watchdog counter overflow is shown in Figure 15.

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

Figure 15. RESET Sequences



SUPPLY, RESET AND CLOCK MANAGEMENT (Cont'd)

7.6 SYSTEM INTEGRITY MANAGEMENT (SI)

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

Note: A reset can also be triggered following the detection of an illegal opcode or prebyte code. Refer to section 12.2.1 on page 95 for further details.

7.6.1 Low Voltage Detector (LVD)

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

The $V_{IT-(LVD)}$ reference value for a voltage drop is lower than the $V_{IT+(LVD)}$ reference value for power-on 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:

– $V_{IT+(LVD)}$ when V_{DD} is rising

– $V_{IT-(LVD)}$ when V_{DD} is falling

The LVD function is illustrated in Figure 16.

Provided the minimum V_{DD} value (guaranteed for the oscillator frequency) is above $V_{IT-(LVD)}$, the MCU can only be in two modes:

– Under full software control

– In static safe reset

In these conditions, secure operation is always ensured for the application without the need for external reset hardware.

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

Notes:

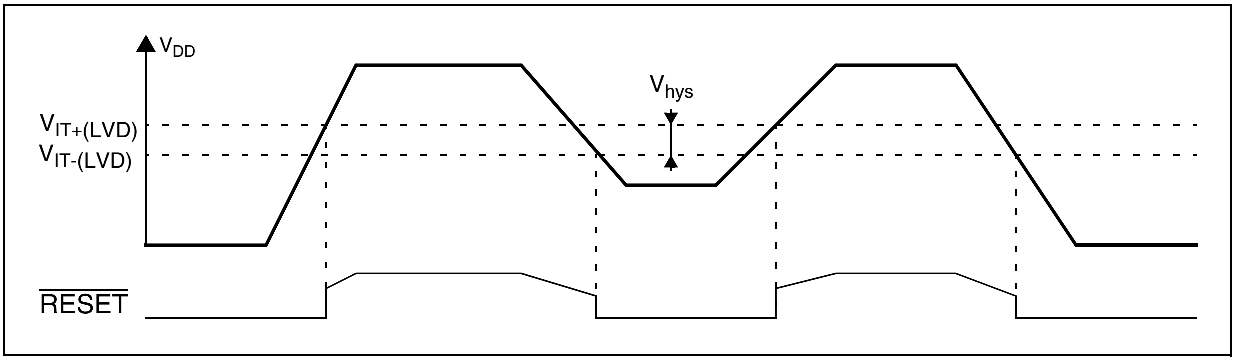
The LVD allows the device to be used without any external RESET circuitry.

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

Use of LVD with capacitive power supply: With this type of power supply, if power cuts occur in the application, it is recommended to pull V_{DD} down to 0V to ensure optimum restart conditions. Refer to circuit example in Figure 96 on page 119 and note 4.

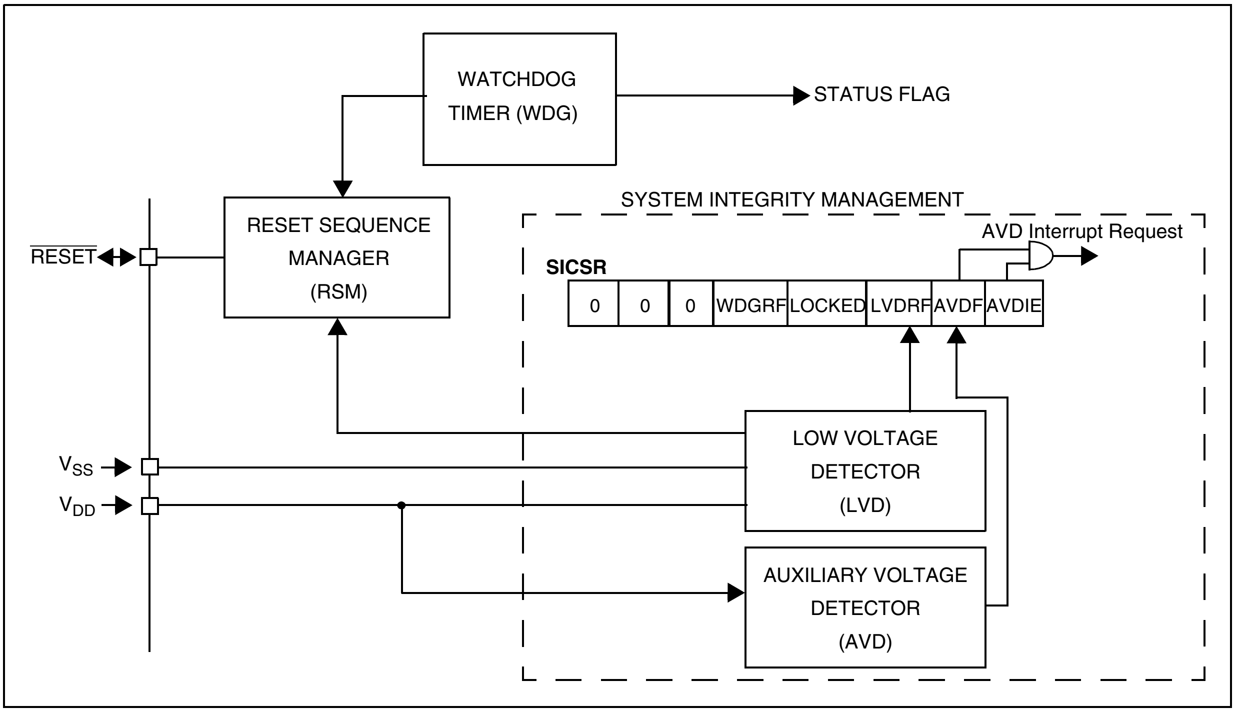
For the application to function correctly, it is recommended to make sure that the V_{DD} supply voltage rises monotonously when the device is exiting from RESET.

Figure 16. Low Voltage Detector vs Reset



SUPPLY, RESET AND CLOCK MANAGEMENT (Cont'd)

Figure 17. Reset and Supply Management Block Diagram



7.6.2 Auxiliary Voltage Detector (AVD)

The Auxiliary Voltage Detector (AVD) function is based on an analog comparison between a $V_{IT-(AVD)}$ and a $V_{IT+(AVD)}$ reference value and the V_{DD} main supply voltage (V_{AVD}). The $V_{IT-(AVD)}$ reference value for falling voltage is lower than the $V_{IT+(AVD)}$ reference value for rising voltage in order to avoid parasitic detection (hysteresis).

The output of the AVD comparator is directly readable by the application software through a real time status bit (AVDF) in the SICSR register. This bit is read only.

Caution: The AVD functions only if the LVD is enabled through the option byte.

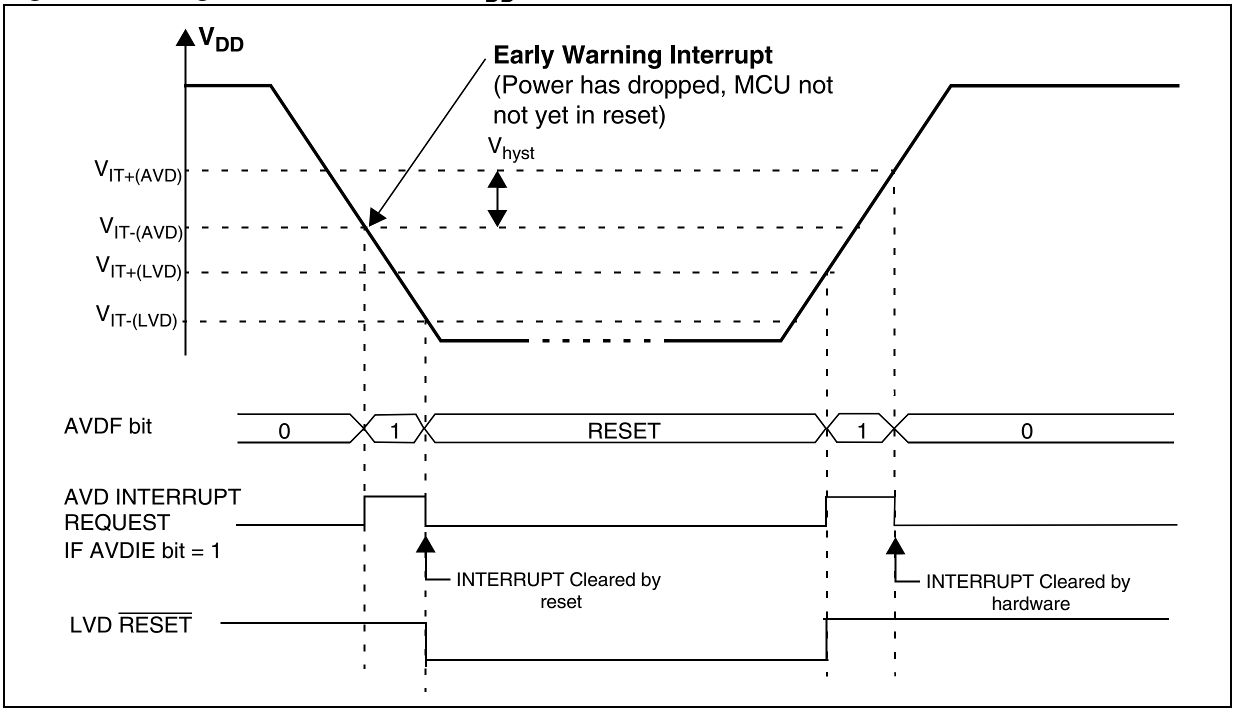
7.6.2.1 Monitoring the V_{DD} Main Supply

If the AVD interrupt is enabled, an interrupt is generated when the voltage crosses the $V_{IT+(LVD)}$ or $V_{IT-(AVD)}$ threshold (AVDF bit is set).

In the case of a drop in voltage, the AVD interrupt acts as an early warning, allowing the software to shut down safely before the LVD resets the microcontroller. See Figure 18 on page 29.

SUPPLY, RESET AND CLOCK MANAGEMENT (Cont'd)

Figure 18. Using the AVD to Monitor V_{DD}



7.6.3 Low-Power Modes

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

7.6.3.1 Interrupts

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

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

SUPPLY, RESET AND CLOCK MANAGEMENT (Cont'd)

7.6.4 Register Description

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

Read/Write

Reset Value: 0110 0xx0 (6xh)

7							0
Res	CR1	CR0	WDGRF	LOCKED	LVDRF	AVDF	AVDIE

Bit 7 = **Reserved** (should be 0)

Bits 6:5 = **CR[1:0]** *RC Oscillator Frequency Adjustment bits*

These bits, as well as CR[9:2] bits in the RCCR register must be written immediately after reset to adjust the RC oscillator frequency and to obtain an accuracy of 1%. Refer to section 7.3 on page 22.

Bit 4 = **WDGRF** *Watchdog Reset flag*
This bit indicates that the last Reset was generated by the Watchdog peripheral. It is set by hardware (Watchdog reset) and cleared by software (writing zero) or an LVD Reset (to ensure a stable cleared state of the WDGRF flag when CPU starts).
Combined with the LVDRF flag information, the flag description is given in the following table.

RESET Sources	LVDRF	WDGRF
External RESET pin	0	0
Watchdog	0	1
LVD	1	X

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

Bit 2 = **LVDRF** *LVD reset flag*
This bit indicates that the last Reset was generated by the LVD block. It is set by hardware (LVD reset) and cleared by software (by reading). When the LVD is disabled by OPTION BYTE, the LVDRF bit value is undefined.

Bit 1 = **AVDF** *Voltage Detector Flag*
This read-only bit is set and cleared by hardware. If the AVDIE bit is set, an interrupt request is generated when the AVDF bit is set. Refer to Figure 18 and to Section 7.6.2.1 for additional details.
0: V_{DD} over AVD threshold
1: V_{DD} under AVD threshold

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

Application notes

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

PLL TEST REGISTER (PLLTST)

Read/Write

Reset Value: 0000 0000(00h)

7							0
PLLdiv2	0	0	0	0	0	0	0

Bit 7: **PLLdiv2** *PLL clock divide by 2*

This bit is read or write by software and cleared by hardware after reset. This bit divides the PLL output clock by 2.
0: PLL output clock
1: Divide by 2 of PLL output clock

Refer to "Clock Management Block Diagram" on page 23

Note: Write of this bit is effective after 2 Tcpu cycles (if system clock is 8 MHz) or else 1 cycle (if system clock is 4 MHz), that is, effective time is 250ns.

Bits 6:0: **Reserved**, must always be cleared.

8 INTERRUPTS

The ST7 core may be interrupted by one of two different methods: Maskable hardware interrupts as listed in the Interrupt Mapping Table and a non-maskable software interrupt (TRAP). The Interrupt processing flowchart is shown in Figure 19.
The maskable interrupts must be enabled by clearing the I bit in order to be serviced. However, disabled interrupts may be latched and processed when they are enabled (see external interrupts subsection).

Note: After reset, all interrupts are disabled.

When an interrupt has to be serviced:

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

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

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

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

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

Note: As a consequence of the IRET instruction, the I bit is cleared and the main program resumes.

Priority Management

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

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

Interrupts and Low Power Mode

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

8.1 NON MASKABLE SOFTWARE INTERRUPT

This interrupt is entered when the TRAP instruction is executed regardless of the state of the I bit. It is serviced according to the flowchart in Figure 19.

8.2 EXTERNAL INTERRUPTS

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

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

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

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

8.3 PERIPHERAL INTERRUPTS

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

– The I bit of the CC register is cleared.

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

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

Clearing an interrupt request is done by:

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

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

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

INTERRUPTS (Cont'd)

Figure 19. Interrupt Processing Flowchart

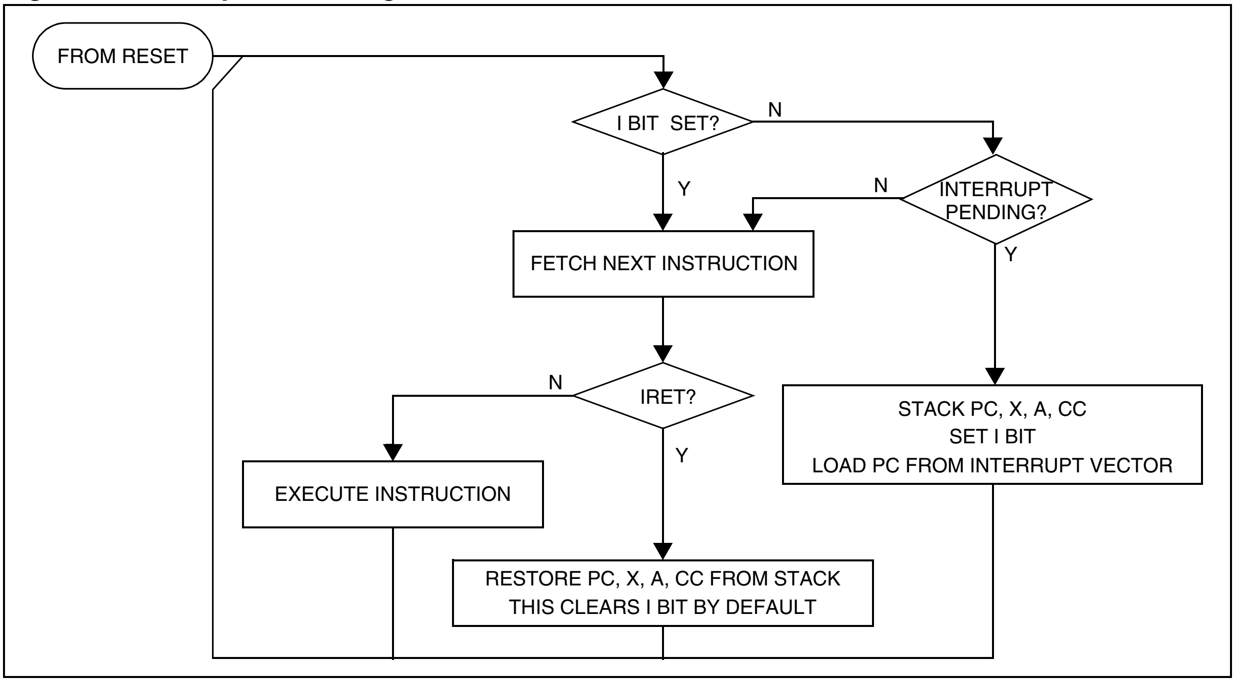


Table 5. Interrupt Mapping

N°	Source Block	Description	Register Label	Priority Order	Exit from HALT or AWUFH	Address Vector
	RESET	Reset	N/A	Highest Priority	yes	FFFEh-FFFFh
	TRAP	Software Interrupt			no	FFFCh-FFFDh
0	AWU	Auto Wake-Up Interrupt	AWUCSR		yes[1)]	FFFAh-FFFBh
1	ei0	External Interrupt 0	N/A		yes	FFF8h-FFF9h
2	ei1	External Interrupt 1				FFF6h-FFF7h
3	ei2	External Interrupt 2				FFF4h-FFF5h
4	ei3	External Interrupt 3				FFF2h-FFF3h
5	LITE TIMER	LITE TIMER RTC2 interrupt	LTCSR2		no	FFF0h-FFF1h
6	Not used					FFEEh-FFEFh
7	SI	AVD interrupt	SICSR		no	FFECh-FFEDh
8	AT TIMER	AT TIMER Output Compare Interrupt or Input Capture Interrupt	PWMxCSR or ATCSR		no	FFEAh-FFEBh
9		AT TIMER Overflow Interrupt	ATCSR		yes[2)]	FFE8h-FFE9h
10	LITE TIMER	LITE TIMER Input Capture Interrupt	LTCSR		no	FFE6h-FFE7h
11		LITE TIMER RTC1 Interrupt	LTCSR		yes[2)]	FFE4h-FFE5h
12	SPI	SPI Peripheral Interrupts	SPICSR	Lowest Priority	yes	FFE2h-FFE3h
13	AT TIMER	AT TIMER Overflow Interrupt	ATCSR2		no	FFE0h-FFE1h

Notes:
1. This interrupt exits the MCU from "Auto Wake-Up from Halt" mode only.
2. These interrupts exit the MCU from "ACTIVE HALT" mode only.

INTERRUPTS (Cont'd)

EXTERNAL INTERRUPT CONTROL REGISTER (EICR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
IS31	IS30	IS21	IS20	IS11	IS10	IS01	IS00

Bits 7:6 = **IS3[1:0]** *ei3 sensitivity*
These bits define the interrupt sensitivity for ei3 (Port B0) according to Table 6.

Bits 5:4 = **IS2[1:0]** *ei2 sensitivity*
These bits define the interrupt sensitivity for ei2 (Port B3) according to Table 6.

Bits 3:2 = **IS1[1:0]** *ei1 sensitivity*
These bits define the interrupt sensitivity for ei1 (Port A7) according to Table 6.

Bits 1:0 = **IS0[1:0]** *ei0 sensitivity*
These bits define the interrupt sensitivity for ei0 (Port A0) according to Table 6.

Notes:

1. These 8 bits can be written only when the I bit in the CC register is set.

2. Changing the sensitivity of a particular external interrupt clears this pending interrupt. This can be used to clear unwanted pending interrupts. Refer to section "External Interrupt Function" on page 43

Table 6. Interrupt Sensitivity Bits

ISx1	ISx0	External Interrupt Sensitivity
0	0	Falling edge and low level
	1	Rising edge only
1	0	Falling edge only
	1	Rising and falling edge

.

EXTERNAL INTERRUPT SELECTION REGISTER (EISR)

Read/Write

Reset Value: 0000 1100 (0Ch)

7							0
ei31	ei30	ei21	ei20	ei11	ei10	ei01	ei00

Bits 7:6 = **ei3[1:0]** *ei3 pin selection*
These bits are written by software. They select the Port B I/O pin used for the ei3 external interrupt according to the table below.

External Interrupt I/O pin selection

ei31	ei30	I/O Pin
0	0	PB0*
0	1	PB1
1	0	PB2

* Reset State

Bits 5:4 = **ei2[1:0]** *ei2 pin selection*
These bits are written by software. They select the Port B I/O pin used for the ei2 external interrupt according to the table below.

External Interrupt I/O pin selection

ei21	ei20	I/O Pin
0	0	PB3*
	1	PB4[1)]
1	0	PB5
	1	PB6

* Reset State

Notes:

1. PB4 cannot be used as an external interrupt in HALT mode.

INTERRUPTS (Cont'd)

Bits 3:2 = **ei1[1:0]** *ei1 pin selection*
These bits are written by software. They select the Port A I/O pin used for the ei1 external interrupt according to the table below.

External Interrupt I/O pin selection

ei11	ei10	I/O Pin
0	0	PA4
	1	PA5
1	0	PA6
	1	PA7*

* Reset State

Bits 1:0 = **ei0[1:0]** *ei0 pin selection*
These bits are written by software. They select the Port A I/O pin used for the ei0 external interrupt according to the table below.

External Interrupt I/O pin selection

ei01	ei00	I/O Pin
0	0	PA0*
	1	PA1
1	0	PA2
	1	PA3

* Reset State

9 POWER SAVING MODES

9.1 INTRODUCTION

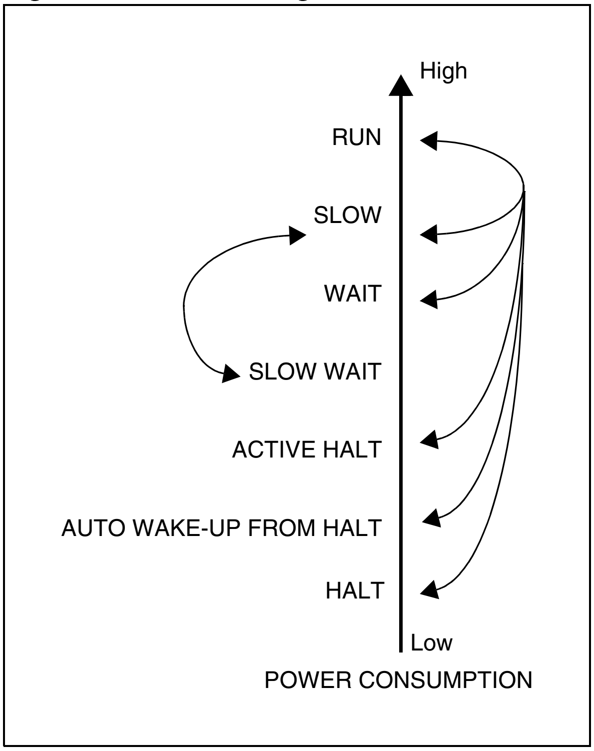
To give a large measure of flexibility to the application in terms of power consumption, five main power saving modes are implemented in the ST7 (see Figure 20):

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

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

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

Figure 20. Power Saving Mode Transitions



9.2 SLOW MODE

This mode has two targets:

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

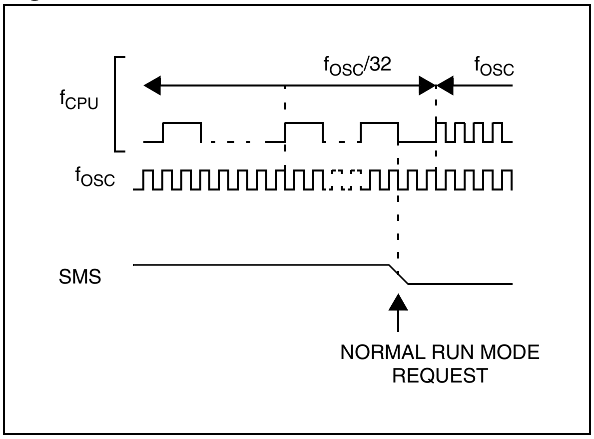
– To adapt the internal clock frequency (f_{CPU}) to the available supply voltage.

SLOW mode is controlled by the SMS bit in the MCCSR register which enables or disables SLOW mode.

In this mode, the oscillator frequency is divided by 32. The CPU and peripherals are clocked at this lower frequency.

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

Figure 21. SLOW Mode Clock Transition



POWER SAVING MODES (Cont'd)

9.3 WAIT MODE

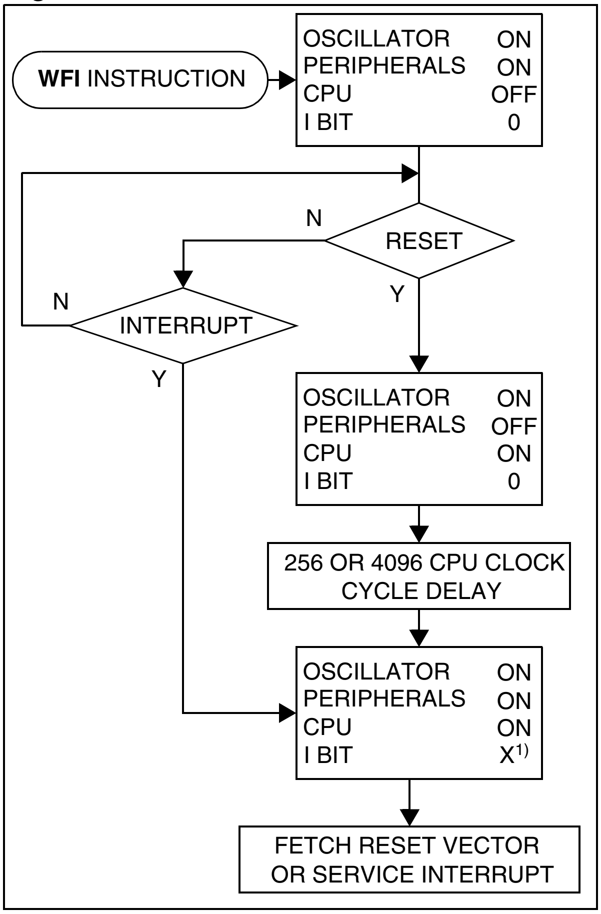
WAIT mode places the MCU into a low power consumption mode by stopping the CPU.

This power saving mode is selected by calling the 'WFI' instruction.

All peripherals remain active. During WAIT mode, the I bit of the CC register is cleared to enable all interrupts. All other registers and memory remain unchanged. The MCU remains in WAIT mode until an interrupt or reset occurs, whereupon it wakes up and the Program Counter branches to the starting address of the interrupt or reset service routine.

Refer to Figure 22.

Figure 22. WAIT Mode Flowchart



Notes:

1. Before servicing an interrupt, the CC register is pushed on the stack. The I bit of the CC register is set during the interrupt routine and cleared when the CC register is popped.

POWER SAVING MODES (Cont'd)

9.4 HALT MODE

The HALT mode is the lowest power consumption mode of the MCU. It is entered by executing the 'HALT' instruction when ACTIVE HALT is disabled (see section 9.5 on page 38 for more details) and when the AWUEN bit in the AWUCSR register is cleared.

The MCU can exit HALT mode on reception of either a specific interrupt (see Table 5, "Interrupt Mapping," on page 32) or a RESET. When exiting HALT mode by means of a RESET or an interrupt, the oscillator is immediately turned on and the 256 or 4096 CPU cycle delay is used to stabilize the oscillator. After the start-up delay, the CPU resumes operation by servicing the interrupt or by fetching the reset vector which woke it up (see Figure 24).

When entering HALT mode, the I bit in the CC register is forced to 0 to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes up immediately.

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

The compatibility of Watchdog operation with HALT mode is configured by the "WDGHALT" option bit of the option byte. The HALT instruction, when executed while the Watchdog system is enabled, can generate a Watchdog RESET (see section 15.1 on page 126 for more details).

Figure 23. HALT Timing Overview

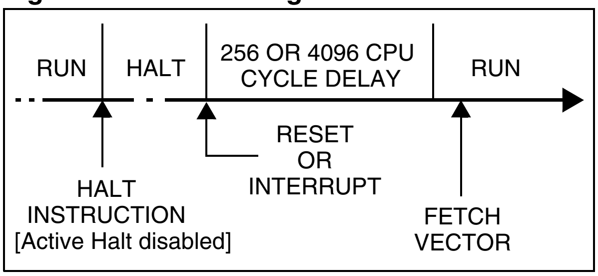
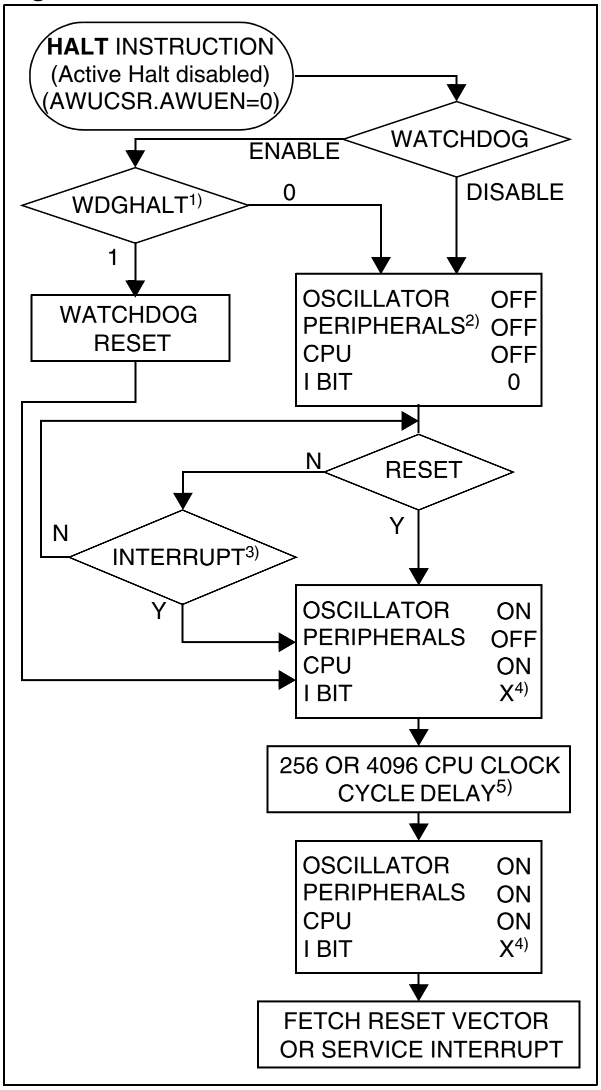


Figure 24. HALT Mode Flowchart



Notes:

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

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

3. Only some specific interrupts can exit the MCU from HALT mode (such as external interrupt). Refer to Table 5, "Interrupt Mapping," on page 32 for more details.

4. Before servicing an interrupt, the CC register is pushed on the stack. The I bit of the CC register is set during the interrupt routine and cleared when the CC register is popped.

5. If the PLL is enabled by option byte, it outputs the clock after a delay of $t_{STARTUP}$ (see Figure 11 on page 22).

POWER SAVING MODES (Cont'd)

9.4.1 Halt Mode Recommendations

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

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

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

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

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

9.5 ACTIVE HALT MODE

ACTIVE HALT mode is the lowest power consumption mode of the MCU with a real time clock available. It is entered by executing the 'HALT' instruction. The decision to enter either in ACTIVE HALT or HALT mode is given by the LTCSR/ATCSR register status as shown in the following table:

LTCSR1 TB1IE bit	ATCSR OVFIE bit	ATCSR CK1 bit	ATCSR CK0 bit	Meaning
0	x	x	0	ACTIVE HALT mode disabled
	0		x	
1	x	x	x	ACTIVE HALT mode enabled
x	1	0	1	

The MCU exits ACTIVE HALT mode on reception of a specific interrupt (see Table 5, "Interrupt Mapping," on page 32) or a RESET.

– When exiting ACTIVE HALT mode by means of a RESET, a 256 or 4096 CPU cycle delay occurs. After the start-up delay, the CPU resumes operation by fetching the reset vector which woke it up (see Figure 26).

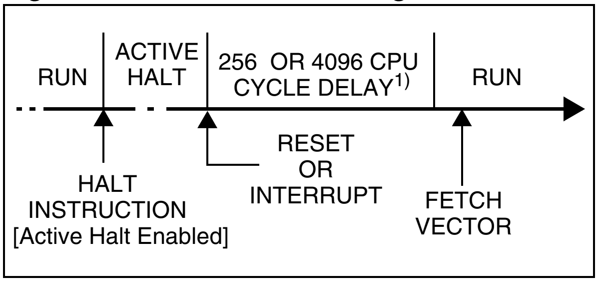
– When exiting ACTIVE HALT mode by means of an interrupt, the CPU immediately resumes operation by servicing the interrupt vector which woke it up (see Figure 26).

When entering ACTIVE HALT mode, the I bit in the CC register is cleared to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes up immediately (see Note 3).

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

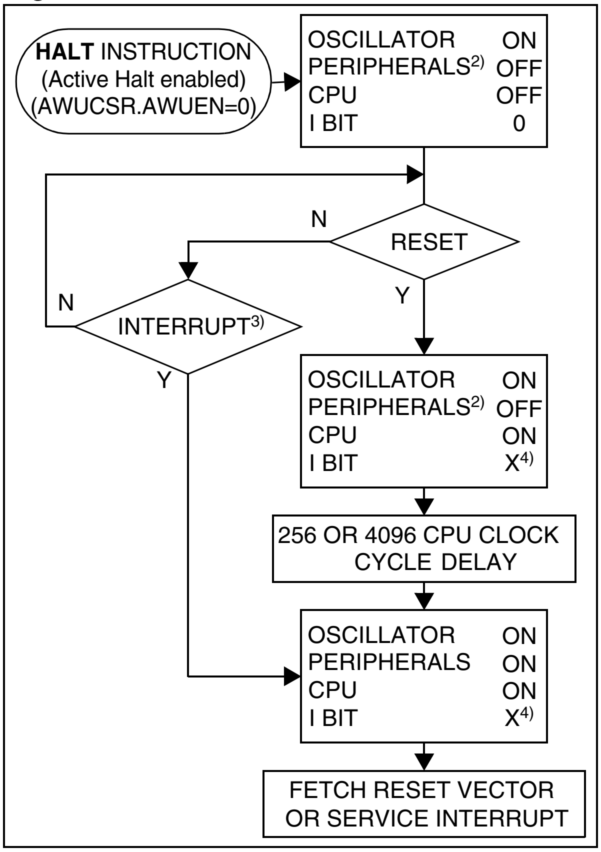
Note: As soon as ACTIVE HALT is enabled, executing a HALT instruction while the Watchdog is active does not generate a RESET.
This means that the device cannot exceed a defined delay in this power saving mode.

Figure 25. ACTIVE HALT Timing Overview



POWER SAVING MODES (Cont'd)

Figure 26. ACTIVE HALT Mode Flowchart



Notes:

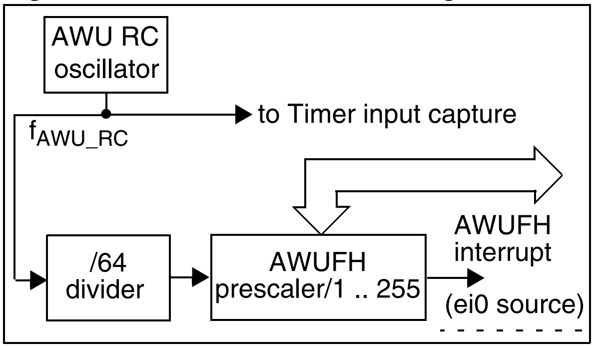
1. This delay occurs only if the MCU exits ACTIVE HALT mode by means of a RESET.
2. Peripherals clocked with an external clock source can still be active.
3. Only the RTC1 interrupt and some specific interrupts can exit the MCU from ACTIVE HALT mode. Refer to Table 5, "Interrupt Mapping," on page 32 for more details.
4. Before servicing an interrupt, the CC register is pushed on the stack. The I bit of the CC register is set during the interrupt routine and cleared when the CC register is popped.

9.6 AUTO WAKE-UP FROM HALT MODE

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

It is entered by executing the HALT instruction when the AWUEN bit in the AWUCSR register has been set.

Figure 27. AWUFH Mode Block Diagram



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

To compensate for any frequency dispersion of the AWU RC oscillator, it can be calibrated by measuring the clock frequency f_{AWU_RC} and then calculating the right prescaler value. Measurement mode is enabled by setting the AWUM bit in the AWUCSR register in Run mode. This connects f_{AWU_RC} to the input capture of the 12-bit Auto-Reload timer, allowing the f_{AWU_RC} to be measured using the main oscillator clock as a reference timebase.

POWER SAVING MODES (Cont'd)

Similarities with Halt mode

The following AWUFH mode behavior is the same as normal HALT mode:

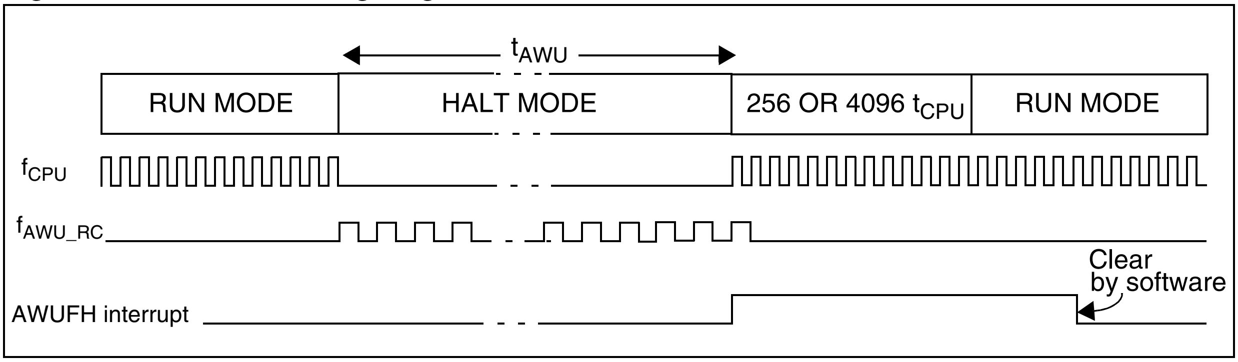
– The MCU can exit AWUFH mode by means of any interrupt with exit from Halt capability or a reset (see Section 9.4 HALT MODE).

– When entering AWUFH mode, the I bit in the CC register is forced to 0 to enable interrupts. Therefore, if an interrupt is pending, the MCU wakes up immediately.

– In AWUFH mode, the main oscillator is turned off, stopping all internal processing, including the operation of the on-chip peripherals. None of the peripherals are clocked except those which receive their clock supply from another clock generator (such as an external or auxiliary oscillator like the AWU oscillator).

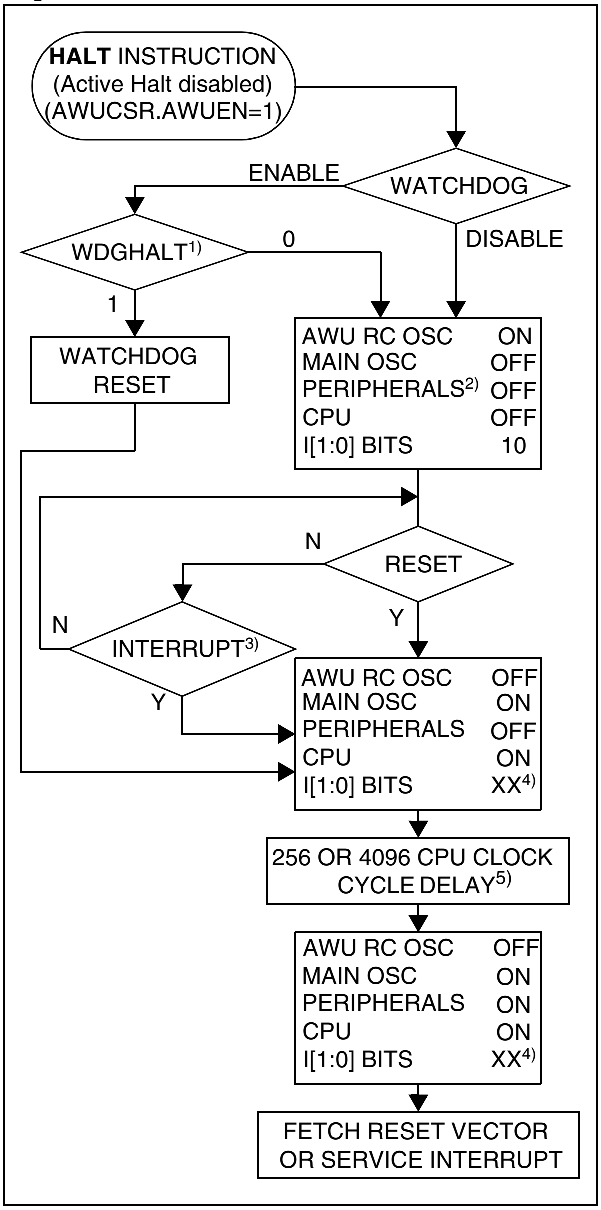
– The compatibility of Watchdog operation with AWUFH mode is configured by the WDGHALT option bit in the option byte. Depending on this setting, the HALT instruction, when executed while the Watchdog system is enabled, can generate a Watchdog RESET.

Figure 28. AWUF Halt Timing Diagram



POWER SAVING MODES (Cont'd)

Figure 29. AWUFH Mode Flowchart



Notes:

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

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

3. Only an AWUFH interrupt and some specific interrupts can exit the MCU from HALT mode (such as external interrupt). Refer to Table 5, "Interrupt Mapping," on page 32 for more details.

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

5. If the PLL is enabled by the option byte, it outputs the clock after an additional delay of $t_{STARTUP}$ (see Figure 11).

POWER SAVING MODES (Cont'd)

9.6.0.1 Register Description

AWUFH CONTROL/STATUS REGISTER (AWUCSR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	AWUF	AWUM	AWUEN

Bits 7:3 = **Reserved.**

Bit 1 = **AWUF** *Auto Wake-Up Flag*
This bit is set by hardware when the AWU module generates an interrupt and cleared by software on reading AWUCSR. Writing to this bit does not change its value.
0: No AWU interrupt occurred
1: AWU interrupt occurred

Bit 1 = **AWUM** *Auto Wake-Up Measurement*
This bit enables the AWU RC oscillator and connects its output to the input capture of the 12-bit Auto-Reload timer. This allows the timer to measure the AWU RC oscillator dispersion and then compensate this dispersion by providing the right value in the AWUPRE register.
0: Measurement disabled
1: Measurement enabled

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

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

7							0
AWUPR7	AWUPR6	AWUPR5	AWUPR4	AWUPR3	AWUPR2	AWUPR1	AWUPR0

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

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

In AWU mode, the period that the MCU stays in Halt Mode (t_{AWU} in Figure 28 on page 40) is defined by

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

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

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

Table 7. AWU Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0049h	**AWUPR** Reset Value	AWUPR7 1	AWUPR6 1	AWUPR5 1	AWUPR4 1	AWUPR3 1	AWUPR2 1	AWUPR1 1	AWUPR0 1
004Ah	**AWUCSR** Reset Value	0	0	0	0	0	AWUF	AWUM	AWUEN

10 I/O PORTS

10.1 INTRODUCTION

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

10.2 FUNCTIONAL DESCRIPTION

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

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

Figure 30 shows the generic I/O block diagram.

10.2.1 Input Modes

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

If an OR bit is available, different input modes can be configured by software: Floating or pull-up. Refer to Section 10.3 I/O PORT IMPLEMENTATION for configuration.

Notes:

1. Writing to the DR modifies the latch value but does not change the state of the input pin.
2. Do not use read/modify/write instructions (BSET/BRES) to modify the DR register.

10.2.1.1 External Interrupt Function

External interrupt capability is selected using the EISR register. If EISR bits are <> 0, the corresponding pin is used as external interrupt. In this case, the ORx bit can select the pin as either interrupt floating or interrupt pull-up. In this configuration, a signal edge or level input on the I/O generates an interrupt request via the corresponding interrupt vector (eix).

Falling or rising edge sensitivity is programmed independently for each interrupt vector. The External Interrupt Control Register (EICR) or the Miscellaneous Register controls this sensitivity, depending on the device.

A device may have up to seven external interrupts. Several pins may be tied to one external interrupt vector. Refer to “PIN DESCRIPTION” on page 5 to see which ports have external interrupts.

If several I/O interrupt pins on the same interrupt vector are selected simultaneously, they are logically combined. For this reason, if one of the interrupt pins is tied low, it may mask the others.

External interrupts are hardware interrupts. Fetching the corresponding interrupt vector automatically clears the request latch. Changing the sensitivity of a particular external interrupt clears this pending interrupt. This can be used to clear unwanted pending interrupts.

Spurious interrupts

When enabling/disabling an external interrupt by setting/resetting the related OR register bit, a spurious interrupt is generated if the pin level is low and its edge sensitivity includes falling/rising edge. This is due to the edge detector input, which is switched to '1' when the external interrupt is disabled by the OR register.

To avoid this unwanted interrupt, a "safe" edge sensitivity (rising edge for enabling and falling edge for disabling) must be selected before changing the OR register bit and configuring the appropriate sensitivity again.

Caution: If a pin level change occurs during these operations (asynchronous signal input), as interrupts are generated according to the current sensitivity, it is advised to disable all interrupts before and to re-enable them after the complete previous sequence in order to avoid an external interrupt occurring on the unwanted edge.

This corresponds to the following steps:

1. To enable an external interrupt:
   – Set the interrupt mask with the SIM instruction (in cases where a pin level change could occur)
   – Select rising edge
   – Enable the external interrupt through the OR register
   – Select the desired sensitivity if different from rising edge
   – Reset the interrupt mask with the RIM instruction (in cases where a pin level change could occur)
2. To disable an external interrupt:

I/O PORTS (Cont'd)

– Set the interrupt mask with the SIM instruction SIM (in cases where a pin level change could occur)

– Select falling edge

– Disable the external interrupt through the OR register

– Select rising edge

– Reset the interrupt mask with the RIM instruction (in cases where a pin level change could occur)

10.2.2 Output Modes

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

If an OR bit is available, different output modes can be selected by software: Push-pull or open-drain. Refer to "I/O PORT IMPLEMENTATION" on page 47 for configuration.

DR Value and Output Pin Status

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

10.2.3 Alternate Functions

Many ST7 I/Os have one or more alternate functions. These may include output signals from, or input signals to, on-chip peripherals. The Device Pin Description table describes which peripheral signals can be input/output to which ports.

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

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

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

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

I/O PORTS (Cont'd)

Figure 30. I/O Port General Block Diagram

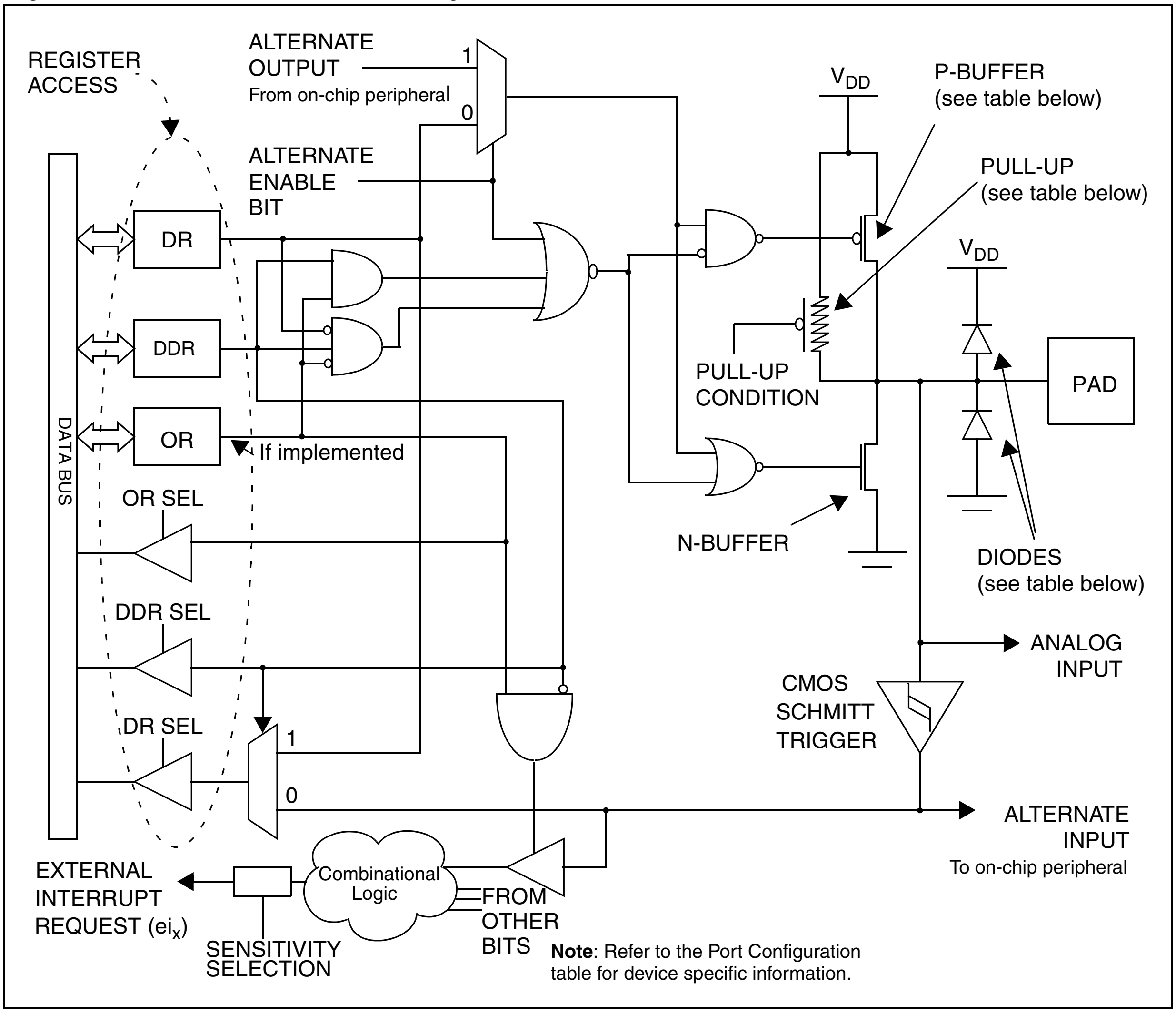


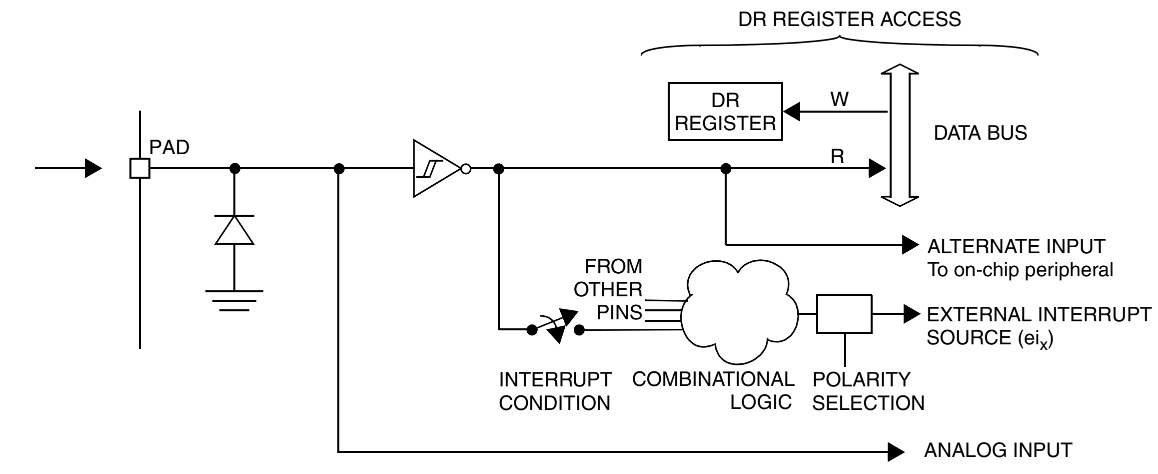
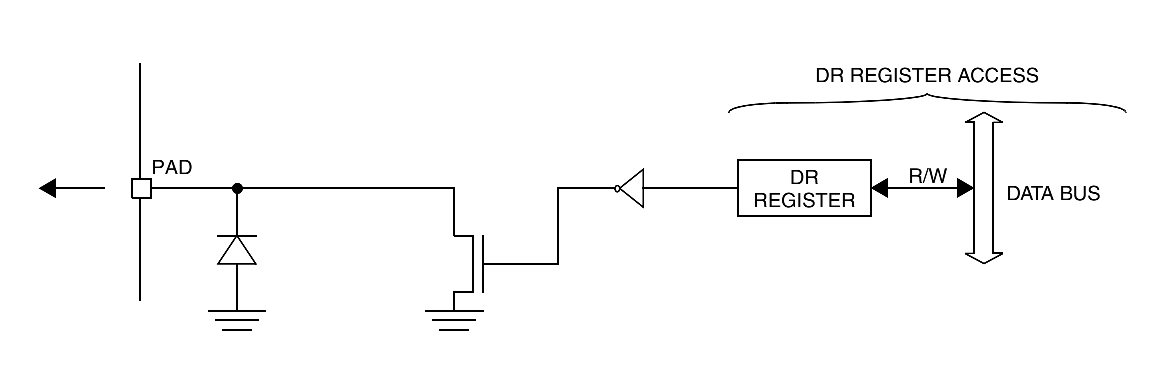
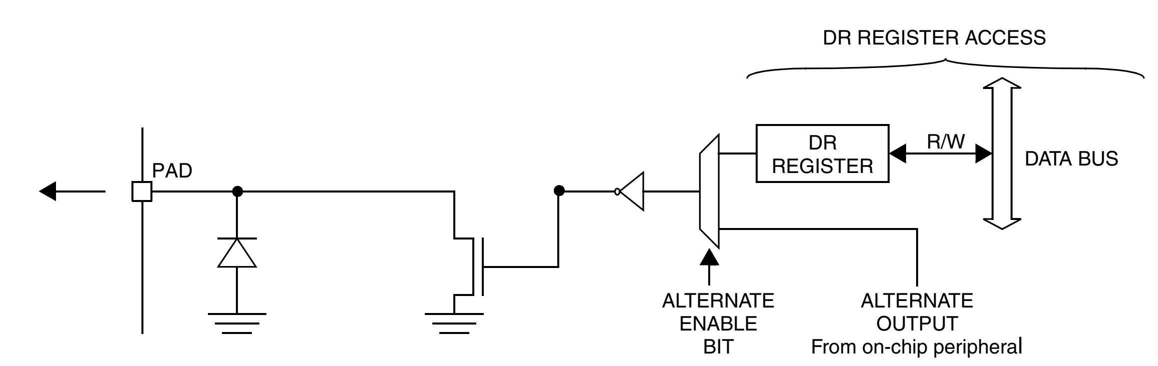
Table 8. Port Mode Options

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		

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

I/O PORTS (Cont'd)

Table 9. I/O Configurations

<table>
<tr><th></th><th>Hardware Configuration</th></tr>
<tr><td>INPUT 1)</td><td>
</td></tr>
<tr><td>OPEN-DRAIN OUTPUT 2)</td><td>
</td></tr>
<tr><td>PUSH-PULL OUTPUT 2)</td><td>
</td></tr>
</table>

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

Analog alternate function

Configure the I/O as floating input to use an ADC input. The analog multiplexer (controlled by the ADC registers) switches the analog voltage present on the selected pin to the common analog rail, connected to the ADC input.

Analog Recommendations

Do not change the voltage level or loading on any I/O while conversion is in progress. Do not have clocking pins located close to a selected analog pin.

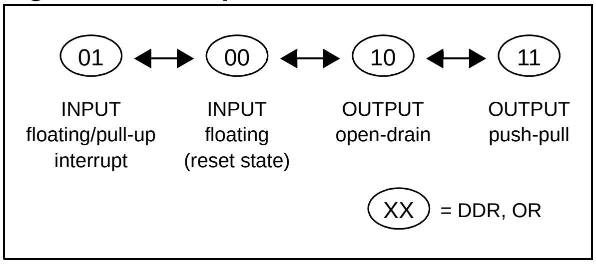
WARNING: The analog input voltage level must be within the limits stated in the absolute maximum ratings.

10.3 I/O PORT IMPLEMENTATION

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

Switching these I/O ports from one state to another should be done in a sequence that prevents unwanted side effects. Recommended safe transitions are illustrated in Figure 31. Other transitions are potentially risky and should be avoided, since they may present unwanted side-effects such as spurious interrupt generation.

Figure 31. Interrupt I/O Port State Transitions



10.4 UNUSED I/O PINS

Unused I/O pins must be connected to fixed voltage levels. Refer to section 13.8 on page 113.

10.5 LOW-POWER MODES

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

10.6 INTERRUPTS

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

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

Related Documentation

SPI Communication between ST7 and EEPROM (AN970)

S/W implementation of I2C bus master (AN1045)

Software LCD driver (AN1048)

I/O PORTS (Cont'd)

10.7 DEVICE-SPECIFIC I/O PORT CONFIGURATION

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

Standard Ports

PA7:0, PB6:0

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

PC1:0 (multiplexed with OSC1,OSC2)

MODE	DDR
floating input	0
push-pull output	1

The selection between OSC1 or PC0 and OSC2 or PC1 is done by the option byte (refer to section 15.1 on page 126). Interrupt capability is not available on PC1:0.

Port C is not present on ROM devices.

Note: PCOR not implemented but p-transistor always active in output mode (refer to Figure 30 on page 45).

Interrupt Ports

Ports where the external interrupt capability is selected using the EISR register

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

Table 10. Port Configuration (Standard Ports)

Port	Pin name	Input		Output	
		OR = 0	OR = 1	OR = 0	OR = 1
Port A	PA7:0	floating	pull-up	open drain	push-pull
Port B	PB6:0				

Note: On ports where the external interrupt capability is selected using the EISR register, the configuration is as follows:

Port	Pin name	Input		Output	
		OR = 0	OR = 1	OR = 0	OR = 1
Port A	PA7:0	floating	pull-up interrupt	open drain	push-pull
Port B	PB6:0				

Table 11. I/O Port Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0000h	**PADR** Reset Value	MSB 1	1	1	1	1	1	1	LSB 1
0001h	**PADDR** Reset Value	MSB 0	0	0	0	0	0	0	LSB 0

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0002h	**PAOR** Reset Value	MSB 0	 1	 0	 0	 0	 0	 0	LSB 0
0003h	**PBDR** Reset Value	MSB 1	 1	 1	 1	 1	 1	 1	LSB 1
0004h	**PBDDR** Reset Value	MSB 0	 0	 0	 0	 0	 0	 0	LSB 0
0005h	**PBOR** Reset Value	MSB 0	 0	 0	 0	 0	 0	 0	LSB 0
0006h	**PCDR** **Reset Value**	MSB 0	 0	 0	 0	 0	 0	 1	LSB 1
0007h	**PCDDR** **Reset Value**	MSB 0	 0	 0	 0	 0	 0	 0	LSB 0

10.8 MULTIPLEXED INPUT/OUTPUT PORTS

OSC1/PC0 are multiplexed on one pin (pin20) and OSC2/PC1 are multiplexed on another pin (pin19).

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 upon expiration of a programmed time period, unless the program refreshes the counter's contents before the T6 bit is cleared.

11.1.2 Main Features

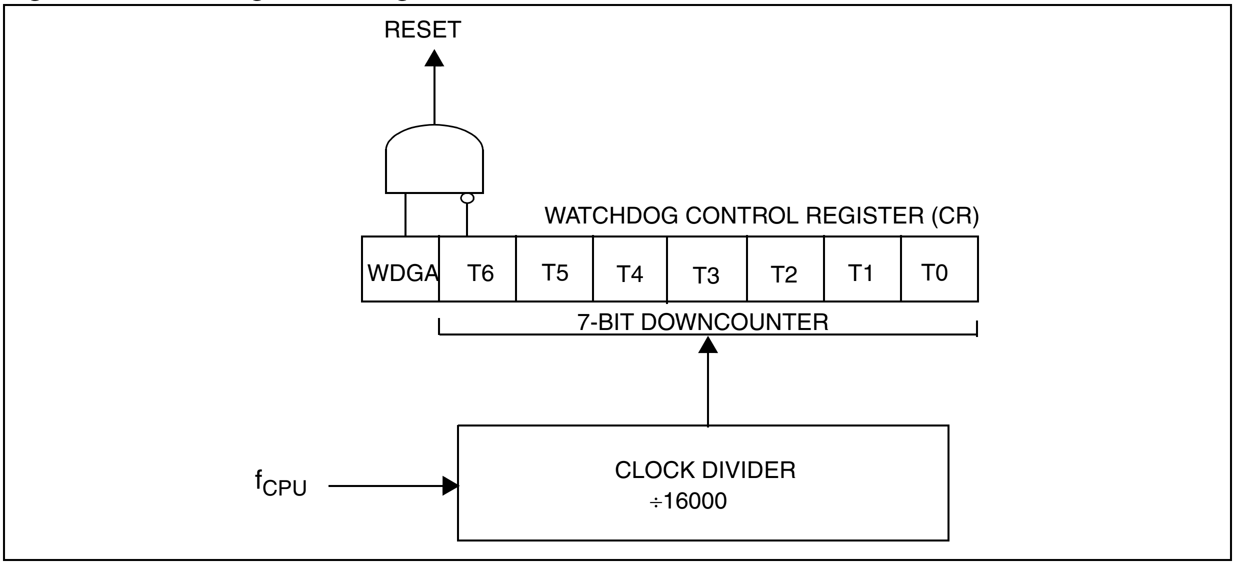
- Programmable free-running downcounter (64 increments of 16000 CPU cycles)
- Programmable reset
- Reset (if Watchdog activated) when the T6 bit reaches zero
- Optional reset on HALT instruction (configurable by option byte)
- Hardware Watchdog selectable by option byte

11.1.3 Functional Description

The counter value stored in the CR register (bits T[6:0]) is decremented every 16000 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 30µs.

Figure 32. Watchdog Block Diagram



ON-CHIP PERIPHERALS (Cont'd)

The application program must write in the CR register at regular intervals during normal operation to prevent an MCU reset. This downcounter is free-running: It counts down, even if the Watchdog is disabled. The value to be stored in the CR register must be between FFh and C0h (see Table 12 .Watchdog Timing):

– 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.

Following a reset, the Watchdog is disabled. Once activated, it can be disabled only by a reset.

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

If the Watchdog is activated, the HALT instruction generates a Reset.

Table 12.Watchdog Timing

f_{CPU} = 8 MHz		
WDG Counter Code	**min [ms]**	**max [ms]**
C0h	1	2
FFh	127	128

Notes:

1. The timing variation shown in Table 12 is due to the unknown status of the prescaler when writing to the CR register.
2. The number of CPU clock cycles applied during the RESET phase (256 or 4096) must be taken into account in addition to these timings.

11.1.4 Hardware Watchdog Option

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

Refer to the Option Byte description in section 15 on page 126.

11.1.4.1 Using Halt Mode with the WDG (WDGHALT Option)

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

11.1.5 Interrupts

None.

11.1.6 Register Description

CONTROL REGISTER (WDGCR)

Read/Write

Reset Value: 0111 1111 (7Fh)

7							0
WDGA	T6	T5	T4	T3	T2	T1	T0

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

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

Bits 6:0 = **T[6:0]** *7-bit 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 13. Watchdog Timer Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
002Eh	**WDGCR** Reset Value	WDGA 0	T6 1	T5 1	T4 1	T3 1	T2 1	T1 1	T0 1

11.2 DUAL 12-BIT AUTORELOAD TIMER 3 (AT3)

11.2.1 Introduction

The 12-bit Autoreload Timer can be used for general-purpose timing functions. It is based on one or two free-running 12-bit upcounters with an input capture register and four PWM output channels. There are seven external pins:

– 4 PWM outputs

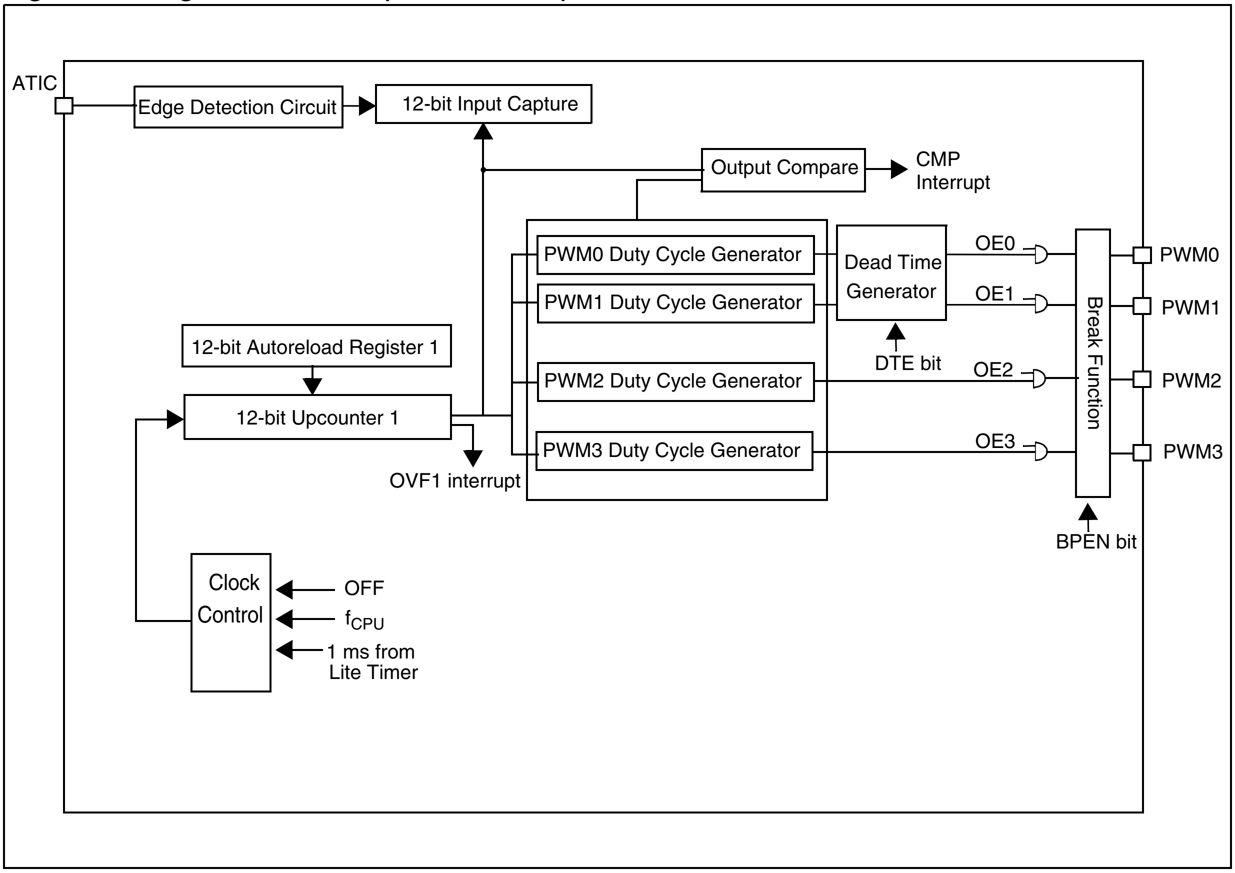
– ATIC/LTIC pins for the Input Capture function

– BREAK pin for forcing a break condition on the PWM outputs

11.2.2 Main Features

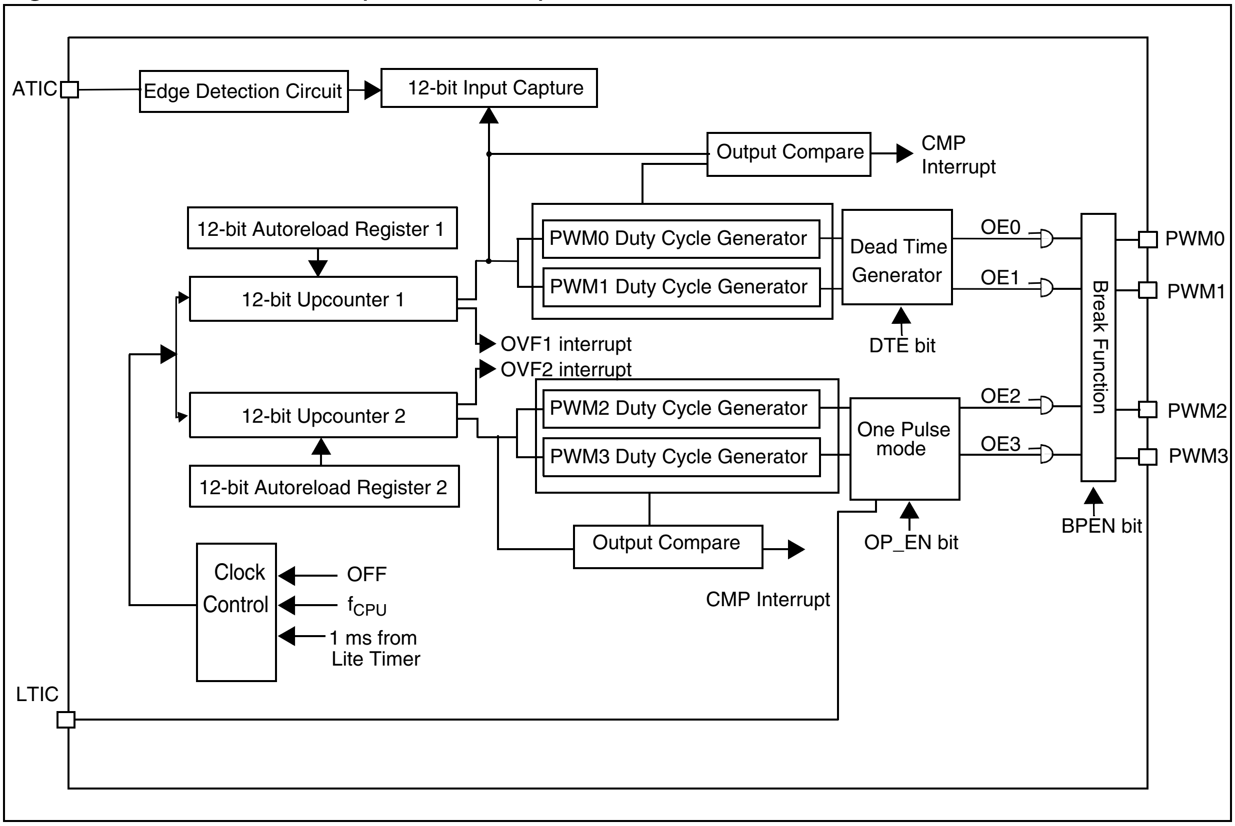
- Single Timer or Dual Timer mode with two 12-bit upcounters (CNTR1/CNTR2) and two 12-bit autoreload registers (ATR1/ATR2)
- Maskable overflow interrupts
- PWM mode
  – Generation of four independent PWMx signals
  – Dead time generation for Half-Bridge driving mode with programmable dead time
  – Frequency 2 kHz to 4 MHz (@ 8 MHz f_{CPU})
  – Programmable duty-cycles
  – Polarity control
  – Programmable output modes
- Output Compare Mode
- Input Capture Mode
  – 12-bit input capture register (ATICR)
  – Triggered by rising and falling edges
  – Maskable IC interrupt
  – Long range input capture
- Break control
- Flexible Clock control
- One Pulse mode on PWM2/3 (available only on Flash devices)
- Force Update (available only on Flash devices)

Figure 33. Single Timer Mode (ENCNTR2 = 0)



DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

Figure 34. Dual Timer Mode (ENCNTR2 = 1)



DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

11.2.3 Functional Description

11.2.3.1 PWM Mode

This mode allows up to four Pulse Width Modulated signals to be generated on the PWMx output pins.

PWM Frequency

The four PWM signals can have the same frequency (f_{PWM}) or can have two different frequencies. This is selected by the ENCNTR2 bit which enables single timer or dual timer mode (see Figure 33 and Figure 34).

The frequency is controlled by the counter period and the ATR register value. In dual timer mode, PWM2 and PWM3 can be generated with a different frequency controlled by CNTR2 and ATR2.

$$f_{PWM} = f_{COUNTER} / (4096 - ATR)$$

Following the above formula,

– If $f_{COUNTER}$ is 4 MHz, the maximum value of f_{PWM} is 2 MHz (ATR register value = 4094),the minimum value is 1 kHz (ATR register value = 0).

– If $f_{COUNTER}$ is 32 MHz, the maximum value of f_{PWM} is 8 MHz (ATR register value = 4092), the minimum value is 8 kHz (ATR register value = 0).

Notes:

1. The maximum value of ATR is 4094 because it must be lower than the DC4R value, which in this case must be 4095.

2. To update the DCRx registers at 32 MHz, the following precautions must be taken:

– if the PWM frequency is < 1 MHz and the TRANx bit is set asynchronously, it should be set twice after a write to the DCRx registers.

– if the PWM frequency is > 1 MHz, the TRANx bit should be set along with FORCEx bit with the same instruction (use a load instruction and not two bset instructions).

Duty Cycle

The duty cycle is selected by programming the DCRx registers. These are preload registers. The DCRx values are transferred in Active duty cycle registers after an overflow event if the corresponding transfer bit (TRANx bit) is set.

The TRAN1 bit controls the PWMx outputs driven by Counter 1 and the TRAN2 bit controls the PWMx outputs driven by Counter 2.

PWM generation and output compare are done by comparing these active DCRx values with the counter.

The maximum available resolution for the PWMx duty cycle is:

$$Resolution = 1 / (4096 - ATR)$$

where ATR is equal to 0. With this maximum resolution, 0% and 100% duty cycle can be obtained by changing the polarity.

At reset, the counter starts counting from 0.

When an upcounter overflow occurs (OVF event), the preloaded Duty cycle values are transferred to the active Duty Cycle registers and the PWMx signals are set to a high level. When the upcounter matches the active DCRx value, the PWMx signals are set to a low level. To obtain a signal on a PWMx pin, the contents of the corresponding active DCRx register must be greater than the contents of the ATR register.

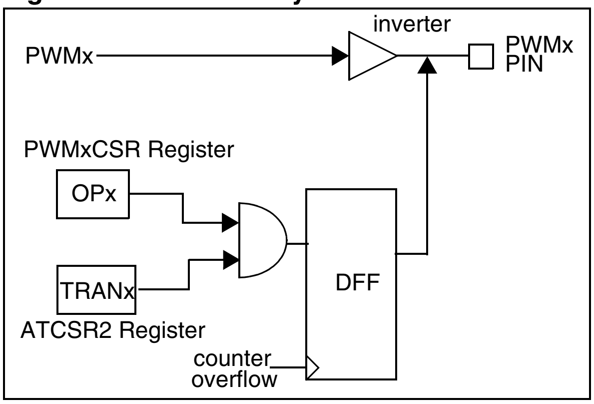
Note for ROM devices only: The PWM can be enabled/disabled only in overflow ISR, otherwise the first pulse of PWM can be different from expected one because no force overflow function is present.

The maximum value of ATR is 4094 because it must be lower than the DCR value, which in this case must be 4095.

Polarity Inversion

The polarity bits can be used to invert any of the four output signals. The inversion is synchronized with the counter overflow if the corresponding transfer bit in the ATCSR2 register is set (reset value). See Figure 35.

Figure 35. PWM Polarity Inversion



The Data Flip Flop (DFF) applies the polarity inversion when triggered by the counter overflow input.

Output Control

The PWMx output signals can be enabled or disabled using the OEx bits in the PWMCR register.

DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

Figure 36. PWM Function

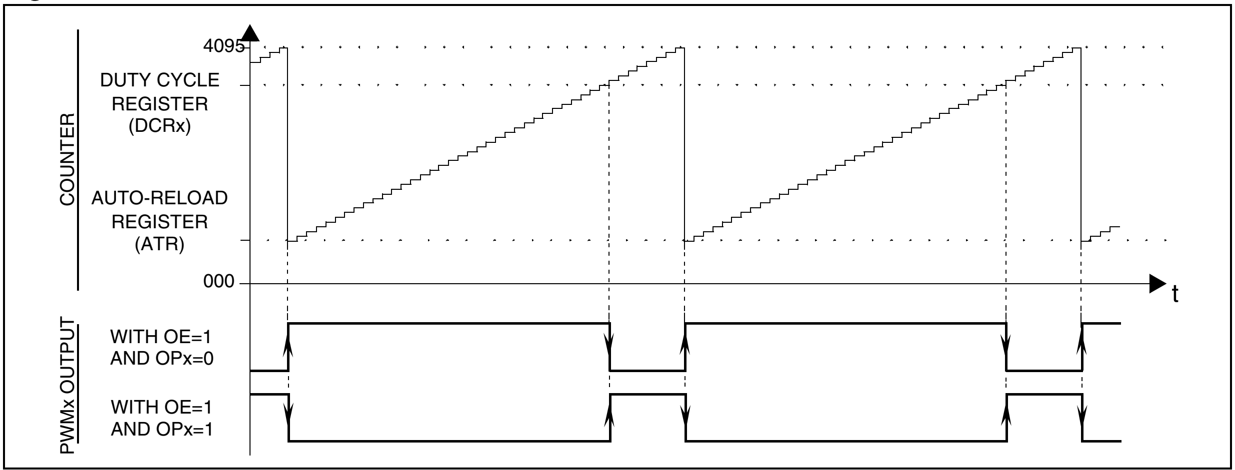
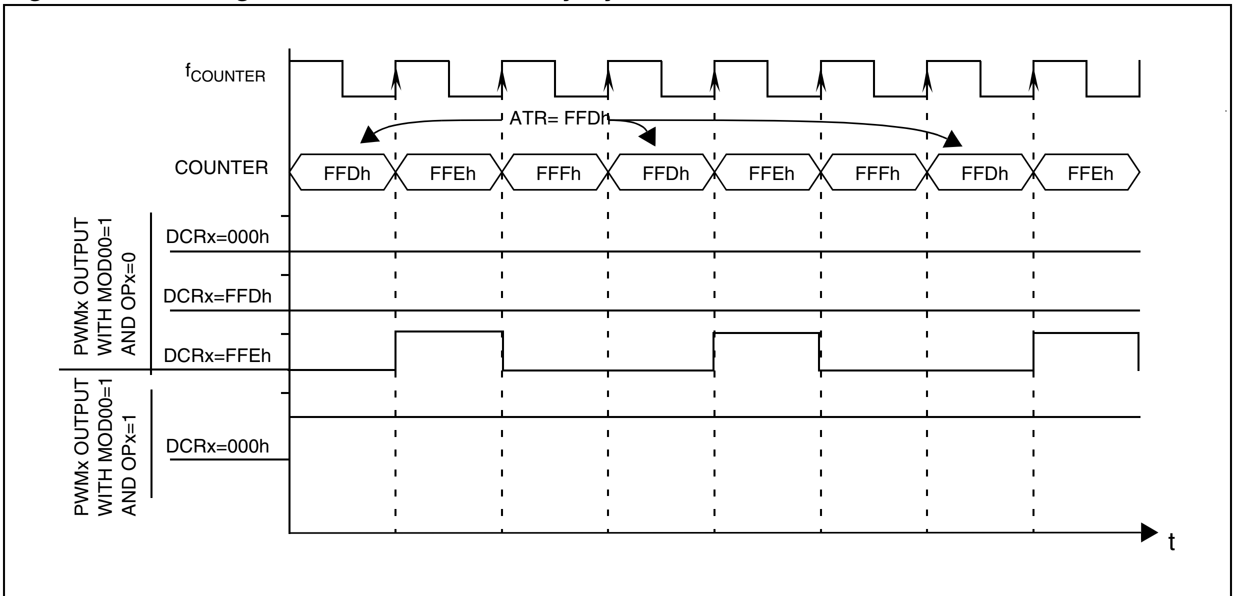


Figure 37. PWM Signal from 0% to 100% Duty Cycle



DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

11.2.3.2 Dead Time Generation

A dead time can be inserted between PWM0 and PWM1 using the DTGR register. This is required for half-bridge driving where PWM signals must not be overlapped. The non-overlapping PWM0/ PWM1 signals are generated through a programmable dead time by setting the DTE bit.

Dead time value = DT[6:0] x Tcounter1

DTGR[7:0] is buffered inside so as to avoid deforming the current PWM cycle. The DTGR effect will take place only after an overflow.

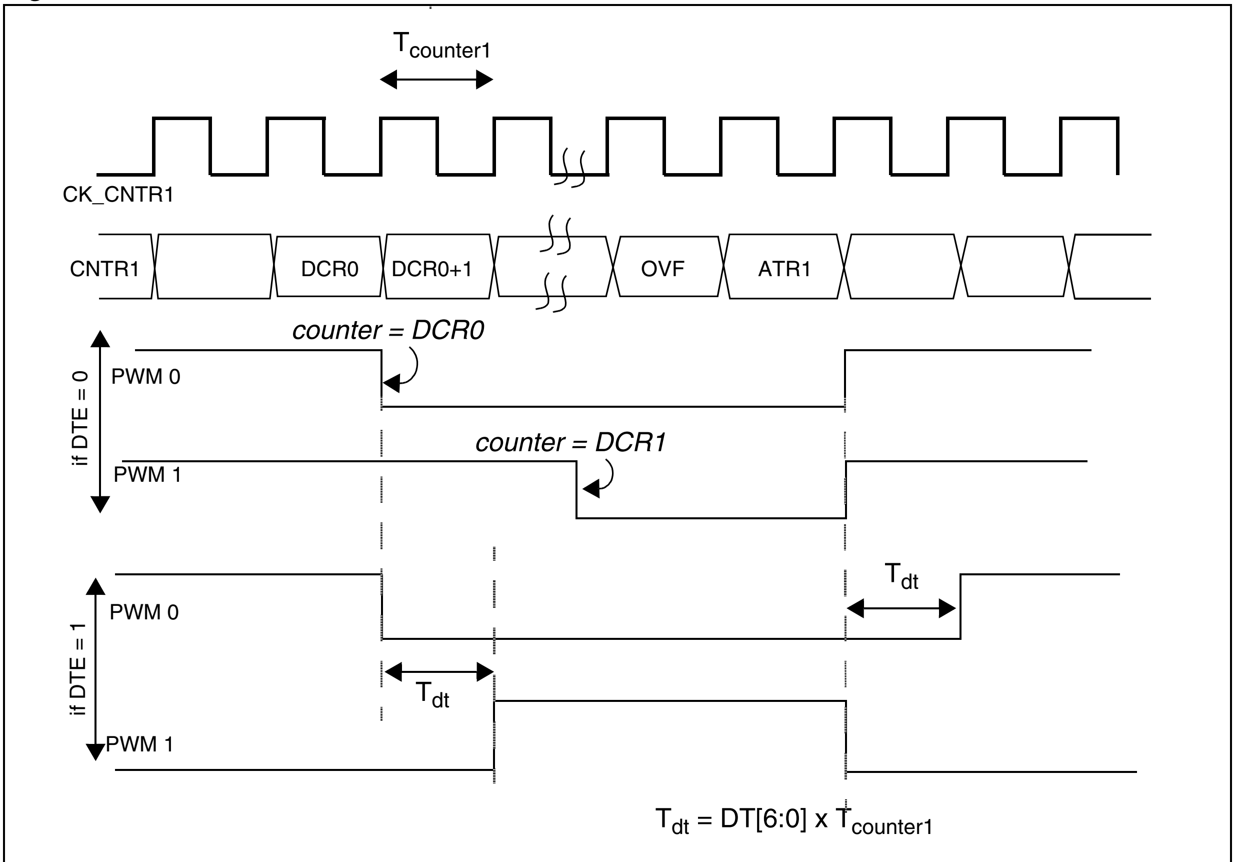
Notes:

1. Dead time is generated only when DTE = 1 and DT[6:0] ≠ 0. If DTE is set and DT[6:0] = 0, PWM output signals will be at their reset state.

2. Half-Bridge driving is possible only if polarities of PWM0 and PWM1 are not inverted, that is, if OP0 and OP1 are not set. If polarity is inverted, overlapping PWM0/PWM1 signals will be generated.

3. Dead Time generation does not work at 1msec timebase.

Figure 38. Dead Time Generation



In the above example, when the DTE bit is set:

– PWM goes low at DCR0 match and goes high at ATR1+Tdt

– PWM1 goes high at DCR0+Tdt and goes low at ATR match.

With this programmable delay (Tdt), the PWM0 and PWM1 signals which are generated are not overlapped.

DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

11.2.3.3 Break Function

The break function can be used to perform an emergency shutdown of the application being driven by the PWM signals.

The break function is activated by the external BREAK pin. In order to use the break function it must be previously enabled by software setting the BPEN bit in the BREAKCR register.

The Break active level can be programmed by the BREDGE bit in the BREAKCR register (in ROM devices the active level is not programmable; the break active level is low level). When an active level is detected on the BREAK pin, the BA bit is set and the break function is activated. In this case, the PWM signals are forced to BREAK value if the respective OEx bit is set in the PWMCR register.

Software can set the BA bit to activate the break function without using the BREAK pin. The BREN1 and BREN2 bits in the BREAKEN Register are used to enable the break activation on the two counters respectively. In Dual Timer Mode, the break for PWM2 and PWM3 is enabled by the BREN2 bit. In Single Timer Mode, the BREN1 bit enables the break for all PWM channels. In ROM devices, BREN1 and BREN2 are both forced by hardware at high level and all PWMs are enabled.

When a break function is activated (BA bit = 1 and BREN1/BREN2 = 1):

– The break pattern (PWM[3:0] bits in the BREAKCR is forced directly on the PWMx output pins if respective OEx is set (after the inverter).

– The 12-bit PWM counter CNTR1 is put to its reset value, that is 00h (if BREN1 = 1).

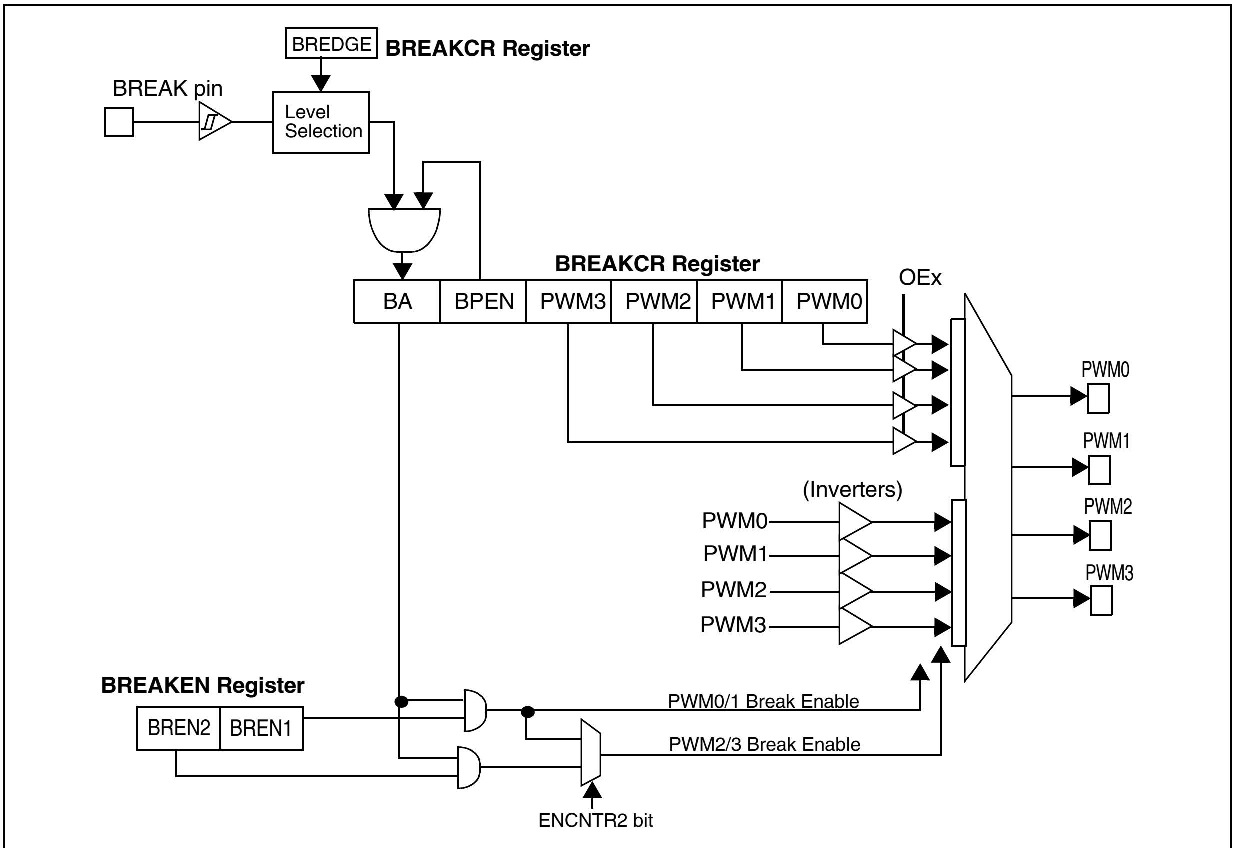
– The 12-bit PWM counter CNTR2 is put to its reset value, that is 00h (if BREN2 = 1).

– ATR1, ATR2, Preload and Active DCRx are put to their reset values.

– Counters stop counting.

When the break function is deactivated after applying the break (BA bit goes from 1 to 0 by software), the Timer takes the control of the PWM ports.

Figure 39. Block Diagram of Break Function



DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

11.2.3.4 Output Compare Mode

To use this function, load a 12-bit value in the Preload DCRxH and DCRxL registers.

When the 12-bit upcounter CNTR1 reaches the value stored in the Active DCRxH and DCRxL registers, the CMPFx bit in the PWMxCSR register is set and an interrupt request is generated if the CMPIE bit is set.

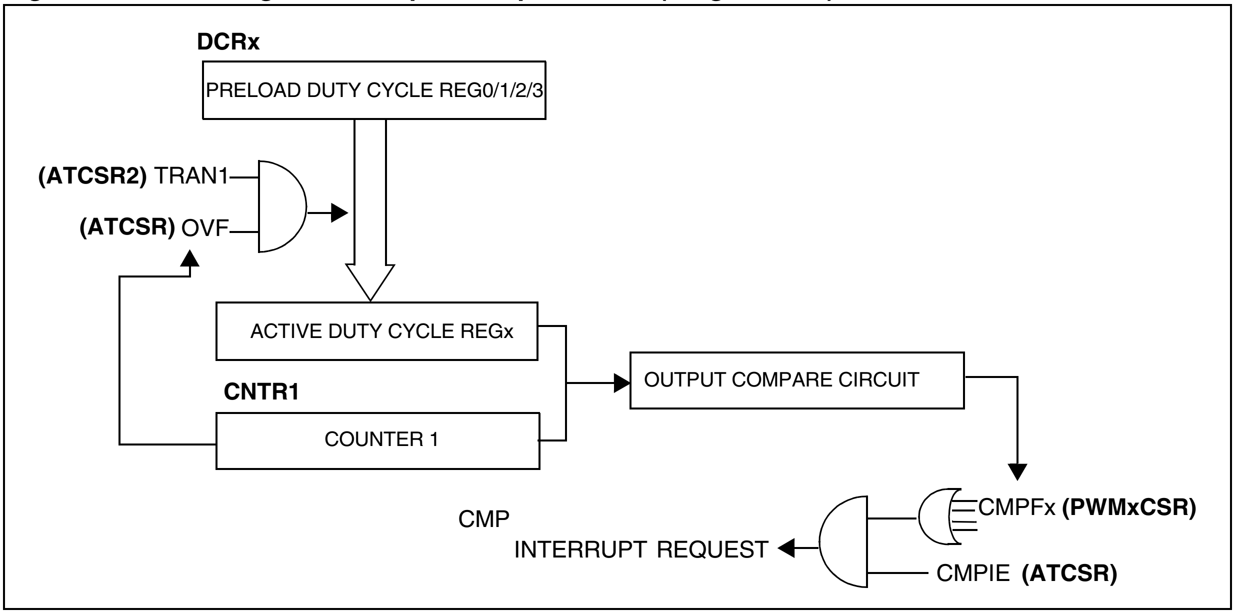
In Single Timer mode the output compare function is performed only on CNTR1. The difference between both the modes is that in Single Timer mode, CNTR1 can be compared with any of the four DCR registers, and in Dual Timer mode, CNTR1 is compared with DCR0 or DCR1 and CNTR2 is compared with DCR2 or DCR3. In ROM devices, the CNTR2 counter is not used for this comparison.

Notes:

1. The output compare function is only available for DCRx values other than 0 (reset value).

2. Duty cycle registers are buffered internally. The CPU writes in Preload Duty Cycle Registers and these values are transferred to Active Duty Cycle Registers after an overflow event if the corresponding transfer bit (TRANx bit) is set. Output compare is done by comparing these active DCRx values with the counters.

Figure 40. Block Diagram of Output Compare Mode (Single Timer)



DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

11.2.3.5 Input Capture Mode

The 12-bit ATICR register is used to latch the value of the 12-bit free running upcounter CNTR1 after a rising or falling edge is detected on the ATIC pin. When an input capture occurs, the ICF bit is set and the ATICR register contains the value of the free running upcounter. An IC interrupt is generated if the ICIE bit is set. The ICF bit is reset by reading the ATICRH/ATICRL register when the ICF bit is set. The ATICR is a read only register and always contains the free running upcounter value which corresponds to the most recent input capture. Any further input capture is inhibited while the ICF bit is set.

Figure 41. Block Diagram of Input Capture Mode

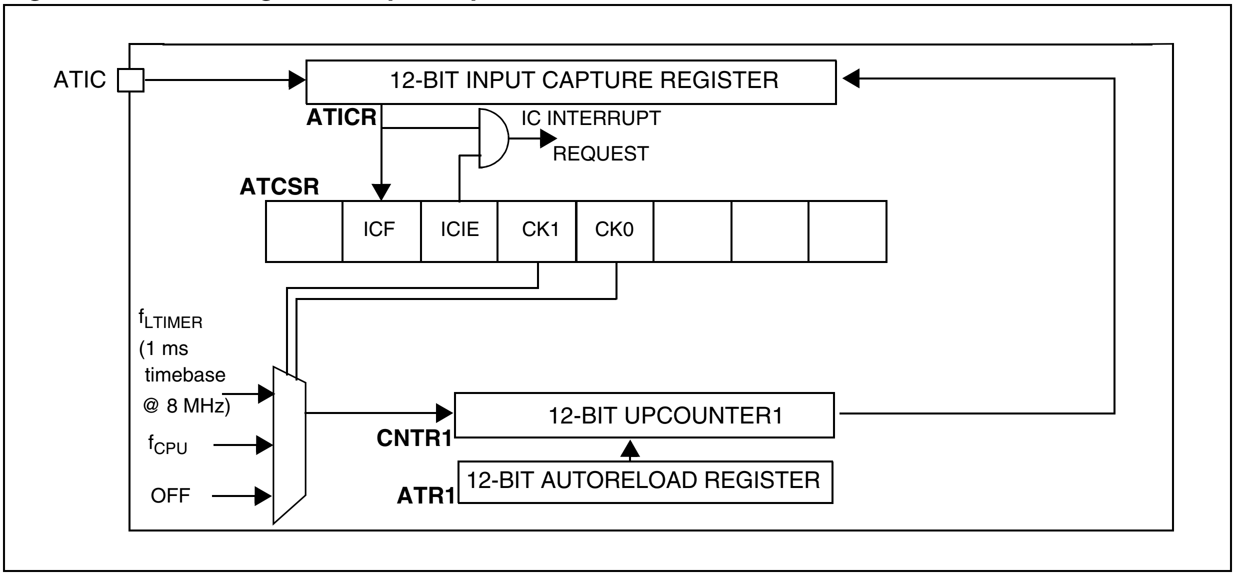
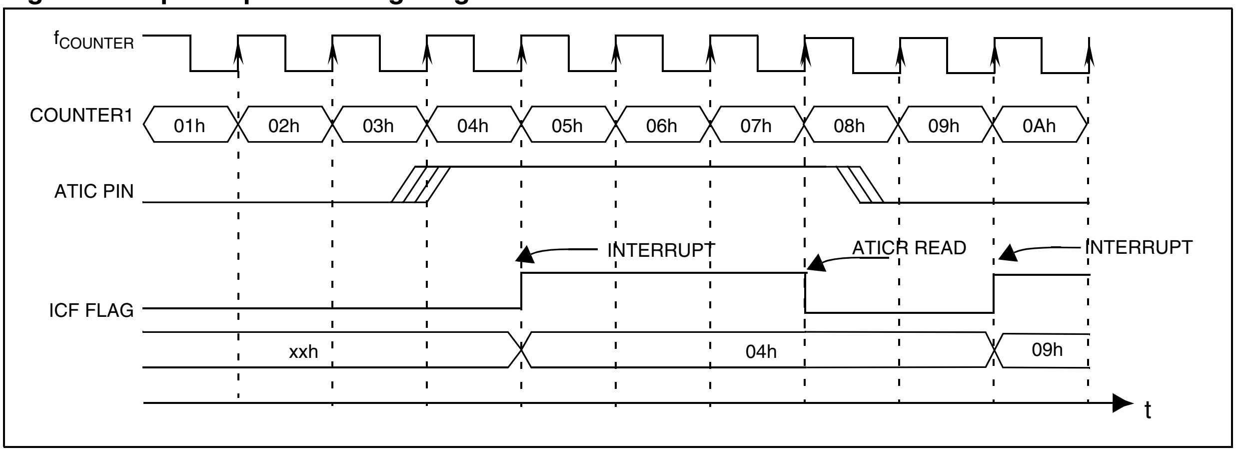


Figure 42. Input Capture Timing Diagram



DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

■ **Long Input Capture**

Pulses that last more than 8µs can be measured with an accuracy of 4µs if f_{OSC} = 8 MHz under the following conditions:

– The 12-bit AT3 Timer is clocked by the Lite Timer (RTC pulse: CK[1:0] = 01 in the ATCSR register)

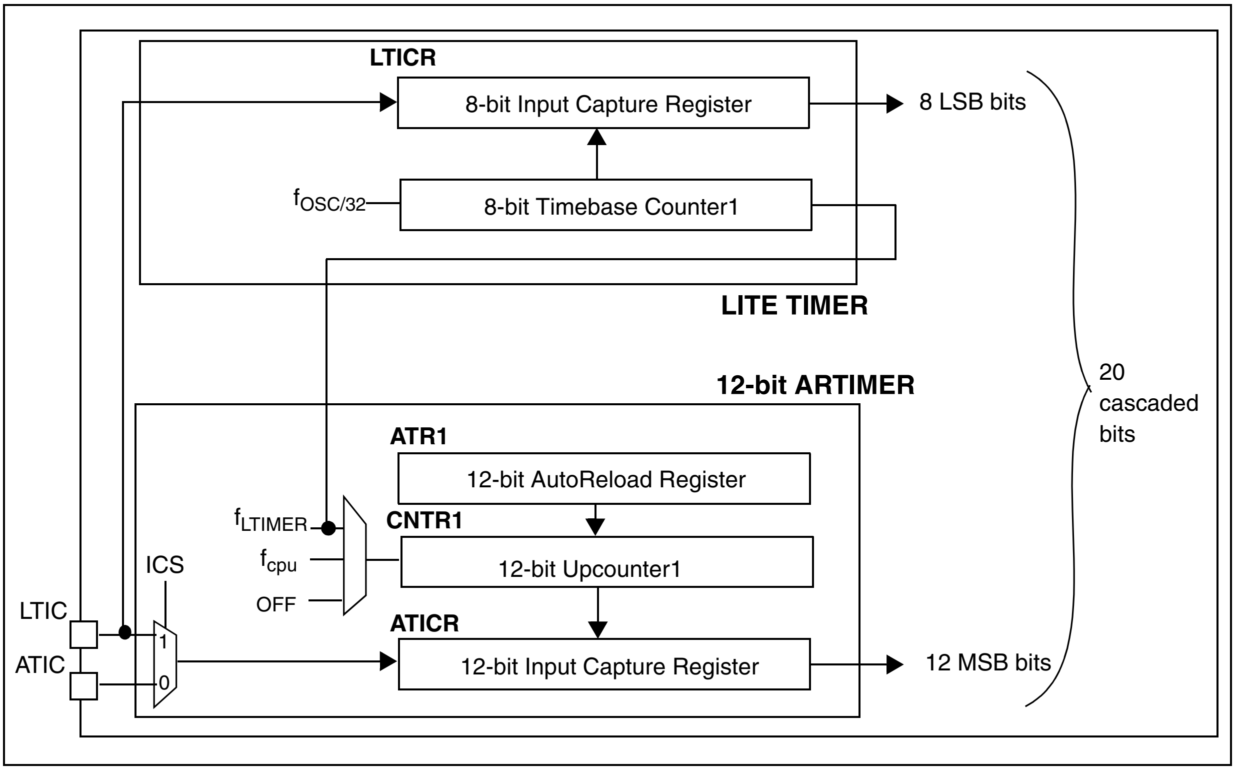
– The ICS bit in the ATCSR2 register is set so that the LTIC pin is used to trigger the AT3 Timer capture.

– The signal to be captured is connected to LTIC pin

– Input Capture registers LTICR, ATICRH and ATICRL are read

This configuration allows to cascade the Lite Timer and the 12-bit AT3 Timer to get a 20-bit input capture value. Refer to Figure 43.

Figure 43. Long Range Input Capture Block Diagram



Notes:

1. Since the input capture flags (ICF) for both timers (AT3 Timer and LT Timer) are set when signal transition occurs, software must mask one interrupt by clearing the corresponding ICIE bit before setting the ICS bit.

2. If the ICS bit changes (from 0 to 1 or from 1 to 0), a spurious transition might occur on the input capture signal because of different values on LTIC and ATIC. To avoid this situation, it is recommended to do the following:

– First, reset both ICIE bits.

– Then set the ICS bit.

– Reset both ICF bits.

– Finally, set the ICIE bit of desired interrupt.

3. How to compute a pulse length with long input capture feature:

As both timers are used, computing a pulse length is not straight-forward. The procedure is as follows:

– At the first input capture on the rising edge of the pulse, we assume that values in the registers are as follows:
LTICR = LT1
ATICRH = ATH1
ATICRL = ATL1
Hence ATICR1 [11:0] = ATH1 & ATL1

Refer to Figure 44 on page 61.

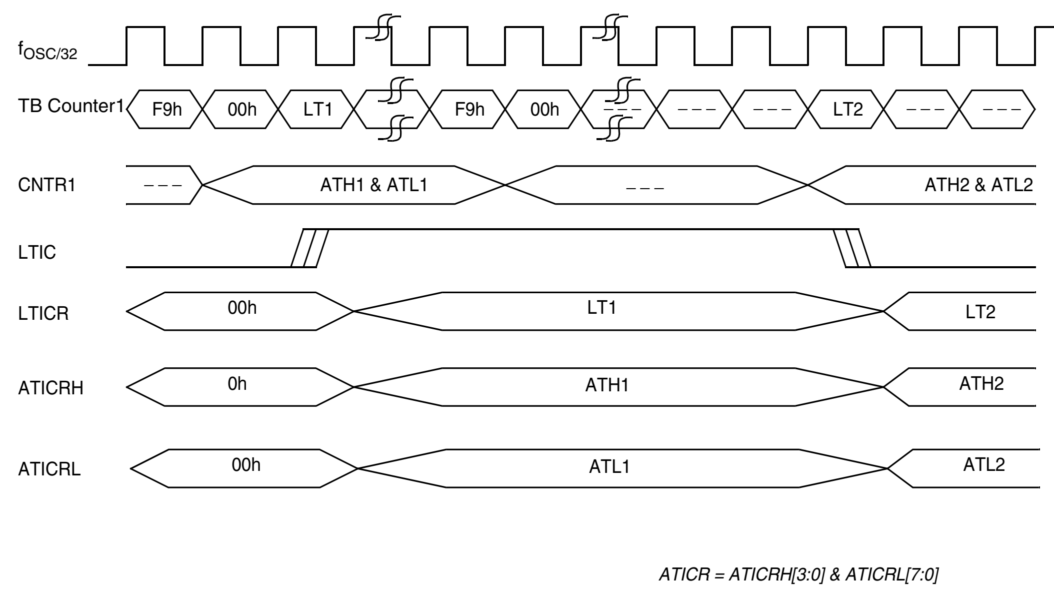
DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

– At the second input capture on the falling edge of the pulse, we assume that the values in the registers are as follows:
LTICR = LT2
ATICRH = ATH2
ATICRL = ATL2
Hence ATICR2 [11:0] = ATH2 & ATL2

Now pulse width P between first capture and second capture will be:

P = decimal (F9 – LT1 + LT2 + 1) * 0.004ms + decimal ((FFF * N) + N + ATICR2 - ATICR1 – 1) * 1ms where N = No of overflows of 12-bit CNTR1.

Figure 44. Long Range Input Capture Timing Diagram



DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

11.2.3.6 One Pulse Mode (available only on Flash devices)

One Pulse mode can be used to control PWM2/3 signal with an external LTIC pin. This mode is available only in dual timer mode that is only for CNTR2, when the OP_EN bit in PWM3CSR register is set.

One Pulse mode is activated by the external LTIC input. The active edge of the LTIC pin is selected by the OPEDGE bit in the PWM3CSR register.

After obtaining the active edge of the LTIC pin, CNTR2 is reset (000h) and PWM3 is set to high. CNTR2 starts counting from 000h and when it reaches the active DCR3 value, PWM3 goes low. Until this time, any further transitions on the LTIC signal will have no effect. If there are LTIC transitions after CNTR2 reaches the DCR3 value, CNTR2 is reset again and PWM3 goes high.

If there are no more LTIC active edges after the first active edge, CNTR2 counts until it reaches the ARR2 value, it is then reset and PWM3 is set to high. The counter again starts counting from 000h. When it reaches the active DCR3 value, PWM3 goes low, after which the counter counts until it reaches ARR2, it is reset and PWM3 is set to high again, and the cycle continues in this manner.

The same operation applies for PWM2, but in this case the comparison is done on DCR2

OP_EN and OPEDGE bits take effect on the fly and are not synchronized with the Counter 2 overflow.

The output bit OP2/3 can be used to invert the polarity of PWM2/3 in One Pulse mode. The update of these bits (OP2/3) is synchronized with the Counter 2 overflow, provided the TRAN2 bit is set.

Notes:

1. The time taken from activation of LTIC input and CNTR2 reset is between 1 and 2 T_{cpu} cycles, that is 125n to 250ns (with 8 MHz f_{cpu}).

2. To avoid spurious interrupts, the LiteTimer input capture interrupt should be disabled while 12-bit ARTimer is in One Pulse mode.

3. Priority of various conditions is as follows for PWM3:

Break > One Pulse mode with active LTIC edge > Forced overflow by s/w > One Pulse mode without active LTIC edge > normal PWM operation.

4. It is possible to update DCR2/3 and OP2/3 at the Counter 2 reset because the update is synchronized with the counter reset. This is managed by the overflow interrupt which is generated if the counter is reset either due to ARR match or active pulse at LTIC pin.

5. DCR2/3 and OP2/3 update in One Pulse mode is done dynamically using force update in software.

6. DCR3 update in this mode is not synchronized with any event. That may lead to a longer next PWM3 cycle duration than expected just after the change (refer to Figure 47).

7. In One Pulse mode, the ATR2 value must be greater than the DCR2/3 value for PWM2/3 (opposite to normal PWM mode).

8. If there is an active edge on the LTIC pin after the counter has reset due to an ARR2 match, then the timer again is reset and appears as modified Duty cycle, depending on whether the new DCR value is less than or more than the previous value.

9. The TRAN2 bit should be set along with the FORCE2 bit with the same instruction after a write to the DCR register.

10. ARR2 value should be changed after an overflow in One Pulse mode to avoid any irregular PWM cycle.

11. When exiting from One Pulse mode, the OP_EN bit in the PWM3CSR register should be reset first and then the ENCNTR2 bit (if Counter 2 must be stopped).

How to enter One Pulse mode:

1. Load ATR2H/ATR2L with required value.
2. Load DCR3H/DCR3L for PWM3. ATR2 value must be greater than DCR3.
3. Set OP3 in PWM3CSR if polarity change is required.
4. Select CNTR2 by setting ENCNTR2 bit in ATCSR2.
5. Set TRAN2 bit in ATCSR2 to enable transfer.
6. "Wait for Overflow" by checking the OVF2 flag in ATCSR2.
7. Select counter clock using CK<1:0> bits in ATCSR.
8. Set OP_EN bit in PWM3CSR to enable One Pulse mode.
9. Enable PWM3 by OE3 bit of PWMCR.

The "Wait for Overflow" in step 6 can be replaced by forced update.

Follow the same procedure for PWM2 with the bits corresponding to PWM2.

Note: When break is applied in One Pulse mode, DUAL 12-BIT AUTORELOAD TIMER 3, CNTR2, DCR2/3 and ATR2 registers are reset. Conse-

DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

quently, these registers must be initialized again when break is removed.

Figure 45. Block Diagram of One Pulse Mode

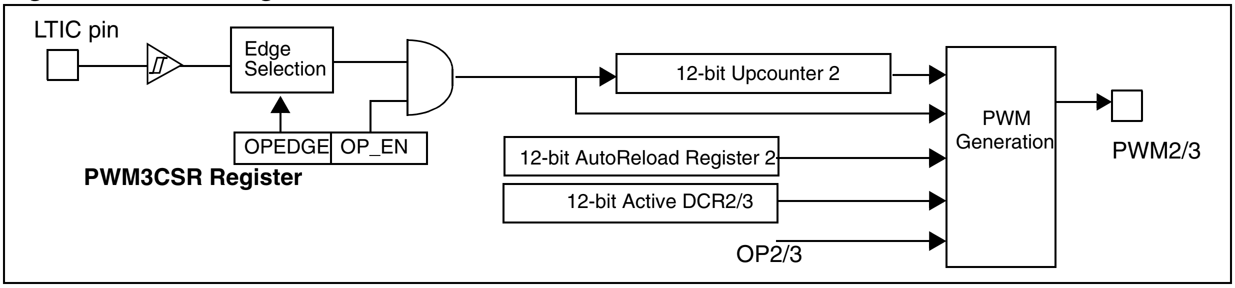


Figure 46. One Pulse Mode Timing Diagram

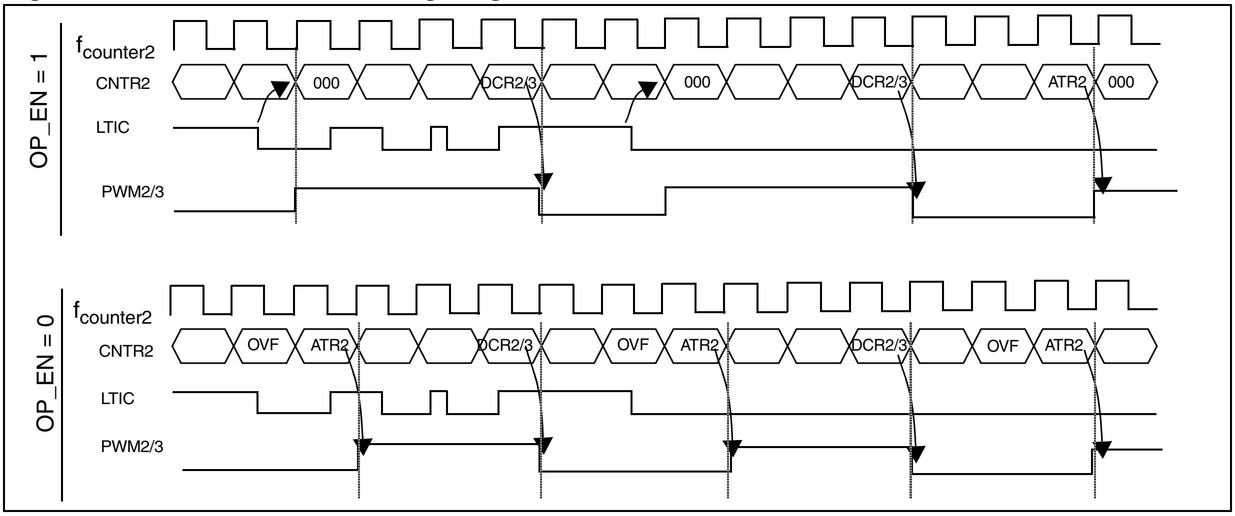
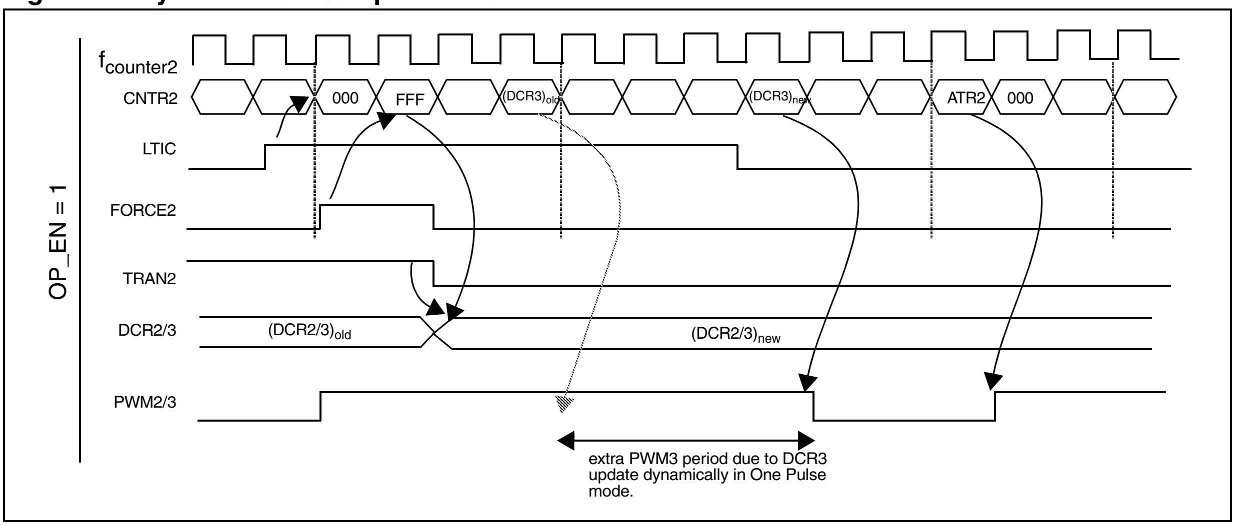


Figure 47. Dynamic DCR2/3 update in One Pulse Mode



DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

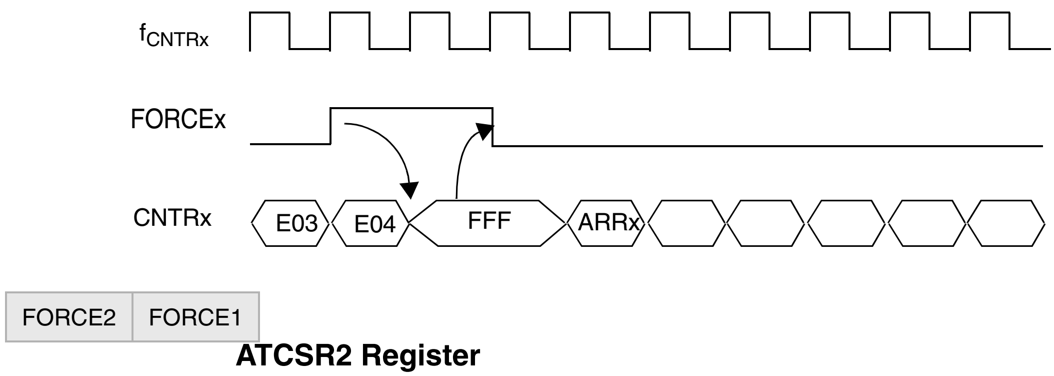
11.2.3.7 Force Update (available only on Flash devices)

In order not to wait for the $counter_x$ overflow to load the value into active DCRx registers, a programmable $counter_x$ overflow is provided. For both counters, a separate bit is provided which when set, starts the counters with the overflow value, that is FFFh. After overflow, the counters start counting from their respective autoreload register values.

These bits are FORCE1 and FORCE2 in the ATCSR2 register. FORCE1 is used to force an overflow on Counter 1 and FORCE2 is used for Counter 2. These bits are set by software and reset by hardware after the respective counter overflow event has occurred.

This feature can be used at any time. All related features such as PWM generation, Output Compare, Input Capture and One Pulse can be used this way.

Figure 48. Force Overflow Timing Diagram



11.2.4 Low Power Modes

Mode	Description
WAIT	No effect on AT timer
HALT	AT timer halted

11.2.5 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt	Exit from Active Halt
Overflow Event	OVF1	OVFIE1	Yes	No	Yes
AT3 IC Event	ICF	ICIE			No
CMP Event	CMPFx	CMPIE			
Overflow Event2	OVF2	OVFIE2			

Note: The CMP and AT3 IC events are connected to the same interrupt vector. The OVF event is mapped on a separate vector (see Interrupts chapter). They generate an interrupt if the enable bit is set in the ATCSR register and the interrupt mask in the CC register is reset (RIM instruction).

DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

11.2.6 Register Description

TIMER CONTROL STATUS REGISTER (ATCSR)
Read / Write
Reset Value: 0x00 0000 (x0h)

7							0
0	ICF	ICIE	CK1	CK0	OVF1	OVFIE1	CMPIE

Bit 7 = **Reserved,** must be kept cleared

Bit 6 = **ICF** *Input Capture Flag*
This bit is set by hardware and cleared by software by reading the ATICR register (a read access to ATICRH or ATICRL will clear this flag). Writing to this bit does not change the bit value.

0: No input capture
1: An input capture has occurred

Bit 5 = **ICIE** *IC Interrupt Enable*
This bit is set and cleared by software.

0: Input capture interrupt disabled
1: Input capture interrupt enabled

Bits 4:3 = **CK[1:0]** *Counter Clock Selection*
These bits are set and cleared by software and cleared by hardware after a reset. They select the clock frequency of the counter.

Counter Clock Selection	CK1	CK0
OFF	0	0
f_{LTIMER} (1 ms timebase @ 8 MHz)	0	1
f_{CPU}	1	0

Bit 2 = **OVF1** *Overflow Flag*
This bit is set by hardware and cleared by software by reading the ATCSR register. It indicates the transition of the counter CNTR1 from FFFh to ATR1 value.

0: No counter overflow occurred
1: Counter overflow occurred

Bit 1 = **OVFIE1** *Overflow Interrupt Enable*
This bit is read/write by software and cleared by hardware after a reset.

0: Overflow interrupt disabled.
1: Overflow interrupt enabled.

Bit 0 = **CMPIE** *Compare Interrupt Enable*
This bit is read/write by software and cleared by hardware after a reset. It can be used to mask the interrupt generated when any of the CMPFx bit is set.

0: Output compare interrupt disabled.
1: Output Compare interrupt enabled.

COUNTER REGISTER 1 HIGH (CNTR1H)
Read only
Reset Value: 0000 0000 (00h)

15							8
0	0	0	0	CNTR1_11	CNTR1_10	CNTR1_9	CNTR1_8

COUNTER REGISTER 1 LOW (CNTR1L)
Read only
Reset Value: 0000 0000 (00h)

7							0
CNTR1_7	CNTR1_6	CNTR1_5	CNTR1_4	CNTR1_3	CNTR1_2	CNTR1_1	CNTR1_0

Bits 15:12 = **Reserved,** must be kept cleared

Bits 11:0 = **CNTR1[11:0]** *Counter Value*
This 12-bit register is read by software and cleared by hardware after a reset. The counter CNTR1 increments continuously as soon as a counter clock is selected. To obtain the 12-bit CNTR1 value, software should read the counter value in two consecutive read operations, LSB first. When a counter overflow occurs, the counter restarts from the value specified in the ATR1 register.

DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

AUTORELOAD REGISTER (ATR1H)
Read / Write
Reset Value: 0000 0000 (00h)

15							8
0	0	0	0	ATR11	ATR10	ATR9	ATR8

AUTORELOAD REGISTER (ATR1L)
Read / Write
Reset Value: 0000 0000 (00h)

7							0
ATR7	ATR6	ATR5	ATR4	ATR3	ATR2	ATR1	ATR0

Bits 15:12 = **Reserved,** must be kept cleared

Bits 11:0 = **ATR1[11:0]** *Autoreload Register 1*
This is a 12-bit register which is written by software. The ATR1 register value is automatically loaded into the upcounter CNTR1 when an overflow occurs. The register value is used to set the PWM frequency.

PWM OUTPUT CONTROL REGISTER (PWMCR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
0	OE3	0	OE2	0	OE1	0	OE0

Bits 7:0 = **OE[3:0]** *PWMx Output Enable.*
These bits are set and cleared by software and cleared by hardware after a reset.

0: PWM mode disabled. PWMx Output Alternate Function disabled (I/O pin free for general purpose I/O)
1: PWM mode enabled

PWMx CONTROL STATUS REGISTER (PWMxCSR)
Read / Write
Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	OP_EN	OPEDGE	OPx	CMPFx

Bits 7:4 = **Reserved**, must be kept cleared

Bit 3 = **OP_EN** *One Pulse Mode Enable (not applicable to ROM devices)*
This bit is read/write by software and cleared by hardware after a reset. This bit enables the One Pulse feature for PWM2 and PWM3. **(Only available for PWM3CSR)**

0: One Pulse mode disable for PWM2/3.
1: One Pulse mode enable for PWM2/3.

Bit 2 = **OPEDGE** *One Pulse Edge Selection (not applicable to ROM devices)*

This bit is read/write by software and cleared by hardware after a reset. This bit selects the polarity of the LTIC signal for One Pulse feature. This bit will be effective only if OP_EN bit is set. **(Only available for PWM3CSR)**

0: Falling edge of LTIC is selected.
1: Rising edge of LTIC is selected.

Bit 1 = **OPx** *PWMx Output Polarity*
This bit is read/write by software and cleared by hardware after a reset. This bit selects the polarity of the PWM signal.

0: The PWM signal is not inverted.
1: The PWM signal is inverted.

Bit 0 = **CMPFx** PWMx *Compare Flag*
This bit is set by hardware and cleared by software by reading the PWMxCSR register. It indicates that the upcounter value matches the Active DCRx register value.

0: Upcounter value does not match DCRx value.
1: Upcounter value matches DCRx value.

DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

BREAK CONTROL REGISTER (BREAKCR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
0	BREDGE	BA	BPEN	PWM3	PWM2	PWM1	PWM0

Bit 7 = **Reserved,** must be kept cleared

Bit 6 = **BREDGE** *Break Input Edge Selection (not applicable to ROM devices)*
This bit is read/write by software and cleared by hardware after reset. It selects the active level of Break signal.

0: Low level of Break selected as active level.
1: High level of Break selected as active level.

Bit 5 = **BA** *Break Active*
This bit is read/write by software, cleared by hardware after reset and set by hardware when the BREAK pin is low. It activates/deactivates the Break function.

0: Break not active
1: Break active

Bit 4 = **BPEN** *Break Pin Enable*
This bit is read/write by software and cleared by hardware after Reset.

0: Break pin disabled
1: Break pin enabled

Bits 3:0 = **PWM[3:0]** *Break Pattern*
These bits are read/write by software and cleared by hardware after a reset. They are used to force the four PWMx output signals into a stable state when the Break function is active and corresponding OEx bit is set.

PWMx DUTY CYCLE REGISTER HIGH (DCRxH)

Read / Write

Reset Value: 0000 0000 (00h)

15							8
0	0	0	0	DCR11	DCR10	DCR9	DCR8

PWMx DUTY CYCLE REGISTER LOW (DCRxL)

Read / Write

Reset Value: 0000 0000 (00h)

7							0
DCR7	DCR6	DCR5	DCR4	DCR3	DCR2	DCR1	DCR0

Bits 15:12 = **Reserved,** must be kept cleared

Bits 11:0 = **DCRx[11:0]** *PWMx Duty Cycle Value*
This 12-bit value is written by software. It defines the duty cycle of the corresponding PWM output signal (see Figure 36).

In PWM mode (OEx = 1 in the PWMCR register) the DCR[11:0] bits define the duty cycle of the PWMx output signal (see Figure 36). In Output Compare mode, they define the value to be compared with the 12-bit upcounter value.

INPUT CAPTURE REGISTER HIGH (ATICRH)

Read only

Reset Value: 0000 0000 (00h)

15							8
0	0	0	0	ICR11	ICR10	ICR9	ICR8

INPUT CAPTURE REGISTER LOW (ATICRL)

Read only

Reset Value: 0000 0000 (00h)

7							0
ICR7	ICR6	ICR5	ICR4	ICR3	ICR2	ICR1	ICR0

Bits 15:12 = **Reserved,** must be kept cleared

DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

Bits 11:0 = **ICR[11:0]** *Input Capture Data*
This is a 12-bit register which is readable by software and cleared by hardware after a reset. The ATICR register contains the captured value of the 12-bit CNTR1 register when a rising or falling edge occurs on the ATIC or LTIC pin (depending on ICS). Capture will only be performed when the ICF flag is cleared.

BREAK ENABLE REGISTER (BREAKEN)

Read/Write
Reset Value: 0000 0011 (03h)

7							0
0	0	0	0	0	0	BREN2	BREN1

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

Bit 1 = **BREN2** *Break Enable for Counter 2 (forced at high level in ROM devices)*
This bit is read/write by software. It enables the break functionality for Counter 2 if BA bit is set in BREAKCR. It controls PWM2/3 if ENCNTR2 bit is set.

0: No Break applied for CNTR2
1: Break applied for CNTR2

Bit 0 = **BREN1** *Break Enable for Counter 1 (forced at high level in ROM devices)*
This bit is read/write by software. It enables the break functionality for Counter 1. If BA bit is set, it controls PWM0/1 by default, and controls PWM2/3 also if ENCNTR2 bit is reset.

0: No Break applied for CNTR1
1: Break applied for CNTR1

TIMER CONTROL REGISTER2 (ATCSR2)

Read/Write
Reset Value: 0000 0011 (03h)

7							0
FORCE 2	FORCE 1	ICS	OVFIE2	OVF2	ENCNT R2	TRAN2	TRAN1

Bit 7 = **FORCE2** *Force Counter 2 Overflow (not applicable to ROM devices)*
This bit is read/set by software. When set, it loads FFFh in the CNTR2 register. It is reset by hardware one CPU clock cycle after Counter 2 overflow has occurred.

0: No effect on CNTR2
1: Loads FFFh in CNTR2

Note: This bit must not be reset by software

Bit 6 = **FORCE1** *Force Counter 1 Overflow (forced at high level in ROM devices)*
This bit is read/set by software. When set, it loads FFFh in CNTR1 register. It is reset by hardware one CPU clock cycle after Counter 1 overflow has occurred.

0: No effect on CNTR1
1: Loads FFFh in CNTR1

Note: This bit must not be reset by software

Bit 5 = **ICS** *Input Capture Shorted*
This bit is read/write by software. It allows the AT-timer CNTR1 to use the LTIC pin for long input capture.

0: ATIC for CNTR1 input capture
1: LTIC for CNTR1 input capture

Bit 4 = **OVFIE2** *Overflow Interrupt 2 Enable*
This bit is read/write by software and controls the overflow interrupt of Counter 2.

0: Overflow interrupt disabled
1: Overflow interrupt enabled

Bit 3 = **OVF2** *Overflow Flag*
This bit is set by hardware and cleared by software by reading the ATCSR2 register. It indicates the transition of the Counter 2 from FFFh to ATR2 value.

0: No counter overflow occurred
1: Counter overflow occurred

Bit 2 = **ENCNTR2** *Enable Counter 2* for PWM2/3

This bit is read/write by software and switches the PWM2/3 operation to the CNTR2 counter. If this bit is set, PWM2/3 will be generated using CNTR2.

0: PWM2/3 is generated using CNTR1.
1: PWM2/3 is generated using CNTR2.

Note: Counter 2 becomes frozen when the ENCNTR2 bit is reset. When ENCNTR2 is set again, the counter will restart from the last value.

DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

Bit 1 = **TRAN2** *Transfer Enable 2*
This bit is read/write by software, cleared by hardware after each completed transfer and set by hardware after reset. It controls the transfers on CNTR2.

It allows the value of the Preload DCRx registers to be transferred to the Active DCRx registers after the next overflow event.

The OPx bits are transferred to the shadow OPx bits in the same way.

Notes:

1. DCR2/3 transfer is controlled using this bit if ENCNTR2 bit is set.
2. This bit must not be reset by software.

Bit 0 = **TRAN1** *Transfer Enable 1*
This bit is read/write by software, cleared by hardware after each completed transfer and set by hardware after reset. It controls the transfers on CNTR1. It allows the value of the Preload DCRx registers to be transferred to the Active DCRx registers after the next overflow event.

The OPx bits are transferred to the shadow OPx bits in the same way.

Notes:

1. DCR0,1 transfers are always controlled using this bit.
2. DCR2/3 transfer is controlled using this bit if ENCNTR2 is reset.
3.This bit must not be reset by software

AUTORELOAD REGISTER2 (ATR2H)

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

15							8
0	0	0	0	ATR11	ATR10	ATR9	ATR8

AUTORELOAD REGISTER2 (ATR2L)

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

7							0
ATR7	ATR6	ATR5	ATR4	ATR3	ATR2	ATR1	ATR0

Bits 15:12 = **Reserved,** must be kept cleared

Bits 11:0 = **ATR2[11:0]** *Autoreload Register 2*
This is a 12-bit register which is written by software. The ATR2 register value is automatically loaded into the upcounter CNTR2 when an overflow of CNTR2 occurs. The register value is used to set the PWM2/PWM3 frequency when ENCNTR2 is set.

DEAD TIME GENERATOR REGISTER (DTGR)

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

7							0
DTE	DT6	DT5	DT4	DT3	DT2	DT1	DT0

Bit 7 = **DTE** *Dead Time Enable*
This bit is read/write by software. It enables a dead time generation on PWM0/PWM1.

0: No Dead time insertion.
1: Dead time insertion enabled.

Bits 6:0 = **DT[6:0]** *Dead Time Value*
These bits are read/write by software. They define the dead time inserted between PWM0/PWM1. Dead time is calculated as follows:

Dead Time = DT[6:0] x Tcounter1

Note:

1. If DTE is set and DT[6:0] = 0, PWM output signals are at their reset state.

DUAL 12-BIT AUTORELOAD TIMER 3 (Cont'd)

Table 14. Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0D	**ATCSR** Reset Value	0	ICF 0	ICIE 0	CK1 0	CK0 0	OVF1 0	OVFIE1 0	CMPIE 0
0E	**CNTR1H** Reset Value	0	0	0	0	CNTR1_11 0	CNTR1_10 0	CNTR1_9 0	CNTR1_8 0
0F	**CNTR1L** Reset Value	CNTR1_7 0	CNTR1_8 0	CNTR1_7 0	CNTR1_6 0	CNTR1_3 0	CNTR1_2 0	CNTR1_1 0	CNTR1_0 0
10	**ATR1H** Reset Value	0	0	0	0	ATR11 0	ATR10 0	ATR9 0	ATR8 0
11	**ATR1L** Reset Value	ATR7 0	ATR6 0	ATR5 0	ATR4 0	ATR3 0	ATR2 0	ATR1 0	ATR0 0
12	**PWMCR** Reset Value	0	OE3 0	0	OE2 0	0	OE1 0	0	OE0 0
13	**PWM0CSR** Reset Value	0	0	0	0	0	0	OP0 0	CMPF0 0
14	**PWM1CSR** Reset Value	0	0	0	0	0	0	OP1 0	CMPF1 0
15	**PWM2CSR** Reset Value	0	0	0	0	0	0	OP2 0	CMPF2 0
16	**PWM3CSR** Reset Value	0	0	0	0	OP_EN 0	OPEDGE 0	OP3 0	CMPF3 0
17	**DCR0H** Reset Value	0	0	0	0	DCR11 0	DCR10 0	DCR9 0	DCR8 0
18	**DCR0L** Reset Value	DCR7 0	DCR6 0	DCR5 0	DCR4 0	DCR3 0	DCR2 0	DCR1 0	DCR0 0
19	**DCR1H** Reset Value	0	0	0	0	DCR11 0	DCR10 0	DCR9 0	DCR8 0
1A	**DCR1L** Reset Value	DCR7 0	DCR6 0	DCR5 0	DCR4 0	DCR3 0	DCR2 0	DCR1 0	DCR0 0
1B	**DCR2H** Reset Value	0	0	0	0	DCR11 0	DCR10 0	DCR9 0	DCR8 0
1C	**DCR2L** Reset Value	DCR7 0	DCR6 0	DCR5 0	DCR4 0	DCR3 0	DCR2 0	DCR1 0	DCR0 0
1D	**DCR3H** Reset Value	0	0	0	0	DCR11 0	DCR10 0	DCR9 0	DCR8 0
1E	**DCR3L** Reset Value	DCR7 0	DCR6 0	DCR5 0	DCR4 0	DCR3 0	DCR2 0	DCR1 0	DCR0 0
1F	**ATICRH** Reset Value	0	0	0	0	ICR11 0	ICR10 0	ICR9 0	ICR8 0
20	**ATICRL** Reset Value	ICR7 0	ICR6 0	ICR5 0	ICR4 0	ICR3 0	ICR2 0	ICR1 0	ICR0 0

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
21	**ATCSR2** Reset Value	FORCE2 0	FORCE1 0	ICS 0	OVFIE2 0	OVF2 0	ENCNTR2 0	TRAN2 1	TRAN1 1
22	**BREAKCR** Reset Value	BRSEL 0	BREDGE 0	BA 0	BPEN 0	PWM3 0	PWM2 0	PWM1 0	PWM0 0
23	**ATR2H** Reset Value	0	0	0	0	ATR11 0	ATR10 0	ATR9 0	ATR8 0
24	**ATR2L** Reset Value	ATR7 0	ATR6 0	ATR5 0	ATR4 0	ATR3 0	ATR2 0	ATR1 0	ATR0 0
25	**DTGR** Reset Value	DTE 0	DT6 0	DT5 0	DT4 0	DT3 0	DT2 0	DT1 0	DT0 0
26	**BREAKEN** Reset Value	0	0	0	0	0	0	BREN2 1	BREN1 1

11.3 LITE TIMER 2 (LT2)

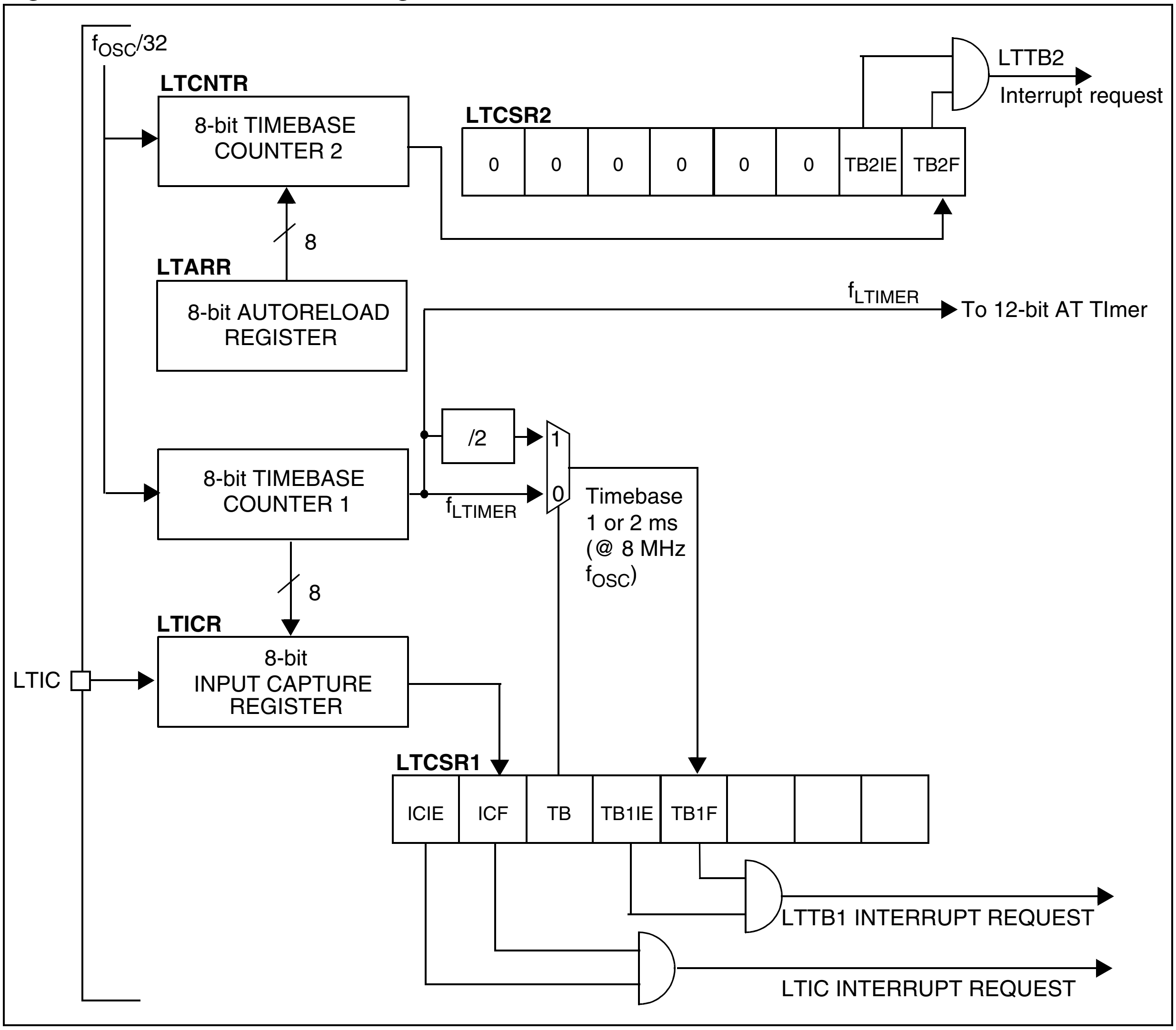
11.3.1 Introduction

The Lite Timer can be used for general-purpose timing functions. It is based on two free-running 8-bit upcounters and an 8-bit input capture register.

11.3.2 Main Features

- Realtime Clock
  – One 8-bit upcounter 1 ms or 2 ms timebase period (@ 8 MHz f_{OSC})
  – One 8-bit upcounter with autoreload and programmable timebase period from 4µs to 1.024ms in 4µs increments (@ 8 MHz f_{OSC})
  – 2 Maskable timebase interrupts
- Input Capture
  – 8-bit input capture register (LTICR)
  – Maskable interrupt with wake-up from Halt mode capability

Figure 49. Lite Timer 2 Block Diagram



LITE TIMER (Cont'd)

11.3.3 Functional Description

11.3.3.1 Timebase Counter 1

The 8-bit value of Counter 1 cannot be read or written by software. After an MCU reset, it starts incrementing from 0 at a frequency of $f_{OSC}/32$. An overflow event occurs when the counter rolls over from F9h to 00h. If f_{OSC} = 8 MHz, then the time period between two counter overflow events is 1 ms. This period can be doubled by setting the TB bit in the LTCSR1 register.

When Counter 1 overflows, the TB1F bit is set by hardware and an interrupt request is generated if the TB1IE bit is set. The TB1F bit is cleared by software reading the LTCSR1 register.

11.3.3.2 Input Capture

The 8-bit input capture register is used to latch the free-running upcounter (Counter 1) 1 after a rising or falling edge is detected on the LTIC pin. When an input capture occurs, the ICF bit is set and the LTICR1 register contains the MSB of Counter 1. An interrupt is generated if the ICIE bit is set. The ICF bit is cleared by reading the LTICR register.

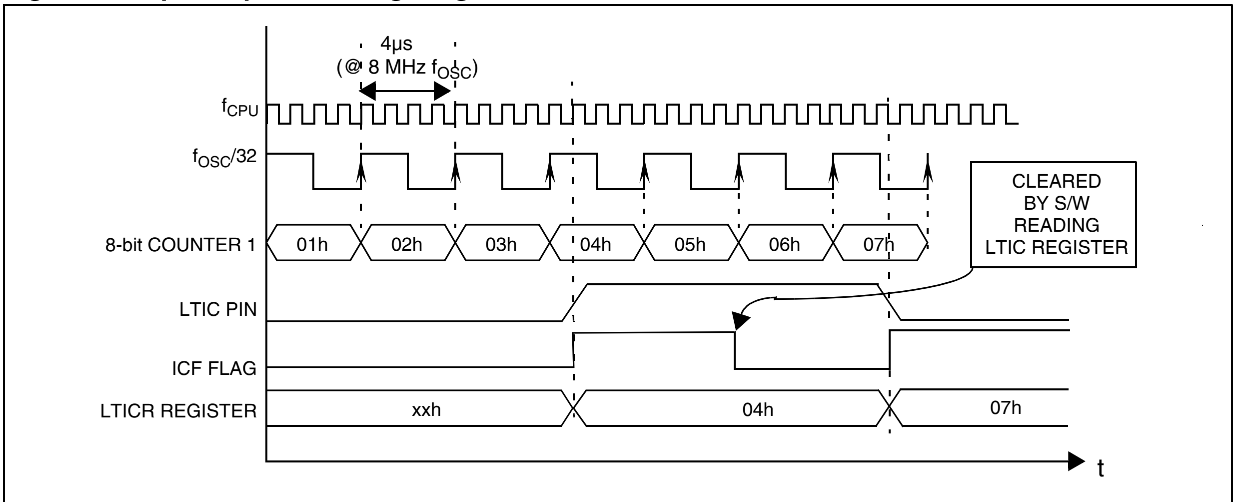
The LTICR is a read-only register and always contains the data from the last input capture. Input capture is inhibited if the ICF bit is set.

11.3.3.3 Timebase Counter 2

Counter 2 is an 8-bit autoreload upcounter. It can be read by accessing the LTCNTR register. After an MCU reset, it increments at a frequency of $f_{OSC}/32$ starting from the value stored in the LTARR register. A counter overflow event occurs when the counter rolls over from FFh to the LTARR reload value. Software can write a new value at any time in the LTARR register, this value will be automatically loaded in the counter when the next overflow occurs.

When Counter 2 overflows, the TB2F bit in the LTCSR2 register is set by hardware and an interrupt request is generated if the TB2IE bit is set. The TB2F bit is cleared by software reading the LTCSR2 register.

Figure 50. Input Capture Timing Diagram.



LITE TIMER (Cont'd)

11.3.4 Low Power Modes

Mode	Description
SLOW	No effect on Lite timer (this peripheral is driven directly by f_{OSC}/32)
WAIT	No effect on Lite timer
ACTIVE HALT	No effect on Lite timer
HALT	Lite timer stops counting

11.3.5 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Active Halt	Exit from Halt
Timebase 1 Event	TB1F	TB1IE	Yes	Yes	No
Timebase 2 Event	TB2F	TB2IE		No	
IC Event	ICF	ICIE		No	

Note: The TBxF and ICF interrupt events are connected to separate interrupt vectors (see Interrupts chapter).

They generate an interrupt if the enable bit is set in the LTCSR1 or LTCSR2 register and the interrupt mask in the CC register is reset (RIM instruction).

11.3.6 Register Description

LITE TIMER CONTROL/STATUS REGISTER 2 (LTCSR2)
Read / Write
Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	TB2IE	TB2F

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

Bit 1 = **TB2IE** *Timebase 2 Interrupt enable*
This bit is set and cleared by software.
0: Timebase (TB2) interrupt disabled
1: Timebase (TB2) interrupt enabled

Bit 0 = **TB2F** *Timebase 2 Interrupt Flag*
This bit is set by hardware and cleared by software reading the LTCSR register. Writing to this bit has no effect.
0: No Counter 2 overflow
1: A Counter 2 overflow has occurred

LITE TIMER AUTORELOAD REGISTER (LTARR)
Read / Write
Reset Value: 0000 0000 (00h)

7							0
AR7	AR7	AR7	AR7	AR3	AR2	AR1	AR0

Bits 7:0 = **AR[7:0]** *Counter 2 Reload Value*
These bits register is read/write by software. The LTARR value is automatically loaded into Counter 2 (LTCNTR) when an overflow occurs.

LITE TIMER COUNTER 2 (LTCNTR)
Read only
Reset Value: 0000 0000 (00h)

7							0
CNT7	CNT7	CNT7	CNT7	CNT3	CNT2	CNT1	CNT0

Bits 7:0 = **CNT[7:0]** *Counter 2 Reload Value*
This register is read by software. The LTARR value is automatically loaded into Counter 2 (LTCNTR) when an overflow occurs.

LITE TIMER CONTROL/STATUS REGISTER (LTCSR1)
Read / Write
Reset Value: 0x00 0000 (x0h)

7							0
ICIE	ICF	TB	TB1IE	TB1F	-	-	-

Bit 7 = **ICIE** *Interrupt Enable*
This bit is set and cleared by software.
0: Input Capture (IC) interrupt disabled
1: Input Capture (IC) interrupt enabled

LITE TIMER (Cont'd)

Bit 6 = **ICF** *Input Capture Flag*
This bit is set by hardware and cleared by software by reading the LTICR register. Writing to this bit does not change the bit value.
0: No input capture
1: An input capture has occurred

Note: After an MCU reset, software must initialize the ICF bit by reading the LTICR register

Bit 5 = **TB** *Timebase period selection*
This bit is set and cleared by software.
0: Timebase period = t_{OSC} * 8000 (1ms @ 8 MHz)
1: Timebase period = t_{OSC} * 16000 (2ms @ 8 MHz)

Bit 4 = **TB1IE** *Timebase Interrupt enable*
This bit is set and cleared by software.
0: Timebase (TB1) interrupt disabled
1: Timebase (TB1) interrupt enabled

Bit 3 = **TB1F** *Timebase Interrupt Flag*
This bit is set by hardware and cleared by software reading the LTCSR register. Writing to this bit has no effect.
0: No counter overflow
1: A counter overflow has occurred

Bits 2:0 = Reserved

LITE TIMER INPUT CAPTURE REGISTER (LTICR)
Read only
Reset Value: 0000 0000 (00h)

7							0
ICR7	ICR6	ICR5	ICR4	ICR3	ICR2	ICR1	ICR0

Bits 7:0 = **ICR[7:0]** *Input Capture Value*
These bits are read by software and cleared by hardware after a reset. If the ICF bit in the LTCSR is cleared, the value of the 8-bit up-counter will be captured when a rising or falling edge occurs on the LTIC pin.

Table 15. Lite Timer Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
08	**LTCSR2** Reset Value	0	0	0	0	0	0	TB2IE 0	TB2F 0
09	**LTARR** Reset Value	AR7 0	AR6 0	AR5 0	AR4 0	AR3 0	AR2 0	AR1 0	AR0 0
0A	**LTCNTR** Reset Value	CNT7 0	CNT6 0	CNT5 0	CNT4 0	CNT3 0	CNT2 0	CNT1 0	CNT0 0
0B	**LTCSR1** Reset Value	ICIE 0	ICF x	TB 0	TB1IE 0	TB1F 0	0	0	0
0C	**LTICR** Reset Value	ICR7 0	ICR6 0	ICR5 0	ICR4 0	ICR3 0	ICR2 0	ICR1 0	ICR0 0

ON-CHIP PERIPHERALS (cont'd)

11.4 SERIAL PERIPHERAL INTERFACE (SPI)

11.4.1 Introduction

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

11.4.2 Main Features

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

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

11.4.3 General Description

Figure 51 on page 77 shows the serial peripheral interface (SPI) block diagram. There are three registers:

– SPI Control Register (SPICR)

– SPI Control/Status Register (SPICSR)

– SPI Data Register (SPIDR)

The SPI is connected to external devices through four pins:

– MISO: Master In / Slave Out data

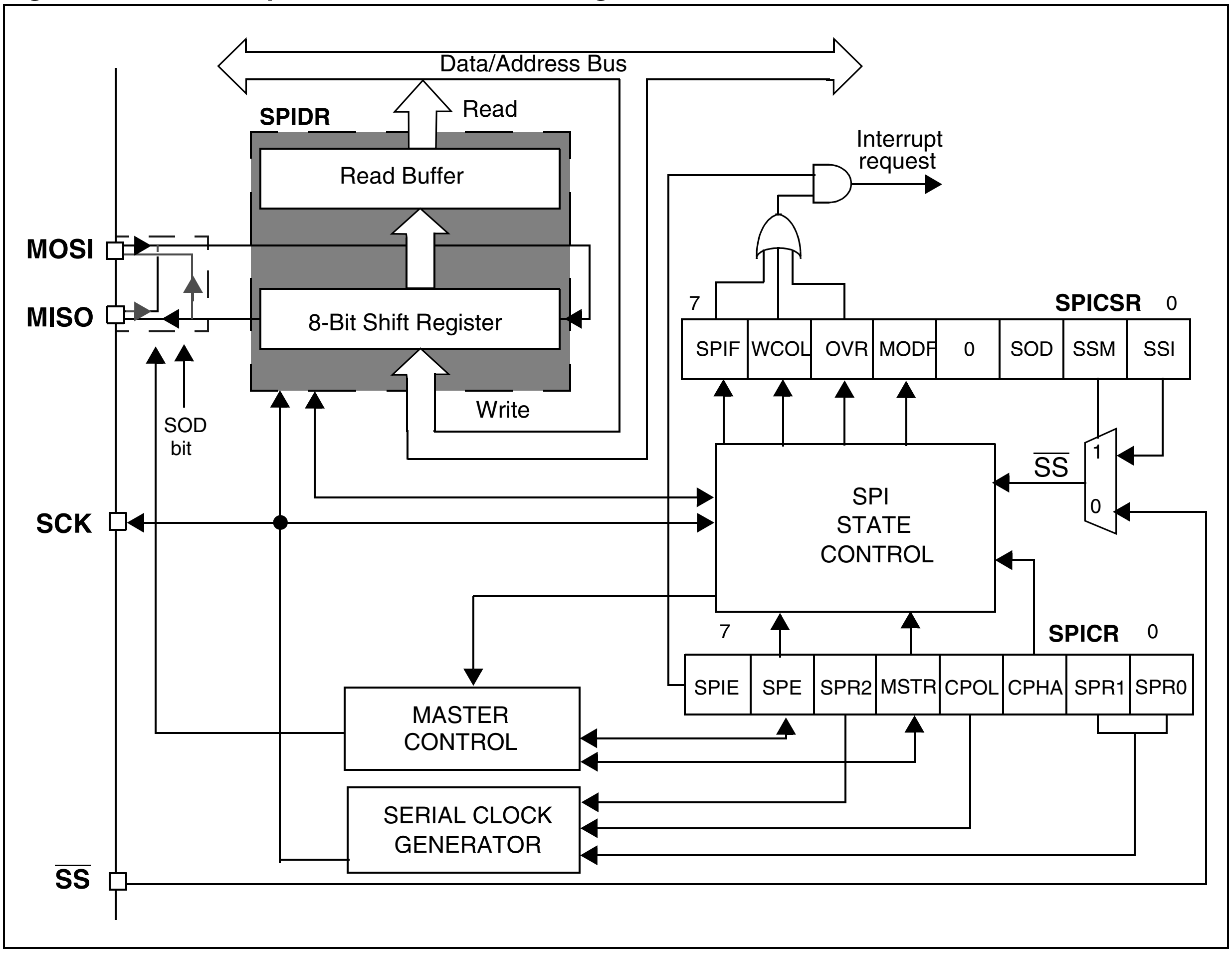
– MOSI: Master Out / Slave In data

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

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

SERIAL PERIPHERAL INTERFACE (SPI) (cont'd)

Figure 51. Serial Peripheral Interface Block Diagram



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.4.3.1 Functional Description

A basic example of interconnections between a single master and a single slave is illustrated in Figure 52.

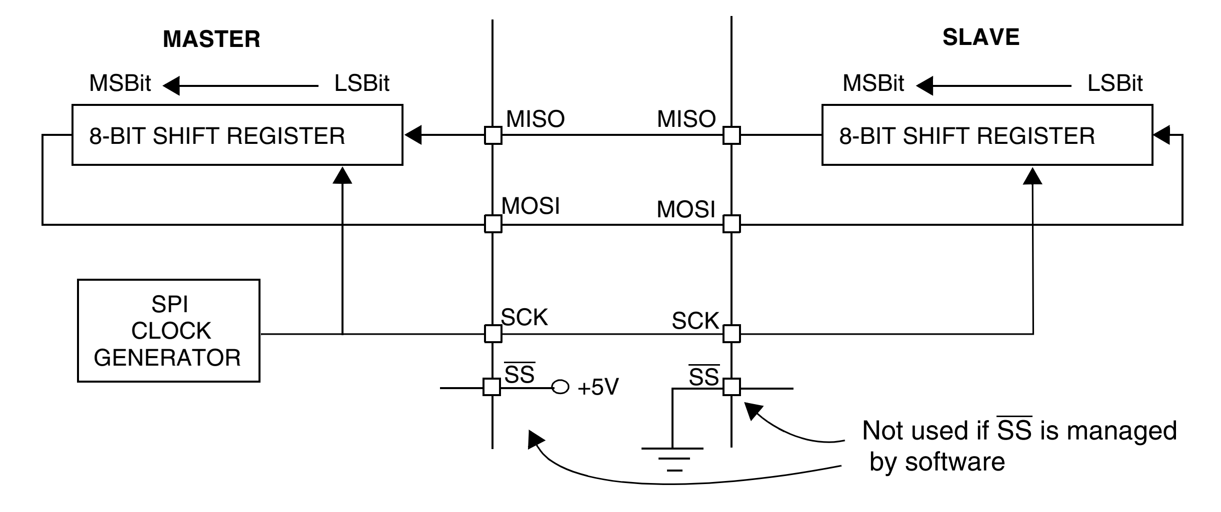
The MOSI pins are connected together and the MISO pins are connected together. In this way data is transferred serially between master and slave (most significant bit first).

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 55 on page 81) but master and slave must be programmed with the same timing mode.

Figure 52. Single Master/ Single Slave Application



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.4.3.2 Slave Select Management

As an alternative to using the $\overline{SS}$ pin to control the Slave Select signal, the application can choose to manage the Slave Select signal by software. This is configured by the SSM bit in the SPICSR register (see Figure 54).

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 53):

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

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

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

– $\overline{SS}$ internal must be held low during byte transmission and pulled high between each byte to allow the slave to write to the shift register. If $\overline{SS}$ is not pulled high, a Write Collision error will occur when the slave writes to the shift register (see Section 11.4.5.3).

Figure 53. Generic $\overline{SS}$ Timing Diagram

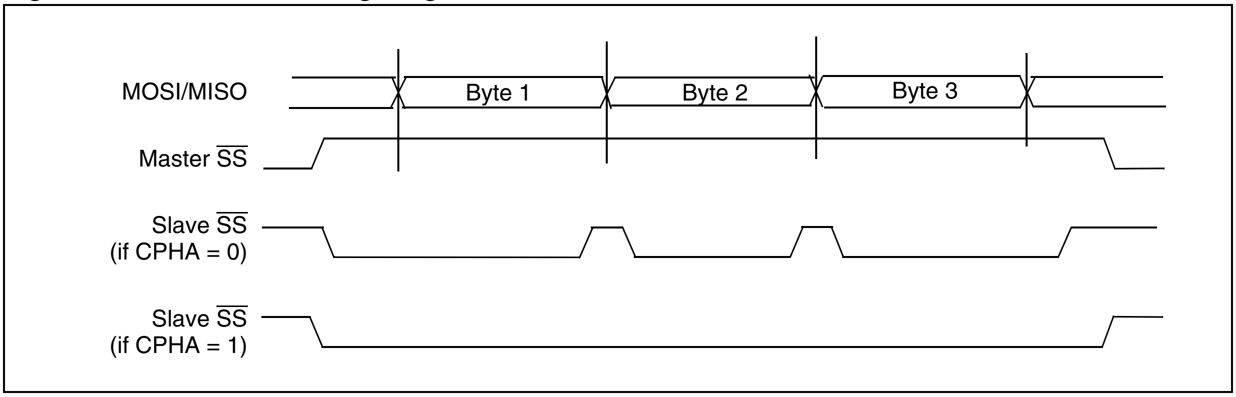
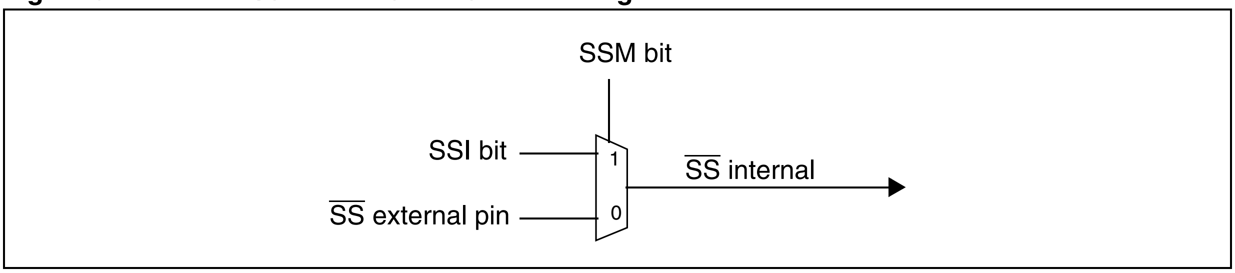


Figure 54. Hardware/Software Slave Select Management



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.4.3.3 Master Mode Operation

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

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

How to operate the SPI in master mode

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

1. Write to the SPICR register:
   – Select the clock frequency by configuring the SPR[2:0] bits.
   – Select the clock polarity and clock phase by configuring the CPOL and CPHA bits. Figure 55 shows the four possible configurations.
   Note: The slave must have the same CPOL and CPHA settings as the master.
2. Write to the SPICSR register:
   – Either set the SSM bit and set the SSI bit or clear the SSM bit and tie the SS pin high for the complete byte transmit sequence.
3. Write to the SPICR register:
   – Set the MSTR and SPE bits
   Note: MSTR and SPE bits remain set only if SS is high).

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

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

11.4.3.4 Master Mode Transmit Sequence

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

When data transfer is complete:

– The SPIF bit is set by hardware.

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

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

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

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

11.4.3.5 Slave Mode Operation

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

To operate the SPI in slave mode:

1. Write to the SPICSR register to perform the following actions:
   – Select the clock polarity and clock phase by configuring the CPOL and CPHA bits (see Figure 55).
   Note: The slave must have the same CPOL and CPHA settings as the master.
   – Manage the SS pin as described in Section 11.4.3.2 and Figure 53. If CPHA = 1 SS must be held low continuously. If CPHA = 0 SS must be held low during byte transmission and pulled up between each byte to let the slave write in the shift register.
2. Write to the SPICR register to clear the MSTR bit and set the SPE bit to enable the SPI I/O functions.

11.4.3.6 Slave Mode Transmit Sequence

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

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

When data transfer is complete:

– The SPIF bit is set by hardware.

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

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

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

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

The SPIF bit can be cleared during a second transmission; however, it must be cleared before the second SPIF bit in order to prevent an Overrun condition (see Section 11.4.5.2).

SERIAL PERIPHERAL INTERFACE (Cont'd)

11.4.4 Clock Phase and Clock Polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits (See Figure 55).

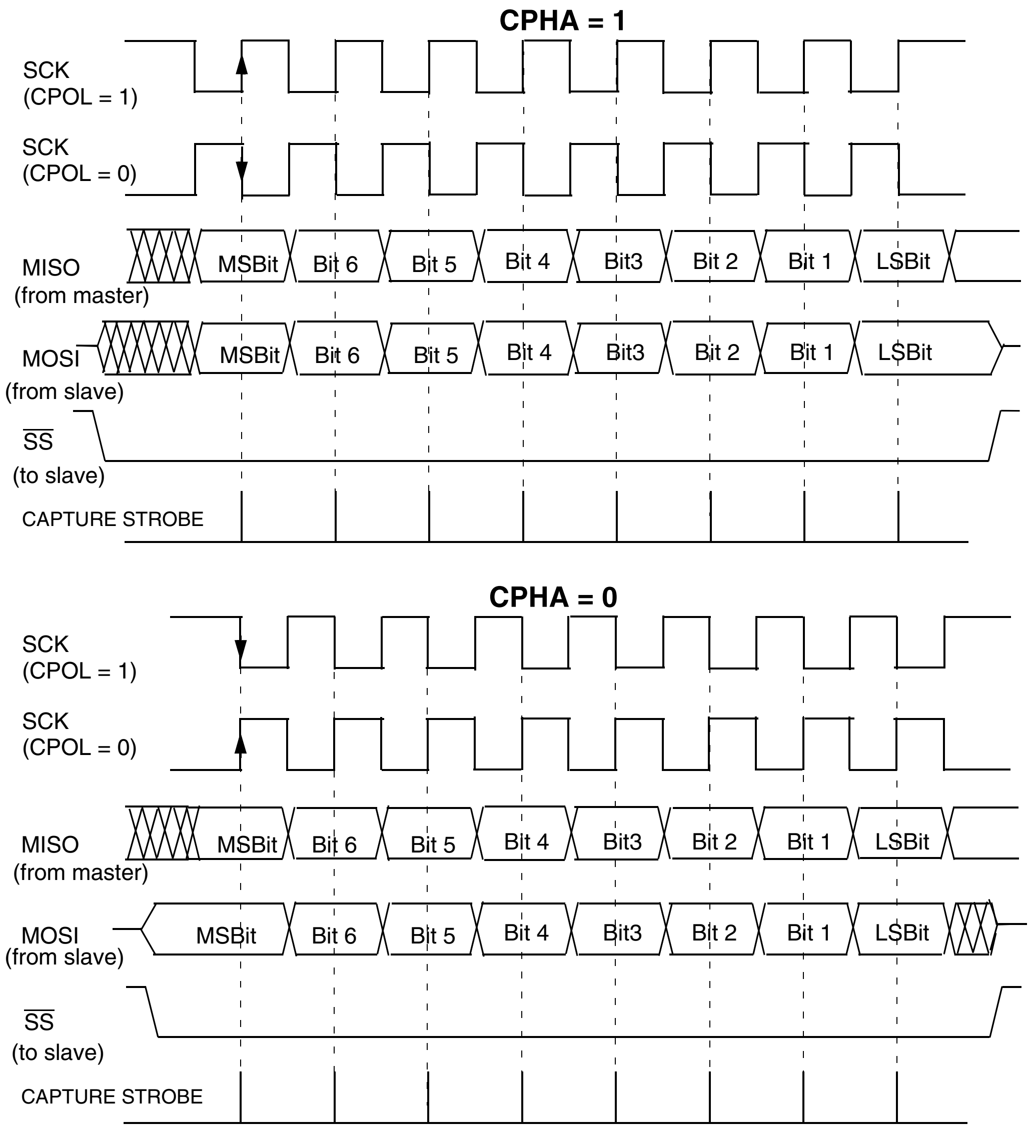
Note: The idle state of SCK must correspond to the polarity selected in the SPICSR register (by pulling up SCK if CPOL = 1 or pulling down SCK if CPOL = 0).

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

Figure 55 shows an SPI transfer with the four combinations of the CPHA and CPOL bits. The diagram may be interpreted as a master or slave timing diagram where the SCK pin, the MISO pin and the MOSI pin are directly connected between the master and the slave device.

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

Figure 55. Data Clock Timing Diagram



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

SERIAL PERIPHERAL INTERFACE (Cont'd)

11.4.5 Error Flags

11.4.5.1 Master Mode Fault (MODF)

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

When a Master mode fault occurs:

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

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

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

Clearing the MODF bit is done through a software sequence:

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

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

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

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

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

11.4.5.2 Overrun Condition (OVR)

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

When an Overrun occurs:

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

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

The OVR bit is cleared by reading the SPICSR register.

11.4.5.3 Write Collision Error (WCOL)

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

Write collisions can occur both in master and slave mode. See also Section 11.4.3.2 Slave Select Management.

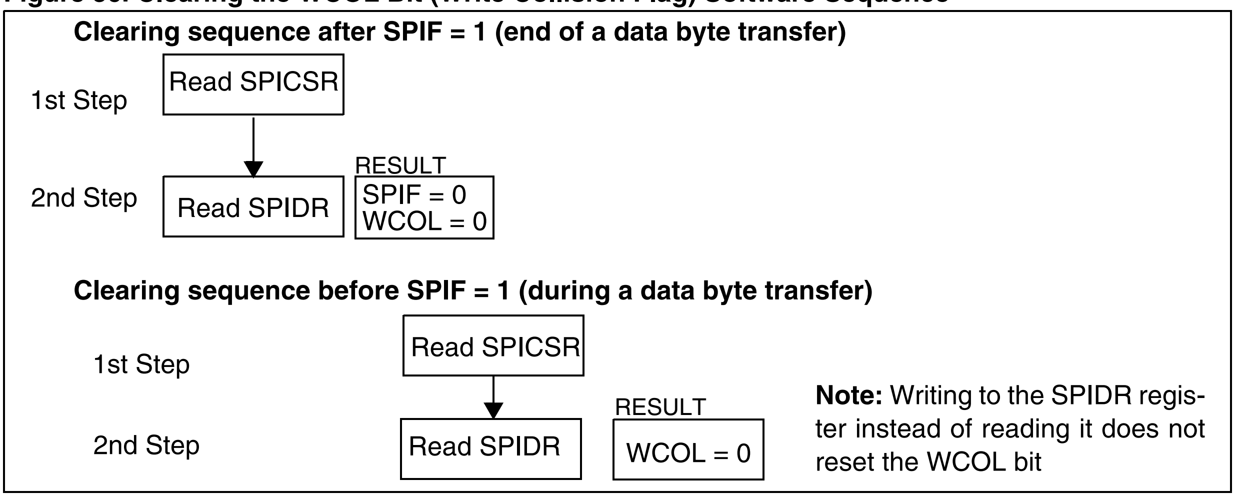
Note: A "read collision" will never occur since the received data byte is placed in a buffer in which access is always synchronous with the CPU operation.

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

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

Clearing the WCOL bit is done through a software sequence (see Figure 56).

Figure 56. Clearing the WCOL Bit (Write Collision Flag) Software Sequence



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.4.5.4 Single Master and Multimaster Configurations

There are two types of SPI systems:

– Single Master System

– Multimaster System

Single Master System

A typical single master system may be configured using a device as the master and four devices as slaves (see Figure 57).

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

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

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

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

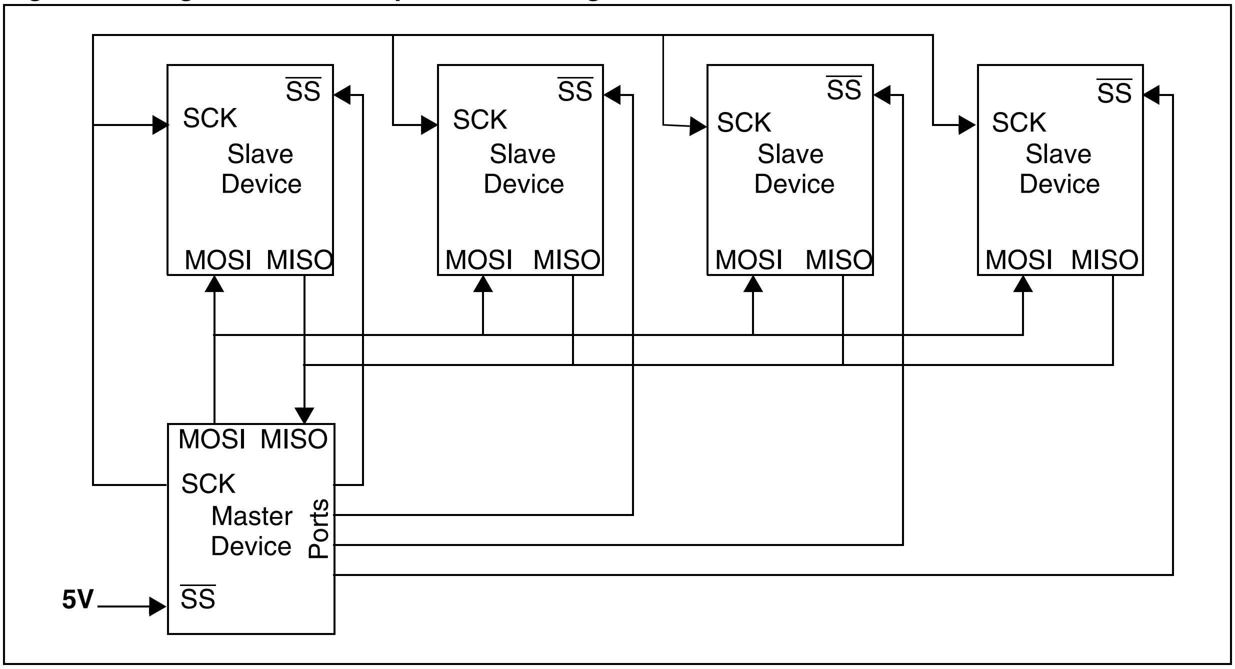
Other transmission security methods can use ports for handshake lines or data bytes with command fields.

Multimaster System

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

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

Figure 57. Single Master / Multiple Slave Configuration



SERIAL PERIPHERAL INTERFACE (Cont'd)

11.4.6 Low Power Modes

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

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

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

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

Caution: The SPI can wake up the device from HALT mode only if the Slave Select signal (external $\overline{SS}$ pin or the SSI bit in the SPICSR register) is low when the device enters HALT mode. So, if Slave selection is configured as external (see Section 11.4.3.2), make sure the master drives a low level on the $\overline{SS}$ pin when the slave enters HALT mode.

11.4.7 Interrupts

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

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

11.4.8 Register Description

CONTROL REGISTER (SPICR)

Read/Write

Reset Value: 0000 xxxx (0xh)

7							0
SPIE	SPE	SPR2	MSTR	CPOL	CPHA	SPR1	SPR0

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

Bit 6 = **SPE** *Serial Peripheral Output Enable.*
This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$ = 0 (see Section 11.4.5.1 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 16 SPI Master mode SCK Frequency.
0: Divider by 2 enabled
1: Divider by 2 disabled

Note: This bit has no effect in slave mode.

Bit 4 = **MSTR** *Master Mode.*
This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$ = 0 (see Section 11.4.5.1 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 16. SPI Master mode SCK Frequency

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

SERIAL PERIPHERAL INTERFACE (Cont'd)

CONTROL/STATUS REGISTER (SPICSR)

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

7							0
SPIF	WCOL	OVR	MODF	-	SOD	SSM	SSI

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

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

Bit 6 = **WCOL** *Write Collision status (Read only).*
This bit is set by hardware when a write to the SPIDR register is done during a transmit sequence. It is cleared by a software sequence (see Figure 56).
0: No write collision occurred
1: A write collision has been detected

Bit 5 = **OVR** *SPI 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 11.4.5.2). An interrupt is generated if SPIE = 1 in the SPICR register. The OVR bit is cleared by software reading the SPICSR register.
0: No overrun error
1: Overrun error detected

Bit 4 = **MODF** *Mode Fault flag (Read only).*
This bit is set by hardware when the $\overline{SS}$ pin is pulled low in master mode (see Section 11.4.5.1 Master Mode Fault (MODF)). An SPI interrupt can be generated if SPIE = 1 in the SPICR register. This bit is cleared by a software sequence (An access to the SPICSR register while MODF = 1 followed by a write to the SPICR register).
0: No master mode fault detected
1: A fault in master mode has been detected

Bit 3 = Reserved, must be kept cleared.

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

Bit 1 = **SSM** *$\overline{SS}$ Management.*
This bit is set and cleared by software. When set, it disables the alternate function of the SPI $\overline{SS}$ pin and uses the SSI bit value instead. See Section 11.4.3.2 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 51).

SERIAL PERIPHERAL INTERFACE (Cont'd)

Table 17. SPI Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0031h	**SPIDR** Reset Value	MSB x	 x	 x	 x	 x	 x	 x	LSB x
0032h	**SPICR** Reset Value	SPIE 0	SPE 0	SPR2 0	MSTR 0	CPOL x	CPHA x	SPR1 x	SPR0 x
0033h	**SPICSR** Reset Value	SPIF 0	WCOL 0	OVR 0	MODF 0	 0	SOD 0	SSM 0	SSI 0

ON-CHIP PERIPHERALS (Cont'd)

11.5 10-BIT A/D CONVERTER (ADC)

11.5.1 Introduction

The on-chip Analog to Digital Converter (ADC) peripheral is a 10-bit, successive approximation converter with internal sample and hold circuitry. This peripheral has up to seven multiplexed analog input channels (refer to device pinout description) that allow the peripheral to convert the analog voltage levels from up to seven different sources.

The result of the conversion is stored in a 10-bit Data Register. The A/D converter is controlled through a Control/Status Register.

11.5.2 Main Features

- 10-bit conversion
- Up to 7 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 58.

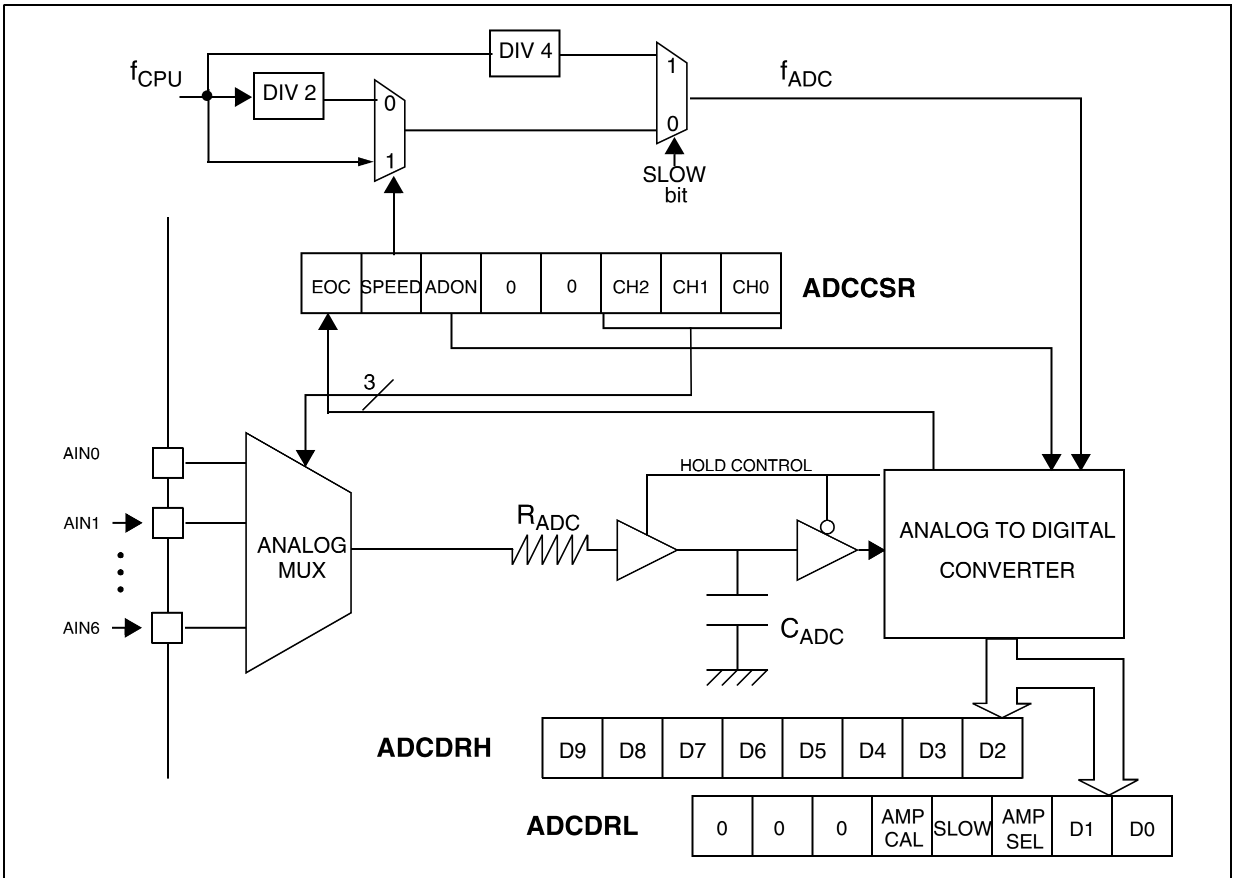
11.5.3 Functional Description

11.5.3.1 Analog Power Supply

V_{DDA} and V_{SSA} are the high and low level reference voltage pins. In some devices (refer to device pinout 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.

Figure 58. ADC Block Diagram



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

11.5.3.2 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 V_{DDA} (high-level voltage reference) then the conversion result is FFh in the ADCDRH register and 03h in the ADCDRL register (without overflow indication).

If the input voltage (V_{AIN}) is lower than V_{SSA} (low-level voltage reference) then the conversion result in the ADCDRH and ADCDRL registers is 00 00h.

The A/D converter is linear and the digital result of the conversion is stored in the ADCDRH and ADCDRL registers. The accuracy of the conversion is described in the Electrical Characteristics Section.

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

11.5.3.3 A/D Conversion

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

In the ADCCSR register:

– Select the CH[2:0] bits to assign the analog channel to convert.

ADC Conversion mode

In the ADCCSR register:

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

When a conversion is complete:

– The EOC bit is set by hardware.

– The result is in the ADCDR registers.

A read to the ADCDRH resets the EOC bit.

To read the 10 bits, perform the following steps:

1. Poll the EOC bit
2. Read ADCDRL
3. Read ADCDRH. This clears EOC automatically.

To read only 8 bits, perform the following steps:

1. Poll EOC bit
2. Read ADCDRH. This clears EOC automatically.

11.5.4 Low-Power Modes

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

Mode	Description
WAIT	No effect on A/D converter
HALT	A/D Converter disabled. After wake-up from HALT mode, the A/D converter requires a stabilization time t_{STAB} (see Electrical Characteristics) before accurate conversions can be performed.

11.5.5 Interrupts

None.

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

11.5.6 Register Description

CONTROL/STATUS REGISTER (ADCCSR)

Read/Write (Except bit 7 read only)

Reset Value: 0000 0000 (00h)

7							0
EOC	SPEED	ADON	0	0	CH2	CH1	CH0

Bit 7 = **EOC** *End of Conversion*
This bit is set by hardware. It is cleared by software reading the ADCDRH register.
0: Conversion is not complete
1: Conversion complete

Bit 6 = **SPEED** *ADC clock selection*
This bit is set and cleared by software. It is used together with the SLOW bit to configure the ADC clock speed. Refer to the table in the SLOW bit description (ADCDRL register).

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

Bits 4:3 = **Reserved.** Must be kept cleared.

Bits 2:0 = **CH[2:0]** *Channel Selection*
These bits are set and cleared by software. They select the analog input to convert.

Channel Pin*	CH2	CH1	CH0
AIN0	0	0	0
AIN1	0	0	1
AIN2	0	1	0
AIN3	0	1	1
AIN4	1	0	0
AIN5	1	0	1
AIN6	1	1	0

*The number of channels is device dependent. Refer to the device pinout description.

DATA REGISTER HIGH (ADCDRH)

Read Only

Reset Value: xxxx xxxx (xxh)

7							0
D9	D8	D7	D6	D5	D4	D3	D2

Bits 7:0 = **D[9:2]** *MSB of Analog Converted Value*

AMP CONTROL/DATA REGISTER LOW (ADCDRL)

Read/Write

Reset Value: 0000 00xx (0xh)

7							0
0	0	0	-	SLOW	-	D1	D0

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

Bit 4 = **Reserved**. Must be kept cleared.

Bit 3 = **SLOW** *Slow mode*
This bit is set and cleared by software. It is used together with the SPEED bit in the ADCCSR register to configure the ADC clock speed as shown on the table below.

f_{ADC}	SLOW	SPEED
$f_{CPU}/2$	0	0
f_{CPU}	0	1
$f_{CPU}/4$	1	x

Note: Max f_{ADC} allowed = 4 MHz (see section 13.11 on page 122)

Bit 2 = **Reserved.** Must be kept cleared.

Bits 1:0 = **D[1:0]** *LSB of Analog Converted Value*

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

Table 18. ADC Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0034h	**ADCCSR** Reset Value	EOC 0	SPEED 0	ADON 0	0 0	0 0	CH2 0	CH1 0	CH0 0
0035h	**ADCDRH** Reset Value	D9 x	D8 x	D7 x	D6 x	D5 x	D4 x	D3 x	D2 x
0036h	**ADCDRL** Reset Value	0 0	0 0	0 0	AMPCAL 0	SLOW 0	AMPSEL 0	D1 x	D0 x

12 INSTRUCTION SET

12.1 ST7 ADDRESSING MODES

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

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

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

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

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

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

Table 19. ST7 Addressing Mode Overview

Mode			Syntax	Destination/ Source	Pointer Address (Hex.)	Pointer Size (Hex.)	Length (Bytes)
Inherent			nop				+ 0
Immediate			ld A,#$55				+ 1
Short	Direct		ld A,$10	00..FF			+ 1
Long	Direct		ld A,$1000	0000..FFFF			+ 2
No Offset	Direct	Indexed	ld A,(X)	00..FF			+ 0 (with X register) + 1 (with Y register)
Short	Direct	Indexed	ld A,($10,X)	00..1FE			+ 1
Long	Direct	Indexed	ld A,($1000,X)	0000..FFFF			+ 2
Short	Indirect		ld A,[$10]	00..FF	00..FF	byte	+ 2
Long	Indirect		ld A,[$10.w]	0000..FFFF	00..FF	word	+ 2
Short	Indirect	Indexed	ld A,([$10],X)	00..1FE	00..FF	byte	+ 2
Long	Indirect	Indexed	ld A,([$10.w],X)	0000..FFFF	00..FF	word	+ 2
Relative	Direct		jrne loop	PC-128/PC+127[1)]			+ 1
Relative	Indirect		jrne [$10]	PC-128/PC+127[1)]	00..FF	byte	+ 2
Bit	Direct		bset $10,#7	00..FF			+ 1
Bit	Indirect		bset [$10],#7	00..FF	00..FF	byte	+ 2
Bit	Direct	Relative	btjt $10,#7,skip	00..FF			+ 2
Bit	Indirect	Relative	btjt [$10],#7,skip	00..FF	00..FF	byte	+ 3

Note:
1. At the time the instruction is executed, the Program Counter (PC) points to the instruction following JRxx.

ST7 ADDRESSING MODES (Cont'd)

12.1.1 Inherent

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

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

12.1.2 Immediate

Immediate instructions have 2 bytes, the first byte contains the opcode, the second byte contains the operand value.

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

12.1.3 Direct

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

The direct addressing mode consists of two submodes:

Direct (short)

The address is a byte, thus requires only 1 byte after the opcode, but only allows 00 - FF addressing space.

Direct (long)

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

12.1.4 Indexed (No Offset, Short, Long)

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

The indirect addressing mode consists of three submodes:

Indexed (No Offset)

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

Indexed (Short)

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

Indexed (long)

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

12.1.5 Indirect (Short, Long)

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

The pointer address follows the opcode. The indirect addressing mode consists of two submodes:

Indirect (short)

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

Indirect (long)

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

ST7 ADDRESSING MODES (Cont'd)

12.1.6 Indirect Indexed (Short, Long)

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

The indirect indexed addressing mode consists of two submodes:

Indirect Indexed (Short)

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

Indirect Indexed (Long)

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

Table 20. Instructions Supporting Direct, Indexed, Indirect and Indirect Indexed Addressing Modes

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

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

12.1.7 Relative Mode (Direct, Indirect)

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

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

The relative addressing mode consists of two submodes:

Relative (Direct)

The offset follows the opcode.

Relative (Indirect)

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

12.2 INSTRUCTION GROUPS

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

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

Using a prebyte

The instructions are described with 1 to 4 bytes.

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

The whole instruction becomes:

PC-2 End of previous instruction

PC-1 Prebyte

PC Opcode

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

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

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

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

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

12.2.1 Illegal Opcode Reset

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

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

INSTRUCTION GROUPS (Cont'd)

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

INSTRUCTION GROUPS (Cont'd)

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

13 ELECTRICAL CHARACTERISTICS

13.1 PARAMETER CONDITIONS

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

13.1.1 Minimum and Maximum Values

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

Data based on characterization results, design simulation and/or technology characteristics is indicated in the table footnotes and is 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 is based on $T_A = 25°C$, $V_{DD} = 5V$ (for the $4.5V \le V_{DD} \le 5.5V$ voltage range) and $V_{DD} = 3.3V$ (for the $3V \le V_{DD} \le 3.6V$ voltage range). They are given only as design guidelines and are not tested.

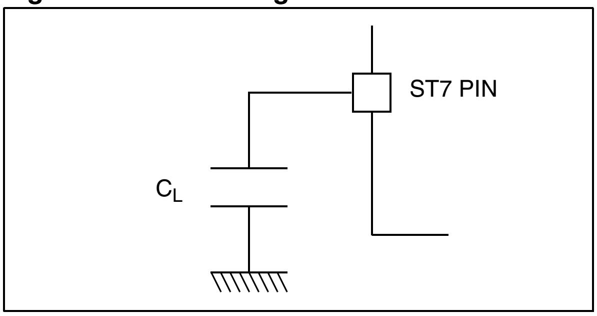
13.1.3 Typical Curves

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

13.1.4 Loading Capacitor

The loading conditions used for pin parameter measurement are shown in Figure 59.

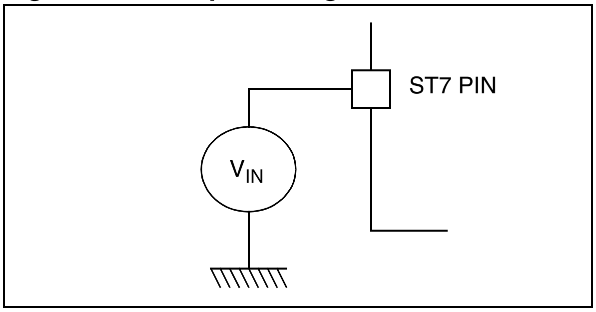
Figure 59. Pin Loading Conditions



13.1.5 Pin Input Voltage

The input voltage measurement on a pin of the device is described in Figure 60.

Figure 60. Pin Input Voltage



ELECTRICAL CHARACTERISTICS (Cont'd)

13.2 ABSOLUTE MAXIMUM RATINGS

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

13.2.1 Voltage Characteristics

Symbol	Ratings	Maximum value	Unit
V_{DD} - V_{SS}	Supply voltage	7.0	V
V_{IN}	Input voltage on any pin[1) 2)]	V_{SS} - 0.3 to V_{DD} + 0.3	
$V_{ESD(HBM)}$	Electrostatic discharge voltage (Human Body Model)	See section 13.7.3 on page 112	
$V_{ESD(MM)}$	Electrostatic discharge voltage (Machine Model)		

13.2.2 Current Characteristics

Symbol	Ratings	Maximum value	Unit
I_{VDD}	Total current into V_{DD} power lines (source)[3)]	75	mA
I_{VSS}	Total current out of V_{SS} ground lines (sink)[3)]	150	
I_{IO}	Output current sunk by any standard I/O and control pin	20	
	Output current sunk by any high sink I/O pin	40	
	Output current source by any I/Os and control pin	- 25	
$I_{INJ(PIN)}$[2)4)]	Injected current on ISPSEL pin	± 5	
	Injected current on $\overline{RESET}$ pin		
	Injected current on OSC1 and OSC2 pins		
	Injected current on PB0 pin[5)]	+ 5	
	Injected current on any other pin[6)]	± 5	
$\Sigma I_{INJ(PIN)}$[2)]	Total injected current (sum of all I/O and control pins)[6)]	± 20	

13.2.3 Thermal Characteristics

Symbol	Ratings	Value	Unit
T_{STG}	Storage temperature range	-65 to +150	°C
T_J	Maximum junction temperature (see Table 23, "THERMAL CHARACTERISTICS," on page 124)		

Notes:

1. Directly connecting the $\overline{RESET}$ and I/O pins to V_{DD} or V_{SS} could damage the device if an unintentional internal reset is generated or an unexpected change of the I/O configuration occurs (for example, due to a corrupted program counter). To guarantee safe operation, this connection must be made through a pull-up or pull-down resistor (typical: 4.7kΩ for $\overline{RESET}$, 10kΩ for I/Os). Unused I/O pins must be tied in the same way to V_{DD} or V_{SS} according to their reset configuration.
2. $I_{INJ(PIN)}$ must never be exceeded. This is implicitly insured if V_{IN} maximum is respected. If V_{IN} maximum cannot be respected, the injection current must be limited externally to the $I_{INJ(PIN)}$ value. A positive injection is induced by $V_{IN} > V_{DD}$ while a negative injection is induced by $V_{IN} < V_{SS}$.
3. All power (V_{DD}) and ground (V_{SS}) lines must always be connected to the external supply.
4. Negative injection disturbs the analog performance of the device. In particular, it induces leakage currents throughout the device including the analog inputs. To avoid undesirable effects on the analog functions, care must be taken:
- Analog input pins must have a negative injection less than 0.8 mA (assuming that the impedance of the analog voltage is lower than the specified limits)
- Pure digital pins must have a negative injection less than 1.6mA. In addition, it is recommended to inject the current as far as possible from the analog input pins.
5. No negative current injection allowed on PB0 pin.
6. When several inputs are submitted to a current injection, the maximum $\Sigma I_{INJ(PIN)}$ is the absolute sum of the positive and negative injected currents (instantaneous values). These results are based on characterization with $\Sigma I_{INJ(PIN)}$ maximum current injection on four I/O port pins of the device.

ELECTRICAL CHARACTERISTICS (Cont'd)

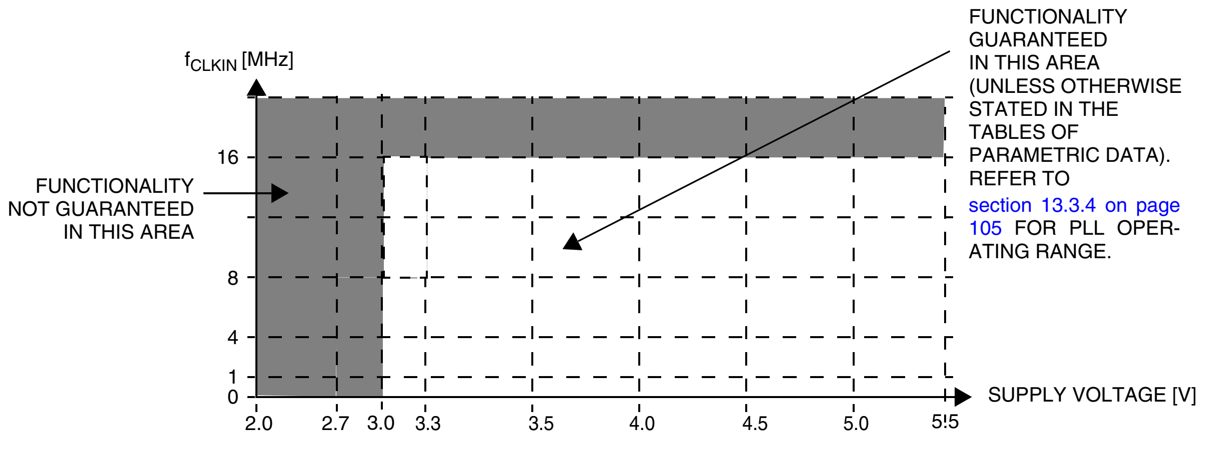
13.3 OPERATING CONDITIONS

13.3.1 General Operating Conditions

T_A = -40 to +125°C, unless otherwise specified.

Symbol	Parameter	Conditions	Min	Max	Unit
V_{DD}	Supply voltage	f_{OSC} = 16 MHz max T_A = -40°C to T_A max	3.0	5.5	V
f_{CLKIN}	External clock frequency on CLKIN pin	$V_{DD} \geq 3V$	0	16	MHz
T_A	Ambient temperature range	A Suffix version	-40	+85	°C
		C Suffix version		+125	

Figure 61. f_{CLKIN} Maximum Operating Frequency vs V_{DD} Supply Voltage



Note: For further information on clock management block diagram for f_{CLKIN} description, refer to Figure 12 in section 7 on page 21.

OPERATING CONDITIONS (Cont'd)

The RC oscillator and PLL characteristics are temperature-dependent.

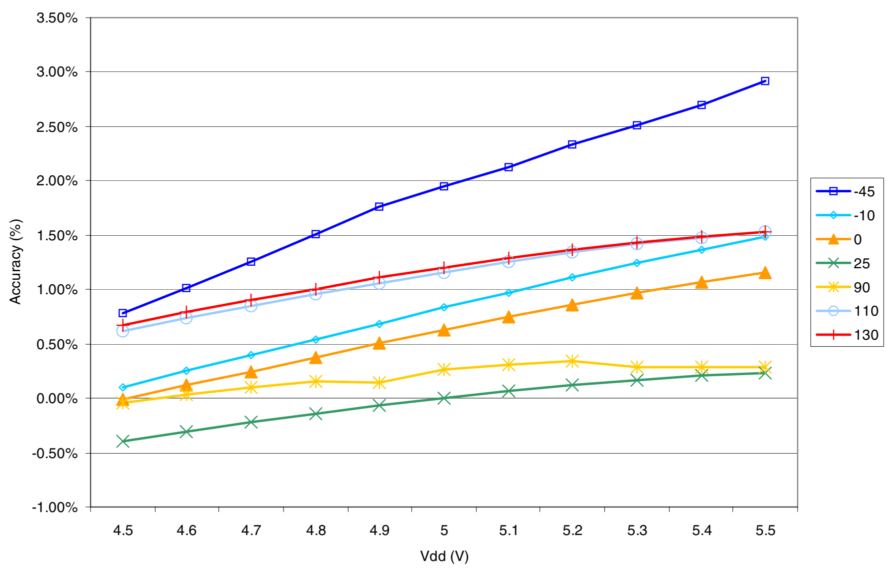
13.3.1.1 Operating Conditions (tested for T_A = -40 to +125°C) @ V_{DD} = 4.5 to 5.5V

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{RC}[1]	Internal RC oscillator frequency	RCCR = FF (reset value), T_A = 25°C, V_{DD} = 5V		700		kHz
		RCCR = RCCR0[2], T_A = 25°C, V_{DD} = 5V	992	1000	1008	
ACC_{RC}	Accuracy of Internal RC oscillator with RCCR = RCCR0[2][3]	T_A = 25°C, V_{DD} = 5V	-0.8		+0.8	%
		T_A = 25°C, V_{DD} = 4.5 to 5.5V[4]	-1		+1	
		T_A = -40 to +125°C, V_{DD} = 4.5 to 5.5V[4]	-3.5		+7	
$I_{DD(RC)}$	RC oscillator current consumption	T_A = 25°C, V_{DD} = 5V		600[4][5]		µA
$t_{su(RC)}$	RC oscillator setup time				10[2]	µs
f_{PLL}	x8 PLL input clock			1		MHz
t_{LOCK}	PLL Lock time[8]			2		ms
t_{STAB}	PLL Stabilization time[8]			4		
ACC_{PLL}	x8 PLL Accuracy	f_{RC} = 1 MHz @ T_A = 25°C, V_{DD} = 4.5 to 5.5V[4]		0.1[7]		%
		f_{RC} = 1 MHz @ T_A = -40 to +125°C, V_{DD} = 5V				
JIT_{PLL}	PLL jitter ($\Delta f_{CPU}/f_{CPU}$)			1[6]		
$I_{DD(PLL)}$	PLL current consumption	T_A = 25°C		600[4]		µA

Notes:

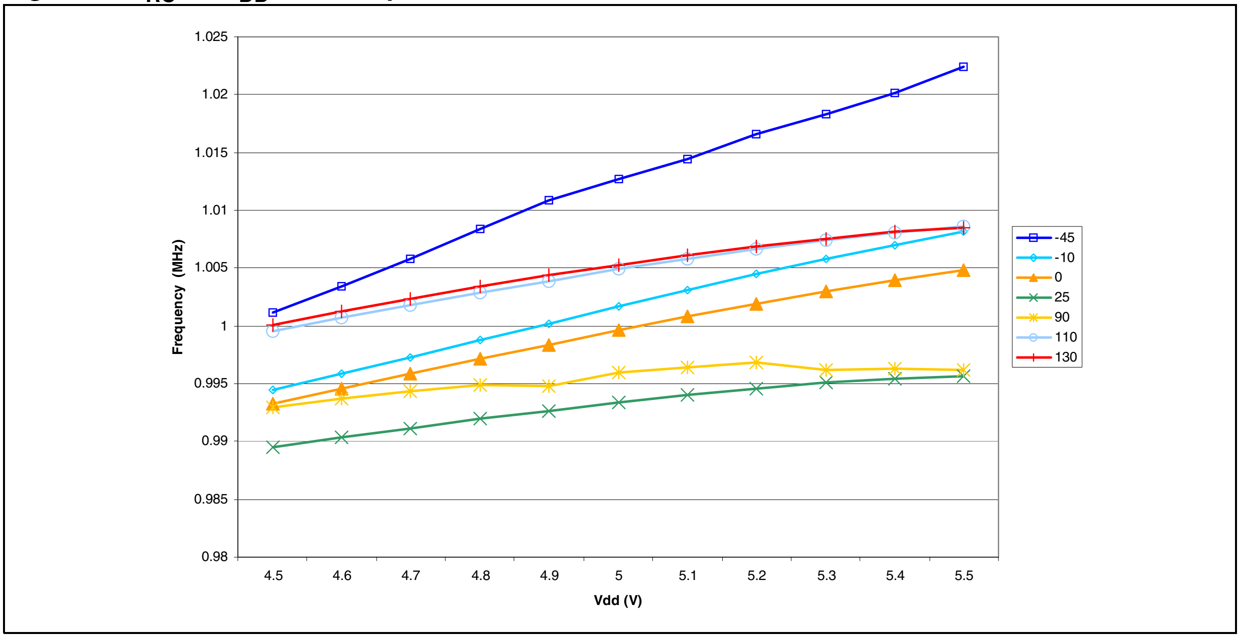
1. If the RC oscillator clock is selected, to improve clock stability and frequency accuracy, it is recommended to place a decoupling capacitor, typically 100nF, between the V_{DD} and V_{SS} pins as close as possible to the ST7 device.
2. See "INTERNAL RC OSCILLATOR ADJUSTMENT" on page 21.
3. Minimum value is obtained for hot temperature and max value is obtained for cold temperature.
4. Data based on characterization results, not tested in production
5. Measurement made with RC calibrated at 1 MHz.
6. Guaranteed by design.
7. Averaged over a 4ms period. After the LOCKED bit is set, a period of t_{STAB} is required to reach ACC_{PLL} accuracy.
8. After the LOCKED bit is set ACC_{PLL} is maximum 10% until t_{STAB} has elapsed. See Figure 11 on page 22.

Figure 62. Typical Accuracy with RCCR = RCCR0 vs V_{DD} = 4.5 to 5.5V and Temperature



OPERATING CONDITIONS (Cont'd)

Figure 63. f_{RC} vs V_{DD} and Temperature for Calibrated RCCR0



13.3.1.2 Operating Conditions (tested for T_A = -40 to +125°C) @ V_{DD} = 3.0 to 3.6V[1]

Symbol	Parameter[1]	Conditions	Min	Typ	Max	Unit
f_{RC}[2]	Internal RC oscillator frequency	RCCR = FF (reset value), T_A = 25°C, V_{DD} = 3.3V		700		kHz
		RCCR = RCCR1[3], T_A = 25°C, V_{DD} = 3.3V	992	1000	1008	
ACC_{RC}	Accuracy of Internal RC oscillator when calibrated with RCCR = RCCR1[3][4]	T_A = 25°C, V_{DD} = 3.3V	-0.8		+0.8	%
		T_A = 25°C, V_{DD} = 3 to 3.6V	-1		+1	
		T_A= -40 to +125°C, V_{DD} = 3 to 3.6V	-5		+6.5	
$I_{DD(RC)}$	RC oscillator current consumption	T_A = 25°C, V_{DD} = 3.3V		400[5]		µA
$t_{su(RC)}$	RC oscillator setup time	T_A = 25°C, V_{DD} = 3.3V			10[3]	µs
f_{PLL}	x4 PLL input clock			0.7		MHz
t_{LOCK}	PLL lock time[8]			2		ms
t_{STAB}	PLL stabilization time[8]			4		
ACC_{PLL}	x4 PLL accuracy	f_{RC} = 1 MHz @ T_A = -40 to +125°C, V_{DD} = 3.3V		0.1[7]		%
		f_{RC} = 1 MHz @ T_A = 25°C, V_{DD} = 3 to 3.6V		0.1[7]		
JIT_{PLL}	PLL jitter ($\Delta f_{CPU}/f_{CPU}$)			1[6]		
$I_{DD(PLL)}$	PLL current consumption	T_A = 25°C		190		µA

Notes:
1. Data based on characterization results, not tested in production.
2. If the RC oscillator clock is selected, to improve clock stability and frequency accuracy, it is recommended to place a decoupling capacitor, typically 100nF, between the VDD and VSS pins as close as possible to the ST7 device.
3. See "INTERNAL RC OSCILLATOR ADJUSTMENT" on page 21.
4. Minimum value is obtained for hot temperature and maximum value is obtained for cold temperature.
5. Measurement made with RC calibrated at 1 MHz.
6. Guaranteed by design.
7. Averaged over a 4ms period. After the LOCKED bit is set, a period of tSTAB is required to reach ACCPLL accuracy.
8. After the LOCKED bit is set, ACCPLL is maximum 10% until tSTAB has elapsed. See Figure 11 on page 22.

OPERATING CONDITIONS (Cont'd)

Figure 64. Typical Accuracy with RCCR = RCCR1 vs V_{DD} = 3 to 3.6V and Temperature

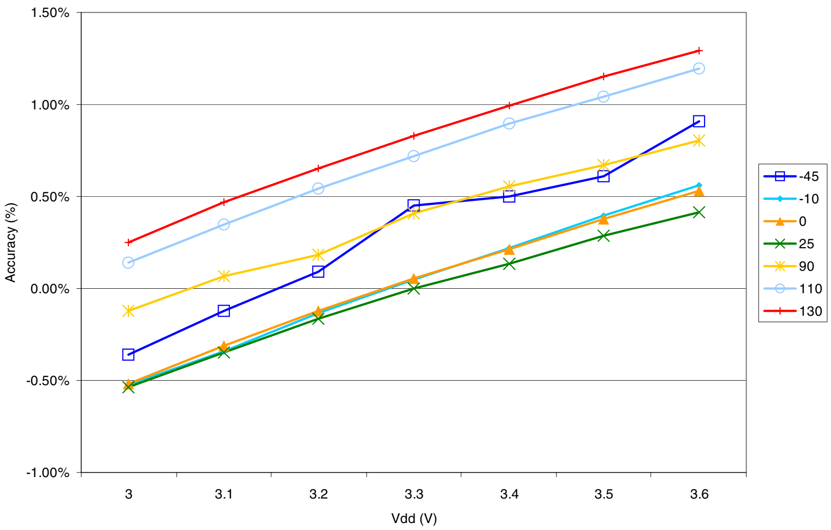
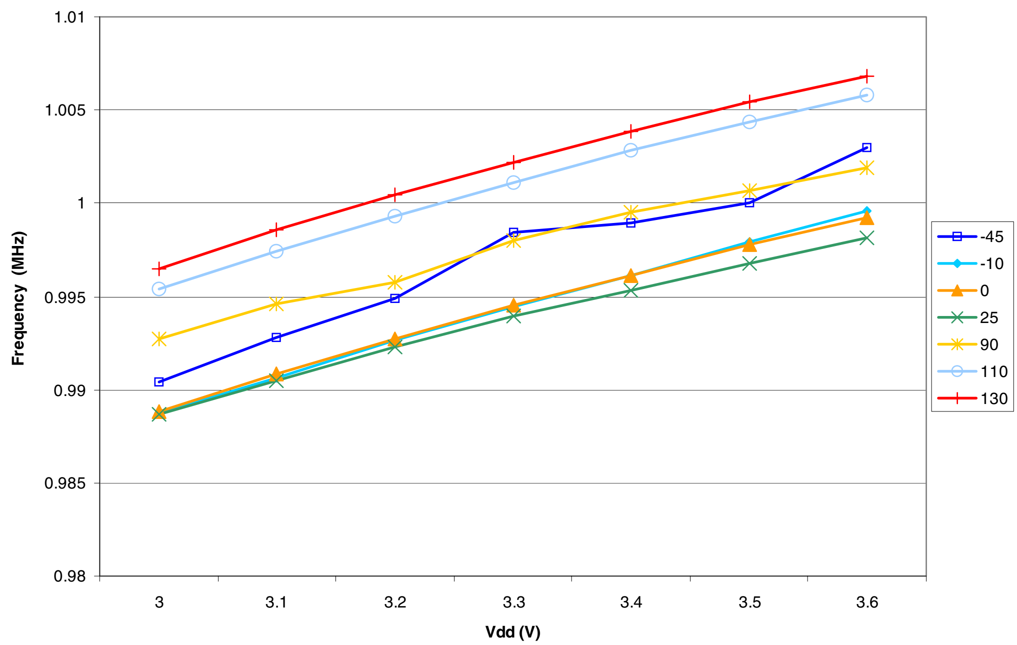
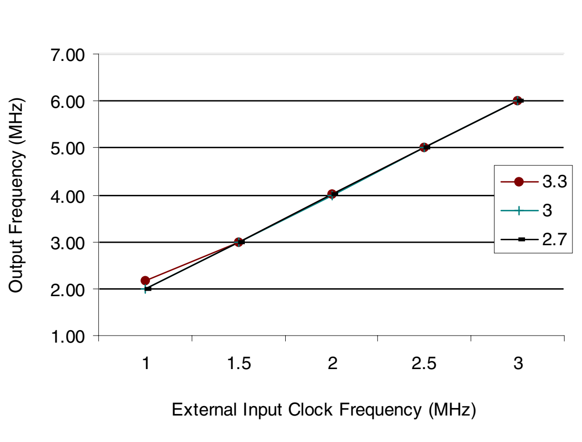


Figure 65. f_{RC} vs V_{DD} and Temperature for Calibrated RCCR1



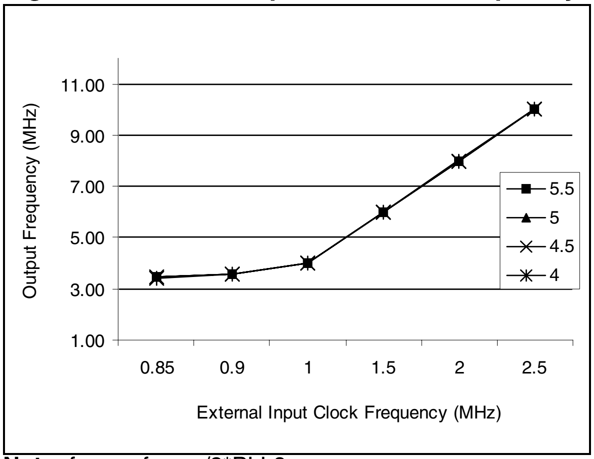
OPERATING CONDITIONS (Cont'd)

Figure 66. PLLx4 Output vs CLKIN Frequency



Note: $f_{OSC} = f_{CLKIN}/2*PLL4$

Figure 67. PLLx8 Output vs CLKIN Frequency



Note: $f_{OSC} = f_{CLKIN}/2*PLL8$

OPERATING CONDITONS (Cont'd)

13.3.2 Operating Conditions with Low Voltage Detector (LVD)

T_A = -40 to +125°C, unless otherwise specified.

Symbol	Parameter	Conditions[1]	Min	Typ	Max	Unit
$V_{IT+(LVD)}$	Reset release threshold (V_{DD} rise)		3.80[2]	4.20	4.60	V
$V_{IT-(LVD)}$	Reset generation threshold (V_{DD} fall)		3.70	4.00	4.35[2]	V
V_{hys}	LVD voltage threshold hysteresis	$V_{IT+(LVD)}-V_{IT-(LVD)}$		200		mV
Vt_{POR}	V_{DD} rise time rate [3][5]		0.02[2]		100[2]	ms/V
$t_{g(VDD)}$	Filtered glitch delay on V_{DD}	Not detected by the LVD			150[4]	ns
$I_{DD(LVD)}$	LVD/AVD current consumption			200		µA

Notes:

1. LVD functionality guaranteed only within the V_{DD} operating range specified in section 13.3.1 on page 100.
2. Not tested in production.
3. Not tested in production. The V_{DD} rise time rate condition is needed to insure a correct device power-on and LVD reset. When the V_{DD} slope is outside these values, the LVD may not ensure a proper reset of the MCU.
4. Based on design simulation.
5. Use of LVD with capacitive power supply: With this type of power supply, if power cuts occur in the application, it is recommended to pull V_{DD} down to 0V to ensure optimum restart conditions. Refer to circuit example in Figure 96 on page 119 and note 4.

13.3.3 Auxiliary Voltage Detector (AVD) Thresholds

T_A = -40 to +125°C, unless otherwise specified.

Symbol	Parameter	Conditions[1]	Min	Typ	Max	Unit
$V_{IT+(AVD)}$	1 ==> 0 AVDF flag toggle threshold (V_{DD} rise)		TBD[2]	4.40		V
$V_{IT-(AVD)}$	0 ==> 1 AVDF flag toggle threshold (V_{DD} fall)			4.15	TBD[2]	V
V_{hys}	AVD voltage threshold hysteresis	$V_{IT+(AVD)}-V_{IT-(AVD)}$		170		mV
ΔV_{IT-}	Voltage drop between AVD flag set and LVD reset activation	V_{DD} fall		0.15		V

Notes:

1. LVD functionality guaranteed only within the V_{DD} operating range specified in section 13.3.1 on page 100.
2. Data under characterization, not tested in production.

13.3.4 Internal RC Oscillator and PLL

The ST7 internal clock can be supplied by an internal RC oscillator and PLL (selectable by option byte).

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{DD(RC)}$	Internal RC oscillator operating voltage	Refer to operating range of V_{DD} with T_A, section 13.3.1 on page 100	3.0		5.5	V
$V_{DD(x4PLL)}$	x4 PLL operating voltage[1]		3.0		3.6	V
$V_{DD(x8PLL)}$	x8 PLL operating voltage		3.6		5.5	V
$t_{STARTUP}$	PLL Start-up time			60		PLL input clock (f_{PLL}) cycles

Notes:

1. x4 PLL option only applicable on Flash devices.

ELECTRICAL CHARACTERISTICS (Cont'd)

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 obtain the total device consumption, the two current values must be added (except for HALT mode for which the clock is stopped).

13.4.1 Supply Current

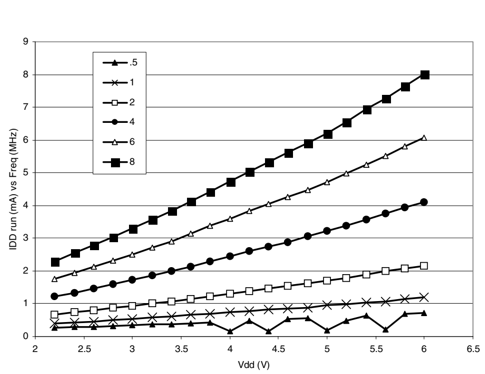
T_A = -40 to +125°C, unless otherwise specified.

Symbol	Parameter	Conditions		Typ	Max	Unit
I_{DD}	Supply current in RUN mode	V_{DD} = 5.5V	f_{CPU} = 8 MHz[1)]	7	9	mA
	Supply current in WAIT mode		f_{CPU} = 8 MHz[2)]	3	3.6	
	Supply current in SLOW mode		f_{CPU} = 250 kHz[3)]	0.7	0.9	
	Supply current in SLOW WAIT mode		f_{CPU} = 250 kHz[4)]	0.5	0.8	
	Supply current in HALT mode[5)]		$-40°C \leq T_A \leq +125°C$	1	6	µA
	Supply current in AWUFH mode[6)7)]		$-40°C \leq T_A \leq +125°C$	20		
	Supply current in ACTIVE HALT mode		$-40°C \leq T_A \leq +125°C$			mA

Notes:

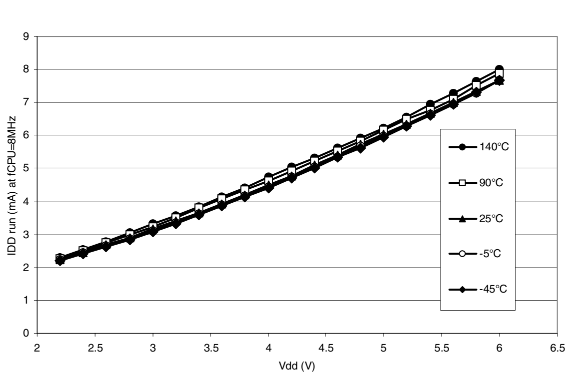
1. CPU running with memory access, all I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (CLKIN) driven by external square wave, LVD disabled.
2. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (CLKIN) driven by external square wave, LVD disabled.
3. SLOW mode selected with f_{CPU} based on f_{OSC} divided by 32. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (CLKIN) driven by external square wave, LVD disabled.
4. SLOW-WAIT mode selected with f_{CPU} based on f_{OSC} divided by 32. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load), all peripherals in reset state; clock input (CLKIN) driven by external square wave, LVD disabled.
5. All I/O pins in output mode with a static value at V_{SS} (no load), LVD disabled. Data based on characterization results, tested in production at V_{DD} max and f_{CPU} max.
6. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load). Data tested in production at V_{DD} max. and f_{CPU} max.
7. This consumption refers to the Halt period only and not the associated run period which is software dependent.

Figure 68. Typical I_{DD} in RUN vs f_{CPU}



Note: Graph displays data beyond the normal operating range of 3V - 5.5V

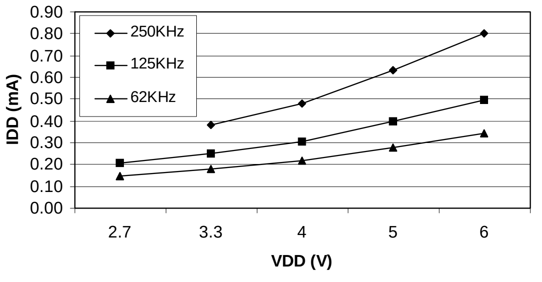
Figure 69. Typical I_{DD} in RUN at f_{CPU} = 8 MHz



Note: Graph displays data beyond the normal operating range of 3V - 5.5V

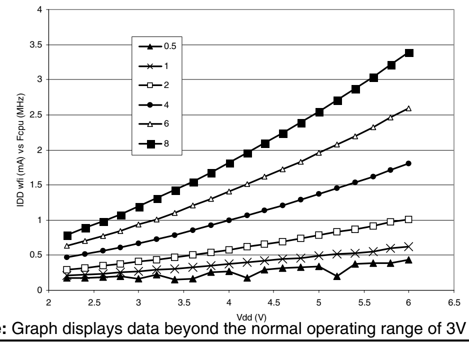
SUPPLY CURRENT CHARACTERISTICS (Cont'd)

Figure 70. Typical I_{DD} in SLOW vs f_{CPU}



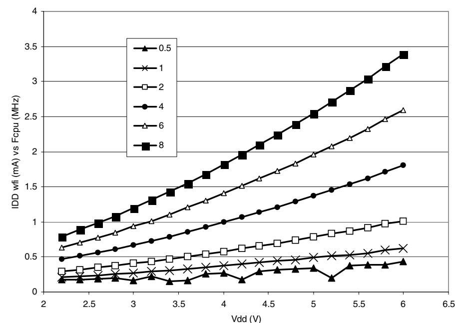
Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 71. Typical I_{DD} in WAIT vs f_{CPU}



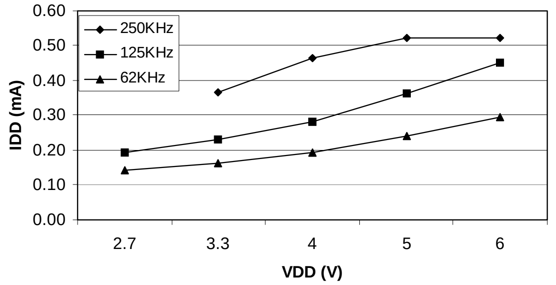
Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 72. Typical I_{DD} in WAIT at f_{CPU} = 8 MHz



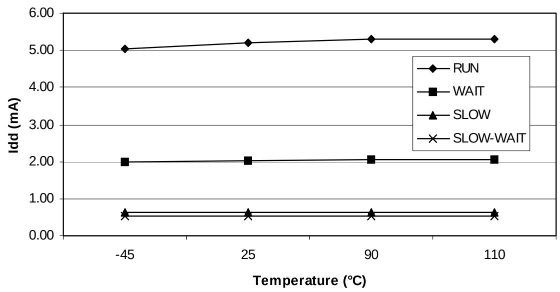
Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 73. Typical I_{DD} in SLOW-WAIT vs f_{CPU}



Note: Graph displays data beyond the normal operating range of 3V - 5.5V

Figure 74. Typical I_{DD} vs Temperature at V_{DD} = 5V and f_{CPU} = 8 MHz



SUPPLY CURRENT CHARACTERISTICS (Cont'd)

13.4.2 On-chip Peripherals

Symbol	Parameter	Conditions		Typ	Unit
$I_{DD(AT)}$	12-bit Auto-Reload Timer supply current[1)]	f_{CPU} = 4 MHz	V_{DD} = 3.3V	150	µA
		f_{CPU} = 8 MHz	V_{DD} = 5V	1000	
$I_{DD(SPI)}$	SPI supply current[2)]	f_{CPU} = 4 MHz	V_{DD} = 3.3V	50	
		f_{CPU} = 8 MHz	V_{DD} = 5V	200	
$I_{DD(ADC)}$	ADC supply current when converting[3)]	f_{ADC} = 4 MHz	V_{DD} = 3.3V	250	
			V_{DD} = 5V	1100	

Notes:

1. Data based on a differential I_{DD} measurement between reset configuration (timer stopped) and a timer running in PWM mode at f_{cpu} = 8 MHz.
2. Data based on a differential I_{DD} measurement between reset configuration and a permanent SPI master communication (data sent equal to 55h).
3. Data based on a differential I_{DD} measurement between reset configuration and continuous A/D conversions.

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[1)]	Conditions	Min	Typ[2)]	Max	Unit
$t_{c(INST)}$	Instruction cycle time	f_{CPU} = 8 MHz	2	3	12	t_{CPU}
			250	375	1500	ns
$t_{v(IT)}$	Interrupt reaction time[3)] $t_{v(IT)} = \Delta t_{c(INST)} + 10$		10		22	t_{CPU}
			1.25		2.75	µs

Notes:

1. Guaranteed by Design. Not tested in production.
2. Data based on typical application software.
3. Time measured between interrupt event and interrupt vector fetch. $Dt_{c(INST)}$ is the number of t_{CPU} cycles needed to finish the current instruction execution.

13.5.2 Auto Wake-Up from Halt Oscillator (AWU)[1)]

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{AWU}	AWU oscillator frequency		50	125	250	kHz
t_{RCSRT}	AWU oscillator start-up time				50	µs

Notes:

1. Guaranteed by Design. Not tested in production.

CLOCK AND TIMING CHARACTERISTICS (Cont'd)

13.5.3 Crystal and Ceramic Resonator Oscillators

The ST7 internal clock can be supplied with ten different Crystal/Ceramic resonator oscillators. All the information given in this paragraph are based on characterization results with specified typical external components. In the application, the resonator and the load capacitors must be placed as close as possible to the oscillator pins in order to minimize output distortion and start-up stabilization time. Refer to the crystal/ceramic resonator manufacturer for more details (frequency, package, accuracy...).

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{CrOSC}	Crystal oscillator frequency[1]		2		16	MHz
C_{L1} C_{L2}	Recommended load capacitance versus equivalent serial resistance of the crystal or ceramic resonator (R_S)		See table below			pF

Supplier	f_{CrOSC} (MHz)	Typical Ceramic Resonators[2]		CL1[4] (pF)	CL2[4] (pF)	Rd (Ω)	Supply Voltage Range (V)	Temperature Range (°C)
		Type[3]	Reference					
Murata	1	SMD	CSBFB1M00J58-R0	220	220	2.2k	3.6 to 5.5	-40 to +125
		LEAD	CSBLA1M00J58-B0					
	2	SMD	CSTCC2M00G56Z-R0	(47)	(47)	0	3.0 to 5.5	
	4	SMD	CSTCR4M00G53Z-R0	(15)	(15)			
		LEAD	CSTLS4M00G53Z-B0					
	8	SMD	CSTCE8M00G52Z-R0	(10)	(10)			
		LEAD	CSTLS8M00G53Z-B0	(15)	(15)			
	12	SMD	CSTCE12M0G52Z-R0	(10)	(10)		3.6 to 5.5	
	16	SMD	CSTCE16M0V51Z-R0	(5)	(5)			
		LEAD	CSTLS16M0X51Z-B0					

Notes:

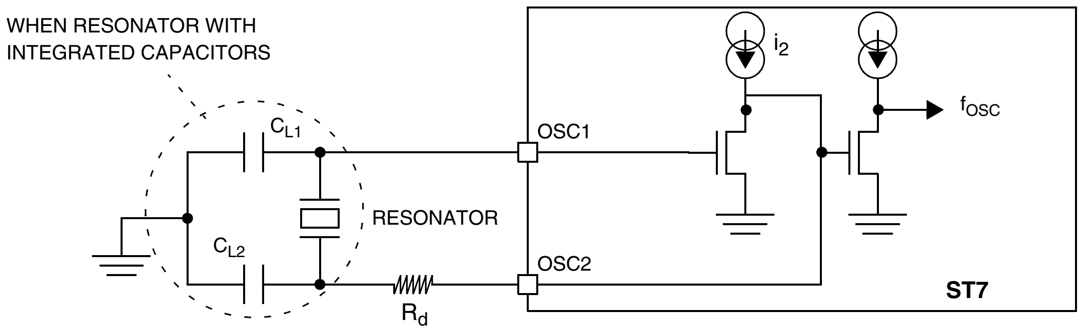
1. When PLL is used, please refer to the PLL characteristics chapter and to "SUPPLY, RESET AND CLOCK MANAGEMENT" on page 21 chapter (f_{CrOSC} min. is 8 MHz with PLL).

2. Resonator characteristics are given by the ceramic resonator manufacturer. For more information on these resonators, please consult www.murata.com.

3. SMD = [-R0: Plastic tape package (∅ =180mm)]
LEAD = [-B0: Bulk]

4. () means load capacitor built in resonator

Figure 75. Typical Application with a Crystal or Ceramic Resonator



ELECTRICAL CHARACTERISTICS (Cont'd)

13.6 MEMORY CHARACTERISTICS

13.6.1 RAM and Hardware Registers

T_A = -40 to +125°C, unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{RM}	Data retention mode[1]	HALT mode (or RESET)	1.6			V

13.6.2 FLASH Program Memory

T_A = -40 to +85°C, unless otherwise specified

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DD}	Operating voltage for Flash write/erase	Refer to operating range of V_{DD} with T_A, section 13.3.1 on page 100	3.0		5.5	V
t_{prog}	Programming time for 1~32 bytes[2]	T_A = –40 to +85°C		5	10	ms
	Programming time for 1.5 Kbytes	T_A = 25°C		0.24	0.48	s
t_{RET}[4]	Data retention	T_A = 55°C[3]	20			years
N_{RW}	Write erase cycles	T_{PROG} = 25°C	1K			cycles
		T_{PROG} = 85°C	300			
I_{DD}	Supply current	Read / Write / Erase modes f_{CPU} = 8 MHz, V_{DD} = 5.5V			2.6[5]	mA
		No Read/No Write Mode			100	µA
		Power down mode / HALT		0	0.1	

13.6.3 EEPROM Data Memory

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

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DD}	Operating voltage for EEPROM write/ erase	Refer to operating range of V_{DD} with T_A, section 13.3.1 on page 100	3.0		5.5	V
t_{prog}	Programming time for 1~32 bytes	T_A = –40 to +125°C		5	10	ms
t_{RET}[4]	Data retention with 1k cycling (T_{PROG} = –40 to +125°C	T_A = 55°C[3]	20			years
	Data retention with 10k cycling (T_{PROG} = –40 to +125°C)		10			
	Data retention with 100k cycling (T_{PROG} = –40 to +125°C)		1			

Notes:

1. Minimum V_{DD} supply voltage without losing data stored in RAM (in HALT mode or under RESET) or in hardware registers (only in HALT mode). Guaranteed by construction, not tested in production.
2. Up to 32 bytes can be programmed at a time.
3. The data retention time increases when the T_A decreases.
4. Data based on reliability test results and monitored in production.
5. Guaranteed by Design. Not tested in production.

ELECTRICAL CHARACTERISTICS (Cont'd)

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**: A Burst of Fast Transient voltage (positive and negative) is applied to V_{DD} and V_{SS} through a 100pF capacitor, until a functional disturbance occurs. This test conforms with the IEC 1000-4-4 standard.

A device reset allows normal operations to resume. The test results are given in the table below based on the EMS levels and classes defined in application note AN1709.

13.7.1.1 Designing Hardened Software to Avoid Noise Problems

EMC characterization and optimization are performed at component level with a typical application environment and simplified MCU software. It should be noted that good EMC performance is highly dependent on the user application and the software in particular.

Therefore, it is recommended that EMC software optimization and prequalification tests are made relative to the EMC level requested for the user's 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 behavior is detected, the software can be hardened to prevent unrecoverable errors occurring (see application note AN1015).

Symbol	Parameter	Conditions	Level/ Class
V_{FESD}	Voltage limits to be applied on any I/O pin to induce a functional disturbance	V_{DD} = 5V, T_A = 25°C, f_{OSC} = 8 MHz, conforms to IEC 1000-4-2	2B
V_{FFTB}	Fast transient voltage burst limits to be applied through 100pF on V_{DD} and V_{DD} pins to induce a functional disturbance	V_{DD} = 5V, T_A = 25°C, f_{OSC} = 8 MHz, conforms to IEC 1000-4-4	3B

13.7.2 Electro Magnetic Interference (EMI)

Based on a simple application running on the product (toggling two LEDs through the I/O ports), the product is monitored in terms of emission. This emission test is in line with the norm SAE J 1752/3 which specifies the board and the loading of each pin.

Symbol	Parameter	Conditions	Monitored Frequency Band	Max vs [f_{OSC}/f_{CPU}]		Unit
				8/4 MHz	16/8 MHz	
S_{EMI}	Peak level[1)]	V_{DD} = 5V, T_A = 25°C, SO20 package, conforming to SAE J 1752/3	0.1 MHz to 30 MHz	15	20	dBμV
			30 MHz to 130 MHz	17	21	
			130 MHz to 1 GHz	12	15	
			SAE EMI Level	3		-

Notes:

1. Data based on characterization results, not tested in production.

EMC CHARACTERISTICS (Cont'd)

13.7.3 Absolute Maximum Ratings (Electrical Sensitivity)

Based on two different tests (ESD and LU) using specific measurement methods, the product is stressed in order to determine its performance in terms of electrical sensitivity. For more details, refer to application note AN1181.

13.7.3.1 Electro-Static Discharge (ESD)

Electro-Static Discharges (a positive then a negative pulse separated by 1 second) are applied to the pins of each sample according to each pin combination. The sample size depends on the number of supply pins in the device (3 parts*(n+1) supply pin). Two models can be simulated: Human Body Model and Machine Model. This test conforms to the JESD22-A114A/A115A standard.

Absolute Maximum Ratings

Symbol	Ratings	Conditions	Maximum value[1)]	Unit
$V_{ESD(HBM)}$	Electro-static discharge voltage (Human Body Model)	$T_A = 25°C$	8000	V
$V_{ESD(MM)}$	Electro-static discharge voltage (Machine Model)		400	
$V_{ESD(CDM)}$	Electro-static discharge voltage (Charge Device Model)		1000	

Notes:
1. Data based on characterization results, not tested in production.

13.7.3.2 Static and Dynamic Latch-Up

- **LU**: Three complementary static tests are required on six parts to assess the latch-up performance. A supply overvoltage (applied to each power supply pin) and a current injection (applied to each input, output and configurable I/O pin) are performed on each sample. This test conforms to the EIA/JESD 78 IC latch-up standard. For more details, refer to application note AN1181.

Electrical Sensitivities

Symbol	Parameter	Conditions	Class[1)]
LU	Static latch-up class	$T_A = 25°C$ $T_A = 125°C$	A
DLU	Dynamic latch-up class	$V_{DD} = 5.5V$, $f_{OSC} = 4$ MHz, $T_A = 25°C$	A

Notes:
1. Class description: A Class is an STMicroelectronics internal specification. All its limits are higher than the JEDEC specifications, which means when a device belongs to Class A it exceeds the JEDEC standard. Class B strictly covers all the JEDEC criteria (international standard).

ELECTRICAL CHARACTERISTICS (Cont'd)

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	Max	Unit
V_{IL}	Input low level voltage			V_{SS} - 0.3		$0.3xV_{DD}$	V
V_{IH}	Input high level voltage			$0.7xV_{DD}$		V_{DD} + 0.3	
V_{hys}	Schmitt trigger voltage hysteresis[1)]				400		mV
I_L	Input leakage current	$V_{SS} \leq V_{IN} \leq V_{DD}$				±1	µA
I_S	Static current consumption induced by each floating input pin[2)]	Floating input mode			400		
R_{PU}	Weak pull-up equivalent resistor[3)]	$V_{IN} = V_{SS}$	V_{DD} = 5V	50	120	250	kΩ
			V_{DD} = 3V		160		
C_{IO}	I/O pin capacitance				5		pF
$t_{f(IO)out}$	Output high to low level fall time[1)]	C_L = 50pF Between 10% and 90%			25		ns
$t_{r(IO)out}$	Output low to high level rise time[1)]				25		
$t_{w(IT)in}$	External interrupt pulse time[4)]			1			t_{CPU}

Notes:
1. Data based on validation/design results.
2. Configuration not recommended, all unused pins must be kept at a fixed voltage: Using the output mode of the I/O for example or an external pull-up or pull-down resistor (see Figure 76). Static peak current value taken at a fixed V_{IN} value, based on design simulation and technology characteristics, not tested in production. This value depends on V_{DD} and temperature values.
3. The R_{PU} pull-up equivalent resistor is based on a resistive transistor (corresponding I_{PU} current characteristics described in Figure 77 on page 115).
4. To generate an external interrupt, a minimum pulse width must be applied on an I/O port pin configured as an external interrupt source.

Figure 76. Two Typical Applications with Unused I/O Pin

Caution: During normal operation the ICCCLK pin must be pulled up, internally or externally (external pull-up of 10k mandatory in noisy environment). This is to avoid entering ICC mode unexpectedly during a reset.
Note: I/O can be left unconnected if it is configured as output (0 or 1) by the software. This has the advantage of greater EMC robustness and lower cost.

I/O PORT PIN CHARACTERISTICS (Cont'd)

13.8.2 Output Driving Current

Subject to general operating conditions for V_{DD}, f_{CPU} and T_A (-40 to +125°C), unless otherwise specified.

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{OL}[1]	Output low level voltage for a standard I/O pin when eight pins are sunk at same time (see Figure 78)	V_{DD} = 5V	I_{IO} = +5mA			1.0	V
			I_{IO} = +2mA			0.4	
	Output low level voltage for a high sink I/O pin when four pins are sunk at same time (see Figure 81)		I_{IO} = +20mA			1.3	
			I_{IO} = +8mA			0.75	
V_{OH}[2]	Output high level voltage for an I/O pin when four pins are sourced at same time (see Figure 87)		I_{IO} = -5mA	V_{DD} - 1.5			
			I_{IO} = -2mA	V_{DD} - 0.8			
V_{OL}[1)3)]	Output low level voltage for a standard I/O pin when eight pins are sunk at same time (see Figure 77)	V_{DD} = 3.3V	I_{IO} = +2mA, $T_A \leq$ +85°C			0.5	
	Output low level voltage for a high sink I/O pin when four pins are sunk at same time		I_{IO} = +8mA, $T_A \leq$ +85°C				
V_{OH}[2)3)]	Output high level voltage for an I/O pin when four pins are sourced at same time (Figure 86)		I_{IO} = -2mA, $T_A \leq$ +85°C	V_{DD} - 0.8			

Notes:

1. The I_{IO} current sunk must always respect the absolute maximum rating specified in section 13.2.2 on page 99 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 on page 99 and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VDD}.
3. Not tested in production, based on characterization results.

I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 77. Typical V_{OL} at V_{DD} = 3.3V (Standard)

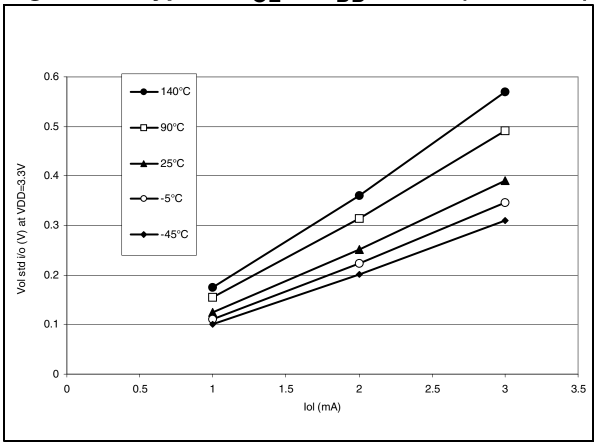


Figure 78. Typical V_{OL} at V_{DD} = 5V (Standard)

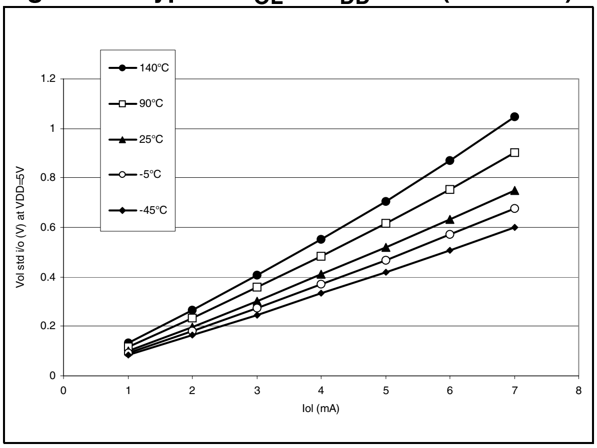


Figure 79. Typical V_{OL} at V_{DD} = 3.3V (Port C)

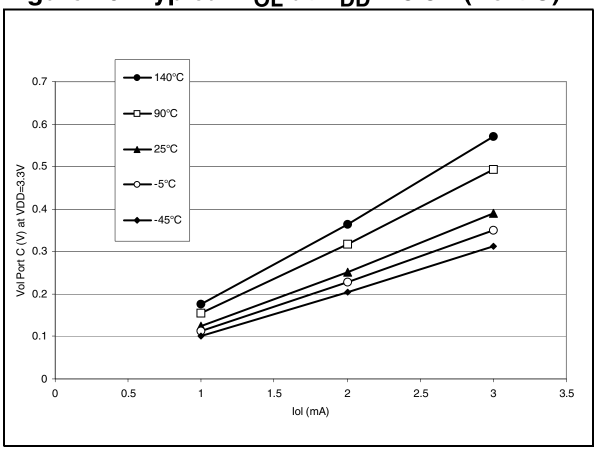


Figure 80. Typical V_{OL} at V_{DD} = 5V (Port C)

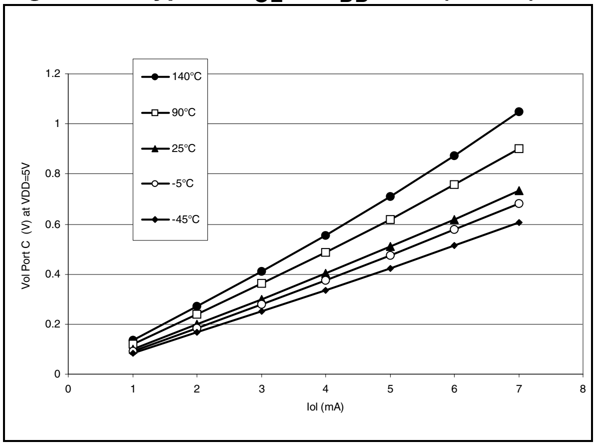


Figure 81. Typical V_{OL} at V_{DD} = 3.3V (High-sink)

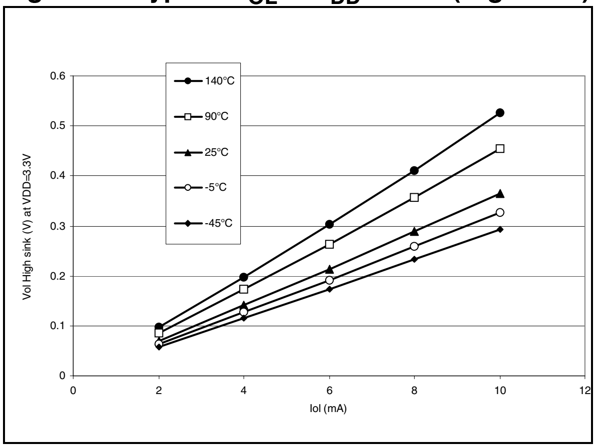
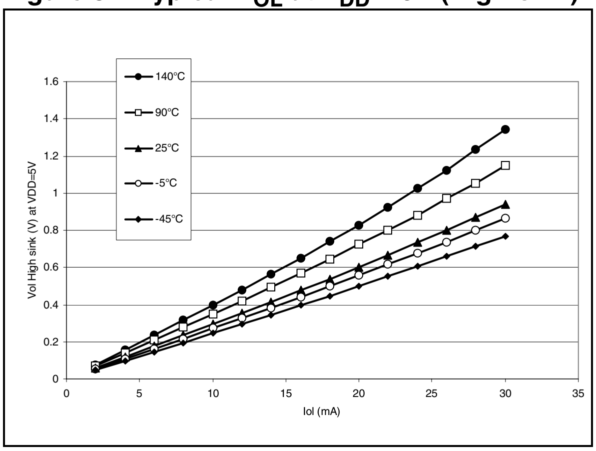


Figure 82. Typical V_{OL} at V_{DD} = 5V (High-sink)



I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 83. Typical V_{OL} vs V_{DD} (Standard I/Os)

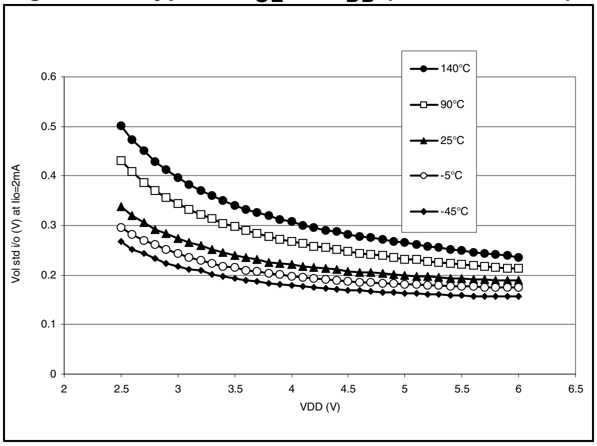


Figure 84. Typical V_{OL} vs V_{DD} (High-sink)

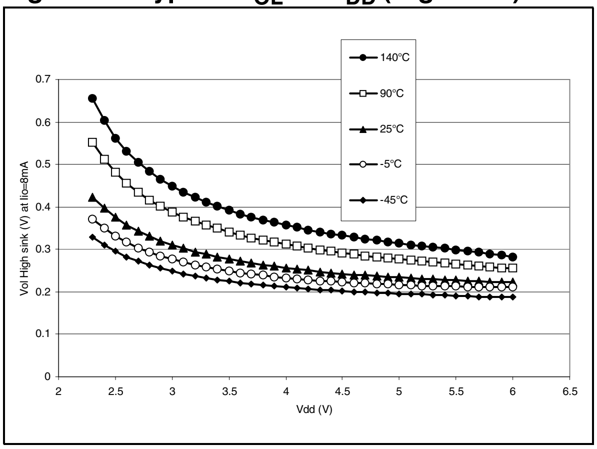


Figure 85. Typical V_{OL} vs V_{DD} (Port C)

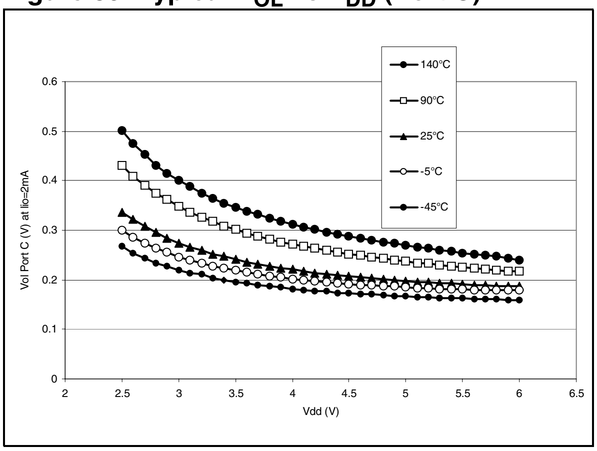


Figure 86. Typical V_{DD} - V_{OH} at V_{DD} = 3.3V

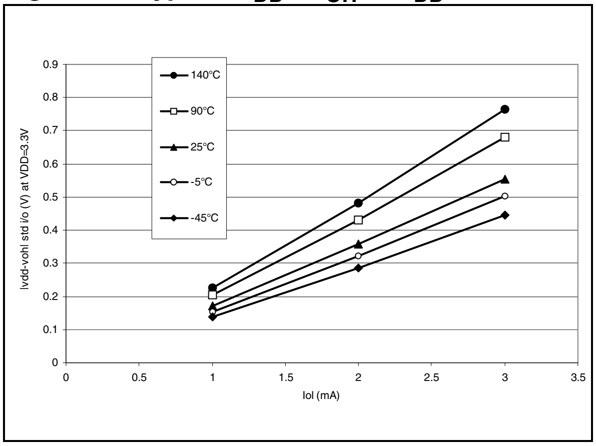


Figure 87. Typical V_{DD} - V_{OH} at V_{DD} = 5V

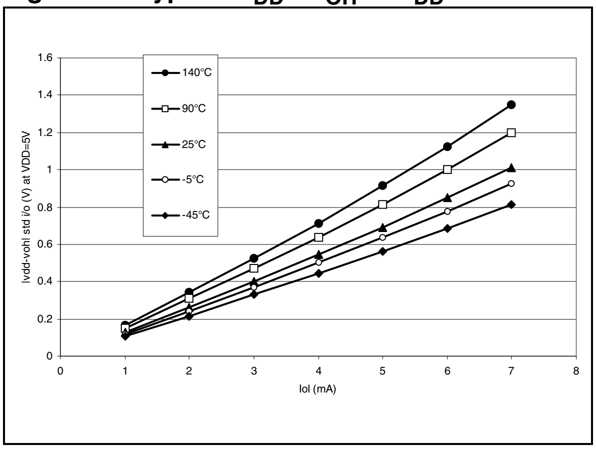
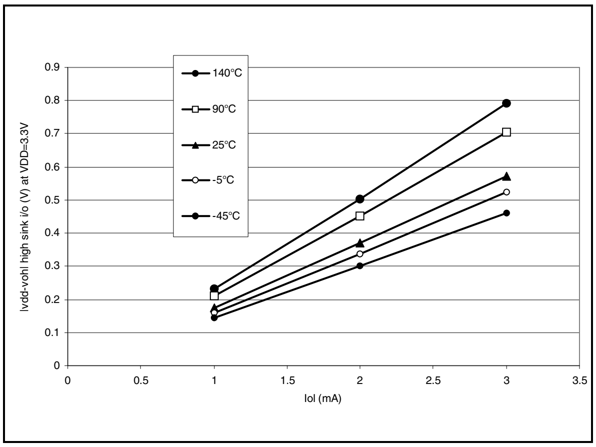


Figure 88. Typical V_{DD} - V_{OH} at V_{DD} = 3.3V (HS)



I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 89. Typical V_{DD} - V_{OH} at V_{DD} = 5V (HS)

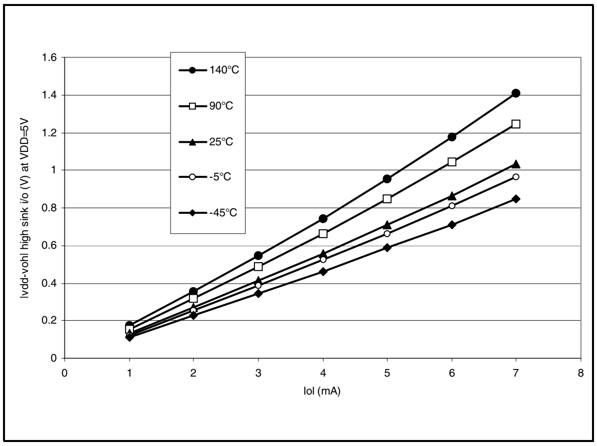


Figure 90. Typical V_{DD} - V_{OH} at V_{DD} = 3.3V (Port C)

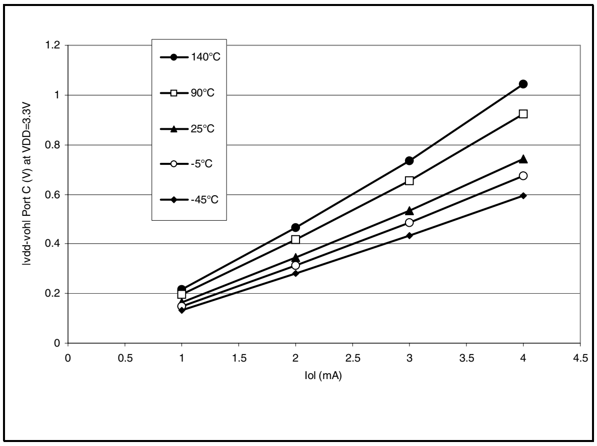


Figure 91. Typical V_{DD}-V_{OH} at V_{DD} = 5V (Port C)

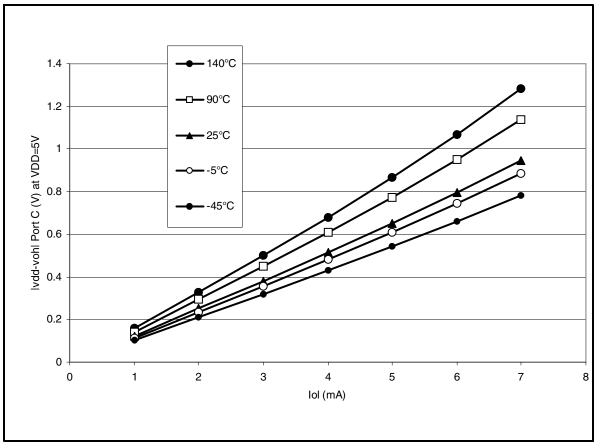


Figure 92. Typical V_{DD} - V_{OH} vs V_{DD} (Standard)

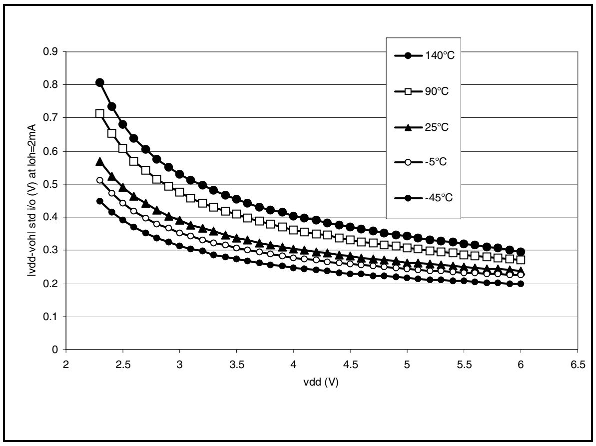


Figure 93. Typical V_{DD}-V_{OH} vs V_{DD} (High Sink)

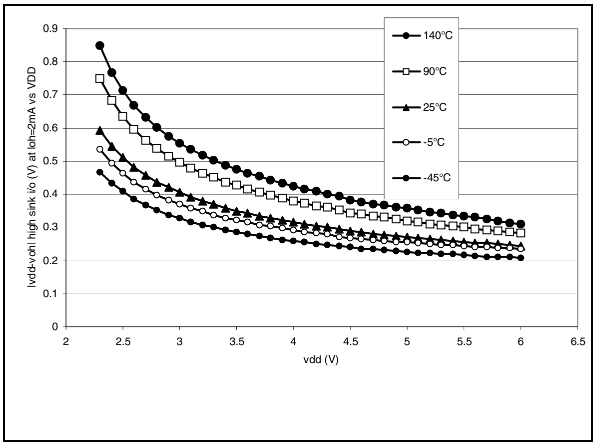
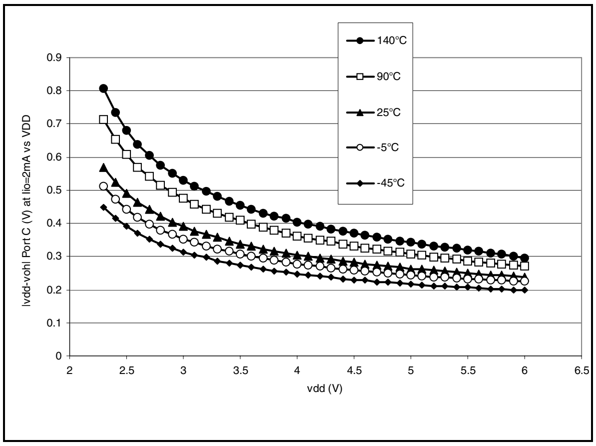


Figure 94. Typical V_{DD}-V_{OH} vs V_{DD} (Port C)



ELECTRICAL CHARACTERISTICS (Cont'd)

13.9 CONTROL PIN CHARACTERISTICS

13.9.1 Asynchronous $\overline{\text{RESET}}$ Pin

T_A = -40 to +125°C, unless otherwise specified.

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{IL} 1)	Input low-level voltage			V_{ss} - 0.3		$0.3xV_{DD}$	V
V_{IH} 1)	Input high-level voltage			$0.7xV_{DD}$		V_{DD} + 0.3	V
V_{hys}	Schmitt trigger voltage hysteresis 1)				2		V
V_{OL} 1)	Output low-level voltage 2)	V_{DD} = 5V	I_{IO} = +5mA, $T_A \leq$ +125°C		0.5	1.0	V
			I_{IO} = +2mA, $T_A \leq$ +125°C		0.2	0.4	V
R_{ON}	Pull-up equivalent resistor 1)3)	V_{DD} = 5V		20	40	80	kΩ
		V_{DD} = 3.3V 1)		40	70	120	kΩ
$t_{w(RSTL)out}$	Generated reset pulse duration	Internal reset sources			30		µs
$t_{h(RSTL)in}$	External reset pulse hold time 4)			20			µs
$t_{g(RSTL)in}$	Filtered glitch duration				200		ns

Notes:

1. Data based on characterization results, not tested in production.
2. The I_{IO} current sunk must always respect the absolute maximum rating specified in section 13.2.2 on page 99 and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VSS}.
3. The R_{ON} pull-up equivalent resistor is based on a resistive transistor. Specified for voltages on $\overline{\text{RESET}}$ pin between V_{ILmax} and V_{DD}.
4. To guarantee the reset of the device, a minimum pulse must be applied to the $\overline{\text{RESET}}$ pin. All short pulses applied on $\overline{\text{RESET}}$ pin with a duration below $t_{h(RSTL)in}$ can be ignored.

CONTROL PIN CHARACTERISTICS (Cont'd)

Figure 95. RESET Pin Protection When LVD Is Enabled[1)2)3)4)]

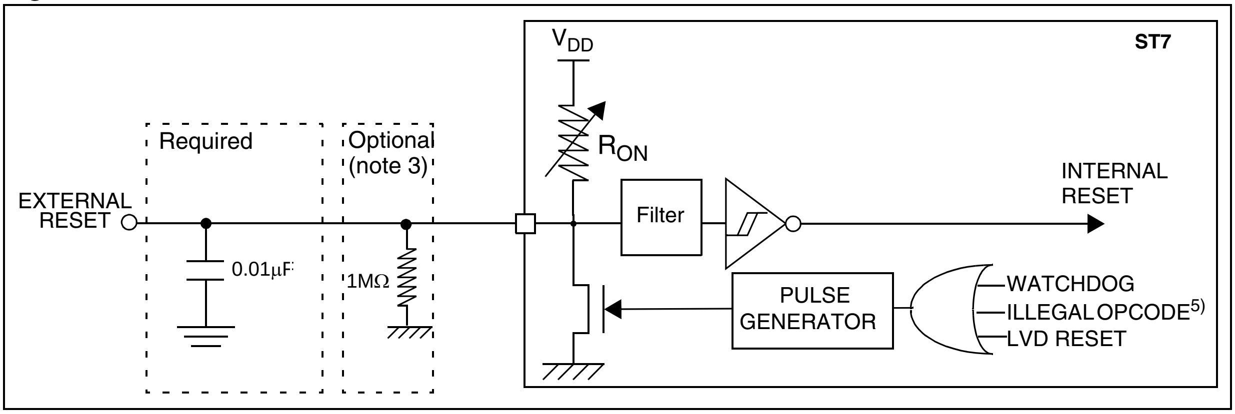
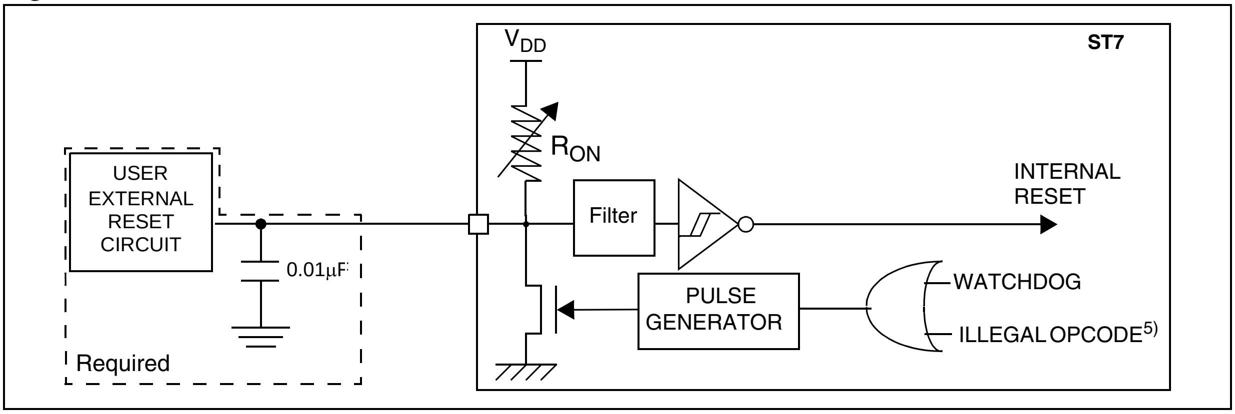


Figure 96. RESET Pin Protection When LVD Is Disabled[1)]



Note 1:

- The reset network protects the device against parasitic resets.
- The output of the external reset circuit must have an open-drain output to drive the ST7 reset pad. Otherwise the device can be damaged when the ST7 generates an internal reset (LVD or Watchdog).
- Whatever the reset source is (internal or external), the user must ensure that the level on the RESET pin can go below the V_{IL} maximum level specified in section 13.9.1 on page 118. Otherwise the reset is not taken into account internally.
- Because the reset circuit is designed to allow the internal RESET to be output in the RESET pin, the user must ensure that the current sunk on the RESET pin is less than the absolute maximum value specified for $I_{INJ(RESET)}$ in section 13.2.2 on page 99.

Note 2: When the LVD is enabled, it is recommended not to connect a pull-up resistor or capacitor. A 10nF pull-down capacitor is required to filter noise on the reset line.

Note 3: If a capacitive power supply is used, it is recommended to connect a 1MΩ pull-down resistor to the RESET pin to discharge any residual voltage induced by the capacitive effect of the power supply (this adds 5µA to the power consumption of the MCU).

Note 4: Tips when using the LVD:

- 1. Check that all recommendations related to ICCCLK and reset circuit have been applied (see caution in Table 1 on page 5 and notes above)
- 2. Check that the power supply is properly decoupled (100nF + 10µF close to the MCU). Refer to AN1709 and AN2017. If this cannot be done, it is recommended to put a 100nF + 1MΩ pull-down on the RESET pin.
- 3. The capacitors connected on the RESET pin and also the power supply are key to avoid any start-up marginality. In most cases, steps 1 and 2 above are sufficient for a robust solution. Otherwise: Replace 10nF pull-down on the RESET pin with a 5µF to 20µF capacitor."

Note 5: Please refer to section 12.2.1 on page 95 for more details on illegal opcode reset conditions.

ELECTRICAL CHARACTERISTICS (Cont'd)

13.10 COMMUNICATION INTERFACE CHARACTERISTICS

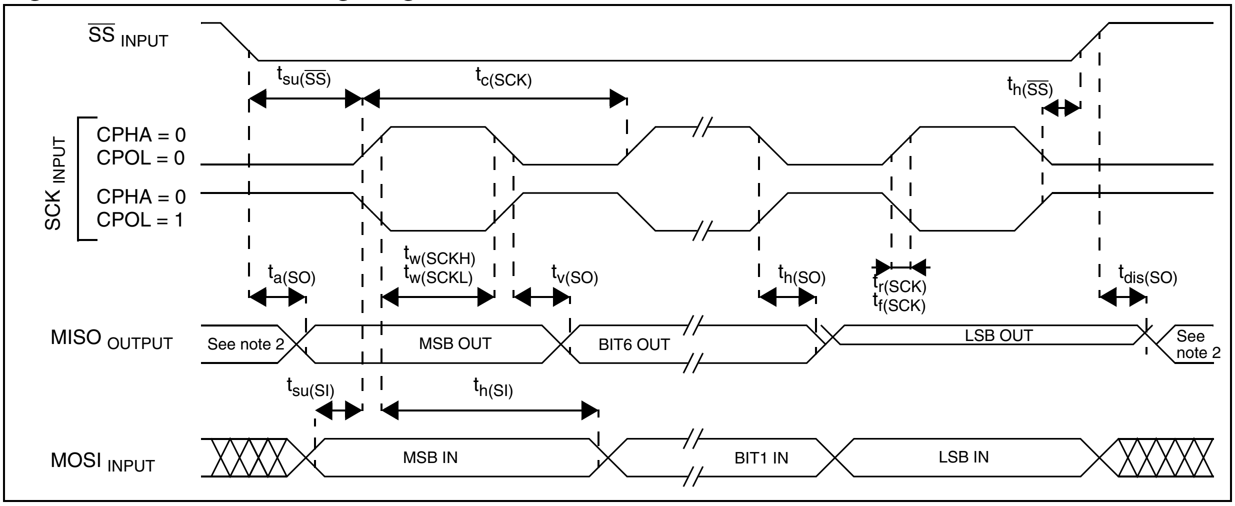
13.10.1 SPI - Serial Peripheral Interface

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 ($\overline{SS}$, SCK, MOSI, MISO).

Symbol	Parameter	Conditions	Min	Max	Unit
f_{SCK} = 1 / $t_{c(SCK)}$	SPI clock frequency	Master, f_{CPU} = 8 MHz	f_{CPU} / 128 = 0.0625	f_{CPU} / 4 = 2	MHz
		Slave, f_{CPU} = 8 MHz	0	f_{CPU} / 2 = 4	
$t_{r(SCK)}$	SPI clock rise and fall time		See I/O port pin description		
$t_{f(SCK)}$					
$t_{su(\overline{SS})}$ 1)	$\overline{SS}$ setup time 4)	Slave	(4 x T_{CPU}) + 50		ns
$t_{h(\overline{SS})}$ 1)	$\overline{SS}$ hold time		120		
$t_{w(SCKH)}$ 1)	SCK high and low time	Master	100		
$t_{w(SCKL)}$ 1)		Slave	90		
$t_{su(MI)}$ 1)	Data input setup time	Master	100		
$t_{su(SI)}$ 1)		Slave			
$t_{h(MI)}$ 1)	Data input hold time	Master			
$t_{h(SI)}$ 1)		Slave			
$t_{a(SO)}$ 1)	Data output access time	Slave	0	120	
$t_{dis(SO)}$ 1)	Data output disable time			240	
$t_{v(SO)}$ 1)	Data output valid time	Slave (after enable edge)		120	
$t_{h(SO)}$ 1)	Data output hold time		0		
$t_{v(MO)}$ 1)	Data output valid time	Master (after enable edge)		120	
$t_{h(MO)}$ 1)	Data output hold time		0		

Figure 97. SPI Slave Timing Diagram with CPHA = 0 3)



Notes:

1. Data based on design simulation, not tested in production.
2. When no communication is on-going, the data output line of the SPI (MOSI in master mode, MISO in slave mode) has its alternate function capability released. In this case, the pin status depends on the I/O port configuration.
3. Measurement points are done at CMOS levels: 0.3 x V_{DD} and 0.7 x V_{DD}.
4. Depends on f_{CPU}. For example, if f_{CPU} = 8 MHz, then T_{CPU} = 1 / f_{CPU} = 125ns and $t_{su(\overline{SS})}$ = 550ns.

COMMUNICATION INTERFACE CHARACTERISTICS (Cont'd)

Figure 98. SPI Slave Timing Diagram with CPHA = 1[1)]

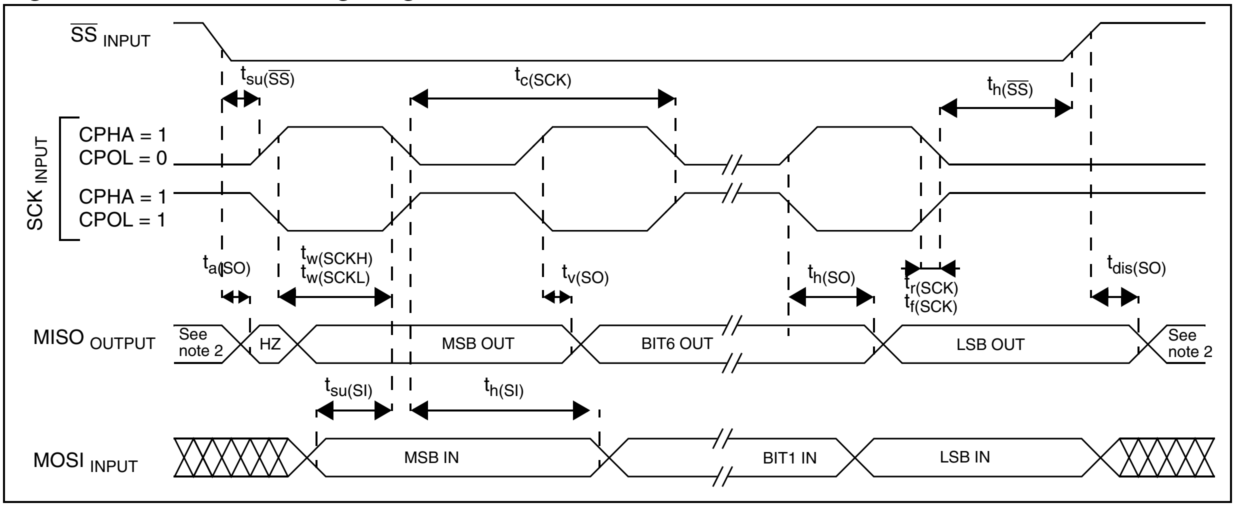
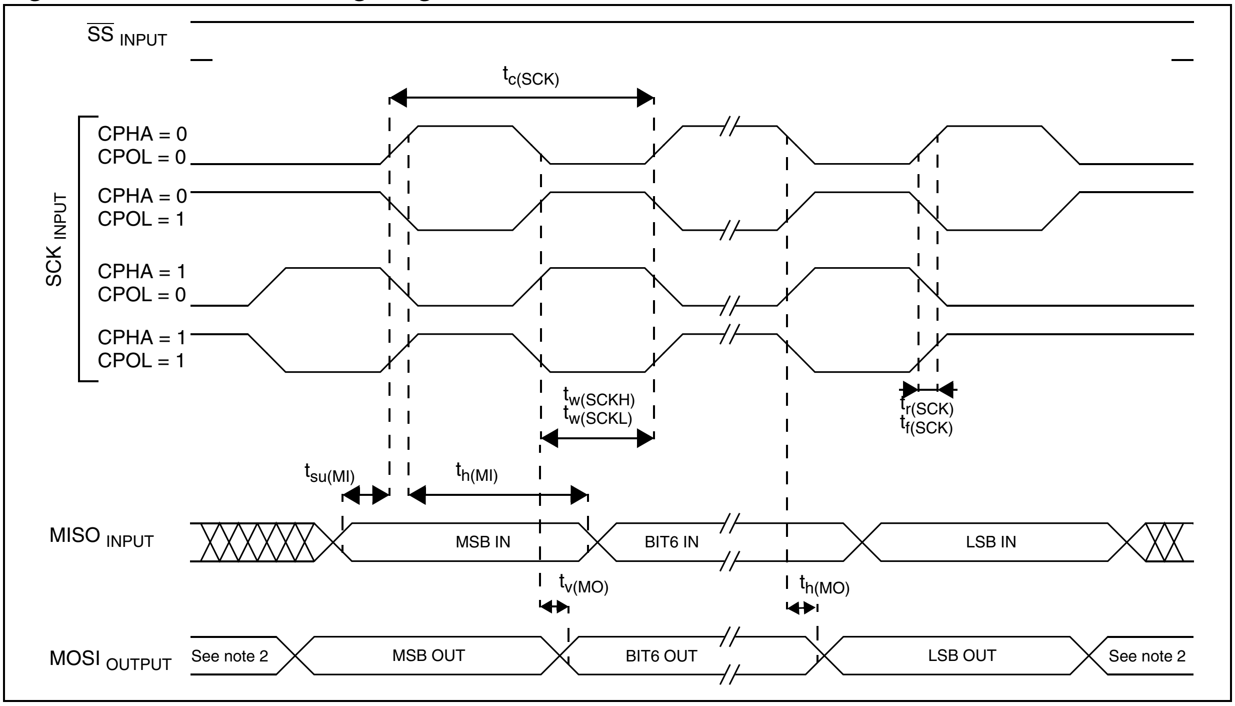


Figure 99. SPI Master Timing Diagram[1)]



Notes:

1. Measurement points are done at CMOS levels: 0.3 x V_{DD} and 0.7 x V_{DD}.
2. When no communication is on-going, the alternate function capability of the SPI's data output line (MOSI in master mode, MISO in slave mode) is released. In this case, the pin status depends on the I/O port configuration.

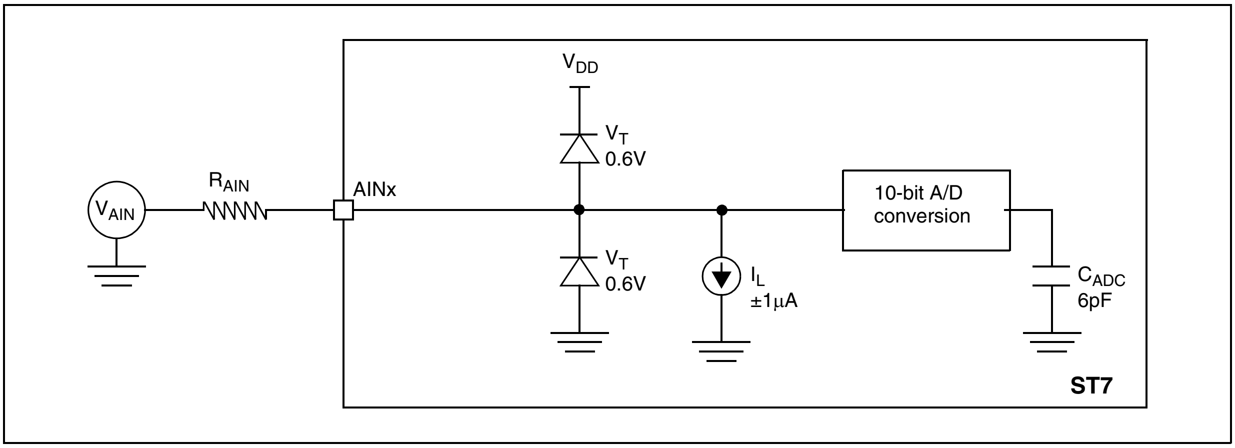
ELECTRICAL CHARACTERISTICS (Cont'd)

13.11 10-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 voltage range 2)		V_{SSA}		V_{DDA}	V
R_{AIN}	External input resistor				10 3)	kΩ
C_{ADC}	Internal sample and hold capacitor			6		pF
t_{STAB}	Stabilization time after ADC enable	f_{CPU} = 8 MHz, f_{ADC} = 4 MHz		0 4)		µs
t_{ADC}	Conversion time (Sample+Hold)			3.5		
	- Sample capacitor loading time - Hold conversion time			4 10		1 / f_{ADC}
I_{ADC}	Analog part			1		mA
	Digital part			0.2		

Figure 100. Typical Application with ADC



Notes:

1. Unless otherwise specified, typical data is based on T_A = 25°C and V_{DD} - V_{SS} = 5V. They are given only as design guidelines and are not tested.
2. When V_{DDA} and V_{SSA} pins are not available on the pinout, the ADC refers to V_{DD} and V_{SS}.
3. Any added external serial resistor downgrades 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.

Related application notes:

Understanding and minimizing ADC conversion errors (AN1636)
Software techniques for compensating ST7 ADC errors (AN1711)

10-BIT ADC CHARACTERISTICS (Cont'd)

Table 21. ADC Accuracy with $3V \le V_{DD} \le 3.6V$

Symbol	Parameter	Conditions	Typ	Max[3)]	Unit
$\|E_T\|$	Total unadjusted error		2.5		
$\|E_O\|$	Offset error		0.9		
$\|E_G\|$	Gain error	$f_{CPU} = 4$ MHz, $f_{ADC} = 2$ MHz[1)2)]	1.3	TBD	LSB
$\|E_D\|$	Differential linearity error		1.8		
$\|E_L\|$	Integral linearity error				

Table 22. ADC Accuracy with $4.5V \le V_{DD} \le 5.5V$

Symbol	Parameter	Conditions	Typ	Max	Unit
$\|E_T\|$	Total unadjusted error		4	6[1)]	
$\|E_O\|$	Offset error		3	5[1)]	
$\|E_G\|$	Gain error	$f_{CPU} = 8$ MHz, $f_{ADC} = 4$ MHz[1)2)]	1	4[1)]	LSB
$\|E_D\|$	Differential linearity error		1.5	3[3)]	
$\|E_L\|$	Integral linearity error				

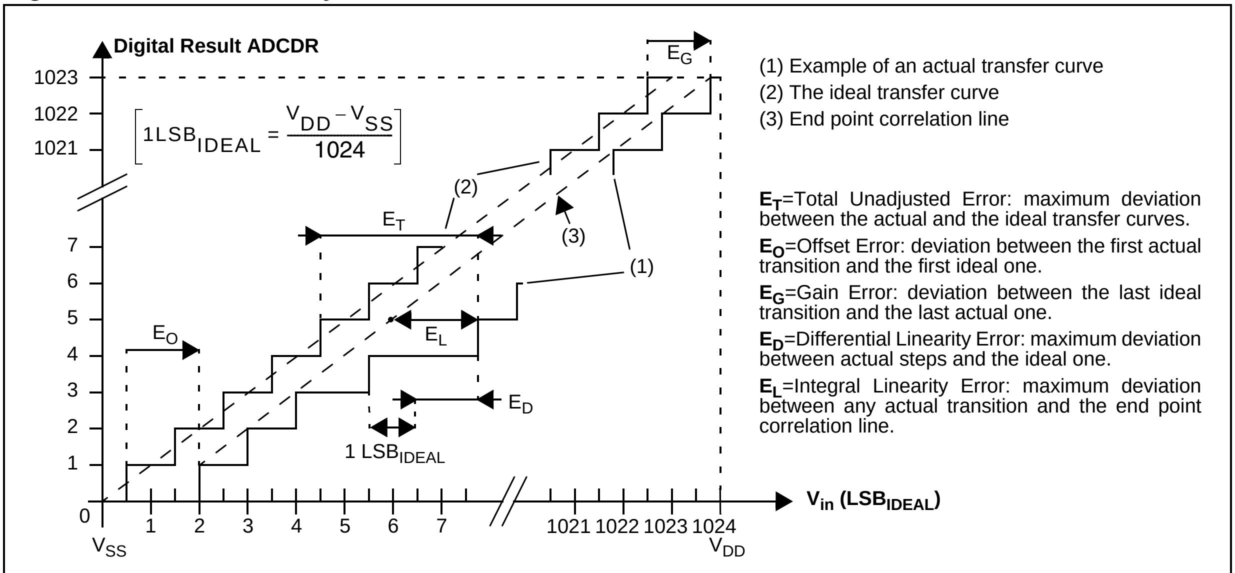
Notes:

1. Data based on characterization results over the whole temperature range, monitored in production.

2. ADC accuracy vs.negative injection current: Injecting negative current on any of the analog input pins may reduce the accuracy of the conversion being performed on another analog input.
The effect of negative injection current on robust pins is specified in section 13.11 on page 122
Any positive injection current within the limits specified for $I_{INJ(PIN)}$ and $\Sigma I_{INJ(PIN)}$ in section 13.8 on page 113 does not affect the ADC accuracy.

3. Data based on characterization results, monitored in production to guarantee 99.73% within ± max value from -40°C to +125°C (± 3σ distribution limits).

Figure 101. ADC Accuracy Characteristics



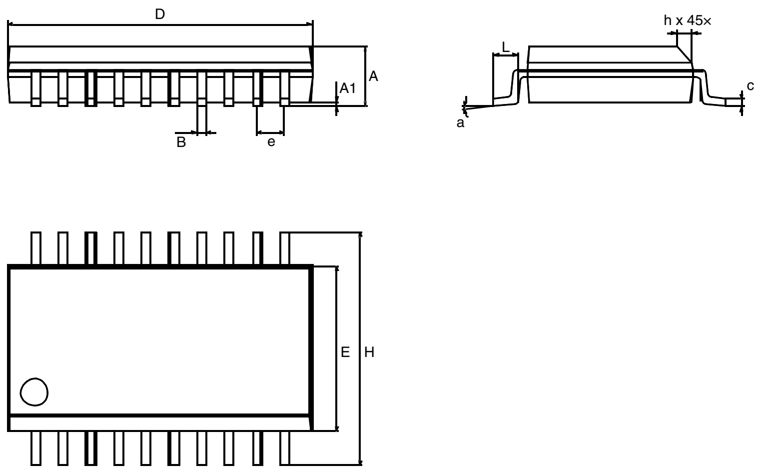
14 PACKAGE CHARACTERISTICS

In order to meet environmental requirements, ST offers these devices in ECOPACK® packages. These packages have a lead-free second level interconnect. The category of second level interconnect is marked on the package and on the inner box label, in compliance with JEDEC Standard JESD97. The maximum ratings related to soldering conditions are also marked on the inner box label.

ECOPACK is an ST trademark. ECOPACK specifications are available at www.st.com.

14.1 PACKAGE MECHANICAL DATA

Figure 102. 20-Pin Plastic Small Outline Package, 300-mil Width



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A	2.35		2.65	0.093		0.104
A1	0.10		0.30	0.004		0.012
B	0.33		0.51	0.013		0.020
C	0.23		0.32	0.009		0.013
D	12.60		13.00	0.496		0.512
E	7.40		7.60	0.291		0.299
e		1.27			0.050	
H	10.00		10.65	0.394		0.419
h	0.25		0.75	0.010		0.030
α	0°		8°	0°		8°
L	0.40		1.27	0.016		0.050
	Number of Pins					
N	20					

Table 23. THERMAL CHARACTERISTICS

Symbol	Ratings	Value	Unit
R_{thJA}	Package thermal resistance (junction to ambient)	85	°C/W
T_{Jmax}	Maximum junction temperature[1)]	150	°C
P_{Dmax}	Power dissipation[2)]	300	mW

Notes:

1. The maximum chip-junction temperature is based on technology characteristics.
2. The maximum power dissipation is obtained from the formula $P_D = (T_J - T_A) / R_{thJA}$.

The power dissipation of an application can be defined by the user with the formula: $P_D = P_{INT} + P_{PORT}$, where P_{INT} is the chip internal power ($I_{DD} x V_{DD}$) and P_{PORT} is the port power dissipation depending on the ports used in the application.

PACKAGE CHARACTERISTICS (Cont'd

14.2 SOLDERING INFORMATION

In accordance with the RoHS European directive, all STMicroelectronics packages have been converted to lead-free technology, named ECOPACK^{TM}.

- ECOPACK^{TM} packages are qualified according to the JEDEC STD-020C compliant soldering profile.
- Detailed information on the STMicroelectronics ECOPACK^{TM} transition program is available on www.st.com/stonline/leadfree/, with specific technical application notes covering the main technical aspects related to lead-free conversion (AN2033, AN2034, AN2035, AN2036).

Backward and forward compatibility:

The main difference between Pb and Pb-free soldering process is the temperature range.

– ECOPACK^{TM} SO packages are fully compatible with lead (Pb) containing soldering process (see application note AN2034)

– SO Pb-packages are compatible with lead-free soldering process, nevertheless it's the customer's duty to verify that the Pb-packages maximum temperature (mentioned on the inner box label) is compatible with their lead-free soldering temperature.

Table 24. Soldering Compatibility (wave and reflow soldering process)

Package	Plating material devices	Pb solder paste	Pb-free solder paste
SO	NiPdAu (Nickel-palladium-Gold)	Yes	Yes*

* Assemblers must verify that the Pb-package maximum temperature (mentioned on the inner box label) is compatible with their lead-free soldering process.

15 DEVICE CONFIGURATION AND ORDERING INFORMATION

Each device is available for production in user programmable versions (FLASH). ST7L1 devices are shipped to customers with a default program memory content (FFh). This implies that FLASH devices must be configured by the customer using the Option Bytes.

15.1 OPTION BYTES

The 2 option bytes select the hardware configuration of the microcontroller.

The option bytes are accessed only in programming mode (for example, using a standard ST7 programming tool).

OPTION BYTE 0

OPT7 = **Reserved**, must always be 0.

OPT6 = **Reserved**, must always be 1.

OPT5:4 = **CLKSEL** *Clock Source Selection*
When the internal RC oscillator is not selected (Option OSC = 1), these option bits select the clock source: Resonator oscillator or external clock

Clock Source		Port C	CLKSEL	
Resonator		Ext. Osc Enabled/ Port C Disabled	0	0
External Clock source: CLKIN	on PB4	Ext. Osc Disabled/ Port C Enabled	0	1
	on PC0		1	1
Reserved			1	0

Note: When the internal RC oscillator is selected, the CLKSEL option bits must be kept at their default value in order to select the 256 clock cycle delay (see Section 7.5).

OPT3:2 = **SEC[1:0]** *Sector 0 size definition*
These option bits indicate the size of sector 0 according to the following table.

Sector 0 Size	SEC1	SEC0
0.5k	0	0
1k		1
2k	1	0
4k		1

OPT1 = **FMP_R** *Read-out protection*
Readout protection, when selected, provides a protection against program memory content extraction and against write access to Flash memory. Erasing the option bytes when the FMP_R option is selected causes the whole memory to be erased first and the device can be reprogrammed. Refer to the *ST7 Flash Programming Reference Manual* and section 4.5 on page 12 for more details
0: Read-out protection off
1: Read-out protection on

OPT0 = **FMP_W** *FLASH write protection*
This option indicates if the FLASH program memory is write protected.
Warning: When this option is selected, the program memory (and the option bit itself) can never be erased or programmed again.
0: Write protection off
1: Write protection on

	OPTION BYTE 0 (7 … 0)								OPTION BYTE 1 (7 … 0)							
	Res.	Res.	CLKSEL		SEC1	SEC0	FMP R	FMP W	PLL x4x8	PLL OFF	Res.	OSC	LVD1	LVD0	WDG SW	WDG HALT
Default Value	0	1	1	1	0	1	0	0	1	1	1	0	1	1	1	1

OPTION BYTES (Cont'd)

OPTION BYTE 1

OPT7 = **PLLx4x8** *PLL Factor selection (must be set to 1 for ROM devices).*
0: PLLx4
1: PLLx8

OPT6 = **PLLOFF** *PLL disable.*
0: PLL enabled
1: PLL disabled (by-passed)

OPT5 = **Reserved.** Must be set to 1.

OPT4 = **OSC** *RC Oscillator selection*
0: RC oscillator on
1: RC oscillator off

Note:

– If the RC oscillator is selected, then to improve clock stability and frequency accuracy, it is recommended to place a decoupling capacitor, typically 100nF, between the V_{DD} and V_{SS} pins as close as possible to the ST7 device.

OPT3:2 = **LVD[1:0]** *Low voltage detection selection*
These option bits enable the voltage detection block (LVD and AVD) with a selected threshold to the LVD and AVD.

Configuration	LVD1	LVD0
LVD Off	1	1
LVD On (Highest Voltage Threshold)		0

OPT1 = **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)

OPT0 = **WDG HALT** *Watchdog Reset on Halt*
This option bit determines if a RESET is generated when entering HALT mode while the Watchdog is active.
0: No Reset generation when entering HALT mode
1: Reset generation when entering HALT mode

Table 25. List of Valid Option Combinations

Operating conditions				Option Bits		
V_{DD} range	Clock Source	PLL	Typ f_{CPU}	OSC	PLLOFF	PLLx4x8
3.0 to 3.6V	Internal RC 1%	off	1 MHz @ 3.3V	0	1	1
		x4	4 MHz @ 3.3V		0	0
		x8	-	-	-	-
	External clock or resonator (depending on OPT5:4 selection)	off	0 to 4 MHz	1	1	1
		x4	4 MHz		0	0
		x8	-	-	-	-
4.5 to 5.5V	Internal RC 1%	off	1 MHz @ 5V	0	1	1
		x4	-	-	-	-
		x8	8 MHz @ 5V	0	0	1
	External clock or resonator (depending on OPT5:4 selection)	off	0 to 8 MHz	1	1	1
		x4	-	-	-	-
		x8	8 MHz	1	0	1

DEVICE CONFIGURATION AND ORDERING INFORMATION (Cont'd)

15.2 DEVICE ORDERING INFORMATION

Table 26. Supported Part Numbers

Part Number	Program Memory (Bytes)	RAM (Bytes)	Data EEPROM (Bytes)	Temperature Range	Package
ST7FL15F1MAE	4K FLASH	256	-	-40°C to +85°C	SO20
ST7FL15F1MCE				-40°C to +125°C	
ST7FL19F1MAE			128	-40°C to +85°C	
ST7FL19F1MCE				-40°C to +125°C	
ST7PL15F1MAE	4K FASTROM		-	-40°C to +85°C	
ST7PL15F1MCE				-40°C to +125°C	
ST7PL19F1MAE			128	-40°C to +85°C	
ST7PL19F1MCE				-40°C to +125°C	
ST7L15F1MAE	4K ROM		-	-40°C to +85°C	
ST7L15F1MCE				-40°C to +125°C	
ST7L19F1MAE			128	-40°C to +85°C	
ST7L19F1MCE				-40°C to +125°C	

ST7L1 FASTROM AND ROM MICROCONTROLLER OPTION LIST
(Last update: August 2006)

Customer . . . .
Address . . . .
. . . .
Contact . . . .
Phone No . . . .
Reference FASTROM Code*: . . . .
*FASTROM code name is assigned by STMicroelectronics.
FASTROM code must be sent in .S19 format. .Hex extension cannot be processed.

Device Type/Memory Size/Package (check only one option):

FASTROM DEVICE:	4K
SO20:	[ ] ST7PL15 [ ] ST7PL19

ROM DEVICE:	4K
SO20:	[ ] ST7L15 [ ] ST7L19

Conditioning (check only one option):
[ ] Tape and Reel [ ] Tube

Special Marking: [ ] No [ ] Yes "_ _ _ _ _ _ _ _ " (8 char. max.)
Authorized characters are letters, digits, '.', '-', '/' and spaces only.

Temperature range: [ ] -40°C to +85°C [ ] -40°C to +125°C
AWUCK Selection [ ] 32 kHz Oscillator [ ] AWU RC Oscillator
Clock Source Selection: [ ] Resonator:
- [ ] LP: Low power resonator (1 to 2 MHz)
- [ ] MP: Medium power resonator (2 to 4 MHz)
- [ ] MS: Medium speed resonator (4 to 8 MHz)
- [ ] HS: High speed resonator (8 to 16 MHz)

[ ] External Clock
- [ ] on PB4
- [ ] on OSC1

[ ] Internal RC Oscillator

Sector 0 size: [ ] 0.5K [ ] 1K [ ] 2K [ ] 4K
Readout Protection: [ ] Disabled [ ] Enabled
FLASH Write Protection [ ] Disabled [ ] Enabled
PLL [ ] Disabled [ ] Enabled
LVD Reset [ ] Disabled [ ] Enabled (Highest voltage threshold)

Watchdog Selection: [ ] Software Activation [ ] Hardware Activation
Watchdog Reset on Halt: [ ] Disabled [ ] Enabled

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

Important note: Not all configurations are available. See section 15.1 on page 126 for authorized option byte combinations.

Please download the latest version of this option list from:
http://www.st.com/mcu > **downloads** > **ST7 microcontrollers** > **Option list**

DEVICE CONFIGURATION AND ORDERING INFORMATION (Cont'd)

15.3 DEVELOPMENT TOOLS

Development tools for the ST7 microcontrollers include a complete range of hardware systems and software tools from STMicroelectronics and third-party tool suppliers. The range of tools includes solutions to help you evaluate microcontroller peripherals, develop and debug your application, and program your microcontrollers.

15.3.1 Evaluation Tools and Starter Kits

ST offers complete, affordable **starter kits** and full-featured **evaluation boards** that allow you to evaluate microcontroller features and quickly start developing ST7 applications. Starter kits are complete, affordable hardware/software tool packages that include features and samples to help you quickly start developing your application. ST evaluation boards are open-design, embedded systems, which are developed and documented to serve as references for your application design. They include sample application software to help you demonstrate, learn about and implement your ST7's features.

15.3.2 Development and Debugging Tools

Application development for ST7 is supported by fully optimizing **C Compilers** and the **ST7 Assembler-Linker** toolchain, which are all seamlessly integrated in the ST7 integrated development environments in order to facilitate the debugging and fine-tuning of your application. The Cosmic C Compiler is available in a free version that outputs up to 16 Kbytes of code.

The range of hardware tools includes full-featured **ST7-EMU3 series emulators,** cost effective **ST7-DVP3 series emulators** and the low-cost **RLink** in-circuit debugger/programmer. These tools are supported by the **ST7 Toolset** from STMicroelectronics, which includes the STVD7 integrated development environment (IDE) with high-level language debugger, editor, project manager and integrated programming interface.

15.3.3 Programming Tools

During the development cycle, the **ST7-DVP3** and **ST7-EMU3 series emulators** and the **RLink** provide in-circuit programming capability for programming the Flash microcontroller on your application board.

ST also provides dedicated a low-cost dedicated in-circuit programmer, the **ST7-STICK**, as well as **ST7 Socket Boards** which provide all the sockets required for programming any of the devices in a specific ST7 sub-family on a platform that can be used with any tool with in-circuit programming capability for ST7.

For production programming of ST7 devices, ST's third-party tool partners also provide a complete range of gang and automated programming solutions, which are ready to integrate into your production environment.

15.3.4 Order Codes for Development and Programming Tools

Table 27 below lists the ordering codes for the ST7L1 development and programming tools. For additional ordering codes for spare parts and accessories, refer to the online product selector at www.st.com/mcu.

Table 27. ST7L1 Development and Programming Tools

Supported Products	In-circuit Debugger, RLink Series[1]	Emulator		Programming Tool	
	Starter Kit without Demo Board	DVP Series	EMU Series	In-circuit Programmer	ST Socket Boards and EPBs
ST7FL15	STX-RLINK[2)6)]	ST7MDT10-DVP3[4)]	ST7MDT10-EMU3	ST7-STICK[3)5)] STX-RLINK[6)]	ST7SB10-123[3)]
ST7FL19					

Notes:

1. Available from ST or from Raisonance, www.raisonance.com
2. USB connection to PC
3. Add suffix /EU, /UK or /US for the power supply for your region
4. Includes connection kit for DIP16/SO16 only. See "How to order an EMU or DVP" in ST product and tool selection guide for connection kit ordering information
5. Parallel port connection to PC
6. RLink with ST7 tool set

15.4 ST7 APPLICATION NOTES

Table 28. ST7 Application Notes

IDENTIFICATION	DESCRIPTION
Application Examples	
AN1658	Serial Numbering Implementation
AN1720	Managing the Read-out Protection in Flash Microcontrollers
AN1755	A High Resolution/precision Thermometer Using ST7 and NE555
AN1756	Choosing a DALI Implementation Strategy with ST7DALI
AN1812	A High Precision, Low Cost, Single Supply ADC for Positive and Negative Input Voltages
Example Drivers	
AN 969	SCI Communication Between ST7 and PC
AN 970	SPI Communication Between ST7 and EEPROM
AN 971	I²C Communication 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 (Sinusoïd)
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 Peripherals Registers
AN1083	ST72141 BLDC Motor Control Software and Flowchart Example
AN1105	ST7 pCAN Peripheral Driver
AN1129	PWM Management for BLDC Motor Drives Using the ST72141
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	Emulated 16-bit Slave SPI
AN1475	Developing an ST7265X Mass Storage Application
AN1504	Starting a PWM Signal Directly at High Level Using the ST7 16-bit Timer
AN1602	16-bit Timing Operations Using ST7262 or ST7263B ST7 USB MCUs
AN1633	Device Firmware Upgrade (DFU) Implementation in ST7 Non-USB Applications
AN1712	Generating a High Resolution Sinewave Using ST7 PWMART
AN1713	SMBus Slave Driver for ST7 I2C Peripherals
AN1753	Software UART Using 12-bit ART
AN1947	ST7MC PMAC Sine Wave Motor Control Software Library

Table 28. ST7 Application Notes

IDENTIFICATION	DESCRIPTION
General Purpose	
AN1476	Low Cost Power Supply for Home Appliances
AN1526	ST7FLITE0 Quick Reference Note
AN1709	EMC Design for ST Microcontrollers
AN1752	ST72324 Quick Reference Note
Product Evaluation	
AN 910	Performance Benchmarking
AN 990	ST7 Benefits vs Industry Standard
AN1077	Overview of Enhanced CAN Controllers for ST7 and ST9 MCUs
AN1086	U435 Can-Do Solutions for Car Multiplexing
AN1103	Improved B-EMF detection for Low Speed, Low Voltage with ST72141
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 Applications to ST72F264
AN1604	How to Use ST7MDT1-TRAIN with ST72F264
AN2200	Guidelines for Migrating ST7LITE1x Applications to ST7FLITE1xB
Product Optimization	
AN 982	Using ST7 with Ceramic Resonator
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
AN1181	Electrostatic Discharge Sensitive Measurement
AN1324	Calibrating the RC Oscillator of the ST7FLITE0 MCU Using the Mains
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
AN1605	Using an Active RC to Wakeup the ST7LITE0 from Power Saving Mode
AN1636	Understanding and Minimizing ADC Conversion Errors
AN1828	PIR (Passive Infrared) Detector Using the ST7FLITE05/09/SUPERLITE
AN1946	Sensorless BLDC Motor Control and BEMF Sampling Methods with ST7MC
AN1953	PFC for ST7MC Starter Kit
AN1971	ST7LITE0 Microcontrolled Ballast
Programming and Tools	
AN 978	ST7 Visual DeVELOP Software Key Debugging Features
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

Table 28. ST7 Application Notes

IDENTIFICATION	DESCRIPTION
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 an ST72324 Target Application
AN1477	Emulated Data EEPROM with Xflash Memory
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
AN1576	In-application Programming (IAP) Drivers for ST7 HDFLASH or XFLASH MCUs
AN1577	Device Firmware Upgrade (DFU) Implementation for ST7 USB Applications
AN1601	Software Implementation for ST7DALI-EVAL
AN1603	Using the ST7 USB Device Firmware Upgrade Development Kit (DFU-DK)
AN1635	ST7 Customer ROM Code Release Information
AN1754	Data Logging Program for Testing ST7 Applications via ICC
AN1796	Field Updates for FLASH Based ST7 Applications Using a PC Comm Port
AN1900	Hardware Implementation for ST7DALI-EVAL
AN1904	ST7MC Three-phase AC Induction Motor Control Software Library
AN1905	ST7MC Three-phase BLDC Motor Control Software Library
System Optimization	
AN1711	Software Techniques for Compensating ST7 ADC Errors
AN1827	Implementation of SIGMA-DELTA ADC with ST7FLITE05/09
AN2009	PWM Management for 3-Phase BLDC Motor Drives Using the ST7FMC
AN2030	Back EMF Detection During PWM On Time by ST7MC

16 REVISION HISTORY

Table 29. Revision History

Date	Revision	Main changes
16-Aug-2006	1	Initial release

Please Read Carefully:

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com