THE DATASHEET OF

ST72F561R9TATR

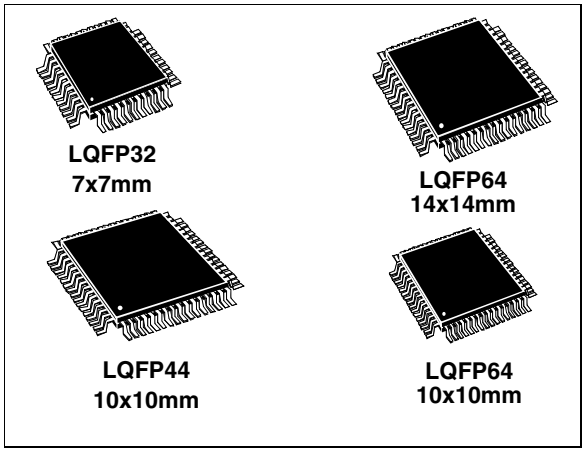
www.win-source.net 0086-755-83957316

ST72561

8-bit MCU with Flash or ROM, 10-bit ADC, 5 timers, SPI, LINSCI™, active CAN

Features

- **Memories**
  - 16K to 60K High Density Flash (HDFlash) or ROM with read-out protection capability. In-Application Programming and In-Circuit Programming for HDFlash devices
  - 1 to 2K RAM
  - HDFlash endurance: 100 cycles, data retention 40 years at 85°C
- **Clock, Reset and Supply Management**
  - Low power crystal/ceramic resonator oscillators and bypass for external clock
  - PLL for 2x frequency multiplication
  - 5 power saving modes: Halt, Auto Wake Up From Halt, Active Halt, Wait and Slow
- **Interrupt Management**
  - Nested interrupt controller
  - 14 interrupt vectors plus TRAP and RESET
  - TLI top level interrupt (on 64-pin devices)
  - Up to 21 external interrupt lines (on 4 vectors)
- **Up to 48 I/O Ports**
  - Up to 48 multifunctional bidirectional I/O lines
  - Up to 36 alternate function lines
  - Up to 6 high sink outputs
- **5 Timers**
  - 16-bit timer with 2 input captures, 2 output compares, external clock input, PWM and pulse generator modes
  - 8-bit timer with 1 or 2 input captures, 1 or 2 output compares, PWM and pulse generator modes
  - 8-bit PWM auto-reload timer with 1 or 2 input captures, 2 or 4 independent PWM output channels, output compare and time base interrupt, external clock with event detector



  - Main clock controller with real-time base and clock output
  - Window watchdog timer
- **Up to 4 Communications Interfaces**
  - SPI synchronous serial interface
  - Master/slave LINSCI™ asynchronous serial interface
  - Master-only LINSCI™ asynchronous serial interface
  - CAN 2.0B active
- **Analog Peripheral (Low Current Coupling)**
  - 10-bit A/D converter with up to 16 inputs
  - Up to 9 robust ports (low current coupling)
- **Instruction Set**
  - 8-bit data manipulation
  - 63 basic instructions
  - 17 main addressing modes
  - 8 x 8 unsigned multiply instruction
- **Development Tools**
  - Full hardware/software development package

Table 1. Device Summary

Features	ST72561AR9/ST72561R9/ ST72561J9/ST72561K9	ST72561AR7	ST72561AR6/ST72561R6/ ST72561J6/ST72561K6	ST72561AR4/ST72561J4/ ST72561K4
Program memory - bytes	60K	48K	32K	16K
RAM (stack) - bytes	2K (256)	2K (256)	1.5K (256)	1K (256)
Operating Supply	4.5V to 5.5V			
CPU Frequency	External Resonator Osc. w/ PLLx2/8 MHz			
Max. Temp. Range	-40°C to +125°C			
Packages	LQFP64 10x10mm (AR), LQFP64 14x14mm (R) LQFP44 10x10mm (J), LQFP32 7x7mm (K)			

Rev. 7

Table of Contents

1 DESCRIPTION 4
2 PIN DESCRIPTION 5
3 REGISTER AND MEMORY MAP 11
4 FLASH PROGRAM MEMORY 15
4.1 INTRODUCTION 15
4.2 MAIN FEATURES 15
4.3 STRUCTURE 15
4.4 ICC INTERFACE 16
4.5 ICP (IN-CIRCUIT PROGRAMMING) 17
4.6 IAP (IN-APPLICATION PROGRAMMING) 17
4.7 RELATED DOCUMENTATION 17
4.8 REGISTER DESCRIPTION 17
5 CENTRAL PROCESSING UNIT 18
5.1 INTRODUCTION 18
5.2 MAIN FEATURES 18
5.3 CPU REGISTERS 18
6 SUPPLY, RESET AND CLOCK MANAGEMENT 21
6.1 PHASE LOCKED LOOP 21
6.2 MULTI-OSCILLATOR (MO) 22
6.3 RESET SEQUENCE MANAGER (RSM) 23
6.4 SYSTEM INTEGRITY MANAGEMENT (SI) 25
7 INTERRUPTS 29
7.1 INTRODUCTION 29
7.2 MASKING AND PROCESSING FLOW 29
7.3 INTERRUPTS AND LOW POWER MODES 31
7.4 CONCURRENT & NESTED MANAGEMENT 31
7.5 INTERRUPT REGISTER DESCRIPTION 32
7.6 EXTERNAL INTERRUPTS 35
8 POWER SAVING MODES 38
8.1 INTRODUCTION 38
8.2 SLOW MODE 38
8.3 WAIT MODE 39
8.4 HALT MODE 40
8.5 ACTIVE HALT MODE 41
8.6 AUTO WAKE-UP FROM HALT MODE 43
9 I/O PORTS 46
9.1 INTRODUCTION 46
9.2 FUNCTIONAL DESCRIPTION 46
9.3 I/O PORT IMPLEMENTATION 49
9.4 LOW POWER MODES 49
9.5 INTERRUPTS 49
9.6 I/O PORT REGISTER CONFIGURATIONS 50

Table of Contents

10 ON-CHIP PERIPHERALS 53
10.1 WINDOW WATCHDOG (WWDG) 53
10.2 MAIN CLOCK CONTROLLER WITH REAL TIME CLOCK MCC/RTC 59
10.3 PWM AUTO-RELOAD TIMER (ART) 62
10.4 16-BIT TIMER 72
10.5 8-BIT TIMER (TIM8) 91
10.6 SERIAL PERIPHERAL INTERFACE (SPI) 109
10.7 LINSCI SERIAL COMMUNICATION INTERFACE (LIN MASTER/SLAVE) 121
10.8 LINSCI SERIAL COMMUNICATION INTERFACE (LIN MASTER ONLY) 152
10.9 BECAN CONTROLLER (BECAN) 169
10.1010-BIT A/D CONVERTER (ADC) 208
11 INSTRUCTION SET 212
11.1 CPU ADDRESSING MODES 212
11.2 INSTRUCTION GROUPS 215
12 ELECTRICAL CHARACTERISTICS 218
12.1 PARAMETER CONDITIONS 218
12.2 ABSOLUTE MAXIMUM RATINGS 219
12.3 OPERATING CONDITIONS 220
12.4 SUPPLY CURRENT CHARACTERISTICS 222
12.5 CLOCK AND TIMING CHARACTERISTICS 225
12.6 AUTO WAKEUP FROM HALT OSCILLATOR (AWU) 228
12.7 MEMORY CHARACTERISTICS 229
12.8 EMC CHARACTERISTICS 230
12.9 I/O PORT PIN CHARACTERISTICS 233
12.10CONTROL PIN CHARACTERISTICS 238
12.11TIMER PERIPHERAL CHARACTERISTICS 241
12.12COMMUNICATION INTERFACE CHARACTERISTICS 242
12.1310-BIT ADC CHARACTERISTICS 244
13 PACKAGE CHARACTERISTICS 248
13.1 PACKAGE MECHANICAL DATA 248
13.2 THERMAL CHARACTERISTICS 250
13.3 SOLDERING AND GLUEABILITY INFORMATION 250
14 DEVICE CONFIGURATION AND ORDERING INFORMATION 251
14.1 FLASH OPTION BYTES 251
14.2 TRANSFER OF CUSTOMER CODE 253
15 DEVELOPMENT TOOLS 256
16 IMPORTANT NOTES 257
16.1 ALL DEVICES 257
16.2 FLASH/FASTROM DEVICES ONLY 260
16.3 ROM DEVICES ONLY 262
17 REVISION HISTORY 263

1 DESCRIPTION

The ST72561 devices are members of the ST7 microcontroller family designed for mid-range applications with CAN (Controller Area Network) and LIN (Local Interconnect Network) interface.

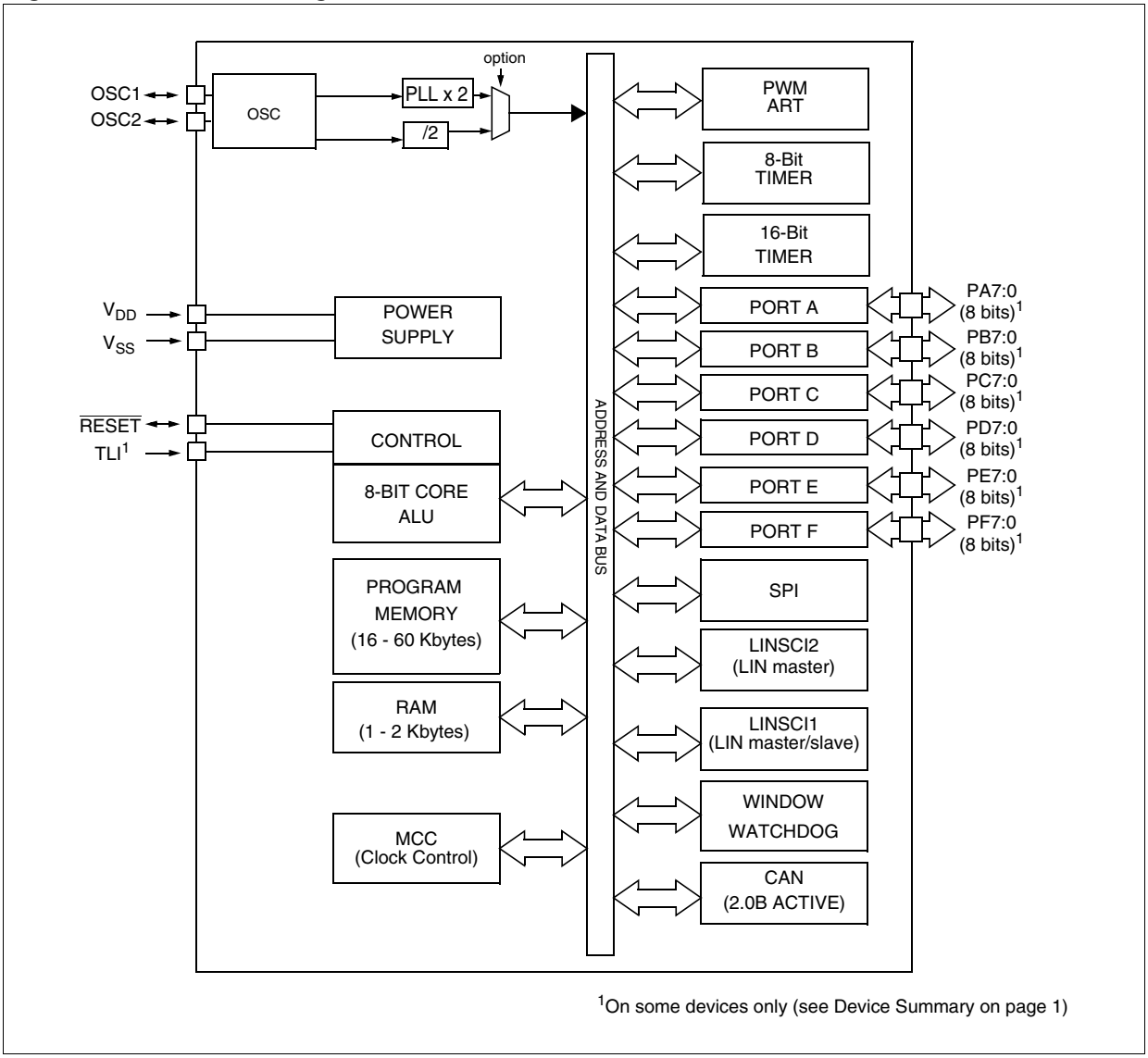
All devices are based on a common industry-standard 8-bit core, featuring an enhanced instruction set and are available with Flash or ROM program memory. The ST7 family architecture offers both power and flexibility to software developers, enabling the design of highly efficient and compact application code.

The on-chip peripherals include an A/D converter, a PWM Autoreload timer, 2 general purpose timers, 2 asynchronous serial interfaces, and an SPI interface.

For power economy, microcontroller can switch dynamically into WAIT, SLOW, Active-Halt, Auto Wake-up from HALT (AWU) or HALT mode when the application is in idle or stand-by state.

Typical applications are consumer, home, office and industrial products.

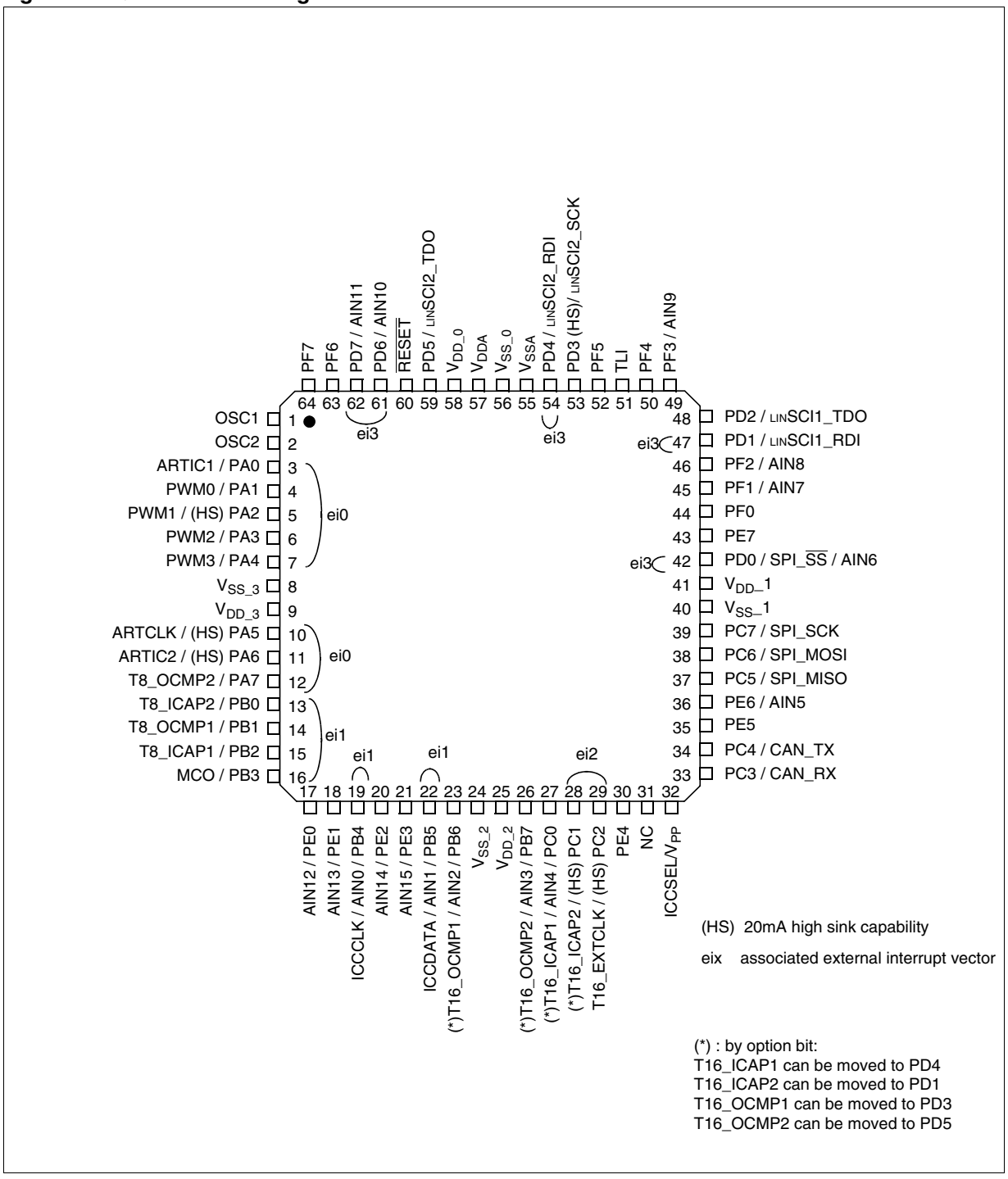
Figure 1. Device Block Diagram



1On some devices only (see Device Summary on page 1)

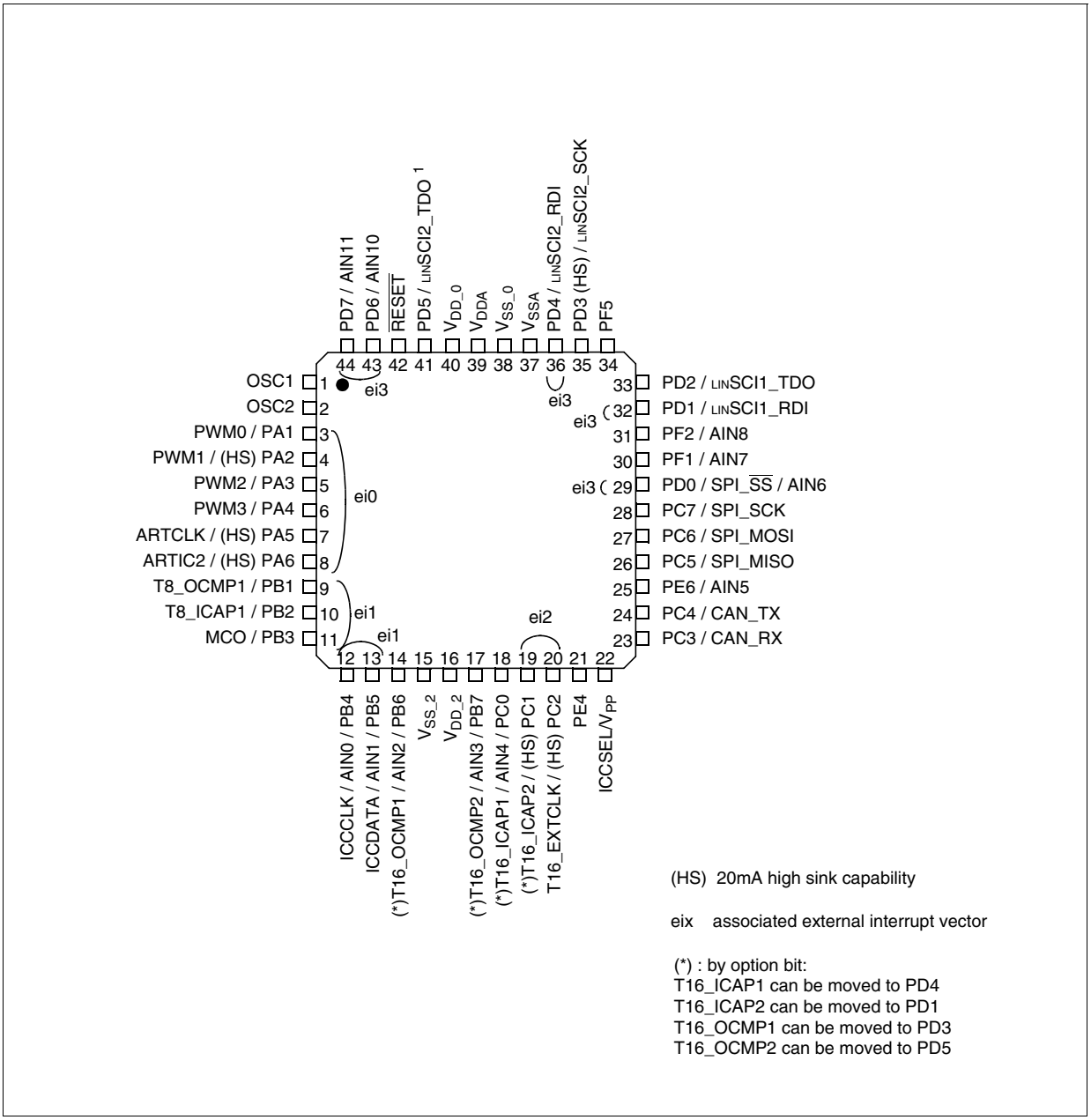
2 PIN DESCRIPTION

Figure 2. LQFP 64-Pin Package Pinout



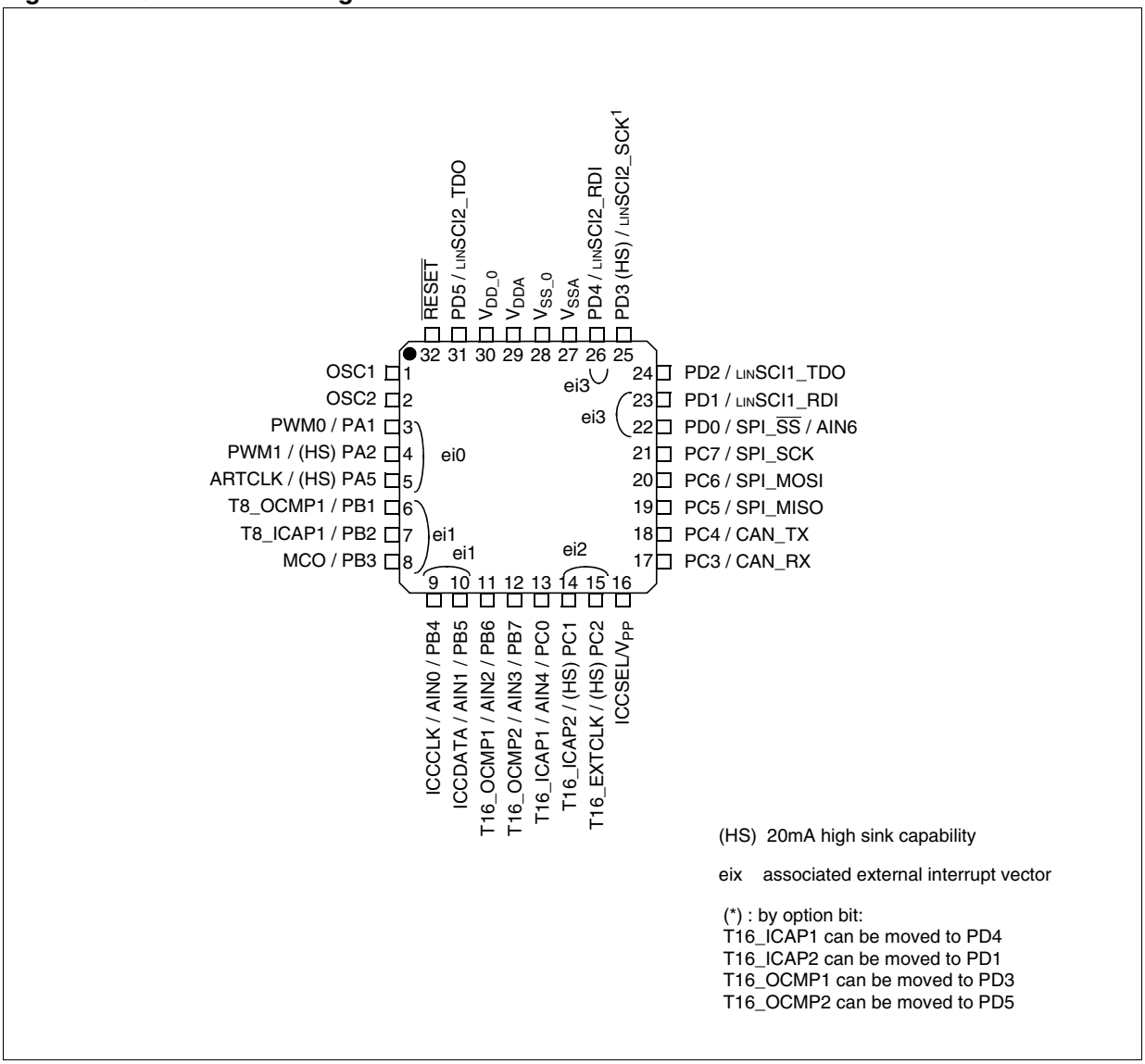
PIN DESCRIPTION (Cont'd)

Figure 3. LQFP 44-Pin Package Pinout



PIN DESCRIPTION (Cont'd)

Figure 4. LQFP 32-Pin Package Pinout



For external pin connection guidelines, refer to "ELECTRICAL CHARACTERISTICS" on page 219.

PIN DESCRIPTION (Cont'd)

For external pin connection guidelines, refer to "ELECTRICAL CHARACTERISTICS" on page 219.

Legend / Abbreviations for Table 2:

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

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

T_T= TTL 0.8V / 2V with Schmitt trigger

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

Port and control configuration:

– Input: float = floating, wpu = weak pull-up, int = interrupt[1], ana = analog, RB = robust

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

Refer to "I/O PORTS" on page 46 for more details on the software configuration of the I/O ports.

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

Table 2. Device Pin Description

Pin n°			Pin Name	Type	Level		Port						Main function (after reset)	Alternate function	
							Input				Output				
LQFP64	LQFP44	LQFP32			Input	Output	float	wpu	int	ana	OD	PP			
1	1	1	OSC1[3]	I									External clock input or Resonator oscillator inverter input		
2	2	2	OSC2[3]	I/O									Resonator oscillator inverter output		
3	-	-	PA0 / ARTIC1	I/O	C_T		**X**	ei0			X	X	Port A0	ART Input Capture 1	
4	3	3	PA1 / PWM0	I/O	C_T		**X**		ei0		X	X	Port A1	ART PWM Output 0	
5	4	4	PA2 (HS) / PWM1	I/O	C_T	HS	**X**	ei0			X	X	Port A2	ART PWM Output 1	
6	5	-	PA3 / PWM2	I/O	C_T		**X**		ei0		X	X	Port A3	ART PWM Output 2	
7	6	-	PA4 / PWM3	I/O	C_T		**X**	ei0			X	X	Port A4	ART PWM Output 3	
8	-	-	V_{SS_3}	S									Digital Ground Voltage		
9	-	-	V_{DD_3}	S									Digital Main Supply Voltage		
10	7	5	PA5 (HS) / ARTCLK	I/O	C_T	HS	**X**		ei0		X	X	Port A5	ART External Clock	
11	8	-	PA6 (HS) / ARTIC2	I/O	C_T	HS	**X**	ei0			X	X	Port A6	ART Input Capture 2	
12	-	-	PA7 / T8_OCMP2	I/O	C_T		**X**		ei0		X	X	Port A7	TIM8 Output Compare 2	
13	-	-	PB0 /T8_ICAP2	I/O	C_T		**X**	ei1			X	X	Port B0	TIM8 Input Capture 2	
14	9	6	PB1 /T8_OCMP1	I/O	C_T		**X**		ei1		X	X	Port B1	TIM8 Output Compare 1	
15	10	7	PB2 / T8_ICAP1	I/O	C_T		**X**	ei1			X	X	Port B2	TIM8 Input Capture 1	
16	11	8	PB3 / MCO	I/O	C_T		**X**		ei1		X	X	Port B3	Main clock out (f_{OSC2})	
17	-	-	PE0 / AIN12	I/O	T_T		**X**	X		RB	X	X	Port E0	ADC Analog Input 12	
18	-	-	PE1 / AIN13	I/O	T_T		**X**	X		RB	X	X	Port E1	ADC Analog Input 13	
19	12	9	PB4 / AIN0 / ICCCLK	I/O	C_T		**X**	ei1		RB	X	X	Port B4	ICC Clock input	ADC Analog Input 0
20	-	-	PE2 / AIN14	I/O	T_T		**X**	X		RB	X	X	Port E2	ADC Analog Input 14	
21	-	-	PE3 / AIN15	I/O	T_T		**X**	X		RB	X	X	Port E3	ADC Analog Input 15	
22	13	10	PB5 / AIN1 / ICCDATA	I/O	C_T		**X**		ei1	RB	X	X	Port B5	ICC Data input	ADC Analog Input 1

Pin n°			Pin Name	Type	Level		Port						Main function (after reset)	Alternate function	
							Input				Output				
LQFP64	LQFP44	LQFP32			Input	Output	float	wpu	int	ana	OD	PP			
23	14	11	PB6 / AIN2 / T16_OCMP1	I/O	C_T		**X**	X		RB	X	X	Port B6	TIM16 Output Compare 1	ADC Analog Input 2
24	15	-	V_{SS_2}	S									Digital Ground Voltage		
25	16	-	V_{DD_2}	S									Digital Main Supply Voltage		
26	17	12	PB7 /AIN3 / T16_OCMP2	I/O	C_T		**X**	X		RB	X	X	Port B7	TIM16 Output Compare 2	ADC Analog Input 3
27	18	13	PC0 / AIN4 / T16_ICAP1	I/O	C_T		**X**	X		RB	X	X	Port C0	TIM16 Input Capture 1	ADC Analog Input 4
28	19	14	PC1 (HS) / T16_ICAP2	I/O	C_T	HS	**X**	ei2			X	X	Port C1	TIM16 Input Capture 2	
29	20	15	PC2 (HS) / T16_EXTCLK	I/O	C_T	HS	**X**		ei2		X	X	Port C2	TIM16 External Clock input	
30	21	-	PE4	I/O	T_T		**X**	X			X	X	Port E4		
31	-	-	NC	Not Connected											
32	22	16	V_{PP}	I									Flash programming voltage. Must be tied low in user mode.		
33	23	17	PC3 / CANRX	I/O	C_T		**X**	X			X	X	Port C3	CAN Receive Data Input	
34	24	18	PC4 / CANTX	I/O	C_T		**X**					X[2]	Port C4	CAN Transmit Data Output	
35	-	-	PE5	I/O	T_T		**X**	X			X	X	Port E5		
36	25	-	PE6 / AIN5	I/O	T_T		**X**	X		X	X	X	Port E6	ADC Analog Input 5	
37	26	19	PC5 /MISO	I/O	C_T		**X**	X			X	X	Port C5	SPI Master In/Slave Out	
38	27	20	PC6 / MOSI	I/O	C_T		**X**	X			X	X	Port C6	SPI Master Out/Slave In	
39	28	21	PC7 /SCK	I/O	C_T		**X**	X			X	X	Port C7	SPI Serial Clock	
40	-	-	V_{SS_1}	S									Digital Ground Voltage		
41	-	-	V_{DD_1}	S									Digital Main Supply Voltage		
42	29	22	PD0 / $\overline{SS}$/ AIN6	I/O	C_T		**X**	ei3		X	X	X	Port D0	SPI Slave Select	ADC Analog Input 6
43	-	-	PE7	I/O	T_T		**X**	X			X	X	Port E7		
44	-	-	PF0	I/O	T_T		**X**	X			X	X	Port F0		
45	30	-	PF1 / AIN7	I/O	T_T		**X**	X		X	X	X	Port F1	ADC Analog Input 7	
46	31	-	PF2 / AIN8	I/O	T_T		**X**	X		X	X	X	Port F2	ADC Analog Input 8	
47	32	23	PD1 / SCI1_RDI	I/O	C_T		**X**		ei3		X	X	Port D1	LINSCI1 Receive Data input	
48	33	24	PD2 / SCI1_TDO	I/O	C_T		**X**	X			X	X	Port D2	LINSCI1 Transmit Data output	
49	-	-	PF3 / AIN9	I/O	T_T		**X**	X		X	X	X	Port F3	ADC Analog Input 9	
50	-	-	PF4	I/O	T_T		**X**	X			X	X	Port F4		
51	-	-	TLI	I	C_T		**X**		X				Top level interrupt input pin		
52	34	-	PF5	I/O	T_T		**X**	X			X	X	Port F5		
53	35	25	PD3 (HS) / SCI2_SCK	I/O	C_T	HS	**X**	X			X	X	Port D3	LINSCI2 Serial Clock Output	

Pin n°			Pin Name	Type	Level		Port						Main function (after reset)	Alternate function
							Input				Output			
LQFP64	LQFP44	LQFP32			Input	Output	float	wpu	int	ana	OD	PP		
54	36	26	PD4 / SCI2_RDI	I/O	C_T		**X**		ei3		X	X	Port D4	LINSCI2 Receive Data input
55	37	27	V_{SSA}	S									Analog Ground Voltage	
56	38	28	V_{SS_0}	S									Digital Ground Voltage	
57	39	29	V_{DDA}	I									Analog Reference Voltage for ADC	
58	40	30	V_{DD_0}	S									Digital Main Supply Voltage	
59	41	31	PD5 / SCI2_TDO	I/O	C_T		**X**	X			X	X	Port D5	LINSCI2 Transmit Data output
60	42	32	$\overline{\text{RESET}}$	I/O	C_T								Top priority non maskable interrupt.	
61	43	-	PD6 / AIN10	I/O	C_T		**X**	ei3		X	X	X	Port D6	ADC Analog Input 10
62	44	-	PD7 / AIN11	I/O	C_T		**X**		ei3	X	X	X	Port D7	ADC Analog Input 11
63	-	-	PF6	I/O	T_T		**X**	X			X	X	Port F6	
64	-	-	PF7	I/O	T_T		**X**	X			X	X	Port F7	

Notes:

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

2. Input mode can be used for general purpose I/O, output mode only for CANTX.

3. OSC1 and OSC2 pins connect a crystal/ceramic resonator, or an external source to the on-chip oscillator; see Section 6 and Section 12.5 "CLOCK AND TIMING CHARACTERISTICS" for more details.

4. On the chip, each I/O port has eight pads. Pads that are not bonded to external pins are in input pull-up configuration after reset. The configuration of these pads must be kept at reset state to avoid added current consumption.

3 REGISTER AND MEMORY MAP

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

The available memory locations consist of 128 bytes of register locations, up to 2 Kbytes of RAM and up to 60 Kbytes of user program memory.

The RAM space includes up to 256 bytes for the stack from 0100h to 01FFh.The highest address bytes contain the user reset and interrupt vectors.

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

Figure 5. Memory Map

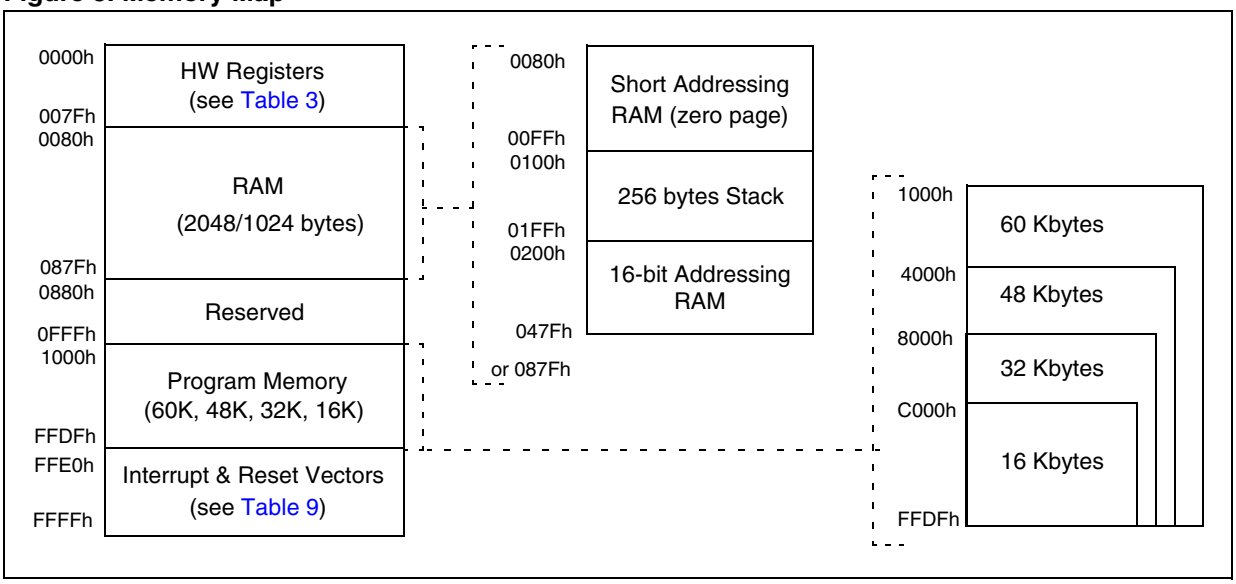


Table 3. Hardware Register Map

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	00h[1] 00h 00h	R/W[2] R/W[2] R/W[2]
0003h 0004h 0005h	Port B	PBDR PBDDR PBOR	Port B Data Register Port B Data Direction Register Port B Option Register	00h[1] 00h 00h	R/W[2] R/W[2] R/W[2]
0006h 0007h 0008h	Port C	PCDR PCDDR PCOR	Port C Data Register Port C Data Direction Register Port C Option Register	00h[1] 00h 00h	R/W[2] R/W[2] R/W[2]
0009h 000Ah 000Bh	Port D	PDDR PDDDR PDOR	Port D Data Register Port D Data Direction Register Port D Option Register	00h[1] 00h 00h	R/W[2] R/W[2] R/W[2]
000Ch 000Dh 000Eh	Port E	PEDR PEDDR PEOR	Port E Data Register Port E Data Direction Register Port E Option Register	00h[1] 00h 00h	R/W[2] R/W[2] R/W[2]

Address	Block	Register Label	Register Name	Reset Status	Remarks
000Fh 0010h 0011h	Port F	PFDR PFDDR PFOR	Port F Data Register Port F Data Direction Register Port F Option Register	00h[1] 00h 00h	R/W[2] R/W[2] R/W[2]
0012h to 0020h	Reserved Area (15 bytes)				
0021h 0022h 0023h	SPI	SPIDR SPICR SPICSR	SPI Data I/O Register SPI Control Register SPI Control/Status Register	xxh 0xh 00h	R/W R/W R/W
0024h	FLASH	FCSR	Flash Control/Status Register	00h	R/W
0025h 0026h 0027h 0028h 0029h 002Ah	ITC	ISPR0 ISPR1 ISPR2 ISPR3 EICR0 EICR1	Interrupt Software Priority Register 0 Interrupt Software Priority Register 1 Interrupt Software Priority Register 2 Interrupt Software Priority Register 3 External Interrupt Control Register 0 External Interrupt Control Register 1	FFh FFh FFh FFh 00h 00h	R/W R/W R/W R/W R/W R/W
002Bh 002Ch	AWU	AWUCSR AWUPR	Auto Wake up f. Halt Control/Status Register Auto Wake Up From Halt Prescaler	00h FFh	R/W R/W
002Dh 002Eh	CKCTRL	SICSR MCCSR	System Integrity Control / Status Register Main Clock Control / Status Register	0xh 00h	R/W R/W
002Fh 0030h	WWDG	WDGCR WDGWR	Watchdog Control Register Watchdog Window Register	7Fh 7Fh	R/W R/W
0031h 0032h 0033h 0034h 0035h 0036h 0037h 0038h 0039h 003Ah 003Bh	PWM ART	PWMDCR3 PWMDCR2 PWMDCR1 PWMDCR0 PWMCR ARTCSR ARTCAR ARTARR ARTICCSR ARTICR1 ARTICR2	Pulse Width Modulator Duty Cycle Register 3 PWM Duty Cycle Register 2 PWM Duty Cycle Register 1 PWM Duty Cycle Register 0 PWM Control register Auto-Reload Timer Control/Status Register Auto-Reload Timer Counter Access Register Auto-Reload Timer Auto-Reload Register ART Input Capture Control/Status Register ART Input Capture Register 1 ART Input Capture register 2	00h 00h 00h 00h 00h 00h 00h 00h 00h 00h 00h	R/W R/W R/W R/W R/W R/W R/W R/W R/W Read Only Read Only
003Ch 003Dh 003Eh 003Fh 0040h 0041h 0042h 0043h 0044h	8-BIT TIMER	T8CR2 T8CR1 T8CSR T8IC1R T8OC1R T8CTR T8ACTR T8IC2R T8OC2R	Timer Control Register 2 Timer Control Register 1 Timer Control/Status Register Timer Input Capture 1 Register Timer Output Compare 1 Register Timer Counter Register Timer Alternate Counter Register Timer Input Capture 2 Register Timer Output Compare 2 Register	00h 00h 00h xxh 00h FCh FCh xxh 00h	R/W R/W Read Only Read Only R/W Read Only Read Only Read Only R/W
0045h 0046h 0047h	ADC	ADCCSR ADCDRH ADCDRL	Control/Status Register Data High Register Data Low Register	00h 00h 00h	R/W Read Only Read Only

Address	Block	Register Label	Register Name	Reset Status	Remarks
0048h	LINSCI1 (LIN Master/ Slave)	SCI1ISR	SCI1 Status Register	C0h	Read Only
0049h		SCI1DR	SCI1 Data Register	xxh	R/W
004Ah		SCI1BRR	SCI1 Baud Rate Register	00h	R/W
004Bh		SCI1CR1	SCI1 Control Register 1	xxh	R/W
004Ch		SCI1CR2	SCI1 Control Register 2	00h	R/W
004Dh		SCI1CR3	SCI1Control Register 3	00h	R/W
004Eh		SCI1ERPR	SCI1 Extended Receive Prescaler Register	00h	R/W
004Fh		SCI1ETPR	SCI1 Extended Transmit Prescaler Register	00h	R/W
0050h	Reserved Area (1 byte)				
0051h	16-BIT TIMER	T16CR2	Timer Control Register 2	00h	R/W
0052h		T16CR1	Timer Control Register 1	00h	R/W
0053h		T16CSR	Timer Control/Status Register	00h	R/W
0054h		T16IC1HR	Timer Input Capture 1 High Register	xxh	Read Only
0055h		T16IC1LR	Timer Input Capture 1 Low Register	xxh	Read Only
0056h		T16OC1HR	Timer Output Compare 1 High Register	80h	R/W
0057h		T16OC1LR	Timer Output Compare 1 Low Register	00h	R/W
0058h		T16CHR	Timer Counter High Register	FFh	Read Only
0059h		T16CLR	Timer Counter Low Register	FCh	Read Only
005Ah		T16ACHR	Timer Alternate Counter High Register	FFh	Read Only
005Bh		T16ACLR	Timer Alternate Counter Low Register	FCh	Read Only
005Ch		T16IC2HR	Timer Input Capture 2 High Register	xxh	Read Only
005Dh		T16IC2LR	Timer Input Capture 2 Low Register	xxh	Read Only
005Eh		T16OC2HR	Timer Output Compare 2 High Register	80h	R/W
005Fh		T16OC2LR	Timer Output Compare 2 Low Register	00h	R/W
0060h	LINSCI2 (LIN Master)	SCI2SR	SCI2 Status Register	C0h	Read Only
0061h		SCI2DR	SCI2 Data Register	xxh	R/W
0062h		SCI2BRR	SCI2 Baud Rate Register	00h	R/W
0063h		SCI2CR1	SCI2 Control Register 1	xxh	R/W
0064h		SCI2CR2	SCI2 Control Register 2	00h	R/W
0065h		SCI2CR3	SCI2 Control Register 3	00h	R/W
0066h		SCI2ERPR	SCI2 Extended Receive Prescaler Register	00h	R/W
0067h		SCI2ETPR	SCI2 Extended Transmit Prescaler Register	00h	R/W

Address	Block	Register Label	Register Name	Reset Status	Remarks
0068h	Active CAN	CMCR	CAN Master Control Register		R/W
0069h		CMSR	CAN Master Status Register		R/W
006Ah		CTSR	CAN Transmit Status Register		R/W
006Bh		CTPR	CAN Transmit Priority Register		R/W
006Ch		CRFR	CAN Receive FIFO Register		R/W
006Dh		CIER	CAN Interrupt Enable Register		R/W
006Eh		CDGR	CAN Diagnosis Register		R/W
006Fh		CPSR	CAN Page Selection Register		R/W
0070h		PAGES	PAGE REGISTER 0		R/W
0071h			PAGE REGISTER 1		R/W
0072h			PAGE REGISTER 2		R/W
0073h			PAGE REGISTER 3		R/W
0074h			PAGE REGISTER 4		R/W
0075h			PAGE REGISTER 5		R/W
0076h			PAGE REGISTER 6		R/W
0077h			PAGE REGISTER 7		R/W
0078h			PAGE REGISTER 8		R/W
0079h			PAGE REGISTER 9		R/W
007Ah			PAGE REGISTER 10		R/W
007Bh			PAGE REGISTER 11		R/W
007Ch			PAGE REGISTER 12		R/W
007Dh			PAGE REGISTER 13		R/W
007Eh			PAGE REGISTER 14		R/W
007Fh			PAGE REGISTER 15		R/W

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

Notes:

1. The contents of the I/O port DR registers are readable only in output configuration. In input configuration, the values of the I/O pins are returned instead of the DR register contents.
2. The bits associated with unavailable pins must always keep their reset value.

4 FLASH PROGRAM MEMORY

4.1 INTRODUCTION

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

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

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

4.2 MAIN FEATURES

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

4.3 STRUCTURE

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

Depending on the overall Flash memory size in the microcontroller device, there are up to three user sectors (see Table 3). Each of these sectors can be erased independently to avoid unnecessary erasing of the whole Flash memory when only a partial erasing is required.

The first two sectors have a fixed size of 4 Kbytes (see Figure 6). They are mapped in the upper part of the ST7 addressing space so the reset and interrupt vectors are located in Sector 0 (F000h-FFFFh).

Table 4. Sectors available in Flash devices

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

4.3.1 Read-out Protection

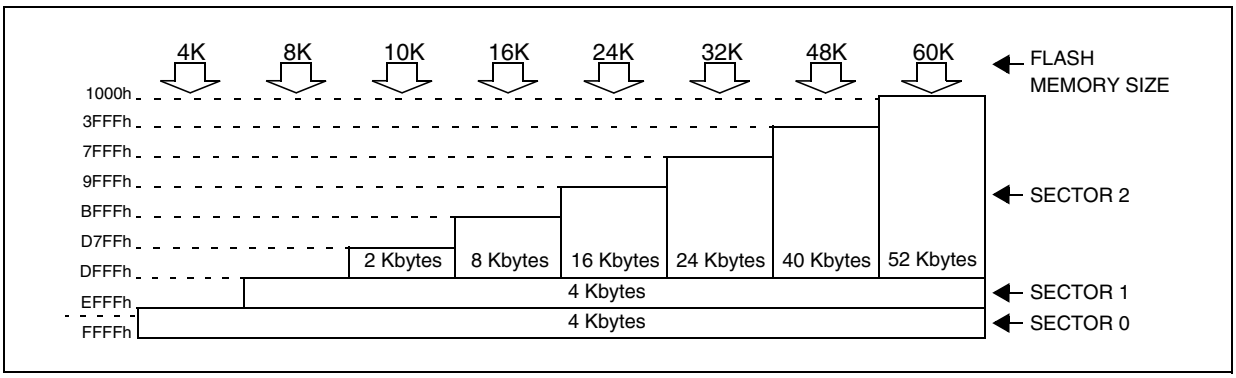
Read-out protection, when selected, provides a protection against Program Memory content extraction and against write access to Flash memory. Even if no protection can be considered as totally unbreakable, the feature provides a very high level of protection for a general purpose microcontroller.

In Flash devices, this protection is removed by reprogramming the option. In this case, the entire program memory is first automatically erased and the device can be reprogrammed.

Read-out protection selection depends on the device type:

- In Flash devices it is enabled and removed through the FMP_R bit in the option byte.
- In ROM devices it is enabled by mask option specified in the Option List.

Figure 6. Memory Map and Sector Address



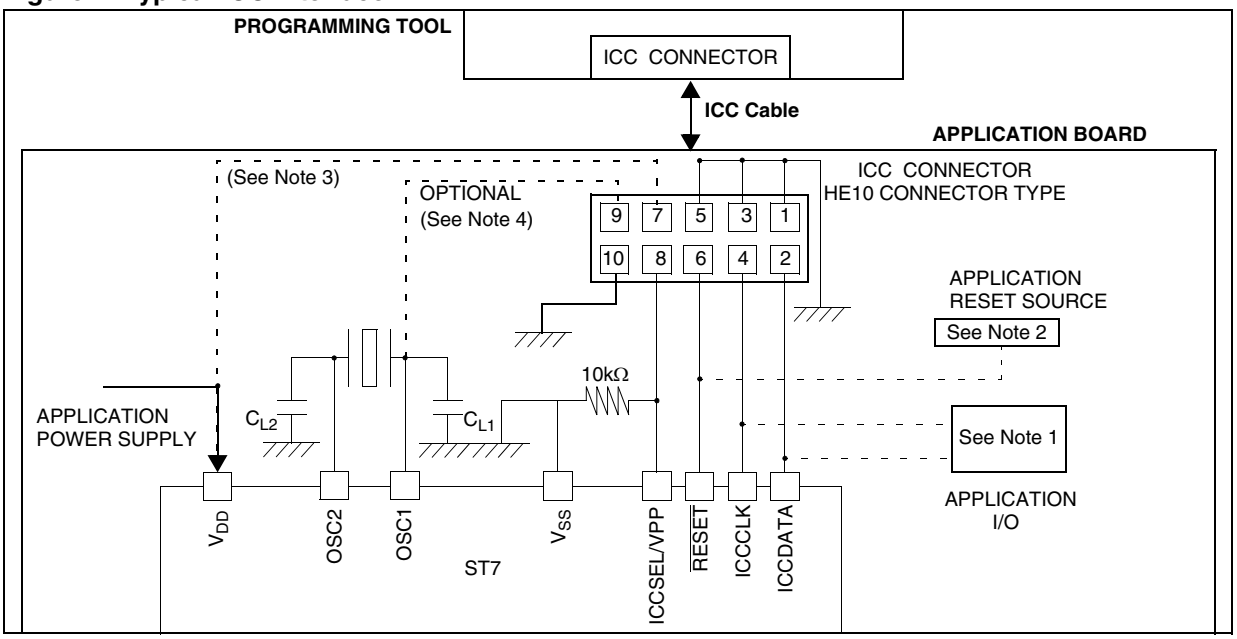
FLASH PROGRAM MEMORY (Cont'd)

4.4 ICC INTERFACE

ICC (In-Circuit Communication) needs a minimum of four and up to six pins to be connected to the programming tool (see Figure 7). These pins are:

- $\overline{\text{RESET}}$: device reset
- V_{SS}: device power supply ground
- ICCCLK: ICC output serial clock pin
- ICCDATA: ICC input/output serial data pin
- ICCSEL/V_{PP}: programming voltage
- OSC1(or OSCIN): main clock input for external source (optional)
- V_{DD}: application board power supply (see Figure 7, Note 3)

Figure 7. 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 has to implemented in case another device forces the signal. Refer to the Programming Tool documentation for recommended resistor values.

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

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

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

FLASH PROGRAM MEMORY (Cont'd)

4.5 ICP (IN-CIRCUIT PROGRAMMING)

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

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

When using an STMicroelectronics or third-party programming tool that supports ICP and the specific microcontroller device, the user needs only to implement the ICP hardware interface on the application board (see Figure 7). For more details on the pin locations, refer to the device pinout description.

4.6 IAP (IN-APPLICATION PROGRAMMING)

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

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

4.7 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*.

4.8 REGISTER DESCRIPTION

FLASH CONTROL/STATUS REGISTER (FCSR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	0	0

This register is reserved for use by Programming Tool software. It controls the Flash programming and erasing operations.

Table 5. Flash Control/Status Register Address and Reset Value

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0024h	**FCSR** Reset Value	0	0	0	0	0	0	0	0

5 CENTRAL PROCESSING UNIT

5.1 INTRODUCTION

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

5.2 MAIN FEATURES

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

5.3 CPU REGISTERS

The six CPU registers shown in Figure 8 are not present in the memory mapping and are accessed by specific instructions.

Accumulator (A)

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

Index Registers (X and Y)

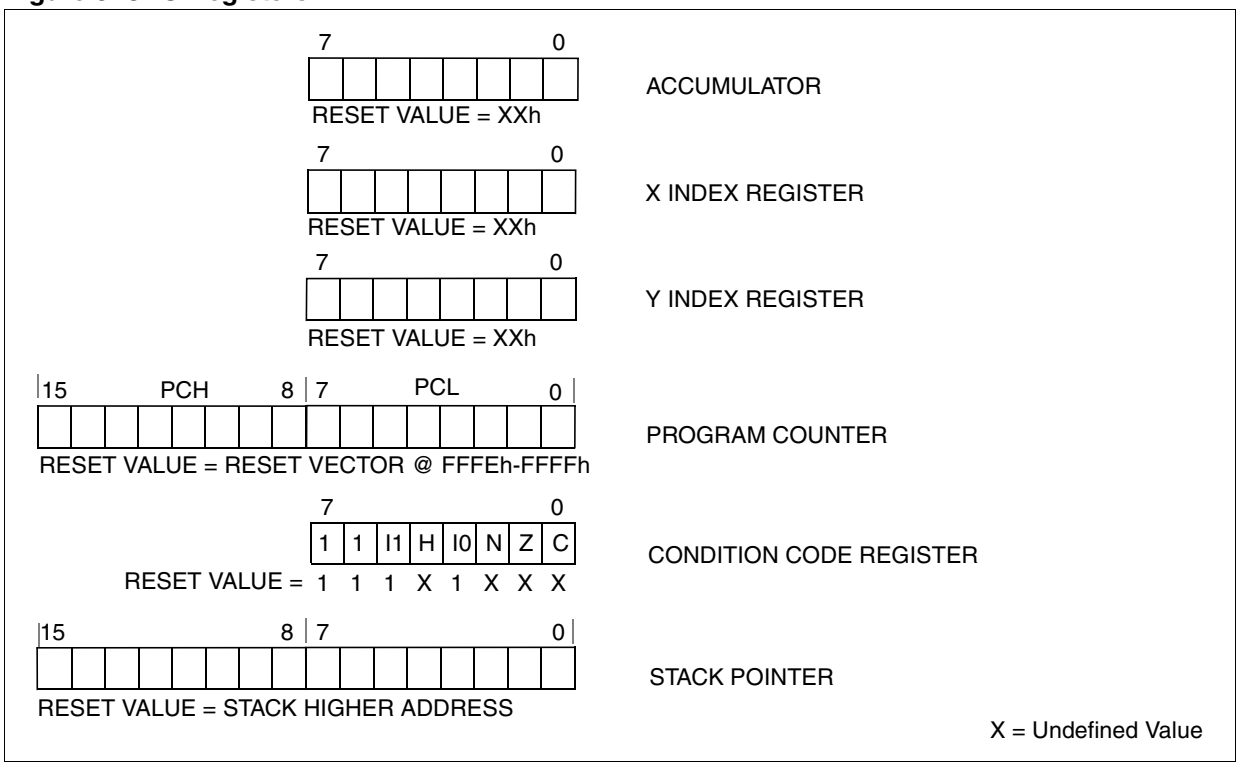
These 8-bit registers are used to create effective addresses or as temporary storage areas for data manipulation. (The Cross-Assembler generates a precede instruction (PRE) to indicate that the following instruction refers to the Y register.)

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

Program Counter (PC)

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

Figure 8. CPU Registers



CENTRAL PROCESSING UNIT (Cont'd)

Condition Code Register (CC)

Read/Write

Reset Value: 111x1xxx

7							0
1	1	I1	H	I0	N	Z	C

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

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

Arithmetic Management Bits

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

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

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

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

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

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

This bit is accessed by the JRMI and JRPL instructions.

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

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

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

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

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

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

Interrupt Management Bits

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

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

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

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

See the interrupt management chapter for more details.

CENTRAL PROCESSING UNIT (Cont'd)

Stack Pointer (SP)

Read/Write

Reset Value: 01 FFh

15							8
0	0	0	0	0	0	0	1
7							0
SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0

The Stack Pointer is a 16-bit register which is always pointing to the next free location in the stack. It is then decremented after data has been pushed onto the stack and incremented before data is popped from the stack (see Figure 9).

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

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

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

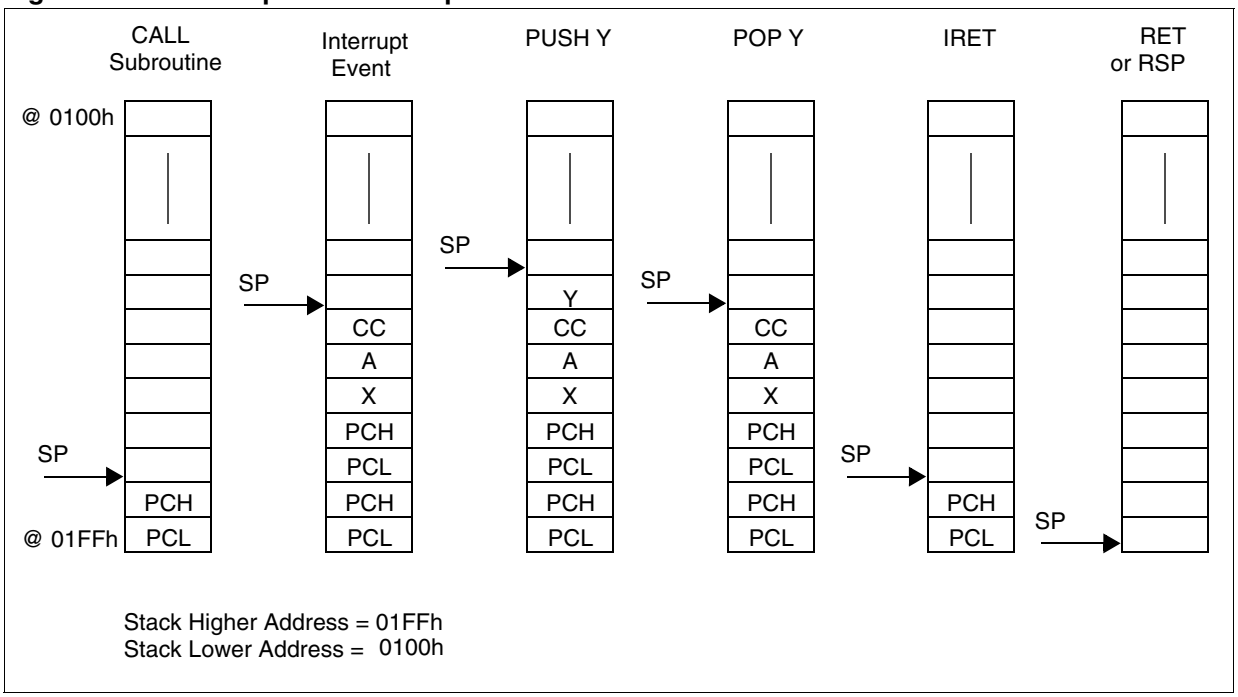
The stack is used to save the return address during a subroutine call and the CPU context during an interrupt. The user may also directly manipulate the stack by means of the PUSH and POP instructions. In the case of an interrupt, the PCL is stored at the first location pointed to by the SP. Then the other registers are stored in the next locations as shown in Figure 9.

– When an interrupt is received, the SP is decremented and the context is pushed on the stack.

– On return from interrupt, the SP is incremented and the context is popped from the stack.

A subroutine call occupies two locations and an interrupt five locations in the stack area.

Figure 9. Stack Manipulation Example



6 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. An overview is shown in Figure 11.

For more details, refer to dedicated parametric section.

Main features

- Optional PLL for multiplying the frequency by 2
- Reset Sequence Manager (RSM)
- Multi-Oscillator Clock Management (MO)
  - 4 Crystal/Ceramic resonator oscillators
- System Integrity Management (SI)
  - Main supply Low voltage detection (LVD)
  - Auxiliary Voltage detector (AVD) with interrupt capability for monitoring the main supply

6.1 PHASE LOCKED LOOP

If the clock frequency input to the PLL is in the range 2 to 4 MHz, the PLL can be used to multiply the frequency by two to obtain an f_{OSC2} of 4 to 8 MHz. The PLL is enabled by option byte. If the PLL is disabled, then $f_{OSC2} = f_{OSC}/2$.

Caution: The PLL is not recommended for applications where timing accuracy is required. See "PLL Characteristics" on page 228.

Figure 10. PLL Block Diagram

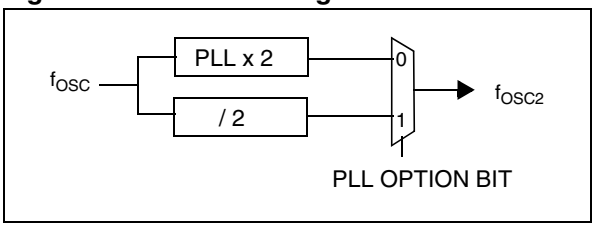
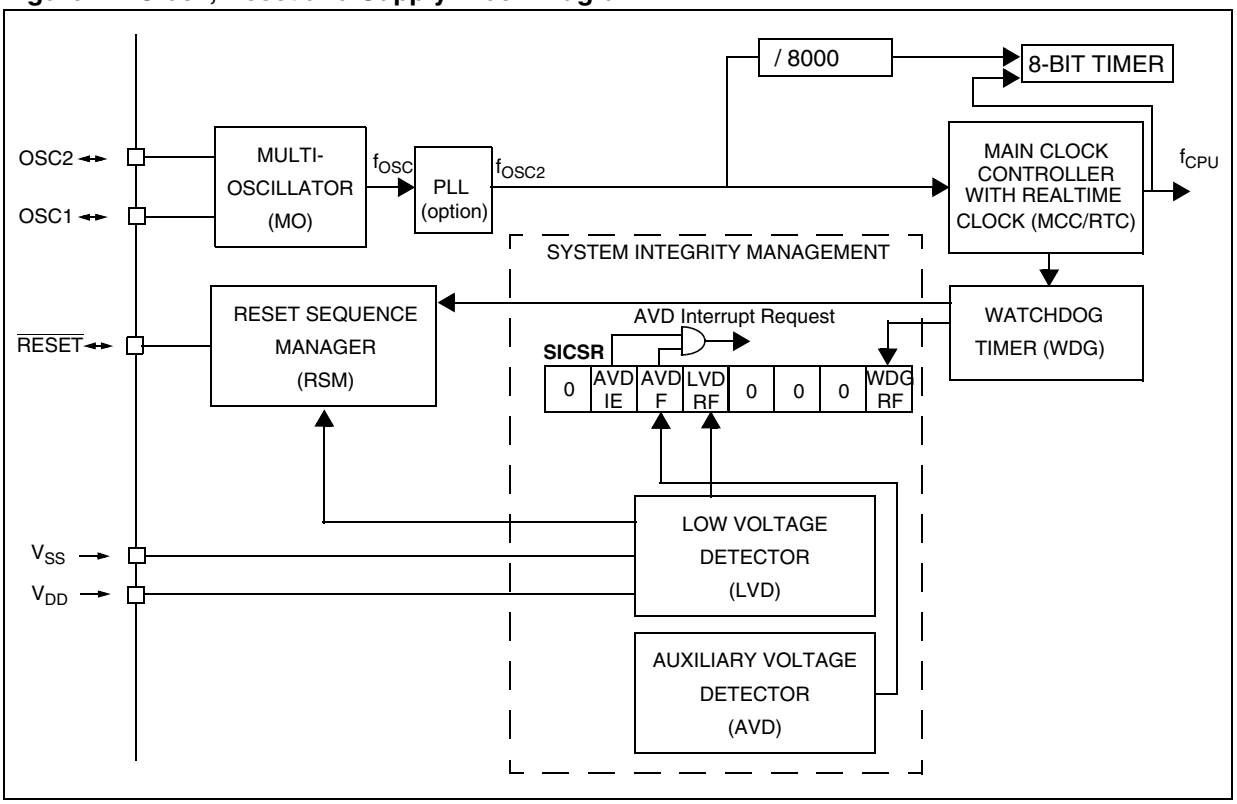


Figure 11. Clock, Reset and Supply Block Diagram



6.2 MULTI-OSCILLATOR (MO)

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

- an external source
- a crystal or ceramic resonator oscillator

Each oscillator is optimized for a given frequency range in terms of consumption and is selectable through the option byte. The associated hardware configuration are shown in Table 6. Refer to the electrical characteristics section for more details.

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

External Clock Source

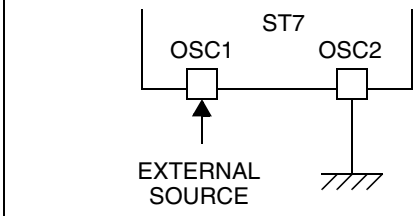
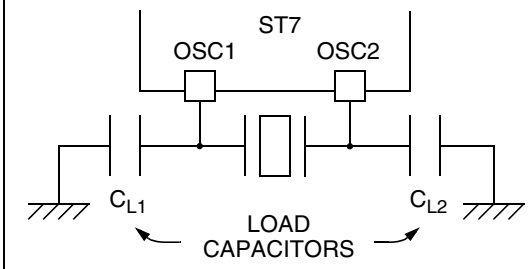
In external clock mode, a clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC1 pin while the OSC2 pin is tied to ground.

Crystal/Ceramic Oscillators

This family of oscillators has the advantage of producing a very accurate rate on the main clock of the ST7. The selection within a list of five oscillators with different frequency ranges must be done by option byte in order to reduce consumption (refer to Section 14.1 on page 252 for more details on the frequency ranges). 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. The loading capacitance values must be adjusted according to the selected oscillator.

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

Table 6. ST7 Clock Sources

	Hardware Configuration
External Clock	
Crystal/Ceramic Resonators	

6.3 RESET SEQUENCE MANAGER (RSM)

6.3.1 Introduction

The reset sequence manager includes three RESET sources as shown in Figure 2:

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

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

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

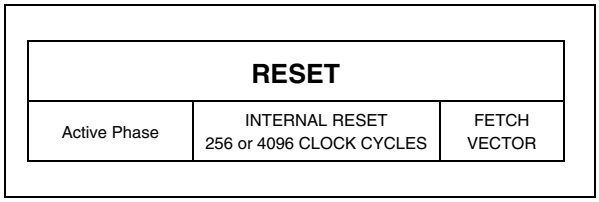
The basic RESET sequence consists of three phases as shown in Figure 1:

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

The 256 or 4096 CPU clock cycle delay allows the oscillator to stabilize and ensures that recovery has taken place from the Reset state. The shorter or longer clock cycle delay should be selected by option byte to correspond to the stabilization time of the external oscillator used in the application.

The RESET vector fetch phase duration is two clock cycles.

Figure 12. RESET Sequence Phases



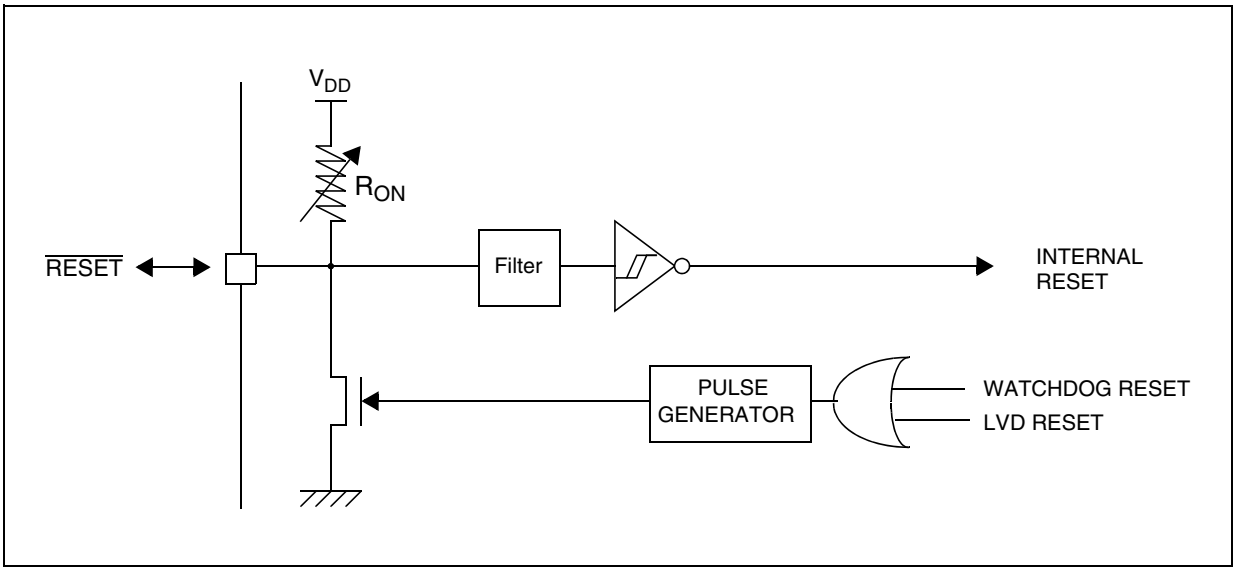
Caution: When the ST7 is unprogrammed or fully erased, the Flash is blank and the RESET vector is not programmed. For this reason, it is recommended to keep the RESET pin in low state until programming mode is entered, in order to avoid unwanted behavior.

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

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

A RESET signal originating from an external source must have a duration of at least $t_{h(RSTL)in}$ in order to be recognized (see Figure 3). This detection is asynchronous and therefore the MCU can enter reset state even in HALT mode.

Figure 13. Reset Block Diagram



RESET SEQUENCE MANAGER (Cont'd)

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

6.3.3 External Power-On RESET

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

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

6.3.4 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 $\overline{\text{RESET}}$ pin acts as an output that is pulled low when $V_{DD} < V_{IT+}$ (rising edge) or $V_{DD} < V_{IT-}$ (falling edge) as shown in Figure 3.

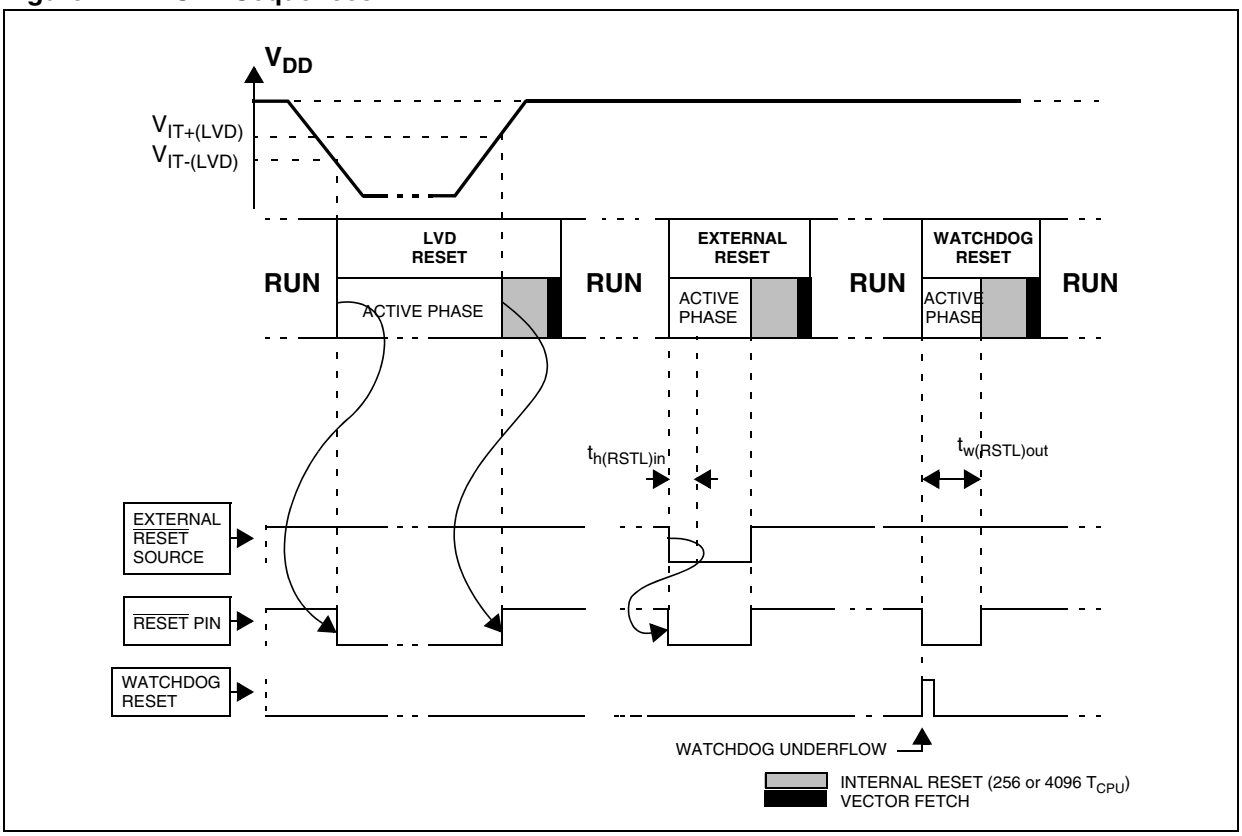
The LVD filters spikes on V_{DD} larger than $t_{g(VDD)}$ to avoid parasitic resets.

6.3.5 Internal Watchdog RESET

The RESET sequence generated by a internal Watchdog counter overflow is shown in Figure 3.

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

Figure 14. RESET Sequences



6.4 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.

6.4.1 Low Voltage Detector (LVD)

The Low Voltage Detector function (LVD) generates a static reset when the V_{DD} supply voltage is below a $V_{IT-(LVD)}$ 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 in order to avoid a parasitic reset when the MCU starts running and sinks current on the supply (hysteresis).

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

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

The LVD function is illustrated in Figure 15.

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.

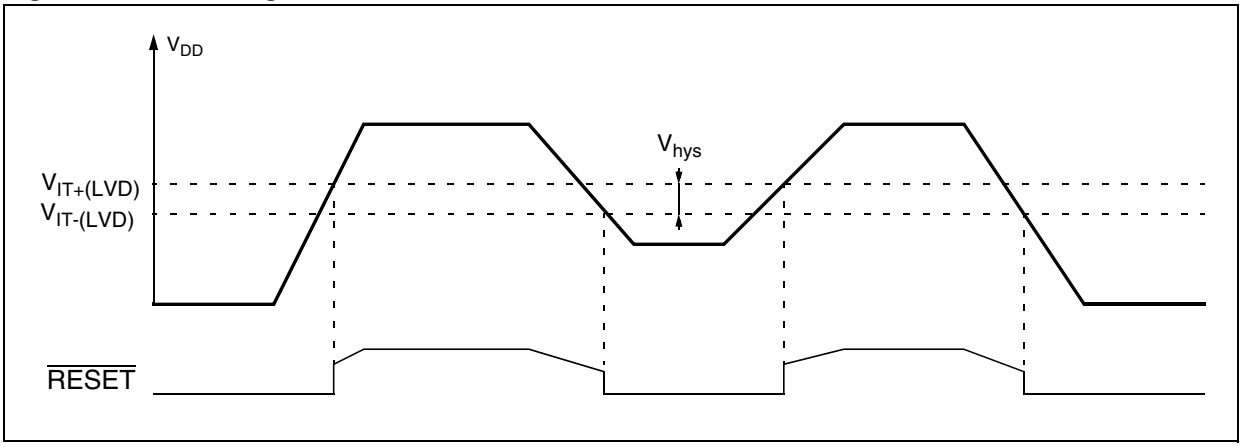
Notes:

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

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

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

Figure 15. Low Voltage Detector vs Reset



SYSTEM INTEGRITY MANAGEMENT (Cont'd)

6.4.2 Auxiliary Voltage Detector (AVD)

The Voltage Detector function (AVD) is based on an analog comparison between a $V_{IT-(AVD)}$ and $V_{IT+(AVD)}$ reference value and the V_{DD} main supply. 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 function is active only if the LVD is enabled through the option byte.

6.4.2.1 Monitoring the V_{DD} Main Supply

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

In the case of a drop in voltage, the AVD interrupt acts as an early warning, allowing software to shut down safely before the LVD resets the microcontroller. See Figure 16.

The interrupt on the rising edge is used to inform the application that the V_{DD} warning state is over.

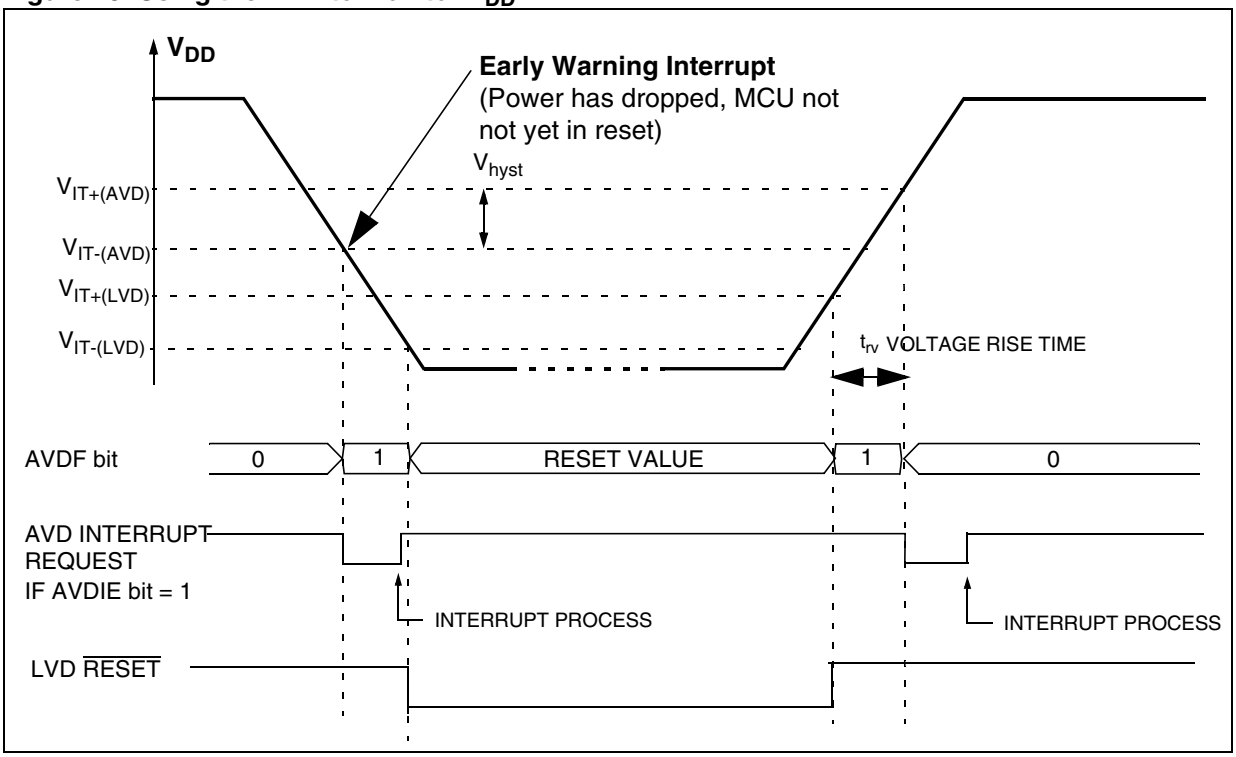
If the voltage rise time t_{rv} is less than 256 or 4096 CPU cycles (depending on the reset delay selected by option byte), no AVD interrupt will be generated when $V_{IT+(AVD)}$ is reached.

If t_{rv} is greater than 256 or 4096 cycles then:

– If the AVD interrupt is enabled before the $V_{IT+(AVD)}$ threshold is reached, then two AVD interrupts will be received: The first when the AVDIE bit is set and the second when the threshold is reached.

– If the AVD interrupt is enabled after the $V_{IT+(AVD)}$ threshold is reached, then only one AVD interrupt occurs.

Figure 16. Using the AVD to Monitor V_{DD}



SYSTEM INTEGRITY MANAGEMENT (Cont'd)

6.4.3 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.

6.4.3.1 Interrupts

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

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

SYSTEM INTEGRITY MANAGEMENT (Cont'd)

6.4.4 Register Description

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

Read/Write

Reset Value: 000x 000x (00h)

7							0
0	AVD IE	AVD F	LVD RF	0	0	0	WDG RF

Bit 7 = Reserved, must be kept cleared.

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

Bit 5 = **AVDF** *Voltage Detector flag*
This read-only bit is set and cleared by hardware. If the AVDIE bit is set, an interrupt request is generated when the AVDF bit changes value. Refer to Figure 16 and to Section 6.4.2.1 for additional details.
0: V_{DD} over $V_{IT+(AVD)}$ threshold
1: V_{DD} under $V_{IT-(AVD)}$ threshold

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

Bits 3:1 = Reserved, must be kept cleared.

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

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

Application notes

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

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

7 INTERRUPTS

7.1 INTRODUCTION

The ST7 enhanced interrupt management provides the following features:

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

This interrupt management is based on:

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

– Interrupt software priority registers (ISPRx),

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

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

7.2 MASKING AND PROCESSING FLOW

The interrupt masking is managed by the I1 and I0 bits of the CC register and the ISPRx registers which give the interrupt software priority level of each interrupt vector (see Table 6). The processing flow is shown in Figure 17.

When an interrupt request has to be serviced:

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

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

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

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

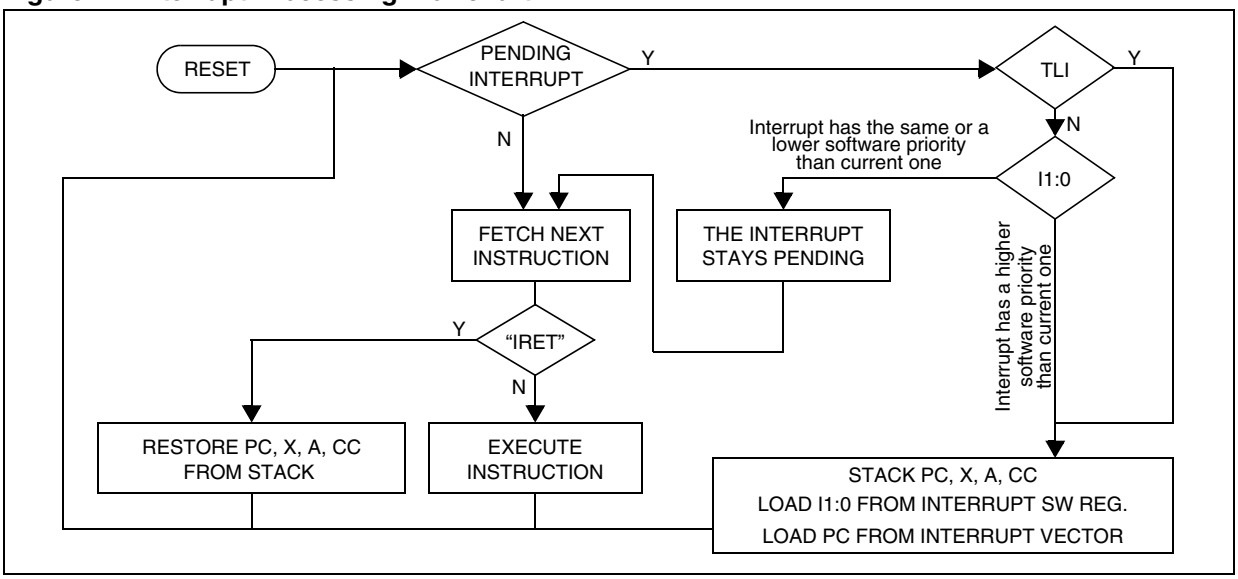
The interrupt service routine should end with the IRET instruction which causes the contents of the saved registers to be recovered from the stack.

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

Table 7. Interrupt Software Priority Levels

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

Figure 17. Interrupt Processing Flowchart



INTERRUPTS (Cont'd)

Servicing Pending Interrupts

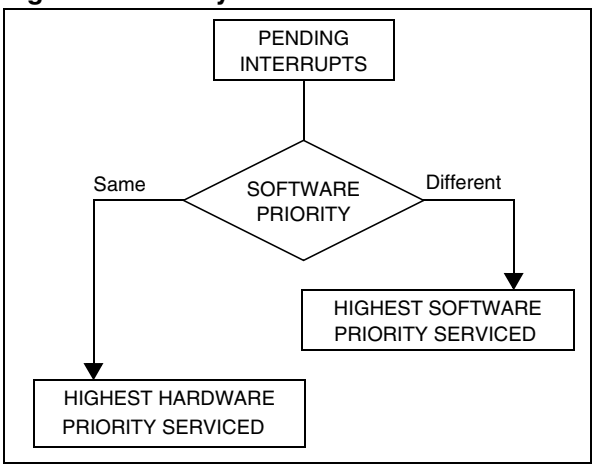
As several interrupts can be pending at the same time, the interrupt to be taken into account is determined by the following two-step process:

– the highest software priority interrupt is serviced,

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

Figure 18 describes this decision process.

Figure 18. Priority Decision Process



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

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

Different Interrupt Vector Sources

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

Non-Maskable Sources

These sources are processed regardless of the state of the I1 and I0 bits of the CC register (see Figure 17). After stacking the PC, X, A and CC registers (except for RESET), the corresponding vector is loaded in the PC register and the I1 and I0 bits of the CC are set to disable interrupts (level 3). These sources allow the processor to exit HALT mode.

- TRAP (Non Maskable Software Interrupt)

This software interrupt is serviced when the TRAP instruction is executed. It will be serviced according to the flowchart in Figure 17 as a TLI.
Caution: TRAP can be interrupted by a TLI.

- RESET

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

Maskable Sources

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

- TLI (Top Level Hardware Interrupt)

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

- External Interrupts

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

- Peripheral Interrupts

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

INTERRUPTS (Cont'd)

7.3 INTERRUPTS AND LOW POWER MODES

All interrupts allow the processor to exit the WAIT low power mode. On the contrary, only external and other specified interrupts allow the processor to exit from the HALT modes (see column "Exit from HALT" in "Interrupt Mapping" table). When several pending interrupts are present while exiting HALT mode, the first one serviced can only be an interrupt with exit from HALT mode capability and it is selected through the same decision process shown in Figure 18.

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

7.4 CONCURRENT & NESTED MANAGEMENT

The following Figure 19 and Figure 20 show two different interrupt management modes. The first is called concurrent mode and does not allow an interrupt to be interrupted, unlike the nested mode in Figure 20. The interrupt hardware priority is given in this order from the lowest to the highest: MAIN, IT4, IT3, IT2, IT1, IT0, TLI. The software priority is given for each interrupt.

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

Figure 19. Concurrent Interrupt Management

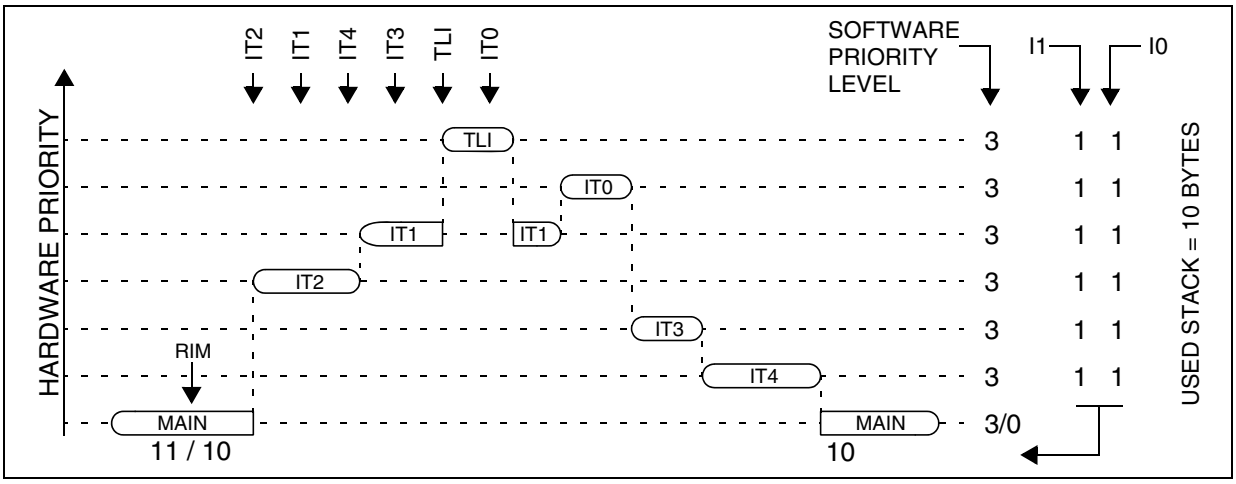
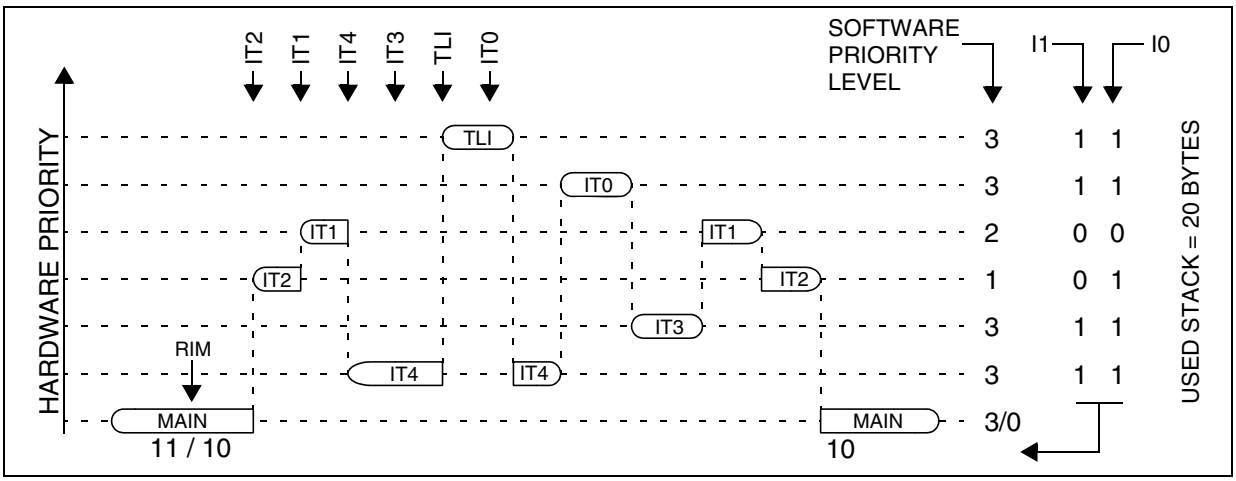


Figure 20. Nested Interrupt Management



INTERRUPTS (Cont'd)

7.5 INTERRUPT REGISTER DESCRIPTION

CPU CC REGISTER INTERRUPT BITS

Read/Write

Reset Value: 111x 1010 (xAh)

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

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

These two bits indicate the current interrupt software priority.

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

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

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

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

INTERRUPT SOFTWARE PRIORITY REGISTERS (ISPRX)

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

Reset Value: 1111 1111 (FFh)

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

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

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

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

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

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

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

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

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

INTERRUPTS (Cont'd)

Table 8. Dedicated Interrupt Instruction Set

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

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

INTERRUPTS (Cont'd)

Table 9. Interrupt Mapping

N°	Source Block	Description	Register Label	Priority Order	Exit from HALT[1)]	Address Vector
	RESET	Reset	N/A	Highest Priority	yes	FFFEh-FFFFh
	TRAP	Software interrupt			no	FFFCh-FFFDh
0	TLI	External top level interrupt	EICR		yes	FFFAh-FFFBh
1	MCC/RTC	Main clock controller time base interrupt	MCCSR		yes	FFF8h-FFF9h
2	ei0/AWUFH	External interrupt ei0/ Auto wake-up from Halt	EICR/ AWUCSR		yes[2)]	FFF6h-FFF7h
3	ei1/AVD	External interrupt ei1/Auxiliary Voltage Detector	EICR/ SICSR			FFF4h-FFF5h
4	ei2	External interrupt ei2	EICR			FFF2h-FFF3h
5	ei3	External interrupt ei3	EICR			FFF0h-FFF1h
6	CAN	CAN peripheral interrupt - RX	CIER		no	FFEEh-FFEFh
7	CAN	CAN peripheral interrupt - TX / ER / SC	CIER		yes[3)]	FFECh-FFEDh
8	SPI	SPI peripheral interrupts	SPICSR		yes	FFEAh-FFEBh
9	TIMER8	8-bit TIMER peripheral interrupts	T8_TCR1		no	FFE8h-FFE9h
10	TIMER16	16-bit TIMER peripheral interrupts	TCR1		no	FFE6h-FFE7h
11	LINSCI2	LINSCI2 Peripheral interrupts	SCI2CR1		no	FFE4h-FFE5h
12	LINSCI1	LINSCI1 Peripheral interrupts (LIN Master/ Slave)	SCI1CR1		no[4)]	FFE2h-FFE3h
13	PWM ART	8-bit PWM ART interrupts	PWMCR	Lowest Priority	yes	FFE0h-FFE1h

Notes:

1. Valid for HALT and ACTIVE HALT modes except for the MCC/RTC interrupt source which exits from ACTIVE HALT mode only.
2. Except AVD interrupt
3. Exit from Halt only when a wake-up condition is detected, generating a Status Change interrupt. See Section 10.8.6 on page 160.
4. It is possible to exit from Halt using the external interrupt which is mapped on the RDI pin.

INTERRUPTS (Cont'd)

7.6 EXTERNAL INTERRUPTS

7.6.1 I/O Port Interrupt Sensitivity

The external interrupt sensitivity is controlled by the ISxx bits in the EICR register (Figure 21). This control allows up to four fully independent external interrupt source sensitivities.

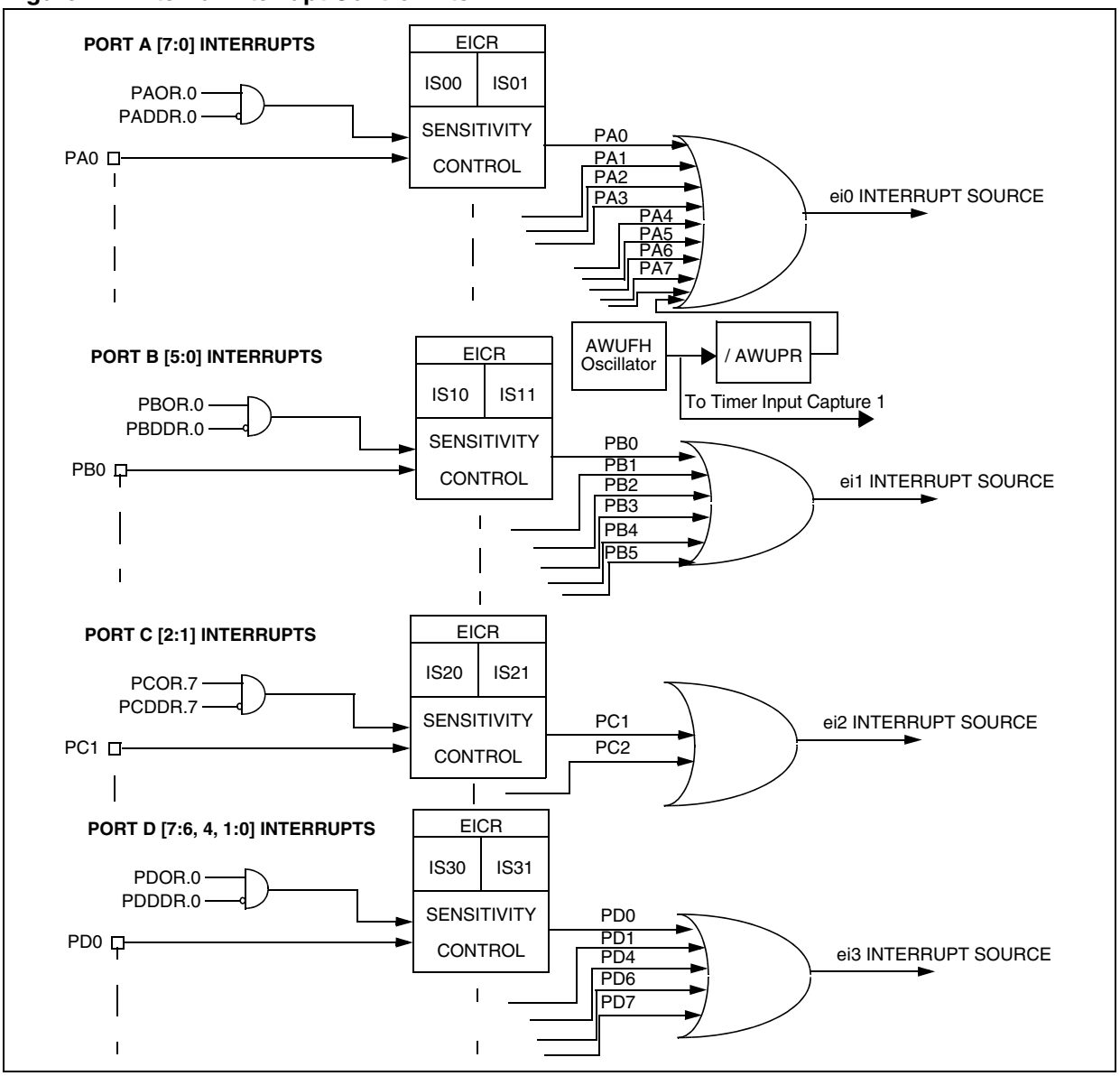
Each external interrupt source can be generated on four (or five) different events on the pin:

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

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

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

Figure 21. External Interrupt Control Bits



INTERRUPTS (Cont'd)

7.6.2 Register Description

EXTERNAL INTERRUPT CONTROL REGISTER 0 (EICR0)
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*
The interrupt sensitivity, defined using the IS3[1:0] bits, is applied to the ei3 external interrupts:

IS31	IS30	External Interrupt Sensitivity
0	0	Falling edge & low level
0	1	Rising edge only
1	0	Falling edge only
1	1	Rising and falling edge

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

Bits 5:4 = **IS2[1:0]** *ei2 sensitivity*
The interrupt sensitivity, defined using the IS2[1:0] bits, is applied to the ei2 external interrupts:

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

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

Bits 3:2 = **IS1[1:0]** *ei1 sensitivity*
The interrupt sensitivity, defined using the IS1[1:0] bits, is applied to the ei1 external interrupts:

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

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

Bits 1:0 = **IS0[1:0]** *ei0 sensitivity*
The interrupt sensitivity, defined using the IS0[1:0] bits, is applied to the ei0 external interrupts:

IS01	IS00	External Interrupt Sensitivity
0	0	Falling edge & low level
0	1	Rising edge only
1	0	Falling edge only
1	1	Rising and falling edge

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

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

7							0
0	0	0	0	0	0	TLIS	TLIE

BIts 7:2 = Reserved

Bit 1 = **TLIS** *Top Level Interrupt sensitivity*
This bit configures the TLI edge sensitivity. It can be set and cleared by software only when TLIE bit is cleared.
0: Falling edge
1: Rising edge

Bit 0 = **TLIE** *Top Level Interrupt enable*
This bit allows to enable or disable the TLI capability on the dedicated pin. It is set and cleared by software.
0: TLI disabled
1: TLI enabled

Notes:

– A parasitic interrupt can be generated when clearing the TLIE bit.

– In some packages, the TLI pin is not available. In this case, the TLIE bit must be kept low to avoid parasitic TLI interrupts.

INTERRUPTS (Cont'd)

Table 10. Nested Interrupts Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0025h		ei1		ei0		CLKM		TLI	
	ISPR0	I1_3	I0_3	I1_2	I0_2	I1_1	I0_1		
	Reset Value	1	1	1	1	1	1	1	1
0026h		CAN TX/ER/SC		CAN RX		ei3		ei2	
	ISPR1	I1_7	I0_7	I1_6	I0_6	I1_5	I0_5	I1_4	I0_4
	Reset Value	1	1	1	1	1	1	1	1
0027h		LINSCI 2		TIMER 16		TIMER 8		SPI	
	ISPR2	I1_11	I0_11	I1_10	I0_10	I1_9	I0_9	I1_8	I0_8
	Reset Value	1	1	1	1	1	1	1	1
0028h						ART		LINSCI 1	
	ISPR3					I1_13	I0_13	I1_12	I0_12
	Reset Value	1	1	1	1	1	1	1	1
0029h	**EICR0**	IS31	IS30	IS21	IS20	IS11	IS10	IS01	IS00
	Reset Value	0	0	0	0	0	0	0	0
002Ah	**EICR1**							TLIS	TLIE
	Reset Value	0	0	0	0	0	0	0	0

8 POWER SAVING MODES

8.1 INTRODUCTION

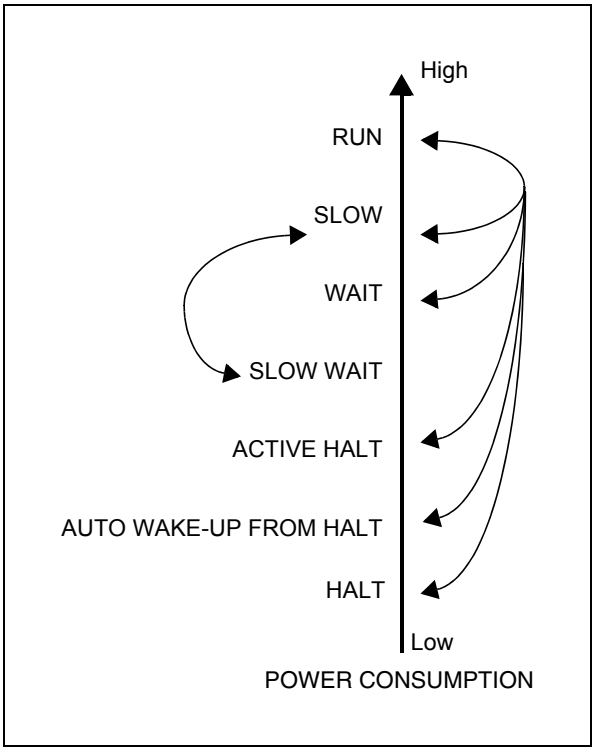
To give a large measure of flexibility to the application in terms of power consumption, five main power saving modes are implemented in the ST7 (see Figure 22):

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

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

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

Figure 22. Power Saving Mode Transitions



8.2 SLOW MODE

This mode has two targets:

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

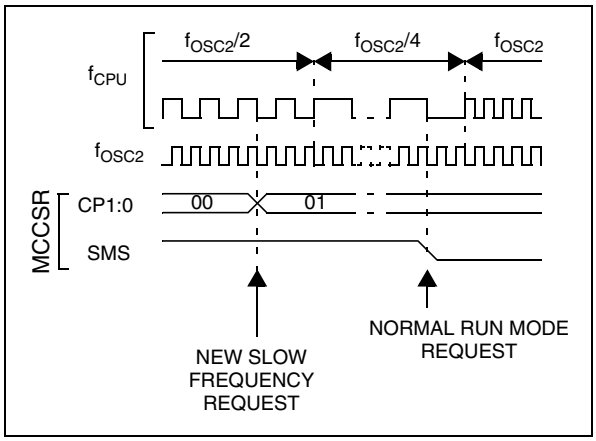
– To adapt the internal clock frequency (f_{CPU}) to the available supply voltage.

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

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

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

Figure 23. SLOW Mode Clock Transitions



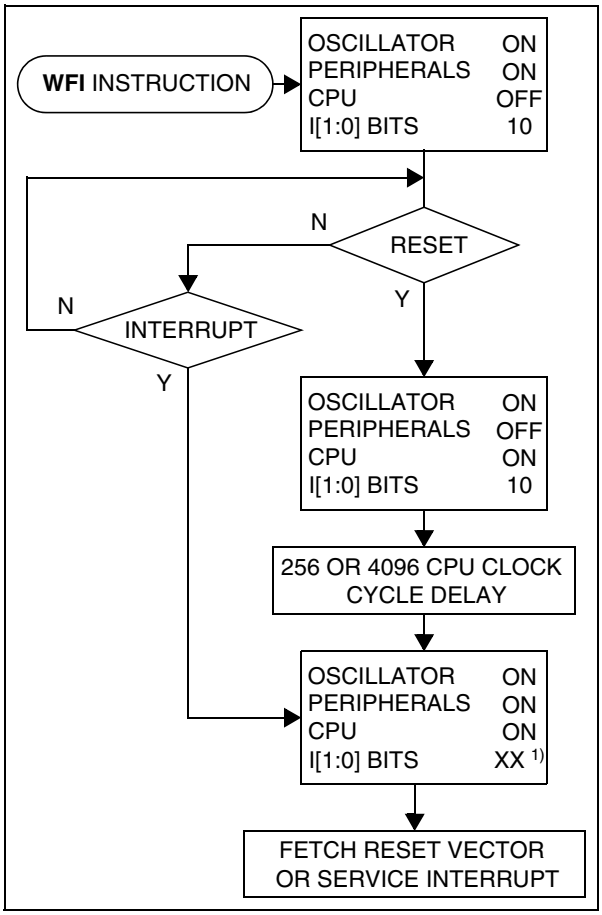
POWER SAVING MODES (Cont'd)

8.3 WAIT MODE

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

Refer to Figure 24

Figure 24. WAIT Mode Flow-chart



Note:

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

POWER SAVING MODES (Cont'd)

8.4 HALT MODE

The HALT mode is the lowest power consumption mode of the MCU. It is entered by executing the 'HALT' instruction when the OIE bit of the Main Clock Controller Status register (MCCSR) is cleared (see Section 10.2 on page 59 for more details on the MCCSR register) and when the AWUEN bit in the AWUCSR register is cleared.

The MCU can exit HALT mode on reception of either a specific interrupt (see Table 9, "Interrupt Mapping," on page 34) 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 26).

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

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

The compatibility of Watchdog operation with HALT mode is configured by the "WDGHALT" option bit of the option byte. The HALT instruction when executed while the Watchdog system is enabled, can generate a Watchdog RESET (see Section 10.1 on page 53 for more details).

Figure 25. HALT Timing Overview

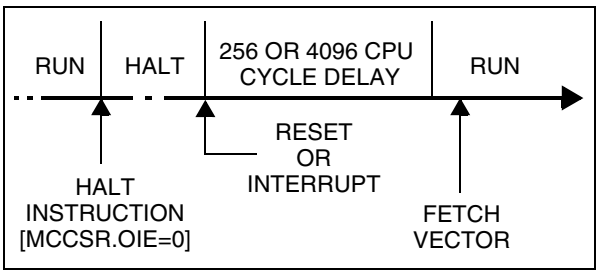
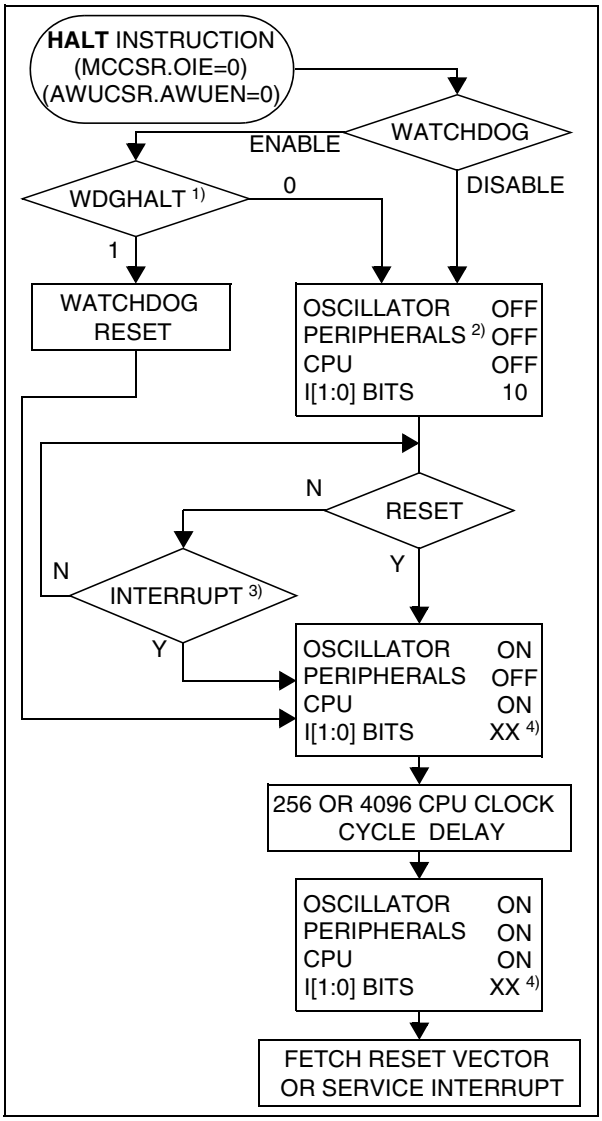


Figure 26. HALT Mode Flow-chart



Notes:

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

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

3. Only some specific interrupts can exit the MCU from HALT mode (such as external interrupt). Refer to Table 9, "Interrupt Mapping," on page 34 for more details.

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

POWER SAVING MODES (Cont'd)

Halt Mode Recommendations

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

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

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

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

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

8.5 ACTIVE HALT MODE

ACTIVE HALT mode is the lowest power consumption mode of the MCU with a real time clock available. It is entered by executing the 'HALT' instruction when MCC/RTC interrupt enable flag (OIE bit in MCCSR register) is set and when the AWUEN bit in the AWUCSR register is cleared (See "Register Description" on page 45.)

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

The MCU can exit ACTIVE HALT mode on reception of the RTC interrupt and some specific interrupts (see Table 9, "Interrupt Mapping," on page 34) or a RESET. When exiting ACTIVE HALT mode by means of a RESET a 4096 or 256 CPU cycle delay occurs (depending on the option byte). After the start up delay, the CPU resumes operation by servicing the interrupt or by fetching the reset vector which woke it up (see Figure 28).

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

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

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

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

POWER SAVING MODES (Cont'd)

Figure 27. ACTIVE HALT Timing Overview

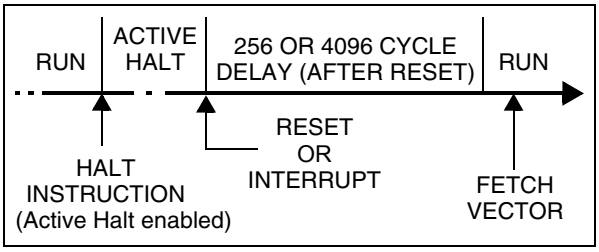
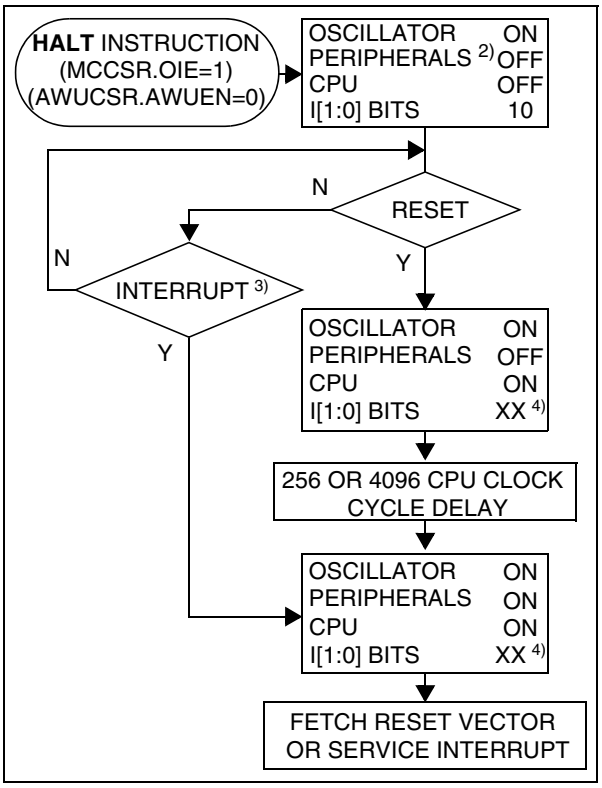


Figure 28. ACTIVE HALT Mode Flow-chart



Notes:

1. This delay occurs only if the MCU exits ACTIVE HALT mode by means of a RESET.
2. Peripheral clocked with an external clock source can still be active.
3. Only the RTC interrupt and some specific interrupts can exit the MCU from ACTIVE HALT mode (such as external interrupt). Refer to Table 9, "Interrupt Mapping," on page 34 for more details.
4. Before servicing an interrupt, the CC register is pushed on the stack. The I[1:0] bits in the CC register are set to the current software priority level of the interrupt routine and restored when the CC register is popped.

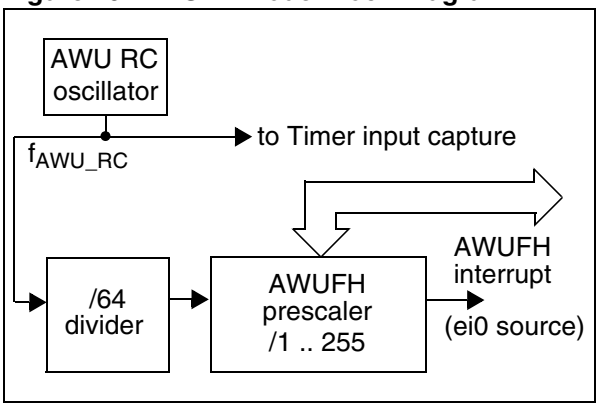
POWER SAVING MODES (Cont'd)

8.6 AUTO WAKE-UP FROM HALT MODE

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

It is entered by executing the HALT instruction when the AWUEN bit in the AWUCSR register has been set and the OIE bit in the MCCSR register is cleared (see Section 10.2 on page 59 for more details).

Figure 29. 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 ICAP1 input of the 16-bit timer, allowing the f_{AWU_RC} to be measured using the main oscillator clock as a reference timebase.

Similarities with Halt mode

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

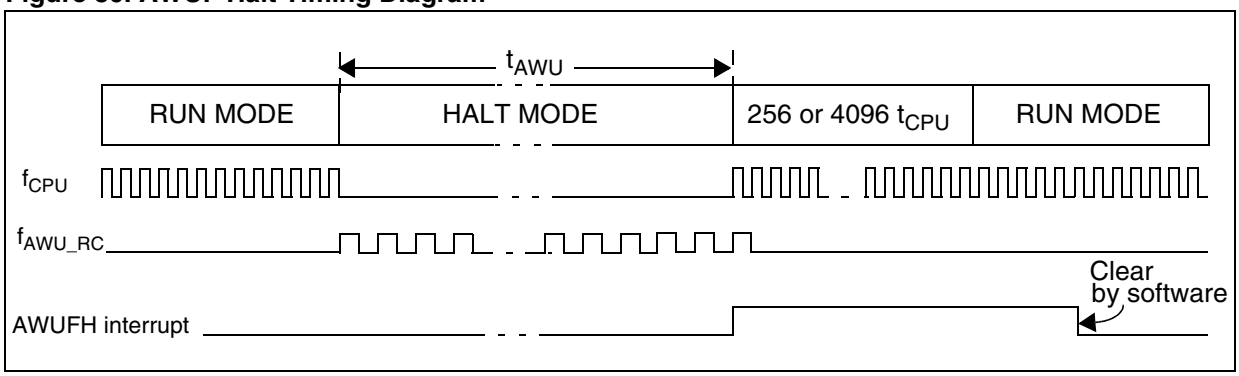
– The MCU can exit AWUFH mode by means of any interrupt with exit from Halt capability or a reset (see Section 8.4 "HALT MODE").

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

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

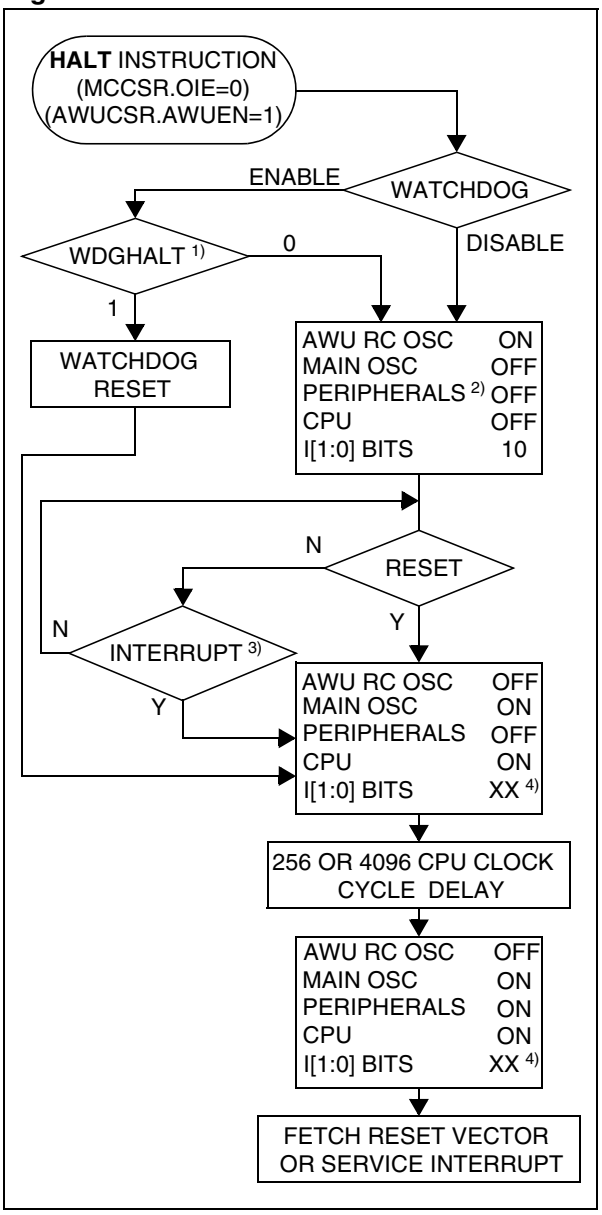
– The compatibility of Watchdog operation with AWUFH mode is configured by the WDGHALT option bit in the option byte. Depending on this setting, the HALT instruction when executed while the Watchdog system is enabled, can generate a Watchdog RESET.

Figure 30. AWUF Halt Timing Diagram



POWER SAVING MODES (Cont'd)

Figure 31. AWUFH Mode Flow-chart



Notes:

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

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

3. Only an AWUFH interrupt and some specific interrupts can exit the MCU from HALT mode (such as external interrupt). Refer to Table 9, "Interrupt Mapping," on page 34 for more details.

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

POWER SAVING MODES (Cont'd)

8.6.1 Register Description

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

7							0
0	0	0	0	0	AWUF	AWUM	AWUEN

Bits 7:3 = Reserved.

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

Bit 1 = **AWUM** *Auto Wake-Up Measurement*
This bit enables the AWU RC oscillator and connects its output to the ICAP1 input of the 16-bit timer. This allows the timer to be used to measure the AWU RC oscillator dispersion and then compensate this dispersion by providing the right value in the AWUPR register.
0: Measurement disabled
1: Measurement enabled

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

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

7							0
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 (See note)
01h	1
...	...
FEh	254
FFh	255

In AWU mode, the period that the MCU stays in Halt Mode (t_{AWU} in Figure 30) 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 11. AWU Register Map and Reset Values

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

9 I/O PORTS

9.1 INTRODUCTION

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

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

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

9.2 FUNCTIONAL DESCRIPTION

Each port has two main registers:

– Data Register (DR)

– Data Direction Register (DDR)

and one optional register:

– Option Register (OR)

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

The following description takes into account the OR register, (for specific ports which do not provide this register refer to the I/O Port Implementation section). The generic I/O block diagram is shown in Figure 32

9.2.1 Input Modes

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

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

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

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

External interrupt function

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

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

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

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

9.2.2 Output Modes

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

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

DR register value and output pin status:

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

9.2.3 Alternate Functions

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

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

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

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

I/O PORTS (Cont'd)

Figure 32. I/O Port General Block Diagram

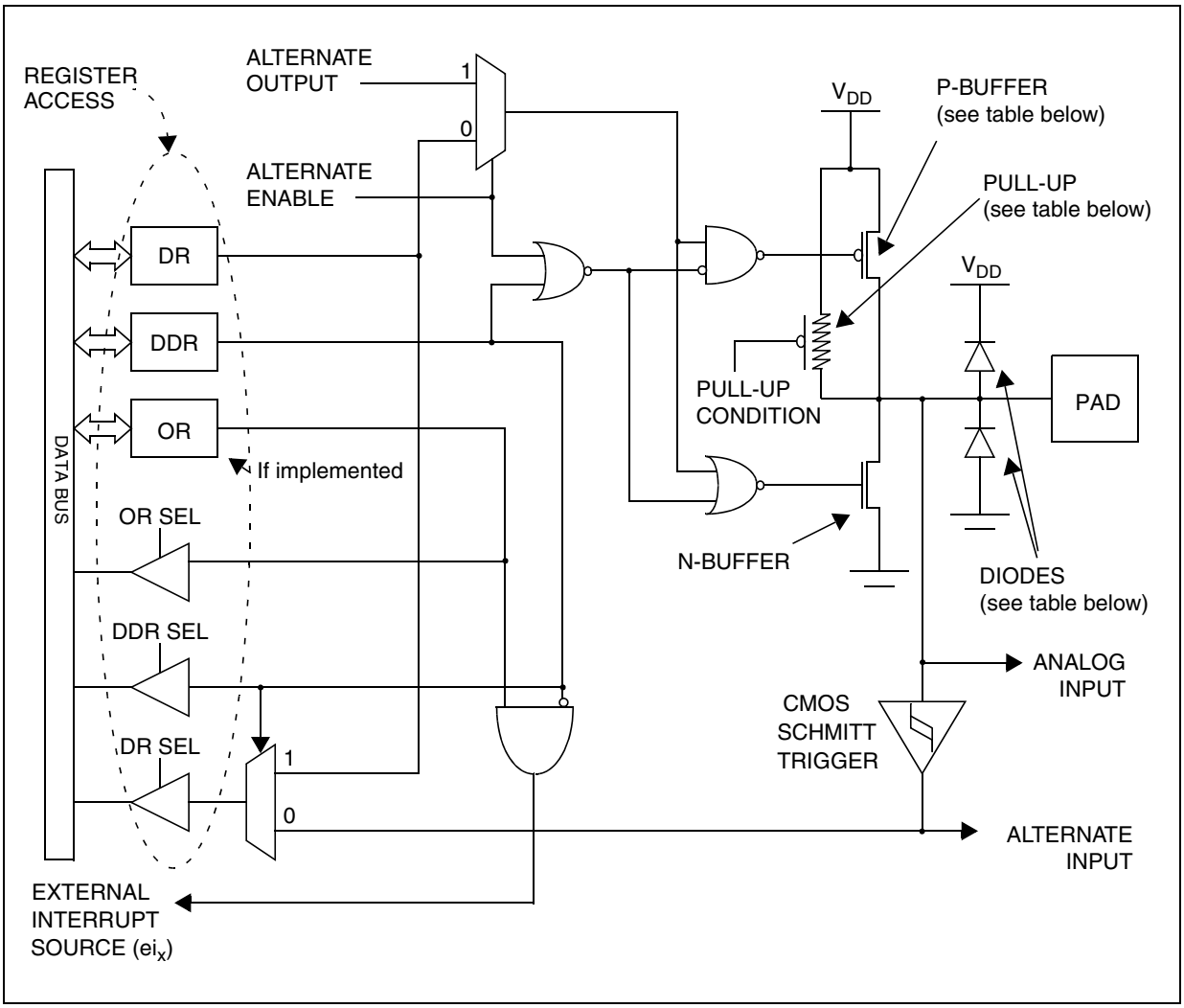


Table 12. I/O Port Mode Options

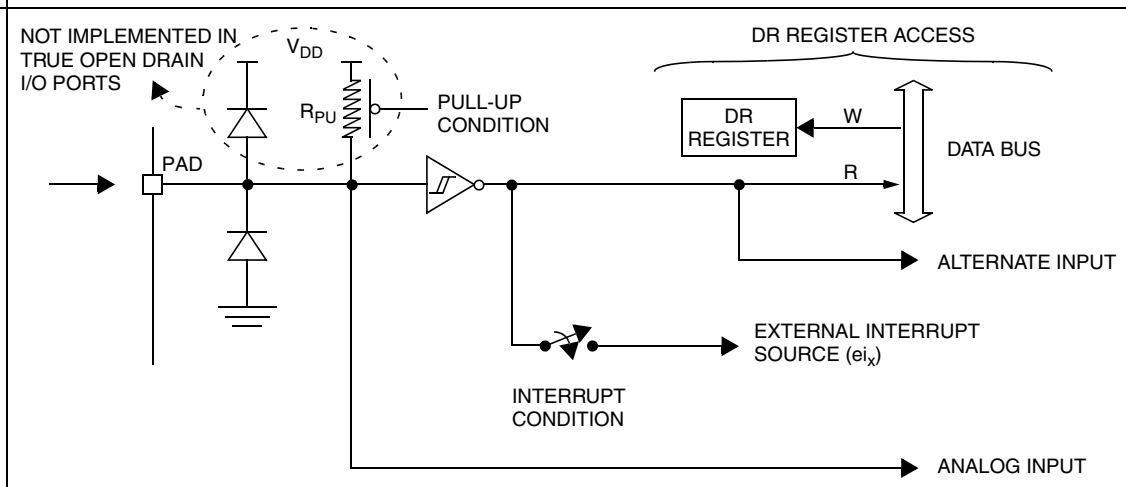
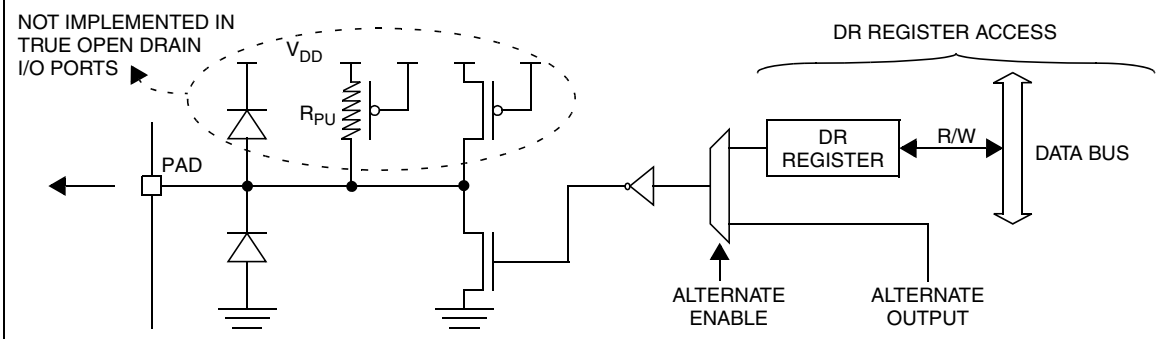
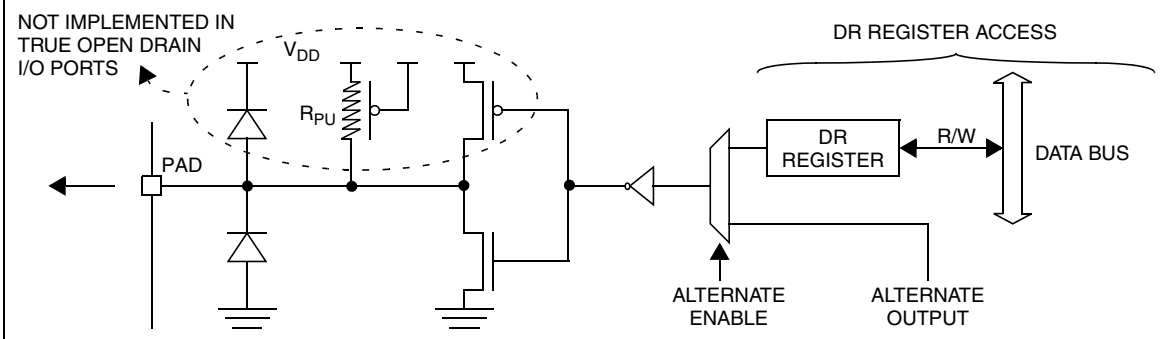
Configuration Mode		Pull-Up	P-Buffer	Diodes	
				to V_{DD}	to V_{SS}
Input	Floating with/without Interrupt	Off	Off	On	On
	Pull-up with/without Interrupt	On			
Output	Push-pull	Off	On		
	Open Drain (logic level)		Off		
	True Open Drain	NI	NI	NI (see note)	

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

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

I/O PORTS (Cont'd)

Table 13. I/O Port Configurations

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

Notes:

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

I/O PORTS (Cont'd)

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

Analog alternate function

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

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

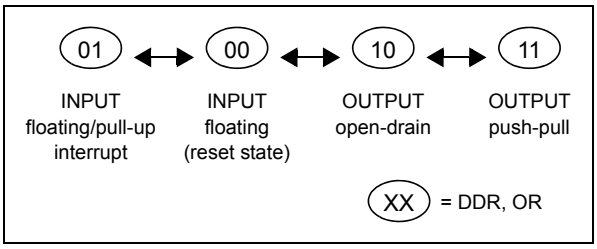
WARNING: The analog input voltage level must be within the limits stated in the absolute maximum ratings.

9.3 I/O PORT IMPLEMENTATION

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

Switching these I/O ports from one state to another should be done in a sequence that prevents unwanted side effects. Recommended safe transitions are illustrated in Figure 33 on page 49. Other transitions are potentially risky and should be avoided, since they are likely to present unwanted side-effects such as spurious interrupt generation.

Figure 33. Interrupt I/O Port State Transitions



9.4 LOW POWER MODES

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

9.5 INTERRUPTS

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

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

I/O PORTS (Cont'd)

9.6 I/O PORT REGISTER CONFIGURATIONS

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

9.6.1 Standard Ports

PB7:6, PC0, PC3, PC7:5, PD3:2, PD5, PE7:0, PF7:0

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

9.6.2 Interrupt Ports

PA0,2,4,6; PB0,2,4; PC1; PD0,6

(with pull-up)

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

PA1,3,5,7; PB1,3,5; PC2; PD1,4,7

(without pull-up)

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

9.6.3 Pull-up Input Port (CANTX requirement)

PC4

MODE
pull-up input

The PC4 port cannot operate as a general purpose output. The CAN peripheral controls it directly when enabled. Otherwise, PC4 is a pull-up input.

If DDR = 1 it is still possible to read the port through the DR register.

I/O PORTS (Cont'd)

Table 14. Port Configuration

Port	Pin name	Input		Output	
		OR = 0	OR = 1	OR = 0	OR = 1
Port A	PA0	floating	pull-up interrupt (ei0)	open drain	push-pull
	PA1		floating interrupt (ei0)		
	PA2		pull-up interrupt (ei0)		
	PA3		floating interrupt (ei0)		
	PA4		pull-up interrupt (ei0)		
	PA5		floating interrupt (ei0)		
	PA6		pull-up interrupt (ei0)		
	PA7		floating interrupt (ei0)		
Port B	PB0	floating	pull-up interrupt (ei1)	open drain	push-pull
	PB1		floating interrupt (ei1)		
	PB2		pull-up interrupt (ei1)		
	PB3		floating interrupt (ei1)		
	PB4		pull-up interrupt (ei1)		
	PB5		floating interrupt (ei1)		
Port C	PC0	floating	pull-up	open drain	push-pull
	PC1		pull-up interrupt (ei2)		
	PC2		floating interrupt (ei2)		
	PC3		pull-up		
	PC4	pull-up		controlled by CANTX *	
	PC7:5	floating	pull-up	open drain	push-pull
Port D	PD0	floating	pull-up interrupt (ei3)	open drain	push-pull
	PD1		floating interrupt (ei3)		
	PD3:2		pull-up		
	PD4		floating interrupt (ei3)		
	PD5		pull-up		
	PD6		pull-up interrupt (ei3)		
	PD7		floating interrupt (ei3)		
Port E	PE7:0	floating (TTL)	pull-up (TTL)	open drain	push-pull
Port F	PF7:0	floating (TTL)	pull-up (TTL)	open drain	push-pull

* Note: When the CANTX alternate function is selected, the I/O port operates in output push-pull mode.

I/O PORTS (Cont'd)

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

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

10 ON-CHIP PERIPHERALS

10.1 WINDOW WATCHDOG (WWDG)

10.1.1 Introduction

The Window Watchdog is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The Watchdog circuit generates an MCU reset on expiry of a programmed time period, unless the program refreshes the contents of the downcounter before the T6 bit becomes cleared. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter has reached the window register value. This implies that the counter must be refreshed in a limited window.

10.1.2 Main Features

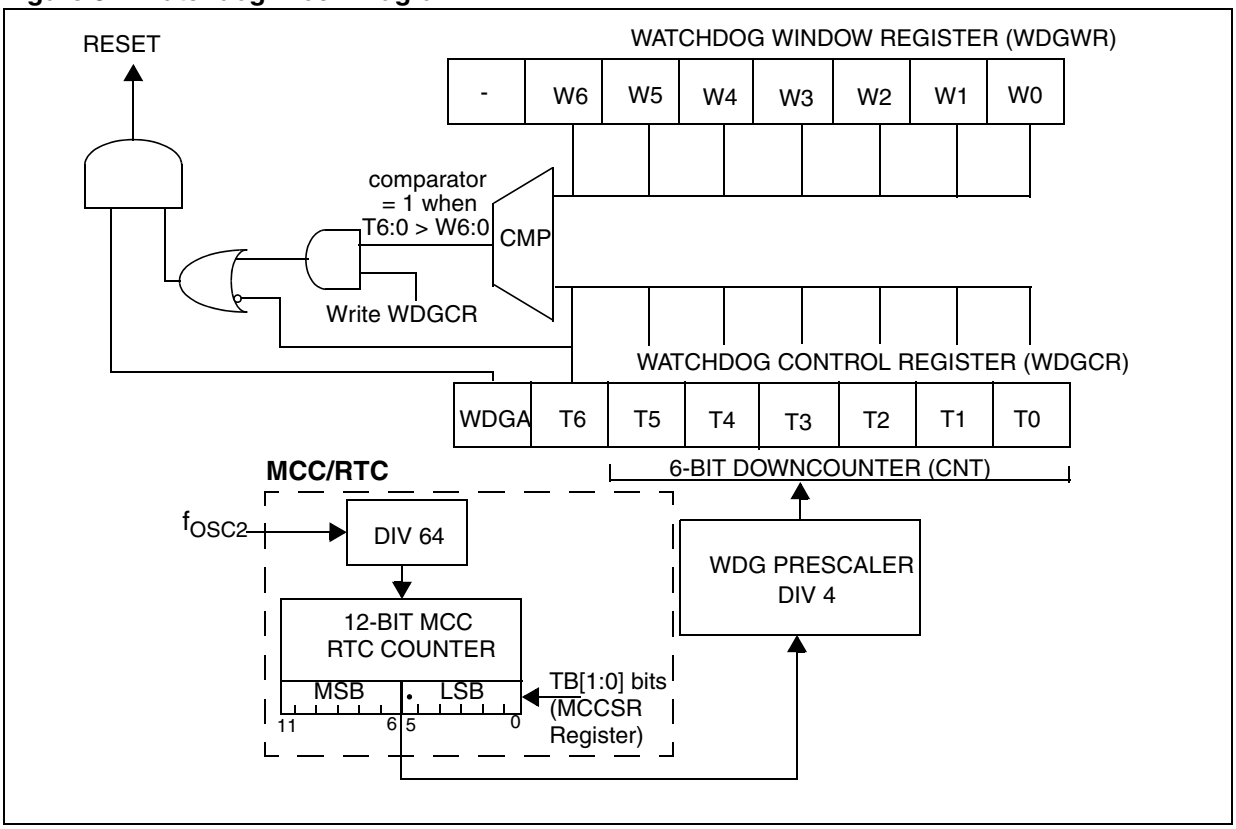
- Programmable free-running downcounter
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes less than 40h
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window (see Figure 4)
- Hardware/Software Watchdog activation (selectable by option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

10.1.3 Functional Description

The counter value stored in the WDGCR register (bits T[6:0]), is decremented every 16384 f_{OSC2} cycles (approx.), and the length of the timeout period can be programmed by the user in 64 increments.

If the watchdog is activated (the WDGA bit is set) and when the 7-bit downcounter (T[6:0] bits) rolls over from 40h to 3Fh (T6 becomes cleared), it initiates a reset cycle pulling low the reset pin for typically 30µs. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

Figure 34. Watchdog Block Diagram



WINDOW WATCHDOG (Cont'd)

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

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

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

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

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

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

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

10.1.4 Using Halt Mode with the WDG

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

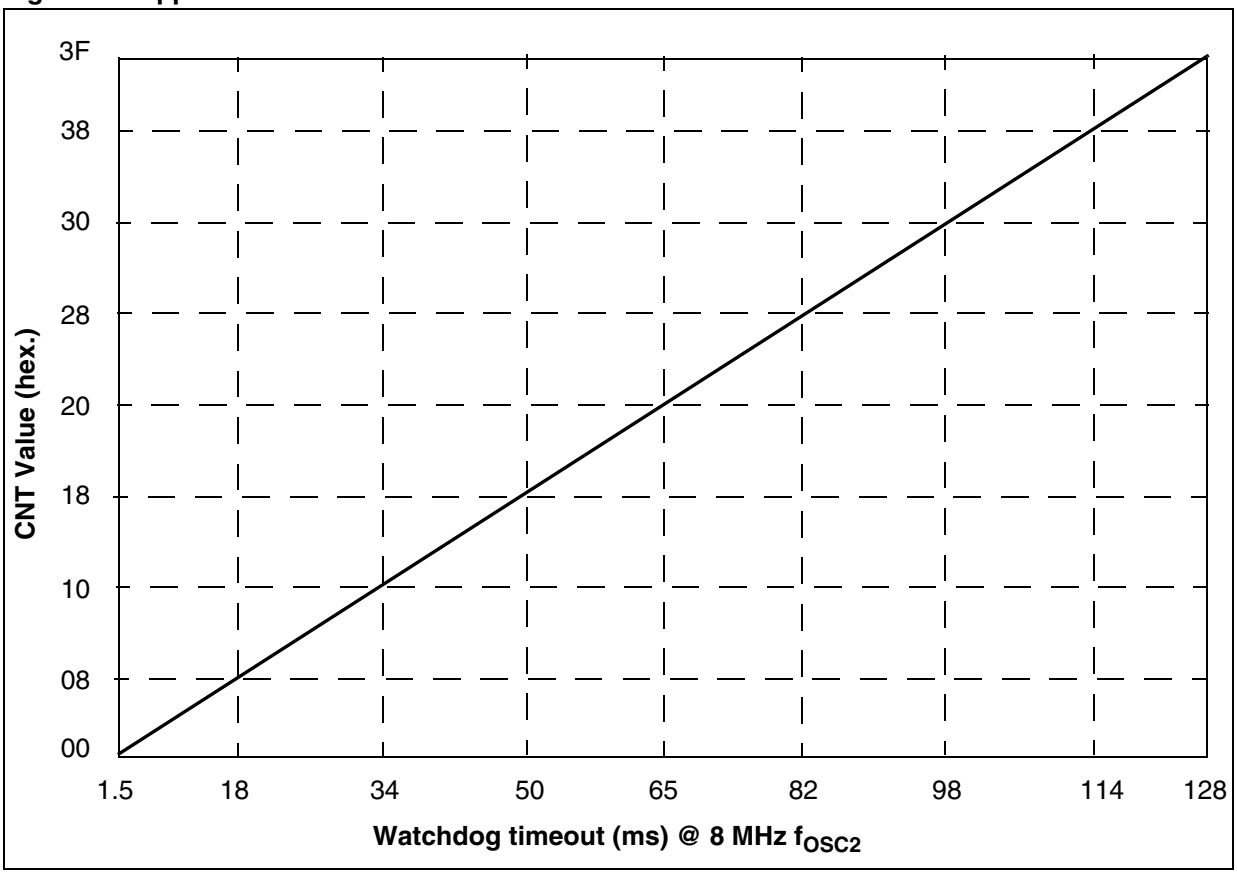
WINDOW WATCHDOG (Cont'd)

10.1.5 How to Program the Watchdog Timeout

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

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

Figure 35. Approximate Timeout Duration



WINDOW WATCHDOG (Cont'd)

Figure 36. Exact Timeout Duration (t_{min} and t_{max})

WHERE:

$t_{min0} = (LSB + 128) \times 64 \times t_{OSC2}$

$t_{max0} = 16384 \times t_{OSC2}$

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

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

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

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

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

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

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

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

IF $CNT \leq \left\lceil \frac{MSB}{4} \right\rceil$ **THEN** $t_{max} = t_{max0} + 16384 \times CNT \times t_{osc2}$

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

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

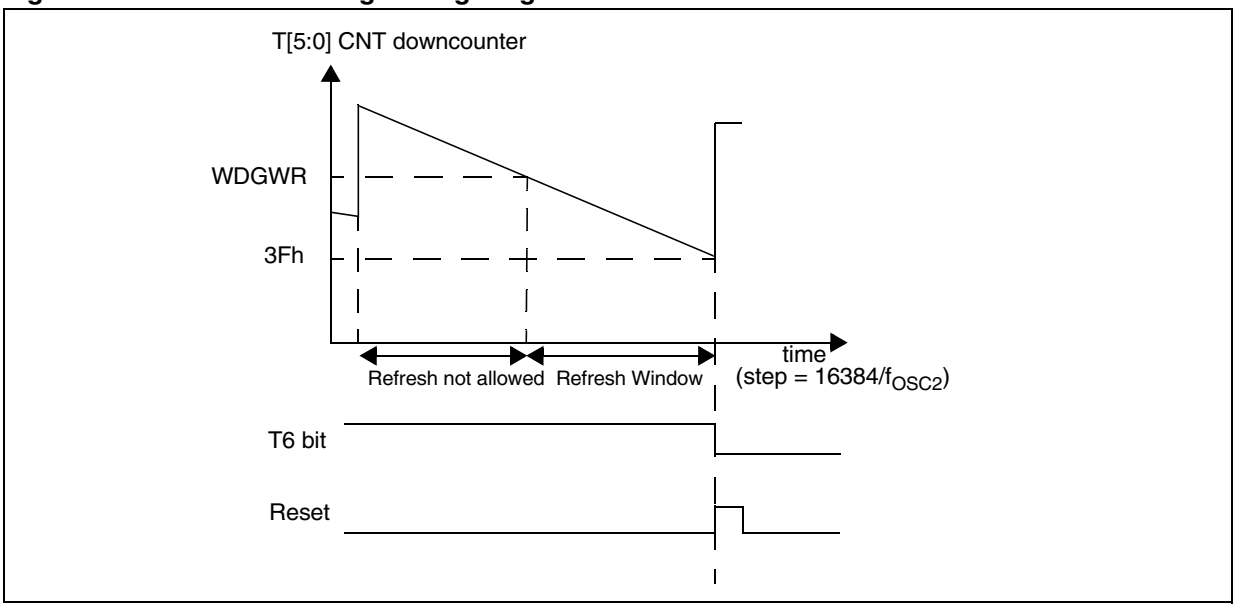
Example:

With 2ms timeout selected in MCCSR register

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

WINDOW WATCHDOG (Cont'd)

Figure 37. Window Watchdog Timing Diagram



10.1.6 Low Power Modes

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

10.1.7 Hardware Watchdog Option

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

10.1.8 Using Halt Mode with the WDG (WDGHALT option)

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

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

WINDOW WATCHDOG (Cont'd)

10.1.9 Interrupts

None.

10.1.10 Register Description

CONTROL REGISTER (WDGCR)

Read/Write

Reset Value: 0111 1111 (7Fh)

7							0
WDGA	T6	T5	T4	T3	T2	T1	T0

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

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

0: Watchdog disabled

1: Watchdog enabled

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

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

WINDOW REGISTER (WDGWR)

Read/Write

Reset Value: 0111 1111 (7Fh)

7							0
-	W6	W5	W4	W3	W2	W1	W0

Bit 7 = Reserved

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

Figure 38. Watchdog Timer Register Map and Reset Values

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

ON-CHIP PERIPHERALS (Cont'd)

10.2 MAIN CLOCK CONTROLLER WITH REAL TIME CLOCK MCC/RTC

The Main Clock Controller consists of three different functions:

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

Each function can be used independently and simultaneously.

10.2.1 Programmable CPU Clock Prescaler

The programmable CPU clock prescaler supplies the clock for the ST7 CPU and its internal peripherals. It manages SLOW power saving mode (See Section 8.2 "SLOW MODE" for more details).

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

10.2.2 Clock-out Capability

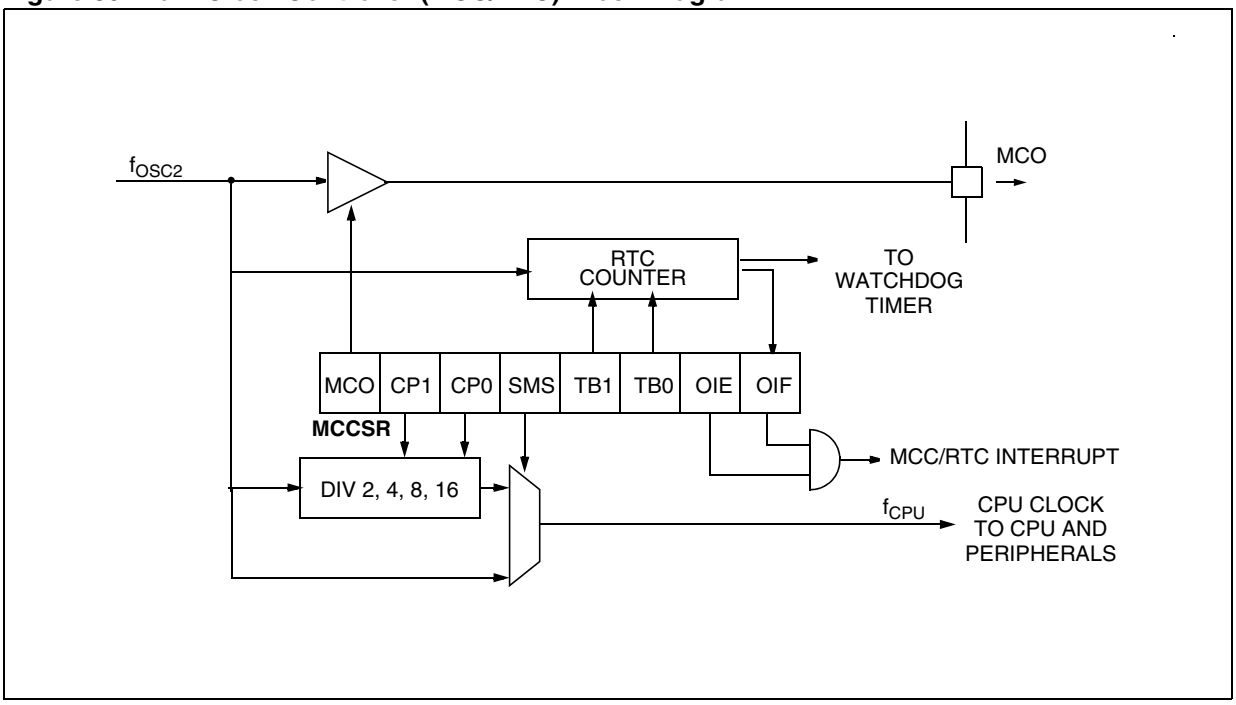
The clock-out capability is an alternate function of an I/O port pin that outputs a f_{OSC2} clock to drive external devices. It is controlled by the MCO bit in the MCCSR register.

10.2.3 Real Time Clock Timer (RTC)

The counter of the real time clock timer allows an interrupt to be generated based on an accurate real time clock. Four different time bases depending directly on f_{OSC2} are available. The whole functionality is controlled by 4 bits of the MCCSR register: TB[1:0], OIE and OIF.

When the RTC interrupt is enabled (OIE bit set), the ST7 enters ACTIVE HALT mode when the HALT instruction is executed. See Section 8.5 "ACTIVE HALT MODE" for more details.

Figure 39. Main Clock Controller (MCC/RTC) Block Diagram



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

10.2.4 Low Power Modes

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

10.2.5 Interrupts

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

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

Note:

The MCC/RTC interrupt wakes up the MCU from ACTIVE HALT mode, not from HALT or AWUF HALT mode.

10.2.6 Register Description

MCC CONTROL/STATUS REGISTER (MCCSR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
MCO	CP1	CP0	SMS	TB1	TB0	OIE	OIF

Bit 7 = **MCO** *Main clock out selection*
This bit enables the MCO alternate function on the corresponding I/O port. It is set and cleared by software.
0: MCO alternate function disabled (I/O pin free for general-purpose I/O)
1: MCO alternate function enabled (f_{OSC2} on I/O port)

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

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

Bit 4 = **SMS** *Slow mode select*
This bit is set and cleared by software.
0: Normal mode. $f_{CPU} = f_{OSC2}$
1: Slow mode. f_{CPU} is given by CP1, CP0
See Section 8.2 "SLOW MODE" and Section 10.2 "MAIN CLOCK CONTROLLER WITH REAL TIME CLOCK MCC/RTC" for more details.

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

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

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

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

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

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

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

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

Table 16. Main Clock Controller Register Map and Reset Values

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

ON-CHIP PERIPHERALS (Cont'd)

10.3 PWM AUTO-RELOAD TIMER (ART)

10.3.1 Introduction

The Pulse Width Modulated Auto-Reload Timer on-chip peripheral consists of an 8-bit auto reload counter with compare/capture capabilities and of a 7-bit prescaler clock source.

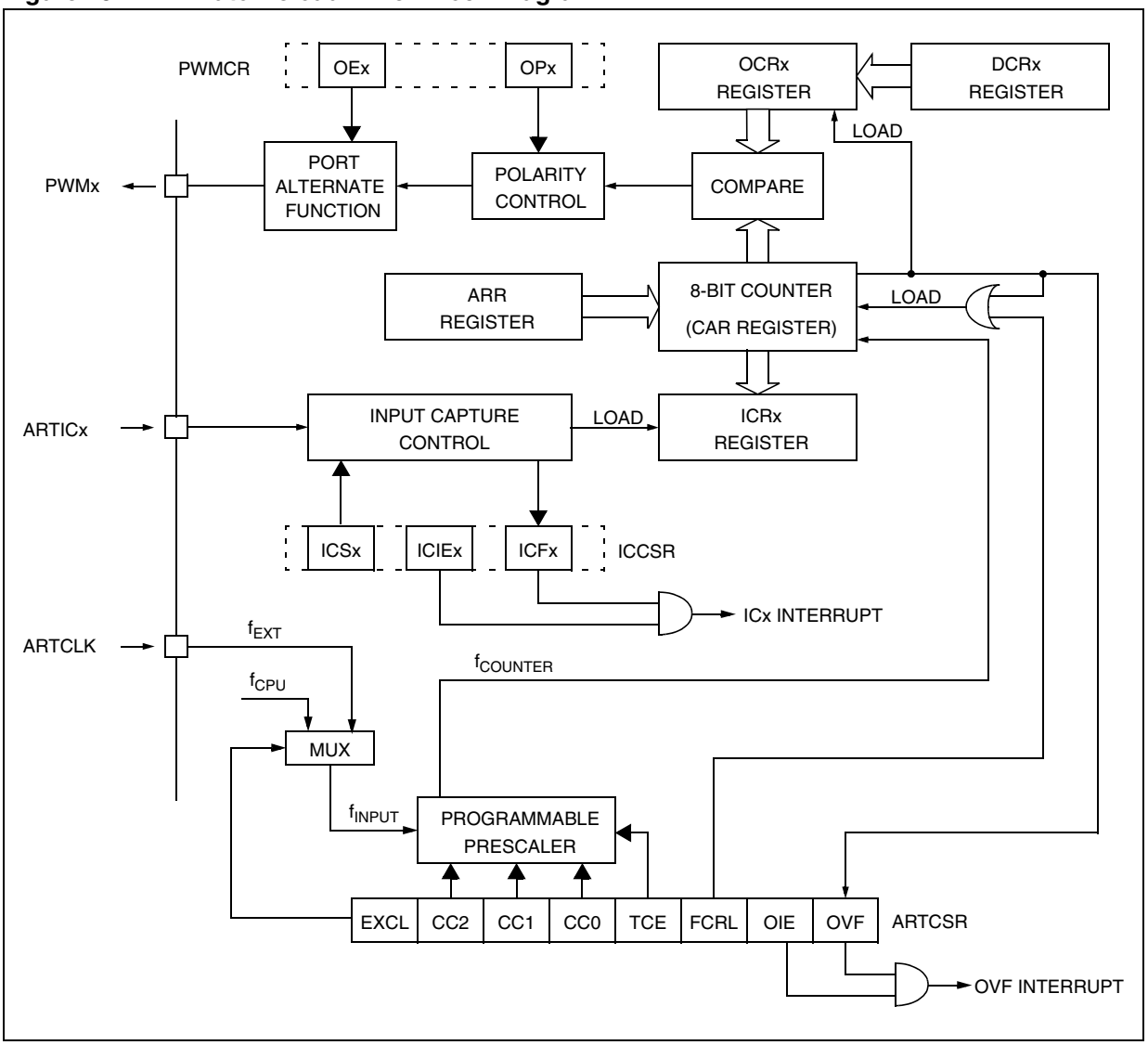
These resources allow five possible operating modes:

– Generation of up to four independent PWM signals
– Output compare and Time base interrupt
– Up to two input capture functions
– External event detector
– Up to two external interrupt sources

The three first modes can be used together with a single counter frequency.

The timer can be used to wake up the MCU from WAIT and HALT modes.

Figure 40. PWM Auto-Reload Timer Block Diagram



PWM AUTO-RELOAD TIMER (Cont'd)

10.3.2 Functional Description

Counter

The free running 8-bit counter is fed by the output of the prescaler, and is incremented on every rising edge of the clock signal.

It is possible to read or write the contents of the counter on the fly by reading or writing the Counter Access register (ARTCAR).

When a counter overflow occurs, the counter is automatically reloaded with the contents of the ARTARR register (the prescaler is not affected).

Counter clock and prescaler

The counter clock frequency is given by:

$f_{COUNTER} = f_{INPUT} / 2^{CC[2:0]}$

The timer counter's input clock (f_{INPUT}) feeds the 7-bit programmable prescaler, which selects one of the eight available taps of the prescaler, as defined by CC[2:0] bits in the Control/Status Register (ARTCSR). Thus the division factor of the prescaler can be set to 2^n (where n = 0, 1,..7).

This f_{INPUT} frequency source is selected through the EXCL bit of the ARTCSR register and can be either the f_{CPU} or an external input frequency f_{EXT}.

The clock input to the counter is enabled by the TCE (Timer Counter Enable) bit in the ARTCSR register. When TCE is reset, the counter is stopped and the prescaler and counter contents are frozen. When TCE is set, the counter runs at the rate of the selected clock source.

Counter and Prescaler Initialization

After RESET, the counter and the prescaler are cleared and $f_{INPUT} = f_{CPU}$.

The counter can be initialized by:

– Writing to the ARTARR register and then setting the FCRL (Force Counter Re-Load) and the TCE (Timer Counter Enable) bits in the ARTCSR register.

– Writing to the ARTCAR counter access register,

In both cases the 7-bit prescaler is also cleared, whereupon counting will start from a known value.

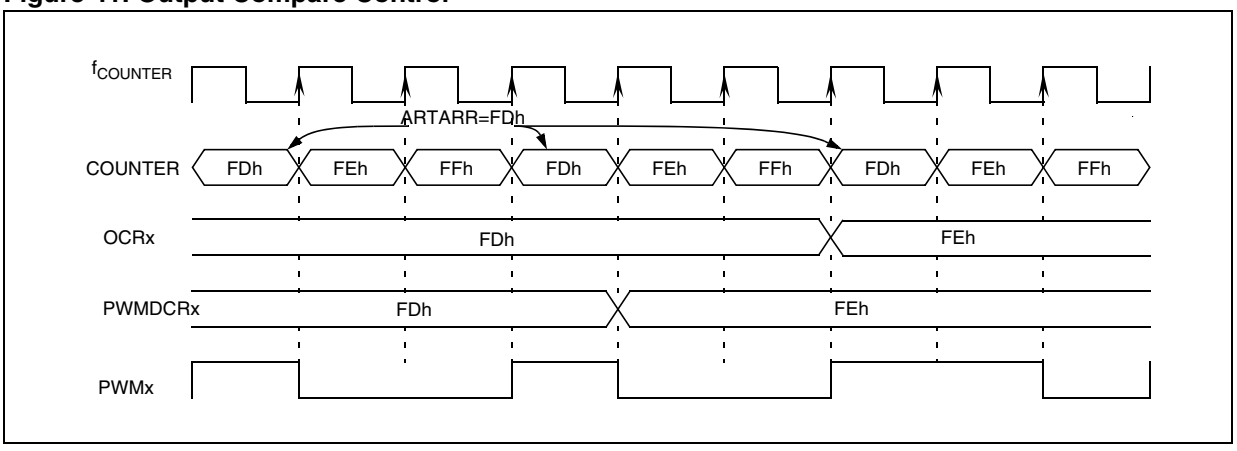
Direct access to the prescaler is not possible.

Output compare control

The timer compare function is based on four different comparisons with the counter (one for each PWMx output). Each comparison is made between the counter value and an output compare register (OCRx) value. This OCRx register can not be accessed directly, it is loaded from the duty cycle register (PWMDCRx) at each overflow of the counter.

This double buffering method avoids glitch generation when changing the duty cycle on the fly.

Figure 41. Output Compare Control



PWM AUTO-RELOAD TIMER (Cont'd)

Independent PWM signal generation

This mode allows up to four Pulse Width Modulated signals to be generated on the PWMx output pins with minimum core processing overhead. This function is stopped during HALT mode.

Each PWMx output signal can be selected independently using the corresponding OEx bit in the PWM Control register (PWMCR). When this bit is set, the corresponding I/O pin is configured as output push-pull alternate function.

The PWM signals all have the same frequency which is controlled by the counter period and the ARTARR register value.

$f_{PWM} = f_{COUNTER} / (256 - ARTARR)$

When a counter overflow occurs, the PWMx pin level is changed depending on the corresponding OPx (output polarity) bit in the PWMCR register. When the counter reaches the value contained in one of the output compare register (OCRx) the corresponding PWMx pin level is restored.

It should be noted that the reload values will also affect the value and the resolution of the duty cycle of the PWM output signal. To obtain a signal on a PWMx pin, the contents of the OCRx register must be greater than the contents of the ARTARR register.

The maximum available resolution for the PWMx duty cycle is:

Resolution = 1 / (256 - ARTARR)

Note: To get the maximum resolution (1/256), the ARTARR register must be 0. With this maximum resolution, 0% and 100% can be obtained by changing the polarity.

Figure 42. PWM Auto-reload Timer Function

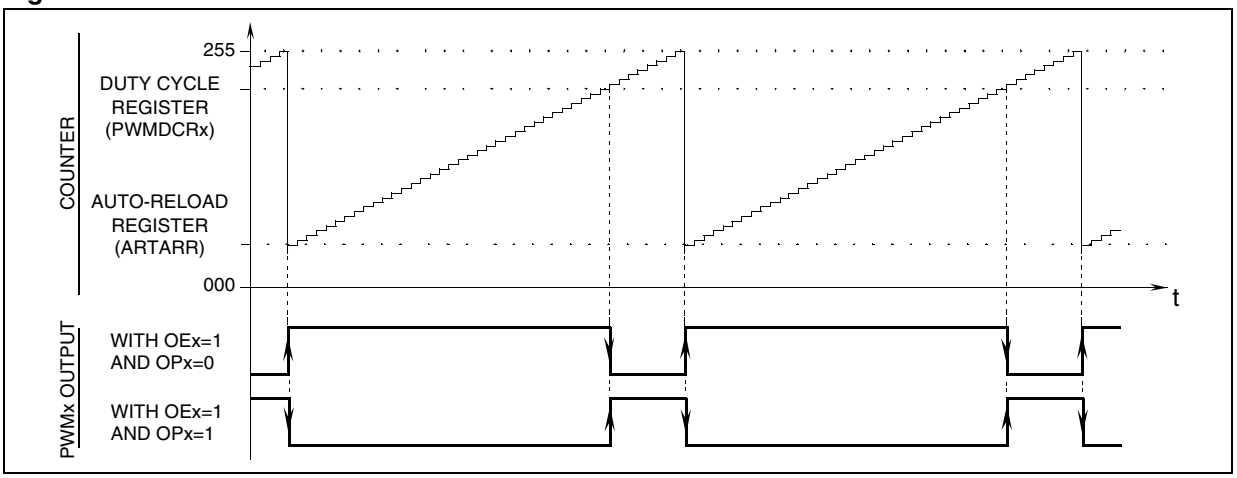
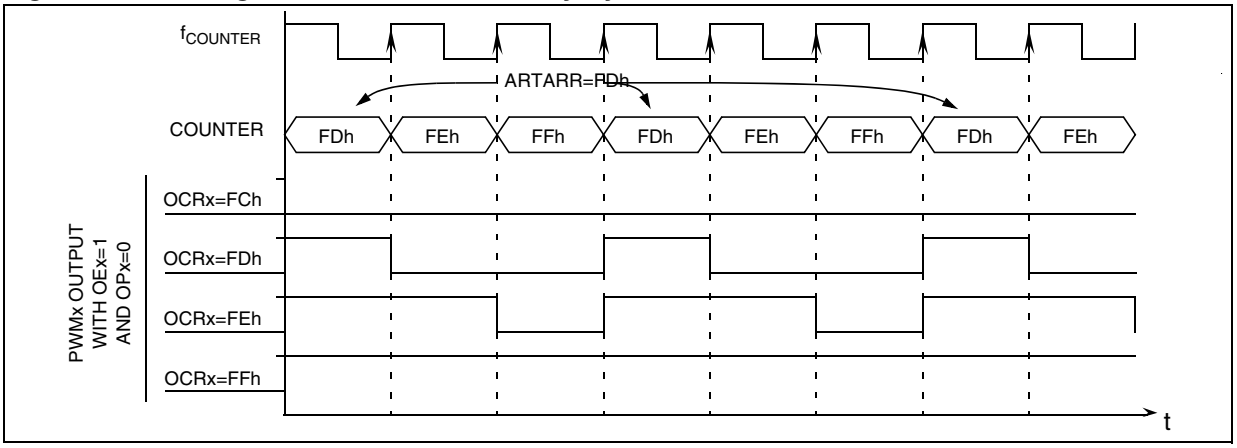


Figure 43. PWM Signal from 0% to 100% Duty Cycle



PWM AUTO-RELOAD TIMER (Cont'd)

Output compare and Time base interrupt

On overflow, the OVF flag of the ARTCSR register is set and an overflow interrupt request is generated if the overflow interrupt enable bit, OIE, in the ARTCSR register, is set. The OVF flag must be reset by the user software. This interrupt can be used as a time base in the application.

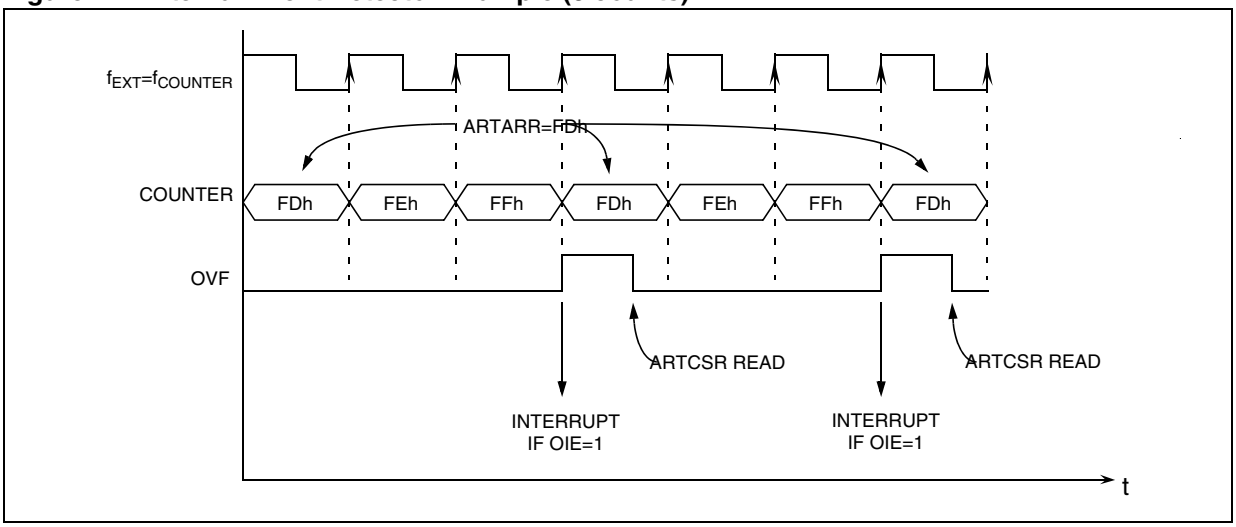
External clock and event detector mode

Using the f_{EXT} external prescaler input clock, the auto-reload timer can be used as an external clock event detector. In this mode, the ARTARR register is used to select the n_{EVENT} number of events to be counted before setting the OVF flag.

n_{EVENT} = 256 - ARTARR

Caution: The external clock function is not available in HALT mode. If HALT mode is used in the application, prior to executing the HALT instruction, the counter must be disabled by clearing the TCE bit in the ARTCSR register to avoid spurious counter increments.

Figure 44. External Event Detector Example (3 counts)



PWM AUTO-RELOAD TIMER (Cont'd)

Input Capture Function

Input Capture mode allows the measurement of external signal pulse widths through ARTICRx registers.

Each input capture can generate an interrupt independently on a selected input signal transition. This event is flagged by a set of the corresponding CFx bits of the Input Capture Control/Status register (ARTICCSR).

These input capture interrupts are enabled through the CIEx bits of the ARTICCSR register.

The active transition (falling or rising edge) is software programmable through the CSx bits of the ARTICCSR register.

The read only input capture registers (ARTICRx) are used to latch the auto-reload counter value when a transition is detected on the ARTICx pin (CFx bit set in ARTICCSR register). After fetching the interrupt vector, the CFx flags can be read to identify the interrupt source.

Note: After a capture detection, data transfer in the ARTICRx register is inhibited until the next read (clearing the CFx bit).
The timer interrupt remains pending while the CFx flag is set when the interrupt is enabled (CIEx bit set). This means, the ARTICRx register has to be read at each capture event to clear the CFx flag.

The timing resolution is given by auto-reload counter cycle time ($1/f_{COUNTER}$).

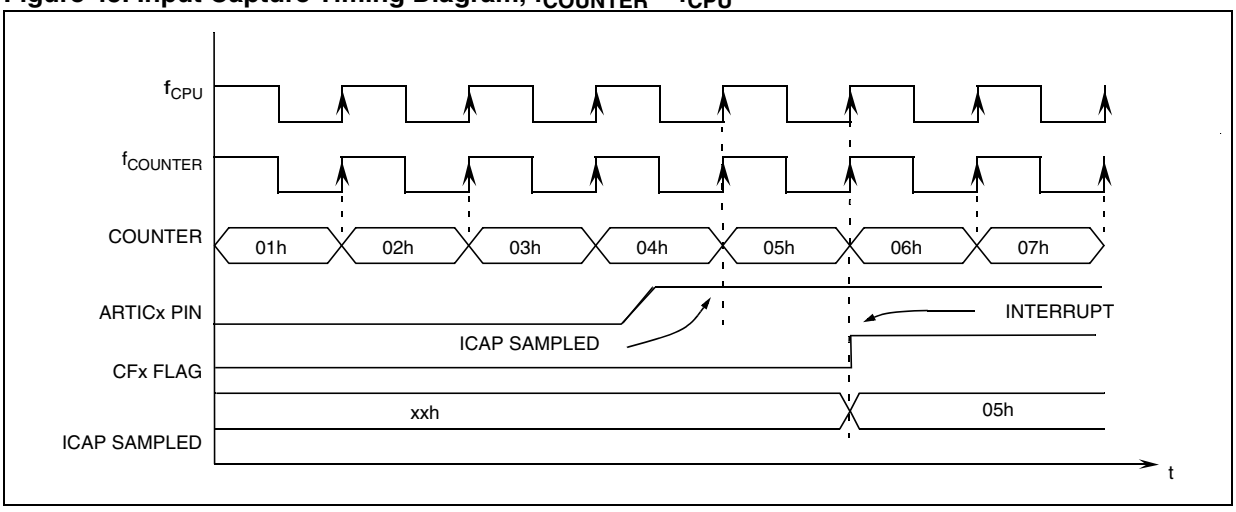
Note: During HALT mode, input capture is inhibited (the ARTICRx is never reloaded) and only the external interrupt capability can be used.

Note: The ARTICx signal is synchronized on CPU clock. It takes two rising edges until ARTICRx is latched with the counter value. Depending on the prescaler value and the time when the ICAP event occurs, the value loaded in the ARTICRx register may be different.

If the counter is clocked with the CPU clock, the value latched in ARTICRx is always the next counter value after the event on ARTICx occurred (Figure 45).

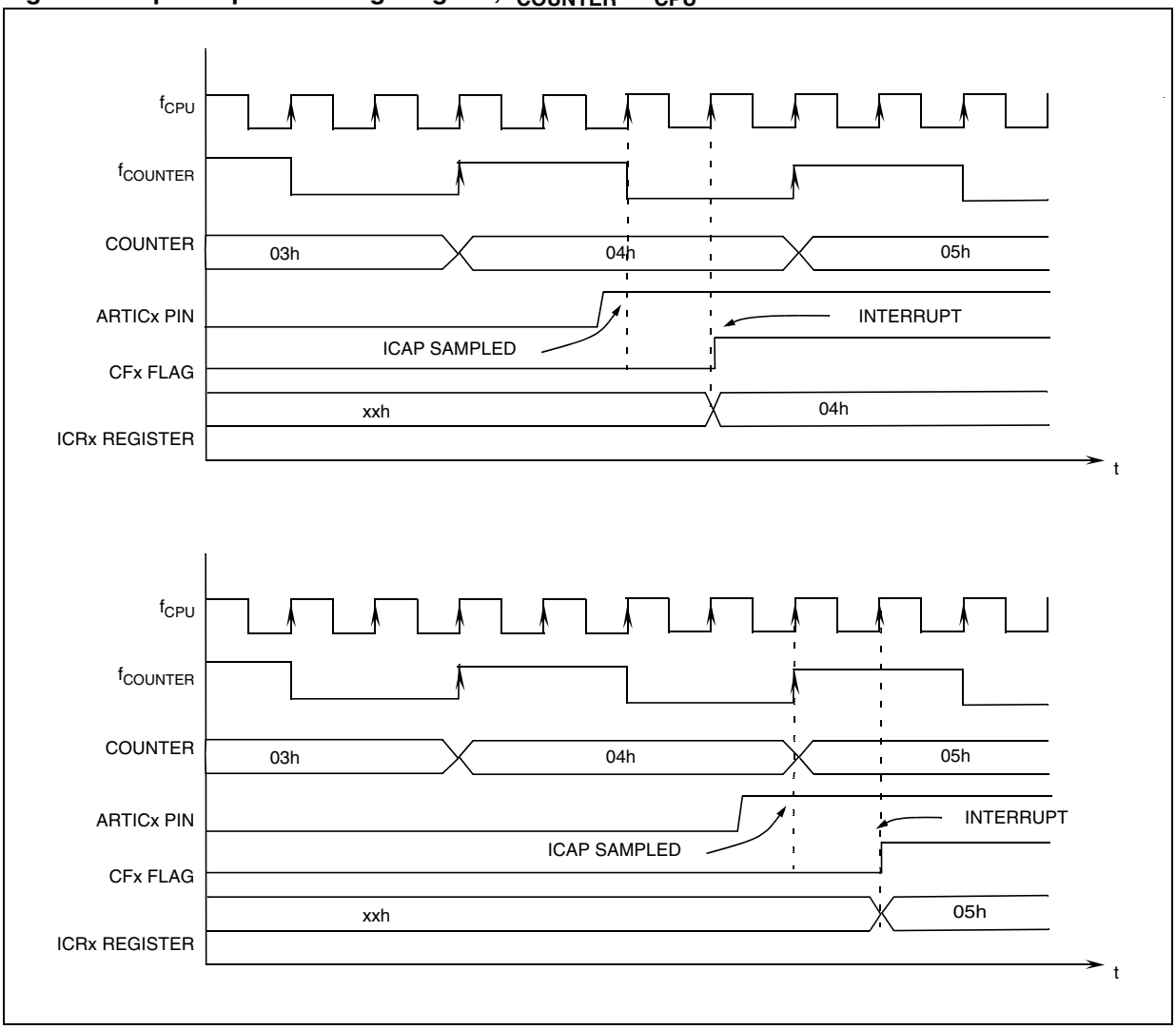
If the counter clock is prescaled, it depends on the position of the ARTICx event within the counter cycle (Figure 46).

Figure 45. Input Capture Timing Diagram, $f_{COUNTER} = f_{CPU}$



PWM AUTO-RELOAD TIMER (Cont'd)

Figure 46. input Capture Timing Diagram, $f_{COUNTER} = f_{CPU} / 4$



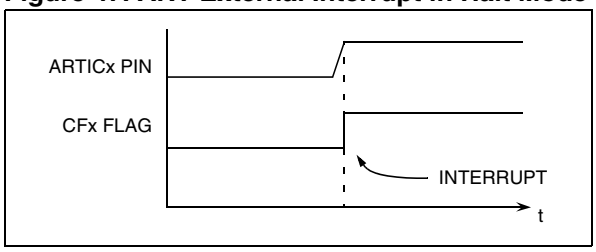
External Interrupt Capability

This mode allows the Input capture capabilities to be used as external interrupt sources. The interrupts are generated on the edge of the ARTICx signal.

The edge sensitivity of the external interrupts is programmable (CSx bit of ARTICCSR register) and they are independently enabled through CIEx bits of the ARTICCSR register. After fetching the interrupt vector, the CFx flags can be read to identify the interrupt source.

During HALT mode, the external interrupts can be used to wake up the micro (if the CIEx bit is set). In this case, the interrupt synchronization is done directly on the ARTICx pin edge (Figure 47).

Figure 47. ART External Interrupt in Halt Mode



ON-CHIP PERIPHERALS (Cont'd)

10.3.3 Register Description

CONTROL / STATUS REGISTER (ARTCSR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
EXCL	CC2	CC1	CC0	TCE	FCRL	OIE	OVF

Bit 7 = **EXCL** *External Clock*
This bit is set and cleared by software. It selects the input clock for the 7-bit prescaler.
0: CPU clock.
1: External clock.

Bit 6:4 = **CC[2:0]** *Counter Clock Control*
These bits are set and cleared by software. They determine the prescaler division ratio from f_{INPUT}.

$f_{COUNTER}$	With f_{INPUT}=8 MHz	CC2	CC1	CC0
f_{INPUT}	8 MHz	0	0	0
f_{INPUT} / 2	4 MHz	0	0	1
f_{INPUT} / 4	2 MHz	0	1	0
f_{INPUT} / 8	1 MHz	0	1	1
f_{INPUT} / 16	500 kHz	1	0	0
f_{INPUT} / 32	250 kHz	1	0	1
f_{INPUT} / 64	125 kHz	1	1	0
f_{INPUT} / 128	62.5 kHz	1	1	1

Bit 3 = **TCE** *Timer Counter Enable*
This bit is set and cleared by software. It puts the timer in the lowest power consumption mode.
0: Counter stopped (prescaler and counter frozen).
1: Counter running.

Bit 2 = **FCRL** *Force Counter Re-Load*
This bit is write-only and any attempt to read it will yield a logical zero. When set, it causes the contents of ARTARR register to be loaded into the counter, and the content of the prescaler register to be cleared in order to initialize the timer before starting to count.

Bit 1 = **OIE** *Overflow Interrupt Enable*
This bit is set and cleared by software. It allows to enable/disable the interrupt which is generated when the OVF bit is set.
0: Overflow Interrupt disable.
1: Overflow Interrupt enable.

Bit 0 = **OVF** *Overflow Flag*
This bit is set by hardware and cleared by software reading the ARTCSR register. It indicates the transition of the counter from FFh to the ARTARR value.
0: New transition not yet reached
1: Transition reached

COUNTER ACCESS REGISTER (ARTCAR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
CA7	CA6	CA5	CA4	CA3	CA2	CA1	CA0

Bit 7:0 = **CA[7:0]** *Counter Access Data*

These bits can be set and cleared either by hardware or by software. The ARTCAR register is used to read or write the auto-reload counter "on the fly" (while it is counting).

AUTO-RELOAD REGISTER (ARTARR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0

Bit 7:0 = **AR[7:0]** *Counter Auto-Reload Data*

These bits are set and cleared by software. They are used to hold the auto-reload value which is automatically loaded in the counter when an overflow occurs. At the same time, the PWM output levels are changed according to the corresponding OPx bit in the PWMCR register.

This register has two PWM management functions:

– Adjusting the PWM frequency
– Setting the PWM duty cycle resolution

PWM Frequency vs Resolution:

ARTARR value	Resolution	f_{PWM}	
		Min	Max
0	8-bit	~0.244 kHz	31.25 kHz
[0..127]	> 7-bit	~0.244 kHz	62.5 kHz
[128..191]	> 6-bit	~0.488 kHz	125 kHz
[192..223]	> 5-bit	~0.977 kHz	250 kHz
[224..239]	> 4-bit	~1.953 kHz	500 kHz

ON-CHIP PERIPHERALS (Cont'd)

PWM CONTROL REGISTER (PWMCR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
OE3	OE2	OE1	OE0	OP3	OP2	OP1	OP0

Bit 7:4 = **OE[3:0]** *PWM Output Enable*
These bits are set and cleared by software. They enable or disable the PWM output channels independently acting on the corresponding I/O pin.
0: PWM output disabled.
1: PWM output enabled.

Bit 3:0 = **OP[3:0]** *PWM Output Polarity*
These bits are set and cleared by software. They independently select the polarity of the four PWM output signals.

PWMx output level		OPx
Counter <= OCRx	Counter > OCRx	
1	0	0
0	1	1

Note: When an OPx bit is modified, the PWMx output signal polarity is immediately reversed.

DUTY CYCLE REGISTERS (PWMDCRx)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0

Bit 7:0 = **DC[7:0]** *Duty Cycle Data*

These bits are set and cleared by software.

A PWMDCRx register is associated with the OCRx register of each PWM channel to determine the second edge location of the PWM signal (the first edge location is common to all channels and given by the ARTARR register). These PWMDCR registers allow the duty cycle to be set independently for each PWM channel.

ON-CHIP PERIPHERALS (Cont'd)

INPUT CAPTURE CONTROL / STATUS REGISTER (ARTICCSR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
0	0	CS2	CS1	CIE2	CIE1	CF2	CF1

Bit 7:6 = Reserved, always read as 0.

Bit 5:4 = **CS[2:1]** *Capture Sensitivity*
These bits are set and cleared by software. They determine the trigger event polarity on the corresponding input capture channel.
0: Falling edge triggers capture on channel x.
1: Rising edge triggers capture on channel x.

Bit 3:2 = **CIE[2:1]** *Capture Interrupt Enable*
These bits are set and cleared by software. They enable or disable the Input capture channel interrupts independently.
0: Input capture channel x interrupt disabled.
1: Input capture channel x interrupt enabled.

Bit 1:0 = **CF[2:1]** *Capture Flag*
These bits are set by hardware and cleared by software reading the corresponding ARTICRx register. Each CFx bit indicates that an input capture x has occurred.
0: No input capture on channel x.
1: An input capture has occurred on channel x.

INPUT CAPTURE REGISTERS (ARTICRx)

Read only

Reset Value: 0000 0000 (00h)

7							0
IC7	IC6	IC5	IC4	IC3	IC2	IC1	IC0

Bit 7:0 = **IC[7:0]** *Input Capture Data*

These read only bits are set and cleared by hardware. An ARTICRx register contains the 8-bit auto-reload counter value transferred by the input capture channel x event.

PWM AUTO-RELOAD TIMER (Cont'd)

Table 17. PWM Auto-Reload Timer Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0031h	**PWMDCR3**	DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0
	Reset Value	0	0	0	0	0	0	0	0
0032h	**PWMDCR2**	DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0
	Reset Value	0	0	0	0	0	0	0	0
0033h	**PWMDCR1**	DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0
	Reset Value	0	0	0	0	0	0	0	0
0034h	**PWMDCR0**	DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0
	Reset Value	0	0	0	0	0	0	0	0
0035h	**PWMCR**	OE3	OE2	OE1	OE0	OP3	OP2	OP1	OP0
	Reset Value	0	0	0	0	0	0	0	0
0036h	**ARTCSR**	EXCL	CC2	CC1	CC0	TCE	FCRL	RIE	OVF
	Reset Value	0	0	0	0	0	0	0	0
0037h	**ARTCAR**	CA7	CA6	CA5	CA4	CA3	CA2	CA1	CA0
	Reset Value	0	0	0	0	0	0	0	0
0038h	**ARTARR**	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
	Reset Value	0	0	0	0	0	0	0	0
0039h	**ARTICCSR**			CE2	CE1	CS2	CS1	CF2	CF1
	Reset Value	0	0	0	0	0	0	0	0
003Ah	**ARTICR1**	IC7	IC6	IC5	IC4	IC3	IC2	IC1	IC0
	Reset Value	0	0	0	0	0	0	0	0
003Bh	**ARTICR2**	IC7	IC6	IC5	IC4	IC3	IC2	IC1	IC0
	Reset Value	0	0	0	0	0	0	0	0

10.4 16-BIT TIMER

10.4.1 Introduction

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

It may be used for a variety of purposes, including pulse length measurement of up to two input signals (*input capture*) or generation of up to two output waveforms (*output compare* and *PWM*).

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

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

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

10.4.2 Main Features

- Programmable prescaler: f_{CPU} divided by 2, 4 or 8
- Overflow status flag and maskable interrupt
- External clock input (must be at least four times slower than the CPU clock speed) with the choice of active edge
- 1 or 2 Output Compare functions each with:
  - 2 dedicated 16-bit registers
  - 2 dedicated programmable signals
  - 2 dedicated status flags
  - 1 dedicated maskable interrupt
- 1 or 2 Input Capture functions each with:
  - 2 dedicated 16-bit registers
  - 2 dedicated active edge selection signals
  - 2 dedicated status flags
  - 1 dedicated maskable interrupt
- Pulse width modulation mode (PWM)
- One Pulse mode
- Reduced Power Mode
- 5 alternate functions on I/O ports (ICAP1, ICAP2, OCMP1, OCMP2, EXTCLK)*

The Block Diagram is shown in Figure 48.

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

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

10.4.3 Functional Description

10.4.3.1 Counter

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

Counter Register (CR):

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

Alternate Counter Register (ACR)

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

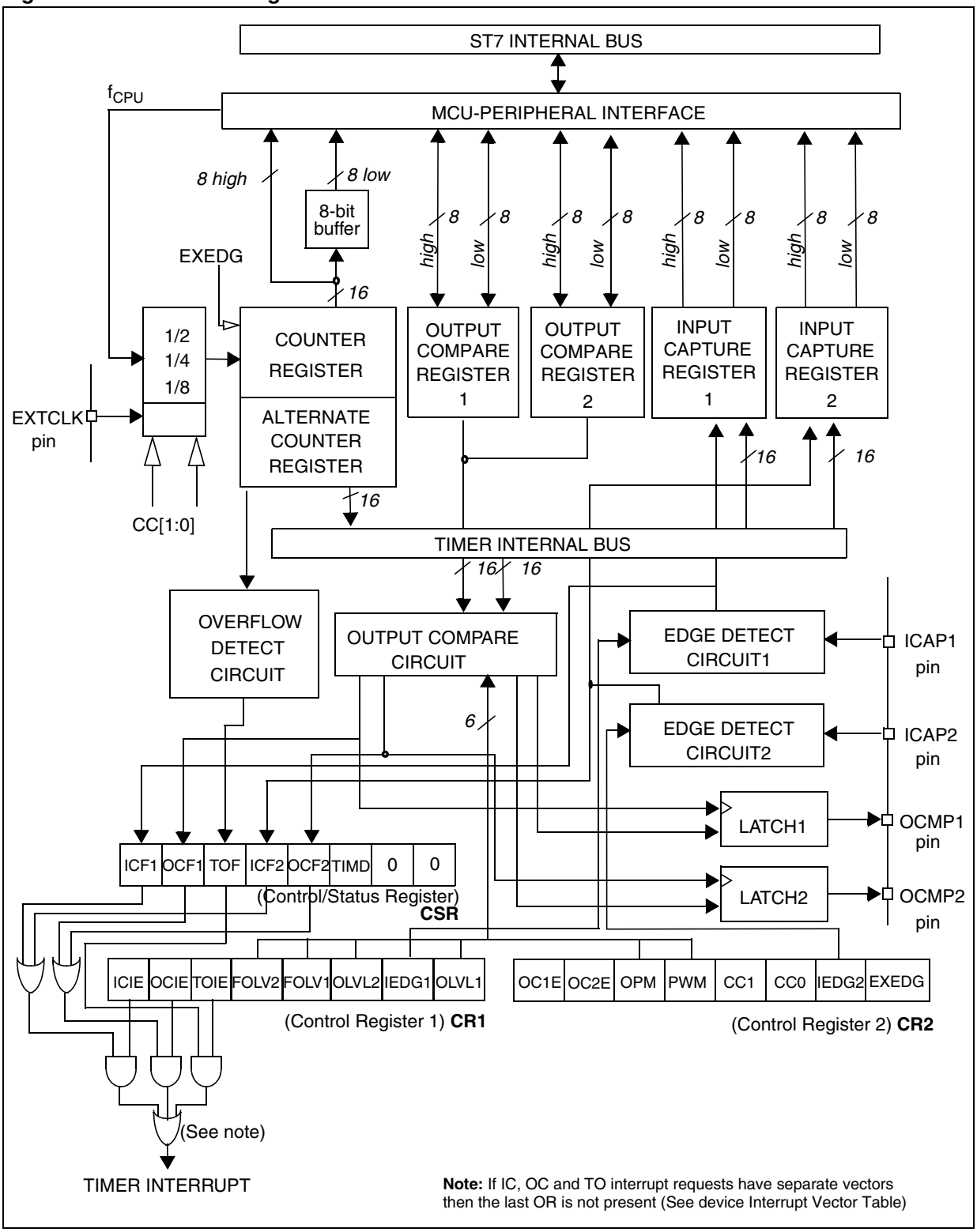
These two read-only 16-bit registers contain the same value but with the difference that reading the ACLR register does not clear the TOF bit (Timer overflow flag), located in the Status register, (SR), (see note at the end of paragraph titled 16-bit read sequence).

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

The timer clock depends on the clock control bits of the CR2 register, as illustrated in Table 17 Clock Control Bits. The value in the counter register repeats every 131072, 262144 or 524288 CPU clock cycles depending on the CC[1:0] bits.
The timer frequency can be $f_{CPU}/2$, $f_{CPU}/4$, $f_{CPU}/8$ or an external frequency.

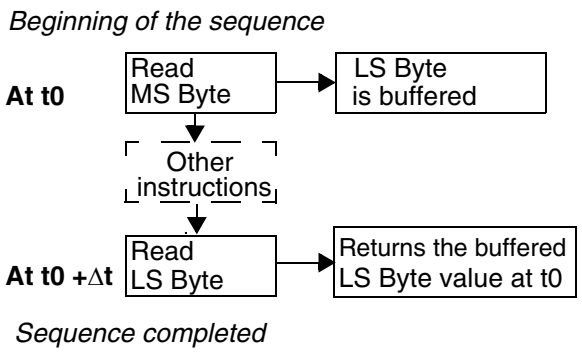
16-BIT TIMER (Cont'd)

Figure 48. Timer Block Diagram



16-BIT TIMER (Cont'd)

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



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

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

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

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

– The TOF bit of the SR register is set.

– A timer interrupt is generated if:

  – TOIE bit of the CR1 register is set and

  – I bit of the CC register is cleared.

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

Clearing the overflow interrupt request is done in two steps:

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

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

The timer is not affected by WAIT mode.

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

10.4.3.2 External Clock

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

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

The counter is synchronized with the falling edge of the internal CPU clock.

A minimum of four falling edges of the CPU clock must occur between two consecutive active edges of the external clock; thus the external clock frequency must be less than a quarter of the CPU clock frequency.

16-BIT TIMER (Cont'd)

Figure 49. Counter Timing Diagram, Internal Clock Divided by 2

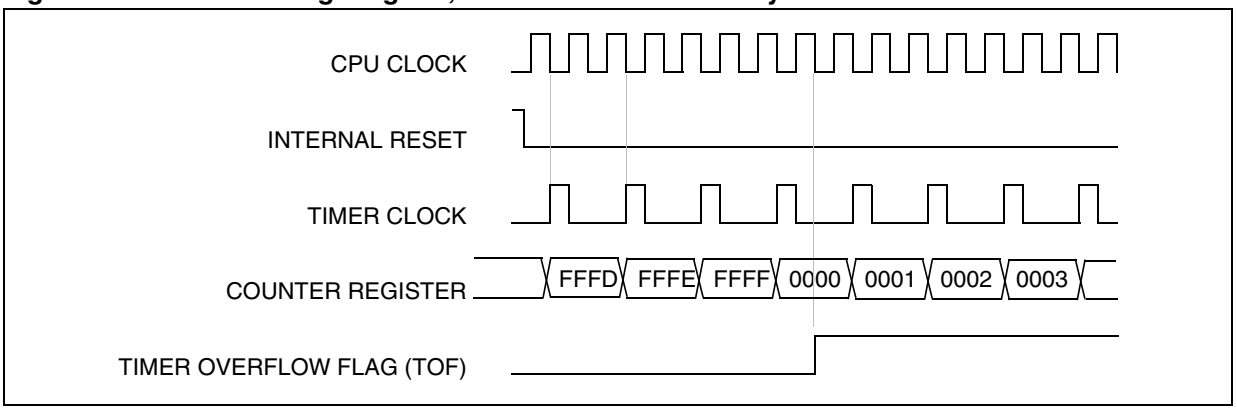


Figure 50. Counter Timing Diagram, Internal Clock Divided by 4

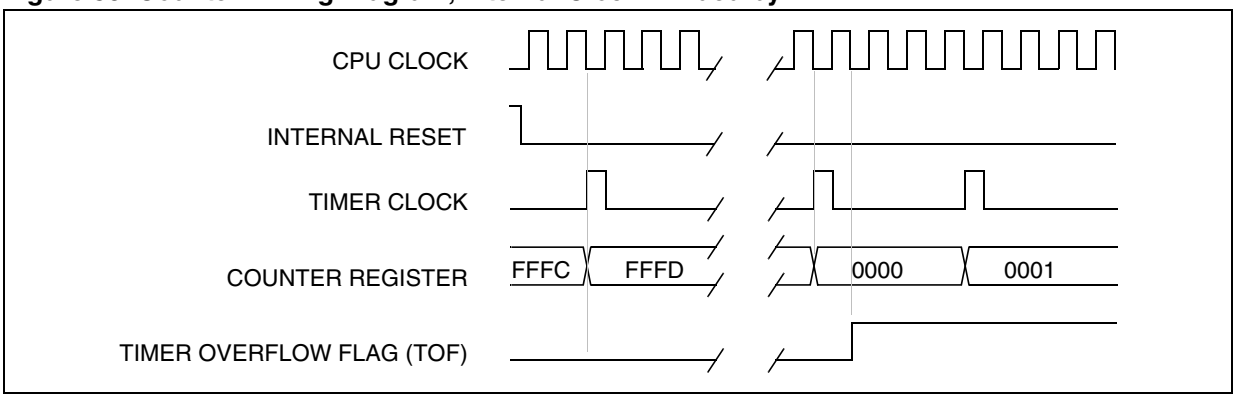
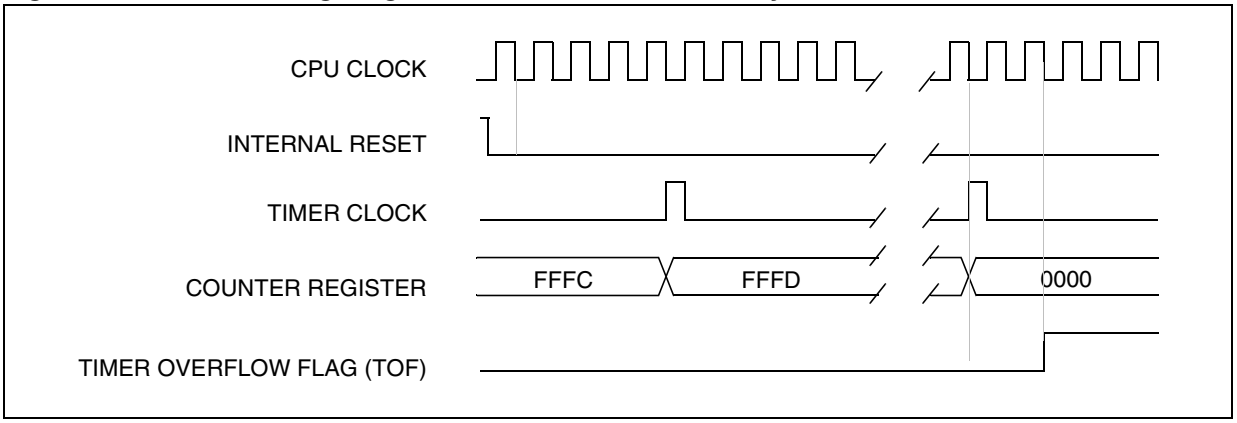


Figure 51. Counter Timing Diagram, Internal Clock Divided By 8



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

16-BIT TIMER (Cont'd)

10.4.3.3 Input Capture

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

The two 16-bit input capture registers (IC1R and IC2R) are used to latch the value of the free running counter after a transition is detected on the ICAP*i* pin (see Figure 52).

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

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

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

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

Procedure:

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

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

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

And select the following in the CR1 register:

– Set the ICIE bit to generate an interrupt after an input capture coming from either the ICAP1 pin or the ICAP2 pin

– Select the edge of the active transition on the ICAP1 pin with the IEDG1 bit (the ICAP1pin must be configured as floating input or input with pull-up without interrupt if this configuration is available).

When an input capture occurs:

– ICF*i* bit is set.

– The IC*i*R register contains the value of the free running counter on the active transition on the ICAP*i* pin (see Figure 53).

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

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

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

Notes:

1. After reading the IC*i*HR register, transfer of input capture data is inhibited and ICF*i* will never be set until the IC*i*LR register is also read.
2. The IC*i*R register contains the free running counter value which corresponds to the most recent input capture.
3. The two input capture functions can be used together even if the timer also uses the two output compare functions.
4. In One Pulse mode and PWM mode only Input Capture 2 can be used.
5. The alternate inputs (ICAP1 and ICAP2) are always directly connected to the timer. So any transitions on these pins activates the input capture function.
   Moreover if one of the ICAP*i* pins is configured as an input and the second one as an output, an interrupt can be generated if the user toggles the output pin and if the ICIE bit is set.
   This can be avoided if the input capture function *i* is disabled by reading the IC*i*HR (see note 1).
6. The TOF bit can be used with interrupt generation in order to measure events that go beyond the timer range (FFFFh).

16-BIT TIMER (Cont'd)

Figure 52. Input Capture Block Diagram

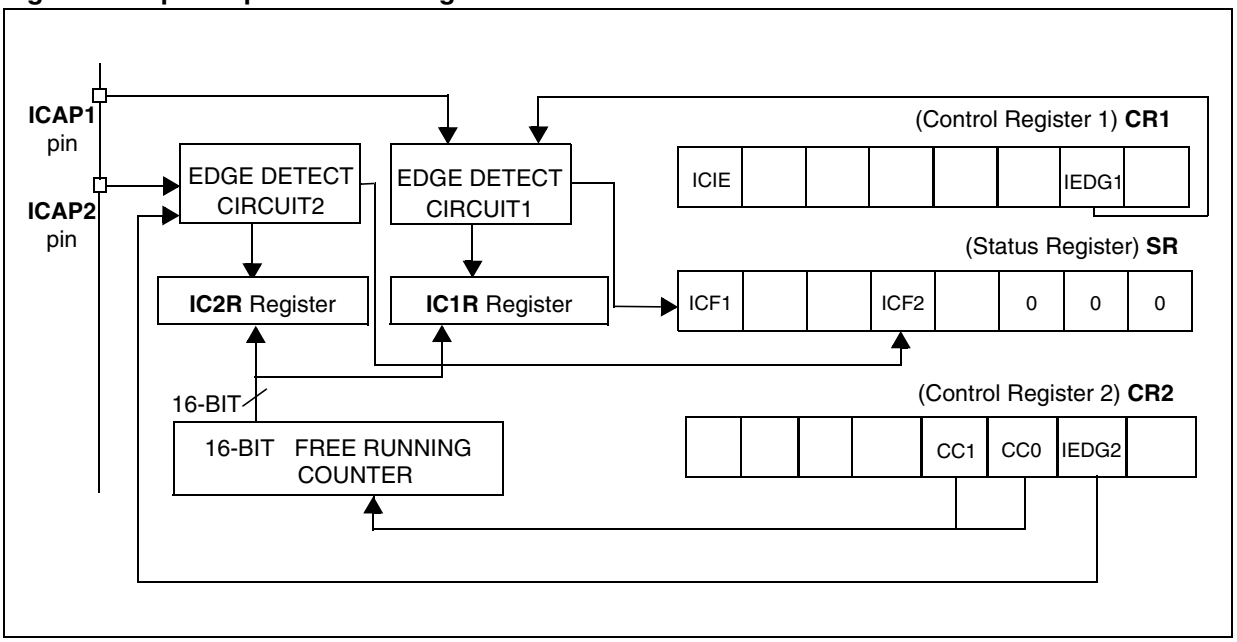
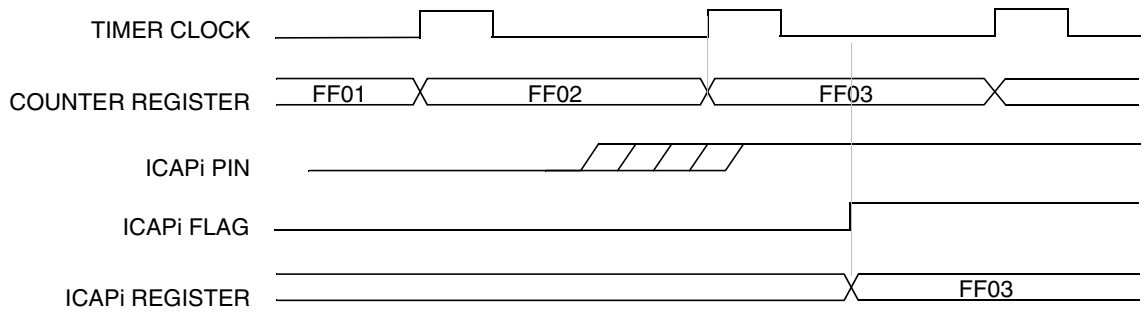


Figure 53. Input Capture Timing Diagram



Note: The rising edge is the active edge.

16-BIT TIMER (Cont'd)

10.4.3.4 Output Compare

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

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

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

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

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

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

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

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

Procedure:

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

- Set the OC*i*E bit if an output is needed then the OCMP*i* pin is dedicated to the output compare *i* signal.
- Select the timer clock (CC[1:0]) (see Table 17 Clock Control Bits).

And select the following in the CR1 register:

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

When a match is found between OC*i*R register and CR register:

- OCF*i* bit is set.
- The OCMP*i* pin takes OLVL*i* bit value (OCMP*i* pin latch is forced low during reset).
- A timer interrupt is generated if the OCIE bit is set in the CR1 register and the I bit is cleared in the CC register (CC).

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

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

Where:

Δt = Output compare period (in seconds)

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

PRESC = Timer prescaler factor (2, 4 or 8 depending on CC[1:0] bits, see Table 17 Clock Control Bits)

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

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

Where:

Δt = Output compare period (in seconds)

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

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

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

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

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

16-BIT TIMER (Cont'd)

Notes:

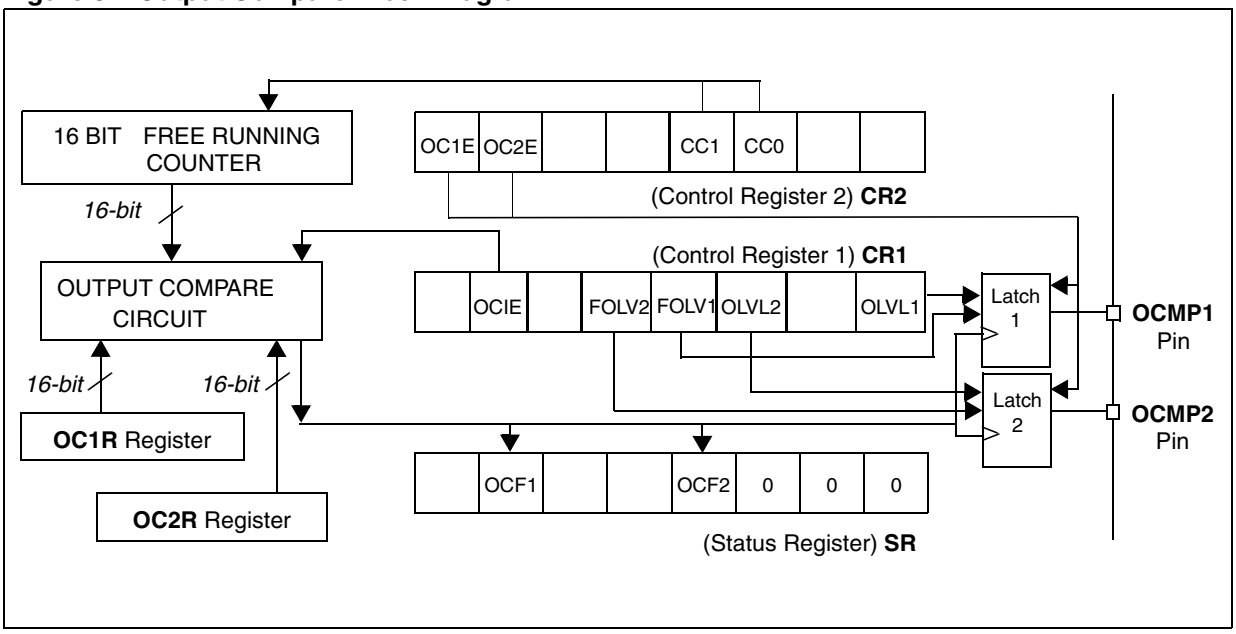
1. After a processor write cycle to the OC*i*HR register, the output compare function is inhibited until the OC*i*LR register is also written.
2. If the OC*i*E bit is not set, the OCMP*i* pin is a general I/O port and the OLVL*i* bit will not appear when a match is found but an interrupt could be generated if the OCIE bit is set.
3. In both internal and external clock modes, OCFi and OCMPi are set while the counter value equals the OCiR register value (see Figure 55 on page 80 for an example with $f_{CPU}/2$ and Figure 56 on page 80 for an example with $f_{CPU}/4$). This behavior is the same in OPM or PWM mode.
4. The output compare functions can be used both for generating external events on the OCMP*i* pins even if the input capture mode is also used.
5. The value in the 16-bit OC*i*R register and the OLV*i* bit should be changed after each successful comparison in order to control an output waveform or establish a new elapsed timeout.

Forced Compare Output capability

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

The FOLVL*i* bits have no effect in both One Pulse mode and PWM mode.

Figure 54. Output Compare Block Diagram



16-BIT TIMER (Cont'd)

Figure 55. Output Compare Timing Diagram, $f_{TIMER} = f_{CPU}/2$

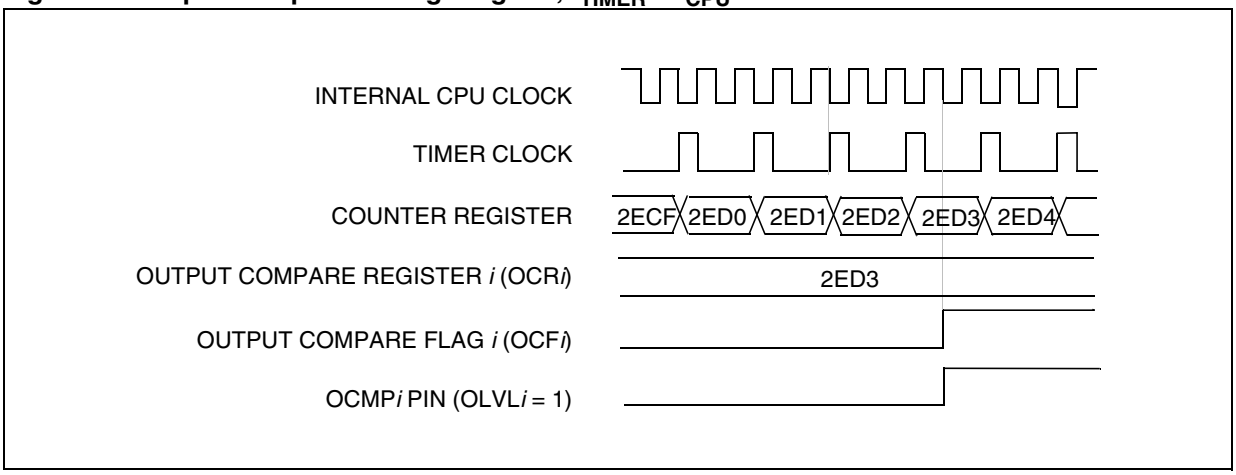
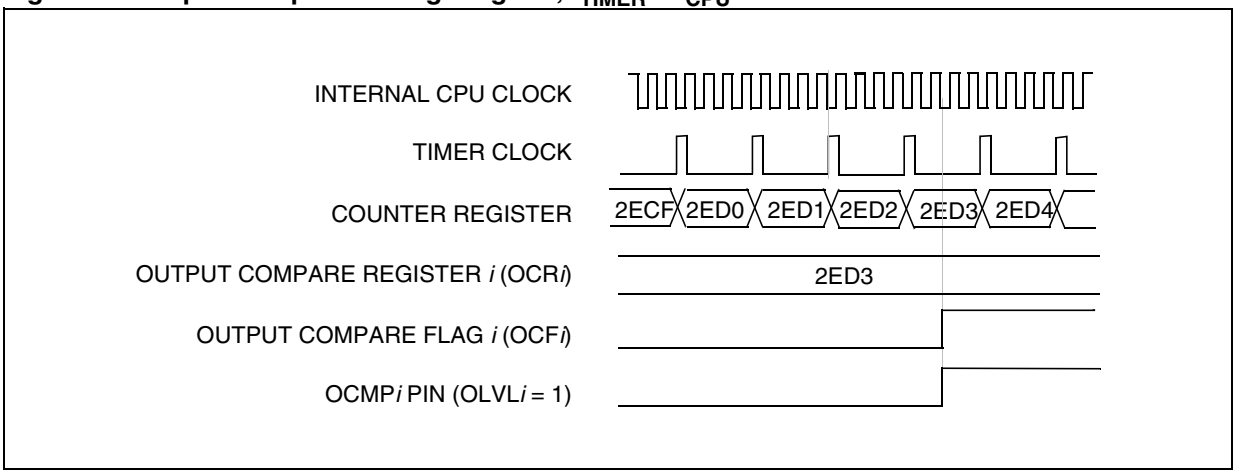


Figure 56. Output Compare Timing Diagram, $f_{TIMER} = f_{CPU}/4$



16-BIT TIMER (Cont'd)

10.4.3.5 One Pulse Mode

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

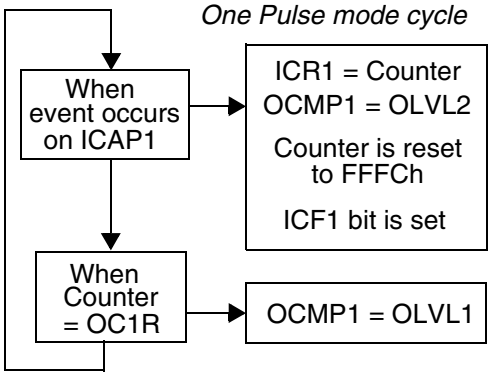
The One Pulse mode uses the Input Capture1 function and the Output Compare1 function.

Procedure:

To use One Pulse mode:

1. Load the OC1R register with the value corresponding to the length of the pulse (see the formula in the opposite column).
2. Select the following in the CR1 register:
   - Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after the pulse.
   - Using the OLVL2 bit, select the level to be applied to the OCMP1 pin during the pulse.
   - Select the edge of the active transition on the ICAP1 pin with the IEDG1 bit (the ICAP1 pin must be configured as floating input).
3. Select the following in the CR2 register:
   - Set the OC1E bit, the OCMP1 pin is then dedicated to the Output Compare 1 function.
   - Set the OPM bit.
   - Select the timer clock CC[1:0] (see Table 17 Clock Control Bits).

One Pulse mode cycle



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

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

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

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

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

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

Where:

t = Pulse period (in seconds)

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

PRESC = Timer prescaler factor (2, 4 or 8 depending on the CC[1:0] bits, see Table 17 Clock Control Bits)

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

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

Where:

t = Pulse period (in seconds)

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

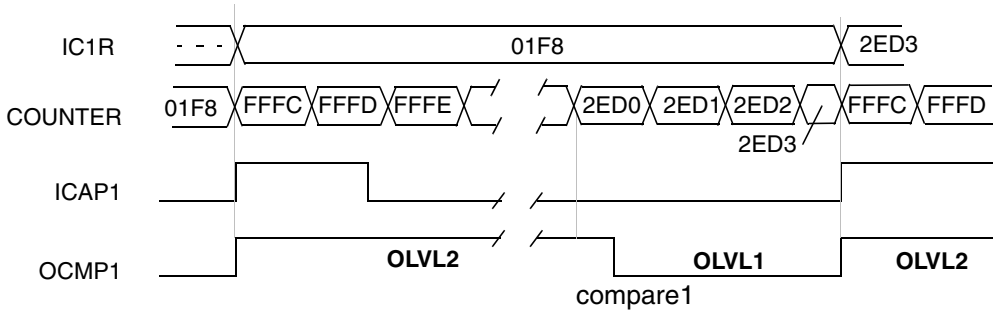
When the value of the counter is equal to the value of the contents of the OC1R register, the OLVL1 bit is output on the OCMP1 pin, (See Figure 57).

Notes:

1. The OCF1 bit cannot be set by hardware in One Pulse mode but the OCF2 bit can generate an Output Compare interrupt.
2. When the Pulse Width Modulation (PWM) and One Pulse mode (OPM) bits are both set, the PWM mode is the only active one.
3. If OLVL1 = OLVL2 a continuous signal will be seen on the OCMP1 pin.
4. The ICAP1 pin can not be used to perform input capture. The ICAP2 pin can be used to perform input capture (ICF2 can be set and IC2R can be loaded) but the user must take care that the counter is reset each time a valid edge occurs on the ICAP1 pin and ICF1 can also generates interrupt if ICIE is set.
5. When One Pulse mode is used OC1R is dedicated to this mode. Nevertheless OC2R and OCF2 can be used to indicate a period of time has been elapsed but cannot generate an output waveform because the level OLVL2 is dedicated to the One Pulse mode.

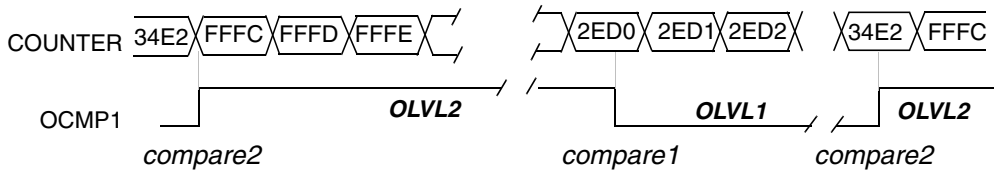
16-BIT TIMER (Cont'd)

Figure 57. One Pulse Mode Timing Example



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

Figure 58. Pulse Width Modulation Mode Timing Example with 2 Output Compare Functions



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

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

16-BIT TIMER (Cont'd)

10.4.3.6 Pulse Width Modulation Mode

Pulse Width Modulation (PWM) mode enables the generation of a signal with a frequency and pulse length determined by the value of the OC1R and OC2R registers.

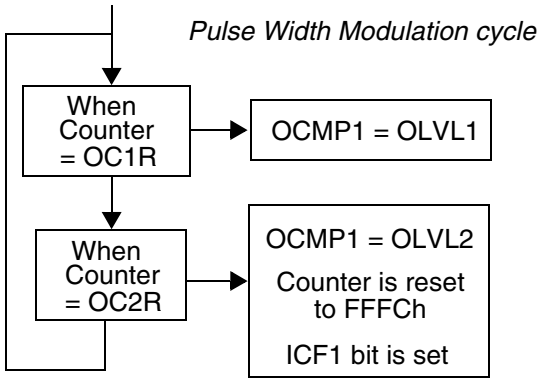
Pulse Width Modulation mode uses the complete Output Compare 1 function plus the OC2R register, and so this functionality can not be used when PWM mode is activated.

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

Procedure

To use Pulse Width Modulation mode:

1. Load the OC2R register with the value corresponding to the period of the signal using the formula in the opposite column.
2. Load the OC1R register with the value corresponding to the period of the pulse if (OLVL1 = 0 and OLVL2 = 1) using the formula in the opposite column.
3. Select the following in the CR1 register:
   - Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after a successful comparison with the OC1R register.
   - Using the OLVL2 bit, select the level to be applied to the OCMP1 pin after a successful comparison with the OC2R register.
4. Select the following in the CR2 register:
   - Set OC1E bit: the OCMP1 pin is then dedicated to the output compare 1 function.
   - Set the PWM bit.
   - Select the timer clock (CC[1:0]) (see Table 17 Clock Control Bits).



If OLVL1 = 1 and OLVL2 = 0 the length of the positive pulse is the difference between the OC2R and OC1R registers.

If OLVL1 = OLVL2 a continuous signal will be seen on the OCMP1 pin.

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

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

Where:

t = Signal or pulse period (in seconds)

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

PRESC = Timer prescaler factor (2, 4 or 8 depending on CC[1:0] bits, see Table 17 Clock Control Bits)

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

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

Where:

t = Signal or pulse period (in seconds)

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

The Output Compare 2 event causes the counter to be initialized to FFFCh (See Figure 58)

Notes:

1. After a write instruction to the OC*i*HR register, the output compare function is inhibited until the OC*i*LR register is also written.
2. The OCF1 and OCF2 bits cannot be set by hardware in PWM mode therefore the Output Compare interrupt is inhibited.
3. The ICF1 bit is set by hardware when the counter reaches the OC2R value and can produce a timer interrupt if the ICIE bit is set and the I bit is cleared.
4. In PWM mode the ICAP1 pin can not be used to perform input capture because it is disconnected to the timer. The ICAP2 pin can be used to perform input capture (ICF2 can be set and IC2R can be loaded) but the user must take care that the counter is reset each period and ICF1 can also generates interrupt if ICIE is set.
5. When the Pulse Width Modulation (PWM) and One Pulse mode (OPM) bits are both set, the PWM mode is the only active one.

16-BIT TIMER (Cont'd)

10.4.4 Low Power Modes

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

10.4.5 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
Input Capture 1 event/Counter reset in PWM mode	ICF1	ICIE	Yes	No
Input Capture 2 event	ICF2			
Output Compare 1 event (not available in PWM mode)	OCF1	OCIE		
Output Compare 2 event (not available in PWM mode)	OCF2			
Timer Overflow event	TOF	TOIE		

Note: The 16-bit Timer interrupt events are connected to the same interrupt vector (see Interrupts chapter). These events generate an interrupt if the corresponding Enable Control Bit is set and the interrupt mask in the CC register is reset (RIM instruction).

10.4.6 Summary of Timer Modes

MODES	TIMER RESOURCES			
	Input Capture 1	Input Capture 2	Output Compare 1	Output Compare 2
Input Capture (1 and/or 2)	Yes	Yes	Yes	Yes
Output Compare (1 and/or 2)				
One Pulse Mode	No	Not Recommended[1)]	No	Partially [2)]
PWM Mode		Not Recommended[3)]		No

1) See note 4 in Section 10.4.3.5 "One Pulse Mode"

2) See note 5 in Section 10.4.3.5 "One Pulse Mode"

3) See note 4 in Section 10.4.3.6 "Pulse Width Modulation Mode"

16-BIT TIMER (Cont'd)

10.4.7 Register Description

Each Timer is associated with three control and status registers, and with six pairs of data registers (16-bit values) relating to the two input captures, the two output compares, the counter and the alternate counter.

CONTROL REGISTER 1 (CR1)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
ICIE	OCIE	TOIE	FOLV2	FOLV1	OLVL2	IEDG1	OLVL1

Bit 7 = **ICIE** *Input Capture Interrupt Enable.*
0: Interrupt is inhibited.
1: A timer interrupt is generated whenever the ICF1 or ICF2 bit of the SR register is set.

Bit 6 = **OCIE** *Output Compare Interrupt Enable.*
0: Interrupt is inhibited.
1: A timer interrupt is generated whenever the OCF1 or OCF2 bit of the SR register is set.

Bit 5 = **TOIE** *Timer Overflow Interrupt Enable.*
0: Interrupt is inhibited.
1: A timer interrupt is enabled whenever the TOF bit of the SR register is set.

Bit 4 = **FOLV2** *Forced Output Compare 2.*
This bit is set and cleared by software.
0: No effect on the OCMP2 pin.
1: Forces the OLVL2 bit to be copied to the OCMP2 pin, if the OC2E bit is set and even if there is no successful comparison.

Bit 3 = **FOLV1** *Forced Output Compare 1.*
This bit is set and cleared by software.
0: No effect on the OCMP1 pin.
1: Forces OLVL1 to be copied to the OCMP1 pin, if the OC1E bit is set and even if there is no successful comparison.

Bit 2 = **OLVL2** *Output Level 2.*
This bit is copied to the OCMP2 pin whenever a successful comparison occurs with the OC2R register and OCxE is set in the CR2 register. This value is copied to the OCMP1 pin in One Pulse mode and Pulse Width Modulation mode.

Bit 1 = **IEDG1** *Input Edge 1.*
This bit determines which type of level transition on the ICAP1 pin will trigger the capture.
0: A falling edge triggers the capture.
1: A rising edge triggers the capture.

Bit 0 = **OLVL1** *Output Level 1.*
The OLVL1 bit is copied to the OCMP1 pin whenever a successful comparison occurs with the OC1R register and the OC1E bit is set in the CR2 register.

16-BIT TIMER (Cont'd)

CONTROL REGISTER 2 (CR2)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
OC1E	OC2E	OPM	PWM	CC1	CC0	IEDG2	EXEDG

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

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

Bit 5 = **OPM** *One Pulse Mode.*
0: One Pulse mode is not active.
1: One Pulse mode is active, the ICAP1 pin can be used to trigger one pulse on the OCMP1 pin; the active transition is given by the IEDG1 bit. The length of the generated pulse depends on the contents of the OC1R register.

Bit 4 = **PWM** *Pulse Width Modulation.*
0: PWM mode is not active.
1: PWM mode is active, the OCMP1 pin outputs a programmable cyclic signal; the length of the pulse depends on the value of OC1R register; the period depends on the value of OC2R register.

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

The timer clock mode depends on these bits:

Table 18. Clock Control Bits

Timer Clock	CC1	CC0
f_{CPU} / 4	0	0
f_{CPU} / 2		1
f_{CPU} / 8	1	0
External Clock (where available)		1

Note: If the external clock pin is not available, programming the external clock configuration stops the counter.

Bit 1 = **IEDG2** *Input Edge 2.*
This bit determines which type of level transition on the ICAP2 pin will trigger the capture.
0: A falling edge triggers the capture.
1: A rising edge triggers the capture.

Bit 0 = **EXEDG** *External Clock Edge.*
This bit determines which type of level transition on the external clock pin EXTCLK will trigger the counter register.
0: A falling edge triggers the counter register.
1: A rising edge triggers the counter register.

16-BIT TIMER (Cont'd)

CONTROL/STATUS REGISTER (CSR)

Read/Write (bits 7:3 read only)

Reset Value: xxxx x0xx (xxh)

7							0
ICF1	OCF1	TOF	ICF2	OCF2	TIMD	0	0

Bit 7 = **ICF1** *Input Capture Flag 1.*
0: No input capture (reset value).
1: An input capture has occurred on the ICAP1 pin or the counter has reached the OC2R value in PWM mode. To clear this bit, first read the SR register, then read or write the low byte of the IC1R (IC1LR) register.

Bit 6 = **OCF1** *Output Compare Flag 1.*
0: No match (reset value).
1: The content of the free running counter has matched the content of the OC1R register. To clear this bit, first read the SR register, then read or write the low byte of the OC1R (OC1LR) register.

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

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

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

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

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

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

16-BIT TIMER (Cont'd)

INPUT CAPTURE 1 HIGH REGISTER (IC1HR)

Read Only
Reset Value: Undefined

This is an 8-bit read only register that contains the high part of the counter value (transferred by the input capture 1 event).

7							0
MSB							LSB

INPUT CAPTURE 1 LOW REGISTER (IC1LR)

Read Only
Reset Value: Undefined

This is an 8-bit read only register that contains the low part of the counter value (transferred by the input capture 1 event).

7							0
MSB							LSB

OUTPUT COMPARE 1 HIGH REGISTER (OC1HR)

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

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

7							0
MSB							LSB

OUTPUT COMPARE 1 LOW REGISTER (OC1LR)

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

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

7							0
MSB							LSB

16-BIT TIMER (Cont'd)

OUTPUT COMPARE 2 HIGH REGISTER (OC2HR)

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

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

7							0
MSB							LSB

OUTPUT COMPARE 2 LOW REGISTER (OC2LR)

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

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

7							0
MSB							LSB

COUNTER HIGH REGISTER (CHR)

Read Only
Reset Value: 1111 1111 (FFh)

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

7							0
MSB							LSB

COUNTER LOW REGISTER (CLR)

Read Only
Reset Value: 1111 1100 (FCh)

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

7							0
MSB							LSB

ALTERNATE COUNTER HIGH REGISTER (ACHR)

Read Only
Reset Value: 1111 1111 (FFh)

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

7							0
MSB							LSB

ALTERNATE COUNTER LOW REGISTER (ACLR)

Read Only
Reset Value: 1111 1100 (FCh)

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

7							0
MSB							LSB

INPUT CAPTURE 2 HIGH REGISTER (IC2HR)

Read Only
Reset Value: Undefined

This is an 8-bit read only register that contains the high part of the counter value (transferred by the Input Capture 2 event).

7							0
MSB							LSB

INPUT CAPTURE 2 LOW REGISTER (IC2LR)

Read Only
Reset Value: Undefined

This is an 8-bit read only register that contains the low part of the counter value (transferred by the Input Capture 2 event).

7							0
MSB							LSB

16-BIT TIMER (Cont'd)

Table 19. 16-Bit Timer Register Map

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
51	**CR2**	OC1E	OC2E	OPM	PWM	CC1	CC0	IEDG2	EXEDG
52	**CR1**	ICIE	OCIE	TOIE	FOLV2	FOLV1	OLVL2	IEDG1	OLVL1
53	**CSR**	ICF1	OCF1	TOF	ICF2	OCF2	TIMD		
54	**IC1HR**	MSB							LSB
55	**IC1LR**	MSB							LSB
56	**OC1HR**	MSB							LSB
57	**OC1LR**	MSB							LSB
58	**CHR**	MSB							LSB
59	**CLR**	MSB							LSB
5A	**ACHR**	MSB							LSB
5B	**ACLR**	MSB							LSB
5C	**IC2HR**	MSB							LSB
5D	**IC2LR**	MSB							LSB
5E	**OC2HR**	MSB							LSB
5F	**OC2LR**	MSB							LSB

10.5 8-BIT TIMER (TIM8)

10.5.1 Introduction

The timer consists of a 8-bit free-running counter driven by a programmable prescaler.

It may be used for a variety of purposes, including pulse length measurement of up to two input signals (*input capture*) or generation of up to two output waveforms (*output compare* and *PWM*).

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

10.5.2 Main Features

- Programmable prescaler: f_{CPU} divided by 2, 4 , 8 or f_{OSC2} divided by 8000.
- Overflow status flag and maskable interrupt
- Output compare functions with
  - 2 dedicated 8-bit registers
  - 2 dedicated programmable signals
  - 2 dedicated status flags
  - 1 dedicated maskable interrupt
- Input capture functions with
  - 2 dedicated 8-bit registers
  - 2 dedicated active edge selection signals
  - 2 dedicated status flags
  - 1 dedicated maskable interrupt
- Pulse width modulation mode (PWM)
- One pulse mode
- Reduced Power Mode
- 4 alternate functions on I/O ports (ICAP1, ICAP2, OCMP1, OCMP2)*

The Block Diagram is shown in Figure 59.

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

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

10.5.3 Functional Description

10.5.3.1 Counter

The main block of the Programmable Timer is a 8-bit free running upcounter and its associated 8-bit registers.

These two read-only 8-bit registers contain the same value but with the difference that reading the ACTR register does not clear the TOF bit (Timer overflow flag), located in the Status register, (SR).

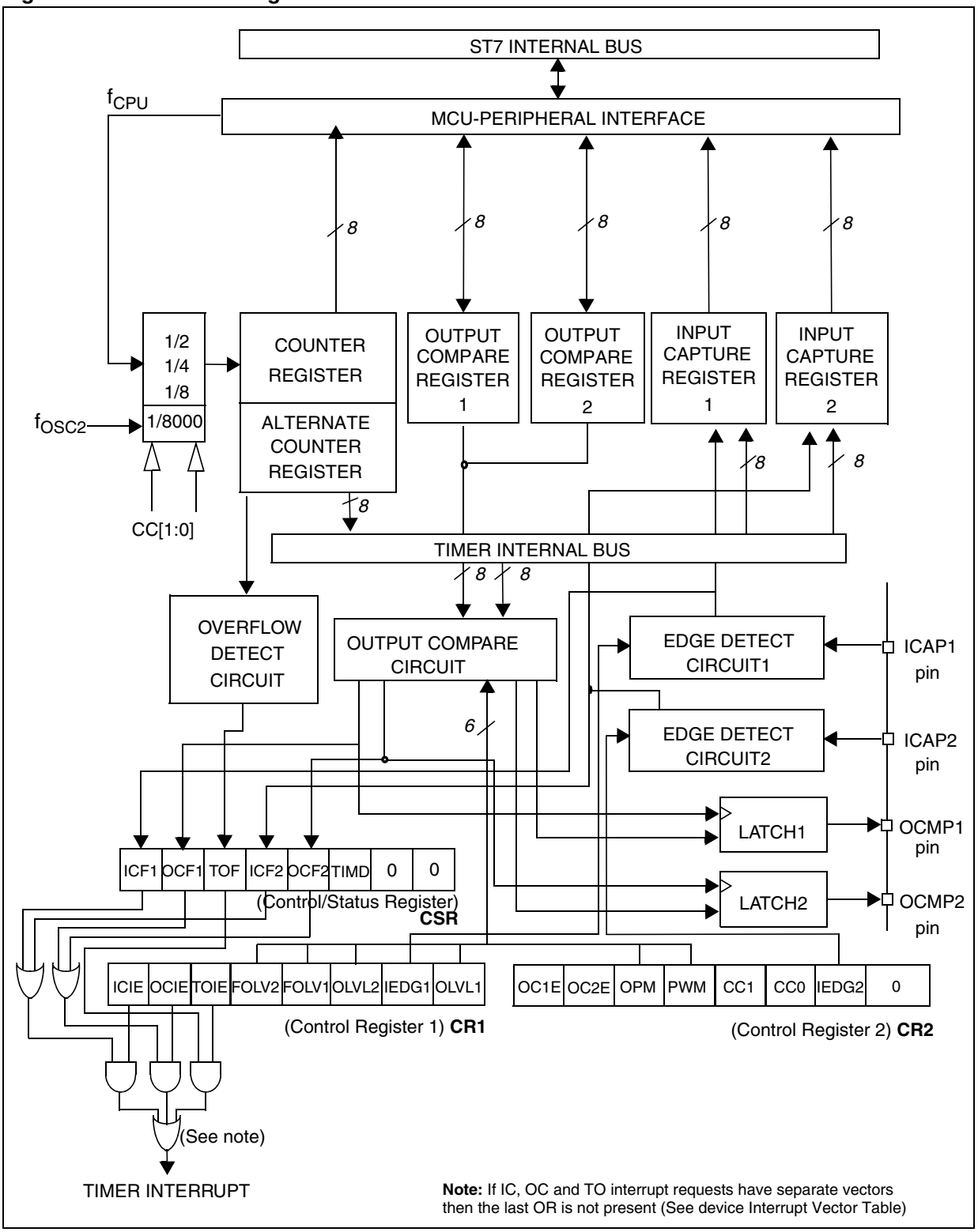
Writing in the CTR register or ACTR register resets the free running counter to the FCh value.
Both counters have a reset value of FCh (this is the only value which is reloaded in the 8-bit timer). The reset value of both counters is also FCh in One Pulse mode and PWM mode.

The timer clock depends on the clock control bits of the CR2 register, as shown in Table 19 Clock Control Bits. The value in the counter register repeats every 512, 1024, 2048 or 20480000 f_{CPU} clock cycles depending on the CC[1:0] bits.
The timer frequency can be $f_{CPU}/2$, $f_{CPU}/4$, $f_{CPU}/8$ or $f_{OSC2}/8000$.

For example, if $f_{OSC2}/8000$ is selected, and f_{OSC2} = 8 MHz, the timer frequency will be 1 ms. Refer to Table 19 on page 105.

8-BIT TIMER (Cont'd)

Figure 59. Timer Block Diagram



8-BIT TIMER (Cont'd)

Whatever the timer mode used (input capture, output compare, one pulse mode or PWM mode) an overflow occurs when the counter rolls over from FFh to 00h then:

– The TOF bit of the SR register is set.

– A timer interrupt is generated if:

  – TOIE bit of the CR1 register is set and

  – I bit of the CC register is cleared.

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

Clearing the overflow interrupt request is done in two steps:

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

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

The timer is not affected by WAIT mode.

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

8-BIT TIMER (Cont'd)

Figure 60. Counter Timing Diagram, Internal Clock Divided by 2

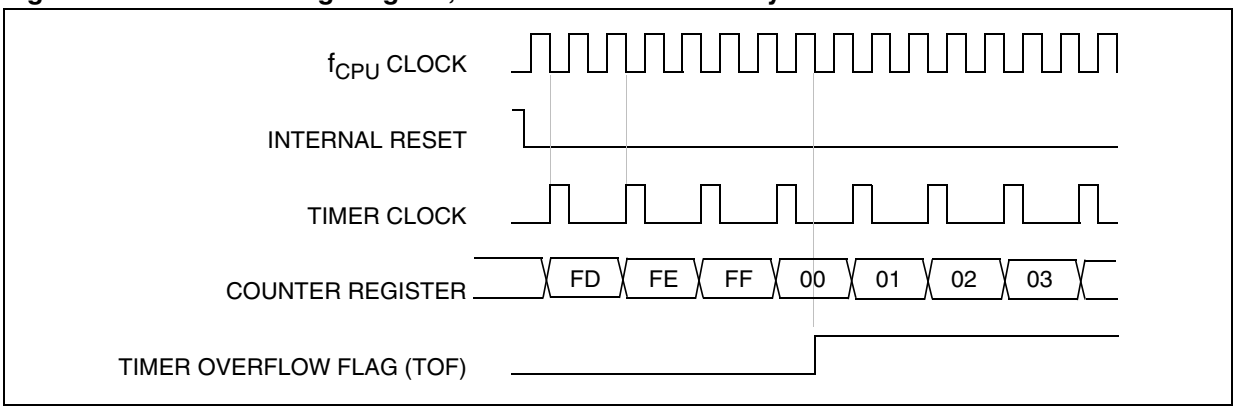


Figure 61. Counter Timing Diagram, Internal Clock Divided by 4

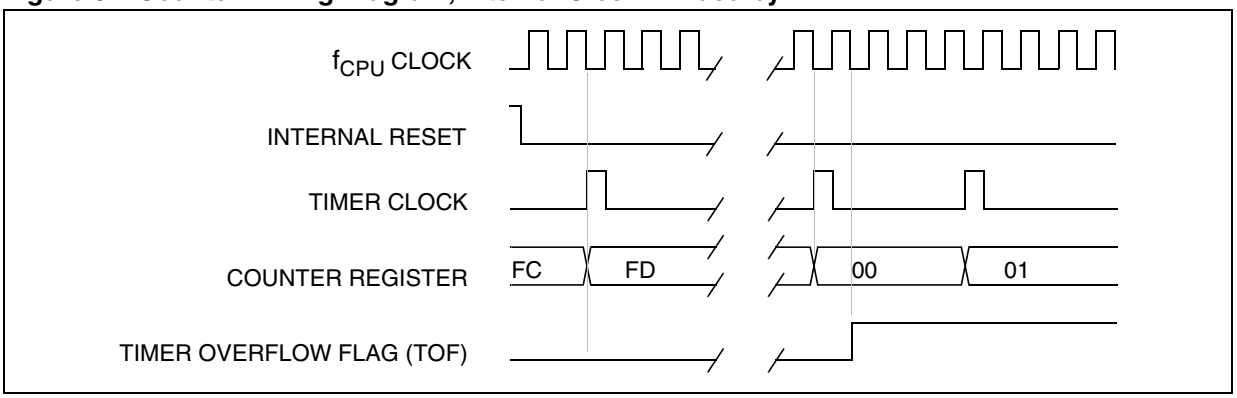
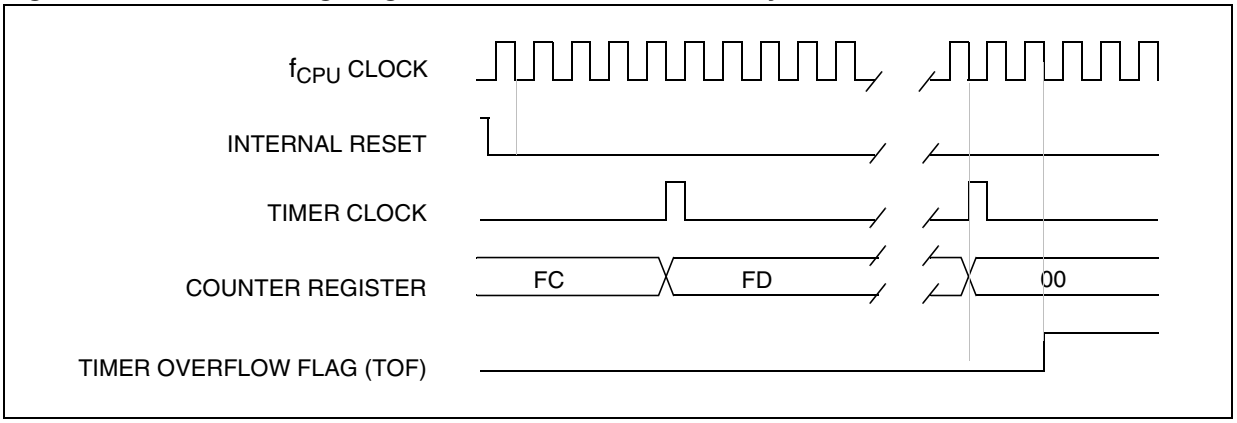


Figure 62. Counter Timing Diagram, Internal Clock Divided by 8



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

8-BIT TIMER (Cont'd)

10.5.3.2 Input Capture

In this section, the index, *i*, may be 1 or 2 because there are two input capture functions in the 8-bit timer.

The two 8-bit input capture registers (IC1R and IC2R) are used to latch the value of the free running counter after a transition is detected on the ICAP*i* pin (see Figure 63).

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

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

Timing resolution is one count of the free running counter (see Table 19 Clock Control Bits).

Procedure:

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

– Select the timer clock (CC[1:0]) (see Table 19 Clock Control Bits).

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

And select the following in the CR1 register:

– Set the ICIE bit to generate an interrupt after an input capture coming from either the ICAP1 pin or the ICAP2 pin

– Select the edge of the active transition on the ICAP1 pin with the IEDG1 bit (the ICAP1 pin must be configured as floating input or input with pull-up without interrupt if this configuration is available).

When an input capture occurs:

– ICF*i* bit is set.

– The IC*i*R register contains the value of the free running counter on the active transition on the ICAP*i* pin (see Figure 64).

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

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

1. Reading the SR register while the ICF*i* bit is set.

2. An access (read or write) to the IC*i*R register.

Notes:

1. The IC*i*R register contains the free running counter value which corresponds to the most recent input capture.

2. The two input capture functions can be used together even if the timer also uses the two output compare functions.

3. Once the ICIE bit is set both input capture features may trigger interrupt requests. If only one is needed in the application, the interrupt routine software needs to discard the unwanted capture interrupt. This can be done by checking the ICF1 and ICF2 flags and resetting them both.

4. In One pulse Mode and PWM mode only Input Capture 2 can be used.

5. The alternate inputs (ICAP1 and ICAP2) are always directly connected to the timer. So any transitions on these pins activates the input capture function.

Moreover if one of the ICAP*i* pins is configured as an input and the second one as an output, an interrupt can be generated if the user toggles the output pin and if the ICIE bit is set.

6. The TOF bit can be used with interrupt generation in order to measure events that go beyond the timer range (FFh).

8-BIT TIMER (Cont'd)

Figure 63. Input Capture Block Diagram

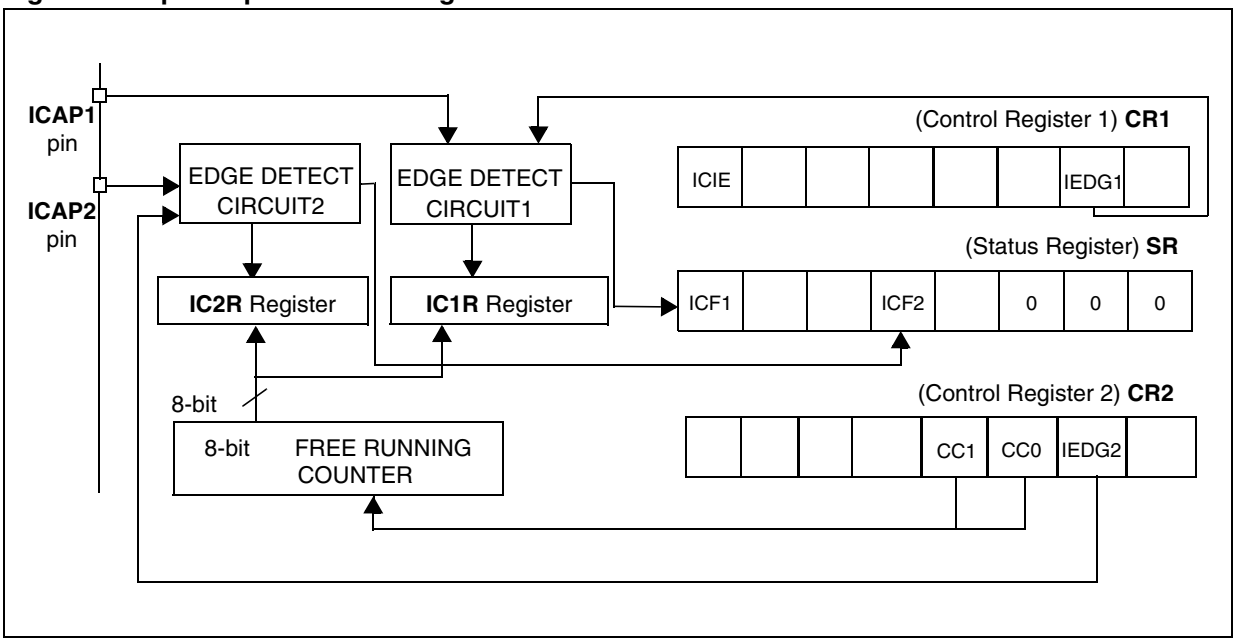
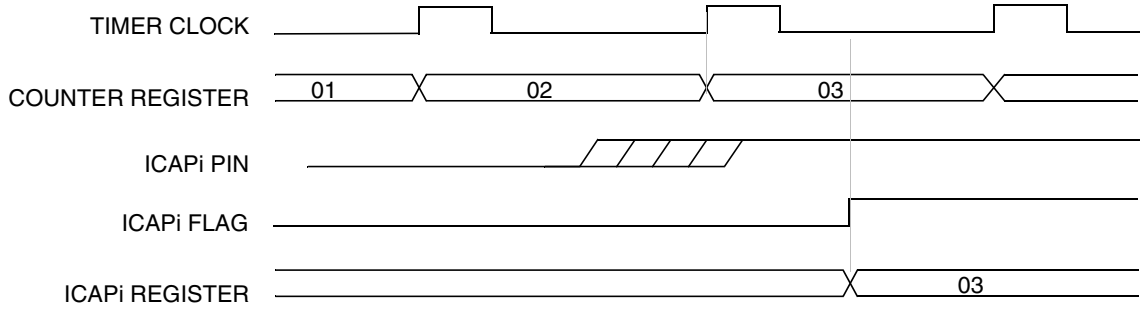


Figure 64. Input Capture Timing Diagram



Note: The rising edge is the active edge.

8-BIT TIMER (Cont'd)

10.5.3.3 Output Compare

In this section, the index, *i*, may be 1 or 2 because there are two output compare functions in the 8-bit timer.

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

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

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

Two 8-bit registers Output Compare Register 1 (OC1R) and Output Compare Register 2 (OC2R) contain the value to be compared to the counter register each timer clock cycle.

These registers are readable and writable and are not affected by the timer hardware. A reset event changes the OC*i*R value to 00h.

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

Procedure:

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

- Set the OC*i*E bit if an output is needed then the OCMP*i* pin is dedicated to the output compare *i* signal.
- Select the timer clock (CC[1:0]) (see Table 19 Clock Control Bits).

And select the following in the CR1 register:

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

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

- OCF*i* bit is set.
- The OCMP*i* pin takes OLVL*i* bit value (OCMP*i* pin latch is forced low during reset).
- A timer interrupt is generated if the OCIE bit is set in the CR1 register and the I bit is cleared in the CC register (CC).

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

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

Where:

Δt = Output compare period (in seconds)

f_{CPU} = PLL output x2 clock frequency in hertz (or $f_{OSC}/2$ if PLL is not enabled)

PRESC = Timer prescaler factor (2, 4, 8 or 8000 depending on CC[1:0] bits, see Table 19 Clock Control Bits)

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

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

8-BIT TIMER (Cont'd)

Notes:

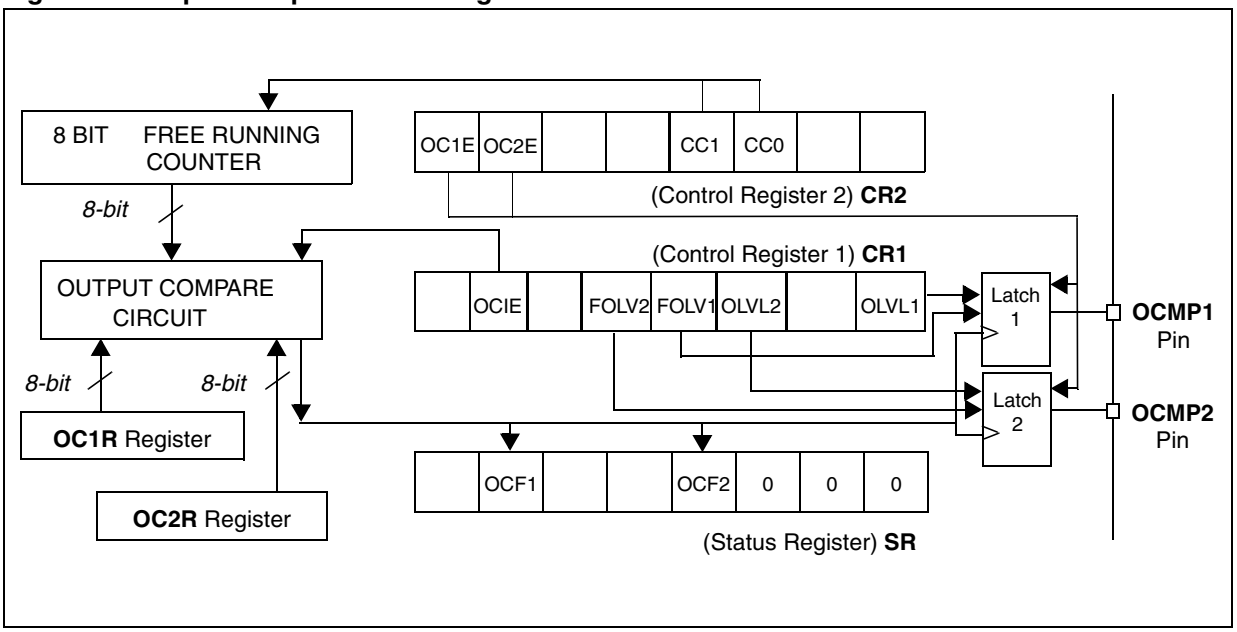
1. Once the OCIE bit is set both output compare features may trigger interrupt requests. If only one is needed in the application, the interrupt routine software needs to discard the unwanted compare interrupt. This can be done by checking the OCF1 and OCF2 flags and resetting them both.
2. If the OC*i*E bit is not set, the OCMP*i* pin is a general I/O port and the OLVL*i* bit will not appear when a match is found but an interrupt could be generated if the OCIE bit is set.
3. When the timer clock is $f_{CPU}/2$, OCF*i* and OCMP*i* are set while the counter value equals the OC*i*R register value (see Figure 66 on page 99). This behaviour is the same in OPM or PWM mode.
   When the timer clock is $f_{CPU}/4$, $f_{CPU}/8$ or $f_{CPU}/8000$, OCF*i* and OCMP*i* are set while the counter value equals the OC*i*R register value plus 1 (see Figure 67 on page 99).
4. The output compare functions can be used both for generating external events on the OCMP*i* pins even if the input capture mode is also used.
5. The value in the 8-bit OC*i*R register and the OLV*i* bit should be changed after each successful comparison in order to control an output waveform or establish a new elapsed timeout.

Forced Compare Output capability

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

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

Figure 65. Output Compare Block Diagram



8-BIT TIMER (Cont'd)

Figure 66. Output Compare Timing Diagram, $f_{TIMER} = f_{CPU}/2$

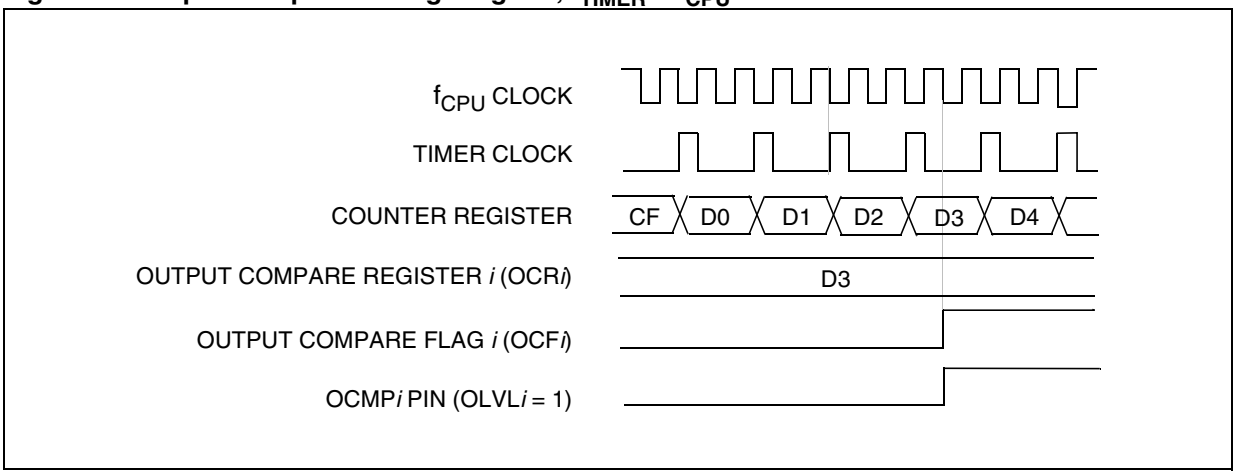
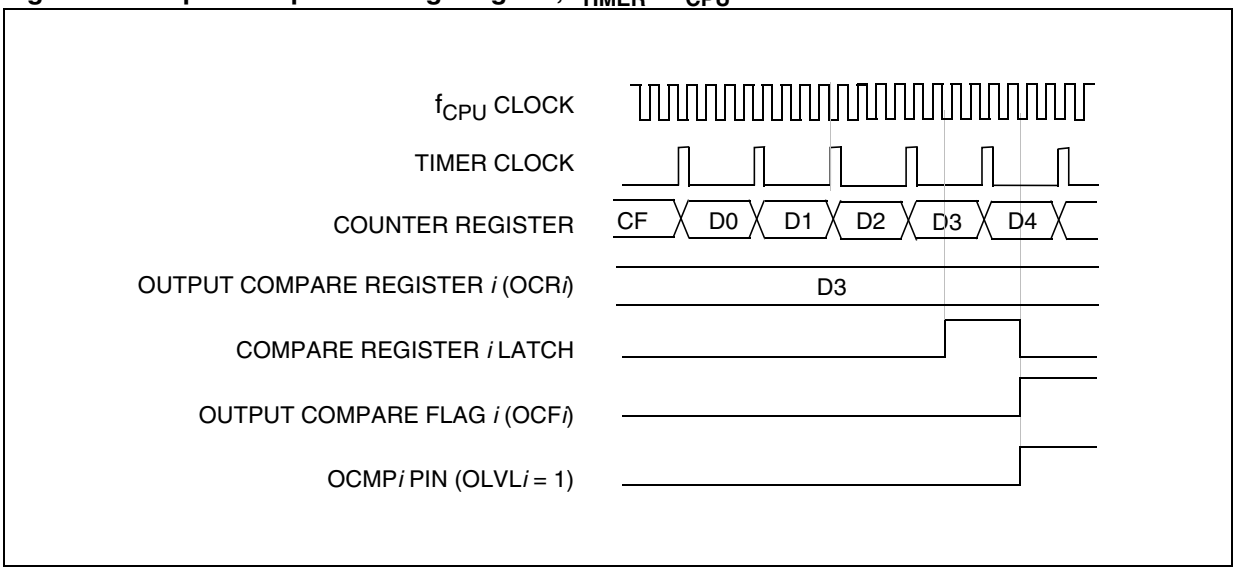


Figure 67. Output Compare Timing Diagram, $f_{TIMER} = f_{CPU}/4$



8-BIT TIMER (Cont'd)

10.5.3.4 One Pulse Mode

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

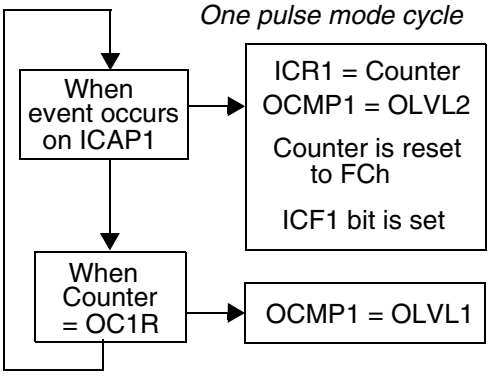
The one pulse mode uses the Input Capture1 function and the Output Compare1 function.

Procedure:

To use one pulse mode:

1. Load the OC1R register with the value corresponding to the length of the pulse (see the formula in the opposite column).
2. Select the following in the CR1 register:
   - Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after the pulse.
   - Using the OLVL2 bit, select the level to be applied to the OCMP1 pin during the pulse.
   - Select the edge of the active transition on the ICAP1 pin with the IEDG1 bit (the ICAP1 pin must be configured as floating input).
3. Select the following in the CR2 register:
   - Set the OC1E bit, the OCMP1 pin is then dedicated to the Output Compare 1 function.
   - Set the OPM bit.
   - Select the timer clock CC[1:0] (see Table 19 Clock Control Bits).

One pulse mode cycle



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

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

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

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

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

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

Where:

t = Pulse period (in seconds)

f_{CPU} = PLL output x2 clock frequency in hertz (or $f_{OSC}/2$ if PLL is not enabled)

PRESC = Timer prescaler factor (2, 4, 8 or 8000 depending on the CC[1:0] bits, see Table 19 Clock Control Bits)

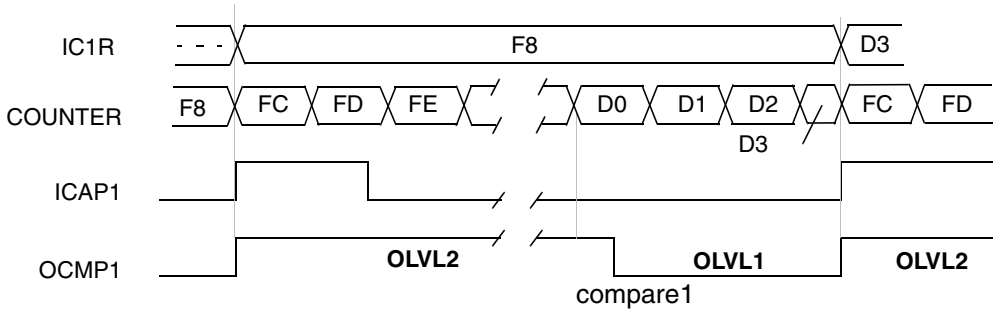
When the value of the counter is equal to the value of the contents of the OC1R register, the OLVL1 bit is output on the OCMP1 pin, (See Figure 68).

Notes:

1. The OCF1 bit cannot be set by hardware in one pulse mode but the OCF2 bit can generate an Output Compare interrupt.
2. When the Pulse Width Modulation (PWM) and One Pulse Mode (OPM) bits are both set, the PWM mode is the only active one.
3. If OLVL1=OLVL2 a continuous signal will be seen on the OCMP1 pin.
4. The ICAP1 pin can not be used to perform input capture. The ICAP2 pin can be used to perform input capture (ICF2 can be set and IC2R can be loaded) but the user must take care that the counter is reset each time a valid edge occurs on the ICAP1 pin and ICF1 can also generates interrupt if ICIE is set.
5. When one pulse mode is used OC1R is dedicated to this mode. Nevertheless OC2R and OCF2 can be used to indicate a period of time has been elapsed but cannot generate an output waveform because the level OLVL2 is dedicated to the one pulse mode.

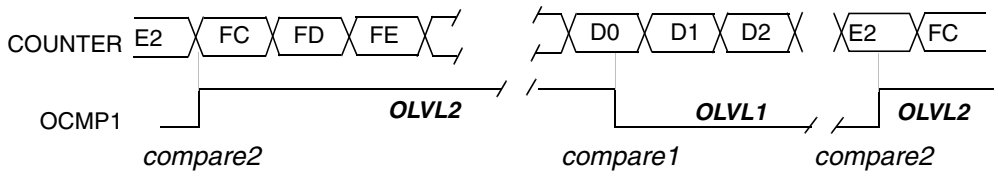
8-BIT TIMER (Cont'd)

Figure 68. One Pulse Mode Timing Example



Note: IEDG1 = 1, OC1R = D0h, OLVL1 = 0, OLVL2 = 1

Figure 69. Pulse Width Modulation Mode Timing Example



Note: OC1R = D0h, OC2R = E2, OLVL1 = 0, OLVL2 = 1

8-BIT TIMER (Cont'd)

10.5.3.5 Pulse Width Modulation Mode

Pulse Width Modulation (PWM) mode enables the generation of a signal with a frequency and pulse length determined by the value of the OC1R and OC2R registers.

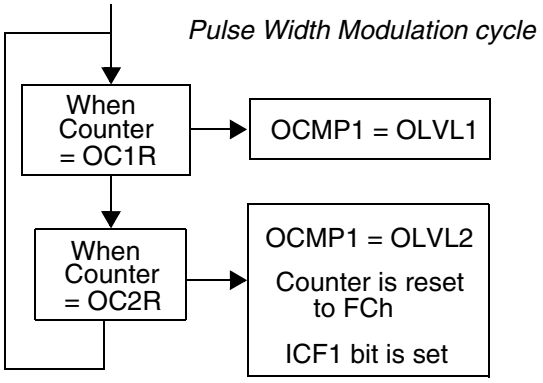
Pulse Width Modulation mode uses the complete Output Compare 1 function plus the OC2R register, and so this functionality can not be used when PWM mode is activated.

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

Procedure

To use pulse width modulation mode:

1. Load the OC2R register with the value corresponding to the period of the signal using the formula in the opposite column.
2. Load the OC1R register with the value corresponding to the period of the pulse if (OLVL1 = 0 and OLVL2 = 1) using the formula in the opposite column.
3. Select the following in the CR1 register:
   - Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after a successful comparison with the OC1R register.
   - Using the OLVL2 bit, select the level to be applied to the OCMP1 pin after a successful comparison with the OC2R register.
4. Select the following in the CR2 register:
   - Set OC1E bit: the OCMP1 pin is then dedicated to the output compare 1 function.
   - Set the PWM bit.
   - Select the timer clock (CC[1:0]) (see Table 19 Clock Control Bits).



If OLVL1 = 1 and OLVL2 = 0 the length of the positive pulse is the difference between the OC2R and OC1R registers.

If OLVL1 = OLVL2 a continuous signal will be seen on the OCMP1 pin.

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

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

Where:

t = Signal or pulse period (in seconds)

f_{CPU} = PLL output x2 clock frequency in hertz (or $f_{OSC}/2$ if PLL is not enabled)

PRESC = Timer prescaler factor (2, 4, 8 or 8000 depending on CC[1:0] bits, see Table 19 Clock Control Bits)

The Output Compare 2 event causes the counter to be initialized to FCh (See Figure 69)

Notes:

1. The OCF1 and OCF2 bits cannot be set by hardware in PWM mode therefore the Output Compare interrupt is inhibited.
2. The ICF1 bit is set by hardware when the counter reaches the OC2R value and can produce a timer interrupt if the ICIE bit is set and the I bit is cleared.
3. In PWM mode the ICAP1 pin can not be used to perform input capture because it is disconnected to the timer. The ICAP2 pin can be used to perform input capture (ICF2 can be set and IC2R can be loaded) but the user must take care that the counter is reset each period and ICF1 can also generates interrupt if ICIE is set.
4. When the Pulse Width Modulation (PWM) and One Pulse Mode (OPM) bits are both set, the PWM mode is the only active one.

8-BIT TIMER (Cont'd)

10.5.4 Low Power Modes

Mode	Description
WAIT	No effect on 8-bit Timer. Timer interrupts cause the device to exit from WAIT mode.
HALT	8-bit Timer registers are frozen. In HALT mode, the counter stops counting until Halt mode is exited. Counting resumes from the previous count when the MCU is woken up by an interrupt with "exit from HALT mode" capability or from the counter reset value when the MCU is woken up by a RESET. If an input capture event occurs on the ICAP*i* pin, the input capture detection circuitry is armed. Consequently, when the MCU is woken up by an interrupt with "exit from HALT mode" capability, the ICF*i* bit is set, and the counter value present when exiting from HALT mode is captured into the IC*i*R register.

10.5.5 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
Input Capture 1 event/Counter reset in PWM mode	ICF1	ICIE	Yes	No
Input Capture 2 event	ICF2			
Output Compare 1 event (not available in PWM mode)	OCF1	OCIE		
Output Compare 2 event (not available in PWM mode)	OCF2			
Timer Overflow event	TOF	TOIE		

Note: The 8-bit Timer interrupt events are connected to the same interrupt vector (see Interrupts chapter). These events generate an interrupt if the corresponding Enable Control Bit is set and the interrupt mask in the CC register is reset (RIM instruction).

10.5.6 Summary of Timer modes

MODES	AVAILABLE RESOURCES			
	Input Capture 1	Input Capture 2	Output Compare 1	Output Compare 2
Input Capture (1 and/or 2)	Yes	Yes	Yes	Yes
Output Compare (1 and/or 2)				
One Pulse Mode	No	Not Recommended[1)]	No	Partially [2)]
PWM Mode		Not Recommended[3)]		No

1) See note 4 in "One Pulse Mode" on page 100

2) See note 5 in "One Pulse Mode" on page 100

3) See note 4 in "Pulse Width Modulation Mode" on page 102

8-BIT TIMER (Cont'd)

10.5.7 Register Description

Each Timer is associated with three control and status registers, and with six data registers (8-bit values) relating to the two input captures, the two output compares, the counter and the alternate counter.

CONTROL REGISTER 1 (CR1)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
ICIE	OCIE	TOIE	FOLV2	FOLV1	OLVL2	IEDG1	OLVL1

Bit 7 = **ICIE** *Input Capture Interrupt Enable.*
0: Interrupt is inhibited.
1: A timer interrupt is generated whenever the ICF1 or ICF2 bit of the SR register is set.

Bit 6 = **OCIE** *Output Compare Interrupt Enable.*
0: Interrupt is inhibited.
1: A timer interrupt is generated whenever the OCF1 or OCF2 bit of the SR register is set.

Bit 5 = **TOIE** *Timer Overflow Interrupt Enable.*
0: Interrupt is inhibited.
1: A timer interrupt is enabled whenever the TOF bit of the SR register is set.

Bit 4 = **FOLV2** *Forced Output Compare 2.*
This bit is set and cleared by software.
0: No effect on the OCMP2 pin.
1: Forces the OLVL2 bit to be copied to the OCMP2 pin, if the OC2E bit is set and even if there is no successful comparison.

Bit 3 = **FOLV1** *Forced Output Compare 1.*
This bit is set and cleared by software.
0: No effect on the OCMP1 pin.
1: Forces OLVL1 to be copied to the OCMP1 pin, if the OC1E bit is set and even if there is no successful comparison.

Bit 2 = **OLVL2** *Output Level 2.*
This bit is copied to the OCMP2 pin whenever a successful comparison occurs with the OC2R register and OCxE is set in the CR2 register. This value is copied to the OCMP1 pin in One Pulse Mode and Pulse Width Modulation mode.

Bit 1 = **IEDG1** *Input Edge 1.*
This bit determines which type of level transition on the ICAP1 pin will trigger the capture.
0: A falling edge triggers the capture.
1: A rising edge triggers the capture.

Bit 0 = **OLVL1** *Output Level 1.*
The OLVL1 bit is copied to the OCMP1 pin whenever a successful comparison occurs with the OC1R register and the OC1E bit is set in the CR2 register.

8-BIT TIMER (Cont'd)

CONTROL REGISTER 2 (CR2)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
OC1E	OC2E	OPM	PWM	CC1	CC0	IEDG2	0

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

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

Bit 5 = **OPM** *One Pulse Mode.*
0: One Pulse Mode is not active.
1: One Pulse Mode is active, the ICAP1 pin can be used to trigger one pulse on the OCMP1 pin; the active transition is given by the IEDG1 bit. The length of the generated pulse depends on the contents of the OC1R register.

Bit 4 = **PWM** *Pulse Width Modulation.*
0: PWM mode is not active.
1: PWM mode is active, the OCMP1 pin outputs a programmable cyclic signal; the length of the pulse depends on the value of OC1R register; the period depends on the value of OC2R register.

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

The timer clock mode depends on these bits:

Table 20. Clock Control Bits

Timer Clock	CC1	CC0
f_{CPU} / 4	0	0
f_{CPU} / 2	0	1
f_{CPU} / 8	1	0
f_{OSC2} / 8000*	1	1

* Not available in Slow mode in ST72F561.

Bit 1 = **IEDG2** *Input Edge 2.*
This bit determines which type of level transition on the ICAP2 pin will trigger the capture.
0: A falling edge triggers the capture.
1: A rising edge triggers the capture.

Bit 0 = Reserved, must be kept at 0.

8-BIT TIMER (Cont'd)

CONTROL/STATUS REGISTER (CSR)

Read Only (except bit 2 R/W)

Reset Value: 0000 0000 (00h)

7							0
ICF1	OCF1	TOF	ICF2	OCF2	TIMD	0	0

Bit 7 = **ICF1** *Input Capture Flag 1.*
0: No input capture (reset value).
1: An input capture has occurred on the ICAP1 pin or the counter has reached the OC2R value in PWM mode. To clear this bit, first read the SR register, then read or write the the IC1R register.

Bit 6 = **OCF1** *Output Compare Flag 1.*
0: No match (reset value).
1: The content of the free running counter has matched the content of the OC1R register. To clear this bit, first read the SR register, then read or write the OC1R register.

Bit 5 = **TOF** *Timer Overflow Flag.*
0: No timer overflow (reset value).
1: The free running counter rolled over from FFh to 00h. To clear this bit, first read the SR register, then read or write the CTR register.

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

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

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

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

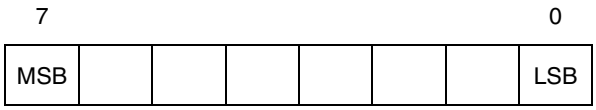
Bits 1:0 = Reserved, must be kept cleared.

8-BIT TIMER (Cont'd)

INPUT CAPTURE 1 REGISTER (IC1R)

Read Only
Reset Value: Undefined

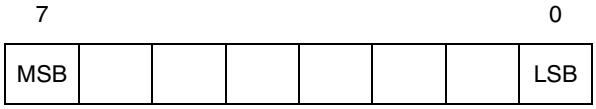
This is an 8-bit read only register that contains the counter value (transferred by the input capture 1 event).



OUTPUT COMPARE 1 REGISTER (OC1R)

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

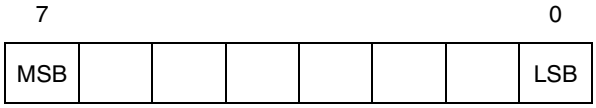
This is an 8-bit register that contains the value to be compared to the CTR register.



OUTPUT COMPARE 2 REGISTER (OC2R)

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

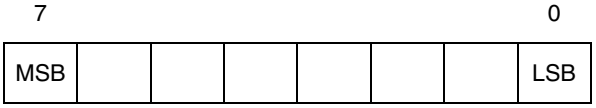
This is an 8-bit register that contains the value to be compared to the CTR register.



COUNTER REGISTER (CTR)

Read Only
Reset Value: 1111 1100 (FCh)

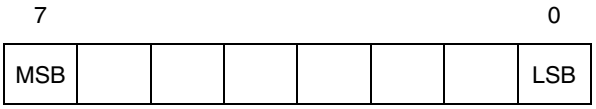
This is an 8-bit register that contains the counter value. A write to this register resets the counter. An access to this register after accessing the CSR register clears the TOF bit.



ALTERNATE COUNTER REGISTER (ACTR)

Read Only
Reset Value: 1111 1100 (FCh)

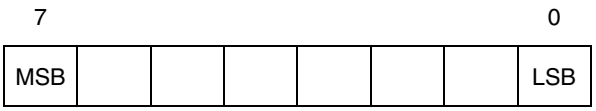
This is an 8-bit register that contains the counter value. A write to this register resets the counter. An access to this register after an access to CSR register does not clear the TOF bit in the CSR register.



INPUT CAPTURE 2 REGISTER (IC2R)

Read Only
Reset Value: Undefined

This is an 8-bit read only register that contains the counter value (transferred by the Input Capture 2 event).



8-BIT TIMER (Cont'd)

10.5.8 8-bit Timer Register Map

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
3C	**CR2**	OC1E	OC2E	OPM	PWM	CC1	CC0	IEDG2	0
3D	**CR1**	ICIE	OCIE	TOIE	FOLV2	FOLV1	OLVL2	IEDG1	OLVL1
3E	**CSR**	ICF1	OCF1	TOF	ICF2	OCF2	TIMD		
3F	**IC1R**	MSB							LSB
40	**OC1R**	MSB							LSB
41	**CTR**	MSB							LSB
42	**ACTR**	MSB							LSB
43	**IC2R**	MSB							LSB
44	**OC2R**	MSB							LSB

ON-CHIP PERIPHERALS (cont'd)

10.6 SERIAL PERIPHERAL INTERFACE (SPI)

10.6.1 Introduction

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

10.6.2 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.

10.6.3 General Description

Figure 70 on page 110 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

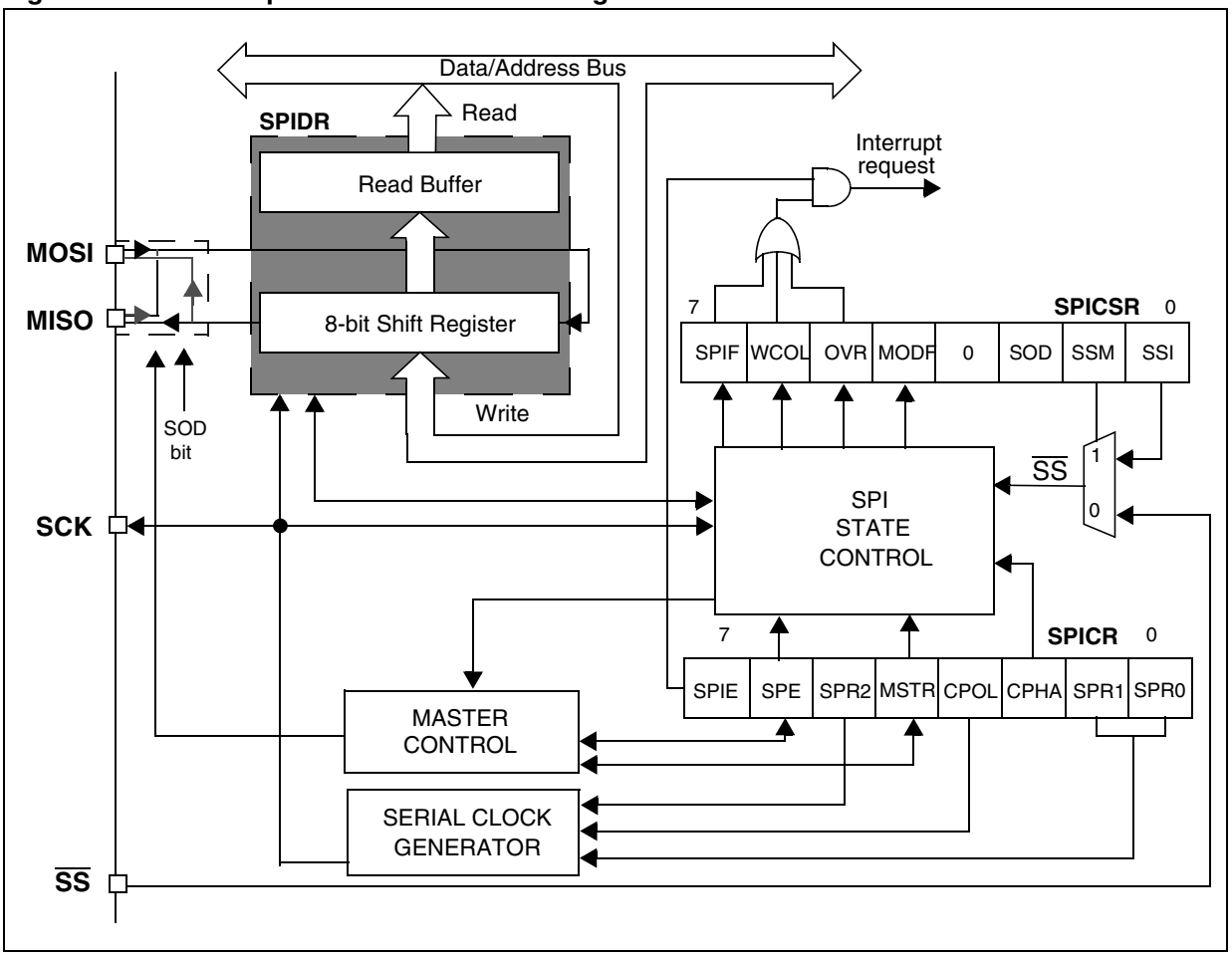
– MOSI: Master Out / Slave In data

– 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 70. Serial Peripheral Interface Block Diagram



SERIAL PERIPHERAL INTERFACE (cont'd)

10.6.3.1 Functional Description

A basic example of interconnections between a single master and a single slave is illustrated in Figure 71.

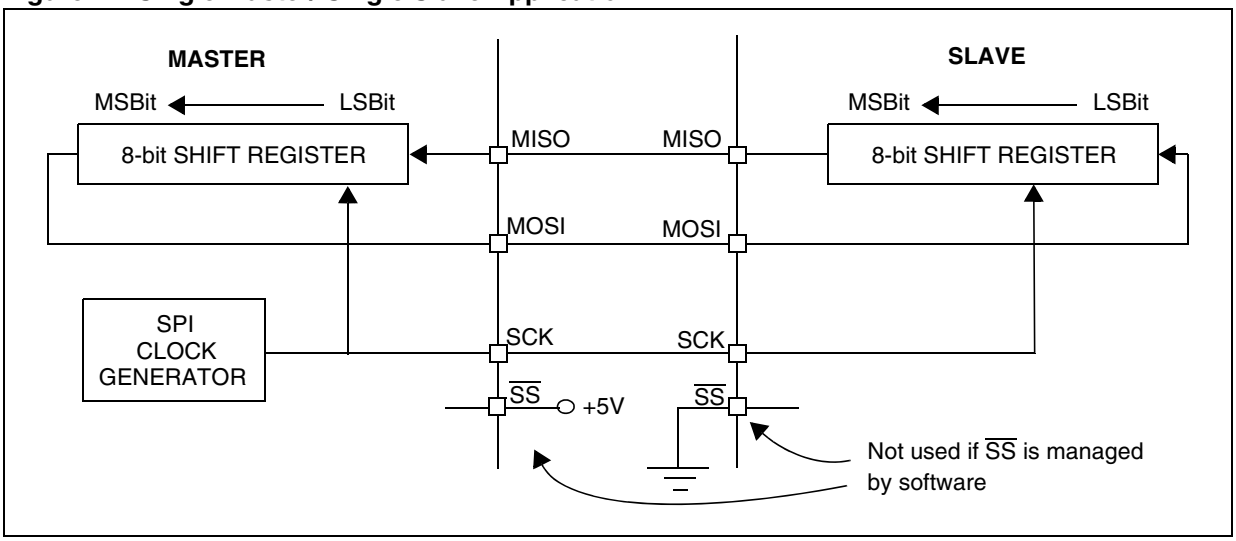
The MOSI pins are connected together and the MISO pins are connected together. In this way data is transferred serially between master and slave (most significant bit first).

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 74 on page 114) but master and slave must be programmed with the same timing mode.

Figure 71. Single Master/ Single Slave Application



SERIAL PERIPHERAL INTERFACE (cont'd)

10.6.3.2 Slave Select Management

As an alternative to using the $\overline{SS}$ pin to control the Slave Select signal, the application can choose to manage the Slave Select signal by software. This is configured by the SSM bit in the SPICSR register (see Figure 73).

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 72):

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

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

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

– $\overline{SS}$ internal must be held low during byte transmission and pulled high between each byte to allow the slave to write to the shift register. If $\overline{SS}$ is not pulled high, a Write Collision error will occur when the slave writes to the shift register (see Section 10.6.5.3).

Figure 72. Generic $\overline{SS}$ Timing Diagram

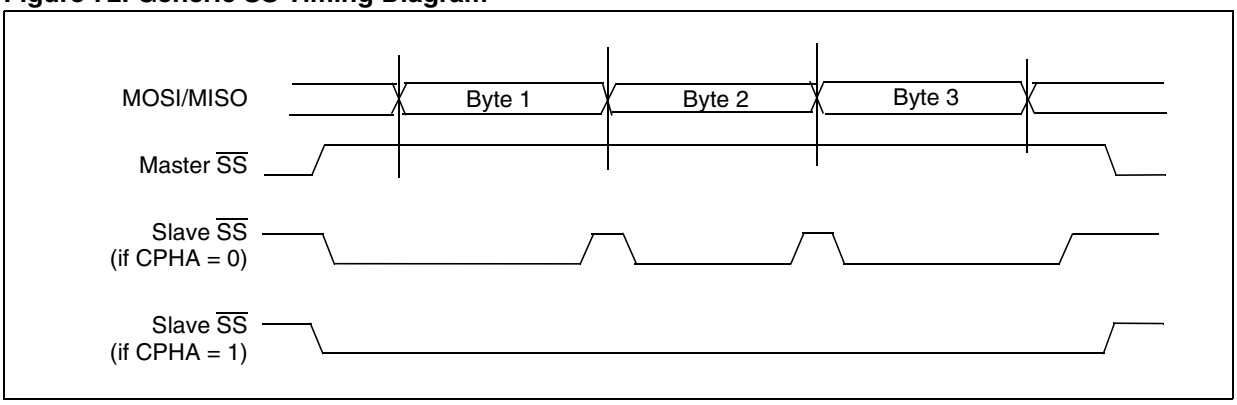
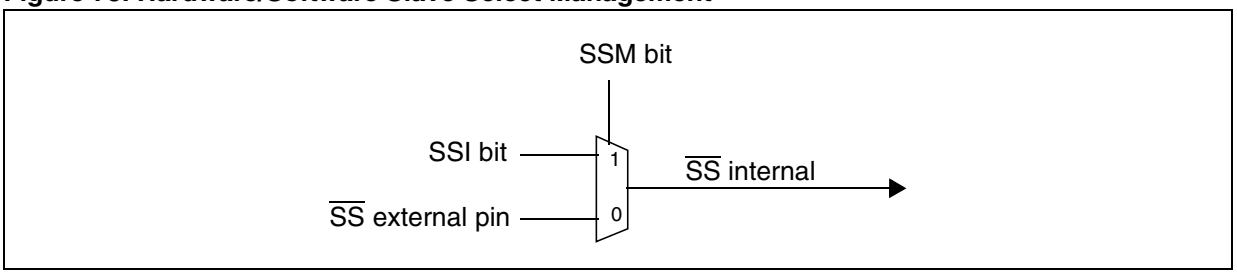


Figure 73. Hardware/Software Slave Select Management



SERIAL PERIPHERAL INTERFACE (cont'd)

10.6.3.3 Master Mode Operation

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

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

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 74 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.

10.6.3.4 Master Mode Transmit Sequence

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

When data transfer is complete:

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

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

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

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

10.6.3.5 Slave Mode Operation

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

To operate the SPI in slave mode:

1. Write to the SPICSR register to perform the following actions:
   - Select the clock polarity and clock phase by configuring the CPOL and CPHA bits (see Figure 74).
     Note: The slave must have the same CPOL and CPHA settings as the master.
   - Manage the $\overline{SS}$ pin as described in Section 10.6.3.2 and Figure 72. If CPHA = 1 $\overline{SS}$ must be held low continuously. If CPHA = 0 $\overline{SS}$ must be held low during byte transmission and pulled up between each byte to let the slave write in the shift register.
2. Write to the SPICR register to clear the MSTR bit and set the SPE bit to enable the SPI I/O functions.

10.6.3.6 Slave Mode Transmit Sequence

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

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

When data transfer is complete:

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

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

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

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

The SPIF bit can be cleared during a second transmission; however, it must be cleared before the second SPIF bit in order to prevent an Overrun condition (see Section 10.6.5.2).

SERIAL PERIPHERAL INTERFACE (cont'd)

10.6.4 Clock Phase and Clock Polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits (See Figure 74).

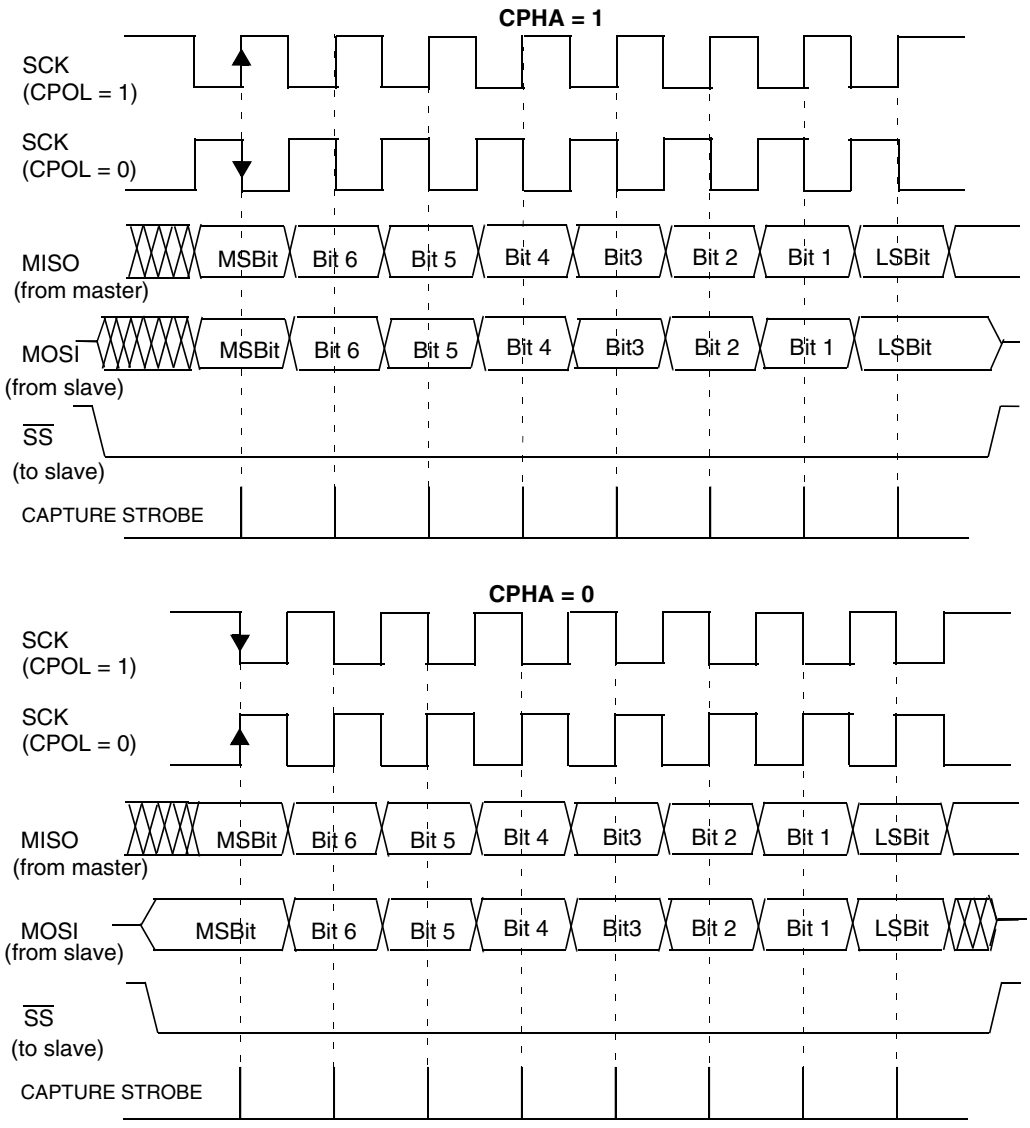
Note: The idle state of SCK must correspond to the polarity selected in the SPICSR register (by pulling up SCK if CPOL = 1 or pulling down SCK if CPOL = 0).

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

Figure 74 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 74. Data Clock Timing Diagram



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

SERIAL PERIPHERAL INTERFACE (cont'd)

10.6.5 Error Flags

10.6.5.1 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.

10.6.5.2 Overrun Condition (OVR)

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

When an Overrun occurs:

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

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

The OVR bit is cleared by reading the SPICSR register.

10.6.5.3 Write Collision Error (WCOL)

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

Write collisions can occur both in master and slave mode. See also Section 10.6.3.2 "Slave Select Management".

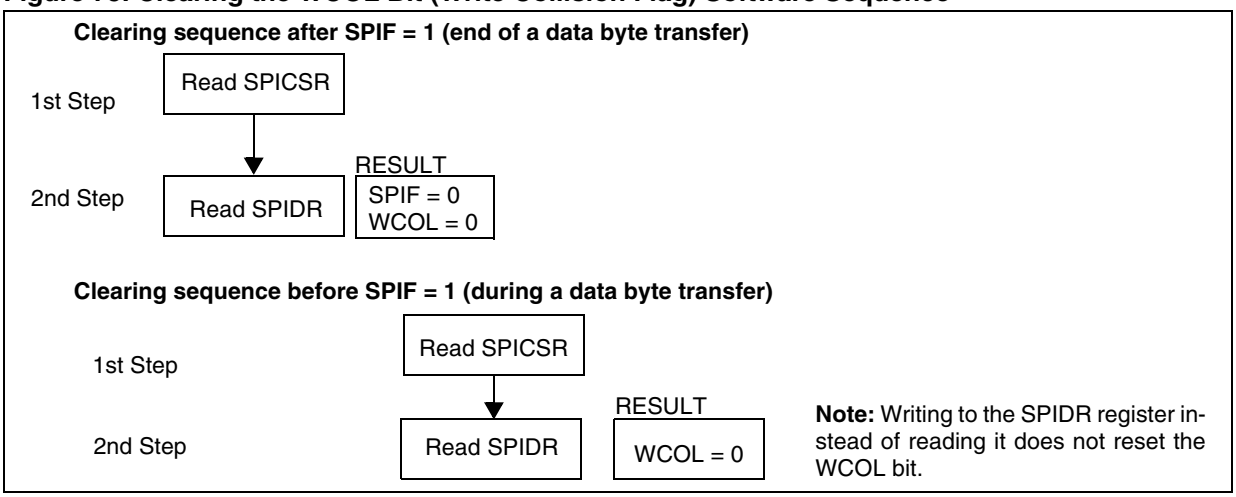
Note: A "read collision" will never occur since the received data byte is placed in a buffer in which access is always synchronous with the CPU operation.

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 75).

Figure 75. Clearing the WCOL Bit (Write Collision Flag) Software Sequence



SERIAL PERIPHERAL INTERFACE (cont'd)

10.6.5.4 Single Master and Multimaster Configurations

There are two types of SPI systems:

– Single Master System

– Multimaster System

Single Master System

A typical single master system may be configured using a device as the master and four devices as slaves (see Figure 76).

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.

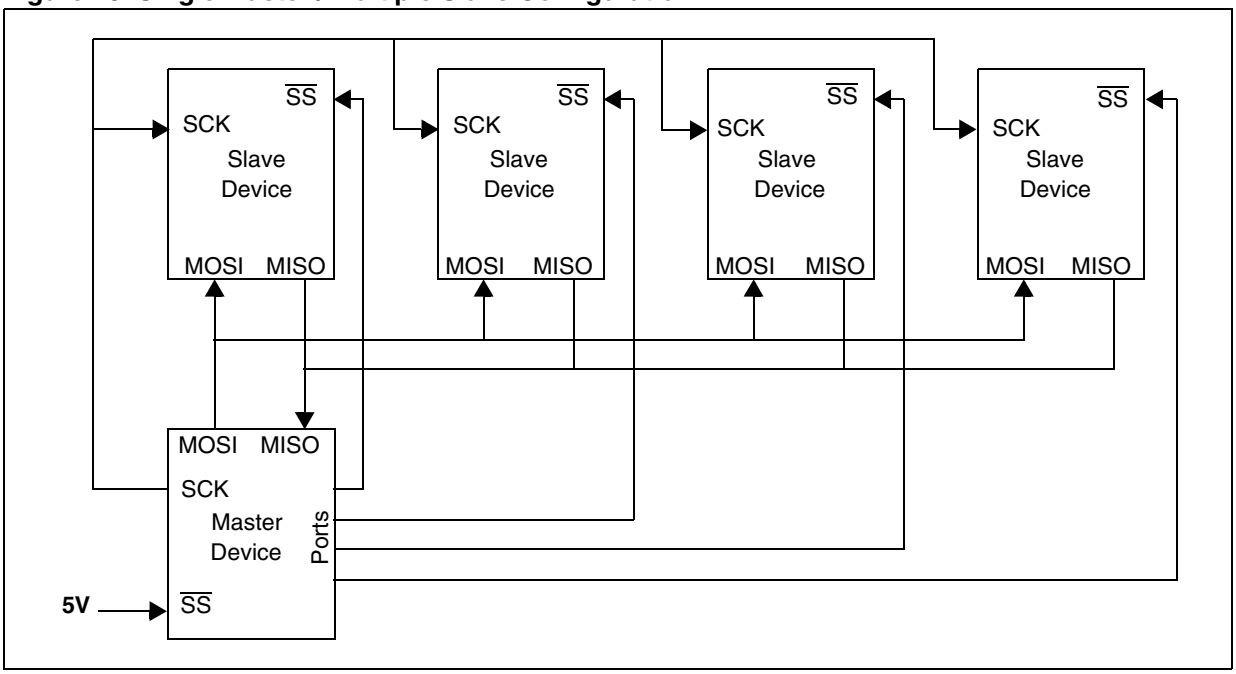
Other transmission security methods can use ports for handshake lines or data bytes with command fields.

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 76. Single Master / Multiple Slave Configuration



SERIAL PERIPHERAL INTERFACE (cont'd)

10.6.6 Low Power Modes

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

10.6.6.1 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 10.6.3.2), make sure the master drives a low level on the $\overline{SS}$ pin when the slave enters HALT mode.

10.6.7 Interrupts

Interrupt Event	Event Flag	Enable Control Bit	Exit from Wait	Exit from Halt
SPI End of Transfer Event	SPIF	SPIE	Yes	Yes
Master Mode Fault Event	MODF			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).

10.6.8 Register Description

SPI CONTROL REGISTER (SPICR)

Read/Write

Reset Value: 0000 xxxx (0xh)

7							0
SPIE	SPE	SPR2	MSTR	CPOL	CPHA	SPR1	SPR0

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

Bit 6 = **SPE** *Serial Peripheral Output Enable*
This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$ = 0 (see Section 10.6.5.1 "Master Mode Fault (MODF)"). The SPE bit is cleared by reset, so the SPI peripheral is not initially connected to the external pins.
0: I/O pins free for general purpose I/O
1: SPI I/O pin alternate functions enabled

Bit 5 = **SPR2** *Divider Enable*
This bit is set and cleared by software and is cleared by reset. It is used with the SPR[1:0] bits to set the baud rate. Refer to Table 20 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 10.6.5.1 "Master Mode Fault (MODF)").
0: Slave mode
1: Master mode. The function of the SCK pin changes from an input to an output and the functions of the MISO and MOSI pins are reversed.

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

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

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

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

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

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

Table 21. SPI Master Mode SCK Frequency

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

SERIAL PERIPHERAL INTERFACE (cont'd)

SPI CONTROL/STATUS REGISTER (SPICSR)

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

7							0
SPIF	WCOL	OVR	MODF	-	SOD	SSM	SSI

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

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

Bit 6 = **WCOL** *Write Collision status (Read only)*
This bit is set by hardware when a write to the SPIDR register is done during a transmit sequence. It is cleared by a software sequence (see Figure 75).
0: No write collision occurred
1: A write collision has been detected

Bit 5 = **OVR** S*PI Overrun error (Read only)*
This bit is set by hardware when the byte currently being received in the shift register is ready to be transferred into the SPIDR register while SPIF = 1 (See Section 10.6.5.2). 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 10.6.5.1 "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 10.6.3.2 "Slave Select Management".
0: Hardware management ($\overline{SS}$ managed by external pin)
1: Software management (internal $\overline{SS}$ signal controlled by SSI bit. External $\overline{SS}$ pin free for general-purpose I/O)

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

SPI DATA I/O REGISTER (SPIDR)

Read/Write
Reset Value: Undefined

7							0
D7	D6	D5	D4	D3	D2	D1	D0

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

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

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

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

A read to the SPIDR register returns the value located in the buffer and not the content of the shift register (see Figure 70).

SERIAL PERIPHERAL INTERFACE (Cont'd)

Table 22. SPI Register Map and Reset Values

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

10.7 LINSCI SERIAL COMMUNICATION INTERFACE (LIN MASTER/SLAVE)

10.7.1 Introduction

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

The LIN-dedicated features support the LIN (Local Interconnect Network) protocol for both master and slave nodes.

This chapter is divided into SCI Mode and LIN mode sections. For information on general SCI communications, refer to the SCI mode section. For LIN applications, refer to both the SCI mode and LIN mode sections.

10.7.2 SCI Features

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

10.7.3 LIN Features

– LIN Master
  – 13-bit LIN Synch Break generation

– LIN Slave
  – Automatic Header Handling
  – Automatic baud rate resynchronization based on recognition and measurement of the LIN Synch Field (for LIN slave nodes)
  – Automatic baud rate adjustment (at CPU frequency precision)
  – 11-bit LIN Synch Break detection capability
  – LIN Parity check on the LIN Identifier Field (only in reception)
  – LIN Error management
  – LIN Header Timeout
  – Hot plugging support

LINSCI™ SERIAL COMMUNICATION INTERFACE (cont'd)

10.7.4 General Description

The interface is externally connected to another device by two pins:

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

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

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

– An Idle Line prior to transmission or reception

– A start bit

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

– A Stop bit indicating that the character is complete

This interface uses three types of baud rate generator:

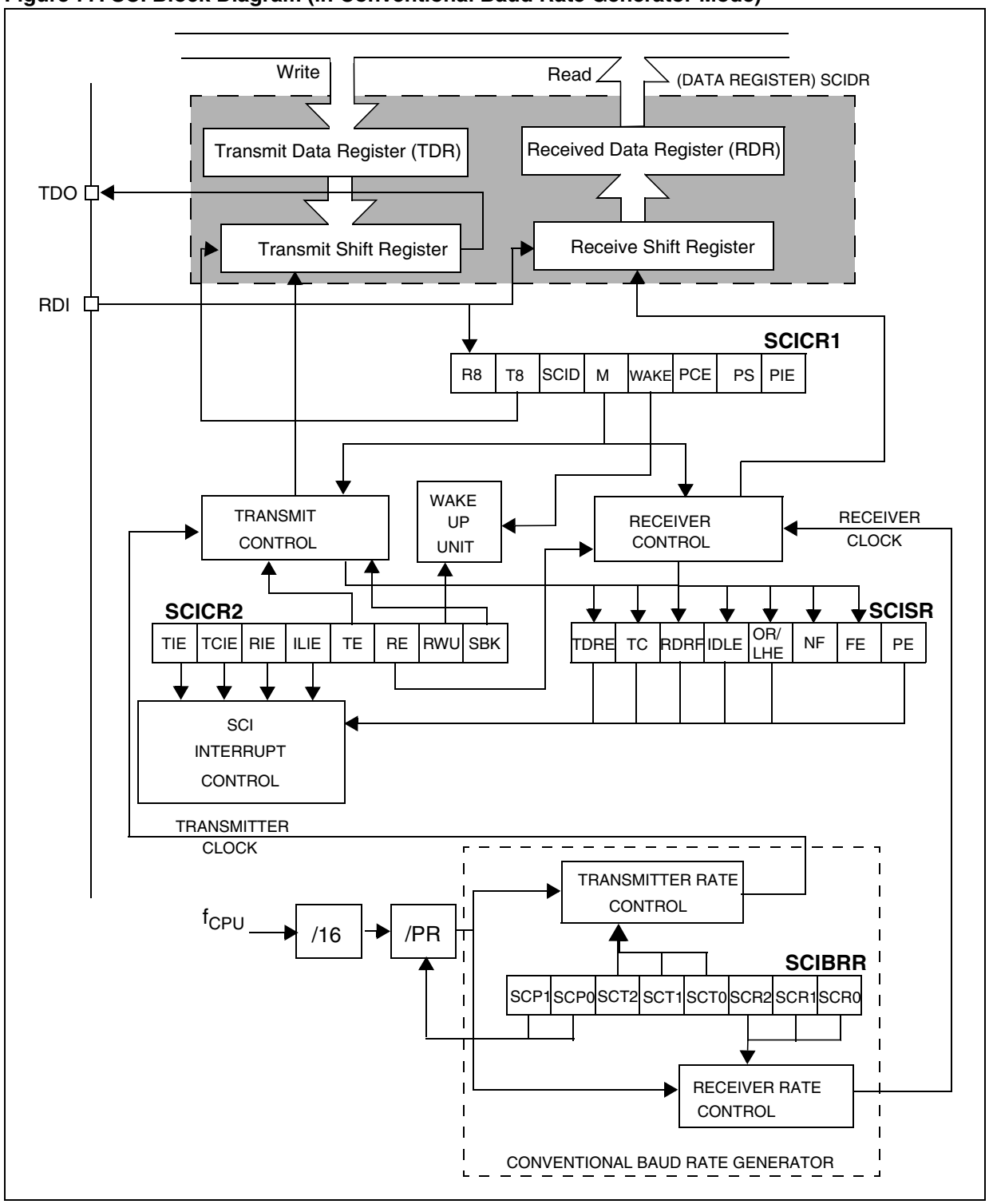
– A conventional type for commonly-used baud rates

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

– A LIN baud rate generator with automatic resynchronization

LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

Figure 77. SCI Block Diagram (in Conventional Baud Rate Generator Mode)



LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

10.7.5 SCI Mode - Functional Description

Conventional Baud Rate Generator Mode

The block diagram of the Serial Control Interface in conventional baud rate generator mode is shown in Figure 1.

It uses four registers:

– 2 control registers (SCICR1 and SCICR2)

– A status register (SCISR)

– A baud rate register (SCIBRR)

Extended Prescaler Mode

Two additional prescalers are available in extended prescaler mode. They are shown in Figure 3.

– An extended prescaler receiver register (SCIERPR)

– An extended prescaler transmitter register (SCIETPR)

10.7.5.1 Serial Data Format

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

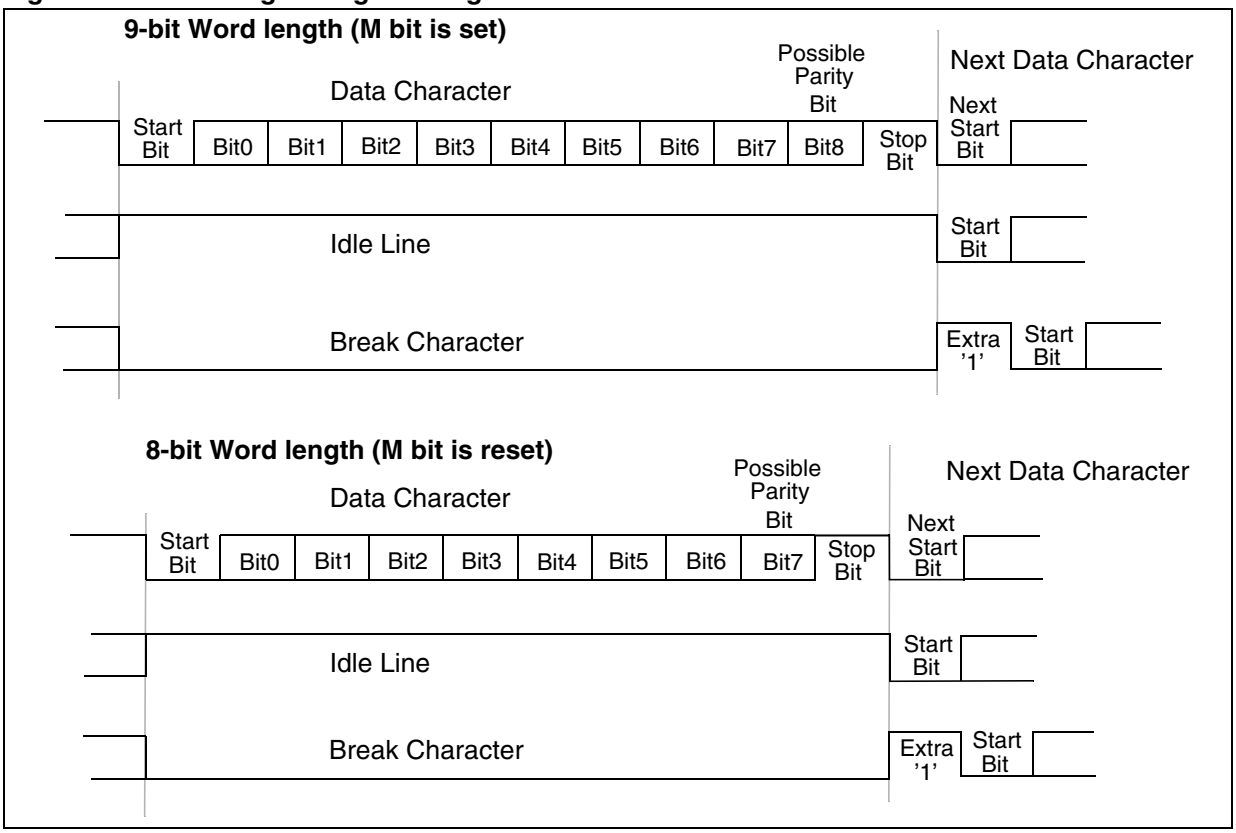
The TDO pin is in low state during the start bit.

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

An Idle character is interpreted as a continuous logic high level for 10 (or 11) full bit times.

A Break character is a character with a sufficient number of low level bits to break the normal data format followed by an extra "1" bit to acknowledge the start bit.

Figure 78. Word Length Programming



LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

10.7.5.2 Transmitter

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

Character Transmission

During an SCI transmission, data shifts out least significant bit first on the TDO pin. In this mode, the SCIDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see Figure 1).

Procedure

– Select the M bit to define the word length.

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

– Set the TE bit to send a preamble of 10 (M = 0) or 11 (M = 1) consecutive ones (Idle Line) as first transmission.

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

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

The TDRE bit is set by hardware and it indicates:

– The TDR register is empty.

– The data transfer is beginning.

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

This flag generates an interrupt if the TIE bit is set and the I[1:0] bits are cleared in the CCR register.

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

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

When a character transmission is complete (after the stop bit) the TC bit is set and an interrupt is generated if the TCIE is set and the I[1:0] bits are cleared in the CCR register.

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

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

Break Characters

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

As long as the SBK bit is set, the SCI sends break characters to the TDO pin. After clearing this bit by software, the SCI inserts a logic 1 bit at the end of the last break character to guarantee the recognition of the start bit of the next character.

Idle Line

Setting the TE bit drives the SCI to send a preamble of 10 (M = 0) or 11 (M = 1) consecutive '1's (idle line) before the first character.

In this case, clearing and then setting the TE bit during a transmission sends a preamble (idle line) after the current word. Note that the preamble duration (10 or 11 consecutive '1's depending on the M bit) does not take into account the stop bit of the previous character.

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

LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

10.7.5.3 Receiver

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

Character reception

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

Procedure

– Select the M bit to define the word length.

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

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

When a character is received:

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

– An interrupt is generated if the RIE bit is set and the I[1:0] bits are cleared in the CCR register.

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

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

1. An access to the SCISR register

2. A read to the SCIDR register.

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

Idle Line

When an idle line is detected, there is the same procedure as a data received character plus an interrupt if the ILIE bit is set and the I[1:0] bits are cleared in the CCR register.

Overrun Error

An overrun error occurs when a character is received when RDRF has not been reset. Data can not be transferred from the shift register to the TDR register as long as the RDRF bit is not cleared.

When an overrun error occurs:

– The OR bit is set.

– The RDR content will not be lost.

– The shift register will be overwritten.

– An interrupt is generated if the RIE bit is set and the I[1:0] bits are cleared in the CCR register.

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

Noise Error

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

When noise is detected in a character:

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

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

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

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

Framing Error

A framing error is detected when:

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

– A break is received.

When the framing error is detected:

– the FE bit is set by hardware

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

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

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

Break Character

– When a break character is received, the SCI handles it as a framing error. To differentiate a break character from a framing error, it is necessary to read the SCIDR. If the received value is 00h, it is a break character. Otherwise it is a framing error.

LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

10.7.5.4 Conventional Baud Rate Generation

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

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

with:

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

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

(see SCT[2:0] bits)

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

(see SCR[2:0] bits)

All these bits are in the SCIBRR register.

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

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

10.7.5.5 Extended Baud Rate Generation

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

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

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

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

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

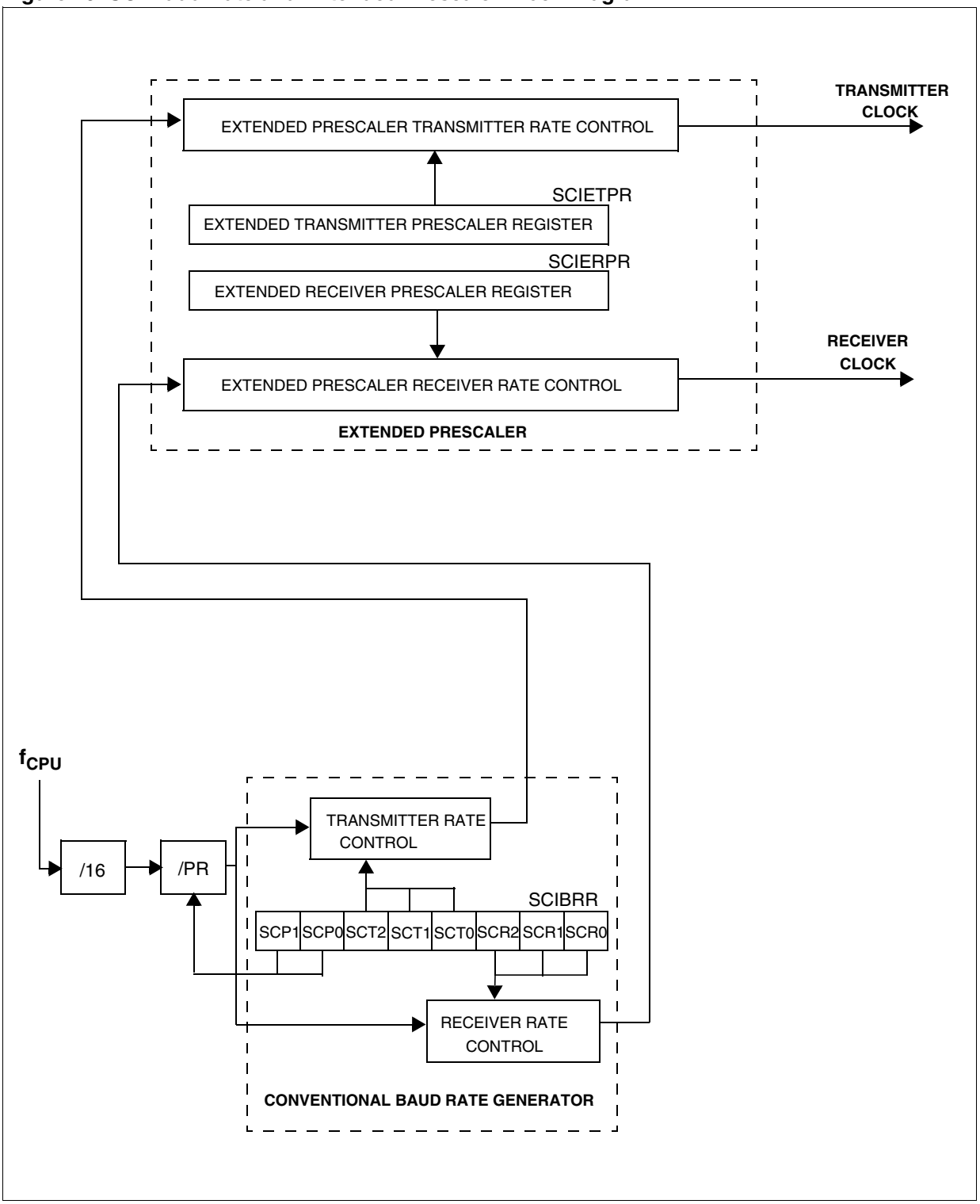
with:

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

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

LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

Figure 79. SCI Baud Rate and Extended Prescaler Block Diagram



LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

10.7.5.6 Receiver Muting and Wake-up Feature

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

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

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

All the reception status bits can not be set.

All the receive interrupts are inhibited.

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

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

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

Idle Line Detection

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

This feature is useful in a multiprocessor system when the first characters of the message determine the address and when each message ends by an idle line: As soon as the line becomes idle, every receivers is waken up and analyse the first characters of the message which indicates the addressed receiver. The receivers which are not addressed set RWU bit to enter in mute mode. Consequently, they will not treat the next characters constituting the next part of the message. At the end of the message, an idle line is sent by the transmitter: this wakes up every receivers which are ready to analyse the addressing characters of the new message.

In such a system, the inter-characters space must be smaller than the idle time.

Address Mark Detection

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

This feature is useful in a multiprocessor system when the most significant bit of each character (except for the break character) is reserved for Address Detection. As soon as the receivers received an address character (most significant bit = '1'), the receivers are waken up. The receivers which are not addressed set RWU bit to enter in mute mode. Consequently, they will not treat the next characters constituting the next part of the message.

10.7.5.7 Parity Control

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

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

Table 23. Character Formats

M bit	PCE bit	Character format
0	0	ǀ SB ǀ 8 bit data ǀ STB ǀ
	1	ǀ SB ǀ 7-bit data ǀ PB ǀ STB ǀ
1	0	ǀ SB ǀ 9-bit data ǀ STB ǀ
	1	ǀ SB ǀ 8-bit data ǀ PB ǀ STB ǀ

Legend: SB = Start Bit, STB = Stop Bit, PB = Parity Bit

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

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

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

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

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

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

LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

10.7.6 Low Power Modes

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

10.7.7 Interrupts

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

The SCI interrupt events are connected to the same interrupt vector (see Interrupts chapter).

These events generate an interrupt if the corresponding Enable Control Bit is set and the interrupt mask in the CC register is reset (RIM instruction).

LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

10.7.8 SCI Mode Register Description

STATUS REGISTER (SCISR)

Read Only

Reset Value: 1100 0000 (C0h)

7							0
TDRE	TC	RDRF	IDLE	OR[1]	NF[1]	FE[1]	PE[1]

Bit 7 = **TDRE** *Transmit data register empty*

This bit is set by hardware when the content of the TDR register has been transferred into the shift register. An interrupt is generated if the TIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a write to the SCIDR register).

0: Data is not transferred to the shift register

1: Data is transferred to the shift register

Bit 6 = **TC** *Transmission complete*

This bit is set by hardware when transmission of a character containing Data is complete. An interrupt is generated if TCIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a write to the SCIDR register).

0: Transmission is not complete

1: Transmission is complete

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

Bit 5 = **RDRF** *Received data ready flag*

This bit is set by hardware when the content of the RDR register has been transferred to the SCIDR register. An interrupt is generated if RIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).

0: Data is not received

1: Received data is ready to be read

Bit 4 = **IDLE** *Idle line detected*

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if the ILIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).

0: No Idle Line is detected

1: Idle Line is detected

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

Bit 3 = **OR** *Overrun error*

The OR bit is set by hardware when the word currently being received in the shift register is ready to be transferred into the RDR register whereas RDRF is still set. An interrupt is generated if RIE = 1 in the SCICR2 register. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).

0: No Overrun error

1: Overrun error detected

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

Bit 2 = **NF** *Character Noise flag*

This bit is set by hardware when noise is detected on a received character. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).

0: No noise

1: Noise is detected

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

Bit 1 = **FE** *Framing error*

This bit is set by hardware when a desynchronization, excessive noise or a break character is detected. It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).

0: No Framing error

1: Framing error or break character detected

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

Bit 0 = **PE** *Parity error*

This bit is set by hardware when a byte parity error occurs (if the PCE bit is set) in receiver mode. It is cleared by a software sequence (a read to the status register followed by an access to the SCIDR data register). An interrupt is generated if PIE = 1 in the SCICR1 register.

0: No parity error

1: Parity error detected

LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

CONTROL REGISTER 1 (SCICR1)

Read/Write
Reset Value: x000 0000 (x0h)

7							0
R8	T8	SCID	M	WAKE	PCE[1)]	PS	PIE

[1)]This bit has a different function in LIN mode, please refer to the LIN mode register description.

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

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

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

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

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

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

Note: If the LINE bit is set, the WAKE bit is deactivated and replaced by the LHDM bit.

Bit 2 = **PCE** *Parity control enable*
This bit is set and cleared by software. It selects the hardware parity control (generation and detection for byte parity, detection only for LIN parity).
0: Parity control disabled
1: Parity control enabled

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

Bit 0 = **PIE** *Parity interrupt enable*
This bit enables the interrupt capability of the hardware parity control when a parity error is detected (PE bit set). The parity error involved can be a byte parity error (if bit PCE is set and bit LPE is reset) or a LIN parity error (if bit PCE is set and bit LPE is set).
0: Parity error interrupt disabled
1: Parity error interrupt enabled

LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

CONTROL REGISTER 2 (SCICR2)

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

7							0
TIE	TCIE	RIE	ILIE	TE	RE	RWU[1]	SBK[1]

[1] This bit has a different function in LIN mode, please refer to the LIN mode register description.

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

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

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

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

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

Notes:

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

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

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

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

Notes:

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

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

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

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

DATA REGISTER (SCIDR)

Read/Write
Reset Value: Undefined

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

7							0
DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0

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

LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

BAUD RATE REGISTER (SCIBRR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
SCP1	SCP0	SCT2	SCT1	SCT0	SCR2	SCR1	SCR0

Note: When LIN slave mode is disabled, the SCIBRR register controls the conventional baud rate generator.

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

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

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

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

Bits 2:0 = **SCR[2:0]** *SCI Receiver rate divider*
These 3 bits, in conjunction with the SCP[1:0] bits define the total division applied to the bus clock to yield the receive rate clock in conventional Baud Rate Generator mode.

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

LINSCI™ SERIAL COMMUNICATION INTERFACE (SCI Mode) (cont'd)

EXTENDED RECEIVE PRESCALER DIVISION REGISTER (SCIERPR)

Read/Write

Reset Value: 0000 0000 (00h)

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

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

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

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

EXTENDED TRANSMIT PRESCALER DIVISION REGISTER (SCIETPR)

Read/Write

Reset Value:0000 0000 (00h)

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

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

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

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

Note: In LIN slave mode, the Conventional and Extended Baud Rate Generators are disabled.

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode)

10.7.9 LIN Mode - Functional Description.

The block diagram of the Serial Control Interface, in LIN slave mode is shown in Figure 5.

It uses six registers:

– 3 control registers: SCICR1, SCICR2 and SCICR3

– 2 status registers: the SCISR register and the LHLR register mapped at the SCIERPR address

– A baud rate register: LPR mapped at the SCIBRR address and an associated fraction register LPFR mapped at the SCIETPR address

The bits dedicated to LIN are located in the SCICR3. Refer to the register descriptions in Section 0.1.10 for the definitions of each bit.

10.7.9.1 Entering LIN Mode

To use the LINSCI in LIN mode the following configuration must be set in SCICR3 register:

– Clear the M bit to configure 8-bit word length.

– Set the LINE bit.

Master

To enter master mode the LSLV bit must be reset In this case, setting the SBK bit will send 13 low bits.

Then the baud rate can programmed using the SCIBRR, SCIERPR and SCIETPR registers.

In LIN master mode, the Conventional and / or Extended Prescaler define the baud rate (as in standard SCI mode)

Slave

Set the LSLV bit in the SCICR3 register to enter LIN slave mode. In this case, setting the SBK bit will have no effect.

In LIN Slave mode the LIN baud rate generator is selected instead of the Conventional or Extended Prescaler. The LIN baud rate generator is common to the transmitter and the receiver.

Then the baud rate can be programmed using LPR and LPRF registers.

Note: It is mandatory to set the LIN configuration first before programming LPR and LPRF, because the LIN configuration uses a different baud rate generator from the standard one.

10.7.9.2 LIN Transmission

In LIN mode the same procedure as in SCI mode has to be applied for a LIN transmission.

To transmit the LIN Header the proceed as follows:

– First set the SBK bit in the SCICR2 register to start transmitting a 13-bit LIN Synch Break

– reset the SBK bit

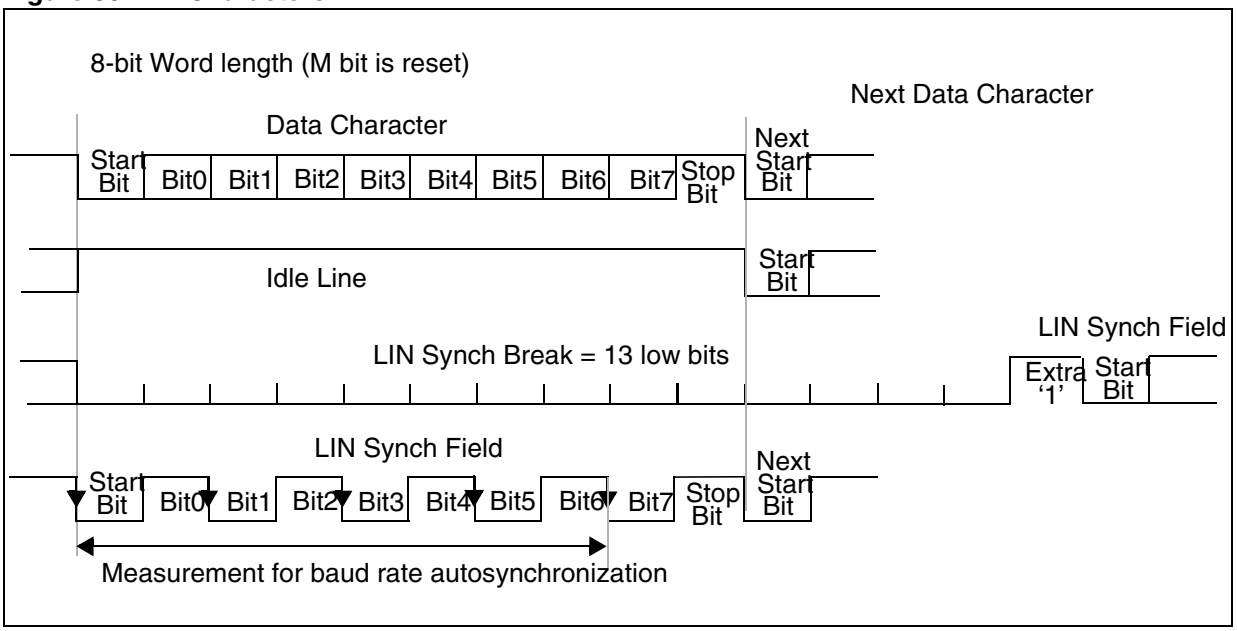
– Load the LIN Synch Field (0x55) in the SCIDR register to request Synch Field transmission

– Wait until the SCIDR is empty (TDRE bit set in the SCISR register)

– Load the LIN message Identifier in the SCIDR register to request Identifier transmission.

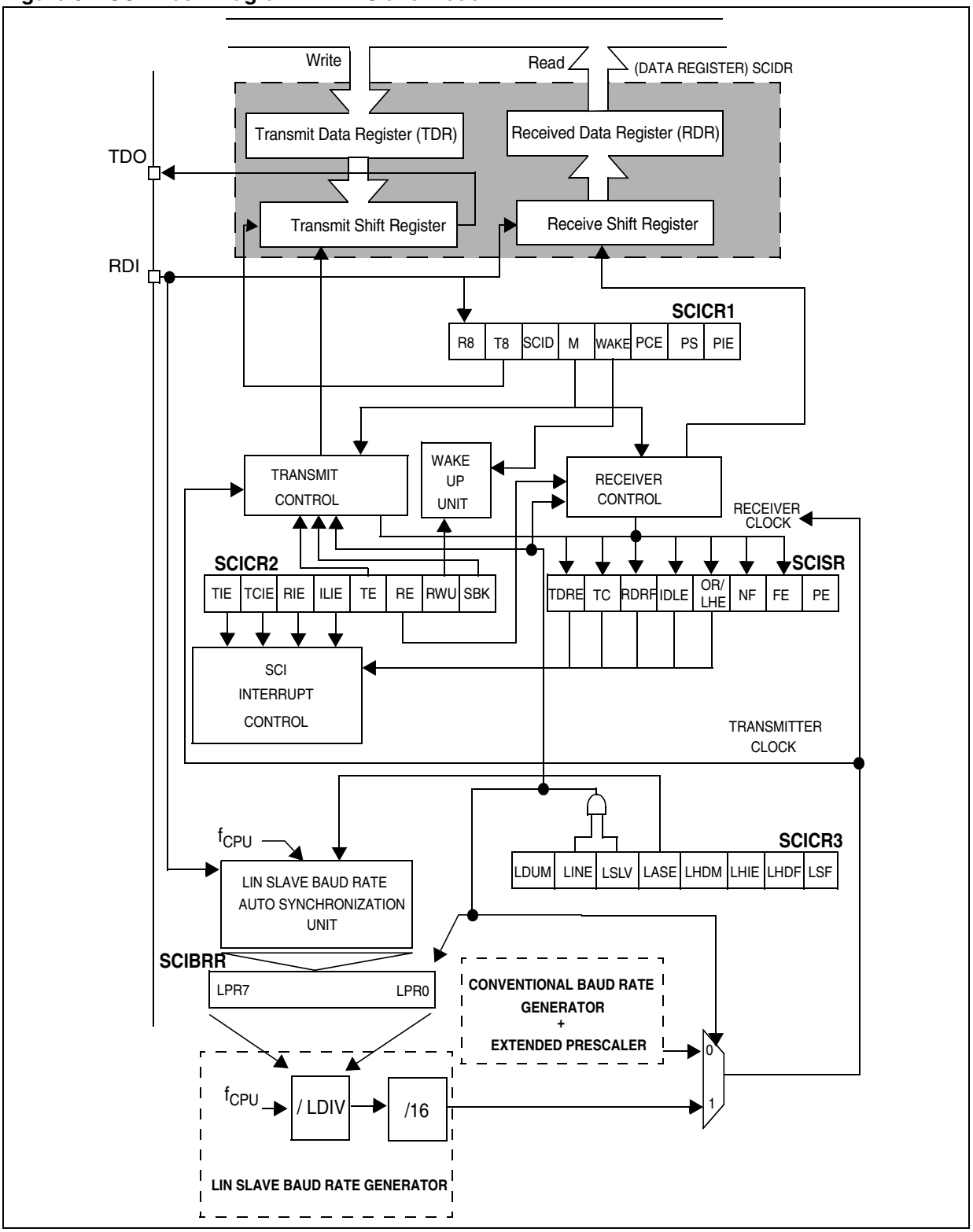
LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode) (cont'd)

Figure 80. LIN Characters



LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode) (cont'd)

Figure 81. SCI Block Diagram in LIN Slave Mode



LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode) (cont'd)

10.7.9.3 LIN Reception

In LIN mode the reception of a byte is the same as in SCI mode but the LINSCI has features for handling the LIN Header automatically (identifier detection) or semiautomatically (Synch Break detection) depending on the LIN Header detection mode. The detection mode is selected by the LHDM bit in the SCICR3.

Additionally, an automatic resynchronization feature can be activated to compensate for any clock deviation, for more details please refer to Section 0.1.9.5 LIN Baud Rate.

LIN Header Handling by a Slave

Depending on the LIN Header detection method the LINSCI will signal the detection of a LIN Header after the LIN Synch Break or after the Identifier has been successfully received.

Note:

It is recommended to combine the Header detection function with Mute mode. Putting the LINSCI in Mute mode allows the detection of Headers only and prevents the reception of any other characters.

This mode can be used to wait for the next Header without being interrupted by the data bytes of the current message in case this message is not relevant for the application.

Synch Break Detection (LHDM = 0):

When a LIN Synch Break is received:

– The RDRF bit in the SCISR register is set. It indicates that the content of the shift register is transferred to the SCIDR register, a value of 0x00 is expected for a Break.

– The LHDF flag in the SCICR3 register indicates that a LIN Synch Break Field has been detected.

– An interrupt is generated if the LHIE bit in the SCICR3 register is set and the I[1:0] bits are cleared in the CCR register.

– Then the LIN Synch Field is received and measured.

  – If automatic resynchronization is enabled (LASE bit = 1), the LIN Synch Field is not transferred to the shift register: There is no need to clear the RDRF bit.

  – If automatic resynchronization is disabled (LASE bit = 0), the LIN Synch Field is received as a normal character and transferred to the SCIDR register and RDRF is set.

Note:

In LIN slave mode, the FE bit detects all frame error which does not correspond to a break.

Identifier Detection (LHDM = 1):

This case is the same as the previous one except that the LHDF and the RDRF flags are set only after the entire header has been received (this is true whether automatic resynchronization is enabled or not). This indicates that the LIN Identifier is available in the SCIDR register.

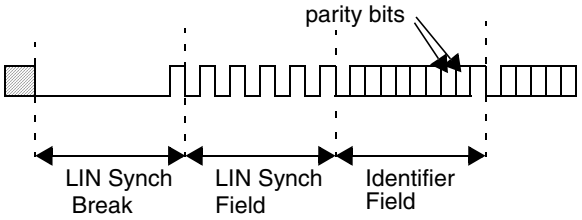
Notes:

During LIN Synch Field measurement, the SCI state machine is switched off: No characters are transferred to the data register.

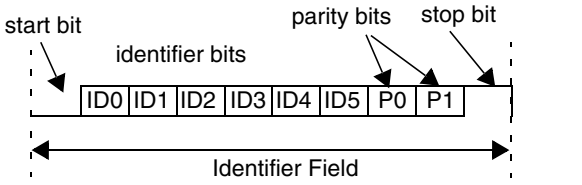
LIN Slave parity

In LIN Slave mode (LINE and LSLV bits are set) LIN parity checking can be enabled by setting the PCE bit.

In this case, the parity bits of the LIN Identifier Field are checked. The identifier character is recognized as the third received character after a break character (included):



The bits involved are the two MSB positions (7th and 8th bits if M = 0; 8th and 9th bits if M = 1) of the identifier character. The check is performed as specified by the LIN specification:



$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4$

$P1 = \overline{ID1 \oplus ID3 \oplus ID4 \oplus ID5}$

M = 0

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode) (cont'd)

10.7.9.4 LIN Error Detection

LIN Header Error Flag

The LIN Header Error Flag indicates that an invalid LIN Header has been detected.

When a LIN Header Error occurs:

– The LHE flag is set

– An interrupt is generated if the RIE bit is set and the I[1:0] bits are cleared in the CCR register.

If autosynchronization is enabled (LASE bit = 1), this can mean that the LIN Synch Field is corrupted, and that the SCI is in a blocked state (LSF bit is set). The only way to recover is to reset the LSF bit and then to clear the LHE bit.

– The LHE bit is reset by an access to the SCISR register followed by a read of the SCIDR register.

LHE/OVR Error Conditions

When Auto Resynchronization is disabled (LASE bit = 0), the LHE flag detects:

– That the received LIN Synch Field is not equal to 55h.

– That an overrun occurred (as in standard SCI mode)

– Furthermore, if LHDM is set it also detects that a LIN Header Reception Timeout occurred (only if LHDM is set).

When the LIN auto-resynchronization is enabled (LASE bit = 1), the LHE flag detects:

– That the deviation error on the Synch Field is outside the LIN specification which allows up to +/-15.5% of period deviation between the slave and master oscillators.

– A LIN Header Reception Timeout occurred. If $T_{HEADER} > T_{HEADER_MAX}$ then the LHE flag is set. Refer to Figure 6. (only if LHDM is set to 1)

– An overflow during the Synch Field Measurement, which leads to an overflow of the divider registers. If LHE is set due to this error then the SCI goes into a blocked state (LSF bit is set).

– That an overrun occurred on Fields other than the Synch Field (as in standard SCI mode)

Deviation Error on the Synch Field

The deviation error is checking by comparing the current baud rate (relative to the slave oscillator) with the received LIN Synch Field (relative to the master oscillator). Two checks are performed in parallel:

– The first check is based on a measurement between the first falling edge and the last falling edge of the Synch Field. Let us refer to this period deviation as D:

If the LHE flag is set, it means that:

D > 15.625%

If LHE flag is not set, it means that:

D < 16.40625%

If 15.625% ≤ D < 16.40625%, then the flag can be either set or reset depending on the dephasing between the signal on the RDI line and the CPU clock.

– The second check is based on the measurement of each bit time between both edges of the Synch Field: this checks that each of these bit times is large enough compared to the bit time of the current baud rate.

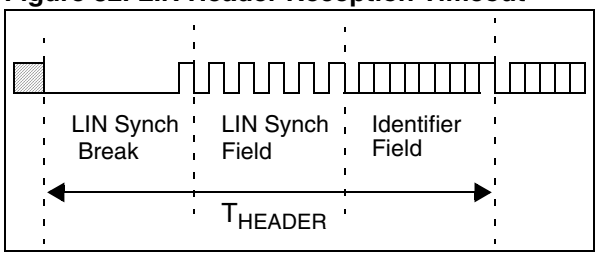
When LHE is set due to this error then the SCI goes into a blocked state (LSF bit is set).

LIN Header Time-out Error

When the LIN Identifier Field Detection Method is used (by configuring LHDM to 1) or when LIN auto-resynchronization is enabled (LASE bit = 1), the LINSCI automatically monitors the T_{HEADER_MAX} condition given by the LIN protocol.

If the entire Header (up to and including the STOP bit of the LIN Identifier Field) is not received within the maximum time limit of 57 bit times then a LIN Header Error is signalled and the LHE bit is set in the SCISR register.

Figure 82. LIN Header Reception Timeout



The time-out counter is enabled at each break detection. It is stopped in the following conditions:
- A LIN Identifier Field has been received
- An LHE error occurred (other than a timeout error).
- A software reset of LSF bit (transition from high to low) occurred during the analysis of the LIN Synch Field or

If LHE bit is set due to this error during the LIN Synchr Field (if LASE bit = 1) then the SCI goes into a blocked state (LSF bit is set).

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode) (cont'd)

If LHE bit is set due to this error during Fields other than LIN Synch Field or if LASE bit is reset then the current received Header is discarded and the SCI searches for a new Break Field.

Note on LIN Header Time-out Limit

According to the LIN specification, the maximum length of a LIN Header which does not cause a timeout is equal to 1.4 * (34 + 1) = 49 T_{BIT_MASTER}.

T_{BIT_MASTER} refers to the master baud rate.

When checking this timeout, the slave node is desynchronized for the reception of the LIN Break and Synch fields. Consequently, a margin must be allowed, taking into account the worst case: This occurs when the LIN identifier lasts exactly 10 T_{BIT_MASTER} periods. In this case, the LIN Break and Synch fields last 49 - 10 = 39T_{BIT_MASTER} periods.

Assuming the slave measures these first 39 bits with a desynchronized clock of 15.5%. This leads to a maximum allowed Header Length of:

39 x (1/0.845) T_{BIT_MASTER} + 10T_{BIT_MASTER}

= 56.15 T_{BIT_SLAVE}

A margin is provided so that the time-out occurs when the header length is greater than 57 T_{BIT_SLAVE} periods. If it is less than or equal to 57 T_{BIT_SLAVE} periods, then no timeout occurs.

LIN Header Length

Even if no timeout occurs on the LIN Header, it is possible to have access to the effective LIN header Length (T_{HEADER}) through the LHL register. This allows monitoring at software level the T_{FRAME_MAX} condition given by the LIN protocol.

This feature is only available when LHDM bit = 1 or when LASE bit = 1.

Mute Mode and Errors

In mute mode when LHDM bit = 1, if an LHE error occurs during the analysis of the LIN Synch Field or if a LIN Header Time-out occurs then the LHE bit is set but it does not wake up from mute mode. In this case, the current header analysis is discarded. If needed, the software has to reset LSF bit. Then the SCI searches for a new LIN header.

In mute mode, if a framing error occurs on a data (which is not a break), it is discarded and the FE bit is not set.

When LHDM bit = 1, any LIN header which respects the following conditions causes a wake-up from mute mode:

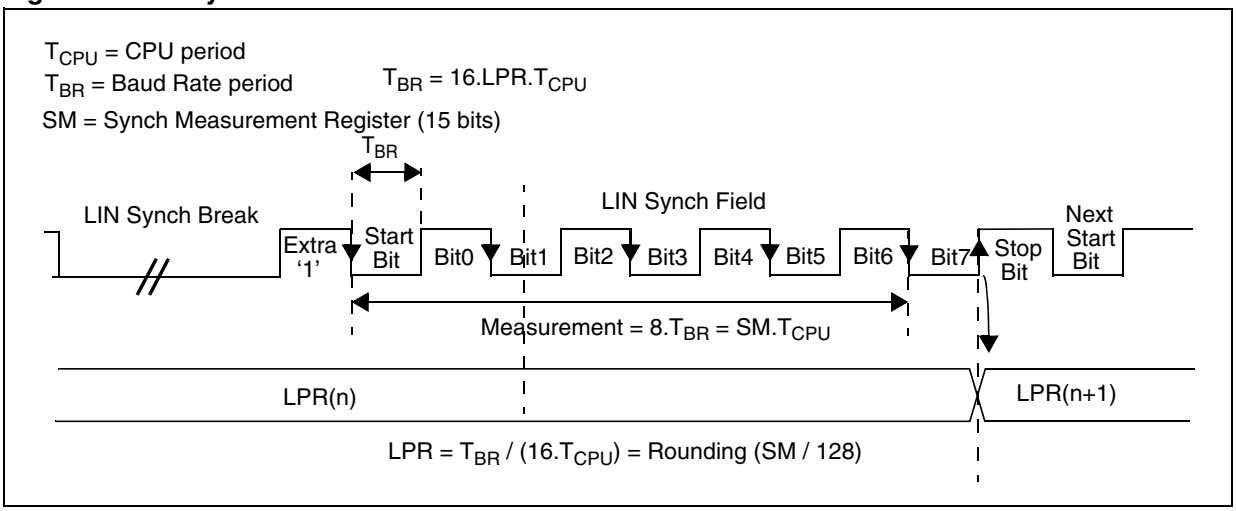
- A valid LIN Break Field (at least 11 dominant bits followed by a recessive bit)

- A valid LIN Synch Field (without deviation error)

- A LIN Identifier Field without framing error. Note that a LIN parity error on the LIN Identifier Field does not prevent wake-up from mute mode.

- No LIN Header Time-out should occur during Header reception.

Figure 83. LIN Synch Field Measurement



LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode) (cont'd)

10.7.9.5 LIN Baud Rate

Baud rate programming is done by writing a value in the LPR prescaler or performing an automatic resynchronization as described below.

Automatic Resynchronization

To automatically adjust the baud rate based on measurement of the LIN Synch Field:

– Write the nominal LIN Prescaler value (usually depending on the nominal baud rate) in the LPFR / LPR registers.

– Set the LASE bit to enable the Auto Synchronization Unit.

When Auto Synchronization is enabled, after each LIN Synch Break, the time duration between five falling edges on RDI is sampled on f_{CPU} and the result of this measurement is stored in an internal 15-bit register called SM (not user accessible) (see Figure 7). Then the LDIV value (and its associated LPFR and LPR registers) are automatically updated at the end of the fifth falling edge. During LIN Synch field measurement, the SCI state machine is stopped and no data is transferred to the data register.

10.7.9.6 LIN Slave Baud Rate Generation

In LIN mode, transmission and reception are driven by the LIN baud rate generator

Note: LIN Master mode uses the Extended or Conventional prescaler register to generate the baud rate.

If LINE bit = 1 and LSLV bit = 1 then the Conventional and Extended Baud Rate Generators are disabled: the baud rate for the receiver and transmitter are both set to the same value, depending on the LIN Slave baud rate generator:

$$Tx = Rx = \frac{f_{CPU}}{(16*LDIV)}$$

with:

LDIV is an unsigned fixed point number. The mantissa is coded on 8 bits in the LPR register and the fraction is coded on 4 bits in the LPFR register.

If LASE bit = 1 then LDIV is automatically updated at the end of each LIN Synch Field.

Three registers are used internally to manage the auto-update of the LIN divider (LDIV):

- LDIV_NOM (nominal value written by software at LPR/LPFR addresses)

- LDIV_MEAS (results of the Field Synch measurement)

- LDIV (used to generate the local baud rate)

The control and interactions of these registers, explained in Figure 8 and Figure 9, depend on the LDUM bit setting (LIN Divider Update Method).

Note:

As explained in Figure 8 and Figure 9, LDIV can be updated by two concurrent actions: a transfer from LDIV_MEAS at the end of the LIN Sync Field and a transfer from LDIV_NOM due to a software write of LPR. If both operations occur at the same time, the transfer from LDIV_NOM has priority.

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode) (cont'd)

Figure 84. LDIV Read / Write Operations When LDUM = 0

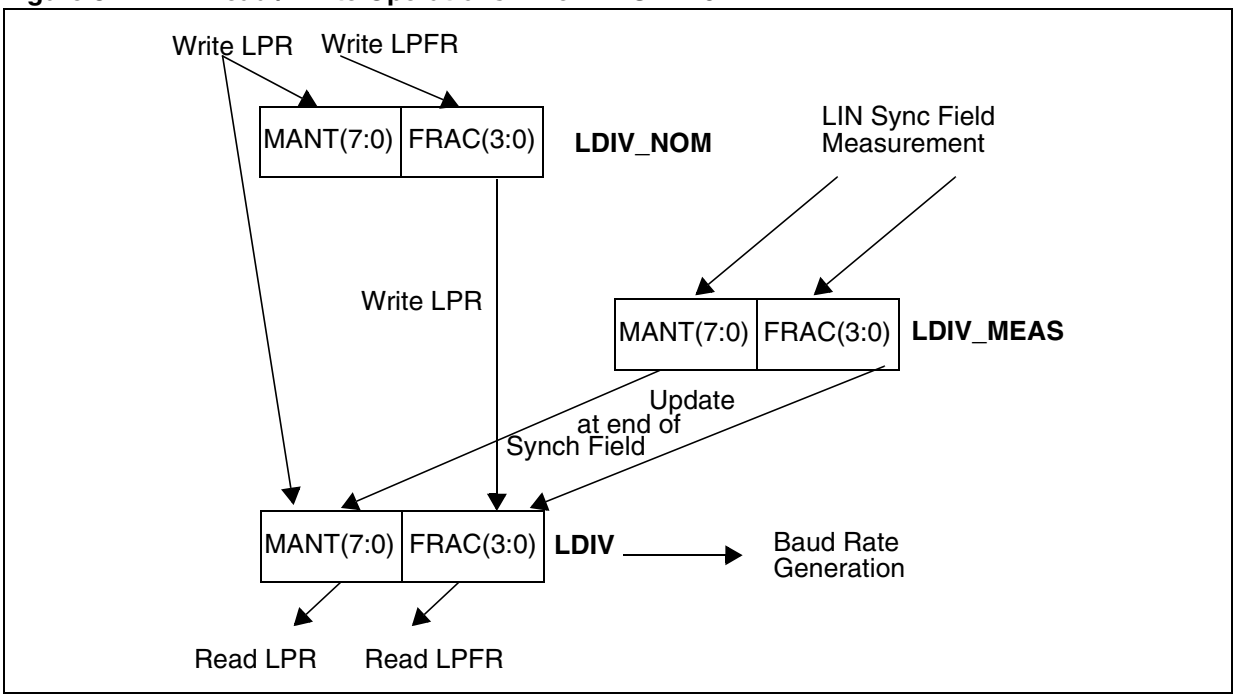
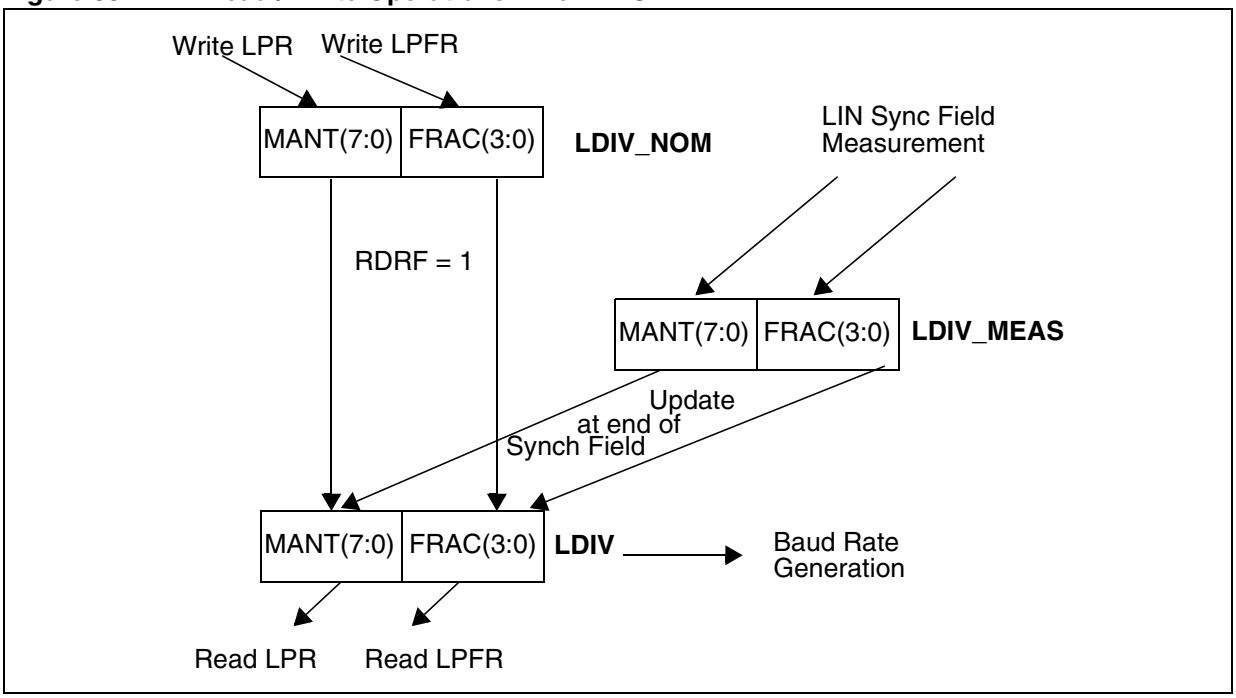


Figure 85. LDIV Read / Write Operations When LDUM = 1



LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode) (cont'd)

10.7.9.7 LINSCI Clock Tolerance

LINSCI Clock Tolerance when unsynchronized

When LIN slaves are unsynchronized (meaning no characters have been transmitted for a relatively long time), the maximum tolerated deviation of the LINSCI clock is +/-15%.

If the deviation is within this range then the LIN Synch Break is detected properly when a new reception occurs.

This is made possible by the fact that masters send 13 low bits for the LIN Synch Break, which can be interpreted as 11 low bits (13 bits -15% = 11.05) by a "fast" slave and then considered as a LIN Synch Break. According to the LIN specification, a LIN Synch Break is valid when its duration is greater than t_{SBRKTS} = 10. This means that the LIN Synch Break must last at least 11 low bits.

Note: If the period desynchronization of the slave is +15% (slave too slow), the character "00h" which represents a sequence of 9 low bits must not be interpreted as a break character (9 bits + 15% = 10.35). Consequently, a valid LIN Synch break must last at least 11 low bits.

LINSCI Clock Tolerance when Synchronized

When synchronization has been performed, following reception of a LIN Synch Break, the LINSCI, in LIN mode, has the same clock deviation tolerance as in SCI mode, which is explained below:

During reception, each bit is oversampled 16 times. The mean of the 8th, 9th and 10th samples is considered as the bit value.

Consequently, the clock frequency should not vary more than 6/16 (37.5%) within one bit.

The sampling clock is resynchronized at each start bit, so that when receiving 10 bits (one start bit, 1 data byte, 1 stop bit), the clock deviation should not exceed 3.75%.

10.7.9.8 Clock Deviation Causes

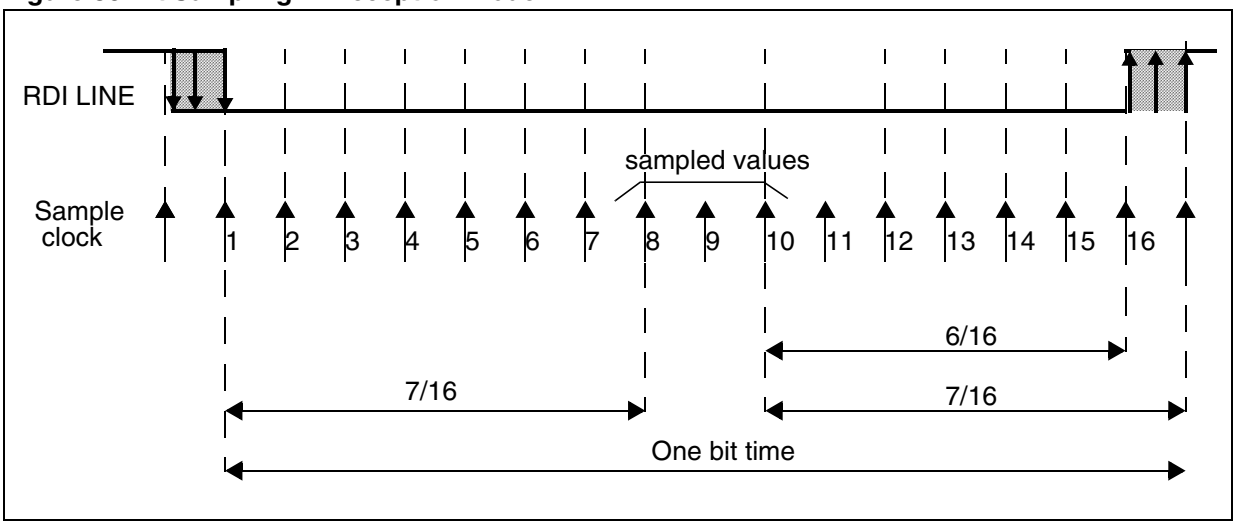
The causes which contribute to the total deviation are:

- D_{TRA}: Deviation due to transmitter error.
  Note: The transmitter can be either a master or a slave (in case of a slave listening to the response of another slave).
- D_{MEAS}: Error due to the LIN Synch measurement performed by the receiver.
- D_{QUANT}: Error due to the baud rate quantization of the receiver.
- D_{REC}: Deviation of the local oscillator of the receiver: This deviation can occur during the reception of one complete LIN message assuming that the deviation has been compensated at the beginning of the message.
- D_{TCL}: Deviation due to the transmission line (generally due to the transceivers)

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

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

Figure 86.Bit Sampling in Reception Mode



LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode) (cont'd)

10.7.9.9 Error due to LIN Synch measurement

The LIN Synch Field is measured over eight bit times.

This measurement is performed using a counter clocked by the CPU clock. The edge detections are performed using the CPU clock cycle.

This leads to a precision of 2 CPU clock cycles for the measurement which lasts 16*8*LDIV clock cycles.

Consequently, this error (D_{MEAS}) is equal to:

$2 / (128*LDIV_{MIN})$.

$LDIV_{MIN}$ corresponds to the minimum LIN prescaler content, leading to the maximum baud rate, taking into account the maximum deviation of +/-15%.

10.7.9.10 Error due to Baud Rate Quantization

The baud rate can be adjusted in steps of 1 / (16 * LDIV). The worst case occurs when the "real" baud rate is in the middle of the step.

This leads to a quantization error (D_{QUANT}) equal to $1 / (2*16*LDIV_{MIN})$.

10.7.9.11 Impact of Clock Deviation on Maximum Baud Rate

The choice of the nominal baud rate ($LDIV_{NOM}$) will influence both the quantization error (D_{QUANT}) and the measurement error (D_{MEAS}). The worst case occurs for $LDIV_{MIN}$.

Consequently, at a given CPU frequency, the maximum possible nominal baud rate (LPR_{MIN}) should be chosen with respect to the maximum tolerated deviation given by the equation:

$D_{TRA} + 2 / (128*LDIV_{MIN}) + 1 / (2*16*LDIV_{MIN}) + D_{REC} + D_{TCL} < 3.75\%$

Example:

A nominal baud rate of 20Kbits/s at T_{CPU} = 125ns (8 MHz) leads to $LDIV_{NOM}$ = 25d.

$LDIV_{MIN} = 25 - 0.15*25 = 21.25$

$D_{MEAS} = 2 / (128*LDIV_{MIN}) * 100 = 0.00073\%$

$D_{QUANT} = 1 / (2*16*LDIV_{MIN}) * 100 = 0.0015\%$

LIN Slave systems

For LIN Slave systems (the LINE and LSLV bits are set), receivers wake up by LIN Synch Break or LIN Identifier detection (depending on the LHDM bit).

Hot Plugging Feature for LIN Slave Nodes

In LIN Slave Mute Mode (the LINE, LSLV and RWU bits are set) it is possible to hot plug to a network during an ongoing communication flow. In this case the SCI monitors the bus on the RDI line until 11 consecutive dominant bits have been detected and discards all the other bits received.

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode) (cont'd)

10.7.10 LIN Mode Register Description

STATUS REGISTER (SCISR)
Read Only
Reset Value: 1100 0000 (C0h)

7							0
TDRE	TC	RDRF	IDLE	LHE	NF	FE	PE

Bits 7:4 = Same function as in SCI mode; please refer to Section 0.1.8 SCI Mode Register Description.

Bit 3 = **LHE** *LIN Header Error.*
During LIN Header this bit signals three error types:

– The LIN Synch Field is corrupted and the SCI is blocked in LIN Synch State (LSF bit = 1).

– A timeout occurred during LIN Header reception

– An overrun error was detected on one of the header field (see OR bit description in Section 0.1.8 SCI Mode Register Description).

An interrupt is generated if RIE = 1 in the SCICR2 register. If blocked in the LIN Synch State, the LSF bit must first be reset (to exit LIN Synch Field state and then to be able to clear LHE flag). Then it is cleared by the following software sequence: An access to the SCISR register followed by a read to the SCIDR register.
0: No LIN Header error
1: LIN Header error detected

Note:

Apart from the LIN Header this bit signals an Overrun Error as in SCI mode (see description in Section 0.1.8 SCI Mode Register Description).

Bit 2 = **NF** *Noise flag*

In LIN Master mode (LINE bit = 1 and LSLV bit = 0), this bit has the same function as in SCI mode; please refer to Section 0.1.8 SCI Mode Register Description.

In LIN Slave mode (LINE bit = 1 and LSLV bit = 1) this bit has no meaning.

Bit 1 = **FE** *Framing error.*
In LIN slave mode, this bit is set only when a real framing error is detected (if the stop bit is dominant (0) and at least one of the other bits is recessive (1). It is not set when a break occurs, the LHDF bit is used instead as a break flag (if the LHDM bit = 0). It is cleared by a software sequence (an access to the SCISR register followed by a read to the SCIDR register).
0: No Framing error
1: Framing error detected

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

CONTROL REGISTER 1 (SCICR1)
Read/Write
Reset Value: x000 0000 (x0h)

7							0
R8	T8	SCID	M	WAKE	PCE	PS	PIE

Bits 7:3 = Same function as in SCI mode; please refer to Section 0.1.8 SCI Mode Register Description.

Bit 2 = **PCE** *Parity control enable.*
This bit is set and cleared by software. It selects the hardware parity control for LIN identifier parity check.
0: Parity control disabled
1: Parity control enabled

When a parity error occurs, the PE bit in the SCISR register is set.

Bit 1 = Reserved

Bit 0 = Same function as in SCI mode; please refer to Section 0.1.8 SCI Mode Register Description.

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode) (cont'd)

CONTROL REGISTER 2 (SCICR2)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK

Bits 7:2 Same function as in SCI mode; please refer to Section 0.1.8 SCI Mode Register Description.

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

Notes:

– Mute mode is recommended for detecting only the Header and avoiding the reception of any other characters. For more details, please refer to Section 0.1.9.3 LIN Reception.

– In LIN slave mode, when RDRF is set, the software can not set or clear the RWU bit.

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

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

CONTROL REGISTER 3 (SCICR3)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
LDUM	LINE	LSLV	LASE	LHDM	LHIE	LHDF	LSF

Bit 7 = **LDUM** *LIN Divider Update Method.*
This bit is set and cleared by software and is also cleared by hardware (when RDRF = 1). It is only used in LIN Slave mode. It determines how the LIN Divider can be updated by software.
0: LDIV is updated as soon as LPR is written (if no Auto Synchronization update occurs at the same time).
1: LDIV is updated at the next received character (when RDRF = 1) after a write to the LPR register

Notes:

- If no write to LPR is performed between the setting of LDUM bit and the reception of the next character, LDIV will be updated with the old value.

- After LDUM has been set, it is possible to reset the LDUM bit by software. In this case, LDIV can be modified by writing into LPR / LPFR registers.

Bits 6:5 = **LINE, LSLV** *LIN Mode Enable Bits.*
These bits configure the LIN mode:

LINE	LSLV	Meaning
0	x	LIN mode disabled
1	0	LIN Master Mode
	1	LIN Slave Mode

The LIN Master configuration enables:

The capability to send LIN Synch Breaks (13 low bits) using the SBK bit in the SCICR2 register.

The LIN Slave configuration enables:

– The LIN Slave Baud Rate generator. The LIN Divider (LDIV) is then represented by the LPR and LPFR registers. The LPR and LPFR registers are read/write accessible at the address of the SCIBRR register and the address of the SCIETPR register

– Management of LIN Headers.

– LIN Synch Break detection (11-bit dominant).

– LIN Wake-Up method (see LHDM bit) instead of the normal SCI Wake-Up method.

– Inhibition of Break transmission capability (SBK has no effect)

– LIN Parity Checking (in conjunction with the PCE bit)

Bit 4 = **LASE** *LIN Auto Synch Enable.*
This bit enables the Auto Synch Unit (ASU). It is set and cleared by software. It is only usable in LIN Slave mode.

0: Auto Synch Unit disabled
1: Auto Synch Unit enabled.

Bit 3 = **LHDM** *LIN Header Detection Method*
This bit is set and cleared by software. It is only usable in LIN Slave mode. It enables the Header Detection Method. In addition if the RWU bit in the

SCICR2 register is set, the LHDM bit selects the Wake-Up method (replacing the WAKE bit).
0: LIN Synch Break Detection Method
1: LIN Identifier Field Detection Method

Bit 2 = **LHIE** *LIN Header Interrupt Enable*
This bit is set and cleared by software. It is only usable in LIN Slave mode.
0: LIN Header Interrupt is inhibited.
1: An SCI interrupt is generated whenever LHDF = 1.

Bit 1 = **LHDF** *LIN Header Detection Flag*
This bit is set by hardware when a LIN Header is detected and cleared by a software sequence (an access to the SCISR register followed by a read of the SCICR3 register). It is only usable in LIN Slave mode.
0: No LIN Header detected.
1: LIN Header detected.

Notes: The header detection method depends on the LHDM bit:

– If LHDM = 0, a header is detected as a LIN Synch Break.

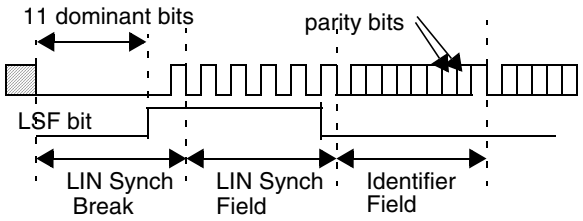
– If LHDM = 1, a header is detected as a LIN Identifier, meaning that a LIN Synch Break Field + a LIN Synch Field + a LIN Identifier Field have been consecutively received.

Bit 0 = **LSF** *LIN Synch Field State*

This bit indicates that the LIN Synch Field is being analyzed. It is only used in LIN Slave mode. In Auto Synchronization Mode (LASE bit = 1), when the SCI is in the LIN Synch Field State it waits or counts the falling edges on the RDI line.

It is set by hardware as soon as a LIN Synch Break is detected and cleared by hardware when the LIN Synch Field analysis is finished (see Figure 11). This bit can also be cleared by software to exit LIN Synch State and return to idle mode.
0: The current character is not the LIN Synch Field
1: LIN Synch Field State (LIN Synch Field undergoing analysis)

Figure 87. LSF Bit Set and Clear



LIN DIVIDER REGISTERS

LDIV is coded using the two registers LPR and LPFR. In LIN Slave mode, the LPR register is accessible at the address of the SCIBRR register and the LPFR register is accessible at the address of the SCIETPR register.

LIN PRESCALER REGISTER (LPR)
Read/Write
Reset Value: 0000 0000 (00h)

7							0
LPR7	LPR6	LPR5	LPR4	LPR3	LPR2	LPR1	LPR0

LPR[7:0] *LIN Prescaler (mantissa of LDIV)*

These 8 bits define the value of the mantissa of the LIN Divider (LDIV):

LPR[7:0]	Rounded Mantissa (LDIV)
00h	SCI clock disabled
01h	1
...	...
FEh	254
FFh	255

Caution: LPR and LPFR registers have different meanings when reading or writing to them. Consequently bit manipulation instructions (BRES or BSET) should never be used to modify the LPR[7:0] bits, or the LPFR[3:0] bits.

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode) (cont'd)

LIN PRESCALER FRACTION REGISTER (LPFR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	LPFR 3	LPFR 2	LPFR 1	LPFR 0

Bits 7:4 = Reserved.

Bits 3:0 = **LPFR[3:0]** *Fraction of LDIV*

These 4 bits define the fraction of the LIN Divider (LDIV):

LPFR[3:0]	Fraction (LDIV)
0h	0
1h	1/16
...	...
Eh	14/16
Fh	15/16

1. When initializing LDIV, the LPFR register must be written first. Then, the write to the LPR register will effectively update LDIV and so the clock generation.
2. In LIN Slave mode, if the LPR[7:0] register is equal to 00h, the transceiver and receiver input clocks are switched off.

Examples of LDIV coding:

Example 1: LPR = 27d and LPFR = 12d

This leads to:

Mantissa (LDIV) = 27d

Fraction (LDIV) = 12/16 = 0.75d

Therefore LDIV = 27.75d

Example 2: LDIV = 25.62d

This leads to:

LPFR = rounded(16*0.62d)

= rounded(9.92d) = 10d = Ah

LPR = mantissa (25.620d) = 25d = 1Bh

Example 3: LDIV = 25.99d

This leads to:

LPFR = rounded(16*0.99d)

= rounded(15.84d) = 16d

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Mode) (cont'd)

LIN HEADER LENGTH REGISTER (LHLR)
Read Only
Reset Value: 0000 0000 (00h).

7							0
LHL7	LHL6	LHL5	LHL4	LHL3	LHL2	LHL1	LHL0

Note: In LIN Slave mode when LASE = 1 or LHDM = 1, the LHLR register is accessible at the address of the SCIERPR register.

Otherwise this register is always read as 00h.

Bits 7:0 = **LHL[7:0]** *LIN Header Length.*

This is a read-only register, which is updated by hardware if one of the following conditions occurs:
- After each break detection, it is loaded with "FFh".
- If a timeout occurs on T_{HEADER}, it is loaded with 00h.
- After every successful LIN Header reception (at the same time than the setting of LHDF bit), it is loaded with a value (LHL) which gives access to the number of bit times of the LIN header length (T_{HEADER}). The coding of this value is explained below:

LHL Coding:

T_{HEADER_MAX} = 57
LHL(7:2) represents the mantissa of (57 - T_{HEADER})
LHL(1:0) represents the fraction (57 - T_{HEADER})

LHL[7:2]	Mantissa (57 - T_{HEADER})	Mantissa (T_{HEADER})
0h	0	57
1h	1	56
...	...	...
39h	56	1
3Ah	57	0
3Bh	58	Never Occurs
...	...	...
3Eh	62	Never Occurs
3Fh	63	Initial value

LHL[1:0]	Fraction (57 - T_{HEADER})
0h	0
1h	1/4
2h	1/2
3h	3/4

Example of LHL coding:

Example 1: LHL = 33h = 001100 11b
LHL(7:3) = 1100b = 12d
LHL(1:0) = 11b = 3d
This leads to:
Mantissa (57 - T_{HEADER}) = 12d
Fraction (57 - T_{HEADER}) = 3/4 = 0.75
Therefore:
(57 - T_{HEADER}) = 12.75d
and T_{HEADER} = 44.25d

Example 2:
57 - T_{HEADER} = 36.21d
LHL(1:0) = rounded(4*0.21d) = 1d
LHL(7:2) = Mantissa (36.21d) = 36d = 24h
Therefore LHL(7:0) = 10010001 = 91h

Example 3:
57 - T_{HEADER} = 36.90d
LHL(1:0) = rounded(4*0.90d) = 4d
The carry must be propagated to the matissa:
LHL(7:2) = Mantissa (36.90d) + 1 = 37d =
Therefore LHL(7:0) = 10110000 = A0h

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master/Slave) (Cont'd)

Table 24. LINSCI1 Register Map and Reset Values

Addr. (Hex.)	Register Name	7	6	5	4	3	2	1	0
48	**SCI1SR**	TDRE	TC	RDRF	IDLE	OR/LHE	NF	FE	PE
	Reset Value	1	1	0	0	0	0	0	0
49	**SCI1DR**	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0
	Reset Value	-	-	-	-	-	-	-	-
4A	**SCI1BRR**	SCP1	SCP0	SCT2	SCT1	SCT0	SCR2	SCR1	SCR0
	LPR (LIN Slave Mode)	LPR7	LPR6	LPR5	LPR4	LPR3	LPR2	LPR1	LPR0
	Reset Value	0	0	0	0	0	0	0	0
4B	**SCI1CR1**	R8	T8	SCID	M	WAKE	PCE	PS	PIE
	Reset Value	x	0	0	0	0	0	0	0
4C	**SCI1CR2**	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
	Reset Value	0	0	0	0	0	0	0	0
4D	**SCI1CR3**	LDUM	LINE	LSLV	LASE	LHDM	LHIE	LHDF	LSF
	Reset Value	0	0	0	0	0	0	0	0
4E	**SCI1ERPR**	ERPR7	ERPR6	ERPR5	ERPR4	ERPR3	ERPR2	ERPR1	ERPR0
	LHLR (LIN Slave Mode)	LHL7	LHL6	LHL5	LHL4	LHL3	LHL2	LHL1	LHL0
	Reset Value	0	0	0	0	0	0	0	0
4F	**SCI1ETPR**	ETPR7	ETPR6	ETPR5	ETPR4	ETPR3	ETPR2	ETPR1	ETPR0
	LPFR (LIN Slave Mode)	0	0	0	0	LPFR3	LPFR2	LPFR1	LPFR0
	Reset Value	0	0	0	0	0	0	0	0

10.8 LINSCI SERIAL COMMUNICATION INTERFACE (LIN Master Only)

10.8.1 Introduction

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

10.8.2 Main Features

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

10.8.3 General Description

The interface is externally connected to another device by three pins (see Figure 88 on page 153). Any SCI bidirectional communication requires a minimum of two pins: Receive Data In (RDI) and Transmit Data Out (TDO):

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

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

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

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

– An Idle Line prior to transmission or reception

– A start bit

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

– A Stop bit indicating that the frame is complete.

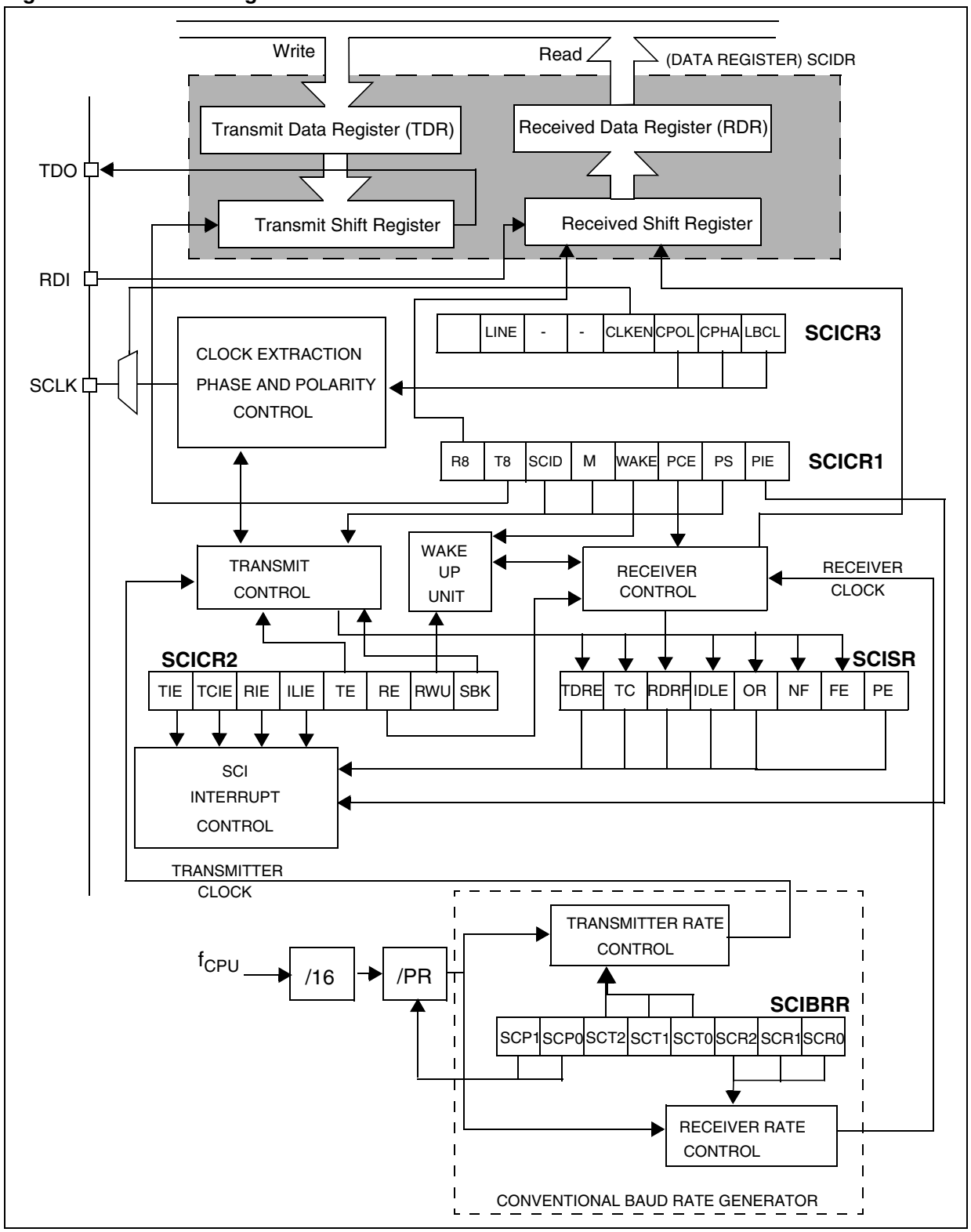
This interface uses two types of baud rate generator:

– A conventional type for commonly-used baud rates,

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

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

Figure 88. SCI Block Diagram



LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

10.8.4 Functional Description

The block diagram of the Serial Control Interface, is shown in Figure 88 on page 153. It contains seven dedicated registers:

– Three control registers (SCICR1, SCICR2 and SCICR3)

– A status register (SCISR)

– A baud rate register (SCIBRR)

– An extended prescaler receiver register (SCIERPR)

– An extended prescaler transmitter register (SCIETPR)

Refer to the register descriptions in Section 10.7.8for the definitions of each bit.

10.8.4.1 Serial Data Format

Word length may be selected as being either 8 or 9 bits by programming the M bit in the SCICR1 register (see Figure 89).

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

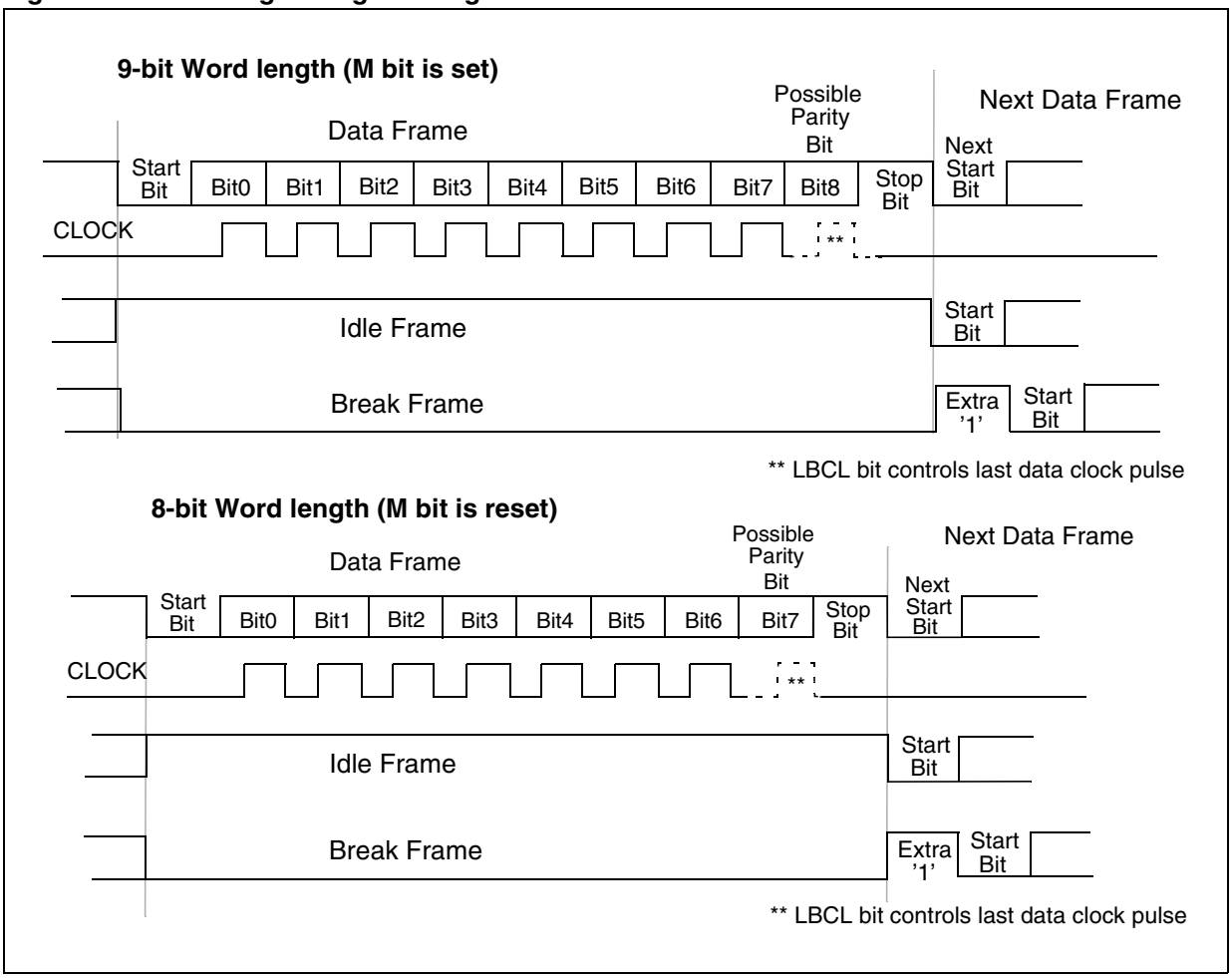
The TDO pin is in high state during the stop bit.

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

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

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

Figure 89. Word Length Programming



LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

10.8.4.2 Transmitter

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

When the transmit enable bit (TE) is set, the data in the transmit shift register is output on the TDO pin and the corresponding clock pulses are output on the SCLK pin.

Character Transmission

During an SCI transmission, data shifts out least significant bit first on the TDO pin. In this mode, the SCIDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see Figure 89).

Procedure

– Select the M bit to define the word length.

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

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

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

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

The TDRE bit is set by hardware and it indicates:

– The TDR register is empty.

– The data transfer is beginning.

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

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

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

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

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

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

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

Break Characters

Setting the SBK bit loads the shift register with a break character. The break frame length depends on the M bit (see Figure 89).

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

Idle Characters

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

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

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

LIN Transmission

The same procedure has to be applied for LIN Master transmission with the following differences:

– Clear the M bit to configure 8-bit word length.

– Set the LINE bit to enter LIN master mode. In this case, setting the SBK bit sends 13 low bits.

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

10.8.4.3 Receiver

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

Character reception

During a SCI reception, data shifts in least significant bit first through the RDI pin. In this mode, the SCIDR register consists or a buffer (RDR) between the internal bus and the received shift register (see Figure 88 on page 153).

Procedure

– Select the M bit to define the word length.

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

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

When a character is received:

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

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

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

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

1. An access to the SCISR register

2. A read to the SCIDR register.

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

Break Character

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

Idle Character

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

Overrun Error

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

When a overrun error occurs:

– The OR bit is set.

– The RDR content is not lost.

– The shift register is overwritten.

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

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

Noise Error

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

When noise is detected in a frame:

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

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

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

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

Framing Error

A framing error is detected when:

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

– A break is received.

When the framing error is detected:

– the FE bit is set by hardware

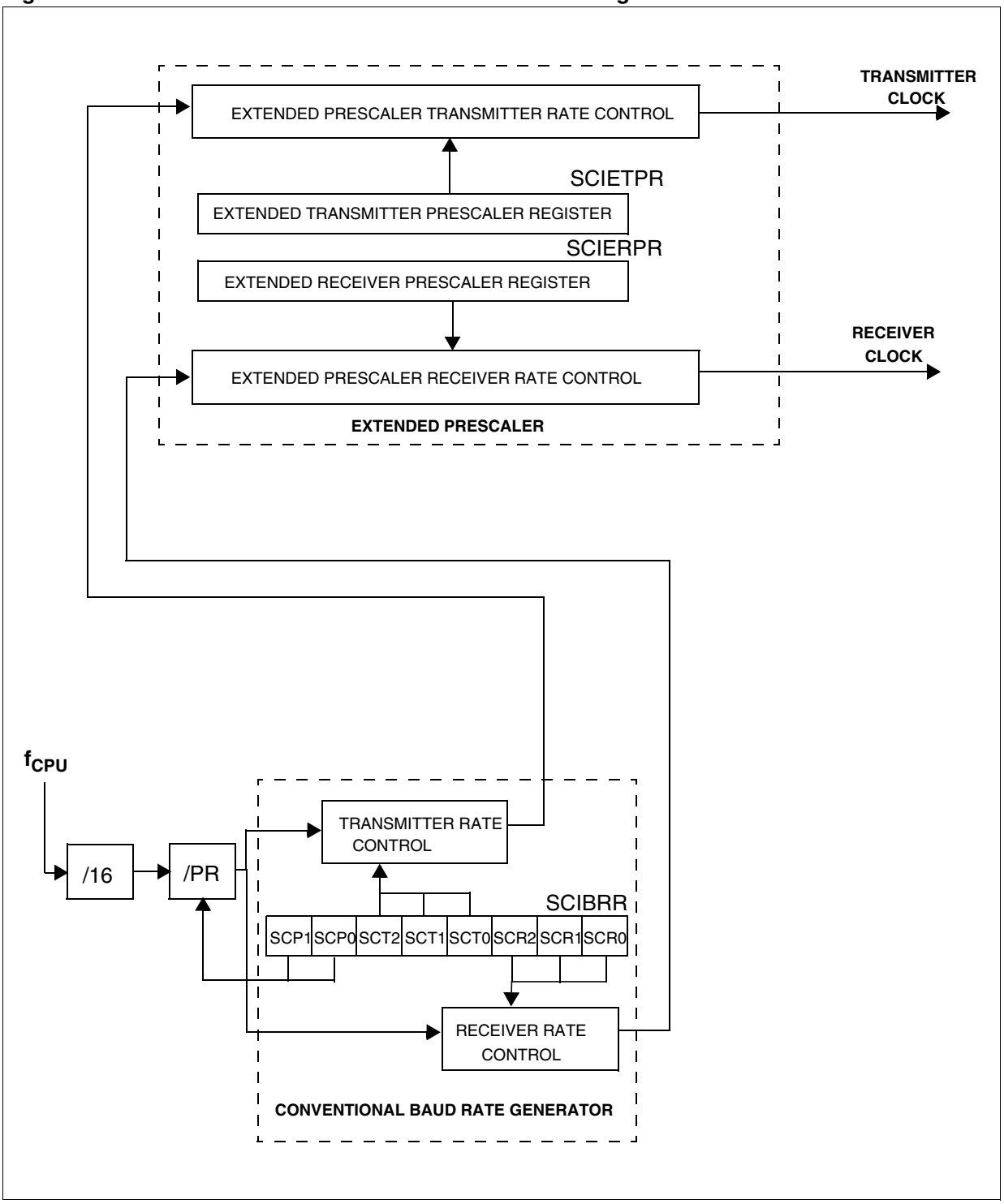
– Data is transferred from the Shift register to the SCIDR register.

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

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

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

Figure 90. SCI Baud Rate and Extended Prescaler Block Diagram



LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

10.8.4.4 Conventional Baud Rate Generation

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

:

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

with:

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

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

(see SCT[2:0] bits)

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

(see SCR[2:0] bits)

All these bits are in the SCIBRR register.

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

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

10.8.4.5 Extended Baud Rate Generation

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

The extended baud rate generator block diagram is described in the Figure 90.

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

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

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

with:

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

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

10.8.4.6 Receiver Muting and Wake-up Feature

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

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

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

All the reception status bits cannot be set.

All the receive interrupts are inhibited.

A muted receiver may be awakened by one of the following two ways:

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

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

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

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

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

10.8.4.7 Parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the SCICR1 register. Depending on the frame length defined by the M bit, the possible SCI frame formats are as listed in Table 24.

Table 25. Frame Formats

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

Legend:

SB: Start Bit

STB: Stop Bit

PB: Parity Bit

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

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

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

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

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

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

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

10.8.5 Low Power Modes

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

10.8.6 Interrupts

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

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

These events generate an interrupt if the corresponding Enable Control Bit is set and the interrupt mask in the CC register is reset (RIM instruction).

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

10.8.7 SCI Synchronous Transmission

The SCI transmitter allows the user to control a one way synchronous serial transmission. The SCLK pin is the output of the SCI transmitter clock. No clock pulses are sent to the SCLK pin during start bit and stop bit. Depending on the state of the LBCL bit in the SCICR3 register, clock pulses are or are not be generated during the last valid data bit (address mark). The CPOL bit in the SCICR3 register allows the user to select the clock polarity, and the CPHA bit in the SCICR3 register allows the user to select the phase of the external clock (see Figure 91, Figure 92 and Figure 93).

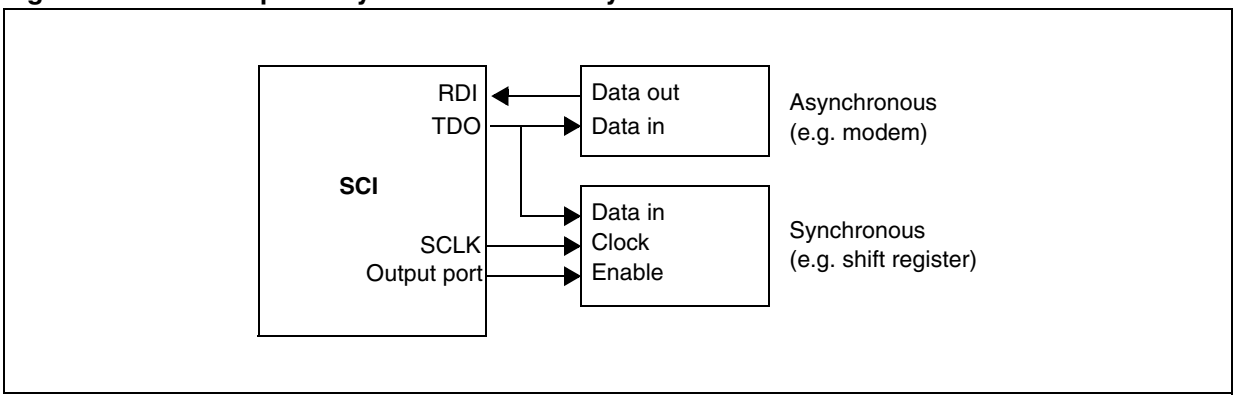
During idle, preamble and send break, the external SCLK clock is not activated.

These options allow the user to serially control peripherals which consist of shift registers, without losing any functions of the SCI transmitter which can still talk to other SCI receivers. These options do not affect the SCI receiver which is independent from the transmitter.

Note: The SCLK pin works in conjunction with the TDO pin. When the SCI transmitter is disabled (TE and RE = 0), the SCLK and TDO pins go into high impedance state.

Note: The LBCL, CPOL and CPHA bits have to be selected before enabling the transmitter to ensure that the clock pulses function correctly. These bits should not be changed while the transmitter is enabled.

Figure 91. SCI Example of Synchronous and Asynchronous Transmission



LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

Figure 92. SCI Data Clock Timing Diagram (M = 0)

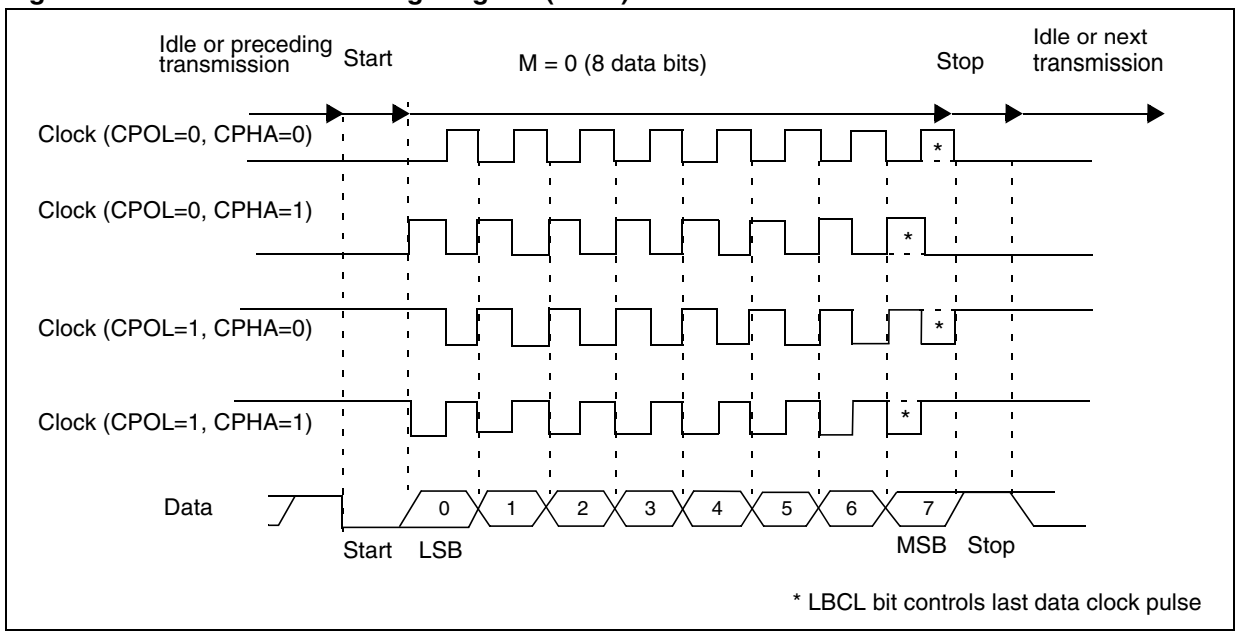
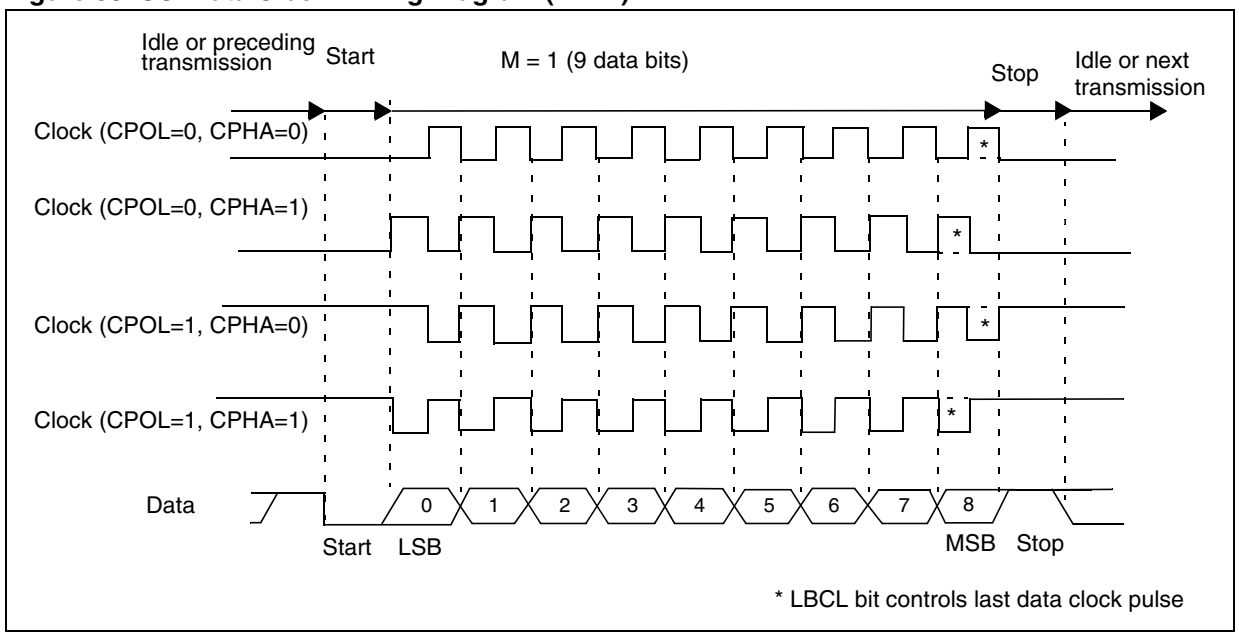


Figure 93. SCI Data Clock Timing Diagram (M = 1)



LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

10.8.8 Register Description

STATUS REGISTER (SCISR)
Read Only
Reset Value: 1100 0000 (C0h)

7							0
TDRE	TC	RDRF	IDLE	OR	NF	FE	PE

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

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

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

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

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

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

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

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

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

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

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

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

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

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

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

CONTROL REGISTER 1 (SCICR1)

Read/Write

Reset Value: x000 0000 (x0h)

7							0
R8	T8	SCID	M	WAKE	PCE	PS	PIE

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

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

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

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

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

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

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

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

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

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

CONTROL REGISTER 2 (SCICR2)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK

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

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

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

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

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

Notes:

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

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

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

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

Notes:

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

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

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

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

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

CONTROL REGISTER 3 (SCICR3)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
-	LINE	-	-	CLKEN	CPOL	CPHA	LBCL

Bit 7 = Reserved, must be kept cleared.

Bit 6 = **LINE** *LIN Mode Enable.*
This bit is set and cleared by software.
0: LIN Mode disabled
1: LIN Master mode enabled

The LIN Master mode enables the capability to send LIN Synch Breaks (13 low bits) using the SBK bit in the SCICR2 register

.In transmission, the LIN Synch Break low phase duration is shown as below:

LINE	M	Number of low bits sent during a LIN Synch Break
0	0	10
	1	11
1	0	13
	1	14

Bits 5:4 = Reserved, forced by hardware to 0.
These bits are not used.

Bit 3 = **CLKEN** *Clock Enable.*
This bit allows the user to enable the SCLK pin.
0: SLK pin disabled
1: SLK pin enabled

Bit 2 = **CPOL** *Clock Polarity.*
This bit allows the user to select the polarity of the clock output on the SCLK pin. It works in conjunction with the CPHA bit to produce the desired clock/data relationship (see Figure 92 and Figure 93).
0: Steady low value on SCLK pin outside transmission window.
1: Steady high value on SCLK pin outside transmission window.

Bit 1 = **CPHA** *Clock Phase*.
This bit allows the user to select the phase of the clock output on the SCLK pin. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see Figure 92 and Figure 93)
0: SCLK clock line activated in middle of data bit.
1: SCLK clock line activated at beginning of data bit.

Bit 0 = **LBCL** *Last bit clock pulse.*
This bit allows the user to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the SCLK pin.
0: The clock pulse of the last data bit is not output to the SCLK pin.
1: The clock pulse of the last data bit is output to the SCLK pin.

Note: The last bit is the 8th or 9th data bit transmitted depending on the 8 or 9 bit format selected by the M bit in the SCICR1 register.

Table 26. SCI clock on SCLK pin

Data format	M bit	LBCL bit	Number of clock pulses on SCLK
8 bit	0	0	7
		1	8
9 bit	1	0	8
		1	9

Note: These 3 bits (**CPOL**, **CPHA**, **LBCL**) should not be written while the transmitter is enabled.

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

DATA REGISTER (SCIDR)

Read/Write

Reset Value: Undefined

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

7							0
DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0

The Data register performs a double function (read and write) since it is composed of two registers, one for transmission (TDR) and one for reception (RDR).
The TDR register provides the parallel interface between the internal bus and the output shift register (see Figure 88 on page 153).
The RDR register provides the parallel interface between the input shift register and the internal bus (see Figure 88).

BAUD RATE REGISTER (SCIBRR)

Read/Write

Reset Value: 0000 0000 (00h)

7							0
SCP1	SCP0	SCT2	SCT1	SCT0	SCR2	SCR1	SCR0

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

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

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

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

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

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

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

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

LINSCI™ SERIAL COMMUNICATION INTERFACE (LIN Master Only) (Cont'd)

EXTENDED RECEIVE PRESCALER DIVISION REGISTER (SCIERPR)

Read/Write

Reset Value: 0000 0000 (00h)

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

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

The extended Baud Rate Generator is activated when a value other than 00h is stored in this register. The clock frequency from the 16 divider (see Figure 90) is divided by the binary factor set in the SCIERPR register (in the range 1 to 255).

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

EXTENDED TRANSMIT PRESCALER DIVISION REGISTER (SCIETPR)

Read/Write

Reset Value:0000 0000 (00h)

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

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

The extended Baud Rate Generator is activated when a value other than 00h is stored in this register. The clock frequency from the 16 divider (see Figure 90) is divided by the binary factor set in the SCIETPR register (in the range 1 to 255).

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

Table 27. Baud Rate Selection

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

LINSCI™ SERIAL COMMUNICATIONS INTERFACE (LIN Master) (Cont'd)

Table 28. LINSCI2 Register Map and Reset Values

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
60	**SCI2SR**	TDRE	TC	RDRF	IDLE	OR	NF	FE	PE
	Reset Value	1	1	0	0	0	0	0	0
61	**SCI2DR**	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0
	Reset Value	-	-	-	-	-	-	-	-
62	**SCI2BRR**	SCP1	SCP0	SCT2	SCT1	SCT0	SCR2	SCR1	SCR0
	Reset Value	0	0	0	0	0	0	0	0
63	**SCI2CR1**	R8	T8	SCID	M	WAKE	PCE	PS	PIE
	Reset Value	-	-	-	-	-			
64	**SCI2CR2**	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
	Reset Value	0	0	0	0	0	0	0	0
65	**SCI2CR3**		LINE	-	-	CLKEN	CPOL	CPHA	LBCL
	Reset Value	0	0	0	0	0	0	0	0
66	**SCI2ERPR**	ERPR7	ERPR6	ERPR5	ERPR4	ERPR3	ERPR2	ERPR1	ERPR0
	Reset Value	0	0	0	0	0	0	0	0
67	**SCI2ETPR**	ETPR7	ETPR6	ETPR5	ETPR4	ETPR3	ETPR2	ETPR1	ETPR0
	Reset Value	0	0	0	0	0	0	0	0

10.9 beCAN CONTROLLER (beCAN)

The beCAN controller (Basic Enhanced CAN), interfaces the CAN network and supports the CAN protocol version 2.0A and B. It has been designed to manage high number of incoming messages efficiently with a minimum CPU load. It also meets the priority requirements for transmit messages.

10.9.1 Main Features

- Supports CAN protocol version 2.0 A, B Active
- Bit rates up to 1Mbit/s

Transmission

- 2 transmit mailboxes
- Configurable transmit priority

Reception

- 1 receive FIFO with three stages
- 6 scalable filter banks
- Identifier list feature
- Configurable FIFO overrun

Management

- Maskable interrupts
- Software-efficient mailbox mapping at a unique address space

10.9.2 General Description

In today's CAN applications, the number of nodes in a network is increasing and often several networks are linked together via gateways. Typically the number of messages in the system (and thus to be handled by each node) has significantly increased. In addition to the application messages, Network Management and Diagnostic messages have been introduced.

– An enhanced filtering mechanism is required to handle each type of message.

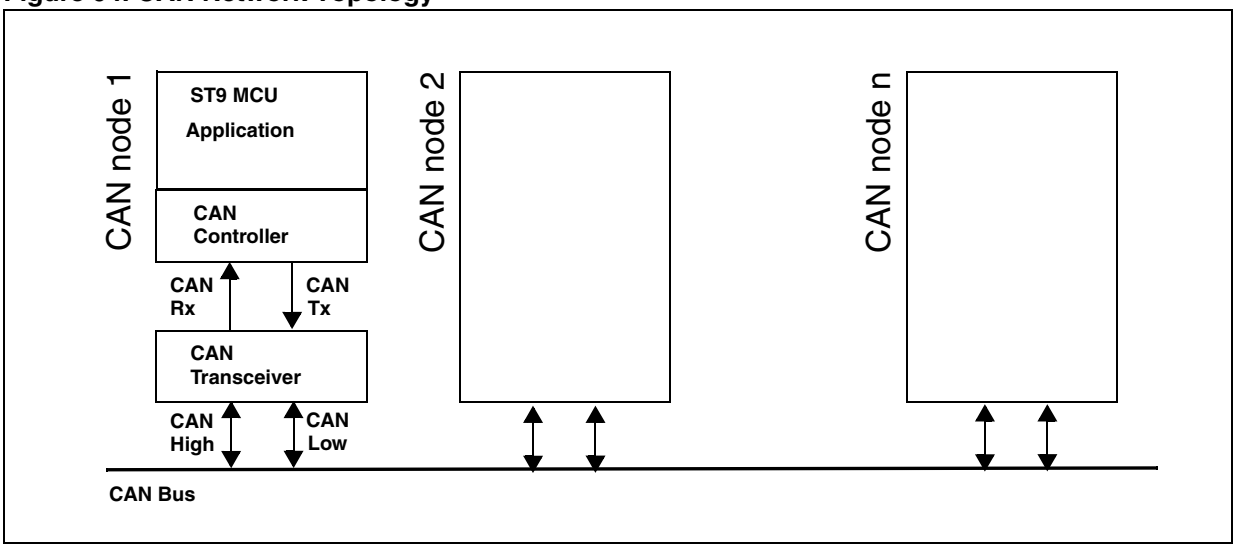
Furthermore, application tasks require more CPU time, therefore real-time constraints caused by message reception have to be reduced.

– A receive FIFO scheme allows the CPU to be dedicated to application tasks for a long time period without losing messages.

The standard HLP (Higher Layer Protocol) based on standard CAN drivers requires an efficient interface to the CAN controller.

– All mailboxes and registers are organized in 16-byte pages mapped at the same address and selected via a page select register.

Figure 94. CAN Network Topology



beCAN CONTROLLER (Cont'd)

CAN 2.0B Active Core

The beCAN module handles the transmission and the reception of CAN messages fully autonomously. Standard identifiers (11-bit) and extended identifiers (29-bit) are fully supported by hardware.

Control, Status and Configuration Registers

The application uses these registers to:

– Configure CAN parameters, e.g.baud rate

– Request transmissions

– Handle receptions

– Manage interrupts

– Get diagnostic information

Tx Mailboxes

Two transmit mailboxes are provided to the software for setting up messages. The Transmission Scheduler decides which mailbox has to be transmitted first.

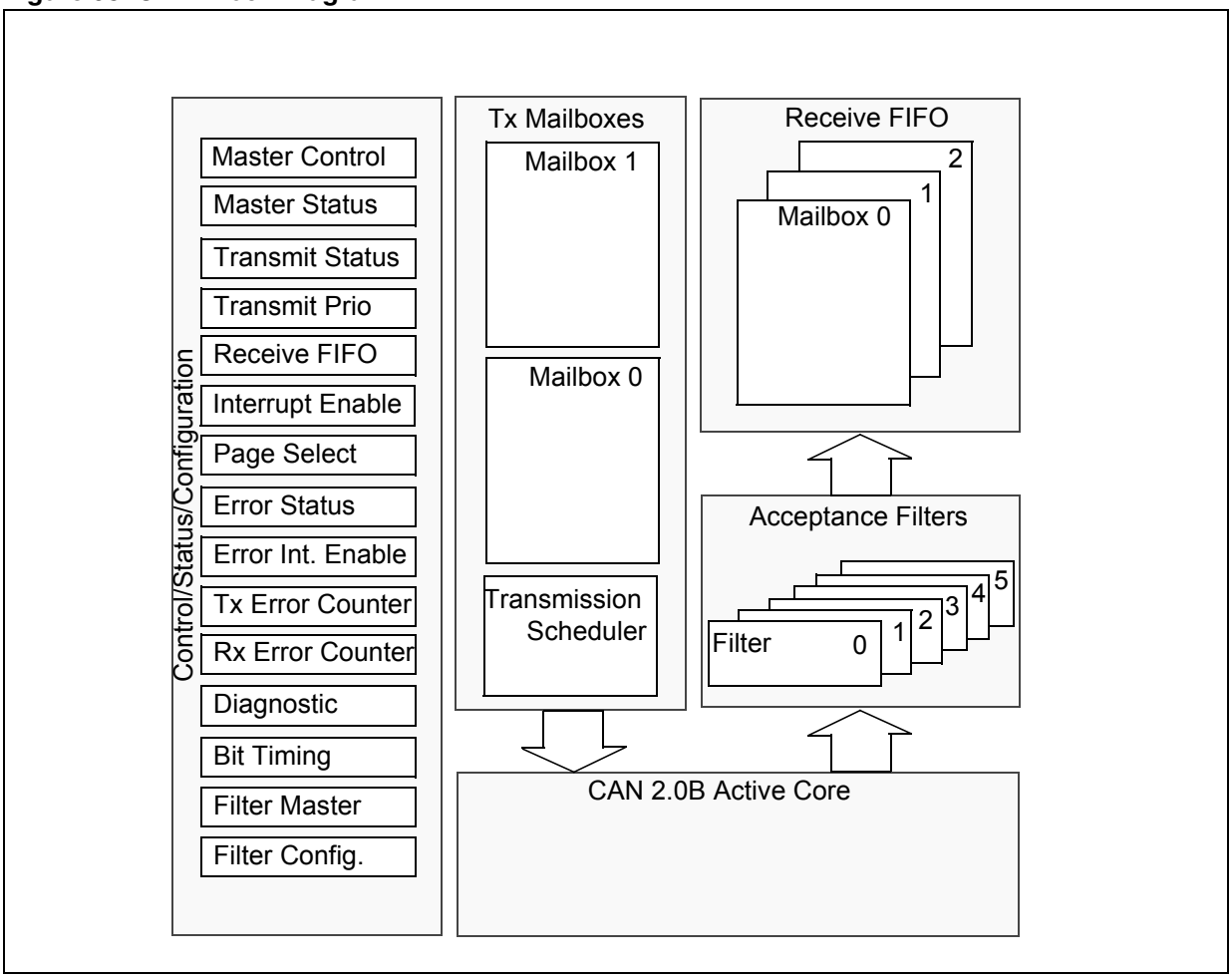
Acceptance Filters

The beCAN provides six scalable/configurable identifier filter banks for selecting the incoming messages the software needs and discarding the others.

Receive FIFO

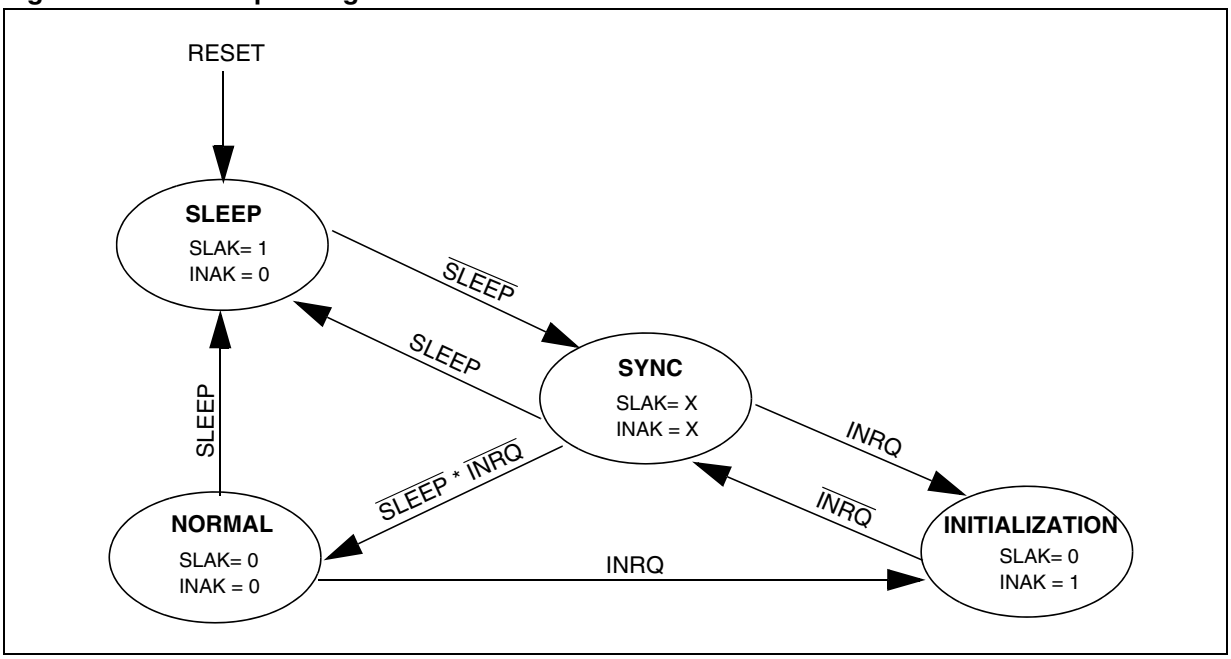
The receive FIFO is used by the CAN controller to store the incoming messages. Three complete messages can be stored in the FIFO. The software always accesses the next available message at the same address. The FIFO is managed completely by hardware.

Figure 95. CAN Block Diagram



beCAN CONTROLLER (Cont'd)

Figure 96. beCAN Operating Modes



10.9.3 Operating Modes

The beCAN has three main operating modes: Initialization, Normal and Sleep. After a hardware reset, beCAN is in Sleep mode to reduce power consumption. The software requests beCAN to enter Initialization or Sleep mode by setting the INRQ or SLEEP bits in the CMCR register. Once the mode has been entered, beCAN confirms it by setting the INAK or SLAK bits in the CMSR register. When neither INAK nor SLAK are set, beCAN is in Normal mode. Before entering Normal mode beCAN always has to **synchronize** on the CAN bus. To synchronize, beCAN waits until the CAN bus is idle, this means 11 consecutive recessive bits have been monitored on CANRX.

10.9.3.1 Initialization Mode

The software initialization can be done while the hardware is in Initialization mode. To enter this mode the software sets the INRQ bit in the CMCR register and waits until the hardware has confirmed the request by setting the INAK bit in the CMSR register.

To leave Initialization mode, the software clears the INQR bit. beCAN has left Initialization mode once the INAK bit has been cleared by hardware.

While in Initialization mode, all message transfers to and from the CAN bus are stopped and the status of the CAN bus output CANTX is recessive (high).

Entering Initialization Mode does not change any of the configuration registers.

To initialize the CAN Controller, software has to set up the Bit Timing registers and the filter banks. If a filter bank is not used, it is recommended to leave it non active (leave the corresponding FACT bit cleared).

10.9.3.2 Normal Mode

Once the initialization has been done, the software must request the hardware to enter Normal mode, to synchronize on the CAN bus and start reception and transmission. Entering Normal mode is done by clearing the INRQ bit in the CMCR register and waiting until the hardware has confirmed the request by clearing the INAK bit in the CMSR register. Afterwards, the beCAN synchronizes with the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (≡ Bus Idle) before it can take part in bus activities and start message transfer.

The initialization of the filter values is independent from Initialization mode but must be done while the filter bank is not active (corresponding FACTx bit cleared). The filter bank scale and mode configuration must be configured in initialization mode.

beCAN CONTROLLER (Cont'd)

10.9.3.3 Low Power Mode (Sleep)

To reduce power consumption, beCAN has a low power mode called Sleep mode. This mode is entered on software request by setting the SLEEP bit in the CMCR register. In this mode, the beCAN clock is stopped. Consequently, software can still access the beCAN registers and mailboxes but the beCAN will not update the status bits.

Example: If software requests entry to **initialization** mode by setting the INRQ bit while beCAN is in **sleep** mode, it will not be acknowledged by the hardware, INAK stays cleared.

beCAN can be woken up (exit Sleep mode) either by software clearing the SLEEP bit or on detection of CAN bus activity.

On CAN bus activity detection, hardware automatically performs the wake-up sequence by clearing the SLEEP bit if the AWUM bit in the CMCR register is set. If the AWUM bit is cleared, software has to clear the SLEEP bit when a wake-up interrupt occurs, in order to exit from sleep mode.

Note: If the wake-up interrupt is enabled (WKUIE bit set in CIER register) a wake-up interrupt will be generated on detection of CAN bus activity, even if the beCAN automatically performs the wake-up sequence.

After the SLEEP bit has been cleared, Sleep mode is exited once beCAN has synchronized with the CAN bus, refer to Figure 3. beCAN Operating Modes. The sleep mode is exited once the SLAK bit has been cleared by hardware.

10.9.3.4 Test Mode

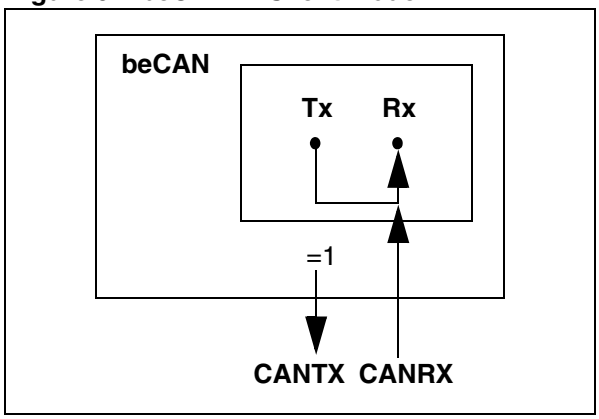
Test mode can be selected by the SILM and LBKM bits in the CDGR register. These bits must be configured while beCAN is in Initialization mode. Once test mode has been selected, beCAN is started in Normal mode.

10.9.3.5 Silent Mode

The beCAN can be put in Silent mode by setting the SILM bit in the CDGR register.

In Silent mode, the beCAN is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the beCAN has to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the CAN Core monitors this dominant bit, although the CAN bus may remain in recessive state. Silent mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames).

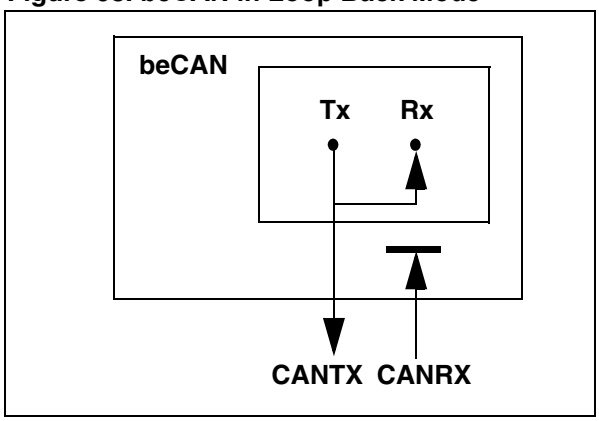
Figure 97. beCAN in Silent Mode



10.9.3.6 Loop Back Mode

The beCAN can be set in Loop Back Mode by setting the LBKM bit in the CDGR register. In Loop Back Mode, the beCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) in the FIFO.

Figure 98. beCAN in Loop Back Mode



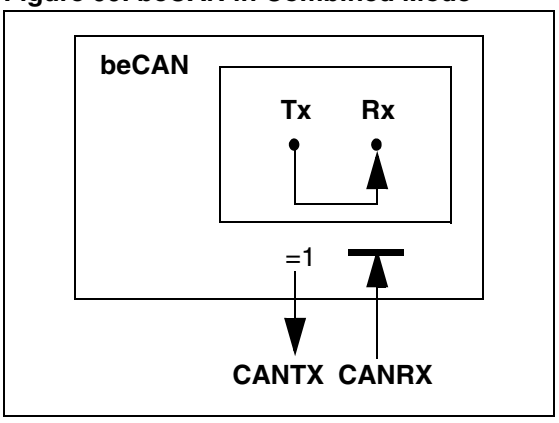
This mode is provided for self-test functions. To be independent of external events, the CAN Core ignores acknowledge errors (no dominant bit sampled in the acknowledge slot of a data / remote frame) in Loop Back Mode. In this mode, the beCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CANRX input pin is disregarded by the beCAN. The transmitted messages can be monitored on the CANTX pin.

beCAN CONTROLLER (Cont'd)

10.9.3.7 Loop Back combined with Silent Mode

It is also possible to combine Loop Back mode and Silent mode by setting the LBKM and SILM bits in the CDGR register. This mode can be used for a "Hot Selftest", meaning the beCAN can be tested like in Loop Back mode but without affecting a running CAN system connected to the CANTX and CANRX pins. In this mode, the CANRX pin is disconnected from the beCAN and the CANTX pin is held recessive.

Figure 99. beCAN in Combined Mode



10.9.4 Functional Description

10.9.4.1 Transmission Handling

In order to transmit a message, the application must select one **empty** transmit mailbox, set up the identifier, the data length code (DLC) and the data before requesting the transmission by setting the corresponding TXRQ bit in the MCSR register. Once the mailbox has left **empty** state, the software no longer has write access to the mailbox registers. Immediately after the TXRQ bit has been set, the mailbox enters **pending** state and waits to become the highest priority mailbox, see *Transmit Priority*. As soon as the mailbox has the highest priority it will be **scheduled** for transmission. The transmission of the message of the scheduled mailbox will start (enter **transmit** state) when the CAN bus becomes idle. Once the mailbox has been successfully transmitted, it will become **empty** again. The hardware indicates a successful transmission by setting the RQCP and TXOK bits in the MCSR and CTSR registers.

If the transmission fails, the cause is indicated by the ALST bit in the MCSR register in case of an Arbitration Lost, and/or the TERR bit, in case of transmission error detection.

Transmit Priority

By Identifier:

When more than one transmit mailbox is pending, the transmission order is given by the identifier of the message stored in the mailbox. The message with the lowest identifier value has the highest priority according to the arbitration of the CAN protocol. If the identifier values are equal, the lower mailbox number will be scheduled first.

By Transmit Request Order:

The transmit mailboxes can be configured as a transmit FIFO by setting the TXFP bit in the CMCR register. In this mode the priority order is given by the transmit request order.

This mode is very useful for segmented transmission.

Abort

A transmission request can be aborted by the user setting the ABRQ bit in the MCSR register. In **pending** or **scheduled** state, the mailbox is aborted immediately. An abort request while the mailbox is in **transmit** state can have two results. If the mailbox is transmitted successfully the mailbox becomes **empty** with the TXOK bit set in the MCSR and CTSR registers. If the transmission fails, the mailbox becomes **scheduled,** the transmission is aborted and becomes **empty** with TXOK cleared. In all cases the mailbox will become **empty** again at least at the end of the current transmission.

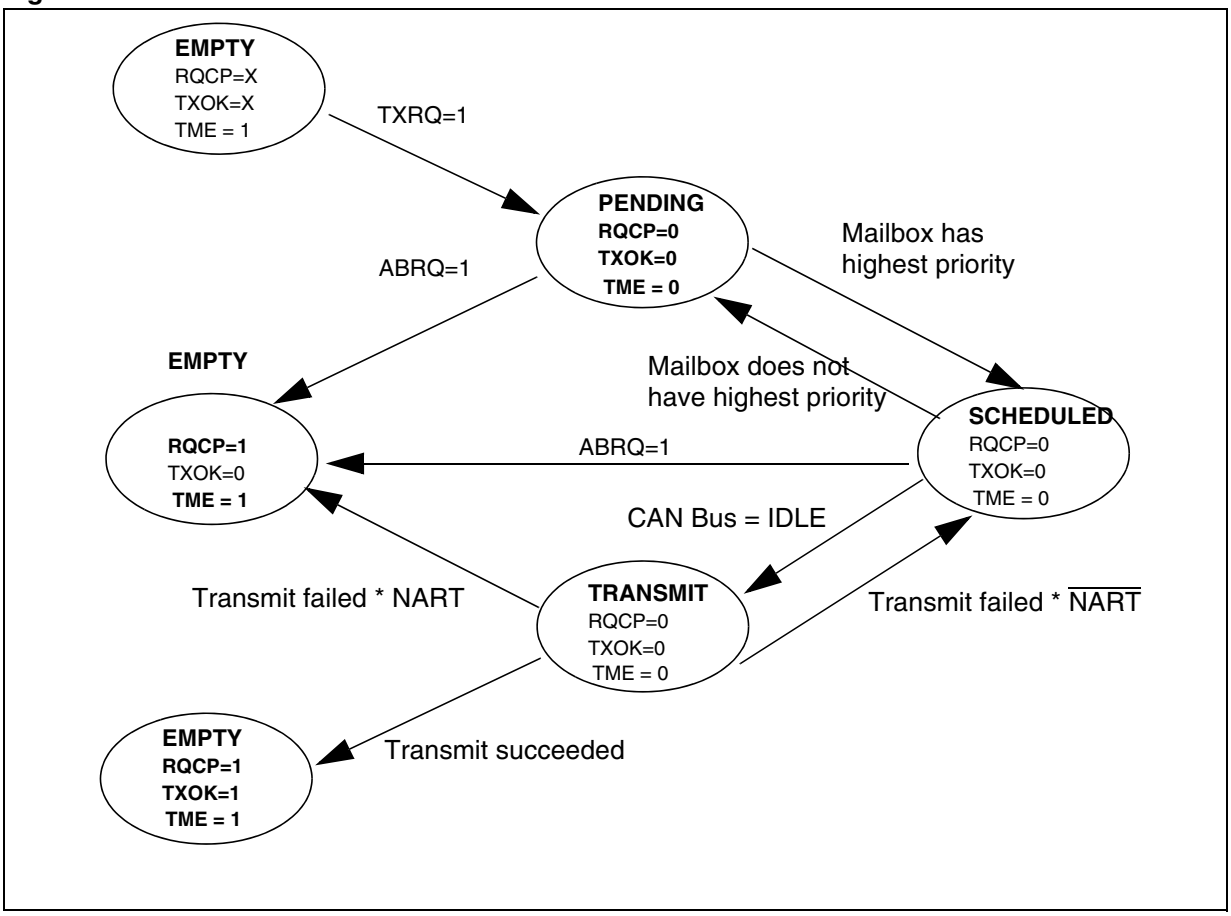
Non-Automatic Retransmission Mode

To configure the hardware in this mode the NART bit in the CMCR register must be set.

In this mode, each transmission is started only once. If the first attempt fails, due to an arbitration loss or an error, the hardware will not automatically restart the message transmission. At the end of the first transmission attempt, the hardware considers the request as completed and sets the RQCP bit in the MCSR register. The result of the transmission is indicated in the MCSR register by the TXOK, ALST and TERR bits.

beCAN CONTROLLER (Cont'd)

Figure 100. Transmit Mailbox States



beCAN CONTROLLER (Cont'd)

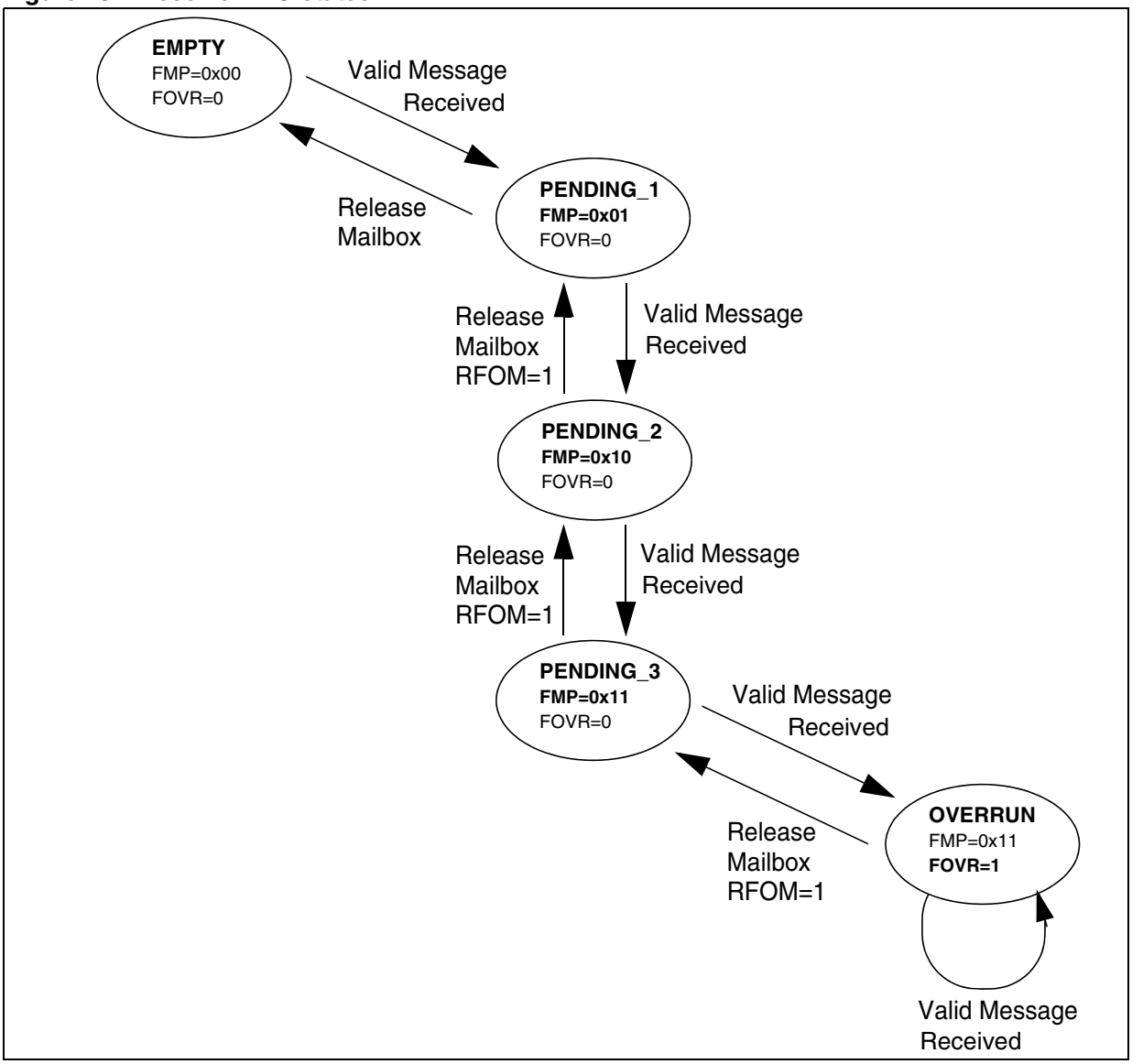
10.9.4.2 Reception Handling

For the reception of CAN messages, three mailboxes organized as a FIFO are provided. In order to save CPU load, simplify the software and guarantee data consistency, the FIFO is managed completely by hardware. The application accesses the messages stored in the FIFO through the FIFO output mailbox.

Valid Message

A received message is considered as valid **when** it has been received correctly according to the CAN protocol (no error until the last but one bit of the EOF field) **and** It passed through the identifier filtering successfully, see Section 0.1.4.3 Identifier Filtering.

Figure 101. Receive FIFO states



beCAN CONTROLLER (Cont'd)

FIFO Management

Starting from the **empty** state, the first valid message received is stored in the FIFO which becomes **pending_1**. The hardware signals the event setting the FMP[1:0] bits in the CRFR register to the value 01b. The message is available in the FIFO output mailbox. The software reads out the mailbox content and releases it by setting the RFOM bit in the CRFR register. The FIFO becomes **empty** again. If a new valid message has been received in the meantime, the FIFO stays in **pending_1** state and the new message is available in the output mailbox.

If the application does not release the mailbox, the next valid message will be stored in the FIFO which enters **pending_2** state (FMP[1:0] = 10b). The storage process is repeated for the next valid message putting the FIFO into **pending_3** state (FMP[1:0] = 11b). At this point, the software must release the output mailbox by setting the RFOM bit, so that a mailbox is free to store the next valid message. Otherwise the next valid message received will cause a loss of message.

Refer also to Section 0.1.4.4 Message Storage.

Overrun

Once the FIFO is in **pending_3** state (that is, the three mailboxes are full) the next valid message reception will lead to an **overrun** and a message will be lost. The hardware signals the overrun condition by setting the FOVR bit in the CRFR register. Which message is lost depends on the configuration of the FIFO:

– If the FIFO lock function is disabled (RFLM bit in the CMCR register cleared) the last message stored in the FIFO will be overwritten by the new incoming message. In this case the latest messages will be always available to the application.

– If the FIFO lock function is enabled (RFLM bit in the CMCR register set) the most recent message will be discarded and the software will have the three oldest messages in the FIFO available.

Reception Related Interrupts

On the storage of the first message in the FIFO - FMP[1:0] bits change from 00b to 01b - an interrupt is generated if the FMPIE bit in the CIER register is set.

When the FIFO becomes full (that is, a third message is stored) the FULL bit in the CRFR register is set and an interrupt is generated if the FFIE bit in the CIER register is set.

On overrun condition, the FOVR bit is set and an interrupt is generated if the FOVIE bit in the CIER register is set.

10.9.4.3 Identifier Filtering

In the CAN protocol the identifier of a message is not associated with the address of a node but related to the content of the message. Consequently a transmitter broadcasts its message to all receivers. On message reception a receiver node decides - depending on the identifier value - whether the software needs the message or not. If the message is needed, it is copied into the RAM. If not, the message must be discarded without intervention by the software.

To fulfil this requirement, the beCAN Controller provides six configurable and scalable filter banks (0-5) in order to receive only the messages the software needs. This hardware filtering saves CPU resources which would be otherwise needed to perform filtering by software. Each filter bank consists of eight 8-bit registers, CFxR[0:7].

Scalable Width

To optimize and adapt the filters to the application needs, each filter bank can be scaled independently. Depending on the filter scale a filter bank provides:

– One 32-bit filter for the STDID[10:0], IDE, EXTID[17:0] and RTR bits.

– Two 16-bit filters for the STDID[10:0], RTR and IDE bits.

– Four 8-bit filters for the STDID[10:3] bits. The other bits are considered as “don’t care”.

– One 16-bit filter and two 8-bit filters for filtering the same set of bits as the 16 and 8-bit filters described above.

Refer to Figure 9. Filter Bank Scale Configuration - Register Organisation.

Furthermore, the filters can be configured in mask mode or in identifier list mode.

Mask mode

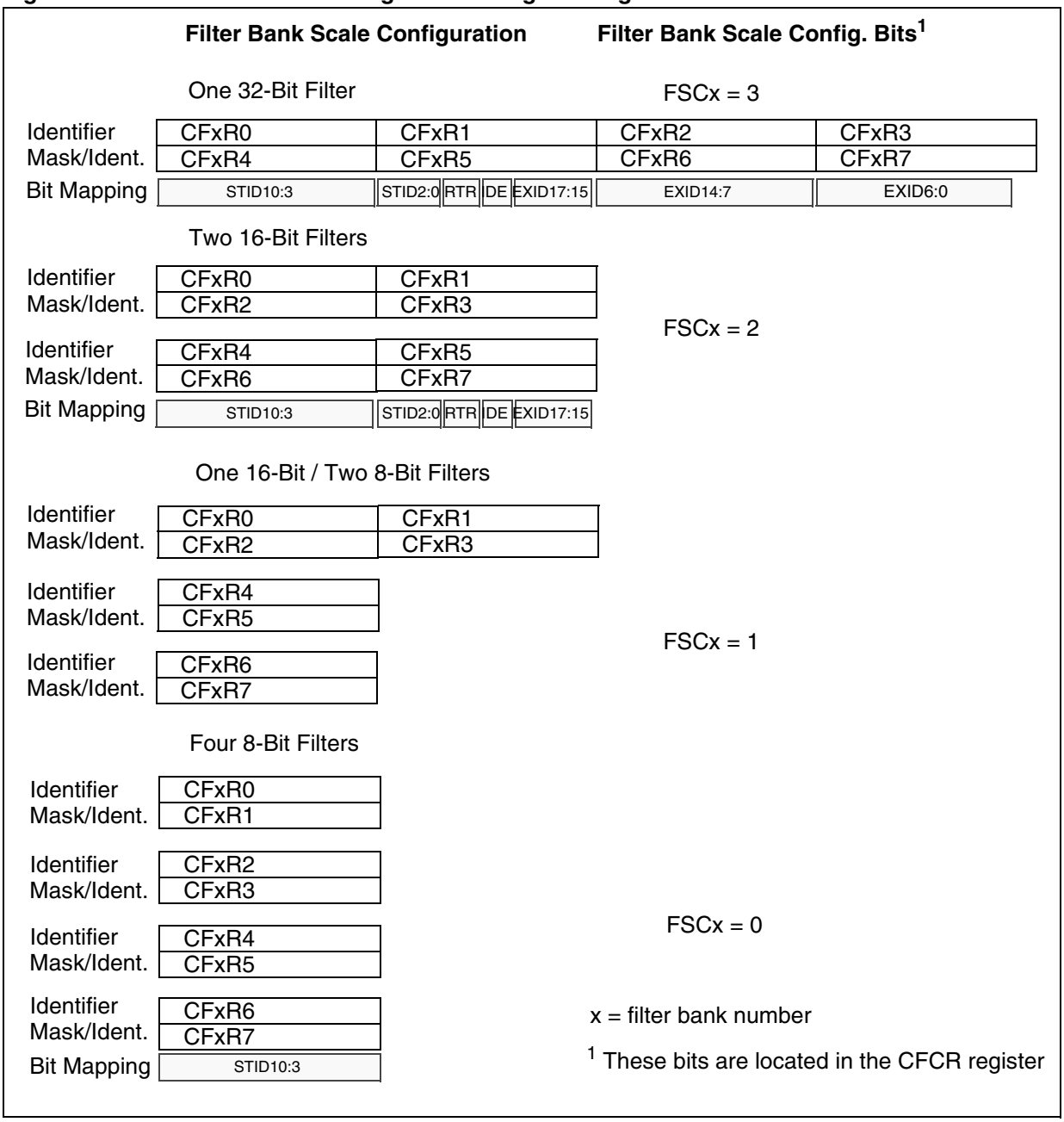
In **mask** mode the identifier registers are associated with mask registers specifying which bits of the identifier are handled as “must match” or as “don’t care”.

Identifier List mode

In **identifier list** mode, the mask registers are used as identifier registers. Thus instead of defining an identifier and a mask, two identifiers are specified, doubling the number of single identifiers. All bits of the incoming identifier must match the bits specified in the filter registers.

beCAN CONTROLLER (Cont'd)

Figure 102. Filter Bank Scale Configuration - Register Organisation



beCAN CONTROLLER (Cont'd)

Filter Bank Scale and Mode Configuration

The filter banks are configured by means of the corresponding CFCRx register. To configure a filter bank this must be deactivated by clearing the FACT bit in the CFCR register. The filter scale is configured by means of the FSC[1:0] bits in the corresponding CFCR register, refer to Figure 9. Filter Bank Scale Configuration - Register Organisation. The **identifier list** or **identifier mask** mode for the corresponding Mask/Identifier registers is configured by means of the FMCLx and FMCHx bits in the CFMR register. The FMCLx bit defines the mode for the two least significant bytes, and the FMCHx bit the mode for the two most significant bytes of filter bank x. **Examples**:

– If filter bank 1 is configured as two 16-bit filters, then the FMCL1 bit defines the mode of the CF1R2 and CF1R3 registers and the FMCH1 bit defines the mode of the CF1R6 and CF1R7 registers.

– If filter bank 2 is configured as four 8-bit filters, then the FMCL2 bit defines the mode of the CF2R1 and CF2R3 registers and the FMCH2 bit defines the mode of the CF2R5 and CF2R7 registers.

Note: In 32-bit configuration, the FMCLx and FMCHx bits must have the same value to ensure that the four Mask/Identifier registers are in the same mode.

To filter a group of identifiers, configure the Mask/Identifier registers in mask mode.

To select single identifiers, configure the Mask/Identifier registers in identifier list mode.

Filters not used by the application should be left deactivated.

Filter Match Index

Once a message has been received in the FIFO it is available to the application. Typically application data are copied into RAM locations. To copy the data to the right location the application has to identify the data by means of the identifier. To avoid this and to ease the access to the RAM locations, the CAN controller provides a Filter Match Index.

This index is stored in the mailbox together with the message according to the filter priority rules. Thus each received message has its associated Filter Match Index.

The Filter Match Index can be used in two ways:

– Compare the Filter Match Index with a list of expected values.

– Use the Filter Match Index as an index on an array to access the data destination location.

For non-masked filters, the software no longer has to compare the identifier.

If the filter is masked the software reduces the comparison to the masked bits only.

Filter Priority Rules

Depending on the filter combination it may occur that an identifier passes successfully through several filters. In this case the filter match value stored in the receive mailbox is chosen according to the following rules:

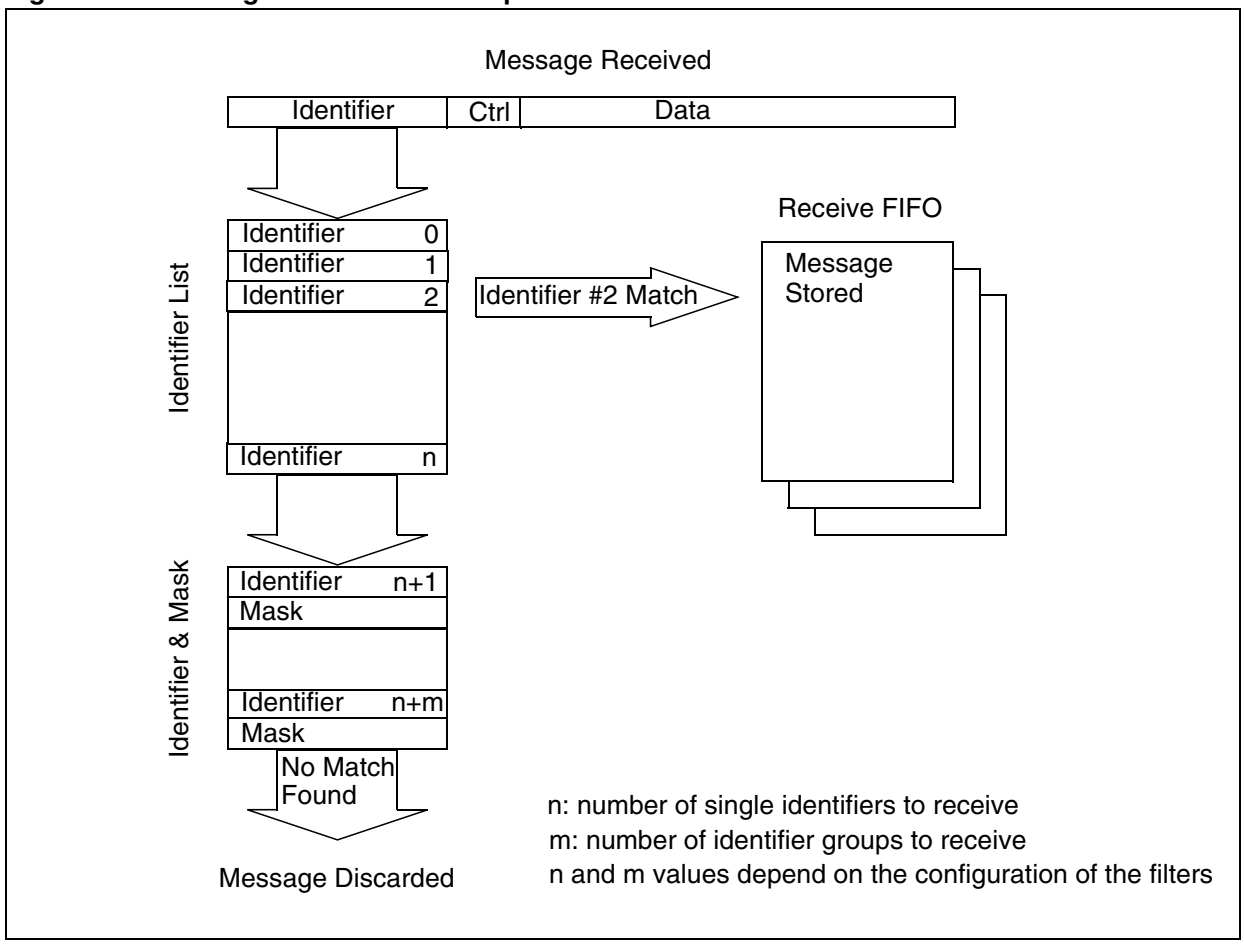
– A filter in identifier list mode prevails on an filter in mask mode.

– A filter with full identifier coverage prevails over filters covering part of the identifier, e.g. 16-bit filters prevail over 8-bit filters.

– Filters configured in the same mode and with identical coverage are prioritized by filter number and register number. The lower the number the higher the priority.

beCAN CONTROLLER (Cont'd)

Figure 103. Filtering Mechanism - example



The example above shows the filtering principle of the beCAN. On reception of a message, the identifier is compared first with the filters configured in identifier list mode. If there is a match, the message is stored in the FIFO and the index of the matching filter is stored in the Filter Match Index. As shown in the example, the identifier matches with Identifier #2 thus the message content and MFMI 2 is stored in the FIFO.

If there is no match, the incoming identifier is then compared with the filters configured in mask mode.

If the identifier does not match any of the identifiers configured in the filters, the message is discarded by hardware without software intervention.

beCAN CONTROLLER (Cont'd)

10.9.4.4 Message Storage

The interface between the software and the hardware for the CAN messages is implemented by means of mailboxes. A mailbox contains all information related to a message; identifier, data, control and status information.

Transmit Mailbox

The software sets up the message to be transmitted in an empty transmit mailbox. The status of the transmission is indicated by hardware in the MCSR register.

Transmit Mailbox Mapping

Offset to Transmit Mailbox base address (bytes)	Register Name
0	MCSR
1	MDLC
2	MIDR0
3	MIDR1
4	MIDR2
5	MIDR3
6	MDAR0
7	MDAR1
8	MDAR2
9	MDAR3
10	MDAR4
11	MDAR5
12	MDAR6
13	MDAR7
14	Reserved
15	Reserved

Receive Mailbox

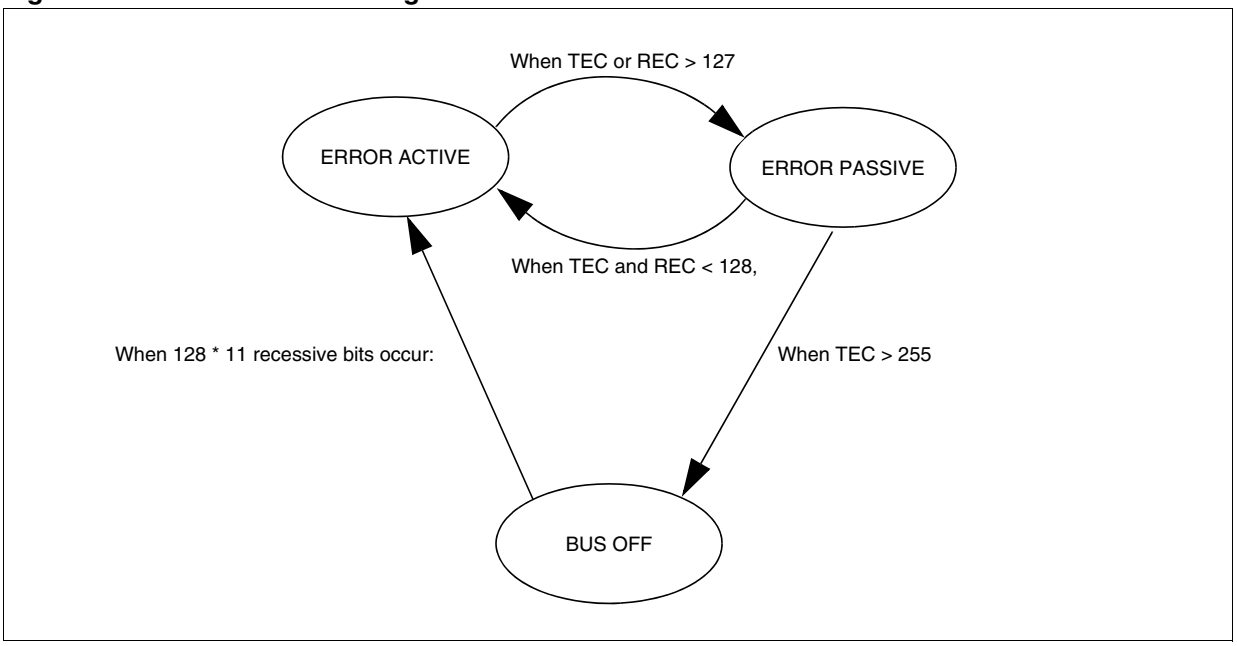
When a message has been received, it is available to the software in the FIFO output mailbox. Once the software has handled the message (e.g. read it) the software must release the FIFO output mailbox by means of the RFOM bit in the CRFR register to make the next incoming message available. The filter match index is stored in the MFMI register.

Receive Mailbox Mapping

Offset to Receive Mailbox base address (bytes)	Register Name
0	MFMI
1	MDLC
2	MIDR0
3	MIDR1
4	MIDR2
5	MIDR3
6	MDAR0
7	MDAR1
8	MDAR2
9	MDAR3
10	MDAR4
11	MDAR5
12	MDAR6
13	MDAR7
14	Reserved
15	Reserved

beCAN CONTROLLER (Cont'd)

Figure 104. CAN Error State Diagram



10.9.4.5 Error Management

The error management as described in the CAN protocol is handled entirely by hardware using a Transmit Error Counter (TECR register) and a Receive Error Counter (RECR register), which get incremented or decremented according to the error condition. For detailed information about TEC and REC management, please refer to the CAN standard.

Both of them may be read by software to determine the stability of the network. Furthermore, the CAN hardware provides detailed information on the current error status in CESR register. By means of CEIER register and ERRIE bit in CIER register, the software can configure the interrupt generation on error detection in a very flexible way.

Bus-Off Recovery

The Bus-Off state is reached when TECR is greater then 255, this state is indicated by BOFF bit in CESR register. In Bus-Off state, the beCAN acts as disconnected from the CAN bus, hence it is no longer able to transmit and receive messages.

Depending on the ABOM bit in the CMCR register beCAN will recover from Bus-Off (become error active again) either automatically or on software request. But in both cases the beCAN has to wait at least for the recovery sequence specified in the CAN standard (128 x 11 consecutive recessive bits monitored on CANRX).

If ABOM is set, the beCAN will start the recovering sequence automatically after it has entered Bus-Off state.

If ABOM is cleared, the software must initiate the recovering sequence by requesting beCAN to enter initialization mode. Then beCAN starts monitoring the recovery sequence when the beCAN is requested to leave the initialisation mode.

Note: In initialization mode, beCAN does not monitor the CANRX signal, therefore it cannot complete the recovery sequence. **To recover, beCAN must be in normal mode**.

beCAN CONTROLLER (Cont'd)

10.9.4.6 Bit Timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

Its operation may be explained simply by splitting nominal bit time into three segments as follows:

– **Synchronization segment (SYNC_SEG)**: a bit change is expected to occur within this time segment. It has a fixed length of one time quantum ($1 \times t_{CAN}$).

– **Bit segment 1 (BS1)**: defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable between 1 and 16 time quanta but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.

– **Bit segment 2 (BS2)**: defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard. Its duration is programmable between 1 and 8 time quanta but may also be automatically shortened to compensate for negative phase drifts.

– **Resynchronization Jump Width (RJW)**: defines an upper bound to the amount of lengthening or shortening of the bit segments. It is programmable between 1 and 4 time quanta.

To guarantee the correct behaviour of the CAN controller, SYNC_SEG + BS1 + BS2 must be greater than or equal to 5 time quanta.

For a detailed description of the CAN resynchronization mechanism and other bit timing configuration constraints, please refer to the Bosch CAN standard 2.0.

As a safeguard against programming errors, the configuration of the Bit Timing Registers CBTR1 and CBTR0 is only possible while the device is in Initialization mode.

Figure 105. Bit Timing

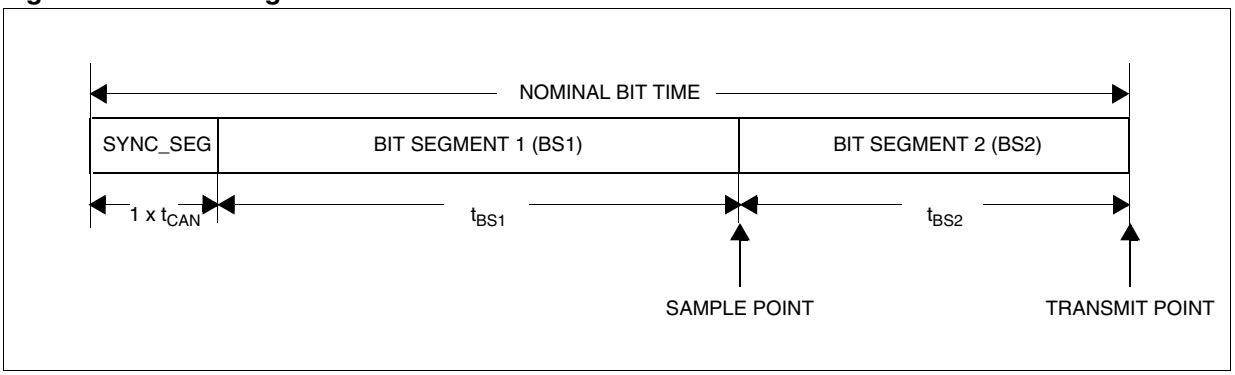
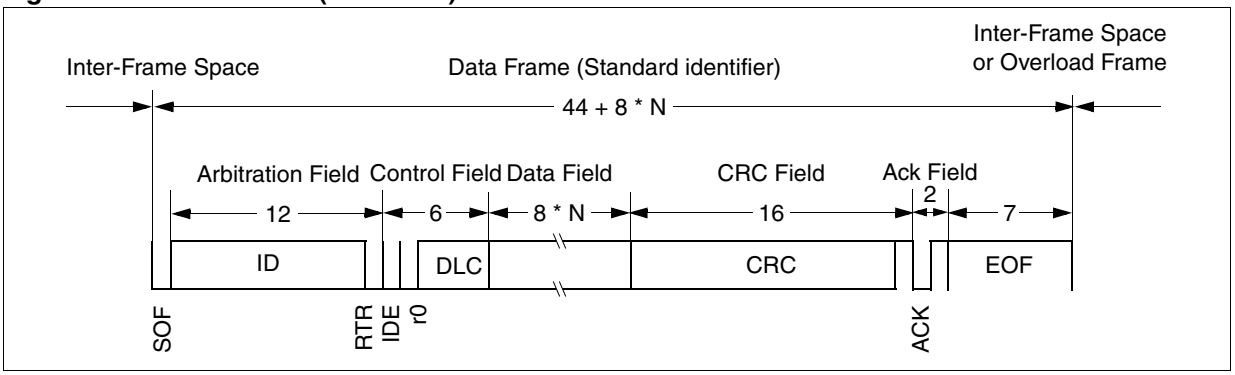
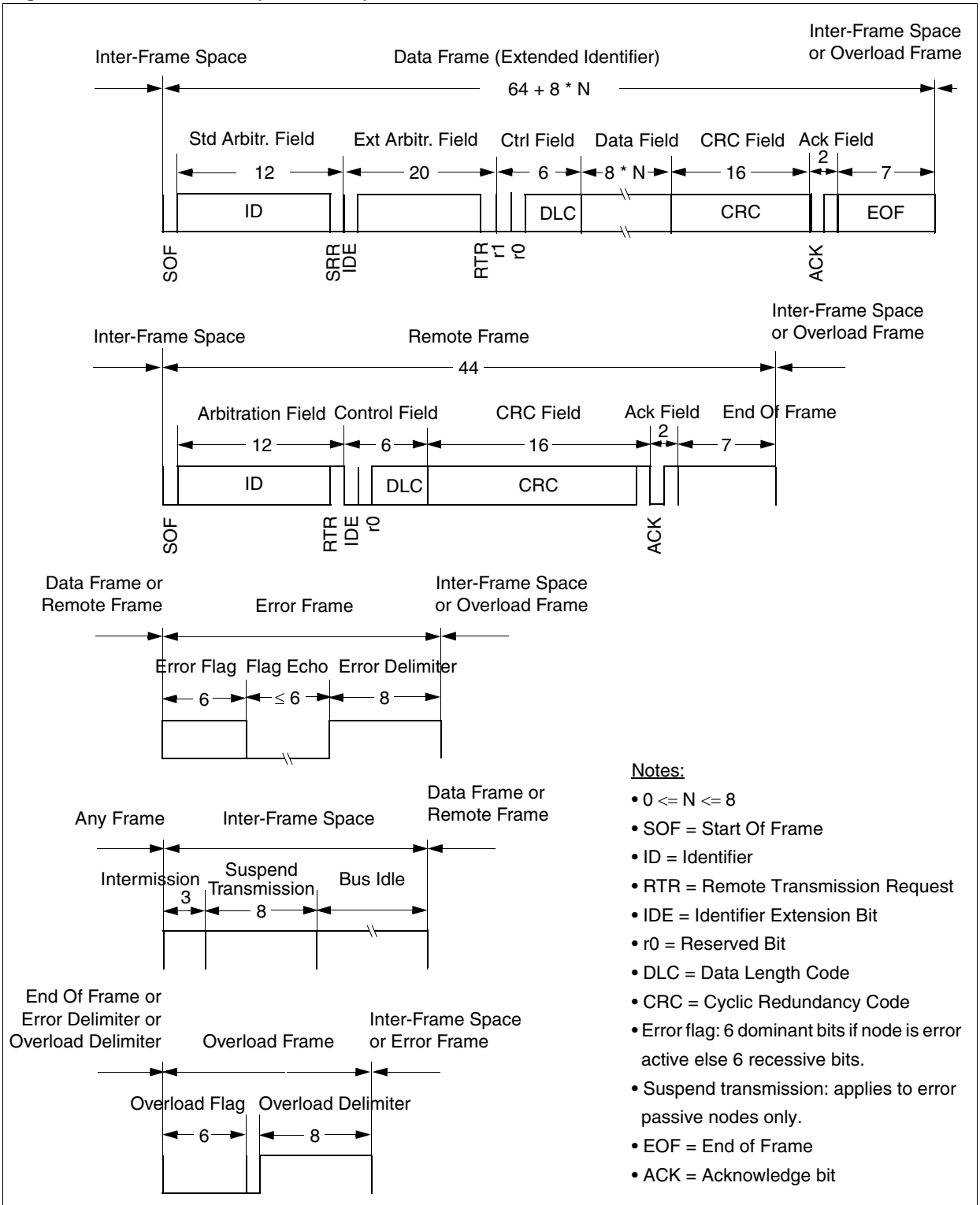


Figure 106. CAN Frames (Part 1of 2)



beCAN CONTROLLER (Cont'd)

Figure 107. CAN Frames (Part 2 of 2)



Notes:

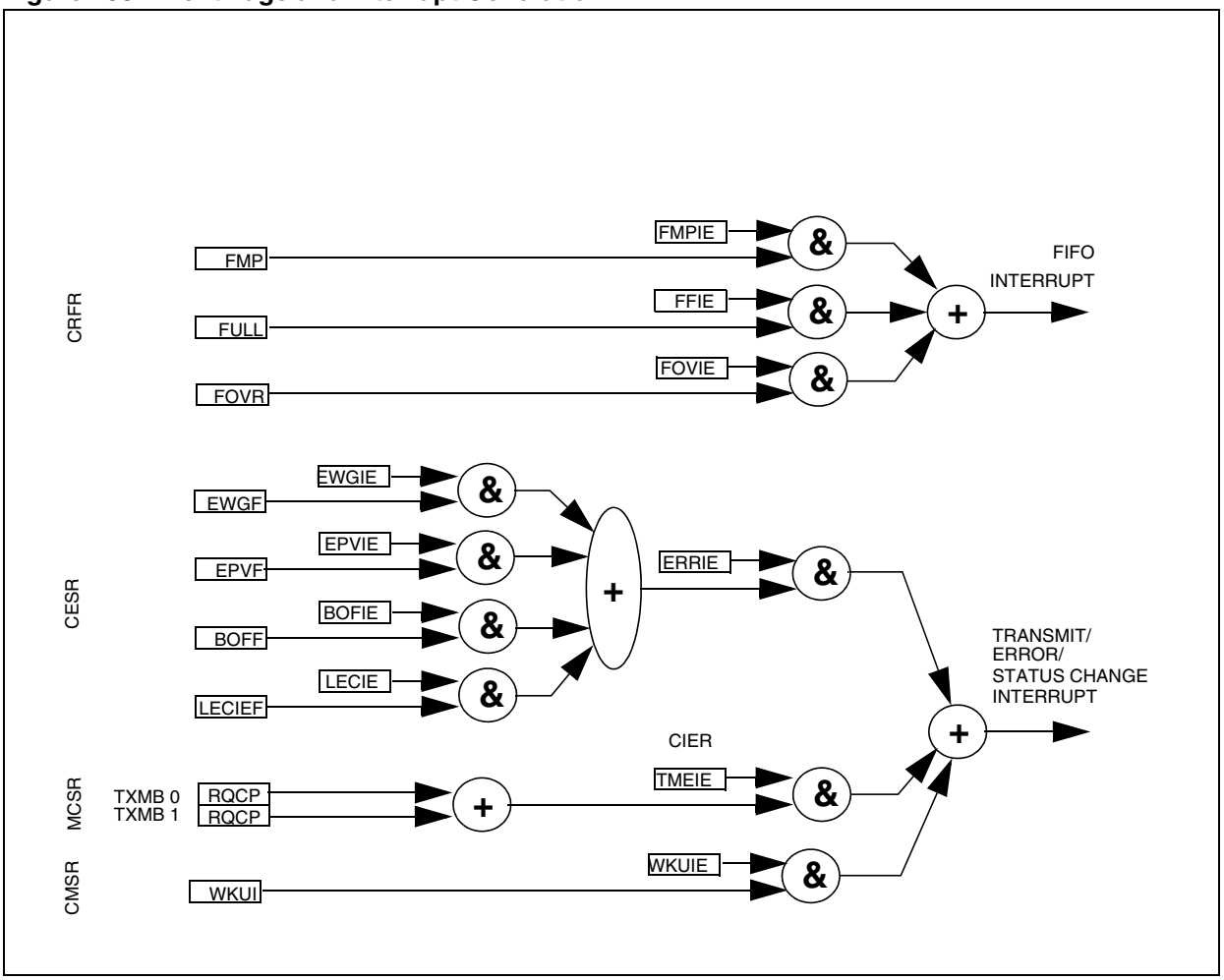
- 0 <= N <= 8
- SOF = Start Of Frame
- ID = Identifier
- RTR = Remote Transmission Request
- IDE = Identifier Extension Bit
- r0 = Reserved Bit
- DLC = Data Length Code
- CRC = Cyclic Redundancy Code
- Error flag: 6 dominant bits if node is error active else 6 recessive bits.
- Suspend transmission: applies to error passive nodes only.
- EOF = End of Frame
- ACK = Acknowledge bit

beCAN CONTROLLER (Cont'd)

10.9.5 Interrupts

Two interrupt vectors are dedicated to beCAN. Each interrupt source can be independently enabled or disabled by means of the CAN Interrupt Enable Register (CIER) and CAN Error Interrupt Enable register (CEIER).

Figure 108. Event flags and Interrupt Generation



beCAN CONTROLLER (Cont'd)

– The **FIFO interrupt** can be generated by the following events:
  – Reception of a new message, FMP bits in the CRFR0 register incremented.
  – FIFO0 full condition, FULL bit in the CRFR0 register set.
  – FIFO0 overrun condition, FOVR bit in the CRFR0 register set.
– The **transmit, error and status change interrupt** can be generated by the following events:
  – Transmit mailbox 0 becomes empty, RQCP0 bit in the CTSR register set.
  – Transmit mailbox 1 becomes empty, RQCP1 bit in the CTSR register set.
  – Error condition, for more details on error conditions please refer to the CAN Error Status register (CESR).
  – Wake-up condition, SOF monitored on the CAN Rx signal.

10.9.6 Register Access Protection

Erroneous access to certain configuration registers can cause the hardware to temporarily disturb the whole CAN network. Therefore the following registers can be modified by software only while the hardware is in initialization mode:

CBTR0, CBTR1, CFCR0, CFCR1, CFMR and CDGR registers.

Although the transmission of incorrect data will not cause problems at the CAN network level, it can severely disturb the application. A transmit mailbox can be only modified by software while it is in empty state (refer to Figure 7. Transmit Mailbox States).

The filters must be deactivated before their value can be modified by software. The modification of the filter configuration (scale or mode) can be done by software only in initialization mode.

beCAN CONTROLLER (Cont'd)

10.9.7 BeCAN Cell Limitations

10.9.7.1 FIFO Corruption

FIFO corruption occurs in the following case:

WHEN the beCAN RX FIFO already holds two messages (that is, FMP == 2)

AND the application releases the FIFO (with the instruction CRFR = B_RFOM;)

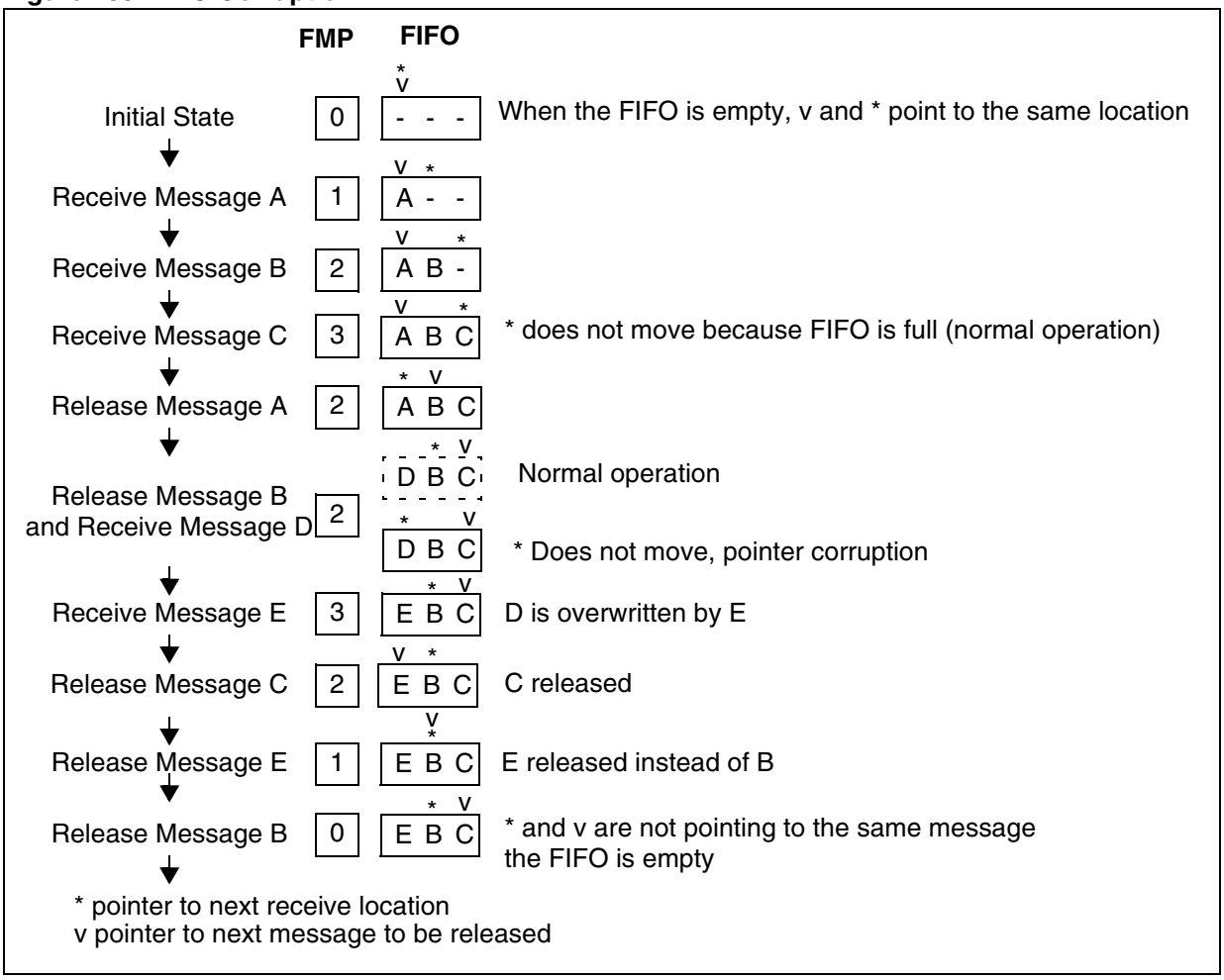
WHILE the beCAN requests the transfer of a new receive message into the FIFO (this lasts one CPU cycle)

THEN the internal FIFO pointer is not updated

BUT the FMP bits are updated correctly

As the FIFO pointer is not updated correctly, this causes the last message received to be overwritten by any incoming message. This means one message is lost as shown in the example in Figure 16. The beCAN will not recover normal operation until a device reset occurs.

Figure 109. FIFO Corruption



beCAN CONTROLLER (Cont'd)

Workaround

To implement the workaround, use the following sequence to release the CAN receive FIFO. This sequence replaces any occurrence of `CRFR |= B_RFOM;`.

Figure 110. Workaround 1

```
if ((CRFR & 0x03) == 0x02)
      while    (( CMSR & 0x20) && ( CDGR & 0x08) )   { };
CRFR |= B_RFOM;
```

Explanation of Workaround 1

First, we need to make sure no interrupt can occur between the test and the release of the FIFO to avoid any added delay.

The workaround checks if the first two FIFO levels are already full (FMP = 2) as the problem happens only in this case.

If FMP $\neq$ 2 we release the FIFO immediately, if FMP = 2, we monitor the reception status of the cell.

The reception status is available in the CMSR register bit 5 (REC bit). **Note:** The REC bit was called RX in older versions of the datasheet.

– If the cell is not receiving, then REC bit in CMSR is at 0, the software can release the FIFO immediately: there is no risk.

– If the cell is receiving, it is important to make sure the release of the mailbox will not happen at the time when the received message is loaded into the FIFO.

We could simply wait for the end of the reception, but this could take a long time (200µs for a 100-bit frame at 500 kHz), so we also monitor the Rx pin of the microcontroller to minimize the time the application may wait in the while loop.

We know the critical window is located at the end of the frame, 6+ CAN bit times after the acknowledge bit (exactly six full bit times plus the time from the beginning of the bit to the sample point). Those bits represent the acknowledge delimiter + the end of frame slot.

We know also that those 6+ bits are in recessive state on the bus, therefore if the CAN Rx pin of the device is at '0', (reflecting a CAN dominant state on the bus), this is early enough to be sure we can release the FIFO before the critical time slot.

Therefore, if the device hardware pin Rx is at 0 and there is a reception on going, its message will be transferred to the FIFO only 6+ CAN bit times later at the earliest (if the dominant bit is the acknowledge) or later if the dominant bit is part of the message.

Compiled with Cosmic C compiler, the workaround generates the following assembly lines:

```
                                        Cycles
if ((CRFR & 0x03) == 0x02)
                    ld      a, CRFR          3
                    and     a,#3             2
                    cp      a,#2             2
                    jrne    _RELEASE         3   test: 10 cycles

while    (( CMSR & 0x20) && ( CDGR & 0x08) )  { };
  _WHILELOOP:
                    btjf    CMSR,#5,_RELEASE     5
                    btjt    CDGR,#3,_WHILELOOP   5   loop: 10 cycles

CRFR |= B_RFOM;
  _RELEASE:
                    bset    CRFR,#5          5   release: 5 cycles
```

beCAN CONTROLLER (Cont'd)

In the worst case configuration, if the CAN cell speed is set to the maximum baud rate, one bit time is 8 CPU cycle. In this case the minimum time between the end of the acknowledge and the critical period is 52 CPU cycles (48 for the 6 bit times + 4 for the (PROP SEG + $T_{Seg\ 1}$). According to the previous code timing, we need less than 15 cycles from the time we see the dominant state to the time we perform the FIFO release (one full loop + the actual release) therefore the application will never release the FIFO at the critical time when this workaround is implemented.

Timing analysis

- Time spent in the workaround

Inside a CAN frame, the longest period that the Rx pin stays in recessive state is 5 bits. At the end of the frame, the time between the acknowledge dominant bit and the end of reception (signaled by REC bit status) is $8t_{CANbit}$, therefore the maximum time spent in the workaround is: $8t_{CANbit}+t_{loop}+t_{test}+t_{release}$ in this case or $8t_{CANbit}+25t_{CPU}$.

At low speed, this time could represent a long delay for the application, therefore it makes sense to evaluate how frequently this delay occurs.

In order to reach the critical FMP = 2, the CAN node needs to receive two messages without servicing them. Then in order to reach the critical window, the cell has to receive a third one and the application has to release the mailbox at the same time, at the end of the reception.

In the application, messages are not processed only if either the interrupt are disabled or higher level interrupts are being serviced.

Therefore if:

$t_{IT\ higher\ level} + t_{IT\ disable} + t_{IT\ CAN} < 2 \times t_{CAN\ frame}$

the application will never wait in the workaround

$t_{IT\ higher\ level}$: This the sum of the duration of all the interrupts with a level strictly higher than the CAN interrupt level

$t_{IT\ disable}$: This is the longest time the application disables the CAN interrupt (or all interrupts)

$t_{IT\ CAN}$: This is the maximum duration between the beginning of the CAN interrupt and the actual location of the workaround

$t_{CAN\ frame}$: This is minimum CAN frame duration

Figure 111. Critical Window Timing Diagram

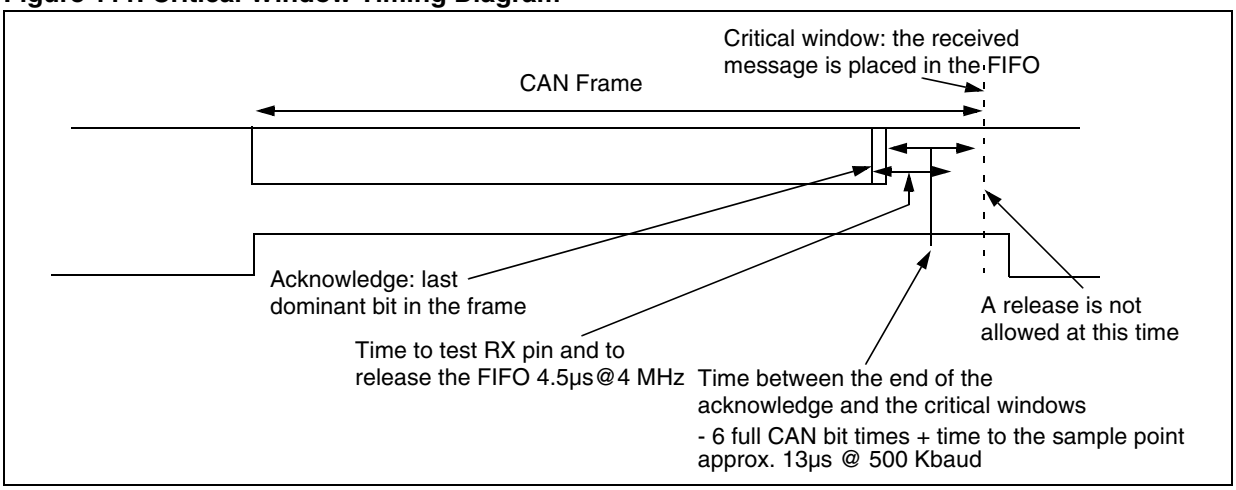
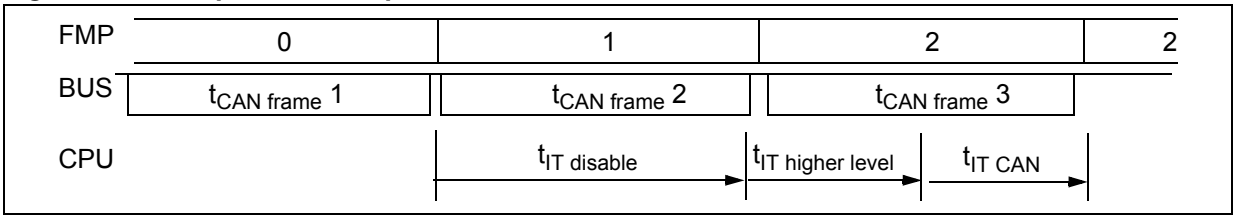


Figure 112. Reception of a Sequence of Frames



beCAN CONTROLLER (Cont'd)

Side-effect of Workround 1

Because the while loop lasts 10 CPU cycles, at high baud rate, it is possible to miss a dominant state on the bus if it lasts just one CAN bit time and the bus speed is high enough (see Table 1).

Table 29. While Loop Timing

f_{CPU}	Software timing: While loop	Minimum baud rate for possible missed dominant bit
8 MHz	1.25 µs	800 Kbaud
4 MHz	2.5 µs	400 Kbaud
f_{CPU}	$10/f_{CPU}$	$f_{CPU}/10$

If this happens, we will continue waiting in the while loop instead of releasing the FIFO immediately. The workaround is still valid because we will not release the FIFO during the critical period. But the application may lose additional time waiting in the while loop as we are no longer able to guarantee a maximum of 6 CAN bit times spent in the workaround.

In this particular case the time the application can spend in the workaround may increase up to a full CAN frame, depending of the frame contents. This case is very rare but happens when a specific sequence is present on in the CAN frame.

The example in Figure 20 shows reception at maximum CAN baud rate: In this case t_{CAN} is $8/f_{CPU}$ and the sampling time is $10/f_{CPU}$.

If the application is using the maximum baud rate and the possible delay caused by the workaround is not acceptable, there is another workaround which reduces the Rx pin sampling time.

Workaround 2 (see Figure 21) first tests that FMP = 2 and the CAN cell is receiving, if not the FIFO can be released immediately. If yes, the program goes through a sequence of test instructions on the RX pin that last longer than the time between the acknowledge dominant bit and the critical time slot. If the Rx pin is in recessive state for more than 8 CAN bit times, it means we are now after the acknowledge and the critical slot. If a dominant bit is read on the bus, we can release the FIFO immediately. This workaround has to be written in assembly language to avoid the compiler optimizing the test sequence.

The implementation shown here is for the CAN bus maximum speed (1 Mbaud @ 8 MHz CPU clock).

Figure 113. Reception at Maximum CAN Baud Rate

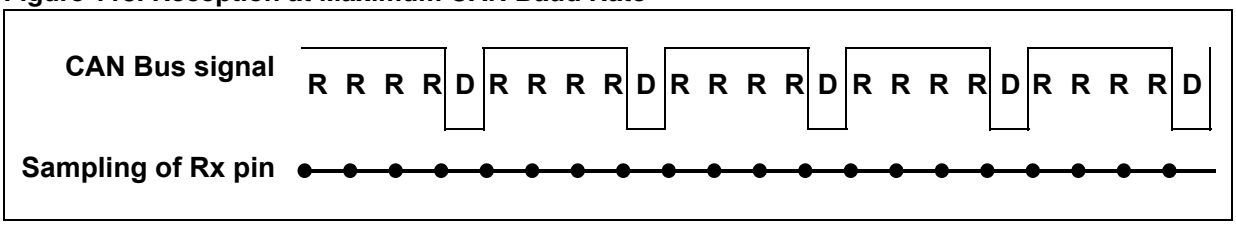


Figure 114. Workaround 2

```
Ld          a, CRFR
And         a,#3
Cp          a,#2            ; test FMP=2 ?
Jrne        _RELEASE        ; if not release


Btjf        CMSR,#5,_RELEASE ; test if reception on going.
                             ; if not release

Btjf        CDGR,#3,_RELEASE ; sample RX pin for 8 CAN bit time
Btjf        CDGR,#3,_RELEASE
Btjf        CDGR,#3,_RELEASE
btjf        CDGR,#3,_RELEASE
btjf        CDGR,#3,_RELEASE
btjf        CDGR,#3,_RELEASE
btjf        CDGR,#3,_RELEASE
btjf        CDGR,#3,_RELEASE
btjf        CDGR,#3,_RELEASE
btjf        CDGR,#3,_RELEASE
btjf        CDGR,#3,_RELEASE
btjf        CDGR,#3,_RELEASE
btjf        CDGR,#3,_RELEASE
btjf        CDGR,#3,_RELEASE

_RELEASE:
bset        CRFR,#5
```

beCAN CONTROLLER (Cont'd)

10.9.8 Register Description

10.9.8.1 Control and Status Registers

CAN MASTER CONTROL REGISTER (CMCR)
Reset Value: 0000 0010 (02h)

7							0
0	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ

Bit 7 = Reserved, must be kept cleared.

Bit 6 = **ABOM** *Automatic Bus-Off Management*
- Read/Set/Clear
This bit controls the behaviour of the CAN hardware on leaving the Bus-Off state.
0: The Bus-Off state is left on software request. Refer to Section 0.1.4.5 Error Management, Bus-Off recovery.
1: The Bus-Off state is left automatically by hardware once 128 x 11 recessive bits have been monitored.

For detailed information on the Bus-Off state please refer to Section 0.1.4.5 Error Management.

Bit 5 = **AWUM** *Automatic Wake-Up Mode*
- Read/Set/Clear

This bit controls the behaviour of the CAN hardware on message reception during sleep mode.
0: The sleep mode is left on software request by clearing the SLEEP bit of the CMCR register.
1: The sleep mode is left automatically by hardware on CAN message detection. The SLEEP bit of the CMCR register and the SLAK bit of the CMSR register are cleared by hardware.

Bit 4 = **NART** *No Automatic Retransmission*
- Read/Set/Clear
0: The CAN hardware will automatically retransmit the message until it has been successfully transmitted according to the CAN standard.
1: A message will be transmitted only once, independently of the transmission result (successful, error or arbitration lost).

Bit 3 = **RFLM** *Receive FIFO Locked Mode*
- Read/Set/Clear
0: Receive FIFO not locked on overrun. Once a receive FIFO is full the next incoming message will overwrite the previous one.
1: Receive FIFO locked against overrun. Once a receive FIFO is full the next incoming message will be discarded.

Bit 2 = **TXFP** Transmit FIFO Priority
- Read/Set/Clear
This bit controls the transmission order when several mailboxes are pending at the same time.
0: Priority driven by the identifier of the message
1: Priority driven by the request order (chronologically)

Bit 1 = **SLEEP** *Sleep Mode Request*
- Read/Set/Clear
This bit is set by software to request the CAN hardware to enter the sleep mode. Sleep mode will be entered as soon as the current CAN activity (transmission or reception of a CAN frame) has been completed.

This bit is cleared by software to exit sleep mode.

This bit is cleared by hardware when the AWUM bit is set and a SOF bit is detected on the CAN Rx signal.

Bit 0 = **INRQ** *Initialization Request*
- Read/Set/Clear
The software clears this bit to switch the hardware into normal mode. Once 11 consecutive recessive bits have been monitored on the Rx signal the CAN hardware is synchronized and ready for transmission and reception. Hardware signals this event by clearing the INAK bit if the CMSR register.

Software sets this bit to request the CAN hardware to enter initialization mode. Once software has set the INRQ bit, the CAN hardware waits until the current CAN activity (transmission or reception) is completed before entering the initialization mode. Hardware signals this event by setting the INAK bit in the CMSR register.

beCAN CONTROLLER (Cont'd)

CAN MASTER STATUS REGISTER (CMSR)

Reset Value: 0000 0010 (02h)

7							0
0	0	REC	TRAN	WKUI	ERRI	SLAK	INAK

Note: To clear a bit of this register the software must write this bit with a one.

Bits 7:4 = Reserved. Forced to 0 by hardware.

Bit 5 = **REC** *Receive*
- Read
The CAN hardware is currently receiver.

Bit 4 = **TRAN** *Transmit*
- Read
The CAN hardware is currently transmitter.

Bit 3 = **WKUI** *Wake-Up Interrupt*
- Read/Clear
This bit is set by hardware to signal that a SOF bit has been detected while the CAN hardware was in sleep mode. Setting this bit generates a status change interrupt if the WKUIE bit in the CIER register is set.

This bit is cleared by software.

Bit 2 = **ERRI** *Error Interrupt*
- Read/Clear
This bit is set by hardware when a bit of the CESR has been set on error detection and the corresponding interrupt in the CEIER is enabled. Setting this bit generates a status change interrupt if the ERRIE bit in the CIER register is set.

This bit is cleared by software.

Bit 1 = **SLAK** *Sleep Acknowledge*
- Read
This bit is set by hardware and indicates to the software that the CAN hardware is now in sleep mode. This bit acknowledges the sleep mode request from the software (set SLEEP bit in CMCR register).

This bit is cleared by hardware when the CAN hardware has left sleep mode. Sleep mode is left when the SLEEP bit in the CMCR register is cleared. Please refer to the AWUM bit of the CMCR register description for detailed information for clearing SLEEP bit.

Bit 0 = **INAK** *Initialization Acknowledge*
- Read
This bit is set by hardware and indicates to the software that the CAN hardware is now in initialization mode. This bit acknowledges the initialization request from the software (set INRQ bit in CMCR register).

This bit is cleared by hardware when the CAN hardware has left the initialization mode and is now synchronized on the CAN bus. To be synchronized the hardware has to monitor a sequence of 11 consecutive recessive bits on the CAN RX signal.

CAN TRANSMIT STATUS REGISTER (CTSR)

Read / Write (

Reset Value: 0000 0000 (00h)

7							0
0	0	TXOK1	TXOK0	0	0	RQCP1	RQCP0

Note: To clear a bit of this register the software must write this bit with a one.

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

Bit 5 = **TXOK1** *Transmission OK for mailbox 1*
- Read
This bit is set by hardware when the transmission request on mailbox 1 has been completed successfully. Please refer to Figure 7.

This bit is cleared by hardware when mailbox 1 is requested for transmission or when the software clears the RQCP1 bit.

beCAN CONTROLLER (Cont'd)

Bit 4 = **TXOK0** *Transmission OK for mailbox 0*
- Read
This bit is set by hardware when the transmission request on mailbox 0 has been completed successfully. Please refer to Figure 7.

This bit is cleared by hardware when mailbox 0 is requested for transmission or when the software clears the RQCP0 bit.

Bits 3:2 = Reserved. Forced to 0 by hardware.

Bit 1 = **RQCP1** *Request Completed for Mailbox 1*
- Read/Clear
This bit is set by hardware to signal that the last request for mailbox 1 has been completed. The request could be a transmit or an abort request.

This bit is cleared by software.

Bit 0 = **RQCP0** *Request Completed for Mailbox 0*
- Read/Clear
This bit is set by hardware to signal that the last request for mailbox 0 has been completed. The request could be a transmit or an abort request.

This bit is cleared by software.

CAN TRANSMIT PRIORITY REGISTER (CTPR)
All bits of this register are read only.

Reset Value: 0000 1100 (0Ch)

7							0
0	LOW1	LOW0	0	TME1	TME0	0	CODE

Bit 7 = Reserved. Forced to 0 by hardware.

Bit 6 = **LOW1** *Lowest Priority Flag for Mailbox 1*
- Read
This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 1 has the lowest priority.

Bit 5 = **LOW0** *Lowest Priority Flag for Mailbox 0*
- Read
This bit is set by hardware when more than one mailbox are pending for transmission and mailbox 0 has the lowest priority.

Note: These bits are set to zero when only one mailbox is pending.

Bit 4 = Reserved. Forced to 0 by hardware.

Bit 3 = **TME1** *Transmit Mailbox 1 Empty*
- Read
This bit is set by hardware when no transmit request is pending for mailbox 1.

Bit 2 = **TME0** *Transmit Mailbox 0 Empty*
- Read
This bit is set by hardware when no transmit request is pending for mailbox 0.

Bit 1:0 = **CODE** *Mailbox Code*
- Read
In case at least one transmit mailbox is free, the code value is equal to the number of the next transmit mailbox free.

In case all transmit mailboxes are pending, the code value is equal to the number of the transmit mailbox with the lowest priority.

beCAN CONTROLLER (Cont'd)

CAN RECEIVE FIFO REGISTERS (CRFR)
Read / Write
Reset Value: 0000 0000 (00h)

7							0
0	0	RFOM	FOVR	FULL	0	FMP1	FMP0

Note: To clear a bit in this register, software must write a "1" to the bit.

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

Bit 5 = **RFOM** *Release FIFO Output Mailbox*
- Read/Set
Set by software to release the output mailbox of the FIFO. The output mailbox can only be released when at least one message is pending in the FIFO. Setting this bit when the FIFO is empty has no effect. If more than one message are pending in the FIFO, the software has to release the output mailbox to access the next message.

Cleared by hardware when the output mailbox has been released.

Bit 4 = **FOVR** *FIFO Overrun*
- Read/Clear
This bit is set by hardware when a new message has been received and passed the filter while the FIFO was full.
This bit is cleared by software.

Bit 3 = **FULL** *FIFO Full*
- Read/Clear
Set by hardware when three messages are stored in the FIFO.
This bit can be cleared by software writing a one to this bit or releasing the FIFO by means of RFOM.

Bit 2 = Reserved. Forced to 0 by hardware.

Bits 1:0 = **FMP[1:0]** *FIFO Message Pending*
- Read

These bits indicate how many messages are pending in the receive FIFO.

FMP is increased each time the hardware stores a new message in to the FIFO. FMP is decreased each time the software releases the output mailbox by setting the RFOM bit.

CAN INTERRUPT ENABLE REGISTER (CIER)

All bits of this register are set and cleared by software.

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

7							0
WKUIE	0	0	0	FOVIE0	FFIE0	FMPIE0	TMEIE

Bit 7 = **WKUIE** *Wake-Up Interrupt Enable*
0: No interrupt when WKUI is set.
1: Interrupt generated when WKUI bit is set.

Bits 6:4 = Reserved. Forced to 0 by hardware.

Bit 3 = **FOVIE** *FIFO Overrun Interrupt Enable*
0: No interrupt when FOVR bit is set.
1: Interrupt generated when FOVR bit is set.

Bit 2 = **FFIE** *FIFO Full Interrupt Enable*
0: No interrupt when FULL bit is set.
1: Interrupt generated when FULL bit is set.

Bit 1 = **FMPIE** *FIFO Message Pending Interrupt Enable*
0: No interrupt on FMP[1:0] bits transition from 00b to 01b.
1: Interrupt generated on FMP[1:0] bits transition from 00b to 01b.

Bit 0 = **TMEIE** *Transmit Mailbox Empty Interrupt Enable*
0: No interrupt when RQCPx bit is set.
1: Interrupt generated when RQCPx bit is set.

beCAN CONTROLLER (Cont'd)

CAN ERROR STATUS REGISTER (CESR)

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

7							0
0	LEC2	LEC1	LEC0	0	BOFF	EPVF	EWGF

Bit 7 = Reserved. Forced to 0 by hardware.

Bits 6:4 = **LEC[2:0]** *Last Error Code*
- Read/Set/Clear
This field holds a code which indicates the type of the last error detected on the CAN bus. If a message has been transferred (reception or transmission) without error, this field will be cleared to '0'. The code 7 is unused and may be written by the CPU to check for update

Table 30. LEC Error Types

Code	Error Type
0	No Error
1	Stuff Error
2	Form Error
3	Acknowledgment Error
4	Bit recessive Error
5	Bit dominant Error
6	CRC Error
7	Set by software

Bit 3 = Reserved. Forced to 0 by hardware.

Bit 2 = **BOFF** *Bus-Off Flag*
- Read
This bit is set by hardware when it enters the bus-off state. The bus-off state is entered on TECR overrun, TEC greater than 255, refer to section 0.1.4.5 on page 14.

Bit 1 = **EPVF** *Error Passive Flag*
- Read
This bit is set by hardware when the Error Passive limit has been reached (Receive Error Counter or Transmit Error Counter greater than 127).

Bit 1 = **EWGF** *Error Warning Flag*
- Read
This bit is set by hardware when the warning limit has been reached. Receive Error Counter or Transmit Error Counter greater than 96.

CAN ERROR INTERRUPT ENABLE REGISTER (CEIER)

All bits of this register are set and clear by software.
Read/Write
Reset Value: 0000 0000 (00h)

7							0
ERRIE	0	0	LECIE	0	BOFIE	EPVIE	EWGIE

Bit 7 = **ERRIE** *Error Interrupt Enable*
0: No interrupt will be generated when an error condition is pending in the CESR.
1: An interrupt will be generation when an error condition is pending in the CESR.

Bits 6:5 = Reserved. Forced to 0 by hardware.

Bit 4 = **LECIE** *Last Error Code Interrupt Enable*
0: ERRI bit will not be set when the error code in LEC[2:0] is set by hardware on error detection.
1: ERRI bit will be set when the error code in LEC[2:0] is set by hardware on error detection.

Bit 3 = Reserved. Forced to 0 by hardware.

Bit 2 = **BOFIE** *Bus-Off Interrupt Enable*
0: ERRI bit will not be set when BOFF is set.
1: ERRI bit will be set when BOFF is set.

Bit 1 = **EPVIE** *Error Passive Interrupt Enable*
0: ERRI bit will not be set when EPVF is set.
1: ERRI bit will be set when EPVF is set.

Bit 0 = **EWGIE** *Error Warning Interrupt Enable*
0: ERRI bit will not be set when EWGF is set.
1: ERRI bit will be set when EWGF is set.

beCAN CONTROLLER (Cont'd)

TRANSMIT ERROR COUNTER REG. (TECR)

Read Only

Reset Value: 00h

7							0
TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0

TEC[7:0] is the least significant byte of the 9-bit Transmit Error Counter implementing part of the fault confinement mechanism of the CAN protocol.

RECEIVE ERROR COUNTER REG. (RECR)

Page: 00h — Read Only

Reset Value: 00h

7							0
REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0

REC[7:0] is the Receive Error Counter implementing part of the fault confinement mechanism of the CAN protocol. In case of an error during reception, this counter is incremented by 1 or by 8 depending on the error condition as defined by the CAN standard. After every successful reception the counter is decremented by 1 or reset to 120 if its value was higher than 128. When the counter value exceeds 127, the CAN controller enters the error passive state.

CAN DIAGNOSIS REGISTER (CDGR)

All bits of this register are set and clear by software.
Read / Write
Reset Value: 0000 1100 (0Ch)

7							0
0	0	0	0	RX	SAMP	SILM	LBKM

Bit 3 = **RX** *CAN Rx Signal*
- Read
Monitors the actual value of the **CAN_RX** Pin.

Bit 2 = **SAMP** *Last Sample Point*
- Read
The value of the last sample point.

Bit 1 = **SILM** *Silent Mode*
- Read/Set/Clear
0: Normal operation
1: Silent Mode

Bit 0 = **LBKM** *Loop Back Mode*
- Read/Set/Clear
0: Loop Back Mode disabled
1: Loop Back Mode enabled

CAN BIT TIMING REGISTER 0 (CBTR0)

This register can only be accessed by the software when the CAN hardware is in configuration mode.
Read / Write
Reset Value: 0000 0000 (00h)

7							0
SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0

Bit 7:6 **SJW[1:0]** *Resynchronization Jump Width*
These bits define the maximum number of time quanta the CAN hardware is allowed to lengthen or shorten a bit to perform the resynchronization. Resynchronization Jump Width = (SJW+1).

Bit 5:0 **BRP[5:0]** *Baud Rate Prescaler*
These bits define the length of a time quantum.
$tq = (BRP+1)/f_{CPU}$

For more information on bit timing, please refer to Section 0.1.4.6 Bit Timing.

CAN BIT TIMING REGISTER 1 (CBTR1)

Read / Write
Reset Value: 0001 0011 (23h)

7							0
0	BS22	BS21	BS20	BS13	BS12	BS11	BS10

Bit 7 = Reserved. Forced to 0 by hardware.

Bits 6:4 **BS2[2:0]** *Time Segment 2*
These bits define the number of time quanta in Time Segment 2.
Time Segment 2 = (BS2+1)

beCAN CONTROLLER (Cont'd)

Bits 3:0 **BS1[3:0]** *Time Segment 1*
These bits define the number of time quanta in Time Segment 1

Time Segment 1 = (BS1+1)

For more information on bit timing, please refer to Section 0.1.4.6 Bit Timing.

CAN FILTER PAGE SELECT REGISTER (CPSR)

All bits of this register are set and cleared by software.
Read / Write
Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	FPS2	FPS1	FPS0

Bits 7:3 = Reserved. Forced to 0 by hardware.

Bits 2:0 = **PS[2:0]** *Page Select*
- Read/Write
This register contains the page number.

Table 31. Filter Page Selection

PS[2:0]	Page Selected
0	Tx Mailbox 0
1	Tx Mailbox 1
2	Acceptance Filter 0:1
3	Acceptance Filter 2:3
4	Acceptance Filter 4:5
5	Reserved
6	Configuration/Diagnosis
7	Receive FIFO

beCAN CONTROLLER (Cont'd)

10.9.8.2 Mailbox Registers

This chapter describes the registers of the transmit and receive mailboxes. Refer to Section 0.1.4.4 Message Storage for detailed register mapping.

Transmit and receive mailboxes have the same registers except:

– MCSR register in a transmit mailbox is replaced by MFMI register in a receive mailbox.

– A receive mailbox is always write protected.

– A transmit mailbox is write enable only while empty, corresponding TME bit in the CTPR register set.

MAILBOX CONTROL STATUS REGISTER (MCSR)

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

7							0
0	0	TERR	ALST	TXOK	RQCP	ABRQ	TXRQ

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

Bit 5 = **TERR** *Transmission Error*
- Read
This bit is updated by hardware after each transmission attempt.
0: The previous transmission was successful
1: The previous transmission failed due to an error

Bit 4 = **ALST** *Arbitration Lost*
- Read
This bit is updated by hardware after each transmission attempt.
0: The previous transmission was successful
1: The previous transmission failed due to an arbitration lost

Bit 3 = **TXOK** *Transmission OK*
- Read
The hardware updates this bit after each transmission attempt.

0: The previous transmission failed
1: The previous transmission was successful

Note: This bit has the same value as the corresponding TXOKx bit in the CTSR register.

Bit 2 = **RQCP** *Request Completed*
- Read/Clear
Set by hardware when the last request (transmit or abort) has been performed.

Cleared by software writing a "1" or by hardware on transmission request.

Note: This bit has the same value as the corresponding RQCPx bit of the CTSR register.

Clearing this bit clears all the status bits (TXOK, ALST and TERR) in the MCSR register and the RQCP and TXOK bits in the CTSR register.

Bit 1 = **ABRQ** *Abort Request for Mailbox*
- Read/Set
Set by software to abort the transmission request for the corresponding mailbox.

Cleared by hardware when the mailbox becomes empty.

Setting this bit has no effect when the mailbox is not pending for transmission.

Bit 0 = **TXRQ** *Transmit Mailbox Request*

- Read/Set
Set by software to request the transmission for the corresponding mailbox.

Cleared by hardware when the mailbox becomes empty.

Note: This register is implemented only in transmit mailboxes. In receive mailboxes, the MFMI register is mapped at this location.

beCAN CONTROLLER (Cont'd)

MAILBOX FILTER MATCH INDEX (MFMI)

This register is read only.

Reset Value: 0000 0000 (00h)

7							0
FMI7	FMI6	FMI5	FMI4	FMI3	FMI2	FMI1	FMI0

Bits 7:0 = **FMI[7:0]** *Filter Match Index*
This register contains the index of the filter the message stored in the mailbox passed through. For more details on identifier filtering please refer to Section 0.1.4.3 - **Filter Match Index** paragraph.

Note: This register is implemented only in receive mailboxes. In transmit mailboxes, the MCSR register is mapped at this location.

MAILBOX IDENTIFIER REGISTERS (MIDR[3:0])

Read / Write
Reset Value: Undefined

MIDR0

7							0
0	IDE	RTR	STID10	STID9	STID8	STID7	STID6

Bit 7 = Reserved. Forced to 0 by hardware.

Bit 6 = **IDE** *Extended Identifier*
This bit defines the identifier type of message in the mailbox.

0: Standard identifier.
1: Extended identifier.

Bit 5 = **RTR** *Remote Transmission Request*

0: Data frame
1: Remote frame

Bits 4:0 = **STID[10:6]** *Standard Identifier*
5 most significant bits of the standard part of the identifier.

MIDR1

7							0
STID5	STID4	STID3	STID2	STID1	STID0	EXID17	EXID16

Bits 7:2 = **STID[5:0]** *Standard Identifier*
6 least significant bits of the standard part of the identifier.

Bits 1:0 = **EXID[17:16]** *Extended Identifier*
2 most significant bits of the extended part of the identifier.

MIDR2

7							0
EXID15	EXID14	EXID13	EXID12	EXID11	EXID10	EXID9	EXID8

Bits 7:0 = **EXID[15:8]** *Extended Identifier*
Bits 15 to 8 of the extended part of the identifier.

MIDR3

7							0
EXID7	EXID6	EXID5	EXID4	EXID3	EXID2	EXID1	EXID0

Bits 7:1 = **EXID[6:0]** *Extended Identifier*
6 least significant bits of the extended part of the identifier.

beCAN CONTROLLER (Cont'd)

MAILBOX DATA LENGTH CONTROL REGISTER (MDLC)

All bits of this register is write protected when the mailbox is not in empty state.

Read / Write
Reset Value: xxxx xxxx (xxh)

7							0
0	0	0	0	DLC3	DLC2	DLC1	DLC0

Bit 7 = Reserved, must be kept cleared.

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

Bits 3:0 = **DLC[3:0]** *Data Length Code*
This field defines the number of data bytes a data frame contains or a remote frame request.

MAILBOX DATA REGISTERS (MDAR[7:0])

All bits of this register are write protected when the mailbox is not in empty state.

Read / Write
Reset Value: Undefined

7							0
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0

Bits 7:0 = **DATA[7:0]** *Data*
A data byte of the message. A message can contain from 0 to 8 data bytes.

beCAN CONTROLLER (Cont'd)

10.9.8.3 CAN Filter Registers

CAN FILTER CONFIGURATION REG.0 (CFCR0)

All bits of this register are set and cleared by software.
Read / Write
Reset Value: 0000 0000 (00h)

7							0
0	FSC11	FSC10	FACT1	0	FSC01	FSC00	FACT0

Note: To modify the FFAx and FSCx bits, the beCAN must be in INIT mode.

Bit 7 = Reserved. Forced to 0 by hardware.

Bits 6:5 = **FSC1[1:0]** *Filter Scale Configuration*
These bits define the scale configuration of Filter 1.

Bit 4 = **FACT1** *Filter Active*
The software sets this bit to activate Filter 1. To modify the Filter 1 registers (CF1R[7:0]), the FACT1 bit must be cleared.
0: Filter 1 is not active
1: Filter 1 is active

Bit 3 = Reserved. Forced to 0 by hardware.

Bits 2:1 = **FSC0[1:0]** *Filter Scale Configuration*
These bits define the scale configuration of Filter 0.

Bit 0 = **FACT0** *Filter Active*
The software sets this bit to activate Filter 0. To modify the Filter 0 registers (CF0R[0:7]), the FACT0 bit must be cleared.
0: Filter 0 is not active
1: Filter 0 is active

CAN FILTER CONFIGURATION REG.1 (CFCR1)

All bits of this register are set and cleared by software.
Read / Write
Reset Value: 0000 0000 (00h)

7							0
0	FSC31	FSC30	FACT3	0	FSC21	FSC20	FACT2

Bit 7 = Reserved. Forced to 0 by hardware.

Bits 6:5 = **FSC3[1:0]** *Filter Scale Configuration*
These bits define the scale configuration of Filter 3.

Bit 4 = **FACT3** *Filter Active*
The software sets this bit to activate filter 3. To modify the Filter 3 registers (CF3R[0:7]) the FACT3 bit must be cleared.
0: Filter 3 is not active
1: Filter 3 is active

Bit 3 = Reserved. Forced to 0 by hardware.

Bits 2:1 = **FSC2[1:0]** *Filter Scale Configuration*
These bits define the scale configuration of Filter 2.

Bit 0 = **FACT2** *Filter Active*
The software sets this bit to activate Filter 2. To modify the Filter 2 registers (CF2R[0:7]), the FACT2 bit must be cleared.
0: Filter 2 is not active
1: Filter 2 is active

beCAN CONTROLLER (Cont'd)

CAN FILTER CONFIGURATION REG.1 (CFCR2)

All bits of this register are set and cleared by software.
Read / Write
Reset Value: 0000 0000 (00h)

7							0
0	FSC51	FSC50	FACT5	0	FSC41	FSC40	FACT4

Bit 7 = Reserved. Forced to 0 by hardware.

Bits 6:5 = **FSC5[1:0]** *Filter Scale Configuration*
These bits define the scale configuration of Filter 5.

Bit 4 = **FACT5** *Filter Active*
The software sets this bit to activate filter 5. To modify the Filter 5 registers (CF5R[0:7]) the FACT5 bit must be cleared.
0: Filter 5 is not active
1: Filter 5 is active

Bit 3 = Reserved. Forced to 0 by hardware.

Bits 2:1 = **FSC4[1:0]** *Filter Scale Configuration*
These bits define the scale configuration of Filter 4.

Bit 0 = **FACT4** *Filter Active*
The software sets this bit to activate Filter 4. To modify the Filter 4 registers (CF4R[0:7]), the FACT4 bit must be cleared.
0: Filter 4 is not active
1: Filter 4 is active

CAN FILTER MODE REGISTER (CFMR0)

All bits of this register are set and cleared by software.
Read / Write
Reset Value: 0000 0000 (00h)

7							0
FMH3	FML3	FMH2	FML2	FMH1	FML1	FMH0	FML0

Bit 7 = **FMH3** *Filter Mode High*
Mode of the high registers of Filter 3.
0: High registers are in mask mode
1: High registers are in identifier list mode

Bit 6 = **FML3** *Filter Mode Low*
Mode of the low registers of Filter 3.
0: Low registers are in mask mode
1: Low registers are in identifier list mode

Bit 5 = **FMH2** *Filter Mode High*
Mode of the high registers of Filter 2.
0: High registers are in mask mode
1: High registers are in identifier list mode

Bit 4 = **FML2** *Filter Mode Low*
Mode of the low registers of Filter 2.
0: Low registers are in mask mode
1: Low registers are in identifier list mode

Bit 3 = **FMH1** *Filter Mode High*
Mode of the high registers of Filter 1.
0: High registers are in mask mode
1: High registers are in identifier list mode

Bit 2 = **FML1** *Filter Mode Low*
Mode of the low registers of filter 1.
0: Low registers are in mask mode
1: Low registers are in identifier list mode

Bit 1 = **FMH0** *Filter Mode High*
Mode of the high registers of filter 0.
0: High registers are in mask mode
1: High registers are in identifier list mode

Bit 0 = **FML0** *Filter Mode Low*
Mode of the low registers of filter 0.
0: Low registers are in mask mode
1: Low registers are in identifier list mode

beCAN CONTROLLER (Cont'd)

CAN FILTER MODE REGISTER (CFMR1)

All bits of this register are set and cleared by software.
Read / Write
Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	FMH5	FML5	FMH4	FML4

Bits 7:4 = Reserved. Forced to 0 by hardware.

Bit 3 = **FMH5** *Filter Mode High*
Mode of the high registers of Filter 5.
0: High registers are in mask mode
1: High registers are in identifier list mode

Bit 2 = **FML5** *Filter Mode Low*
Mode of the low registers of filter 5.
0: Low registers are in mask mode
1: Low registers are in identifier list mode

Bit 1 = **FMH4** *Filter Mode High*
Mode of the high registers of filter 4.
0: High registers are in mask mode
1: High registers are in identifier list mode

Bit 0 = **FML4** *Filter Mode Low*
Mode of the low registers of filter 4.
0: Low registers are in mask mode
1: Low registers are in identifier list mode

FILTER x REGISTER[7:0] (CFxR[7:0])

Read / Write
Reset Value: Undefined

7							0
FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0

In all configurations:

Bits 7:0 = **FB[7:0]** *Filter Bits*

Identifier
Each bit of the register specifies the level of the corresponding bit of the expected identifier.
0: Dominant bit is expected
1: Recessive bit is expected

Mask
Each bit of the register specifies whether the bit of the associated identifier register must match with the corresponding bit of the expected identifier or not.
0: Don't care, the bit is not used for the comparison
1: Must match, the bit of the incoming identifier must have the same level has specified in the corresponding identifier register of the filter.

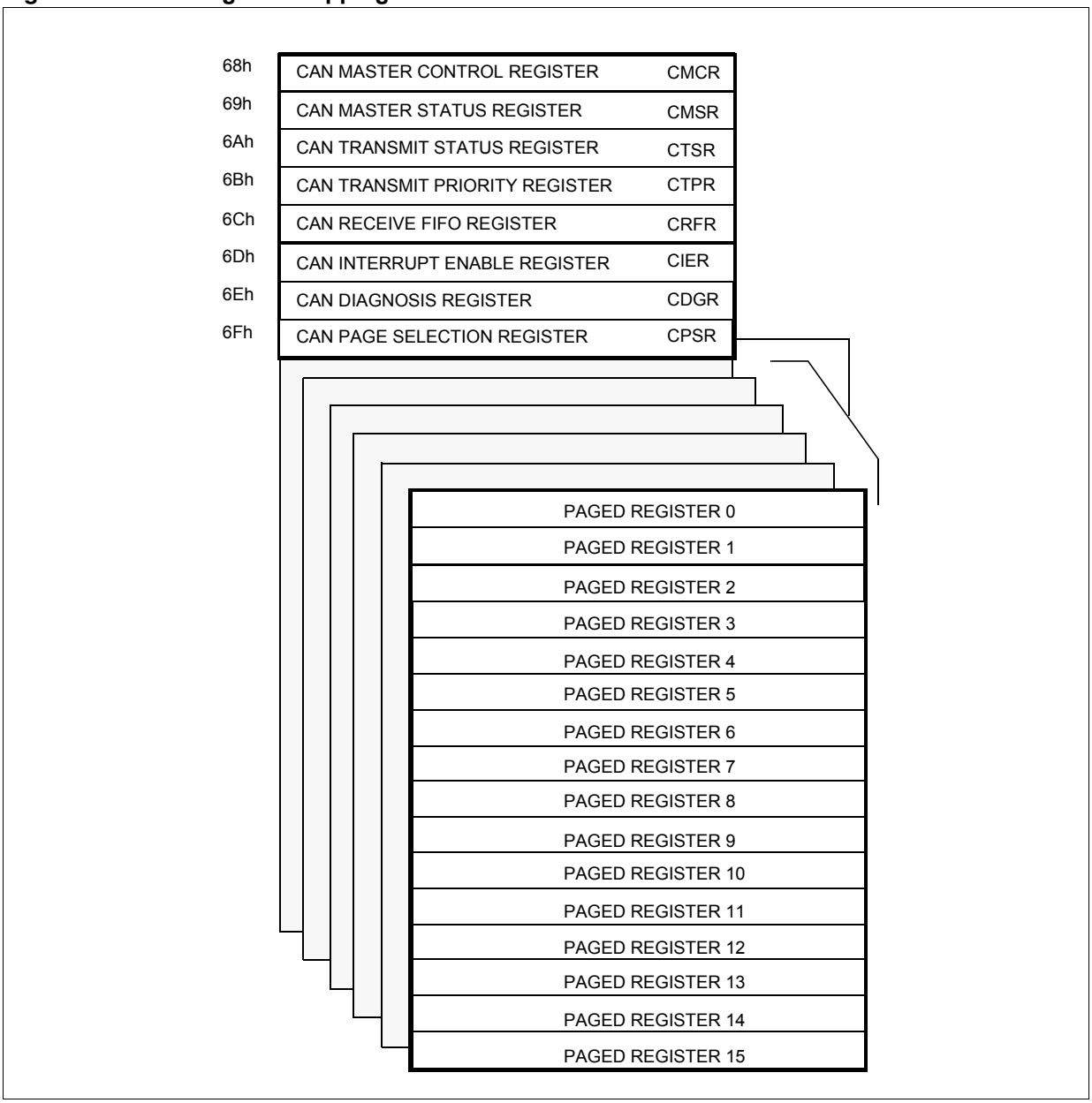
Note: Each filter x is composed of 8 registers, CFxR[7:0]. Depending on the scale and mode configuration of the filter the function of each register can differ. For the filter mapping, functions description and mask registers association, refer to Section 0.1.4.3 Identifier Filtering.

A Mask/Identifier register in **mask mode** has the same bit mapping as in **identifier list** mode.

Note: To modify these registers, the corresponding FACT bit in the CFCR register must be cleared.

beCAN CONTROLLER (Cont'd)

Figure 115. CAN Register Mapping



beCAN CONTROLLER (Cont'd)

10.9.8.4 Page Mapping for CAN

	PAGE 0	PAGE 1	PAGE 2	PAGE 3	PAGE 4
70h	MCSR	MCSR	CF0R0	CF2R0	CF4R0
71h	MDLC	MDLC	CF0R1	CF2R1	CF4R1
72h	MIDR0	MIDR0	CF0R2	CF2R2	CF4R2
73h	MIDR1	MIDR1	CF0R3	CF2R3	CF4R3
74h	MIDR2	MIDR2	CF0R4	CF2R4	CF4R4
75h	MIDR3	MIDR3	CF0R5	CF2R5	CF4R5
76h	MDAR0	MDAR0	CF0R6	CF2R6	CF4R6
77h	MDAR1	MDAR1	CF0R7	CF2R7	CF4R7
78h	MDAR2	MDAR2	CF1R0	CF3R0	CF5R0
79h	MDAR3	MDAR3	CF1R1	CF3R1	CF5R1
7Ah	MDAR4	MDAR4	CF1R2	CF3R2	CF5R2
7Bh	MDAR5	MDAR5	CF1R3	CF3R3	CF5R3
7Ch	MDAR6	MDAR6	CF1R4	CF3R4	CF5R4
7Dh	MDAR7	MDAR7	CF1R5	CF3R5	CF5R5
7Eh	MTSLR	MTSLR	CF1R6	CF3R6	CF5R6
7Fh	MTSHR	MTSHR	CF1R7	CF3R7	CF5R7
	Tx Mailbox 0	Tx Mailbox 1	Acceptance Filter 0:1	Acceptance Filter 2:3	Acceptance Filter 4:5

	PAGE 6	PAGE 7
70h	CESR	MFMI
71h	CEIER	MDLC
72h	TECR	MIDR0
73h	RECR	MIDR1
74h	BTCR0	MIDR2
75h	BTCR1	MIDR3
76h	Reserved	MDAR0
77h	Reserved	MDAR1
78h	CFMR0	MDAR2
79h	CFMR1	MDAR3
7Ah	CFCR0	MDAR4
7Bh	CFCR1	MDAR5
7Ch	CFCR2	MDAR6
7Dh	Reserved	MDAR7
7Eh	Reserved	MTSLR
7Fh	Reserved	MTSHR
	Configuration/Diagnosis	Receive FIFO

beCAN CONTROLLER (Cont'd)

Table 32. beCAN Control and Status Page - Register Map and Reset Values

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
68h	**CMCR**		ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
	Reset Value	0	0	0	0	0	0	1	0
69h	**CMSR**			REC	TRAN	WKUI	ERRI	SLAK	INAK
	Reset Value	0	0	0	0	0	0	1	0
6Ah	**CTSR**			TXOK1	TXOK0			RQCP1	RQCP0
	Reset Value	0	0	0	0	0	0	0	0
6Bh	**CTPR**		LOW1	LOW0		TME1	TME0		CODE0
	Reset Value	0	0	0	1	1	1	0	0
6Ch	**CRFR**			RFOM	FOVR	FULL		FMP1	FMP0
	Reset Value	0	0	0	0	0	0	0	0
6Dh	**CIER**	WKUIE	0	0	0	FOVIE0	FFIE0	FMPIE0	TMEIE
	Reset Value	0				0	0	0	0
6Eh	**CDGR**					RX	SAMP	SILM	LBKM
	Reset Value	0	0	0	0	1	1	0	0
6Fh	**CFPSR**						FPS2	FPS1	FPS0
	Reset Value	0	0	0	0	0	0	0	0

Table 33. beCAN Mailbox Pages - Register Map and Reset Values

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
70h	**MFMI**	FMI7	FMI6	FMI5	FMI4	FMI3	FMI2	FMI1	FMI0
Receive	Reset Value	0	0	0	0	0	0	0	0
70h	**MCSR**			TERR	ALST	TXOK	RQCP	ABRQ	TXRQ
Transmit	Reset Value	0	0	0	0	0	0	0	0
71h	**MDLC**	0				DLC3	DLC2	DLC1	DLC0
	Reset Value	x	x	x	x	x	x	x	x
72h	**MIDR0**		IDE	RTR	STID10	STID9	STID8	STID7	STID6
	Reset Value	x	x	x	x	x	x	x	x
73h	**MIDR1**	STID5	STID4	STID3	STID2	STID1	STID0	EXID17	EXID16
	Reset Value	x	x	x	x	x	x	x	x
74h	**MIDR2**	EXID15	EXID14	EXID13	EXID12	EXID11	EXID10	EXID9	EXID8
	Reset Value	x	x	x	x	x	x	x	x
75h	**MIDR3**	EXID7	EXID6	EXID5	EXID4	EXID3	EXID2	EXID1	EXID0
	Reset Value	x	x	x	x	x	x	x	x
76h:7Dh	**MDAR[0:7]**	MDAR7	MDAR6	MDAR5	MDAR4	MDAR3	MDAR2	MDAR1	MDAR0
	Reset Value	x	x	x	x	x	x	x	x

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
7Eh	**MTSLR**	TIME7	TIME6	TIME5	TIME4	TIME3	TIME2	TIME1	TIME0
	Reset Value	x	x	x	x	x	x	x	x
7Fh	**MTSHR**	TIME15	TIME14	TIME13	TIME12	TIME11	TIME10	TIME9	TIME8
	Reset Value	x	x	x	x	x	x	x	x

Table 34. beCAN Filter Configuration Page - Register Map and Reset Values

Address (Hex.)	Register Name	7	6	5	4	3	2	1	0
70h	**CESR**		LEC2	LEC1	LEC0		BOFF	EPVF	EWGF
	Reset Value	0	0	0	0	0	0	0	0
71h	**CEIER**	ERRIE			LECIE		BOFIE	EPVIE	EWGIE
	Reset Value	0	0	0	0	0	0	0	0
72h	**TECR**	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0
	Reset Value	0	0	0	0	0	0	0	0
73h	**RECR**	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0
	Reset Value	0	0	0	0	0	0	0	0
74h	**CBTR0**	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
	Reset Value	0	0	0	0	0	0	0	0
75h	**CBTR1**		BS22	BS21	BS20	BS13	BS12	BS11	BS10
	Reset Value	0	0	1	0	0	0	1	1
76h	**Reserved**	x	x	x	x	x	x	x	x
77h	**Reserved**	x	x	x	x	x	x	x	x
78h	**CFMR0**	FMH3	FML3	FMH2	FML2	FMH1	FML1	FMH0	FML0
	Reset Value	0	0	0	0	0	0	0	0
79h	**CFMR1**	0	0	0	0	FMH5	FML5	FMH4	FML4
	Reset Value					0	0	0	0
7Ah	**CFCR0**	FFA1	FSC11	FSC10	FACT1	FFA0	FSC01	FSC00	FACT0
	Reset Value	0	0	0	0	0	0	0	0
7Bh	**CFCR1**	FFA3	FSC31	FSC30	FACT3	FFA2	FSC21	FSC20	FACT2
	Reset Value	0	0	0	0	0	0	0	0
7Ch	**CFCR2**	FFA5	FSC51	FSC50	FACT5	FFA4	FSC41	FSC40	FACT4
	Reset Value	0	0	0	0	0	0	0	0

10.10 10-BIT A/D CONVERTER (ADC)

10.10.1 Introduction

The on-chip Analog to Digital Converter (ADC) peripheral is a 10-bit, successive approximation converter with internal sample and hold circuitry. This peripheral has up to 16 multiplexed analog input channels (refer to device pin out description) that allow the peripheral to convert the analog voltage levels from up to 16 different sources.

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

10.10.2 Main Features

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

The block diagram is shown in Figure 116.

10.10.3 Functional Description

10.10.3.1 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.

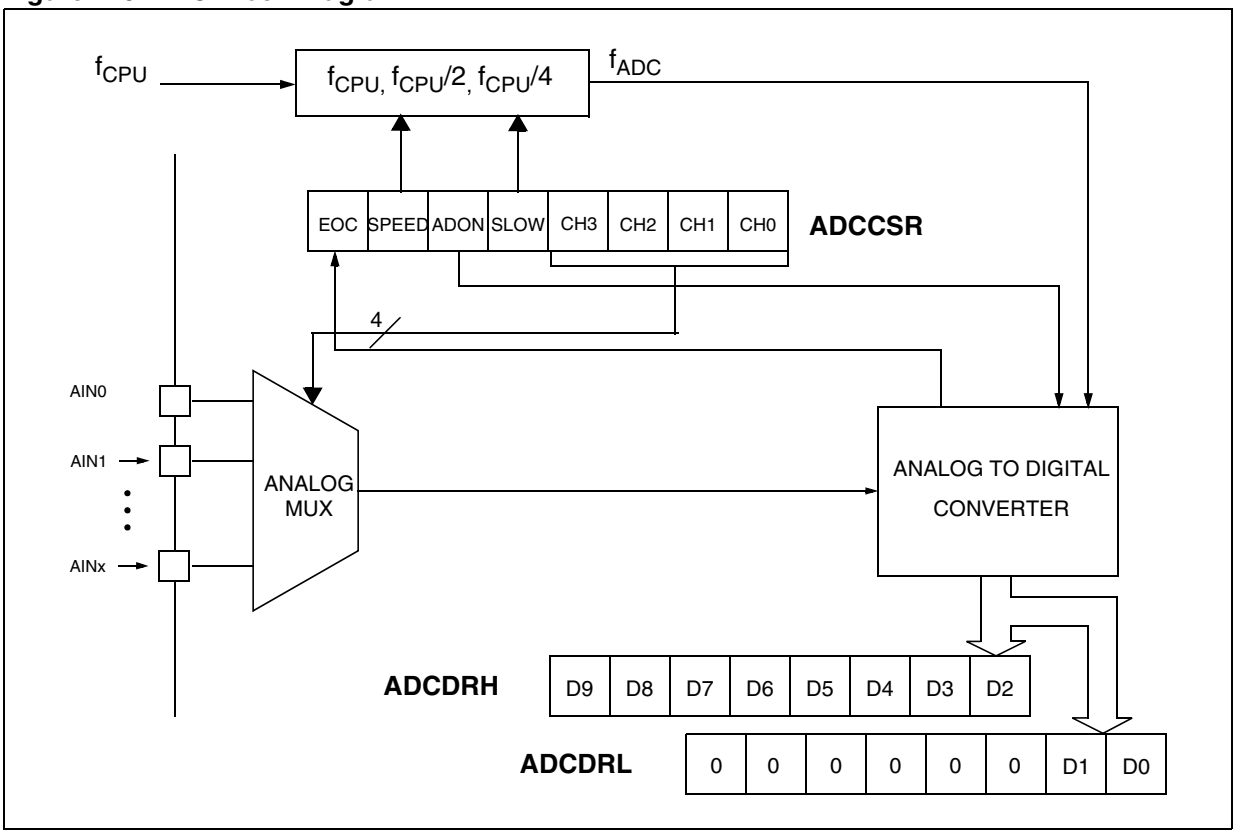
If the input voltage (V_{AIN}) is greater than V_{DDA} (high-level voltage reference) then the conversion result is FFh in the ADCDRH register and 03h in the ADCDRL register (without overflow indication).

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 will result in a loss of accuracy due to leakage and sampling not being completed in the allotted time.

Figure 116. ADC Block Diagram



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

10.10.3.2 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 CS[3:0] 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 EOC bit
2. Read the ADCDRL register
3. Read the ADCDRH register. This clears EOC automatically.

To read only 8 bits, perform the following steps:

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

10.10.3.3 Changing the conversion channel

The application can change channels during conversion. When software modifies the CH[3:0] bits in the ADCCSR register, the current conversion is stopped, the EOC bit is cleared, and the A/D converter starts converting the newly selected channel.

10.10.3.4 ADCDR consistency

If an End Of Conversion event occurs after software has read the ADCDRLSB but before it has read the ADCDRMSB, there would be a risk that the two values read would belong to different samples.

To guarantee consistency:

– The ADCDRL and the ADCDRH registers are locked when the ADCCRL is read
– The ADCDRL and the ADCDRH registers are unlocked when the ADCDRH register is read or when ADON is reset.

This is important, as the ADCDR register will not be updated until the ADCDRH register is read.

10.10.4 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 wakeup from Halt mode, the A/D Converter requires a stabilization time t_{STAB} (see Electrical Characteristics) before accurate conversions can be performed.

10.10.5 Interrupts

None.

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

10.10.6 Register Description

CONTROL/STATUS REGISTER (ADCCSR)

Read/Write (Except bit 7 read only)

Reset Value: 0000 0000 (00h)

7							0
EOC	SPEED	ADON	SLOW	CH3	CH2	CH1	CH0

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

Bit 6 = **SPEED** *A/D clock selection*
This bit is set and cleared by software.

Table 35. A/D Clock Selection

f_{ADC}	SLOW	SPEED
$f_{CPU}/2$	0	0
f_{CPU} (where f_{CPU} <= 4 MHz)	0	1
$f_{CPU}/4$	1	0
$f_{CPU}/2$ (same frequency as SLOW=0, SPEED=0)	1	1

Bit 5 = **ADON** *A/D Converter on*
This bit is set and cleared by software.
0: Disable ADC and stop conversion
1: Enable ADC and start conversion

Bit 4 = **SLOW** *A/D Clock Selection*
This bit is set and cleared by software. It works together with the SPEED bit. Refer to Table 35.

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

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

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

DATA REGISTER (ADCDRH)

Read Only

Reset Value: 0000 0000 (00h)

7							0
D9	D8	D7	D6	D5	D4	D3	D2

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

DATA REGISTER (ADCDRL)

Read Only

Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	0	0	D1	D0

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

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

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

Table 36. ADC Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
45h	**ADCCSR**	EOC	SPEED	ADON	SLOW	CH3	CH2	CH1	CH0
	Reset Value	0	0	0	0	0	0	0	0
46h	**ADCDRH**	D9	D8	D7	D6	D5	D4	D3	D2
	Reset Value	0	0	0	0	0	0	0	0
47h	**ADCDRL**	0	0	0	0	0	0	D1	D0
	Reset Value	0	0	0	0	0	0	0	0

11 INSTRUCTION SET

11.1 CPU ADDRESSING MODES

The CPU 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 CPU 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 37. CPU Addressing Mode Overview

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

INSTRUCTION SET OVERVIEW (Cont'd)

11.1.1 Inherent

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

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

11.1.2 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

11.1.3 Direct

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

The direct addressing mode consists of two submodes:

Direct (short)

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

Direct (long)

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

11.1.4 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 one byte after the opcode and allows 00 - 1FE addressing space.

Indexed (long)

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

11.1.5 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.

INSTRUCTION SET OVERVIEW (Cont'd)

11.1.6 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 38. Instructions Supporting Direct, Indexed, Indirect and Indirect Indexed Addressing Modes

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

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

11.1.7 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 is following the opcode.

Relative (Indirect)

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

INSTRUCTION SET OVERVIEW (Cont'd)

11.2 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 one to four opcodes.

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

The whole instruction becomes:

PC-2 End of previous instruction

PC-1 Prebyte

PC Opcode

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

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

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

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

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

INSTRUCTION SET OVERVIEW (Cont'd)

Mnemo	Description	Function/Example	Dst	Src	I1	H	I0	N	Z	C
ADC	Add with Carry	A = A + M + C	A	M		H		N	Z	C
ADD	Addition	A = A + M	A	M		H		N	Z	C
AND	Logical And	A = A . M	A	M				N	Z	
BCP	Bit compare A, Memory	tst (A . M)	A	M				N	Z	
BRES	Bit Reset	bres Byte, #3	M							
BSET	Bit Set	bset Byte, #3	M							
BTJF	Jump if bit is false (0)	btjf Byte, #3, Jmp1	M							C
BTJT	Jump if bit is true (1)	btjt Byte, #3, Jmp1	M							C
CALL	Call subroutine									
CALLR	Call subroutine relative									
CLR	Clear		reg, M					0	1	
CP	Arithmetic Compare	tst(Reg - M)	reg	M				N	Z	C
CPL	One Complement	A = FFH-A	reg, M					N	Z	1
DEC	Decrement	dec Y	reg, M					N	Z	
HALT	Halt				1		0			
IRET	Interrupt routine return	Pop CC, A, X, PC			I1	H	I0	N	Z	C
INC	Increment	inc X	reg, M					N	Z	
JP	Absolute Jump	jp [TBL.w]								
JRA	Jump relative always									
JRT	Jump relative									
JRF	Never jump	jrf *								
JRIH	Jump if ext. INT pin = 1	(ext. INT pin high)								
JRIL	Jump if ext. INT pin = 0	(ext. INT pin low)								
JRH	Jump if H = 1	H = 1 ?								
JRNH	Jump if H = 0	H = 0 ?								
JRM	Jump if I1:0 = 11	I1:0 = 11 ?								
JRNM	Jump if I1:0 <> 11	I1:0 <> 11 ?								
JRMI	Jump if N = 1 (minus)	N = 1 ?								
JRPL	Jump if N = 0 (plus)	N = 0 ?								
JREQ	Jump if Z = 1 (equal)	Z = 1 ?								
JRNE	Jump if Z = 0 (not equal)	Z = 0 ?								
JRC	Jump if C = 1	C = 1 ?								
JRNC	Jump if C = 0	C = 0 ?								
JRULT	Jump if C = 1	Unsigned <								
JRUGE	Jump if C = 0	Jmp if unsigned >=								
JRUGT	Jump if (C + Z = 0)	Unsigned >								

INSTRUCTION SET OVERVIEW (Cont'd)

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

12 ELECTRICAL CHARACTERISTICS

12.1 PARAMETER CONDITIONS

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

12.1.1 Minimum and Maximum Values

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

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

12.1.2 Typical Values

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

Typical ADC accuracy values are determined by characterization of a batch of samples from a standard diffusion lot over the full temperature range, where 95% of the devices have an error less than or equal to the value indicated (mean±2Σ).

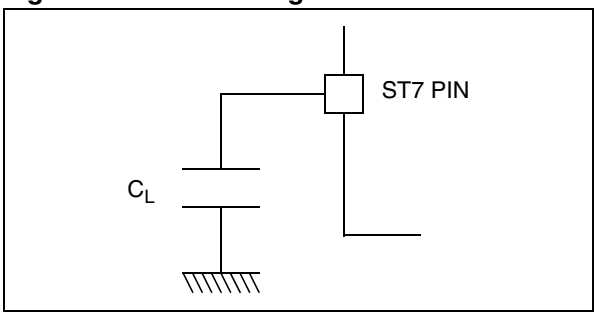
12.1.3 Typical Curves

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

12.1.4 Loading Capacitor

The loading conditions used for pin parameter measurement are shown in Figure 117.

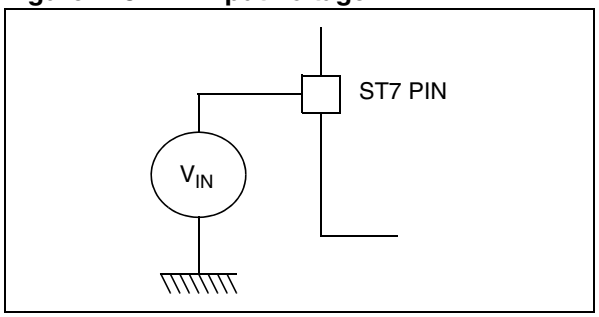
Figure 117. Pin Loading Conditions



12.1.5 Pin Input Voltage

The input voltage measurement on a pin of the device is described in Figure 118.

Figure 118. Pin input voltage



12.2 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.

12.2.1 Voltage Characteristics

Symbol	Ratings	Maximum value	Unit
V_{DD} - V_{SS}	Supply voltage	6.5	V
V_{PP} - V_{SS}	Programming Voltage	13	
V_{IN}	Input voltage on any pin[1)2)]	V_{SS} - 0.3 to V_{DD} + 0.3	
$\lvert\Delta V_{DDx}\rvert$ and $\lvert\Delta V_{SSx}\rvert$	Variations between different digital power pins	50	mV
$\lvert V_{SSA} - V_{SSx}\rvert$	Variations between digital and analog ground pins	50	
$V_{ESD(HBM)}$	Electro-static discharge voltage (Human Body Model)	see Section 12.8.3 on page 233	
$V_{ESD(MM)}$	Electro-static discharge voltage (Machine Model)		

12.2.2 Current Characteristics

Symbol	Ratings	Maximum value	Unit
I_{VDD}	Total current into V_{DD} power lines (source)[3)]	150	mA
I_{VSS}	Total current out of V_{SS} ground lines (sink)[3)]		
I_{IO}	Output current sunk by any standard I/O and control pin	25	
	Output current sunk by any high sink I/O pin	50	
	Output current source by any I/Os and control pin	- 25	
$I_{INJ(PIN)}$[2)4)]	Injected current on V_{PP} pin	± 5	
	Injected current on $\overline{\text{RESET}}$ pin		
	Injected current on OSC1 and OSC2 pins		
	Injected current on PB3 (on Flash devices)	+5	
	Injected current on any other pin[5)]	± 5	
$\Sigma I_{INJ(PIN)}$[2)]	Total injected current (sum of all I/O and control pins)[5)]	± 25	

12.2.3 Thermal Characteristics

Symbol	Ratings	Value	Unit
T_{STG}	Storage temperature range	-65 to +150	°C
T_J	Maximum junction temperature (see Section 13.2 "THERMAL CHARACTERISTICS")		

Notes:

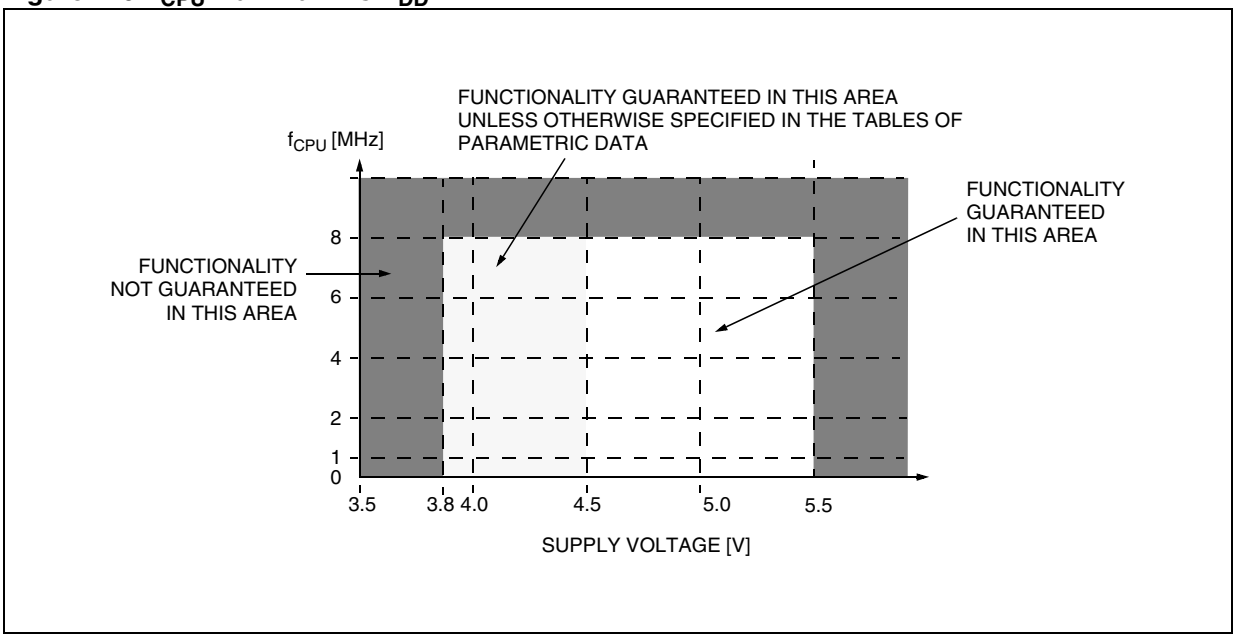
1. Directly connecting the $\overline{\text{RESET}}$ and I/O pins to V_{DD} or V_{SS} could damage the device if an unintentional internal reset is generated or an unexpected change of the I/O configuration occurs (for example, due to a corrupted program counter). To guarantee safe operation, this connection has to be done through a pull-up or pull-down resistor (typical: 4.7kΩ for $\overline{\text{RESET}}$, 10kΩ for I/Os). Unused I/O pins must be tied in the same way to V_{DD} or V_{SS} according to their reset configuration.
2. $I_{INJ(PIN)}$ must never be exceeded. This is implicitly insured if V_{IN} maximum is respected. If V_{IN} maximum cannot be respected, the injection current must be limited externally to the $I_{INJ(PIN)}$ value. A positive injection is induced by $V_{IN}>V_{DD}$ while a negative injection is induced by $V_{IN} < V_{SS}$. For true open-drain pads, there is no positive injection current, and the corresponding V_{IN} maximum must always be respected
3. All power (V_{DD}) and ground (V_{SS}) lines must always be connected to the external supply.
4. Negative injection disturbs the analog performance of the device. See note in “10-BIT ADC CHARACTERISTICS” on page 245.
5. When several inputs are submitted to a current injection, the maximum $\Sigma I_{INJ(PIN)}$ is the absolute sum of the positive and negative injected currents (instantaneous values). These results are based on characterization with $\Sigma I_{INJ(PIN)}$ maximum current injection on four I/O port pins of the device.

12.3 OPERATING CONDITIONS

12.3.1 General Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
f_{CPU}	Internal clock frequency		0	8	MHz
V_{DD}	Extended Operating voltage	No Flash Write/Erase. Analog parameters not guaranteed.	3.8	4.5	V
	Standard Operating Voltage		4.5	5.5	
	Operating Voltage for Flash Write/Erase	V_{PP} = 11.4 to 12.6V	4.5	5.5	
T_A	Ambient temperature range	1 Suffix Version	0	70	°C
		5 Suffix Version	-10	85	
		6 Suffix Version	-40	85	
		7 Suffix Version		105	
		3 Suffix Version		125	

Figure 119. f_{CPU} Maximum vs V_{DD}



Note: It is mandatory to connect all available V_{DD} and V_{DDA} pins to the supply voltage and all V_{SS} and V_{SSA} pins to ground.

12.3.2 Operating Conditions with Low Voltage Detector (LVD)

Subject to general operating conditions for T_A.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{IT+(LVD)}$	Reset release threshold (V_{DD} rise)		4.0[1)]	4.2	4.5	V
$V_{IT-(LVD)}$	Reset generation threshold (V_{DD} fall)		3.8	4.0	4.25[1)]	
$V_{hys(LVD)}$	LVD voltage threshold hysteresis[1)]	$V_{IT+(LVD)}$-$V_{IT-(LVD)}$	150	200	250	mV
Vt_{POR}	V_{DD} rise time rate[1)]		6			µs/V
					100	ms/V
$t_{g(VDD)}$	V_{DD} glitches filtered (not detected) by LVD[1)]	Measured at $V_{IT-(LVD)}$			40	ns

Notes:

1. Data based on characterization results, not tested in production.

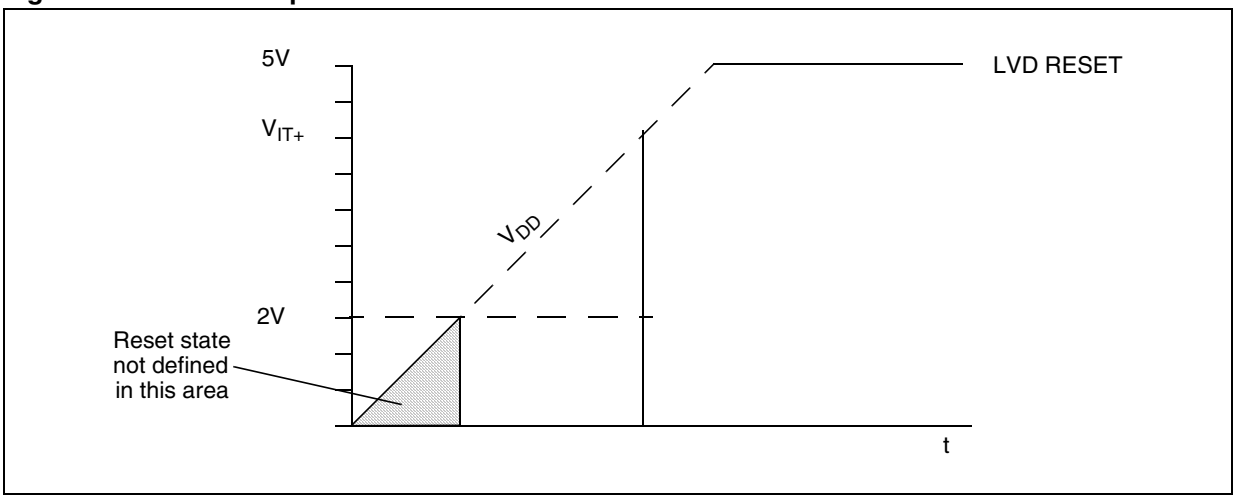
12.3.3 Auxiliary Voltage Detector (AVD) Thresholds

Subject to general operating conditions for T_A.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{IT+(AVD)}$	1⇒0 AVDF flag toggle threshold (V_{DD} rise)		4.4[1)]	4.6	4.9	V
$V_{IT-(AVD)}$	0⇒1 AVDF flag toggle threshold (V_{DD} fall)		4.2	4.4	4.65[1)]	
$V_{hys(AVD)}$	AVD voltage threshold hysteresis	$V_{IT+(AVD)}$-$V_{IT-(AVD)}$		250		mV
ΔV_{IT-}	Voltage drop between AVD flag set and LVD reset activated	$V_{IT-(AVD)}$-$V_{IT-(LVD)}$		450		

1. Data based on characterization results, not tested in production.

Figure 120. LVD Startup Behavior



Note: When the LVD is enabled, the MCU reaches its authorized operating voltage from a reset state. However, in some devices, the reset signal may be undefined until V_{DD} is approximately 2V. As a consequence, the I/Os may toggle when V_{DD} is below this voltage.

Because Flash write access is impossible below this voltage, the Flash memory contents will not be corrupted.

12.4 SUPPLY CURRENT CHARACTERISTICS

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

Symbol	Parameter	Conditions		Flash Devices		ROM Devices		Unit
				Typ [1)]	Max [2)]	Typ [1]	Max [2)]	
I_{DD}	Supply current in RUN mode [3)]	f_{OSC} = 2 MHz, f_{CPU} = 1 MHz f_{OSC} = 4 MHz, f_{CPU} = 2 MHz f_{OSC} = 8 MHz, f_{CPU} = 4 MHz f_{OSC} = 16 MHz, f_{CPU} = 8 MHz		1.8 3.2 6 10	3 5 8 15	1.1 2.2 4.4 8.9	2 3.5 6 12	mA
	Supply current in SLOW mode [3)]	f_{OSC} = 2 MHz, f_{CPU} = 62.5kHz f_{OSC} = 4 MHz, f_{CPU} = 125 kHz f_{OSC} = 8 MHz, f_{CPU} = 250 kHz f_{OSC} = 16 MHz, f_{CPU} = 500 kHz		0.5 0.6 0.85 1.25	2.7 3 3.6 4	0.1 0.2 0.4 0.8	0.2 0.4 0.8 1.5	
	Supply current in WAIT mode [3)]	f_{OSC} = 2 MHz, f_{CPU} = 1 MHz f_{OSC} = 4 MHz, f_{CPU} = 2 MHz f_{OSC} = 8 MHz, f_{CPU} = 4 MHz f_{OSC} = 16 MHz, f_{CPU} = 8 MHz		1 1.8 3.4 6.4	3 4 5 7	0.7 1.4 2.9 5.7	3 4 5 7	
	Supply current in SLOW WAIT mode [2)]	f_{OSC} = 2 MHz, f_{CPU} = 62.5 kHz f_{OSC} = 4 MHz, f_{CPU} = 125 kHz f_{OSC} = 8 MHz, f_{CPU} = 250 kHz f_{OSC} = 16 MHz, f_{CPU} = 500 kHz		0.4 0.5 0.6 0.8	1.2 1.3 1.8 2	0.07 0.14 0.28 0.56	0.12 0.25 0.5 1	
	Supply current in HALT mode [4)]	V_{DD} = 5.5V	-40°C ≤ T_A ≤ +85°C	<1	10	<1	10	µA
			-40°C ≤ T_A ≤ +125°C		50		50	
	Supply current in ACTIVE HALT mode [4)5)]			0.5	1.2	0.18	0.25	mA
	Supply current in AWUFH mode [4)5)]	V_{DD} = 5.5V	-40°C ≤ T_A ≤ +85°C	25	30	25	30	µA
			-40°C ≤ T_A ≤ +125°C		70		70	

Notes:

1. Typical data are based on T_A = 25°C, V_{DD} = 5V (4.5V ≤ V_{DD} ≤ 5.5V range).
2. Data based on characterization results, tested in production at V_{DD} max., f_{CPU} max. and T_A max.
3. Measurements are done in the following conditions:
- Program executed from Flash, CPU running with Flash (for flash devices).
- All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load)
- All peripherals in reset state.
- Clock input (OSC1) driven by external square wave.
- In SLOW and SLOW WAIT mode, f_{CPU} is based on f_{OSC} divided by 32.

   To obtain the total current consumption of the device, add the clock source (Section 12.5.3) and the peripheral power consumption (Section 12.4.2).
4. All I/O pins in input mode with a static value at V_{DD} or V_{SS} (no load). Data based on characterization results, tested in production at V_{DD} max., f_{CPU} max. and T_A max.
5. This consumption refers to the Halt period only and not the associated run period which is software dependent.

SUPPLY CURRENT CHARACTERISTICS (Cont'd)

12.4.1 Supply and Clock Managers

The previous 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).

Symbol	Parameter	Conditions	Typ	Max[1)]	Unit
$I_{DD(RES)}$	Supply current of resonator oscillator[2)3)]		See Section 12.5.3 on page 227		µA
$I_{DD(PLL)}$	PLL supply current	V_{DD} = 5V	360		
$I_{DD(LVD)}$	LVD supply current	HALT mode, V_{DD} = 5V	150	300	

Notes:

1. Data based on characterization results, not tested in production.
2. Data based on characterization results done with the external components specified in Section 12.5.3, not tested in production.
3. As the oscillator is based on a current source, the consumption does not depend on the voltage.

12.4.2 On-Chip Peripherals

T_A = 25°C, f_{CPU} = 8 MHz.

Symbol	Parameter	Conditions		Typ	Unit
$I_{DD(TIM)}$	16-bit Timer supply current[1)]		V_{DD} = 5.0V	50	µA
$I_{DD(TIM8)}$	8-bit Timer supply current [1)]				
$I_{DD(ART)}$	ART PWM supply current[2)]			75	
$I_{DD(SPI)}$	SPI supply current[3)]			400	
$I_{DD(SCI)}$	SCI supply current[4)]				
$I_{DD(ADC)}$	ADC supply current when converting[5)]				
$I_{DD(CAN)}$	CAN supply current [6)]			800	

Notes:

1. Data based on a differential I_{DD} measurement between reset configuration (timer counter running at $f_{CPU}/4$) and timer counter stopped (only TIMD bit set). Data valid for one timer.
2. Data based on a differential I_{DD} measurement between reset configuration (timer stopped) and timer counter enabled (only TCE bit set).
3. Data based on a differential I_{DD} measurement between reset configuration (SPI disabled) and a permanent SPI master communication at maximum speed (data sent equal to 55h).This measurement includes the pad toggling consumption.
4. Data based on a differential I_{DD} measurement between SCI low power state (SCID = 1) and a permanent SCI data transmit sequence. Data valid for one SCI.
5. Data based on a differential I_{DD} measurement between reset configuration and continuous A/D conversions.
6. Data based on a differential I_{DD} measurement between reset configuration (CAN disabled) and a permanent CAN data transmit sequence with RX and TX connected together. This measurement include the pad toggling consumption.

12.5 CLOCK AND TIMING CHARACTERISTICS

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

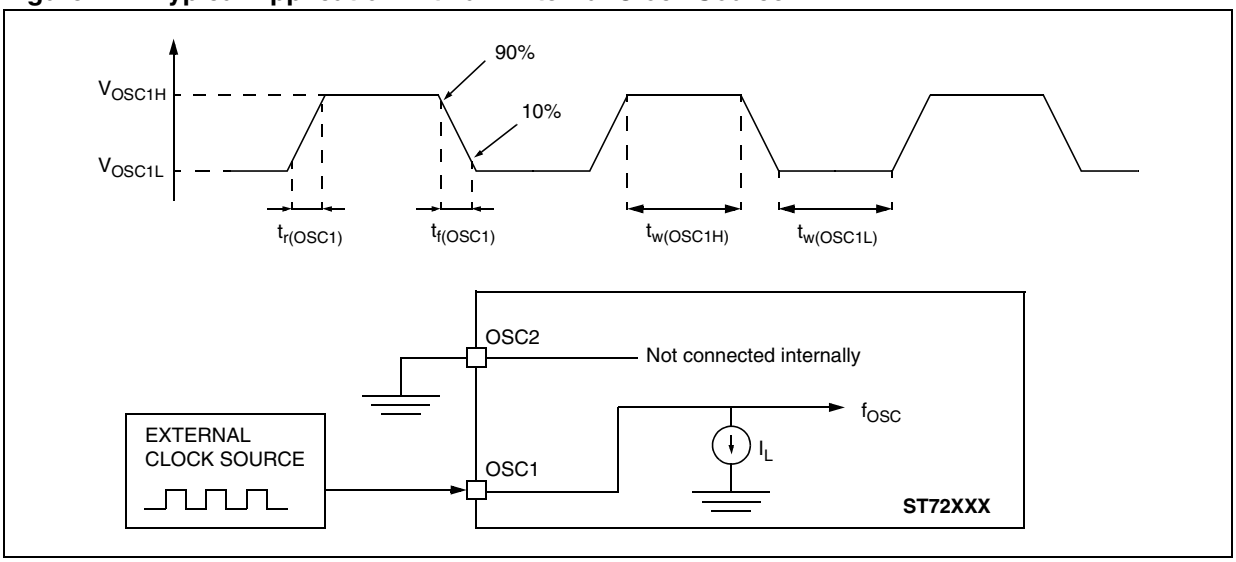
12.5.1 General Timings

Symbol	Parameter	Conditions	Min	Typ[1)]	Max	Unit
$t_{c(INST)}$	Instruction cycle time		2	3	12	t_{CPU}
		f_{CPU} = 8 MHz	250	375	1500	ns
$t_{v(IT)}$	Interrupt reaction time[2)] $t_{v(IT)} = \Delta t_{c(INST)} + 10$		10		22	t_{CPU}
		f_{CPU} = 8 MHz	1.25		2.75	µs

12.5.2 External Clock Source

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{OSC1H}	OSC1 input pin high level voltage	see Figure 121	0.7 x V_{DD}		V_{DD}	V
V_{OSC1L}	OSC1 input pin low level voltage		V_{SS}		0.3 x V_{DD}	
$t_{w(OSC1H)}$ $t_{w(OSC1L)}$	OSC1 high or low time[3)]		25			ns
$t_{r(OSC1)}$ $t_{f(OSC1)}$	OSC1 rise or fall time[3)]				5	
I_L	OSCx Input leakage current	$V_{SS} \le V_{IN} \le V_{DD}$			±1	µA

Figure 121. Typical Application with an External Clock Source



Notes:

1. Data based on typical application software.
2. Time measured between interrupt event and interrupt vector fetch. $\Delta t_{c(INST)}$ is the number of t_{CPU} cycles needed to finish the current instruction execution.
3. Data based on design simulation and/or technology characteristics, not tested in production.

CLOCK AND TIMING CHARACTERISTICS (Cont'd)

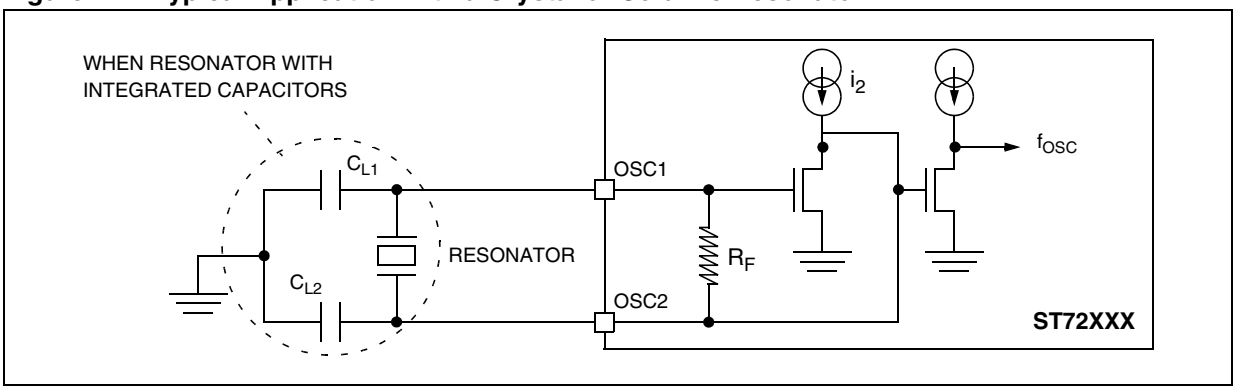
12.5.3 Crystal and Ceramic Resonator Oscillators

The ST7 internal clock can be supplied with four different Crystal/Ceramic resonator oscillators. All the information given in this paragraph is based on characterization results with specified typical external components. In the application, the resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and start-up stabilization time. Refer to the crystal/ceramic resonator manufacturer for more details (frequency, package, accuracy...). [1)2)]

Symbol	Parameter	Conditions		Min	Max	Unit
f_{OSC}	Oscillator Frequency[3)]	LP: Low power oscillator MP: Medium power oscillator MS: Medium speed oscillator HS: High speed oscillator		1 >2 >4 >8	2 4 8 16	MHz
R_F	Feedback resistor			20	40	kΩ
C_{L1} C_{L2}	Recommended load capacitance versus equivalent serial resistance of the crystal or ceramic resonator (R_S)	R_S = 200Ω R_S = 200Ω R_S = 200Ω R_S = 100Ω	LP oscillator MP oscillator MS oscillator HS oscillator	22 22 18 15	56 46 33 33	pF

Symbol	Parameter	Conditions		Typ	Max	Unit
i_2	OSC2 driving current	V_{DD} = 5V V_{IN} = V_{SS}	LP oscillator MP oscillator MS oscillator HS oscillator	80 160 310 610	150 250 460 910	µA

Figure 122. Typical Application with a Crystal or Ceramic Resonator



Notes:

1. Resonator characteristics given by the crystal/ceramic resonator manufacturer.

2. $t_{SU(OSC)}$ is the typical oscillator start-up time measured between V_{DD} = 2.8V and the fetch of the first instruction (with a quick V_{DD} ramp-up from 0 to 5V (< 50µs).

3. The oscillator selection can be optimized in terms of supply current using an high quality resonator with small R_S value. Refer to crystal/ceramic resonator manufacturer for more details.

CLOCK CHARACTERISTICS (Cont'd)

12.5.4 PLL Characteristics

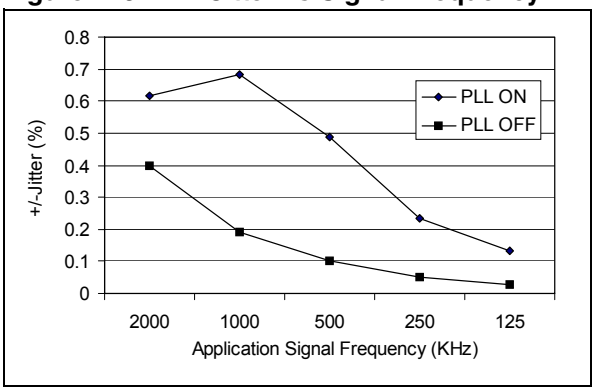
Operating conditions: V_{DD} 3.8 to 5.5V @ T_A 0 to 70°C[1)] or V_{DD} 4.5 to 5.5V @ T_A -40 to 125°C

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{DD(PLL)}$	PLL Voltage Range	T_A = 0 to +70°C	3.8		5.5	
		T_A = -40 to +125°C	4.5			
f_{OSC}	PLL input frequency range		2		4	MHz
$\Delta f_{CPU}/f_{CPU}$	PLL jitter[1)]	f_{OSC} = 4 MHz, V_{DD} = 4.5 to 5.5V		Note 2		%
		f_{OSC} = 2 MHz, V_{DD} = 4.5 to 5.5V				

Notes:

1. Data characterized but not tested.
2. Under characterization

Figure 123. PLL Jitter vs Signal Frequency[1)]



Notes:

1. Measurement conditions: f_{CPU} = 4 MHz, T_A = 25°C

The user must take the PLL jitter into account in the application (for example in serial communication or sampling of high frequency signals). The PLL jitter is a periodic effect, which is integrated over several CPU cycles. Therefore, the longer the period of the application signal, the less it is impacted by the PLL jitter.

Figure 123 shows the PLL jitter integrated on application signals in the range 125 kHz to 2 MHz. At frequencies of less than 125 kHz, the jitter is negligible.

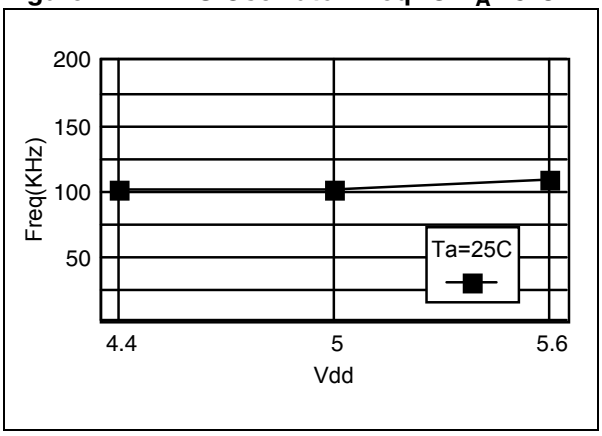
CLOCK CHARACTERISTICS (Cont'd)

12.6 Auto Wakeup from Halt Oscillator (AWU)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{AWU}	AWU oscillator frequency[1)]		50	100	250	kHz
t_{RCSRT}	AWU oscillator startup time			10		µs

1. Data based on characterization results, not tested in production.

Figure 124. AWU Oscillator Freq. @ T_A 25°C



12.7 MEMORY CHARACTERISTICS

12.7.1 RAM and Hardware Registers

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{RM}	Data retention mode[1)]	HALT mode (or RESET)	1.6			V

12.7.2 FLASH Memory

DUAL VOLTAGE HDFLASH MEMORY						
Symbol	**Parameter**	**Conditions**	**Min[2)]**	**Typ**	**Max[2)]**	**Unit**
f_{CPU}	Operating frequency	Read mode	0		8	MHz
		Write / Erase mode	1		8	
V_{PP}	Programming voltage[3)]	$4.5V \leq V_{DD} \leq 5.5V$	11.4		12.6	V
I_{PP}	V_{PP} current[4)5)]	Read (V_{PP} = 12V)			200	µA
		Write / Erase			30	mA
t_{VPP}	Internal V_{PP} stabilization time			$10^{4)}$		µs
t_{RET}	Data retention	T_A = 85°C	40			years
		T_A = 105°C	15			
		T_A = 125°C	7			
N_{RW}	Write erase cycles	T_A = 25°C	100			cycles
T_{PROG} T_{ERASE}	Programming or erasing temperature range		-40	25	85	°C

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). Not tested in production.
2. Data based on characterization results, not tested in production.
3. V_{PP} must be applied only during the programming or erasing operation and not permanently for reliability reasons.
4. Data based on simulation results, not tested in production.
5. In Write / erase mode the I_{DD} supply current consumption is the same as in Run mode (see Section 12.2.2)

12.8 EMC CHARACTERISTICS

Susceptibility tests are performed on a sample basis during product characterization.

12.8.1 Functional EMS (Electro Magnetic Susceptibility)

Based on a simple running application on the product (toggling two LEDs through I/O ports), the product is stressed by two electro magnetic events until a failure occurs (indicated by the LEDs).

- **ESD**: Electro-Static Discharge (positive and negative) is applied on all pins of the device until a functional disturbance occurs. This test conforms with the IEC 1000-4-2 standard.
- **FTB**: A Burst of Fast Transient voltage (positive and negative) is applied to V_{DD} and V_{SS} through a 100pF capacitor, until a functional disturbance occurs. This test conforms with the IEC 1000-4-4 standard.

A device reset allows normal operations to be resumed. The test results are given in the table below based on the EMS levels and classes defined in application note AN1709.

12.8.1.1 Designing hardened software to avoid noise problems

EMC characterization and optimization are performed at component level with a typical application environment and simplified MCU software. It should be noted that good EMC performance is highly dependent on the user application and the software in particular.

Therefore it is recommended that the user applies EMC software optimization and prequalification tests in relation with the EMC level requested for his application.

Software recommendations:

The software flowchart must include the management of runaway conditions such as:

– Corrupted program counter

– Unexpected reset

– Critical Data corruption (control registers...)

Prequalification trials:

Most of the common failures (unexpected reset and program counter corruption) can be reproduced by manually forcing a low state on the RESET pin or the Oscillator pins for 1 second.

To complete these trials, ESD stress can be applied directly on the device, over the range of specification values. When unexpected 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	LQFP64, V_{DD} = 5V, T_A = +25°C, f_{OSC} = 8 MHz, conforms to IEC 1000-4-2	3B
		LQFP44, 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	LQFP64, V_{DD} = 5V, T_A = +25°C, f_{OSC} = 8 MHz, conforms to IEC 1000-4-4	3B
		LQFP44, V_{DD} = 5V, T_A = +25°C, f_{OSC} = 8 MHz, conforms to IEC 1000-4-4	2B

EMC CHARACTERISTICS (Cont'd)

12.8.2 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)]	Flash devices: V_{DD} = 5V, T_A = +25°C, LQFP44 package conforming to SAE J 1752/3	0.1 MHz to 30 MHz	10	13	dBµV
			30 MHz to 130 MHz	12	19	
			130 MHz to 1 GHz	8	14	
			SAE EMI Level	2.5	3	-
		Flash devices: V_{DD} = 5V, T_A = +25°C, LQFP64 package conforming to SAE J 1752/3	0.1 MHz to 30 MHz	31	32	dBµV
			30 MHz to 130 MHz	32	37	
			130 MHz to 1 GHz	11	16	
			SAE EMI Level	3.0	3.5	-
		ROM devices: V_{DD} = 5V, T_A = +25°C, LQFP64 package conforming to SAE J 1752/3	0.1 MHz to 30 MHz	10	18	dBµV
			30 MHz to 130 MHz	15	25	
			130 MHz to 1 GHz	-3	1	
			SAE EMI Level	2.0	2.5	-

Notes:

1. Not tested in production.

EMC CHARACTERISTICS (Cont'd)

12.8.3 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 (see Table 39 and Table 40 below). For more details, refer to application note AN1181.

12.8.3.1 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.

12.8.3.2 Static Latch-Up

- **LU**: Two complementary static tests are required on 6 parts to assess the latch-up performance. A supply overvoltage (applied to each power supply pin) and a current injection (applied to each input, output and configurable I/O pin) are performed on each sample. This test conforms to the EIA/JESD 78 IC latch-up standard.

Table 39. Absolute Maximum Ratings

Symbol	Ratings	Conditions	Maximum value[1)]	Unit
$V_{ESD(HBM)}$	Electro-static discharge voltage (Human Body Model)	T_A = +25°C	2000	V
$V_{ESD(MM)}$	Electro-static discharge voltage (Machine Model)		200	
$V_{ESD(CDM)}$	Electro-static discharge voltage (Charge Device Model)		750 on corner pins, 500 on others	

Notes:

1. Data based on characterization results, not tested in production.

Table 40. Electrical Sensitivities

Symbol	Parameter	Conditions	Class
LU	Static latch-up class	T_A=+125°C conforming to JESD 78	II level A

12.9 I/O PORT PIN CHARACTERISTICS

12.9.1 General Characteristics

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

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{IL}	Input low level voltage[1]	CMOS ports				0.3 x V_{DD}	V
V_{IH}	Input high level voltage[1]			0.7 x V_{DD}			
V_{hys}	Schmitt trigger voltage hysteresis[2]				1		
V_{IL}	Input low level voltage[1]	TTL ports				0.8	
V_{IH}	Input high level voltage[1]			2			
V_{hys}	Schmitt trigger voltage hysteresis[2]				400		mV
$I_{INJ(PIN)}$	Injected Current on PB3	V_{DD} = 5V	Flash devices	0		+4	mA
			ROM devices			±4	
	Injected Current on any other I/O pin					±4	
$\Sigma I_{INJ(PIN)}$[3]	Total injected current (sum of all I/O and control pins)[7]					±25	
I_{lkg}	Input leakage current on robust pins	See "10-BIT ADC CHARACTERISTICS" on page 245					
	Input leakage current[4]	$V_{SS} \le V_{IN} \le V_{DD}$				±1	µA
I_S	Static current consumption[5]	Floating input mode			200		
R_{PU}	Weak pull-up equivalent resistor[6]	$V_{IN} = V_{SS}$	V_{DD} = 5V	50	90	250	kΩ
C_{IO}	I/O pin capacitance				5		pF
$t_{f(IO)out}$	Output high to low level fall time	C_L = 50pF Between 10% and 90%			25		ns
$t_{r(IO)out}$	Output low to high level rise time						
$t_{w(IT)in}$	External interrupt pulse time[7]			1			t_{CPU}

Notes:

1. Data based on characterization results, not tested in production.
2. Hysteresis voltage between Schmitt trigger switching levels. Based on characterization results, not tested.
3. When the current limitation is not possible, the V_{IN} absolute maximum rating must be respected, otherwise refer to $I_{INJ(PIN)}$ specification. A positive injection is induced by $V_{IN} > V_{DD}$ while a negative injection is induced by $V_{IN} < V_{SS}$. Refer to Section 12.2 on page 220 for more details.
4. Leakage could be higher than max. if negative current is injected on adjacent pins.
5. 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 125). Data based on design simulation and/or technology characteristics, not tested in production.
6. The R_{PU} pull-up equivalent resistor is based on a resistive transistor (corresponding I_{PU} current characteristics described in Figure 126).
7. To generate an external interrupt, a minimum pulse width must be applied on an I/O port pin configured as an external interrupt source.

I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 125. Connecting Unused I/O Pins

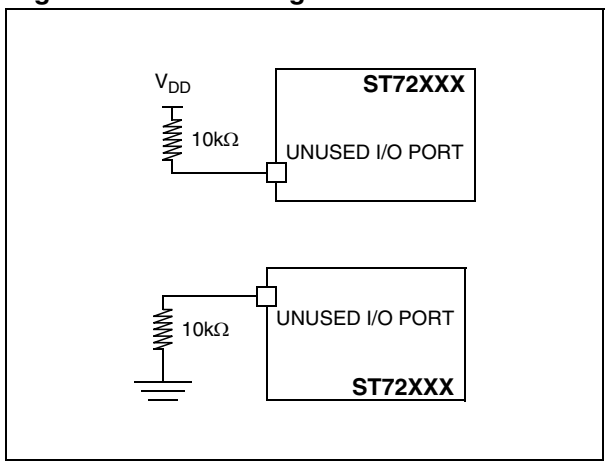


Figure 126. R_{PU} vs V_{DD} with $V_{IN} = V_{SS}$

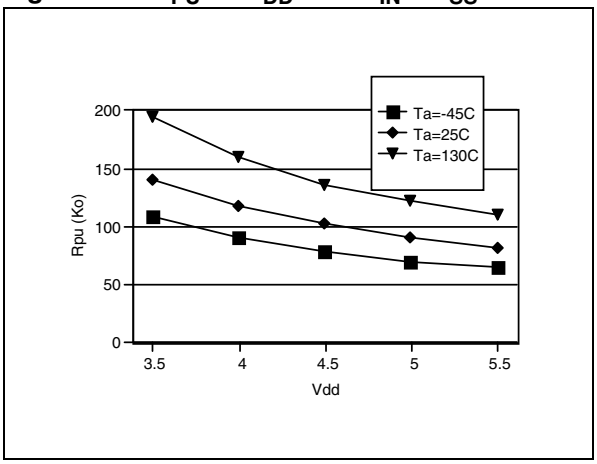
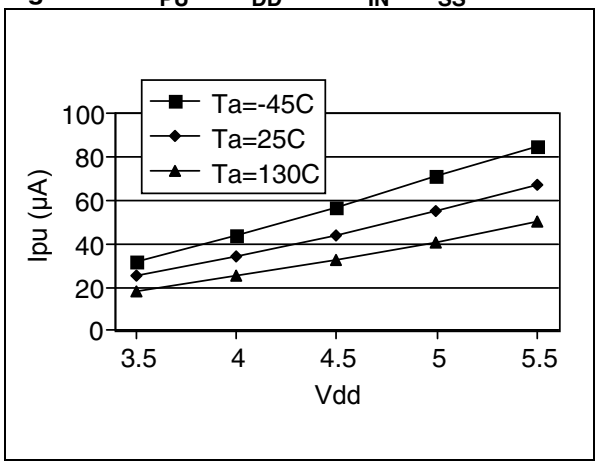


Figure 127. I_{PU} vs V_{DD} with $V_{IN} = V_{SS}$



I/O PORT PIN CHARACTERISTICS (Cont'd)

12.9.2 Output Driving Current

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

Symbol	Parameter	Conditions		Min	Max	Unit
V_{OL} [1]	Output low level voltage for a standard I/O pin when eight pins are sunk at same time (see Figure 128)	V_{DD}=5V	I_{IO}=+5mA		1.2	V
			I_{IO}=+2mA		0.5	
	Output low level voltage for a high sink I/O pin when four pins are sunk at same time (see Figure 129 and Figure 132)		I_{IO}=+20mA, $T_A \le 85°C$ $T_A \ge 85°C$		1.3 1.5	
			I_{IO}=+8mA		0.6	
V_{OH} [2]	Output high level voltage for an I/O pin when four pins are sourced at same time (see Figure 130 and Figure 133)		I_{IO}=-5mA, $T_A \le 85°C$ $T_A \ge 85°C$	V_{DD}-1.4 V_{DD}-1.6		
			I_{IO}=-2mA	V_{DD}-0.7		

Figure 128. Typical V_{OL} at V_{DD} = 5V (Standard)

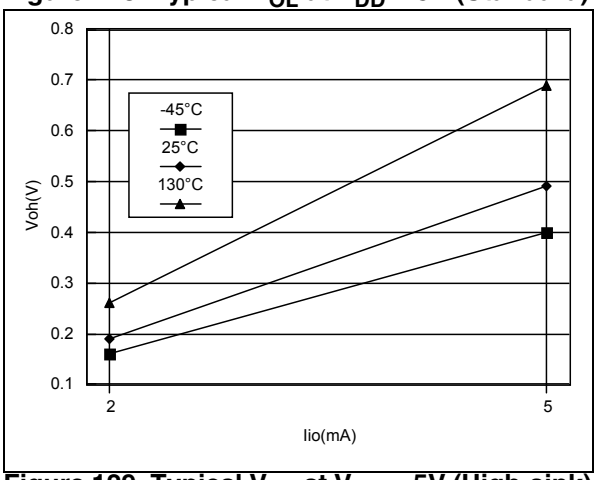


Figure 129. Typical V_{OL} at V_{DD} = 5V (High-sink)

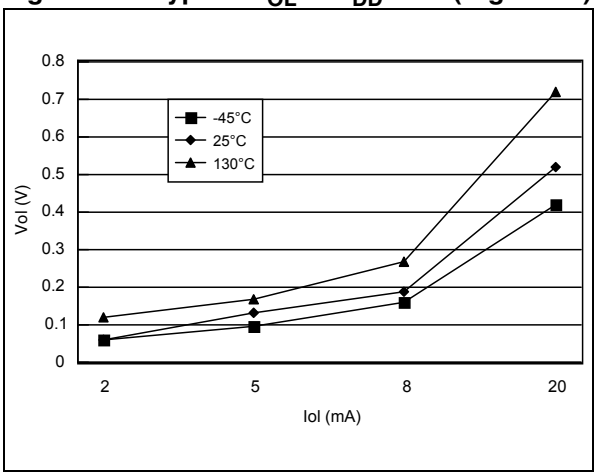
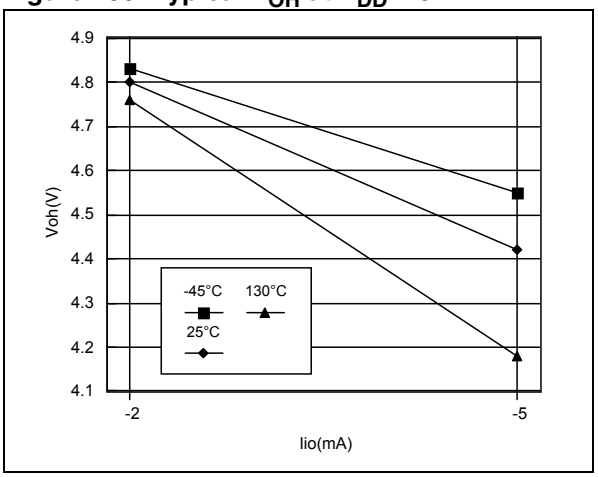


Figure 130. Typical V_{OH} at V_{DD} = 5V



Notes:

1. The I_{IO} current sunk must always respect the absolute maximum rating specified in Section 12.2.2 and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VSS}.
2. The I_{IO} current sourced must always respect the absolute maximum rating specified in Section 12.2.2 and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VDD}. True open drain I/O pins does not have V_{OH}.

I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 131. Typical V_{OL} vs V_{DD} (Standard I/Os)

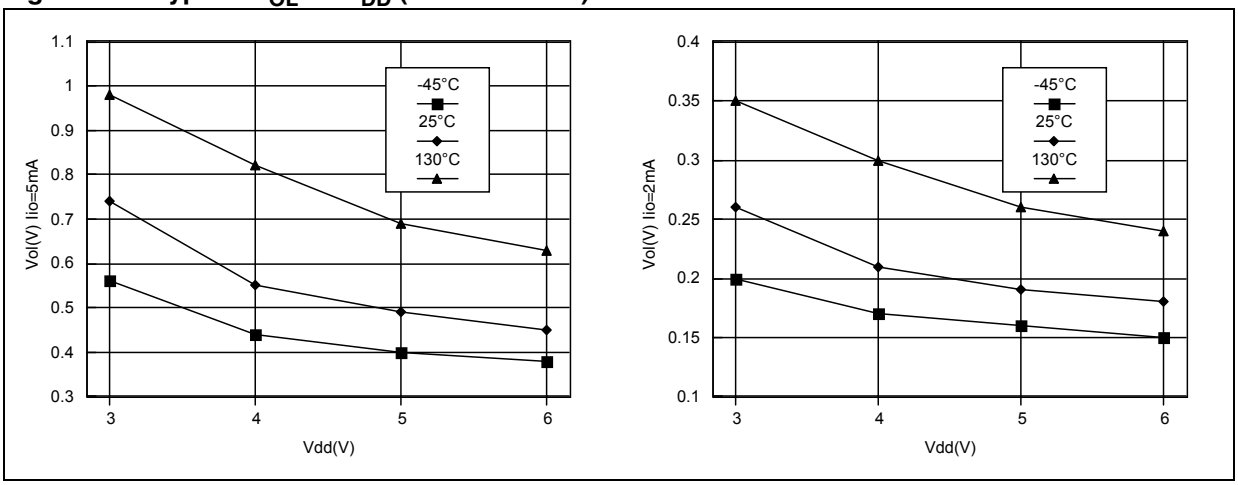
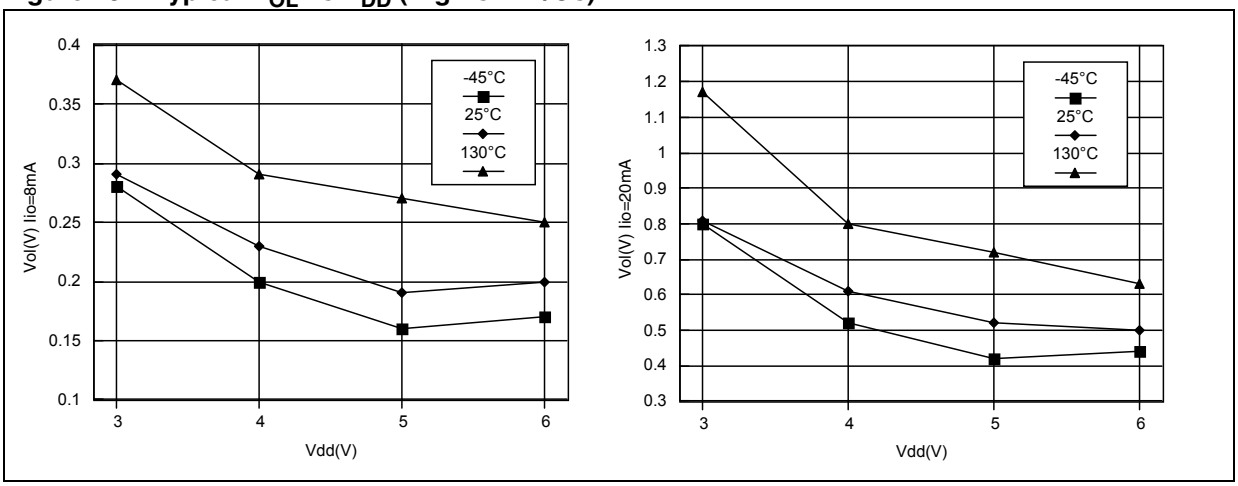
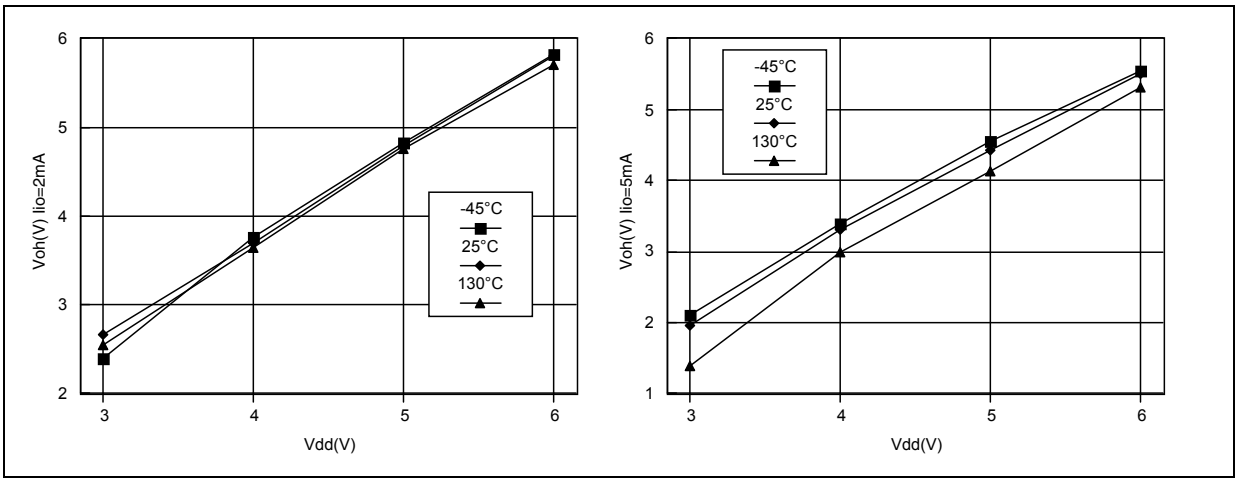


Figure 132. Typical V_{OL} vs V_{DD} (High-sink I/Os)



I/O PORT PIN CHARACTERISTICS (Cont'd)

Figure 133. Typical V_{OH} vs V_{DD}



12.10 CONTROL PIN CHARACTERISTICS

12.10.1 Asynchronous $\overline{\text{RESET}}$ Pin

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

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{IL}	Input low level voltage[1)]					0.3 x V_{DD}	V
V_{IH}	Input high level voltage[1)]			0.7 x V_{DD}			
V_{hys}	Schmitt trigger voltage hysteresis[2)]	V_{DD} = 5V			1.5		
V_{OL}	Output low level voltage[3)]	V_{DD} = 5V	I_{IO} = +5mA		0.68	0.95	
			I_{IO} = +2mA		0.28	0.45	
R_{ON}	Weak pull-up equivalent resistor[4)]	$V_{IN} = V_{SS}$		20	40	80	kΩ
$t_{w(RSTL)out}$	Generated reset pulse duration	Internal reset source			30		µs
$t_{h(RSTL)in}$	External reset pulse hold time[5)]			2.5			
$t_{g(RSTL)in}$	Filtered glitch duration[6)]				200		ns

Notes:

1. Data based on characterization results, not tested in production.
2. Hysteresis voltage between Schmitt trigger switching levels.
3. The I_{IO} current sunk must always respect the absolute maximum rating specified in Section 12.2.2 and the sum of I_{IO} (I/O ports and control pins) must not exceed I_{VSS}.
4. To guarantee the reset of the device, a minimum pulse has to be applied to the $\overline{\text{RESET}}$ pin. All short pulses applied on the RESET pin with a duration below $t_{h(RSTL)in}$ can be ignored.
5. The reset network (the resistor and two capacitors) protects the device against parasitic resets, especially in noisy environments.
6. Data guaranteed by design, not tested in production.

CONTROL PIN CHARACTERISTICS (Cont'd)

Figure 134. $\overline{\text{RESET}}$ Pin Protection When LVD Is Enabled[1)2)]

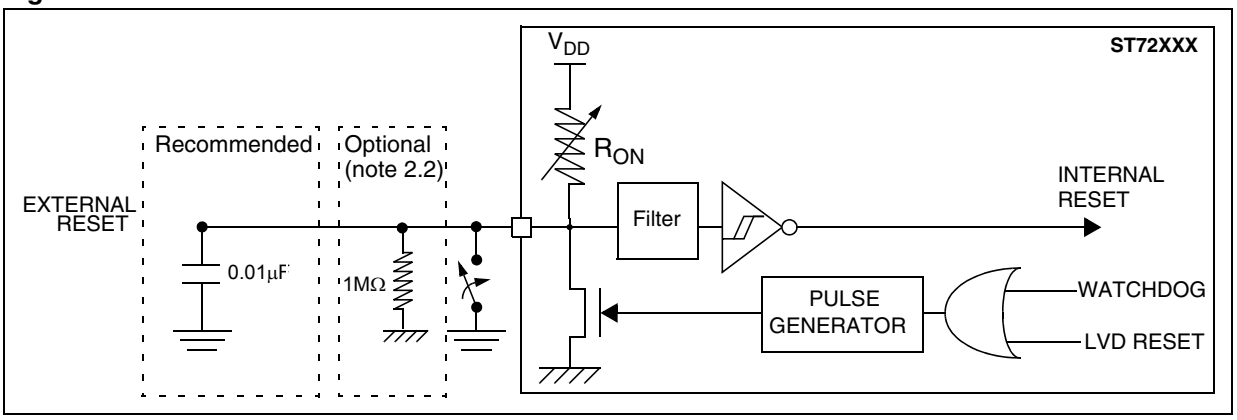
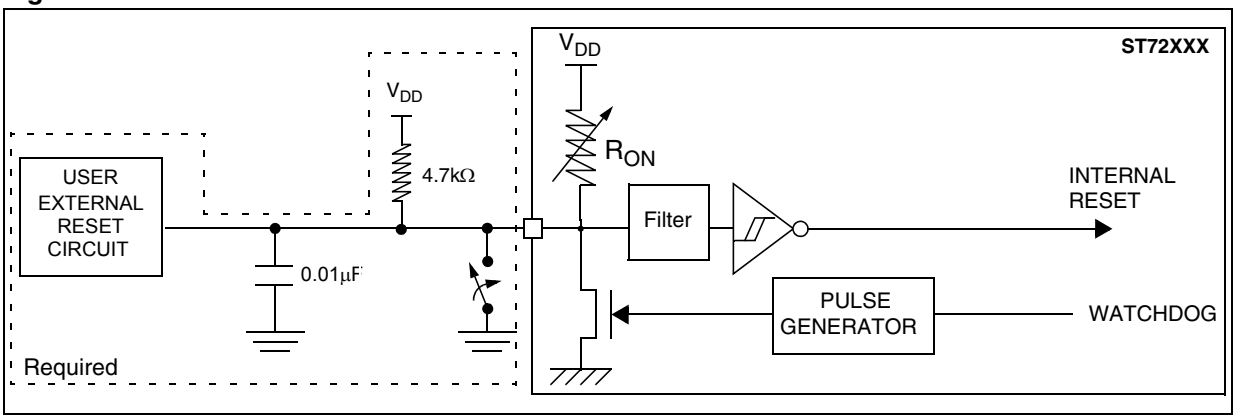


Figure 135. $\overline{\text{RESET}}$ Pin Protection When LVD Is Disabled[1)]



Note 1:

1.1 The reset network protects the device against parasitic resets.

1.2 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).

1.3 Whatever the reset source is (internal or external), the user must ensure that the level on the $\overline{\text{RESET}}$ pin can go below the V_{IL} max. level specified in Section 12.10.1. Otherwise the reset will not be taken into account internally.

1.4 Because the reset circuit is designed to allow the internal RESET to be output in the $\overline{\text{RESET}}$ pin, the user must ensure that the current sunk on the $\overline{\text{RESET}}$ pin (by an external pull-up for example) is less than the absolute maximum value specified for $I_{INJ(RESET)}$ in Section 12.2.2 on page 220.

Note 2:

2.1 When the LVD is enabled, it is mandatory not to connect a pull-up resistor. A 10nF pull-down capacitor is recommended to filter noise on the reset line.

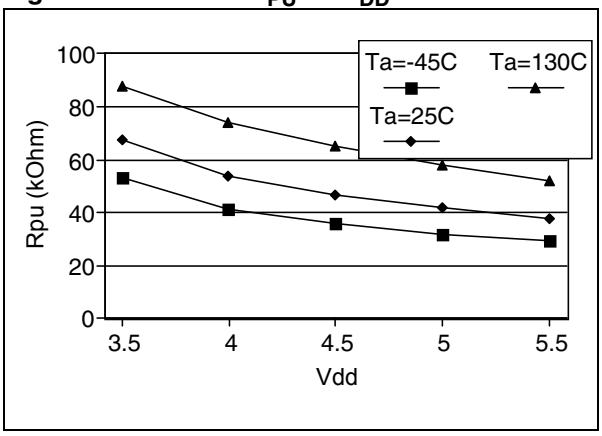
2.2. In case a capacitive power supply is used, it is recommended to connect a1MW pull-down resistor to the $\overline{\text{RESET}}$ pin to discharge any residual voltage induced by this capacitive power supply (this will add 5µA to the power consumption of the MCU).

2.3. Tips when using the LVD:

- 1. Check that all recommendations related to reset circuit have been applied (see notes above)
- 2. Check that the power supply is properly decoupled (100nF + 10µF close to the MCU). Refer to AN1709. If this cannot be done, it is recommended to put a 100nF + 1MW pull-down on the $\overline{\text{RESET}}$ pin.
- 3. The capacitors connected on the RESET pin and also the power supply are key to avoiding 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 $\overline{\text{RESET}}$ pin with a 5µF to 20µF capacitor.

CONTROL PIN CHARACTERISTICS (Cont'd)

Figure 136. RESET R_{PU} vs V_{DD}

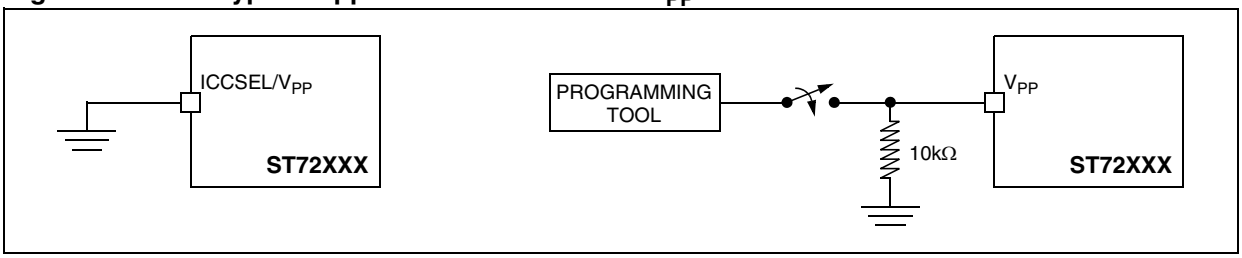


12.10.2 ICCSEL/V_{PP} Pin

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

Symbol	Parameter	Conditions	Min	Max	Unit
V_{IL}	Input low level voltage[1)]		V_{SS}	0.2	V
V_{IH}	Input high level voltage[1)]		V_{DD}-0.1	12.6	
I_L	Input leakage current	$V_{IN} = V_{SS}$		±1	µA

Figure 137. Two Typical Applications with ICCSEL/V_{PP} Pin[2)]



Notes:

1. Data based on design simulation and/or technology characteristics, not tested in production.
2. When ICC mode is not required by the application ICCSEL/V_{PP} pin must be tied to V_{SS}.

12.11 TIMER PERIPHERAL CHARACTERISTICS

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

Refer to I/O port characteristics for more details on the input/output alternate function characteristics (output compare, input capture, external clock, PWM output...).

12.11.1 8-Bit PWM-ART Autoreload Timer

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{res(PWM)}$	PWM resolution time		1			t_{CPU}
		f_{CPU} = 8 MHz	125			ns
f_{EXT}	ART external clock frequency		0		$f_{CPU}/2$	MHz
f_{PWM}	PWM repetition rate					
Res_{PWM}	PWM resolution				8	bit
V_{OS}	PWM/DAC output step voltage	V_{DD} = 5V, Res = 8-bits		20		mV
$t_{COUNTER}$	Timer clock period when internal clock is selected	f_{CPU} = 8 MHz	1		128	t_{CPU}
			0.125		16	µs

12.11.2 8-Bit Timer

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{w(ICAP)in}$	Input capture pulse time		1			t_{CPU}
$t_{res(PWM)}$	PWM resolution time		2			
		f_{CPU} = 8 MHz	250			ns
f_{PWM}	PWM repetition rate		0		$f_{CPU}/4$	MHz
Res_{PWM}	PWM resolution				8	bit
$t_{COUNTER}$	Timer clock period	f_{CPU} = 8 MHz	2		8000	t_{CPU}
			0.250		1000	µs

12.11.3 16-Bit Timer

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{w(ICAP)in}$	Input capture pulse time		1			t_{CPU}
$t_{res(PWM)}$	PWM resolution time		2			
		f_{CPU} = 8 MHz	250			ns
f_{EXT}	Timer external clock frequency		0		$f_{CPU}/4$	MHz
f_{PWM}	PWM repetition rate					
Res_{PWM}	PWM resolution				16	bit
$t_{COUNTER}$	Timer clock period when internal clock is selected	f_{CPU} = 8 MHz	2		8	t_{CPU}
			0.250		1	µs

12.12 COMMUNICATION INTERFACE CHARACTERISTICS

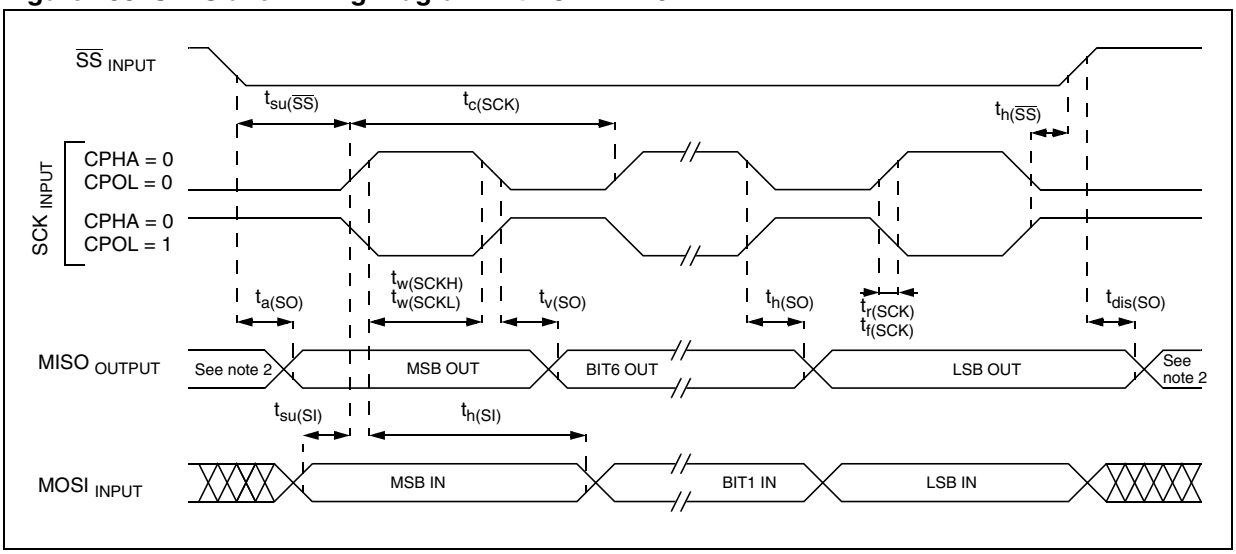
12.12.1 SPI - Serial Peripheral Interface

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	$(4 \times T_{CPU}) + 50$		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)		90	
$t_{h(SO)}$ [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 138. SPI Slave Timing Diagram with CPHA = 0[3]



Notes:

1. Data based on design simulation and/or characterization results, not tested in production.
2. When no communication is on-going the data output line of the SPI (MOSI in master mode, MISO in slave mode) has its alternate function capability released. In this case, the pin status depends on the I/O port configuration.
3. Measurement points are done at CMOS levels: 0.3 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 139. SPI Slave Timing Diagram with CPHA = 1[1)]

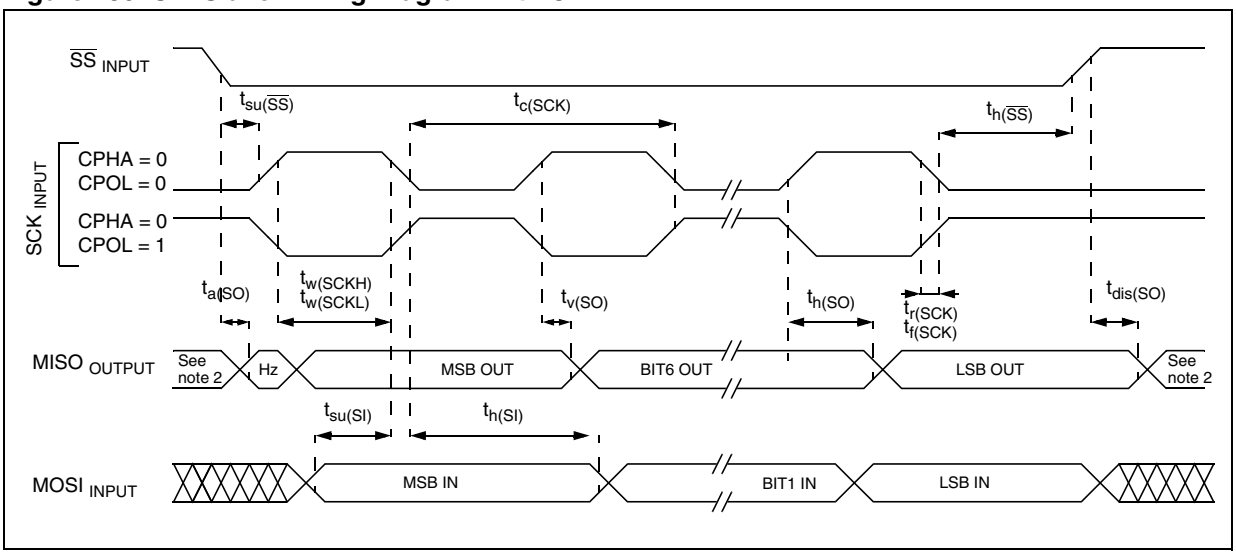
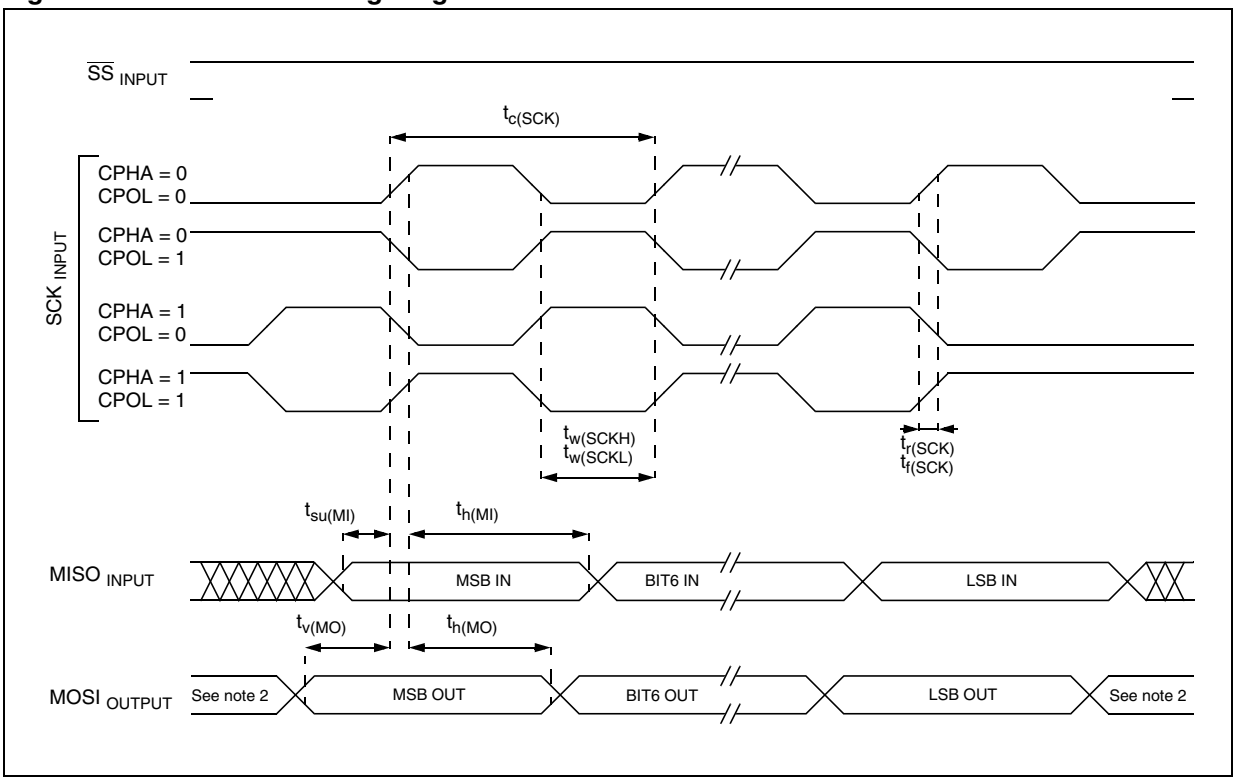


Figure 140. SPI Master Timing Diagram[1)]



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 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.

COMMUNICATIONS INTERFACE CHARACTERISTICS (Cont'd)

12.12.2 CAN - Controller Area Network Interface

Subject to general operating condition for V_{DD}, f_{O-SC}, and T_A unless otherwise specified.
Refer to I/O port characteristics for more details on the input/output alternate function characteristics (CANTX and CANRX).

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$t_{p(RX:TX)}$	CAN controller propagation time[1)]				60	ns

1. Data based on characterization results, not tested in production.

12.13 10-BIT ADC CHARACTERISTICS

Subject to general operating conditions for V_{DD}, f_{CPU}, and T_A unless otherwise specified.

Symbol	Parameter	Conditions	Min[1)]	Typ	Max[1)]	Unit
f_{ADC}	ADC clock frequency		0.4		4	MHz
V_{AIN}	Conversion voltage range[2)]		V_{SSA}		V_{DDA}	V
R_{AIN}	External input impedance				see Figure 141 and Figure 142	kΩ
C_{AIN}	External capacitor on analog input					pF
f_{AIN}	Variation frequency of analog input signal					Hz
I_{lkg}	Negative input leakage current on robust analog pins (refer to Table 2 on page 8)	$V_{IN} < V_{SS}$, $\lvert I_{IN} \rvert < 400\mu A$ on adjacent robust analog pin		5	6	µA
C_{ADC}	Internal sample and hold capacitor			6		pF
t_{CONV}	Conversion time	f_{ADC} = 4 MHz		3.5		µs
				14		$1/f_{ADC}$
I_{ADC}	Analog part	Sunk on V_{DDA}[2)]			3.6	mA
	Digital part	Sunk on V_{DD}			0.2	

Notes:
1. Data based on characterization results, not tested in production.
2. When V_{DDA} and V_{SSA} pins are not available on the pinout, the ADC refers to V_{DD} and V_{SS}.

ADC CHARACTERISTICS (Cont'd)

Figure 141. R_{AIN} Max vs f_{ADC} with C_{AIN} = 0pF[1)2)]

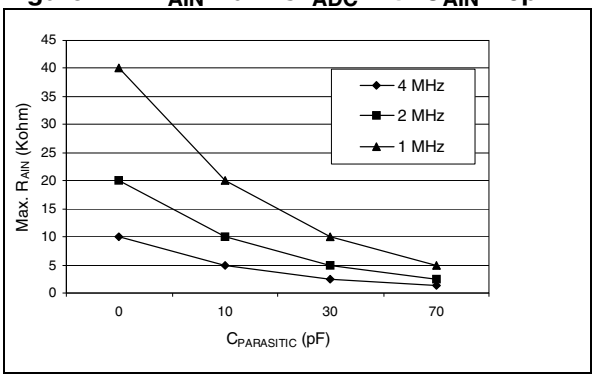


Figure 142. Recommended C_{AIN}/R_{AIN} Values[3)]

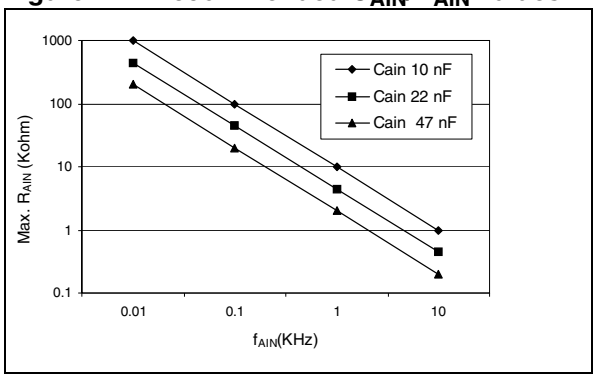
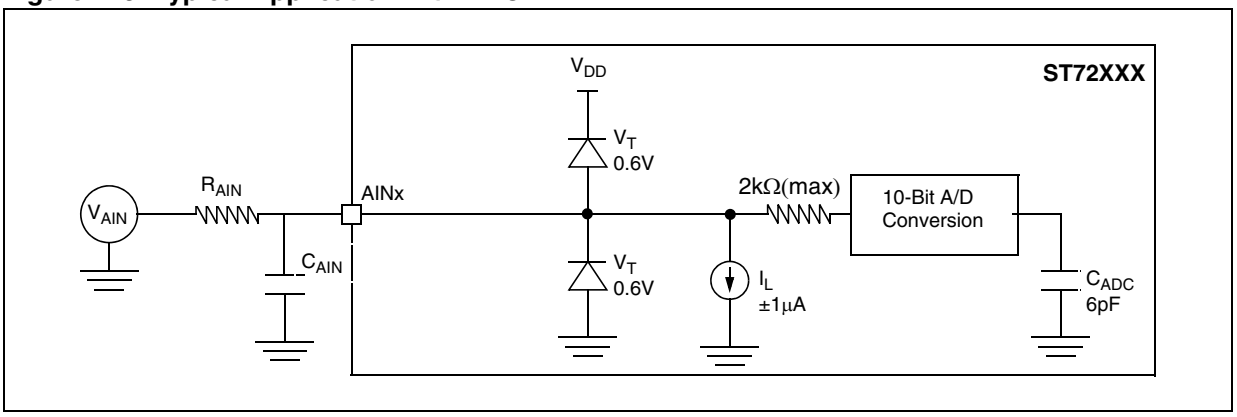


Figure 143. Typical Application with ADC



Notes:

1. $C_{PARASITIC}$ represents the capacitance of the PCB (dependent on soldering and PCB layout quality) plus the pad capacitance (3pF). A high $C_{PARASITIC}$ value will downgrade conversion accuracy. To remedy this, f_{ADC} should be reduced.

2. Any added external serial resistor will downgrade the ADC accuracy (especially for resistance greater than 10kΩ). Data based on characterization results, not tested in production.

3. This graph shows that depending on the input signal variation (f_{AIN}), C_{AIN} can be increased for stabilization time and reduced to allow the use of a larger serial resistor (R_{AIN}). It is valid for all f_{ADC} frequencies ≤ 4 MHz.

ADC CHARACTERISTICS (Cont'd)

12.13.0.1 Analog Power Supply and Reference Pins

Depending on the MCU pin count, the package may feature separate V_{DDA} and V_{SSA} analog power supply pins. These pins supply power to the A/D converter cell and function as the high and low reference voltages for the conversion. In smaller packages V_{DDA} and V_{SSA} pins are not available and the analog supply and reference pads are internally bonded to the V_{DD} and V_{SS} pins.

Separation of the digital and analog power pins allow board designers to improve A/D performance. Conversion accuracy can be impacted by voltage drops and noise in the event of heavily loaded or badly decoupled power supply lines (see Section 12.13.0.2 "General PCB Design Guidelines").

12.13.0.2 General PCB Design Guidelines

To obtain best results, some general design and layout rules should be followed when designing the application PCB to shield the noise-sensitive, analog physical interface from noise-generating CMOS logic signals.

– Use separate digital and analog planes. The analog ground plane should be connected to the digital ground plane via a single point on the PCB.

– Filter power to the analog power planes. It is recommended to connect capacitors, with good high frequency characteristics, between the power and ground lines, placing 0.1µF and optionally, if needed 10pF capacitors as close as possible to the ST7 power supply pins and a 1 to 10µF capacitor close to the power source (see Figure 144).

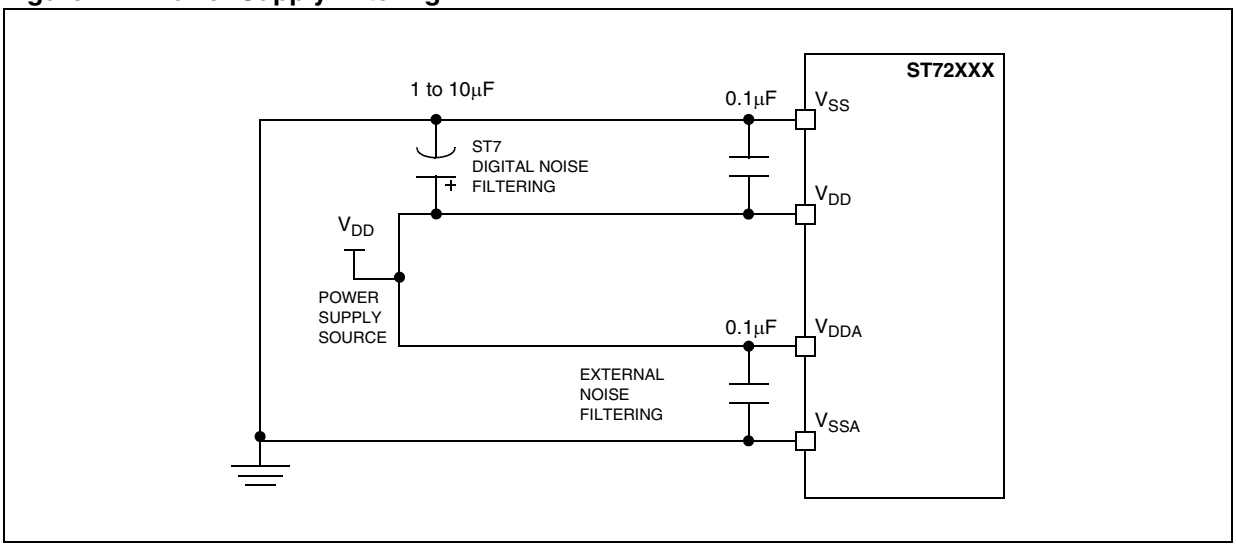
– The analog and digital power supplies should be connected in a star network. Do not use a resistor, as V_{DDA} is used as a reference voltage by the A/D converter and any resistance would cause a voltage drop and a loss of accuracy.

– Properly place components and route the signal traces on the PCB to shield the analog inputs. Analog signals paths should run over the analog ground plane and be as short as possible. Isolate analog signals from digital signals that may switch while the analog inputs are being sampled by the A/D converter. Do not toggle digital outputs on the same I/O port as the A/D input being converted.

12.13.0.3 Software Filtering of Spurious Conversion Results

For EMC performance reasons, it is recommended to filter A/D conversion outliers using software filtering techniques.

Figure 144. Power Supply Filtering

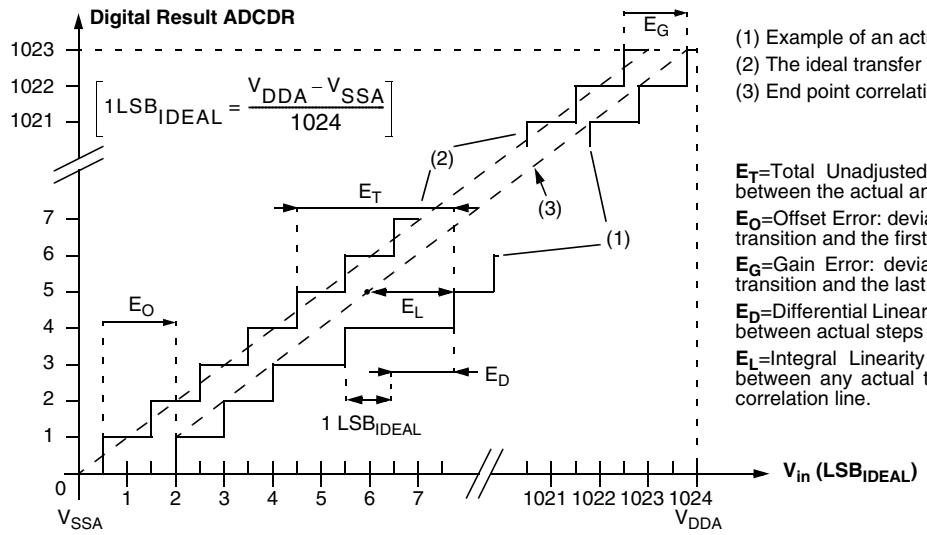


ADC CHARACTERISTICS (Cont'd)

ADC Accuracy with f_{CPU} = 8 MHz, f_{ADC} = 4 MHz R_{AIN} < 10kΩ, V_{DD} = 5V

Symbol	Parameter	Conditions	Typ	Max	Unit
$\|E_T\|$	Total unadjusted error[1)]		3.2	5	LSB
$\|E_O\|$	Offset error[1)]		1	4	
$\|E_G\|$	Gain Error[1)]		0.7		
$\|E_D\|$	Differential linearity error[1)]		1.5	2.3	
$\|E_L\|$	Integral linearity error[1)]		1.2	3.6	

Figure 145. ADC Accuracy Characteristics



(1) Example of an actual transfer curve
(2) The ideal transfer curve
(3) End point correlation line

E_T=Total Unadjusted Error: maximum deviation between the actual and the ideal transfer curves.

E_O=Offset Error: deviation between the first actual transition and the first ideal one.

E_G=Gain Error: deviation between the last ideal transition and the last actual one.

E_D=Differential Linearity Error: maximum deviation between actual steps and the ideal one.

E_L=Integral Linearity Error: maximum deviation between any actual transition and the end point correlation line.

Notes:

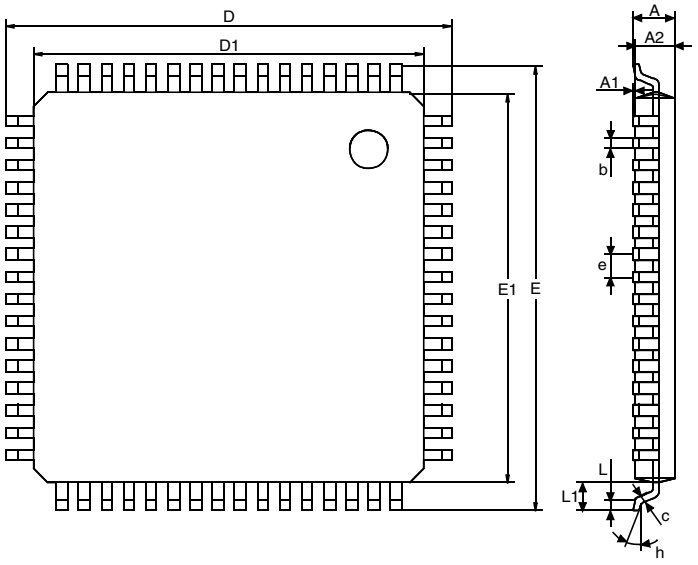
1. Data based on characterization results, not tested in production. ADC Accuracy vs. Negative Injection Current: Injecting negative current on any of the standard (non-robust) analog input pins should be avoided as this significantly reduces the accuracy of the conversion being performed on another analog input. It is recommended to add a Schottky diode (pin to ground) to standard analog pins which may potentially inject negative current. The effect of negative injection current on robust pins is specified in Section 12.9.

Any positive injection current within the limits specified for $I_{INJ(PIN)}$ and $\Sigma I_{INJ(PIN)}$ in Section 12.9 does not affect the ADC accuracy.

13 PACKAGE CHARACTERISTICS

13.1 PACKAGE MECHANICAL DATA

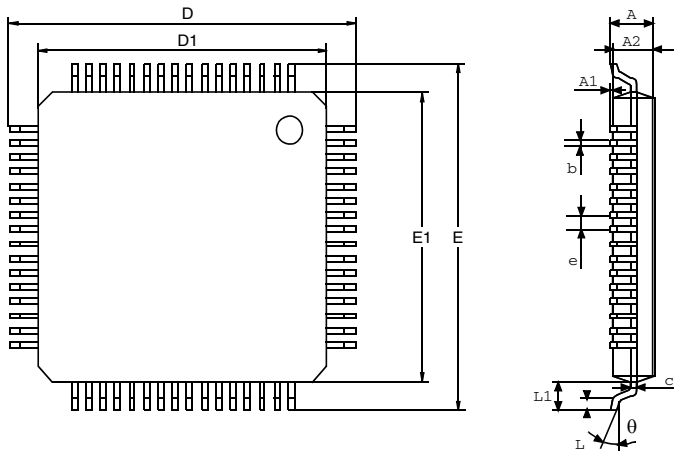
Figure 146. 64-Pin Low Profile Quad Flat Package (14x14)



Dim.	mm			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A			1.60			0.0630
A1	0.05		0.15	0.0020		0.0059
A2	1.35	1.40	1.45	0.0531	0.0551	0.0571
b	0.30	0.37	0.45	0.0118	0.0146	0.0177
c	0.09		0.20	0.0035		0.0079
D		16.00			0.6299	
D1		14.00			0.5512	
E		16.00			0.6299	
E1		14.00			0.5512	
e		0.80			0.0315	
θ	0°	3.5°	7°	0°	3.5°	7°
L	0.45	0.60	0.75	0.0177	0.0236	0.0295
L1		1.00			0.0394	
	Number of Pins					
N	64					

Note 1. Values in inches are converted from mm and rounded to 4 decimal digits.

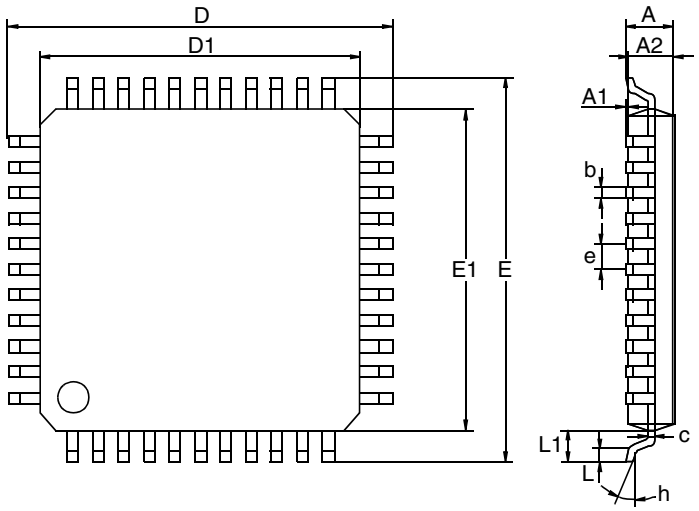
Figure 147. 64-Pin Low Profile Quad Flat Package (10 x10)



Dim.	mm			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A			1.60			0.0630
A1	0.05		0.15	0.0020		0.0059
A2	1.35	1.40	1.45	0.0531	0.0551	0.0571
b	0.17	0.22	0.27	0.0067	0.0087	0.0106
c	0.09		0.20	0.0035		0.0079
D		12.00			0.4724	
D1		10.00			0.3937	
E		12.00			0.4724	
E1		10.00			0.3937	
e		0.50			0.0197	
θ	0°	3.5°	7°	0°	3.5°	7°
L	0.45	0.60	0.75	0.0177	0.0236	0.0295
L1		1.00			0.0394	
	Number of Pins					
N	64					

Note 1. Values in inches are converted from mm and rounded to 4 decimal digits.

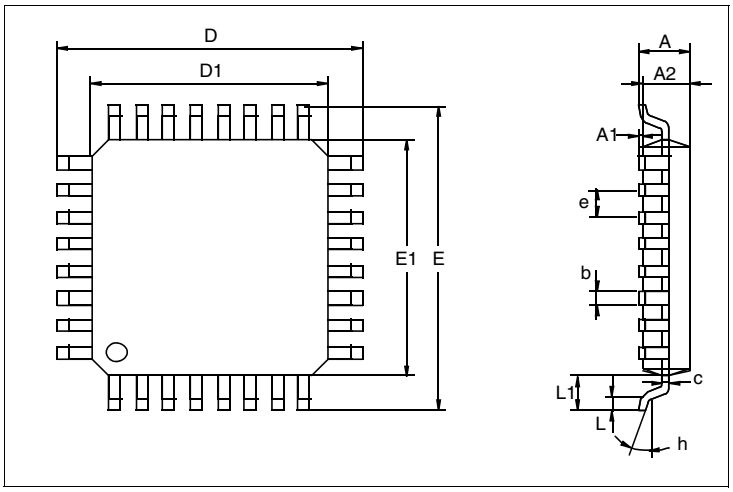
Figure 148. 44-Pin Low Profile Quad Flat Package (10x10)



Dim.	mm			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A			1.60			0.0630
A1	0.05		0.15	0.0020		0.0059
A2	1.35	1.40	1.45	0.0531	0.0551	0.0571
b	0.30	0.37	0.45	0.0118	0.0146	0.0177
C	0.09		0.20	0.0035		0.0079
D		12.00			0.4724	
D1		10.00			0.3937	
E		12.00			0.4724	
E1		10.00			0.3937	
e		0.80			0.0315	
θ	0°	3.5°	7°	0°	3.5°	7°
L	0.45	0.60	0.75	0.0177	0.0236	0.0295
L1		1.00			0.0394	
	Number of Pins					
N	44					

Note 1. Values in inches are converted from mm and rounded to 4 decimal digits.

Figure 149. 32-Pin Low Profile Quad Flat Package (7x7)



Dim.	mm			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A			1.60			0.0630
A1	0.05		0.15	0.0020		0.0059
A2	1.35	1.40	1.45	0.0531	0.0551	0.0571
b	0.30	0.37	0.45	0.0118	0.0146	0.0177
C	0.09		0.20	0.0035		0.0079
D		9.00			0.3543	
D1		7.00			0.2756	
E		9.00			0.3543	
E1		7.00			0.2756	
e		0.80			0.0315	
θ	0°	3.5°	7°	0°	3.5°	7°
L	0.45	0.60	0.75	0.0177	0.0236	0.0295
L1		1.00			0.0394	
	Number of Pins					
N	32					

Note 1. Values in inches are converted from mm and rounded to 4 decimal digits.

13.2 THERMAL CHARACTERISTICS

Symbol	Ratings	Value	Unit
R_{thJA}	Package thermal resistance (junction to ambient) LQFP64 LQFP44 LQFP32	 60 52 70	°C/W
P_D	Power dissipation[1]	500	mW
T_{Jmax}	Maximum junction temperature[2]	150	°C

Notes:

1. 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.

2. The maximum chip-junction temperature is based on technology characteristics.

13.3 SOLDERING AND GLUEABILITY INFORMATION

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 DEVICE CONFIGURATION AND ORDERING INFORMATION

Each device is available for production in user programmable versions (FLASH) as well as in factory coded versions (ROM/FASTROM).

ST72561 devices are ROM versions. ST72P561 devices are Factory Advanced Service Technique ROM (FASTROM) versions: They are factory-programmed HDFlash devices.

ST72F561 FLASH devices are shipped to customers with a default content (FFh), while ROM factory coded parts contain the code supplied by the customer. This implies that FLASH devices have to be configured by the customer using the Option Bytes while the ROM devices are factory-configured.

14.1 FLASH OPTION BYTES

The option bytes allows the hardware configuration of the microcontroller to be selected. They have no address in the memory map and can be accessed only in programming mode (for example using a standard ST7 programming tool). The default content of the FLASH is fixed to FFh. To program directly the FLASH devices using ICP, FLASH devices are shipped to customers with a reserved internal clock source enabled. In masked ROM devices, the option bytes are fixed in hardware by the ROM code (see option list).

OPTION BYTE 0

OPT7 = **WDGHALT** *Watchdog reset on HALT*
This option bit determines if a RESET is generated when entering HALT mode while the Watchdog is active.
0: No Reset generation when entering Halt mode
1: Reset generation when entering Halt mode

OPT6 = **WDGSW** *Hardware or software watchdog*
This option bit selects the watchdog type.
0: Hardware (watchdog always enabled)
1: Software (watchdog to be enabled by software)

OPT5 = Reserved, must be kept at default value.

OPT4 = **LVD** *Voltage detection*
This option bit enables the voltage detection block (LVD).

Selected Low Voltage Detector	VD
LVD Off	1
LVD On	0

OPT3 = **PLL OFF** *PLL activation*
This option bit activates the PLL which allows multiplication by two of the main input clock frequency. The PLL is guaranteed only with an input frequency between 2 and 4 MHz.
0: PLL x2 enabled
1: PLL x2 disabled
Caution: The PLL can be enabled only if the “OSC RANGE” (OPT11:10) bits are configured to “MP - 2~4 MHz”. Otherwise, the device functionality is not guaranteed.

	STATIC OPTION BYTE 0								STATIC OPTION BYTE 1							
	7							0	7							0
	WDG		Reserved	LVD	PLLOFF	PKG		FMP_R	AFI_MAP		OSCTYPE		OSCRANGE		Reserved	RSTC
	HALT	SW				1	0		1	0	1	0	1	0		
Default(*)	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1

(*): Option bit values programmed by ST

FLASH OPTION BYTES (Cont'd)

OPT2:1 = **PKG[1:0]** *Package selection*
These option bits select the device package.

Selected Package	PKG	
	1	**0**
LQFP 64	1	x
LQFP 44	0	1
LQFP 32	0	0

Note: Pads that are not bonded to external pins are in input pull-up configuration when the package selection option bits have been properly programmed. The configuration of these pads must be kept in reset state to avoid added current consumption.

OPT0 = **FMP_R** *Flash memory read-out protection*
Read-out 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 user memory to be erased first, and the device can be reprogrammed. Refer to Section 4.3.1 and the *ST7 Flash Programming Reference Manual* for more details.
0: Read-out protection enabled
1: Read-out protection disabled

OPTION BYTE 1

OPT7:6 = **AFI_MAP[1:0]** *AFI Mapping*
These option bits allow the mapping of some of the Alternate Functions to be changed.

AFI Mapping 1	AFI_MAP(1)
T16_OCMP1 on PD3 T16_OCMP2 on PD5 T16_ICAP1 on PD4 LINSCI2_SCK not available LINSCI2_TDO not available LINSCI2_RDI not available	0
T16_OCMP1 on PB6 T16_OCMP2 on PB7 T16_ICAP1 on PC0 LINSCI2_SCK on PD3 LINSCI2_TDO on PD5 LINSCI2_RDI on PD4	1

AFI Mapping 0	AFI_MAP(0)
T16_ICAP2 is mapped on PD1	0
T16_ICAP2 is mapped on PC1	1

OPT5:4 = **OSCTYPE[1:0]** *Oscillator Type*
These option bits select the ST7 main clock source type.

Clock Source	OSCTYPE	
	1	**0**
Resonator Oscillator	0	0
Reserved	0	1
Reserved internal clock source (used only in ICC mode)	1	0
External Source	1	1

OPT3:2 = **OSCRANGE[1:0]** *Oscillator range*
If the resonator oscillator type is selected, these option bits select the resonator oscillator. This selection corresponds to the frequency range of the resonator used. If external source is selected with the OSCTYPE option, then the OSCRANGE option must be selected with the corresponding range.

Typ. Freq. Range		OSCRANGE	
		1	**0**
LP	1~2 MHz	0	0
MP	2~4 MHz	0	1
MS	4~8 MHz	1	0
HS	8~16 MHz	1	1

OPT1 = Reserved

OPT0 = **RSTC** *RESET clock cycle selection*
This option bit selects the number of CPU cycles inserted during the RESET phase and when exiting HALT mode. For resonator oscillators, it is advised to select 4096 due to the long crystal stabilization time.
0: Reset phase with 4096 CPU cycles
1: Reset phase with 256 CPU cycles

DEVICE CONFIGURATION AND ORDER INFORMATION (cont'd)

14.2 TRANSFER OF CUSTOMER CODE

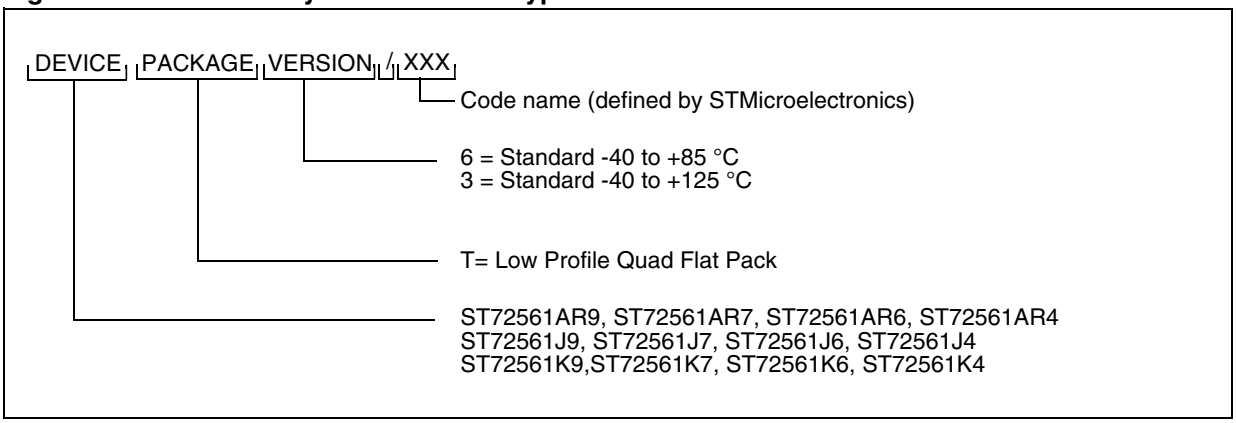
Customer code is made up of the ROM/FASTROM contents and the list of the selected options (if any). The ROM/FASTROM contents are to be sent on diskette, or by electronic means, with the S19 hexadecimal file generated by the development tool. All unused bytes must be set to FFh.

The selected options are communicated to STMicroelectronics using the correctly completed OPTION LIST appended.

Refer to application note AN1635 for information on the counter listing returned by ST after code has been transferred.

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

Figure 150. ROM Factory Coded Device Types



TRANSFER OF CUSTOMER CODE (cont'd)

Figure 151. FASTROM Factory Coded Device Types

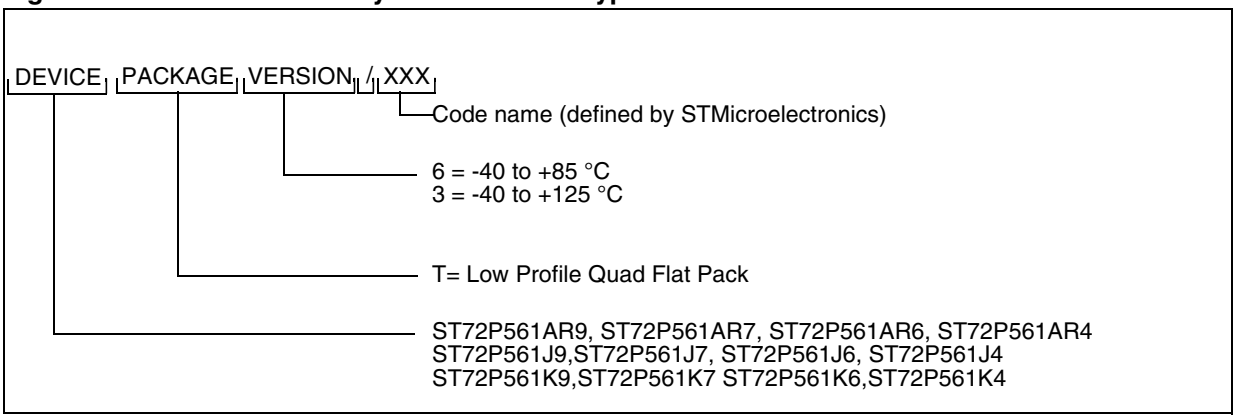
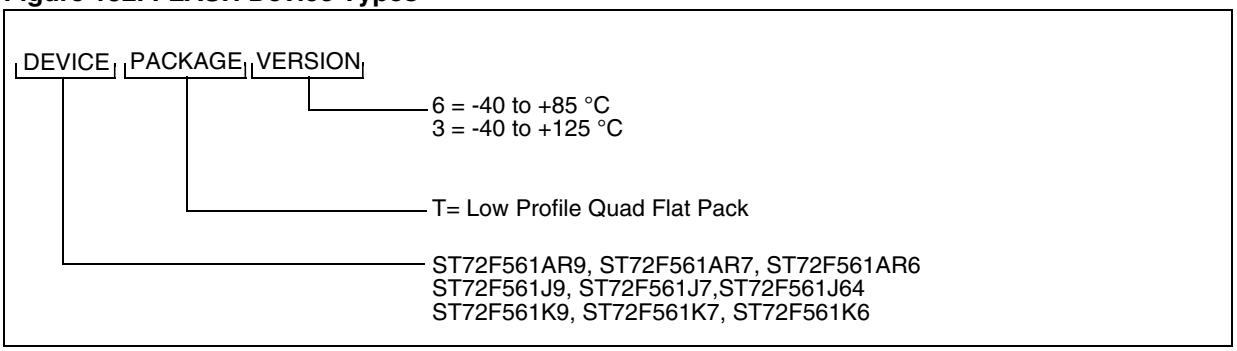


Figure 152. FLASH Device Types



TRANSFER OF CUSTOMER CODE (Cont'd)

ST72561 MICROCONTROLLER OPTION LIST
(Last update: September 2006)

Customer . . . . . .
Address . . . . . .
. . . . . .
Contact . . . . . .
Phone No . . . . . .
Reference/ROM Code* . . . . . .

*The ROM/FASTROM code name is assigned by STMicroelectronics.
ROM/FASTROM code must be sent in .S19 format. .Hex extension cannot be processed.

Device Type/Memory Size/Package (check only one option)

ROM:	Package	60K	48K	32K	16K
	LQFP64 10x10:	[ ] ST72561AR9	[ ] ST72561AR7	[ ] ST72561AR6	[ ] ST72561AR4
	LQFP44:	[ ] ST72561J9	[ ] ST72561J7	[ ] ST72561J6	[ ] ST72561J4
	LQFP32:	[ ] ST72561K9	[ ] ST72561K7	[ ] ST72561K6	[ ] ST72561K4

FASTROM:	Package	60K	48K	32K	16K
	LQFP64 10x10:	[ ] ST72P561AR9	[ ] ST72P561AR7	[ ] ST72P561AR6	[ ] ST72P561AR4
	LQFP44:	[ ] ST72P561J9	[ ] ST72P561J7	[ ] ST72P561J6	[ ] ST72P561J4
	LQFP32:	[ ] ST72P561K9	[ ] ST72P561K7	[ ] ST72P561K6	[ ] ST72P561K4

Conditioning: [ ] Tray [ ] Tape & Reel
Special Marking: [ ] No [ ] Yes "_ _ _ _ _ _ _ _ _ _ " (10 char. max)
Authorized characters are letters, digits, '.', '-', '/' and spaces only.

Temp. Range. Please refer to datasheet for specific sales conditions:

-	Temp. Range
[ ]	-40°C to +85°C
[ ]	-40°C to +125°C

Clock Source Selection: [ ] Resonator: [ ] External Source
Oscillator/External source range:
[ ] LP: Low power (1 to 2 MHz)
[ ] MP: Medium power (2 to 4 MHz)
[ ] MS: Medium speed (4 to 8 MHz)
[ ] HS: High speed (8 to 16 MHz)
LVD [ ] Disabled [ ] Enabled
PLL[1] [ ] Disabled [ ] Enabled
Watchdog Selection [ ] Software Activation [ ] Hardware Activation
Watchdog Reset on Halt [ ] Reset [ ] No Reset

Read-out Protection [ ] Disabled [ ] Enabled

Reset Delay [ ] 256 Cycles [ ] 4096 Cycles

LINSCI2 Mapping [ ] Not available (AFIMAP[1] = 0) [ ] Mapped (AFIMAP[1] = 1)
T16_ICAP2 Mapping [ ] On PD1 (AFIMAP[0] = 0) [ ] On PC1 (AFIMAP[0] = 1)

Comments:
Supply Operating Range in the application: . . . . . .

Notes . . . . . .
Signature . . . . . .
Date . . . . . .

[1] If PLL is enabled, medium power (2 to 4 MHz range) has to be selected (MP)
Please download the latest version of this option list from:
http://www.st.com

15 DEVELOPMENT TOOLS

Full details of tools available for the ST7 from third party manufacturers can be obtained from the STMicroelectronics Internet site:

➟ www.st.com/mcu

Tools from iSystem and Hitex include C compliers, emulators and gang programmers.

Note: Before designing the board layout, it is recommended to check the overall dimensions of the socket as they may be greater than the dimensions of the device.

For footprint and other mechanical information about these sockets and adapters, refer to the manufacturer's datasheet.

ST Programming Tools

- ST7MDT25-EPB: For in-socket or ICC programming
- ST7-STICK: For ICC programming

16 IMPORTANT NOTES

16.1 ALL DEVICES

16.1.1 RESET Pin Protection with LVD Enabled

As mentioned in note 2 below Figure 134 on page 240, 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.

16.1.2 Clearing Active Interrupts Outside Interrupt Routine

When an active interrupt request occurs at the same time as the related flag or interrupt mask is being cleared, the CC register may be corrupted.

Concurrent interrupt context

The symptom does not occur when the interrupts are handled normally, that is, when:

– The interrupt request is cleared (flag reset or interrupt mask) within its own interrupt routine

– The interrupt request is cleared (flag reset or interrupt mask) within any interrupt routine

– The interrupt request is cleared (flag reset or interrupt mask) in any part of the code while this interrupt is disabled

If these conditions are not met, the symptom can be avoided by implementing the following sequence:

Perform SIM and RIM operation before and after resetting an active interrupt request

Example:

```
SIM
reset flag or interrupt mask
RIM
```

Nested interrupt context

The symptom does not occur when the interrupts are handled normally, that is, when:

– The interrupt request is cleared (flag reset or interrupt mask) within its own interrupt routine

– The interrupt request is cleared (flag reset or interrupt mask) within any interrupt routine with higher or identical priority level

– The interrupt request is cleared (flag reset or interrupt mask) in any part of the code while this interrupt is disabled

If these conditions are not met, the symptom can be avoided by implementing the following sequence:

```
PUSH CC
SIM
reset flag or interrupt mask
POP CC
```

16.1.3 External Interrupt Missed

To avoid any risk of generating a parasitic interrupt, the edge detector is automatically disabled for one clock cycle during an access to either DDR and OR. Any input signal edge during this period will not be detected and will not generate an interrupt.

This case can typically occur if the application refreshes the port configuration registers at intervals during runtime.

Workaround

The workaround is based on software checking the level on the interrupt pin before and after writing to the PxOR or PxDDR registers. If there is a level change (depending on the sensitivity programmed for this pin) the interrupt routine is invoked using the call instruction with three extra PUSH instructions before executing the interrupt routine (this is to make the call compatible with the IRET instruction at the end of the interrupt service routine).

But detection of the level change does ensure that edge occurs during the critical 1 cycle duration and the interrupt has been missed. This may lead to occurrence of same interrupt twice (one hardware and another with software call).

To avoid this, a semaphore is set to '1' before checking the level change. The semaphore is changed to level '0' inside the interrupt routine. When a level change is detected, the semaphore status is checked and if it is '1' this means that the last interrupt has been missed. In this case, the interrupt routine is invoked with the call instruction.

There is another possible case, that is, if writing to PxOR or PxDDR is done with global interrupts disabled (interrupt mask bit set). In this case, the semaphore is changed to '1' when the level change is detected. Detecting a missed interrupt is done after the global interrupts are enabled (interrupt mask bit reset) and by checking the status of the semaphore. If it is '1' this means that the last interrupt was missed and the interrupt routine is invoked with the call instruction.

To implement the workaround, the following software sequence is to be followed for writing into the PxOR/PxDDR registers. The example is for Port PF1 with falling edge interrupt sensitivity. The

IMPORTANT NOTES (Cont'd)

software sequence is given for both cases (global interrupt disabled/enabled).

Case 1: Writing to PxOR or PxDDR with Global Interrupts Enabled:

```
LD A,#01
LD sema,A      ; set the semaphore to '1'
LD A,PFDR
AND A,#02
LD  X,A        ; store the level before writing to PxOR/PxDDR
LD A,#$90
LD PFDDR,A  ; Write to PFDDR
LD A,#$ff
LD  PFOR,A     ; Write to PFOR
LD A,PFDR
AND A,#02
LD Y,A         ; store the level after writing to PxOR/PxDDR
LD A,X         ; check for falling edge
cp A,#02
jrne OUT
TNZ Y
jrne OUT
LD A,sema      ; check the semaphore status if edge is detected
CP A,#01
jrne OUT
call call_routine; call the interrupt routine
OUT:LD A,#00
LD sema,A
.call_routine    ; entry to call_routine
PUSH A
PUSH X
PUSH CC
.ext1_rt         ; entry to interrupt routine
LD A,#00
LD sema,A
IRET
```

Case 2: Writing to PxOR or PxDDR with Global Interrupts Disabled:

```
SIM            ; set the interrupt mask
LD A,PFDR
AND A,#$02
LD  X,A        ; store the level before writing to PxOR/PxDDR
LD A,#$90
LD PFDDR,A; Write into PFDDR
LD A,#$ff
LD PFOR,A          ; Write to PFOR
LD A,PFDR
AND A,#$02
LD Y,A         ; store the level after writing to PxOR/PxDDR
LD A,X         ; check for falling edge
cp A,#$02
jrne OUT
TNZ Y
jrne OUT
LD A,#$01
LD sema,A   ; set the semaphore to '1' if edge is detected
RIM             ; reset the interrupt mask
LD A,sema   ; check the semaphore status
CP A,#$01
jrne OUT
call call_routine; call the interrupt routine
RIM
OUT:           RIM
JP while_loop
.call_routine ; entry to call_routine
PUSH A
PUSH X
PUSH CC
.ext1_rt       ; entry to interrupt routine
LD A,#$00
LD sema,A
IRET
```

16.1.4 Unexpected Reset Fetch

If an interrupt request occurs while a "POP CC" instruction is executed, the interrupt controller does not recognise the source of the interrupt and, by default, passes the RESET vector address to the CPU.

Workaround

To solve this issue, a "POP CC" instruction must always be preceded by a "SIM" instruction.

IMPORTANT NOTES (Cont'd)

16.1.5 Header Time-out Does Not Prevent Wake-up from Mute Mode

Normally, when LINSCI is configured in LIN slave mode, if a header time-out occurs during a LIN header reception (that is, header length > 57 bits), the LIN Header Error bit (LHE) is set, an interrupt occurs to inform the application but the LINSCI should stay in mute mode, waiting for the next header reception.

Problem Description

The LINSCI sampling period is Tbit / 16. If a LIN Header time-out occurs between the 9th and the 15th sample of the Identifier Field Stop Bit (refer to Figure 153), the LINSCI wakes up from mute mode. Nevertheless, LHE is set and LIN Header Detection Flag (LHDF) is kept cleared.

In addition, if LHE is reset by software before this 15th sample (by accessing the SCISR register and reading the SCIDR register in the LINSCI interrupt routine), the LINSCI will generate another LINSCI interrupt (due to the RDRF flag setting).

Impact on application

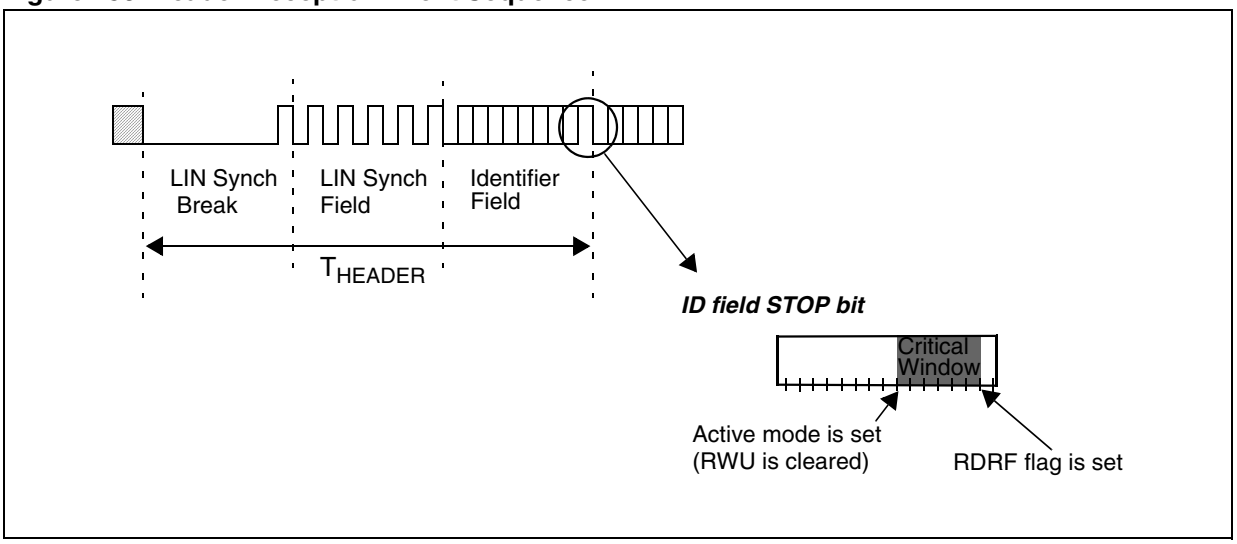
Software may execute the interrupt routine twice after header reception.

Moreover, in reception mode, as the receiver is no longer in mute mode, an interrupt will be generated on each data byte reception.

Workaround

The problem can be detected in the LINSCI interrupt routine. In case of timeout error (LHE is set and LHLR is loaded with 00h), the software can check the RWU bit in the SCICR2 register. If RWU is cleared, it can be set by software. Refer to Figure 154 on page 261. Workaround is shown in bold characters.

Figure 153. Header Reception Event Sequence



IMPORTANT NOTES (Cont'd)

Figure 154.LINSCI Interrupt Routine

```
@interrupt void LINSCI_IT ( void ) /* LINSCI interrupt routine */
{
      /* clear flags */
      SCISR_buffer = SCISR;
      SCIDR_buffer = SCIDR;

      if ( SCISR_buffer & LHE )/* header error ? */
      {
            if (!LHLR)/* header time-out? */
            {
                  if ( !(SCICR2 & RWU) )/* active mode ? */
                  {
                        _asm("sim");/* disable interrupts */
                        SCISR;
                        SCIDR;/* Clear RDRF flag */
                        SCICR2 |= RWU;/* set mute mode */
                        SCISR;
                        SCIDR;/* Clear RDRF flag */
                        SCICR2 |= RWU;/* set mute mode */
                        _asm("rim");/* enable interrupts */
                  }
            }
      }
}
```

Example using Cosmic compiler syntax

16.1.6 TIMD set simultaneously with OC interrupt

If the 16-bit timer is disabled at the same time the output compare event occurs then the output compare flag gets locked and cannot be cleared before the timer is enabled again.

Impact on the application: If output compare interrupt is enabled, then the output compare flag cannot be cleared in the timer interrupt routine. Consequently the interrupt service routine is called repeatedly and the application get stuck which causes the watchdog reset if enabled by the application.

Workaround: Disable the timer interrupt before disabling the timer. Again while enabling, first enable the timer, then the timer interrupts.

Perform the following to disable the timer:

– TACR1 or TBCR1 = 0x00h; // Disable the compare interrupt

– TACSR | or TBCSR | = 0x40; // Disable the timer

– Perform the following to enable the timer again:

– TACSR & or TBCSR &= ~0x40; // Enable the timer

– TACR1 or TBCR1 = 0x40; // Enable the compare interrupt

16.1.7 CAN FIFO Corruption

The beCAN FIFO gets corrupted when a message is received and simultaneously a message is released while FMP = 2. For details and a description of the workaround refer to Section 10.9.7.1 on page 187.

16.2 FLASH/FASTROM DEVICES ONLY

16.2.1 LINSCI Wrong Break Duration

SCI mode

A single break character is sent by setting and resetting the SBK bit in the SCICR2 register. In some cases, the break character may have a longer duration than expected:

- 20 bits instead of 10 bits if M = 0

- 22 bits instead of 11 bits if M = 1

In the same way, as long as the SBK bit is set, break characters are sent to the TDO pin. This may lead to generate one break more than expected.

Occurrence

The occurrence of the problem is random and proportional to the baud rate. With a transmit frequency of 19200 baud (f_{CPU} = 8 MHz and

SCIBRR = 0xC9), the wrong break duration occurrence is around 1%.

Workaround

If this wrong duration is not compliant with the communication protocol in the application, software can request that an Idle line be generated before the break character. In this case, the break duration is always correct assuming the application is not doing anything between the idle and the break. This can be ensured by temporarily disabling interrupts.

The exact sequence is:

- Disable interrupts
- Reset and Set TE (IDLE request)
- Set and Reset SBK (Break Request)
- Re-enable interrupts

LIN mode

If the LINE bit in the SCICR3 is set and the M bit in the SCICR1 register is reset, the LINSCI is in LIN master mode. A single break character is sent by setting and resetting the SBK bit in the SCICR2 register. In some cases, the break character may have a longer duration than expected:

- 24 bits instead of 13 bits

IMPORTANT NOTES (Cont'd)

Occurrence

The occurrence of the problem is random and proportional to the baud rate. With a transmit frequency of 19200 baud (f_{CPU} = 8 MHz and SCIBRR = 0xC9), the wrong break duration occurrence is around 1%.

Analysis

The LIN protocol specifies a minimum of 13 bits for the break duration, but there is no maximum value. Nevertheless, the maximum length of the header is specified as (14+10+10+1) x 1.4 = 49 bits. This is composed of:

- the synch break field (14 bits)
- the synch field (10 bits)
- the identifier field (10 bits)

Every LIN frame starts with a break character. Adding an idle character increases the length of each header by 10 bits. When the problem occurs, the header length is increased by 11 bits and becomes ((14+11)+10+10+1) = 45 bits.

To conclude, the problem is not always critical for LIN communication if the software keeps the time between the sync field and the ID smaller than 4 bits, that is, 208µs at 19200 baud.

The workaround is the same as for SCI mode but considering the low probability of occurrence (1%), it may be better to keep the break generation sequence as it is.

16.2.2 16-bit and 8-bit Timer PWM Mode

In PWM mode, the first PWM pulse is missed after writing the value FFFCh in the OC1R or OC2R register.

16.3 ROM DEVICES ONLY

16.3.1 16-bit Timer PWM Mode Buffering Feature Change

In all devices, the frequency and period of the PWM signal are controlled by comparing the counter with a 16-bit buffer updated by the OCiHR and OCiLR registers. In ROM devices, contrary to the description in Section 10.5.3.5 on page 103, the output compare function is not inhibited after a write instruction to the OCiHR register. Instead the buffer update at the end of the PWM period is inhibited until OCiLR is written. This improved buffer handling is fully compatible with applications written for Flash devices.

17 REVISION HISTORY

Date	Revision	Main changes
24-Oct-2005	4	Added standard version 16K ROM/Flash devices Modified data retention in Section 12.7 Added “6” and “3” standard version device type coding to Figure 152 and Figure 150 on page 254 Modified power consumption Section 12.4 Added CDM in Section 12.8.3.1 Added “External interrupt missed” Section 16.1.3
26-Sep-2006	5	Replaced TQFP with LQFP packages throughout document Changed device summary on page 1 Changed Section 9.2.1 on page 46 Changed title of Section 9.6 on page 50 from “I/O Port Implementation” to “I/O Port Register Configurations” Changed Section 10.6.3.3 on page 114 Corrected name of bit 5 in SPICSR register in Table 22 on page 121 Changed Section 12.5.4 on page 228 Added links to Table 39 and Table 40 in Section 12.8.3 on page 233 Removed EMC protection circuitry in Figure 135 on page 240 (device works correctly without these components) Changed Section 12.12.1 on page 243 Changed title of Figure 146 on page 249 Changed title of Figure 124 on page 210 Changed title of Figure 147 on page 249 Changed notes in Section 13.2 on page 251 Changed AFI mapping for “OPTION BYTE 1” on page 253 Changed Figure 150 on page 254, Figure 151 on page 255 and Figure 152 on page 255 Changed “ST72561 MICROCONTROLLER OPTION LIST” on page 256 Deleted Section 15.1.5 “Clearing active interrupts outside interrupt routine” (text already exists in Section 16.1.2 on page 258) Added Section 16.1.5 on page 260 Removed automotive part numbers, see separate ST72561-auto datasheet
19-Jun-2007	6	Added EMI characteristics for LQFP44 in Section 12.8. Updated ADC accuracy characteristics in Section 12.13.
02-Oct-2008	7	Updated static latchup Section 12.8.3.2 on page 233 Updated Flash data retention characteristics in Section 12.7.2. Added Section 16.1.6 on page 261

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2008 STMicroelectronics - All rights reserved

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

Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View ST72F561R9TATR on WIN SOURCE
- STMicroelectronics Information

Optimize Your Supply Chain with WIN SOURCE Solutions

- Global Sourcing Solution
- Obsolete Management
- Cost Control Management
- Shortage Management
- Alternative Solution
- Excess Inventory Management