



## STM8L05xx, STM8L15xx, STM8L162x, STM8AL31xx and STM8AL3Lxx microcontroller family

### Introduction

This reference manual targets application developers. It provides complete information on how to use the STM8L05xx, STM8L15xx, STM8L16xx, STM8AL31xx and STM8AL3Lxx microcontroller memory and peripherals. The STM8L05xx/STM8L15xx/STM8L16xx/STM8AL31xx/STM8AL3Lxx is a family of microcontrollers with different memory densities, packages and peripherals. These products are designed for ultralow power applications. Refer to the product datasheet for the complete list of available peripherals.

For ordering information, pin description, mechanical and electrical device characteristics, please refer to the product datasheet. For information on the STM8 SWIM communication protocol and debug module, please refer to the user manual (UM0470). For information on the STM8 core, please refer to the STM8 CPU programming manual (PM0044). For information on programming, erasing and protection of the internal Flash memory please refer to the STM8L Flash programming manual (PM0054).

**Table 1. Applicable products**

Type	Part numbers
Microcontrollers	<b>Value line low density STM8L05xx devices:</b> STM8L051x3 microcontrollers with 8-KB Flash <b>Value line medium density STM8L05xx devices:</b> STM8L052x6 microcontrollers with 32-KB Flash <b>Value line high density STM8L05xx devices:</b> STM8L052x8 microcontrollers with 64-KB Flash <b>Low density STM8L15x devices:</b> STM8L151C2/K2/G2/F2, STM8L151C3/K3/G3/F3 microcontrollers with 4-KB or 8-KB Flash <b>Medium density STM8L15xx devices:</b> STM8L151C4/K4/G4, STM8L151C6/K6/G6, STM8L152C4/K4 and STM8L152C6/K6 microcontrollers with 16-KB or 32-KB Flash <b>Medium density STM8AL31xx/STM8AL3Lxx devices:</b> STM8AL3168, STM8AL3166, STM8AL3148, STM8AL3146, STM8AL3138, STM8AL3136, STM8AL3L68, STM8AL3L66, STM8AL3L48, STM8AL3L46 microcontrollers with 8-KB, 16-KB or 32-KB Flash <b>Medium+ density STM8L15xx devices:</b> STM8L151R6 and STM8L152R6 microcontrollers with 32-KB Flash (Wider range of peripherals than medium density devices) <b>High density STM8L15xx devices:</b> STM8L151x8 and the STM8L152x8 microcontrollers with 64-KB Flash (Same peripheral set as medium+) <b>High density STM8L16xx devices:</b> STM8L162x8 microcontrollers with 64-KB Flash (Same peripheral set as high density STM8L152 devices plus the AES hardware accelerator)

# Contents

<b>1</b>	<b>Central processing unit (CPU) . . . . .</b>	<b>30</b>
1.1	Introduction . . . . .	30
1.2	CPU registers . . . . .	30
1.2.1	Description of CPU registers . . . . .	30
1.2.2	STM8 CPU register map . . . . .	34
1.3	Global configuration register (CFG_GCR) . . . . .	34
1.3.1	Activation level . . . . .	34
1.3.2	SWIM disable . . . . .	35
1.3.3	Description of global configuration register (CFG_GCR) . . . . .	35
1.3.4	Global configuration register map and reset values . . . . .	35
<b>2</b>	<b>Boot ROM . . . . .</b>	<b>36</b>
<b>3</b>	<b>Flash program memory and data EEPROM . . . . .</b>	<b>37</b>
3.1	Introduction . . . . .	37
3.2	Glossary . . . . .	37
3.3	Main Flash memory features . . . . .	38
3.4	Memory organization . . . . .	39
3.4.1	Low density device memory organization . . . . .	39
3.4.2	Medium density device memory organization . . . . .	40
3.4.3	Medium+ density device memory organization . . . . .	41
3.4.4	High density device memory organization . . . . .	42
3.4.5	Proprietary code area (PCODE) . . . . .	43
3.4.6	User boot area (UBC) . . . . .	43
3.4.7	Data EEPROM (DATA) . . . . .	46
3.4.8	Main program area . . . . .	46
3.4.9	Option bytes . . . . .	46
3.5	Memory protection . . . . .	47
3.5.1	Readout protection . . . . .	47
3.5.2	Memory access security system (MASS) . . . . .	47
3.5.3	Enabling write access to option bytes . . . . .	49
3.6	Memory programming . . . . .	49
3.6.1	Read-while-write (RWW) . . . . .	49

3.6.2	Byte programming .....	49
3.6.3	Word programming .....	50
3.6.4	Block programming .....	50
3.6.5	Option byte programming .....	52
3.7	Flash low power modes .....	52
3.8	ICP and IAP .....	52
3.9	Flash registers .....	57
3.9.1	Flash control register 1 (FLASH_CR1) .....	57
3.9.2	Flash control register 2 (FLASH_CR2) .....	58
3.9.3	Flash program memory unprotecting key register (FLASH_PUKR) .....	58
3.9.4	Data EEPROM unprotection key register (FLASH_DUKR) .....	59
3.9.5	Flash status register (FLASH_IAPSR) .....	59
3.9.6	Flash register map and reset values .....	60
<b>4</b>	<b>Single wire interface module (SWIM) and debug module (DM) .....</b>	<b>61</b>
4.1	Introduction .....	61
4.2	Main features .....	61
4.3	SWIM modes .....	62
<b>5</b>	<b>Memory and register map .....</b>	<b>63</b>
5.1	Register description abbreviations .....	63
<b>6</b>	<b>Power control (PWR) .....</b>	<b>64</b>
6.1	Power supply .....	64
6.2	Power-on reset (POR)/power-down reset (PDR) .....	65
6.3	Brownout reset (BOR) .....	66
6.4	Programmable voltage detector (PWD) .....	67
6.5	Internal voltage reference (VREFINT) .....	68
6.6	Voltage regulator .....	69
6.7	PWR registers .....	70
6.7.1	Power control and status register 1 (PWR_CSR1) .....	70
6.7.2	PWR control and status register 2 (PWR_CSR2) .....	71
6.7.3	PWR register map and reset values .....	71
<b>7</b>	<b>Low power modes .....</b>	<b>72</b>
7.1	Slowing down the system clocks .....	73

---

7.2	Peripheral clock gating (PCG) . . . . .	73
7.3	Wait mode (WFI or WFE mode) . . . . .	73
7.4	Wait for interrupt (WFI) mode . . . . .	74
7.5	Wait for event (WFE) mode . . . . .	74
7.5.1	WFE registers . . . . .	75
7.5.2	WFE register map and reset values . . . . .	80
7.6	Low power run mode . . . . .	81
7.6.1	Entering Low power run mode . . . . .	81
7.6.2	Exiting Low power run mode . . . . .	81
7.7	Low power wait mode . . . . .	81
7.8	Halt mode . . . . .	82
7.8.1	Entering Halt mode . . . . .	82
7.8.2	Exiting Halt mode . . . . .	82
7.9	Active-halt mode . . . . .	83
<b>8</b>	<b>Reset (RST) . . . . .</b>	<b>84</b>
8.1	“Reset state” and “under reset” definitions . . . . .	84
8.2	External reset (NRST pin) . . . . .	84
8.2.1	Asynchronous external reset description . . . . .	84
8.2.2	Configuring NRST/PA1 pin as general purpose output . . . . .	85
8.3	Internal reset . . . . .	85
8.3.1	Power-on reset (POR) . . . . .	85
8.3.2	Independent watchdog reset . . . . .	85
8.3.3	Window watchdog reset . . . . .	85
8.3.4	SWIM reset . . . . .	85
8.3.5	Illegal opcode reset . . . . .	85
8.4	RST registers . . . . .	86
8.4.1	Reset pin configuration register (RST_CR) . . . . .	86
8.4.2	Reset status register (RST_SR) . . . . .	86
8.4.3	RST register map and reset values . . . . .	87
<b>9</b>	<b>Clock control (CLK) . . . . .</b>	<b>88</b>
9.1	Introduction . . . . .	88
9.2	HSE clock . . . . .	89
9.3	HSI clock . . . . .	91

9.4	LSE clock .....	91
9.5	LSI clock .....	92
9.6	System clock sources .....	92
9.6.1	System startup .....	92
9.6.2	System clock switching procedures .....	93
9.7	Peripheral clock gating (PCG) .....	96
9.8	Clock security system (CSS) .....	96
9.8.1	Clock security system on HSE .....	96
9.8.2	Clock security system on LSE .....	97
9.8.3	CSS on LSE control and status register (CSSLSE_CSR) .....	98
9.8.4	CSS on LSE register map and reset values .....	99
9.9	RTC and LCD clock .....	99
9.10	BEEP clock .....	99
9.11	Configurable clock output capability (CCO) .....	99
9.12	Clock-independent system clock sources for TIM2/TIM3 .....	100
9.13	CLK interrupts .....	100
9.14	CLK registers .....	101
9.14.1	System clock divider register (CLK_CKDIVR) .....	101
9.14.2	Clock RTC register (CLK_CRTCR) .....	101
9.14.3	Internal clock register (CLK_ICKCR) .....	103
9.14.4	Peripheral clock gating register 1 (CLK_PCKENR1) .....	104
9.14.5	Peripheral clock gating register 2 (CLK_PCKENR2) .....	105
9.14.6	Peripheral clock gating register 3 (CLK_PCKENR3) .....	106
9.14.7	Configurable clock output register (CLK_CCOR) .....	107
9.14.8	External clock register (CLK_ECKCR) .....	108
9.14.9	System clock status register (CLK_SCSR) .....	109
9.14.10	System clock switch register (CLK_SWR) .....	110
9.14.11	Switch control register (CLK_SWCR) .....	110
9.14.12	Clock security system register (CLK_CSSR) .....	111
9.14.13	Clock BEEP register (CLK_CBEEPR) .....	112
9.14.14	HSI calibration register (CLK_HSICALR) .....	112
9.14.15	HSI clock calibration trimming register (CLK_HSITRIMR) .....	113
9.14.16	HSI unlock register (CLK_HSIUNLCKR) .....	113
9.14.17	Main regulator control status register (CLK_REGCSR) .....	114
9.14.18	CLK register map and reset values .....	115

---

<b>10</b>	<b>General purpose I/O ports (GPIO) . . . . .</b>	<b>116</b>
10.1	Introduction . . . . .	116
10.2	GPIO main features . . . . .	116
10.3	Port configuration and usage . . . . .	117
10.3.1	Input modes . . . . .	118
10.3.2	Output modes . . . . .	119
10.4	Reset configuration . . . . .	119
10.5	Unused I/O pins . . . . .	119
10.6	Low power modes . . . . .	119
10.7	Input mode details . . . . .	119
10.7.1	Alternate function input . . . . .	119
10.7.2	Interrupt capability . . . . .	120
10.8	Output mode details . . . . .	120
10.8.1	Alternate function output . . . . .	120
10.8.2	Slope control . . . . .	120
10.9	GPIO registers . . . . .	121
10.9.1	Port x output data register (Px_ODR) . . . . .	121
10.9.2	Port x pin input register (Px_IDR) . . . . .	121
10.9.3	Port x data direction register (Px_DDR) . . . . .	122
10.9.4	Port x control register 1 (Px_CR1) . . . . .	122
10.9.5	Port x control register 2 (Px_CR2) . . . . .	123
10.9.6	Peripheral alternate function remapping . . . . .	123
10.9.7	GPIO register map and reset values . . . . .	123
<b>11</b>	<b>Routing interface (RI) and system configuration controller (SYSCFG) . . . . .</b>	<b>124</b>
11.1	Introduction . . . . .	124
11.2	RI main features . . . . .	124
11.2.1	RI functional description . . . . .	126
11.2.2	I/O groups . . . . .	126
11.2.3	TIM1 input capture routing . . . . .	128
11.2.4	TIM2 & TIM3 routing . . . . .	129
11.2.5	Comparator routing . . . . .	130
11.2.6	DAC routing . . . . .	130
11.2.7	Internal reference voltage routing . . . . .	132
11.3	RI interrupts . . . . .	132

11.4	RI registers . . . . .	132
11.4.1	Timer input capture routing register 1 (RI_ICR1) . . . . .	132
11.4.2	Timer input capture routing register 2 (RI_ICR2) . . . . .	133
11.4.3	I/O input register 1 (RI_IOIR1) . . . . .	133
11.4.4	I/O input register 2 (RI_IOIR2) . . . . .	133
11.4.5	I/O input register 3 (RI_IOIR3) . . . . .	134
11.4.6	I/O control mode register 1 (RI_IOCMR1) . . . . .	134
11.4.7	I/O control mode register 2 (RI_IOCMR2) . . . . .	134
11.4.8	I/O control mode register 3 (RI_IOCMR3) . . . . .	135
11.4.9	I/O switch register 1 (RI_IOSR1) . . . . .	135
11.4.10	I/O switch register 2 (RI_IOSR2) . . . . .	137
11.4.11	I/O switch register 3 (RI_IOSR3) . . . . .	138
11.4.12	IO group control register (RI_IOGCR) . . . . .	139
11.4.13	Analog switch register 1 (RI_ASCR1) . . . . .	141
11.4.14	Analog switch register 2 (RI_ASCR2) . . . . .	141
11.4.15	Resistor control register (RI_RCR) . . . . .	142
11.4.16	Control register (RI_CR) . . . . .	143
11.4.17	IO mask register 1 (RI_IOMR1) . . . . .	144
11.4.18	IO mask register 2 (RI_IOMR2) . . . . .	144
11.4.19	IO mask register 3 (RI_IOMR3) . . . . .	145
11.4.20	IO mask register 4 (RI_IOMR4) . . . . .	145
11.4.21	I/O input register 4 (RI_IOIR4) . . . . .	146
11.4.22	I/O control mode register 4 (RI_IOCMR4) . . . . .	146
11.4.23	I/O switch register 4 (RI_IOSR4) . . . . .	147
11.4.24	RI register map and reset values . . . . .	148
11.5	SYSCFG registers . . . . .	150
11.5.1	SYSCFG remap control register 1 (SYSCFG_RMPCR1) . . . . .	150
11.5.2	SYSCFG remap control register 2 (SYSCFG_RMPCR2) . . . . .	151
11.5.3	SYSCFG remap control register 3 (SYSCFG_RMPCR3) . . . . .	152
11.5.4	SYSCFG register map and reset values . . . . .	153
<b>12</b>	<b>Interrupt controller (ITC) . . . . .</b>	<b>154</b>
12.1	ITC introduction . . . . .	154
12.2	Interrupt masking and processing flow . . . . .	154
12.2.1	Servicing pending interrupts . . . . .	155
12.2.2	Interrupt sources . . . . .	156
12.3	Interrupts and low power modes . . . . .	157

---

12.4	Activation level/low power mode control . . . . .	158
12.5	Concurrent and nested interrupt management . . . . .	158
12.5.1	Concurrent interrupt management mode . . . . .	158
12.5.2	Nested interrupt management mode . . . . .	159
12.6	External interrupts . . . . .	160
12.7	Interrupt instructions . . . . .	161
12.8	Interrupt mapping . . . . .	161
12.9	ITC and EXTI registers . . . . .	162
12.9.1	CPU condition code register interrupt bits (CCR) . . . . .	162
12.9.2	Software priority register x (ITC_SPRx) . . . . .	163
12.9.3	External interrupt control register 1 (EXTI_CR1) . . . . .	163
12.9.4	External interrupt control register 2 (EXTI_CR2) . . . . .	165
12.9.5	External interrupt control register 3 (EXTI_CR3) . . . . .	166
12.9.6	External interrupt control register 4 (EXTI_CR4) . . . . .	167
12.9.7	External interrupt status register 1 (EXTI_SR1) . . . . .	167
12.9.8	External interrupt status register 2 (EXTI_SR2) . . . . .	168
12.9.9	External interrupt port select register (EXTI_CONF1) . . . . .	169
12.9.10	External interrupt port select register (EXTI_CONF2) . . . . .	170
12.9.11	ITC and EXTI register map and reset values . . . . .	171
<b>13</b>	<b>Direct memory access controller (DMA) . . . . .</b>	<b>172</b>
13.1	DMA introduction . . . . .	172
	Glossary . . . . .	172
13.2	DMA main features . . . . .	173
13.3	DMA functional description . . . . .	174
13.3.1	DMA transactions . . . . .	174
13.3.2	DMA arbiter . . . . .	175
13.3.3	DMA channels . . . . .	175
13.3.4	DMA1 request mapping . . . . .	182
13.3.5	DMA hardware request description . . . . .	184
13.4	DMA low power modes . . . . .	185
13.5	DMA interrupts . . . . .	186
13.6	DMA registers . . . . .	186
13.6.1	DMA global configuration & status register (DMA_GCSR) . . . . .	186
13.6.2	DMA global interrupt register 1 (DMA_GIR1) . . . . .	187
13.6.3	DMA channel configuration register (DMA_CxCR) . . . . .	187

13.6.4	DMA channel status & priority register (DMA_CxSPR) . . . . .	189
13.6.5	DMA number of data to transfer register (DMA_CxNDTR) . . . . .	190
13.6.6	DMA peripheral address high register (DMA_CxPARH) . . . . .	190
13.6.7	DMA peripheral address low register (DMA_CxPARL) . . . . .	191
13.6.8	DMA channel 3 peripheral address high & memory 1 address high register (DMA_C3PARH_C3M1ARH) . . . . .	191
13.6.9	DMA channel 3 peripheral address low & memory 1 address low register (DMA_C3PARL_C3M1ARL) . . . . .	192
13.6.10	DMA memory 0 address high register (DMA_CxM0ARH) . . . . .	192
13.6.11	DMA memory 0 address low register (DMA_CxM0ARL) . . . . .	193
13.6.12	DMA channel 3 memory 0 extended address register (DMA_C3M0EAR) . . . . .	193
13.6.13	DMA register map and reset values . . . . .	194
<b>14</b>	<b>Analog-to-digital converter (ADC) . . . . .</b>	<b>196</b>
14.1	ADC introduction . . . . .	196
14.2	ADC main features . . . . .	196
14.3	ADC functional description . . . . .	197
14.3.1	General description . . . . .	197
14.3.2	Number of analog channels . . . . .	198
14.3.3	ADC on-off control . . . . .	198
14.3.4	Single conversion mode . . . . .	198
14.3.5	Continuous conversion mode . . . . .	200
14.3.6	ADC clock . . . . .	200
14.3.7	Analog watchdog . . . . .	200
14.3.8	Interrupts . . . . .	201
14.3.9	Channel selection (Scan mode) . . . . .	201
14.3.10	Data integrity . . . . .	202
14.3.11	DMA transfer . . . . .	202
14.3.12	Configurable resolution . . . . .	202
14.3.13	Data alignment . . . . .	202
14.3.14	Programmable sampling time . . . . .	203
14.3.15	Schmitt trigger disabling . . . . .	204
14.3.16	Temperature sensor . . . . .	204
14.3.17	Internal reference voltage conversion . . . . .	205
14.4	ADC low power modes . . . . .	205
14.5	ADC interrupts . . . . .	205

---

14.6	ADC registers . . . . .	206
14.6.1	ADC configuration register 1 (ADC_CR1) . . . . .	206
14.6.2	ADC configuration register 2 (ADC_CR2) . . . . .	207
14.6.3	ADC configuration register 3 (ADC_CR3) . . . . .	208
14.6.4	ADC status register (ADC_SR) . . . . .	209
14.6.5	ADC data register high (ADC_DRH) . . . . .	210
14.6.6	ADC data register low (ADC_DRL) . . . . .	210
14.6.7	ADC high threshold register high (ADC_HTRH) . . . . .	211
14.6.8	ADC high threshold register low (ADC_HTRL) . . . . .	211
14.6.9	ADC low threshold register high (ADC_LTRH) . . . . .	211
14.6.10	ADC low threshold register low (ADC_LTRL) . . . . .	212
14.6.11	ADC channel sequence 1 register (ADC_SQR1) . . . . .	212
14.6.12	ADC channel sequence register 2 (ADC_SQR2) . . . . .	213
14.6.13	ADC channel select scan 3 (ADC_SQR3) . . . . .	213
14.6.14	ADC channel select scan 4 (ADC_SQR4) . . . . .	214
14.6.15	ADC trigger disable 1 (ADC_TRIGR1) . . . . .	214
14.6.16	ADC trigger disable 2 (ADC_TRIGR2) . . . . .	215
14.6.17	ADC trigger disable 3 (ADC_TRIGR3) . . . . .	215
14.6.18	ADC trigger disable 4 (ADC_TRIGR4) . . . . .	215
14.6.19	ADC register map and reset values . . . . .	216
<b>15</b>	<b>Digital-to-analog converter (DAC) . . . . .</b>	<b>217</b>
15.1	DAC introduction . . . . .	217
15.2	DAC main features . . . . .	217
15.3	DAC functional description . . . . .	219
15.3.1	DAC channel x enable . . . . .	219
15.3.2	DAC output buffer enable . . . . .	219
15.3.3	DAC output switch configuration . . . . .	219
15.3.4	DAC data format . . . . .	220
15.3.5	DAC conversion sequence . . . . .	220
15.3.6	DAC output voltage . . . . .	220
15.3.7	DAC trigger selection . . . . .	220
15.3.8	DAC DMA request . . . . .	221
15.3.9	DAC DMA underrun interrupt . . . . .	221
15.3.10	Noise generation . . . . .	221
15.3.11	Triangle-wave generation . . . . .	222
15.3.12	Dual DAC conversion . . . . .	223

---

15.4	DAC registers .....	227
15.4.1	DAC channel x control register 1 (DAC_CHxCR1) .....	227
15.4.2	DAC channel x control register 2 (DAC_CHxCR2) .....	228
15.4.3	DAC software trigger register (DAC_SWTRIGR) .....	229
15.4.4	DAC status register (DAC_SR) .....	229
15.4.5	DAC channel x right aligned data holding register high (DAC_RDHRH) .....	230
15.4.6	DAC channel x right aligned data holding register low (DAC_CHxRDHRL) .....	230
15.4.7	DAC channel x left aligned data holding register high (DAC_CHxLDHRH) .....	230
15.4.8	DAC channel x left aligned data holding register low (DAC_CHxLDHRL) .....	231
15.4.9	DAC channel x 8-bit data holding register (DAC_CHxDHR8) .....	231
15.4.10	DAC channel x dual mode right aligned data holding register high (DAC_DCHxRDHRH) .....	232
15.4.11	DAC channel x dual mode right aligned data holding register low (DAC_DCHxRDHRL) .....	232
15.4.12	DAC channel x dual mode left aligned data holding register high (DAC_DCHxLDHRH) .....	233
15.4.13	DAC channel x left aligned data holding register low (DAC_DCHxLDHRL) .....	233
15.4.14	DAC channel x dual mode 8-bit data holding register (DAC_DCHxDHR8) .....	234
15.4.15	DAC channel x data output register high (DAC_CHxDORH) .....	234
15.4.16	DAC channel x data output register low (DAC_CHxDORL) .....	234
15.4.17	DAC register map and reset values .....	235
16	<b>Comparators (COMP) .....</b>	<b>238</b>
16.1	COMP introduction .....	238
16.2	COMP main features .....	240
16.3	Comparator 1 (COMP1) .....	241
16.4	Comparator 2 (COMP2) .....	242
16.5	Using the comparators in window mode .....	243
16.6	COMP low power modes .....	244
16.7	COMP interrupts .....	244
16.8	COMP registers .....	245

---

16.8.1	Comparator control and status register 1 (COMP_CSR1) . . . . .	245
16.8.2	Comparator control and status register 2 (COMP_CSR2) . . . . .	246
16.8.3	Comparator control and status register 3 (COMP_CSR3) . . . . .	247
16.8.4	Comparator control and status register 4 (COMP_CSR4) . . . . .	248
16.8.5	Comparator control and status register 5 (COMP_CSR5) . . . . .	248
16.8.6	COMP register map and reset values . . . . .	249
<b>17</b>	<b>LCD controller . . . . .</b>	<b>250</b>
17.1	LCD controller introduction . . . . .	250
17.1.1	Definitions . . . . .	250
17.2	LCD controller main features . . . . .	251
17.3	LCD functional description . . . . .	253
17.3.1	General description . . . . .	253
17.3.2	Frequency generator . . . . .	254
17.3.3	Common driver . . . . .	257
17.3.4	Segment driver . . . . .	265
17.3.5	Enabling a segment . . . . .	266
17.3.6	Blink . . . . .	266
17.3.7	Multiplexing COM[7:4] and SEG[43:40], SEG[39:36], or SEG[31:28] . . . . .	266
17.3.8	Generation of LCD voltage levels . . . . .	267
17.3.9	LCD buffer update . . . . .	270
17.4	LCD controller low power modes . . . . .	270
17.5	LCD controller interrupts . . . . .	270
17.6	LCD controller registers . . . . .	271
17.6.1	Control register 1 (LCD_CR1) . . . . .	271
17.6.2	Control register 2 (LCD_CR2) . . . . .	273
17.6.3	Control register 3 (LCD_CR3) . . . . .	274
17.6.4	Frequency selection register (LCD_FRQ) . . . . .	275
17.6.5	Port mask registers (LCD_PM) . . . . .	275
17.6.6	Control register 4 (LCD_CR4) . . . . .	276
17.6.7	LCD display memory (LCD_RAM) . . . . .	277
17.6.8	LCD register map and reset values . . . . .	279
<b>18</b>	<b>Timer overview . . . . .</b>	<b>281</b>
18.1	Timer feature comparison . . . . .	282
18.2	Glossary of timer signal names . . . . .	282

<b>19</b>	<b>16-bit advanced control timer (TIM1) . . . . .</b>	<b>284</b>
19.1	Introduction . . . . .	284
19.2	TIM1 main features . . . . .	285
19.3	TIM1 time base unit . . . . .	287
19.3.1	Reading and writing to the 16-bit counter . . . . .	288
19.3.2	Write sequence for 16-bit TIM1_ARR register . . . . .	288
19.3.3	Prescaler . . . . .	288
19.3.4	Up-counting mode . . . . .	289
19.3.5	Down-counting mode . . . . .	291
19.3.6	Center-aligned mode (up/down counting) . . . . .	293
19.3.7	Repetition down-counter . . . . .	295
19.4	TIM1 clock/trigger controller . . . . .	297
19.4.1	Prescaler clock (CK_PSC) . . . . .	297
19.4.2	Internal clock source (fSYSCLK) . . . . .	298
19.4.3	External clock source mode 1 . . . . .	298
19.4.4	External clock source mode 2 . . . . .	300
19.4.5	Trigger synchronization . . . . .	301
19.4.6	Synchronization between timers . . . . .	305
19.5	TIM1 capture/compare channels . . . . .	311
19.5.1	Write sequence for 16-bit TIM1_CCRI registers . . . . .	312
19.5.2	Input stage . . . . .	313
19.5.3	Input capture mode . . . . .	314
19.5.4	Output stage . . . . .	316
19.5.5	Forced output mode . . . . .	317
19.5.6	Output compare mode . . . . .	317
19.5.7	PWM mode . . . . .	319
19.5.8	Using the break function . . . . .	326
19.5.9	Clearing the OCiREF signal on an external event . . . . .	329
19.5.10	Encoder interface mode . . . . .	330
19.5.11	Timer input XOR function . . . . .	332
19.5.12	Interfacing with Hall sensors . . . . .	332
19.6	TIM1 interrupts . . . . .	334
19.6.1	TIM1 wait-for-event capability . . . . .	334
19.7	TIM1 DMA . . . . .	334
19.7.1	DMA single mode . . . . .	334
19.7.2	DMA burst mode . . . . .	335

19.8	TIM1 registers . . . . .	336
19.8.1	Control register 1 (TIM1_CR1) . . . . .	336
19.8.2	Control register 2 (TIM1_CR2) . . . . .	338
19.8.3	Slave mode control register (TIM1_SMCR) . . . . .	339
19.8.4	External trigger register (TIM1_ETR) . . . . .	340
19.8.5	DMA request enable register (TIM1_DER) . . . . .	342
19.8.6	Interrupt enable register (TIM1_IER) . . . . .	343
19.8.7	Status register 1 (TIM1_SR1) . . . . .	344
19.8.8	Status register 2 (TIM1_SR2) . . . . .	345
19.8.9	Event generation register (TIM1_EGR) . . . . .	346
19.8.10	Capture/compare mode register 1 (TIM1_CCMR1) . . . . .	347
19.8.11	Capture/compare mode register 2 (TIM1_CCMR2) . . . . .	350
19.8.12	Capture/compare mode register 3 (TIM1_CCMR3) . . . . .	351
19.8.13	Capture/compare mode register 4 (TIM1_CCMR4) . . . . .	352
19.8.14	Capture/compare enable register 1 (TIM1_CCER1) . . . . .	353
19.8.15	Capture/compare enable register 2 (TIM1_CCER2) . . . . .	356
19.8.16	Counter high (TIM1_CNTRH) . . . . .	356
19.8.17	Counter low (TIM1_CNTRL) . . . . .	357
19.8.18	Prescaler high (TIM1_PSCRH) . . . . .	357
19.8.19	Prescaler low (TIM1_PSCRL) . . . . .	357
19.8.20	Auto-reload register high (TIM1_ARRH) . . . . .	358
19.8.21	Auto-reload register low (TIM1_ARRL) . . . . .	358
19.8.22	Repetition counter register (TIM1_RCR) . . . . .	358
19.8.23	Capture/compare register 1 high (TIM1_CCR1H) . . . . .	359
19.8.24	Capture/compare register 1 low (TIM1_CCR1L) . . . . .	359
19.8.25	Capture/compare register 2 high (TIM1_CCR2H) . . . . .	360
19.8.26	Capture/compare register 2 low (TIM1_CCR2L) . . . . .	360
19.8.27	Capture/compare register 3 high (TIM1_CCR3H) . . . . .	361
19.8.28	Capture/compare register 3 low (TIM1_CCR3L) . . . . .	361
19.8.29	Capture/compare register 4 high (TIM1_CCR4H) . . . . .	362
19.8.30	Capture/compare register 4 low (TIM1_CCR4L) . . . . .	362
19.8.31	Break register (TIM1_BKR) . . . . .	363
19.8.32	Deadtime register (TIM1_DTR) . . . . .	364
19.8.33	Output idle state register (TIM1_OISR) . . . . .	365
19.8.34	DMA control register 1 (TIM1_DCR1) . . . . .	365
19.8.35	DMA control register 2 (TIM1_DCR2) . . . . .	366
19.8.36	DMA address for burst mode (TIM1_DMAR) . . . . .	366

---

19.8.37	TIM1 register map and reset values	367
<b>20</b>	<b>16-bit general purpose timers (TIM2, TIM3, TIM5)</b>	<b>369</b>
20.1	Introduction	369
20.2	TIMx main features	369
20.3	TIMx functional description	370
20.3.1	Time base unit	370
20.3.2	Clock/trigger controller	371
20.3.3	Capture/compare channels	372
20.3.4	Timer input XOR function	374
20.4	TIMx interrupts	374
20.5	TIMx registers	375
20.5.1	Control register 1 (TIMx_CR1)	375
20.5.2	Control register 2 (TIMx_CR2)	376
20.5.3	Slave mode control register (TIMx_SMCR)	377
20.5.4	External trigger register (TIMx_ETR)	378
20.5.5	DMA request enable register (TIMx_DER)	379
20.5.6	Interrupt enable register (TIMx_IER)	380
20.5.7	Status register 1 (TIMx_SR1)	381
20.5.8	Status register 2 (TIMx_SR2)	382
20.5.9	Event generation register (TIMx_EGR)	383
20.5.10	Capture/compare mode register 1 (TIMx_CCMR1)	384
20.5.11	Capture/compare mode register 2 (TIMx_CCMR2)	386
20.5.12	Capture/compare enable register 1 (TIMx_CCER1)	387
20.5.13	Counter high (TIMx_CNTRH)	388
20.5.14	Counter low (TIMx_CNTRL)	389
20.5.15	Prescaler register (TIMx_PSCR)	389
20.5.16	Auto-reload register high (TIMx_ARRH)	389
20.5.17	Auto-reload register low (TIMx_ARRL)	390
20.5.18	Capture/compare register 1 high (TIMx_CCR1H)	390
20.5.19	Capture/compare register 1 low (TIMx_CCR1L)	391
20.5.20	Capture/compare register 2 high (TIMx_CCR2H)	391
20.5.21	Capture/compare register 2 low (TIMx_CCR2L)	391
20.5.22	Break register (TIMx_BKR)	392
20.5.23	Output idle state register (TIMx_OISR)	394
20.5.24	TIMx register map and reset values	394

---

<b>21</b>	<b>8-bit basic timer (TIM4) . . . . .</b>	<b>396</b>
21.1	Introduction . . . . .	396
21.2	TIM4 main features . . . . .	396
21.3	TIM4interrupts . . . . .	396
21.4	TIM4 clock selection . . . . .	397
21.5	TIM4 registers . . . . .	397
21.5.1	Control register 1 (TIM4_CR1) . . . . .	397
21.5.2	Control register 2 (TIM4_CR2) . . . . .	398
21.5.3	Slave mode control register (TIM4_SMCR) . . . . .	398
21.5.4	DMA request enable register (TIM4_DER) . . . . .	400
21.5.5	Interrupt enable register (TIM4_IER) . . . . .	400
21.5.6	Status register 1 (TIM4_SR) . . . . .	400
21.5.7	Event generation register (TIM4_EGR) . . . . .	401
21.5.8	Counter (TIM4_CNTR) . . . . .	401
21.5.9	Prescaler register (TIM4_PSCR) . . . . .	402
21.5.10	Auto-reload register (TIM4_ARR) . . . . .	402
21.5.11	TIM4 register map and reset values . . . . .	403
<b>22</b>	<b>Infrared (IRTIM) interface . . . . .</b>	<b>404</b>
22.1	Introduction . . . . .	404
22.2	Main features . . . . .	404
22.3	IRTIM register . . . . .	405
22.3.1	Control register (IR_CR) . . . . .	405
22.3.2	IRTIM register map and reset values . . . . .	405
<b>23</b>	<b>Beeper (BEEP) . . . . .</b>	<b>406</b>
23.1	Introduction . . . . .	406
23.2	BEEP functional description . . . . .	407
23.2.1	Beeper operation . . . . .	407
23.2.2	Beeper calibration . . . . .	407
23.2.3	LSI clock frequency measurement . . . . .	407
23.3	BEEP registers . . . . .	408
23.3.1	BEEP control/status register 1 (BEEP_CSR1) . . . . .	408
23.3.2	BEEP control/status register 2 (BEEP_CSR2) . . . . .	408
23.3.3	BEEP register map and reset values . . . . .	409

<b>24</b>	<b>Real-time clock (RTC) . . . . .</b>	<b>410</b>
24.1	Introduction . . . . .	410
24.2	RTC main features . . . . .	411
24.3	RTC functional description . . . . .	411
24.3.1	Clock and prescalers . . . . .	412
24.3.2	Real-time clock and calendar . . . . .	414
24.3.3	Programmable alarm . . . . .	415
24.3.4	Periodic auto-wakeup . . . . .	415
24.3.5	RTC initialization and configuration . . . . .	416
24.3.6	Reading the calendar . . . . .	417
24.3.7	Resetting the RTC . . . . .	418
24.3.8	RTC synchronization (low, medium+ and high density devices only) . . . . .	418
24.3.9	RTC smooth digital calibration (low, medium+ and high density devices only) . . . . .	419
24.3.10	Tamper detection (low, medium+ and high density devices only) . . . . .	421
24.3.11	Calibration clock output . . . . .	421
24.3.12	Alarm output . . . . .	422
24.4	RTC low power modes . . . . .	422
24.5	RTC interrupts . . . . .	422
24.6	RTC registers . . . . .	423
24.6.1	Time register 1 (RTC_TR1) . . . . .	423
24.6.2	Time register 2 (RTC_TR2) . . . . .	423
24.6.3	Time register 3 (RTC_TR3) . . . . .	423
24.6.4	Date register 1 (RTC_DR1) . . . . .	424
24.6.5	Date register 2 (RTC_DR2) . . . . .	424
24.6.6	Date register 3 (RTC_DR3) . . . . .	425
24.6.7	Subsecond register high (RTC_SSRH) . . . . .	425
24.6.8	Subsecond register low (RTC_SSRL) . . . . .	426
24.6.9	Control register 1 (RTC_CR1) . . . . .	426
24.6.10	Control register 2 (RTC_CR2) . . . . .	427
24.6.11	Control register 3 (RTC_CR3) . . . . .	428
24.6.12	Initialization and status register 1 (RTC_ISR1) . . . . .	429
24.6.13	Initialization and status register 2 (RTC_ISR2) . . . . .	430
24.6.14	Synchronous prescaler register high (RTC_SPRERH) . . . . .	431
24.6.15	Synchronous prescaler register low (RTC_SPRERL) . . . . .	432
24.6.16	Asynchronous prescaler register (RTC_APERR) . . . . .	432

---

24.6.17	Wakeup timer register high (RTC_WUTRH) . . . . .	433
24.6.18	Wakeup timer register low (RTC_WUTRL) . . . . .	433
24.6.19	Write protection register (RTC_WPR) . . . . .	434
24.6.20	RTC shift control register high (RTC_SHIFTRH) . . . . .	434
24.6.21	RTC shift control register low (RTC_SHIFTRL) . . . . .	435
24.6.22	Alarm A register 1 (RTC_ALRMAR1) . . . . .	435
24.6.23	Alarm A register 2 (RTC_ALRMAR2) . . . . .	436
24.6.24	Alarm A register 3 (RTC_ALRMAR3) . . . . .	436
24.6.25	Alarm A register 4 (RTC_ALRMAR4) . . . . .	437
24.6.26	Alarm A sub second register high (RTC_ALRMASSRH) . . . . .	437
24.6.27	Alarm A sub second register low (RTC_ALRMASSRL) . . . . .	438
24.6.28	Alarm A subsecond masking register (RTC_ALRMASSMSKR) . . . . .	438
24.6.29	Calibration register high (RTC_CALRH) . . . . .	439
24.6.30	Calibration register low (RTC_CALRL) . . . . .	440
24.6.31	Tamper control register 1 (RTC_TCR1) . . . . .	440
24.6.32	Tamper control register 2 (RTC_TCR2) . . . . .	441
24.6.33	RTC register map and reset values . . . . .	442
<b>25</b>	<b>Independent watchdog (IWDG) . . . . .</b>	<b>445</b>
25.1	Introduction . . . . .	445
25.2	IWDG functional description . . . . .	445
25.3	IWDG registers . . . . .	447
25.3.1	Key register (IWDG_KR) . . . . .	447
25.3.2	Prescaler register (IWDG_PR) . . . . .	447
25.3.3	Reload register (IWDG_RLR) . . . . .	448
25.3.4	IWDG register map and reset values . . . . .	448
<b>26</b>	<b>Window watchdog (WWDG) . . . . .</b>	<b>449</b>
26.1	Introduction . . . . .	449
26.2	WWDG main features . . . . .	449
26.3	WWDG functional description . . . . .	449
26.4	How to program the watchdog timeout . . . . .	451
26.5	WWDG low power modes . . . . .	451
26.6	Hardware watchdog option . . . . .	452
26.7	WWDG interrupts . . . . .	452
26.8	WWDG registers . . . . .	452

26.8.1	Control register (WWDG_CR) . . . . .	452
26.8.2	Window register (WWDG_WR) . . . . .	453
26.9	Window watchdog register map and reset values . . . . .	453
<b>27</b>	<b>AES hardware accelerator (AES) . . . . .</b>	<b>454</b>
27.1	Introduction . . . . .	454
27.2	AES main features . . . . .	454
27.3	AES functional description . . . . .	455
27.4	Modes of operation . . . . .	456
27.4.1	Mode 1: encryption . . . . .	456
27.4.2	Mode 2: Key derivation . . . . .	457
27.4.3	Mode 3: decryption . . . . .	457
27.4.4	Mode 4: key derivation and decryption . . . . .	458
27.5	AES DMA interface . . . . .	459
27.6	Error flags . . . . .	460
27.7	Processing time . . . . .	460
27.8	AES low power modes . . . . .	461
27.9	AES interrupts . . . . .	461
27.10	AES registers . . . . .	462
27.10.1	AES control register (AES_CR) . . . . .	462
27.10.2	AES status register (AES_SR) . . . . .	463
27.10.3	AES data input register (AES_DINR) . . . . .	464
27.10.4	AES data output register (AES_DOUTR) . . . . .	464
27.10.5	AES register map and reset values . . . . .	465
<b>28</b>	<b>Inter-integrated circuit (I<sup>2</sup>C) interface . . . . .</b>	<b>466</b>
28.1	Introduction . . . . .	466
28.2	I <sup>2</sup> C main features . . . . .	467
28.3	I <sup>2</sup> C general description . . . . .	468
28.4	I <sup>2</sup> C functional description . . . . .	470
28.4.1	I <sup>2</sup> C slave mode . . . . .	470
28.4.2	I <sup>2</sup> C master mode . . . . .	472
28.4.3	Error conditions . . . . .	479
28.4.4	SDA/SCL line control . . . . .	480
28.4.5	SMBus . . . . .	480

---

28.4.6	DMA requests . . . . .	483
28.4.7	Packet error checking . . . . .	484
28.5	I <sup>2</sup> C low power modes . . . . .	485
28.6	I <sup>2</sup> C interrupts . . . . .	485
28.7	I <sup>2</sup> C registers . . . . .	487
28.7.1	Control register 1 (I2C_CR1) . . . . .	487
28.7.2	Control register 2 (I2C_CR2) . . . . .	488
28.7.3	Frequency register (I2C_FREQR) . . . . .	490
28.7.4	Own address register LSB (I2C_OAR1L) . . . . .	491
28.7.5	Own address register MSB (I2C_OAR1H) . . . . .	491
28.7.6	Own address register 2 (I2C_OAR2) . . . . .	491
28.7.7	Data register (I2C_DR) . . . . .	492
28.7.8	Status register 1 (I2C_SR1) . . . . .	492
28.7.9	Status register 2 (I2C_SR2) . . . . .	494
28.7.10	Status register 3 (I2C_SR3) . . . . .	495
28.7.11	Interrupt and DMA register (I2C_ITR) . . . . .	497
28.7.12	Clock control register low (I2C_CCRL) . . . . .	498
28.7.13	Clock control register high (I2C_CCRH) . . . . .	499
28.7.14	TRISE register (I2C_TRISER) . . . . .	500
28.7.15	PEC register (I2C_PECR) . . . . .	501
28.7.16	I <sup>2</sup> C register map and reset values . . . . .	501
29	<b>Universal synchronous/asynchronous receiver transmitter (USART) . . . . .</b>	<b>503</b>
29.1	USART introduction . . . . .	503
29.2	USART main features . . . . .	503
29.3	USART functional description . . . . .	504
29.3.1	USART character description . . . . .	507
29.3.2	Transmitter . . . . .	508
29.3.3	Receiver . . . . .	511
29.3.4	High precision baud rate generator . . . . .	514
29.3.5	USART receiver's tolerance to clock deviation . . . . .	517
29.3.6	Parity control . . . . .	517
29.3.7	Multi-processor communication . . . . .	519
29.3.8	USART synchronous communication . . . . .	520
29.3.9	Single wire half duplex communication . . . . .	523
29.3.10	Smartcard . . . . .	523

29.3.11	IrDA SIR ENDEC block . . . . .	525
29.3.12	Continuous communication using DMA . . . . .	527
29.4	USART low power modes . . . . .	530
29.5	USART interrupts . . . . .	530
29.6	USART registers . . . . .	531
29.6.1	Status register (USART_SR) . . . . .	531
29.6.2	Data register (USART_DR) . . . . .	533
29.6.3	Baud rate register 1 (USART_BRR1) . . . . .	533
29.6.4	Baud rate register 2 (USART_BRR2) . . . . .	533
29.6.5	Control register 1 (USART_CR1) . . . . .	534
29.6.6	Control register 2 (USART_CR2) . . . . .	535
29.6.7	Control register 3 (USART_CR3) . . . . .	536
29.6.8	Control register 4 (USART_CR4) . . . . .	537
29.6.9	Control register 5 (USART_CR5) . . . . .	537
29.6.10	Guard time register (USART_GTR) . . . . .	538
29.6.11	Prescaler register (USART_PSCR) . . . . .	539
29.6.12	USART register map and reset values . . . . .	539
<b>30</b>	<b>Serial peripheral interface (SPI) . . . . .</b>	<b>541</b>
30.1	Introduction . . . . .	541
30.2	SPI main features . . . . .	541
30.3	SPI functional description . . . . .	542
30.3.1	General description . . . . .	542
30.3.2	Configuring the SPI in slave mode . . . . .	546
30.3.3	Configuring the SPI master mode . . . . .	546
30.3.4	Configuring the SPI for simplex communications . . . . .	547
30.3.5	Data transmission and reception procedures . . . . .	547
30.3.6	CRC calculation . . . . .	554
30.3.7	Status flags . . . . .	555
30.3.8	Disabling the SPI . . . . .	556
30.3.9	SPI communication using DMA (direct memory addressing) . . . . .	558
30.3.10	Error flags . . . . .	559
30.3.11	SPI low power modes . . . . .	560
30.3.12	SPI interrupts . . . . .	561
30.4	SPI registers . . . . .	561
30.4.1	SPI control register 1 (SPI_CR1) . . . . .	561

---

30.4.2	SPI control register 2 (SPI_CR2) . . . . .	562
30.4.3	SPI interrupt control register (SPI_ICR) . . . . .	563
30.4.4	SPI status register (SPI_SR) . . . . .	564
30.4.5	SPI data register (SPI_DR) . . . . .	565
30.4.6	SPI CRC polynomial register (SPI_CRCPR) . . . . .	565
30.4.7	SPI Rx CRC register (SPI_RXCRCR) . . . . .	565
30.4.8	SPI Tx CRC register (SPI_TXCRCR) . . . . .	566
30.5	SPI register map and reset values . . . . .	566
<b>31</b>	<b>Revision history</b> . . . . .	<b>567</b>

## List of tables

Table 1.	Applicable products .....	1
Table 2.	Interrupt levels .....	33
Table 3.	CPU register map .....	34
Table 4.	CFG_GCR register map .....	35
Table 5.	Block size .....	52
Table 6.	Memory access versus programming method (low density devices) .....	53
Table 7.	Memory access versus programming method (medium density devices) .....	54
Table 8.	Memory access versus programming method (medium+ and high density devices) .....	55
Table 9.	Flash register map .....	60
Table 10.	List of abbreviations .....	63
Table 11.	Internal voltage reference status during Halt/Active-halt mode .....	68
Table 12.	PWR interrupt requests .....	69
Table 13.	PWR register map and reset values .....	71
Table 14.	Low power mode summary .....	72
Table 15.	WFE register map .....	80
Table 16.	RST register map and reset values .....	87
Table 17.	CSS on LSE in low power modes .....	98
Table 18.	CSS on LSE register map .....	99
Table 19.	CLK interrupt requests .....	100
Table 20.	Peripheral clock gating bits (PCKEN 10 to PCKEN 17) .....	104
Table 21.	Peripheral clock gating bits ((PCKEN 20 to PCKEN 27)) .....	105
Table 22.	Peripheral clock gating bits (PCKEN 30 to PCKEN 35) .....	106
Table 23.	CLK register map and reset values .....	115
Table 24.	I/O port configuration summary .....	118
Table 25.	Effect of low power modes on GPIO ports .....	119
Table 26.	GPIO register map .....	123
Table 27.	I/O groups and selection .....	127
Table 28.	TIM1 input capture routing .....	129
Table 29.	RI interrupt requests .....	132
Table 30.	Routing interface register map .....	148
Table 31.	Register map .....	153
Table 32.	Software priority levels .....	155
Table 33.	Vector address map versus software priority bits .....	160
Table 34.	External interrupt sensitivity .....	161
Table 35.	Dedicated interrupt instruction set .....	161
Table 36.	ITC and EXTI register map .....	171
Table 37.	Source and destination addresses .....	181
Table 38.	DMA1 channel request mapping .....	182
Table 39.	SPIx/USARTx DMA requests .....	184
Table 40.	I2C1 DMA requests .....	185
Table 41.	TIMx DMA requests .....	185
Table 42.	DMA behavior in low power modes .....	185
Table 43.	DMA interrupt requests .....	186
Table 44.	DMA register map .....	194
Table 45.	Flag/interrupt configuration .....	201
Table 46.	Authorized sampling cycles .....	203
Table 47.	Behavior in low power modes .....	205

---

Table 48.	Interrupt requests . . . . .	205
Table 49.	Register map . . . . .	216
Table 50.	DAC output switch configuration. . . . .	219
Table 51.	DAC output switch configuration (medium and medium+ and high density devices) . . . . .	219
Table 52.	Sources of conversion trigger (medium, medium+ and high density devices) . . . . .	221
Table 53.	Sources of conversion trigger . . . . .	221
Table 54.	DAC register map (medium, medium+ and high density devices) . . . . .	235
Table 55.	Window interrupts/wakeup modes . . . . .	244
Table 56.	Behavior of comparators in low power modes . . . . .	244
Table 57.	Comparator interrupt requests . . . . .	244
Table 58.	Comparators and routing Interface register map . . . . .	249
Table 59.	Typical frame rate calculation for input frequency of 16.384 kHz . . . . .	256
Table 60.	Typical frame rate calculation for input frequency of 500 kHz . . . . .	257
Table 61.	SEG and COM signal mapping. . . . .	267
Table 62.	LCD behavior in low power modes . . . . .	270
Table 63.	LCD interrupt requests . . . . .	271
Table 64.	LCD register map . . . . .	279
Table 65.	Timer characteristics. . . . .	281
Table 66.	Timer feature comparison . . . . .	282
Table 67.	Glossary of internal timer signals . . . . .	282
Table 68.	Explanation of indices 'i', 'n', and 'x' . . . . .	283
Table 69.	Counting direction versus encoder signals . . . . .	330
Table 70.	Output control for complementary OCi and OCiN channels with break feature. . . . .	354
Table 71.	TIM1 register map. . . . .	367
Table 72.	TIMx internal trigger connection . . . . .	377
Table 73.	Output control bit for OCi channels with break feature (break feature implemented, complementary output not implemented) . . . . .	393
Table 74.	TIMx register map. . . . .	394
Table 75.	TIM4 register map. . . . .	403
Table 76.	IR register map . . . . .	405
Table 77.	BEEP register map . . . . .	409
Table 78.	Effect of low power modes on RTC . . . . .	422
Table 79.	Interrupt control bits . . . . .	422
Table 80.	RTC register map and reset values . . . . .	442
Table 81.	Min/Max IWDG timeout (LSI clock frequency = 38 kHz) . . . . .	446
Table 82.	IWDG register map . . . . .	448
Table 83.	Window watchdog timing example . . . . .	451
Table 84.	Effect of low power modes on WWDG . . . . .	451
Table 85.	WWDG register map and reset values . . . . .	453
Table 86.	Processing time (in clock cycles) . . . . .	460
Table 87.	Effect of low power modes on AES . . . . .	461
Table 88.	AES interrupt requests . . . . .	461
Table 89.	AES register map and reset values . . . . .	465
Table 90.	SMBus vs. I2C . . . . .	481
Table 91.	I <sup>2</sup> C interface behavior in low power modes . . . . .	485
Table 92.	I <sup>2</sup> C Interrupt requests . . . . .	485
Table 93.	I2C_CCR values for SCL frequency table (fSYSCLK= 10 MHz or 16 MHz) . . . . .	500
Table 94.	I <sup>2</sup> C register map . . . . .	501
Table 95.	Noise detection from sampled data . . . . .	513
Table 96.	Baud rate programming and error calculation . . . . .	515

Table 97.	Baud rate programming and error calculation .....	516
Table 98.	Baud rate programming and error calculation .....	516
Table 99.	USART receiver's tolerance when USART_DIV[3:0] is 0 .....	517
Table 100.	USART receiver's tolerance when USART_DIV[3:0] is different from 0 .....	517
Table 101.	Frame formats .....	517
Table 102.	USART interface behavior in low power modes .....	530
Table 103.	USART interrupt requests .....	530
Table 104.	USART register map .....	539
Table 105.	SPI behavior in low power modes .....	560
Table 106.	SPI interrupt requests .....	561
Table 107.	SPI register map and reset values .....	566
Table 108.	Document revision history .....	567

# List of figures

Figure 1.	Programming model . . . . .	31
Figure 2.	Stacking order. . . . .	32
Figure 3.	Low density Flash program and data EEPROM . . . . .	39
Figure 4.	Medium density Flash program and data EEPROM organization . . . . .	40
Figure 5.	Medium+ density Flash program and data EEPROM organization . . . . .	41
Figure 6.	High density Flash program and data EEPROM organization . . . . .	42
Figure 7.	UBC area size definition for low density devices . . . . .	44
Figure 8.	UBC area size definition for medium density devices. . . . .	45
Figure 9.	UBC area size definition for medium + STM8L05xx/15xx devices . . . . .	45
Figure 10.	UBC area size definition for high density devices. . . . .	46
Figure 11.	SWIM pin connection . . . . .	61
Figure 12.	Power supply overview . . . . .	64
Figure 13.	Power on reset/power down reset waveform . . . . .	66
Figure 14.	VDD voltage detection: BOR threshold . . . . .	67
Figure 15.	PVD threshold. . . . .	68
Figure 16.	Reset circuit . . . . .	84
Figure 17.	Clock structure . . . . .	88
Figure 18.	HSE clock sources . . . . .	90
Figure 19.	Clock switching flowchart (automatic mode example) . . . . .	94
Figure 20.	Clock switching flowchart (manual mode example without interrupt) . . . . .	95
Figure 21.	Clock switching flowchart (manual mode example with interrupt) . . . . .	95
Figure 22.	GPIO block diagram . . . . .	117
Figure 23.	Routing interface (RI) block diagram (medium, medium+ and high density devices) . . . . .	125
Figure 24.	Routing interface (RI) block diagram (low density devices) . . . . .	126
Figure 25.	TIM2 and TIM3 interconnections . . . . .	130
Figure 26.	DAC interconnections . . . . .	131
Figure 27.	Internal reference voltage output . . . . .	132
Figure 28.	Interrupt processing flowchart . . . . .	155
Figure 29.	Priority decision process . . . . .	156
Figure 30.	Concurrent interrupt management . . . . .	159
Figure 31.	Nested interrupt management . . . . .	160
Figure 32.	DMA block diagram . . . . .	174
Figure 33.	Endianess in 16-bit mode . . . . .	176
Figure 34.	Regular channel (medium density devices) . . . . .	178
Figure 35.	Regular channel (medium+ and high density devices) . . . . .	178
Figure 36.	Memory channel with MEM=0 (medium density devices) . . . . .	179
Figure 37.	Memory channel with MEM=0 (medium+ and high density devices) . . . . .	179
Figure 38.	Memory channel with MEM=1 (medium density devices) . . . . .	180
Figure 39.	Memory channel with MEM=1 (medium+ and high density devices) . . . . .	180
Figure 40.	DMA1 request mapping (medium density devices) . . . . .	183
Figure 41.	DMA1 request mapping (medium+ and high density devices) . . . . .	183
Figure 42.	ADC block diagram. . . . .	197
Figure 43.	ADC external triggers . . . . .	200
Figure 44.	Sequence example . . . . .	202
Figure 45.	Resolution and data alignment . . . . .	203
Figure 46.	DAC channel block diagram (medium density devices) . . . . .	218
Figure 47.	DAC channel block diagram (medium+ and high density devices) . . . . .	218
Figure 48.	DAC LFSR register calculation algorithm . . . . .	222

---

Figure 49.	DAC triangle wave generation .....	223
Figure 50.	Comparator block diagram .....	239
Figure 51.	COMP1 interconnections .....	241
Figure 52.	COMP2 interconnections .....	242
Figure 53.	COMP2 output redirection .....	243
Figure 54.	Configuring the comparators in window mode .....	243
Figure 55.	Medium + and high density LCD controller block diagram .....	253
Figure 56.	Medium density LCD controller block diagram .....	254
Figure 57.	1/3 bias, 1/4 duty .....	258
Figure 58.	Static duty .....	260
Figure 59.	Liquid crystal display and terminal connection (static duty) .....	261
Figure 60.	Liquid crystal display and terminal connection (1/2 duty, 1/2 bias) .....	262
Figure 61.	Liquid crystal display and terminal connection (1/3 duty, 1/3 bias) .....	263
Figure 62.	Liquid crystal display and terminal connection (1/4 duty, 1/3 bias) .....	264
Figure 63.	Liquid crystal display and terminal connection (1/8 duty, 1/4 bias) .....	265
Figure 64.	Resistive network .....	268
Figure 65.	Contrast control .....	269
Figure 66.	SOF interrupt signal .....	271
Figure 67.	TIM1 general block diagram .....	286
Figure 68.	Time base unit .....	287
Figure 69.	16-bit read sequence for the counter (TIM1_CNTR) .....	288
Figure 70.	Counter in up-counting mode .....	289
Figure 71.	Counter update when ARPE = 0 (ARR not preloaded) with prescaler = 2 .....	290
Figure 72.	Counter update event when ARPE = 1 (TIM1_ARR preloaded) .....	290
Figure 73.	Counter in down-counting mode .....	291
Figure 74.	Counter update when ARPE = 0 (ARR not preloaded) with prescaler = 2 .....	292
Figure 75.	Counter update when ARPE = 1 (ARR preloaded), with prescaler = 1 .....	292
Figure 76.	Counter in center-aligned mode .....	293
Figure 77.	Counter timing diagram, $f_{CK\_CNT} = f_{CK\_PSC}$ , TIM1_ARR = 06h, ARPE = 1 .....	294
Figure 78.	Update rate examples depending on mode and TIM1_RCR register settings .....	296
Figure 79.	Clock/trigger controller block diagram .....	297
Figure 80.	Control circuit in normal mode, $f_{CK\_PSC} = f_{SYSCLK}$ .....	298
Figure 81.	TI2 external clock connection example .....	298
Figure 82.	Control circuit in external clock mode 1 .....	299
Figure 83.	External trigger input block diagram .....	300
Figure 84.	Control circuit in external clock mode 2 .....	300
Figure 85.	Control circuit in trigger mode .....	301
Figure 86.	Control circuit in trigger reset mode .....	302
Figure 87.	Control circuit in trigger gated mode .....	303
Figure 88.	Control circuit in external clock mode 2 + trigger mode .....	304
Figure 89.	Timer chaining system implementation example .....	305
Figure 90.	Trigger/master mode selection blocks .....	306
Figure 91.	Master/slave timer example .....	306
Figure 92.	Gating timer B with OC1REF of timer A .....	307
Figure 93.	Gating timer B with the counter enable signal of timer A (CNT_EN) .....	308
Figure 94.	Triggering timer B with the UEV of timer A (TIMERA-UEV) .....	309
Figure 95.	Triggering timer B with counter enable CNT_EN of timer A .....	310
Figure 96.	Triggering Timer A and B with Timer A TI1 input .....	311
Figure 97.	Capture/compare channel 1 main circuit .....	311
Figure 98.	16-bit read sequence for the TIM1_CCRi register in capture mode .....	312
Figure 99.	Channel input stage block diagram .....	313
Figure 100.	Input stage of TIM 1 channel 1 .....	313

---

Figure 101. PWM input signal measurement . . . . .	315
Figure 102. PWM input signal measurement example . . . . .	316
Figure 103. Channel output stage block diagram . . . . .	316
Figure 104. Detailed output stage of channel with complementary output (channel 1) . . . . .	317
Figure 105. Output compare mode, toggle on OC1 . . . . .	318
Figure 106. Edge-aligned counting mode PWM mode 1 waveforms (ARR = 8) . . . . .	320
Figure 107. Center-aligned PWM waveforms (ARR = 8) . . . . .	321
Figure 108. Example of one-pulse mode . . . . .	322
Figure 109. Complementary output with deadtime insertion . . . . .	324
Figure 110. Deadtime waveforms with a delay greater than the negative pulse . . . . .	324
Figure 111. Deadtime waveforms with a delay greater than the positive pulse . . . . .	324
Figure 112. Six-step generation, COM example (OSSR = 1) . . . . .	326
Figure 113. Behavior of outputs in response to a break (channel without complementary output) . . . . .	328
Figure 114. Behavior of outputs in response to a break (TIM1 complementary outputs) . . . . .	328
Figure 115. OCREFCLR activation . . . . .	329
Figure 116. Example of counter operation in encoder interface mode . . . . .	331
Figure 117. Example of encoder interface mode with IC1 polarity inverted . . . . .	331
Figure 118. Example of Hall sensor interface . . . . .	333
Figure 119. DMA burst modes example . . . . .	335
Figure 120. TIMx block diagram . . . . .	370
Figure 121. Time base unit . . . . .	371
Figure 122. Input stage block diagram . . . . .	372
Figure 123. Input stage of TIM 2 channel 1 . . . . .	372
Figure 124. Output stage . . . . .	373
Figure 125. Output stage of channel 1 . . . . .	373
Figure 126. TIM4 block diagram . . . . .	396
Figure 127. IR internal hardware connections with TIM2 and TIM3 . . . . .	404
Figure 128. BEEP block diagram . . . . .	406
Figure 129. RTC block diagram (low density devices) . . . . .	413
Figure 130. RTC block diagram (medium density devices) . . . . .	413
Figure 131. RTC block diagram (medium+ and high density devices) . . . . .	414
Figure 132. Independent watchdog (IWDG) block diagram . . . . .	445
Figure 133. Watchdog block diagram . . . . .	450
Figure 134. Window watchdog timing diagram . . . . .	451
Figure 135. Block diagram . . . . .	455
Figure 136. Mode 1: encryption . . . . .	456
Figure 137. Mode 2: key derivation . . . . .	457
Figure 138. Mode 3: decryption . . . . .	458
Figure 139. Mode 4: key derivation and decryption . . . . .	458
Figure 140. DMA requests and data transfers during input phase (AES_IN) . . . . .	459
Figure 141. DMA requests during output phase (AES_OUT) . . . . .	460
Figure 142. I <sup>2</sup> C bus protocol . . . . .	468
Figure 143. I <sup>2</sup> C block diagram . . . . .	469
Figure 144. Transfer sequence diagram for slave transmitter . . . . .	471
Figure 145. Transfer sequence diagram for slave receiver . . . . .	472
Figure 146. Transfer sequence diagram for master transmitter . . . . .	474
Figure 147. Method 1: transfer sequence diagram for master receiver . . . . .	475
Figure 148. Method 2: transfer sequence diagram for master receiver when N >2 . . . . .	476
Figure 149. Method 2: transfer sequence diagram for master receiver when N=2 . . . . .	478
Figure 150. Method 2: transfer sequence diagram for master receiver when N=1 . . . . .	478
Figure 151. I <sup>2</sup> C interrupt mapping . . . . .	486
Figure 152. USART block diagram . . . . .	506

---

Figure 153. Word length programming . . . . .	507
Figure 154. Configurable STOP bits . . . . .	509
Figure 155. TC/TXE behavior when transmitting . . . . .	510
Figure 156. Start bit detection . . . . .	511
Figure 157. Data sampling for noise detection . . . . .	513
Figure 158. How to code USART_DIV in the BRR registers . . . . .	515
Figure 159. Mute mode using Idle line detection . . . . .	519
Figure 160. Mute mode using address mark detection . . . . .	520
Figure 161. USART example of synchronous transmission. . . . .	521
Figure 162. USART data clock timing diagram (M=0) . . . . .	522
Figure 163. USART data clock timing diagram (M=1) . . . . .	522
Figure 164. RX data setup/hold time . . . . .	522
Figure 165. ISO 7816-3 asynchronous protocol . . . . .	523
Figure 166. Parity error detection using 1.5 STOP bits . . . . .	525
Figure 167. IrDA SIR ENDEC- block diagram . . . . .	527
Figure 168. IrDA data modulation (3/16) - normal mode . . . . .	527
Figure 169. Transmission using DMA . . . . .	528
Figure 170. Reception using DMA . . . . .	529
Figure 171. USART interrupt mapping diagram . . . . .	531
Figure 172. SPI block diagram. . . . .	542
Figure 173. Single master/ single slave application. . . . .	543
Figure 174. Data clock timing diagram . . . . .	545
Figure 175. TXE/RXNE/BSY behavior in full duplex mode (RXONLY = 0). Case of continuous transfers . . . . .	550
Figure 176. TXE/RXNE/BSY behavior in slave / full duplex mode (BDM = 0, RXONLY = 0). Case of continuous transfers. . . . .	550
Figure 177. TXE/BSY in master transmit-only mode (BDM = 0 and RXONLY = 0). Case of continuous transfers. . . . .	551
Figure 178. TXE/BSY in slave transmit-only mode (BDM = 0 and RXONLY = 0). Case of continuous transfers . . . . .	552
Figure 179. RXNE behavior in receive-only mode (BDM = 0 and RXONLY = 1). Case of continuous transfers . . . . .	553
Figure 180. TXE/BSY behavior when transmitting (BDM = 0 and RXONLY = 0). Case of discontinuous transfers . . . . .	554
Figure 181. Transmission using DMA . . . . .	558
Figure 182. Reception using DMA . . . . .	559

# 1 Central processing unit (CPU)

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

## 1.1 Introduction

The CPU has an 8-bit architecture. Six internal registers allow efficient data manipulations. The CPU is able to execute 80 basic instructions. It features 20 addressing modes and can address six internal registers. For the complete description of the instruction set, refer to the STM8 microcontroller family programming manual (PM0044).

## 1.2 CPU registers

The six CPU registers are shown in the programming model in [Figure 1](#). Following an interrupt, the registers are pushed onto the stack in the order shown in [Figure 2](#). They are popped from stack in the reverse order. The interrupt routine must therefore handle it, if needed, through the POP and PUSH instructions.

### 1.2.1 Description of CPU registers

#### Accumulator (A)

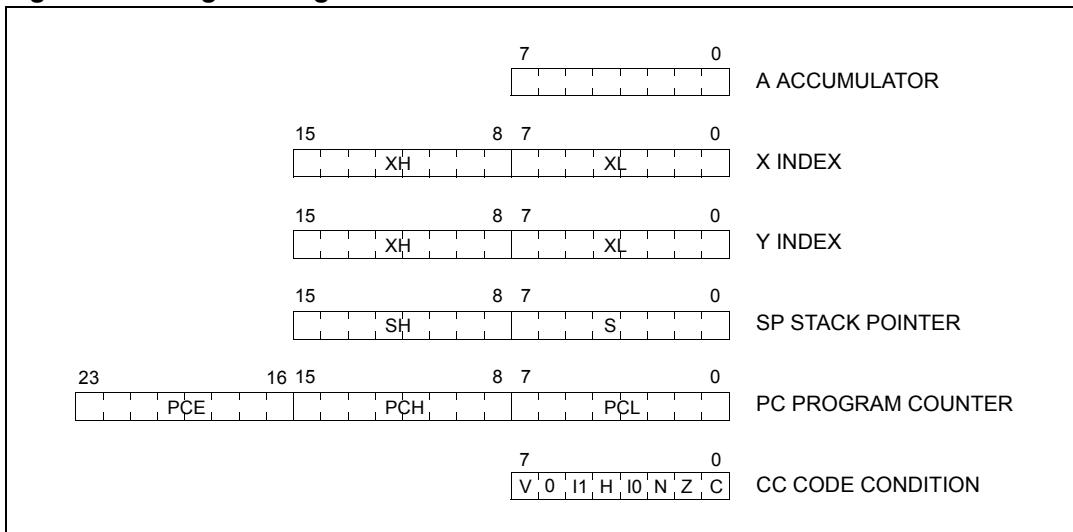
The accumulator is an 8-bit general purpose register used to hold operands and the results of the arithmetic and logic calculations as well as data manipulations.

#### Index registers (X and Y)

These are 16-bit registers used to create effective addresses. They may also be used as a temporary storage area for data manipulations and have an inherent use for some instructions (multiplication/division). In most cases, the cross assembler generates a PRECODE instruction (PRE) to indicate that the following instruction refers to the Y register.

#### Program counter (PC)

The program counter is a 24-bit register used to store the address of the next instruction to be executed by the CPU. It is automatically refreshed after each processed instruction. As a result, the STM8 core can access up to 16 Mbytes of memory.

**Figure 1. Programming model**

### Stack pointer (SP)

The stack pointer is a 16-bit register. It contains the address of the next free location of the stack. Depending on the product, the most significant bits can be forced to a preset value.

The stack is used to save the CPU context on subroutine calls or interrupts. The user can also directly use it through the POP and PUSH instructions.

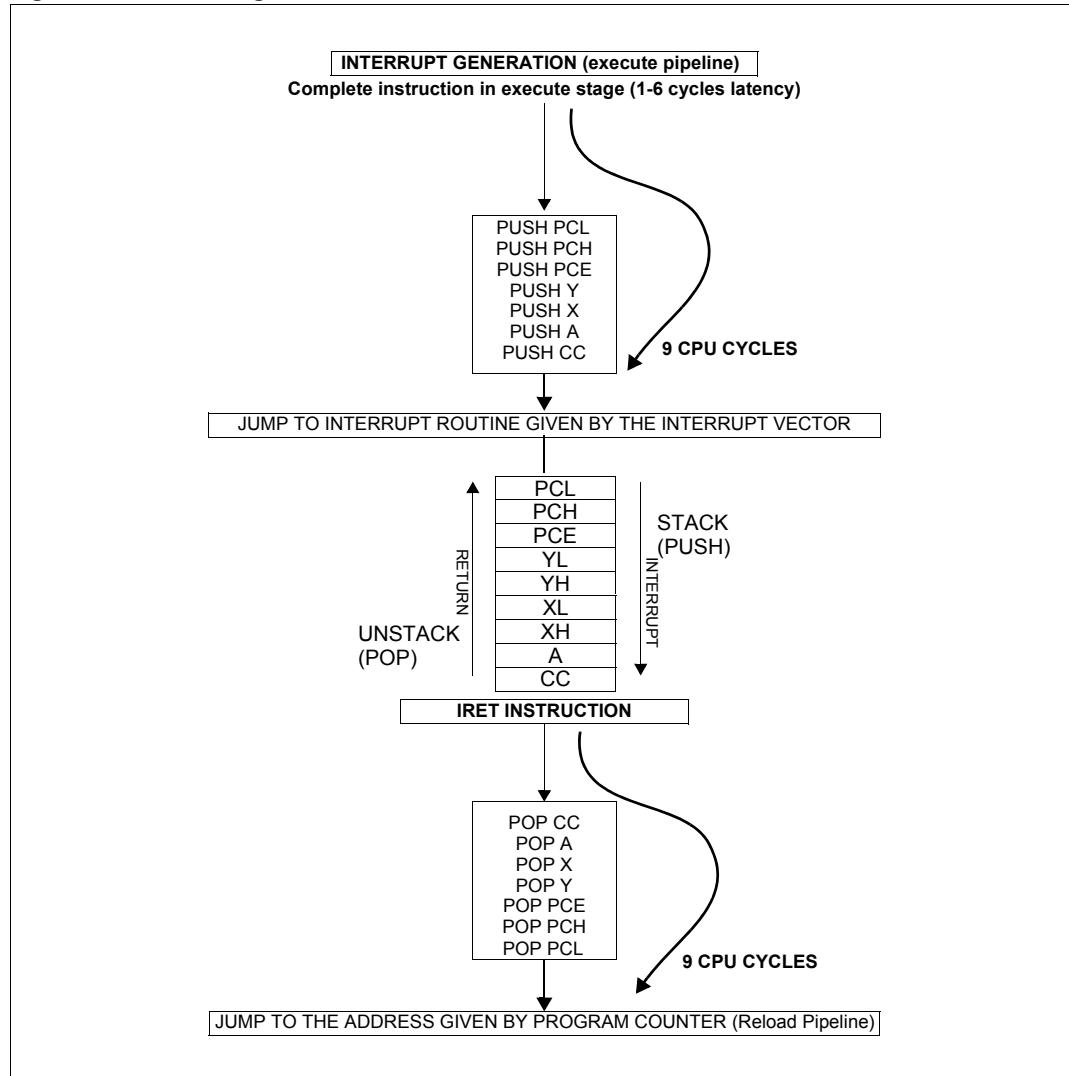
The stack pointer can be initialized by the startup function provided with the C compiler. For applications written in C language, the initialization is then performed according to the address specified in the linker file for C users. If you use your own linker file or startup file, make sure the stack pointer is initialized properly (with the address given in the datasheets). For applications written in assembler, you can use either the startup function provided by ST or write your own by initializing the stack pointer with the correct address.

The stack pointer is decremented after data has been pushed onto the stack and incremented after data is popped from the stack. It is up to the application to ensure that the lower limit is not exceeded.

A subroutine call occupies two or three locations. An interrupt occupies nine locations to store all the internal registers (except SP). For more details refer to [Figure 2](#).

**Note:** *The WFI/HALT instructions save the context in advance. If an interrupt occurs while the CPU is in one of these modes, the latency is reduced.*

Figure 2. Stacking order



### Condition code register (CC)

The condition code register is an 8-bit register which indicates the result of the instruction just executed as well as the state of the processor. The 6th bit (MSB) of this register is reserved. These bits can be individually tested by a program and specified action taken as a result of their state. The following paragraphs describe each bit:

- V: Overflow

When set, V indicates that an overflow occurred during the last signed arithmetic operation, on the MSB result bit. See the INC, INCW, DEC, DECW, NEG, NEGW, ADD, ADDW, ADC, SUB, SUBW, SBC, CP, and CPW instructions.

- I1: Interrupt mask level 1

The I1 flag works in conjunction with the I0 flag to define the current interruptability level as shown in [Table 2](#). These flags can be set and cleared by software through the RIM, SIM, HALT, WFI, WFE, IRET, TRAP, and POP instructions and are automatically set by hardware when entering an interrupt service routine.

**Table 2. Interrupt levels**

Interruptability	Priority	I1	I0
Interruptable main	Lowest	1	0
Interruptable level 1		0	1
Interruptable level 2		0	0
Non interruptable	Highest	1	1

- H: Half carry bit

The H bit is set to 1 when a carry occurs between the bits 3 and 4 of the ALU during an ADD or ADC instruction. The H bit is useful in BCD arithmetic subroutines.

- I0: Interrupt mask level 0

See Flag I1.

- N: Negative

When set to 1, this bit indicates that the result of the last arithmetic, logical or data manipulation is negative (i.e. the most significant bit is a logic 1).

- Z: Zero

When set to 1, this bit indicates that the result of the last arithmetic, logical or data manipulation is zero.

- C: Carry

When set, C indicates that a carry or borrow out of the ALU occurred during the last arithmetic operation on the MSB operation result bit. This bit is also affected during bit test, branch, shift, rotate and load instructions. See the ADD, ADC, SUB, and SBC instructions.

In a division operation, C indicates if trouble occurred during execution (quotient overflow or zero division). See the DIV instruction.

In bit test operations, C is the copy of the tested bit. See the BTJF and BTJT instructions.

In shift and rotate operations, the carry is updated. See the RRC, RLC, SRL, SLL, and SRA instructions.

This bit can be set, reset or complemented by software using the SCF, RCF, and CCF instructions.

**Example: Addition**

$$\$B5 + \$94 = "C" + \$49 = \$149$$

C	7	0
0	1   0   1   1   0   1   0   1	
+ 0	1   0   0   1   0   1   0   0	
= 1	0   1   0   0   1   0   0   1	

**1.2.2 STM8 CPU register map**

The CPU registers are mapped in the STM8 address space as shown in [Table 3](#). These registers can only be accessed by the debug module but not by memory access instructions executed in the core.

**Table 3. CPU register map**

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	A	MSB	-	-	-	-	-	-	LSB
0x01	PCE	MSB	-	-	-	-	-	-	LSB
0x02	PCH	MSB	-	-	-	-	-	-	LSB
0x03	PCL	MSB	-	-	-	-	-	-	LSB
0x04	XH	MSB	-	-	-	-	-	-	LSB
0x05	XL	MSB	-	-	-	-	-	-	LSB
0x06	YH	MSB	-	-	-	-	-	-	LSB
0x07	YL	MSB	-	-	-	-	-	-	LSB
0x08	SPH	MSB	-	-	-	-	-	-	LSB
0x09	SPL	MSB	-	-	-	-	-	-	LSB
0x0A	CC	V	0	I1	H	I0	N	Z	C

**1.3 Global configuration register (CFG\_GCR)****1.3.1 Activation level**

The MCU activation level is configured by programming the AL bit in the CFG\_GCR register.

For information on the use of this bit refer to [Section 12.4: Activation level/low power mode control on page 158](#).

### 1.3.2 SWIM disable

By default, after an MCU reset, the SWIM pin is configured to allow communication with an external tool for debugging or Flash/EEPROM programming. This pin can be configured by the application for use as a general purpose I/O. This is done by setting the SWD bit in the CFG\_GCR register.

### 1.3.3 Description of global configuration register (CFG\_GCR)

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved						AL	SWD
						rw	rw

Bits 7:2 Reserved

#### Bit 1 AL: Activation level

This bit is set and cleared by software. It configures main or interrupt-only activation.

0: Main activation level. An IRET instruction causes the context to be retrieved from the stack and the main program continues after the WFI instruction.

1: Interrupt-only activation level. An IRET instruction causes the CPU to go back to WFI/Halt mode without restoring the context.

#### Bit 0 SWD: SWIM disable

0: SWIM mode enabled

1: SWIM mode disabled

When SWIM mode is enabled, the SWIM pin cannot be used as general purpose I/O.

### 1.3.4 Global configuration register map and reset values

The CFG\_GCR is mapped in the STM8 address space. Refer to the corresponding datasheets for the base address.

**Table 4. CFG\_GCR register map**

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	CFG_GCR Reset value	- 0	- 0	- 0	- 0	- 0	- 0	AL 0	SWD 0

## 2 Boot ROM

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

The internal 2 Kbyte boot ROM contains the bootloader code. Its main task is to download the application program to the internal Flash/EEPROM through the USART1, USART2, USART3 (USARTs in asynchronous mode), SPI1 or SPI2 interfaces and program the code, data, option bytes and interrupt vectors in the internal Flash/EEPROM. Refer to product datasheets for details on available communication ports.

To perform bootloading in LIN mode, a different bootloader communication protocol is implemented on UART2/UART3 and USART1.

The boot loader starts executing after reset. Refer to the STM8 bootloader user manual (UM0560) for more details.

## 3 Flash program memory and data EEPROM

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

### 3.1 Introduction

The embedded Flash program memory and data EEPROM memories are controlled by a common set of registers. Using these registers, the application can program or erase memory contents and set write protection. The application can also program the device option bytes.

### 3.2 Glossary

- Block  
A block is a set of bytes that can be programmed or erased in one single programming operation. Operations that are performed at block level are faster than standard programming and erasing. Refer to [Table 5](#) for the details on block size.
- Page  
A page is a set of blocks.  
Dedicated option bytes can be used to configure, by increments of one page, the size of the user boot code and proprietary code (when available).

### 3.3 Main Flash memory features

- **Low density device EEPROM** is divided into three memory arrays (see [Section 3.4: Memory organization](#) for details on the memory mapping):
  - Up to 8 Kbytes of embedded Flash program
  - 256 Bytes of data EEPROM
  - Up to 64 option bytes (one block)

Flash program and data EEPROM areas are controlled by a common set of registers, allowing the memory content to be programmed or erased, and write protection to be activated.

No RWW is supported on those devices.
- **Medium and medium+ density device EEPROM** is divided into three memory arrays (see [Section 3.4: Memory organization](#) for details on the memory mapping):
  - Up to 32 Kbytes of embedded Flash program
  - Up to 1 Kbyte of data EEPROM
  - Up to 128 option bytes (one block)

Flash program and data EEPROM areas are controlled by a common set of registers, allowing the memory content to be programmed or erased, and write protection to be activated.
- **High density device EEPROM** is divided into three memory arrays (see [Section 3.4: Memory organization](#) for details on the memory mapping):
  - 64 Kbytes of embedded Flash program
  - Up to 2 Kbyte of data EEPROM
  - Up to 128 option bytes (one block)

Flash program and data EEPROM areas are controlled by a common set of registers, allowing the memory content to be programmed or erased, and write protection to be activated.
- Programming modes
  - Byte programming and automatic fast byte programming (without erase operation)
  - Word programming
  - Block programming and fast block programming mode (without erase operation)
  - Interrupt generation on end of program/erase operation and on illegal program operation.
- Read-while-write capability (RWW)
- In-application programming (IAP) and in-circuit programming (ICP) capabilities
- Protection features
  - Memory readout protection (ROP)
  - Program memory write protection with memory access security system (MASS keys)
  - Data memory write protection with memory access security system (MASS keys)
  - Programmable write protected user boot code area (UBC).
  - Automatic readout protection of proprietary code area when available
- Memory state configurable to operating or power-down mode ( $I_{DDQ}$ ) in wait mode (Low power wait) and Run mode (Low power run)

## 3.4 Memory organization

### 3.4.1 Low density device memory organization

Low density EEPROM is split into three memory arrays:

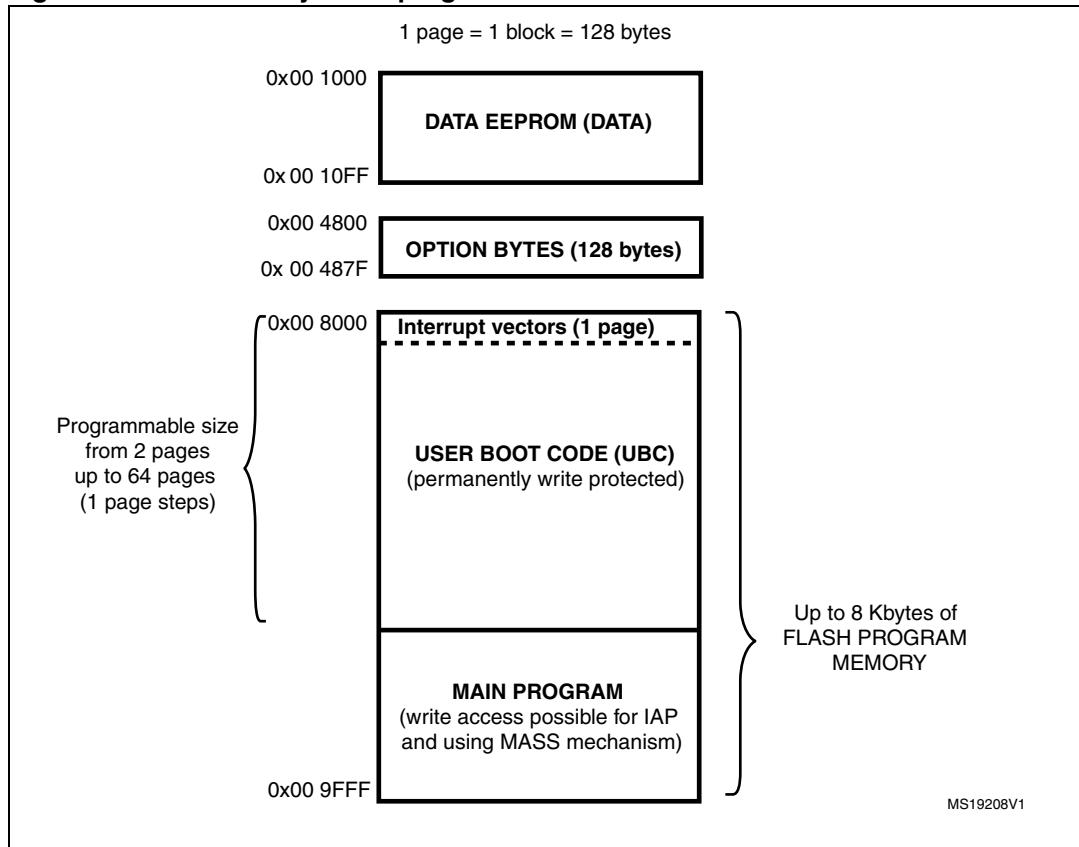
- Up to 8 Kbytes of Flash program memory divided into 128 pages of 64 bytes each. The memory array is organized in 32-bit words (4 bytes per word). It includes two areas:
  - The user boot code area (UBC)
  - The main program area

The first block (128 bytes) of the Flash program memory (starting from address 0x00 8000) contains the interrupt vectors.

- 256 bytes of data EEPROM organized in 4 pages of 64 bytes each.
- One block of option bytes (64 bytes) located in a separate memory array.

See the following figure for a description of the memory organization.

**Figure 3. Low density Flash program and data EEPROM**



### 3.4.2 Medium density device memory organization

Medium density EEPROM is split into three memory arrays:

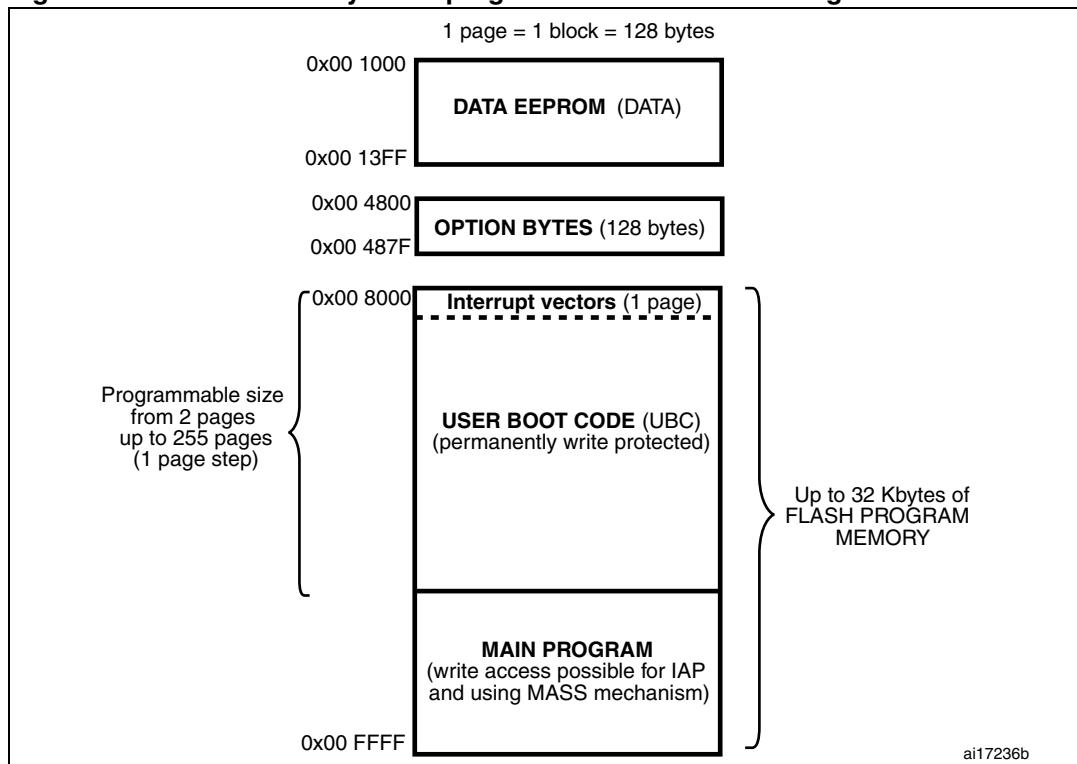
- Up to 32 Kbytes of Flash program memory divided into 256 pages of 128 bytes each. The memory array is organized in 32-bit words (4 bytes per word). It includes two areas:
  - The user boot code area (UBC)
  - The main program area

The first block (128 bytes) of the Flash program memory (starting from address 0x00 8000) contains the interrupt vectors.

- Up to 1 Kbyte of data EEPROM organized in 8 pages of 128 bytes each.
- One block of option bytes (128 bytes) located in a separate memory array.

See the following figure for a description of the memory organization.

**Figure 4. Medium density Flash program and data EEPROM organization**



### 3.4.3 Medium+ density device memory organization

Medium+ density EEPROM is split into three memory arrays:

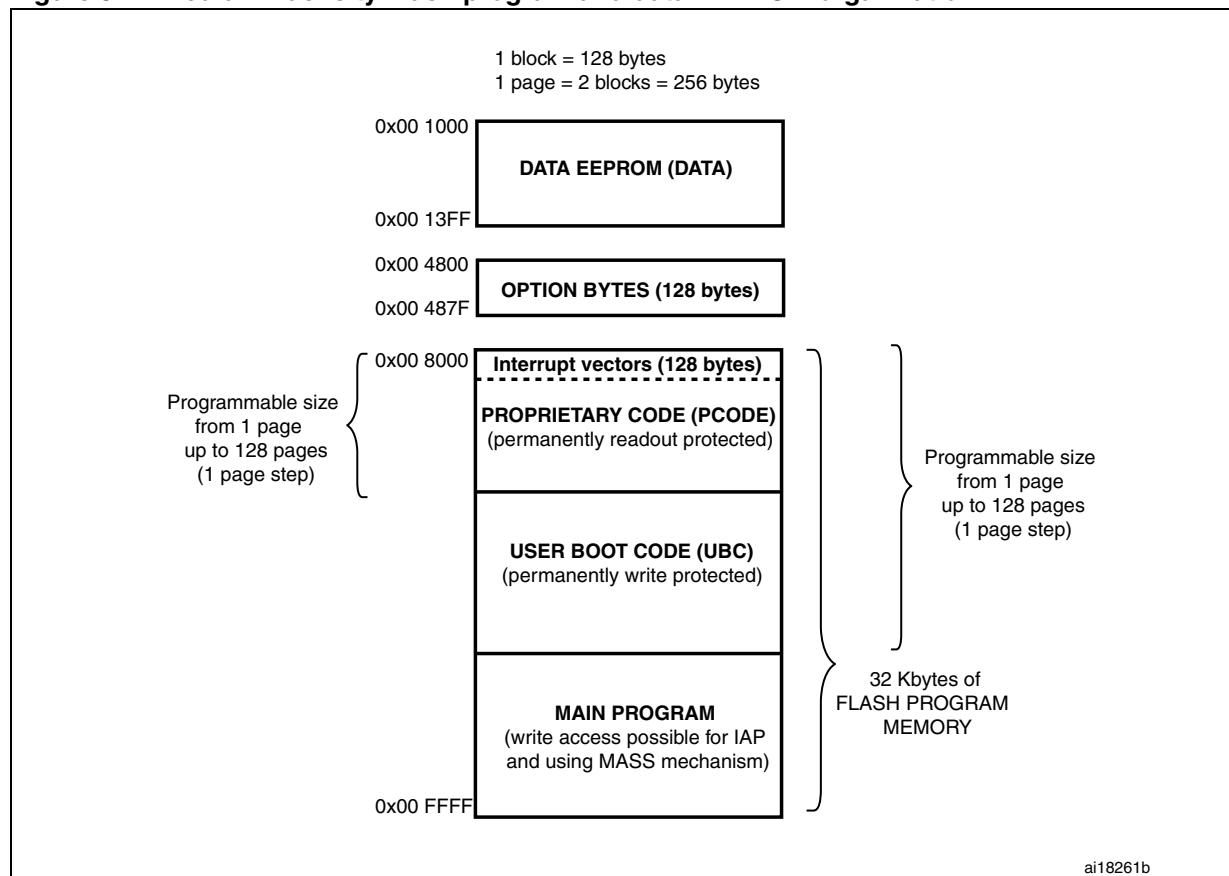
- 32 Kbytes of Flash program memory divided into 128 pages of 256 bytes each. The memory array is organized in 32-bit words (4 bytes per word). It includes three areas:
  - The user boot code area (UBC)
  - The proprietary code area (PCODE)
  - The main program area

The first block (128 bytes) of the Flash program memory (starting from address 0x00 8000) contains the interrupt vectors.

- Up to 1 Kbyte of data EEPROM organized in 4 pages of 256 bytes each.
- One block of option bytes (128 bytes) located in a separate memory array.

See [Figure 6](#) for a description of the memory organization.

**Figure 5. Medium+ density Flash program and data EEPROM organization**



### 3.4.4 High density device memory organization

High density EEPROM is split into three memory arrays:

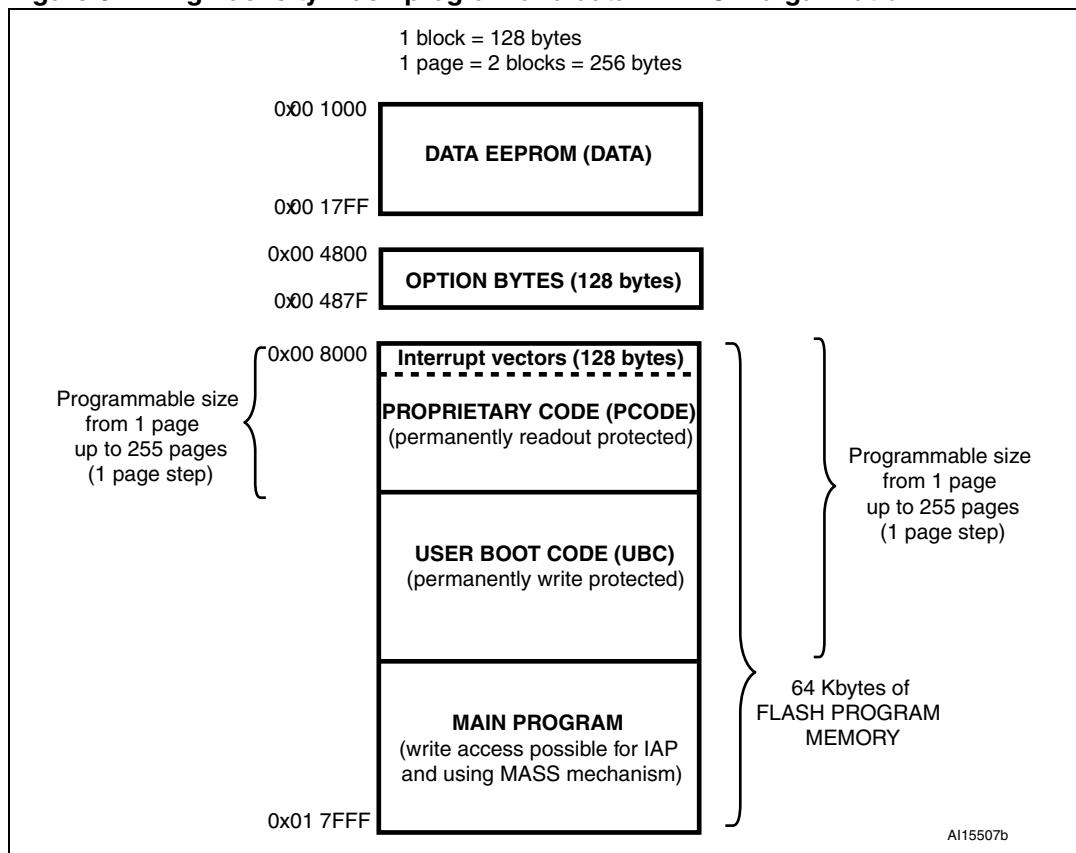
- 64 Kbytes of Flash program memory divided into 256 pages of 256 bytes each. The memory array is organized in 32-bit words (4 bytes per word). It includes three areas:
  - The user boot code area (UBC)
  - The proprietary code area (PCODE)
  - The main program area

The first block (128 bytes) of the Flash program memory (starting from address 0x00 8000) contains the interrupt vectors.

- Up to 2 Kbytes of data EEPROM organized in 8 pages of 256 bytes each.
- One block of option bytes (128 bytes) located in a separate memory array.

See [Figure 6](#) for a description of the memory organization.

**Figure 6. High density Flash program and data EEPROM organization**



### 3.4.5 Proprietary code area (PCODE)

The proprietary code area (PCODE) can be used to protect proprietary software libraries used to drive peripherals. It is available only in low, medium + and high density devices.

The size of the PCODE area can be configured through the PCODE option byte (PCODESIZE) in ICP mode (using the SWIM interface). This option byte specifies the number of pages (256-byte granularity) allocated for the PCODE area starting from address 0x00 8000. Once programmed, the PCODE option byte cannot be erased, and the size of the PCODE area remains fixed.

The minimum meaningful size of the PCODE area is 1 page (256 bytes) of which 128 bytes are used to store the interrupt vectors.

While the PCODE is enabled the TRAP interrupt vector is write protected.

In medium+ density devices, the maximum size of the PCODE area is 128 pages ranging from address 0x00 8000 to 0x00 FFFF. In high density devices, the maximum size of the PCODE area is 255 pages ranging from address 0x00 8000 to 0x01 7EFF.

The PCODE area is automatically readout protected except for the TRAP interrupt vector (see [Section 3.5.1: Readout protection](#), [Table 6: Memory access versus programming method \(low density devices\)](#), [Table 7: Memory access versus programming method \(medium density devices\)](#) and [Table 8: Memory access versus programming method \(medium+ and high density devices\)](#)). The readout protection cannot be disabled in this area. This means that the content of the PCODE area cannot be read or modified.

The PCODE area can be accessed only through the TRAP vector.

### 3.4.6 User boot area (UBC)

The user boot area (UBC) contains the reset and the interrupt vectors. It can be used to store the IAP and communication routines. The UBC area has a second level of protection to prevent unintentional erasing or modification during IAP programming. This means that it is always write protected and the write protection cannot be unlocked using the MASS keys.

The size of the UBC area can be obtained by reading the UBC option byte.

The size of the UBC area can be configured in ICP mode (using the SWIM interface) through the UBC option byte. The UBC option byte specifies the number of pages allocated for the UBC area starting from address 0x00 8000.

In medium density devices the minimum meaningful size of the UBC area is of 2 pages of which 1 is used to store the interrupt vectors.

In low, medium+ and high density devices, the minimum meaningful size of the UBC area is of 1 page of which 128 bytes are used to store the interrupt vectors.

Note:

*When a PCODE area has been defined, the minimum size of the UBC should be PCODESIZE+1. The portion of the UBC available to store the boot code is the area located between the end of the PCODE area and the end of the defined UBC area.*

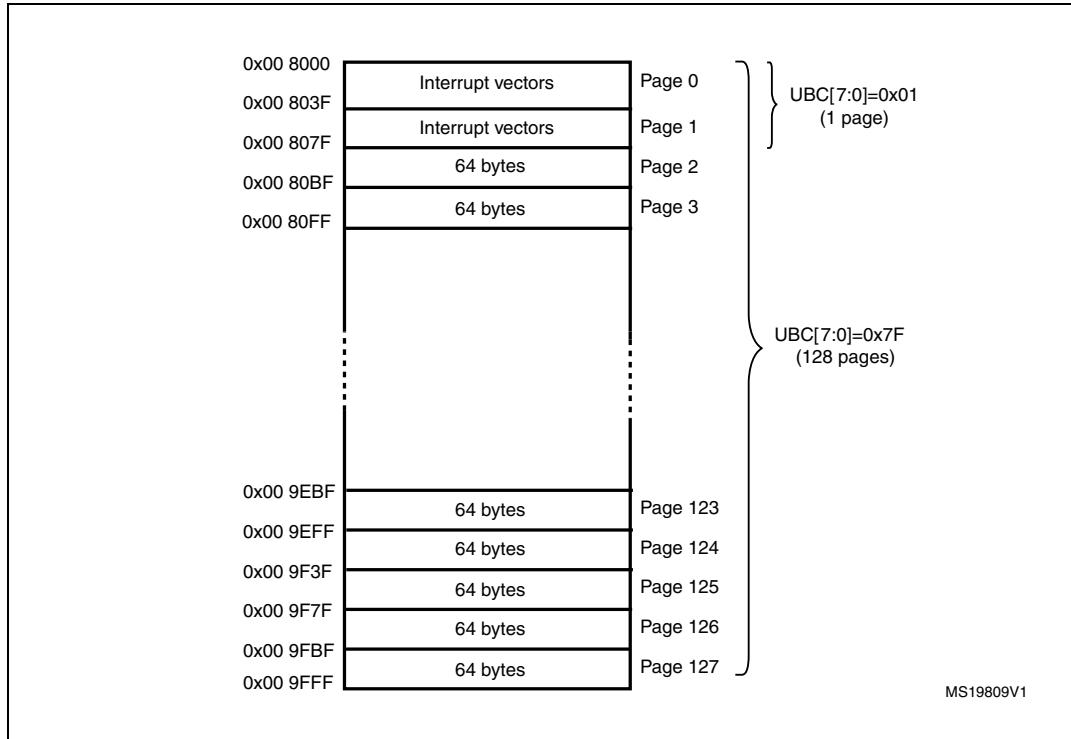
In high density devices, the maximum size of the boot area is 255 pages ranging from address 0x00 8000 to 0x01 7EFF (including the interrupt vectors).

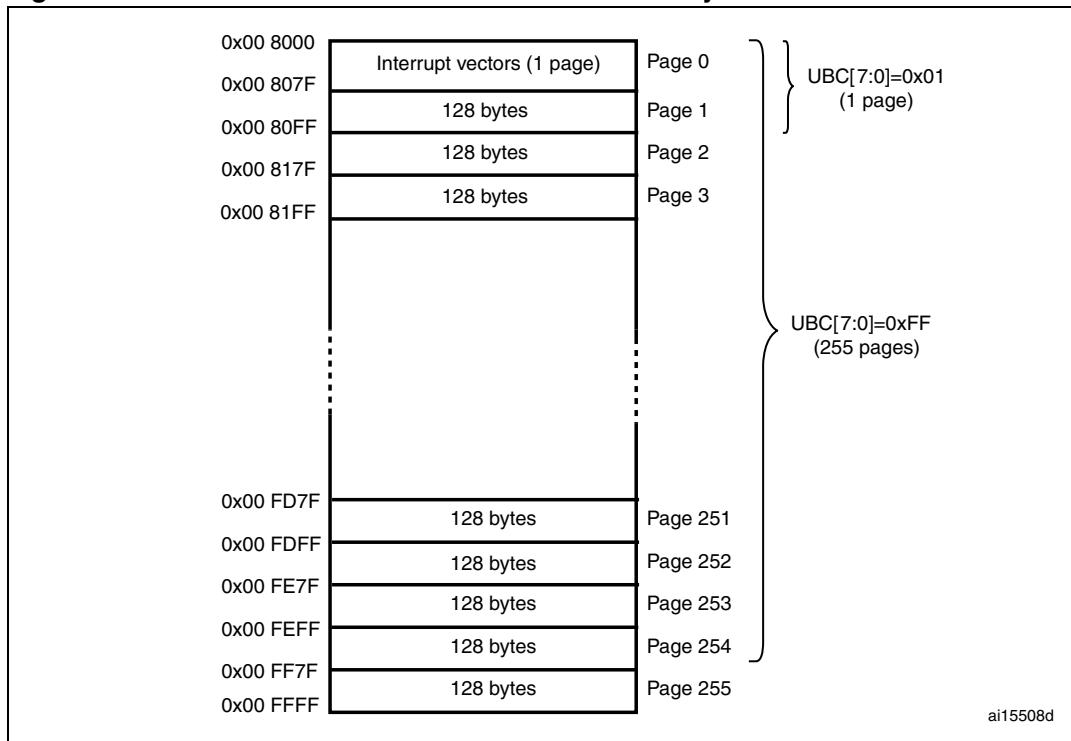
In medium+ density devices, the maximum size of the boot area is 128 pages ranging from address 0x00 8000 to 0x00 FFFF (including the interrupt vectors).

In medium density devices, the maximum size of the boot area is 255 pages ranging from address 0x00 8000 to 0x00 FF7F (including the interrupt vectors).

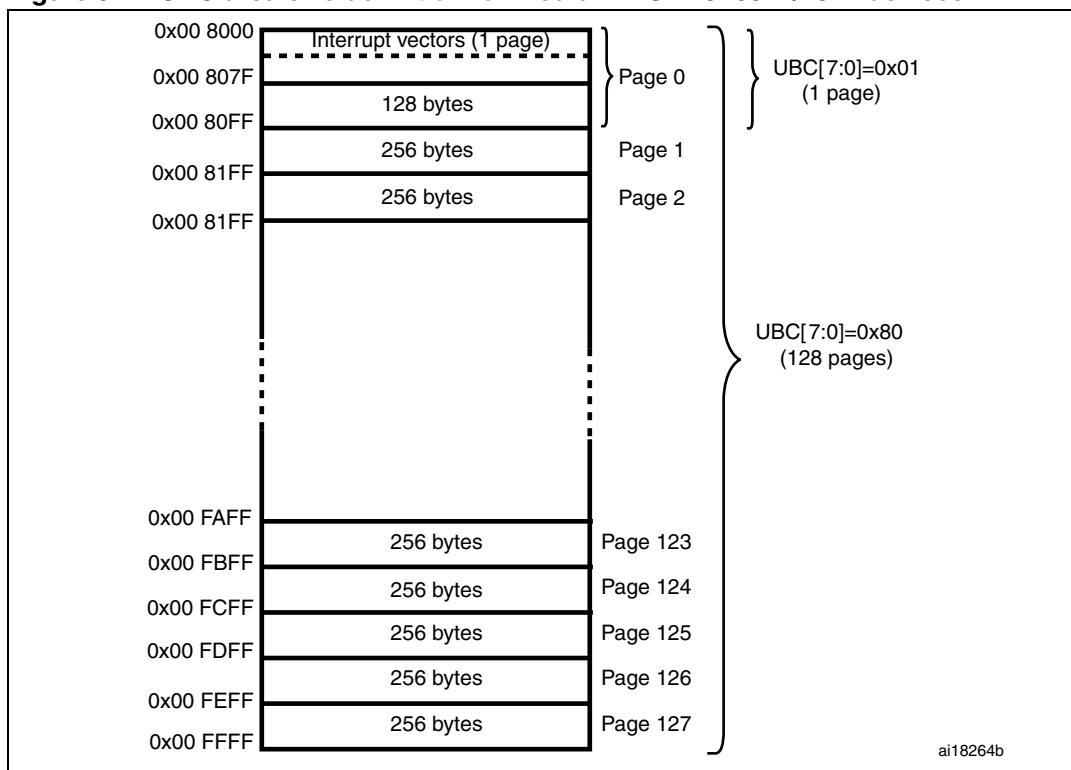
Refer to [Figure 7](#), [Figure 8](#), [Figure 9](#) and [Figure 10](#) for a description of the UBC area memory mapping and to the option byte section in the datasheets for more details on the UBC option byte.

**Figure 7. UBC area size definition for low density devices**

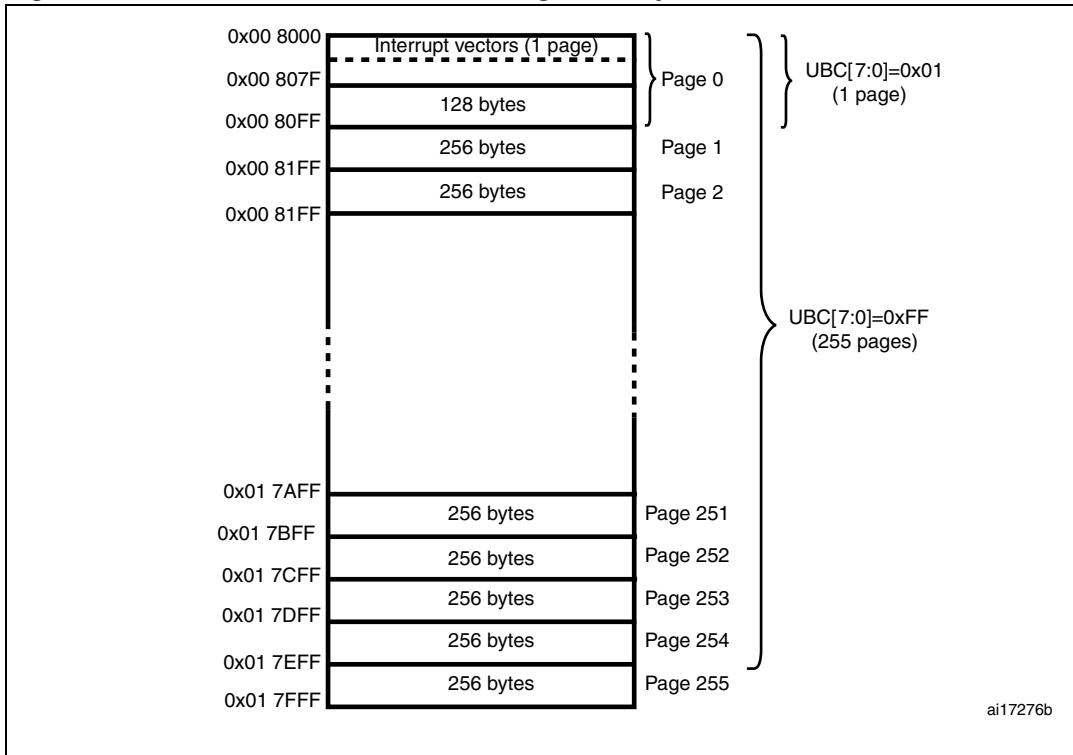


**Figure 8. UBC area size definition for medium density devices**

1. UBC[7:0]= 0x00 means no memory space is allocated for the UBC area.

**Figure 9. UBC area size definition for medium + STM8L05xx/15xx devices**

1. UBC[7:0]= 0x00 means no memory space is allocated for the UBC area.

**Figure 10. UBC area size definition for high density devices**

1. UBC[7:0]= 0x00 means no memory space is allocated for the UBC area.

### 3.4.7 Data EEPROM (DATA)

The data EEPROM area can be used to store application data. By default, the DATA area is write protected to prevent unintentional modification when the main program is updated in IAP mode. The write protection can be unlocked only by using a specific MASS key sequence (refer to [Enabling write access to the DATA area](#)).

The size of the DATA area is 256 bytes in low density STM8L15xx devices, 1 Kbyte in medium and medium+ density STM8L15xx devices, 1 Kbyte in medium density STM8AL31xx/3Lxx devices and 2 Kbytes in high density STM8L15xx/16xx devices. It starts from address 0x00 1000 (see [Figure 4](#), [Figure 5](#) and [Figure 6](#)).

The size of the DATA area is limited to 256 bytes for STM8L05xx value line devices.

### 3.4.8 Main program area

The main program is the area which starts at the end of the UBC or PCODE (when available) and ends at address 0x00 9FFF on low density STM8L05xx/15xx devices, 0x00 FFFF on medium and medium+ density STM8L05xx/15xx, 0x00 FFFF on medium density STM8AL31xx/3Lxx devices and 0x01 7FFF on high density STM8L05xx/15xx/16xx devices. It is used to store the application code (see [Figure 4](#), [Figure 5](#) and [Figure 6](#)).

### 3.4.9 Option bytes

The option bytes are used to configure device hardware features and memory protection. They are located in a dedicated memory array of one block.

The option bytes can be modified both in ICP/SWIM and in IAP mode, except for ROP and UBC and PCODESIZE (when available).

Refer to the option byte section in the datasheet for more information on option bytes, and to the STM8 SWIM protocol and debug module user manual (UM0470) for details on how to program them.

## 3.5 Memory protection

### 3.5.1 Readout protection

Readout protection is removed by programming the ROP option byte to 0xAA. When readout protection is enabled, reading or modifying the Flash program memory and DATA area in ICP mode (using the SWIM interface) is forbidden, whatever the write protection settings.

Even if no protection can be considered as totally unbreakable, the readout feature provides a very high level of protection for a general purpose microcontroller.

The readout protection can be disabled on the program memory, UBC, PCODE (when available) and DATA areas, by reprogramming the ROP option byte in ICP mode. In this case, the Flash program memory, the DATA area and the option bytes are automatically erased and the device can be reprogrammed.

By default, the PCODE area is always readout protected except for the interrupt vector TRAP (see [Section 3.4.5: Proprietary code area \(PCODE\)](#)).

Refer to [Table 7: Memory access versus programming method \(medium density devices\)](#) for details on memory access when readout protection is enabled or disabled.

### 3.5.2 Memory access security system (MASS)

After reset, the main program and DATA areas are protected against unintentional write operations. They must be unlocked before attempting to modify their content. This unlock mechanism is managed by the memory access security system (MASS).

The UBC area specified in the UBC option byte is always write protected (see [Section 3.4.6: User boot area \(UBC\)](#)).

Once the memory has been modified, it is recommended to enable the write protection again to protect the memory content against corruption.

#### Enabling write access to the main program memory

After a device reset, it is possible to disable the main program memory write protection by writing consecutively two values called MASS keys to the FLASH\_PUKR register (see [Section 3.9.3: Flash program memory unprotecting key register \(FLASH\\_PUKR\)](#)). These programmed keys are then compared to two hardware key values:

- First hardware key: 0b0101 0110 (0x56)
- Second hardware key: 0b1010 1110 (0xAE)

The following steps are required to disable write protection of the main program area:

1. Write a first 8-bit key into the FLASH\_PUKR register. When this register is written for the first time after a reset, the data bus content is not latched into the register, but compared to the first hardware key value (0x56).
2. If the key available on the data bus is incorrect, the FLASH\_PUKR register remains locked until the next reset. Any new write commands sent to this address are discarded.
3. If the first hardware key is correct when the FLASH\_PUKR register is written for the second time, the data bus content is still not latched into the register, but compared to the second hardware key value (0xAE).
4. If the key available on the data bus is incorrect, the write protection on program memory remains locked until the next reset. Any new write commands sent to this address is discarded.
5. If the second hardware key is correct, the main program memory is write unprotected and the PUL bit of the FLASH\_IAPSR is set (see [Section 3.9.5: Flash status register \(FLASH\\_IAPSR\)](#) register).

Before starting programming, the application must verify that PUL bit is effectively set. The application can choose, at any time, to disable again write access to the Flash program memory by clearing the PUL bit.

### Enabling write access to the DATA area

After a device reset, it is possible to disable the DATA area write protection by writing consecutively two values called MASS keys to the FLASH\_DUKR register (see [Section 3.9.6: Flash register map and reset values](#)). These programmed keys are then compared to two hardware key values:

- First hardware key: 0b1010 1110 (0xAE)
- Second hardware key: 0b0101 0110 (0x56)

The following steps are required to disable write protection of the DATA area:

1. Write a first 8-bit key into the FLASH\_DUKR register. When this register is written for the first time after a reset, the data bus content is not latched into the register, but compared to the first hardware key value (0xAE).
2. If the key available on the data bus is incorrect, the application can re-enter two MASS keys to try unprotecting the DATA area.
3. If the first hardware key is correct, the FLASH\_DUKR register is programmed with the second key. The data bus content is still not latched into the register, but compared to the second hardware key value (0x56).
4. If the key available on the data bus is incorrect, the data EEPROM area remains write protected until the next reset. Any new write command sent to this address is ignored.
5. If the second hardware key is correct, the DATA area is write unprotected and the DUL bit of the FLASH\_IAPSR register is set (see [Section 3.9.5: Flash status register \(FLASH\\_IAPSR\)](#)).

Before starting programming, the application must verify that the DATA area is not write protected by checking that the DUL bit is effectively set. The application can choose, at any time, to disable again write access to the DATA area by clearing the DUL bit.

### 3.5.3 Enabling write access to option bytes

The write access to the option byte area can be enabled by setting the OPT bit in the *Flash control register 2 (FLASH\_CR2)*.

## 3.6 Memory programming

The main program memory, and the DATA area must be unlocked before attempting to perform any program operation. The unlock mechanism depends on the memory area to be programmed as described in *Section 3.5.2: Memory access security system (MASS)*.

*Note:* *The PCODE area is always readout protected. The only way to reprogram it is to reset the ROP option byte, thus erasing the Flash program memory, DATA area, and option bytes.*

### 3.6.1 Read-while-write (RWW)

The RWW feature allows write operations to be performed on data EEPROM while reading and executing the program memory. Execution time is therefore optimized. The opposite operation is not allowed: Data memory cannot be read while writing to the program memory.

This RWW feature is always enabled and can be used at any time.

*Note:* *The RWW feature is not available on low density devices. Refer to the datasheets for addition information.*

### 3.6.2 Byte programming

The main program memory and the DATA area can be programmed at byte level. To program one byte, the application writes directly to the target address.

- In the main program memory:  
The application stops for the duration of the byte program operation.
- In DATA area:
  - Program execution stops during the “End of high voltage” operation, that is when the HVOFF flag is set, and the byte program operation is performed using the read-while-write (RWW) capability in IAP mode.

To erase a byte, simply write 0x00 at the corresponding address.

The application can read the FLASH\_IAPSR register to verify that the programming or erasing operation has been correctly executed:

- EOP flag is set after a successful programming operation
- WR\_PG\_DIS is set when the software has tried to write to a protected page. In this case, the write procedure is not performed.

As soon as one of these flags are set, a Flash interrupt is generated if it has been previously enabled by setting the IE bit of the FLASH\_CR1 register.

#### Automatic fast byte programming

The programming duration can vary according to the initial content of the target address. If the word (4 bytes) containing the byte to be programmed is not empty, the whole word is automatically erased before the program operation. On the contrary if the word is empty, no erase operation is performed and the programming time is shorter (see  $t_{PROG}$  in *Table “Flash program memory” in the datasheet*).

However, the programming time can be fixed by setting the FIX bit of the FLASH\_CR1 register to force the program operation to systematically erase the byte whatever its content (see [Section 3.9.1: Flash control register 1 \(FLASH\\_CR1\)](#)). The programming time is consequently fixed and equal to the sum of the erase and write time (see  $t_{PROG}$  in [Table "Flash program memory" in the datasheet](#)).

*Note:* *To write a byte fast (no erase), the whole word (4 bytes) into which it is written must be erased beforehand. Consequently, It is not possible to do two fast writes to the same word (without an erase before the second write): The first write will be fast but the second write to the other byte will require an erase.*

### 3.6.3 Word programming

A word write operation allows an entire 4-byte word to be programmed in one shot, thus minimizing the programming time.

As for byte programming, word operation is available both for the main program memory and data EEPROM. The read-while-write (RWW) capability is also available when a word programming operation is performed on the data EEPROM. Refer to the datasheets for additional information.

- In the main program memory:  
The application stops for the duration of the byte program operation.
- In DATA area:
  - Program execution does not stop, and the byte program operation is performed using the read-while-write (RWW) capability in IAP mode.

To program a word, the WPRG bit in the FLASH\_CR2 register must be previously set to enable word programming mode (see [Section 3.9.2: Flash control register 2 \(FLASH\\_CR2\)](#)). Then, the 4 bytes of the word to be programmed must be loaded starting with the first address. The programming cycle starts automatically when the 4 bytes have been written.

As for byte operation, the EOP and the WR\_PG\_DIS control flags of FLASH\_IAPSR, together with the Flash interrupt, can be used to determine if the operation has been correctly completed.

### 3.6.4 Block programming

Block program operations are much faster than byte or word program operations. In a block program operation, a whole block is programmed or erased in a single programming cycle. Refer to [Table 5](#) for details on the block size according to the devices.

Block operations can be performed both to the main program memory and DATA area:

- In the main program memory:  
Block program operations to the main program memory have to be executed totally from RAM.
- In the DATA area:
  - DATA block operations can be executed from the main program memory. However, the data loading phase (see below) has to be executed from RAM.

There are three possible block operations:

- Block programming, also called standard block programming: The block is automatically erased before being programmed.
- Fast block programming: No previous erase operation is performed.
- Block erase

During block programming, interrupts are masked by hardware.

### Standard block programming

A standard block program operation allows a whole block to be written in one shot. The block is automatically erased before being programmed.

To program a whole block in standard mode, the PRG bit in the FLASH\_CR2 register must be previously set to enable standard block programming (see [Section 3.9.2: Flash control register 2 \(FLASH\\_CR2\)](#)). Then, the block of data to be programmed must be loaded sequentially to the destination addresses in the main program memory or DATA area. This causes all the bytes of data to be latched. To start programming the whole block, all bytes of data must be written. All bytes written in a programming sequence must be in the same block. This means that they must have the same high address: Only the six least significant bits of the address can change. When the last byte of the target block is loaded, the programming starts automatically. It is preceded by an automatic erase operation of the whole block.

The EOP and the WR\_PG\_DIS control flags of the FLASH\_IAPSR together with the Flash interrupt can be used to determine if the operation has been correctly completed.

### Fast block programming

Fast block programming allows programming without first erasing the memory contents. Fast block programming is therefore twice as fast as standard programming.

This mode is intended only for programming parts that have already been erased. It is very useful for programming blank parts with the complete application code, as the time saving is significant.

Fast block programming is performed by using the same sequence as standard block programming. To enable fast block programming mode, the FPRG bit of the FLASH\_CR2 registers must be previously set.

The EOP and WR\_PG\_DIS bits of the FLASH\_IAPSR register can be checked to determine if the fast block programming operation has been correctly completed.

**Caution:** The data programmed in the block are not guaranteed when the block is not blank before the fast block program operation.

### Block erasing

A block erase allows a whole block to be erased.

To erase a whole block, the ERASE bit in the FLASH\_CR2 register must be previously set to enable block erasing (see [Section 3.9.2: Flash control register 2 \(FLASH\\_CR2\)](#)). The block is then erased by writing '0x00 00 00 00' to any word inside the block. The word start address must end with '0', '4', '8', or 'C'.

The EOP and the WR\_PG\_DIS control flags of the FLASH\_IAPSR together with the Flash interrupt can be used to determine if the operation has been correctly completed.

**Table 5. Block size**

STM8 microcontroller family	Block size
Low density STM8L05xx/15xx	64 bytes
Medium density STM8L05xx/15xx and STM8AL31xx/3Lxx	128 bytes
Medium+ density STM8L05xx/15xx	128 bytes
High density STM8L05xx/15xx/16xx	128 bytes

### 3.6.5 Option byte programming

Option byte programming is very similar to data EEPROM byte programming.

The application writes directly to the target address. The program does not stop and the write operation is performed using the RWW capability.

Refer to the datasheet for details of the option byte contents.

### 3.7 Flash low power modes

All STM8L05xx/15xx/16xx and STM8AL31xx/3Lxx Flash program memory and data EEPROM have one low consumption mode,  $I_{DDQ}$ . In  $I_{DDQ}$  mode, the memory is switched off. It is used for any of the device low power modes: Halt, active-halt, low power wait, and low power run.

When the EEPM bit is set in FLASH\_CR1 register, the Flash program memory and data EEPROM automatically enter  $I_{DDQ}$  mode when the code is executed from RAM or when the device is in Wait mode.

Refer to [Section 3.9.1: Flash control register 1 \(FLASH\\_CR1\)](#) for details on WAITM and EEPM bits.

When the Flash program memory and data EEPROM exit from  $I_{DDQ}$  mode, the recovery time is lower than 2.8  $\mu$ s and depends on supply voltage and temperature.

### 3.8 ICP and IAP

The in-circuit programming (ICP) method is used to update the entire content of the memory, using the SWIM interface to load the user application into the microcontroller. ICP offers quick and efficient design iterations and eliminates unnecessary package handling or socketing of devices. The SWIM interface (single wire interface module) uses the SWIM pin to connect to the programming tool.

In contrast to the ICP method, in-application programming (IAP) can use any communication interface supported by the microcontroller (I/Os, I<sup>2</sup>C, SPI, USART...) to download the data to be programmed in the memory. IAP allows the Flash program memory content to be reprogrammed during application execution. Nevertheless, part of the application must have been previously programmed in the Flash program memory using ICP.

Refer to the STM8L Flash programming manual (PM0054) and STM8 SWIM protocol and debug manual (UM0470) for more information on programming procedures.

**Table 6. Memory access versus programming method<sup>(1)</sup> (low density devices)**

Mode	ROP	Memory area	Access from core
User mode, IAP, and bootloader	Readout protection enabled	Interrupt vectors except for TRAP	R/W <sup>(2)</sup> /E
		TRAP	R/W <sup>(3)</sup> /E
		Proprietary code area (PCODE) when available	R/E
		Main program	R/W/E <sup>(4)</sup>
		Data EEPROM area (DATA)	R/W <sup>(5)</sup>
		Option bytes	R
	Readout protection disabled	Interrupt vectors except for TRAP	R/W <sup>(2)</sup> /E
		TRAP	R/W <sup>(3)</sup> /E
		Proprietary code area (PCODE) when available	R/E <sup>(6)</sup>
		Main program	R/W/E <sup>(4)</sup>
		Data EEPROM area (DATA)	R/W <sup>(5)</sup>
		Option bytes	R/W <sup>(7)</sup>
SWIM active (ICP mode)	Readout protection enabled	Interrupt vectors except for TRAP	P
		TRAP	P
		Proprietary code area (PCODE) when available	P <sup>(6)</sup>
		Main program	P
		Data EEPROM area (DATA)	P
		Option bytes	P/W <sub>ROP</sub> <sup>(8)</sup>
	Readout protection disabled	Interrupt vectors except for TRAP	R/W <sup>(2)</sup> /E
		TRAP	R/W/E
		Proprietary code area (PCODE) when available	R/E <sup>(6)</sup>
		Main program	R/W/E <sup>(4)</sup>
		Data EEPROM area (DATA)	R/W <sup>(5)</sup>
		Option bytes	R/W <sup>(7)</sup>

1. R/W/E = Read, write, and execute  
R/E = Read and execute (write operation forbidden)  
R = Read (write and execute operations forbidden)  
P = The area cannot be accessed (read, execute and write operations forbidden)  
P/W<sub>ROP</sub> = Protected, write forbidden except for ROP option byte.
2. When no UBC area has been defined, the interrupt vectors, except for TRAP, can be modified in user/IAP mode.
3. If a PCODE area has been defined, the TRAP vector cannot be modified in user and IAP mode, otherwise TRAP follows the same rules as other interrupt vectors.
4. The Flash program memory is write protected (locked) until the correct MASS key is written in the FLASH\_PUKR. It is possible to lock the memory again by resetting the PUL bit in the FLASH\_IAPSR register. Unlocking can only be done once between two resets. If incorrect keys are provided, the device must be reset and new keys programmed.
5. The data memory is write protected (locked) until the correct MASS key is written in the FLASH\_DUKR. It is possible to lock the memory again by resetting the DUL bit in the FLASH\_IAPSR register. If incorrect keys are provided, another key program sequence can be performed without resetting the device.
6. The PCODE area can be read and executed only in privileged mode through TRAP vectors. The PCODE cannot be directly accessed through the SWIM.

7. The option bytes are write protected (locked) until the correct MASS key is written in the FLASH\_DUKR (with OPT set to 1). It is possible to lock the memory again by resetting the DUL bit in the FLASH\_IAPSR register. If incorrect keys are provided, another key program sequence can be performed without resetting the device.
8. When ROP is removed, the whole memory is erased, including option bytes.

**Table 7. Memory access versus programming method<sup>(1)</sup> (medium density devices)**

Mode	ROP	Memory area	Access from core
User mode, IAP, and bootloader	Readout protection enabled	Interrupt vectors except for TRAP	R/W <sup>(2)</sup> /E
		TRAP	R/W/E
		User boot code area (UBC)	R/E
		Main program	R/W/E <sup>(3)</sup>
		Data EEPROM area (DATA)	R/W <sup>(4)</sup>
		Option bytes	R
	Readout protection disabled	Interrupt vectors except for TRAP	R/W <sup>(2)</sup> /E
		TRAP	R/W/E
		User boot code area (UBC)	R/E <sup>(5)</sup>
		Main program	R/W/E <sup>(3)</sup>
		Data EEPROM area (DATA)	R/W <sup>(4)</sup>
SWIM active (ICP mode)	Readout protection enabled	Option bytes	R/W <sup>(6)</sup>
		Interrupt vectors except for TRAP	P
		TRAP	P
		User boot code area (UBC)	P
		Main program	P
		Data EEPROM area (DATA)	P
	Readout protection disabled	Option bytes	P/W <sub>ROP</sub> <sup>(7)</sup>
		Interrupt vectors except for TRAP	R/W <sup>(2)</sup> /E
		TRAP	R/W/E
		User boot code area (UBC)	R/E <sup>(5)</sup>
		Main program	R/W/E <sup>(3)</sup>
		Data EEPROM area (DATA)	R/W <sup>(4)</sup>
		Option bytes	R/W <sup>(6)</sup>

1. R/W/E = Read, write, and execute

R/E = Read and execute (write operation forbidden)

R = Read (write and execute operations forbidden)

P = The area cannot be accessed (read, execute and write operations forbidden)

P/W<sub>ROP</sub> = Protected, write forbidden except for ROP option byte.

2. When no UBC area has been defined, the interrupt vectors, except for TRAP, can be modified in user/IAP mode.
3. The Flash program memory is write protected (locked) until the correct MASS key is written in the FLASH\_PUKR. It is possible to lock the memory again by resetting the PUL bit in the FLASH\_IAPSR register. Unlocking can only be done once between two resets. If incorrect keys are provided, the device must be reset and new keys programmed.

4. The data memory is write protected (locked) until the correct MASS key is written in the FLASH\_DUKR. It is possible to lock the memory again by resetting the DUL bit in the FLASH\_IAPSR register. If incorrect keys are provided, another key program sequence can be performed without resetting the device.
5. To program the UBC area the application must first clear the UBC option byte.
6. The option bytes are write protected (locked) until the correct MASS key is written in the FLASH\_DUKR (with OPT set to 1). It is possible to lock the memory again by resetting the DUL bit in the FLASH\_IAPSR register. If incorrect keys are provided, another key program sequence can be performed without resetting the device.
7. When ROP is removed, the whole memory is erased, including option bytes.

**Table 8. Memory access versus programming method<sup>(1)</sup>  
(medium+ and high density devices)**

Mode	ROP	Memory area	Access from core
User mode, IAP, and bootloader	Readout protection enabled	Interrupt vectors except for TRAP	R/W <sup>(2)</sup> /E
		TRAP	R/W <sup>(3)</sup> /E
		Proprietary code area (PCODE) when available	R/E <sup>(6)</sup>
		User boot code area (UBC)	R/E
		Main program	R/W/E <sup>(4)</sup>
		Data EEPROM area (DATA)	R/W <sup>(5)</sup>
		Option bytes	R
	Readout protection disabled	Interrupt vectors except for TRAP	R/W <sup>(2)</sup> /E
		TRAP	R/W <sup>(3)</sup> /E
		Proprietary code area (PCODE) when available	R/E <sup>(6)</sup>
		User boot code area (UBC)	R/E <sup>(7)</sup>
		Main program	R/W/E <sup>(4)</sup>
		Data EEPROM area (DATA)	R/W <sup>(5)</sup>
		Option bytes	R/W <sup>(8)</sup>

**Table 8. Memory access versus programming method<sup>(1)</sup>  
(medium+ and high density devices) (continued)**

Mode	ROP	Memory area	Access from core
SWIM active (ICP mode)	Readout protection enabled	Interrupt vectors except for TRAP	P
		TRAP	P
		Proprietary code area (PCODE) when available	P <sup>(6)</sup>
		User boot code area (UBC)	P
		Main program	P
		Data EEPROM area (DATA)	P
		Option bytes	P/W <sub>ROP</sub> <sup>(9)</sup>
	Readout protection disabled	Interrupt vectors except for TRAP	R/W <sup>(2)</sup> /E
		TRAP	R/W/E
		Proprietary code area (PCODE) when available	R/E <sup>(6)</sup>
		User boot code area (UBC)	R/E <sup>(7)</sup>
		Main program	R/W/E <sup>(4)</sup>
		Data EEPROM area (DATA)	R/W <sup>(5)</sup>
		Option bytes	R/W <sup>(7)</sup>

1. R/W/E = Read, write, and execute  
R/E = Read and execute (write operation forbidden)  
R = Read (write and execute operations forbidden)  
P = The area cannot be accessed (read, execute and write operations forbidden)  
P/W<sub>ROP</sub> = Protected, write forbidden except for ROP option byte.
2. When no UBC area has been defined, the interrupt vectors, except for TRAP, can be modified in user/IAP mode.
3. If a PCODE area has been defined, the TRAP vector cannot be modified in user and IAP mode, otherwise TRAP follows the same rules as other interrupt vectors.
4. The Flash program memory is write protected (locked) until the correct MASS key is written in the FLASH\_PUKR. It is possible to lock the memory again by resetting the PUL bit in the FLASH\_IAPSR register. Unlocking can only be done once between two resets. If incorrect keys are provided, the device must be reset and new keys programmed.
5. The data memory is write protected (locked) until the correct MASS key is written in the FLASH\_DUKR. It is possible to lock the memory again by resetting the DUL bit in the FLASH\_IAPSR register. If incorrect keys are provided, another key program sequence can be performed without resetting the device.
6. The PCODE area can be read and executed only in privileged mode through TRAP vectors. The PCODE cannot be directly accessed through the SWIM.
7. To program the UBC area the application must first clear the UBC option byte.
8. The option bytes are write protected (locked) until the correct MASS key is written in the FLASH\_DUKR (with OPT set to '1'). It is possible to lock the memory again by resetting the DUL bit in the FLASH\_IAPSR register. If incorrect keys are provided, another key program sequence can be performed without resetting the device.
9. When ROP is removed, the whole memory is erased, including option bytes.

## 3.9 Flash registers

### 3.9.1 Flash control register 1 (FLASH\_CR1)

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved				EEPM	WAITM	IE	FIX
				rw	rw	rw	rw

Bits 7: Reserved

Bit 3 **EEPM**: Flash program and data EEPROM I<sub>DDQ</sub> mode selection during run, low power run and low power wait mode

0: EEPM is cleared by hardware just after a Flash program or data EEPROM memory access. Clearing this bit by software does not exit the memory from I<sub>DDQ</sub> mode.

1: When the code is executed from RAM, the Flash program and data EEPROM enter I<sub>DDQ</sub> mode if the EEPM is set.

Flash program and data EEPROM remain in I<sub>DDQ</sub> mode till the next Flash program or data EEPROM memory access.

Refer to [Section 7.6: Low power run mode on page 81](#).

Bit 2 **WAITM**: Flash program and data EEPROM I<sub>DDQ</sub> mode during wait mode

This bit is set and cleared by software.

0: Flash program and data EEPROM not in I<sub>DDQ</sub> mode when the device is in wait mode

1: Flash program and data EEPROM in I<sub>DDQ</sub> mode when the device is in wait mode.

Bit 1 **IE**: Flash Interrupt enable

This bit is set and cleared by software.

0: Interrupt disabled

1: Interrupt enabled. An interrupt is generated if the EOP or WR\_PG\_DIS flag in the FLASH\_IAPSR register is set.

Bit 0 **FIX**: Fixed Byte programming time

This bit is set and cleared by software.

0: Standard programming time of (1/2 t<sub>prog</sub>) if the memory is already erased and t<sub>prog</sub> otherwise.

1: Programming time fixed at t<sub>prog</sub>.

### 3.9.2 Flash control register 2 (FLASH\_CR2)

Address offset: 0x01

Reset value: 0x00

7	6	5	4	3	2	1	0
OPT	WPRG	ERASE	FPRG	Reserved			PRG
rw	rw	rw	rw				rw

Bit 7 **OPT**: Write option bytes

This bit is set and cleared by software.

0: Write access to option bytes disabled

1: Write access to option bytes enabled

Bit 6 **WPRG**: Word programming

This bit is set by software and cleared by hardware when the operation is completed.

0: Word program operation disabled

1: Word program operation enabled

Bit 5 **ERASE<sup>(1)</sup>**: Block erasing

This bit is set by software and cleared by hardware when the operation is completed.

0: Block erase operation disabled

1: Block erase operation enabled

Bit 4 **FPRG<sup>(1)</sup>**: Fast block programming

This bit is set by software and cleared by hardware when the operation is completed.

0: Fast block program operation disabled

1: Fast block program operation enabled

Bits 3:1 Reserved

Bit 0 **PRG**: Standard block programming

This bit is set by software and cleared by hardware when the operation is completed.

0: Standard block programming operation disabled

1: Standard block programming operation enabled (automatically first erasing)

1. The ERASE and FPRG bits are locked when the memory is busy.

### 3.9.3 Flash program memory unprotected key register (FLASH\_PUKR)

Address offset: 0x02

Reset value: 0x00

7	6	5	4	3	2	1	0
MASS_PRG KEYS							
rw							

Bits 7:0 **PUK [7:0]**: Main program memory unlock keys

This byte is written by software (all modes). It returns 0x00 when read.

Refer to [Enabling write access to the main program memory on page 47](#) for the description of main program area write unprotection mechanism.

### 3.9.4 Data EEPROM unprotection key register (FLASH\_DUKR)

Address offset: 0x03

Reset value: 0x00

7	6	5	4	3	2	1	0
MASS_DATA KEYS							
rw							

Bits 7:0 **DUK[7:0]**: Data EEPROM write unlock keys

This byte is written by software (all modes). It returns 0x00 when read.

Refer to [Enabling write access to the DATA area on page 48](#) for the description of main program area write unprotection mechanism.

### 3.9.5 Flash status register (FLASH\_IAPSR)

Address offset: 0x04

Reset value: 0x40

7	6	5	4	3	2	1	0
Reserved	HVOFF	Reserved	DUL	EOP	PUL	WR_PG_DIS	
res.	r		rc_w0	rc_r	rc_w0	rc_r	

Bit 7 Reserved.

Bit 6 **HVOFF**: End of high voltage flag

This bit is set and cleared by hardware.

0: HV ON, start of actual programming

1: HV OFF, end of high voltage

Bit 3 **DUL**: Data EEPROM area unlocked flag

This bit is set by hardware and cleared by software by programming it to 0.

0: Data EEPROM area write protection enabled

1: Data EEPROM area write protection has been disabled by writing the correct MASS keys

Bit 2 **EOP**: End of programming (write or erase operation) flag

This bit is set by hardware. It is cleared by software by reading the register, or when a new write/erase operation starts.

0: No EOP event occurred

1: An EOP operation occurred. An interrupt is generated if the IE bit is set in the FLASH\_CR1 register.

Bit 1 **PUL**: Flash Program memory unlocked flag

This bit is set by hardware and cleared by software by programming it to 0.

0: Write protection of main Program area enabled

1: Write protection of main Program area has been disabled by writing the correct MASS keys.

Bit 0 **WR\_PG\_DIS**: Write attempted to protected page flag

This bit is set by hardware and cleared by software by reading the register.

0: No WR\_PG\_DIS event occurred

1: A write attempt to a write protected page occurred. An interrupt is generated if the IE bit is set in the FLASH\_CR1 register.

### 3.9.6 Flash register map and reset values

For details on the Flash register boundary addresses, refer to the general hardware register map in the datasheets.

**Table 9. Flash register map**

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	FLASH_CR1	- 0	- 0	- 0	- 0	EEPM 0	WAITM 0	IE 0	FIX 0
0x01	FLASH_CR2	OPT 0	WPRG 0	ERASE 0	FPRG 0	- 0	- 0	- 0	PRG 0
0x02	FLASH_PUKR	PUK7 0	PUK6 0	PUK5 0	PUK4 0	PUK3 0	PUK2 0	PUK1 0	PUK0 0
0x03	FLASH_DUKR	DUK7 0	DUK6 0	DUK5 0	DUK4 0	DUK3 0	DUK2 0	DUK1 0	DUK0 0
0x04	FLASH_IAPSR	- 0	HVOFF 1	- 0	- 0	DUL 0	EOP 0	PUL 0	WR_PG_DIS 0

## 4 Single wire interface module (SWIM) and debug module (DM)

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

### 4.1 Introduction

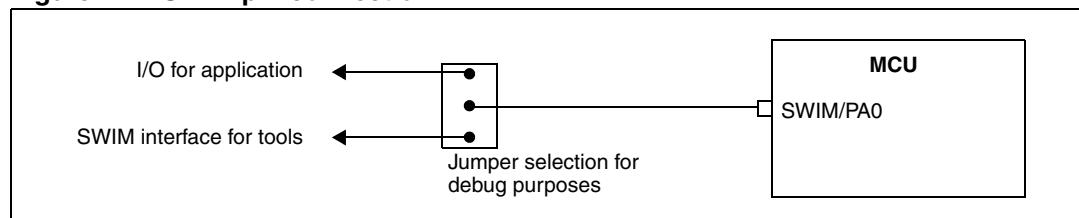
In-circuit debugging mode or in-circuit programming mode are managed through a single wire hardware interface featuring ultrafast memory programming. Coupled with an in-circuit debugging module, it also offers a non-intrusive emulation mode, making the in-circuit debugger extremely powerful, close in performance to a full-featured emulator.

### 4.2 Main features

- Based on an asynchronous, high sink (8 mA), open-drain, bidirectional communication.
- Allows reading or writing any part of memory space.
- Access to CPU registers (A, X, Y, CC, SP). They are memory mapped for read or write access.
- Non intrusive read/write on the fly to the RAM and peripheral registers.

SWIM pin can be used as a standard I/O with some restrictions if you also want to use it for debug. The most secure way is to provide on the PCB a strap option.

**Figure 11. SWIM pin connection**



## 4.3 SWIM modes

After a power-on reset, the SWIM is reset and enters OFF mode.

1. **OFF:** Default state after power-on reset. The SWIM pin cannot be used by the application as an I/O.
2. **I/O:** This state is entered by software writing to the SWD bit in the *Global configuration register (CFG\_GCR)*. In this state, the SWIM pin can be used by the application as a standard I/O pin. In case of a reset, the SWIM goes back to OFF mode.
3. **SWIM:** This state is entered when a specific sequence is performed on the SWIM pin. In this state, the SWIM pin is used by the host tool to control the STM8 with 3 commands (SRST system reset, ROTF read on the fly, WOTF write on the fly).

*Note:* Refer to the STM8 SWIM communication Protocol and Debug Module User Manual for a description of the SWIM and Debug module (DM) registers.

## 5 Memory and register map

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

For details on the memory map, I/O port hardware register map and CPU/SWIM/debug module/interrupt controller registers, refer to the product datasheets.

### 5.1 Register description abbreviations

In the register descriptions of each chapter in this reference manual, the following abbreviations are used:

**Table 10. List of abbreviations**

Abbreviation	Description
<b>read/write (rw)</b>	Software can read and write to these bits.
<b>read-only (r)</b>	Software can only read these bits.
<b>write only (w)</b>	Software can only write to this bit. Reading the bit returns a meaningless value.
<b>read/write once (rwo)</b>	Software can only write once to this bit but can read it at any time. Only a reset can return this bit to its reset value.
<b>read/clear (rc_w1)</b>	Software can read and clear this bit by writing 1. Writing '0' has no effect on the bit value.
<b>read/clear (rc_w0)</b>	Software can read and clear this bit by writing 0. Writing '1' has no effect on the bit value.
<b>read/set (rs)</b>	Software can read and set this bit. Writing '0' has no effect on the bit value.
<b>read/clear by read (rc_r)</b>	Software can read this bit. Reading this bit automatically clears it to '0'. Writing '0' has no effect on the bit value.
<b>Reserved (Res.)</b>	Reserved bit, must be kept at reset value.

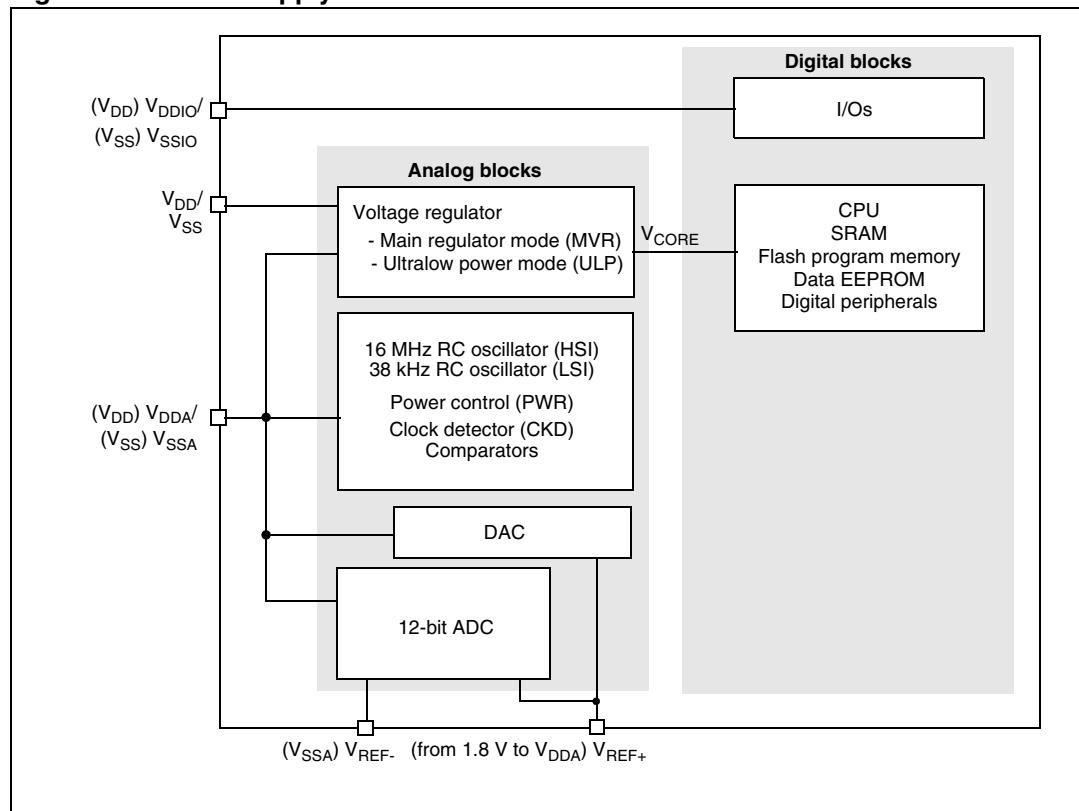
## 6 Power control (PWR)

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

### 6.1 Power supply

The device requires a 1.65 V to 3.6 V operating voltage supply ( $V_{DD}$ ) (1.8 V to 3.6 V for STM8L05xx value line devices). An internal voltage regulator is used to supply  $V_{CORE}$  to the digital core, SRAM, Flash memory and data EEPROM.

**Figure 12. Power supply overview**



**Note:** In the above figure, the DAC is not available on low density or STM8L05x value line devices.

### External power supply pins:

The external power supply pins must be connected as follows:

- $V_{SS}$ ;  $V_{DD} = 1.8$  to  $3.6$  V, down to  $1.65$  V<sup>(1)</sup> at power-down: external power supply for I/Os and for the internal regulator. Provided externally through  $V_{DD}$  pins, the corresponding ground pin is  $V_{SS}$ .
- $V_{SSA}$ ;  $V_{DDA} = 1.8$  to  $3.6$  V, down to  $1.65$  V<sup>(1)</sup> at power-down: external power supplies for analog peripherals (minimum voltage to be applied to  $V_{DDA}$  is  $1.8$  V when the ADC1 is used).  $V_{DDA}$  and  $V_{SSA}$  must be connected to  $V_{DD}$  and  $V_{SS}$ , respectively.
- $V_{SSIO}$ ;  $V_{DDIO} = 1.8$  to  $3.6$  V, down to  $1.65$  V<sup>(1)</sup> at power-down: external power supplies for I/Os.  $V_{DDIO}$  and  $V_{SSIO}$  must be connected to  $V_{DD}$  and  $V_{SS}$ , respectively.
- $V_{REF+}$ ;  $V_{REF-}$  (for ADC1): external reference voltage for ADC1. Must be provided externally through  $V_{REF+}$  and  $V_{REF-}$  pin.
- $V_{REF+}$  (for DAC)<sup>(1)</sup>: external voltage reference for DAC must be provided externally through  $V_{REF+}$ .

## 6.2

### Power-on reset (POR)/power-down reset (PDR)

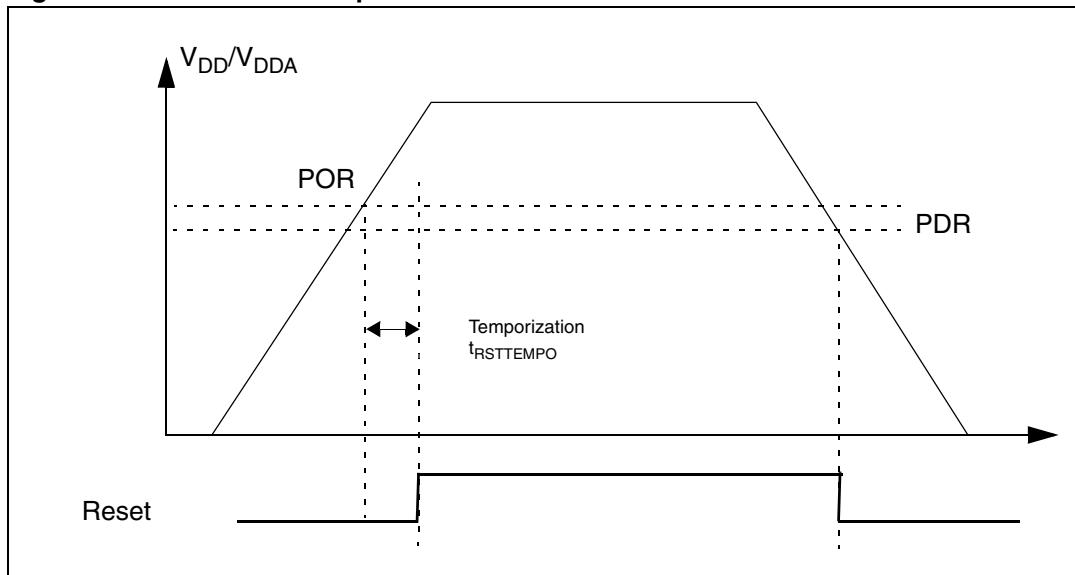
The device has an integrated POR/PDR circuitry that allows operation down to  $1.5$  V typical. During power-on, the device remains in Reset mode when  $V_{DD}/V_{DDA}$  is below a specified threshold,  $V_{POR}$ , without the need for an external reset circuit. The POR feature is always enabled and the POR threshold is around  $1.5$  V. During power-down, the PDR keeps the device under reset when the supply voltage ( $V_{DD}$ ) drops below the  $V_{PDR}$  threshold. The PDR feature is always enabled and the PDR threshold is  $1.5$  V.

In low density devices, medium density devices and in the medium+ and high density devices operating from  $1.8$ V to  $3.6$ V (device part numbers without "D" suffix), the BOR starts to work above the POR threshold (see [Section 6.3: Brownout reset \(BOR\) on page 66](#)). To ensure the minimum operating voltage ( $1.65$  V), the BOR should be configured to BOR Level 0.

When the BOR is disabled, a "grey zone" exists between the minimum operating voltage ( $1.65$  V)<sup>(1)</sup> and the  $V_{POR}/V_{PDR}$  threshold. This means that  $V_{DD}$  can be lower than  $1.65$  V without device reset until the  $V_{PDR}$  threshold is reached. For more details concerning the power on/power down reset threshold, refer to the electrical characteristics of the datasheet.

---

1. Not applicable on STM8L05xx value line devices. Refer to datasheets for more details.

**Figure 13. Power on reset/power down reset waveform**

### 6.3 Brownout reset (BOR)

For low density devices, medium density devices and for medium+ and high density devices operating from 1.8 V to 3.6 V, the BOR is always active at power-on, keeping the MCU under reset till the application operating threshold is reached.

At power-down, the BOR threshold is also used to generate a reset.

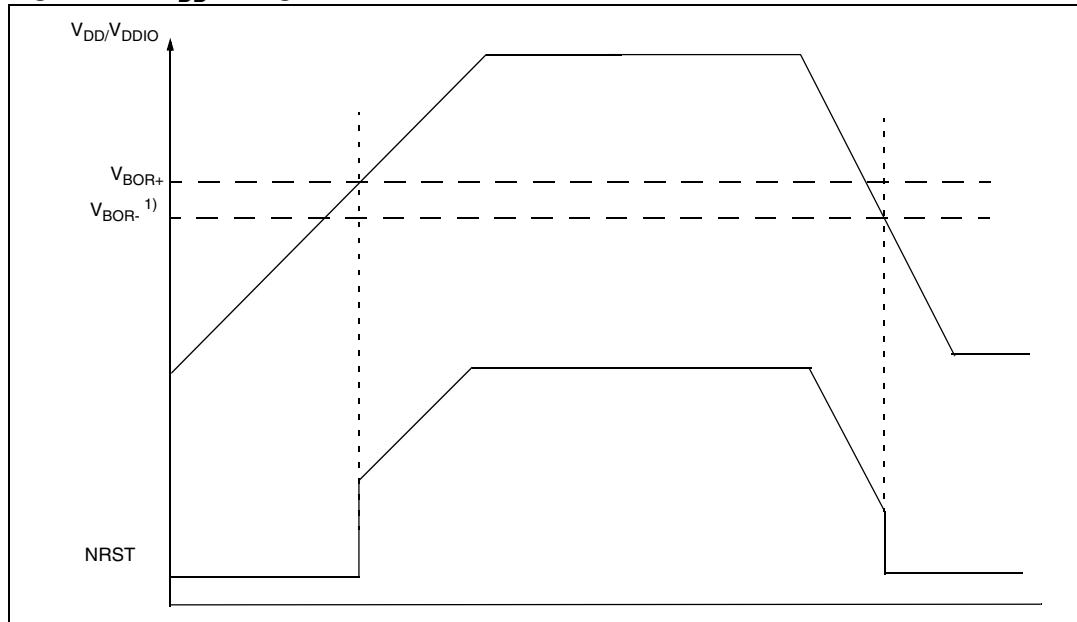
This BOR threshold ( $V_{BOR}$ ) can be configured using the option bytes. Five different levels can be selected, starting from 1.8 V to 3 V . If the BOR is disabled, the reset threshold is  $V_{PDR}$  at power-down: this guarantees a  $V_{DD}$  min. value of 1.65 V<sup>(1)</sup>. Refer to [Figure 14](#).

The power control unit (PWR) has an internal reference voltage which is used by the BOR or by other analog features.

To reduce the power consumption in Halt mode, it is possible to automatically switch off the internal voltage reference using the ULP bit in the PWR\_CSR2 register. In this case, when the MCU wakes up from Halt mode there is an additional delay for the internal reference voltage startup. This delay can be configured using the FWU bit in the PWR\_CSR2 register (see [Table 11](#)).

Enabling the BOR guarantees that the MCU is safely used as it cannot run when  $V_{DD}$  is below the operating range.

1. Not applicable on STM8L05xx value line devices. Refer to datasheets for more details.

**Figure 14.**  $V_{DD}$  voltage detection: BOR threshold

1. When BOR is disabled at power-down, this threshold is  $V_{PDR}$ .

## 6.4

## Programmable voltage detector (PVD)

The device features an embedded programmable voltage detector (PVD) that monitors the  $V_{DD}/V_{DDA}$  power supply and compares it to the  $V_{PVD}$  threshold.

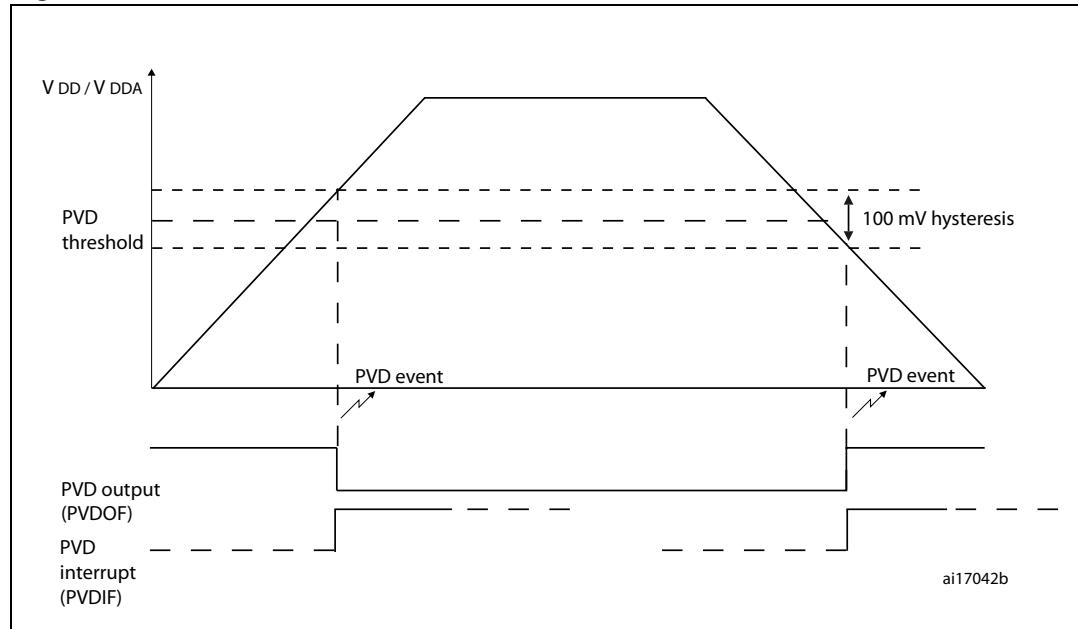
When  $V_{DD}/V_{DDA}$  drops below the  $V_{PVD}$  threshold and/or when  $V_{DD}/V_{DDA}$  rises higher than the  $V_{PVD}$  threshold, the PVD output changes and a PVD event is generated. Every PVD event sets the interrupt flag (PVDIF) in the PWR\_CSR1 register. The current state of the PVD output can be monitored through the PVDOF flag status in the PWR\_CSR1 register. If the PVDIEN bit in the PWR\_CSR1 register is set, the PVD interrupt is enabled and is generated when the PVDIF flag is set. To prevent some spurious interrupts, as the PVDIF flag must be cleared by software, the user must make sure the PVDIF flag has been cleared before the PVD interrupt is enabled and before returning from the interrupt service. The PVD is enabled by software by setting the PVDE bit in the PWR\_CSR1 register.

This PVD has 7 different levels between 1.85 V and 3.05 V, selected by the PLS[2:0] bits in the PWR\_CSR1 register, in steps of around 200 mV. Refer to the “Power-up / power-down operating conditions” section in the product datasheet.

The external reference voltage on the PVD\_IN input pin can be selected by setting the PLS[2:0] bits to 111.

*Note:*

*The voltage injected on PVD\_IN (PE6) is compared to the  $V_{REFINT}$  internal voltage reference.*

**Figure 15. PVD threshold**

## 6.5 Internal voltage reference (VREFINT)

The functions managed through the internal voltage reference ( $V_{REFINT}$ ) are BOR, PVD, ADC, LCD internal VLCD booster and comparators. The internal voltage reference ( $V_{REFINT}$ ) is always enabled. The internal voltage reference consumption is not negligible in Halt and Active-halt mode. To reduce power consumption in these modes, the ULP bit (Ultra low power) in the PWR\_CSR2 register can be set to disable the internal voltage reference during Halt and Active-halt modes. In this case, when exiting from these modes, the functions managed through the internal voltage reference are not reliable during the internal voltage reference startup time  $t_{VREFINT}$  (up to 3 ms, refer to datasheet). To reduce the wakeup time, the device can exit from Halt/Active-halt mode without waiting for the internal voltage reference startup time. This is performed by setting the FWU bit (Fast wakeup) in the PWR\_CSR2 register before entering halt/Active-halt mode. The  $V_{REFINTF}$  flag in the PWR\_CSR register indicates that the internal voltage reference is ready. Consequently, the analog functions listed above can be used.

**Table 11. Internal voltage reference status during Halt/Active-halt mode<sup>(1)</sup>**

ULP bit	FWU bit	Meaning	
0	x	Internal reference voltage always on. Internal reference voltage consumption ( $I_{REFINT}$ ) is added in Halt/Active-halt mode. <sup>(2)</sup>	
1	0	Internal reference voltage stopped in Halt/Active-halt mode	Internal reference voltage wakeup time $t_{VREFINT}$ is added to allow the internal reference voltage start. <sup>(2)</sup>
1	1		Fast wakeup time is forced without waiting the start of the internal reference voltage. In this case, the analog features do not work immediately after wakeup.

1. The FWU bit or the ULP bit must not be changed before the internal reference voltage is properly stabilized. The VREFINTF bit in the PWR\_CSR2 register can be used to check the status of the internal reference voltage.
2. Refer to the product datasheet for the values of wakeup times.

## 6.6 Voltage regulator

The device has an internal voltage regulator for generating the 1.8 V power supply ( $V_{CORE}$ ) for the core and peripherals.

This regulator has two different modes:

- Main voltage regulator mode (MVR) for Run, Wait for interrupt (WFI) and Wait for event (WFE) modes.
- Ultralow power regulator mode (ULP) for Halt and Active-halt modes.

The regulator mode switches modes automatically, depending in some cases on some register control bits. Refer to [Table 14](#).

**Table 12. PWR interrupt requests**

Interrupt event	Event flag	Enable control bit	Exit from Wait/Low power wait	Exit from Halt/Active-halt
PVD interrupt flag	PVDIF	PVDIEN	No	Yes

## 6.7 PWR registers

### 6.7.1 Power control and status register 1 (PWR\_CSR1)

Address offset: 0x00

Reset value after power-on Reset: 0x00

7	6	5	4	3	2	1	0
Reserved	PVDOF	PVDIF	PVDIEN	PLS[2:0]			PVDE
	r	rc_w1	rw	rw	rw	rw	rw

Bit 7 Reserved, must be kept cleared.

Bit 6 **PVDOF**: PVD output flag

This bit is set and cleared by hardware.

0:  $V_{DD}/V_{DDA}$  is above the  $V_{PVD}$  threshold

1:  $V_{DD}/V_{DDA}$  is below the  $V_{PVD}$  threshold

Bit 5 **PVDIF**: PVD interrupt flag

This bit is set by hardware and cleared by software writing “1”.

0: No PVD event occurred

1: PVD interrupt pending. This bit is set by hardware when a PVD event occurs.

Bit 4 **PVDIEN**: PVD interrupt enable

This bit is set and cleared by software.

0: PVD interrupt disabled

1: PVD interrupt enabled

Bits 3:1 **PLS[2:0]**: PVD level selection

These bits are set and cleared by software.

000: PVD threshold = 1.85 V typ.

001: PVD threshold = 2.05 V typ.

010: PVD threshold = 2.26 V typ.

011: PVD threshold = 2.45 V typ.

100: PVD threshold = 2.65 V typ.

101: PVD threshold = 2.85 V typ.

110: PVD Threshold = 3.05 V typ.

111: Threshold = PVD\_IN input pin

For more details, refer to the “Power-up / power-down operating conditions” section in the product datasheet.

Bit 0 **PVDE**: Power voltage detector (PVD) enable

This bit is set and cleared by software.

0: PVD off

1: PVD on

## 6.7.2 PWR control and status register 2 (PWR\_CSR2)

Address offset: 0x01

Reset value after power-on Reset: 0x00

7	6	5	4	3	2	1	0
Reserved					FWU	ULP	VREFINTF
					rw	rw	r

Bits 7:3 Reserved, must be kept cleared.

Bit 2 **FWU**: Fast wake up configuration

Refer to [Table 11](#).

*Note:* The FWU bit must not be changed before the internal reference voltage is properly stabilized. The VREFINTF bit in the PWR\_CSR2 register can be used to check the status of the internal reference voltage.

Bit 1 **ULP**: Ultralow power configuration

Refer to [Table 11](#).

*Note:* The FWU bit must not be changed before the internal reference voltage is properly stabilized. The VREFINTF bit in the PWR\_CSR2 register can be used to check the status of the internal reference voltage.

Bit 0 **VREFINTF**: Internal reference voltage status flag

This bit is set and cleared by hardware

0: Internal reference voltage off or not yet stable

1: Internal reference voltage on

## 6.7.3 PWR register map and reset values

**Table 13. PWR register map and reset values**

Address offset <sup>(1)</sup>	Register name	7	6	5	4	3	2	1	0
0x00	PWR_CSR1	-	PVDOF	PVDIF	PVDIEN	PLS2	PLS1	PLS0	PVDE
	Reset value	0	0	0	0	0	0	0	0
0x01	PWR_CSR2	-	-	-	-	-	FWU	ULP	VREFINTF
	Reset value	0	0	0	0	0	0	0	0

1. Please refer to the “general hardware register map” table in the datasheet for details on register addresses.

## 7 Low power modes

By default, the microcontroller is in Run mode after a system or power reset. However the device supports five low power modes to achieve the best compromise between low power consumption, short startup time and available wakeup sources:

- **Wait mode:** The CPU clock is stopped, but selected peripherals keep running. An internal or external interrupt, event or Reset can be used to exit the microcontroller from Wait mode. Refer to [Section 7.3: Wait mode \(WFI or WFE mode\) on page 73](#)
- **Low power run mode:** The CPU and the selected peripherals are running. Execution is done from RAM with a low speed oscillator (LSI or LSE). Flash and data EEPROM are stopped and the voltage regulator is configured in Ultra Low Power mode. The microcontroller enters Low power run mode by software and can exit from this mode by software or by a reset.  
All interrupts must be masked. They cannot be used to exit the microcontroller from this mode.
- **Low power wait mode:** This mode is entered when executing a Wait for event in Low power run mode. It is similar to Low power run mode except that the CPU clock is stopped. The wakeup from this mode is triggered by a Reset or by an internal or external event (peripheral event generated by the timers, serial interfaces, DMA controller, comparators and I/O ports). When the wakeup is triggered by an event, the system goes back to Low power run mode.  
All interrupts must be masked. They cannot be used to exit the microcontroller from this mode.
- **Active-halt mode:** The CPU and peripheral clocks are stopped, except the RTC. The wakeup can be triggered by RTC interrupts, external interrupts or reset.
- **Halt mode:** The CPU and peripheral clocks are stopped, the device remains powered on. The wakeup is triggered by an external interrupt or reset. A few peripherals also have wakeup from Halt capability. Switching off the internal reference voltage further reduces power consumption. Through software configuration (see [Table 11](#)) it is also possible to have a fast wakeup time of 6 µs, without waiting for the internal reference voltage startup time.

**Table 14. Low power mode summary**

Mode	Entry	Oscillator	CPU	Peripheral	Wakeup	Voltage regulator mode
Wait	WFI	On	Off	On	All internal or external interrupts, reset	MVR
	WFE	On	Off	On	All internal or external interrupts, wakeup events, reset	MVR
Low power run mode	Software sequence	LSI or LSE clock	On	On	Software sequence, reset	ULP
Low power wait mode	Software sequence +WFE	LSI or LSE clock	Off	On	Internal or external event, reset	ULP

**Table 14. Low power mode summary (continued)**

Mode	Entry	Oscillator	CPU	Peripheral	Wakeup	Voltage regulator mode
Active-halt	HALT <sup>(1)</sup>	Off except LSI or LSE clock	Off	Off except RTC and possibly LCD	External interrupts, RTC interrupt, reset	MVR/ULP depending on CLK_ICKCR register
Halt	HALT <sup>(1)</sup>	Off	Off	Off	External interrupts, reset	ULP

1. Before executing the HALT instruction, the application must clear any pending peripheral interrupt by clearing the interrupt pending bit in the corresponding peripheral configuration register. Otherwise, the HALT instruction is not executed and program execution continues.

In addition, the power consumption in Run mode can be reduced by one of the following means:

- Slowing down the system clocks
- Gating the peripherals clocks when they are unused.

## 7.1 Slowing down the system clocks

In Run mode, choosing the clock frequency is very important to ensure the best compromise between performance and consumption. The selection is done by programming the prescaler registers. These prescalers can also be used to slow down peripherals before entering low power modes.

*Note:* *In applications where the MCU can be halted for certain periods, the power consumption can be minimized by keeping a fast clock (high performance execution) during active periods, in order to reduce the ratio between active periods and Halt (that is “zero”-consumption) periods.*

## 7.2 Peripheral clock gating (PCG)

For additional power saving, you can use Peripheral Clock Gating (PCG). This can be done at any time by selectively enabling or disabling the SYSCLK clock connection to individual peripherals. These settings are effective in both Run and Wait modes.

Each PCG state represents a specific power or low power level. It is controlled by the Peripheral clock enable registers (CLK\_PCKENR1, CLK\_PCKENR2 & CLK\_PCKENR3).

## 7.3 Wait mode (WFI or WFE mode)

Wait mode is entered from Run mode by executing a WFI (Wait For Interrupt) or WFE (Wait For Event) instruction: this stops the CPU but allows the other peripherals and interrupt controller to continue running. The consumption decreases accordingly. Wait mode can be combined with PCG to further reduce power consumption of the device.

In Wait mode, all the registers and RAM contents are preserved and the clock configuration selected through the Clock divider register (CLK\_CKDIVR) remains unchanged.

## 7.4 Wait for interrupt (WFI) mode

Wait for interrupt mode is entered from Run mode by executing a WFI (Wait For Interrupt) instruction.

When an internal or external interrupt request occurs, the CPU wakes up from WFI mode, serves the interrupt and resumes processing.

*Note:* *In an interrupt based application, where most of the processing is done through the interrupt routines, the main program may be suspended by setting the activation level bit (AL) in the CFG\_GCR register. Setting this bit causes the CPU to return to WFI mode without restoring the main execution context. This saves power by removing both the save/restore context activity and the need for a main software loop execution for power management (in order to return to WFI mode).*

## 7.5 Wait for event (WFE) mode

Wait for event mode is entered from Run mode by executing a WFE instruction.

Interrupt requests during this mode are served normally, depending on the value of the I0 and I1 bits in the CPU\_CC register.

Peripheral events can be generated by the timers, serial interfaces, DMA controller, comparators and I/O ports. These are enabled by the WFE\_CRx registers.

When a peripheral event is enabled, the corresponding interrupts are not served and you have to clear the corresponding flag status.

There are two ways to wake up the CPU from WFE mode:

- **Interrupts:** when an interrupt occurs, the CPU wakes up from WFE mode and serves the interrupt. After processing the interrupt, the processor goes back to WFE mode.
- **Wakeup events:** when a wakeup event occurs, the CPU wakes up and resumes processing. As the processing resumes directly after the WFE instruction, there is no context save/restore activity (this saves time and power consumption).

Further power consumption reduction may be achieved using this mode together with execution from RAM. In some very low power applications, when the main software routine is short and has a low execution time, this routine can be moved to RAM and executed from RAM. As the Flash program memory is not used at wakeup, the power consumption is then reduced during run time.

At any time, another routine (stored in the Flash program memory) can be executed by software by simply calling/jumping to this routine.

## 7.5.1 WFE registers

### WFE control register 1 (WFE\_CR1)

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
EXTI_EV3	EXTI_EV2	EXTI_EV1	EXTI_EV0	TIM1_EV1	TIM1_EV0	TIM2_EV1	TIM2_EV0
rw							

#### Bit 7 **EXTI\_EV3**: External interrupt event 3

This bit is written by software to select the external interrupt sources used to wake up the CPU from WFE mode.

0: No wakeup event generated on external interrupts from pin 3 of all ports

1: External interrupts from pin 3 of all ports configured to generate wakeup events

#### Bit 6 **EXTI\_EV2**: External interrupt event 2

This bit is written by software to select the external interrupt sources used to wake up the CPU from WFE mode.

0: No wakeup event generated on external interrupts from pin 2 of all ports

1: External interrupts from pin 2 of all ports configured to generate wakeup events

#### Bit 5 **EXTI\_EV1**: External interrupt event 1

This bit is written by software to select the external interrupt sources used to wake up the CPU from WFE mode.

0: No wakeup event generated on external interrupts from pin 1 of all ports

1: External interrupts from pin 1 of all ports configured to generate wakeup events

#### Bit 4 **EXTI\_EV0**: External interrupt event 0

This bit is written by software to select the external interrupt sources used to wake up the CPU from WFE mode.

0: No wakeup event generated on external interrupts from pin 0 of all ports

1: External interrupts from pin 0 of all ports configured to generate wakeup events

#### Bit 3 **TIM1\_EV1**: TIM1 interrupt event 1

This bit is written by software to configure the TIM1 interrupt sources used to wake up the CPU from WFE mode.

0: No wakeup event generated on TIM1 capture and compare interrupts

1: TIM1 capture and compare interrupts logically ORed and configured to generate wakeup events

#### Bit 2 **TIM1\_EV0**: TIM1 interrupt event 0

This bit is written by software to configure the TIM1 interrupt sources used to wake up the CPU from WFE mode.

0: No wakeup event generated on TIM1 update, trigger and break interrupts

1: TIM1 update, trigger and break interrupts logically ORed and configured to generate wakeup events

#### Bit 1 **TIM2\_EV1**: TIM2 interrupt event 1

This bit is written by software to configure the TIM2 interrupt sources used to wake up the CPU from WFE mode.

0: No wakeup event generated on TIM2 capture and compare interrupts

1: TIM2 capture and compare interrupts logically ORed and configured to generate wakeup events

Bit 0 **TIM2\_EVO:** TIM2 interrupt event 0

This bit is written by software to configure the TIM2 interrupt sources used to wake up the CPU from WFE mode

0: No wakeup event generated on TIM2 update, trigger and break interrupts

1: TIM2 update, trigger and break interrupts logically ORed and configured to generate wakeup events

**WFE control register 2 (WFE\_CR2)**

Address offset: 0x01

Reset value: 0x00

7	6	5	4	3	2	1	0
ADC1_COMP_EV	EXTI_EVE_F	EXTI_EVD_H	EXTI_EVB_G	EXTI_EV7	EXTI_EV6	EXTI_EV5	EXTI_EV4
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **ADC1\_COMP\_EV**

This bit is written by software to select the interrupt sources from ADC1 and comparators used to wake up the CPU from WFE mode.

0: No wakeup event generated on interrupts from ADC1 and comparators

1: Interrupts from ADC1 and comparators are configured to generate wakeup events

Bit 6 **EXTI\_EVE\_F:** External interrupt event on Port E or Port F

This bit is written by software to select the external interrupt sources on Port E or Port F used to wake up the CPU from WFE mode.

0: No wakeup event generated on external interrupts from Port E or Port F

1: External interrupts from Port E or Port F configured to generate wakeup events

Bit 5 **EXTI\_EVD:** External interrupt event on Port D or Port H<sup>(1)</sup>

This bit is written by software to select the external interrupt sources on Port D or Port H<sup>(1)</sup> used to wake up the CPU from WFE mode.

0: No wakeup event generated on external interrupts from Port D or Port H<sup>(1)</sup>

1: External interrupts from Port D or Port H<sup>(1)</sup> configured to generate wakeup events

Bit 4 **EXTI\_EVB:** External interrupt event on Port B or Port G<sup>(2)</sup>

This bit is written by software to select the external interrupt sources on Port B or Port G<sup>(2)</sup> used to wake up the CPU from WFE mode.

0: No wakeup event generated on external interrupts from Port B or Port G<sup>(2)</sup>

1: External interrupts from Port B or Port G<sup>(2)</sup> configured to generate wakeup events

Bit 3 **EXTI\_EV7:** External interrupt event 7

This bit is written by software to select the external interrupt sources used to wake up the CPU from WFE mode.

0: No wakeup event generated on external interrupts from pin 7 of all ports

1: External interrupts from pin 7 of all ports configured to generate wakeup events

Bit 2 **EXTI\_EV6:** External interrupt event 6

This bit is written by software to select the external interrupt sources used to wake up the CPU from WFE mode.

0: No wakeup event generated on external interrupts from pin 6 of all ports

1: External interrupts from pin 6 of all ports configured to generate wakeup events

**Bit 1 EXTI\_EV5:** External interrupt event 5

This bit is written by software to select the external interrupt sources used to wake up the CPU from WFE mode.

- 0: No wakeup event generated on external interrupts from pin 5 of all ports
- 1: External interrupts from pin 5 of all ports configured to generate wakeup events

**Bit 0 EXTI\_EV4:** External interrupt event 4

This bit is written by software to select the external interrupt sources used to wake up the CPU from WFE mode.

- 0: No wakeup event generated on external interrupts from pin 4 of all ports
- 1: External interrupts from pin 4 of all ports configured to generate wakeup events

1. External interrupt on port D for medium density devices and on port D or H for low, medium+ and high density devices.

2. External interrupt on port B for medium density devices and on port B or G for low, medium+ and high density devices.

**WFE control register 3 (WFE\_CR3)**

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
DMA1CH23_EV	DMA1CH01_EV	USART1_EV	I2C1_EV	SPI1_EV	TIM4_EV	TIM3_EV1	TIM3_EV0
rw	rw	rw	rw	rw	rw	rw	rw

**Bit 7 DMA1CH23\_EV:** DMA1 channel 2 and 3 interrupt event

This bit is written by software to select the DMA1 channel 2 and 3 interrupt sources used to wake up the CPU from WFE mode.

- 0: No wakeup event generated on DMA1 channel 2 and 3
- 1: DMA1 channel 2 and 3 interrupts configured to generate wakeup events

**Bit 6 DMA1CH01\_EV:** DMA1 channel 0 and 1 interrupt event

This bit is written by software to select the DMA1 channel 0 and 1 interrupt sources used to wake up the CPU from WFE mode.

- 0: No wakeup event generated on DMA1 channel 0 and 1
- 1: DMA1 channel 0and 1interrupts configured to generate wakeup events

**Bit 5 USART1\_EV:** USART1 Rx and Tx interrupt event

This bit is written by software to select the USART1 Rx and Tx interrupt sources used to wake up the CPU from WFE mode

- 0: No wakeup event generated on USART1 Rx and Tx
- 1: USART1 Rx and Tx interrupts configured to generate wakeup events

**Bit 4 I2C1\_EV:** I2C1 Rx and Tx interrupt event

This bit is written by software to select the I2C1 Rx and Tx interrupt sources used to wake up the CPU from WFE mode

- 0: No wakeup event generated on I2C1 Rx and Tx
- 1: I2C1 Rx and Tx interrupts configured to generate wakeup events

**Bit 3 SPI1\_EV:** SPI1Rx and Tx interrupt event

This bit is written by software to select the SPI1Rx and Tx interrupt sources used to wake up the CPU from WFE mode

0: No wakeup event generated on SPI1Rx and Tx

1: SPI1Rx and Tx interrupts configured to generate wakeup events

**bit 2 TIM4\_EV:** TIM4 interrupt event

This bit is written by software to configure the TIM4 interrupt sources used to wake up the CPU from WFE mode

0: No wakeup event generated on TIM4 update and trigger interrupts

1: TIM4 update and trigger interrupts logically ORed and configured to generate wakeup events

**Bit 1 TIM3\_EV1:** TIM3 interrupt event 1

This bit is written by software to configure the TIM3 interrupt sources used to wake up the CPU from WFE mode.

0: No wakeup event generated on TIM3 capture and compare interrupts

1: TIM3 capture and compare interrupts logically ORed and configured to generate wakeup events

**Bit 0 TIM3\_EV0:** TIM3 interrupt event 0

This bit is written by software to configure the TIM3 interrupt sources used to wake up the CPU from WFE mode.

0: No wakeup event generated on TIM3 update, trigger and break interrupts

1: TIM3 update, trigger and break interrupts logically ORed and configured to generate wakeup events

### WFE control register 4 (WFE\_CR4)

Address offset: 0x03

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved	AES_EV	TIM5_EV1	TIM5_EV0	USART3_EV	USART2_EV	SPI2_EV	RTC_CSSLSE_EV	
rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 Reserved

Bit 6 **AES\_EV**: AES interrupt event

This bit is written by software to configure the AES interrupt source used to wake up the CPU from WFE mode.

- 0: No wakeup event generated from AES peripheral
- 1: AES interrupt configured to generate wakeup event

Bit 5 **TIM5\_EV1**: TIM5 interrupt event 1

This bit is written by software to configure the TIM5 interrupt sources used to wake up the CPU from WFE mode.

- 0: No wakeup event generated on TIM5 capture and compare interrupts
- 1: TIM5 capture and compare interrupts logically ORed and configured to generate wakeup events

Bit 4 **TIM5\_EV0**: TIM5 interrupt event 0

This bit is written by software to configure the TIM5 interrupt sources used to wake up the CPU from WFE mode.

- 0: No wakeup event generated on TIM5 update, trigger and break interrupts
- 1: TIM5 update, trigger and break interrupts logically ORed and configured to generate wakeup events

Bit 3 **USART3\_EV**: USART3 Rx and Tx interrupt event

- 0: No wakeup event generated from USART3 Rx and Tx
- 1: USART3 Rx and Tx interrupts configured to generate wakeup events

Bit 2 **USART2\_EV**: USART2 Rx and Tx interrupt event

This bit is written by software to select the USART2 Rx and Tx interrupt sources used to wake up the CPU from WFE mode

- 0: No wakeup event generated from USART2 Rx and Tx
- 1: USART2 Rx and Tx interrupts configured to generate wakeup events

Bit 1 **SPI2\_EV**: SPI2 Rx and Tx interrupt event

This bit is written by software to select the SPI2 Rx and Tx interrupt sources used to wake up the CPU from WFE mode.

- 0: No wakeup event generated from SPI2 Rx and Tx
- 1: SPI2 Rx and Tx interrupts configured to generate wakeup events

Bit 0 **RTC\_CSSLSE\_EV**: “RTC” or “CSS on LSE” interrupt event

This bit is written by software to select the interrupt source (“RTC” or “CSS on LSE”) used to wake up the CPU from WFE mode.

- 0: No wakeup event generated from RTC or CSS on LSE interrupt events
- 1: RTC or CSS on LSE interrupt events configured to generate wakeup events

## 7.5.2 WFE register map and reset values

**Table 15. WFE register map**

Address offset <sup>(1)</sup>	Register name	7	6	5	4	3	2	1	0
0x00	WFE_CR1 Reset value	EXTI_EV3_0	EXTI_EV2_0	EXTI_EV1_0	EXTI_EVO_0	TIM1_EV1_0	TIM1_EVO_0	TIM2_EV1_0	TIM2_EVO_0
0x01	WFE_CR2 Reset value	ADC1_CO_MP_EV_0	EXTI_EVE_F_0	EXTI_EVD_0	EXTI_EVB_0	EXTI_EV7_0	EXTI_EV6_0	EXTI_EV5_0	EXTI_EV4_0
0x02	WFE_CR3 Reset value	DMA1CH23_EV_0	DMA1CH01_EV_0	USART1_EV_0	I2C1_EV_0	SPI1_EV_0	TIM4_EV_0	TIM3_EV1_0	TIM3_EVO_0
0x03 <sup>(2)</sup>	WFE_CR4 Reset value	Reserved	AES_EV_0	TIM5_EV1_0	TIM5_EVO_0	USART3_EV_0	USART2_EV_0	SPI2_EV_0	RTC_CSSLSE_EV_0

1. Please refer to the “general hardware register map” table in the datasheet for details on register addresses.
2. Available on low, medium+ and high density devices only. On low density devices, only the RTC\_CSSLSE\_EV bit is available in this register.

## 7.6 Low power run mode

This mode is based on code fetch from RAM with the regulator in Ultralow power mode (ULP) and EEPROM in IDDQ.

While using this mode you have to switch from high speed clock sources to low speed clock sources.

*Note:* *The ADC cannot be used in Low Power Run mode and must be disabled.*

### 7.6.1 Entering Low power run mode

This mode is entered by executing the following software sequence:

1. Jump to RAM
2. Switch system clock to LSI or LSE clock sources
3. Switch off the high speed oscillators, the ADC and all unused peripherals
4. Mask all interrupts
5. Switch off the Flash/Data EEPROM by setting EEPM bit in FLASH\_CR1 register
6. Add a software delay loop to ensure Flash/Data EEPROM off status
7. Configure the ultralow power mode for the regulator by setting the REGOFF bit in the CLK\_REGCSR register

### 7.6.2 Exiting Low power run mode

The only way to exit this mode is to follow these steps:

1. Switch on the main regulator by resetting the REGOFF bit in the CLK\_REGCSR register. The REGREADY flag in the CLK\_REGCSR register is set when the regulator is ready.
2. Switch on the Flash/Data EEPROM by resetting EEPM bit in FLASH\_CR1 register. The EEREADY flag in the CLK\_REGCSR register is set when the Flash/Data EEPROM is ready.
3. Reset interrupt mask.
4. Switch on what is necessary and jump to Flash/Data EEPROM if needed.

## 7.7 Low power wait mode

This mode is entered by executing a WFE instruction when the MCU is in Low power run mode. It can be exited only by means of an external or internal event. In this case the MCU returns to Low power run mode.

## 7.8 Halt mode

In this mode the system clock is stopped. This means that the CPU and all the peripherals clocked by SYSCLK or by derived clocks are disabled, except for the following cases:

- The HSI clock is not stopped if used by SWIM
- The system clock source is not stopped if a Flash/Data EEPROM write operation is in progress
- The LSI clock is not stopped if used by the SWIM, the IWDG or if the “IWDG\_HALT” option bit is disabled.

In Halt mode, none of the peripherals are clocked and the digital part of the MCU consumes almost no power.

### 7.8.1 Entering Halt mode

The MCU enters Halt mode when a HALT instruction is executed.

*Note:*

*It is recommended not to enter Halt/Active-halt mode from the Low power run mode. Otherwise, the only safe way to exit one of these two modes is to reset the MCU.*

Before executing a HALT instruction, the application must clear all pending peripheral interrupts by clearing the interrupt pending bit in the corresponding peripheral configuration register. Otherwise, the HALT instruction is not executed and program execution continues.

However, the Halt procedure can be delayed if one of the following flags is set:

- SWBSY flag in the CLK\_SWCR register
- EEBUSY flag in the CLK\_CLK\_REGCSR register
- RTCWSBSY flag in the CLK\_CRTCR register
- BEEPSWBSY flag in the CLK\_CBEPR register when BEEP in Active-halt mode enabled.

If SAHALT bit is set in the CLK\_ICKCR register the main regulator (MVR) will be switched off without taking into account that some high-speed clock may be used by the system.

### 7.8.2 Exiting Halt mode

Wakeup from Halt mode is triggered by an external interrupt sourced by a general purpose I/O port configured as interrupt input or by an alternate function pin capable of triggering a peripheral interrupt.

The system clock is restarted with a frequency depending on the FHW bit in the CLK\_ICKCR register. If the FHW bit is set, HSI/8 is the system clock, otherwise the system clock is the last selected clock source before entering Halt mode.

In an interrupt based application, where most of the processing is done through the interrupt routines, the main program may be suspended by setting the activation level bit (AL) in the CPU configuration register. Setting this bit causes the CPU to return to Halt mode when executing the return from interrupt, without restoring the main execution context.

Power consumption is reduced as there is no save/restore context activity and no need for a main software loop execution for power management (in order to return to WFI mode).

After a wake up from Halt mode, the LCD clock switches from RTCCLK to SYSCLK. To have a stable clock signal without glitches, 2 RTCCLK cycles are needed for synchronization reasons. Consequently, read/write access to LCD registers is not possible during this period.

## 7.9 Active-halt mode

Active-halt mode is similar to Halt mode.

*Note:* *It is recommended not to enter Halt/Active-Halt mode from the Low power run mode. Otherwise, the only safe way to exit one of these two modes is to reset the MCU.*

In Active-halt mode, the main oscillator, the CPU and almost all peripherals are stopped. Only oscillator or the LSE crystal is running to drive the SWIM, beeper, IWDG, RTC and LCD if enabled.

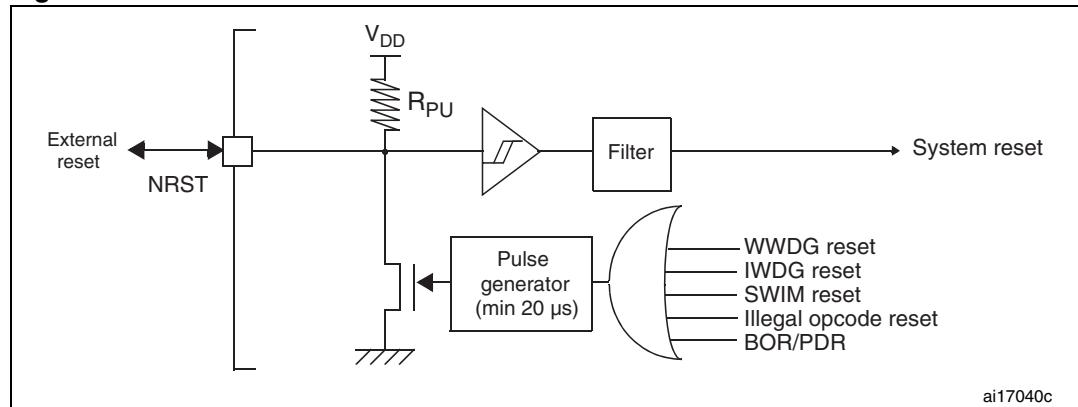
## 8 Reset (RST)

There are 6 reset sources:

- External reset through the NRST pin (this pin can also be configured as general purpose output)
- Power-on reset (POR)/ power-down reset (PDR)
- Independent watchdog reset (IWDG)
- Window watchdog reset (WWDG)
- Illegal opcode reset (ILLOP)
- SWIM reset

These sources act on the NRST pin. The reset service routine vector is fixed at address 0x8000 in the memory map.

**Figure 16. Reset circuit**



ai17040c

### 8.1 “Reset state” and “under reset” definitions

When a reset occurs, there is a reset phase from the external pin pull-down to the internal reset signal release. During this phase, the microcontroller sets some hardware configurations before going to the reset vector.

At the end of this phase, most of the registers are configured with their “reset state” values. During the reset phase, i.e. “under reset”, some pin configurations may be different from their “reset state” configuration.

### 8.2 External reset (NRST pin)

#### 8.2.1 Asynchronous external reset description

The NRST pin is both an input and an open-drain output with an integrated R<sub>PU</sub> weak pull-up resistor.

A low pulse of minimum 300 ns on the NRST pin is needed to generate an external reset. The reset detection is asynchronous and therefore the MCU can enter reset even in Halt mode.

The NRST pin also acts as an open-drain output for resetting external devices.

Any reset pulse from an internal reset source holds the NRST output low for at least 20 µs. The internal R<sub>P</sub>U weak pull-up ensures a high level on the NRST pin when the reset is not forced low.

## 8.2.2 Configuring NRST/PA1 pin as general purpose output

To optimize the number of available pins, the NRST pin (external reset) can be configured as a general purpose push-pull output (PA1).

For security, this configuration can be performed once only after reset, by writing a specified key (0xD0) to the [Reset pin configuration register \(RST\\_CR\)](#).

When the PA1 pin is configured as a general purpose output, the MCU can be reset only by a power-on reset (POR) or other internal reset source.

## 8.3 Internal reset

For internal reset sources, the NRST pin is kept low during the delay phase generated by the pulse generator.

Each internal reset source is linked to a specific flag bit in the [Reset status register \(RST\\_SR\)](#). These flags are set at reset state depending on the given reset source. Consequently, they can be used to identify the last reset source. They are cleared by software by writing the logic value '1'.

*Note:* All flags besides the POR flag are reset at POR.

### 8.3.1 Power-on reset (POR)

During power-on, the POR keeps the device in reset mode until V<sub>DD</sub> reaches a specified threshold. The POR reset is then held for a specified time to ensure that V<sub>DD</sub> has reached the minimum operating voltage. See Electrical parameters section in the datasheet for more details.

### 8.3.2 Independent watchdog reset

Refer to the independent watchdog chapter for details.

A reset can be triggered by the application software using the Independent watchdog.

### 8.3.3 Window watchdog reset

Refer to the window watchdog chapter for details.

### 8.3.4 SWIM reset

An external device connected to the SWIM interface can request the SWIM block to generate an MCU reset.

### 8.3.5 Illegal opcode reset

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

## 8.4 RST registers

### 8.4.1 Reset pin configuration register (RST\_CR)

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
RSTPIN_KEY							
rwo	rwo	rwo	rwo	rwo	rwo	rwo	rwo

Bits 7:0

**RSTPIN\_KEY[7:0]:** Reset pin configuration key

0x00: NRST/PA1 configured as reset pin (default reset value)

0xD0: NRST/PA1 configured as general purpose output

These bits are write once only. They can also be read at any time.

*Note:* Writing any value beside 0xD0 is equivalent to writing 0x00.

### 8.4.2 Reset status register (RST\_SR)

Address offset: 0x01

Reset value after power-on Reset: 0x01

7	6	5	4	3	2	1	0
Reserved	BORF	WWDGF	SWIMF	ILLOPF	IWDGF	PORF	
	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 7:6 Reserved, must be kept cleared.

Bit 5 **BORF:** Brownout reset flag

This bit is set by hardware and cleared by software writing “1”.

0: No BOR reset occurred

1: An BOR reset occurred

Bit 4 **WWDGF:** Window Watchdog reset flag

This bit is set by hardware and cleared by software writing “1”.

0: No WWDG reset occurred

1: A WWDG reset occurred

Bit 3 **SWIMF:** SWIM reset flag

This bit is set by hardware and cleared by software writing “1”.

0: No SWIM reset occurred

1: A SWIM reset occurred

Bit 2 **ILLOPF:** Illegal opcode reset flag

This bit is set by hardware and cleared by software writing “1”.

0: No ILLOP reset occurred

1: An ILLOP reset occurred

Bit 1 **IWDGF:** Independent Watchdog reset flag

This bit is set by hardware and cleared by software writing “1”.

0: No IWDG reset occurred

1: An IWDG reset occurred

Bit 0 **PORF**: Power-on Reset (POR) flag

This bit is set by hardware and cleared by software writing “1”.

0: No POR occurred

1: A POR occurred

### 8.4.3 RST register map and reset values

**Table 16. RST register map and reset values**

Address offset <sup>(1)</sup>	Register name	7	6	5	4	3	2	1	0
0x00	RST_CR Reset value	RSTPIN _KEY7 0	RSTPIN _KEY6 0	RSTPIN _KEY5 0	RSTPIN _KEY4 0	RSTPIN _KEY3 0	RSTPIN _KEY2 0	RSTPIN _KEY1 0	RSTPIN _KEY0 0
0x01	RST_SR Reset value	- 0	- 0	BORF 0	WWDG 0	SWIMF 0	ILLOPF 0	IWDGF 0	PORF 1

1. Please refer to the “general hardware register map” table in the datasheet for details on register addresses.

## 9 Clock control (CLK)

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

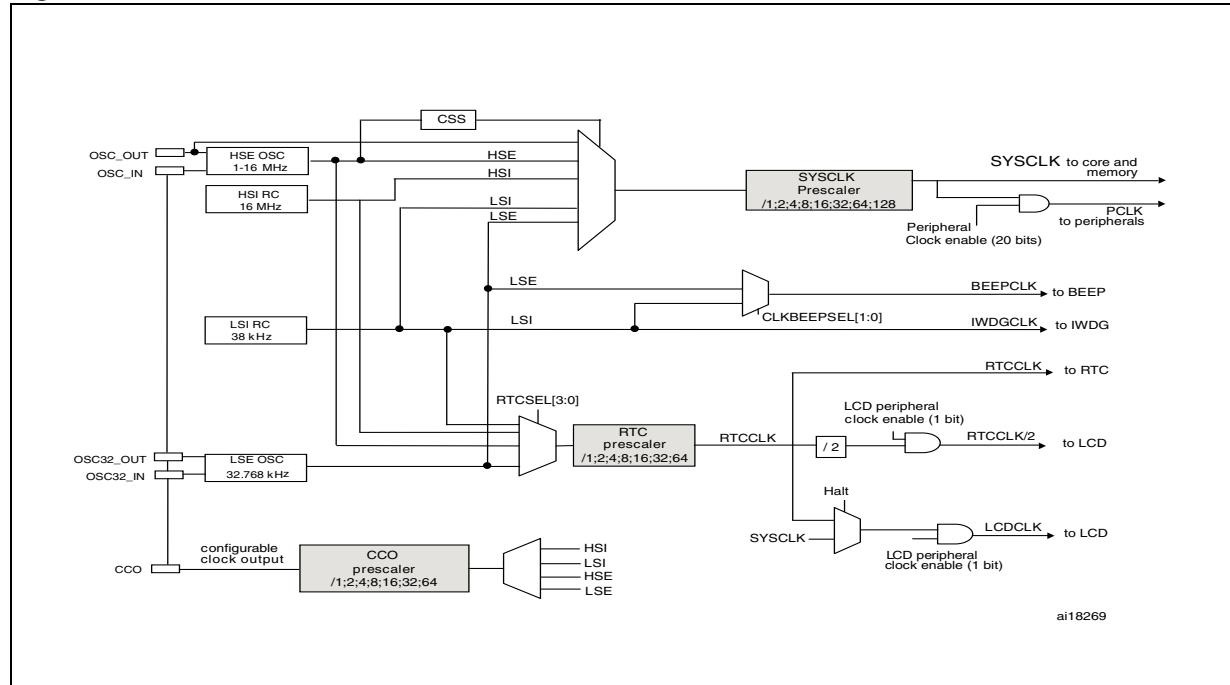
### 9.1 Introduction

The clock controller is designed to be very robust and at the same time easy to use. Its purpose is to obtain the best performance in the application while at the same time get the full benefit of all the microcontroller power saving capabilities.

You can manage the clock distribution to the CPU and to the various peripherals, in order to optimize the power consumption.

A safe and glitch-free switch mechanism allows you to switch the system clock division factor on the fly, by means of clock prescaler.

**Figure 17. Clock structure**



1. The HSE clock source can be either an external crystal/ceramic resonator or an external source (HSE bypass). Refer to [Section 9.2: HSE clock](#).
2. The LSE clock source can be either an external crystal/ceramic resonator or a external source (LSE bypass). Refer to [Section 9.4: LSE clock](#).
3. The Peripheral Clock Enable is 13 bits in low density devices and 14 bits in medium density devices and 20 bits in medium+ and high density devices.
4. The CSS\_LSE bit is available on low density, medium+ and high density devices only.

Four different clock sources can be used to drive the system clock:

- 16 MHz high-speed internal (HSI) factory-trimmed RC clock
- 1 to 16 MHz high-speed external (HSE) oscillator clock
- 32.768 kHz low-speed external (LSE) oscillator clock
- 38 kHz low-speed internal (LSI) low-consumption clock

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

These four clocks can be used with a programmable prescaler (factor 1 to 128) to drive the system clock (SYSCLK). The SYSCLK is used to clock the core, memory and peripherals. After reset, the device restarts by default with the HSI clock divided by 8. The prescaler ratio and clock source can be changed by the application program as soon as the code execution starts.

All the peripheral clocks are derived from the system clock (SYSCLK) except:

- The BEEP clock which is derived from the LSE or LSI clocks. This selection is made by programming the CLKBEEPSEL[1:0] bits in the Clock BEEP register (CLK\_CBEPR).
- The RTC and the LCD clock which is derived from the LSE, LSI HSI or HSE (HSI and HSE are divided by a programmable prescaler in CLK\_CRTCR register). For more information about the RTC and LCD clock source configuration please refer to [Section 9.9: RTC and LCD clock](#)
- The IWDG clock which is always the LSI clock.

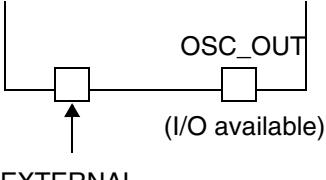
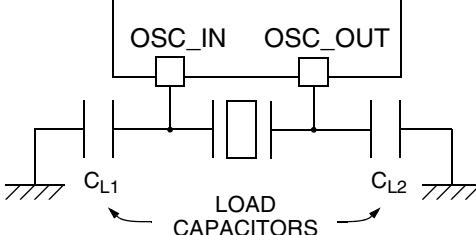
## 9.2 HSE clock

The high-speed external clock signal (HSE) can be generated from two possible clock sources:

- HSE external crystal/ceramic resonator
- HSE user external clock

*Note:* *The resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and start-up stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.*

**Figure 18. HSE clock sources**

Hardware configuration	
External clock	 <p>EXTERNAL SOURCE</p> <p>(I/O available)</p>
Crystal/ceramic resonators	 <p>LOAD CAPACITORS</p>

**External crystal/ceramic resonator (HSE crystal)**

The 1 to 16 MHz external oscillator has the advantage of producing a very accurate rate on the main clock with 50% duty cycle.

The associated hardware configuration is shown in [Figure 18](#). Refer to the electrical characteristics section in the product datasheet for more details.

At start up the clock signal produced by the oscillator is not stable, and by default a delay of 1 oscillator cycle is inserted before the clock signal is released. You can program another stabilization time in the HSECNT option byte, please refer to option bytes section in the datasheet.

The HSERDY flag in the External clock register (CLK\_ECKCR) indicates if the high-speed external oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware.

The HSE crystal can be switched on and off using the HSEON bit in the External clock register (CLK\_ECKCR).

**External source (HSE bypass)**

In this mode, an external clock source must be provided. It can have a frequency up to 16 MHz. The application can select this mode by setting the HSEBYP and HSEON bits in the External clock register CLK\_ECKCR. The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC\_IN pin while the OSC\_OUT pin is available as standard I/O. See [Figure 18](#).

**Note:** Before selecting this mode, you have to make sure that the HSE clock source is not used by the RTC, output or involved in a switching operation.

## 9.3 HSI clock

The HSI clock signal is generated from an internal 16 MHz RC.

At startup the system clock source is automatically selected as HSI RC clock output divided by 8 (HSI/8).

The HSI RC oscillator has the advantage of providing a 16-MHz system clock source with 50% duty cycle at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator however, even with calibration the frequency is less accurate than an external crystal oscillator or ceramic resonator.

### Calibration

Each device is factory-calibrated. After reset, the factory calibration value is automatically loaded in the [HSI calibration register \(CLK\\_HSICALR\)](#) and used for trimming the HSI oscillator. The read only content of the CLK\_HSICALR register can be used as a reference for a calibration by the user.

If the application is subject to voltage or temperature variations, this may affect the RC oscillator speed. You can trim the HSI frequency in the application using the [HSI clock calibration trimming register \(CLK\\_HSITRIMR\)](#). This register provides the internal HSI calibration register value.

Before writing to this register, you have to unlock the hardware write protection and then use the following procedure:

1. Two consecutive write accesses to the [HSI unlock register \(CLK\\_HSIUNLCKR\)](#), the first one with the value 0xAC and the second one with the value 0x35.
2. Write the HSI trimming value to the [HSI clock calibration trimming register \(CLK\\_HSITRIMR\)](#).

**Note:** Any other attempt to read/write data from/to CLK registers breaks the unlocking process and the hardware protection is set again.

When this procedure is properly completed, register CLK\_HSITRIMR is locked again and the written HSI trimming value is the calibration value used to trim the HSI oscillator.

### Backup source

The HSI clock can also be used as a backup source (auxiliary clock) if the HSE crystal oscillator fails. Refer to [Section 9.8: Clock security system \(CSS\) on page 96](#).

### Fast wakeup feature

If the FHWU bit in the [Internal clock register \(CLK\\_ICKCR\)](#) is set, this automatically selects the HSI clock as system clock after the MCU wakeup from Halt or Active-halt mode. Otherwise, the same clock source as that used before entering Halt or Active-halt mode is selected.

## 9.4 LSE clock

The LSE crystal is a 32.768 kHz low speed external crystal or ceramic resonator. It has the advantage of providing a low power and highly accurate clock source to the real-time clock peripheral (RTC) for clock/calendar or other timing functions.

The LSE crystal is switched on and off using the LSEON bit in the external clock register (CLK\_ECKCR). It is also automatically switched on when it is selected as RTC, CCO, Beep or system clock source. At startup, the LSE crystal output clock signal is transmitted to the MCU after the LSERDY flag has been set by hardware in the external clock register (CLK\_ECKCR).

*Note:* *It is forbidden to enable CCO before the end of the LSE startup time  $t_{SU(LSE)}$ .*

*It is also recommended to wait for this LSE startup time before enabling other features using the LSE, in order to have a stabilized frequency.*

### External source (LSE bypass)

In this mode, an external clock source must be provided. It must have a frequency of 32.768 kHz. The application can select this mode by setting the LSEBYP and LSEON bits in the External clock register (CLK\_ECKCR). The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC32\_IN pin while the OSC32\_OUT pin is available as standard I/O. See [Figure 18](#).

*Note:* *Before selecting this mode, you have to make sure that the LSE clock source is not used by the RTC output or involved in a switching operation.*

## 9.5 LSI clock

The LSI RC acts as a low power clock source that can be kept running in Active-halt and Run modes for the independent watchdog (IWDG). The clock frequency is around 38 kHz. For more details, refer to the electrical characteristics section in the product datasheet.

The LSI RC can be switched on and off using the LSION bit in the internal clock register (CLK\_IKCCR).

To detect a sequence of entry into SWIM mode, the LSI is automatically kept enabled in all modes except Halt and Active-halt modes. Consequently, the LSI can be switched off by clearing the LSION bit only if no peripheral is clocked by the LSI and after SWIM has been disabled by setting the SWD bit in the CFG\_GCR register. The LSI is not kept ON by the SWIM in Halt and Active-halt mode.

The LSIRDY flag in the Internal clock register (CLK\_IKCCR) indicates if the low-speed internal oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware.

## 9.6 System clock sources

### 9.6.1 System startup

For fast system startup, after a reset the clock controller configures the system clock source as HSI RC clock output divided by 8 (HSI/8). This is to take advantage of the short stabilization time of the HSI oscillator.

Once the system clock is released, the user program can switch the system clock to another clock source (LSI, HSE, or LSE) and/or modify the prescaler ratio.

## 9.6.2 System clock switching procedures

The clock switching feature provides an easy to use, fast and secure way for the application to switch from one system clock source to another.

To switch clock sources, you can proceed in one of two ways:

- Automatic switching
- Manual switching

### Automatic switching

The automatic switching enables the user to launch a clock switch with a minimum number of instructions. The software can continue doing other operations without taking care of the switch event exact time.

Refer to the flowchart in [Figure 19](#).

Use the following procedure:

1. Enable the switching mechanism by setting the SWEN bit in the [\*Switch control register \(CLK\\_SWCR\)\*](#).
2. Write the 8-bit value used to select the target clock source in the [\*System clock switch register \(CLK\\_SWR\)\*](#). The SWBSY bit in the CLK\_SWCR register is set by hardware, and the target source oscillator starts. The old clock source continues to drive the CPU and peripherals.

As soon as the target clock source is ready (stabilized), the content of the CLK\_SWR register is copied to the [\*System clock status register \(CLK\\_SCSR\)\*](#).

The SWBSY bit is cleared and the new clock source replaces the old one. If the SWIEN bit is set, the SWIF flag in the CLK\_SWCR is set and an interrupt is generated.

### Manual switching

The manual switching is not as immediate as the automatic switching but it offers to the user a precise control of the switch event time.

Refer to the flowchart in [Figure 20](#).

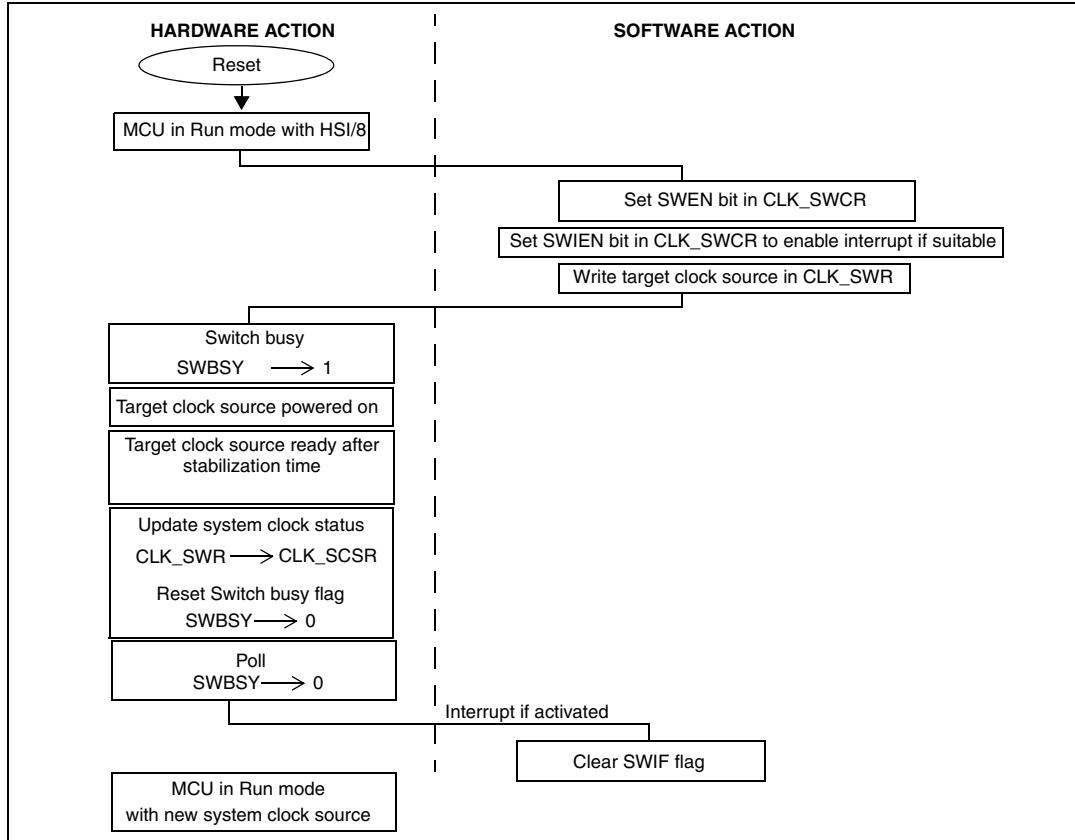
1. Write the 8-bit value used to select the target clock source in the [\*System clock switch register \(CLK\\_SWR\)\*](#). Then the SWBSY bit is set by hardware, and the target source oscillator starts. The old clock source continues to drive the CPU and peripherals.
2. The software has to wait until the target clock source is ready (stabilized). This is indicated by the SWIF flag in the CLK\_SWCR register and by an interrupt if the SWIEN bit is set.
3. The final software action is to set, at the chosen time, the SWEN bit in the CLK\_SWCR register to execute the switch.

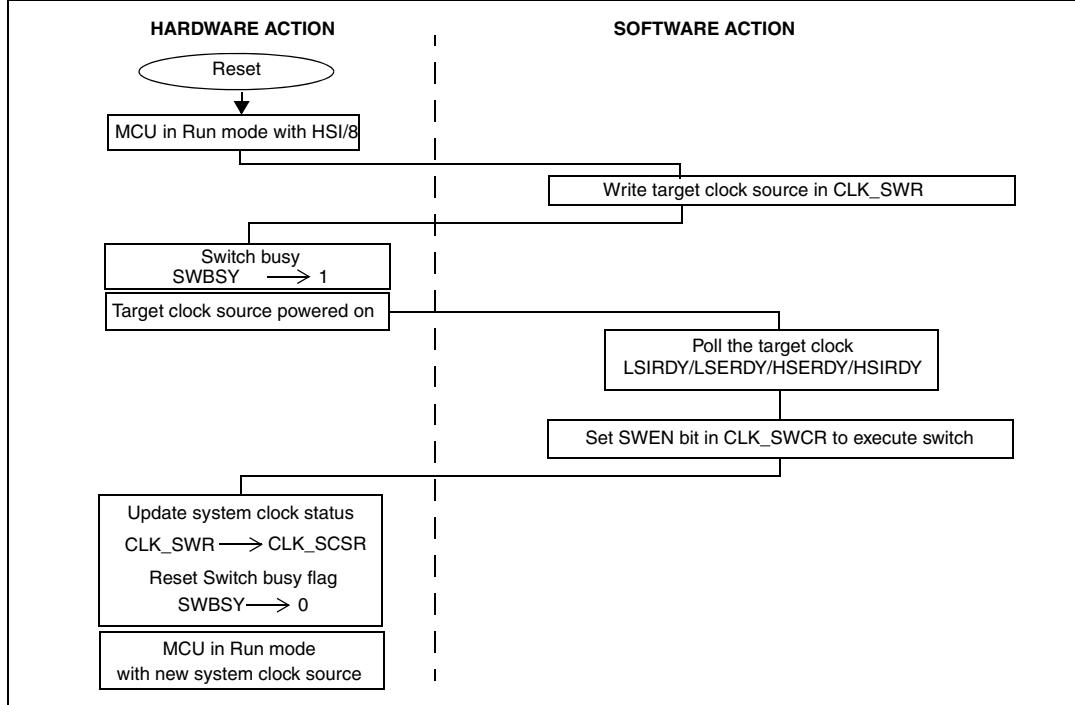
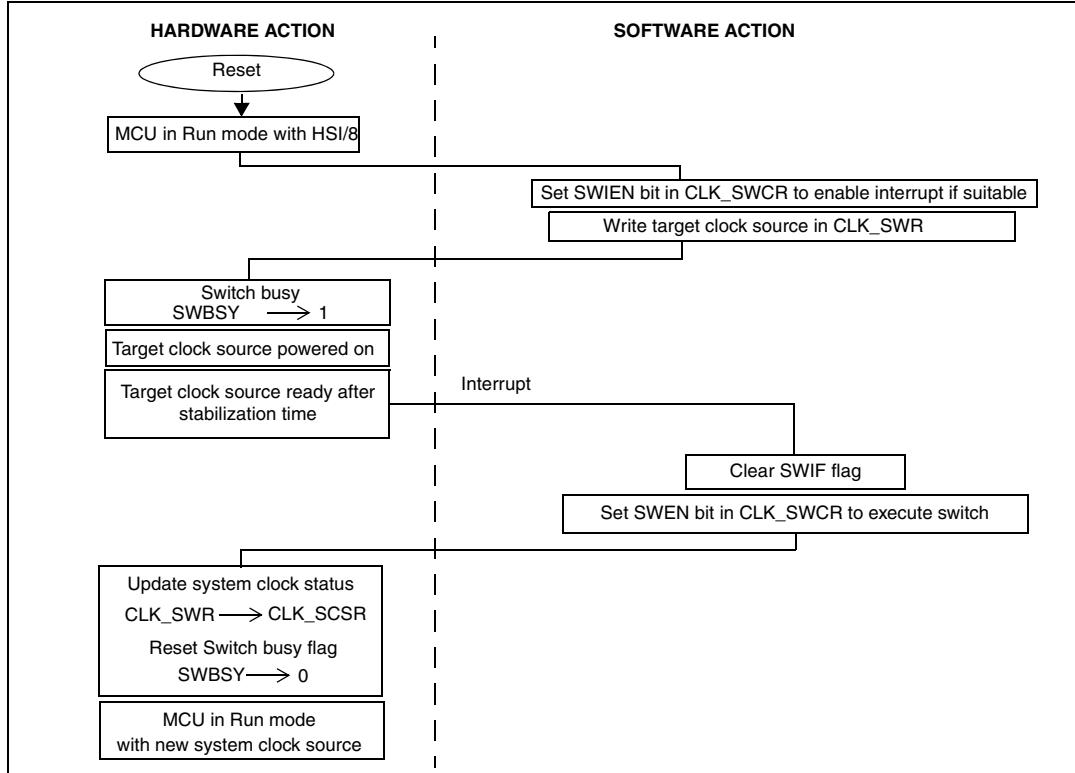
In both manual and automatic switching modes, the old system clock source will not be powered off automatically in case it is required by other blocks (the LSI RC may be used to drive the independent watchdog for example). The clock source can be powered off using the bits in the Internal clock register (CLK\_ICKCR) and the external clock register (CLK\_ECKCR).

If the clock switch does not work for any reason, software can reset the current switch operation by clearing the SWBSY flag. This will restore the CLK\_SWR register to its previous content (old system clock).

**Note:** After having reset a clock master switch procedure by clearing the SWBSY flag, the application must wait until a period of at least two clock cycles has elapsed before generating any new clock master switch request.

**Figure 19. Clock switching flowchart (automatic mode example)**



**Figure 20. Clock switching flowchart (manual mode example without interrupt)****Figure 21. Clock switching flowchart (manual mode example with interrupt)**

## 9.7 Peripheral clock gating (PCG)

Peripheral clock Gating (PCG) mode selectively enables or disables the system clock (SYSCLK) connection to peripherals at any time in run or slow mode to optimize power consumption.

After a device reset, all peripheral clocks are disabled. The only bit which is enabled by default at reset state is PCKEN27 as it is used for the BootROM. Software has to be properly written to switch off the ROM clock after the Bootloader execution.

You can enable the clock to any peripheral by setting the corresponding PCKEN bit in the CLK\_PCKENRx peripheral clock gating registers.

- To enable a peripheral, first enable the corresponding PCKEN bit in the CLK\_PCKENR registers and then set the peripheral enable bit in the peripheral control registers.
- To disable properly the peripheral, first disable the appropriate bit in the peripheral control registers and then stop the corresponding clock.

*Note:* *The beeper, RTC and LCD are fed by a specific clock different from SYSCLK, so that they continue to run even if the clock gating to the peripheral registers is asserted.*

## 9.8 Clock security system (CSS)

### 9.8.1 Clock security system on HSE

The clock security system (CSS) monitors HSE crystal clock source failures when HSE is used as the system clock. If the HSE clock fails due to a broken or disconnected resonator or any other reason, the clock controller activates a stall-safe recovery mechanism by automatically switching SYSCLK to the HSI with the same division factor as that used before the HSE clock failure. Once selected, the auxiliary clock source remains enabled until the microcontroller is reset. The application can enable the clock security system by setting the CSSEN bit in the [Clock security system register \(CLK\\_CSSR\)](#). For safety reason, once CSS is enabled it cannot be disabled until the next device reset.

The following conditions must be met to detect HSE quartz crystal failures through the CSS:

- HSE on and stabilized (HSEON=1 and HSERDY=1 in the [External clock register \(CLK\\_ECKCR\)](#))
- CSS function enabled (CSSEN=1 in the [Clock security system register \(CLK\\_CSSR\)](#))

If HSE is the current system clock when a failure is detected, the CSS performs the following actions:

- The HSI is switched on (if it was off) (HSION bit in [Internal clock register \(CLK\\_IKCR\)](#) register is set).
- The HSE is switched off (HSEON bit in the CLK\_ECKCR register is reset)
- The AUX bit is set to indicate that the HSI is used as system clock source. The division factor of the system clock is not modified.
- All CLK registers are write protected except the [System clock divider register \(CLK\\_CKDIVR\)](#). The application can still change the system clock prescaler.
- The CSSD bit is set in the CLK\_CSSR register and an interrupt is generated if the CSSDIE bit is set.

*Note:* *CSSD and AUX flags are cleared only by device reset.*

If HSE is not the current system clock source when a failure is detected, the CSS performs the following actions:

- The HSE is switched off (HSEON bit in the CLK\_ECKCR is reset)
- The CSSD bit is set in the CLK\_CSSR register and an interrupt is generated if the CSSDIE bit is set.

When the CSSDIE bit is set, if the HSE clock fails, the CSS interrupt is triggered and is executed indefinitely until the CSS interrupt enable bit is cleared. As a consequence, the application must clear the CSSDIE bit in CLK\_CSSR register in the CLK interrupt service routine (ISR).

When the HSE is selected as system clock source, if the HSE fails before reaching the required stabilization time, it is not detected (CSSD equal to zero) and the system clock remains fed by the previous clock. In this case, the application must abort the clock switching procedure by clearing the SWBSY bit in the *Switch control register (CLK\_SWCR)*.

## 9.8.2 Clock security system on LSE

A second clock security system is implemented on low, medium+ and high density devices to monitor LSE crystal clock source failures when LSE is used as RTC clock. The LSE failure detection is made through the LSI oscillator.

This feature is implemented externally to the RTC peripheral and has no impact on clock controller registers. To work properly, it requires an LSE clock frequency greater than LSI/4.

Depending on the configuration of the SWITCHEN bit in the CSSLSE\_CSR register, the RTC clock can be automatically switched from LSE to LSI when a LSE failure is detected.

The application can enable the clock security system on LSE and switch to LSI by setting respectively the CSSEN bit and the SWITCHEN bit in the CSSLSE\_CSR register.

For safety reason, once the CSS on LSE is enabled, it cannot be disabled until the next power-on reset.

The following conditions must be met to detect LSE quartz crystal failures through the CSS on LSE when LSE is used as RTC clock source:

- Enable the peripheral clock by setting the CSS\_LSE bit in the CLK\_PCKENR3 register
- Enable the LSI clock by setting the LSION bit in the CLK\_ICKCR register
- Configure the LSE as RTC clock source through the CLK\_CRTCR register.

*Note:*

*If the CSS on LSE is enabled without configuring the LSE as RTC clock source, a false failure will be detected, due to the fact that the RTC clock is missing.*

- Enable the CSS on LSE by setting the CSSEN bit in the CSSLSE\_CSR register

*Note:*

*Before enabling the CSS on LSE, the user must wait until the LSE startup time  $t_{SU(LSE)}$  has elapsed*

- If needed, enable the automatic clock switch from LSE to LSI on LSE failures by setting the SWITCHEN bit in the CSSLSE\_CSR register.
- When an LSE failure is detected:
  - The CSSF bit is set in the CSSLSE\_CSR register and an interrupt is generated if the CSSIE bit is set.
  - If the SWITCHEN bit is set : the RTC clock source is switched to LSI and the SWITCHF bit is set in the CSSLSE\_CSR register. The RTC clock source remains

the LSI until the next power-on reset. The CLK\_CRTCR register is not updated by the switch and cannot control the RTC clock until the next power-on.

- Note:** Only the RTC peripheral is clocked by LSI instead of LSE when the SWITCHF bit is set.
- If the SWITCHEN bit is reset: the RTC clock source remains the LSE clock.

### CSS on LSE low power modes

The following table describes the behavior of the CSS on LSE in low power modes.

**Table 17. CSS on LSE in low power modes**

Mode	Description
Halt/ Active-halt	CSSF interrupts cause the device to exit from Active-halt or Halt mode

## 9.8.3 CSS on LSE control and status register (CSSLSE\_CSR)

Address offset: 0x00

Power-on reset value: 0x00

Reset value: 0xXX (the registers are not impacted by a system reset. They are reset at power-on).

7	6	5	4	3	2	1	0
Reserved		SWITCHF	CSSF	CSSIE	SWITCHEN	CSSEN	
r		r	rw	rw	rwo	rwo	

Bits 7:5 Reserved, must be kept cleared.

#### Bit 4 **SWITCHF:** RTC clock switch flag

This bit is set by hardware and can be cleared only by power-on reset.

0: No RTC clock switch occurred or no failure on LSE detected

1: RTC clock switch to LSI completed

#### Bit 3 **CSSF:** CSS on LSE flag

This bit is set by hardware and cleared by software by writing 0.

0: CSS on LSE is OFF or no LSE crystal clock disturbance is detected.

1: An LSE crystal clock disturbance is detected.

#### Bit 2 **CSSIE:** Clock security system on LSE interrupt enable

This bit is set and cleared by software.

0: Clock security system on LSE interrupt disabled

1: Clock security system on LSE interrupt enabled

#### Bit 1 **SWITCHEN:** RTC clock switch to LSI in case of LSE failure enable

This bit is set only by software and can be cleared only by power-on reset.

0: Clock switch to LSI in case of LSE failure OFF

1: Clock switch to LSI in case of LSE failure ON

#### Bit 0 **CSSEN:** Clock security system on LSE enable

This bit is set only by software and can be cleared only by power-on reset.

0: Clock security system on LSE OFF

1: Clock security system on LSE ON

### 9.8.4 CSS on LSE register map and reset values

**Table 18. CSS on LSE register map**

Address offset <sup>(1)</sup>	Register name	7	6	5	4	3	2	1	0
0x00	CSSLSE_CSR Power-on reset value	-0	-0	-0	SWITCHF0	CSSF0	CSSIE0	SWITCHEN0	CSSEN0

1. Please refer to the “general hardware register map” table in the datasheet for details on register addresses.

## 9.9 RTC and LCD clock

The RTC has two clock sources:

- RTCCLK used for RTC timer/counter  
RTCCLK can be either the HSE, LSE, HSI or LSI clock. This selection is performed by programming the RTCSEL[3:0] bits in the Clock RTC register (CLK\_CRTCR). These clocks can be optionally divided by a programmable divider (factor 1 to 64) by programming the RTCDIV[2:0] bits in the Clock RTC register (CLK\_CRTCR). When the HSE or HSI clock is selected as RTCCLK source, this clock must be divided to have a maximum of 1 MHz as input for the RTCCLK.
- SYSCLK used for RTC register read/write accesses  
SYSCLK is gated by bit 2 of the Peripheral clock gating register 2 (CLK\_PCKENR2).

The LCD has two clock sources:

- RTCCLK divided by 2 used to generate the LCD frame rate  
This clock is gated by bit 3 in the Peripheral clock gating register 2 (CLK\_PCKENR2). As a consequence, even if the RTC is not used in the application, the RTCCLK must be configured to drive the LCD.
- LCDCLK used for LCD register read/write access  
This clock is derived from SYSCLK by setting the bit 3 in the Peripheral clock gating register 2 (CLK\_PCKENR2). In Active-halt mode the LCDCLK source is RTCCLK instead of SYSCLK.

## 9.10 BEEP clock

The BEEP clock sources can be either the LSE or LSI clocks. They can be selected by programming the CLKBEEPSEL[1:0] bits in the *Clock BEEP register (CLK\_CBEEPR)*.

## 9.11 Configurable clock output capability (CCO)

The configurable clock output (CCO) capability allows you to output a clock on the external CCO pin. You can select one of 4 clock signals as CCO clock:

- $f_{HSE}$
- $f_{HSI}$
- $f_{LSE}$
- $f_{LSI}$

**Note:** 50% duty cycle is not guaranteed on all possible prescaled values

The selection is controlled by the CCOSEL[3:0] bits in the [Configurable clock output register \(CLK\\_CCOR\)](#). The clock frequency can then be prescaled. The division factor depends on the CCODIV[2:0] bits.

The sequence to really output the chosen clock starts with the selection of the desired clock and the division factor for the dedicated I/O pin (see “pin description” chapter). This I/O must be set at 1 in the corresponding Px\_CR1 register to be set as input with pull-up or push-pull output. The PxCR2 register should be configured to match the maximum output speed of the desired clock.

The CCOSWBSY is set to indicate that the configurable clock output system is operating. As long as the CCOBWSY bit is set, the CCOSEL and CCODIV bits are write protected.

The CCO automatically activates the target oscillator if needed.

To disable the clock output the user has to clear the CCOSEL bits. The CCOSWBSY flag remains at 1 till the shut down is completed. The time between the clear of CCOSEL bits and the reset of the flag can be relatively long, for instance in case the selected clock output is very slow compared to  $f_{CPU}$ .

## 9.12 Clock-independent system clock sources for TIM2/TIM3

In some applications using the 32.768 kHz clock as a time base for the RTC, it may be interesting to have time bases that work completely independently from the system clock. The user can then schedule tasks without having to take into account the system clock.

For this purpose, the LSE clock is internally redirected to the 2 timers' ETR inputs (TIM2 and TIM3) which are used as additional clock sources. Refer to [Section 11.5.2: SYSCFG remap control register 2 \(SYSCFG\\_RMPCR2\) on page 151](#).

**Note:** The ETR prescaler of the corresponding timer must be set to have a ratio of at least 4 with respect to the selected system clock.

The LSE clock must be properly switched on as defined in the CLK configuration registers.

## 9.13 CLK interrupts

The following interrupts can be generated by the clock controller:

- System clock source switch event
- Clock security system event

Both interrupts are individually maskable.

**Table 19. CLK interrupt requests**

Interrupt event	Event flag	Enable control bit	Exit from Wait/Low power wait	Exit from Halt/Active-halt
CSS event	CSSD	CSSDIE	Yes	No
System clock switch event	SWIF	SWIEN	Yes	No

## 9.14 CLK registers

### 9.14.1 System clock divider register (CLK\_CKDIVR)

Address offset: 0x00

Reset value: 0x03

7	6	5	4	3	2	1	0
Reserved					CKM[2:0]		
-	-	-	-	-	rw	rw	rw

Bits 7:3 Reserved, must be kept cleared.

Bits 2:0 **CKM[2:0]**: System clock prescaler

- 000: System clock source/1
- 001: System clock source /2
- 010: System clock source /4
- 011: System clock source /8
- 100: System clock source /16
- 101: System clock source /32
- 110: System clock source /64
- 111: System clock source /128

These bits are written by software to define the system clock prescaling factor.

### 9.14.2 Clock RTC register (CLK\_CRTCR)

Address offset: 0x01

Power-on reset value: 0x00

Reset value: not affected (the content of this register is not affected by system resets)

7	6	5	4	3	2	1	0
RTCDIV2	RTCDIV1	RTCDIV0	RTCSEL3	RTCSEL2	RTCSEL1	RTCSEL0	RTCSWBSY
rw	r						

Bits 7:5 **RTCDIV[2:0]**: Clock RTC prescaler

These bits are written by software to select the clock RTC division factor.

- 000: RTC clock source/1
- 001: RTC clock source /2
- 010: RTC clock source /4
- 011: RTC clock source /8
- 100: RTC clock source /16
- 101: RTC clock source /32
- 110: RTC clock source /64
- 111: RTC clock source /128

*Note:* Any write with a non valid target code will be skipped.

The content of these bits is frozen and write protected when RTCSWBSY is set.

**Bits 4:1 RTCSEL[3:0]: Configurable RTC clock source selection**

These bits are written by software to select the clock source to be used by the RTC.

0000: No clock selected

0001: HSI clock used as RTC clock source

0010: LSI clock used as RTC clock source

0100: HSE clock used as RTC clock source

1000: LSE clock used as RTC clock source

*Note: Any write with a non valid target code will be skipped.*

*The content of these bits is frozen and write protected when RTCSWBSY is set.*

**Bit 0 RTCSWBSY: The system is busy during a RTC clock change**

This bit is set by hardware any time a valid clock change for RTC is required.

It is reset when the RTC clock change procedure is complete.

### 9.14.3 Internal clock register (CLK\_ICKCR)

Address offset: 0x02

Reset value: 0x11

	7	6	5	4	3	2	1	0
Reserved	BEEPAHALT	FHWU	SAHALT	LSIRDY	LSION	HSIRDY	HSION	
	rw	rw	rw	r	rw	r	rw	

Bit 7 Reserved, must be kept cleared.

Bit 6 **BEEPAHALT**: BEEP clock Halt/Active-halt mode

This bit is set and cleared by software.

0: BEEP clock is switched off during Halt mode or Active-halt mode

1: BEEP clock is kept running during Halt mode

Bit 5 **FHWU**: Fast wakeup from Halt/Active-halt modes

This bit is set and cleared by software.

0: Fast wakeup from Halt/Active-halt modes disabled: system clock is the last selected clock source before entering Halt/Active-halt mode

1: Fast wakeup from Halt/Active-halt modes enabled: HSI/8 used as system clock source after wakeup from Halt/Active-halt

Bit 4 **SAHALT**: Slow Active-halt mode

This bit is set and cleared by software. When it is set, the main voltage regulator is powered off as soon as the MCU enters Active-halt mode, so the wakeup time is longer.

0: MVR regulator ON in Active-halt mode

1: MVR regulator OFF in Active-halt mode

Bit 3 **LSIRDY**: Low speed internal oscillator ready

This bit is set and cleared by hardware.

0: LSI clock not ready

1: LSI clock ready

Bit 2 **LSION**: Low speed internal RC oscillator enable

This bit is set and cleared by software. It is set by hardware whenever the LSI oscillator is required, for example:

- When switching to the LSI clock (see CLK\_SWR register)
- When LSI is selected as the active CCO source (see CLK\_CCOR register)
- When LSI is selected as the active RTC clock source (see CLK\_CRTCR register)
- When LSI is selected as the active BEEP clock source (see CLK\_CBEEPR register)
- When LSI measurement is enabled (MSR bit set in the BEEP\_CSR1 register)

It cannot be cleared when LSI is selected

● as system clock source (CLK\_SCSR register)

● as active CCO source

● as clock source for the BEEP peripheral while BEEPAHALT bit is set

● or as active clock source for RTC.

0: Low speed internal RC OFF

1: Low speed internal RC ON

Bit 1 **HSIRDY**: High-speed internal oscillator ready

This bit is set and cleared by hardware.

0: HSI clock not ready

1: HSI clock ready

Bit 0 **HSION**: High-speed internal RC oscillator ON

This bit is set and cleared by software. It is set by hardware whenever the HSI oscillator is required, for example:

- When activated as safe oscillator by the CSS
- When switching to HSI clock (see CLK\_SWR register)
- When HSI is selected as the active CCO source, RTC clock
- When exiting Halt/Active-halt in fast wake up mode

It cannot be cleared when HSI is selected as system clock (CLK\_SCSR register), as active CCO source, as active RTC clock or if the safe oscillator (AUX) is enabled.

0: High-speed internal RC OFF

1: High-speed internal RC ON

#### 9.14.4 Peripheral clock gating register 1 (CLK\_PCKENR1)

Address offset: 0x3

Reset value: 0x00

7	6	5	4	3	2	1	0
PCKEN1[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **PCKEN1[7:0]**: Peripheral clock enable

These bits are written by software to enable or disable the SYSCLK clock to the corresponding peripheral. See [Table 20](#)

- 0: SYSCLK to peripheral disabled
- 1: SYSCLK to peripheral enabled

**Table 20. Peripheral clock gating bits (PCKEN 10 to PCKEN 17)**

Control bit	Peripheral
PCKEN17	DAC
PCKEN16	BEEP
PCKEN15	USART1
PCKEN14	SPI1
PCKEN13	I2C1
PCKEN12	TIM4
PCKEN11	TIM3
PCKEN10	TIM2

### 9.14.5 Peripheral clock gating register 2 (CLK\_PCKENR2)

Address offset: 0x04

Reset value: 0x80

7	6	5	4	3	2	1	0
PCKEN27	Reserved	PCKEN2[5:0]					
		rw	rw	rw	rw	rw	rw

Bit 7 **PCKEN27**: Peripheral clock enable

These bits are written by software to enable or disable the SYSCLK clock to the corresponding peripheral. See [Table 21](#)

- 0: SYSCLK to peripheral disabled
- 1: SYSCLK to peripheral enabled

Bit 6 Reserved

Bits 5:0 **PCKEN2[5:0]**: Peripheral clock enable

These bits are written by software to enable or disable the SYSCLK clock to the corresponding peripheral. See [Table 21](#)

- 0: SYSCLK to peripheral disabled
- 1: SYSCLK to peripheral enabled

**Table 21. Peripheral clock gating bits ((PCKEN 20 to PCKEN 27))**

Control bit	Peripheral
PCKEN27 <sup>(1)</sup>	Boot ROM
PCKEN25	COMP1 and COMP2
PCKEN24	DMA1
PCKEN23	LCD
PCKEN22	RTC
PCKEN21	TIM1
PCKEN20	ADC1

1. The only bit which is enabled by default at reset state is PCKEN27 as it is used for the Boot ROM. Software has to be properly written to switch off the ROM clock after the Bootloader execution.

### 9.14.6 Peripheral clock gating register 3 (CLK\_PCKENR3)

Address offset: 0x10

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved	PCKEN3[5:0]							
	rw	rw	rw	rw	rw	rw	rw	rw

**Note:** This peripheral is available in low density, medium+ and high density devices only. In low density devices, only bit 5 is available.

Bits 7:0 **PCKEN3[5:0]:** Peripheral clock enable

These bits are written by software to enable or disable the SYSCLK clock to the corresponding peripheral. See [Table 22](#)

- 0: SYSCLK to peripheral disabled
- 1: SYSCLK to peripheral enabled

**Table 22. Peripheral clock gating bits (PCKEN 30 to PCKEN 35)**

Control bit	Peripheral
PCKEN35	CSS_LSE
PCKEN34	USART3
PCKEN33	USART2
PCKEN32	SPI2
PCKEN31	TIM5
PCKEN30	AES

### 9.14.7 Configurable clock output register (CLK\_CCOR)

Address offset: 0x05

Reset value: 0x00

7	6	5	4	3	2	1	0
CCODIV[2:0]			CCOSEL[3:0]				CCOSWBSY
rw	rw	rw	rw	rw	rw	rw	r

Bits 7:5 **CCODIV[2:0]**: Configurable clock output prescaler

These bits are written by software to select the clock CCO division factor.

- 000: CCO
- 001: CCO divided by 2
- 010: CCO divided by 4
- 011: CCO divided by 8
- 100: CCO divided by 16
- 101: CCO divided by 32
- 110: CCO divided by 64
- 111: CCO divided by 64

*Note: Any write with a non valid target code will be skipped.*

*The content of these bits is frozen and write protected when CCOSWBSY is set.*

Bits 4:1 **CCOSEL[3:0]**: Configurable clock output selection.

These bits are written by software to select the source of the output clock available on the CCO pin.

- 0000: Clock output disabled, no clock output on CCO pin
- 0001: HSI clock output on CCO pin
- 0010: LSI clock output on CCO pin
- 0100: HSE clock output on CCO pin
- 1000: LSE clock output on CCO pin

*Note: Any write with a non valid target code will be skipped.*

*The content of these bits is frozen and write protected when CCOSWBSY is set.*

Bit 0 **CCOSWBSY**: Configurable clock output switch busy

This bit is set and cleared by hardware. It indicates that the selected CCO clock source is being switched-on and stabilized. While CCOSWBSY is set, the CCOSEL bits and CCODIV bits are write-protected. CCOSWBSY remains set until the CCO clock is enabled.

- 0: CCO clock not busy
- 1: CCO clock busy

### 9.14.8 External clock register (CLK\_ECKCR)

Address offset: 0x06

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved	LSEBYP	HSEBYP	LSERDY	LSEON	HSERDY	HSEON		
	rw	rw	r	rw	r	rw		

Bits 7:6 Reserved, must be kept cleared.

Bit 5 **LSEBYP**: Low speed external clock bypass

Set and reset by software to bypass the oscillator. This bit can be written only when the external low speed oscillator is disabled.

- 0: LSE oscillator not bypassed
- 1: LSE oscillator bypassed

Bit 4 **HSEBYP**: High-speed external clock bypass

Set and reset by software in debug for bypassing the oscillator with the external clock. This bit can be written only if the external 1-16 MHz oscillator is disabled.

- 0: external 1-16 MHz oscillator not bypassed
- 1: external 1-16 MHz oscillator bypassed with external clock

Bit 3 **LSERDY**: Low speed external crystal oscillator ready

This bit is set and cleared by hardware.

- 0: LSE clock not ready
- 1: LSE clock ready (LSE clock is stabilized and available)

Note: the user must wait for the LSE startup time  $t_{SU(LSE)}$  to get a stabilized frequency.

Bit 2 **LSEON**: Low speed external crystal oscillator enable

This bit is set and cleared by software. It can be used to switch the external crystal oscillator on or off. It is set by hardware in the following cases:

- When switching to LSE clock (see CLK\_SWR register)
- When LSE is selected as the active CCO source (see CLK\_CCOR register)
- When LSE is selected as the active RTC source (see CLK\_CRTCR register)
- When LSE is selected as the active BEEP source (see CLK\_CBEPR register)

It cannot be cleared when LSE is selected

- as system clock source (CLK\_SCSR register),
- as active CCO source
- as clock source for the BEEP peripheral and BEEPAHALT bit is set
- or as active clock source for RTC.

- 0: LSE clock off
- 1: LSE clock on

Bit 1 **HSERDY**: High-speed external crystal oscillator ready

This bit is set and cleared by hardware.

- 0: HSE clock not ready
- 1: HSE clock ready (HSE clock is stabilized and available)

Bit 0 **HSEON**: High-speed external crystal oscillator enable

This bit is set and cleared by software. It can be used to switch the external crystal oscillator ON or OFF. It is set by hardware in the following cases:

- When switching to HSE clock (see CLK\_SWR register)
- When HSE is selected as the active CCO source (see CLK\_CCOR register)
- When HSE is selected as the active RTC source (see CLK\_CRTCR register)

It cannot be cleared when HSE is selected as system clock (indicated in CLK\_SCSR register) or as the active CCO source or as active RTC clock source.

0: HSE clock OFF

1: HSE clock ON

### 9.14.9 System clock status register (CLK\_SCSR)

Address offset: 0x07

Reset value: 0x01

7	6	5	4	3	2	1	0
CKM[7:0]							
r	r	r	r	r	r	r	r

Bits 7:0 **CKM[7:0]**: System clock status bits

These bits are set and cleared by hardware. They indicate the currently selected system clock source.

0x01: HSI selected as system clock source (reset value)

0x02: LSI selected as system clock source

0x04: HSE selected as system clock source

0x08: LSE selected as system clock source

### 9.14.10 System clock switch register (CLK\_SWR)

Address offset: 0x08

Reset value: 0x01

7	6	5	4	3	2	1	0
SWI[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **SWI[7:0]:** System clock selection bits

These bits are written by software to select the system clock source. Their contents are write protected while a clock switch is ongoing (while the SWBSY bit is set). They are set to the reset value (HSI) if the AUX bit is set in the CLK\_CSSR register. If Fast halt wakeup mode is selected (FHW bit =1 in CLK\_ICKCR register) then these bits are set by hardware to 0x01 (HSI selected) when resuming from Halt/Active-halt mode.

- 0x01: HSI selected as system clock source (reset value)
- 0x02: LSI selected as system clock source
- 0x04: HSE selected as system clock source
- 0x08: LSE selected as system clock source

### 9.14.11 Switch control register (CLK\_SWCR)

Address offset: 0x09

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved				SWIF	SWIEN	SWEN	SWBSY
				rc_w0	rw	rw	rw

Bits 7:4 Reserved, must be kept cleared.

Bit 3 **SWIF:** Clock switch interrupt flag

This bit is set by hardware when the SWIEN bit is set and cleared by software writing 0. Its meaning depends on the status of the SWEN bit. Refer to [Figure 19](#) and [Figure 20](#).

- **In manual switching mode (SWEN=0):**
  - 0: Target clock source not ready
  - 1: Target clock source ready
- **In automatic switching mode (SWEN=1):**
  - 0: No clock switch event occurred
  - 1: Clock switch event occurred

Bit 2 **SWIEN:** Clock switch interrupt enable

This bit is set and cleared by software.

- 0: Clock switch interrupt disabled
- 1: Clock switch interrupt enabled

Bit 1 **SWEN:** Switch start/stop

This bit is set and cleared by software. Writing a 1 to this bit enables switching the system clock to the source defined in the CLK\_SWR register.

- 0: Disable clock switch execution
- 1: Enable clock switch execution

**Bit 0 SWBSY:** Switch busy

This bit is set and cleared by hardware. It can be cleared by software to reset the clock switch process.

0: No clock switch ongoing

1: Clock switch ongoing

### 9.14.12 Clock security system register (CLK\_CSSR)

Address offset: 0x0A

Reset value: 0x00

7	6	5	4	3	2	1	0
			CSSDGON	CSSD	CSSDIE	AUX	CSSEN
Reserved			rw	rc_w0	rw	r	rwo

Bits 7:5 Reserved, must be kept cleared.

**Bit 4 CSSDGON:** CSS deglitcher system

This bit, when set, avoids any clock glitch generated during the HSE switch-off executed by the CSS mechanism.

**Bit 3 CSSD:** Clock security system detection

This bit is set by hardware and only cleared by device reset.

0: CSS is OFF or no HSE crystal clock disturbance detected.

1: HSE crystal clock disturbance detected.

**Bit 2 CSSDIE:** Clock security system detection interrupt enable

This bit is set and cleared by software.

0: Clock security system interrupt disabled

1: Clock security system interrupt enabled

**Bit 1 AUX:** Auxiliary oscillator connected to system clock

This bit is set and cleared by hardware.

0: Auxiliary oscillator is OFF.

1: Auxiliary oscillator (HSI) is ON and selected as current system clock source.

**Bit 0 CSSEN:** Clock security system enable

This bit can be written once-only by software. It cannot be reset before the next device reset.

0: Clock security system OFF

1: Clock security system ON

### 9.14.13 Clock BEEP register (CLK\_CBEPR)

Address offset: 0x0B

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved					CLKBEEPSEL1	CLKBEEPSEL0	BEEPSWBSY
					rw	rw	r

Bits 7:3 Reserved

Bits 2:1 **CLKBEEPSEL[1:0]**: Configurable BEEP clock source selection.

These bits are set by software to select the clock source to be used by the BEEP.

00: No clock selected

01: LSI clock used as BEEP clock source

10: LSE clock used as BEEP clock source

*Note: Any write with a non valid target code will be skipped.*

*The content of these bits is frozen and write protected when BEEPSWBSY is set.*

Bit 0 **BEEPSWBSY**: System busy during BEEP clock change

This bit is set by hardware any time a valid clock change for BEEP is required.

It is reset when the BEEP clock change procedure is complete.

### 9.14.14 HSI calibration register (CLK\_HSICALR)

Address offset: 0x0C

Reset value: 0XX where X is undefined

7	6	5	4	3	2	1	0
HSICAL[7:0]							
r	r	r	r	r	r	r	r

Bits 7:0 **HSICAL[7:0]**: HSI calibration

This register is initially loaded with the factory calibration value used to trim the HSI oscillator.

### 9.14.15 HSI clock calibration trimming register (CLK\_HSITRIMR)

Address offset: 0x0D

Reset value: 0x00

7	6	5	4	3	2	1	0
HSITRIM[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **HSITRIM[7:0]**: HSI trimming value.

To ensure the best HSI clock accuracy, the value to be written to this register should be within the following range: [(HSICALR register value) -12] < x < [(HSICALR register value) + 8].

*Note: Once this register configured, its value is used instead of the HSICALR register values. These bits are used by the application to choose/change the HSI oscillator trimming value. A hardware protection can be provided to avoid erroneous write access to this register, refer to the CLK\_HSIUNLCK register.*

### 9.14.16 HSI unlock register (CLK\_HSIUNLCKR)

Address offset: 0x0E

Reset value: 0x00

7	6	5	4	3	2	1	0
HSIUNLCK[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **HSIUNLCK[7:0]**: HSI unlock mechanism

This register is used by the application for both unlocking the hardware write protection of HSITRIM register and enabling the use of HSITRIM as trimming value for HSI oscillator.

The HSITRIM unlock/enable procedure consists in:

- 1) two consecutive write accesses at this address, the first one with the value 0xAC and the second one with the value 0x35
- 2) a write access to the HSITRIM register.

*Note: When this procedure is correctly completed the HSITRIM will be locked again*

### 9.14.17 Main regulator control status register (CLK\_REGCSR)

Address offset: 0x0F

Reset value: 0xB9

7	6	5	4	3	2	1	0
EEREADY	EEBUSY	LSEPD	HSEPD	LSIPD	HSIPD	REGOFF	REGREADY
r	r	r	r	r	r	rw	r

**Bit 7 EEREADY:** Flash program memory and Data EEPROM ready

This bit indicated if the Flash program memory and Data EEPROM are ready

0: Flash program memory and Data EEPROM not ready

1: Flash program memory and Data EEPROM ready

**Bit 6 EEBUSY:** Flash program memory and Data EEPROM busy

This bit indicates if the Flash program memory and Data EEPROM ready are busy

0: Flash program memory and Data EEPROM not busy

1: Flash program memory and Data EEPROM busy

**Bit 5 LSEPD:** LSE power-down

This bit indicates the status of the LSE oscillator

0: LSE oscillator ON

1: LSE oscillator OFF

**Bit 4 HSEPD:** HSE power-down

This bit indicates the status of the HSE oscillator

0: HSE oscillator ON

1: HSE oscillator OFF

**Bit 3 LSIPD:** LSI power-down

This bit indicates the status of the LSI oscillator

0: LSI oscillator ON

1: LSI oscillator OFF

**Bit 2 HSIPD:** HSI power-down

This bit indicates the status of the oscillator

0: HSI oscillator ON

1: HSI oscillator OFF

**Bit 1 REGOFF:** Main regulator OFF

This bit enables the direct switch-off of the main regulator regardless of the REGREADY flag status

0: The main regulator follows the standard functionality

1: The main regulator is switched off and all clocks are provided without taking into account the REGREADY flag status

**Bit 0 REGREADY:** Main regulator ready

The REGREADY bit indicates the main regulator status: this bit is set when the main regulator is ready to provide the full power.

### 9.14.18 CLK register map and reset values

**Table 23. CLK register map and reset values**

Address offset <sup>(1)</sup>	Register name	7	6	5	4	3	2	1	0
0x00	CLK_CKDIVR Reset value	-0	-0	-0	-0	-0	CKM20	CKM11	CKM01
0x01	CLK_CRTCR Reset value	RTC DIV20	RTC DIV10	RTC DIV00	RTC SEL30	RTC SEL20	RTC SEL10	RTC SEL00	RTC SWBSY0
0x02	CLK_ICKCR Reset value	-0	BEEPAHALT0	FHW0	SAHALT1	LSIRDY0	LSION0	HSIRDY0	HSION1
0x03	CLK_PCKENR1 Reset value	PCKEN170	PCKEN160	PCKEN150	PCKEN140	PCKEN130	PCKEN120	PCKEN110	PCKEN100
0x04	CLK_PCKENR2 Reset value	PCKEN271	-0	PCKEN250	PCKEN240	PCKEN230	PCKEN220	PCKEN210	PCKEN200
0x05	CLK_CCOR Reset value	CCODIV20	CCODIV10	CCODIV00	CCOSEL30	CCOSEL20	CCOSEL10	CCOSEL00	CCOSWBSY0
0x06	CLK_ECKCR Reset value	-0	-0	LSEBYP0	HSEBYP0	LSERDY0	LSEON0	HSERDY0	HSEON0
0x07	CLK_SCSR Reset value	CKM70	CKM60	CKM50	CKM40	CKM30	CKM20	CKM10	CKM01
0x08	CLK_SWR Reset value	SWI70	SWI60	SWI50	SWI40	SWI30	SWI20	SWI10	SWI01
0x09	CLK_SWCR Reset value	-x	-x	-x	-x	SWIF0	SWIEN0	SWEN0	SWBSY0
0x0A	CLK_CSSR Reset value	-0	-0	-0	CSSDGON0	CSSD0	CSSDIE0	AUX0	CSSEN0
0x0B	CLK_CBEEP Reset value	-0	-0	-0	-0	-0	CLKBEEP SEL10	CLKBEEP SEL00	BEEP SWBSY0
0x0C	CLK_HSICALR Reset value	HSICAL70	HSICAL60	HSICAL50	HSICAL40	HSICAL30	HSICAL20	HSICAL10	HSICAL00
0x0D	CLK_HSITRIMR Reset value	HSITRIM70	HSITRIM60	HSITRIM50	HSITRIM40	HSITRIM30	HSITRIM20	HSITRIM10	HSITRIM00
0x0E	CLK_HSIUNLCKR Reset value	HSIUNLCK70	HSIUNLCK60	HSIUNLCK50	HSIUNLCK40	HSIUNLCK30	HSIUNLCK20	HSIUNLCK10	HSIUNLCK00
0x0F	CLK_REGCSR Reset value	EEREADYx	EEBUSYx	LSEPD1	HSEPD1	LSIPD1	HSIPD0	REGOFF0	REGREADYx
0x10 <sup>(2)</sup>	CLK_PCKENR3 Reset value	-0	-0	PCKEN350	PCKEN340	PCKEN330	PCKEN320	PCKEN310	PCKEN300

1. Please refer to the “general hardware register map” table in the datasheet for details on register addresses.
2. This register is available in low density, medium+ and high density devices only. In low density devices, only bit 5 is available.

## 10 General purpose I/O ports (GPIO)

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

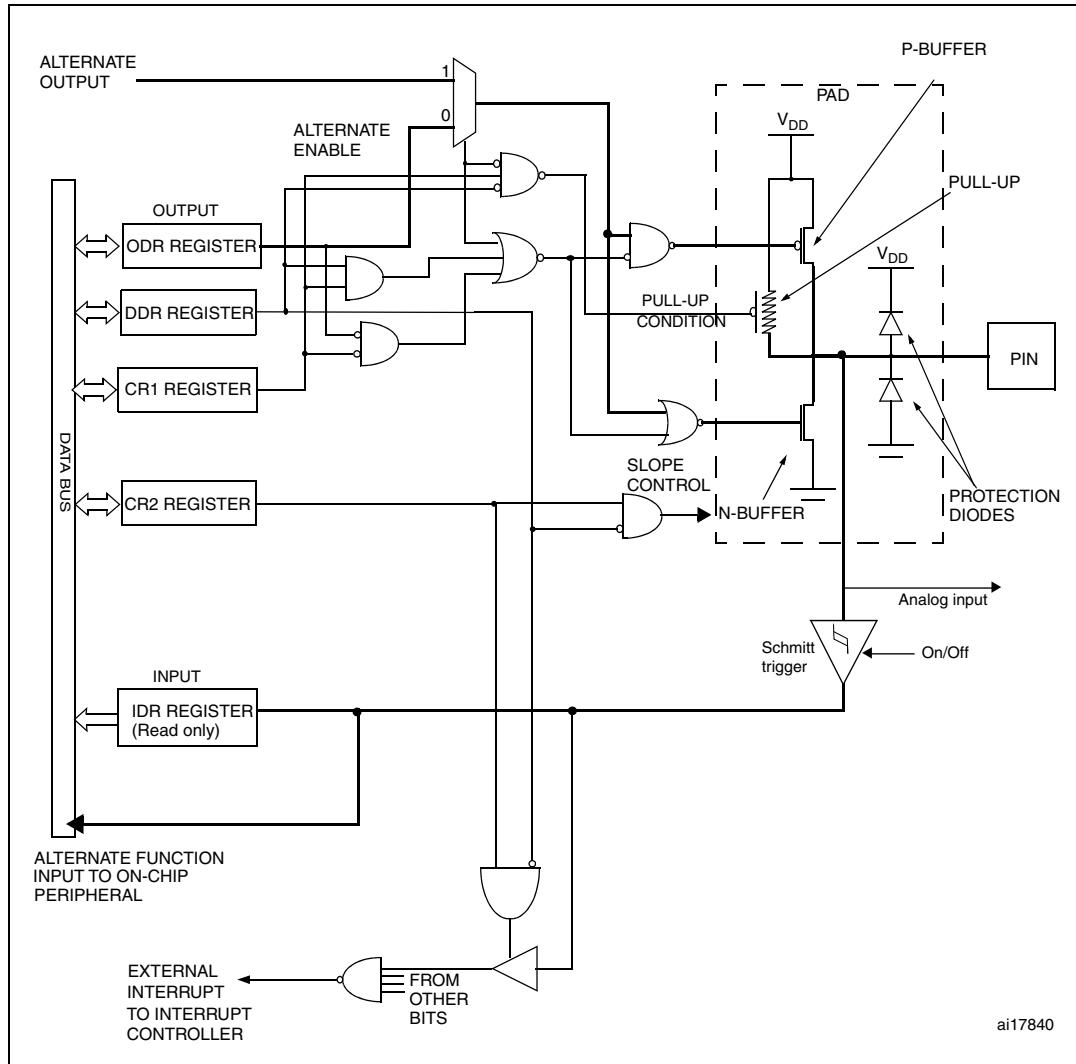
### 10.1 Introduction

General purpose input/output ports are used for data transfers between the chip and the external world. An I/O port can contain up to eight pins. Each pin can be individually programmed as a digital input or digital output. In addition, some ports may have alternate functions like analog inputs, external interrupts, input/output for on-chip peripherals. Only one alternate function can be mapped to a pin at a time.

An output data register, input data register, data direction register and two configuration registers are associated with each port. A particular port will behave as an input or output depending on the status of the data direction register of the port.

### 10.2 GPIO main features

- Port bits can be configured individually
- Selectable input modes: floating input or input with pull-up
- Selectable output modes: push-pull output or pseudo-open-drain.
- Separate registers for data input and output
- External interrupts can be enabled and disabled individually
- Output slope control for reduced EMC noise
- Alternate function I/Os for on-chip peripherals
- Read-modify-write possible on data output latch
- I/O state guaranteed in voltage range 1.6 V to  $V_{DDIOMax}$

**Figure 22. GPIO block diagram**

**Note:** In the open-drain, 3.6 V tolerant and 5 V tolerant I/Os, protection diode to  $V_{DD}$  is not implemented.

### 10.3 Port configuration and usage

An output data register (ODR), pin input register (IDR), data direction register (DDR) are always associated with each port.

The control register 1 (CR1) and control register 2 (CR2) allow input/output options. An I/O pin is programmed using the corresponding bits in the DDR, ODR, CR1 and CR2 registers.

Bit  $n$  in the registers corresponds to pin  $n$  of the Port.

The various configurations are summarized in [Table 24](#).

**Table 24. I/O port configuration summary**

Mode	DDR bit	CR1 bit	CR2 bit	Function	Pull-up	P-buffer	Diodes	
							to V <sub>DD</sub>	to V <sub>SS</sub>
Input	0	0	0	Floating without interrupt	Off	Off	On <sup>(1)</sup>	On
	0	1	0	Pull-up without interrupt	On			
	0	0	1	Floating with interrupt	Off			
	0	1	1	Pull-up with interrupt	On			
Output	1	0	0	Open drain output	Off	Off	On <sup>(1)</sup>	On
	1	1	0	Push-pull output		On		
	1	0	1	Open drain output, fast mode		Off		
	1	1	1	Push-pull, fast mode	Off	On		
	1	x	x	True open drain (on specific pins)	Not implemented		Not implemented <sup>(2)</sup>	

1. In 3.6 V and 5 V tolerant I/Os, protection diode to V<sub>DD</sub> is not implemented.
2. The diode connected to V<sub>DD</sub> is not implemented in true open drain pads. A local protection between the pad and V<sub>OL</sub> is implemented to protect the device against positive stress.

**Warning:** On some packages, some ports must be considered as active even if they do not exist on the package. To avoid spurious effects, configure them as pull-up inputs without interrupt at startup, and keep them in this state when changing the port configuration. Refer to the datasheet for additional information.

### 10.3.1 Input modes

Clearing the DDRx bit selects input mode. In this mode, reading a IDR bit returns the digital value of the corresponding I/O pin.

Refer to [Section 10.7: Input mode details on page 119](#) for information on analog input, external interrupts and Schmitt trigger enable/disable.

As shown in [Table 24](#), four different input modes can be theoretically be configured by software: floating without interrupt, floating with interrupt, pull-up without interrupt or pull-up with interrupt. However in practice, not all ports have external interrupt capability or pull-ups. You should refer to the datasheet pin-out description for details on the actual hardware capability of each port.

### 10.3.2 Output modes

Setting the DDRx bit selects output mode. In this mode, writing to the ODR bits applies a digital value to the I/O through the latch. Reading IDR bit returns the digital value from the corresponding I/O pin. Using the CR1, CR2 registers, different output modes can be configured by software: Push-pull output, Open-drain output.

Refer to [Section 10.8: Output mode details on page 120](#) for more information.

## 10.4 Reset configuration

All I/O pins are generally input floating under reset (i.e. during the reset phase) and at reset state (i.e. after reset release). However, a few pins may have a different behavior. Refer to the datasheet pinout description for all details.

## 10.5 Unused I/O pins

Unused I/O pins must be connected to fixed voltage levels and configured as input floating. Either connect a pull-up or pull-down to the unused I/O pins, or use the internal weak pull-up if it is available on the pins. It is also possible to configure them in output mode.

## 10.6 Low power modes

**Table 25. Effect of low power modes on GPIO ports**

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 wakeup from Halt mode.

## 10.7 Input mode details

### 10.7.1 Alternate function input

Some I/Os can be used as alternate function input. For example as the port may be used as the input capture input to a timer. Alternate function inputs are not selected automatically, you select them by writing to a control bit in the registers of the corresponding peripheral. For Alternate Function input, you should select floating or pull-up input configuration in the DDR and CR1 registers.

### 10.7.2 Interrupt capability

Each I/O can be configured as an input with interrupt capability by setting the CR2x bit while the I/O is in input mode. In this configuration, a signal edge or level input on the I/O generates an interrupt request.

Falling or rising edge sensitivity is programmed independently for each interrupt vector in the EXTI\_CR[2:1] registers.

External interrupt capability is only available if the port is configured in input mode.

#### Interrupt masking

Interrupts can be enabled/disabled individually by programming the corresponding bit in the configuration register (Px\_CR2). At reset state, the interrupts are disabled.

## 10.8 Output mode details

### 10.8.1 Alternate function output

Alternate function outputs provide a direct path from a peripheral to an output or to an I/O pad, taking precedence over the port bit in the data output latch register (Px\_ODR) and forcing the Px\_DDR corresponding bit to 1.

An alternate function output can be push-pull or pseudo-open drain depending on the peripheral and Control register 1 (Px\_CR1) and slope can be controlled depending on the Control register 2 (Px\_CR2) values.

#### *Examples:*

SPI outputs must be set-up as push-pull. The slope of SPI outputs is controlled by hardware and configured in fast mode to enable an optimal operation. The user must then keep the CR2 slope control bit cleared to avoid spurious interrupts.

### 10.8.2 Slope control

The maximum frequency that can be applied to an I/O can be controlled by software using the CR2 bit. Low frequency operation with improved EMC behavior is selected at reset. Higher frequency (up to 10 MHz) can be selected if needed. This feature can be applied in either open drain or push-pull output mode on I/O ports of output type O3 or O4. Refer to the pin description tables in the datasheets for the specific output type information for each pin.

## 10.9 GPIO registers

The bit of each port register drives the corresponding pin of the port.

### 10.9.1 Port x output data register (Px\_ODR)

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw							

Bits 7:0 **ODR[7:0]**: Output data register bits

Writing to the ODR register when in output mode applies a digital value to the I/O through the latch.  
Reading the ODR returns the previously latched value in the register.

In Input mode, writing in the ODR register, latches the value in the register but does not change the pin state. The ODR register is always cleared after reset. Bit read-modify-write instructions (BSET, BRST) can be used on the DR register to drive an individual pin without affecting the others.

### 10.9.2 Port x pin input register (Px\_IDR)

Address offset: 0x01

Reset value: 0XX

7	6	5	4	3	2	1	0
IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r

Bits 7:0 **IDR[7:0]**: Pin input values

The pin register can be used to read the pin value irrespective of whether port is in input or output mode. This register is read-only.

0: Low logic level

1: High logic level

*Note:* Px\_IDR reset value depends on the external circuitry.

### 10.9.3 Port x data direction register (Px\_DDR)

Address offset: 0x02

Reset value: 0x00

7	6	5	4	3	2	1	0
DDR7	DDR6	DDR5	DDR4	DDR3	DDR2	DDR1	DDR0
rw							

Bits 7:0 **DDR[7:0]**: Data direction bits

These bits are set and cleared by software to select input or output mode for a particular pin of a port.

0: Input mode

1: Output mode

### 10.9.4 Port x control register 1 (Px\_CR1)

Address offset: 0x03

Reset value: 0x00 except for PA\_CR1 which reset value is 0x01.

7	6	5	4	3	2	1	0
C17	C16	C15	C14	C13	C12	C11	C10
rw							

Bits 7:0 **C1[7:0]**: Control bits

These bits are set and cleared by software. They select different functions in input mode and output mode (see).

– **In input mode (DDR = 0):**

0: Floating input

1: Input with pull-up

– **In output mode (DDR = 1):**

0: Pseudo open drain

1: Push-pull, slope control for the output depends on the corresponding CR2 bit

Note: This bit has no effect on true open drain ports (refer to pin marked "T" in datasheet pin description table).

### 10.9.5 Port x control register 2 (Px\_CR2)

Address offset: 0x04

Reset value: 0x00

7	6	5	4	3	2	1	0
C27	C26	C25	C24	C23	C22	C21	C20
rw							

Bits 7:0 **C2[7:0]**: Control bits

These bits are set and cleared by software. They select different functions in input mode and output mode. In input mode, the CR2 bit enables the interrupt capability if available. If the I/O does not have interrupt capability, setting the CR2 bit has no effect. In output mode, setting the bit increases the speed of the I/O. This applies to ports with O3 and O4 output types (see pin description table).

- **In input mode (DDR = 0):**

- 0: External interrupt disabled
- 1: External interrupt enabled

- **In output mode (DDR = 1):**

- 0: Output speed up to 2 MHz
- 1: Output speed up to 10 MHz

### 10.9.6 Peripheral alternate function remapping

Some peripheral alternate functions can be remapped to different I/O ports through two remapping registers. Refer to SYSCFG remap control register 1 (SYSCFG\_RMPCR1) and SYSCFG remap control register 2 (SYSCFG\_RMPCR2) in [Section 11: Routing interface \(RI\) and system configuration controller \(SYSCFG\)](#).

### 10.9.7 GPIO register map and reset values

Each GPIO port has five registers mapped as shown in [Table 26](#). Refer to the register map in the corresponding datasheet for the base address for each port.

*Note:* At reset state, all ports are input floating. Exceptions are indicated in the pin description table of the corresponding datasheet.

**Table 26. GPIO register map**

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	Px_ODR	ODR7 0	ODR6 0	ODR5 0	ODR4 0	ODR3 0	ODR2 0	ODR1 0	ODR0 0
0x01	Px_IDR	IDR7 x	IDR6 x	IDR5 x	IDR4 x	IDR3 x	IDR2 x	IDR1 x	IDR0 x
0x02	Px_DDR	DDR7 0	DDR6 0	DDR5 0	DDR4 0	DDR3 0	DDR2 0	DDR1 0	DDR0 0
0x03	Px_CR1 <sup>(1)</sup>	C17 0	C16 0	C15 0	C14 0	C13 0	C12 0	C11 0	C10 0
0x04	Px_CR2	C27 0	C26 0	C25 0	C24 0	C23 0	C22 0	C21 0	C20 0

1. PA\_CR1 reset value is 0x01.

## 11 Routing interface (RI) and system configuration controller (SYSCFG)

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

### 11.1 Introduction

The system configuration controller offers remapping capabilities of some alternate functions on different I/O ports and on TIM4 & ADC1 DMA channels. To use an alternate function, the corresponding peripheral must be enabled in the peripheral registers. Alternate function remapping does not affect the GPIO capabilities of the I/O ports (see [Section 10: General purpose I/O ports \(GPIO\) on page 116](#)). Refer to [Section 11.5: SYSCFG registers](#) for remapping capabilities on TIM4 & ADC1 DMA channels.

The routing interface provides high flexibility by allowing the software routing of I/Os to the input captures of TIM1. It also controls the routing of internal analog signals to ADC1, COMP1, COMP2, DAC and internal reference voltage  $V_{REFINT}$ . It also provides a set of registers for efficiently managing up to 20 capacitive sensing channels (low density devices) and up to 16 capacitive sensing channels (medium, medium+ and high density devices).

Refer to [Section 1.: Comparators are not available on STM8L05xx value line devices..](#)

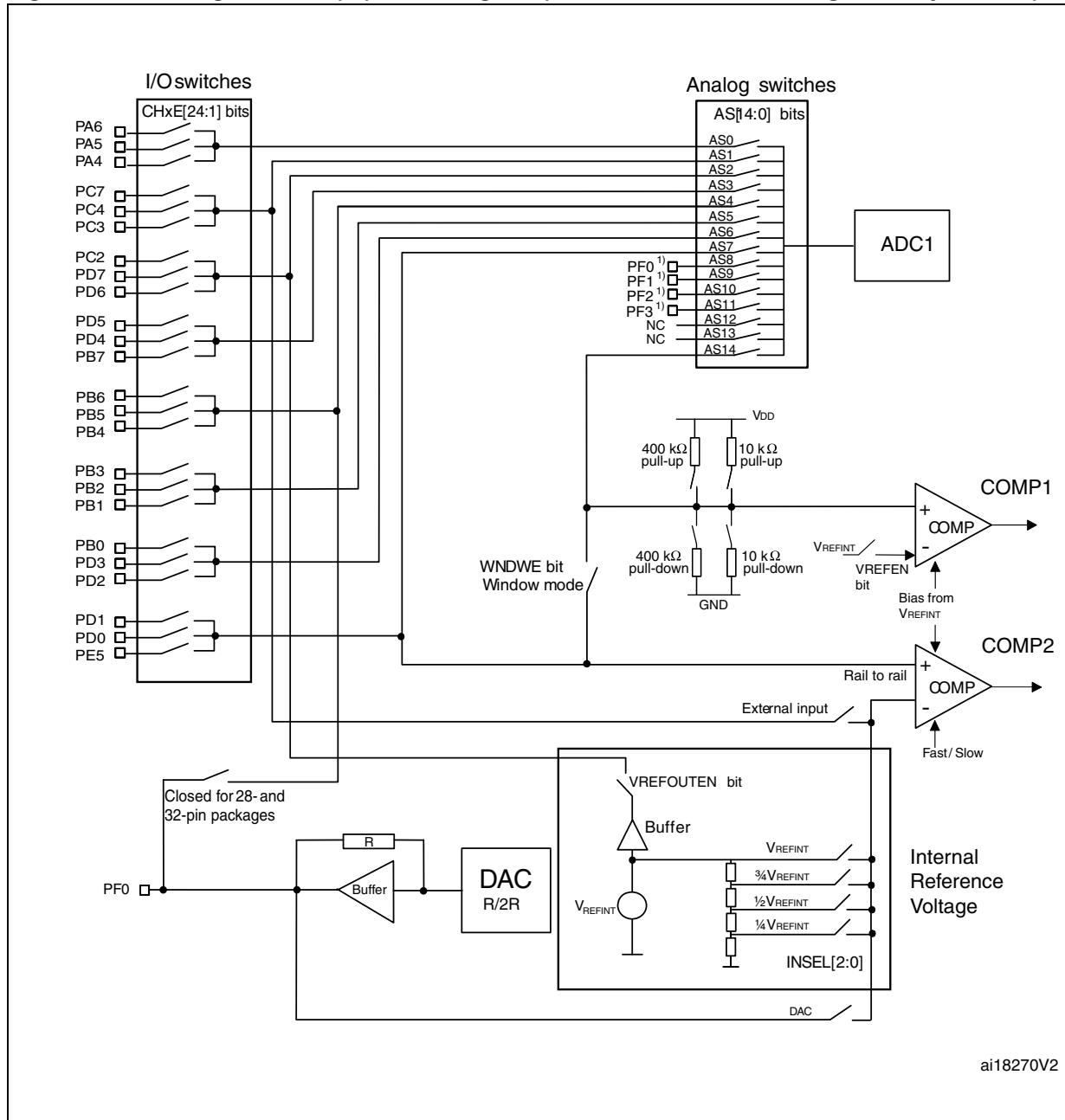
### 11.2 RI main features

- 28 programmable I/O switches (low density devices) and 24 programmable I/O switches (medium, medium+ and high density devices) for signal routing
- 10 programmable analog switches (low and medium density devices) or 13 programmable analog switches (medium+ and high density devices) for signal routing
- COMP1 and COMP2 input and output routing<sup>(1)</sup>
- TIM1 input capture 2 and 3 routing selection from selectable I/Os (medium, medium+ and high density devices).
- DAC output routing to selectable I/Os for 28 and 32-pin packages<sup>(1)</sup> (medium, medium+ and high density devices).
- Internal reference voltage routing to selectable I/Os.
- Software or hardware management of the charge transfer acquisition sequence.

---

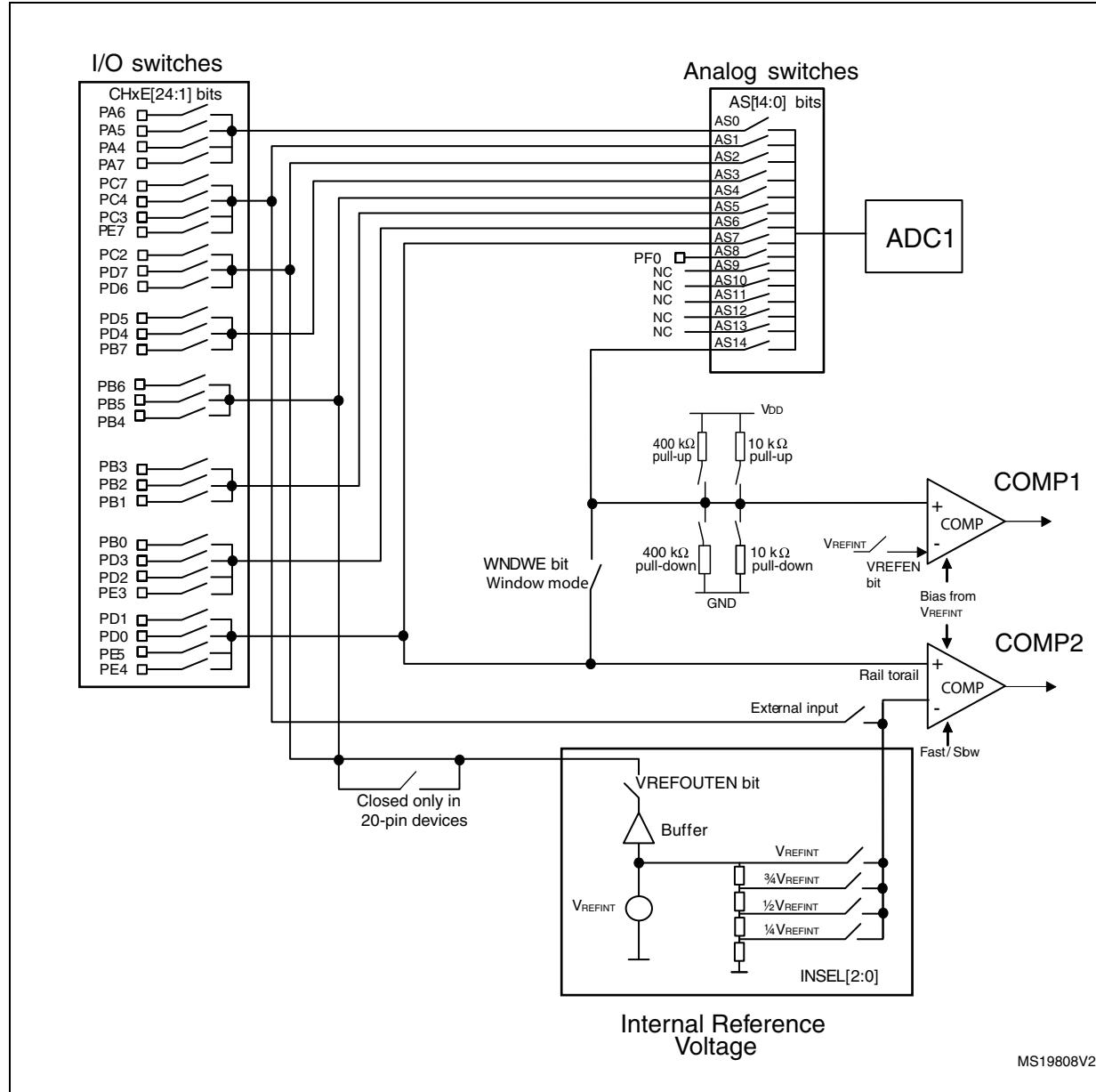
1. Not available on STM8L05xx value line devices.

Figure 23. Routing interface (RI) block diagram (medium, medium+ and high density devices)



1. On medium+ and high density devices only.
2. DAC and comparators are not available on STM8L05xx value line devices.

Figure 24. Routing interface (RI) block diagram (low density devices)



- Comparators are not available on STM8L05xx value line devices.

### 11.2.1 RI functional description

The RI registers can be accessed only when the comparator clock is enabled by setting the PCKEN25 bit in the CLK\_PCKENR2 register. Refer to [Section 9.14.5 on page 105](#).

### 11.2.2 I/O groups

On low density devices, 28 general purpose I/Os are grouped into 4 groups of three I/Os each and into groups of 4 I/Os each. On medium, medium+ and high density devices, 24

general purpose I/Os are grouped into 4 groups of three I/Os each. [Table 27](#) shows the I/O groups and the control registers used to route them to the analog blocks.

- Two blocks of switches control the routing of signals to the analog blocks.
  - I/O switches controlled by the RI\_IOSRx registers
  - Analog switches controlled by the RI\_ASCRx registers
- An additional set of switches controls the routing to the TIM1 timer inputs (not shown in block diagram)
- A complementary set of registers controls the configuration of the I/Os and are designed to manage up to 20 capacitive sensing channels on low density devices and up to 16 capacitive sensing channels on medium, medium+ and high density devices (not shown in the block diagram).

When the I/Os are switched to analog mode (I/O switches CHxE are set or ADC used), the Schmitt trigger is disabled by default. At that time, the bit in the Px\_IDR register associated with the I/O is always read as 0 whatever the level on the pin. The Schmitt trigger can be enabled by setting the STE bit in the COMP\_CSR1 register but it acts as a simple comparator in this configuration mode. In that case, it is possible to read the I/O state through the Px\_IDR register while the hysteresis is kept disabled to reduce the power consumption of the device. The state of an I/O switched to analog mode can be read through the RI\_IOIRx registers anyway whatever the STE bit value.

- **When I/Os are used as ADC inputs:** the I/O switch and analog switches are controlled directly by the ADC. The corresponding CHxE and ASx bits in the RI registers are not used and must be kept cleared (switches left open).
- **When I/Os are used as inputs/outputs for analog blocks other than the ADC:** the I/O switch and analog switch have to be controlled by the RI\_IOSRx and RI\_ASCRx registers. The corresponding CHxE and ASx bits in the RI registers must be set by software to close the switches and cleared to open the switches.

**Table 27. I/O groups and selection**

I/O Group	GPIO port	I/O switch control bit RI_IOSRx registers	ADC1 input	Analog switch control bit in RI_ASCRx register	Comparator <sup>(1)</sup> / DAC <sup>(1)</sup> /V <sub>REFINT</sub> functions
Group 1	PA6	CH1E bit	ADC1_IN0	AS0 bit	COMP1+ in
	PA5	CH2E bit	ADC1_IN1		
	PA4	CH3E bit	ADC1_IN2		
	PA7 <sup>(2)</sup>	CH29E bit	-		
Group 2	PC7	CH4E bit	ADC1_IN3	AS1 bit	COMP1+ in COMP2- in
	PC4	CH5E bit	ADC1_IN4		
	PC3	CH6E bit	ADC1_IN5		
	PE7 <sup>(2)</sup>	CH26E bit	ADC1_IN25		
Group 3	PC2	CH7E bit	ADC1_IN6	AS2 bit	COMP1+ in V <sub>REFINT</sub> out
	PD7	CH8E bit	ADC1_IN7		
	PD6	CH9E bit	ADC1_IN8		

**Table 27. I/O groups and selection (continued)**

I/O Group	GPIO port	I/O switch control bit RI_IOSRx registers	ADC1 input	Analog switch control bit in RI_ASCR1 register	Comparator <sup>(1)</sup> / DAC <sup>(1)</sup> /V <sub>REFINT</sub> functions
Group 4	PD5	CH10E bit	ADC1_IN9	AS3 bit	COMP1+ in
	PD4	CH11E bit	ADC1_IN10		
	PB7	CH12E bit	ADC1_IN11		
Group 5	PB6	CH13E bit	ADC1_IN12	AS4 bit	COMP1+ in DAC out
	PB5	CH14E bit	ADC1_IN13		
	PB4	CH15E bit	ADC1_IN14		
Group 6	PB3	CH16E bit	ADC1_IN15	AS5 bit	COMP1+ in
	PB2	CH17E bit	ADC1_IN16		
	PB1	CH18E bit	ADC1_IN17		
Group 7	PB0	CH19E bit	ADC1_IN18	AS6 bit	COMP1+ in
	PD3	CH20E bit	ADC1_IN19		
	PD2	CH21E bit	ADC1_IN20		
	PE3 <sup>(2)</sup>	CH27E bit	ADC1_IN26		
Group 8	PD1	CH22E bit	ADC1_IN21	AS7 bit	COMP1+ in COMP2+ in
	PD0	CH23E bit	ADC1_IN22		
	PE5	CH24E bit	ADC1_IN23		
	PE4 <sup>(2)</sup>	CH28E bit	ADC1_IN27		
NA	PF0	NA	ADC1_IN24	AS8 bit	COMP1+ in
NA	PF1 <sup>(3)</sup>	NA	ADC1_IN25	AS9 bit	COMP1+ in
NA	PF2 <sup>(2)</sup>	NA	ADC1_IN26	AS10 bit	COMP1+ in
NA	PF3 <sup>(2)</sup>	NA	ADC1_IN27	AS11 bit	COMP1+ in

1. DAC and comparators are not available on STM8L05xx value line devices.

2. On low density devices only

3. On medium+ and high density devices only.

### 11.2.3 TIM1 input capture routing

TIM1 is not available in low density devices.

After reset, the Timer 1 Input Capture 2 and Input Capture 3 signals are connected to the I/O port assigned in the datasheet pinout (default routing). The I/O routing can be changed by programming registers RI\_ICR1 and RI\_ICR2. Refer to [Table 28](#).

**Table 28. TIM1 input capture routing**

<b>IC2CS[4:0] / IC3CS[4:0] bit value</b>	<b>I/O port routed to TIM1.IC2 input</b>	<b>I/O port routed to TIM1.IC3 input</b>
00000	default routing (PD4)	default routing (PD5)
00001	PF0	Reserved (medium density devices) PF1 (medium+ and high density devices)
00010	Reserved (medium density devices) PF2 (medium+ and high density devices)	Reserved (medium density devices) PF3 (medium+ and high density devices)
00011	PE0	PE1
00100	PE2	PE3
00101	PE4	PE5
00110	PE6	PE7
00111	PD0	PD1
01000	PD2	PD3
01001	PD4	PD5
01010	PD6	PD7
01011	PC0	PC1
01100	PC2	PC3
01101	PC4	PC5
01110	PC6	PC7
01111	PB0	PB1
10000	PB2	PB3
10001	PB4	PB5
10010	PB6	PB7
10011	PA0	PA2
10100	PA3	PA4
10101	PA5	PA6
10110	PA7	default routing (PD5)
other values	default routing (PD4)	default routing (PD5)

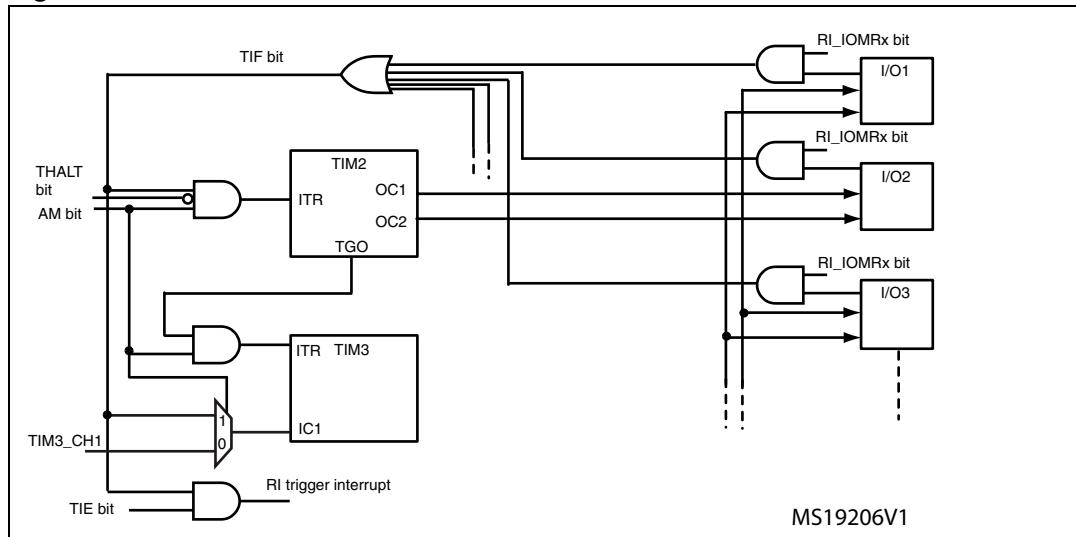
#### 11.2.4 TIM2 & TIM3 routing

*Note:* Only available on low density devices.

To reduce the CPU load required for the management of the 20 capacitive sensing channels, low density devices feature a hardware mode for handling the charger transfer acquisition sequence. This is done by using timer TIM2 to control the state of the I/Os in the I/O groups. Timer TIM3 is used to count the number of charge transfer cycles generated before the voltage across  $C_S$  reaches  $V_{IH}$ .

Refer to [Figure 25](#).

**Figure 25. TIM2 and TIM3 interconnections**



When using the hardware acquisition mode:

- TIM2 OC1 controls the state of the electrode I/O
  - When OC1 is high, the electrode I/O is set to output push-pull high.
  - When OC1 is low, the electrode I/O is set to input floating mode
- TIM2 OC2 controls both the sampling capacitor and the electrode I/O analog switch
  - When OC2 is high, the analog switch is closed
  - When OC2 is low, the analog switch is open

### 11.2.5 Comparator routing

Refer to [Section 16.3: Comparator 1 \(COMP1\) on page 241](#) for the description of the COMP1 interconnections.

Refer to [Section 16.4: Comparator 2 \(COMP2\) on page 242](#) for the description of the COMP2 interconnections.

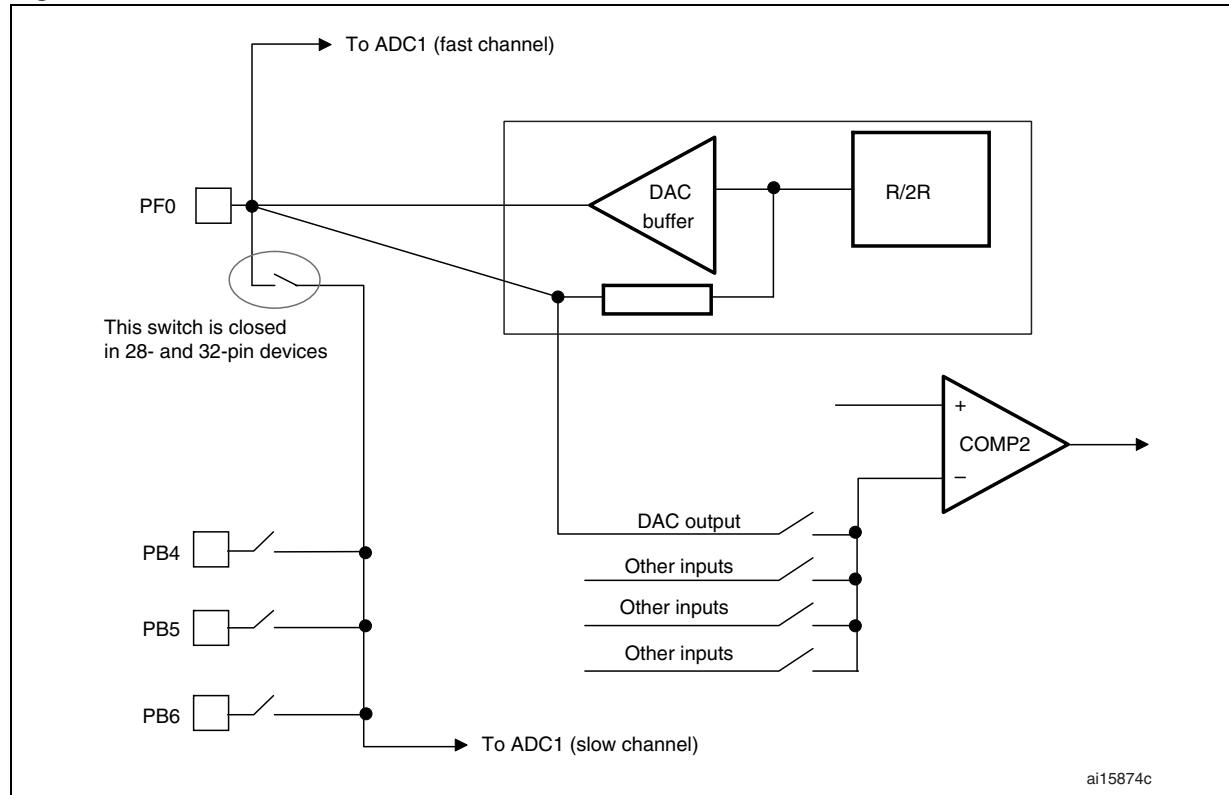
### 11.2.6 DAC routing

DAC is not available on low density and STM8L05xx value line devices.

In medium density devices, for 28 and 32-pin devices, the DAC\_OUT1 can be routed to any I/O of group 5 by setting the corresponding I/O switch bit (CH13E in the RI\_IOSR1 register, CH14E in the RI\_IOSR2 register or CH15E in the RI\_IOSR3 register).

In medium+ and high density devices, for 48-pin devices, the DAC\_OUT2 can be routed to any I/O of group 5 by setting the corresponding I/O switch bit (CH13E in the RI\_IOSR1 register, CH14E in the RI\_IOSR2 register or CH15E in the RI\_IOSR3 register).

*Note:* In this case group 5 cannot be used for ADC1.

**Figure 26. DAC interconnections**

ai15874c

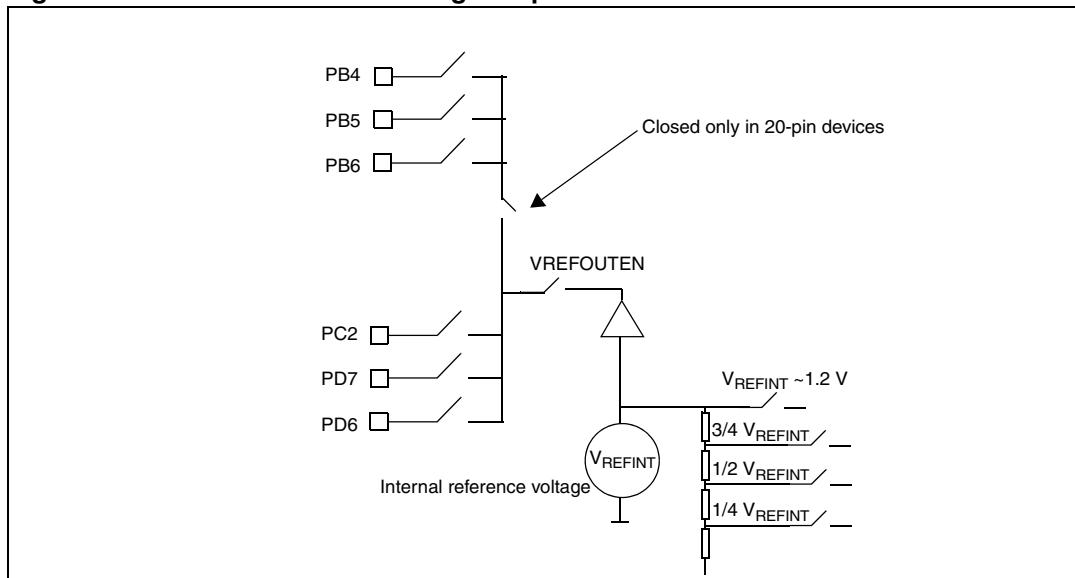
The DAC\_OUT1 can be routed to the COMP2 inverting input by writing the value 0b110 in the INSEL[2:1] bits in the COMP\_CSR3 register.

In medium density devices, for 48-pin packages, the DAC\_OUT1 is connected to the ADC1\_IN24 input.

In medium+ and high density devices, for 64 and 80-pin packages, the DAC\_OUT2 is connected to the ADC1\_IN25 input.

### 11.2.7 Internal reference voltage routing

**Figure 27. Internal reference voltage output**



Internal reference voltage output can be routed to any I/O of group 3 following this procedure:

1. Set the bit VREFOUTEN in COMP\_CSR3
2. Close the I/O switch of any I/O of group 3 by setting CH7E in RI\_IOSR1 or CH8E in RI\_IOSR2 or CH9E in RI\_IOSR3.

## 11.3 RI interrupts

**Table 29. RI interrupt requests**

Interrupt event	Event flag	Enable control bit	Exit from Wait/ Low power wait	Exit from Halt/ Active-halt
Interrupt event trigger	TIF	TIE	Yes	No

## 11.4 RI registers

### 11.4.1 Timer input capture routing register 1 (RI\_ICR1)

Address offset: 0x01

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved			IC2CS[4:0]				

Bits 7:5 Reserved, must be kept cleared.

Bits 4:0 **IC2CS[4:0]**: TIM1 Input Capture 2 I/O selection

These bits are set and cleared by software. They select which I/O port is routed to Timer 1 input Capture 2. Refer to [Table 28](#)

*Note:* This register is not available on low density devices.

### 11.4.2 Timer input capture routing register 2 (RI\_ICR2)

Address offset: 0x02

Reset value: 0x00

7	6	5	4	3	2	1	0		
Reserved			IC2CS[4:0]						
			rw	rw	rw	rw	rw		

Bits 7:5 Reserved, must be kept cleared.

Bits 4:0 **IC3CS[4:0]**: TIM1 Input Capture 3 I/O selection

These bits are set and cleared by software. They select which I/O port is routed to Timer 1 Input Capture 3. Refer to [Table 28](#).

*Note:* This register is not available on low density devices.

### 11.4.3 I/O input register 1 (RI\_IOIR1)

Address offset: 0x03

Reset value: 0xXX where X is undefined

7	6	5	4	3	2	1	0
CH22I	CH19I	CH16I	CH13I	CH10I	CH7I	CH4I	CH1I
r	r	r	r	r	r	r	r

Bits 7:0 **CHxI**: I/O pin input value

These bits return the corresponding I/O pin value irrespective of the port configuration (similar to the Px\_IDR register). This register is read-only.

0: Low logic level

1: High logic level

*Note:* This register controls the first I/O of each group of I/Os.

### 11.4.4 I/O input register 2 (RI\_IOIR2)

Address offset: 0x04

Reset value: 0xXX where X is undefined

7	6	5	4	3	2	1	0
CH23I	CH20I	CH17I	CH14I	CH11I	CH8I	CH5I	CH2I
r	r	r	r	r	r	r	r

Bits 7:0 **CHxI**: I/O pin input value

These bits return the corresponding I/O pin value irrespective of the port configuration (similar to the Px\_IDR register). This register is read-only.

- 0: Low logic level
- 1: High logic level

*Note:* This register controls the second I/O of each group of I/Os.

#### 11.4.5 I/O input register 3 (RI\_IOIR3)

Address offset: 0x05

Reset value: 0xXX where X is undefined

7	6	5	4	3	2	1	0
CH24I	CH21I	CH18I	CH15I	CH12I	CH9I	CH6I	CH3I
r	r	r	r	r	r	r	r

Bits 7:0 **CHxI**: I/O pin input value

These bits return the corresponding I/O pin value irrespective of the port configuration (similar to the Px\_IDR register). This register is read-only.

- 0: Low logic level
- 1: High logic level

*Note:* This register controls the third I/O of each group of I/Os.

#### 11.4.6 I/O control mode register 1 (RI\_IOCMR1)

Address offset: 0x06

Reset value: 0x00

7	6	5	4	3	2	1	0
CH22M	CH19M	CH16M	CH13M	CH10M	CH7M	CH4M	CH1M
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CHxM**: I/O control mode

These bits are set and cleared by software to select how the I/O is controlled.

0: I/O x is controlled by the standard GPIO registers. The I/O x switch is directly controlled by the RI\_IOSR1 register.

1: I/O x is set to protected mode and is only controlled by the RI\_IOSR1 and RI\_IGCR registers (standard GPIO registers have no effect on the configuration of I/O x).

*Note:* This register controls the first I/O of each group of I/Os.

#### 11.4.7 I/O control mode register 2 (RI\_IOCMR2)

Address offset: 0x07

Reset value: 0x00

7	6	5	4	3	2	1	0
CH23M	CH20M	CH17M	CH14M	CH11M	CH8M	CH5M	CH2M
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CHxM:** I/O control mode

These bits are set and cleared by software to select how the I/O is controlled.

0: I/O x is controlled by the standard GPIO registers. The I/O x switch is directly controlled by the RI\_IOSR2 register.

1: I/O x is set to protected mode and is only controlled by the RI\_IOSR2 and RI\_IGCR registers (standard GPIO registers have no effect on the configuration of I/O x).

*Note: This register controls the second I/O of each group of I/Os.*

#### 11.4.8 I/O control mode register 3 (RI\_IOCMR3)

Address offset: 0x08

Reset value: 0x00

7	6	5	4	3	2	1	0
CH24M	CH21M	CH18M	CH53M	CH12M	CH9M	CH6M	CH3M
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CHxM:** I/O control mode

These bits are set and cleared by software to select how the I/O is controlled.

0: I/O x is controlled by the standard GPIO registers. The I/O x switch is directly controlled by the RI\_IOSR3 register.

1: I/O x is set to protected mode and is only controlled by the RI\_IOSR3 and RI\_IGCR registers (standard GPIO registers have no effect on the configuration of I/O x).

*Note: This register controls the third I/O of each group of I/Os.*

#### 11.4.9 I/O switch register 1 (RI\_IOSR1)

Address offset: 0x09

Reset value: 0x00

7	6	5	4	3	2	1	0
CH22E	CH19E	CH16E	CH13E	CH10E	CH7E	CH4E	CH1E
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CHxE**: I/O switch control

**If the corresponding RI\_IOCMRx bit is cleared:**

These bits are set and cleared by software to open and close the I/O switches. If the I/O is used for the ADC1 input, the switch must be left open to allow the ADC1 to control it.

0: I/O x switch is open (I/O x is controlled by the GPIO registers or ADC1).

1: I/O x switch is closed (analog channel enabled).

**If the corresponding RI\_IOCMRx bit is set and the AM bit is cleared in the RI\_CR register (software management of the charge transfer acquisition sequence):**

These bits are set and cleared by software. They define the configuration of the I/O.

0: I/O x is forced to output push-pull low mode.

1: I/O x configuration is determined by bit 0 and bit 1 in the RI\_IOGCR register.

**If the corresponding RI\_IOCMRx bit is set and the AM bit is also set in the RI\_CR register (hardware management of the charge transfer acquisition sequence):**

These bits are set and cleared by software. They define if the corresponding I/O x state is controlled by the TIM2 timer.

0: I/O x is forced to output push-pull low mode

1: I/O x configuration is determined by the timer TIM2

*Note: This register controls the first I/O of each group of I/Os.*

### 11.4.10 I/O switch register 2 (RI\_IOSR2)

Address offset: 0x0A

Reset value: 0x00

7	6	5	4	3	2	1	0
CH23E	CH20E	CH17E	CH14E	CH11E	CH8E	CH5E	CH2E

Bits 7:0 **CHxE**: I/O switch control.

**If the corresponding RI\_IOCMRx bit is cleared:**

These bits are set and cleared by software to open and close the I/O switches. If the I/O is used for the ADC1 input, the switch must be left open to allow the ADC1 to control it.

- 0: I/O x switch is open (I/O x is controlled by the GPIO registers or ADC1).
- 1: I/O x switch is closed (analog channel enabled).

**If the corresponding RI\_IOCMRx bit is set and the AM bit is cleared in the RI\_CR register (software management of the charge transfer acquisition sequence):**

These bits are set and cleared by software. They define the configuration of the I/O.

- 0: I/O x is forced to output push-pull low mode.
- 1: I/O x configuration is determined by bit 2 and bit 3 in the RI\_IOGCR register.

**If the corresponding RI\_IOCMRx bit is set and the AM bit is also set in the RI\_CR register (hardware management of the charge transfer acquisition sequence):**

These bits are set and cleared by software. They define if the corresponding I/O x state is controlled by the TIM2 timer.

- 0: I/O x is forced to output push-pull low mode
- 1: I/O x configuration is determined by the timer TIM2

*Note: This register controls the second I/O of each group of I/Os.*

### 11.4.11 I/O switch register 3 (RI\_IOSR3)

Address offset: 0x0B

Reset value: 0x00

7	6	5	4	3	2	1	0
CH24E	CH21E	CH18E	CH15E	CH12E	CH9E	CH6E	CH3E

rw rw rw rw rw rw rw rw

Bits 7:0 **CHxE**: I/O Switch control.

**If the corresponding RI\_IOCMRx bit is cleared:**

These bits are set and cleared by software to open and close the I/O switches. If the I/O is used for the ADC1 input, the switch must be left open to allow the ADC1 to control it.

- 0: I/O x switch is open (I/O x is controlled by the GPIO registers or ADC1).
- 1: I/O x switch is closed (analog channel enabled).

**If the corresponding RI\_IOCMRx bit is set and the AM bit is cleared in the RI\_CR register (software management of the charge transfer acquisition sequence):**

These bits are set and cleared by software. They define the configuration of the I/O.

- 0: I/O x is forced to output push-pull low mode.
- 1: I/O x configuration is determined by bit 4 and bit 5 in the RI\_IOGCR register.

**If the corresponding RI\_IOCMRx bit is set and the AM bit is also set in the RI\_CR register (hardware management of the charge transfer acquisition sequence):**

These bits are set and cleared by software. They define if the corresponding I/O x state is controlled by the TIM2 timer.

- 0: I/O x is forced to output push-pull low mode
- 1: I/O x configuration is determined by the timer TIM2

*Note: This register controls the third I/O of each group of I/Os.*

### 11.4.12 IO group control register (RI\_IOGCR)

Address offset: 0x0C

Reset value: 0xFF

7	6	5	4	3	2	1	0
IOM4[1:0]		IOM3[1:0]		IOM2[1:0]		IOM1[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:6 **IOM4[1:0]**: I/O mode 4

**If the bit AM in the RI\_CR register is cleared (software management of the charge transfer acquisition sequence):**

These bits are set and cleared by software. They define the configuration of the I/O if the corresponding RI\_IOCMRx bit is set.

00: I/O x is forced to output push-pull low mode.

01: I/O x is forced to output push-pull high mode.

10: I/O x is forced to input floating mode with the I/O switch open.

11: I/O x is forced to input floating mode with the I/O switch closed. The I/O x Schmitt trigger hysteresis is disabled but the I/O pin value can still be read.

**If the AM bit in the RI\_CR register is set (hardware management of the charge transfer acquisition sequence):**

These bits are set and cleared by software. They define the usage of the I/O.

00: I/O used for the sampling capacitor

01: I/O used for the electrode

Other: reserved

*Note:* These bits control the fourth I/O of each group of I/Os (only available on low density devices).

Bits 5:4 **IOM3[1:0]**: I/O mode 3

**If the bit AM in the RI\_CR register is cleared (software management of the charge transfer acquisition sequence):**

These bits are set and cleared by software. They define the configuration of the I/O if the corresponding RI\_IOCMRx bit is set.

00: I/O x is forced to output push-pull low mode.

01: I/O x is forced to output push-pull high mode.

10: I/O x is forced to input floating mode with the I/O switch open.

11: I/O x is forced to input floating mode with the I/O switch closed. The I/O x Schmitt trigger hysteresis is disabled but the I/O pin value can still be read.

**On low density devices, if the AM bit in the RI\_CR register is set (hardware management of the charge transfer acquisition sequence):**

These bits are set and cleared by software. They define the usage of the I/O.

00: I/O used for the sampling capacitor

01: I/O used for the electrode

Other: reserved

*Note:* These bits control the third I/O of each group of I/Os .

Bits 3:2 **IOM2[1:0]**: I/O mode 2

*If the bit AM in the RI\_CR register is cleared (software management of the charge transfer acquisition sequence):*

These bits are set and cleared by software. They define the configuration of the I/O if the corresponding RI\_IOCMRx bit is set.

00: I/O x is forced to output push-pull low mode.

01: I/O x is forced to output push-pull high mode.

10: I/O x is forced to input floating mode with the I/O switch open.

11: I/O x is forced to input floating mode with the I/O switch closed. The I/O x Schmitt trigger hysteresis is disabled but the I/O pin value can still be read.

*On low density devices, if the AM bit in the RI\_CR register is set (hardware management of the charge transfer acquisition sequence):*

These bits are set and cleared by software. They define the usage of the I/O.

00: I/O used for the sampling capacitor

01: I/O used for the electrode

Other: reserved

*Note:* These bits control the second I/O of each group of I/Os.

Bits 1:0 **IOM1[1:0]**: I/O mode 1

*If the bit AM in the RI\_CR register is cleared (software management of the charge transfer acquisition sequence):*

These bits are set and cleared by software. They define the configuration of the I/O if the corresponding RI\_IOCMRx bit is set.

00: I/O x is forced to output push-pull low mode.

01: I/O x is forced to output push-pull high mode.

10: I/O x is forced to input floating mode with the I/O switch open.

11: I/O x is forced to input floating mode with the I/O switch closed. The I/O x Schmitt trigger hysteresis is disabled but the I/O pin value can still be read.

*On low density devices, if the AM bit in the RI\_CR register is set (hardware management of the charge transfer acquisition sequence):*

These bits are set and cleared by software. They define the usage of the I/O.

00: I/O used for the sampling capacitor

01: I/O used for the electrode

Other: reserved

*Note:* These bits control the first I/O of each group of I/Os.

### 11.4.13 Analog switch register 1 (RI\_ASCR1)

Address offset: 0x0D

Reset value: 0x00

7	6	5	4	3	2	1	0
AS7	AS6	AS5	AS4	AS3	AS2	AS1	AS0

Bits 7:0 **ASx**: Analog switch control

These bits are set and cleared by software to control the analog switches. If the I/O is used for ADC1 input, the switch must be left open to allow the ADC1 to control it.

0: Analog switch open

1: Analog switch closed

### 11.4.14 Analog switch register 2 (RI\_ASCR2)

Address offset: 0x0E

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved	AS14	Reserved		AS11 <sup>(1)</sup>	AS10 <sup>(1)</sup>	AS9 <sup>(1)</sup>	AS8

Bit 7 Reserved, must be kept cleared

Bit 6 **AS14**: Analog switch 14

This bit is set and cleared by software to control AS14 analog switch. If the I/O is used for ADC1 input, the switch must be left open to allow the ADC1 to control it.

0: Analog switch open

1: Analog switch closed

Bits 5:4 Reserved, must be kept cleared

Bit 3<sup>(1)</sup> **AS11**: Analog switch 11<sup>(1)</sup>

This bit is set and cleared by software to control AS11 analog switch. If the I/O is used for ADC1 input, the switch must be left open to allow the ADC1 to control it.

0: Analog switch open

1: Analog switch closed

Bit 2<sup>(1)</sup> **AS10**: Analog switch 10<sup>(1)</sup>

This bit is set and cleared by software to control AS10 analog switch. If the I/O is used for ADC1 input, the switch must be left open to allow the ADC1 to control it.

0: Analog switch open

1: Analog switch closed

Bit 1<sup>(1)</sup> **AS9**: Analog switch 9<sup>(1)</sup>

This bit is set and cleared by software to control AS9 analog switch. If the I/O is used for ADC1 input, the switch must be left open to allow the ADC1 to control it.

0: Analog switch open

1: Analog switch closed

**Bit 0 AS8:** Analog switch control

This bit is set and cleared by software to control AS8 analog switch. If the I/O is used for ADC1 input, the switch must be left open to allow the ADC1 to control it.

- 0: Analog switch open
- 1: Analog switch closed

1. Available on low, medium+ and high density devices only. Reserved on medium density devices.

#### 11.4.15 Resistor control register (RI\_RCR)

Address offset: 0x0F

Reset value: 0x00

7	6	5	4	3	2	1	0
				400KPD	10KPD	400KPU	10KPU
Reserved				rw	rw	rw	rw

Bits 7:4 Reserved, must be kept cleared

**Bit 3 400KPD:** 400 kΩ pull-down resistor

This bit enables the 400 kΩ pull-down resistor.

- 0: 400 kΩ pull-down resistor disabled
- 1: 400 kΩ pull-down resistor enabled

**Bit 2 10KPD:** 10 kΩ pull-down resistor

This bit enables the 10 kΩ pull-down resistor.

- 0: 10 kΩ pull-down resistor disabled
- 1: 10 kΩ pull-down resistor enabled

**Bit 1 400KPU:** 400 kΩ pull-up resistor

This bit enables the 400 kΩ pull-up resistor.

- 0: 400K pull-up resistor disabled
- 1: 400K pull-up resistor enabled

**Bit 0 10KPU:** 10 kΩ pull-up resistor

This bit enables the 10 kΩ pull-up resistor.

- 0: 10 kΩ pull-up resistor disabled
- 1: 10 kΩ pull-up resistor enabled

**Note:** 1 To connect the pull-up or pull-down to ADC1 input, the corresponding ADC1 analog switch must be enabled using the corresponding register.

2 To avoid extra power consumption, only one resistor should be enabled at a time.

### 11.4.16 Control register (RI\_CR)

Address offset: 0x20

Reset value: 0x00

7	6	5	4	3	2	1	0
				THALT	AM	TIF	TIE
Reserved				rw	rw	rc_w1	rw

Bits 7:4 Reserved, must be kept cleared.

Bit 3 **THALT**: *Timer halted mode*

This bit is set and cleared by software to select Timer halted mode. In this mode, TIM2 and TIM3 are halted when a trigger event occurs and remain halted until the event is cleared. This allows the event to be processed.

- 0: Timer halted mode disabled
- 1: Timer halted mode enabled

Bit 2 **AM**: *Acquisition mode*

This bit is set and cleared by software. It selects how the charge transfer acquisition sequence is managed.

- 0: Charge transfer acquisition sequence managed by software
- 1: Charge transfer acquisition sequence managed by hardware

Bit 1 **TIF**: *Trigger interrupt flag*

This flag is set by hardware when a trigger event occurs caused by  $V_{IH}$  voltage being reached on the sampling capacitor I/O and only if the corresponding I/O mask bit is not set. It is cleared by software writing “1”. If the bit TIE is set, an interrupt is generated.

- 0: No trigger event occurred
- 1: Trigger event pending

Bit 0 **TIE**: *Trigger interrupt enable*

This bit is set and cleared by software. It enables the generation of an interrupt on trigger event.

- 0: Trigger interrupt disabled
- 1: Trigger interrupt enabled

*Note:* This register is only available on low density devices.

### 11.4.17 IO mask register 1 (RI\_IOMR1)

Address offset: 0x21

Reset value: 0x00

7	6	5	4	3	2	1	0
CH22M	CH19M	CH16M	CH13M	CH10M	CH7M	CH4M	CH1M
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CHxM: I/O mask**

These bits are set and cleared by software to mask events from an I/O or to avoid triggering multiple events from an I/O.

0: Event from the I/O is allowed

1: Event from the I/O is masked

*Note:* This register controls the first I/O of each group of I/Os and is only available on low density devices.

### 11.4.18 IO mask register 2 (RI\_IOMR2)

Address offset: 0x22

Reset value: 0x00

7	6	5	4	3	2	1	0
CH23M	CH20M	CH17M	CH14M	CH11M	CH8M	CH5M	CH2M
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CHxM: I/O mask**

These bits are set and cleared by software to mask event from an I/O or to avoid triggering multiple events from an I/O.

0: Event from the I/O is allowed

1: Event from the I/O is masked

*Note:* This register controls the second I/O of each group of I/Os and is only available on low density devices.

### 11.4.19 IO mask register 3 (RI\_IOMR3)

Address offset: 0x23

Reset value: 0x00

7	6	5	4	3	2	1	0
CH24M	CH21M	CH18M	CH15M	CH12M	CH9M	CH6M	CH3M
rw	rw	rw	rw	rw	rw	rw	rw

#### Bits 7:0 CHxM: I/O mask

These bits are set and cleared by software to mask events from an I/O or to avoid triggering multiple events from an I/O.

0: Event from the I/O is allowed

1: Event from the I/O is masked

*Note:* This register controls the first I/O of each group of I/Os and is only available on low density devices.

### 11.4.20 IO mask register 4 (RI\_IOMR4)

Address offset: 0x24

Reset value: 0x00

7	6	5	4	3	2	1	0
CH28M	CH27M	Reserved			CH26M	CH29M	
rw	rw				rw	rw	

#### Bits 7:6 CHxM: I/O mask

These bits are set and cleared by software to mask events from an I/O or to avoid triggering multiple events from an I/O.

0: Event from the I/O is allowed

1: Event from the I/O is masked

Bits 5:2 Reserved, must be kept cleared.

#### Bits 1:0 CHxM: I/O mask

These bits are set and cleared by software to mask events from an I/O or to avoid triggering multiple events from an I/O.

0: Event from the I/O is allowed

1: Event from the I/O is masked

*Note:* This register controls the first I/O of each group of I/Os. and is only available on low density devices.

### 11.4.21 I/O input register 4 (RI\_IOIR4)

Address offset: 0x25

Reset value: 0xXX where X is undefined

7	6	5	4	3	2	1	0
CH28I	CH27I			Reserved		CH26I	CH29I
r	r					r	r

Bits 7:6 **CHxI**: I/O pin input value

These bits return the corresponding I/O pin value irrespective of the port configuration (similar to the Px\_IDR register). This register is read-only.

0: Low logic level

1: High logic level

Bits 5:2 Reserved, must be kept cleared.

Bits 1:0 **CHxI**: I/O pin input value

These bits return the corresponding I/O pin value irrespective of the port configuration (similar to the Px\_IDR register). This register is read-only.

0: Low logic level

1: High logic level

**Note:** *This register controls the fourth I/O of each group of I/Os and this register is only available on low density devices.*

### 11.4.22 I/O control mode register 4 (RI\_IOCMR4)

Address offset: 0x26

Reset value: 0x00

7	6	5	4	3	2	1	0
CH28M	CH27M			Reserved		CH26M	CH29M
rw	rw					rw	rw

Bits 7:6 **CHxM**: I/O control mode

These bits are set and cleared by software to select how the I/O is controlled.

0: I/O x is controlled by the standard GPIO registers. The I/O x switch is directly controlled by the RI\_IOSR2 register.

1: I/O x is set to protected mode and is only controlled by the RI\_IOSR2 and RI\_IOPCR registers (standard GPIO registers have no effect on the configuration of I/O x).

Bits 5:2 Reserved, must be kept cleared.

Bits 1:0 **CHxM**: I/O control mode

These bits are set and cleared by software to select how the I/O is controlled.

0: I/O x is controlled by the standard GPIO registers. The I/O x switch is directly controlled by the RI\_IOSR2 register.

1: I/O x is set to protected mode and is only controlled by the RI\_IOSR2 and RI\_IOPCR registers (standard GPIO registers have no effect on the configuration of I/O x).

**Note:** *This register controls the fourth I/O of each group of I/Os and is only available on low density devices.*

### 11.4.23 I/O switch register 4 (RI\_IOSR4)

Address offset: 0x27

Reset value: 0x00

7	6	5	4	3	2	1	0
CH28E	CH27E			Reserved		CH26E	CH29E
rw	rw					rw	rw

Bits 7:6 **CHxE: I/O switch control**

**If the corresponding RI\_IOCMRx bit is cleared:**

These bits are set and cleared by software to open and close the I/O switches. If the I/O is used for the ADC1 input, the switch must be left open to allow the ADC1 to control it.

0: I/O x switch is open (I/O x is controlled by the GPIO registers or ADC1).

1: I/O x switch is closed (analog channel enabled).

**If the corresponding RI\_IOCMRx bit is set and the AM bit is cleared in the RI\_CR register (software management of the charge transfer acquisition sequence):**

These bits are set and cleared by software. They define the configuration of the I/O.

0: I/O x is forced to output push-pull low mode.

1: I/O x configuration is determined by bit 6 and bit 7 in the RI\_IOGCR register.

**If the corresponding RI\_IOCMRx bit is set and the AM bit is also set in the RI\_CR register (hardware management of the charge transfer acquisition sequence):**

These bits are set and cleared by software. They define if the corresponding I/O x state is controlled by the TIM2 timer.

0: I/O x is forced to output push-pull low mode

1: I/O x configuration determined by the timer TIM2

Bits 5:2 Reserved, must be kept cleared.

Bits 1:0 **CHxE: I/O switch control**

**If the corresponding RI\_IOCMRx bit is cleared:**

These bits are set and cleared by software to open and close the I/O switches. If the I/O is used for the ADC1 input, the switch must be left open to allow the ADC1 to control it.

0: I/O x switch is open (I/O x is controlled by the GPIO registers or ADC1).

1: I/O x switch is closed (analog channel enabled).

**If the corresponding RI\_IOCMRx bit is set AND the AM bit is cleared in the RI\_CR register (software management of the charge transfer acquisition sequence):**

These bits are set and cleared by software. They define the configuration of the I/O.

0: I/O x is forced to output push-pull low mode.

1: I/O x configuration is determined by bit 6 and bit 7 in the RI\_IOGCR register.

**If the corresponding RI\_IOCMRx bit is set and the AM is also set in the RI\_CR register (hardware management of the charge transfer acquisition sequence):**

These bits are set and cleared by software. They define if the corresponding I/O x state is controlled by the timer TIM2.

0: I/O x is forced to output push-pull low mode

1: I/O x configuration determined by the timer TIM2

*Note: This register controls the fourth I/O of each group of I/Os.*

*Note: This register controls the fourth I/O of each group of I/Os and is only available on low density devices.*

### 11.4.24 RI register map and reset values

**Table 30.** Routing interface register map

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	Reserved Reset value	-0	-0	-0	-0	-0	-0	-0	-0
0x01	RI_ICR1 <sup>(1)</sup> Reset value	-0	-0	-0	IC2CS40	IC2CS30	IC2CS20	IC2CS10	IC2CS00
0x02	RI_ICR2 <sup>(1)</sup> Reset value	-0	-0	-0	IC3CS40	IC3CS30	IC3CS20	IC3CS10	IC3CS00
0x03	RI_IOIR1 Reset value	CH22Ix	CH19Ix	CH16Ix	CH13Ix	CH10Ix	CH7Ix	CH4Ix	CH1Ix
0x04	RI_IOIR2 Reset value	CH23Ix	CH20Ix	CH17Ix	CH14Ix	CH11Ix	CH8Ix	CH5Ix	CH2Ix
0x05	RI_IOIR3 Reset value	CH24Ix	CH21Ix	CH18Ix	CH15Ix	CH12Ix	CH9Ix	CH6Ix	CH3Ix
0x06	RI_IOCMR1 Reset value	CH22M0	CH19M0	CH16M0	CH13M0	CH10M0	CH7M0	CH4M0	CH1M0
0x07	RI_IOCMR2 Reset value	CH23M0	CH20M0	CH17M0	CH14M0	CH11M0	CH8M0	CH5M0	CH2M0
0x08	RI_IOCMR3 Reset value	CH24M0	CH21M0	CH18M0	CH15M0	CH12M0	CH9M0	CH6M0	CH3M0
0x09	RI_IOSR1 Reset value	CH22E0	CH19E0	CH16E0	CH13E0	CH10E0	CH7E0	CH4E0	CH1E0
0x0A	RI_IOSR2 Reset value	CH23E0	CH20E0	CH17E0	CH14E0	CH11E0	CH8E0	CH5E0	CH2E0
0x0B	RI_IOSR3 Reset value	CH24E0	CH21E0	CH18E0	CH15E0	CH12E0	CH9E0	CH6E0	CH3E0
0x0C	RI_IOGCR Reset value	IOM41 <sup>(2)</sup> 1 <sup>(2)</sup>	IOM40 <sup>(2)</sup> 1 <sup>(2)</sup>	IOM311	IOM301	IOM211	IOM201	IOM111	IOM101
0x0D	RI_ASCR1 Reset value	AS70	AS60	AS50	AS40	AS30	AS20	AS10	AS00
0x0E	RI_ASCR2 Reset value	-0	AS140	-0	-0	AS11 <sup>(3)</sup> 0	AS10 <sup>(3)</sup> 0	AS9 <sup>(3)</sup> 0	AS80
0x0F	RI_RCR Reset value	-0	-0	-0	-0	400KPD0	10KPD0	400KPU0	10KPU0
0x20	RI_CR <sup>(4)</sup> Reset value	-0	-0	-0	-0	THALT0	AM0	TIF0	TIE0
0x21	RI_IOMR1 <sup>(4)</sup> Reset value	CH22M0	CH19M0	CH16M0	CH13M0	CH10M0	CH7M0	CH4M0	CH1M0
0x22	RI_IOMR2 <sup>(4)</sup> Reset value	CH23M0	CH20M0	CH17M0	CH14M0	CH11M0	CH8M0	CH5M0	CH2M0
0x23	RI_IOMR3 <sup>(4)</sup> Reset value	CH24M0	CH21M0	CH18M0	CH15M0	CH12M0	CH9M0	CH6M0	CH3M0
0x24	RI_IOMR4 <sup>(4)</sup> Reset value	CH28M0	CH27M0	-0	-0	-0	-0	CH26M0	CH29M0
0x25	RI_IOIR4 <sup>(4)</sup> Reset value	CH28Ix	CH27Ix	-x	-x	-x	-x	CH26Ix	CH29Ix

**Table 30. Routing interface register map (continued)**

<b>Address offset</b>	<b>Register name</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
0x26	RI_IOCMR4 <sup>(4)</sup> Reset value	CH28M 0	CH27M 0	- 0	- 0	- 0	- 0	CH26M 0	CH29M 0
0x27	RI_IOSR4 <sup>(4)</sup> Reset value	CH28E 0	CH27E 0	- 0	- 0	- 0	- 0	CH26E 0	CH29E 0

1. On medium, medium+ and high density devices only.
2. These bits are available on low density devices only. Reset values are 0 for the other devices.
3. On medium+ and high density devices only.
4. On low density devices only

## 11.5 SYSCFG registers

### 11.5.1 SYSCFG remap control register 1 (SYSCFG\_RMPCR1)

Address offset: 0x01

Reset value: 0x0C

7	6	5	4	3	2	1	0
SPI1_REMAP1	USART1CK_REMAP	USART1TR_REMAP[1:0]		TIM4DMA_REMAP[1:0]		ADC1DMA_REMAP[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **SPI1\_REMAP1**: SPI1 remapping

When this bit is reset:

- SPI1\_MISO is mapped on PB7
- SPI1\_MOSI is mapped on PB6
- SPI1\_SCK is mapped on PB5
- SPI1\_NSS is mapped on PB4

When this bit is set:

- SPI1\_MISO is mapped on PA2
- SPI1\_MOSI is mapped on PA3
- SPI1\_SCK is mapped on PC6
- SPI1\_NSS is mapped on PC5

Bit 6 **USART1CK\_REMAP**: USART1\_CK remapping

0: USART1\_CK mapped on PC4

1: USART1\_CK mapped on PA0

Bits 5:4 **USART1TR\_REMAP[1:0]**: USART1\_TX and USART1\_RX remapping

00: USART1\_TX on PC3 and USART1\_RX on PC2

01: USART1\_TX on PA2 and USART1\_RX on PA3

10: USART1\_TX on PC5 and USART1\_RX on PC6

11: Reserved

Bits 3:2 **TIM4DMA\_REMAP[1:0]**: TIM4 DMA channel remapping

00: TIM4 DMA request/acknowledge mapped on DMA1 channel 0

01: TIM4 DMA request/acknowledge mapped on DMA1 channel 1

10: TIM4 DMA request/acknowledge mapped on DMA1 channel 2

11: TIM4 DMA request/acknowledge mapped on DMA1 channel 3

Bits 1:0 **ADC1DMA\_REMAP[1:0]**: ADC1 DMA channel remapping

00: ADC1 DMA request/acknowledge mapped on DMA1 channel 0

01: ADC1 DMA request/acknowledge mapped on DMA1 channel 1

10: ADC1 DMA request/acknowledge mapped on DMA1 channel 2

11: ADC1 DMA request/acknowledge mapped on DMA1 channel 3

### 11.5.2 SYSCFG remap control register 2 (SYSCFG\_RMPCR2)

Address offset: 0x02

Reset value: 0x00

7	6	5	4	3	2	1	0
TIM23BKIN_REMAP	TIM3TRIG_REMAP2	SPI2_REMAP	TIM3TRIGLSE_REMAP	TIM2TRIGLSE_REMAP	TIM3TRIG_REMAP1	TIM2TRIG_REMAP	ADC1TRIG_REMAP
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7<sup>(1)</sup> **TIM23BKIN\_REMAP**: TIM2 break input and TIM3 break input remapping

0: TIM2\_BKIN & TIM3\_BKIN mapped on PA4 & PA5

1: TIM2\_BKIN & TIM3\_BKIN mapped on PG0 & PG1 (only on 64- and 80-pin packages)

Bit 6<sup>(1)</sup> **TIM3TRIG\_REMAP2**: TIM3 trigger remapping

0: TIM3\_TRIG mapped on PD1 if TRIM3TRIG\_REMAP1=0 or on PA5 if TRIM3TRIG\_REMAP1=1

1: TIM3\_TRIG mapped on PG3 (only on 64- and 80-pin packages)

Bit 5<sup>(1)</sup> **SPI2\_REMAP**: SPI2 remapping

When this bit is reset:

- SPI2\_MISO is mapped on PG7
- SPI2\_MOSI is mapped on PG6
- SPI2\_SCK is mapped on PG5
- SPI2\_NSS is mapped on PG4

When this bit is set:

- SPI2\_MISO is mapped on PI3
- SPI2\_MOSI is mapped on PI2
- SPI2\_SCK is mapped on PI1
- SPI2\_NSS is mapped on PI0

Bit 4 **TIM3TRIGLSE\_REMAP**: TIM3 trigger controlled by LSE

TIM3 trigger controlled by the LSE oscillator.

0: TIM3\_TRIG mapped as defined in the TIM3TRIG\_REMAP1 bit

1: TIM3\_TRIG mapped on OSC32\_IN (LSE oscillator input)

*Note: When the TIM2TRIGLSE\_REMAP or TIM3TRIGLSE\_REMAP are set, the ETR prescaler of the corresponding timer must be set to have a ratio of at least 4 with respect to the selected system clock.*

*LSE needs to be properly switched on as defined in the CLK configuration registers.*

Bit 3 **TIM2TRIGLSE\_REMAP**: TIM2 trigger controlled by LSE

TIM2 trigger controlled by the LSE oscillator.

0: TIM2\_TRIG mapped as defined in the TIM2TRIG\_REMAP bit

1: TIM2\_TRIG mapped on OSC32\_IN (LSE oscillator input)

*Note: When the TIM2TRIGLSE\_REMAP or TIM3TRIGLSE\_REMAP are set, the ETR prescaler of the corresponding timer must be set to have a ratio of at least 4 with respect to the selected system clock.*

*LSE needs to be properly switched on as defined in the CLK configuration registers.*

Bit 2 **TIM3TRIG\_REMAP1**: TIM3 trigger remapping

0: TIM3\_TRIG mapped on PD1

1: TIM3\_TRIG mapped on PA5

Bit 1 **TIM2TRIG\_REMAP**: TIM2 trigger remapping

- 0: TIM2\_TRIG mapped on PB3
- 1: TIM2\_TRIG mapped on PA4

Bit 0 **ADC1TRIG\_REMAP**: ADC1 trigger remapping

- 0: ADC1\_TRIG mapped on PA6
- 1: ADC1\_TRIG mapped on PD0

1. These bits are reserved in low and medium density devices.

### 11.5.3 SYSCFG remap control register 3 (SYSCFG\_RMPCR3)

Address offset: 0x00

Reset value: 0x00

**Note:** *This register is available in low density devices (with 20-pin packages), medium+ and high density devices only.*

	7	6	5	4	3	2	1	0
Reserved		CCO_REMAP	TIM3_CH2_REMAP	TIM3_CH1_REMAP	USART3CK_REMAP	USART3TR_REMAP	SPI1_REMAP2	
	rw	rw	rw	rw	rw	rw	rw	

Bits 7:6 Reserved (medium+ and high density devices only)

Bit 7 **TIM2\_CH2** TIM2 channel 2 remapping (20-pin package low density devices only)

- 0: TIM2\_CH2 is mapped on PB2
- 1: TIM2\_CH2 is mapped on PC6

Bit 6 **TIM2\_CH1** TIM2 channel 1 remapping (20-pin package low density devices only)

- 0: TIM2\_CH1 is mapped on PB0
- 1: TIM2\_CH1 is mapped on PC5

Bit 5 **CCO\_REMAP**: Configurable clock output remapping

- 0: CCO mapped on PC4
- 1: CCO mapped on PE2

Bit 4 **TIM3\_CH2\_REMAP**: TIM3 channel 2 remapping

- 0: TIM3\_CH2 mapped on PD0
- 1: TIM3\_CH2 mapped on PI3 (80-pin packages only)

Bit 3 **TIM3\_CH1\_REMAP**: TIM3 channel 1 remapping

- 0: TIM3\_CH1 mapped on PB1
- 1: TIM3\_CH1 mapped on PI0 (80-pin packages only)

Bit 2 **USART3CK\_REMAP**: USART3\_CK remapping

- 0: USART3\_CK mapped on PG2
- 1: USART3\_CK mapped on PF2 (80-pin packages only)

Bit 1 **USART3TR\_REMAP**: USART3\_TX and USART3\_RX remapping

- 0: USART3\_TX mapped on PG1 and USART3\_RX mapped on PG0
- 1: USART3\_TX mapped on PF0 and USART3\_RX mapped on PF1

Bit 0 **SPI1\_REMAP2**: SPI1 remapping (80-pin packages only)

When this bit is reset:

- SPI1\_MISO is mapped on PB7
- SPI1\_MOSI is mapped on PB6
- SPI1\_SCK is mapped on PB5
- SPI1\_NSS is mapped on PB4

When this bit is set:

- SPI1\_MISO is mapped on PF0
- SPI1\_MOSI is mapped on PF1
- SPI1\_SCK is mapped on PF2
- SPI1\_NSS is mapped on PF3

#### 11.5.4 SYSCFG register map and reset values

**Table 31. Register map**

Offset address	Register name	7	6	5	4	3	2	1	0
0x00	SYSCFG_RMPCR3 <sup>(2)</sup> Reset value	Reserved 000 <sup>(1)</sup>		CCO_REMAP_0	TIM3_CH2_REMAP_0	TIM3_CH1_REMAP_0	USART3CK_REMAP_0	USART3TRREMAPP_0	SPI1_REMAP2_0
0x01	SYSCFG_RMPCR1 Reset value	SPI1_REMAP1_0	USART1CK_REMAP_0	USART1TR_REMAP[1:0] 00	TIM4DMA_REMAP[1:0] 11			ADC1DMA_REMAP[1:0] 00	
0x02	SYSCFG_RMPCR2 Reset value	TIM23BKIN_REMAP <sup>(2)</sup> _0	TIM3TRIG_REMAP2 <sup>(2)</sup> _0	SPI2_REMAP <sup>(2)</sup> _0	TIM3TRIGLSE_REMAP_0	TIM2TRIGLSE_REMAP_0	TIM3TRIG_REMAP1_0	TIM2TRIG_REMAP_0	ADC1TRIG_REMAP_0

1. On low density devices, bit 6 = TIM2\_CH1\_REMAP and bit 7 = TIM2\_CH2\_REMAP.

2. On low, medium+ and high density devices only.

## 12 Interrupt controller (ITC)

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

### 12.1 ITC introduction

- Management of hardware interrupts
  - External interrupt capability on all I/O pins with dedicated interrupt vector per port and dedicated flag per pin
  - Peripheral interrupt capability
- Management of software interrupt (TRAP)
- Nested or concurrent interrupt management with flexible interrupt priority and level management:
  - Up to 4 software programmable nesting levels
  - Up to 32 interrupt vectors fixed by hardware
  - 2 non maskable events: RESET, TRAP
- Up to 40 external interrupt sources on 11 vectors

This interrupt management is based on:

- Bit I1 and I0 of the CPU Condition Code register (CCR)
- Software priority registers (ITC\_SPRx)
- Reset vector located at 0x00 8000 at the beginning of program memory. The Reset initialization routine is programmed in ROM by STMicroelectronics.
- Fixed interrupt vector addresses located at the high addresses of the memory map (0x00 8004 to 0x00 807C) sorted by hardware priority order.

### 12.2 Interrupt masking and processing flow

The interrupt masking is managed by bits I1 and I0 of the CCR register and by the ITC\_SPRx registers which set the software priority level of each interrupt vector (see [Table 32](#)). The processing flow is shown in [Figure 28](#).

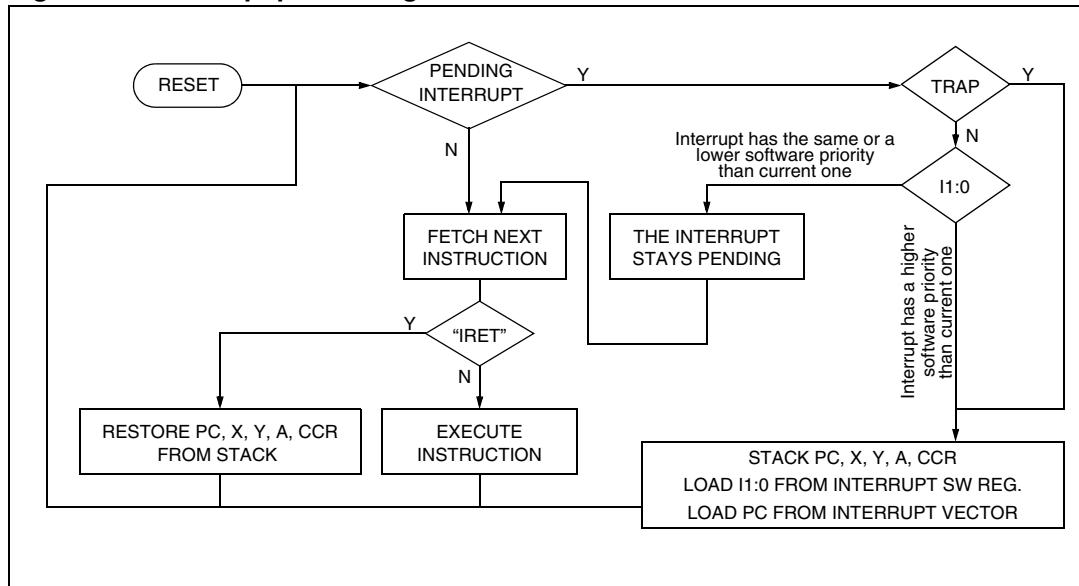
When an interrupt request has to be serviced:

1. Normal processing is suspended at the end of the current instruction execution.
2. The PC, X,Y, A and CCR registers are saved onto the stack.
3. Bits I1 and I0 of CCR register are set according to the values in the ITC\_SPRx registers corresponding to the serviced interrupt vector.
4. 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 .

The interrupt service routine should end with the IRET instruction which causes the content of the saved registers to be recovered from the stack. As a consequence of the IRET instruction, bits I1 and I0 are restored from the stack and the program execution resumes.

**Table 32. Software priority levels**

Software priority	Level	I1	I0
Level 0 (main)	Low ↓ High	1	0
Level 1		0	1
Level 2		0	0
Level 3 (= software priority disabled)		1	1

**Figure 28. Interrupt processing flowchart**

**Caution:** If the interrupt mask bits I0 and I1 are set within an interrupt service routine (ISR) with the instruction SIM, removal of the interrupt mask with RIM causes the software priority to be set to level 0.

To restore the correct priority when disabling and enabling interrupts inside an ISR, follow the procedures presented in [Table 32](#) for disabling and enabling the interrupts.

### 12.2.1 Servicing pending interrupts

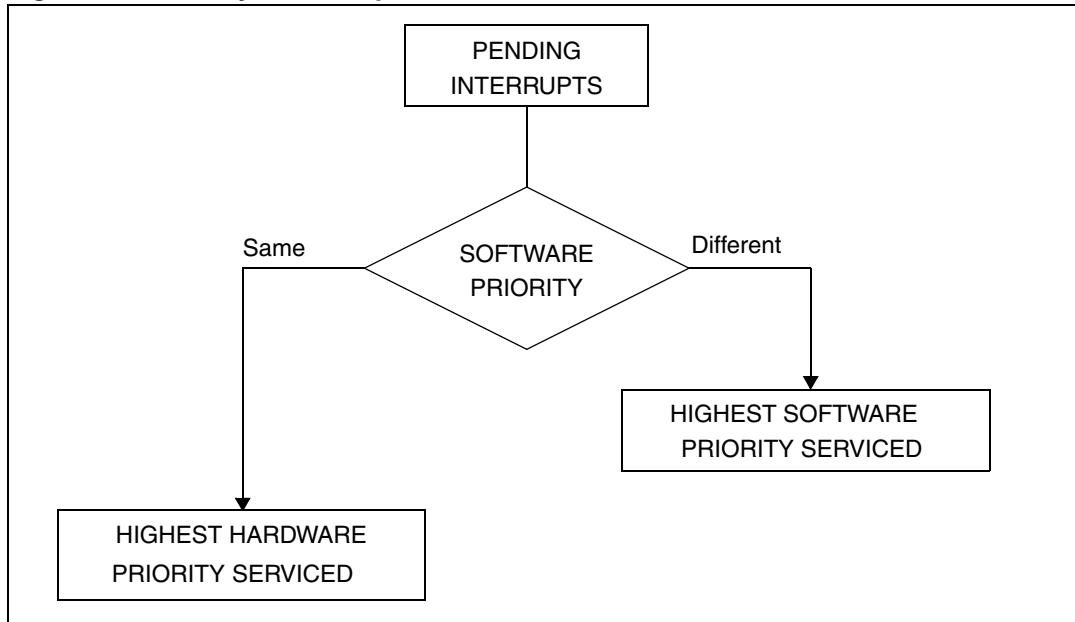
Several interrupts can be pending at the same time. The interrupt to be taken into account is determined by the following two-step process:

1. The highest software priority interrupt is serviced.
2. If several interrupts have the same software priority then the interrupt with the highest hardware priority is serviced first.

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.*
  - 2 *RESET and TRAP are considered as having the highest software priority in the decision process.*

See [Figure 29](#) for a description of pending interrupt servicing process.

**Figure 29. Priority decision process**

## 12.2.2 Interrupt sources

Two interrupt source types are managed by the STM8 interrupt controller:

- Non-maskable interrupts: RESET and TRAP
- Maskable interrupts: external interrupts or interrupts issued by internal peripherals

### Non-maskable interrupt sources

Non-maskable interrupt sources are processed regardless of the state of bits I1 and I0 of the CCR register (see [Figure 28](#)). PC, X, Y, A and CCR registers are stacked only when a TRAP interrupt occurs. The corresponding vector is then loaded in the PC register and bits I1 and I0 of the CCR register are set to disable interrupts (level 3).

- TRAP (non-maskable software interrupt)

This software interrupt source is serviced when the TRAP instruction is executed. It is serviced according to the flowchart shown in [Figure 28](#).

A TRAP interrupt does not allow the processor to exit from Halt mode.

- RESET

The RESET interrupt source has the highest STM8 software and hardware priorities. This means that all the interrupts are disabled at the beginning of the reset routine. They must be re-enabled by the RIM instruction (see [Table 35: Dedicated interrupt instruction set](#)).

A RESET interrupt allows the processor to exit from Halt mode.

See RESET chapter for more details on RESET interrupt management.

### Maskable interrupt sources

Maskable interrupt vector sources are serviced if the corresponding interrupt is enabled and if its own interrupt software priority in ITC\_SPRx registers is higher than the one currently being serviced (I1 and I0 in CCR register). If one of these two conditions is not met, the interrupt is latched and remains pending.

- External interrupts

External interrupts can be used to wake up the MCU from Halt mode. The device sensitivity to external interrupts can be selected by software through the External Interrupt Control registers (EXTI\_CRx).

When several input pins connected to the same interrupt line are selected simultaneously, they are logically ORed.

When external level-triggered interrupts are latched, if the given level is still present at the end of the interrupt routine, the interrupt remains activated except if it has been inactivated in the routine.

- Peripheral interrupts

A few peripheral interrupts cause the MCU to wake up from Halt mode. See the interrupt vector table in the datasheet.

A peripheral interrupt occurs when a specific flag is set in the peripheral status register and the corresponding enable bit is set in the peripheral control register.

The standard sequence for clearing a peripheral interrupt performs an access to the status register followed by a read or write to an associated register. The clearing sequence resets the internal latch. A pending interrupt (that is an interrupt waiting to be serviced) is therefore lost when the clear sequence is executed.

## 12.3 Interrupts and low power modes

All interrupts allow the processor to exit from Wait mode.

Only a Reset or an event allows the processor to exit from Low power wait mode. This mode is entered by executing a WFE instruction in Low power run mode. The wakeup by an event makes the system go back to Low power run mode (refer to [for more details](#)).

Only external and other specific interrupts allow the processor to exit from Halt and Active-halt mode (see wakeup from halt and wakeup from Active-halt in the interrupt vector table in the datasheet).

When several pending interrupts are present while waking up from Halt mode, the first interrupt serviced can only be an interrupt with exit-from-Halt mode capability. It is selected through the decision process shown in [Figure 29](#). If the highest priority pending interrupt cannot wake up the device from Halt mode, it will be serviced next.

If any internal or external interrupt (from a timer for example) occurs while the HALT instruction is executing, the HALT instruction is completed but the interrupt invokes the wakeup process immediately after the HALT instruction has finished executing. In this case the MCU is actually waking up from Halt mode to Run mode, with the corresponding delay of  $t_{WUH}$  as specified in the datasheet.

Before executing the HALT instruction, the user program must clear any pending peripheral interrupts (clear interrupt pending bit in the corresponding peripheral configuration registers). Otherwise, the HALT instruction is not executed and program execution continues.

## 12.4 Activation level/low power mode control

The MCU activation level is configured by programming the AL bit in the CFG\_GCR register (see global configuration register (CFG\_GCR)).

This bit is used to control the low power modes of the MCU. In very low power applications, the MCU spends most of the time in WFI/Halt mode and is woken up (through interrupts) at specific moments in order to execute a specific task. Some of these recurring tasks are short enough to be treated directly in an ISR (interrupt service routine), rather than going back to the main program. To cover this case, you can set the AL bit before entering Low power mode (by executing WFI/HALT instruction). Consequently, the interrupt routine causes the device to return to low power mode., then the interrupt routine returns directly to Low power mode. The run time/ISR execution is reduced due to the fact that the register context is saved only on the first interrupt.

As a consequence, all the operations can be executed in ISR in very simple applications. In more complex ones, an interrupt routine may relaunch the main program by simply resetting the AL bit.

For example, an application may need to be woken up by the auto-wakeup unit (AWU) every 50 ms in order to check the status of some pins/sensors/push-buttons. Most of the time, as these pins are not active, the MCU can return to Low power mode without running the main program. If one of these pins is active, the ISR decides to launch the main program by resetting the AL bit.

## 12.5 Concurrent and nested interrupt management

STM8 devices feature two interrupt management modes:

- Concurrent mode
- Nested mode

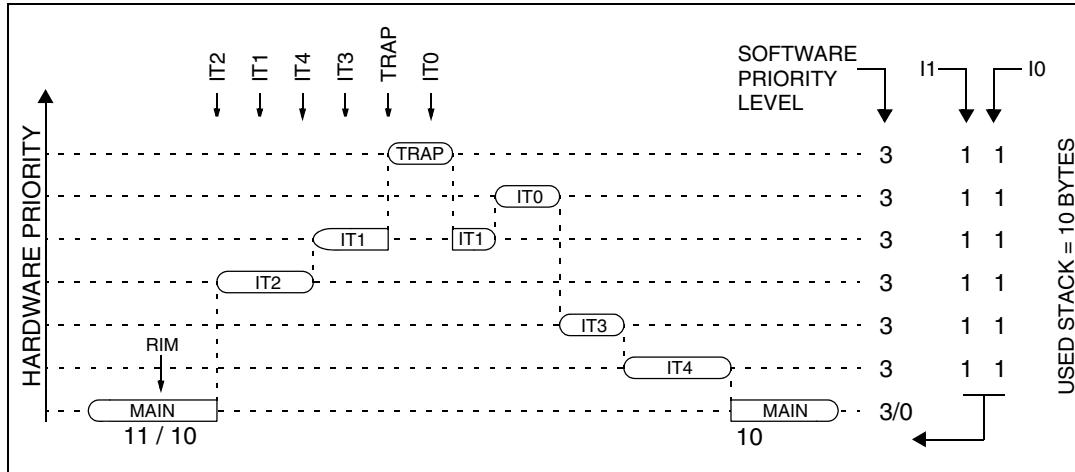
### 12.5.1 Concurrent interrupt management mode

In this mode, all interrupts are interrupt priority level 3 so that none of them can be interrupted, except by a RESET, or TRAP.

The hardware priority is given in the following order from the lowest to the highest priority, that is: MAIN, IT4, IT3, IT2, IT1, IT0, TRAP (same priority), and RESET.

*Figure 30* shows an example of concurrent interrupt management mode.

**Figure 30. Concurrent interrupt management**



## 12.5.2 Nested interrupt management mode

In this mode, interrupts are allowed during interrupt routines. This mode is activated as soon as an interrupt priority level lower than level 3 is set.

The hardware priority is given in the following order from the lowest to the highest priority, that is: MAIN, IT4, IT3, IT2, IT1, IT0, and TRAP.

The software priority is configured for each interrupt vector by setting the corresponding I1\_x and I0\_x bits of the ITC\_SPRx register. I1\_x and I0\_x bits have the same meaning as I1 and I0 bits of the CCR register (see *Table 33*).

Level 0 can not be programmed (I1\_x=1, I0\_x=0). In this case, the previously stored value is kept. For example: if previous value is 0xCF, and programmed value equals 64h, the result is 44h.

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

**Caution:** If bits I1\_x and I0\_x are modified while the interrupt x is executed, the device operates as follows: if the interrupt x is still pending (new interrupt or flag not cleared) and the new software priority is higher than the previous one, then the interrupt x is re-entered. Otherwise, the software priority remains unchanged till the next interrupt request (after the IRET of the interrupt x).

During the execution of an interrupt routine, the HALT, POPCC, RIM, SIM and WFI instructions change the current software priority till the next IRET instruction or one of the previously mentioned instructions is issued. See *Section 12.7* for the list of dedicated interrupt instructions.

*Figure 31* shows an example of nested interrupt management mode.

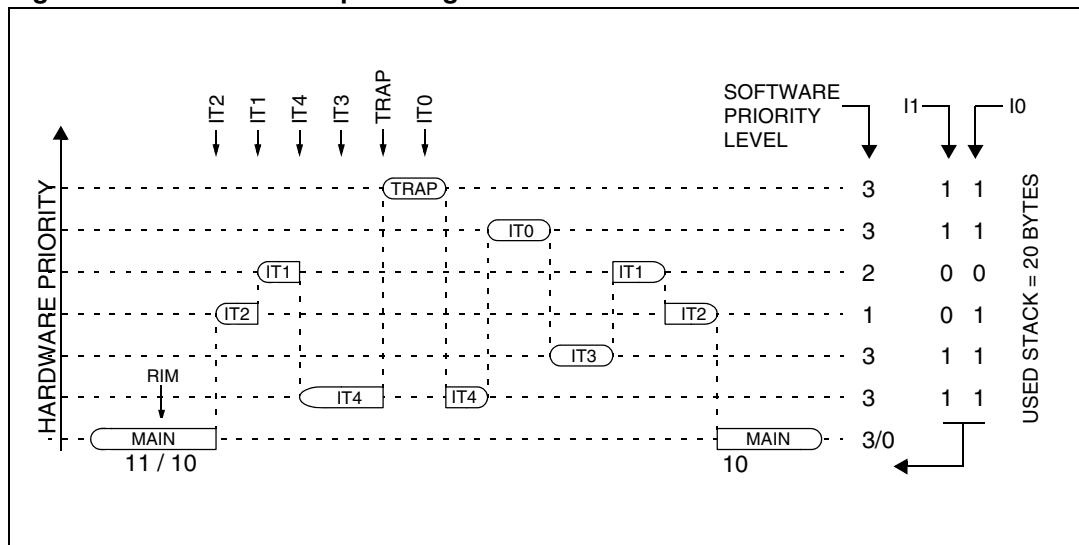
---

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

---

**Table 33.** Vector address map versus software priority bits

Vector address	ITC_SPRx bits
0x00 8008h	I1_0 and I0_0 bits
0x00 800Ch	I1_1 and I0_1 bits
...	...
0x00 807Ch	I1_29 and I0_29 bits

**Figure 31.** Nested interrupt management

## 12.6 External interrupts

Eleven interrupt vectors are dedicated to external Interrupt events:

- EXTIB/G - 8 lines on Port B or G: PB[7:0] or PG[7:0]
- EXTD/H - 8 lines on Port D or H: PD[7:0] or PH[7:0]
- EXTE/F - 8 lines on Port E or F: PE[7:0] or PF[7:0]
- EXTI0 - 6 lines on Port A/B/C/D/E/F, bit 0: Px[0]
- EXTI1 - 5 lines on Port A/B/C/D/E, bit 1: Px[1]
- EXTI2 - 5 lines on Port A/B/C/D/E, bit 2: Px[2]
- EXTI3 - 5 lines on Port A/B/C/D/E, bit 3: Px[3]
- EXTI4 - 5 lines on Port A/B/C/D/E, bit 4: Px[4]
- EXTI5 - 5 lines on Port A/B/C/D/E, bit 5: Px[5]
- EXTI6 - 5 lines on Port A/B/C/D/E, bit 6: Px[6]
- EXTI7 - 5 lines on Port A/B/C/D/E, bit 7: Px[7]

Note:

*Ports G, H and I are available on medium+ and high density devices only.*

To generate an interrupt, the corresponding GPIO port must be configured in input mode with interrupts enabled. Refer to the register description in the GPIO chapter for details.

When an external interrupt occurs, the corresponding bit is set in the EXTI\_SRx status register. This indicates a pending interrupt. Clearing this bit, writing a 1 in it, clears the corresponding pending external interrupt.

The interrupt sensitivity must be configured in the external interrupt control register 1 (EXTI\_CR1), external interrupt control register 2 (EXTI\_CR2), and external interrupt control register 3 (EXTI\_CR3) (see [Section 12.9.5](#), [Section 12.9.7](#) and [Section 12.9.8](#)).

**Table 34. External interrupt sensitivity**

GPIO port	Interrupt sensitivity	Configuration register
EXTI0 to EXTI3 on port A, B, C, D, E and F	Falling edge and low level	EXTI_CR1
EXTI4 to EXTI7 of port A, B, C, D and E	Rising edge only Falling edge only	EXTI_CR2
EXTIE and EXITF	Rising and falling edge	EXTI_CR3

## 12.7 Interrupt instructions

[Table 35](#) shows the interrupt instructions.

**Table 35. 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 CCR, A, X, Y, 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 CCR from the stack	Memory => CCR	I1	H	I0	N	Z	C
PUSH CC	Push CC on the stack	CC =>Memory						
RIM	Enable interrupt (level 0 set)	Load 10 in I1:0 of CCR	1		0			
SIM	Disable interrupt (level 3 set)	Load 11 in I1:0 of CCR	1		1			
TRAP	Software trap	Software NMI	1		1			
WFI	Wait for interrupt		1		0			
WFE	Wait for event		1		0			

## 12.8 Interrupt mapping

Refer to the corresponding device datasheet for the table of interrupt vector addresses.

## 12.9 ITC and EXTI registers

### 12.9.1 CPU condition code register interrupt bits (CCR)

Address: refer to the general hardware register map table in the datasheet.

Reset value: 0x28

7	6	5	4	3	2	1	0
V	-	I1	H	I0	N	Z	C
r	r	rw	r	rw	r	r	r

Bits 5, 3<sup>(1)</sup> **I[1:0]**: Software interrupt priority bits<sup>(2)</sup>

These two bits indicate the software priority of the current interrupt request. When an interrupt request occurs, the software priority of the corresponding vector is loaded automatically from the software priority registers (ITC\_SPRx).

The I[1:0] bits can be also set/cleared by software using the RIM, SIM, HALT, WFI, IRET or PUSH/POP instructions (see [Figure 31: Nested interrupt management](#)).

I1	I0	Priority	Level
1	0	Level 0 (main)	↓ Low High
0	1	Level 1	
0	0	Level 2	
1	1	Level 3 (= software priority disabled*)	

1. Refer to the central processing section for details on the other CCR bits.

2. TRAP and RESET events can interrupt a level-3 program.

## 12.9.2 Software priority register x (ITC\_SPRx)

Address offset: 0x00 to 0x07

Reset value: 0xFF

	7	6	5	4	3	2	1	0
ITC_SPR1	VECT3SPR[1:0]		VECT2SPR[1:0]		VECT1SPR[1:0]		VECT0SPR[1:0]	
ITC_SPR2	VECT7SPR[1:0]		VECT6SPR[1:0]		VECT5SPR[1:0]		VECT4SPR[1:0]	
ITC_SPR3	VECT11SPR[1:0]		VECT10SPR[1:0]		VECT9SPR[1:0]		VECT8SPR[1:0]	
ITC_SPR4	VECT15SPR[1:0]		VECT14SPR[1:0]		VECT13SPR[1:0]		VECT12SPR[1:0]	
ITC_SPR5	VECT19SPR[1:0]		VECT18SPR[1:0]		VECT17SPR[1:0]		VECT16SPR[1:0]	
ITC_SPR6	VECT23SPR[1:0]		VECT22SPR[1:0]		VECT21SPR[1:0]		VECT20SPR[1:0]	
ITC_SPR7	VECT27SPR[1:0]		VECT26SPR[1:0]		VECT25SPR[1:0]		VECT24SPR[1:0]	
ITC_SPR8	Reserved				VECT29SPR[1:0]		VECT28SPR[1:0]	
	rw				rw		rw	

Bits 7:0 **VECTxSPR[1:0]**: Vector x software priority bits

These eight read/write registers (ITC\_SPR1 to ITC\_SPR8) are written by software to define the software priority of each interrupt vector.

The list of vectors is given in [Table 33: Vector address map versus software priority bits](#).

Refer to [Section 12.9.1: CPU condition code register interrupt bits \(CCR\)](#) for the values to be programmed for each priority.

Reserved

ITC\_SPR8 bits 7:4 are forced to 1 by hardware.

*Note:* It is forbidden to write 10 (priority level 0). If 10 is written, the previous value is kept and the interrupt priority remains unchanged.

## 12.9.3 External interrupt control register 1 (EXTI\_CR1)

Address offset: 0x00

Reset value: 0x00

	7	6	5	4	3	2	1	0
P3IS[1:0]		P2IS[1:0]		P1IS[1:0]		P0IS[1:0]		
rw		rw		rw		rw		

Bits 7:6 **P3IS[1:0]**: Portx bit 3 external interrupt sensitivity bits

These bits can only be written when I1 and I0 in the CCR register are both set to 1 (level 3). They define the sensitivity of bit 3 of Port A, B, C, D and/or E external interrupts.

00: Falling edge and low level

01: Rising edge only

10: Falling edge only

11: Rising and falling edge

**Bits 5:4 P2IS[1:0]: Portx bit 2 external interrupt sensitivity bits**

These bits can only be written when I1 and I0 in the CCR register are both set to 1 (level 3). They define the sensitivity of bit 2 of Port A, B, C, D and/or E external interrupts.

- 00: Falling edge and low level
- 01: Rising edge only
- 10: Falling edge only
- 11: Rising and falling edge

**Bits 3:2 P1IS[1:0]: Portx bit 1 external interrupt sensitivity bits**

These bits can only be written when I1 and I0 in the CCR register are both set to 1 (level 3). They define the sensitivity of bit 1 of Port A, B, C, D and/or E external interrupts.

- 00: Falling edge and low level
- 01: Rising edge only
- 10: Falling edge only
- 11: Rising and falling edge

**Bits 1:0 P0IS[1:0]: Portx bit 0 external interrupt sensitivity bits**

These bits can only be written when I1 and I0 in the CCR register are both set to 1 (level 3). They define the sensitivity of bit 0 of Port A, B, C, D, E and/or F external interrupts.

- 00: Falling edge and low level
- 01: Rising edge only
- 10: Falling edge only
- 11: Rising and falling edge

## 12.9.4 External interrupt control register 2 (EXTI\_CR2)

Address offset: 0x01

Reset value: 0x00

7	6	5	4	3	2	1	0
P7IS[1:0]	P6IS[1:0]	P5IS[1:0]	P4IS[1:0]				
rw	rw	rw	rw				

### Bits 7:6 P7IS[1:0]: Portx bit 7 external interrupt sensitivity bits

These bits can only be written when I1 and I0 in the CCR register are both set to 1 (level 3). They define the sensitivity of the bit 7 of Port A, B, C, D and/or E external interrupts.

- 00: Falling edge and low level
- 01: Rising edge only
- 10: Falling edge only
- 11: Rising and falling edge

### Bits 5:4 P6IS[1:0]: Portx bit 6 external interrupt sensitivity bits

These bits can only be written when I1 and I0 in the CCR register are both set to 1 (level 3). They define the sensitivity of the bit 6 of Port A, B, C, D and/or E external interrupts.

- 00: Falling edge and low level
- 01: Rising edge only
- 10: Falling edge only
- 11: Rising and falling edge

### Bits 3:2 P5IS[1:0]: Portx bit 5 external interrupt sensitivity bits

These bits can only be written when I1 and I0 in the CCR register are both set to 1 (level 3). They define the sensitivity of the bit 5 of Port A, B, C, D and/or E external interrupts.

- 00: Falling edge and low level
- 01: Rising edge only
- 10: Falling edge only
- 11: Rising and falling edge

### Bits 1:0 P4IS[1:0]: Portx bit 4 external interrupt sensitivity bits

These bits can only be written when I1 and I0 in the CCR register are both set to 1 (level 3). They define the sensitivity of the bit 4 of Port A, B, C, D and/or E external interrupts.

- 00: Falling edge and low level
- 01: Rising edge only
- 10: Falling edge only
- 11: Rising and falling edge

### 12.9.5 External interrupt control register 3 (EXTI\_CR3)

Address offset: 0x02

Reset value: 0x00

7	6	5	4	3	2	1	0
PFIS[1:0]	PEIS[1:0]		PDIS[1:0]		PBIS[1:0]		
rw	rw		rw		rw		

#### Bits 7:6 **PFIS[1:0]:** Port F external interrupt sensitivity bits

These bits can only be written when I1 and I0 in the CCR register are both set to 1 (level 3). They define the sensitivity of the Port F external interrupts, when EXTIF for Port F[3:0] and/or Port F[7:4] is enabled.

- 00: Falling edge and low level
- 01: Rising edge only
- 10: Falling edge only
- 11: Rising and falling edge

#### Bits 5:4 **PEIS[1:0]:** Port E external interrupt sensitivity bits

These bits can only be written when I1 and I0 in the CCR register are both set to 1 (level 3). They define the sensitivity of the Port E external interrupts, when EXTIE for Port E[3:0] and/or Port E[7:4] is enabled.

- 00: Falling edge and low level
- 01: Rising edge only
- 10: Falling edge only
- 11: Rising and falling edge

#### Bits 3:2 **PDIS[1:0]:** Port D external interrupt sensitivity bits

These bits can only be written when I1 and I0 in the CCR register are both set to 1 (level 3). They define the sensitivity of the Port D external interrupts, when EXTID for Port D[3:0] and/or Port D[7:4] is enabled.

- 00: Falling edge and low level
- 01: Rising edge only
- 10: Falling edge only
- 11: Rising and falling edge

#### Bits 1:0 **PBIS[1:0]:** Port B external interrupt sensitivity bits

These bits can only be written when I1 and I0 in the CCR register are both set to 1 (level 3). They define the sensitivity of the Port B external interrupts, when EXTIB for Port B[3:0] and/or PortB[7:4] is enabled.

- 00: Falling edge and low level
- 01: Rising edge only
- 10: Falling edge only
- 11: Rising and falling edge

## 12.9.6 External interrupt control register 4 (EXTI\_CR4)

Address offset: 0x0A

Reset value: 0x00

*Note:* This register is available in medium+ and high density devices only

7	6	5	4	3	2	1	0
Reserved				PHIS[1:0]		PGIS[1:0]	
				rw		rw	

Bits 7:4 Reserved.

Bits 3:2 **PHIS[1:0]:** Port H external interrupt sensitivity bits

These bits can only be written when I1 and I0 in the CCR register are both set to 1 (level 3). They define the sensitivity of the Port H external interrupts, when EXTH for Port H[3:0] and/or Port H[7:4] is enabled.

00: Falling edge and low level

01: Rising edge only

10: Falling edge only

11: Rising and falling edge

Bits 1:0 **PGIS[1:0]:** Port G external interrupt sensitivity bits

These bits can only be written when I1 and I0 in the CCR register are both set to 1 (level 3). They define the sensitivity of the Port G external interrupts, when EXTG for Port G[3:0] and/or Port G[7:4] is enabled.

00: Falling edge and low level

01: Rising edge only

10: Falling edge only

11: Rising and falling edge

## 12.9.7 External interrupt status register 1 (EXTI\_SR1)

Address offset: 0x03

Reset value: 0x00

7	6	5	4	3	2	1	0
P7F	P6F	P5F	P4F	P3F	P2F	P1F	P0F
rc_w1							

Bits 7:0 **PxF:** Port A/B/C/D/E/F bit x external interrupt flag

These bits are set by hardware when an interrupt event occurs on the corresponding pin. They are cleared by writing a '1' by software.

0: No interrupt

1: External interrupt pending

## 12.9.8 External interrupt status register 2 (EXTI\_SR2)

Address offset: 0x04

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved			PHF <sup>(1)</sup>	PGF <sup>(1)</sup>	PFF	PEF	PDF	PBF
			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 7:6 Reserved.

Bit 5<sup>(1)</sup> **PHF:** Port H external interrupt flag

This bit is set by hardware when an interrupt event occurs on the corresponding pin. It is cleared by writing a '1' by software.

0: No interrupt

1: External interrupt pending

Bit 4<sup>(1)</sup> **PGF:** Port G external interrupt flag

This bit is set by hardware when an interrupt event occurs on the corresponding pin. It is cleared by writing a '1' by software.

0: No interrupt

1: External interrupt pending

Bit 3 **PFF:** Port F external interrupt flag

This bit is set by hardware when an interrupt event occurs on the corresponding pin. It is cleared by writing a '1' by software.

0: No interrupt

1: External interrupt pending

Bit 2 **PEF:** Port E external interrupt flag

This bit is set by hardware when an interrupt event occurs on the corresponding pin. It is cleared by writing a '1' by software.

0: No interrupt

1: External interrupt pending

Bit 1 **PDF:** Port D external interrupt flag

This bit is set by hardware when an interrupt event occurs on the corresponding pin. It is cleared by writing a '1' by software.

0: No interrupt

1: External interrupt pending

Bit 0 **PBF:** Port B external interrupt flag

This bit is set by hardware when an interrupt event occurs on the corresponding pin. It is cleared by writing a '1' by software.

0: No interrupt

1: External interrupt pending

1. This bit is available in medium+ and high density devices only. It is reserved in medium density devices.

### 12.9.9 External interrupt port select register (EXTI\_CONF1)

Address offset: 0x05

Reset value: 0x00

7	6	5	4	3	2	1	0
PFES	PFLIS	PEHIS	PELIS	PDHIS	PDLIS	PBHIS	PBLIS

Bit 7 **PFES:** Port F or port E external interrupt select

0: Port E is used for interrupt generation

1: Port F is used for interrupt generation

Bit 6 **PFLIS:** Port F[3:0] external interrupt select

It selects pins PF[3:0] for EXTIF interrupt.

0: PF[3:0] are used for EXTI3-EXTI0 interrupt generation

1: PF[3:0] are used for EXTIF interrupt generation

Bit 5 **PEHIS:** Port E[7:4] external interrupt select

It selects pins PE[7:4] for EXTIE interrupt.

0: PE[7:4] are used for EXTI7-EXTI4 interrupt generation

1: PE[7:4] are used for EXTIE interrupt generation

Bit 4 **PELIS:** Port E[3:0] external interrupt select

It selects pins PE[3:0] for EXTIE interrupt.

0: PE[3:0] are used for EXTI3-EXTI0 interrupt generation

1: PE[3:0] are used for EXTIE interrupt generation

Bit 3 **PDHIS:** Port D[7:4] external interrupt select

It selects pins PD[7:4] for EXTID interrupt.

0: PD[7:4] are used for EXTI7-EXTI4 interrupt generation

1: PD[7:4] are used for EXTID interrupt generation

Bit 2 **PDLIS:** Port D[3:0] external interrupt select

It selects pins PD[3:0] for EXTID interrupt.

0: PD[3:0] are used for EXTI3-EXTI0 interrupt generation

1: PD[3:0] are used for EXTID interrupt generation

Bit 1 **PBHIS:** Port B[7:4] external interrupt select

It selects pins PB[7:4] for EXTIB interrupt.

0: PB[7:4] are used for EXTI7-EXTI4 interrupt generation

1: PB[7:4] are used for EXTIB interrupt generation

Bit 0 **PBLIS:** Port B[3:0] external interrupt select

It selects pins PB[3:0] for EXTIB interrupt.

0: PB[3:0] are used for EXTI3-EXTI0 interrupt generation

1: PB[3:0] are used for EXTIB interrupt generation

### 12.9.10 External interrupt port select register (EXTI\_CONF2)

Address offset: 0x0B

Reset value: 0x00

*Note:* This register is available in medium+ and high density devices only

7	6	5	4	3	2	1	0
Reserved	PHDS	PGBS	PHHIS	PHLIS	PGHIS	PGLIS	PFHIS
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 Reserved

Bit 6 **PHDS:** Port H or port D external interrupt select

0: Port D is used for interrupt generation

1: Port H is used for interrupt generation

Bit 5 **PGBS:** Port G or port B external interrupt select

0: Port B is used for interrupt generation

1: Port G is used for interrupt generation

Bit 4 **PHHIS:** Port H[7:4] external interrupt select

It selects pins PH[7:4] for EXTIH interrupt.

0: PH[7:4] are used for EXTI7-EXTI4 interrupt generation

1: PH[7:4] are used for EXTIH interrupt generation

Bit 3 **PHLIS:** Port H[3:0] external interrupt select

It selects pins PH[3:0] for EXTIH interrupt.

0: PH[3:0] are used for EXTI3-EXTI0 interrupt generation

1: PH[3:0] are used for EXTIH interrupt generation

Bit 2 **PGHIS:** Port G[7:4] external interrupt select

It selects pins PG[7:4] for EXTIG interrupt.

0: PG[7:4] are used for EXTI7-EXTI4 interrupt generation

1: PG[7:4] are used for EXTIG interrupt generation

Bit 1 **PGLIS:** Port G[3:0] external interrupt select

It selects pins PG[3:0] for EXTIG interrupt.

0: PG[3:0] are used for EXTI3-EXTI0 interrupt generation

1: PG[3:0] are used for EXTIG interrupt generation

Bit 0 **PFHIS:** Port F[7:4] external interrupt select

It selects pins PF[7:4] for EXTIF interrupt.

0: PF[7:4] are used for EXTI7-EXTI4 interrupt generation

1: PF[7:4] are used for EXTIF interrupt generation

### 12.9.11 ITC and EXTI register map and reset values

**Table 36. ITC and EXTI register map**

Add. offset	Register name	7	6	5	4	3	2	1	0
<b>ITC-SPR block<sup>(1)</sup></b>									
0x00	ITC_SPR1 Reset value	VECT3SPR1 1	VECT3SPR0 1	VECT2SPR1 1	VECT2SPR0 1	VECT1SPR1 1	VECT1SPR0 1	VECT0SPR1 1	VECT0SPR0 1
0x01	ITC_SPR2 Reset value	VECT7SPR1 1	VECT7SPR0 1	VECT6SPR1 1	VECT6SPR0 1	VECT5SPR1 1	VECT5SPR0 1	VECT4SPR1 1	VECT4SPR0 1
0x02	ITC_SPR3 Reset value	VECT11SPR1 1	VECT11SPR0 1	VECT10SPR1 1	VECT10SPR0 1	VECT9SPR1 1	VECT9SPR0 1	VECT8SPR1 1	VECT8SPR0 1
0x03	ITC_SPR4 Reset value	VECT15SPR1 1	VECT15SPR0 1	VECT14SPR1 1	VECT14SPR0 1	VECT13SPR1 1	VECT13SPR0 1	VECT12SPR1 1	VECT12SPR0 1
0x04	ITC_SPR5 Reset value	VECT19SPR1 1	VECT19SPR0 1	VECT17SPR1 1	VECT17SPR0 1	VECT17SPR1 1	VECT17SPR0 1	VECT16SPR1 1	VECT16SPR0 1
0x05	ITC_SPR6 Reset value	VECT23SPR1 1	VECT23SPR0 1	VECT22SPR1 1	VECT22SPR0 1	VECT21SPR1 1	VECT21SPR0 1	VECT20SPR1 1	VECT20SPR0 1
0x06	ITC_SPR7 Reset value	VECT27SPR1 1	VECT27SPR0 1	VECT26SPR1 1	VECT26SPR0 1	VECT25SPR1 1	VECT25SPR0 1	VECT24SPR1 1	VECT24SPR0 1
0x07	ITC_SPR8 Reset value	- 1	- 1	- 1	- 1	VECT29SPR1 1	VECT29SPR0 1	VECT28SPR1 1	VECT28SPR0 1
<b>ITC-EXTI block<sup>(2)</sup></b>									
0x00	EXTI_CR1	P3IS1 0	P3ISO 0	P2IS1 0	P2ISO 0	P1IS1 0	P1ISO 0	P0IS1 0	P0ISO 0
0x01	EXTI_CR2	P7IS1 0	P7ISO 0	P6IS1 0	P6ISO 0	P5IS1 0	P5ISO 0	P4IS1 0	P4ISO 0
0x02	EXTI_CR3	PFIS1 0	PFISO 0	PEIS1 0	PEISO 0	PDIS1 0	PDIS0 0	PBIS1 0	PBISO 0
0x03	EXTI_SR1	P7F 0	P6F 0	P5F 0	P4F 0	P3F 0	P2F 0	P1F 0	P0F 0
0x04	EXTI_SR2	- 0	- 0	- 0	- 0	PFF 0	PEF 0	PDF 0	PBF 0
0x05	EXTI_CONF 1 0	PFES 0	PFLIS 0	PEHIS 0	PELIS 0	PDHIS 0	PDLIS 0	PBHIS 0	PBLIS 0
0x06 to 0x09	Reserved area								
0x0A <sup>(3)</sup>	EXTI_CR4 <sup>(3)</sup>	- 0	- 0	- 0	- 0	PHIS1 0	PHISO 0	PGIS1 0	PGISO 0
0x0B <sup>(3)</sup>	EXTI_CONF 2 <sup>(3)</sup>	- 0	PHDS 0	PGBS 0	PHHIS 0	PHLIS 0	PGHIS 0	PGLIS 0	PFHIS 0

1. The address offsets are expressed for the ITC-SPR block base address (see Table CPU/SWIM/debug module/interrupt controller registers in the datasheet).
2. The address offsets are expressed for the ITC-EXTI block base address (see Table General hardware register map in the datasheet).
3. These registers are available in medium+ and high density devices only.

## 13 Direct memory access controller (DMA)

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

### 13.1 DMA introduction

Direct memory access (DMA) is used to provide high-speed data transfer between peripherals and memory as well as between memory to memory. Data can be quickly moved by DMA without any CPU actions. This keeps CPU resources free for other operations.

The DMA controller has 4 channels. Each channel is dedicated to managing memory access requests from one or more peripherals. It has an arbiter for handling the priority between DMA requests.

### Glossary

The term **DMA** refers to direct memory access.

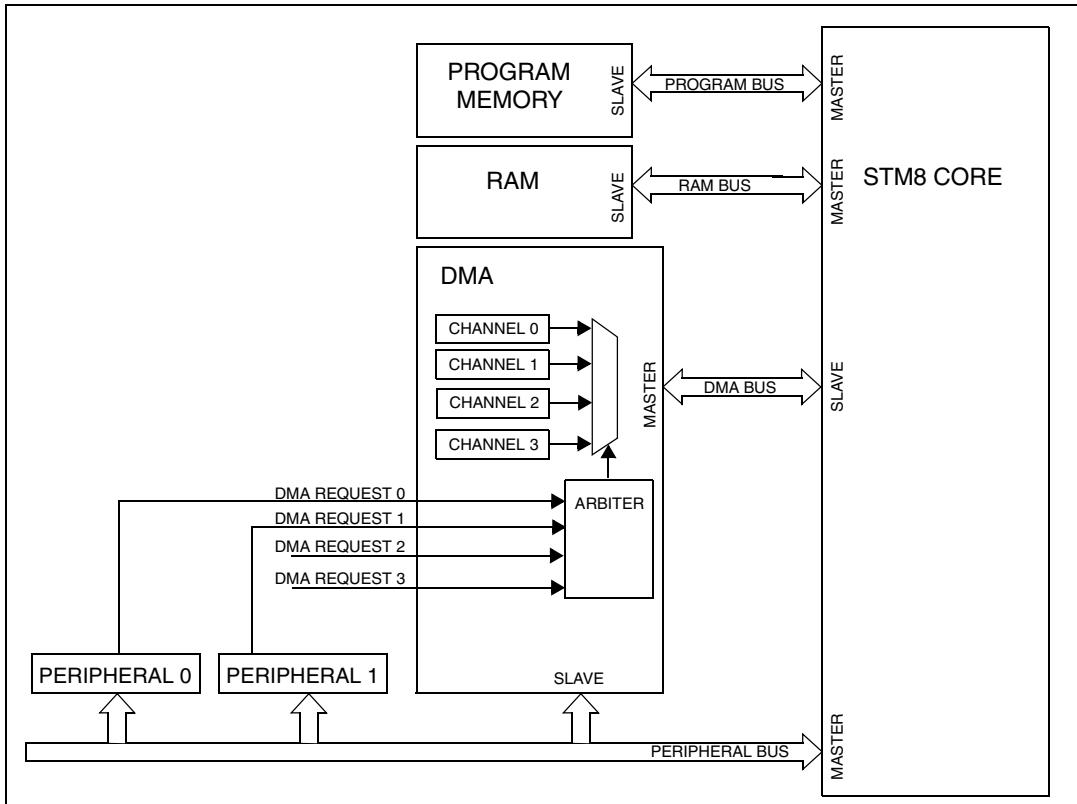
A **DMA transaction** consists of a complete DMA read/write operation on a set of software-programmable data blocks. A DMA transaction can be divided into single DMA transfers.

A **DMA transfer** consists of a single read/write operation on a data block. It cannot be interrupted.

A **data block** is either an 8-bit or a 16-bit data depending on the selected transfer size.

## 13.2 DMA main features

- 4 channels shared among several peripherals (refer to [Table 38: DMA1 channel request mapping on page 182](#))
- Data transfers from peripherals to memory, from memory to peripherals and from memory to memory
- Hardware/software arbitration between each channel depending on the priority level
- Programmable number of “data to be transferred” : up to 255 data block (byte or word) transactions
- Incrementing and decrementing addressing mode
- Channel priority programmable by hardware and software
- Optional interrupt on Half transaction and End of transaction
- Two transfer sizes supported (8-bit and 16-bit data), programmable by software
- Priorities between requests from channels: they can be software-programmable (4 levels consisting of *very high*, *high*, *medium*, *low*) or hardware-programmable in case of equality (request 1 has priority over request 2, etc.).
- Software trigger also supported by memory channels depending on the hardware configuration (memory channel).
- 2 flags (DMA Half transfer, DMA Transfer complete) logically ORed together in a single interrupt request for each channel
- Circular buffer management (auto-reload mode)
- Capability to suspend and resume a DMA transfer.
- Capability to operate in low power modes (Wait for interrupt or Wait for event)

**Figure 32. DMA block diagram**

## 13.3 DMA functional description

The DMA controller performs direct memory transfer by sharing the address and data bus with the STM8™ core. The DMA request may stop the CPU access to the bus for some bus cycles, when the CPU and DMA are targeting the same destination (memory or peripheral). The arbitration is performed inside the STM8™ core and is managed in a different way depending on the chosen bus (peripheral bus, RAM bus or program bus). Refer to the STM8 core description for further information.

Besides, the DMA controller can signal to the STM8 core that the current access must have priority over the CPU. There are two ways to do this:

- the application can specify the timeout duration (number of wait cycles starting from the last request) by configuring the TO[5:0] bits in the DMA\_GCSR register. Then the DMA waits until this timeout has elapsed before requesting from the core a high priority access to the bus.
- the application can also program a channel so that it always takes priority over the CPU.

### 13.3.1 DMA transactions

After an event, the peripheral sends a request signal to the DMA controller. The DMA controller serves the request depending on channel priorities. As soon as the DMA controller accesses the peripheral, an Acknowledge (one cycle pulse) is sent to the

peripheral by the DMA controller. If the peripheral has no other pending request, it releases its request signal as soon as it gets the Acknowledge from the DMA controller.

### 13.3.2 DMA arbiter

The arbiter manages the channel requests based on their priority and it launches the peripheral/memory access sequences.

The priorities are managed in two stages:

- Software: each channel priority can be configured in the DMA\_CxSPR register. There are four levels:
  - Very high priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: if 2 requests have the same software priority level, the channel with the lowest number gets priority versus the channel with the highest number. For example, channel 1 gets priority over channel 3.

*Note:* *Each DMA request is stored into a queue and is served when all requests with higher priority inside the sequence are completed.*

*A channel with a very high priority takes also priority over the CPU.*

### 13.3.3 DMA channels

Four channels are available: three regular channels (channel 0, channel 1 and channel 2) and one memory channel (channel 3).

- The regular channels handle DMA transfers between a peripheral register located at a fixed address and a memory area addressed by an auto-increment/decrement pointer.
- The memory channel is a regular channel that can also handle data transfer between two memory areas managed by two memory pointers.

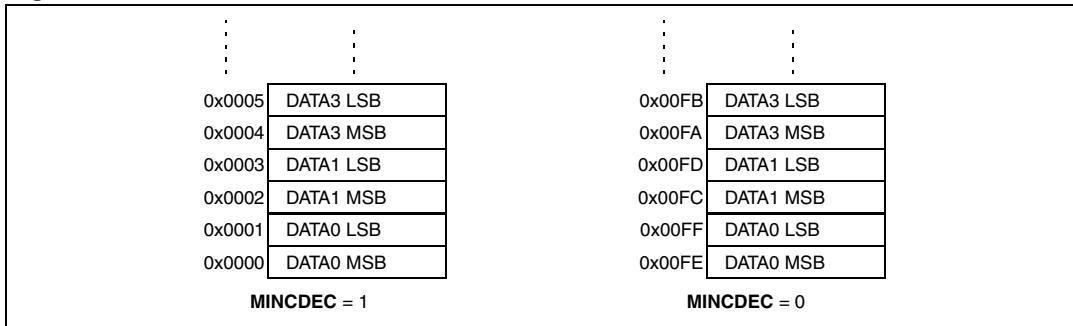
When the channel is configured for a memory transfer, DMA requests are ignored and the transfer is triggered through software: the transaction from memory area 0 to memory area 1 starts as soon as the channel is enabled and stops once the total number of bytes has been transferred.

The amount of data to be transferred (up to 255) is programmable depending on the value in the DMA\_CxNDTR register. The DMA\_CxNDTR register, which contains the amount of data items to be transferred, is decremented after each transfer.

#### Programmable data size

Transfer data size (8-bit or 16-bit) is user programmable through the TSIZE bit in the DMA\_CxSPR register.

*Note:* *When operating in 16-bit mode, the system handles big-endian addressing in both increment or decrement modes. The destination and source pointers must contain the address of the MSB (even address).*

**Figure 33. Endianess in 16-bit mode**

When configured for 16-bit data transfer ( $\text{TSIZE} = 1$ ), a transfer consists of four consecutive 8-bit read and write operations:

- Read the MSB from the source at even address
- Read the LSB from the source at odd address
- Write the MSB to the destination at even address
- Write the LSB to the destination at odd address

To ensure data coherence, the two read operations as well as the two write operations are not divisible.

**Note:**

*This does not prevent the software from synchronizing the CPU and the DMA 16-bit accesses: the software must make sure the DMA will not read the 16-bit data while the CPU has only written half of the 16-bit data to the source area. In the same way, the software must make sure the DMA will not write the 16-bit data while the CPU has already started to read half of the 16-bit data.*

*The initial value address alignment is handled by software.*

### Pointer incrementation

During a memory-to-peripheral or a peripheral-to-memory transaction, the memory pointer is automatically post-incremented or post-decremented after each transaction depending on the MINCDEC bit in the DMA\_CxCR register while the peripheral pointer is always fixed.

If incremented mode is enabled, the address of the next transfer is the address of the previous one incremented by 1. The first transfer address is the address programmed in the DMA\_CxPARH/L and DMA\_CxM0ARH/L registers in medium density devices and in the DMA\_CxPARH/L, DMA\_CxM0ARH/L and DMA\_C3M0EAR (for channel 3) registers in medium+ and high density devices.

If the transfer data size is programmed to 16-bit mode ( $\text{TSIZE} = 1$ ), the address is incremented/decremented by 1 after each byte transfer (by 2 after the two byte transfers).

If the channel is configured in non-circular mode, no DMA request is served after the end of the transfer (that is, once the number of data to be transferred reaches zero).

### Channel configuration procedure

The following sequence should be followed to configure a DMA channel x (where x is the channel number):

1. Set the peripheral register address in the DMA\_CxPARL/H registers. The data is then moved from (or to) this address to (or from) the memory after the peripheral event.

2. Set the memory address in the DMA\_CxPARH/L and DMA\_CxM0ARH/L registers in medium density devices and in the DMA\_CxPARH/L, DMA\_CxM0ARH/L and DMA\_C3M0EAR (for channel 3) registers in medium+ and high density devices. The data is then written to or read from this memory after the peripheral event.
3. Configure the total number of data to be transferred in the DMA\_CxNDTR register. After each peripheral event, this value is then decremented.
4. Configure the channel priority using the PL[1:0] bits in the DMA\_CxSPR register.
5. Configure data transfer direction, circular mode, memory incremented mode, transfer data size, and interrupt after half and/or full transfer in the DMA\_CxCR register.
6. Activate the channel by setting the EN bit (Channel enable bit) in the DMA\_CxCR register.
7. Once all channels are correctly configured (steps 1 to 6), it is possible to enable the DMA through the GEN bit (Global enable bit) in the DMA\_GCSR register.

As soon as the channels are enabled, they can serve any DMA request from the peripherals connected with them.

Once half of the bytes are transferred, the HTIF bit (Half-transfer interrupt flag) is set and an interrupt is generated if the HTIE bit (Half-transfer interrupt enable) is set. At the end of the transfer, the TCIF bit (Transfer complete interrupt flag) is set and an interrupt is generated if the TCIE bit (Transfer complete interrupt enable) is set.

### **Circular mode (auto-reload mode)**

Circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the CIRC bit (circular buffer mode bit) in the DMA\_CxCR register.

When circular mode is activated, the number of data to be transferred is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.

### **Memory-to-memory mode**

This mode is available only for channel 3 which is a memory channel.

Channel 3 can also handle data transfers between two memories, without being triggered by a request from a peripheral. This mode is called Memory to Memory mode.

If the MEM bit in the DMA\_C3CR register is set, then the channel initiates transfers as soon as it is enabled by software by setting the EN bit (Channel enable bit) in the DMA\_C3CR. Peripheral requests are ignored.

The transfer stops once the DMA\_C3NDTR register reaches zero.

Memory-to-memory mode must not be used at the same time as Circular mode.

When the channel is configured in memory-to-memory mode:

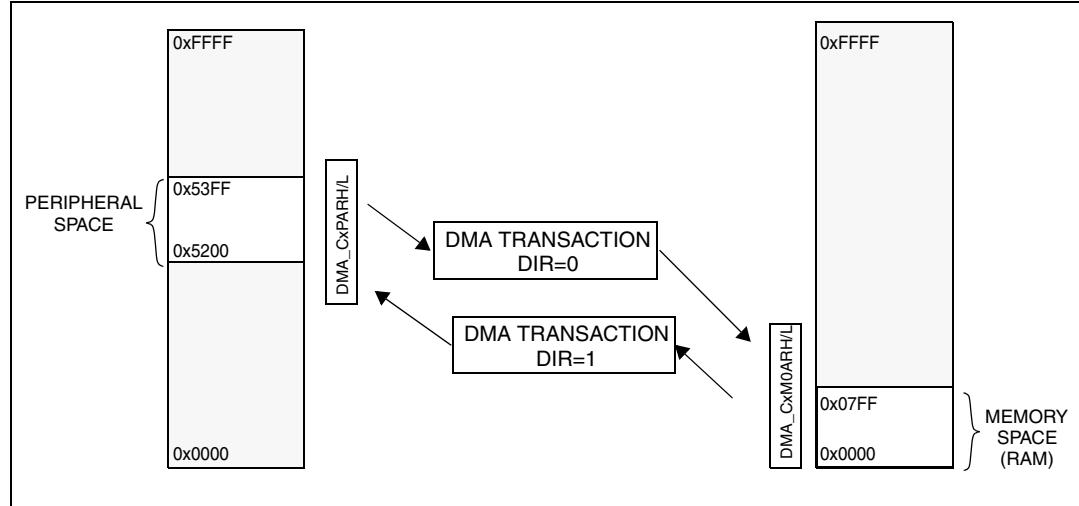
- the Auto-reload mode is disabled and the CIRC bit in the DMA\_C3CR register is “don’t care”.
- the DIR bit (Data transfer direction bit) in the DMA\_C3CR register is “don’t care”: the source is always the memory pointer address 0 (DMA\_CxPARH/L and DMA\_CxM0ARH/L registers in medium density devices and DMA\_CxPARH/L, DMA\_CxM0ARH/L and DMA\_C3M0EAR registers in high density devices) and the destination is always the memory pointer address 1 (DMA\_C3PARL\_C3M1ARL and DMA\_C3PARH\_C3M1ARH registers).

- the source pointer address 0 is always incremented while the destination pointer address 1 is incremented or decremented depending on the MINCDEC bit in the DMA\_C3CR register.

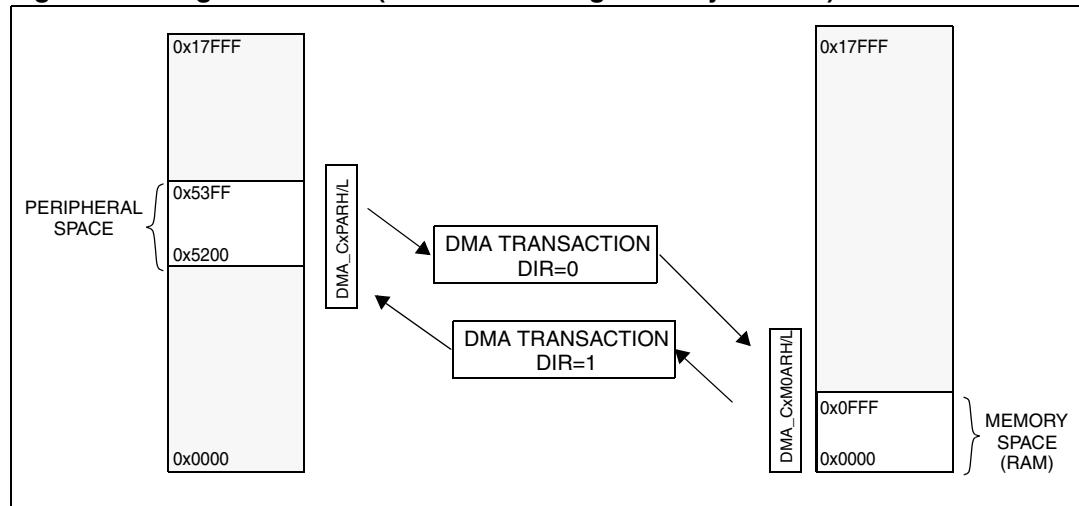
### Source and destination addresses

Channels 0, 1 and 2 can only address peripherals with addresses comprised between 0x5200 and 0x53FF, or within RAM memory (addresses comprised between 0x0000 and 0x07FF for medium density devices and between 0x0000 and 0xFFFF for medium+ and high density devices).

**Figure 34. Regular channel (medium density devices)**



**Figure 35. Regular channel (medium+ and high density devices)**

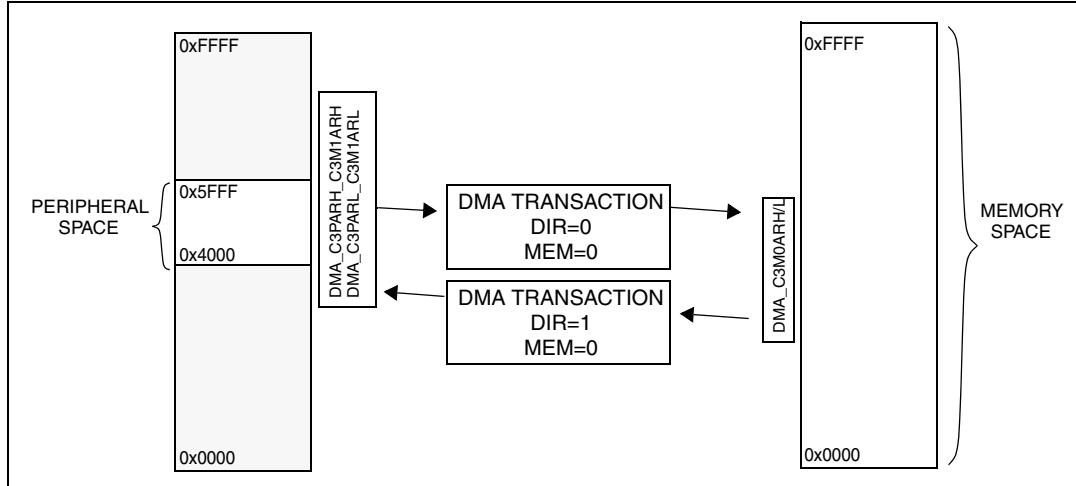


Channel 3, which is a memory channel, can address more memory space:

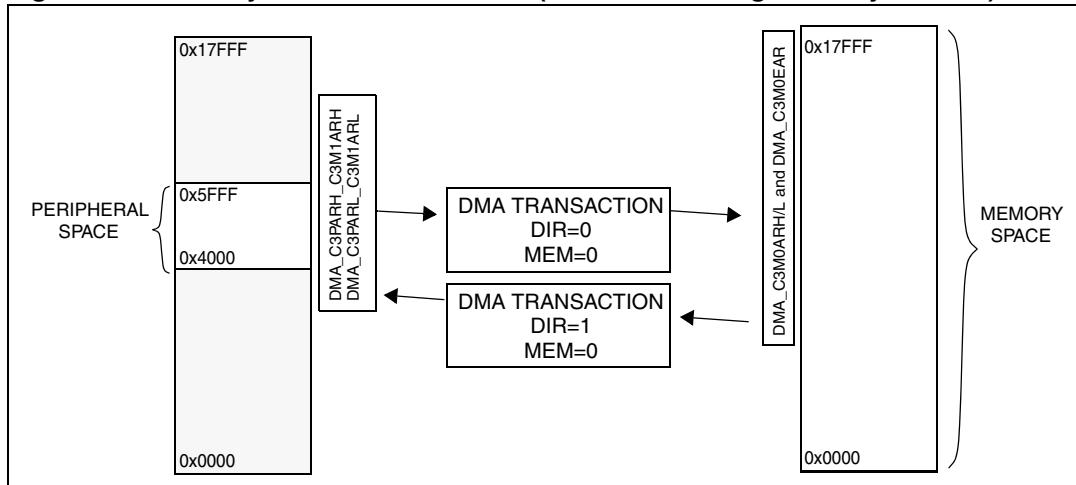
- When performing peripheral-to-memory or memory-to-peripheral transactions (MEM=0), the memory address is comprised between 0x0000 and 0xFFFF for medium density devices and between 0x0000 and 0x17FFF for medium+ and high density devices, the peripheral address is comprised between 0x4000 and 0x5FFF.

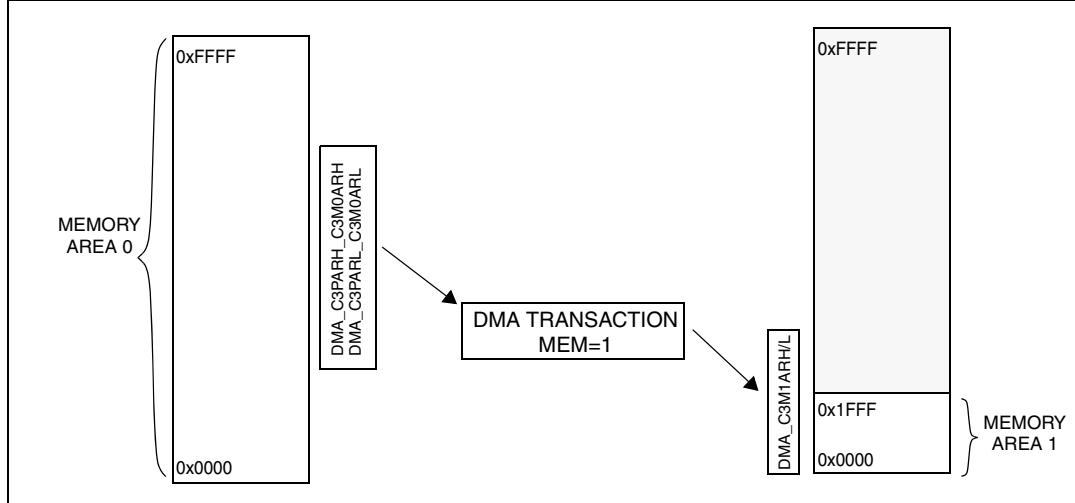
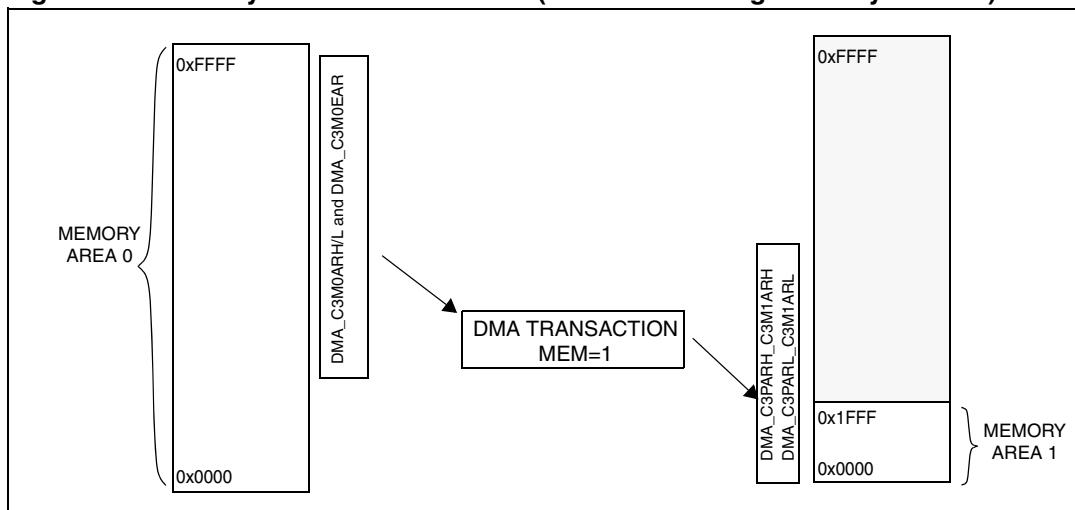
- When performing memory-to-memory transactions (MEM=1), the source memory address is comprised between 0x0000 and 0xFFFF for medium density devices and between 0x0000 and 0x17FFF for medium+ and high density devices while the destination memory address is comprised between 0x0000 and 0x1FFF (which includes the Data EEPROM).

**Figure 36. Memory channel with MEM=0 (medium density devices)**



**Figure 37. Memory channel with MEM=0 (medium+ and high density devices)**



**Figure 38. Memory channel with MEM=1 (medium density devices)****Figure 39. Memory channel with MEM=1 (medium+ and high density devices)**

**Table 37. Source and destination addresses**

Channel	Transfer direction	Source address range	Destination address range
Regular channels <i>Channel 0, Channel1, Channel2</i>	Peripheral to memory	0x5200 to 0x53FF	0x0000 to 0x07FF <sup>(1)</sup> 0x0000 to 0x0FFF <sup>(2)</sup>
	Memory to peripheral	0x0000 to 0x07FF <sup>(1)</sup> 0x0000 to 0x0FFF <sup>(2)</sup>	0x5200 to 0x53FF
Memory channel <i>Channel 3</i>	Peripheral to memory	0x4000 to 0x5FFF	0x0000 to 0xFFFF <sup>(1)</sup> 0x0000 to 0x17FFF <sup>(2)</sup> (memory area 0)
	Memory to peripheral	0x0000 to 0xFFFF <sup>(1)</sup> 0x0000 to 0x17FFF <sup>(2)</sup> (memory area 0)	0x4000 to 0x5FFF
	Memory to memory	0x0000 to 0xFFFF <sup>(1)</sup> 0x0000 to 0x17FFF <sup>(2)</sup> (memory area 0)	0x0000 to 0x1FFF (memory area 1)

1. On medium density devices.
2. On medium+ and high density devices.

### DMA transaction suspension

A DMA transaction can be suspended at any time (even during the transfer) by disabling the EN bit (Channel enable bit) in the DMA\_CxCR register or by disabling each channel using the GEN bit (Global Enable bit) in the DMA\_GCSR register.

If the channel is disabled when a DMA data transfer is ongoing, the channel is effectively disabled only once the current data transfer is completed.

Re-enabling the DMA channel resumes the DMA transaction.

When a DMA transaction is suspended, the software must modify the DMA registers. Otherwise, the DMA transaction may not resume properly. If the number of data to transfer is modified, re-enabling the DMA causes a new transaction to be started instead of the current transaction to be resumed.

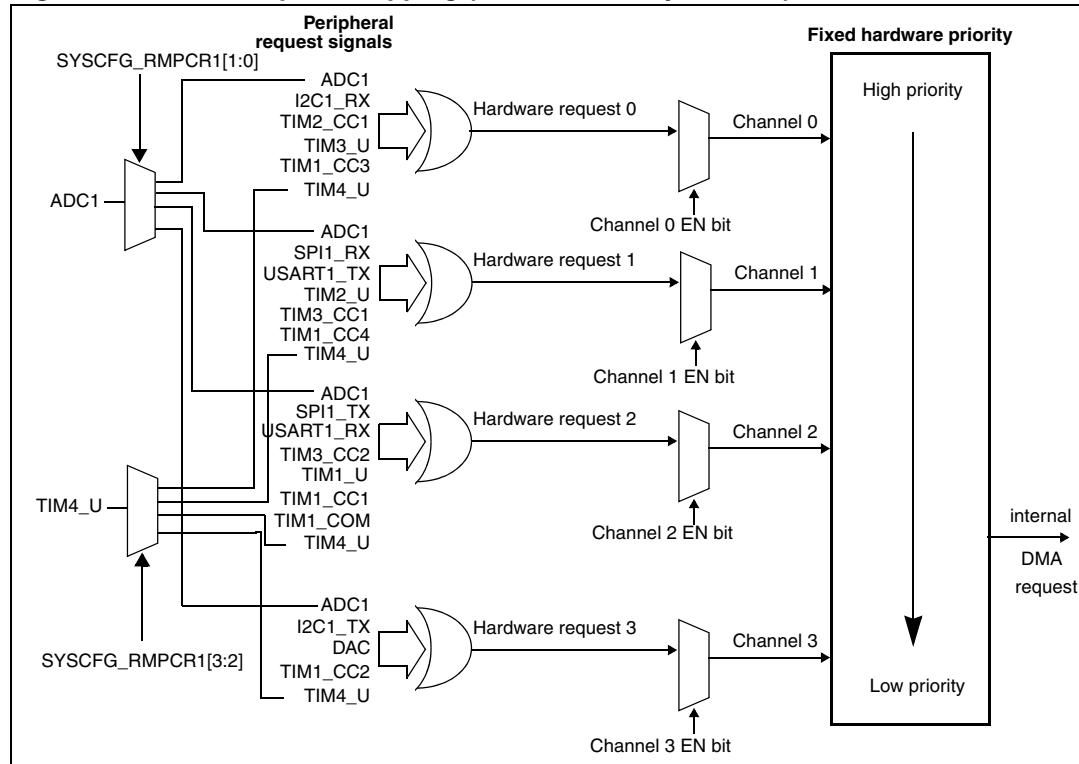
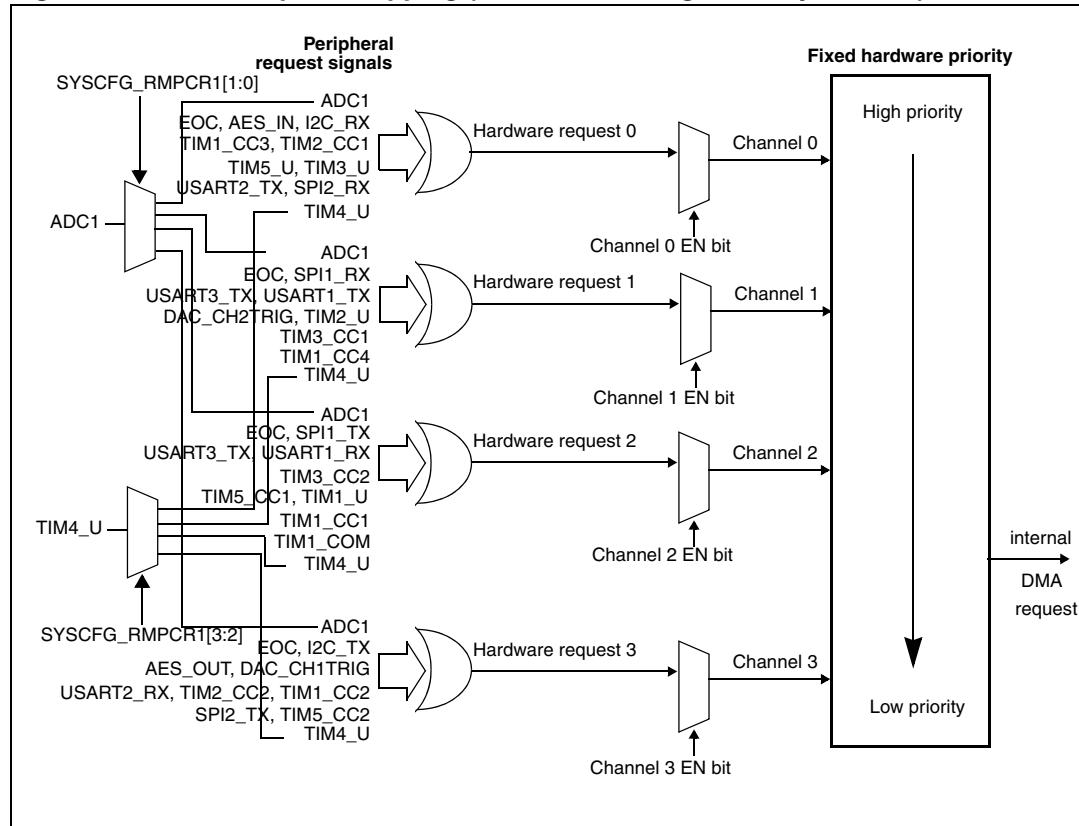
### 13.3.4 DMA1 request mapping

The four hardware requests from the peripherals are simply logically ORed together before entering the DMA. This means that only one hardware request must be enabled at a time. Refer to [Figure 40 on page 183](#). Each OR function allows you to connect ADC1, SPIx, I2C1, USARTx, DAC and TIMx (x=1,2,3,) DMA requests to one of the four channels. The table below lists the DMA requests for each channel.

**Table 38. DMA1 channel request mapping**

IPs	Channel 0	Channel 1	Channel 2	Channel 3
<b>ADC1<sup>(1)</sup></b>	EOC	EOC	EOC	EOC
<b>SPI1</b>		SPI1_RX	SPI1_TX	
<b>AES</b>	AES_IN			AES_OUT
<b>I2C</b>	I2C_RX			I2C_TX
<b>USART1</b>		USART1_TX	USART1_RX	
<b>DAC</b>		DAC_CH2TRIG		DAC_CH1TRIG
<b>TIM2</b>	TIM2_CC1	TIM2_U		TIM2_CC2
<b>TIM3</b>	TIM3_U	TIM3_CC1	TIM3_CC2	
<b>TIM1</b>	TIM1_CC3	TIM1_CC4	TIM1_U TIM1_CC1 TIM1_COM	TIM1_CC2
<b>USART2</b>	USART2_TX			USART2_RX
<b>USART3</b>		USART3_TX	USART3_RX	
<b>SPI2</b>	SPI2_RX			SPI2_TX
<b>TIM5</b>	TIM5_U		TIM5_CC1	TIM5_CC2
<b>TIM4<sup>(2)</sup></b>	TIM4_U	TIM4_U	TIM4_U	TIM4_U

- Note:
- 1 *ADC1 can be mapped on each of the four channels: depending on the SYSCFG\_RMPCR1[1:0] bits (please refer to the ADC chapter and [Section 11.5: SYSCFG registers](#)). The default mapping is Channel 0.*
  - 2 *TIM4 can be mapped on each of the four channels: depending on the SYSCFG\_RMPCR1[3:2] bits (please refer to Timer chapter and [Section 11.5: SYSCFG registers](#)). The default mapping is Channel 3.*

**Figure 40. DMA1 request mapping (medium density devices)****Figure 41. DMA1 request mapping (medium+ and high density devices)**

### 13.3.5 DMA hardware request description

#### AES DMA requests

The AES accelerator provides an interface to connect to the DMA controller. The DMA must be configured to transfer bytes. The AES can be associated with two distinct DMA request channels:

- A DMA request channel for the inputs : When the DMAEN bit is set in the AES\_CR register, the AES initiates a DMA request (AES\_IN) during the input phase each time it requires a byte to be written into the AES\_DINR register. The DMA channel must be configured in memory-to-peripheral mode (bit DTD=1 in the corresponding DMA\_CxCR register).
- A DMA request channel for the outputs : When the DMAEN bit is enabled, the AES initiates a DMA request (AES\_OUT) during the output phase each time it requires a byte to be written into the AES\_DOUTR register. The DMA channel must be configured in peripheral-to-memory mode (bit DTD=0 in the corresponding DMA\_CxCR register).

#### SPIx DMA requests

The Transmission DMA request and the Reception DMA request can be independently enabled or disabled by programming the control bits inside the SPI.

#### USARTTx DMA requests

The Transmission DMA request and the Reception DMA request can be independently enabled or disabled by programming the control bits inside the USART.

**Table 39. SPIx/USARTTx DMA requests**

Request	Description
USARTTx_RX SPIx_RX	Reception requests: DMA transfer request when one character has been received.
USARTTx_TX SPIx_TX	Transmission requests: DMA transfer request when the transmit buffer is empty (Data to be transmitted has been loaded into the shift register)

#### I2C1 DMA requests

DMA requests are generated only for data transfers by the I2C1 data register (I2C1\_DR)

- in transmission when it becomes empty
- in reception when it is full.

When the number of data transfers that have been programmed in the DMA controller is reached, the DMA controller sends an EOT signal (End of transfer) to the I<sup>2</sup>C interface and generates a DMA interrupt.

**Table 40. I2C1 DMA requests**

Request	Description
I2C1_TX	<i>Master transmitter:</i> During an EOT interrupt from the DMA controller, DMA requests must be disabled. The DMA controller has then to wait until a BTF (byte transfer finished) event before programming the STOP condition.
I2C1_RX	<i>Master receiver:</i> The DMA controller sends a hardware signal EOT_1 corresponding to the (number of bytes-1). If LAST=1 in ITDMA register, I2C automatically sends a NACK after the next byte following EOT_1. DMA controller End of transfer interrupt allows to program the STOP condition.

**ADC1 DMA requests**

DMA requests are asserted when the conversion of a selected channel is completed.

**TIMx DMA requests**

TIMx DMA requests can be independently enabled/disabled by programming the DMA control bit inside the Timer.

**Table 41. TIMx DMA requests**

Request	Description
TIMx_UP <sup>(1)</sup>	Update event request: DMA transfer request at each update event
TIMx_CC1 <sup>(1)</sup>	Capture/Compare 1 request: DMA transfer request at each Cap/Com 1 event
TIMx_CC2 <sup>(1)</sup>	Capture/Compare 2 request: DMA transfer request at each Cap/Com 2 event
TIM1_CC3	Capture/Compare 3 request: DMA transfer request at each Cap/Com 3 event
TIM1_CC4	Capture/Compare 4 request: DMA transfer request at each Cap/Com 4 event
TIM1_COM	DMA transfer request at each commutation (COM) event: that is, when CCxE, CCxNE and OCxM Capture/Compare control bits have been updated

1. x= 1, 2, 3 and 4 in medium density devices and x= 1, 2, 3 , 4 and 5 in medium+ and high density devices.

**DAC DMA requests**

DAC DMA requests are generated when the external trigger occurs, either using a software trigger (SWTRIG bit in the DAC\_SWTRIGR register) or a hardware trigger (TIM4\_TRGO in medium density devices and TIM4\_TRGO / TIM5\_TRGO / external trigger input PE4 in medium+ and high density devices). The request indicates that the CHxDHR registers have been transferred to the CHxDOR registers.

## 13.4 DMA low power modes

**Table 42. DMA behavior in low power modes**

Mode	Description
Wait, Low power wait	DMA transfers are still operating
Halt/Active-halt	DMA transfers are stopped

## 13.5 DMA interrupts

**Table 43. DMA interrupt requests**

Interrupt event	Event flag	Enable control bit	Exit from Wait/Low power wait	Exit from Halt/Active-halt
Half transaction interrupt flag	HTIF	HTIE	Yes	No
Transaction complete interrupt flag	TCIF	TCIE	Yes	No

To ensure that a transaction is really completed when using a peripheral to trigger a DMA request, the software must choose carefully between the DMA internal interrupt flag and the peripheral interrupt.

The TCIF flag rises when the whole programmed transaction is completed by the DMA, but this does not necessarily mean that the peripheral ended the data transmission.

*Note:* The reset operation of the HTIF and TCIF bits has priority over the set operation.

## 13.6 DMA registers

The DMA controller has two sets of registers:

- a set of global registers used to globally enable/disable the DMA and to identify quickly the interrupt source
- a set of control, status and pointer registers dedicated for each channel.

In the following description, the name of the registers dedicated to each channel starts with DMA\_Cx where x represents the channel number (numbered from 0 to 3).

### 13.6.1 DMA global configuration & status register (DMA\_GCSR)

Address offset: 0x00

Reset value: 0xFC

7	6	5	4	3	2	1	0
TO[5:0]						GB	GEN
rw	rw	rw	rw	rw	rw	r	rw

Bits 7:2 **TO[5:0]: Timeout**

These bits define the timeout duration (number of cycles to wait starting from the last request). Then the DMA waits until this timeout has elapsed before asking to the core a high priority access to the bus. These bits are write protected if GEN bit and GB bit is set.

When programmed to 0x00, there is no timeout and once a request is served, the DMA immediately asks to the CPU a high priority access to the bus.

Bit 1 **GB: Global busy**

- 0: There is no ongoing DMA transfer
- 1: There is an ongoing DMA transfer

*Note:* The GB bit is logically ORed with all BUSY flags of each channel.

Bit 0 **GEN: Global enable**

- 0: All DMA channels are disabled
- 1: The DMA channels are locally enabled depending on the EN bit in the DMA\_CxCR register.

### 13.6.2 DMA global interrupt register 1 (DMA\_GIR1)

Address offset: 0x01

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	IFC3	IFC2	IFC1	IFC0
r	r	r	r	r	r	r	r

Bits 7: 4 Reserved

Bit 3 **IFC3**: Interrupt flag channel 3

- 0: No pending interrupt on channel 3.
- 1: At least one pending interrupt on channel 3.

Bit 2 **IFC2**: Interrupt flag channel 2

- 0: No pending interrupt on channel 2.
- 1: At least one pending interrupt on channel 2.

Bit 1 **IFC1**: Interrupt Flag Channel 1

- 0: No pending interrupt on channel 1.
- 1: At least one pending interrupt on channel 1.

Bit 0 **IFC0**: Interrupt flag channel 0

- 0: No pending interrupt on channel 0.
- 1: At least one pending interrupt on channel 0.

**Note:** This register is useful to quickly identify the channel which has generated an interrupt without checking all status registers of each channel. Then the software can check the corresponding channel status register and handle the interrupt software management. For each channel, the dedicated flag is set by hardware as soon as one of the two interrupt flags of the corresponding channel is set. The software resets it when clearing the interrupt flag of the DMA\_CxSPR register of the channel which has generated the interrupt.

### 13.6.3 DMA channel configuration register (DMA\_CxCR)

There is one control register per channel where x represents the regular channel number (numbered from 0 to 3).

Address offset: Refer to [Table 44: DMA register map on page 194](#)

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved	MEM (rw) or Reserved (r)	MINCDEC	CIRC	DIR	HTIE	TCIE	EN
r	r/rw	rw	rw	rw	rw	rw	rw

This register is write protected when the DMA channel is enabled (EN bit and GEN bit set) and when the channel is busy (BUSY bit set).

Bit 7 Reserved

Bit 6 **MEM**: Memory transfer enabled

- 0: The channel works as a regular channel.
- 1: The channel works as a memory channel.

*Note: This bit is implemented only for channel 3 (memory channel). Otherwise it is a reserved bit.*

Bit 5 **MINCDEC**: Memory increment/decrement mode

- 0: Decrement mode.
- 1: Increment mode.

This bit defines if the memory address pointer will be incremented or decremented but the peripheral address pointer is never incremented or decremented.

*Note: In case of memory to memory transfer, only the destination memory pointer (memory pointer address 1) can be incremented or decremented. This bit has no effect on the source memory pointer (Memory pointer address 0) which is always incremented (it cannot be decremented).*

Bit 4 **CIRC**: Circular buffer mode (Auto-reload mode)

- 0: Circular mode disabled.
- 1: Circular mode enabled.

*Note: This bit is don't care if MEM = 1 (in this case, the Auto-reload mode is always disabled)*

Bit 3 **DIR**: Data transfer direction

- 0: Transfer from the peripheral to the memory.
- 1: Transfer from the memory to the peripheral.

*Note: This bit is don't care if MEM = 1*

Bit 2 **HTIE**: Half-transaction interrupt enable

- 0: Half-transaction interrupt disabled.
- 1: Half-transaction interrupt enabled.

Bit 1 **TCIE**: Transaction complete interrupt enable

- 0: Transaction complete interrupt disabled.
- 1: Transaction complete interrupt enabled.

Bit 0 **EN**: Channel enable

- 0: Channel disabled.
- 1: Channel disable.

*Note:*

- 1 At the end of a transaction (when the current data counter reaches zero), the DMA channel is NOT automatically disabled (for interrupt mask management).
- 2 When the channel is disabled, all pointer registers remain as they were.

### 13.6.4 DMA channel status & priority register (DMA\_CxSPR)

There is one status and priority register (SPR) per channel where  $x$  represents the channel number (numbered from 0 to 3).

Address offset: Refer to [Table 44: DMA register map on page 194](#)

Reset value: 0x00

7	6	5	4	3	2	1	0
BUSY	PEND	PL[1:0]		TSIZE	HTIF	TCIF	Reserved
r	r	rw	rw	rw	rc_w0	rc_w0	r

Bit 7 **BUSY**: Channel busy

- 0: There is no ongoing DMA transfer
- 1: There is an ongoing DMA transfer

Bit 6 **PEND**: Channel pending

- 0: There is no pending request
- 1: There is a DMA pending request

*Note: If MEM = 1, this bit is don't care.*

Bit 5:4 **PL[1:0]**: Channel priority level

This bit is used for software arbitration between channels

- 00: Low
- 01: Medium
- 10: High
- 11: Very high (DMA takes precedence over the CPU).

*Note: These bits are write protected when the channel is enabled (EN and GEN bits set) and when the channel is busy (BUSY bit set).*

Bit 3 **TSIZE**: Transfer size

This bit is used to perform a 16-bit wide transfer to ensure data coherence

- 0: 8-bit mode
- 1: 16-bit mode

*Note: This bit is write protected when the channel is enabled (EN and GEN bits set) and when the channel is busy (BUSY bit set).*

Bit 2 **HTIF**: Half transaction interrupt flag

- 0: No event.
- 1: Half transaction completed.

This bit is set by hardware and can be cleared by software writing 0 (even if the DMA channel is enabled).

*Note: The reset operation of the HTIF bit has priority over the set operation.*

Bit 1 **TCIF**: Transaction complete interrupt flag

- 0: No event.
- 1: Transaction completed.

This bit is set by hardware and can be cleared by software writing 0 (even if the DMA channel is enabled).

*Note: The reset operation of the TCIF bit has priority over the set operation.*

Bit 0 Reserved

### 13.6.5 DMA number of data to transfer register (DMA\_CxNDTR)

There is one “number of data to transfer register” per channel where  $x$  represents the channel number (numbered from 0 to 3).

Address offset: Refer to [Table 44: DMA register map on page 194](#)

Reset value: 0x00

7	6	5	4	3	2	1	0
NDT[7:0]							
rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r

Bits 7:0 **NDT[7:0]**: Number of data to transfer

Number of data (byte or word) to be transferred (0 up to 255).

This register can only be written when the channel is disabled. When the channel is enabled, this register is write protected and can be read to indicate the remaining data to be transmitted.

Once the transaction is completed, this register remains to zero.

If the channel is configured in auto-reload mode (CIRC = 1), it is automatically reloaded by the value previously programmed.

If a request occurs and the number of data to transfer is zero, the DMA does not serve the transaction.

### 13.6.6 DMA peripheral address high register (DMA\_CxPARH)

Address offset: Refer to [Table 44: DMA register map on page 194](#)

There is one “peripheral address high register” per regular channel where  $x$  represents the channel number (numbered from 0 to 2).

This register is write protected when the DMA channel is enabled (EN = 1 and GEN = 1) and when the channel is busy (BUSY = 1).

Reset value: 0x52

7	6	5	4	3	2	1	0
PA[15:8]							
r	r	r	r	r	r	r	rw

Bits 7:0 **PA[15:8]**: Peripheral address pointer (MSB)

The PA pointer is the source address when DIR = 0 or the destination address when DIR = 1.

Only bit 0 is accessible and the other bits are fixed to allow a value range from 0x52 to 0x53.

### 13.6.7 DMA peripheral address low register (DMA\_CxPARL)

There is one “peripheral address low register” per regular channel where  $x$  represents the regular channel number (numbered from 0 to 2).

Address offset: Refer to [Table 44: DMA register map on page 194](#).

Reset value: 0x00

This register is write protected when the DMA channel is enabled (EN and GEN bits set) and when the channel is busy (BUSY bit set).

7	6	5	4	3	2	1	0
PA[7:0]							
rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r

Bits 7:0 **PA[7:0]**: Peripheral address pointer (LSB)

The PA Pointer is the source address if DIR = 0 or the destination address if DIR = 1.

### 13.6.8 DMA channel 3 peripheral address high & memory 1 address high register (DMA\_C3PARH\_C3M1ARH)

Address offset: Refer to [Table 44: DMA register map on page 194](#). This register is write protected when the DMA channel is enabled (EN= 1 and GEN = 1) and when the channel is busy (BUSY = 1).

It has two different meanings depending on the MEM bit configuration:

- DMA channel 3 peripheral address high (DMA\_C3PARH)

*Note:* *This definition is valid when the MEM bit is reset.*

Reset value: 0x40

7	6	5	4	3	2	1	0
PA[15:8]							
r	r	r	rw/r	rw/r	rw/r	rw/r	rw/r

Bits 7:0 **PA[15:8]**: Peripheral address pointer (MSB)

The PA pointer is the source address when DIR = 0 or the destination address when DIR = 1.

Only bits 0, 1, 2, 3 and 4 are accessible and the other bits are fixed to allow a value range from 0x40 to 0x5F.

- DMA channel 3 memory 1 address high (DMA\_C3M1ARH)

*Note:* *This definition is valid when the MEM bit is set.*

Reset value: N/A

7	6	5	4	3	2	1	0
M1A[15:8]							
r	r	r	rw/r	rw/r	rw/r	rw/r	rw/r

Bits 7:0 **M1A[15:8]**: Memory 1 address pointer (MSB)

The M1A pointer is the destination address when performing memory-to-memory transfers. Only bits 0, 1, 2, 3 and 4 are accessible. The other bits are fixed to allow a value range from 0x00 to 0x1F.

### 13.6.9 DMA channel 3 peripheral address low & memory 1 address low register (DMA\_C3PARL\_C3M1ARL)

Address offset: Refer to [Table 44: DMA register map on page 194](#)

Reset value: 0x00

This register is write protected when the DMA channel is enabled (EN and GEN bits set) and when the channel is busy (BUSY bit set).

This register has two different meanings depending on the MEM bit configuration:

- DMA channel 3 peripheral address low (DMA\_C3PARL)

*Note: This definition is valid when the MEM bit is reset.*

7	6	5	4	3	2	1	0
PA[7:0]							
rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r

Bits 7:0 **PA[7:0]**: Peripheral address pointer (LSB)

The PA Pointer is the source address if DIR = 0 or the destination address if DIR = 1.

- DMA channel 3 memory 1 address low (DMA\_C3M1ARL)

*Note: This definition is valid when the MEM bit is set.*

7	6	5	4	3	2	1	0
M1A[7:0]							
rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r

Bits 7:0 **M1A[7:0]**: Memory 1 address pointer (LSB)

The M1A pointer is the destination address when performing memory-to-memory transfers.

### 13.6.10 DMA memory 0 address high register (DMA\_CxM0ARH)

Address offset: Refer to [Table 44: DMA register map on page 194](#)

Reset value: 0x00

This register is write protected when the DMA channel is enabled (EN = 1 and GEN = 1) and when the channel is busy (BUSY = 1).

7	6	5	4	3	2	1	0
MOA[15:8]							
rw/r or r	rw/r or r	rw/r or r	rw/r	rw/r or r	rw/r	rw/r	rw/r

Bits 7:0 **MOA[15:8]**: Memory 0 address pointer (MSB)

- When MEM = 0, for regular channels or memory channels, the MOA pointer is the destination address when DIR = 0 or the source address when DIR = 1.
- When MEM = 1, for memory channels, the MOA pointer is always the source address. All bits are accessible to allow a value range from 0x00 to 0xFF.
- For regular channels, only bits 0,1 and 2 are accessible and the other bits are fixed to allow a value range from 0x00 to 0x07.

### 13.6.11 DMA memory 0 address low register (DMA\_CxM0ARL)

Address offset: Refer to [Table 44: DMA register map on page 194](#)

Reset value: 0x00

This register is write protected when the DMA channel is enabled (EN= 1 and GEN = 1) and when the channel is busy (BUSY = 1).

7	6	5	4	3	2	1	0
MOA[7:0]							
rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r	rw/r

Bits 7:0 **MOA[7:0]**: Memory 0 address pointer (LSB)

For regular channels or memory channels when MEM = 0, the MOA Pointer is the destination address when DIR = 0 or the source address when DIR = 1.

For memory channels when MEM = 1, the MOA Pointer is always the source address.

### 13.6.12 DMA channel 3 memory 0 extended address register (DMA\_C3M0EAR)

Address offset: Refer to [Table 44: DMA register map on page 194](#)

Reset value: 0x00

**Note:** *This register is available in medium+ and high density devices only.*

It is write protected when the DMA channel is enabled (EN= 1 and GEN = 1) and when the channel is busy (BUSY = 1).

7	6	5	4	3	2	1	0
Reserved							MOA16
r							rw/r

Bit 0 **MOA16**: Memory 0 address pointer 16

For regular channels or memory channels when MEM = 0, the MOA pointer is the destination address when DIR = 0 or the source address when DIR = 1.

For memory channels when MEM = 1, the MOA Pointer is always the source address.

Bits 7:1 Reserved

### 13.6.13 DMA register map and reset values

**Table 44. DMA register map**

Offset address	Register name	7	6	5	4	3	2	1	0
0x00	DMA_GCSR Reset value	T05 1	T04 1	T03 1	T02 1	TO1 1	TO0 1	GB 0	GEN 0
0x01	DMA_GIR1 Reset value	- 0	- 0	- 0	- 0	IFC3 0	IFC2 0	IFC1 0	IFC0 0
DMA channel 0 registers									
0x05	DMA_C0CR Reset value	- 0	- 0	MINCDEC 0	CIRC 0	DIR 0	HTIE 0	TCIE 0	EN 0
0x06	DMA_C0SPR Reset value	BUSY 0	PEND 0	PL1 0	PL0 0	TSIZE 0	HTIF 0	TCIF 0	- 0
0x07	DMA_CONDTR Reset value	NDT7 0	NDT6 0	NDT5 0	NDT4 0	NDT3 0	NDT2 0	NDT1 0	NDT0 0
0x08	DMA_C0PARH Reset value	PA15 0	PA14 1	PA13 0	PA12 1	PA11 0	PA10 0	PA9 1	PA8 0
0x09	DMA_C0PTRL Reset value	PA7 0	PA6 0	PA5 0	PA4 0	PA3 0	PA2 0	PA1 0	PA0 0
0x0A	Reserved area								
0x0B	DMA_C0M0ARH Reset value	M0A15 0	M0A14 0	M0A13 0	M0A12 0	M0A11 0	M0A10 0	M0A9 0	M0A8 0
0x0C	DMA_C0M0ARL Reset value	M0A7 0	M0A6 0	M0A5 0	M0A4 0	M0A3 0	M0A2 0	M0A1 0	M0A0 0
DMA channel 1 registers									
0x0F	DMA_C1CR Reset value	- 0	- 0	MINCDEC 0	CIRC 0	DIR 0	HTIE 0	TCIE 0	EN 0
0x10	DMA_C1SPR Reset value	BUSY 0	PEND 0	PL1 0	PL0 0	TSIZE 0	HTIF 0	TCIF 0	- 0
0x11	DMA_C1NDTR Reset value	NDT7 0	NDT6 0	NDT5 0	NDT4 0	NDT3 0	NDT2 0	NDT1 0	NDT0 0
0x12	DMA_C1PARH Reset value	PA15 0	PA14 1	PA13 0	PA12 1	PA11 0	PA10 0	PA9 1	PA8 0
0x13	DMA_C1PTRL Reset value	PA7 0	PA6 0	PA5 0	PA4 0	PA3 0	PA2 0	PA1 0	PA0 0
0x14	Reserved area								
0x15	DMA_C1M0ARH Reset value	M0A15 0	M0A14 0	M0A13 0	M0A12 0	M0A11 0	M0A10 0	M0A9 0	M0A8 0
0x16	DMA_C1M0ARL Reset value	M0A7 0	M0A6 0	M0A5 0	M0A4 0	M0A3 0	M0A2 0	M0A1 0	M0A0 0
DMA channel 2 registers									
0x19	DMA_C2CR Reset value	- 0	- 0	MINCDEC 0	CIRC 0	DIR 0	HTIE 0	TCIE 0	EN 0
0x1A	DMA_C2SPR Reset value	BUSY 0	PEND 0	PL1 0	PL0 0	TSIZE 0	HTIF 0	TCIF 0	- 0
0x1B	DMA_C2NDTR Reset value	NDT7 0	NDT6 0	NDT5 0	NDT4 0	NDT3 0	NDT2 0	NDT1 0	NDT0 0
0x1C	DMA_C2PARH Reset value	PA15 0	PA14 1	PA13 0	PA12 1	PA11 0	PA10 0	PA9 1	PA8 0

**Table 44. DMA register map (continued)**

Offset address	Register name	7	6	5	4	3	2	1	0
0x1D	DMA_C2PARL Reset value	PA7 0	PA6 0	PA5 0	PA4 0	PA3 0	PA2 0	PA1 0	PA0 0
0x1E Reserved area									
0x1F	DMA_C2M0ARH Reset value	M0A15 0	M0A14 0	M0A13 0	M0A12 0	M0A11 0	M0A10 0	M0A9 0	M0A8 0
0x20	DMA_C2M0ARL Reset value	M0A7 0	M0A6 0	M0A5 0	M0A4 0	M0A3 0	M0A2 0	M0A1 0	M0A0 0
DMA channel 3 registers									
0x23	DMA_C3CR Reset value	- 0	MEM 0	MINCDEC 0	CIRC 0	DIR 0	HTIE 0	TCIE 0	EN 0
0x24	DMA_C3SPR Reset value	BUSY 0	PEND 0	PL1 0	PL0 0	TSIZE 0	HTIF 0	TCIF 0	- 0
0x25	DMA_C3NDTR Reset value	NDT7 0	NDT6 0	NDT5 0	NDT4 0	NDT3 0	NDT2 0	NDT1 0	NDT0 0
0x26	DMA_C3PARH_C3M1ARH	PA[15:8] (reset value: 0x40) or M1A[15:8]							
0x27	DMA_C3PARL_C3M1ARL Reset value	PA[7:0] or M1A[7:0] 0000 0000							
0x28	DMA_C3M0EAR <sup>(1)</sup> Reset value	- 0	- 0	- 0	- 0	- 0	- 0	- 0	M0A16 0
0x29	DMA_C3M0ARH Reset value	M0A15 0	M0A14 0	M0A13 0	M0A12 0	M0A11 0	M0A10 0	M0A9 0	M0A8 0
0x2A	DMA_C3M0ARL Reset value	M0A7 0	M0A6 0	M0A5 0	M0A4 0	M0A3 0	M0A2 0	M0A1 0	M0A0 0

1. Available in medium+ and high density devices only. Reserved in medium density devices.

## 14 Analog-to-digital converter (ADC)

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

### 14.1 ADC introduction

The analog-to-digital converter is used to convert the analog voltage signals to digital values. Up to 28 analog channels are available. A/D conversion can be performed in single or continuous mode.

### 14.2 ADC main features

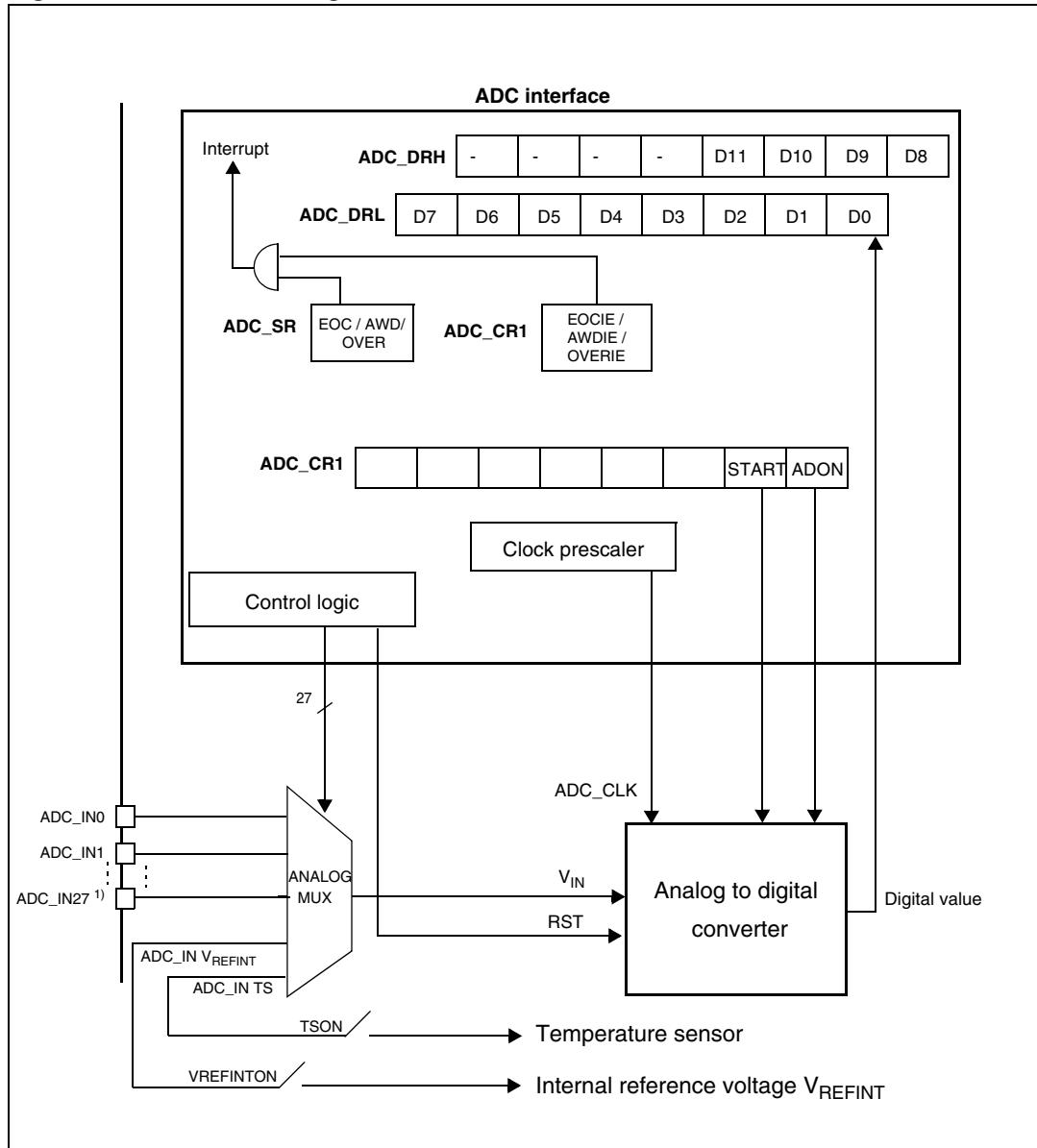
- Configurable resolution (up to 12-bit data width)
- Number of analog channels:
  - Medium density devices:  
25 analog channels : 1 fast channel (1 µs) + 24 slow channels
  - Low, medium+ and high density devices:  
28 analog channels : 4 fast channels (1 µs) + 24 slow channels
- 2 internal channels connected to temperature sensor and internal reference voltage
- Configurable single or continuous conversion
- Prescalable ADC clock
- Analog watchdog
- Separate interrupt generation at end of conversion, watchdog or overrun event
- Multiple channel conversion (scan mode)
- Data integrity mechanism
- DMA capability
- Programmable sampling time
- Schmitt trigger disabling capability
- Conversion time which can be up to 1 µs when SYCLK = 16 MHz
- Voltage range: 1.8 V to 3.6 V
  - Maximum conversion rate obtained from 2.4 V to 3.6 V
  - ADC at a lower speed between 1.8 V and 2.4 V
  - ADC functionality not guaranteed below 1.8 V

## 14.3 ADC functional description

### 14.3.1 General description

*Figure 42* shows the block diagram of the complete system ADC interface.

**Figure 42. ADC block diagram**



1. ADC\_IN27 for medium+ and high density devices.  
ADC\_IN24 for medium density devices.
2. Only basic features are shown in this diagram

### 14.3.2 Number of analog channels

Up to 30 analog input channels are available:

- Up to 27 analog input channels in the medium density devices.
- Up to 28 analog input channels in the low density devices.
- Up to 30 analog input channels in the medium+ and high density devices.

They can be classified into three groups:

- Slow channels: channels 0 to 23 with a sampling time selected through the SMP1 bits in the ADC\_CR2 register
- Fast channels: channel 24 (medium density devices) or channels 24 to 27 (medium+ and high density devices) with a sampling time selected through the SMP2 bits in the ADC\_CR3 register
- Fast internal channels: channels V<sub>REFINT</sub> and TS with a sampling time selected through the SMP2 bits in the ADC\_CR3 register.

### 14.3.3 ADC on-off control

The ADC can be powered-on by setting the ADON bit in the ADC\_CR1 register. When the ADON bit is set, it wakes up the ADC from Power-down mode. Conversion should be started only when the ADC power-up time ( $t_{WKUP}$ ) has elapsed and before the ADC maximum idle delay ( $t_{IDLE}$ ) has elapsed. The software can stop conversion and put the ADC in Power-down mode by resetting the ADON bit.

### 14.3.4 Single conversion mode

In this mode, only one input channel must be selected in the ADC\_SQRx registers (if more channels are selected, the highest selected channel is measured). In addition, the DMAOFF bit must be set in the ADC\_SQR1 register (to disable DMA transfer). The input channel is then converted and the ADC conversion stops (one simple conversion). The converted value is stored into the ADC\_DR data register. An interrupt (EOC) can be generated after the end of conversion. The time between 2 conversions must be lower than the ADC maximum idle delay ( $t_{IDLE}$ ). In case the time between 2 conversions is greater than  $t_{IDLE}$ , the ADC must be powered-off between the 2 conversions (by clearing the ADON bit).

Another possibility is to discard the first conversion (occurring in a time greater than tidle after previous one) and keep the next one

*Note:* If the DMAOFF bit in the ADC\_SQR1 register is reset (and if DMA is properly programmed) the conversion is then performed in single scan mode.

In the single conversion mode, the ADC does one conversion. The conversion can be started in two different ways:

- by software: conversion is performed by setting the START bit in the ADC\_CR1 register. The START bit is then reset by hardware.

*Note:* The channel selection is performed using the ADC\_SQRx registers.

*Before starting a conversion, the software should wait for the stabilization time ( $t_{WKUP}$ ).*

- by hardware: three external triggers can start a conversion (ADC\_TRIGR1, ADC\_TRIGR2 or ADC\_TRIGR3). The selection of one of these three triggers is made through the EXTSEL[1:0] bits in the ADC\_CR2 register. The conversion can be triggered either on the rising edge, on the falling edge or on both edges of the signal, depending on the TRIG\_EDGE[1:0] bits in the ADC\_CR2 register.

**Note:** Any start event occurring during the conversion will be ignored.

Once the conversion is complete:

- the EOC flag is set
- and an interrupt is produced if the EOCIE bit is set in the ADC\_CR1 register.

The EOC flag can be reset by software or by reading the LSB of the converted data.

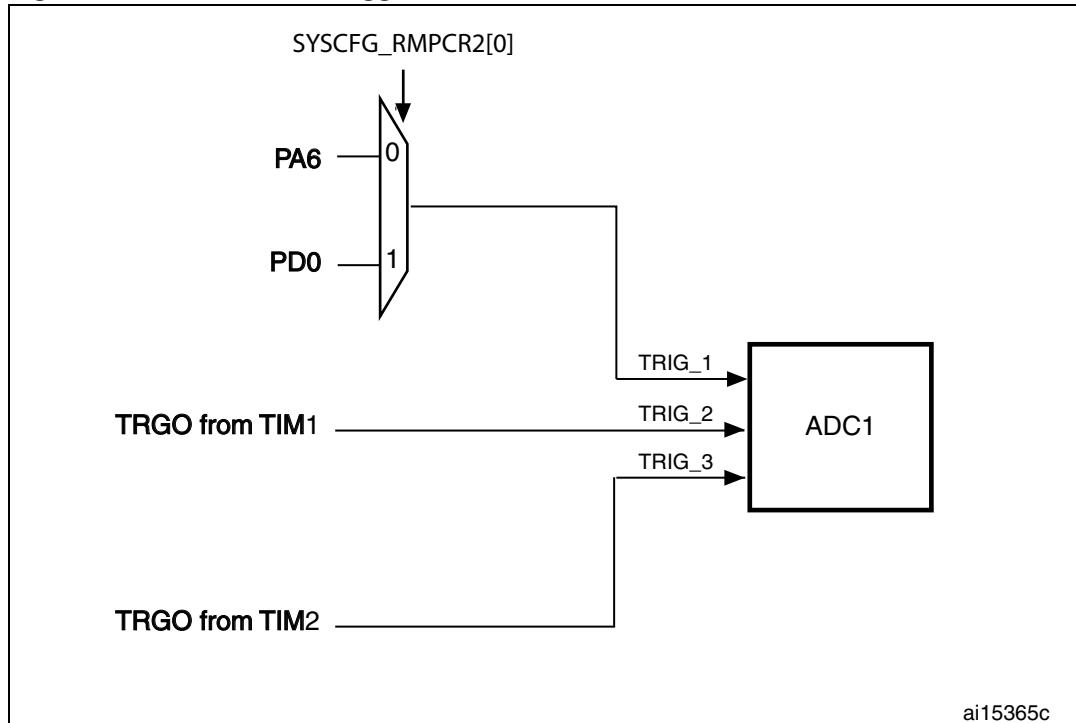
Further single conversions can be initiated by simply setting the START bit or by rising an external trigger.

- Note:**
- 1 *The channel must not be changed during a conversion.*
  - 2 *The user must avoid generating triggers before the end of an ongoing conversion.*
  - 3 *The trigger edge configuration must be set before enabling the triggers*
  - 4 *Even if no channel is programmed, a conversion can be started and the EOC flag will be set.*

### ADC external triggers

Three external trigger sources can be selected:

- ADC trigger 1.  
The trigger can be performed
  - either from PA6 (*if the ADC\_TRIG bit in the SYSCFG\_RMPCCR2 register is reset; refer to Alternate function remapping section in the datasheet*)
  - or directly from PD0 (*if ADC\_TRIG bit in the SYSCFG\_RMPCCR2 register is set; refer to Alternate function remapping section in the datasheet*)
- ADC trigger 2.  
The trigger is performed directly from Timer 1 trigger output (TIM1\_TRGO)
- ADC trigger 3.  
The trigger is performed directly from Timer 2 trigger output (TIM2\_TRGO)

**Figure 43. ADC external triggers**

#### 14.3.5 Continuous conversion mode

In this mode, the ADC does not stop after conversion but continues to the next channel in the selected channel sequence. Conversions continue until the CONT bit and the ADON bits are set and the converted values are transferred through the DMA to the RAM or EEPROM memory buffer.

As the EOC interrupt is generated only after the end of the conversion of the selected channel sequence, the result of each channel conversion cannot be read from the ADC\_DR register.

To save the conversion of each channel in a memory (RAM or EEPROM), the DMA must be used in peripheral to memory mode. If the CONT bit is reset during a conversion, the current selected channel sequence conversions end with the last selected channel and then the ADC stops.

*Note:* The CONT bit must not be set again during a conversion.

#### 14.3.6 ADC clock

The ADC clock provided by the clock controller can be either the ADC system clock (CK) or the ADC system clock divided by 2. The selection of the frequency feeding the clock ( $f_{ADC\_CLK} = CK$  or  $f_{ADC\_CLK} = CK/2$ ) is done through the PRESC bit in the ADC\_CR2 register.

#### 14.3.7 Analog watchdog

The analog watchdog status bit (AWD) in the ADC\_SR register is set when the analog voltage converted by the ADC is above or below a reference voltage threshold defined by the higher/lower thresholds programmed into the ADC\_HTRH/L and ADC\_LTRH/L registers.

In scan mode, the channel where the analog watchdog is enabled is the one selected through the CHSEL[4:0] bits in the ADC\_CR3 register.

**Note:**

*The threshold values must be set/changed before starting the conversion.*

*In scan mode a watchdog flag rise occurring during a conversion does not stop the next conversions of the sequence. This avoids having to start the ADC again every time an analog watchdog is enabled.*

*The action related to the analog watchdog is not necessarily linked to the other tasks using the other ADC channels.*

### 14.3.8 Interrupts

An interrupt can be generated when an EOC event occurs (end of conversion in single mode or end of last scan conversion in scan mode) when the analog watchdog status bit (AWD) is set in the ADC\_SR register or, in scan mode, when a DMA request is not serviced and a new conversion is completed.

Individual interrupt lines are available as well as a common interrupt line. The following table summarizes the possible flag/interrupt configuration.

**Table 45. Flag/interrupt configuration**

Flag	Description	Interrupt generation
OVER	Rises when a new converted data is ready and the previous one is not yet read by the DMA (scan mode)	yes
AWD	Rises when the converted data is outside the reference voltage threshold	yes
EOC	Rises when an end of conversion occurs (single mode) Rises when the end of conversion of last channel of the sequence occurs (scan mode)	yes

### 14.3.9 Channel selection (Scan mode)

This mode works automatically in continuous mode. It can also be used in single mode by resetting the DMAOFF bit in the ADC\_SQR1 register.

In scan mode, the selected channel sequence is taken into account (like in single mode) and the selected channels are converted and transferred through DMA to memory (DMA should be properly programmed).

If the single mode is selected, the ADC converts the selected channel sequence and it is stopped after the last channel conversion. For the next selected channel sequence, a scan conversion is necessary to restart the ADC. An interrupt (EOC) can be generated after the end of the channel sequence conversion.

Three registers are available to select the channel or the sequence of channels used for conversion in Scan mode: ADC\_SQR1, ADC\_SQR2 and ADC\_SQR3.

**Note:**

*This feature must be used in conjunction with the DMA controller.*

*The channel selection must be performed after enabling the ADC peripheral (by setting the ADON bit in the ADC\_CR1 register).*

A single conversion is initiated on each channel of a sequence. After the end of conversion, the next channel of the sequence is automatically converted.

The channel selection is performed using the ADC\_SQRx registers. A logic 1 in the position 'n' of these registers means that channel 'n' is in the list of channels to be converted.

For example, *Figure 44* below shows how to select the sequence: 0; 2; 6; 8; 10; 14; 18.

**Figure 44. Sequence example**

ADC_SQR1	ADC_SQR2	ADC_SQR3	ADC_SQR4
0 0 0 0 0 0 0 0	0 0 0 0 0 1 0 0	0 1 0 0 0 1 0 1	0 1 0 0 0 1 0 1
27 <sup>1</sup> 26 <sup>1</sup> 25 <sup>1</sup> 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 V <sub>REFIN</sub>			

1. Available on medium+ and high density devices only.

If the CONT bit in the ADC\_CR1 register is set (continuous mode selected), the conversion does not stop at channel 'n' but restarts from the first channel of the sequence. After each conversion, a DMA ADC request is sent and the DMA controller is used to transfer the converted data to the RAM.

*Note:* *The ADC\_SQRx registers must not be changed during a conversion.*

#### 14.3.10 Data integrity

An 8-bit shadow register is used to store the LSB data when the MSB is read. The LSB data is read from the shadow register.

This guarantees the data consistency if a new data from the ADC is coming between the MSB and LSB data read. Consequently, the software must read the MSB before reading the LSB to be sure that both LSB and MSB are related to the same data.

*Note:* *An MSB read operation must always be followed by an LSB read operation.*

#### 14.3.11 DMA transfer

When a scan conversion is performed, a DMA ADC request is sent as soon as an end of conversion is detected, to signal that a data is ready to be transferred.

If a new conversion is complete and the DMA has not completed the transfer of previous conversion data, the overrun flag is risen and an interrupt is generated (if enabled).

The DMA transfer can be disabled by setting the DMAOFF bit in the ADC\_SQR1 register.

#### 14.3.12 Configurable resolution

It is possible to reduce the conversion time by reducing the ADC resolution.

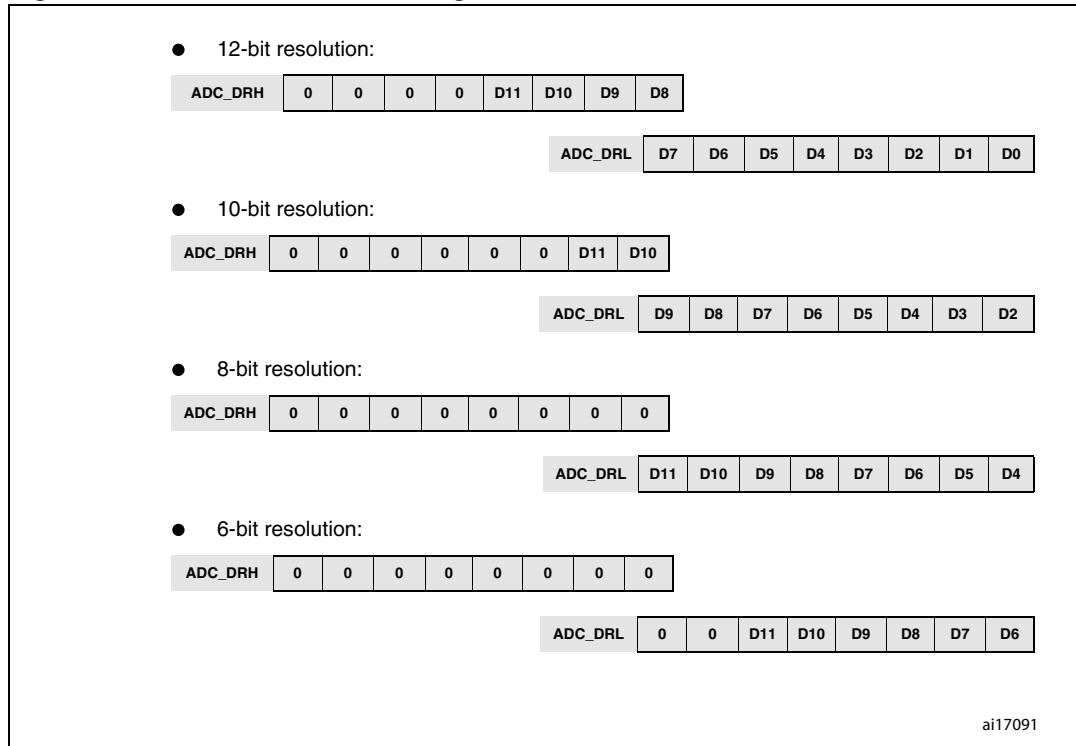
The RES[1:0] bits in the ADC\_CR1 register are used to configure the resolution to 6, 8, 10 or 12 bits. The converted data is received from the ADC through its serial output.

When the desired number of bits is achieved, a new conversion can start immediately.

*Note:* *The resolution bits can be changed only when no ADC conversion is ongoing.*

#### 14.3.13 Data alignment

The converted data stored after conversion are right aligned and their configuration depends on the selected resolution, as shown in the following figure:

**Figure 45. Resolution and data alignment**

#### 14.3.14 Programmable sampling time

The ADC input voltage is sampled during a number of cycles which is selected using:

- the SMP1[2:0] bits in the ADC\_CR2 register for the first 24 channels
- and the SMP2[2:0] bits in the ADC\_CR3 register for channels 24 to 27 (or channel 24 in medium density devices), V<sub>REFINT</sub> and TS.

The following table shows the allowed values of sampling cycles.

**Table 46. Authorized sampling cycles**

Bit configuration		Sampling cycles
SMP1[2:0] for channels 0 to 23 & SMP2[2:0] for channels 24 to 27 (or for channel 24 in medium density devices) V <sub>REFINT</sub> and TS	000	4
	001	9
	010	16
	011	24
	100	48
	101	96
	110	192
	111	384

**Note:** A different sampling time can be selected for slow channels and for fast channels.

The sampling time must not be changed during a conversion.

### 14.3.15 Schmitt trigger disabling

The Schmitt trigger can be disabled to reduce the consumption on some pins used as analog inputs.

As there are 2 internal channels, only 25 Schmitt triggers can be disabled on medium density devices and 28 Schmitt triggers on medium+ and high density devices.

*Note:* In order to disable the Schmitt trigger on the ADC channels which are shared with the comparators inputs, it is also required to disable the Schmitt trigger through the COMP\_CSR4 and/or COMP\_CSR5 registers.

If the control of the I/O using the Routing interface is enabled (with the corresponding bit set in the RI\_IOCMRn register), the setting of the STE bit in the COMP\_CSR1 register takes priority over the setting of the ADC\_TRIGRn registers.

### 14.3.16 Temperature sensor

The temperature sensor is not available on STM8L05xx value line devices.

The temperature sensor can be used to measure the internal temperature of the device. It is internally connected to the ADC TS (temperature sensor) input channel that is used to convert the sensor output voltage into a digital value.

*Note:* When it is not used, this sensor can be put in power-down mode.

The TSON bit in the ADC\_TRIGR1 register must be set to enable the internal ADC TS channel connection. This connection must be enabled only if the temperature sensor conversion is required.

The internal temperature sensor can also be used to detect temperature variations. The output voltage of the temperature sensor is factory measured at high temperature and the result of the ADC conversion is stored in a specific data address: the TS\_Factory\_CONV\_V90 byte represents the LSB of the V90 12-bit ADC conversion result while the MSB have a fixed value: 0x3.

To reduce the temperature sensor error, the user can measure it at ambient temperature (25°C) to redefine more accurately the average slope (avg\_slope) and the offset.

#### How to read the temperature

To read the temperature from the sensor, use the following procedure:

1. Select the ADC1 TS (temperature sensor) input channel.
2. Select a sample time of 10 µs.
3. Set the TSON bit in the ADC\_TRIGR1 register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting  $V_{SENSE}$  data in the ADC data register (ADC\_DRx).
6. Calculate the temperature using the following formulae:

$$T[^\circ K] = \frac{V_{SENSE}}{\text{AvgSlope}}$$

$$T[^\circ\text{C}] = \frac{V_{\text{SENSE}}}{\text{AvgSlope}} - 273.15$$

Where

Avg\_Slope = estimation of the average slope of the “Temperature vs.  $V_{\text{SENSE}}$ ” curve (given in V/ $^\circ\text{K}$ ).

Refer to the Electrical characteristics section for the Avg\_Slope value.

*Note:* When the sensor wakes up from power-down mode, a stabilization time is required before a correct voltage can be output.

After power-on, the ADC also needs a stabilization time. To minimize this delay, the ADON and TSON bits should be set at the same time.

#### 14.3.17 Internal reference voltage conversion

The internal reference voltage is internally connected to the  $V_{\text{REFINT}}$  channel. This analog input channel is used to convert the internal reference voltage into a digital value.

The VREFINTON bit in the ADC\_TRIGR1 register must be set to enable the internal reference voltage. This reference voltage must be enabled only if its conversion is required.

The internal reference voltage is factory measured and the result of the ADC conversion is stored in a specific data address: the VREFINT\_Factory\_CONV byte represents the LSB of the VREFINT 12-bit ADC conversion result while the MSB have a fixed value: 0x6.

### 14.4 ADC low power modes

**Table 47. Behavior in low power modes**

Mode	Description
Wait/ Low power wait	ADC interface is active. Interrupt events cause the device to exit from Wait or Low power wait mode.
Halt/ Active-halt	ADC interface is not active.

### 14.5 ADC interrupts

**Table 48. Interrupt requests**

Interrupt event	Event flag	Enable control bit	Exit from Wait / Low power wait	Exit from Halt / Active-halt
End of conversion flag	EOC	EOCIE	Yes	No
Analog watchdog flag	AWD	AWDIE	Yes	No
Overrun flag	OVER	OVERIE	Yes	No

## 14.6 ADC registers

### 14.6.1 ADC configuration register 1 (ADC\_CR1)

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
OVERIE	RES[1:0]		AWDIE	EOCIE	CONT	START	ADON
rw	rw		rw	rw	rw	rw	rw

Bit 7 **OVERIE**: Overrun interrupt enable

This bit is set and cleared by software. If set it enables the interrupt generated by an overrun event.

- 0: Overrun interrupt disabled
- 1: Overrun interrupt enabled

Bits 6:5 **RES[1:0]**: Configurable resolution

These bits are set and cleared by software. These bits are used to configure the ADC resolution.

- 00: 12-bit resolution
- 01: 10-bit resolution
- 10: 8-bit resolution
- 11: 6-bit resolution

Bit 4 **AWDIE**: Analog watchdog interrupt enable

This bit is set and cleared by software. If set it enables the interrupt generated by the analog watchdog.

- 0: Analog watchdog interrupts disabled
- 1: Analog watchdog interrupts enabled

Bit 3 **EOCIE**: Interrupt enable for EOC

This bit is set and cleared by software. It enables the interrupt at the end of conversion.

- 0: EOC interrupt disabled
- 1: EOC interrupt enabled

Bit 2 **CONT**: Continuous conversion

This bit is set and cleared by software. If set, conversion takes place continuously till this bit is reset.

- 0: Single conversion mode
- 1: Continuous conversion mode

Bit1 **START**: Conversion start

This bit is set by software and cleared by hardware.

If set, it starts a conversion (if enabled). It is automatically reset by hardware after one ADC clock cycle.

*Note: If this bit is set during a conversion, it will not be considered.*

Bit 0 **ADON**: A/D converter ON / OFF

This bit is set and reset by software. It wakes up the ADC from Power down mode.

- 0: ADC disabled (Power-down mode)
- 1: ADC enabled (wakeup from Power-down mode)

### 14.6.2 ADC configuration register 2 (ADC\_CR2)

Address offset: 0x01

Reset value: 0x00

7	6	5	4	3	2	1	0				
PRESC	TRIG_EDGE1	TRIG_EDGE0	EXTSEL1	EXTSEL0	SMTP1[2:0]						
rw	rw	rw	rw	r	rw	rw	rw				

Bit 7 **PRESC**: Clock prescaler

This bit is set and cleared by software. If set, it divides the ADC clock frequency by 2.

- 0:  $f_{(ADC\_CLK)} = CK$
- 1:  $f_{(ADC\_CLK)} = CK/2$

Bits 6:5 **TRIG\_EDGE[1:0]**: Active edge for external triggers

These bits are set and cleared by software. They select the active edges for external triggers.

- 00: Reserved
- 01: Rising edge sensitive
- 10: Falling edge sensitive
- 11: Both rising and falling edge sensitive

Bits 4:3 **EXTSEL[1:0]**: External event selection

These two bits select the software start or one of 3 external events that can trigger a conversion.

- 00: Triggers disabled, software start enabled.
- 01: Trigger 1 enabled
- 10: Trigger 2 enabled
- 11: Trigger 3 enabled

Bits 2:0 **SMTP1[2:0]**: Sampling time selection

These bits are set/reset by software. They are used to select one of the following sampling times for the first 24 channels.

- 000: 4 ADC clock cycles
- 001: 9 ADC clock cycles
- 010: 16 ADC clock cycles
- 011: 24 ADC clock cycles
- 100: 48 ADC clock cycles
- 101: 96 ADC clock cycles
- 110: 192 ADC clock cycles
- 111: 384 ADC clock cycles

### 14.6.3 ADC configuration register 3 (ADC\_CR3)

Address offset: 0x02

Reset value: 0x1F

7	6	5	4	3	2	1	0
SMTP2[2:0]				CHSEL[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw

#### Bit 7:5 **SMTP2[2:0]: Sampling time selection**

These bits are set/reset by software. They are used to select one of the following sampling times for channels 24, V<sub>REFINT</sub> and TS.

- 000: 4 ADC clock cycles
- 001: 9 ADC clock cycles
- 010: 16 ADC clock cycles
- 011: 24 ADC clock cycles
- 100: 48 ADC clock cycles
- 101: 96 ADC clock cycles
- 110: 192 ADC clock cycles
- 111: 384 ADC clock cycles

#### Bit 4:0 **CHSEL[4:0]: Channel selection**

These bits are set and cleared by software. They are used to select the channel to be checked by the analog watchdog.

- 00000: ADC channel 0 is selected
- 00001: ADC channel 1 is selected
- ...
- 10111: ADC channel 23 is selected
- 11000: ADC channel 24 is selected
- 11001: ADC channel 25 is selected <sup>(1)</sup>
- 11010: ADC channel 26 is selected <sup>(1)</sup>
- 11011: ADC channel 27 is selected <sup>(1)</sup>
- 11100: ADC channel V<sub>REFINT</sub> is selected
- 11101: ADC channel TS is selected<sup>(2)</sup>

1. This configuration is “reserved” in medium density devices.
2. This configuration is “reserved” in STM8L05xx value line devices.

#### 14.6.4 ADC status register (ADC\_SR)

Address offset: 0x03

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved					OVER	AWD	EOC
					rw_0	rw_0	rw_0

Bits 7:3 Reserved, forced by hardware to 0.

**Bit 2 OVER:** Overrun flag

This bit is reset by software writing 0 to it or by hardware when the ADC is put in power-down mode. It is set by hardware when, after a conversion, a second conversion has completed and the DMA has not read the first conversion value.

It cannot be set by software.

0: No overrun occurred

1: Overrun occurred

**Bit 1 AWD:** Analog watchdog flag

This bit is reset by software writing 0 to it or by hardware when the ADC is in power-down mode. It is set when the analog voltage converted by the ADC is above or below the reference voltage thresholds defined by the lower/higher thresholds in the ADC\_xTRx registers.

It cannot be set by software.

0: No analog watchdog event occurred

1: Analog watchdog event occurred

**Bit 0 EOC:** End of conversion

This bit is set by hardware at the end of conversion. It is cleared by software by writing '0' to it or by reading the LSB of the converted data or when the ADC is put in power-down mode.

In case of scan conversion, this bit is set at the end of conversion of the last channel of the sequence.

It cannot be set by software.

0: Conversion not complete

1: Conversion complete

### 14.6.5 ADC data register high (ADC\_DRH)

Address offset: 0x04

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved				CONV_DATA[11:8]			
				r	r	r	r

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

Bits 3:0 **CONV\_DATA[11:8]**: Data bits high

These bits are set/reset by hardware and are read only. They contain the 4 MS bits of the converted data. The converted voltage data bits are right aligned and their configuration depends on the programmed resolution, as described below:

12-bit resolution: bits 3:0 = CONV\_DATA[11:8]

10-bit resolution: bits 3:2 = reserved; Bits1:0 : CONV\_DATA[9:8]

8-bit resolution: bits 3:0 = reserved

6-bit resolution: bits 3:0 = reserved

### 14.6.6 ADC data register low (ADC\_DRL)

Address offset: 0x05

Reset value: 0x00

7	6	5	4	3	2	1	0
CONV_DATA[7:0]							
r	r	r	r	r	r	r	r

Bits 7:0 **Data[7:0]**: Data bits low

These bits are set/reset by hardware and are read only. They contain the 8 LS bits of the converted data. The converted voltage data bits are right aligned and their configuration depends on the programmed resolution, as described below:

12-bit resolution: Bits 7:0 = CONV\_DATA[7:0]

10-bit resolution: Bits 7:0 = CONV\_DATA[7:0]

8-bit resolution: Bits 7:0 = CONV\_DATA[7:0]

6-bit resolution: Bits 7:6 = reserved; bits 5: 0 = CONV\_DATA[5:0]

### 14.6.7 ADC high threshold register high (ADC\_HTRH)

Address offset: 0x06

Reset value: 0x0F

7	6	5	4	3	2	1	0
Reserved				HT[11:8]			
				rw	rw	rw	rw

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

Bits 3:0 **HT[11: 8]**: Analog watchdog higher threshold high

These bits are set/reset by software. They define the MSB of the higher threshold for the analog watchdog.

### 14.6.8 ADC high threshold register low (ADC\_HTRL)

Address offset: 0x07

Reset value: 0xFF

7	6	5	4	3	2	1	0
HT[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **HT[7:0]**: Analog watchdog higher threshold low

These bits are set/reset by software. They define the LSB of the higher threshold for the analog watchdog.

### 14.6.9 ADC low threshold register high (ADC\_LTRH)

Address offset: 0x08

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved				LT[11:8]			
				rw	rw	rw	rw

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

Bits 3:0 **LT[11: 8]**: Analog watchdog lower threshold high

These bits are set/reset by software. They define the MSB of the lower threshold for the analog watchdog.

### 14.6.10 ADC low threshold register low (ADC\_LTRL)

Address offset: 0x09

Reset value: 0x00

7	6	5	4	3	2	1	0
LT[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **LT[7:0]** Analog watchdog lower threshold low

These bits are set/reset by software. They define the LSB of the lower threshold for analog watchdog.

**Note:** *The reference voltage threshold data bits are right aligned and their configuration depends on the programmed resolution, as described below:*

12-bit resolution

ADC\_HTRH & ADC\_LTRH Bits 7:4 = reserved ; Bits 3:0 = HT[11:8] or LT[11:8]

ADC\_HTRL & ADC\_LTRL Bits 7:0 = HT[7:0] or LT[7:0]

10-bit resolution

ADC\_HTRH & ADC\_LTRH Bits 7:2 = reserved ; Bits 1:0 = HT[9:8] or LT[9:8]

ADC\_HTRL & ADC\_LTRL Bits 7:0 = HT[7:0] or LT[7:0]

8-bit resolution

ADC\_HTRH & ADC\_LTRH Bits 7:0 = reserved

ADC\_HTRL & ADC\_LTRL Bits 7:0 = HT[7:0] or LT[7:0]

6-bit resolution

ADC\_HTRH & ADC\_LTRH Bits 7:0 = reserved

ADC\_HTRL & ADC\_LTRL Bits 7:6 = reserved ; Bits 5:0 = HT[5:0] or LT[5:0]

### 14.6.11 ADC channel sequence 1 register (ADC\_SQR1)

Address offset: 0x0A

Reset value: 0x00

7	6	5	4	3	2	1	0
DMAOFF	Reserved	CHSEL_STS <sup>(2)</sup>	CHSEL_SVREFINT	CHSEL_S[27:24] <sup>(1)</sup>			
rw	-	rw	rw	rw	rw	rw	rw

Bits 7 **DMAOFF**: DMA disable for a single conversion

0: DMA Enabled

1: DMA Disabled

Bit 6 Reserved, forced by hardware to 0.

Bit 5 **CHSEL\_STS**: Selection of channel TS for scan

These bits are set/reset by software. Channel TS must be kept reset.

A value '1' in the CHSEL\_Sx bit means that channel x is assigned in the scan sequence.

Bit 4 **CHSEL\_SVREFINT**: Selection of channel  $V_{REFINT}$  for scan

These bits are set/reset by software. Channels  $V_{REFINT}$  must be kept reset.

A value '1' in the CHSEL\_Sx bit means that channel x is assigned in the scan sequence.

Bits 3:0<sup>(1)</sup> **CHSEL\_S[27:24]**: Selection of channels 24 to 27 for scan <sup>(1)</sup>

These bits are set/reset by software.

A value '1' in the CHSEL\_Sx bit means that channel x is assigned in the scan sequence.

1. On medium density devices, bits 3:1 are reserved and bit 0 is the CHSEL\_S24 bit (selection of channel 24 for scan).
2. On STM8L05xx value line devices, bit 5 is reserved and must be kept cleared.

*Note:* This register must be modified after ADC is enabled by ADON bit in ADC\_CR1 register.

#### 14.6.12 ADC channel sequence register 2 (ADC\_SQR2)

Address offset: 0x0B

Reset value: 0x00

7	6	5	4	3	2	1	0
CHSEL_S[23:16]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CHSEL\_S[23:16]**: Selection of channels 16 to 23 for scan

These bits are set/reset by software.

A value '1' in the CHSEL\_Sx bit means that channel x is assigned in the scan sequence.

*Note:* This register must be modified after ADC is enabled by ADON bit in ADC\_CR1 register.

#### 14.6.13 ADC channel select scan 3 (ADC\_SQR3)

Address offset: 0x0C

Reset value: 0x00

7	6	5	4	3	2	1	0
CHSEL_S[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CHSEL\_S[15:8]**: Selection of channels 8 to 15 for scan

These bits are set/reset by software.

A value '1' in the CHSEL\_Sx bit means that channel x is assigned in the scan sequence.

*Note:* This register must be modified after ADC is enabled by ADON bit in ADC\_CR1 register.

#### 14.6.14 ADC channel select scan 4 (ADC\_SQR4)

Address offset: 0x0D

Reset value: 0x00

7	6	5	4	3	2	1	0
CHSEL_S[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CHSEL\_S[7:0]**: Selection of channels 0 to 7 for scan

These bits are set/reset by software.

A value ‘1’ in the CHSEL\_Sx bit means that channel x is assigned in the scan sequence.

*Note:* This register must be modified after ADC is enabled by ADON bit in ADC\_CR1 register.

#### 14.6.15 ADC trigger disable 1 (ADC\_TRIGR1)

Address offset: 0x0E

Reset value: 0x00

7	6	5	4	3	2	1	0	
Reserved		TS0N <sup>(2)</sup>	VREFINTON	TRIG[27:24]				
		rw	rw	rw	rw	rw	rw	

Bits 7:6 Reserved, forced by hardware to 0.

Bit 5 **TS0N**: Temperature sensor internal reference voltage enable

This bit is set/reset by software.

0: Temperature sensor internal reference voltage disabled

1: Temperature sensor internal reference voltage enabled

Bit 4 **VREFINTON**: Internal reference voltage enable

This bit is set/reset by software.

0: Internal reference voltage disabled

1: Internal reference voltage enabled

Bits 3:0<sup>(1)</sup> **TRIG[27:24]**: Channels 24 to 27 schmitt trigger disable

These bits are set/reset by software.

A value ‘1’ in the TRIGx bit means that the Schmitt trigger corresponding to channel x is disabled.

1. On medium density devices, bits 3:1 are reserved and bit 0 is the TRIG24 bit (channel 24 schmitt trigger disabled).
2. On STM8L05xx value line devices, bit 5 is reserved and must be kept cleared.

#### 14.6.16 ADC trigger disable 2 (ADC\_TRIGR2)

Address offset: 0x0F

Reset value: 0x00

7	6	5	4	3	2	1	0
TRIG[23:16]							
				rw	rw	rw	rw

Bits 7:0 **TRIG[23:16]**: Channels 16 to 23 schmitt trigger disable

These bits are set/reset by software.

A value '1' in the TRIGx bit means that the Schmitt trigger corresponding to channel x is disabled.

#### 14.6.17 ADC trigger disable 3 (ADC\_TRIGR3)

Address offset: 0x10

Reset value: 0x00

7	6	5	4	3	2	1	0
TRIG[15:8]							
				rw	rw	rw	rw

Bits 7:0 **TRIG[15:8]**: Channels 8 to 15 schmitt trigger disable

These bits are set/reset by software.

A value '1' in the TRIGx bit means that the Schmitt trigger corresponding to channel x is disabled.

#### 14.6.18 ADC trigger disable 4 (ADC\_TRIGR4)

Address offset: 0x11

Reset value: 0x00

7	6	5	4	3	2	1	0
TRIG(7:0)							
				rw	rw	rw	rw

Bits 7:0 **TRIG[7:0]**: Channels 0 to 7 schmitt trigger disable

These bits are set/reset by software.

A value '1' in the TRIGx bit means that the schmitt trigger corresponding to channel x is disabled

### 14.6.19 ADC register map and reset values

**Table 49. Register map**

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	ADC_CR1 Reset value	OVERIE 0	RES[1:0] 00		AWDIE 0	EOCIE 0	CONT 0	START 0	ADON 0
0x01	ADC_CR2 Reset value	PRESC 0	TRIG_ED GE1 0	TRIG_ED GE0 0	EXTSEL1 0	EXTSEL0 0	SMPT1_2 0	SMPT1_1 0	SMPT1_0 0
0x02	ADC_CR3 Reset value	SMTP2_2 0	SMTP2_1 0	SMTP2_0 0	CHSEL4 1	CHSEL3 1	CHSEL2 1	CHSEL1 1	CHSEL0 1
0x03	ADC_SR Reset value	-	-	-	-	-	OVER 0	AWD 0	EOC 0
0x04	ADC_DRH Reset value	-	-	-	-	CONV_DATA[11:8] 0			
0x05	ADC_DRL Reset value	CONV_DATA[7:0] 0							
0x06	ADC_HTRH Reset value	-	-	-	-	HT11 1	HT10 1	HT9 1	HT8 1
0x07	ADC_HTRL Reset value	HT7 1	HT6 1	HT5 1	HT4 1	HT3 1	HT2 1	HT1 1	HT0 1
0x08	ADC_LTRH Reset value	-	-	-	-	LT11 0	LT10 0	LT9 0	LT8 0
0x09	ADC_LTRL Reset value	LT7 0	LT6 0	LT5 0	LT4 0	LT3 0	LT2 0	LT1 0	LT0 0
0x0A	ADC_SQR1 Reset value	DMAOFF 0	-	CHSEL_S TS <sup>(1)</sup> 0	CHSEL_S VREFINT 0	CHSEL_S 27 <sup>(2)</sup> 0	CHSEL_S 26 <sup>(2)</sup> 0	CHSEL_S 25 <sup>(2)</sup> 0	CHSEL_S 24 0
0x0B	ADC_SQR2 Reset value	CHSEL_S[23:16] 0							
0x0C	ADC_SQR3 Reset value	CHSEL_S[15:8] 0							
0x0D	ADC_SQR4 Reset value	CHSEL_S[7:0] 0							
0x0E	ADC_TRIGR1 Reset value	-	-	TS0N <sup>(1)</sup> 0	VREFINTO N 0	-TRIG27 <sup>(2)</sup> 0	-TRIG26 <sup>(2)</sup> 0	TRIG25 <sup>(2)</sup> 0	TRIG24 0
0x0F	ADC_TRIGR2 Reset value	TRIG23 0	TRIG22 0	TRIG21 0	TRIG20 0	TRIG19 0	TRIG18 0	TRIG17 0	TRIG16 0
0x10	ADC_TRIGR3 Reset value	TRIG15 0	TRIG14 0	TRIG13 0	TRIG12 0	TRIG11 0	TRIG10 0	TRIG9 0	TRIG8 0
0x11	ADC_TRIGR4 Reset value	TRIG7 0	TRIG6 0	TRIG5 0	TRIG4 0	TRIG3 0	TRIG2 0	TRIG1 0	TRIG0 0

1. This bit is reserved in STM8L05xx value line devices.

2. This bit is reserved in medium density devices.

## 15 Digital-to-analog converter (DAC)

This section applies to medium density STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L15xx devices and high density STM8L15xx/STM8L16xx devices, unless otherwise specified.

Digital-to-analog converter (DAC) is not available on STM8L05xx value line devices.

### 15.1 DAC introduction

The DAC module is a 12-bit voltage output digital-to-analog converter. The DAC can be configured in 8-bit or 12-bit mode and can be used in conjunction with the DMA controller. In 12-bit mode, the data can be left or right aligned. The DAC has one output channel in the medium density devices and two output channels (each output channel with its own converter) in medium+ and high density devices. An input reference pin  $V_{REF+}$  is available for a better resolution.

In dual DAC channel mode (available on medium+ and high density devices only), conversions can be performed either independently or simultaneously when both channels are grouped together for synchronous update operations.

### 15.2 DAC main features

- 8-bit or 12-bit monotonic output
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- DMA capability
- External triggers for conversion
- Input voltage reference  $V_{REF+}$
- Noise-wave generation (medium+ and high density devices only)
- Triangular-wave generation (medium+ and high density devices only)
- Dual DAC channel for independent or simultaneous conversions (medium+ and high density devices only)

The block diagram of a DAC channel is shown in [Figure 46](#).

Figure 46. DAC channel block diagram (medium density devices)

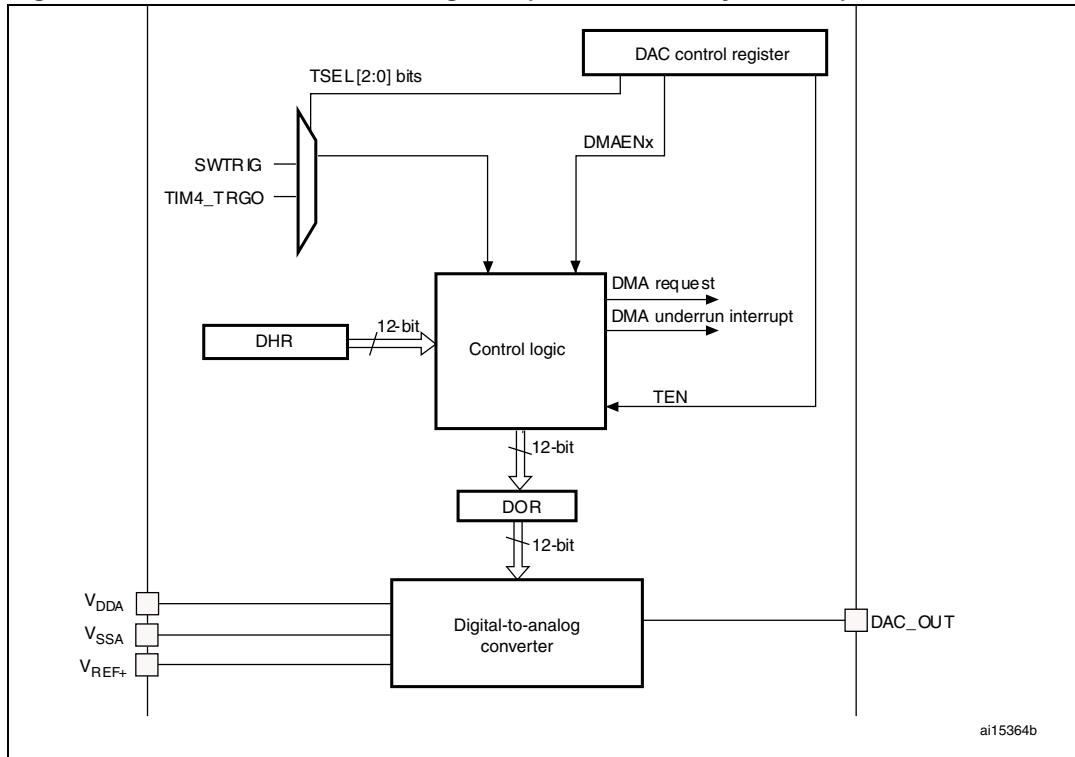
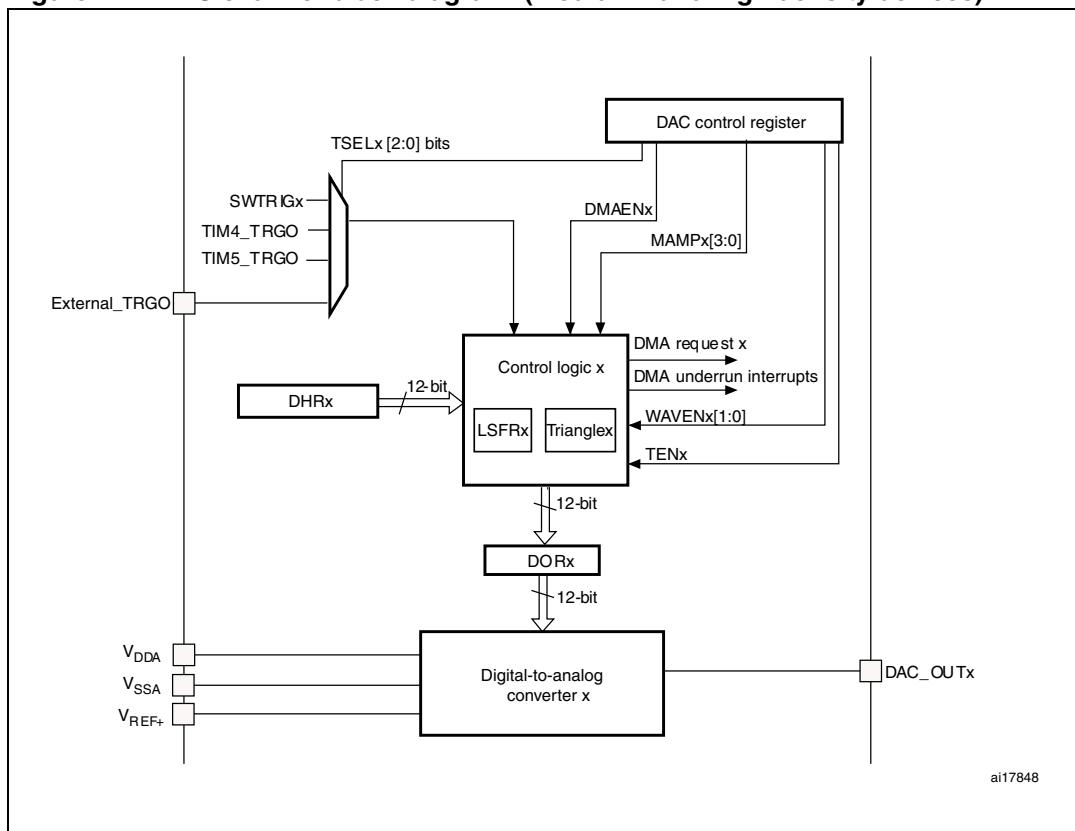


Figure 47. DAC channel block diagram (medium+ and high density devices)



## 15.3 DAC functional description

### 15.3.1 DAC channel x enable

Digital-to-analog conversions can be performed only if the DAC channels have been powered on by setting the EN bit in the DAC\_CHxCR1 register. The DAC channels are then enabled after a  $t_{WAKEUP}$  startup time (refer to the product datasheet).

**Note:** Please make sure the related GPIO is configured in floating input mode with deactivated Schmitt trigger to avoid parasitic current consumption in the I/O structure. Refer to [Section 10.7: Input mode details on page 119](#) for information on analog input.

### 15.3.2 DAC output buffer enable

The DAC integrates one output buffer (on medium density devices) or two output buffers (on medium+ and high density devices) that can be used to reduce the output impedance and to drive external loads directly without having to add an external operational amplifier. It can be enabled and disabled using the BOFF bit in the DAC\_CHxCR1 register.

### 15.3.3 DAC output switch configuration

The following table describes the DAC output switch configuration depending on devices and packages.

See also [Figure 23: Routing interface \(RI\) block diagram \(medium, medium+ and high density devices\)](#).

**Table 51. DAC output switch configuration (medium and medium+ and high density devices)**

Packages	Configuration on medium density devices	Configuration on medium+ and high density devices
64-pin and 80-pin packages	N/A	DAC_OUT1 is connected to PF0 DAC_OUT2 is connected to PF1
48-pin packages	DAC_OUT1 is connected to PF0	DAC_OUT1 is connected to PF0 DAC_OUT2 can be connected to PB4, PB5 or PB6 by closing the corresponding RI switches (see <a href="#">Figure 23: Routing interface (RI) block diagram (medium, medium+ and high density devices)</a> ).
28- and 32-pin packages	DAC_OUT1 can be connected to PB4, PB5 or PB6 by closing the corresponding RI switches (see <a href="#">Figure 23: Routing interface (RI) block diagram (medium, medium+ and high density devices)</a> ).	N/A

### 15.3.4 DAC data format

Depending on the selected configuration mode, the data has to be written in the specified register as described below:

- 8-bit right alignment: data to be loaded to DAC\_DHR8 [7:0] bits by converted data[11:4]
- 12-bit left alignment: data to be loaded to DAC\_LDHRH [7:0] bits by the converted data[11:4] and the DAC\_LDHLR[7:4] bits by the converted data [3:0]
- 12-bit right alignment: data to be loaded to DAC\_RDHRH [3:0] bits by the converted data[11:8] and the DAC\_RDLR[7:0] bits by the converted data [7:0]

### 15.3.5 DAC conversion sequence

To start a conversion, the content of the DAC\_xDHRx and DAC\_DHR8 registers is moved to the DAC\_CHxDORH/L (data output) registers. Depending on the TEN bit in the DAC\_CHxCR1 register, this transfer to DAC\_CHxDORH/L can be performed in two different manners:

- TEN = 0. The transfer is performed when:
  - DAC\_DHR8 is written for 8-bit conversion data in single mode
  - LSB of DAC\_xDHRL is written for 12-bit conversion data in single mode,
- TEN = 1. The transfer is performed when a trigger occurs. Two different triggers (TIM4\_TRGO or SWTRIG) in the medium density devices or three triggers (TIM4\_TRGO, TIM5\_TRGO or SWTRIG) can be selected through the TSEL bits in the DAC\_CHxCR1 register. Refer to [Section 15.3.7: DAC trigger selection](#).

### 15.3.6 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V<sub>REF+</sub>. The analog output voltages on the DAC\_OUTx pin are determined by the following equation:

$$\text{DAC\_OUT } x = V_{\text{REF}} \times \frac{\text{DAC\_CHxDORH/L}}{4096}$$

### 15.3.7 DAC trigger selection

If the TEN bit is set in the DAC\_CHxCR1 register, the following events can trigger a conversion:

- a software trigger (SWTRIG)
- one external event (TIM4\_TRGO: Timer 4 counter channel output) in medium density devices
- three external events (TIM4\_TRGO, TIM5\_TRGO or external pin) in medium + and high-density devices.

The TSEL[2:0] bits can be used to determine which out of the 2 or 4 possible sources will trigger a conversion (SWTRIG or TIM4\_TRGO in medium density devices and SWTRIG, TIM4\_TRGO, TIM5\_TRGO or external pin in medium+ and high density devices):

- If the software trigger (SWTRIG) is selected, the conversion starts once the SWTRIGx bit is set in the DAC\_SWTRIGR. This bit is reset by hardware once the DAC\_DORx registers are loaded with the DAC\_xDHRx values.
- If TIM4\_TRGO, TIM5\_TRGO or external pin is selected, the data transfer is synchronized by a two-stage rising edge synchronizer

*Note:* TSEL[2:0] bits cannot be changed when the EN bit is set in the DAC\_CHxCR1 register.

Refer to the following table.

**Table 52. Sources of conversion trigger (medium, medium+ and high density devices)**

Source	Type	TSEL[2:0]	Medium density	Medium+ and High density
Timer 4 TRGO event	Internal signal from on-chip timers	000	X	X
Timer 5 TRGO event		001		X
PA4 GPIO	External pin	010		x
NA (reserved)		011		
		100		
		101		
		110		
SWTRIG	Software control bit	111	X	X

### 15.3.8 DAC DMA request

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

A DAC channel DMA request is generated when an external trigger occurs while the TEN and DMAENx bits are set. The DAC channel DMA request remains set until a DAC channel DMA Acknowledge comes from the DMA controller. The DAC channel DMA request indicates that DACx\_DHRx registers have been transferred to the DAC\_CHxDORH/L registers.

DAC channel DMA request is not kept in a request queue, so if a second external trigger arrives before the DMA receives the acknowledge for the first external trigger, no new request is serviced and no error is reported.

### 15.3.9 DAC DMA underrun interrupt

A DAC DMA underrun interrupt is generated in DMA mode when the next trigger event occurs while the previous DMA request is still pending. This may happen, for instance, when the DAC trigger frequency is higher than the DMA request servicing process.

For each DAC channel, DMA underrun interrupt can be enabled or disabled through the DMAUDRIE bits in DAC\_CHxCR2 register. If it occurs, it is signalled by the DMAUDR bits in the DAC\_SR register.

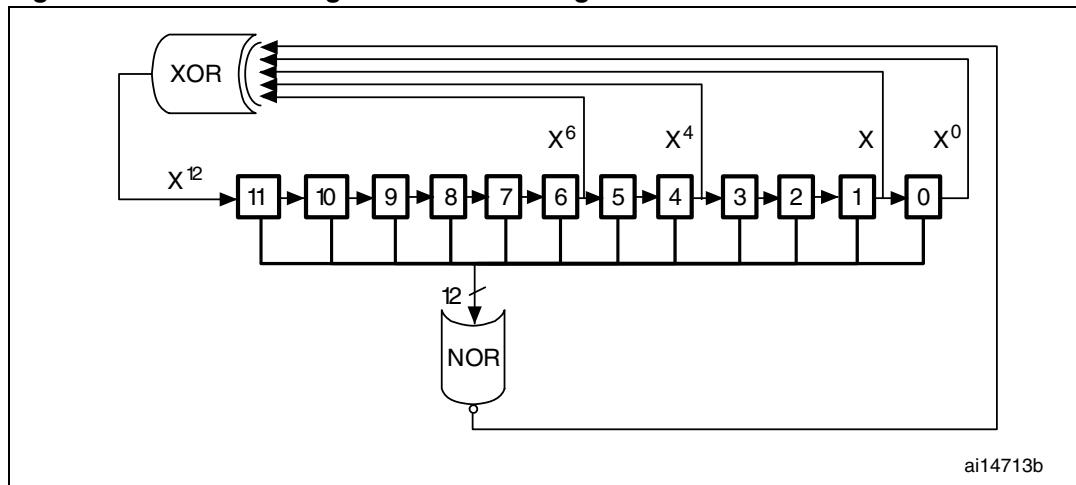
### 15.3.10 Noise generation

In order to generate a variable-amplitude pseudonoise, a linear feedback shift register (LFSRx) register is available. The DAC noise generation is enabled by setting DAC\_CHxCR1/WAVEN to “01” and TENx to 1 in the DAC\_CHxCR1 register. The preloaded value in LFSR is 0xAAA, and the LFSRx output is updated at each conversion. This register is updated three CPU clock cycles after each trigger event, following a specific calculation algorithm (see [Figure 48: DAC LFSR register calculation algorithm on page 222](#)).

If LFSRx is 0x000, a '1' is injected into it (antilock-up mechanism).

The LFSRx value, that may be masked partially or totally by means of the DAC\_CHxCR2/MAMP bits in the DAC\_CHxCR2, is added to the DAC\_DHRx (which can be DAC\_CHxRDHRH/L or DAC\_CHxLDHRH/L or DAC\_CHxDHR8 register depending on the selected data format) contents without overflow and this value is then stored into the DAC\_CHxDORH/L register (see the following figure).

**Figure 48. DAC LFSR register calculation algorithm**



It is possible to reset LFSRx wave generation by resetting the DAC\_CHxCR1/WAVEN[1:0] bits.

*Note:* The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC\_CHxCR1 register.

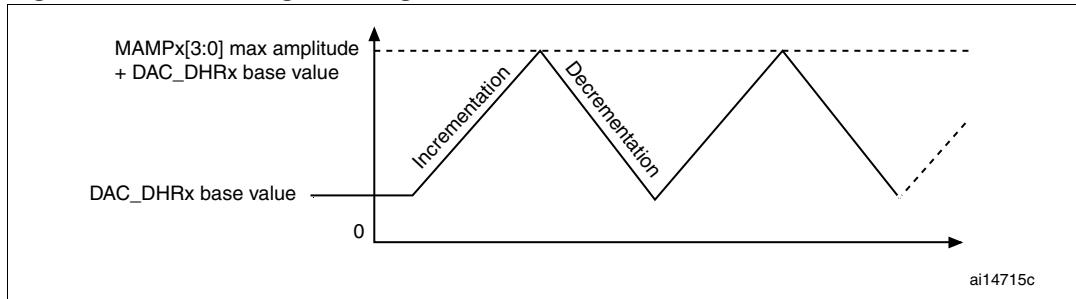
### 15.3.11 Triangle-wave generation

It's possible to add a small-amplitude triangular waveform on a DC or slowly varying signal.

DAC triangle-wave generation is selected DAC\_CH2CR1/TSELsetting DAC\_CHxCR1/WAVEN[1:0] to "10" and DAC\_CHxCR1/TEN bits to 1. The amplitude is configured through the DAC\_CHxCR2/MAMP[3:0] bits in the DAC\_CHxCR2 register. A 12-bit internal triangle counter is incremented three CPU clock cycles after each trigger event. The value of this counter is then added to the DAC\_DHRx (which can be DAC\_CHxRDHRH/L or DAC\_CHxLDHRH/L or DAC\_CHxDHR8 register depending on the selected data format) without overflow and the sum is stored into the DAC\_CHxDORH/L. The triangle counter is incremented while it is less than the maximum amplitude defined by the DAC\_CHxCR2/MAMP[3:0] bits (refer to DAC\_CHxCR2). Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on till WAVENx[1:0] = "10" and TENx = '1' ([Figure 49](#)).

It is possible to reset triangle wavex generation by resetting DAC\_CHxCR1/WAVEN.

*Note:* DAC\_CHxCR2/MAMP[3:0] bits cannot be changed when the DAC\_CHxCR1/EN bit is set.

**Figure 49. DAC triangle wave generation**

1. The DAC trigger must be enabled for triangle generation by setting the TENx bit in the DAC\_CHxCR1 register.
2. The DAC\_CH1CR2/MAMP[3:0] bits must be configured before enabling the DAC. Otherwise, they cannot be changed.

### 15.3.12 Dual DAC conversion

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented. A unique register access is then required to drive both DAC channels at the same time.

Nine possible conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DHRx registers if needed.

All modes (independent and simultaneous modes) are described in the paragraphs below.

#### Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits DAC\_CH1CR1/TEN and DAC\_CH2CR1/TEN
- Configure different trigger sources by setting different values in the DAC\_CH1CR1/TSEL[2:0] and DAC\_CH2CR1/TSEL[2:0] bits
- Load the dual DAC channel data into the desired DAC\_DHRx register (which can be DAC\_CHxRDHRH/L or DAC\_CHxLDHRH/L or DAC\_CHxDHR8 register depending on the selected data format).

When a DAC channel1 trigger arrives, the DAC\_DHR1 (which can be DAC\_CH1RDHRH/L or DAC\_CH1LDHRH/L or DAC\_CH1DHR8 register depending on the selected data format) register is transferred into DAC\_CH1DORH/L (three CPU clock cycles later).

When a DAC channel2 trigger arrives, the DAC\_DHR2(which can be DAC\_CH2RDHRH/L or DAC\_CH2LDHRH/L or DAC\_CH2DHR8 register depending on the selected data format) register is transferred into DAC\_CH2DORH/L (three CPU clock cycles later).

### Independent trigger with same LFSR generation

- Set the two DAC channel trigger enable bits DAC\_CH1CR1/TEN and DAC\_CH2CR1/TEN
- Configure different trigger sources by setting different values in the DAC\_CH1CR1/TSEL[2:0] and DAC\_CH2CR1/TSEL[2:0] bits
- Configure the two DAC channel WAVENx[1:0] bits as “01” and the same LFSR mask value in the DAC\_CHxCR2/MAMP[3:0] bits
- Load the dual DAC channel data into the desired DAC\_DHRx register (which can be DAC\_CHxRDHRH/L or DAC\_CHxLDHRH/L or DAC\_CHxDHR8 register depending on the selected data format)

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask/amplitude, is added to the DAC\_DHR1(which can be DAC\_CH1RDHRH/L or DAC\_CH1LDHRH/L or DAC\_CH1DHR8 register depending on the selected data format) register and the sum is transferred into DAC\_DOR1. Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DAC\_DHR2(which can be DAC\_CH2RDHRH/L or DAC\_CH2LDHRH/L or DAC\_CH2DHR8 register depending on the selected data format) register and the sum is transferred into DAC\_DOR2. Then, the LFSR2 counter is updated.

### Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits DAC\_CH1CR1/TEN and DAC\_CH2CR1/TEN
- Configure different trigger sources by setting different values in the DAC\_CH1CR1/TSEL[2:0] and DAC\_CH2CR1/TSEL[2:0] bits
- Configure the two DAC channel WAVENx[1:0] bits as “01” and set different LFSR masks values in the DAC\_CH1CR2/MAMP[3:0] and DAC\_CH2CR2/MAMP[3:0] bits
- Load the dual DAC channel data into the desired DHR register

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by DAC\_CH1CR2/MAMP[3:0], is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three CPU clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by DAC\_CH2CR2/MAMP[3:0], is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three CPU clock cycles later). Then the LFSR2 counter is updated.

### Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits DAC\_CH1CR1/TEN and DAC\_CH2CR1/TEN
- Configure different trigger sources by setting different values in the DAC\_CH1CR1/TSEL[2:0] and DAC\_CH2CR1/TSEL[2:0] bits
- Configure the two DAC channel WAVENx[1:0] bits as “1x” and set different maximum amplitude values in the DAC\_CH1CR2/MAMP[3:0] and DAC\_CH2CR2/MAMP[3:0] bits
- Load the dual DAC channel data into the desired DHR register

When a DAC channel1 trigger arrives, the 12-bit DAC channel1 triangle counter, with a triangle amplitude configured by DAC\_CH1CR2/MAMP[3:0], is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three CPU clock cycles later). The 12-bit DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by DAC\_CH2CR2/MAMP[3:0], is added to the DHR2 register part and the sum is transferred into DAC\_DOR2 (three CPU clock cycles later). The DAC channel2 triangle counter is then updated.

### **Simultaneous trigger without wave generation**

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits DAC\_CH1CR1/TEN and DAC\_CH2CR1/TEN
- Configure the same trigger source for both DAC channels by setting the same value in the DAC\_CH1CR1/TSEL[2:0] and DAC\_CH2CR1/TSEL[2:0] bits
- Load the dual DAC channel data to the desired DHR register

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC\_DOR1 and DAC\_DOR2, respectively (after three CPU clock cycles).

### **Simultaneous trigger with same LFSR generation**

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits DAC\_CH1CR1/TEN and DAC\_CH2CR1/TEN
- Configure the same trigger source for both DAC channels by setting the same value in the DAC\_CH1CR1/TSEL[2:0] and DAC\_CH2CR1/TSEL[2:0] bits
- Configure the two DAC channel WAVENx[1:0] bits as “01” and the same LFSR mask value in the DAC\_CH1CR2/MAMP[3:0] bits
- Load the dual DAC channel data to the desired DHR register

When a trigger arrives, the LFSR1 counter, with the mask configured by DAC\_CH1CR2/MAMP[3:0], is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three APB1 clock cycles later). The LFSR1 counter is then updated.

At the same time, the LFSR2 counter, with the mask configured by DAC\_CH2CR2/MAMP[3:0], is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three CPU clock cycles later). The LFSR2 counter is then updated.

### **Simultaneous trigger with different LFSR generation**

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits DAC\_CH1CR1/TEN and DAC\_CH2CR1/TEN
- Configure the same trigger source for both DAC channels by setting the same value in the DAC\_CH1CR1/TSEL[2:0] and DAC\_CH2CR1/TSEL[2:0] bits
- Configure the two DAC channel WAVENx[1:0] bits as “01” and set different LFSR masks values using the DAC\_CH1CR2/MAMP[3:0] and DAC\_CH2CR2/MAMP[3:0] bits
- Load the dual DAC channel data into the desired DHR register

When a trigger arrives, the LFSR1 counter, with the mask configured by DAC\_CH1CR2/MAMP[3:0], is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three CPU clock cycles later). The LFSR1 counter is then updated.

At the same time, the LFSR2 counter, with the mask configured by DAC\_CH2CR2/MAMP[3:0], is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three CPU clock cycles later). The LFSR2 counter is then updated.

### Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC channel trigger enable bits DAC\_CH1CR1/TEN and DAC\_CH2CR1/TEN
- Configure the same trigger source for both DAC channels by setting the same value in the DAC\_CH1CR1/TSEL[2:0] and DAC\_CH2CR1/TSEL[2:0] bits
- Configure the two DAC channel WAVENx[1:0] bits as “1x” and set different maximum amplitude values in the DAC\_CH1CR2/MAMP[3:0] and DAC\_CH2CR2/MAMP[3:0] bits
- Load the dual DAC channel data into the desired DHR register

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by DAC\_CH1CR2/MAMP[3:0], is added to the DHR1 register and the sum is transferred into DAC\_DOR1 (three CPU clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by DAC\_CH2CR2/MAMP[3:0], is added to the DHR2 register and the sum is transferred into DAC\_DOR2 (three CPU clock cycles later). Then the DAC channel2 triangle counter is updated.

### Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC\_DHR12RD, DAC\_DHR12LD or DAC\_DHR8RD)

In this configuration, one CPU clock cycle later, the DHR1 and DHR2 registers are transferred into DAC\_DOR1 and DAC\_DOR2, respectively.

## 15.4 DAC registers

### 15.4.1 DAC channel x control register 1 (DAC\_CHxCR1)

Address offset: 0x00 (channel 1) or 0x02 (channel 2 available on high devices only)

Reset value: 0x00

7	6	5	4	3	2	1	0
WAVEN[1:0] <sup>(1)</sup>		TSEL[2:0]			TEN	BOFF	EN
-	-	-	rw	rw	rw	rw	rw
-	-	-	rw	rw	rw	rw	rw

Bits 7:6 <sup>(1)</sup> **WAVEN[1:0]**: DAC channel x noise/triangle waveform generation enable.  
These bits are used only if TEN=1.

00: Wave generation disabled.

01: Noise generation enabled.

1x: Triangle generation enabled.

Bits 5:3 **TSEL[2:0]**: DAC channel x trigger selection.

These bits are only used if TEN=1.

000: TIM4\_TRGO (Timer 0 counter channel output) selected

001: Reserved

010: Reserved

001 <sup>(1)</sup>: TIM5\_TRGO selected

010 <sup>(1)</sup>: External trigger (PE4)

011: Reserved

100: Reserved

101: Reserved

110: Reserved

111: SWTRIG (Software trigger) selected

Bit 2 **TEN**: DAC channel trigger enable.

0: The data written into the data holding register (DHR) is transferred one CPU clock cycle later to the data output register (DORx).

1: The data transfer from the DHRx to the DORx is synchronized by the selected trigger.

Bit 1 **BOFF**: DAC channel output buffer disable.

0: DAC Channel output buffer enabled

1: DAC Channel output buffer disabled

Bit 0 **EN**: DAC channel enable.

0: DAC channel disabled

1: DAC channel enabled

1. Reserved on medium density devices

### 15.4.2 DAC channel x control register 2 (DAC\_CHxCR2)

Address offset: 0x01 (channel 1) or 0x03 (channel 2 available on medium+ and high density devices only)

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved	DMAUDRIE	DMAEN	MAMP[3:0] <sup>(1)</sup>					
	rw	rw	-	-	-	-	-	-

1. Reserved on medium density devices

Bits 7:6 Reserved, forced by hardware at 0.

Bit 5 **DMAUDRIE**: DAC channel DMA underrun interrupt enable.

- 0: DMA underrun interrupt disabled.
- 1: DMA underrun interrupt enabled.

Bit 4 **DMAEN**: DAC DMA enable.

- 0: DMA disabled.
- 1: DMA enabled.

Bits 3:0<sup>(1)</sup> **MAMP[3:0]**: DAC channel x mask/amplitude selector.

- 0000: UnMask bit(0) of LFSR bit/ 2^1-1 triangle amplitude
- 0001: UnMask bit(1:0) of LFSR bit/ 2^2-1 triangle amplitude
- 0010: UnMask bit(2:0) of LFSR bit/ 2^3-1 triangle amplitude
- 0011: UnMask bit(3:0) of LFSR bit/ 2^4-1 triangle amplitude
- 0100: UnMask bit(4:0) of LFSR bit/ 2^5-1 triangle amplitude
- 0101: UnMask bit(5:0) of LFSR bit/ 2^6-1 triangle amplitude
- 0110: UnMask bit(6:0) of LFSR bit/ 2^7-1 triangle amplitude
- 0111: UnMask bit(7:0) of LFSR bit/ 2^8-1 triangle amplitude
- 1000: UnMask bit(8:0) of LFSR bit/ 2^9-1 triangle amplitude
- 1001: UnMask bit(9:0) of LFSR bit/ 2^10-1 triangle amplitude
- 1010: UnMask bit(10:0) of LFSR bit/ 2^11-1 triangle amplitude
- 1x1x: UnMask bit(11:0) of LFSR bit/ 2^12-1 triangle amplitude

1. Reserved on medium density devices.

### 15.4.3 DAC software trigger register (DAC\_SWTRIGR)

Address offset: 0x04

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved						SWTRIG2 <sup>(1)</sup>	SWTRIG1
						rs	rs

1. Reserved on medium density devices.

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

Bit 1<sup>(1)</sup> **SWTRIG2**: DAC channel 2 software trigger.

This bit is set and cleared by software to enable/disable the software trigger.

0: Software trigger disable

1: Software trigger enable

This bit is present only if the dual DAC is implemented (dual=1), otherwise it is forced to 0 by hardware.

Bit 0 **SWTRIG1**: DAC channel 1 software trigger.

This bit is set and cleared by software to enable/disable the software trigger.

0: Software trigger disable

1: Software trigger enable

*Note:* This bit is reset by hardware once the DAC\_DHRx register value is loaded to the DAC\_DORx register

1. Reserved on medium density devices.

### 15.4.4 DAC status register (DAC\_SR)

Address offset: 0x05

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved						DMAUDR2 <sup>(1)</sup>	DMAUDR1
						-	rc_w0

1. Reserved on medium density devices.

Bits 7:2 Reserved, forced by hardware at 0.

Bit 1<sup>(1)</sup> **DMAUDR2**: DAC channel DMA underrun flag.

0: No DMA underrun condition is detected.

1: DMA underrun condition detection.

This bit is present only if the dual DAC is implemented. Otherwise, it is forced to 0 by hardware.

*Note:* This bit generates the DMA underrun interrupt.

Bit 0 **DMAUDR1**: DAC channel DMA underrun flag.

0: No DMA underrun condition is detected.

1: DMA underrun condition detection.

*Note:* This bit generates the DMA underrun interrupt.

1. Reserved on medium density devices.

### 15.4.5 DAC channel x right aligned data holding register high (DAC\_RDHRH)

Address offset: 0x08 (channel 1) or 0x14 (channel 2 available on medium + and high-density devices only)

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved				RDHRH[3:0]			
				rw	rw	rw	rw

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

Bits 3:0 **RDHRH[3:0]**: DAC right aligned data holding register most significant bits.

These bits will be loaded as the 4 most significant bits of the 12-bit digital-to-analog conversion data stored into DHR.

### 15.4.6 DAC channel x right aligned data holding register low (DAC\_CHxRDHRL)

Address offset: 0x09 (channel 1) or 0x15 (channel 2 available on medium+ and high density devices only)

Reset value: 0x00

7	6	5	4	3	2	1	0
RDHRL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **RDHRL[7:0]**: DAC right aligned data holding register least significant bits.

These bits will be loaded as the 8 least significant bits of the 12-bit digital-to-analog conversion data stored into DHR.

*Note: If TEN is not set, write to this register triggers the 12-bit parallel load of DHRx by RDHRH + RDHRL*

### 15.4.7 DAC channel x left aligned data holding register high (DAC\_CHxLDHRH)

Address offset: 0x0C (channel 1) or 0x18 (channel 2 available on medium+ and high density devices only)

Reset value: 0x00

7	6	5	4	3	2	1	0
LDHRH							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **LDHRH[7:0]**: DAC left aligned data holding register most significant bits.

These bits will be loaded as the 8 most significant bits of the 12-bit digital-to-analog conversion data stored into DHR.

### 15.4.8 DAC channel x left aligned data holding register low (DAC\_CHxLDHRL)

Address offset: 0x0D (channel 1), 0x19 (channel 2 available on medium+ and high density devices only)

Reset value: 0x00

7	6	5	4	3	2	1	0
LDHRL[7:4]				Reserved			
rw	rw	rw	rw				

Bits 7:4 **LDHRL[7:4]**: DAC channel x left aligned data holding register least significant bits.

These bits will be loaded as the 4 least significant bits of the 12-bit digital-to-analog conversion data stored into DHR.

*Note: If TENx is not set, write to this register triggers the 12-bits parallel load of DHRx by LDHRH + LDHRL.*

Bits 3:0 Reserved, forced by hardware at 0.

### 15.4.9 DAC channel x 8-bit data holding register (DAC\_CHxDHR8)

Address offset: 0x10 (channel 1) or 0x1C (channel 2 available on medium+ and high density devices only)

Reset value: 0x00

7	6	5	4	3	2	1	0
8DHR							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **8DHR[7:0]**: DAC 8-bit data holding register.

These bits will be loaded as the 8 most significant bits of the 12-bit digital-to-analog conversion data stored into DHR.

*Note: If TEN is not set, writing to this register triggers the 8-bit load of DHR by DHR8[11:4]. 4 LSBits of DHR buffer keep unchanged.*

### 15.4.10 DAC channel x dual mode right aligned data holding register high (DAC\_DCHxRDHRH)

Address offset: 0x20 (channel 1) or 0x22 (channel 2)

Reset value: 0x0000 000

*Note:* This register is available on medium+ and high density devices only.

7	6	5	4	3	2	1	0
Reserved				RDHRH[3:0]			
				rw	rw	rw	rw

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

Bits 3:0 **RDHRH[3:0]**: DAC channel x dual mode right aligned data holding register most significant bits.

These bits will be loaded as the 4 most significant bits of the 12-bit digital-to-analog conversion data stored into DHR.

### 15.4.11 DAC channel x dual mode right aligned data holding register low (DAC\_DCHxRDHRL)

Address offset: 0x21 (channel1) or 0x23 (channel2)

Reset value: 0x0000 0000

*Note:* This register is available on medium+ and high density devices only.

7	6	5	4	3	2	1	0
RDHRL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **RDHRL[7:0]**: DAC channel x right aligned data holding register least significant bits.

These bits will be loaded as the 8 least significant bits of the 12-bit digital-to-analog conversion data stored into DHR.

*Note:* If TEN is not set, write to DAC\_DCH2RDHRL register triggers the 12-bits parallel load of both DHRx by DAC\_DCHxRDHRH + DAC\_DCHxRDHRL

### 15.4.12 DAC channel x dual mode left aligned data holding register high (DAC\_DCHxLDHRH)

Address offset: 0x24 (channel 1), 0x26 (channel 2)

Reset value: 0x0000 0000

*Note:* This register is available on medium+ and high density devices only.

7	6	5	4	3	2	1	0
LDHRH							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **LDHRH[7:0]**: DAC channel x dual mode left aligned data holding register most significant Bits.  
These bits will be loaded as the 8 most significant bits of the 12-bit digital-to-analog conversion data stored into DHR.

### 15.4.13 DAC channel x left aligned data holding register low (DAC\_DCHxLDHRL)

Address offset: 0x25 (Channel1), 0x27 (Channel2)

Reset value: 0x0000 0000

*Note:* This register is available on medium+ and high density devices only.

7	6	5	4	3	2	1	0
LDHRL[7:4]				Reserved			
rw	rw	rw	rw				

Bits 7:4 **LDHRL[7:4]**: DAC channel x data holding register left aligned data least significant bits.  
These bits will be loaded as the 4 least significant bits of the 12-bit digital-to-analog conversion data stored into DHR.

*Note:* If TEN is not set, write to DAC\_DCH2LDHRL register triggers the 12-bits parallel load of both DHRx by DAC\_DCHxLDHRH + DAC\_DCHxLDHRL.

Bits 3:0 Reserved, forced by hardware at 0.

#### 15.4.14 DAC channel x dual mode 8-bit data holding register (DAC\_DCHxDHR8)

Address offset: 0x28 (channel1) or 0x29 (channel2)

Reset value: 0x0000 0000

*Note:* This register is available on medium+ and high density devices only.

7	6	5	4	3	2	1	0
8DHR							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 8DHR[7:0]: DAC channel x dual mode 8-bit data holding register.

These bits will be loaded as the 8 most significant bits of the 12-bit digital-to-analog conversion data stored into DHR.

*Note:* If TEN is not set, write to DAC\_DCH2DHR8 register triggers the 8bit load of both DHRx[11:4] by DAC\_DCHxDHR8. 4LSB of DHRx buffers keep unchanged.

#### 15.4.15 DAC channel x data output register high (DAC\_CHxDORH)

Address offset: 0x2C (channel 1) or 0x30 (channel 2 available on medium+ and high density devices only)

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved				DORH[3:0]			
			r	r	r	r	

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

Bits 3:0 DORH[3:0]: DAC data output register most significant bit.

4 most significant bits of digital data currently under conversion.

#### 15.4.16 DAC channel x data output register low (DAC\_CHxDORL)

Address offset: 0x2D (channel 1), 0x31 (channel 2 available on medium+ and high density devices only)

Reset value: 0x00

7	6	5	4	3	2	1	0
DORL							
r	r	r	r	r	r	r	r

Bits 7:0 DORL[7:0]: DAC data output register least significant bit.

8 least significant bits of digital data currently under conversion.

### 15.4.17 DAC register map and reset values

**Table 54. DAC register map (medium, medium+ and high density devices)**

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	DAC_CH1CR1 Reset value	WAVEN[2:0] <sup>(1)</sup> 0		TSEL[2:0] 0			TEN 0	BOFF 0	EN 0
0x01	DAC_CH1CR2 Reset value	-	-	DMAUDRIE 0	DMAEN 0		MAMP[3:0] 0		
0x02	DAC_CH2CR1 Reset value	WAVEN[2:0] 0		TSEL[2:0] 0			TEN 0	BOFF 0	EN 0
0x03	DAC_CH2CR2 Reset value	-	-	DMAUDRIE 0	DMAEN 0		MAMP[3:0] <sup>(1)</sup> 0		
0x04	DAC_SWTRIGR Reset value	-	-	-	-	-	-	SWTRIG 2 <sup>(1)</sup> 0	SWTRIG1
0x05	DAC_SR Reset value	-	-	-	-	-	-	DMAUD R2 <sup>(1)</sup> 0	DMAUDR 1 0
0x06 to 0x07		Reserved area							
0x08	DAC_CH1RDHR H Reset value	-	-	-	-	RDHRH3 0	RDHRH2 0	RDHRH1 0	RDHRH0 0
0x09	DAC_CH1RDHR L Reset value	RDHRL7 0	RDHRL6 0	RDHRL5 0	RDHRL4 0	RDHRL3 0	RDHRL2 0	RDHRL1 0	RDHRL0 0
0x0A to 0x0B		Reserved area							
0x0C	DAC_CH1LDHR H Reset value	LDHRH7 0	LDHRH6 0	LDHRH5 0	LDHRH4 0	LDHRH3 0	LDHRH2 0	LDHRH1 0	LDHRH0 0
0x0D	DAC_CH1LDHR L Reset value	LDHRL7 0	LDHRL6 0	LDHRL5 0	LDHRL4 0	-	-	-	-
0x0E to 0x0F		Reserved area							
0x10	DAC_CH1DHR8 Reset value	8DHR7 0	8DHR6 0	8DHR5 0	8DHR4 0	8DHR3 0	8DHR2 0	8DHR1 0	8DHR0 0
0x11 to 0x13		Reserved area							
0x14	DAC_CH2RDHR H Reset value	-	-	-	-	RDHRH3 0	RDHRH2 0	RDHRH1 0	RDHRH0 0
0x15	DAC_CH2RDHR L Reset value	RDHRL7 0	RDHRL6 0	RDHRL5 0	RDHRL4 0	RDHRL3 0	RDHRL2 0	RDHRL1 0	RDHRL0 0

**Table 54. DAC register map (medium, medium+ and high density devices) (continued)**

<b>Address offset</b>	<b>Register name</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>	
0x16 to 0x17		Reserved area								
0x18	DAC_CH2LDHR H Reset value	LDHRH7 0	LDHRH6 0	LDHRH5 0	LDHRH4 0	LDHRH3 0	LDHRH2 0	LDHRH1 0	LDHRH0 0	
0x19	DAC_CH2LDHR L Reset value	LDHRL7 0	LDHRL6 0	LDHRL5 0	LDHRL4 0	-	-	-	-	
0x1A to 0x1B		Reserved area								
0x1C	DAC_CH2DHR8 Reset value	8DHR7 0	8DHR6 0	8DHR5 0	8DHR4 0	8DHR3 0	8DHR2 0	8DHR1 0	8DHR0 0	
0x1D to 0x1F		Reserved area								
0x20	DAC_DCH1RDH RH Reset Value	-	-	-	-	RDHRH3 0	RDHRH2 0	RDHRH1 0	RDHRH0 0	
0x21	DAC_DCH1RDH RL Reset Value	RDHRL7 0	RDHRL6 0	RDHRL5 0	RDHRL4 0	RDHRL3 0	RDHRL2 0	RDHRL1 0	RDHRL0 0	
0x22	DAC_DCH2RDH RH Reset Value	-	-	-	-	RDHRH3 0	RDHRH2 0	RDHRH1 0	RDHRH0 0	
0x23	DAC_DCH2RDH RL Reset Value	RDHRL7 0	RDHRL6 0	RDHRL5 0	RDHRL4 0	RDHRL3 0	RDHRL2 0	RDHRL1 0	RDHRL0 0	
0x24	DAC_DCH1LDH RH Reset Value	LDHRH7 0	LDHRH6 0	LDHRH5 0	LDHRH4 0	LDHRH3 0	LDHRH2 0	LDHRH1 0	LDHRH0 0	
0x25	DAC_DCH1LDH RL Reset Value	LDHRL7 0	LDHRL6 0	LDHRL5 0	LDHRL4 0	-	-	-	-	
0x26	DAC_DCH2LDH RH Reset Value	LDHRH7 0	LDHRH6 0	LDHRH5 0	LDHRH4 0	LDHRH3 0	LDHRH2 0	LDHRH1 0	LDHRH0 0	
0x27	DAC_DCH2LDH RL Reset Value	LDHRL7 0	LDHRL6 0	LDHRL5 0	LDHRL4 0	-	-	-	-	
0x28	DAC_DCH1DHR 8 Reset Value	8DHR7 0	8DHR6 0	8DHR5 0	8DHR4 0	8DHR3 0	8DHR2 0	8DHR1 0	8DHR0 0	
0x29	DAC_DCH2DHR 8 Reset Value	8DHR7 0	8DHR6 0	8DHR5 0	8DHR4 0	8DHR3 0	8DHR2 0	8DHR1 0	8DHR0 0	

**Table 54. DAC register map (medium, medium+ and high density devices) (continued)**

<b>Address offset</b>	<b>Register name</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>	
0x2A to 0x2B		Reserved area								
0x2C	DAC_CH1DORH Reset value	-	-	-	-	DORH3 0	DORH2 0	DORH1 0	DORH0 0	
0x2D	DAC_CH1DORL Reset value	DORL7 0	DORL6 0	DORL5 0	DORL4 0	DORL3 0	DORL2 0	DORL1 0	DORL0 0	
0x2E to 0x2F		Reserved area								
0x30	DAC_CH2DORH Reset value	-	-	-	-	DORH3 0	DORH2 0	DORH1 0	DORH0 0	
0x31	DAC_CH2DORL Reset value	DORL7 0	DORL6 0	DORL5 0	DORL4 0	DORL3 0	DORL2 0	DORL1 0	DORL0 0	
0x32 to 0x35		Reserved area								

1. Reserved on medium density devices.

## 16 Comparators (COMP)

This section applies to low density STM8L15xx devices, medium density STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L15xx devices and high density STM8L15xx/STM8L16xx devices, unless otherwise specified.

Comparators (COMP) are not available on STM8L05xx value line devices.

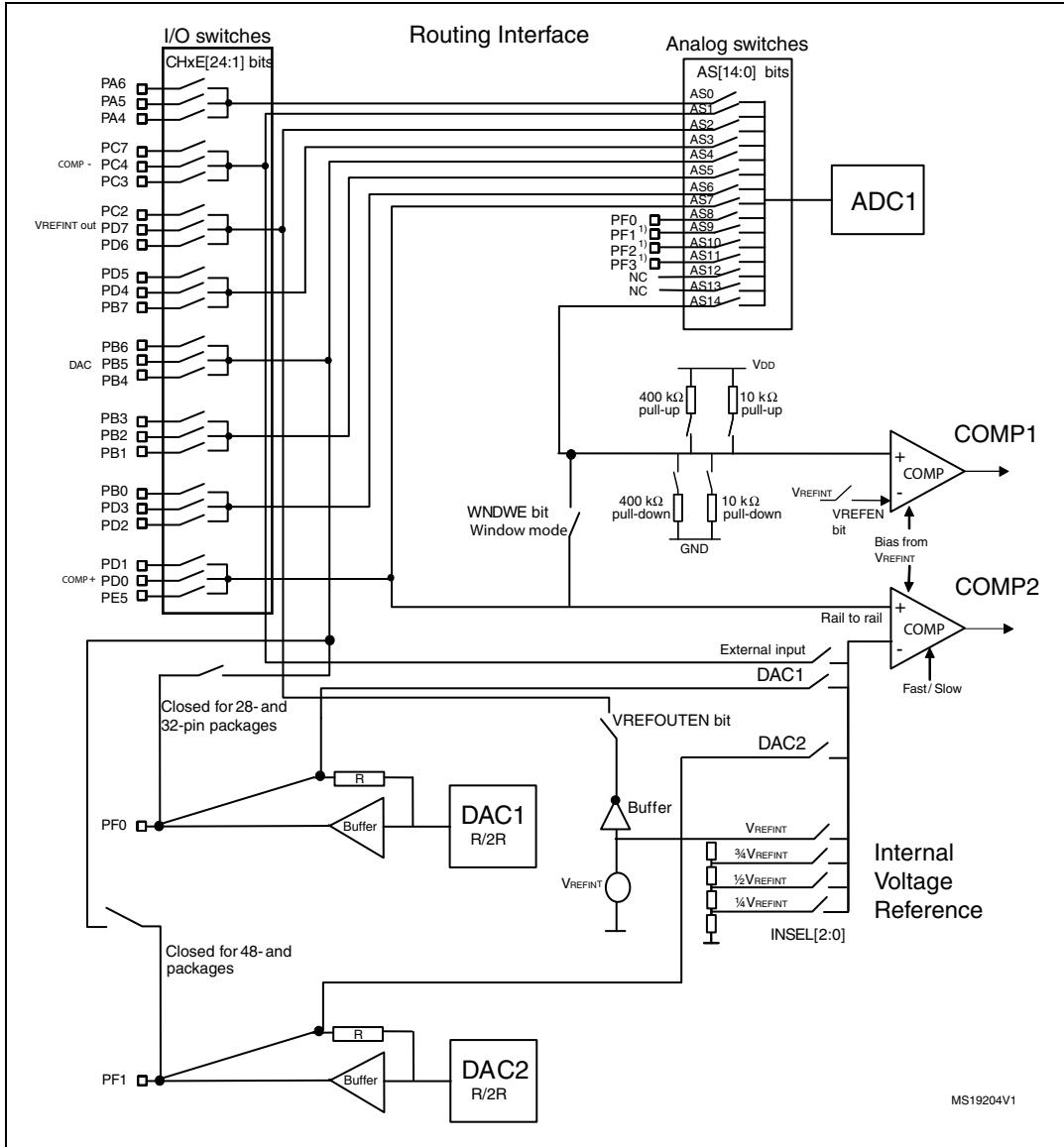
### 16.1 COMP introduction

The STM8L15xx contains two zero-crossing comparators COMP1 and COMP2 that share the same current bias.

*Note:* *For all I/Os used as comparator inputs, the configuration in the GPIO registers must be remain input floating.*

When using the routing interface (see [Section 11.2 on page 124](#)) the comparator inputs can be connected to external I/Os.

Figure 50. Comparator block diagram



1. DAC1/2 available on medium+ and high density devices only. Not connected on medium density devices.

MS19204V1

## 16.2 COMP main features

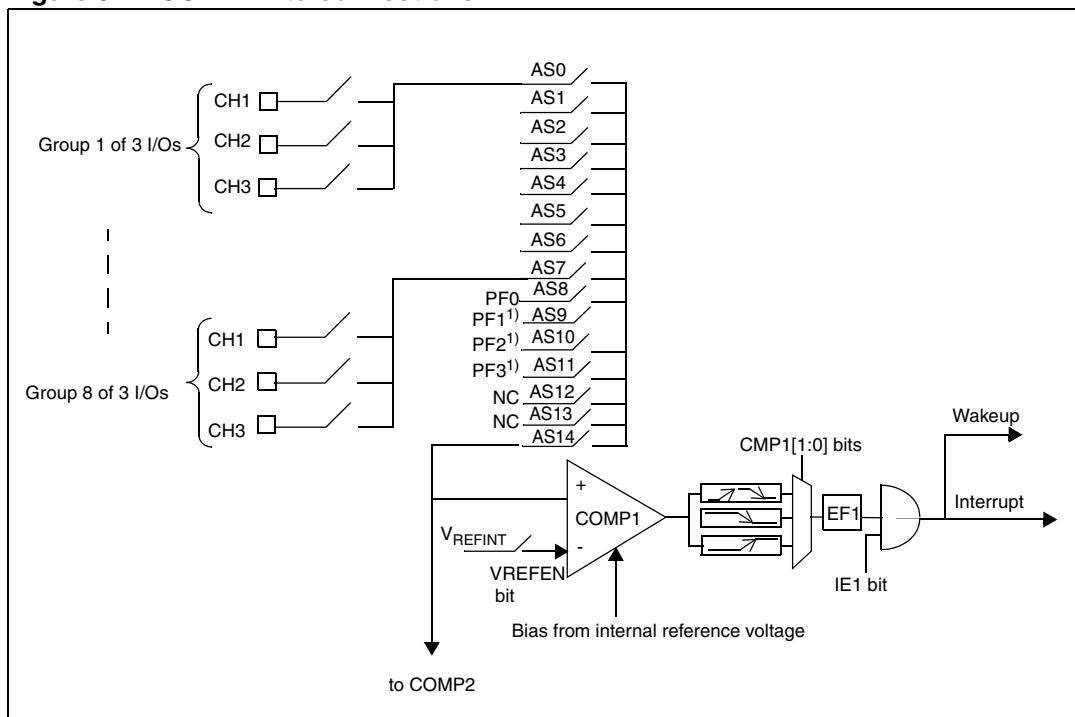
- One comparator (COMP1) with fixed threshold (internal reference voltage). The non-inverting input can be selected from 25 (medium density devices) or 28 (medium+ and high density devices) external I/Os.
- One rail-to-rail comparator (COMP2) with selectable threshold. The non-inverting input can be selected from three I/Os. The threshold can be selected from:
  - Internal reference voltage  $V_{REFINT}$
  - Internal reference voltage submultiple (1/4, 1/2, 3/4)
  - DAC output
  - One of three external I/Os.
- The 2 comparators can be combined into window comparators.
- A zero-crossing can generate a rising edge or a falling edge on comparator outputs depending on a polarity bit
- Each comparator can generate an interrupt with wakeup from Halt capability
- COMP2 output can be redirected to TIM1 BRK or OCREFCLR inputs, or TIM2/TIM3 Input Capture 2.
- COMP2 speed is configurable for optimum speed/consumption ratio.

The block diagram is shown on [Figure 1](#).

## 16.3 Comparator 1 (COMP1)

*Figure 51* shows the comparator 1 interconnections.

**Figure 51. COMP1 interconnections**



1. On medium+ and high density devices only.

Note:

*COMP1 and ADC cannot be used at the same time as they share the same set of analog switches.*

To use the COMP1 comparator, the application must perform the following steps:

1. Set the VREFEN bit in COMP\_CSR3 register to connect VREFINT to comparator 1 inverting input.
2. Close the analog switches to create the path from selected I/O to the non-inverting input:
  - Close analog switch number 14 by setting the AS14 bit in the RI\_ASCR2 register.
  - Close analog switch number ‘n’ corresponding to the I/O group (refer to [Table 27](#)), by setting the ASx bit in the RI\_ASCR1 or RI\_ASCR2 register
3. Close the I/O switch of the I/O to be connected to the COMP1 non-inverting input. The input can be any of the 24 I/Os from the 8 groups of 3 I/Os (refer to [Table 27](#)).
  - Select the correct I/O port by setting the corresponding CHxE bit in the I/O switch registers: RI\_IOSR1, RI\_IOSR2 or RI\_IOSR3 (refer to [Table 27](#))
4. If required, enable the COMP1 interrupt by setting the IE1 bit in the COMP\_CSR1 register
5. Configure the comparator to detect rising edge, falling edge or both rising and falling edges using the CMP1[1:0] bits in the COMP\_CSR1 register.

Note:

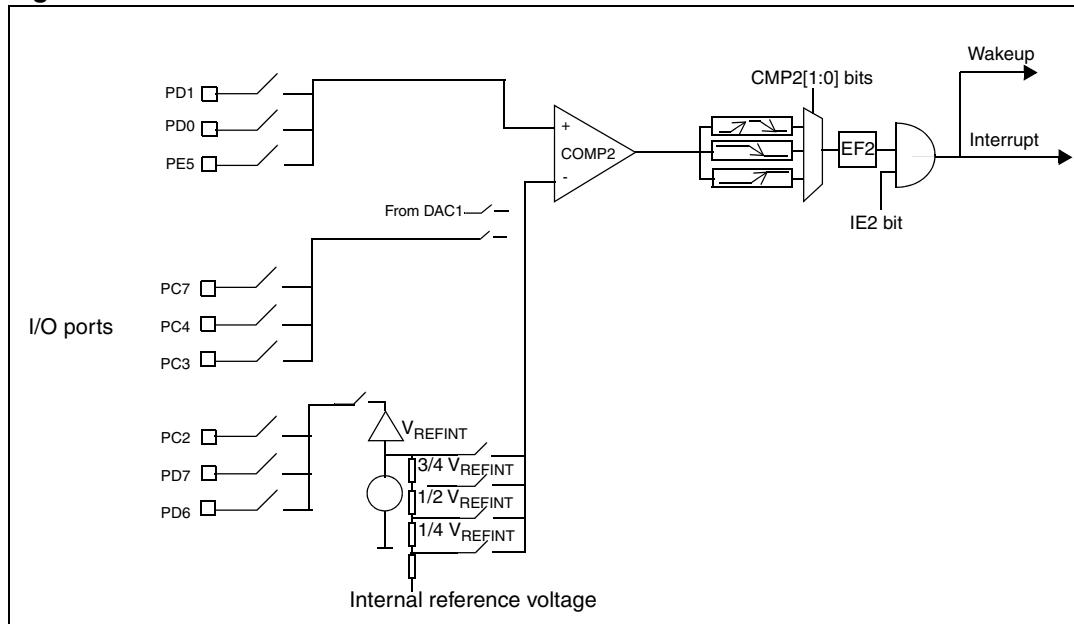
*The channel can be changed once the comparator is enabled.*

*The step 3 is not applicable for PF0, PF1, PF2 and PF3.*

## 16.4 Comparator 2 (COMP2)

*Figure 52* shows the comparator 2 interconnections.

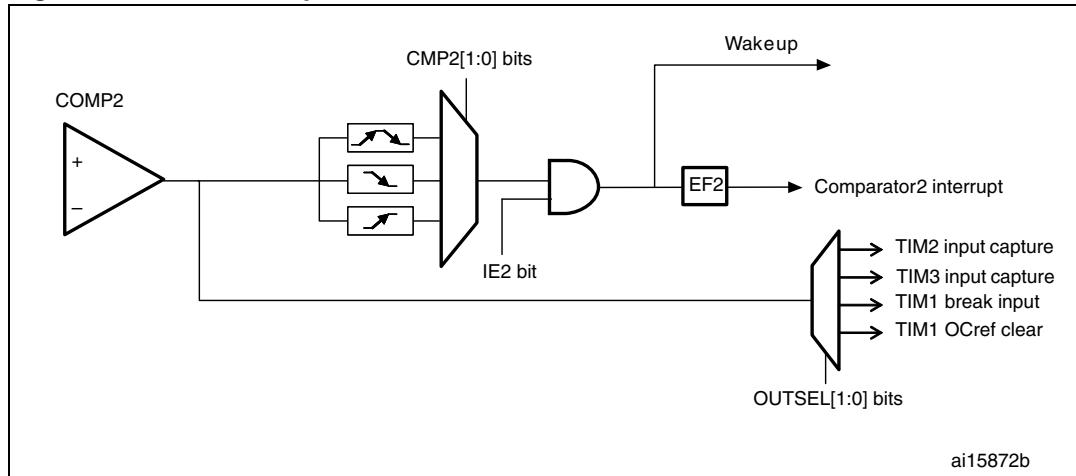
**Figure 52. COMP2 interconnections**



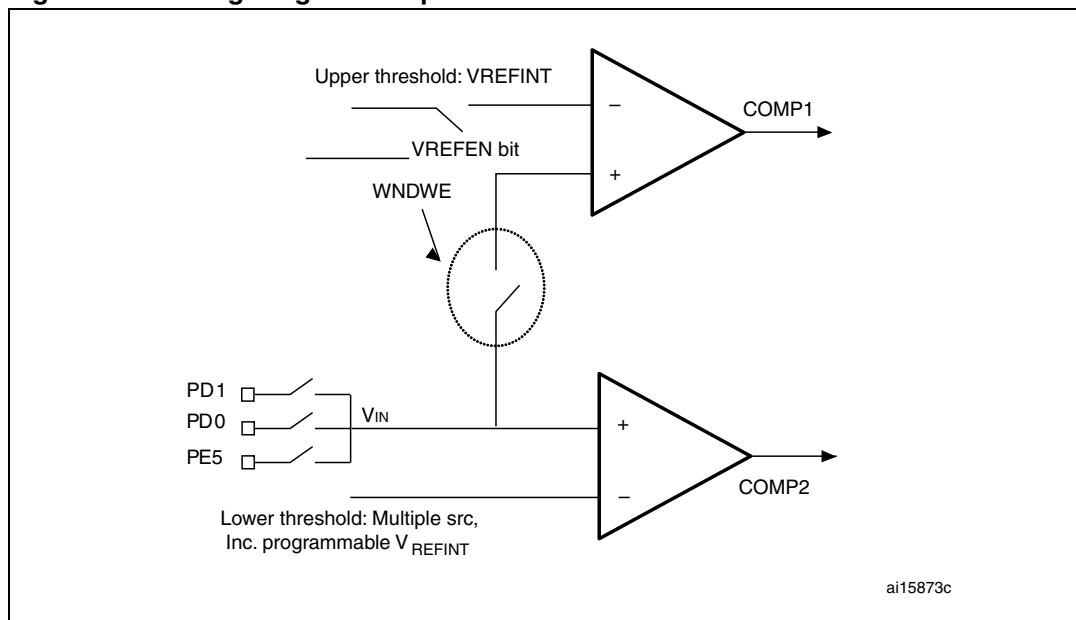
To use the COMP2 comparator, perform the following steps:

1. Select the COMP2 Inverting Input with the INSEL[2:0] bits in the COMP\_CSR3 register.
  - To select an external I/O (any I/O in of group 2), close the I/O channel switch by setting the CH4E bit in the RI\_IOSR1, the CH5E bit in the RI\_IOSR2 or the CH6E bit in the RI\_IOSR3 register (see *Table 27*).
2. Close the I/O channel switch to connect the I/O to the COMP2 non-inverting input. The input can be from any I/O in group 8 (see *Table 27*).
  - Set the CH22E bit in the RI\_IOSR1 register, the CH23E bit in the RI\_IOSR2 or the CH24E bit in the RI\_IOSR3 register.
3. If required, perform the following procedures:
  - Select the speed with the SPEED bit in COMP\_CSR2 register.
  - Redirect the COMP2 output to timer 1, 2 or 3 by configuring the OUTSEL[1:0] bits in the COMP\_CSR3 register (see *Figure 53*).
  - Enable the COMP2 interrupt by setting the IE2 bit in the COMP\_CSR2 register.
4. To detect rising edge, falling edge or both rising and falling edges using the CMP2[1:0] bits in the COMP\_CSR2 register.

*Note:* The channel can be changed once the comparator is enabled.

**Figure 53. COMP2 output redirection**

## 16.5 Using the comparators in window mode

**Figure 54. Configuring the comparators in window mode**

Note:

*In window mode, only the Group 8 (PD0, PD1 and PE5) can be used as a non-inverting input.*

To use the COMP1 and COMP2 comparators in window mode, perform the following steps:

1. Set the VREFEN bit in COMP\_CSR3 register to connect VREFINT to comparator 1 inverting input.
2. Select the COMP2 inverting input as explained in [Section 16.4](#).
3. Enable the Window mode by setting the WNDWE bit in the COMP\_CSR3 register.
4. Select the COMP2 non-inverting input:
  - like for COMP1: follow step 2 and 3 from [Section 16.3](#)
  - like for COMP2: follow step 3 from [Section 16.4](#)
5. Enable the COMP1 comparator using the CMP1[1:0] bits in the COMP\_CSR1 register and the COMP2 comparator using the CMP2[1:0] in the COMP\_CSR2 register. Refer to [Table 55](#) for the different configurations which must be programmed to get the correct wakeup or interrupt event.

**Table 55. Window interrupts/wakeup modes**

Interrupt/wakeup mode	CMP1[1:0] upper threshold	CMP2[1:0] lower threshold
Vin > Upper (above)	0b10	0b00
Vin < Lower (below)	0b00	0b01
Lower < Vin < Upper (inside)	0b01	0b10
Vin < Lower or Vin > Upper (outside)	0b10	0b01

## 16.6 COMP low power modes

**Table 56. Behavior of comparators in low power modes**

Mode	Description
Wait or Low power wait mode	No effect on comparator. Comparator interrupts cause the device to exit from Wait or Low power wait mode.
Active-halt or Halt	No effect on comparator. Comparator interrupts cause the device to exit from Active-halt or Halt mode.

*Note:* Comparators cannot be used to exit the device from Halt/Active-halt mode when the internal reference voltage is stopped using the ULP bit in the PWR\_CSR2 register.

## 16.7 COMP interrupts

**Table 57. Comparator interrupt requests**

Interrupt event	Event flag	Enable control bit	Exit from Wait/Low power wait	Exit from Halt/Active-halt
Comparator 1 event flag	EF1	IE1	Yes	Yes
Comparator 2 event flag	EF2	IE2	Yes	Yes

## 16.8 COMP registers

### 16.8.1 Comparator control and status register 1 (COMP\_CSR1)

Address offset: 0x00

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved	IE1	EF1	CMP1OUT	STE	CMP1[1:0]			
	rw	rc_w0	r	rw	rw	rw	rw	rw

Bits 7:6 Reserved, must be kept cleared

Bit 5 **IE1**: Comparator 1 interrupt enable

This bit enables the comparator 1 interrupt generation when an event is detected.

- 0: Comparator 1 interrupt disabled
- 1: Comparator 1 interrupt enabled

Bit 4 **EF1**: Comparator 1 event flag

This bit is set when the selected edge in COMP1[1:0] on comparator 1 output occurs. It is cleared writing 0 to it. If the comparator interrupt is enabled, then an interrupt is generated.

- 0: No event detected
- 1: Event detected

Bit 3 **CMP1OUT**: Comparator 1 output

This bit is the exact copy of the comparator 1 output.

- 0: Comparator 1 output is low when non inverting input is at lower voltage than inverting input
- 1: Comparator 1 output is high when non-inverting input is at higher voltage than inverting input

Bit 2 **STE**: Schmitt trigger enable

- 0: Schmitt trigger disabled
- 1: Schmitt trigger enabled

*Note: The STE bit modifies the behavior of the Schmitt trigger of the I/Os featuring an analog function (ADC channels, comparator inputs) only when the I/O analog switch is closed.*

Bits 1:0 **CMP1[1:0]**: Comparator 1 configuration

- 00: Comparator 1 disabled
- 01: Event detection on the falling edge of comparator 1 output
- 10: Event detection on the rising edge of comparator 1 output
- 11: Event detection on both rising/falling edges of comparator 1 output

### 16.8.2 Comparator control and status register 2 (COMP\_CSR2)

Address offset: 0x01

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved	IE2	EF2	CMP2OUT	SPEED	CMP2[1:0]			
	rw	rc_w0	r	rw	rw	rw	rw	rw

Bits 7:6 Reserved, must be kept cleared

Bit 5 **IE2**: Comparator 2 Interrupt Enable

This bit enables the comparator 2 interrupt generation when an event is detected.

0: Comparator 2 interrupt disabled

1: Comparator 2 interrupt enabled

Bit 4 **EF2**: Comparator 2 event flag

This bit is set when the selected edge in COMP2[1:0] on comparator 2 output occurs. It is cleared by writing 0 to it. If the comparator interrupt is enabled, then an interrupt is generated.

0: No event detected

1: Event detected

Bit 3 **CMP2OUT**: Comparator 2 output

This bit is the exact copy of the comparator 2 output.

0: Comparator 2 output is low when non inverting input is at lower voltage than inverting input

1: Comparator 2 output is high when non inverting input is at higher voltage than inverting input

Bit 2 **SPEED**: Comparator 2 speed mode

0: Slow speed

1: Fast speed

Bits 1:0 **CMP2[1:0]**: Comparator 2 configuration

00: Comparator 2 disabled

01: Event detection on the falling edge of comparator 2 output

10: Event detection on the rising edge of comparator 2 output

11: Event detection on both rising/falling edges of comparator 2 output

### 16.8.3 Comparator control and status register 3 (COMP\_CSR3)

Address offset: 0x02

Reset value: 0xC0

7	6	5	4	3	2	1	0
OUTSEL[1:0]		INSEL[2:0]			VREFEN	WNDWE	VREFOUTEN
rw	rw	rw	rw	rw	rw	rw	rw

#### Bits 7:6 **OUTSEL[1:0]** Comparator 2 output selection

These bits are written by software to connect the output of COMP2 to a selected timer input.

00: COMP2 output connected to Timer 2 input capture 2 (TIM2 IC2). The corresponding input capture from the I/O is no more available.

01: COMP2 output connected to Timer 3 input capture 2 (TIM3 IC2). The corresponding input capture from the I/O is no more available.

10: COMP2 output connected to Timer 1 break input (TIM1 BRK). The break input from the I/O is no more available.

11: COMP2 output connected to Timer 1 OCREF clear (TIM1 OCREFCLR)

#### Bits 5:3 **INSEL**: Comparator 2 inverting input selection

000 = no selection

001 = I/O (Group 2 of I/Os)

010 = Internal reference voltage  $V_{REFINT}$

011 = 3/4  $V_{REFINT}$

100 = 1/2  $V_{REFINT}$

101 = 1/4  $V_{REFINT}$

110 = DAC1

111 = DAC2

#### Bit 2 **VREFEN**: Internal reference voltage $V_{REFINT}$ enable

0:  $V_{REFINT}$  disconnected from COMP1 inverting input

1:  $V_{REFINT}$  connected to COMP1 inverting input

#### Bit 1 **WNDWE**: Window mode enable

0: Disabled

1: Enabled

#### Bit 0 **VREFOUTEN**: $V_{REFINT}$ output enable

This bit can be set by software to output the internal voltage reference on Group 3 I/Os. Refer to [Figure 27](#).

0: Disabled

1: Enabled

### 16.8.4 Comparator control and status register 4 (COMP\_CSR4)

Address offset: 0x03

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved	NINVTRIG[2:0]				INVTRIG[2:0]			
	rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:6 Reserved, must be kept cleared

Bits 5:3 **NINVTRIG[2:0]**: COMP2 non inverting input

These bits control the Schmitt triggers of all the I/Os belonging to the I/O group 8 (see [Table 27](#)), corresponding to the COMP2 non inverting inputs.

NINVTRIG[0] enables/disables the trigger on pin PE5.

NINVTRIG[1] enables/disables the trigger on pin PD0.

NINVTRIG[2] enables/disables the trigger on pin PD1.

0: Trigger enabled

1: Trigger disabled

Bits 2:0 **INVTRIG[2:0]**: COMP2 inverting input

These bits control the Schmitt triggers of all the I/Os belonging to the I/O group 2 (see [Table 27](#)), corresponding to the COMP2 inverting inputs.

INVTRIG[0] enables/disables the trigger on pin PC3.

INVTRIG[1] enables/disables the trigger on pin PC4.

INVTRIG[2] enables/disables the trigger on pin PC7.

0: Trigger enabled

1: Trigger disabled

**Note:** When the trigger is disabled on an I/O, the associated bit in Px\_IDR register is always read as 0 even if another level is present on the pin.

### 16.8.5 Comparator control and status register 5 (COMP\_CSR5)

Address offset: 0x04

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved	DACTRIG[2:0]				VREFTRIG[2:0]			
	rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:6 Reserved, must be kept cleared

Bits 5:3 **DACTRIG[2:0]**: DAC outputs

These bits control the Schmitt triggers of all the I/Os belonging to the I/O group 5 (see [Table 27](#)), corresponding to the DAC outputs.

DACTRIG[0] enables/disables the trigger on pin PB6.

DACTRIG[1] enables/disables the trigger on pin PB5.

DACTRIG[2] enables/disables the trigger on pin PB4.

0: Trigger enabled

1: Trigger disabled

Bits 2:0 **VREFTRIG[2:0]**: V<sub>REFINT</sub> outputs

These bits control the Schmitt triggers of all the I/Os belonging to the I/O group 3 (see [Table 27](#)), corresponding to the V<sub>REFINT</sub> outputs.

VREFTRIG[0] enables/disables the trigger on pin PC2.

VREFTRIG[1] enables/disables the trigger on pin PD7.

VREFTRIG[2] enables/disables the trigger on pin PD6.

0: Trigger enabled

1: Trigger disabled

**Note:** When the trigger is disabled on an I/O, the associated bit in Px\_IDR register is always read as 0 even if another level is present on the pin.

### 16.8.6 COMP register map and reset values

**Table 58. Comparators and routing Interface register map**

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	COMP_CSR1 Reset value	-0	-0	IE10	EF10	CMP1OUT0	STE0	CMP110	CMP100
0x01	COMP_CSR2 Reset value	-0	-0	IE20	EF20	CMP2OUT0	SPEED0	CMP210	CMP200
0x02	COMP_CSR3 Reset value	OUTSEL11	OUTSEL01	INSEL20	INSEL10	INSEL00	VREFEN0	WINDWE0	VREFOUTEN0
0x03	COMP_CSR4 Reset value	-0	-0	INVTRIG20	INVTRIG10	INVTRIGO0	NINVTRIG20	NINVTRIG10	NINVTRIG00
0x04	COMP_CSR5 Reset value	-0	-0	DACTRIG20	DACTRIG10	DACTRIGO0	VREFTRIG20	VREFTRIG10	VREFTRIG00

## 17 LCD controller

This section applies to medium density STM8L052x/STM8L152x devices, medium density STM8AL3Lxx devices, medium+ density STM8L052x/STM8L152x devices and high density STM8L052x/STM8L152x/STM8L162x devices, unless otherwise specified.

### 17.1 LCD controller introduction

The LCD controller can drive a passive-matrix LCD (liquid crystal display) unit.

- In STM8L05xx value line high density devices, it can interface with 8 common terminals and up to 24 segment terminals to drive up to 192 picture elements (pixels).
- In medium+ and high density devices, it can interface with 8 common terminals and up to 44 segment terminals to drive up to 320 picture elements (pixels).
- In medium density devices, It can interface with 4 common terminals and up to 28 segment terminals to drive up to 112 picture elements (pixels).

The LCD is made up of several segments (pixels or complete symbols) which can be turned visible or invisible. Each segment consists of a layer of liquid crystal molecules aligned between two electrodes. When a voltage greater than a threshold voltage is applied across the liquid crystal, the segment becomes visible. The segment voltage must be alternated to avoid an electrophoresis effect in the liquid crystal (which degrades the display). The waveform across a segment must then be generated so as to avoid having a direct current (DC).

#### 17.1.1 Definitions

##### Glossary

- LCD (Liquid crystal display): a passive display panel with terminals driving segments.
- Segment: the smallest viewing element (a single bar or dot that is used to help create a character on a LCD display).
- Common: electrical connection terminal connected to several segments.
- Duty ratio: number defined as  $1 / (\text{number of common terminals on an LCD display})$ .
- Bias: indicates the number of voltage levels used when driving an LCD. It is defined as  $1 / (\text{number of voltage levels used driving a LCD display} - 1)$ .
- Frame: one period of the waveforms written to a segment.
- Frame rate: number of frames per second, that is, number of times the LCD segments are energized per second.
- Boost circuit: contrast controller circuit.

## 17.2 LCD controller main features

- High-flexibility frame rate control
- **In value line high density devices**
  - Static, 1/2, 1/3, 1/4 and 1/8 duty supported
  - 1/2, 1/3 and 1/4 bias supported
  - LCD data RAM of up to 18 x 8-bit registers which contain pixel (LCD picture element) information (active/inactive)
  - Capability to drive 112 (28x4) or 192 (24x8) pixels
- **In medium+ and high density devices**
  - Static, 1/2, 1/3, 1/4 and 1/8 duty supported
  - 1/2, 1/3 and 1/4 bias supported
  - LCD data RAM of up to 22 x 8-bit registers which contain pixel (LCD picture element) information (active/inactive)
  - Capability to drive 176 (44x4) or 320 (40x8) pixels
- **In medium density devices**
  - Static, 1/2, 1/3 and 1/4 duty supported
  - 1/2 and 1/3 bias supported
  - LCD data RAM of up to 14 x 8-bit registers which contain pixel (LCD picture element) information (active/inactive)
  - Capability to drive 112 (28x4) pixels
- LCD output voltage software selectable
- No need for external analog components:
  - A booster is embedded to generate an internal  $V_{LCD}$  voltage independent from  $V_{DD}$ .
  - Software selection between external and internal  $V_{LCD}$  voltage source. In case of an external source, the internal boost circuit is disabled to reduce power consumption.
  - A resistive network is embedded to generate intermediate  $V_{LCD}$  voltages.
  - The structure of the resistive network is configurable by software to adapt the power consumption to match the capacitive charge required by the LCD panel.
- The contrast can be adjusted using two different methods:
  - When using the internal booster, the software can adjust  $V_{LCD}$  between  $V_{LCDmin}$  and  $V_{LCDmax}$ .
  - Programmable dead time (up to 7 phase periods) between frames.
- Full support of low power modes: the LCD controller can be displayed in Active-halt, Wait, Low power run and Low power wait modes or can be fully disabled to reduce power consumption
- Phase inversion to reduce power consumption and EMI (electromagnetic interference)
- Start of frame interrupt to synchronize the software when updating the LCD data RAM.

- Blink capability:
  - SEG0 COM0, SEG0 COMx, or SEGx COMx which can be programmed to blink at a configurable frequency.
  - Software adjustable blink frequency to achieve around 0.5 Hz, 1 Hz, 2 Hz or 4 Hz.
- Capability to assign some of the SEGx (segment) and COMx (common) pins as standard general purpose IOs when not used.

## 17.3 LCD functional description

### **17.3.1 General description**

The block diagram of the LCD controller is shown in the following figure.

**Figure 55. Medium + and high density LCD controller block diagram**

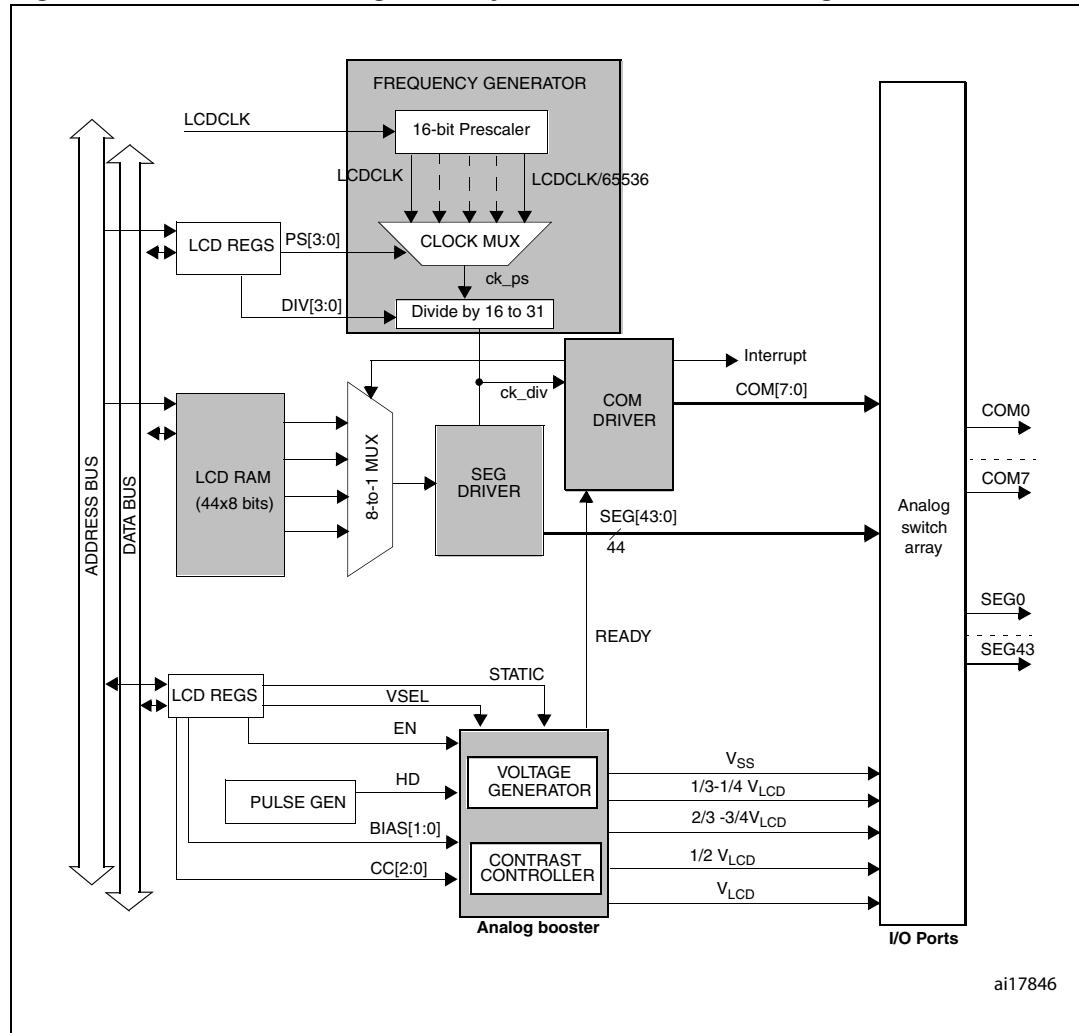
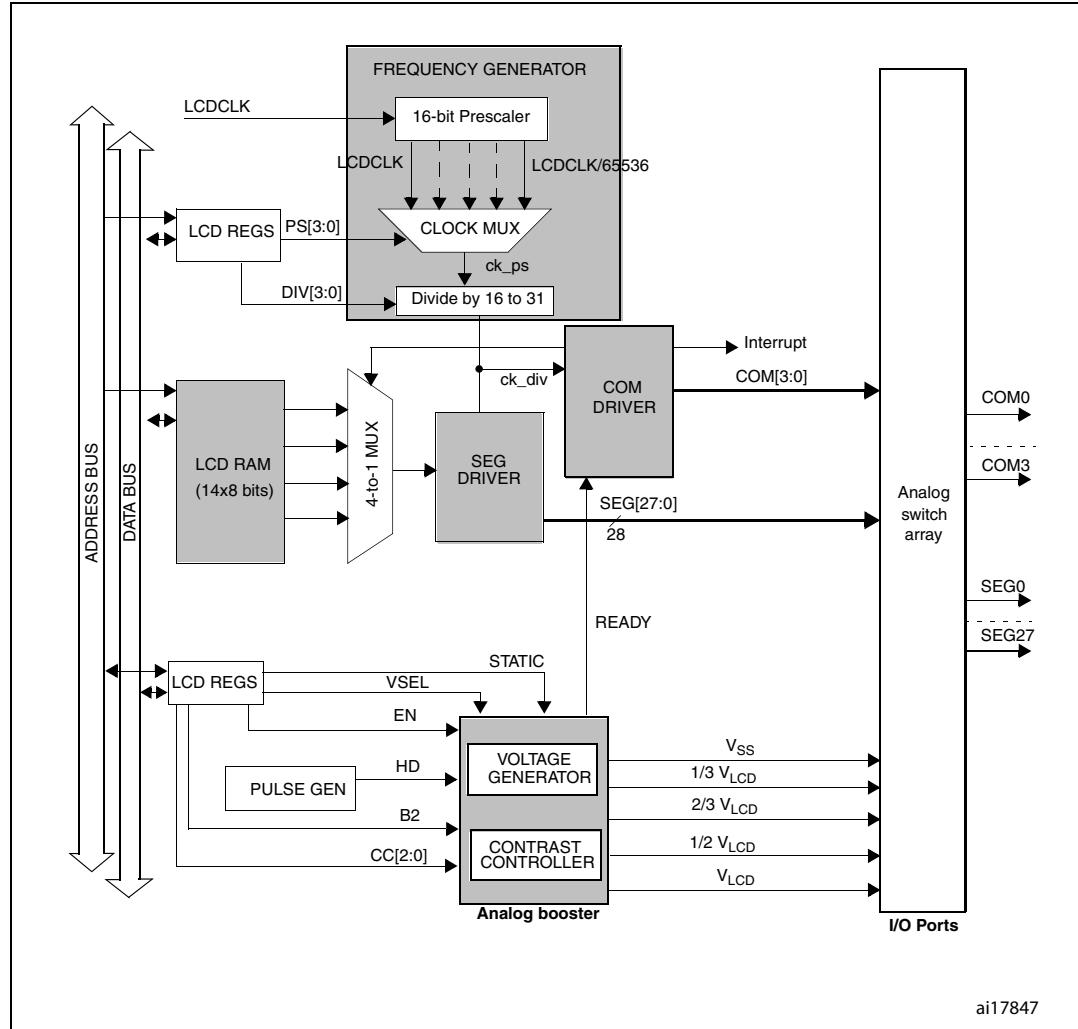


Figure 56. Medium density LCD controller block diagram



### 17.3.2 Frequency generator

The frequency generator allows to achieve various LCD frame rates starting from an LCD input clock frequency which is equal to RTCCLK divided by 2. For more information about the LCD clock source configuration please refer to [Section 9.9: RTC and LCD clock](#).

This clock source must be in the range of 16.384 kHz to 500 kHz and must be stable to obtain an accurate LCD timing and minimize the DC voltage offset across LCD segments. It can be divided by values from 1 up to  $2^{15} \times 31$ . The frequency generator consists of a prescaler (16-bit ripple counter) and a programmable clock divider (factor 16 to 31).

The PS[3:0] bits in the LCD\_FRQ register select the prescaler so as to divide LCDCLK by  $2^{PS[3:0]}$ . If a finer resolution rate is required, the DIV[3:0] bits in the LCD\_FRQ register can be used to further divide the clock by 16 to 31. In this way the user can fine-tune the frequency by linearly scaling (up/down) the clock with the counter.

The output of the frequency generator block is  $f_{ck\_div}$  which constitutes the time base for the entire LCD controller. The  $f_{ck\_div}$  frequency is equivalent to the LCD phase frequency rather than the frame frequency (they are equal only in case of static duty).

The frame frequency ( $f_{frame}$ ) is obtained from  $f_{ck\_div}$  by dividing it by the number of active common terminals (or by multiplying it by the duty ratio). Thus the relation between the input clock frequency ( $f_{LCDCLK}$ ) of the frequency generator and its output clock frequency  $f_{ck\_div}$  is:

$$f_{ck\_div} = \frac{f_{LCDCLK}}{2^{PS} \times (16 + DIV)}$$

$$f_{frame} = f_{ck\_div} \times \text{duty}$$

The frame frequency which must be selected to be within a range of around ~30 Hz to ~100 Hz is a compromise between power consumption and acceptable refresh rate. In addition, a dedicated blink prescaler selects the blink frequency. This frequency is defined as:

$$f_{BLINK} = \frac{f_{ck\_div}}{2^{BLINKF+3}}$$

with  $BLINKF[2:0] = 0,1,2,\dots,7$

The blink frequency achieved is in the range of 0.5 Hz, 1 Hz, 2 Hz or 4 Hz. Some examples of typical frame rate calculation are shown in [Table 59](#) and [Table 60](#).

**Table 59. Typical frame rate calculation for input frequency of 16.384 kHz**

CLK	Duty	PS[3:0]	DIV[3:0]	Ratio	f <sub>LCD</sub>	f <sub>frame</sub>	BLINKF[2:0] divider	f <sub>BLINK</sub>
16.384 kHz	1/8	2	0	64	256 Hz	32 Hz	/512 /256 /64	0.5 Hz 1.0 Hz 4.0 Hz
		1	0	32	512 Hz	64 Hz	/1024 /512 /128	0.5 Hz 1.0 Hz 4.0 Hz
		0	0	16	1024 Hz	128 Hz	/1024 /256	1.0 Hz 4.0 Hz
	1/4	3	0	128	128 Hz	32 Hz	/256 /128 /32	0.5 Hz 1.0 Hz 4.0 Hz
		2	0	64	256 Hz	64 Hz	/512 /256 /64	0.5 Hz 1.0 Hz 4.0 Hz
		1	0	32	512 Hz	128 Hz	/1024 /512 /128	0.5 Hz 1.0 Hz 4.0 Hz
	1/2	4	0	256	64 Hz	32 Hz	/128 /64 /16	0.5 Hz 1.0 Hz 4.0 Hz
		3	0	128	128 Hz	64 Hz	/256 /128 /32	0.5 Hz 1.0 Hz 4.0 Hz
		2	0	64	256 Hz	128 Hz	/512 /256 /64	0.5 Hz 1.0 Hz 4.0 Hz
	static	5	0	512	32 Hz	32 Hz	/64 /32 /8	0.5 Hz 1.0 Hz 4.0 Hz
		4	0	256	64 Hz	64 Hz	/128 /64 /16	0.5 Hz 1.0 Hz 4.0 Hz
		3	0	128	128 Hz	128 Hz	/256 /128 /32	0.5 Hz 1.0 Hz 4.0 Hz

**Table 60.** Typical frame rate calculation for input frequency of 500 kHz

CLK	Duty	PS[3:0]	DIV[3:0]	Ratio	f <sub>LCD</sub>	f <sub>FRAME</sub>	BLINKF[2:0] divider	f <sub>BLINK</sub>
500 kHz	1/8	7	0	2048	244.00 Hz	30.51 Hz	/256 /128 /32	0.47 Hz 0.95 Hz 3.81 Hz
		5	3	608	822.30 Hz	102.79 Hz	/1024 /256	0.80 Hz 3.21 Hz
	1/4	8	0	4096	122.00 Hz	30.51 Hz	/256 /128 /32	0.47 Hz 0.95 Hz 3.81 Hz
		6	3	1216	411.10 Hz	102.79 Hz	/1024 /512 /128	0.40 Hz 0.80 Hz 3.21 Hz
	1/2	9	0	8192	61.00 Hz	30.51 Hz	/128 /64 /16	0.47 Hz 0.95 Hz 3.81 Hz
		7	3	2432	205.60 Hz	102.79 Hz	/512 /256 /64	0.40 Hz 0.80 Hz 3.21 Hz
	static	10	0	16384	30.51 Hz	30.51 Hz	/64 /32 /8	0.47 Hz 0.95 Hz 3.81 Hz
		8	3	4864	102.79 Hz	102.79 Hz	/256 /128 /32	0.40 Hz 0.80 Hz 3.21 Hz

**Note:** The software can decrement the frame frequency by simply incrementing the LCD\_FRQ register.

### 17.3.3 Common driver

Common signals are generated by a common driver which is a programmable ring counter (see [Figure 57](#)).

#### COM signal bias

Each common signal (COMn) has identical waveforms but different phases. It has the maximum amplitude V<sub>LCD</sub> or V<sub>SS</sub> only during the phase n of a frame cycle. During the other phases, the signal amplitude is

- 1/4 V<sub>LCD</sub> or 3/4 V<sub>LCD</sub> in case of 1/4 bias (high density devices only)
- 1/3 V<sub>LCD</sub> or 2/3 V<sub>LCD</sub> in case of 1/3 bias
- 1/2 V<sub>LCD</sub> in case of 1/2 bias.

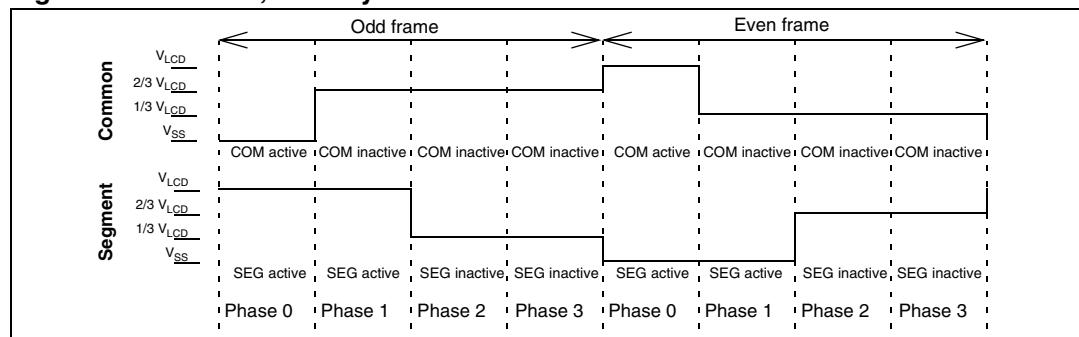
**Note:** The LCD controller generates only one type of LCD waveforms (waveform consuming less power). The waveforms are described in [Figure 59](#), [Figure 60](#), [Figure 61](#) and [Figure 62](#).

Selection between 1/2, 1/3 and 1/4 bias mode can be done by programming the B2 bit in the LCD\_CR1 register.

- the B2 bit in the LCD\_CR1 register and the B4 bit in the LCD\_CR4 register in medium+ and high density devices
- the B2 bit in the LCD\_CR1 register in medium density devices

A pixel is activated when both of its corresponding common and segment lines have their maximum amplitudes during the same phase. As shown in [Figure 57](#), common signal in even frames vs. odd frames are phase inverted in order to reduce EMI (electromagnetic interference). The first frame generated is the odd one followed by an even one.

**Figure 57. 1/3 bias, 1/4 duty**



In case of 1/2 bias (B2 bit set in the LCD\_CR1 register in medium, medium+ and high density devices and B4 bit reset in the LCD\_CR4 register in medium+ and high density devices), an intermediate voltage equal to 1/2  $V_{LCD}$  is generated for even and odd frames;

The input of this common driver is ck\_div which is used to generate common waveforms.

### COM signal duty

#### In medium+ and high density devices:

Common signals are generated depending on the DUTY[1:0] bits in the LCD\_CR1 register and on the DUTY8 bit in the LCD\_CR4 register. Five duty ratios can be selected:

- Static duty (see [Figure 59: Liquid crystal display and terminal connection \(static duty\) on page 261](#))
- 1/2 duty (see [Figure 60: Liquid crystal display and terminal connection \(1/2 duty, 1/2 bias\) on page 262](#))
- 1/3 duty (see [Figure 61: Liquid crystal display and terminal connection \(1/3 duty, 1/3 bias\) on page 263](#))
- 1/4 duty (see [Figure 62: Liquid crystal display and terminal connection \(1/4 duty, 1/3 bias\) on page 264](#))
- 1/8 duty (see [Figure 63: Liquid crystal display and terminal connection \(1/8 duty, 1/4 bias\) on page 265](#))

**In medium density devices:**

Common signals are generated depending on the DUTY[1:0] bits in the LCD\_CR1 register. Four duty ratios can be selected:

- Static duty (see [Figure 59: Liquid crystal display and terminal connection \(static duty\) on page 261](#))
- 1/2 duty (see [Figure 60: Liquid crystal display and terminal connection \(1/2 duty, 1/2 bias\) on page 262](#))
- 1/3 duty (see [Figure 61: Liquid crystal display and terminal connection \(1/3 duty, 1/3 bias\) on page 263](#))
- 1/4 duty (see [Figure 62: Liquid crystal display and terminal connection \(1/4 duty, 1/3 bias\) on page 264](#))

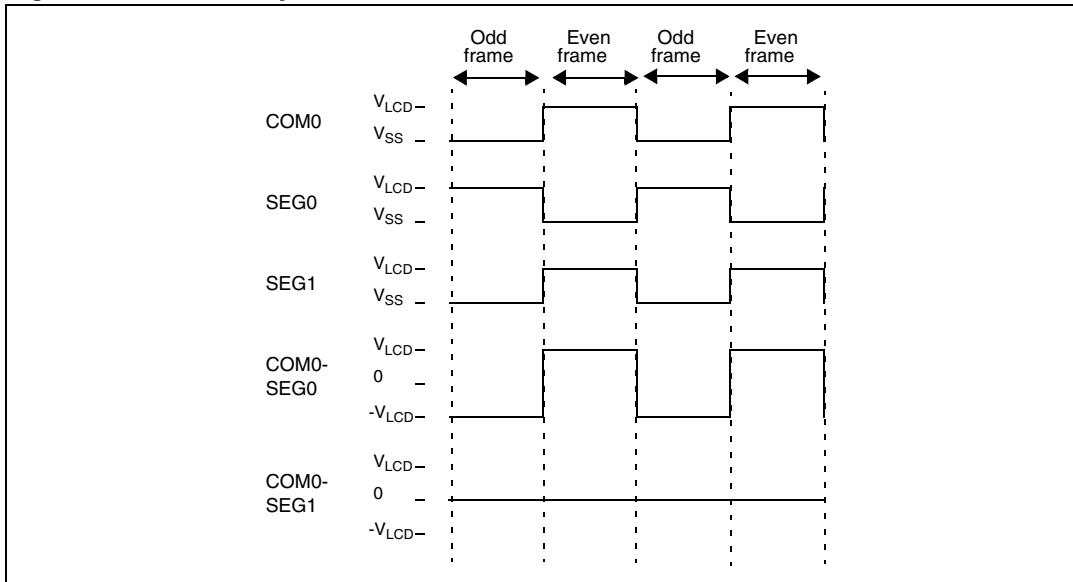
The 'n' value has an impact on COMn behaviors:

- COMn (with 'n' from 0 to 3 in medium density devices and from 0 to 7 in medium+ and high density devices) is active
  - during phase 'n' of an even frame. The COMn pin is then driven to  $V_{LCD}$
  - during phase 'n' of an odd frame. The COMn pin is then driven to  $V_{SS}$
- COMn is inactive during a phase not equal to 'n'
  - In case of 1/4 bias (medium+ and high density devices only), the COMn pin is then driven to 1/4  $V_{LCD}$  during an even frame and to 3/4  $V_{LCD}$  during an odd frame
  - In case of 1/3 bias, the COMn pin is then driven to 1/3  $V_{LCD}$  during an even frame and to 2/3  $V_{LCD}$  during an odd frame
  - In case of 1/2 bias, the COMn pin is then always driven to 1/2  $V_{LCD}$  (odd and even frame).

When static duty is selected, the segment lines are not multiplexed, which means that each segment output corresponds to one pixel. In this way only 28 pixels in the medium density devices and up to 44 pixels in the medium+ and high density devices can be driven. COM0 is always active and only two voltage levels are used for the segment and common lines:  $V_{LCD}$  and  $V_{SS}$ , while COM[3:1] in medium density devices or COM[7:1] in medium+ and high density devices are not used and are driven to  $V_{SS}$ .

A pixel is active if the corresponding segment line gets a voltage different from that of the common line. It is inactive when both voltages are the same. In this case, the LCD has the maximum contrast (see [Figure 58](#) and [Figure 59](#)).

In the following figure, pixel 0 is active while pixel 1 is inactive.

**Figure 58. Static duty**

In each frame there is only one phase, this is why  $f_{frame}$  is equal to  $f_{ck\_div}$ .

In case of 1/4 duty:

If 1/4 duty is selected there are four phases in a frame in which COM0 is active during phase 0, COM1 is active during phase 1, COM2 is active during phase 2, and COM3 is active during phase 3.

In this mode, the segment terminals are multiplexed and each of them control four pixels. A pixel is activated only when both of its corresponding segments and common lines are active in the same phase.

- To deactivate pixel 0 connected to COM0, SEG0 needs to be inactive during the phase 0 when COM0 is active.
- To activate pixel 27 connected to COM1, SEG27 needs to be active during phase 1 when COM1 is active (see [Figure 62](#)).
- To activate pixels from 0 to 27 connected to COM0, SEG[0:27] need to be active during phase 0 when CM0 is active.

These rules can be applied to all other pixels.

## 8 to 1 multiplexer

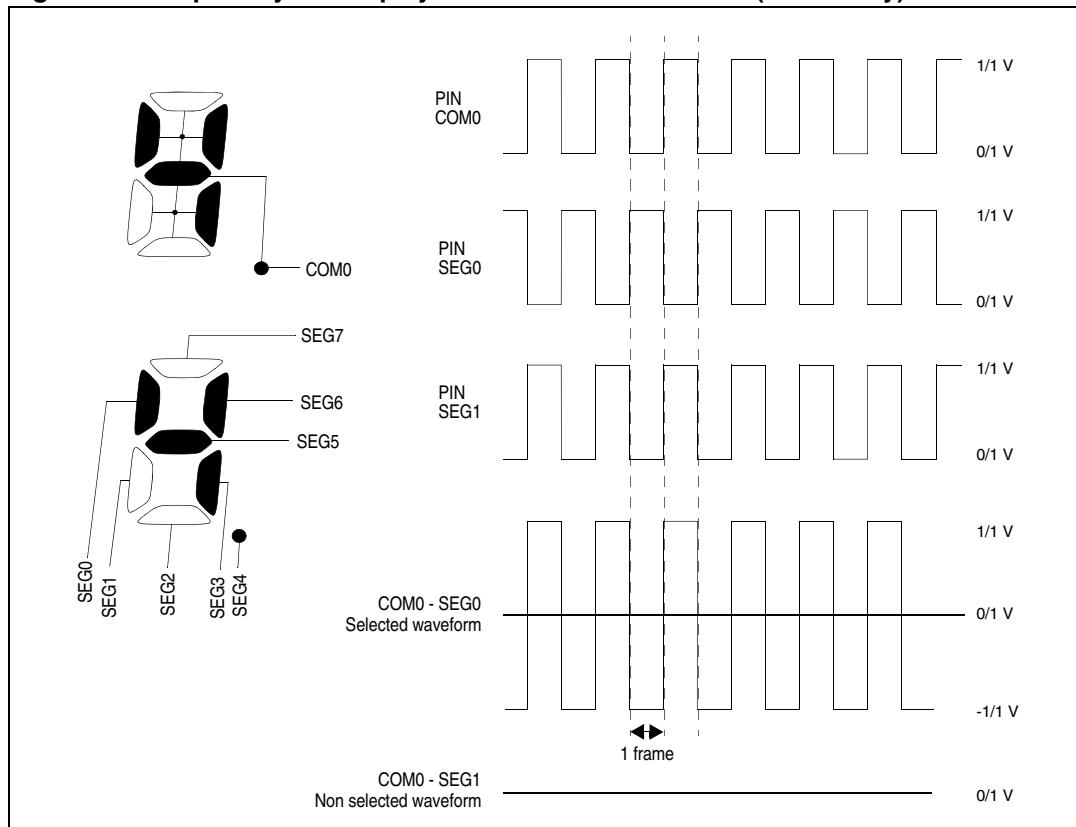
In medium+ and high density devices, when COM[0] is active, the common driver block also drives the 8 to 1 multiplexer shown in [Figure 55: Medium + and high density LCD controller block diagram](#) in order to select the content of RAM register locations corresponding to COM[0]. When COM[7] is active, the output of the 8 to 1 multiplexer is the content of the RAM locations corresponding to COM[7]. Refer to [Section 17.6.7: LCD display memory \(LCD\\_RAM\) on page 277](#).

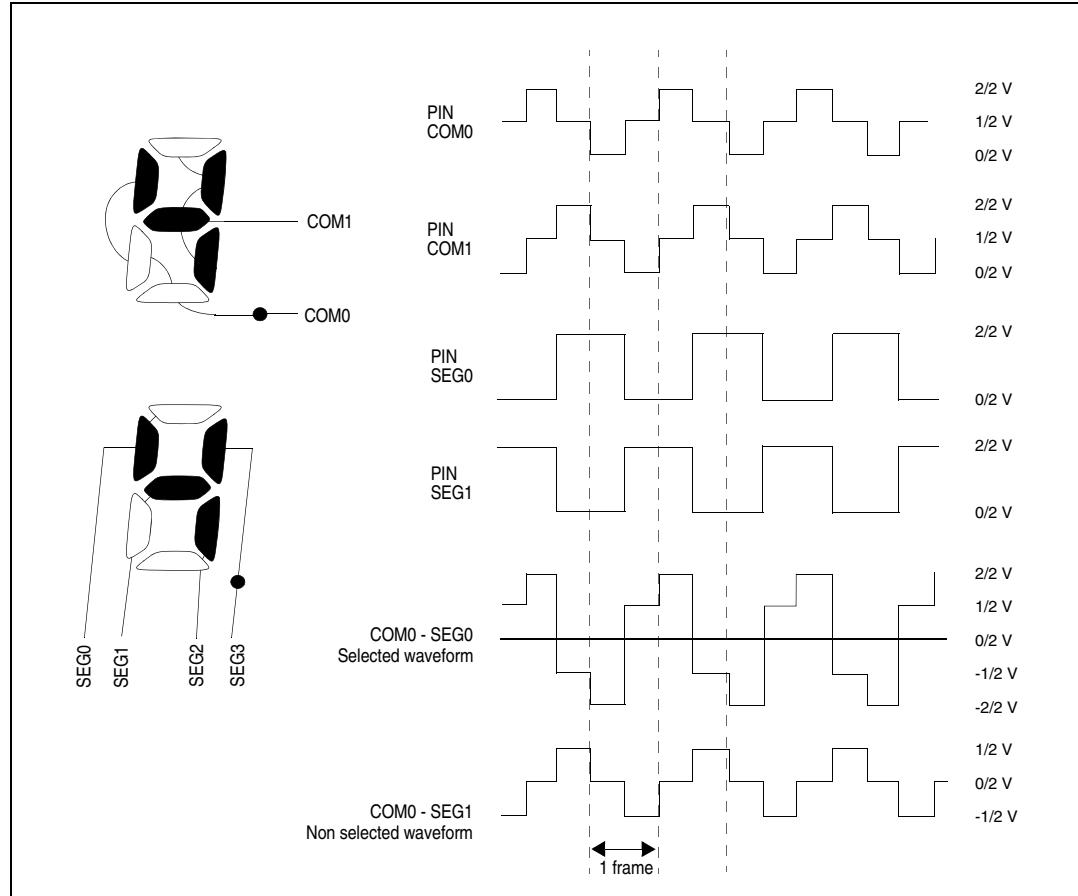
## 4 to 1 multiplexer

In medium density devices, when COM[0] is active, the common driver block also drives the 4 to 1 multiplexer shown in [Figure 56: Medium density LCD controller block diagram](#) in order to select the content of RAM register locations corresponding to COM[0]. When COM[4] is

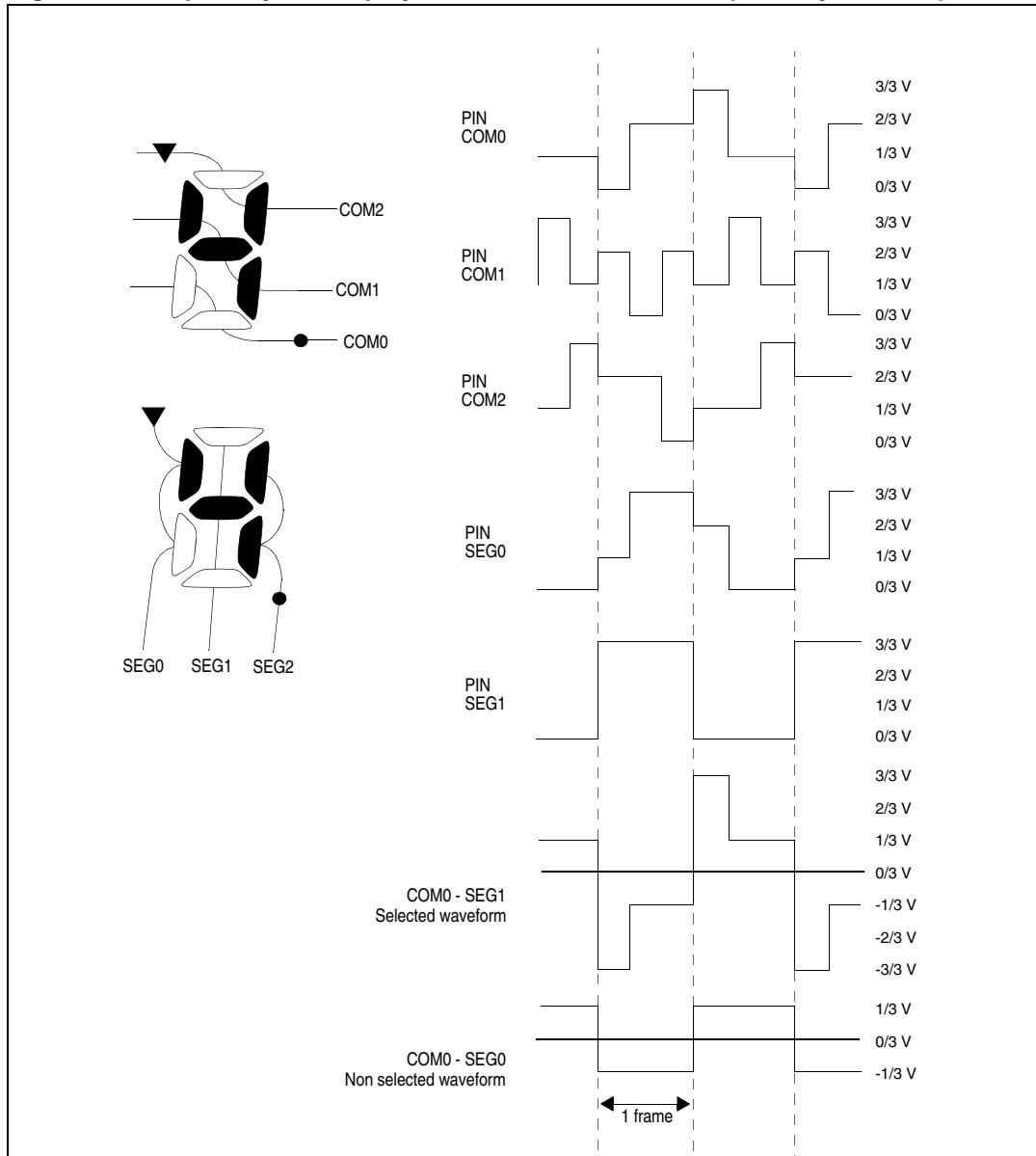
active, the output of the 4 to 1 multiplexer is the content of the RAM locations corresponding to COM[4]. Refer to [Section 17.6.7: LCD display memory \(LCD\\_RAM\) on page 277](#).

**Figure 59. Liquid crystal display and terminal connection (static duty)**

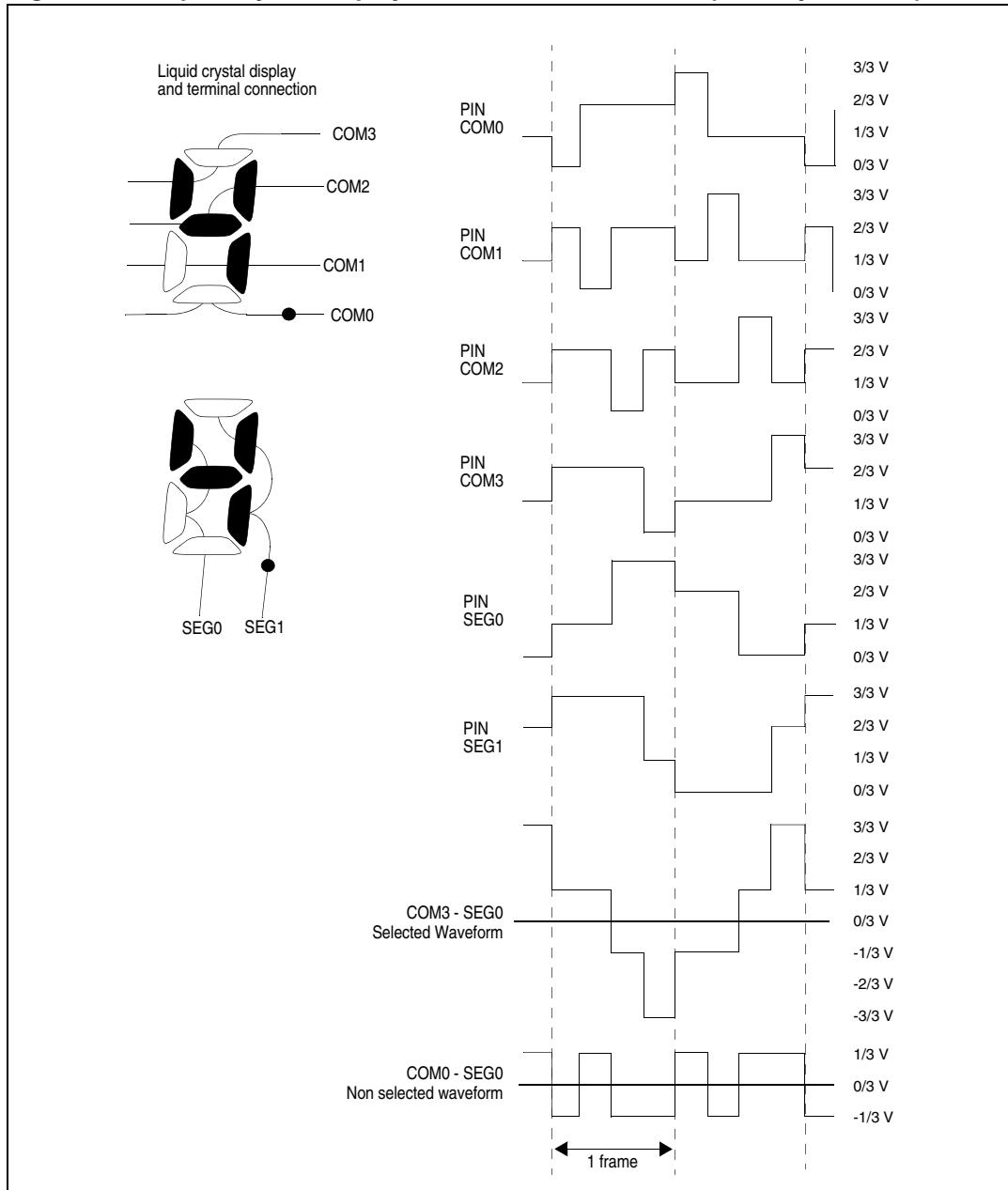


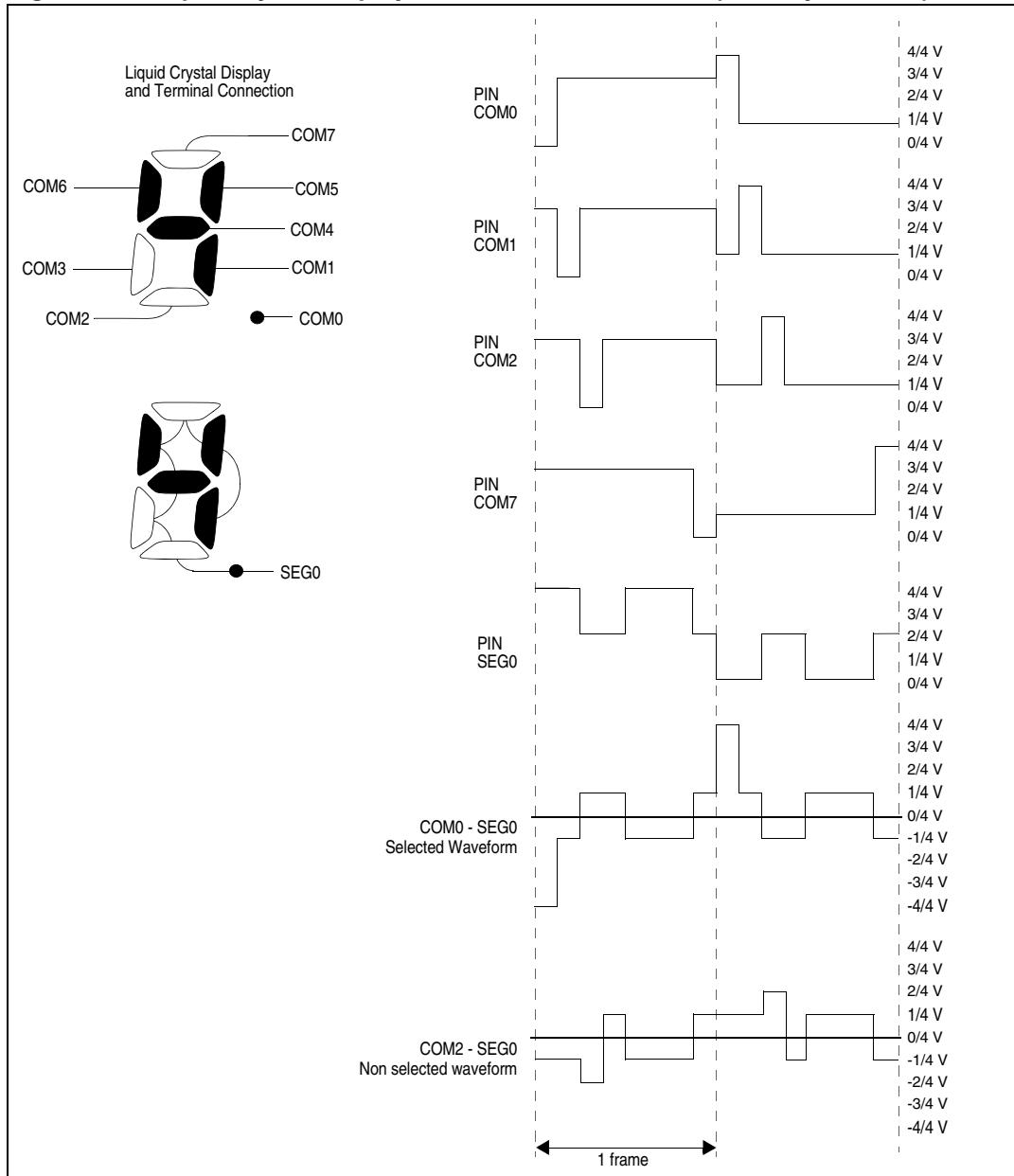
**Figure 60. Liquid crystal display and terminal connection (1/2 duty, 1/2 bias)**

**Figure 61. Liquid crystal display and terminal connection (1/3 duty, 1/3 bias)**



**Figure 62.** Liquid crystal display and terminal connection (1/4 duty, 1/3 bias)



**Figure 63. Liquid crystal display and terminal connection (1/8 duty, 1/4 bias)**

1. Available on medium+ and high density devices only.

#### 17.3.4 Segment driver

The segment driver controls the segments depending on the input pixel data coming from the 4 to 1 multiplexers (in the medium density devices) or from 8 to 1 multiplexers (in the medium+ and high density devices) driven in each phase by the common driver.

In case of 1/4 duty, when COM0 is active, pixel information (active/inactive) relative to pixel connected to COM0 goes through the 4 to 1 multiplexer.

In case of 1/8 duty, when COM0 is active, the pixel information (active/inactive) relative to the pixel connected to COM0 goes through the 8 to 1 multiplexer.

The segment driver can operate in different ways:

- If pixel 'n' is active when COM0 is active:
  - the segment driver drives the SEG<sub>n</sub> pin (with 'n' from 0 to 27 in medium density devices or from 0 to 43 in medium+ and high density devices) to V<sub>SS</sub> in the phase 0 of an even frame
  - the segment driver drives the SEG<sub>n</sub> pin to V<sub>LCD</sub> in the phase 0 of an odd frame
- If pixel 'n' is inactive
  - in case of 1/4 bias (in medium+ and high density devices) the segment driver drives the SEG<sub>n</sub> pin to 1/2 V<sub>LCD</sub> in an even frame and to 1/2 V<sub>LCD</sub> in an odd frame
  - in case of 1/3 bias the segment driver drives the SEG<sub>n</sub> pin to 2/3 V<sub>LCD</sub> in an even frame. the segment driver drives the SEG<sub>n</sub> pin to 1/3 V<sub>LCD</sub> in an odd frame.
  - in case of 1/2 bias SEG<sub>n</sub> pin is driven to V<sub>LCD</sub> in an even frame and to V<sub>SS</sub> in an odd frame (see [Figure 60: Liquid crystal display and terminal connection \(1/2 duty, 1/2 bias\)](#)).

The segment driver also performs LCD pin multiplexing as general I/O.

### 17.3.5 Enabling a segment

Each segment pin (SEG[27:0] or SEG[43:0]) can be enabled or disabled (configured as general purpose I/O) independently through 4 or 6 LCD\_PM registers. If the LCDEN bit in the LCD\_CR3 register is reset and pins are configured as segment pins, segment lines are then pulled down to V<sub>SS</sub> at the end of the even frame.

### 17.3.6 Blink

The segment driver also implements a programmable blink feature. It is possible to select the number of blink pixels through the BLINK[1:0] bits in the LCD\_CR1 register:

- a single pixel (COM0, SEG0),
- all the pixels attached to SEG0 (up to 4),
- or all pixels.

To do this, the corresponding bits in the LCD data RAM must be programmed.

The blink prescaler, through the BLINKF[2:0] bits, selects a wide range of blink frequencies, including 0.5 Hz, 1 Hz, 2 Hz or 4 Hz.

### 17.3.7 Multiplexing COM[7:4] and SEG[43:40], SEG[39:36], or SEG[31:28]

In medium+ and high density devices, the LCD controller can drive Nx4 or (N-4)x8 pixels through the N+4 available LCD pins. The COM4..7 signals and some SEG signals (depending on packages) share the same four pins and the multiplexing is controlled by the DUTY8 bit in the LCD\_CR4 register.

*Note: "N" values can be 44, 40 or 28 respectively on 80-pin, 64-pin or 48-pin packages.*

*For STM8L05xx value line devices, "N" value is 28 on 64-pin or 48-pin packages.*

**Table 61. SEG and COM signal mapping**

Packages	DUTY8 = 0	DUTY8 = 1
80-pin packages	SEG40..43 are mapped respectively on ports PF4...PF7.	COM4...7 are mapped respectively on ports PF4...PF7
64-pin packages	SEG36..39 are mapped respectively on ports PF4...PF7.	COM4...7 are mapped respectively on ports PF4...PF7
48-pin packages	SEG24..27 are mapped respectively on ports PC4, PC7, PE6 and PE7	COM4...7 are mapped respectively on ports PC4, PC7, PE6 and PE7

- On 80-pin packages: The LCD controller can drive 44x4 or 40x8 pixels through the 48 available LCD pins. The COM4..7 signals and the SEG40..43 signals share the same four pins and the multiplexing is controlled by the DUTY8 bit in the LCD\_CR4 register:
    - if DUTY8=0, the SEG40..43 are respectively mapped on PF4..PF7 ports.
    - if DUTY8=1, the COM4..7 are respectively mapped on PF4..PF7 ports.
  - On 64-pin package: The LCD controller can drive 40x4 or 36x8 pixels<sup>(1)</sup> through the 44 available LCD pins. The COM4..7 signals and the SEG36..39 signals share the same four pins and the multiplexing is controlled by the DUTY8 bit in the LCD\_CR4 register:
    - if DUTY8=0, the SEG36..39 are respectively mapped on PF4..PF7 ports.
    - if DUTY8=1, the COM4..7 are respectively mapped on PF4..PF7 ports.
  - On 48-pin package: The LCD controller can drive 28x4 or 24x8 pixels<sup>(1)</sup> through the 32 available LCD pins. The COM4..7 signals and the SEG24..27 signals share the same four pins and the multiplexing is controlled by the DUTY8 bit in the LCD\_CR4 register:
    - if DUTY8=0, the SEG24..27 are respectively mapped PC4, PC7, PE6 and PE7 on ports.
    - if DUTY8=1, the COM4..7 are respectively mapped on PC4, PC7, PE6 and PE7 ports.
1. On STM8L05xx value line devices, the LCD controller can drive 28x4 or 24x8 pixels for 64-pin package and 28x4 pixels for 48-pin package. 80-pin package is not available.

### 17.3.8 Generation of LCD voltage levels

The LCD voltage levels are generated internally using:

- an internal booster which generates  $V_{LCD}$ . The user can also provide the  $V_{LCD}$  voltage externally through the VLCD pin if the VSEL bit is set in the LCD\_CR2 register
- an internal resistor divider network which generates all  $V_{LCD}$  intermediate voltages, as shown in [Figure 64](#).

In fact, two resistive networks are used, one with low value resistors (RL) and one with high value resistors (RH) which are respectively used to increase the current during transitions and to reduce power consumption in static state.

The EN switch follows the following rules:

- If LCDEN bit in the LCD\_CR3 register is set, the EN switch is closed.
- When clearing the LCDEN bit in the LCD\_CR3, the EN switch is open at the end of the even frame in order to avoid a medium voltage level different from 0 during the frame.

The PON[2:0] (Pulse ON duration) bits in the LCD\_CR2 register configure the time during which RL is enabled (see [Figure 55](#)) through a HD (high drive) when the levels of common and segment lines change. A short drive time decreases power consumption, but displays

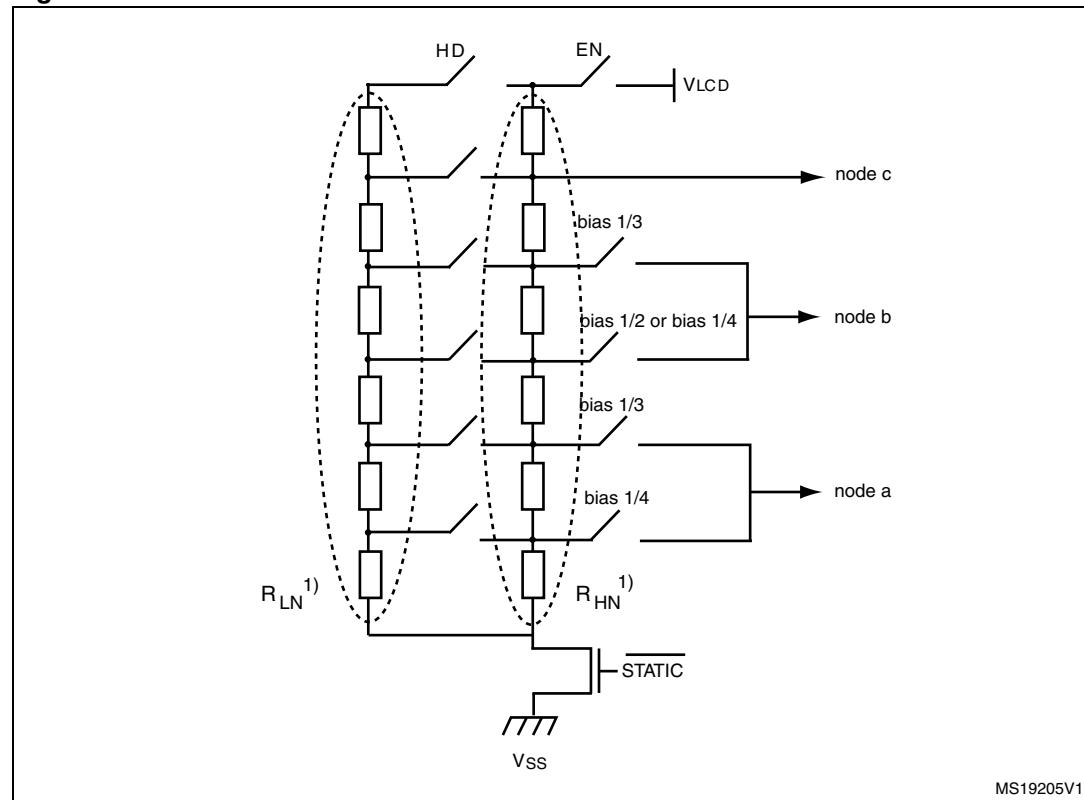
with high internal resistance may need a longer drive time to achieve a satisfactory contrast. The  $R_L$  divider can be always switched on using the HD bit in the LCD\_CR2 register.

The HD switch follows the rules described below:

- If the HD bit and the PON[2:0] bits in the LCD\_CR2 are reset, then HD switch is open.
- If the HD bit in the LCD\_CR2 register is reset and the PON[2:0] bits in the LCD\_CR2 are different from 00 then, the HD switch is closed during the number of pulses defined in the PON[2:0] bits.
- If HD bit in the LCD\_CR2 register is 1 then HD switch is always closed.

The  $V_{LCD}$  value can be chosen among a wide set of values from  $V_{LCDmin}$  to  $V_{LCDmax}$  using the CC[2:0] bits (contrast control bits) in the LCD\_CR2 register (see [Figure 55](#)). New VLCD values take effect at each beginning of a new frame.

**Figure 64. Resistive network**



1.  $R_{LN}$ : Low value resistor network.  $R_{HN}$ : High value resistor network.

- In case of 1/2 bias, one voltage level ( $1/2 V_{LCD}$ ) is generated and node b voltage is  $1/2 V_{LCD}$ .
- In case of 1/3 bias, two intermediate voltage levels ( $1/3 V_{LCD}$ ,  $2/3 V_{LCD}$ ) are generated
  - node a is  $1/3 V_{LCD}$
  - node b is  $2/3 V_{LCD}$
- In case of 1/4 bias (medium+ and high density devices only), three intermediate voltage levels ( $1/4 V_{LCD}$ ,  $1/2 V_{LCD}$  and  $3/4 V_{LCD}$ ) are generated
  - node a is  $1/4 V_{LCD}$
  - node b is  $1/2 V_{LCD}$
  - node c is  $3/4 V_{LCD}$ .

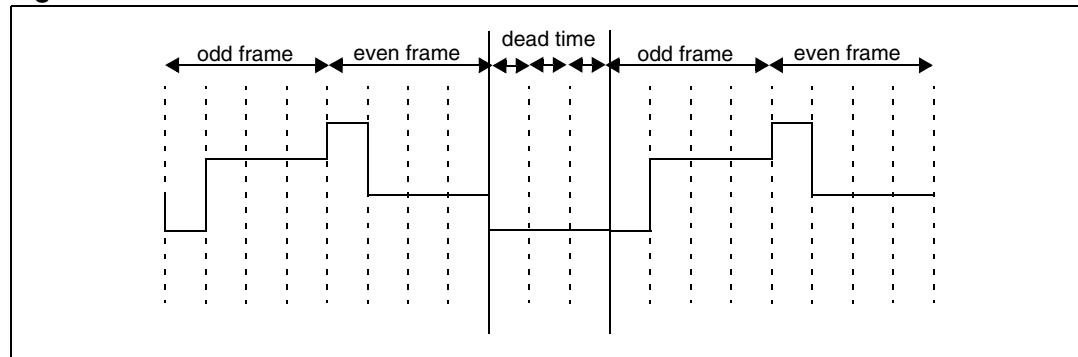
Internal or external  $V_{LCD}$  source can be selected using the VSEL bits of the LCD\_CR2 register.

*Note:* In case the internal  $V_{LCD}$  source is selected (VSEL=0) and the LCD is used in Active-halt mode, the ULP bit (bit 1) in the PWR\_CSR2 register must be programmed to '0' because the internal  $V_{LCD}$  source needs the internal reference voltage.

### Dead time

In addition to using the CC2[2:0] bits, the contrast can be controlled by programming a dead time between each couple of frames where the COM and SEG values are tied to  $V_{SS}$  at the same time. The DEAD[0:2] bits in the LCD\_CR3 register can be used to program a dead time of up to 7 phase periods.

**Figure 65. Contrast control**



### 17.3.9 LCD buffer update

Each time the LCD message is to be changed, LCD\_RAM content is to be updated with new data.

Single buffered memory does not allow data in LCD\_RAM buffer to be updated at any time by the application firmware without affecting the integrity of data displayed. So, to avoid any display unexpected behavior, the application firmware must make sure that LCD\_RAM content is updated synchronously with the start of frame.

To update the LCD\_RAM content, proceed as follow:

- Clear SOF flag by setting SOFC bit in LCD\_CR3
- Wait until SOF active (start of new frame) by polling SOF bit in LCD\_CR3 until it is asserted or waiting until the LCD Interrupt is serviced (if the SOFIE bit is set)
- Update LCD\_RAM buffer

*Note:* *Thereby, LCD\_RAM refresh rate should not be faster than the LCD frame frequency. Otherwise it might result in lower quality display results like pixel blinking.*

## 17.4 LCD controller low power modes

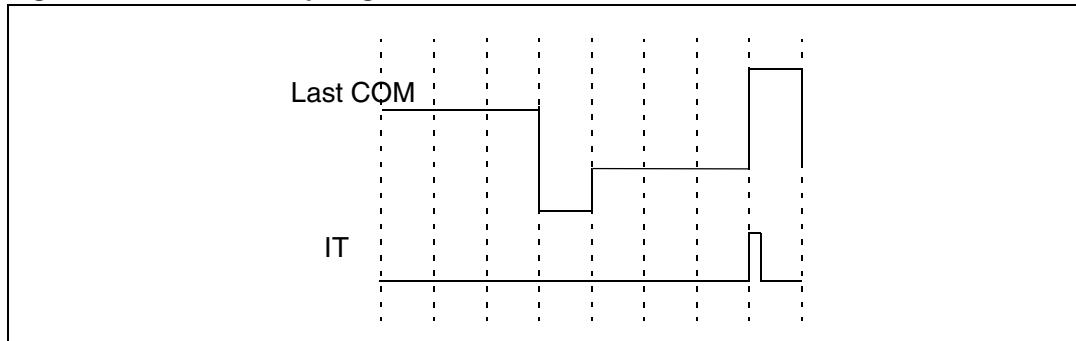
**Table 62. LCD behavior in low power modes**

Mode	Description
Low power wait /Wait	The LCD is still active.
Active-halt	The LCD is still active. If internal VLCD booster is used, ULP bit must be '0' in PWR_CSR2.
Halt	The LCD is not active.

*Note:* *The device can enter Halt mode only when the RTC is OFF, or when HSE or HSI is used as RTC clock source. The device is in Active-halt mode when LSE or LSI is selected.*

## 17.5 LCD controller interrupts

The common driver can also generate a start of new frame flag through the SOF bit in the LCD\_CR3 register. This bit is set at the beginning of the last COM of the even frame. At this moment, the last segments are displayed and the CPU can update the data RAM to avoid displaying flickering data on the LCD panel.

**Figure 66. SOF interrupt signal**

The LCD start of frame interrupt is generated if the SOFIE bit (interrupt enable bit) in the LCD\_CR3 register is set. The start of frame flag is cleared by software by writing 1 to the SOFC bit in the LCD\_CR3 register.

**Table 63. LCD interrupt requests**

Interrupt event	Event flag	Enable control bit	Exit from Wait	Exit from Halt
Start of frame flag	SOF	SOFIE	yes	no

## 17.6 LCD controller registers

### 17.6.1 Control register 1 (LCD\_CR1)

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
BLINK1	BLINK0	BLINKF2	BLINKF1	BLINKF0	DUTY1	DUTY0	B2
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:6 **BLINK[1:0]**: Blink enable

- 00: Blink inactive
- 01: SEG0/COM0 (1 pixel)
- 10: SEG0 COMx (up to 4 pixels)
- 11: SEGx COMx (all pixels)

Bits 5:3 **BLINKF[2:0]**: Blink frequency

- 000:  $f_{LCD}/8$
- 001:  $f_{LCD}/16$
- 010:  $f_{LCD}/32$
- 011:  $f_{LCD}/64$
- 100:  $f_{LCD}/128$
- 101:  $f_{LCD}/256$
- 110:  $f_{LCD}/512$
- 111:  $f_{LCD}/1024$

Bits 2:1 **DUTY**: Duty ratio selection

- 00: Static duty
- 01: 1/2 duty
- 10: 1/3 duty
- 11: 1/4 duty

Bit 0 **B2**: 1/2 Bias selector

- 0: 1/3 bias
- 1: 1/2 bias

## 17.6.2 Control register 2 (LCD\_CR2)

Address offset: 0x01

Reset value: 0x00

7	6	5	4	3	2	1	0
PON2	PON1	PON0	HD	CC2	CC1	CC0	VSEL
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:5 **PON**: Pulse ON duration

These bits specify the pulse duration (number of CLK<sub>PS</sub> pulses). The pulse duration is PON[2:0] clock cycles coming out from the first prescaler thus determining the amount of time the RL resistive network is turned on for each voltage transition on SEG and COM pins. A short drive time decreases power consumption, but displays with a high internal resistance may need longer drive time to achieve a satisfactory contrast.

*Note: The drive time will never be longer than half the prescaled LCD clock period*

- 000: 0 CLK<sub>PS</sub> pulses
- 001: 1 CLK<sub>PS</sub> pulses
- 010: 2 CLK<sub>PS</sub> pulses
- 011: 3 CLK<sub>PS</sub> pulses
- 100: 4 CLK<sub>PS</sub> pulses
- 101: 5 CLK<sub>PS</sub> pulses
- 110: 6 CLK<sub>PS</sub> pulses
- 111: 7 CLK<sub>PS</sub> pulses

Bit 4 **HD**: High drive enable

This bit permanently enables the low resistance divider. Displays with a high internal resistance may need longer drive time to achieve a satisfactory contrast, so the software can set the HD bit if there is no strong power constraint.

- 0: High drive permanent disabled
- 1: High drive permanent enabled

Bits 3:1 **CC[2:0]**: Contrast control

These bits specify one of the V<sub>LCD</sub> maximum voltage (independent of V<sub>DD</sub>).

The new values will have effect every beginning of a new frame. The contrast controller (Boost circuit) is required inside the voltage generator.

- 000: VLCD0
- 001: VLCD1
- 010: VLCD2
- 011: VLCD3
- 100: VLCD4
- 101: VLCD5
- 110: VLCD6
- 111: VLCD7

*Note: Refer to the device datasheet for the VLCDx values.*

Bit 0 **VSEL**: The VSEL bit determines the LCD voltage source

- 0: Internal source
- 1: External source

*Note: Refer to [Section 17.3.8: Generation of LCD voltage levels](#)*

### 17.6.3 Control register 3 (LCD\_CR3)

Address offset: 0x02

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved	LCDEN	SOFIE	SOF	SOFC	DEAD2	DEAD1	DEAD0	
	rw	rw	r	w	rw	rw	rw	

Bit 7 Reserved, tied to 0

Bit 6 **LCDEN**: LCD enable

Clearing this bit during a frame delays the reset at the end of the even frame in order to avoid a medium voltage level different from 0 during the frame.

0: LCD controller disabled

1: LCD controller enabled

Bit 5 **SOFIE**: Interrupt enable

When this bit is set, the LCD start of frame interrupt is enabled.

0: SOF interrupt disabled

1: SOF interrupt enabled

Bit 4 **SOF**: Start of frame flag

This bit is set by hardware at the beginning of a last COM of an even frame. The LCD Interrupt is serviced if the SOFIE bit is set. SOF is cleared by software by setting the SOFC bit.

0: SOF inactive

1: SOF active (start of new frame)

Bit 3 **SOFC**: SOF flag clear

Setting this bit will clear the SOF flag. This bit is reset by hardware automatically.

Bits 2:0 **DEAD[2:0]**: Dead time enable

These bits determine the number of phases between different couples of frames where the COM and the SEG voltage levels are tied to 0 to reduce the contrast.

Dead = 000: no dead time

Dead = 001: 1-phase period dead time

Dead = 010: 2-phase period dead time

Dead = 011: 3-phase period dead time

Dead = 100: 4-phase period dead time

Dead = 101: 5-phase period dead time

Dead = 110: 6-phase period dead time

Dead = 111: 7-phase period dead time

### 17.6.4 Frequency selection register (LCD\_FRQ)

Address offset: 0x03

Reset value: 0x00

7	6	5	4	3	2	1	0
PS3	PS2	PS1	PS0	DIV3	DIV2	DIV1	DIV0
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:4 **PS[3:0]**: CLK 16-bit prescaler

The input clock (CLK) goes to a prescaler. The PS[3:0] bits select the prescaler so as to divide the clock by a factor of  $2^{\text{PS}[3:0]}$ . The prescaled output CLK<sub>PS</sub> can be further divided by setting the DIV[3:0] bits. PS and DIV bits determine the prescaled LCD clock (CLK<sub>LCD</sub>) feeding the LCD module.

Bits 3:0 **DIV[3:0]**: Divider from 16 to 31

These bits set the frequency of the LCD clock such that the period can be DIV[3:0]+16 clocks coming out from the first prescaler. This clock divider provides a more flexible frame rate selection.

### 17.6.5 Port mask registers (LCD\_PM)

Address offset: 0x04 to 0x07 (medium density devices) or 0x04 to 0x09 (medium+ and high density devices)

Reset value: 0x00

	7	6	5	4	3	2	1	0
0x04 (LCD_PM0)	SEG07	SEG06	SEG05	SEG04	SEG03	SEG02	SEG01	SEG00
0x05 (LCD_PM1)	SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG09	SEG08
0x06 (LCD_PM2)	SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
0x07 (LCD_PM3)	SEG31 <sup>(1)(2)</sup>	SEG30 <sup>(1)(2)</sup>	SEG29 <sup>(1)(2)</sup>	SEG28 <sup>(1)(2)</sup>	SEG27 <sup>(2)</sup>	SEG26 <sup>(2)</sup>	SEG25 <sup>(2)</sup>	SEG24 <sup>(2)</sup>
0x08 (LCD_PM4)	SEG39 <sup>(1)</sup>	SEG38 <sup>(1)</sup>	SEG37 <sup>(1)</sup>	SEG36 <sup>(1)</sup>	SEG35 <sup>(1)(2)</sup>	SEG34 <sup>(1)(2)</sup>	SEG33 <sup>(1)(2)</sup>	SEG32 <sup>(1)(2)</sup>
0x09 (LCD_PM5)	Reserved				SEG43 <sup>(1)(2)</sup>	SEG42 <sup>(1)(2)</sup>	SEG41 <sup>(1)(2)</sup>	SEG40 <sup>(1)(2)</sup>
	rw							

1. Available on medium+ and high density devices only. Reserved on medium density devices.

2. Reserved on STM8L05xx value line high density devices.

Bits 7:0 **PM[7:0]**: Port mask for SEG[i]

These bits determine the number of port pins to be used as segment drivers.

Unused pins can be used as general purpose I/Os.

0: Pin configured as general purpose I/O

1: Segment enabled

### 17.6.6 Control register 4 (LCD\_CR4)

Address offset: 0x2F

Reset value: 0x00

*Note:* This register is available in medium+ and high density devices only.

7	6	5	4	3	2	1	0
Reserved					PAGE_COM	DUTY8	B4
					rw	rw	rw

Bits 7:3 Reserved. tied to 0

Bit 2 **PAGE\_COM**: LCD\_RAM page selector

0: Addresses 0x0C to 0x021 give access to the first page of the LCD RAM, corresponding to COM0, 1, 2 and 3

- The S0[i] LCD RAM bit corresponds to SEG[i] of COM0
- The S1[i] LCD RAM bit corresponds to SEG[i] of COM1
- The S2[i] LCD RAM bit corresponds to SEG[i] of COM2
- The S3[i] LCD RAM bit corresponds to SEG[i] of COM3

1: Addresses 0x0C to 0x021 give access to the first page of the LCD RAM, corresponding to COM4, 5, 6 and 7

- The S0[i] LCD RAM bit corresponds to SEG[i] of COM4
- The S1[i] LCD RAM bit corresponds to SEG[i] of COM5
- The S2[i] LCD RAM bit corresponds to SEG[i] of COM6
- The S3[i] LCD RAM bit corresponds to SEG[i] of COM7

Bit 1 **DUTY8**: 1/8 duty enable

0: Duty selected through the DUTY [1:0] bits in the LCD\_CR1 register

1: 1/8 duty enabled

Bit 0 **B4**: 1/4 bias enable

0: 1/2 or 1/3 Bias is selected through the B2 bit in the LCD\_CR1 register

1: 1/4 Bias enabled (bit B2 in the LCD\_CR1 register must be reset).

### 17.6.7 LCD display memory (LCD\_RAM)

Address offset: 0x0C to 0x19 (medium density devices) or 0x0C to 0x21 (medium+ and high density devices)

Reset value: 0x00

	7	6	5	4	3	2	1	0
0x0C				S0[7:0] (COM0 or COM4 <sup>(1)</sup> )				
				rw				
0x0D				S0[15:8] (COM0 or COM4 <sup>(1)</sup> )				
				rw				
0x0E				S0[23:16] (COM0 or COM4 <sup>(1)</sup> )				
				rw				
0x0F		S1[3:0] (COM1 or COM5 <sup>(1)</sup> )			S0[27:24] (COM0 or COM4 <sup>(1)</sup> )			
				rw				
0x10				S1[11:4] (COM1 or COM5 <sup>(1)</sup> )				
				rw				
0x11				S1[19:12] (COM1 or COM5 <sup>(1)</sup> )				
				rw				
0x12				S1[27:20] (COM1 or COM5 <sup>(1)</sup> )				
				rw				
0x13				S2[7:0] (COM2 or COM6 <sup>(1)</sup> )				
				rw				
0x14				S2[15:8] (COM2 or COM6 <sup>(1)</sup> )				
				rw				
0x15				S2[23:16] (COM2 or COM6 <sup>(1)</sup> )				
				rw				
0x16		S3[3:0] (COM3 or COM7 <sup>(1)</sup> )			S2[27:24] (COM2 or COM6 <sup>(1)</sup> )			
				rw				
0x17				S3[11:4] (COM3 or COM7 <sup>(1)</sup> )				
				rw				
0x18				S3[19:12] (COM3 or COM7 <sup>(1)</sup> )				
				rw				
0x19				S3[27:20] (COM3 or COM7 <sup>(1)</sup> )				
				rw				
0x1A <sup>(2)</sup>				S0[35:28] <sup>(2)(3)</sup> (COM0 or COM4)				
				rw				

0x1B <sup>(2)</sup>	S0[43:36] <sup>(2)</sup> (COM0 or COM4)
	rw
0x1C <sup>(2)</sup>	S1[35:28] <sup>(2)(3)</sup> (COM1 or COM5)
	rw
0x1D <sup>(2)</sup>	S1[43:36] <sup>(2)</sup> (COM1 or COM5)
	rw
0x1E <sup>(2)</sup>	S2[35:28] <sup>(2)(3)</sup> (COM2 or COM6)
	rw
0x1F <sup>(2)</sup>	S2[43:36] <sup>(2)</sup> (COM2 or COM6)
	rw
0x20 <sup>(2)</sup>	S3[35:28] <sup>(2)(3)</sup> (COM3 or COM7)
	rw
0x21 <sup>(2)</sup>	S3[43:36] <sup>(2)</sup> (COM3 or COM7)
	rw

1. COM4, 5, 6 and 7 are available on medium+ and high density devices only.
2. Register available on medium + and high density devices only. Reserved on medium density devices.
3. Reserved on STM8L05xx value line high density devices.

Each bit corresponds to one pixel of the LCD display.

- 0: Pixel inactive
- 1: Pixel active

#### In medium+ and high density devices:

When setting the DUTY8 bit in the LCD\_CR4 (to activate 8 COMs), the LCD RAM is accessed through two pages, each being activated by the PAGE\_COM bit in the LCD\_CR4 register:

When PAGE\_COM=0, addresses 0x0C to 0x21 give access to the first page, corresponding to COM0, 1, 2 and 3. Refer to the PAGE\_COM bit description.

When PAGE\_COM=1, addresses 0x0C to 0x21 give access to the second page, corresponding to COM4, 5, 6 and 7. Refer to the PAGE\_COM bit description.

#### In medium density devices:

Addresses 0x0C-0x19 give access to the LCD RAM page, corresponding to COM0,1,2 and 3.

- The LCD RAM bits S0[i] corresponds to SEG[i] of COM0
- The LCD RAM bits S1[i] corresponds to SEG[i] of COM1
- The LCD RAM bits S2[i] corresponds to SEG[i] of COM2
- The LCD RAM bits S3[i] corresponds to SEG[i] of COM3.

### 17.6.8 LCD register map and reset values

**Table 64. LCD register map**

Address offset	Register name	7	6	5	4	3	2	1	0				
0x00	LCD_CR1 Reset value	BLINK1 0	BLINK0 0	BLINKF2 0	BLINKF1 0	BLINKF0 0	DUTY1 0	DUTY0 0	B2 0				
0x01	LCD_CR2 Reset value	PON2 0	PON1 0	PON0 0	HD 0	CC2 0	CC1 0	CC0 0	VSEL 0				
0x02	LCD_CR3 Reset value	Reserved 0	LCDEN 0	SOFIE 0	SOF 0	SOFC 0	DEAD2 0	DEAD1 0	DEAD0 0				
0x03	LCD_FRQ Reset value	PS3 0	PS2 0	PS1 0	PS0 0	DIV3 0	DIV2 0	DIV1 0	DIV0 0				
0x04	LCD_PM0 Reset value	PM_SEG[7:0] 00000000											
0x05	LCD_PM1 Reset value	PM_SEG[15:8] 00000000											
0x06	LCD_PM2 Reset value	PM_SEG[23:16] 00000000											
0x07	LCD_PM3 Reset value	PM_SEG[31:24] <sup>(1)</sup> 00000000											
0x08	LCD_PM4 Reset value	PM_SEG[39:32] <sup>(1)(2)</sup> 00000000											
0x09	LCD_PM5 Reset value	Reserved				SEG[43:40] <sup>(1)(3)</sup> 00000000							
0x0A to 0x0B	Reserved	Reserved											
0x0C	LCD_RAM0 Reset value	S0[7:0] (COM0 or COM4) <sup>(4)</sup> 00000000											
0x0D	LCD_RAM1 Reset value	S0[15:8] (COM0 or COM4) <sup>(4)</sup> 00000000											
0x0E	LCD_RAM2 Reset value	S0[23:16] (COM0 or COM4) <sup>(4)</sup> 00000000											
0x0F	LCD_RAM3 Reset value	S1[3:0] (COM1 or COM5) <sup>(4)</sup> 0000				S0[27:24] (COM0 or COM4) <sup>(4)</sup> 0000							
0x10	LCD_RAM4 Reset value	S1[11:4] (COM1 or COM5) <sup>(4)</sup> 00000000											
0x11	LCD_RAM5 Reset value	S1[19:12] (COM1 or COM5) <sup>(4)</sup> 00000000											
0x12	LCD_RAM6 Reset value	S1[27:20] (COM1 or COM5) <sup>(4)</sup> 00000000											
0x13	LCD_RAM7 Reset value	S2[7:0] (COM2 or COM6) <sup>(4)</sup> 00000000											
0x14	LCD_RAM8 Reset value	S2[15:8] (COM2 or COM6) <sup>(4)</sup> 00000000											
0x15	LCD_RAM9 Reset value	S2[23:16] (COM2 or COM6) <sup>(4)</sup> 00000000											
0x16	LCD_RAM10 Reset value	S3[3:0] (COM3 or COM7) <sup>(4)</sup> 0000				S2[27:24] (COM2 or COM6) <sup>(4)</sup> 0000							
0x17	LCD_RAM11 Reset value	S3[11:4] (COM3 or COM7) <sup>(4)</sup> 00000000											

**Table 64. LCD register map (continued)**

<b>Address offset</b>	<b>Register name</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
0x18	LCD_RAM12 Reset value				S3[19:12] (COM3 or COM7) <sup>(4)</sup> 00000000				
0x19	LCD_RAM13 Reset value				S0[27:20] (COM3 or COM7) <sup>(4)</sup> 00000000				
0x1A	LCD_RAM14 <sup>(3)</sup> Reset value				S0[35:28] (COM0 or COM4) <sup>(3)(5)</sup> 00000000				
0x1B	LCD_RAM15 <sup>(3)</sup> Reset value				S0[43:36] (COM0 or COM4) <sup>(3)</sup> 00000000				
0x1C	LCD_RAM16 <sup>(3)</sup> Reset value				S1[35:28] (COM1 or COM5) <sup>(3)(5)</sup> 00000000				
0x1D	LCD_RAM17 <sup>(3)</sup> Reset value				S1[43:36] (COM1 or COM5) <sup>(3)</sup> 00000000				
0x1E	LCD_RAM18 <sup>(3)</sup> Reset value				S2[35:28] (COM2 or COM6) <sup>(3)(5)</sup> 00000000				
0x1F	LCD_RAM19 <sup>(3)</sup> Reset value				S2[43:36] (COM2 or COM6) <sup>(3)</sup> 00000000				
0x20	LCD_RAM20 <sup>(3)</sup> Reset value				S3[35:28] (COM3 or COM7) <sup>(3)(5)</sup> 00000000				
0x21	LCD_RAM21 <sup>(3)</sup> Reset value				S3[43:36] (COM3 or COM7) <sup>(3)</sup> 00000000				
0x22 to 0x2E	Reserved Reset value				Reserved 00000000				
0x2F	LCD_CR4 <sup>(3)</sup> Reset value				Reserved 00000	PAGE_COM_0	DUTY8_0	B4_0	

1. On STM8L05xx value line devices, refer to datasheet for more details.
2. PM\_SEG[39:28] available on medium+ and high density devices only. Reserved on medium density devices.
3. Available on medium+ and high density devices only. Reserved on medium density devices.
4. COM4, 5, 6 and 7 are available on medium+ and high density devices only and are activated by setting the DUTY8 bit in the LCD\_CR4.
5. Not available on STM8L05xx value line high density devices.

## 18 Timer overview

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

The devices may be equipped with up to three different timer types: Advanced control (TIM1), general purpose (TIM2/TIM3/TIM5), and basic timer (TIM4). The timers share the same architecture, but some have additional unique features. The common timer architecture, which includes identical register mapping and common basic features, simplifies their use and makes it easier to design applications. [Table 65](#) shows the main timer characteristics.

The TIM5 general purpose timer is available on low, medium+ and high density devices only (not on medium density devices). The TIM1 advanced control timer is not available on low density devices. Refer to the product datasheets for more details.

Although the timers do not share any resources, they can be linked together and synchronized.

This section gives a comparison of the different timer features ([Table 66](#)), a glossary of internal timer signal names ([Table 67](#)).

[Section 19: 16-bit advanced control timer \(TIM1\)](#) contains a full description of all the various timer modes. The other timer sections ([Section 20](#) and [Section 21](#)) are more brief and give only specific details on each timer, its block diagram, and register description.

**Table 65. Timer characteristics**

Symbol	Parameter	Min	Typ	Max	Unit
$t_{w(ICAP)in}$	Input capture pulse time	2			$1/f_{SYSCLK}$
$t_{res(TIM)}$	Timer resolution time	1			$1/f_{SYSCLK}$
$Res_{TIM}$	Timer resolution with 16-bit counter		16		bit
	Timer resolution with 8-bit counter		8		bit
$t_{COUNTER}$	Counter clock period when internal clock is selected		1		$1/f_{SYSCLK}$
$t_{MAX\_COUNT}$	Maximum possible count with 16-bit counter			65,536	$1/f_{SYSCLK}$
	Maximum possible count with 8-bit counter			256	$1/f_{SYSCLK}$

## 18.1 Timer feature comparison

Table 66. Timer feature comparison

Timer	Counter resolution	Counter type	Prescaler factor	Capture/compare channels	Complementary outputs	Repetition counter	External trigger input	External break input	Timer synchronization/chaining
TIM1 (advanced control timer)	16-bit	Up/down	Any integer from 1 to 65536	3 + 1	3	Yes	1	1	Yes
TIM2, TIM3 and TIM5 (general purpose timers)		Up/down	Any power of 2 from 1 to 128	2	None	No	1	1	
TIM4 (basic timer)		Up	Any power of 2 from 1 to 32768	0			0	0	

## 18.2 Glossary of timer signal names

Table 67. Glossary of internal timer signals

Internal signal name	Description	Related figures
BI	Break interrupt	<a href="#">Figure 67: TIM1 general block diagram on page 286</a>
CC <i>i</i> : CC1I, CC2I, CC3I, CC4I	Capture/compare interrupt	
CK_CNT	Counter clock	<a href="#">Figure 71: Counter update when ARPE = 0 (ARR not preloaded) with prescaler = 2 on page 290</a>
CK_PSC	Prescaler clock	
CNT_EN	Counter enable	<a href="#">Figure 81: TI2 external clock connection example on page 298</a>
CNT_INIT	Counter initialize	
ETR	External trigger from TIMx_ETR pin	<a href="#">Figure 83: External trigger input block diagram on page 300</a>
ETRF	External trigger filtered	
ETRP	External trigger prescaled	
f <sub>SYSCLK</sub>	Timer peripheral clock from clock controller (CLK)	<a href="#">Figure 17: Clock structure on page 88</a>
IC <i>i</i> : IC1, IC2	Input capture	<a href="#">Figure 100: Input stage of TIM 1 channel 1 on page 313</a>
IC/PS: IC1PS, IC2PS	Input capture prescaled	

**Table 67. Glossary of internal timer signals (continued)**

Internal signal name	Description	Related figures
ITR $i$ : ITR0, ITR1, ITR2, ITR3	Internal trigger input tied to TRGO of other TIM timers	<a href="#">Figure 67: TIM1 general block diagram on page 286</a>
MATCH1	Compare match	<a href="#">Figure 90: Trigger/master mode selection blocks on page 306</a> and <a href="#">Section 19.8.2: Control register 2 (TIM1_CR2) on page 338</a>
OC $i$ : OC1, OC2	Timer output channel	<a href="#">Figure 125: Output stage of channel 1 on page 373</a>
OC $i$ /REF: OC1REF, OC2REF	Output compare reference signal	
TGI	Trigger interrupt	<a href="#">Figure 79: Clock/trigger controller block diagram on page 297</a>
TI $i$ : TI1, TI2	Timer input	<a href="#">Figure 100: Input stage of TIM 1 channel 1 on page 313</a>
TI $i$ F: TI1F, TI2F	Timer input filtered	
TI1_ED	Timer input edge detector	
TI $i$ /FPn: TI1FP1, TI1FP2, TI2FP1, TI2FP2, TI3FP3, TI3FP4, TI4FP3, TI4FP4	Timer input filtered prescaled	
TRC	Trigger capture	<a href="#">Figure 80: Control circuit in normal mode, f<sub>CK_PSC</sub> = f<sub>SYSCLK</sub> on page 298</a>
TRGI	Trigger input to clock/trigger/slave mode controller	
TRGO	Trigger output tied to trigger input INTx of other timers	<a href="#">Figure 67: TIM1 general block diagram on page 286</a>
UEV	Update event	<a href="#">Figure 71: Counter update when ARPE = 0 (ARR not preloaded) with prescaler = 2 on page 290</a>
UIF	Update interrupt	

**Table 68. Explanation of indices ‘ $i$ ’, ‘ $n$ ’, and ‘ $x$ ’<sup>(1)</sup>**

$i$	Signal number: May be 1, 2, 3, 4 depending on the device
	Bit number: May be 1, 2, 3, 4 ..... depending on the device
	Register number: May be 1, 2, 3, 4 depending on the device
$n$	Signal number (when $i$ is already used): May be 1, 2, 3, 4 depending on the device
$x$	Timer number: May be 1, 2, 3, 4, 5, 6 depending on the device
	Don't care (for bits)

1. These indices are used in [Section 19](#), [Section 20](#), and [Section 21](#).

## 19 16-bit advanced control timer (TIM1)

This section gives a description of the full set of timer features.

### 19.1 Introduction

TIM1 consists of a 16-bit up-down auto-reload counter driven by a programmable prescaler.

The timer may be used for a variety of purposes, including:

- Time base generation
- Measuring the pulse lengths of input signals (input capture)
- Generating output waveforms (output compare, PWM and one-pulse mode)
- Interrupt capability on various events (capture, compare, overflow, break, trigger)
- DMA request capability on capture, compare, overflow events
- Synchronization with other timers or external signals (external clock, reset, trigger and enable).

This timer is ideally suited for a wide range of control applications, including those requiring center-aligned PWM capability with complementary outputs and deadtime insertion.

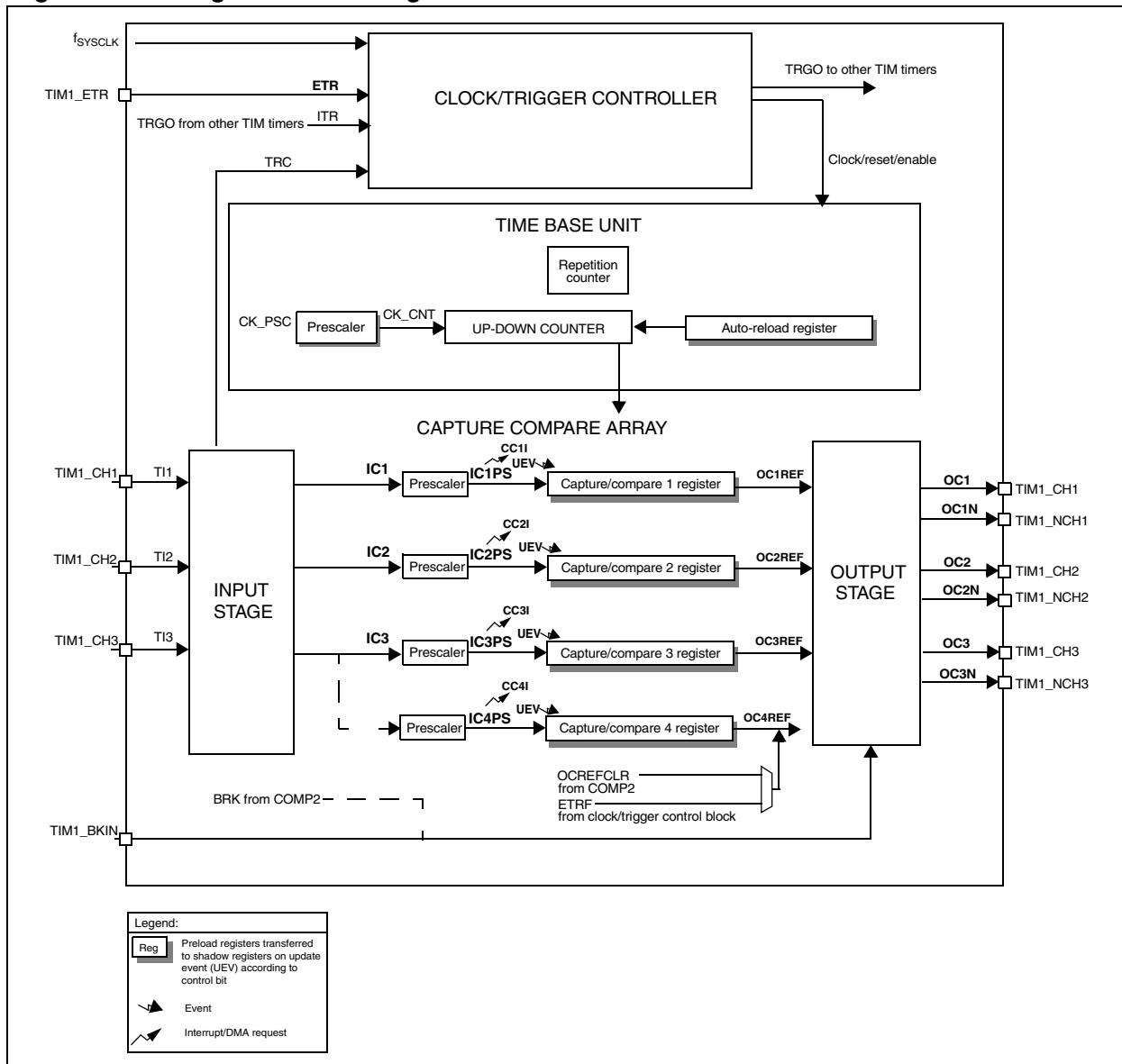
The timer clock can be sourced from internal clocks or from an external source selectable through a configuration register.

## 19.2 TIM1 main features

TIM1 features include:

- 16-bit up, down, up/down counter auto-reload counter
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 16-bit programmable prescaler allowing the counter clock frequency to be divided “on the fly” by any factor between 1 and 65536.
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- 3 independent channels (channel 1, channel 2 and channel 3) that can alternately be configured as:
  - Input capture
  - Output compare
  - PWM generation (edge and center-aligned mode)
  - 6-step PWM generation
  - One-pulse mode output
  - Complementary outputs on three channels with programmable deadtime insertion.
- One additional capture/compare channel (channel 4) not connected to external output
- Break input to put the timer output signals in reset state or in a known state.
- External trigger input pin (ETR)
- OCREFCLR signal or break input can be connected internally to comparator 2 (COMP2) output
- Input capture 2 and input capture 3 channels can be routed from 21 selected I/Os
- Interrupt generation on the following events:
  - Update: Counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input
  - Commutation event
- DMA request generation on the following events:
  - Update: counter overflow/underflow
  - Input capture
  - Output compare
  - Commutation event
- DMA burst transfer mode

Figure 67. TIM1 general block diagram

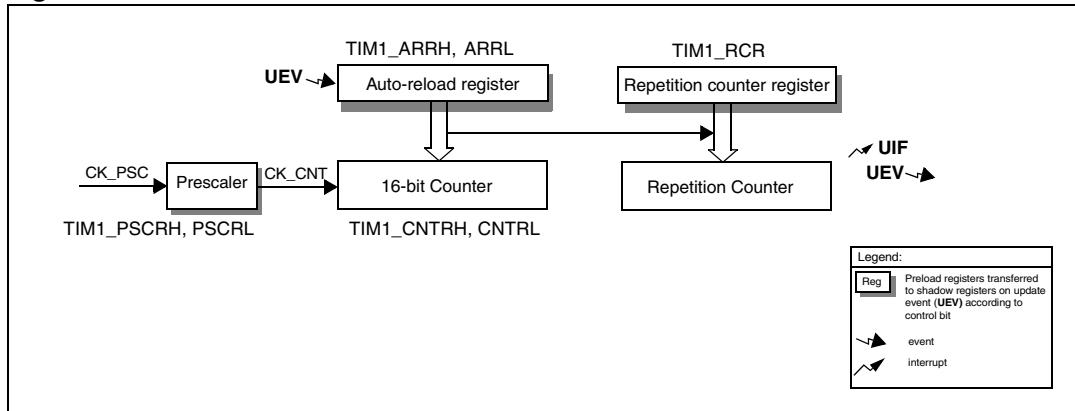


## 19.3 TIM1 time base unit

The timer has a time base unit that includes:

- 16-bit up/down counter
- 16-bit auto-reload register
- Repetition counter
- Prescaler

**Figure 68. Time base unit**



The 16-bit counter, the prescaler, the auto-reload register and the repetition counter register can be written or read by software.

The auto-reload register is composed of a preload register plus a shadow register.

Writing to the auto-reload register can be done in two modes:

- **Auto-reload preload enabled** (ARPE bit set in the TIM1\_CR1 register). In this mode, when data is written to the auto-reload register, it is kept in the preload register and transferred into the shadow register at the next update event (UEV).
- **Auto-reload preload disabled** (ARPE bit cleared in the TIM1\_CR1 register). In this mode, when data is written to the auto-reload register it is transferred into the shadow register immediately.

An update event is generated:

- On a counter overflow or underflow
- By software, setting the UG bit in the TIM1\_EGR register
- By a trigger event from the clock/trigger controller

With preload enabled (ARPE = 1), when an update event occurs: The auto-reload shadow register is updated with the preload value (TIM1\_ARR) and the buffer of the prescaler is reloaded with the preload value (content of the TIM1\_PSCR register).

The UEV can be disabled by setting the UDIS bit in the TIM1\_CR1

The counter is clocked by the prescaler output CK\_CNT, which is enabled only when the counter enable bit (CEN) in TIM1\_CR1 register is set.

*Note:*

*The actual counter enable signal CNT\_EN is set 1 clock cycle after CEN.*

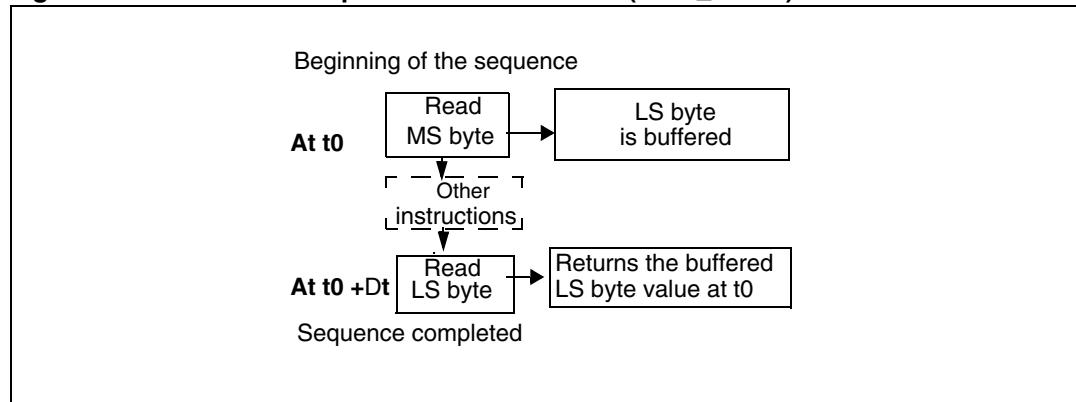
### 19.3.1 Reading and writing to the 16-bit counter

There is no buffering when writing to the counter. Both TIM1\_CNTRH and TIM1\_CNTRL can be written at any time, so it is suggested not to write a new value into the counter while it is running to avoid loading an incorrect intermediate content.

An 8-bit buffer is implemented for the read. Software must read the MS byte first, after which the LS byte value is buffered automatically (see [Figure 69](#)). This buffered value remains unchanged until the 16-bit read sequence is completed.

*Note:* *Do not use the LDW instruction to read the 16-bit counter. It reads the LS byte first and returns an incorrect result.*

**Figure 69. 16-bit read sequence for the counter (TIM1\_CNTR)**



### 19.3.2 Write sequence for 16-bit TIM1\_ARR register

16-bit values are loaded in the TIM1\_ARR register through preload registers. This must be performed by two write instructions, one for each byte. The MS byte must be written first.

The shadow register update is blocked as soon as the MS byte has been written, and stays blocked until the LS byte has been written. Do not use the LDW instruction as this writes the LS byte first which produces incorrect results.

### 19.3.3 Prescaler

The prescaler implementation is as follows:

The TIM1 prescaler is based on a 16-bit counter controlled through a 16-bit register (in TIM1\_PSCR register). It can be changed on the fly as this control register is buffered. It can divide the counter clock frequency by any factor between 1 and 65536.

The counter clock frequency is calculated as follows:

$$f_{CK\_CNT} = f_{CK\_PSC}/(PSCR[15:0]+1)$$

The prescaler value is loaded through a preload register. The shadow register, which contains the current value to be used, is loaded as soon as the LS byte has been written.

To update the 16-bit prescaler, load two bytes in separate write operations starting with the MSB. Do not use the LDW instruction for this purpose as it writes the LSB first.

The new prescaler value is taken into account in the following period (after the next counter update event).

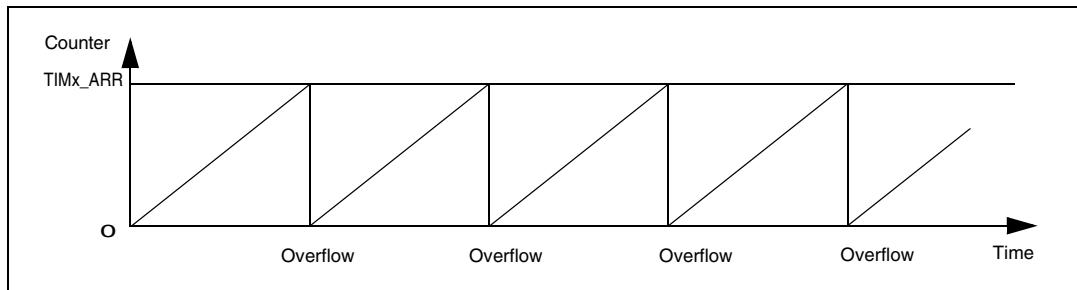
Read operations to the TIM1\_PSCR registers access the preload registers, so no special care needs to be taken to read them.

### 19.3.4 Up-counting mode

In up-counting mode, the counter counts from 0 to a user-defined compare value (content of the TIM1\_ARR register). It then restarts from 0 and generates a counter overflow event and a UEV if the UDIS bit is 0 in the TIM1\_CR1 register.

*Figure 70* shows an example of this counting mode.

**Figure 70. Counter in up-counting mode**



An update event can also be generated by setting the UG bit in the TIM1\_EGR register (either by software or by using the trigger controller).

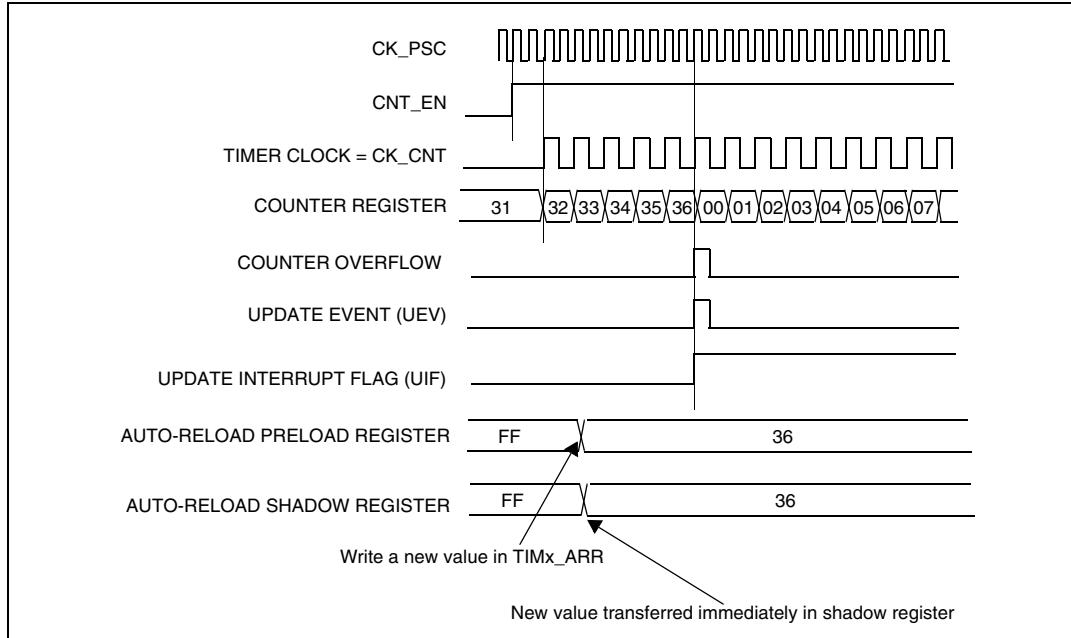
The UEV can be disabled by software by setting the UDIS bit in the TIM1\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. No UEV occurs until the UDIS bit has been written to 0. Note that the counter and the prescaler restart counting from 0 but, the prescaler division factor does not change. In addition, if the URS bit (update request selection) in the TIM1\_CR1 register is set, setting the UG bit generates an UEV without setting the UIF flag. Consequently, no interrupt or DMA request is sent. This avoids generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM1\_SR1 register) is set (depending on the URS bit):

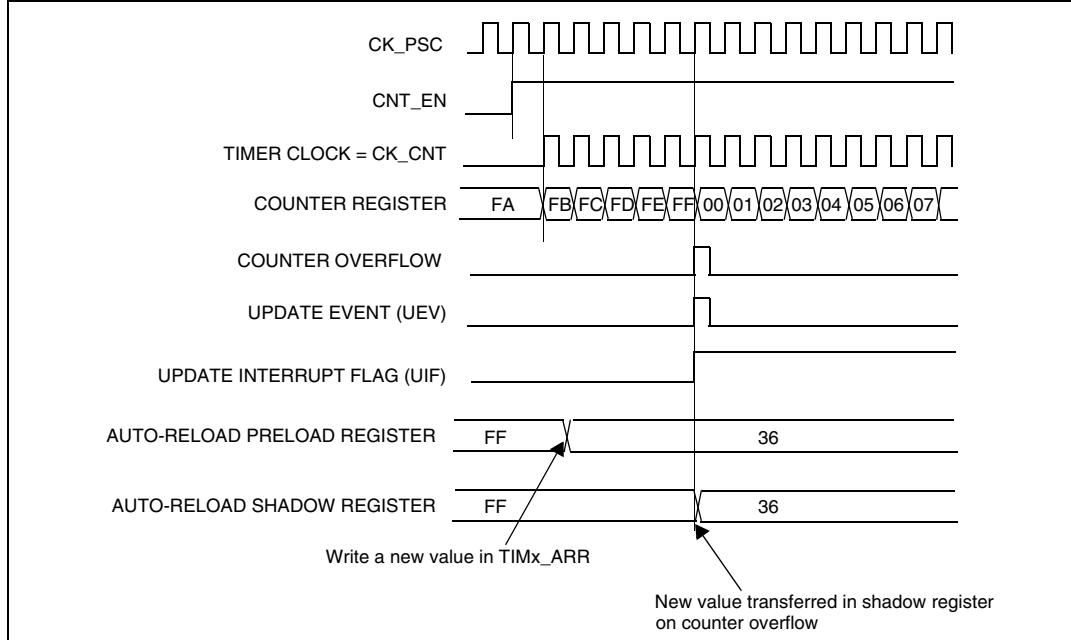
- The auto-reload shadow register is updated with the preload value (TIM1\_ARR).
- The buffer of the prescaler is reloaded with the preload value (content of the TIM1\_PSCR register).

*Figure 71* and *Figure 72* show two examples of the counter behavior for different clock frequencies when TIM1\_ARR = 0x36.

In *Figure 71*, the prescaler divider is set to 2, so the counter clock (CK\_CNT) frequency is at half the frequency of the prescaler clock source (CK\_PSC). The auto-reload preload is disabled (ARPE = 0). Consequently, the shadow register is immediately changed and counter overflow occurs when upcounting reaches 0x36. This generates a UEV.

**Figure 71. Counter update when ARPE = 0 (ARR not preloaded) with prescaler = 2**

In [Figure 72](#) the prescaler divider is set to 1, so CK\_CNT has the same frequency as CK\_PSC. The auto-reload preload is enabled (ARPE = 1), so the next counter overflow occurs at 0xFF. The new auto-reload value register value of 36h is taken into account after the overflow which generates a UEV.

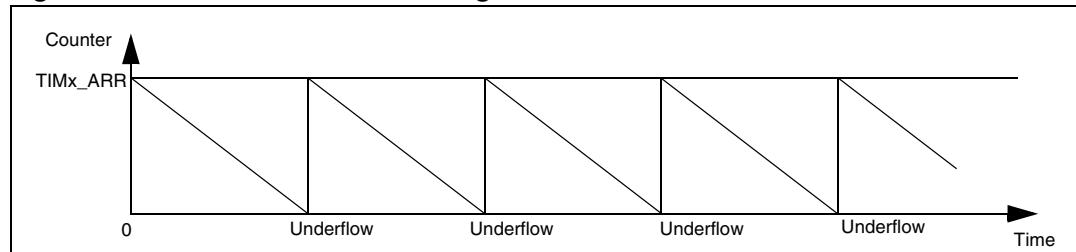
**Figure 72. Counter update event when ARPE = 1 (TIM1\_ARR preloaded)**

### 19.3.5 Down-counting mode

In down-counting mode, the counter counts from the auto-reload value (content of the TIM1\_ARR register) down to 0. It then restarts from the auto-reload value and generates a counter underflow and a UEV, if the UDIS bit is 0 in the TIM1\_CR1 register.

*Figure 73* shows an example of this counting mode.

**Figure 73. Counter in down-counting mode**



An update event can also be generated by setting the UG bit in the TIM1\_EGR register (by software or by using the clock/trigger mode controller).

The UEV update event can be disabled by software by setting the UDIS bit in TIM1\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. No update event occurs until the UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (without any change to the prescale rate).

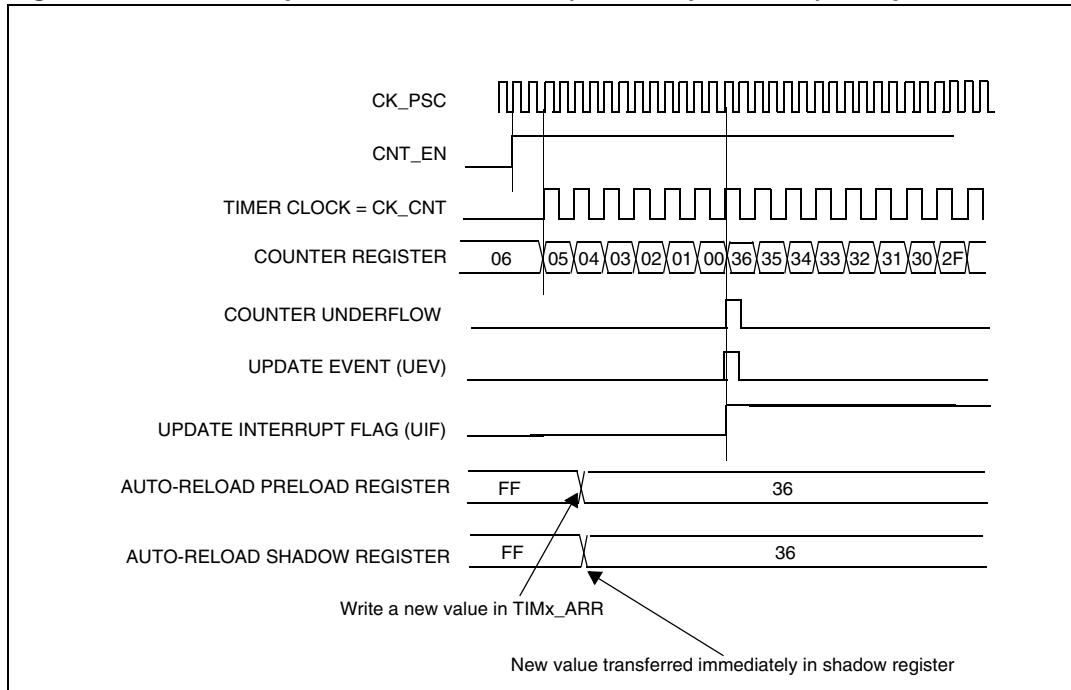
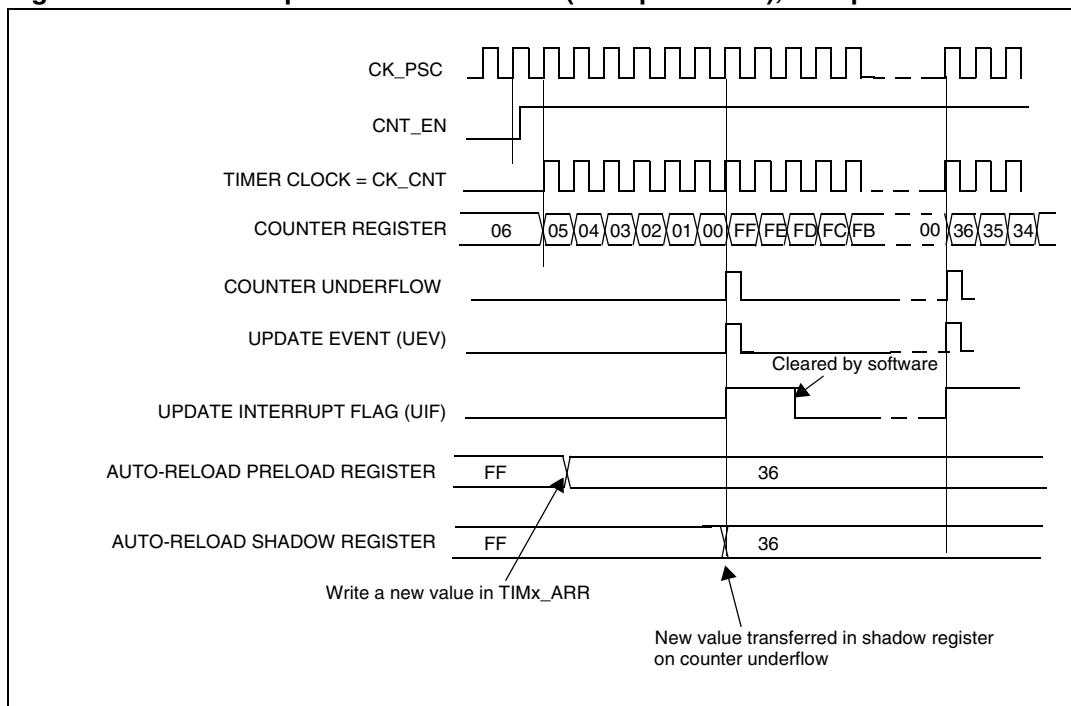
In addition, if the URS bit (update request selection) in the TIM1\_CR1 register is set, setting the UG bit generates a UEV without setting the UIF flag (thus no interrupt or DMA request is sent). This avoids generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIM1\_SR1 register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIM1\_PSCR register),
- The auto-reload shadow register is updated with the preload value (content of the TIM1\_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

*Figure 74* and *Figure 75* show some examples of the counter behavior for different clock frequencies when TIM1\_ARR = 0x36.

In downcounting mode, preload is not normally used. Consequently, the new value is taken into account in the next period (see *Figure 74*).

**Figure 74. Counter update when ARPE = 0 (ARR not preloaded) with prescaler = 2****Figure 75. Counter update when ARPE = 1 (ARR preloaded), with prescaler = 1**

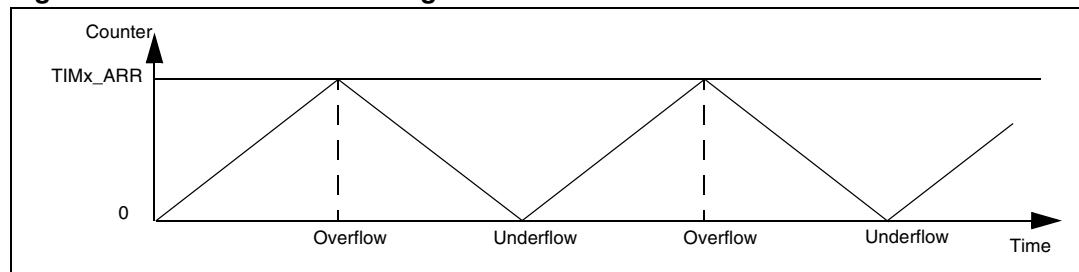
### 19.3.6 Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value of -1 (content of the TIM1\_ARR register). This generates a counter overflow event. The counter then counts down to 0 and generates a counter underflow event. After this, the counter restarts counting from 0.

In this mode, the direction bit (DIR) in the TIM1\_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The [Figure 76](#) shows an example of this counting mode.

**Figure 76. Counter in center-aligned mode**



If the timer has a repetition counter (as in TIM1), the UEV is generated after up and down counting and repeated for the number of times programmed in the repetition counter register (TIM1\_RCR). Otherwise, the UEV is generated at each counter overflow and at each counter underflow.

Setting the UG bit in the TIM1\_EGR register (by software or by using the clock/trigger mode controller) also generates an update event. In this case, the counter and the prescaler restart counting from 0.

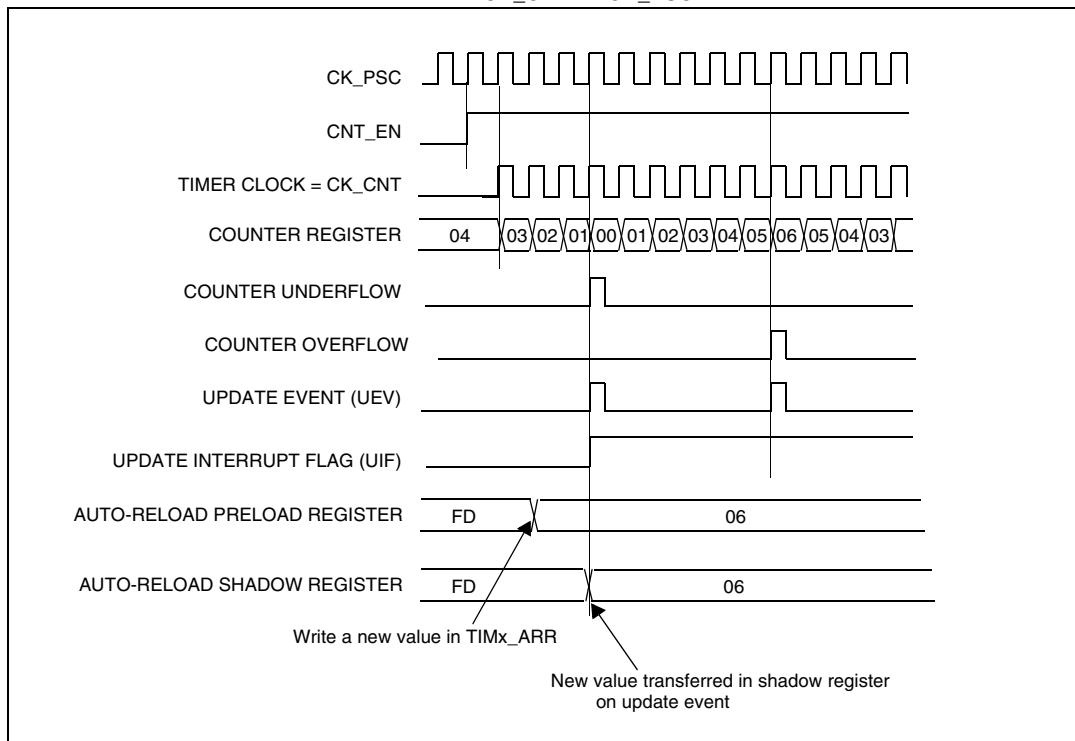
The UEV can be disabled by software by setting the UDIS bit in the TIM1\_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. In this way, no update event occurs until the UDIS bit is written to 0. However, the counter continues counting up and down, based on the current auto-reload value. In timers with a repetition counter, the new update rate is used because the repetition register is not double buffered. For this reason, care must be taken when changing the update rate.

In addition, if the URS bit in the TIM1\_CR1 register is set, setting the UG bit generates a UEV without setting the UIF flag. Consequently, no interrupt or DMA request is sent. This avoids generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all registers are updated and the update flag (the UIF bit in the TIM1\_SR1 register) is set (depending on the URS bit).

- The buffer of the prescaler is reloaded with the preload value (content of the TIM1\_PSCR register).
- The auto-reload shadow register is updated with the preload value (content of the TIM1\_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

Below are some examples of the counter behavior for different clock frequencies.

Figure 77. Counter timing diagram,  $f_{CK\_CNT} = f_{CK\_PSC}$ , TIM1\_ARR = 06h, ARPE = 1

Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter starts counting up or down depending on the value written in the DIR bit in the TIM1\_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value is written in the counter that is greater than the auto-reload value (TIM1\_CNT > TIM1\_ARR). For example, if the counter is counting up, it continues to do so.
  - The direction is updated if 0 or the TIM1\_ARR value are written in the counter but no UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIM1\_EGR register) just before starting the counter. Avoid writing to the counter while it is running.

### 19.3.7 Repetition down-counter

*Section 19.3: TIM1 time base unit* describes how the UEV is generated with respect to counter overflows/underflows. It is generated only when the repetition down-counter reaches zero. This can be useful while generating PWM signals.

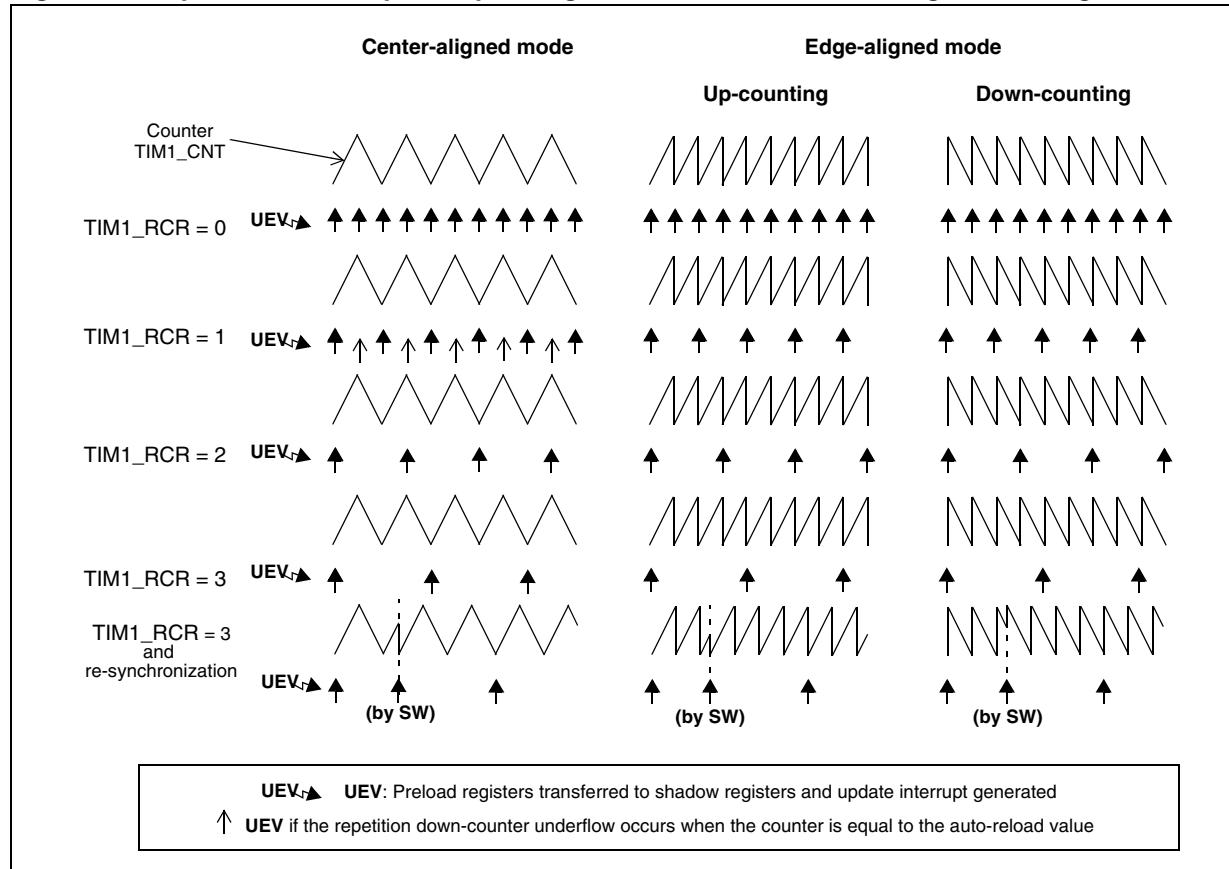
This means that data are transferred from the preload registers to the shadow registers (TIM1\_ARR auto-reload register, TIM1\_PSCR prescaler register, and TIM1\_CCRx capture/compare registers in compare mode) every ‘n’ counter overflow or underflow, where N is the value in the TIM1\_RCR repetition counter register.

The repetition down-counter is decremented:

- At each counter overflow in up-counting mode
- At each counter underflow in down-counting mode
- At each counter overflow and at each counter underflow in center-aligned mode.  
Although this limits the maximum number of repetitions to 128 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is  $2 \times t_{CK\_PSC}$  due to the symmetry of the pattern.

The repetition down-counter is an auto-reload type, the repetition rate of which is maintained as defined by the TIM1\_RCR register value (refer to *Figure 78*). When the UEV is generated by software (by setting the UG bit in the TIM1\_EGR register) or by hardware (through the clock/trigger controller), it occurs immediately irrespective of the value of the repetition down-counter. The repetition down-counter is reloaded with the content of the TIM1\_RCR register.

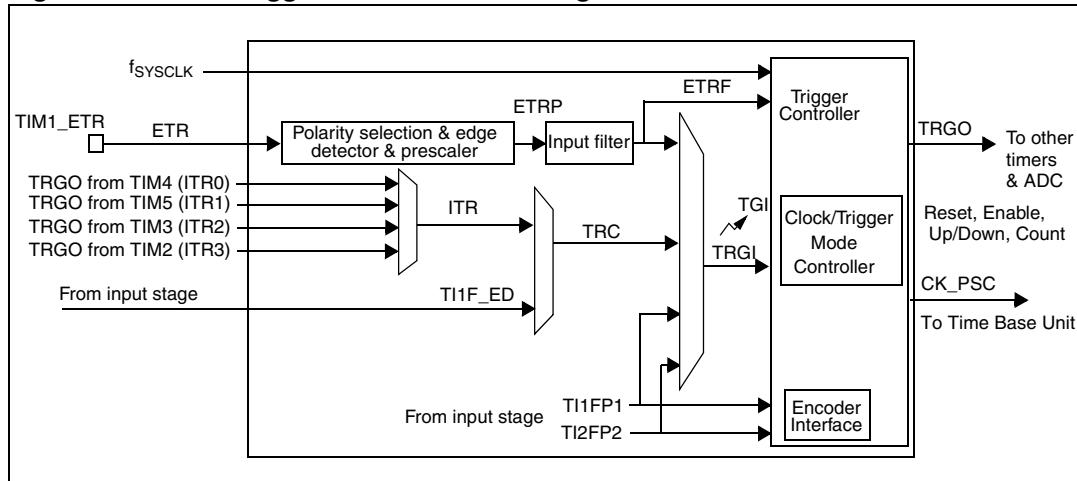
Figure 78. Update rate examples depending on mode and TIM1\_RCR register settings



## 19.4 TIM1 clock/trigger controller

The clock/trigger controller allows the timer clock sources, input triggers, and output triggers to be configured. The block diagram is shown in [Figure 79](#).

**Figure 79. Clock/trigger controller block diagram**



### 19.4.1 Prescaler clock (CK\_PSC)

The time base unit prescaler clock (CK\_PSC) can be provided by the following clock sources:

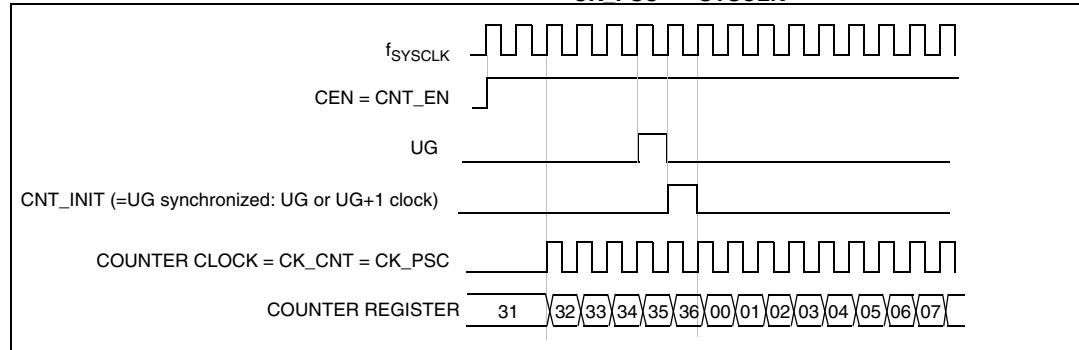
- Internal clock ( $f_{SYSCLK}$ )
- External clock mode 1: External timer input (TIx)
- External clock mode 2: External trigger input (ETR)
- Internal trigger inputs (ITR): using one timer as prescaler for another timer. Refer to [Using one timer as prescaler for another timer on page 306](#) for more details.

### 19.4.2 Internal clock source ( $f_{SYSCLK}$ )

If both the clock/trigger mode controller and the external trigger input are disabled (SMS = 000 in TIM1\_SMCR and ECE = 0 in the TIM1\_ETR register), the CEN, DIR, and UG bits behave as control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock.

The figure below shows the behavior of the control circuit and the up-counter in normal mode, without the prescaler.

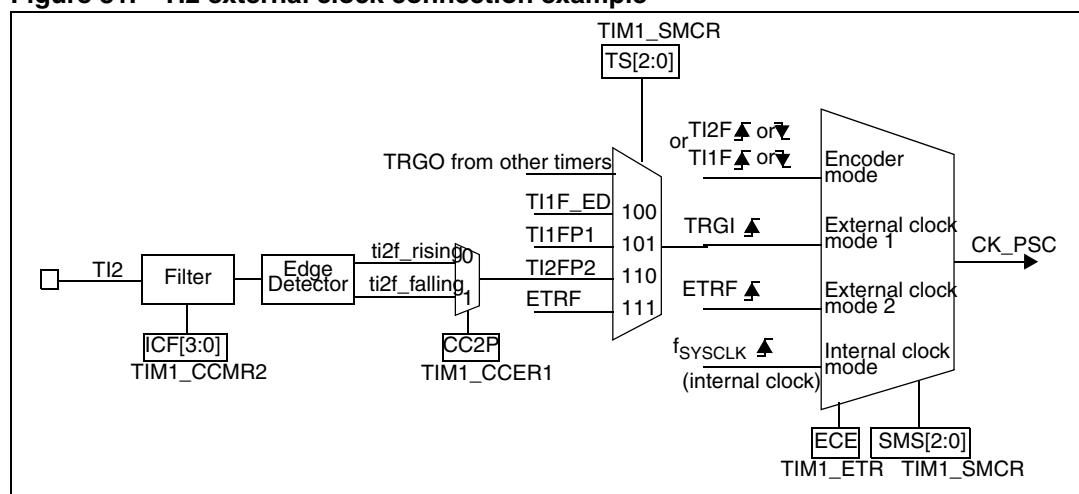
**Figure 80. Control circuit in normal mode,  $f_{CK\_PSC} = f_{SYSCLK}$**



### 19.4.3 External clock source mode 1

The counter can count at each rising or falling edge on a selected timer input. This mode is selected when SMS = 111 in the TIM1\_SMCR register (see [Figure 81](#)).

**Figure 81. TI2 external clock connection example**



### Procedure

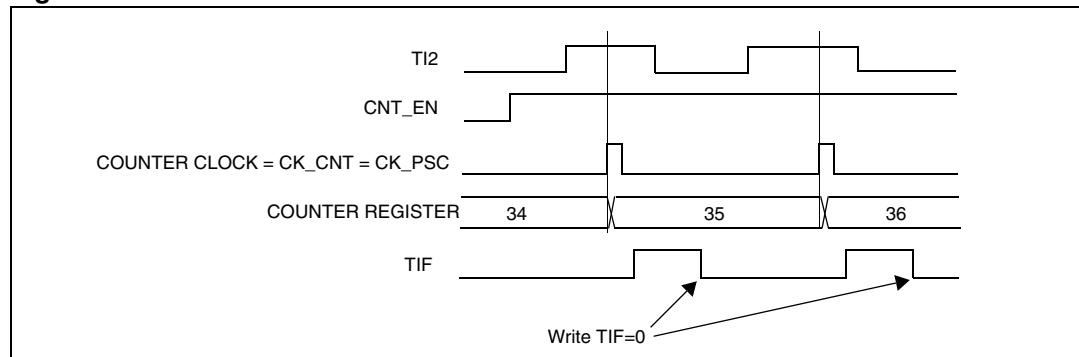
Use the following procedure to configure the up-counter and, for example, to count in response to a rising edge on the TI2 input:

1. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = 01 in the TIM1\_CCMR2 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIM1\_CCMR2 register (if no filter is needed, keep IC2F = 0000).  
*Note: The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits do not need to be configured either as they only select the input capture source.*
3. Select rising edge polarity by writing CC2P = 0 in the TIM1\_CCER1 register.
4. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIM1\_SMCR register.
5. Select TI2 as the input source by writing TS = 110 in the TIM1\_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIM1\_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the trigger flag is set (TIF bit in the TIM1\_SR1 register) and an interrupt request can be sent if enabled (depending on the TIE bit in the TIM1\_IER register).

The delay between the rising edge on TI2 and the actual reset of the counter is due to the resynchronization circuit on TI2 input.

**Figure 82. Control circuit in external clock mode 1**

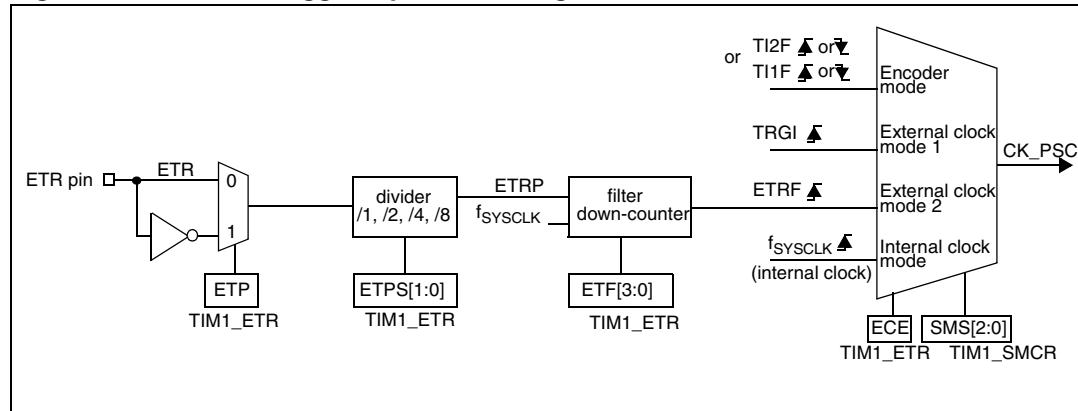


### 19.4.4 External clock source mode 2

The counter can count at each rising or falling edge on the ETR. This mode is selected by writing ECE = 1 in the TIM1\_ETR register.

The [Figure 83](#) gives an overview of the external trigger input block.

**Figure 83. External trigger input block diagram**



#### Procedure

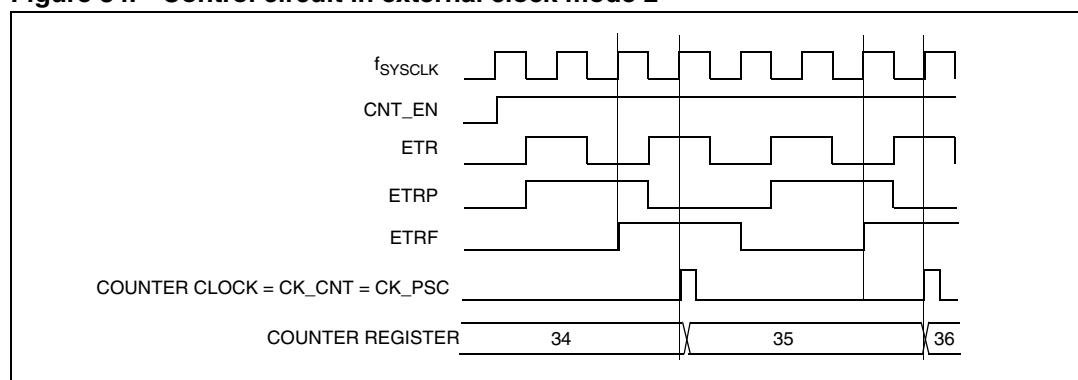
Use the following procedure to configure the up-counter and, for example, to count once every two rising edges on the ETR:

1. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIM1\_ETR register.
2. Set the prescaler by writing ETPS[1:0] = 01 in the TIM1\_ETR register.
3. Select rising edge detection on the ETR pin by writing ETP = 0 in the TIM1\_ETR register.
4. Enable external clock mode 2 by writing ECE = 1 in the TIM1\_ETR register.
5. Enable the counter by writing CEN = 1 in the TIM1\_CR1 register.

The counter counts once every two ETR rising edges.

The delay between the rising edge on the ETR and the actual reset of the counter is due to the resynchronization circuit on the external trigger signal (ETRP).

**Figure 84. Control circuit in external clock mode 2**



### 19.4.5 Trigger synchronization

There are four trigger inputs (refer to [Table 67: Glossary of internal timer signals on page 282](#)):

- ETR
- TI1
- TI2
- TRGO from other timers

The TIM1 timer can be synchronized with an external trigger in three modes: Trigger standard mode, trigger reset mode and trigger gated mode.

#### Trigger standard mode

The counter can start in response to an event on a selected input.

##### Procedure

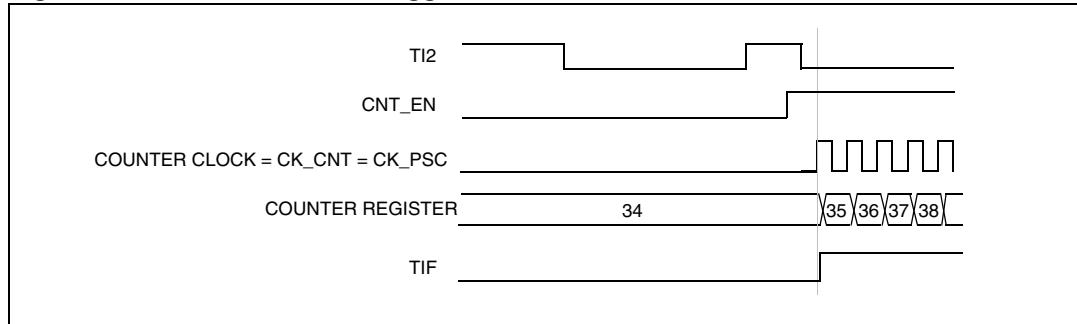
Use the following procedure to start the up-counter in response, for example, to a rising edge on the TI2 input:

1. Configure channel 2 to detect rising edges on TI2. As no filter is required in this example, configure an input filter duration of 0 (IC2F = 0000). The capture prescaler is not used for triggering and does not need to be configured. The CC2S bits select the input capture source and do not need to be configured either. Write CC2P = 0 in the TIM1\_CCER1 register to select rising edge polarity.
2. Configure the timer in trigger mode by writing SMS = 110 in the TIM1\_SMCR register. Select TI2 as the input source by writing TS = 110 in the TIM1\_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual reset of the counter is due to the resynchronization circuit on TI2 input.

**Figure 85. Control circuit in trigger mode**



### Trigger reset mode

The counter and its prescaler can be re-initialized in response to an event on a trigger input. Moreover, if the URS bit from the TIM1\_CR1 register is low, a UEV is generated. Then all the preloaded registers (TIM1\_ARR, TIM1\_CCR) are updated.

#### *Example*

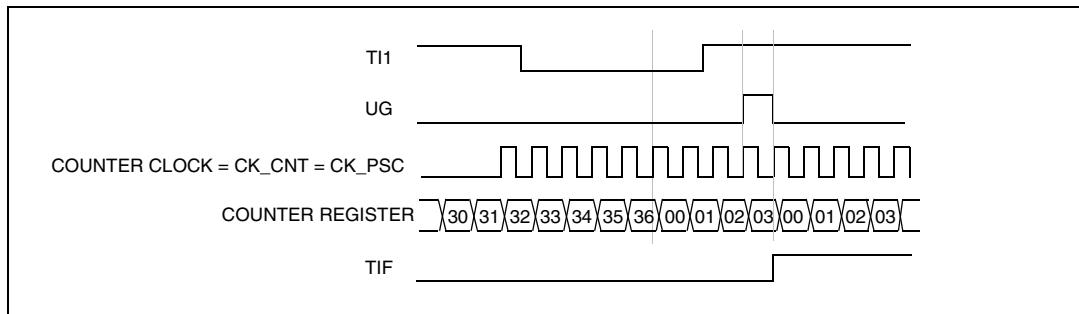
Use the following procedure to clear the up-counter in response to a rising edge on TI1 input:

1. Configure channel 1 to detect rising edges on TI1. As no filter is required in this example, configure an input filter duration of 0 (IC2F = 0000). The capture prescaler is not used for triggering and does not need to be configured. The CC1S bits select the input capture source and do not need to be configured either. Write CC1P = 0 in TIM1\_CCER1 register to validate the polarity (and detect rising edges).
2. Configure the timer in reset mode by writing SMS = 100 in TIM1\_SMCR register. Select TI1 as the input source by writing TS = 101 in the TIM1\_SMCR register.
3. Start the counter by writing CEN = 1 in the TIM1\_CR1 register.

The counter starts counting on the internal clock and behaves normally until the TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIM1\_SR1 register) and an interrupt request can be sent if enabled (depending on the TIE in the TIM1\_IER register).

*Figure 86* shows this behavior when the auto-reload register TIM1\_ARR = 36h. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

**Figure 86. Control circuit in trigger reset mode**



### Trigger gated mode

The counter can be enabled depending on the level of a selected input.

#### *Example*

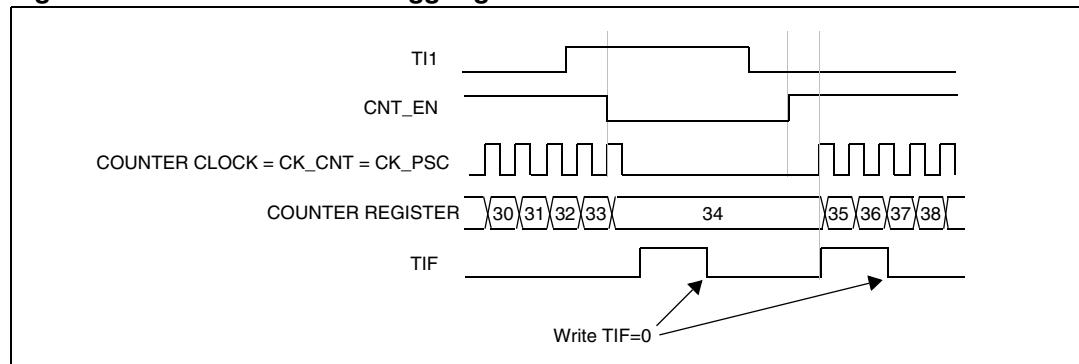
Use the following procedure to enable the up-counter when TI1 input is low:

1. Configure channel 1 to detect low levels on TI1. Configure the input filter duration (IC1F = 0000). The capture prescaler is not used for triggering and does not need to be configured. The CC1S bits select the input capture source and do not need to be configured either. Write CC1P = 1 in the TIM1\_CCER1 register to validate the polarity (and detect low level).
2. Configure the timer in trigger gated mode by writing SMS = 101 in the TIM1\_SMCR register. Select TI1 as the input source by writing TS = 101 in the TIM1\_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIM1\_CR1 register (in trigger gated mode, the counter does not start if CEN = 0 irrespective of the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low. It stops as soon as TI1 becomes high. The TIF flag is set when the counter starts or stops.

The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

**Figure 87. Control circuit in trigger gated mode**



### Combining trigger modes with external clock mode 2

External clock mode 2 can be used with another trigger mode. For example, the ETR can be used as the external clock input, and a different input can be selected as trigger input (in trigger standard mode, trigger reset mode, or trigger gated mode). Care must be taken not to select ETR as TRGI (through the TS bits in the TIM1\_SMCR register).

#### **Example**

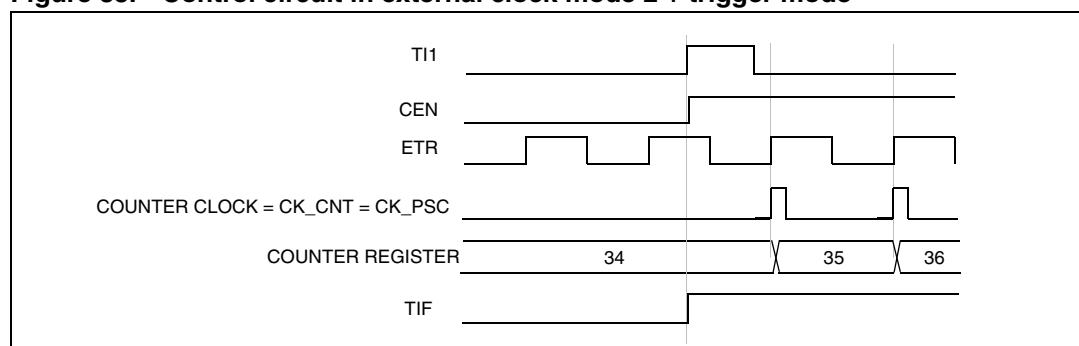
Use the following procedure to enable the up-counter at each rising edge on the ETR as soon as a rising edge occurs on TI1 (standard trigger mode with external ETR clock):

1. Configure the external trigger input circuit by writing to the TIM1\_ETR register. Write ETF = 0000 (no filter needed in this example). Write ETPS = 00 to disable the prescaler, write ETP = 0 to detect rising edges on the ETR, and write ECE = 1 to enable external clock mode 2.
2. Configure channel 1 to detect rising edges on TI1. Configure the input filter duration (IC1F = 0000). The capture prescaler is not used for triggering and does not need to be configured. The CC1S bits select the input capture source and do not need to be configured either. Write CC1P = 0 in the TIM1\_CCER1 register to select rising edge polarity.
3. Configure the timer in trigger mode by writing SMS = 110 in the TIM1\_SMCR register. Select TI1 as the input source by writing TS = 101 in the TIM1\_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. Consequently, the counter counts on the ETR rising edges.

The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input. The delay between the rising edge on the ETR and the actual reset of the counter is due to the resynchronization circuit on the ETRP signal.

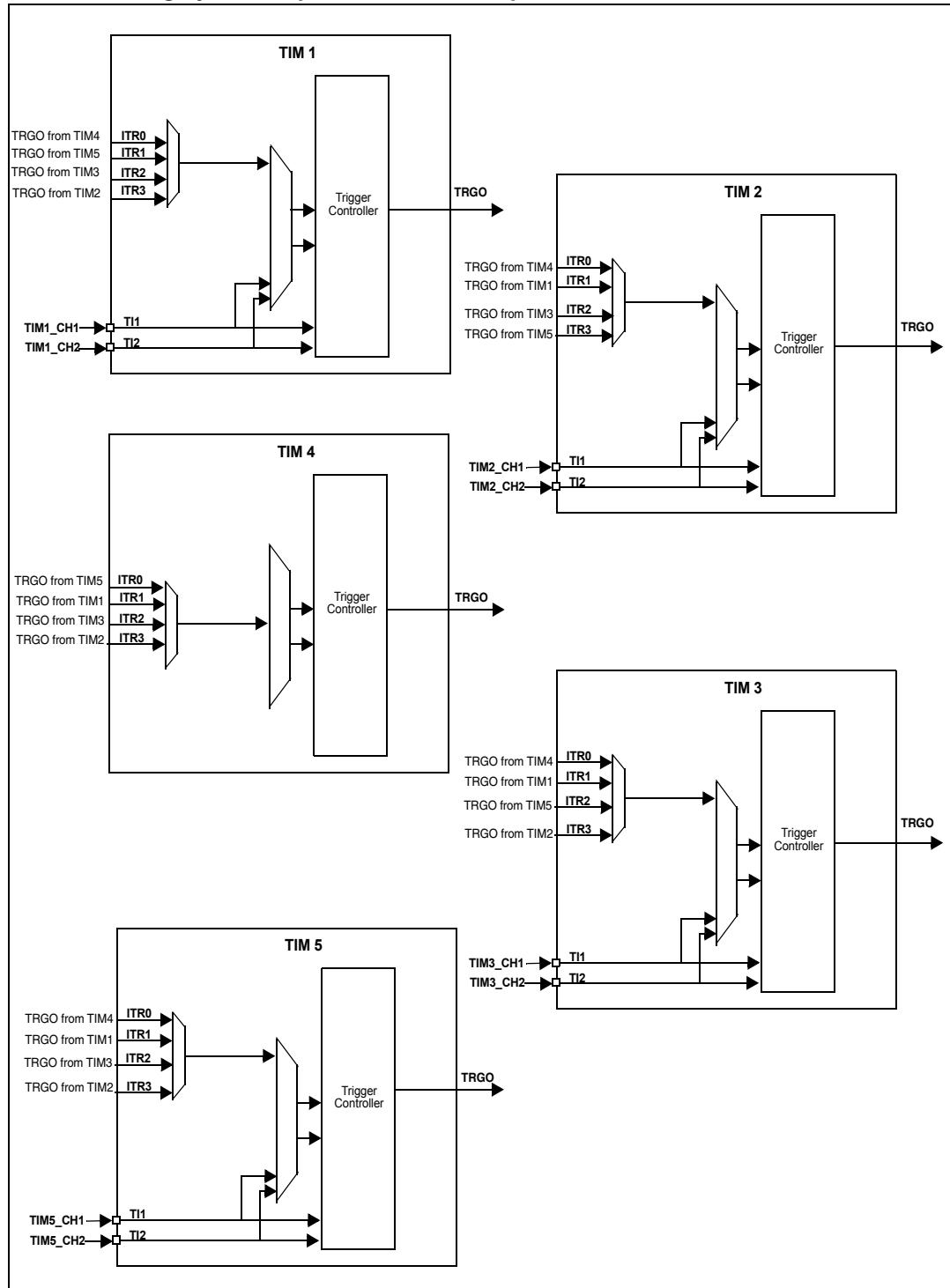
**Figure 88. Control circuit in external clock mode 2 + trigger mode**



### 19.4.6 Synchronization between timers

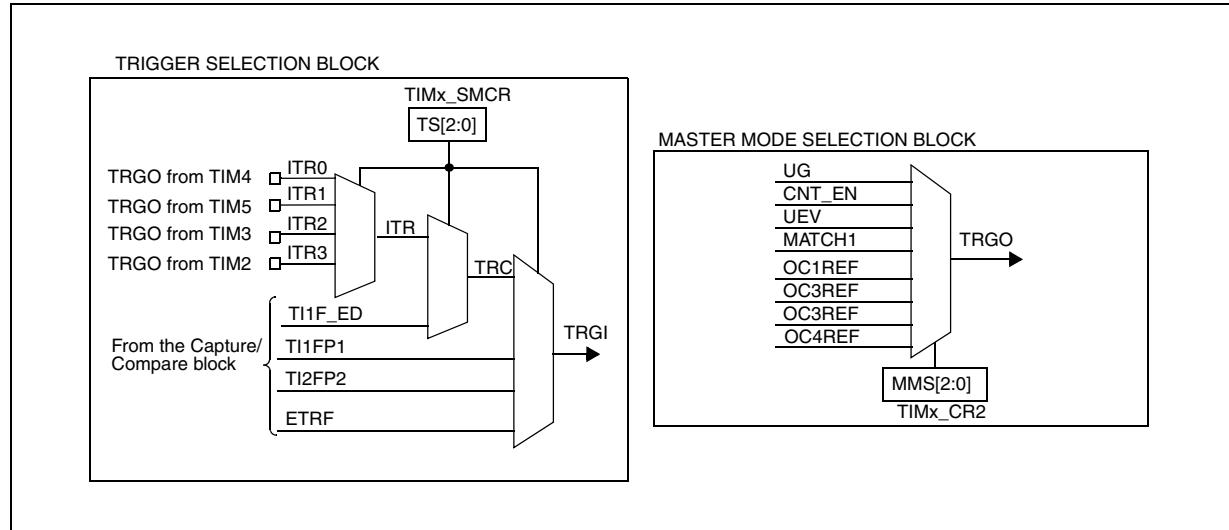
On some products, the timers are linked together internally for timer synchronization or chaining. When one timer is configured in master mode, it can output a trigger (TRGO) to reset, start, stop, or clock the counter of any other timer configured in slave mode.

**Figure 89. Timer chaining system implementation example**



*Figure 90* presents an overview of the trigger selection and the master mode selection blocks.

**Figure 90. Trigger/master mode selection blocks**



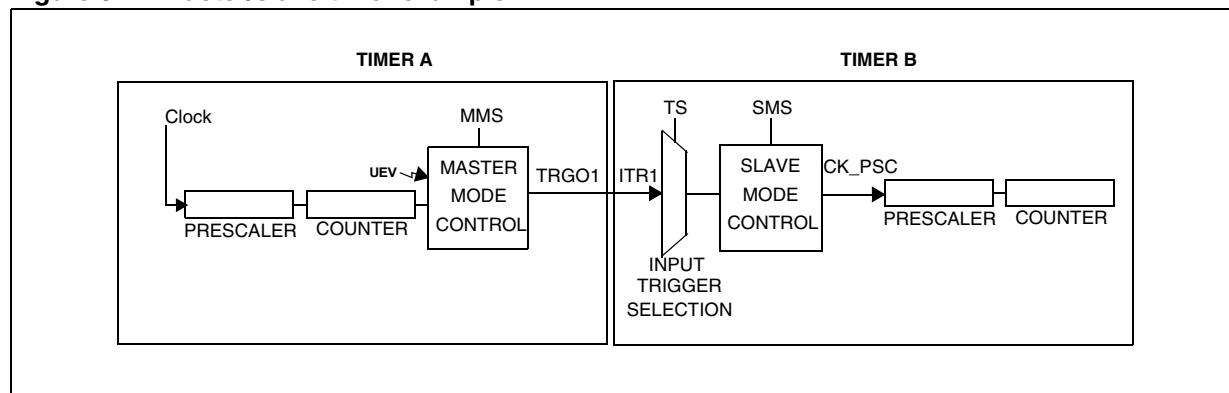
### Using one timer as prescaler for another timer

Refer to *Figure 91* to see how timer A can be configured to act as a prescaler for timer B.

1. Configure timer A in master mode so that it outputs a periodic trigger signal on each UEV. To configure that a rising edge is output on TRGO1 each time an update event is generated, write MMS = 010 in the **TIMx\_CR2** register.
2. Connect the TRGO1 output of timer A to timer B, timer B must be configured in slave mode using ITR1 as the internal trigger. Select this through the TS bits in the **TIMx\_SMCR** register (writing TS = 001).
3. Put the clock/trigger controller in external clock mode 1, by writing SMS = 111 in the **TIMx\_SMCR** register. This causes timer B to be clocked by the rising edge of the periodic timer A trigger signal (which corresponds to the timer A counter overflow).
4. Enable both timers by setting their respective CEN bits (**TIMx\_CR1** register).

**Note:** If OCi is selected on timer A as trigger output (MMS = 1xx), its rising edge is used to clock the counter of timer B.

**Figure 91. Master/slave timer example**



## Using one timer to enable another timer

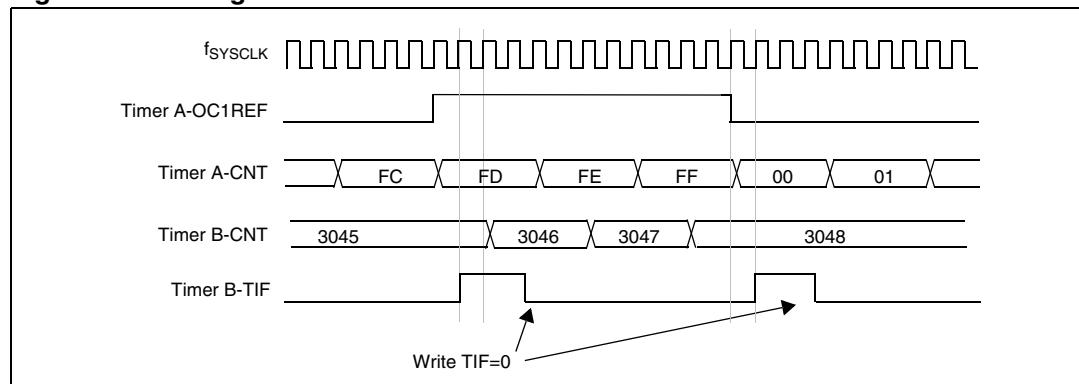
### Example 1

The enable of timer B is controlled with the output compare 1 of timer A (refer to [Figure 92](#) for connections). Timer B counts on the divided internal clock only when OC1REF of timer A is high. Both counter clock frequencies are divided by four by the prescaler compared to  $f_{SYSCLK}$  ( $f_{CK\_CNT} = f_{SYSCLK}/4$ ).

1. Configure timer A master mode to send its output compare 1 reference (OC1REF) signal as trigger output (MMS = 100 in the TIMx\_CR2 register).
2. Configure the timer A OC1REF waveform (TIMx\_CCMR1 register)
3. Configure timer B to get the input trigger from timer A (TS = 001 in the TIMx\_SMCR register).
4. Configure timer B in trigger gated mode (SMS = 101 in TIMx\_SMCR register)
5. Enable timer B by writing 1 in the CEN bit (TIMx\_CR1 register)
6. Start timer A by writing 1 in the CEN bit (TIMx\_CR1 register)

**Note:** *The counter 2 clock is not synchronized with counter 1. This mode only affects the timer B counter enable signal.*

**Figure 92. Gating timer B with OC1REF of timer A**



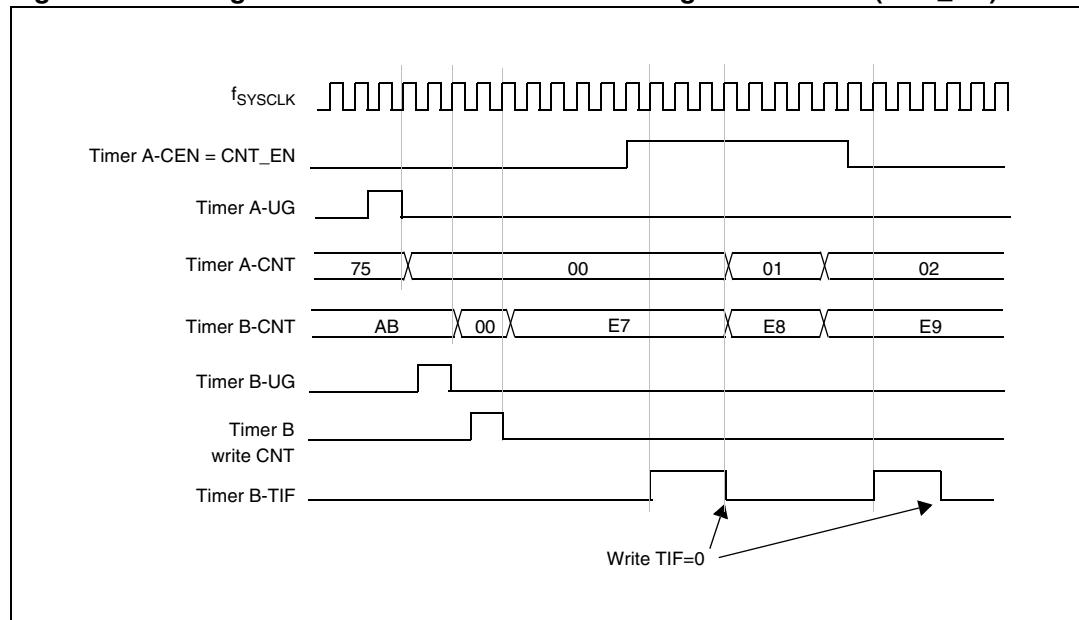
In [Figure 92](#), the timer B counter and prescaler are not initialized before being started. Therefore, they start counting from their current value. It is possible to start from a given value by resetting both timers before starting timer A. In this case, any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx\_EGR registers.

**Example 2**

Timer A and timer B are synchronized. Timer A is the master and starts from 0. Timer B is the slave and starts from E7h. The prescaler ratio is the same for both timers. Timer B stops when timer A is disabled by writing 0 to the CEN bit in the TIMx\_CR1 register:

1. Configure timer A master mode to send its output compare 1 reference (OC1REF) signal as trigger output (MMS = 100 in the TIMx\_CR2 register).
2. Configure the timer A OC1REF waveform (TIMx\_CCMR1 register)
3. Configure timer B to get the input trigger from timer A (TS = 001 in the TIMx\_SMCR register).
4. Configure timer B in trigger gated mode (SMS = 101 in TIMx\_SMCR register)
5. Reset timer A by writing 1 in UG bit (TIMx\_EGR register)
6. Reset timer B by writing 1 in UG bit (TIMx\_EGR register)
7. Initialize timer B to 0xE7 by writing 'E7h' in the timer B counter (TIMx\_CNTRL)
8. Enable timer B by writing 1 in the CEN bit (TIMx\_CR1 register)
9. Start timer A by writing 1 in the CEN bit (TIMx\_CR1 register)
10. Stop timer A by writing 0 in the CEN bit (TIMx\_CR1 register)

**Figure 93. Gating timer B with the counter enable signal of timer A (CNT\_EN)**



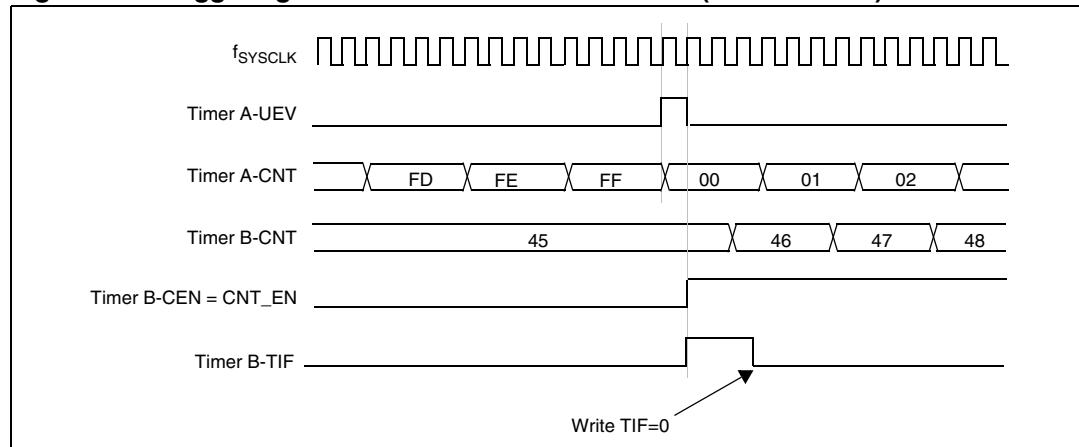
### Using one timer to start another timer

#### *Example 1*

The enable of timer B is set with the UEV of timer A (refer to [Figure 91](#) for connections). Timer B starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the UEV is generated by timer A. When timer B receives the trigger signal, its CEN bit is automatically set and the counter counts until 0 is written to the CEN bit in the TIM1\_CR1 register. Both counter clock frequencies are divided by four by the prescaler compared to f<sub>SYSCLK</sub> ( $f_{CK\_CNT} = f_{SYSCLK}/4$ ).

1. Configure timer A master mode to send its UEV as trigger output (MMS = 010 in the TIM1\_CR2 register).
2. Configure the timer A period (TIM1\_ARR registers)
3. Configure timer B to get the input trigger from timer A (TS = 001 in the TIM1\_SMCR register).
4. Configure timer B in trigger mode (SMS = 110 in TIM1\_SMCR register)
5. Start timer A by writing 1 in the CEN bit (TIM1\_CR1 register)

**Figure 94. Triggering timer B with the UEV of timer A (TIMERA-UEV)**

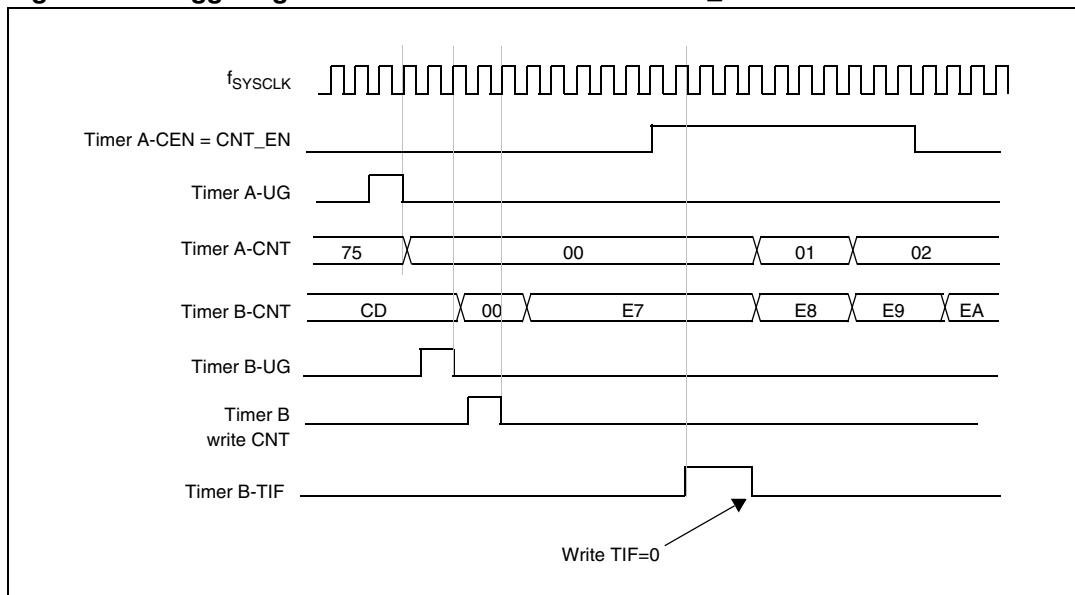


**Example 2**

As in the previous example, both counters can be initialized before starting to count.

*Figure 95* shows the behavior, with the same configuration as in *Figure 93*, but, in trigger standard mode instead of trigger gated mode (SMS = 110 in the TIM1\_SMCR register).

**Figure 95. Triggering timer B with counter enable CNT\_EN of timer A**

**Starting 2 timers synchronously in response to an external trigger****Example**

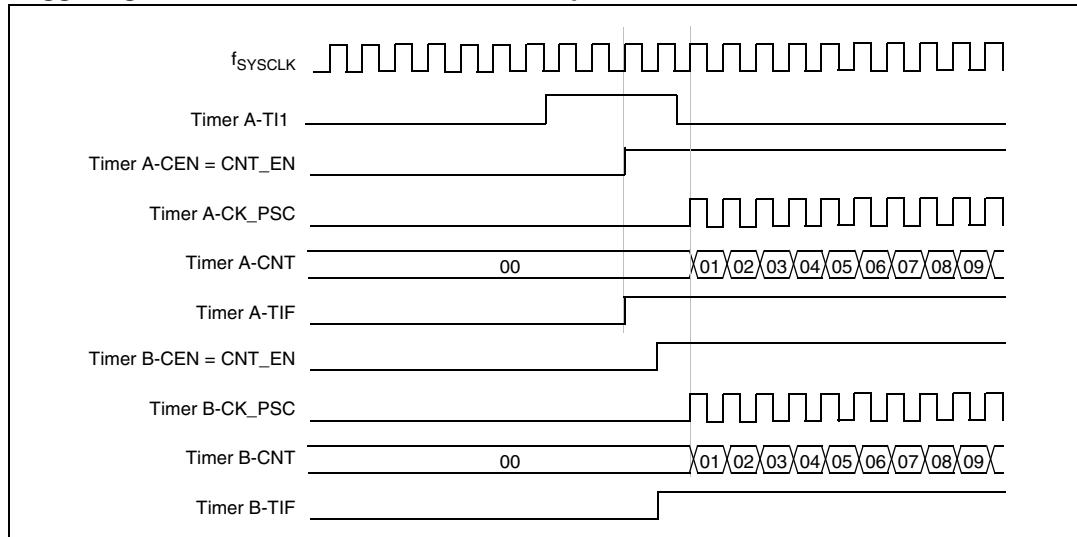
The enable of timer A is set when its TI1 input rises and the enable of timer B is set with the enable of timer A (refer to *Figure 91* for connections). To ensure the counters alignment, timer A must be configured in master/slave mode (slave with respect to TI1, master with respect to timer B).

1. Configure timer A master mode to send its enable as trigger output (MMS = 001 in the TIMx\_CR2 register).
2. Configure timer A slave mode to get the input trigger from TI1 (TS = 100 in the TIMx\_SMCR register).
3. Configure timer A in trigger mode (SMS = 110 in the TIMx\_SMCR register)
4. Configure timer A in master/slave mode by writing MSM = 1 (TIMx\_SMCR register)
5. Configure timer B to get the input trigger from timer A (TS = 001 in the TIMx\_SMCR register).
6. Configure timer B in trigger mode (SMS = 110 in the TIMx\_SMCR register)

When a rising edge occurs on TI1 (timer A), both counters start counting synchronously on the internal clock and both TIF flags are set.

**Note:**

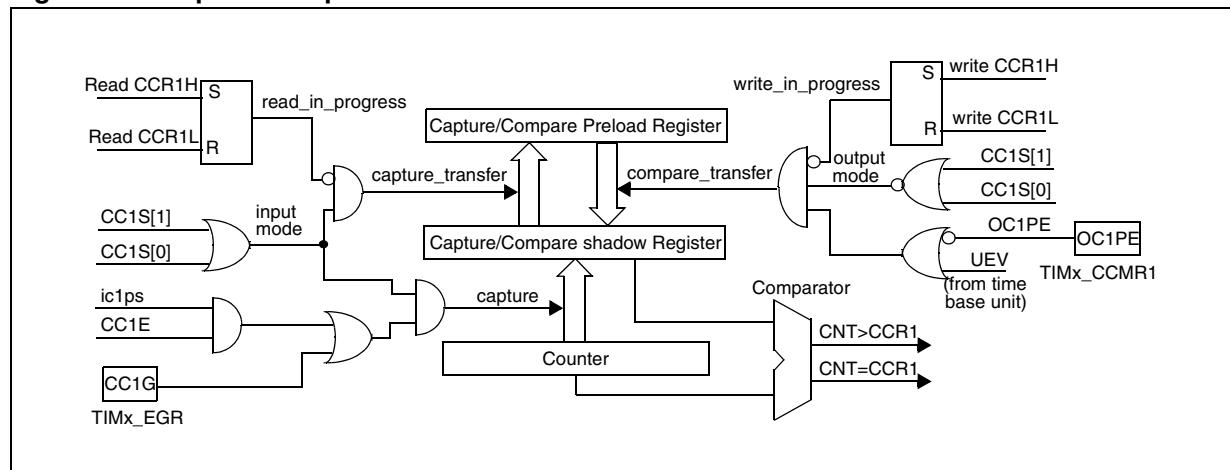
*In this example both timers are initialized before starting (by setting their respective UG bits). Both counters start from 0, but an offset can easily be inserted between them by writing to any of the counter registers (TIMx\_CNT). It can be seen that the master/slave mode inserts a delay between CNT\_EN and CK\_PSC on timer A.*

**Figure 96.** Triggering Timer A and B with Timer A TI1 input

## 19.5 TIM1 capture/compare channels

The timer I/O pins (TIM1\_CC $i$ ) can be configured either for input capture or output compare functions. The choice is made by configuring the CC*i*S channel selection bits in the capture/compare channel mode registers (TIM1\_CCMR*i*), where  $i$  is the channel number.

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

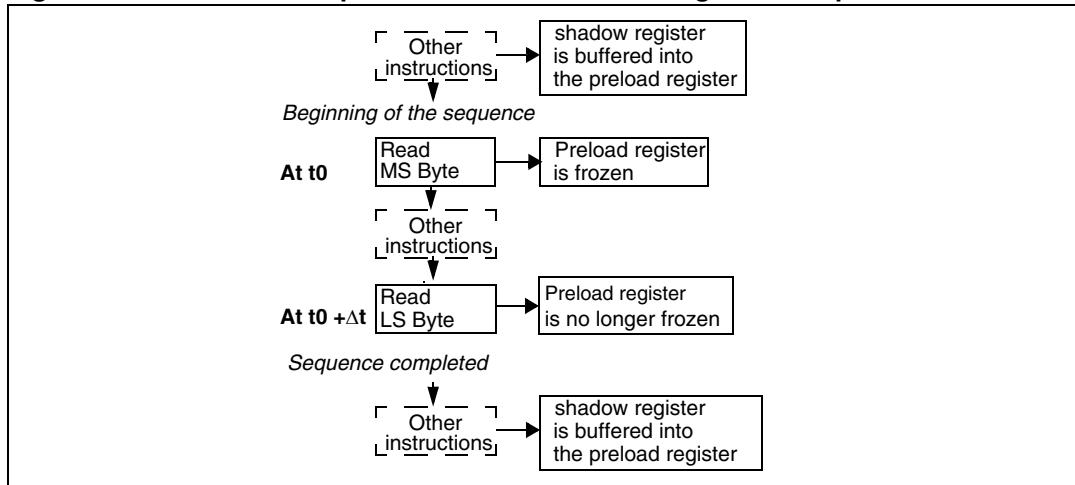
**Figure 97.** Capture/compare channel 1 main circuit

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register. In capture mode, captures are made in the shadow register, which is copied into the preload register. In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

When the channel is configured in output mode ( $CCiS = 00$  in the  $TIM1\_CCMRi$  registers), the  $TIM1\_CCRi$  registers can be accessed without any restriction.

When the channel is configured in input mode, the sequence for reading the  $TIM1\_CCRi$  registers is the same as for the counter (see [Figure 98](#)). When a capture occurs, the content of the counter is captured into the  $TIM1\_CCRi$  shadow registers. Then this value is loaded into the preload register, except during a read sequence, when the preload register is frozen.

**Figure 98. 16-bit read sequence for the  $TIM1\_CCRi$  register in capture mode**



[Figure 98](#) shows the sequence for reading the  $CCRi$  registers in the 16-bit timers. This buffered value remains unchanged until the 16-bit read sequence is completed.

After a complete read sequence, if only the  $TIM1\_CCRiL$  registers are read, they return the LS byte of the count value at the time of the read.

If the MS byte is read after the LS byte, it no longer corresponds to the same captured value as the LS byte.

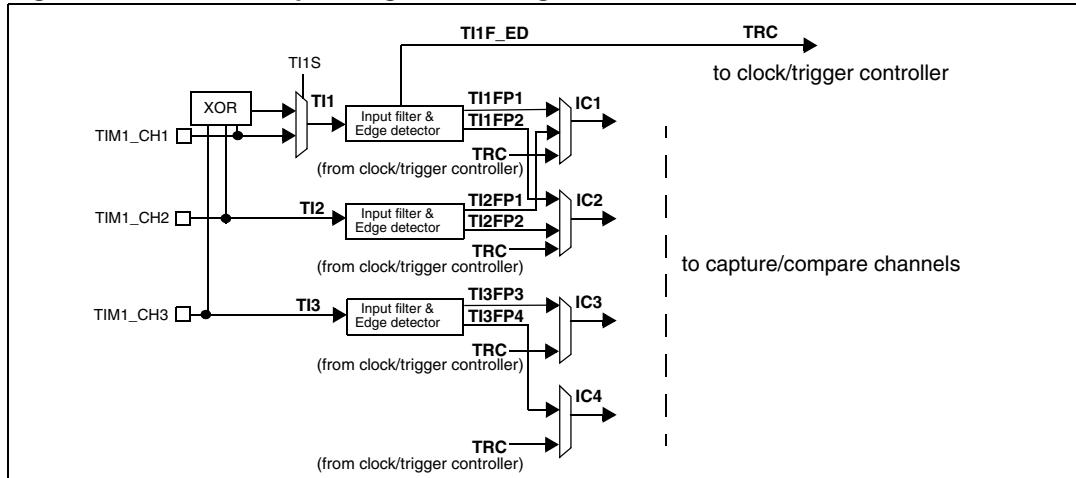
### 19.5.1 Write sequence for 16-bit $TIM1\_CCRi$ registers

16-bit values are loaded in the  $TIM1\_CCRi$  registers through preload registers. This must be performed by two write instructions, one for each byte. The MS byte must be written first.

The shadow register update is blocked as soon as the MS byte has been written, and stays blocked until the LS byte is written. Do not use the LDW instruction, as this writes the LS byte first, and produces incorrect results in this case.

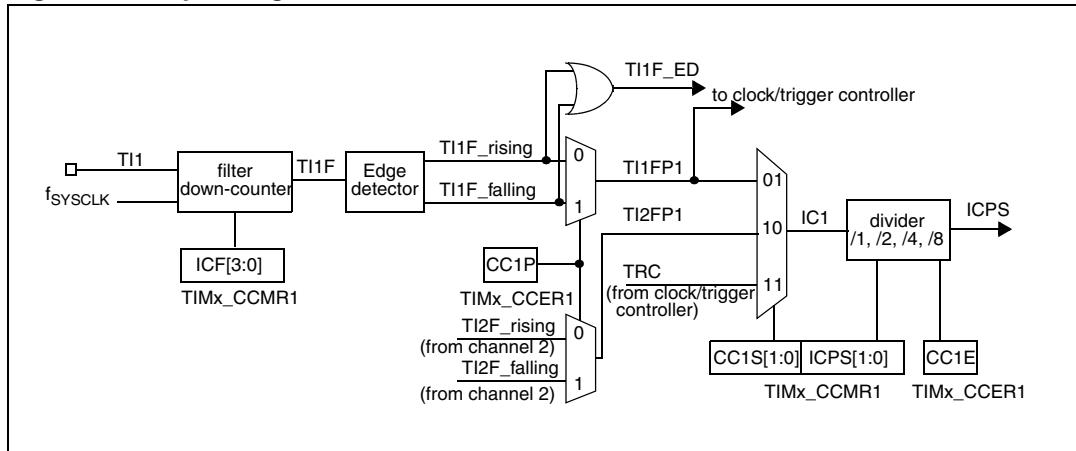
## 19.5.2 Input stage

**Figure 99. Channel input stage block diagram**



*Figure 100* shows how the input stage samples the corresponding TI*i* input to generate a filtered signal TI*iF*. Then, an edge detector with polarity selection, generates a signal (TI*iFPn*) which can be used as trigger input by the clock/trigger controller or as the capture command. The signal is prescaled before entering the capture register (IC*iPS*).

**Figure 100. Input stage of TIM 1 channel 1**



### 19.5.3 Input capture mode

In input capture mode, the capture/compare registers (TIM1\_CCR $i$ ) are used to latch the value of the counter after a transition detected on the corresponding IC $i$  signal. When a capture occurs, the corresponding CC $i$ F flag (TIM1\_SR1 register) is set.

An interrupt or DMA request can be sent if it is enabled, by setting the CC $i$ E bits in the TIM1\_IER register or the CC $x$ DE bit in the TIM1\_DER register. If a capture occurs while the CC $i$ F flag is already high, the over-capture flag CC $i$ OF (TIM1\_SR2 register) is set. CC $i$ F can be cleared by software by writing it to 0 or by reading the captured data stored in the TIM $x$ \_CCR $L$  registers. CC $i$ OF is cleared by writing it to 0.

#### Procedure

The following procedure shows how to capture the counter value in TIM1\_CCR1, for example, when TI1 input rises.

1. Select the active input: For example, to link the TIM1\_CCR1 register to the TI1 input, write the CC1S bits to 01 in the TIM1\_CCMR1 register. This configures the channel in input mode and the TIM1\_CCR1 register becomes read-only.
2. Program the required input filter duration for the signal to be connected to the timer. This is done for each TI $i$  input using the IC $i$ F bits in the TIM1\_CCMR $i$  registers. For example, if the input signal is unstable for up to five t<sub>SYSCLK</sub> cycles when it toggles, the filter duration must be performed for longer than five clock cycles. The filter bits allow a duration of eight cycles to be selected by writing them to 0011 in the TIM $x$ \_CCMR1 register. With this filter setting, a transition on TI1 is valid only when eight consecutive samples with the new level have been detected (sampled at f<sub>SYSCLK</sub> frequency).
3. Select the edge of the active transition on the TI1 channel by writing the CC1P bit to 0 in the TIM1\_CCER1 register (rising edge in this case).
4. Program the input prescaler. In this example, the capture needs to be performed at each valid transition, so the prescaler is disabled (write the IC1PS bits to 00 in the TIM1\_CCMR1 register).
5. Enable capture from the counter into the capture register by setting the CC1E bit in the TIM1\_CCER1 register.
6. If needed, enable the related interrupt request by setting the CC1IE bit in the TIM1\_IER register and/or DMA request by setting the CC1IE bit in the TIM1\_DER register.

When an input capture occurs:

- The TIM1\_CCR1 register gets the value of the counter on the active transition
- The input capture flag (CC1IF) is set. The overcapture flag (CC1OF) is also set if at least two consecutive captures occur while the flag remains uncleared.
- An interrupt is generated depending on the CC1IE bit
- A DMA request is generated depending on the CC1DE bit

To handle the overcapture event (CC1OF flag), it is recommended to read the data before the overcapture flag. This avoids missing an overcapture which could occur after reading the flag and before reading the data.

*Note:*

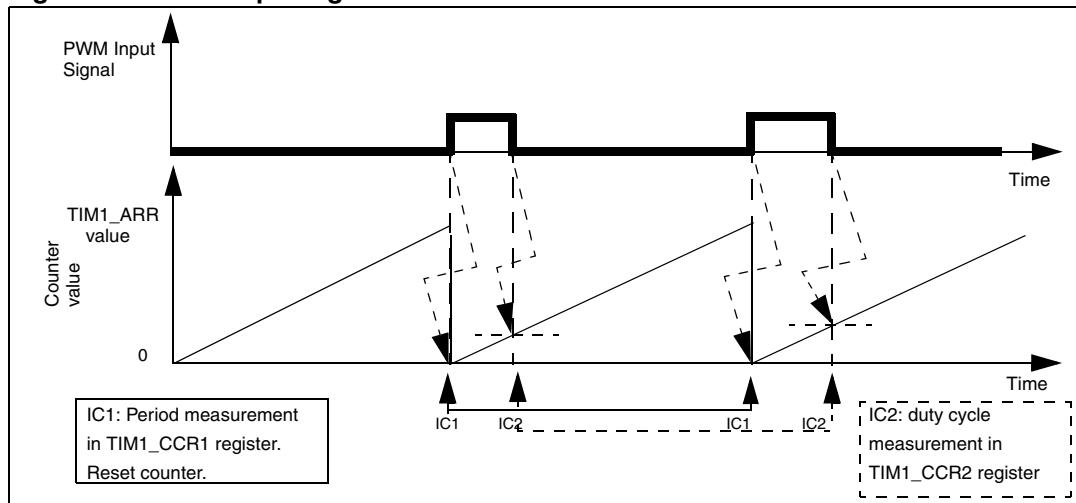
*IC interrupts and/or DMA requests can be generated by software by setting the corresponding CC $i$ G bits in the TIM1\_EGR register.*

### PWM input signal measurement

This mode is a particular case of input capture mode (see [Figure 101](#)). The procedure is the same except:

- Two IC<sub>i</sub> signals are mapped on the same TI<sub>i</sub> input
- These two IC<sub>i</sub> signals are active on edges with opposite polarity
- One of the two TI<sub>i</sub>/FP signals is selected as trigger input and the clock/trigger controller is configured in trigger reset mode.

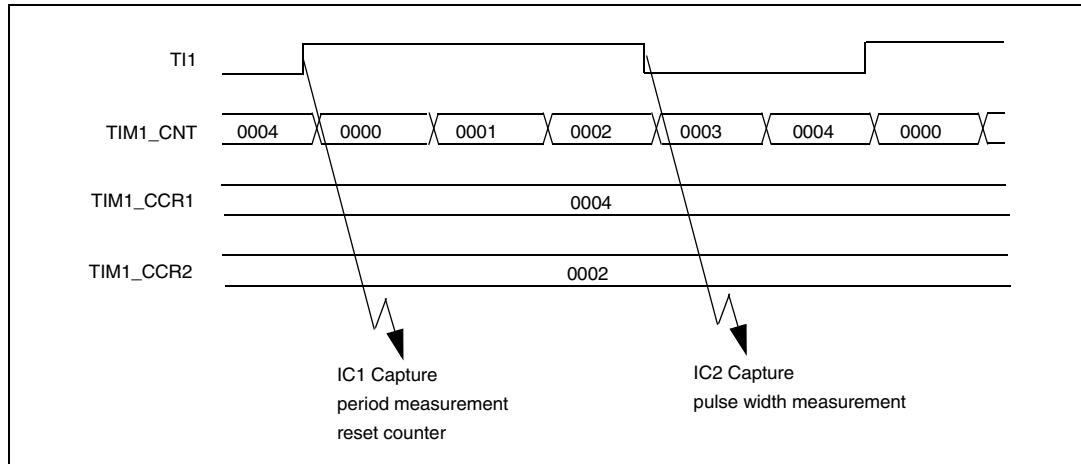
**Figure 101. PWM input signal measurement**



### Procedure

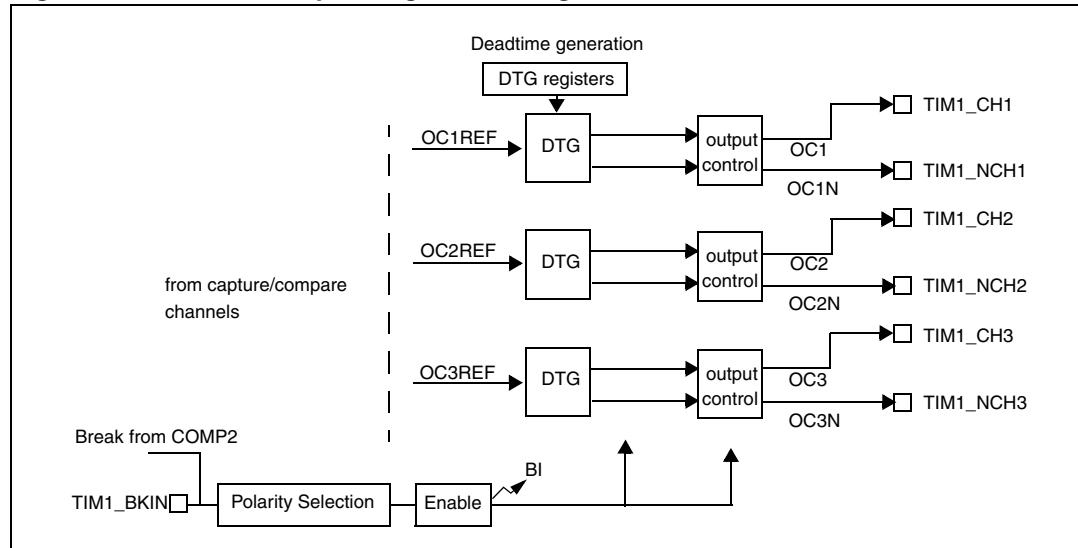
Depending on the f<sub>SYSCLK</sub> frequency and prescaler value, the period (in the TIM1\_CCR1 register) can be measured and the duty cycle (in the TIM1\_CCR2 register) of the PWM can be applied on TI1 using the following procedure:

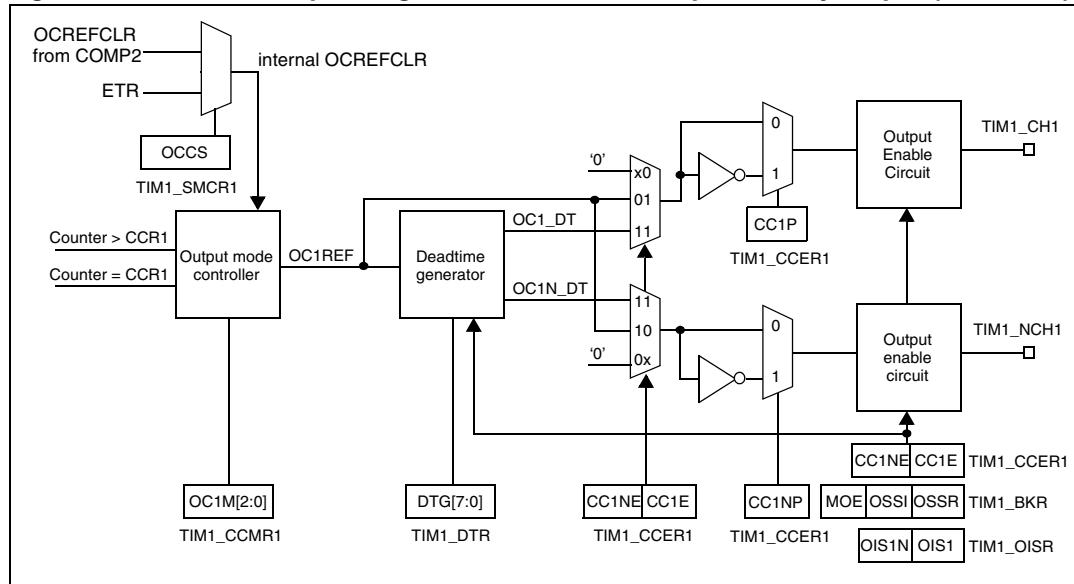
1. Select the active input capture or trigger input for TIM1\_CCR1 by writing the CC1S bits to 01 in the TIM1\_CCMR1 register (TI1FP1 selected).
2. Select the active polarity for TI1FP1 (used for both capture and counter clear in TIMx\_CCR1) by writing the CC1P bit to 0 (TI1FP1 active on rising edge).
3. Select the active input for TIM1\_CCR2 by writing the CC2S bits to 10 in the TIM1\_CCMR2 register (TI1FP2 selected).
4. Select the active polarity for TI1FP2 (used for capture in TIM1\_CCR2) by writing the CC2P bit to 1 (TI1FP2 active on falling edge).
5. Select the valid trigger input by writing the TS bits to 101 in the TIM1\_SMCR register (TI1FP1 selected).
6. Configure the clock/trigger controller in reset mode by writing the SMS bits to 100 in the TIM1\_SMCR register.
7. Enable the captures by writing the CC1E and CC2E bits to 1 in the TIM1\_CCER1 register.

**Figure 102. PWM input signal measurement example**

#### 19.5.4 Output stage

The output stage generates an intermediate waveform called OCi/REF (active high) which is then used for reference. Break functions and polarity act at the end of the chain.

**Figure 103. Channel output stage block diagram**

**Figure 104. Detailed output stage of channel with complementary output (channel 1)**

### 19.5.5 Forced output mode

In output mode (CC/S bits = 00 in the TIM1\_CCMR*i* registers), each output compare signal can be forced to high or low level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal to its active level, write 101 in the OC*i*M bits in the corresponding TIM1\_CCMR*i* registers. OC*i*REF is forced high (OC*i*REF is always active high) and the OC*i* output is forced high or low depending on the CC*i*P polarity bits.

For example, if CC*i*P = 0 (OC*i* active high) => OC*i* is forced high.

The OC*i*REF signal can be forced low by writing the OC*i*M bits to 100 in the TIMx\_CCMRx registers.

Nevertheless, the comparison between the TIM1\_CCR*i* shadow registers and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

### 19.5.6 Output compare mode

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

When a match is found between the capture/compare register and the counter:

- Depending on the output compare mode, the corresponding OC*i* output pin:
  - Keeps its level (OC*i*M = 000),
  - Is set active (OC*i*M = 001),
  - Is set inactive (OC*i*M = 010)
  - Toggles (OC*i*M = 011)
- A flag is set in the interrupt status register (CC*i*F bits in the TIM1\_SR1 register).

- An interrupt is generated if the corresponding interrupt mask is set (CC<sub>i</sub>E bits in the TIM1\_IER register).
- Sends a DMA request if the corresponding enable bit is set (CC<sub>x</sub>DE bit in the TIM1\_DER register)

The output compare mode is defined by the OC<sub>i</sub>M bits in the TIM1\_CCMR<sub>i</sub> registers. The active or inactive level polarity is defined by the CC<sub>i</sub>P bits in the TIM1\_CCER<sub>i</sub> registers.

The TIM1\_CCR<sub>i</sub> registers can be programmed with or without preload registers using the OC<sub>i</sub>PE bits in the TIM1\_CCMR<sub>i</sub> registers.

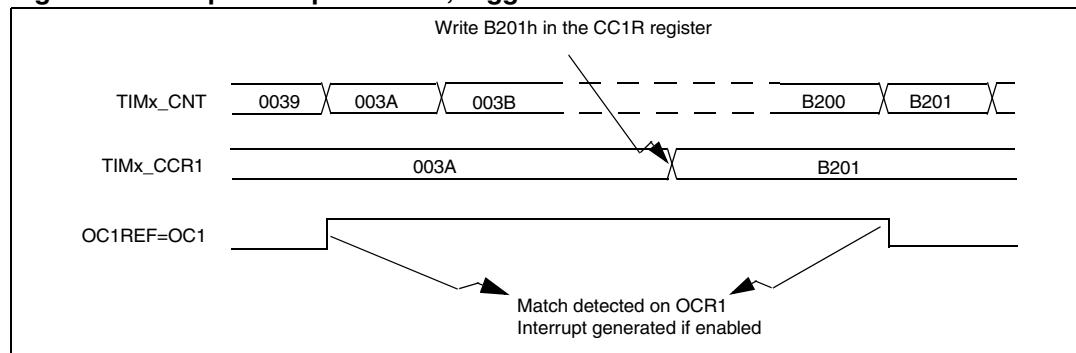
In output compare mode, the UEV has no effect on the OC<sub>i</sub>REF and OC<sub>i</sub> output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse.

#### Procedure

1. Select the counter clock (internal, external, or prescaler).
2. Write the desired data in the TIM1\_ARR and TIM1\_CCR<sub>i</sub> registers.
3. Set the CC<sub>i</sub>E bits if an interrupt request is to be generated.
4. Set the CC<sub>i</sub>DE bit if a DMA request is to be generated.
5. Select the output mode as follows:
  - Write OC<sub>i</sub>M = 011 to toggle the OC<sub>i</sub> output pin when CNT matches CCR<sub>i</sub>
  - Write OC<sub>i</sub>PE = 0 to disable the preload register
  - Write CC<sub>i</sub>P = 0 to select active high polarity
  - Write CC<sub>i</sub>E = 1 to enable the output
6. Enable the counter by setting the CEN bit in the TIMx\_CR1 register

The TIM1\_CCR<sub>i</sub> registers can be updated at any time by software to control the output waveform, provided that the preload registers are not enabled (OC<sub>i</sub>PE = 0). Otherwise, the TIMx\_CCR<sub>i</sub> shadow registers are updated only at the next UEV (see example in [Figure 105](#)).

**Figure 105. Output compare mode, toggle on OC1**



### 19.5.7 PWM mode

Pulse width modulation mode allows you to generate a signal with a frequency determined by the value of the TIM1\_ARR register and a duty cycle determined by the value of the TIM1\_CCR*i* registers.

The PWM mode can be selected independently on each channel (one PWM per OC*i* output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the OC*M* bits in the TIM1\_CCMR*i* registers. The corresponding preload register must be enabled by setting the OC*i*PE bits in the TIM1\_CCMR*i* registers. The auto-reload preload register (in up-counting or center-aligned modes) may be optionally enabled by setting the ARPE bit in the TIM1\_CR1 register.

As the preload registers are transferred to the shadow registers only when an UEV occurs, all registers have to be initialized by setting the UG bit in the TIM1\_EGR register before starting the counter.

OC*i* polarity is software programmable using the CC*i*P bits in the TIM1\_CCER*i* registers. It can be programmed as active high or active low. The OC*i* output is enabled by a combination of CC*E*, MOE, OIS*i*, OSSR and OSSI bits (TIM1\_CCER*i* and TIM1\_BKR registers). Refer to the TIM1\_CCER*i* register descriptions for more details.

In PWM mode (1 or 2), TIM1\_CNT and TIM1\_CCR*i* are always compared to determine whether TIM1\_CCR*i* ≤ TIM1\_CNT or TIM1\_CNT ≤ TIM1\_CCR*i* (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIM1\_CR1 register.

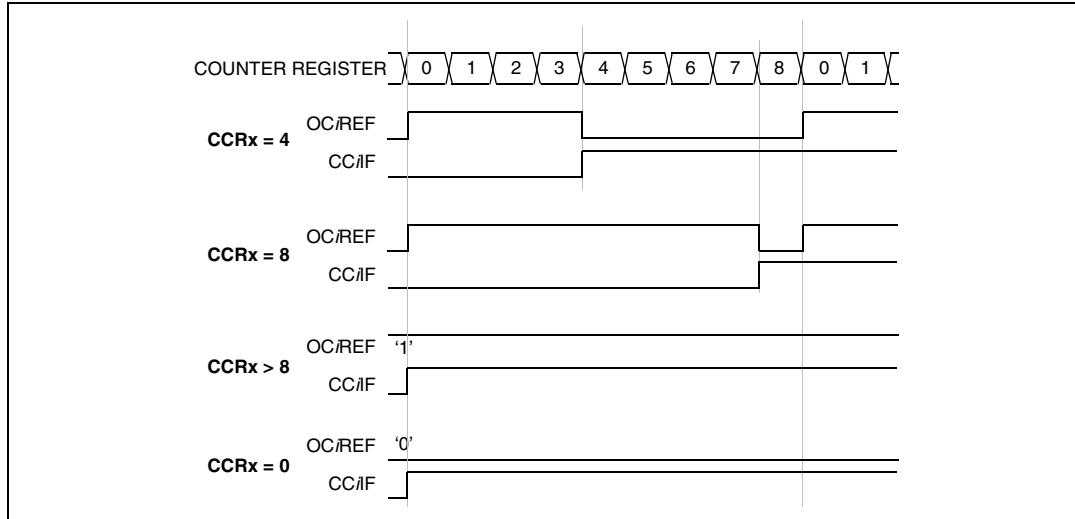
#### PWM edge-aligned mode

##### *Up-counting configuration*

Up-counting is active when the DIR bit in the TIM1\_CR1 register is low.

##### *Example*

This example uses PWM mode 1. The reference PWM signal, OC*i*REF, is high as long as TIM1\_CNT < TIM1\_CCR*i*. Otherwise, it becomes low. If the compare value in TIM1\_CCR*i* is greater than the auto-reload value (in TIM1\_ARR) then OC*i*REF is held at 1. If the compare value is 0, OC*i*REF is held at 0. [Figure 106](#) shows some edge-aligned PWM waveforms in an example where TIM1\_ARR = 8.

**Figure 106. Edge-aligned counting mode PWM mode 1 waveforms (ARR = 8)**

#### Down-counting configuration

Down-counting is active when the DIR bit in the TIM1\_CR1 register is high. Refer to [Down-counting mode on page 291](#)

In PWM mode 1, the reference signal OC/REF is low as long as  $\text{TIM1\_CNT} > \text{TIM1\_CCR}_i$ . Otherwise, it becomes high. If the compare value in the TIM1\_CCR*i* registers is greater than the auto-reload value in the TIM1\_ARR register, OC/REF is held at 1. Zero percent PWM is not possible in this mode.

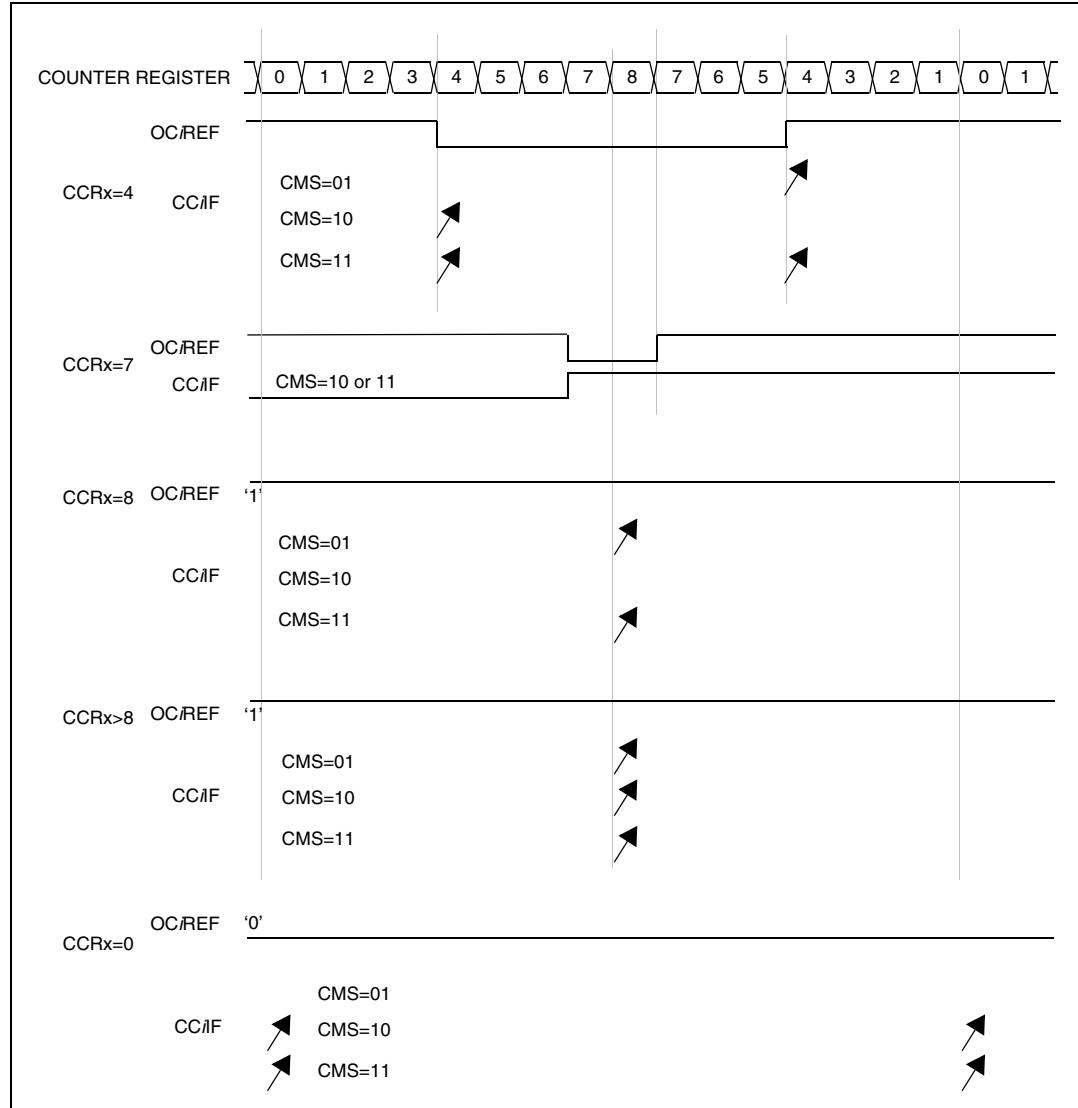
#### PWM center-aligned mode

Center-aligned mode is active when the CMS bits in the TIM1\_CR1 register are different from 00 (all the remaining configurations have the same effect on the OC/REF/OC*i* signals).

The compare flag is set when the counter counts up, down, or up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIM1\_CR1 register is updated by hardware and is read-only in this mode (refer to [Center-aligned mode \(up/down counting\) on page 293](#)).

*Figure 107* shows some center-aligned PWM waveforms in an example where:

- TIM1\_ARR = 8,
- PWM mode is PWM mode 1
- The flag is set (arrow symbol in *Figure 107*) in three different cases:
  - When the counter counts down (CMS = 01)
  - When the counter counts up (CMS = 10)
  - When the counter counts up and down (CMS = 11)

**Figure 107. Center-aligned PWM waveforms (ARR = 8)**

### One-pulse mode

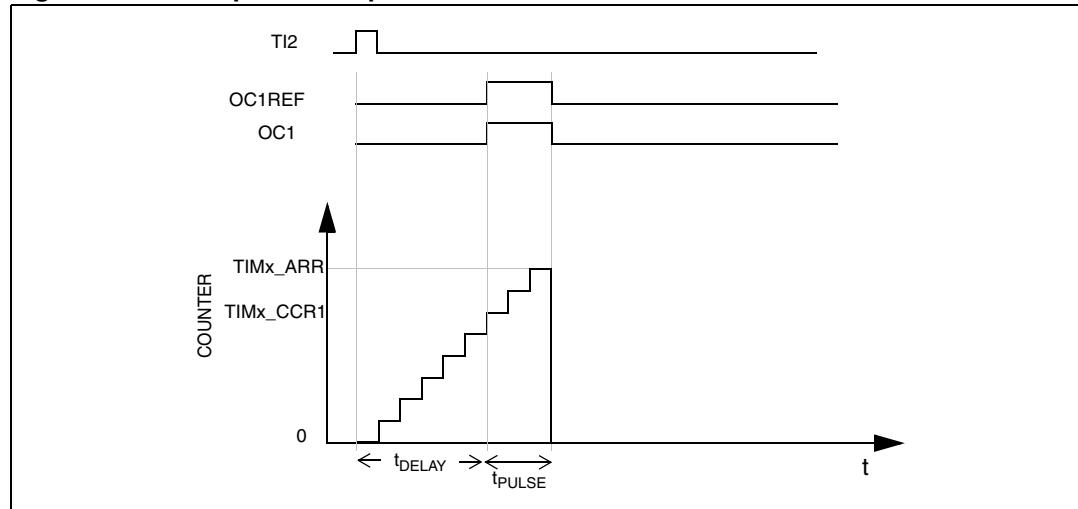
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the clock/trigger controller. Generating the waveform can be done in output compare mode or PWM mode. Select one-pulse mode by setting the OPM bit in the TIM1\_CR1 register. This makes the counter stop automatically at the next UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In up-counting:  $CNT < CCR_i \leq ARR$  (in particular,  $0 < CCR_i$ ),
- In down-counting:  $CNT > CCR_i$

**Figure 108. Example of one-pulse mode**



#### Example

This example shows how to generate a positive pulse on OC1 with a length of  $t_{PULSE}$  and after a delay of  $t_{DELAY}$  as soon as a positive edge is detected on the TI2 input pin.

Follow the procedure below to use IC2 as trigger 1:

- Map IC2 on TI2 by writing CC2S = 01 in the TIM1\_CCMR2 register
- IC2 must detect a rising edge, so write CC2P = 0 in the TIM1\_CCER1 register
- Configure IC2 as trigger for the clock/trigger controller (TRGI) by writing TS = 110 in the TIM1\_SMCR register.
- IC2 is used to start the counter by writing SMS to 110 in the TIM1\_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler) as follows:

- The  $t_{DELAY}$  is defined by the value written in the TIM1\_CCR1 register
- The  $t_{PULSE}$  is defined by the difference between the auto-reload value and the compare value (TIM1\_ARR - TIM1\_CCR1).
- To build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the auto-reload value, enable PWM mode 2 by writing OC1M = 111 in the TIM1\_CCMR1 register. Alternatively, enable the preload registers by writing OC1PE = 1 in the TIM1\_CCMR1 register and ARPE = 0 in the TIM1\_CR1 register (optional). In this case, write the compare value in the TIM1\_CCR1 register and write the auto-reload value in the TIM1\_ARR register. Then, generate an update by setting the UG bit and wait for an external trigger event on TI2. CC1P is written to 0 in this example.

In the example outlined above, the DIR and CMS bits in the TIM1\_CR1 register should be low.

As only one pulse is required, write 1 in the OPM bit in the TIM1\_CR1 register to stop the counter at the next UEV (when the counter rolls over from the auto-reload value back to 0).

#### **Particular case: OC*i* fast enable**

In one-pulse mode, the edge detection on the TI*i* input sets the CEN bit which enables the counter. Then, a comparison between the counter and the compare value makes the output toggle. However, several clock cycles are needed for these operations and this affects the minimum delay ( $t_{DELAY min}$ ) that can be obtained.

To output a waveform with the minimum delay, set the OC<sub>FE</sub> bits in the TIM1\_CCMR<sub>i</sub> registers. OC<sub>i</sub>REF (and OC<sub>i</sub>) are forced in response to the stimulus, without taking the comparison into account. The new level of OC<sub>i</sub>REF (and OC<sub>i</sub>) is the same as if a compare match had occurred. The OC<sub>FE</sub> bits acts only if the channel is configured in PWM1 or PWM2 mode.

#### **Complementary outputs and deadtime insertion**

TIM1 can output two complementary signals per channel. It also manages the switching-off and switching-on instants of the outputs (see [Figure 67: TIM1 general block diagram on page 286](#)).

This time is generally known as deadtime. Deadtimes must be adjusted depending on the characteristics of the devices connected to the outputs (example, intrinsic delays of level-shifters, delays due to power switches).

The polarity of the outputs can be selected (main output OC<sub>i</sub> or complementary OC<sub>i</sub>N) independently for each output. This is done by writing to the CC<sub>i</sub>P and CC<sub>i</sub>NP bits in the TIM1\_CCER<sub>i</sub> registers.

The complementary signals OC<sub>i</sub> and OC<sub>i</sub>N are activated by a combination of several control bits: The CC<sub>i</sub>E and CC<sub>i</sub>NE bits in the TIM1\_CCER<sub>i</sub> register and, if the break feature is implemented, the MOE, OIS<sub>i</sub>, OIS<sub>i</sub>N, OSS<sub>i</sub>, and OSSR bits in the TIM1\_BKR register. Refer to [Table 70: Output control for complementary OC<sub>i</sub> and OC<sub>i</sub>N channels with break feature on page 354](#) for more details. In particular, the deadtime is activated when switching to the IDLE state (when MOE falls to 0).

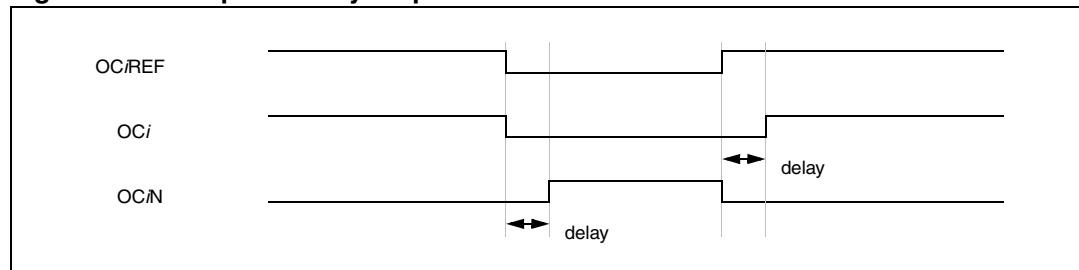
Deadtime insertion is enabled by setting the CC*i* E and CC*i* NE bits, and the MOE bit if the break circuit is present. Each channel embeds an 8-bit deadtime generator. It generates two outputs: OC*i* and OC*i* N from a reference waveform, OC*i* REF. If OC*i* and OC*i* N are active high:

- The OC*i* output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OC*i* N output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

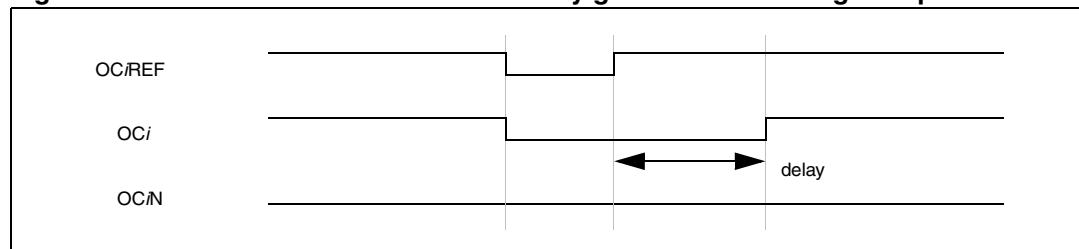
If the delay is greater than the width of the active output (OC*i* or OC*i* N), the corresponding pulse is not generated.

*Figure 109*, *Figure 110*, and *Figure 111* show the relationships between the output signals of the deadtime generator and the reference signal OC*i* REF (where CC*i* P = 0, CC*i* NP = 0, MOE = 1, CC*i* E = 1, and CC*i* NE = 1 in these examples)

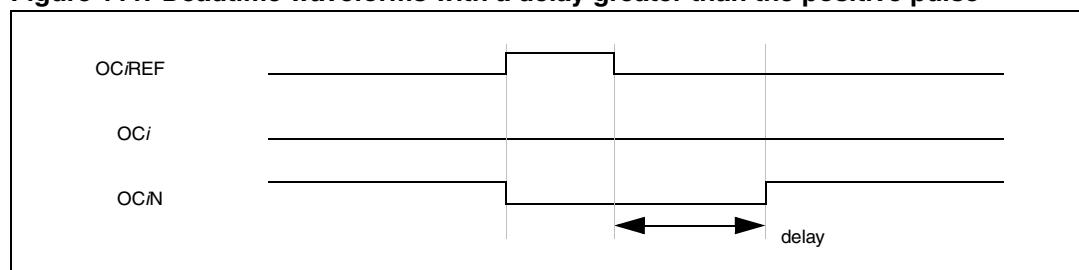
**Figure 109. Complementary output with deadtime insertion**



**Figure 110. Deadtime waveforms with a delay greater than the negative pulse**



**Figure 111. Deadtime waveforms with a delay greater than the positive pulse**



The deadtime delay is the same for each of the channels and is programmable with the DTG bits in the TIM1\_DTR register. Refer to [Section 19.8.32: Deadtime register \(TIM1\\_DTR\) on page 364](#) for delay calculation.

### Re-directing OC<sub>i</sub>REF to OC<sub>i</sub> or OC<sub>i</sub>N

In output mode (forced, output compare, or PWM), OC<sub>i</sub>REF can be re-directed to the OC<sub>i</sub> or OC<sub>i</sub>N outputs by configuring the CC<sub>i</sub>E and CC<sub>i</sub>NE bits in the corresponding TIM1\_CCER<sub>i</sub> registers. This means bypassing the deadtime generator which allows a specific waveform (such as PWM or static active level) to be sent on one output while the complementary output remains at its inactive level. Alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with deadtime.

**Note:** *When only OC<sub>i</sub>N is enabled (CC<sub>i</sub>E = 0, CC<sub>i</sub>NE = 1), it is not complemented and becomes active as soon as OC<sub>i</sub>REF is high. For example, if CC<sub>i</sub>NP = 0 then OC<sub>i</sub>N = OC<sub>i</sub>REF. On the other hand, when both OC<sub>i</sub> and OC<sub>i</sub>N are enabled (CC<sub>i</sub>E = CC<sub>i</sub>NE = 1), OC<sub>i</sub> becomes active when OC<sub>i</sub>REF is high whereas OC<sub>i</sub>N is complemented and becomes active when OC<sub>i</sub>REF is low.*

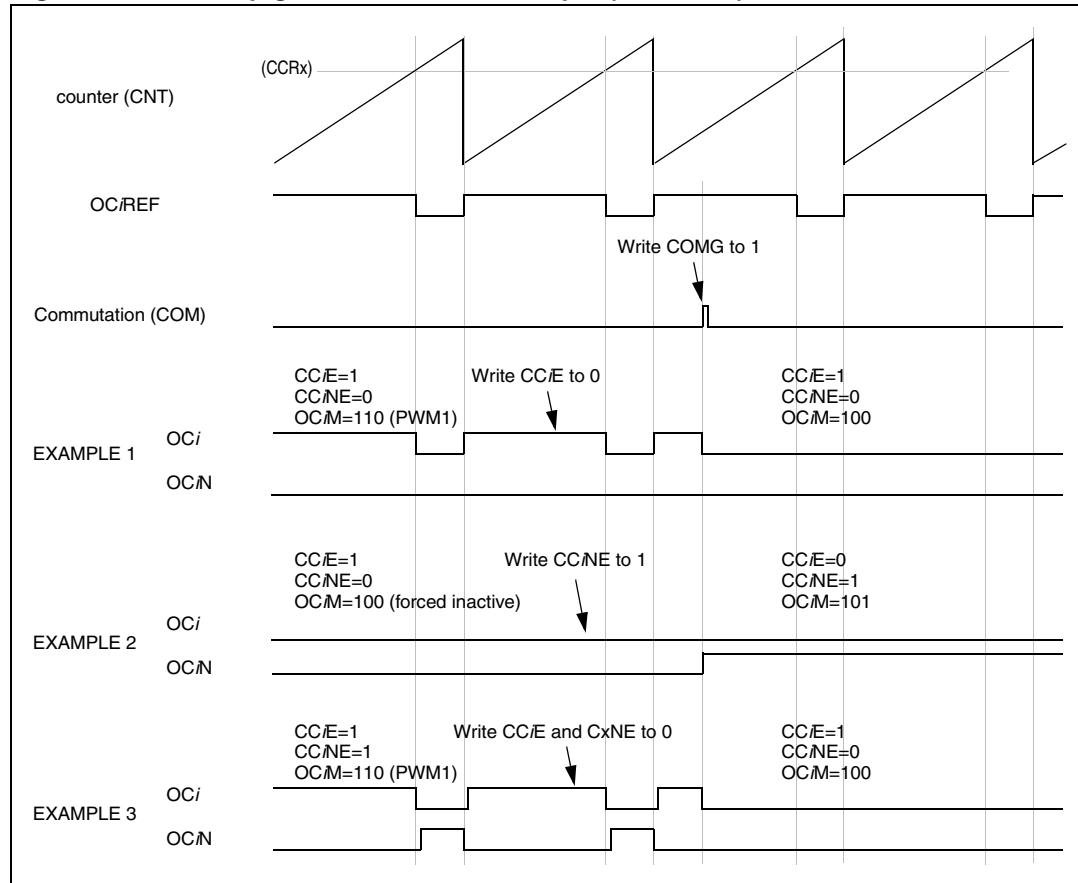
### Six-step PWM generation for motor control

When complementary outputs are implemented on a channel, preload bits are available on the OC<sub>i</sub>M, CC<sub>i</sub>E and CC<sub>i</sub>NE bits. The preload bits are transferred to the active bits at the commutation event (COM). This allows the configuration for the next step to be programmed in advance and for configuration of all the channels to be changed at the same time. The COM event can be generated by software by setting the COMG bit in the TIM1\_EGR register or by hardware trigger (on the rising edge of TRGI).

A flag is set when the COM event occurs (COMIF bit in the TIM1\_SR register) which can generate an interrupt (if the COMIE bit is set in the TIM1\_IER register) or a DMA request (if the COMDE bit is set in the TIM1\_DER register).

*Figure 112* shows the behavior of the OC<sub>i</sub> and OC<sub>i</sub>N outputs when a COM event occurs, for three different examples of programmed configurations.

Figure 112. Six-step generation, COM example (OSSR = 1)



### 19.5.8 Using the break function

The break function is often used in motor control. When using the break function, the output enable signals and inactive levels are modified according to additional control bits (MOE, OSSR and OSSI bits in the TIM1\_BKR register).

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIM1\_BKR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIM1\_BKR register). It results in some delays between the asynchronous and the synchronous signals. For example, if MOE is written to 1 after it has been low, a delay (dummy instruction) must be inserted before it can be read correctly.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or reset state (selected by the OSS1 bit). This happens even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OIS*i* bits in the TIM1\_OISR register as soon as MOE = 0. If OSS1 = 0, the timer releases the enable output otherwise the enable output remains high.
- When complementary outputs are implemented:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, the deadtime generator is reactivated to drive the outputs with the level programmed in the OIS*i* and OIS*i* N bits after a deadtime. Even in this case, OC*i* and OC*i* N cannot be driven to their active level together. Note that because of the resynchronization on MOE, the deadtime duration is a bit longer than usual (around two ck\_tim clock cycles).
- The break status flag (BIF bit in the TIM1\_SR1 register) is set. An interrupt can be generated if the BIE bit in the TIM1\_IER register is set.
- If the AOE bit in the TIM1\_BKR register is set, the MOE bit is automatically set again at the next UEV. This can be used to perform a regulation. Otherwise, MOE remains low until it is written to 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors, or any security components.

*Note:* The break inputs act on signal level. Thus, the MOE bit cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

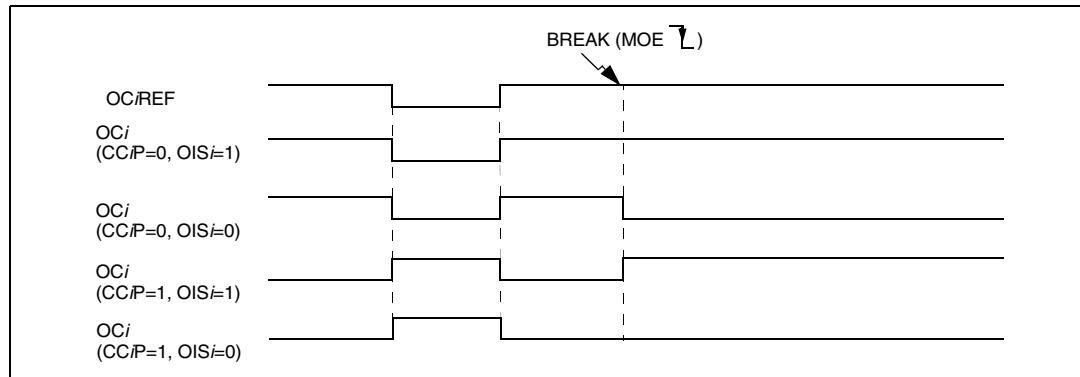
The break can be generated by the break input (BKIN) which has a programmable polarity and can be enabled or disabled by setting or resetting the BKE bit in the TIM1\_BKR register.

In addition to the break inputs and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows the configuration of several parameters (OC*i* polarities and state when disabled, OC*i*M configurations, break enable, and polarity) to be frozen. Three levels of protection can be selected using the LOCK bits in the TIM1\_BKR register. The LOCK bits can be written only once after an MCU reset.

*Figure 113* shows an example of the behavior of the outputs in response to a break.

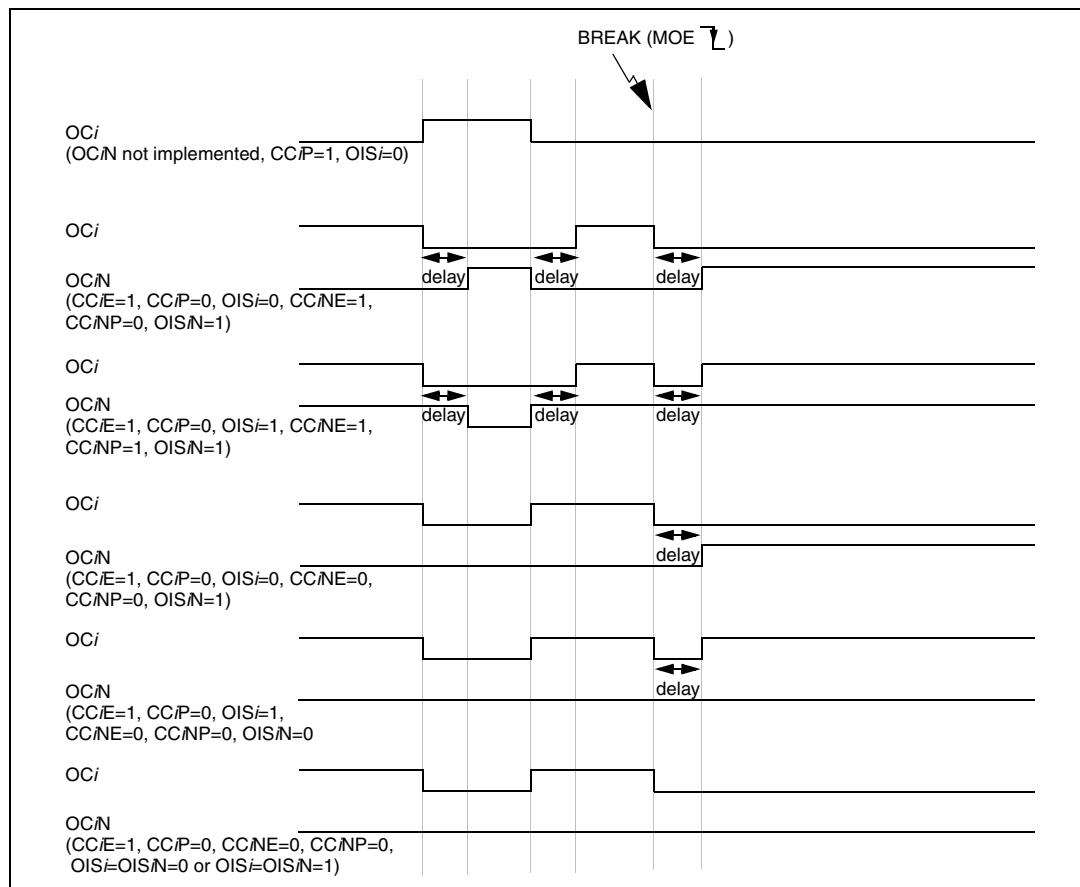
*Note:* The break can also be generated by Comparator 2 through the OUTSEL[1:0] bits in the COMP\_CSR3 register. Refer to the Comparator section.

**Figure 113. Behavior of outputs in response to a break (channel without complementary output)**



*Figure 114* shows an example of behavior of the complementary outputs (TIM1 only) in response to a break.

**Figure 114. Behavior of outputs in response to a break (TIM1 complementary outputs)**



### 19.5.9 Clearing the OCiREF signal on an external event

The OCiREF signal of a given channel can be cleared when a high level is detected on OCREFCLR internal signal (if OCiCE =1 in the TIM1\_CCMR*i* registers, one enable bit per channel). The OCiREF signal remains low until the next UEV occurs. This function can be used in output compare mode and PWM mode only. It does not work in forced mode.

The source of the OCREFCLR internal signal can be selected between the OCREFCLR input and the ETRF (ETR after the filter) by configuring the OCCS bit in the TIM1\_SMCR register

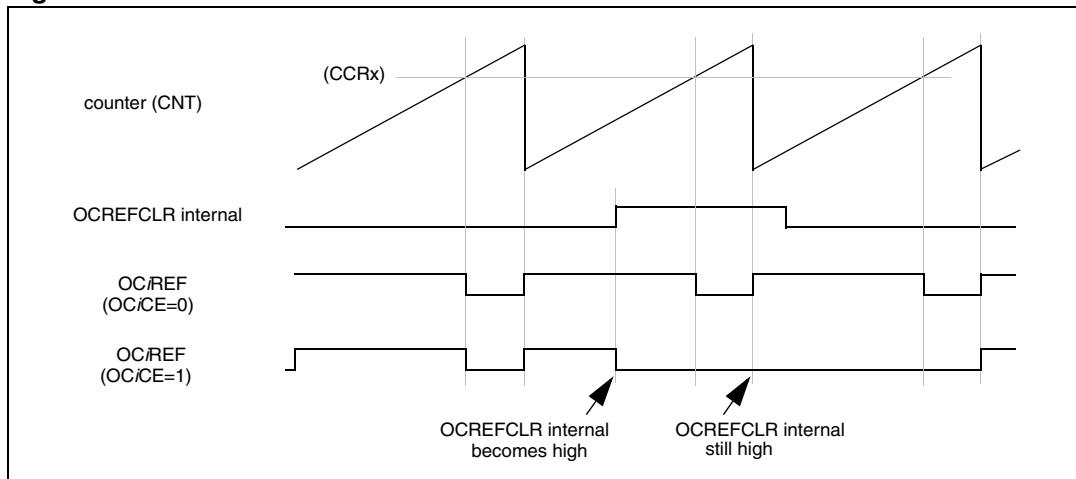
The OCiREF signal can be connected to the output of a comparator and be used for current handling by configuring the external trigger as follows:

1. Switch off the external trigger prescaler by setting bits ETPS[1:0] in the TIM1\_ETR register to 00.
2. Disable external clock mode 2 by setting the ECE bit in the TIM1\_ETR register to 0
3. Configure the external trigger polarity (ETP) and the external trigger filter (ETF) as desired.

Refer to [Figure 83: External trigger input block diagram](#).

[Figure 115](#) shows the behavior of the OCiREF signal when the ETRF input becomes high, for both values of the enable bits OCiCE. In this example, the timer is programmed in PWM mode.

**Figure 115. OCREFCLR activation**



### 19.5.10 Encoder interface mode

Encoder interface mode is typically used for motor control. It can be selected by writing:

- SMS = 001 in the TIM1\_SMCR register if the counter is counting on TI2 edges only
- SMS = 010 if the counter is counting on TI1 edges only
- SMS = 011 if the counter is counting on both TI1 and TI2 edges

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIM1\_CCER1 register. When needed, the input filter can also be programmed.

The two inputs TI1 and TI2 are used to interface an incremental encoder (see [Table 69](#)). If the counter is enabled (when the CEN bit in the TIM1\_CR1 register is written to 1), it is clocked by each valid transition on TI1FP1 or TI2FP2 (see [Figure 100: Input stage of TIM 1 channel 1](#)). The transition sequences of the two inputs (TI1 and TI2) are evaluated and generate count pulses and a direction signal. Depending on the sequence, the counter counts up or down, and the DIR bit in the TIM1\_CR1 register is modified accordingly by hardware. The DIR bit is calculated at each transition based on inputs from either TI1 or TI2, without this being dependent on whether the counter is counting pulses on TI1, TI2 or both.

Encoder interface mode acts as an external clock with direction selection. The counter counts continuously between 0 and the auto-reload value in the TIM1\_ARR register (0 to ARR or ARR down to 0 depending on the direction). TIM1\_ARR must be configured before starting. The capture, compare, prescaler, and trigger output features continue to work as normal in this mode. Encoder mode and external clock mode 2 are not compatible and must not be selected together.

In encoder interface mode, the counter is modified automatically depending on the speed and the direction of the incremental encoder. The content of the counter therefore always represents the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. [Table 69](#) summarizes the possible combinations of counting directions and encoder signals, assuming that TI1 and TI2 do not switch at the same time.

**Table 69. Counting direction versus encoder signals**

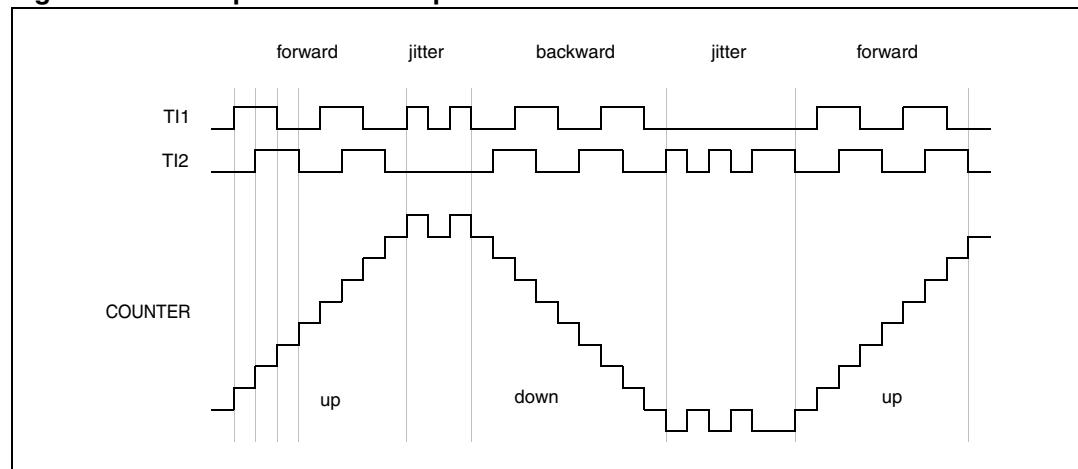
Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No count	No count
	Low	Up	Down	No count	No count
Counting on TI2 only	High	No count	No count	Up	Down
	Low	No count	No count	Down	Up
Counting on both TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

[Figure 116](#) gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near one of the switching points. In the example below, configuration is as follows:

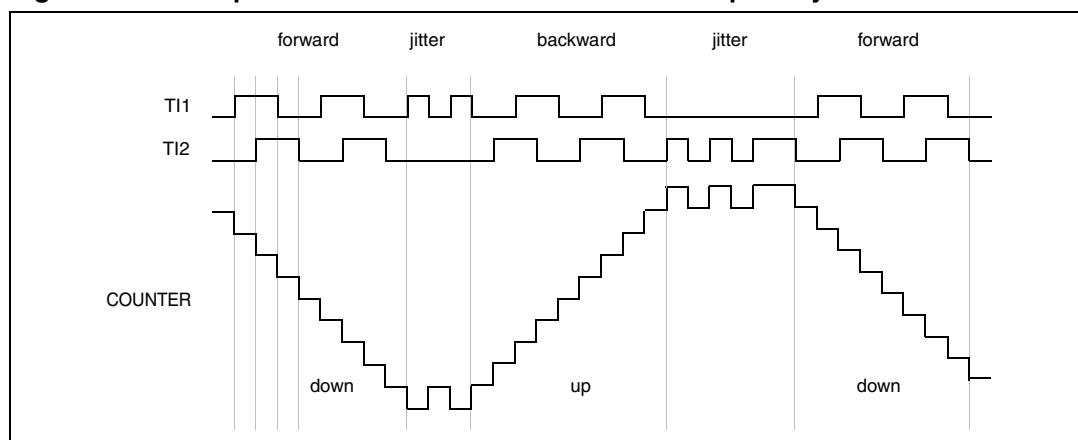
- CC1S = 01 (TIM1\_CCMR1 register, IC1 mapped on TI1)
- CC2S = 01 (TIM1\_CCMR2 register, IC2 mapped on TI2)
- CC1P = 0 (TIM1\_CCER1 register, IC1 non-inverted, IC1=TI1)
- CC2P = 0 (TIM1\_CCER2 register, IC2 non-inverted, IC2=TI2)
- SMS = 011 (TIM1\_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIM1\_CR1 register, counter is enabled)

**Figure 116. Example of counter operation in encoder interface mode**



[Figure 117](#) gives an example of counter behavior when IC1 polarity is inverted (same configuration as [Figure 116](#) except that CC1P =1).

**Figure 117. Example of encoder interface mode with IC1 polarity inverted**



When the timer is configured in encoder interface mode, it provides information on the current position of the sensors. Dynamic information, such as speed, acceleration, and slowdown, can be obtained by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder, which indicates the mechanical zero, can be used for this purpose. Depending on the time between two events, the counter can also be read at regular intervals. This can be done by latching the counter value into a third input capture register, if one is available. In this case, the capture signal must be periodic and can be generated by another timer.

### 19.5.11 Timer input XOR function

If the TI1S bit is set in the TIM1\_CR2 register, the input filter of channel 1 can be connected to the output of an XOR gate, combining the three TIM1\_CH1, TIM1\_CH2 and TIM1\_CH3 input pins.

The XOR output can be used with all timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in the following section.

### 19.5.12 Interfacing with Hall sensors

This is done by using the advanced-control timer (TIM1) to generate PWM signals to drive the motor and another TIMx timer (TIM2, TIM3 or TIM5) referred to as “interfacing timer” in [Figure 118](#). The “interfacing timer” captures the 3 timer input pins connected through an XOR gate to the TI1 input channel (selected by setting the TI1S bit in the TIMx\_CR2 register).

The slave mode controller is configured in reset mode; the slave input is TI1F\_ED. Consequently, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, the capture/compare channel 1 is configured in capture mode and the capture signal is TRC ([See Figure 100: Input stage of TIM 1 channel 1 on page 313](#)). The captured value (which corresponds to the time elapsed between 2 changes on the inputs) gives information about motor speed.

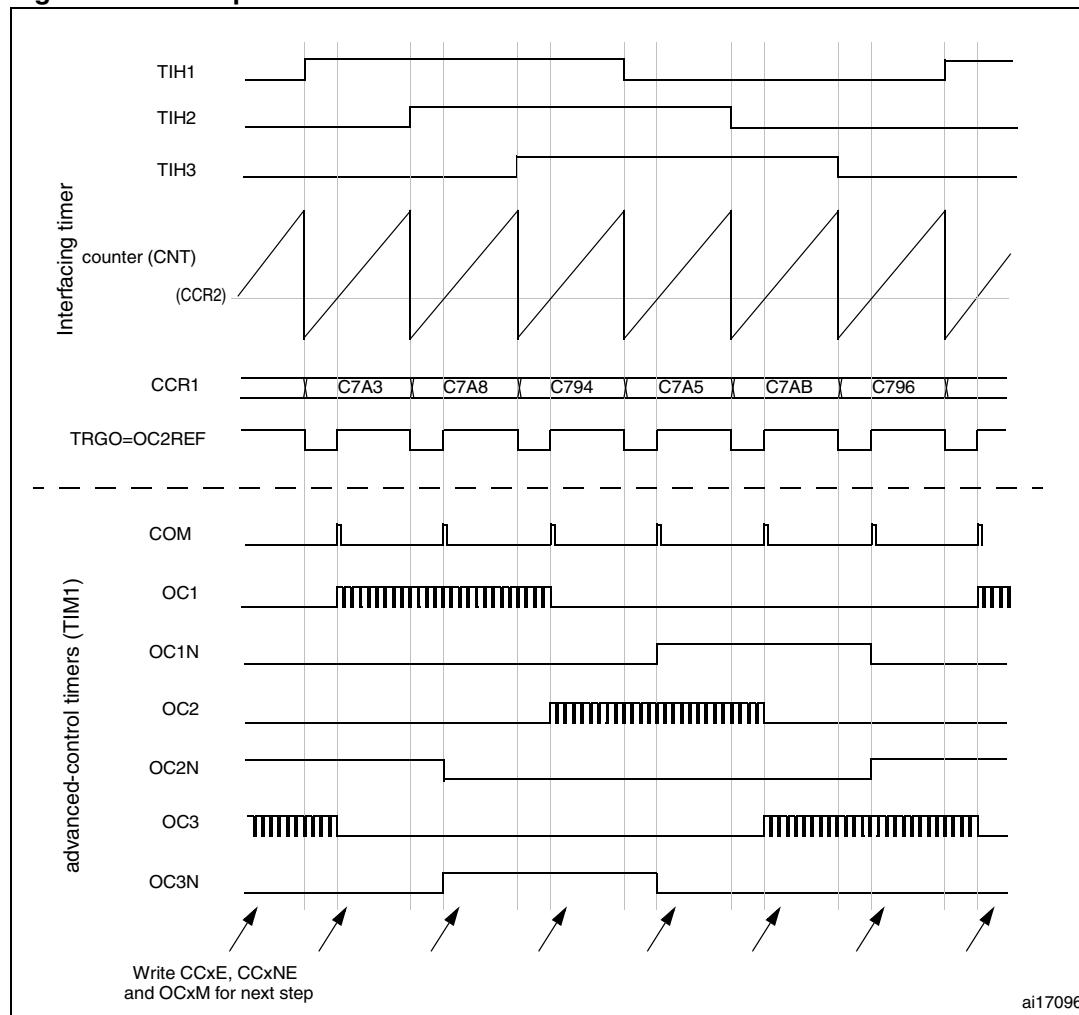
The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (TIM1) (by triggering a COM event). The TIM1 timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so as to generate a positive pulse after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced control timer (TIM1) through the TRGO output.

Example: you want to change the PWM configuration of your TIM1 advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers. Use the following procedure:

- Configure the 3 timer inputs XORed to the TI1 input channel by writing the TI1S bit to 1 in the TIMx\_CR2 register,
- Program the time base: write the TIMx\_ARR to the max value (the counter must be cleared by the TI1 change). Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program the channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx\_CCMR1 register to '11'. You can also program the digital filter if needed,
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to '111' and the CC2S bits to '00' in the TIMx\_CCMR2 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx\_CR2 register to '101'.

In the TIM1 advanced-control timer, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx\_CR2 register) and the COM event is controlled by the trigger input (COMS=1 in the TIMx\_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step.

**Figure 118. Example of Hall sensor interface**



## 19.6 TIM1 interrupts

TIM1 has eight interrupt request sources, mapped on 2 interrupt vectors:

- Break interrupt
- Trigger interrupt
- Commutation interrupt
- Capture/compare 4 interrupt
- Capture/compare 3 interrupt
- Capture/compare 2 interrupt
- Capture/compare 1 interrupt
- Update interrupt (example: overflow, underflow, and counter initialization)

To use the interrupt features for each interrupt channel used, set the desired interrupt enable bits (BIE, TIE, COMIE, CCIE, and UIE) in the TIM1\_IER register to enable interrupt requests.

The different interrupt sources can also be generated by software using the corresponding bits in the TIM1\_EGR register.

### 19.6.1 TIM1 wait-for-event capability

In wait-for-event mode (WFE), TIM1 capture/compare, break, trigger and update interrupts can be used to wake up the device. The interrupt event must have been previously configured through bits TIM1\_EV0 and TIM1\_EV1 in the WFE\_CR1 register (see [Section 9.5: WFE registers](#)).

## 19.7 TIM1 DMA

TIM1 has six DMA request sources:

- Commutation event
- Capture/compare 4 event
- Capture/compare 3 event
- Capture/compare 2 event
- Capture/compare 1 event
- Update event (example: overflow, underflow, counter initialization)

To use the DMA requests set the corresponding “DMA Enable” bit: COMDE, CCDE, UDE in the TIM1\_DER register.

The different DMA requests can be also generated by software using the corresponding bits in the TIM1\_EGR register.

### 19.7.1 DMA single mode

The DMA controller can be configured to transfer a single byte (or two bytes) to or from a single TIM1 register (or two TIM1 registers) by programming the register address in the DMA\_CxPARH and DMA\_CxPTRL registers.

### 19.7.2 DMA burst mode

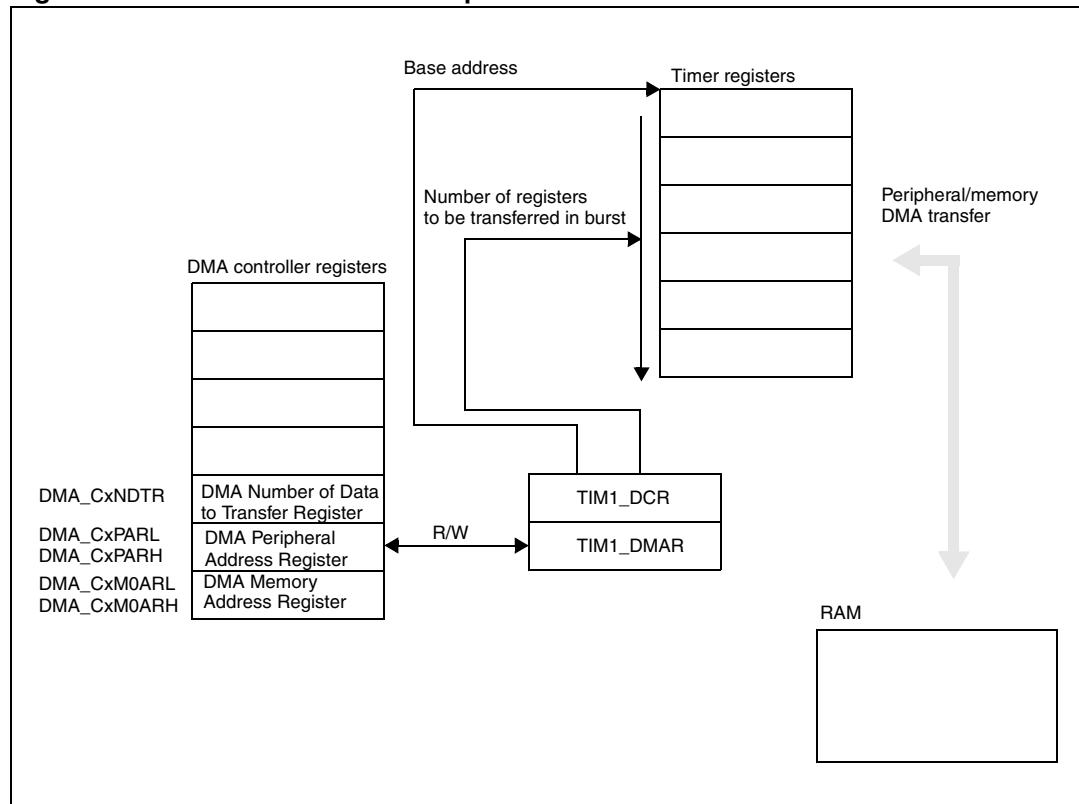
DMA can work in burst mode with TIM1. In this mode, the DMA can transfer a block of data from/to a block of TIM1 registers. The maximum DMA burst length (defined in the DBL[4:0] bits in the TIM1\_DCR2 register) is 32 bytes.

To configure this mode:

1. Write the address of the TIM1\_DMAR register in the DMA\_CxPTRL and DMA\_CxPARH registers.
2. In the TIM1\_DCR register, write the address of the first register to be transferred and the number of registers in the burst.

Refer to [Figure 119](#) for a simplified example of how to configure DMA burst mode.

**Figure 119. DMA burst modes example**



## 19.8 TIM1 registers

### 19.8.1 Control register 1 (TIM1\_CR1)

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **ARPE**: Auto-reload preload enable

- 0: TIM1\_ARR register is not buffered through a preload register. It can be written directly
- 1: TIM1\_ARR register is buffered through a preload register

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

- 00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
- 01: Center-aligned mode 1. The counter counts up and down alternately. Output compare interrupt flags of channels configured in output (CC/S = 00 in TIM1\_CCMR<sub>i</sub> registers) are set only when the counter is counting down.
- 10: Center-aligned mode 2. The counter counts up and down alternately. Output compare interrupt flags of channels configured in output (CC/S = 00 in CCMR<sub>i</sub> registers) are set only when the counter is counting up.
- 11: Center-aligned mode 3. The counter counts up and down alternately. Output compare interrupt flags of channels configured in output (CC/S = 00 in TIM1\_CCMR<sub>i</sub> registers) are set both when the counter is counting up and down.

*Note: It is not allowed to switch from edge-aligned mode to center-aligned mode while the counter is enabled (CEN = 1)*

*Encoder mode (SMS = 001, 010 or 011 in TIM1\_SMCR register) must be disabled in center-aligned mode.*

Bit 4 **DIR**: Direction

- 0: Counter used as up-counter
- 1: Counter used as down-counter

*Note: This bit is read-only when the timer is configured in center-aligned mode or encoder mode.*

Bit 3 **OPM**: One-pulse mode

- 0: Counter is not stopped at update event
- 1: Counter stops counting at the next update event (clearing the CEN bit)

Bit 2 **URS**: Update request source

- 0: When enabled by the UDIS bit, the UIF bit is set and an update interrupt request is sent when one of the following events occurs:

- Registers are updated (counter overflow/underflow)
- UG bit is set by software
- Update event is generated through the clock/trigger controller

- 1: When enabled by the UDIS bit, the UIF bit is set and an update interrupt request is sent only when registers are updated (counter overflow/underflow).

Bit 1 **UDIS**: Update disable.

0: A UEV is generated as soon as a counter overflow occurs, a software update is generated, or a hardware reset is generated by the clock/trigger mode controller. Buffered registers are then loaded with their preload values.

1: A UEV is not generated and shadow registers keep their value (ARR, PSC, CCR<sub>i</sub>). The counter and the prescaler are re-initialized if the UG bit is set or if a hardware reset is received from the clock/trigger mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

*Note: External clock, trigger gated mode, and encoder mode can work only if the CEN bit has been previously set by software. However, trigger mode can set the CEN bit automatically by hardware.*

### 19.8.2 Control register 2 (TIM1\_CR2)

Address offset: 0x01

Reset value: 0x00

7	6	5	4	3	2	1	0
TI1S	MMS[2:0]			CCDS	COMS	Reserved	CCPC
rw	rw	rw	rw	rw	rw		rw

Bit 7 **TI1S:** TI1 (digital filter input) selection

0: TI1 is connected to CH1 input pin

1: TI1 is connected to the 3 inputs CH1, CH2 and CH3 (XORed)

Bits 6:4 **MMS[2:0]:** Master mode selection

These bits select the information to be sent in master mode to the 2 other timers for synchronization (TRGO). The combination is as follows:

000: Reset - The UG bit from the TIM1\_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (clock/trigger mode controller configured in reset mode), the signal on TRGO is delayed compared to the actual reset.

001: Enable - The counter enable signal is used as trigger output (TRGO). It is used to start several timers to control a window in which a slave timer is enabled. The counter enable signal is generated by a logic OR between the CEN control bit and the trigger input when configured in trigger gated mode. When the counter enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIM1\_SMCR register).

010: Update - The update event is selected as trigger output (TRGO)

011: Compare pulse (MATCH1) - The trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurs (TRGO).

100: Compare - OC1REF signal is used as trigger output (TRGO)

101: Compare - OC2REF signal is used as trigger output (TRGO)

110: Compare - OC3REF signal is used as trigger output (TRGO)

111: Compare - OC4REF signal is used as trigger output (TRGO)

Bit 3 **CCDS:** Capture/compare DMA selection

0: TIM1\_CC*i* DMA request sent when a CC*i* (capture/compare) event occurs

1: TIM1\_CC*i* DMA request sent when an update event occurs

Bit 2 **COMS:** Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit.

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

*Note: This bit acts only on channels with complementary outputs.*

Bit 1 Reserved, forced by hardware to 0

Bit 0 **CCPC:** Capture/compare preloaded control

0: The CC*E*, CC*NE*, CC*P*, and CC*NP* bits in the TIM1\_CCER*i* registers and the OC*M* bit in the TIM1\_CCMR*i* registers are not preloaded

1: CC*E*, CC*NE*, CC*P*, CC*NP* and OC*M* bits are preloaded, after having been written, they are updated only when COMG bit is set in the TIM1\_EGR register.

*Note: This bit acts only on channels with complementary outputs.*

### 19.8.3 Slave mode control register (TIM1\_SMCR)

Address offset: 0x02

Reset value: 0x00

7	6	5	4	3	2	1	0
MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **MSM**: Master/slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between TIM1 and another timer (through TRGO).

Bits 6:4 **TS[2:0]**: Trigger selection

This bitfield selects the trigger input (TRGI) to be used to synchronize the counter.

000: Internal trigger ITR0 connected to TIM4 TRGO

001: Internal trigger ITR1 connected to TIM5 TRGO

010: Internal trigger ITR2 connected to TIM3 TRGO

011: Internal trigger ITR3 connected to TIM2 TRGO

100: TI1 edge detector (TI1F\_ED)

101: Filtered timer input 1 (TI1FP1)

110: Filtered timer input 2 (TI2FP2)

111: External trigger input (ETRF)

*Note: These bits must only be changed when they are not used (e.g. when SMS = 000) to avoid incorrect edge detections at the transition.*

Bit 3 **OCCS**: OCREFCLR selection

This bit is set and cleared by software to select the source of the internal OCREFCLR signal.

0: OCREFCLR\_Internal is connected to the OCREFCLR input

1: OCREFCLR\_Internal is connected to ETRF external trigger

Bits 2:0 **SMS[2:0]**: Clock/trigger/slave mode selection

When external signals are selected, the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see input control register and control register description).

000: Clock/trigger controller disabled - If CEN = 1, the prescaler is clocked directly by the internal clock.

001: Encoder mode 1 - Counter counts up or down on TI2FP2 edge depending on TI1FP1 level

010: Encoder mode 2 - Counter counts up or down on TI1FP1 edge depending on TI2FP2 level

011: Encoder mode 3 - Counter counts up or down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

100: Reset mode - Rising edge of the selected trigger signal (TRGI) re-initializes the counter and generates an update of the registers.

101: Trigger gated mode - The counter clock is enabled when the trigger signal (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

110: Trigger standard mode - The counter starts at a rising edge of the trigger TRGI (but, it is not reset). Only the start of the counter is controlled.

111: External clock mode 1 - Rising edges of the selected trigger (TRGI) clock the counter

*Note: Trigger gated mode must not be used if TI1F\_ED is selected as the trigger input (TS = 100).*

*TI1F\_ED outputs 1 pulse for each transition on TI1F, whereas trigger gated mode checks the level of the trigger signal.*

### 19.8.4 External trigger register (TIM1\_ETR)

Address offset: 0x03

Reset value: 0x00

7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **ETP**: External trigger polarity

This bit selects whether ETR or  $\overline{ETR}$  is used for trigger operations

0: ETR is non-inverted, active at high level or rising edge

1: ETR is inverted, active at low level or falling edge

Bit 6 **ECE**: External clock enable

This bit enables external clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

*Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS = 111 and TS = 111 in the TIM1\_SMCR register).*

*It is possible to simultaneously use external clock mode 2 with the following modes: Trigger standard mode, trigger reset mode, and trigger gated mode. Nevertheless, TRGI must not be connected to ETRF in these cases (TS bits must not be 111 in the TIM1\_SMCR register).*

*If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.*

Bits 5:4 **ETPS**: External trigger prescaler

The ETRP frequency must be, at most, 1/4 of  $f_{SYSCLK}$  frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler off

01: ETRP frequency divided by 2

10: ETRP frequency divided by 4

11: ETRP frequency divided by 8

Bits 3:0 **ETF**: External trigger filter.

This bitfield defines the frequency used to sample the ETRP signal and the length of the digital filter applied to it. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at  $f_{SYSCLK}$
- 0001:  $f_{SAMPLING}=f_{SYSCLK}$ ,  $N = 2$
- 0010:  $f_{SAMPLING}=f_{SYSCLK}$ ,  $N = 4$
- 0011:  $f_{SAMPLING}=f_{SYSCLK}$ ,  $N = 8$
- 0100:  $f_{SAMPLING}=f_{SYSCLK}/2$ ,  $N = 6$
- 0101:  $f_{SAMPLING}=f_{SYSCLK}/2$ ,  $N = 8$
- 0110:  $f_{SAMPLING}=f_{SYSCLK}/4$ ,  $N = 6$
- 0111:  $f_{SAMPLING}=f_{SYSCLK}/4$ ,  $N = 8$
- 1000:  $f_{SAMPLING}=f_{SYSCLK}/8$ ,  $N = 6$
- 1001:  $f_{SAMPLING}=f_{SYSCLK}/8$ ,  $N = 8$
- 1010:  $f_{SAMPLING}=f_{SYSCLK}/16$ ,  $N = 5$
- 1011:  $f_{SAMPLING}=f_{SYSCLK}/16$ ,  $N = 6$
- 1100:  $f_{SAMPLING}=f_{SYSCLK}/16$ ,  $N = 8$
- 1101:  $f_{SAMPLING}=f_{SYSCLK}/32$ ,  $N = 5$
- 1110:  $f_{SAMPLING}=f_{SYSCLK}/32$ ,  $N = 6$
- 1111:  $f_{SAMPLING}=f_{SYSCLK}/32$ ,  $N = 8$

### 19.8.5 DMA request enable register (TIM1\_DER)

Address offset: 0x04

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved			COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE
			rw	rw	rw	rw	rw	rw

Bits 7:6 Reserved, must be kept cleared

Bit 5 **COMDE**: Commutation DMA request enable<sup>(1)</sup>

- 0: Commutation DMA request disabled
- 1: Commutation DMA request enabled

Bit 4 **CC4DE**: Capture/compare 4 DMA request enable<sup>(1)</sup>

- 0: CC4 DMA request disabled
- 1: CC4 DMA request enabled

Bit 3 **CC3DE**: Capture/compare 3 DMA request enable<sup>(1)</sup>

- 0: CC3 DMA request disabled
- 1: CC3 DMA request enabled

Bit 2 **CC2DE**: Capture/compare 2 DMA request enable<sup>(1)</sup>

- 0: CC2 DMA request disabled
- 1: CC2 DMA request enabled

Bit 1 **CC1DE**: Capture/compare 1 DMA request enable<sup>(1)</sup>

- 0: CC1 DMA request disabled
- 1: CC1 DMA request enabled

Bit 0 **UDE**: Update DMA request enable<sup>(1)</sup>

- 0: Update DMA request disabled
- 1: Update DMA request enabled

*Note: The conditions for generating a DMA request on the update event are the same as for setting the UIF bit (in the TIM1\_SR1 register). In particular, the DMA request depends on the URS bit (in the TIM1\_CR1 register).*

1. The DMA request is related to the actual event and not to the status bit (in the TIM1\_SR1 register). Thus, no DMA request is sent if the xxDE bit is written to 1 while the corresponding status bit is already set.

### 19.8.6 Interrupt enable register (TIM1\_IER)

Address offset: 0x05

Reset value: 0x00

7	6	5	4	3	2	1	0
BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **BIE**: Break interrupt enable

- 0: Break interrupt disabled
- 1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable

- 0: Trigger interrupt disabled
- 1: Trigger interrupt enabled

Bit 5 **COMIE**: Commutation interrupt enable

- 0: Commutation interrupt disabled
- 1: Commutation interrupt enabled

Bit 4 **CC4IE**: Capture/compare 4 interrupt enable

- 0: CC4 interrupt disabled
- 1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/compare 3 interrupt enable

- 0: CC3 interrupt disabled
- 1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable

- 0: CC2 interrupt disabled
- 1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable

- 0: CC1 interrupt disabled
- 1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

- 0: Update interrupt disabled
- 1: Update interrupt enabled

### 19.8.7 Status register 1 (TIM1\_SR1)

Address offset: 0x06

Reset value: 0x00

7	6	5	4	3	2	1	0
BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
rc_w0							

Bit 7 **BIF:** Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event has occurred

1: An active level has been detected on the break input

Bit 6 **TIF:** Trigger interrupt flag

This flag is set by hardware on a trigger event (the active edge is detected on a TRGI signal, both edges are detected if trigger gated mode is selected). It is cleared by software.

0: No trigger event has occurred

1: Trigger interrupt pending

Bit 5 **COMIF:** Commutation interrupt flag

This flag is set by hardware on a COM (when capture/compare control bits - CCIE, CCNE, OCIM - have been updated). It is cleared by software.

0: No COM has occurred

1: COM interrupt pending

Bit 4 **CC4IF:** Capture/compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF:** Capture/compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF:** Capture/compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF:** Capture/compare 1 interrupt flag

**If channel CC1 is configured as output:**

This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits from TIM1\_CR1 register description). It is cleared by software.

0: No match

1: The content of the counter register TIM1\_CNT matches the content of the TIM1\_CCR1 register

*Note: When the contents of TIMx\_CCRi are greater than the contents of TIMx\_ARR, the CCiIF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode)*

**If channel CC1 is configured as input:**

This bit is set by hardware on a capture. It is cleared by software or by reading the TIM1\_CCR1L register.

0: No input capture has occurred

1: The counter value has been captured in the TIM1\_CCR1 register (an edge has been detected on IC1 which matches the selected polarity).

**Bit 0 UIF:** Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update has occurred

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow or underflow if UDIS = 0 in the TIM1\_CR1 register
- When CNT is re-initialized by software using the UG bit in TIM1\_EGR register, if URS = 0 and UDIS = 0 in the TIM1\_CR1 register.
- When CNT is re-initialized by a trigger event (refer to the TIM1\_SMCR register description), if URS = 0 and UDIS = 0 in the TIM1\_CR1 register.

### 19.8.8 Status register 2 (TIM1\_SR2)

Address offset: 0x07

Reset value: 0x00

7	6	5	4	3	2	1	0
			CC4OF	CC3OF	CC2OF	CC1OF	
Reserved			rc_w0	rc_w0	rc_w0	rc_w0	Reserved

Bits 7:5 Reserved, must be kept cleared

**Bit 4 CC4OF:** Capture/compare 4 overcapture flag

Refer to CC1OF description

**Bit 3 CC3OF:** Capture/compare 3 overcapture flag

Refer to CC1OF description

**Bit 2 CC2OF:** Capture/compare 2 overcapture flag

Refer to CC1OF description

**Bit 1 CC1OF:** Capture/compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected

1: The counter value has been captured in TIM1\_CCR1 register while CC1IF flag was already set

Bit 0 Reserved, must be kept cleared.

### 19.8.9 Event generation register (TIM1\_EGR)

Address offset: 0x08

Reset value: 0x00

7	6	5	4	3	2	1	0
BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
w	w	w	w	w	w	w	w

Bit 7 **BG**: Break generation

This bit is set by software to generate an event. It is automatically cleared by hardware.

0: No action

1: A break event is generated. The MOE bit is cleared and the BIF flag is set. An interrupt is generated if enabled by the BIE bit.

Bit 6 **TG**: Trigger generation

This bit is set by software to generate an event. It is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIM1\_SR1 register. An interrupt is generated if enabled by the TIE bit.

Bit 5 **COMG**: Capture/compare control update generation

This bit can be set by software and is automatically cleared by hardware.

0: No action

1: When the CCPC bit in the TIM1\_CR2 register is set, it allows the CC1E, CC1NE, CC1P, CC1NP, and OC1M bits to be updated.

*Note: This bit acts only on channels that have a complementary output.*

Bit 4 **CC4G**: Capture/compare 4 generation

Refer to CC1G description.

Bit 3 **CC3G**: Capture/compare 3 generation

Refer to CC1G description.

Bit 2 **CC2G**: Capture/compare 2 generation

Refer to CC1G description.

Bit 1 **CC1G**: Capture/compare 1 generation.

This bit is set by software to generate an event. It is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

**If the CC1 channel is configured in output mode:**

The CC1IF flag is set and the corresponding interrupt request is sent if enabled.

**If the CC1 channel is configured in input mode:**

The current value of the counter is captured in the TIM1\_CCR1 register. The CC1IF flag is set, and the corresponding interrupt request is sent if enabled. The CC1OF flag is set if the CC1IF flag is already high.

Bit 0 **UG**: Update generation

This bit can be set by software and is automatically cleared by hardware.

0: No action

1: Re-initializes the counter and generates an update of the registers. Note that the prescaler counter is also cleared. The counter is cleared if center-aligned mode is selected or if DIR = 0 (up-counting). Otherwise, it takes the auto-reload value (TIM1\_ARR) if DIR = 1 (down-counting).

### 19.8.10 Capture/compare mode register 1 (TIM1\_CCMR1)

Address offset: 0x09

Reset value: 0x00

This channel can be used in input (capture mode) or in output (compare mode). The direction of the channel is defined by configuring the CC1S bits. All the other bits of this register have a different function in input and output mode. For a given bit, OC $i$  describes its function when the channel is configured in output, IC $i$  describes its function when the channel is configured in input. Therefore, be aware that the same bit can have a different meaning for the input and output stage.

#### Channel configured in output

7	6	5	4	3	2	1	0
OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw

##### Bit 7 **OC1CE**: Output compare 1 clear enable

This bit is used to enable the clearing of the channel 1 output compare signal (OC1REF) by the OCREFCLR internal signal (see [Section 19.5.9 on page 329](#)).

0: OC1REF is not affected by the OCREFCLR internal signal

1: OC1REF is cleared as soon as a high level is detected on the OCREFCLR internal signal

##### Bits 6:4 **OC1M**: Output compare 1 mode

These bits define the behavior of the output reference signal, OC1REF, from which OC1 is derived. OC1REF is active high whereas OC1 active level depends on the CC1P bit.

000: Frozen - The comparison between the output compare register TIM1\_CCR1 and the counter register TIM1\_CNT has no effect on the outputs.

001: Set channel 1 to active level on match - OC1REF signal is forced high when the counter register TIM1\_CNT matches the capture/compare register 1 (TIM1\_CCR1).

010: Set channel 1 to inactive level on match - OC1REF signal is forced low when the counter register TIM1\_CNT matches the capture/compare register 1 (TIM1\_CCR1).

011: Toggle - OC1REF toggles when TIM1\_CNT = TIM1\_CCR1

100: Force inactive level - OC1REF is forced low

101: Force active level - OC1REF is forced high

110: PWM mode 1 - In up-counting, channel 1 is active as long as TIM1\_CNT < TIM1\_CCR1, otherwise, the channel is inactive. In down-counting, channel 1 is inactive (OC1REF = 0) as long as TIM1\_CNT > TIM1\_CCR1, otherwise, the channel is active (OC1REF = 1).

111: PWM mode 2 - In up-counting, channel 1 is inactive as long as TIM1\_CNT < TIM1\_CCR1, otherwise, the channel is active. In down-counting, channel 1 is active as long as TIM1\_CNT > TIM1\_CCR1, otherwise, the channel is inactive.

*Note: These bits can no longer be modified while LOCK level 3 has been programmed (LOCK bits in TIM1\_BKR register) and CC1S = 00 (the channel is configured in output).*

*In PWM mode 1 or 2, the OC $i$ REF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode (refer to [PWM mode on page 319](#) for more details).*

*On channels that have a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM1\_CR2 register, the OCM active bits take the new value from the preload bits only when a COM is generated.*

**Bit 3 OC1PE:** Output compare 1 preload enable

0: Preload register on TIM1\_CCR1 disabled. TIM1\_CCR1 can be written at anytime. The new value is taken into account immediately.

1: Preload register on TIM1\_CCR1 enabled. Read/write operations access the preload register. TIM1\_CCR1 preload value is loaded in the shadow register at each UEV.

*Note: These bits can no longer be modified while LOCK level 3 has been programmed (LOCK bits in TIM1\_BKR register) and CC1S = 00 (the channel is configured in output).*

*For correct operation, preload registers must be enabled when the timer is in PWM mode. This is not mandatory in one-pulse mode (OPM bit set in TIM1\_CR1 register).*

**Bit 2 OC1FE:** Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on the counter and CCR1 values, even when the trigger is on. The minimum delay to activate CC1 output when an edge occurs on the trigger input, is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on the CC1 output. If this happens, OC is set to the compare level irrespective of the result of the comparison. The delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

**Bits 1:0 CC1S[1:0]:** Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1FP1

10: CC1 channel is configured as input, IC1 is mapped on TI2FP1

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIM1\_SMCR register).

*Note: CC1S bits are writable only when the channel is off (CC1E = 0 in TIM1\_CCER1).*

## Channel configured in input

7	6	5	4	3	2	1	0
IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw

### Bits 7:4 IC1F[3:0]: Input capture 1 filter

This bitfield defines  $f_{\text{SAMPLING}}$ , the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

- 0000: No filter,  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}$
- 0001:  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}$ , N = 2
- 0010:  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}$ , N = 4
- 0011:  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}$ , N = 8
- 0100:  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}/2$ , N = 6
- 0101:  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}/2$ , N = 8
- 0110:  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}/4$ , N = 6
- 0111:  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}/4$ , N = 8
- 1000:  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}/8$ , N = 6
- 1001:  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}/8$ , N = 8
- 1010:  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}/16$ , N = 5
- 1011:  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}/16$ , N = 6
- 1100:  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}/16$ , N = 8
- 1101:  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}/32$ , N = 5
- 1110:  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}/32$ , N = 6
- 1111:  $f_{\text{SAMPLING}} = f_{\text{SYSCLK}}/32$ , N = 8

*Note: Even on channels that have a complementary output, this bit field is not preloaded and does not take into account the content of the CCPC bit (in the TIM1\_CR2 register).*

### Bits 3:2 IC1PSC[1:0]: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E = 0 (TIM1\_CCER register).

- 00: No prescaler, capture is made each time an edge is detected on the capture input
- 01: Capture is made once every 2 events
- 10: Capture is made once every 4 events
- 11: Capture is made once every 8 events

### Bits 1:0 CC1S[1:0]: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) and the used input.

- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, IC1 is mapped on TI1FP1
- 10: CC1 channel is configured as input, IC1 is mapped on TI2FP1
- 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIM1\_SMCR register).

*Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM1\_CCER1).*

### 19.8.11 Capture/compare mode register 2 (TIM1\_CCMR2)

Address offset: 0x0A

Reset value: 0x00

#### Channel configured in output

7	6	5	4	3	2	1	0
OC2CE		OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **OC2CE**: Output compare 2 clear enable

Bits 6:4 **OC2M(2:0)**: Output compare 2 mode

Bit 3 **OC2PE**: Output compare 2 preload enable

Bit 2 **OC2FE**: Output compare 2 fast enable

Bits 1:0 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) and the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2FP2

10: CC2 channel is configured as input, IC2 is mapped on TI1FP2

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIM1\_SMCR register)

*Note: CC2S bits are writable only when the channel is off (CC2E and CC2NE = 0 and updated in TIM1\_CCER1).*

#### Channel configured in input

7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:4 **IC2F**: Input capture 2 filter

Bits 3:2 **IC2PSC(1:0)**: Input capture 2 prescaler

Bits 1:0 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) and the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2FP2

10: CC2 channel is configured as input, IC2 is mapped on TI1FP2

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIM1\_SMCR register).

*Note: CC2S bits are writable only when the channel is off (CC2E and CC2NE = 0 and updated in TIM1\_CCER1).*

### 19.8.12 Capture/compare mode register 3 (TIM1\_CCMR3)

Address offset: 0x0B

Reset value: 0x00

Refer to the CCMR1 register description above.

#### Channel configured in output

7	6	5	4	3	2	1	0
OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 6:4 **OC3M[2:0]**: Output compare 3 mode

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bitfield defines the direction of the channel (input/output) and the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3FP3

10: Reserved

11: CC3 channel is configured as input, IC3 is mapped on TRC. This configuration only works if an internal trigger input is selected through the TS bit in the TIM1\_SMCR register

*Note: CC3S bits are writable only when the channel is off (CC3E and CC3NE = 0 and updated in TIM\_CCER2).*

#### Channel configured in input

7	6	5	4	3	2	1	0
IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
	rw	rw	rw	rw	rw	rw	rw

Bits 7:4 **IC3F**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bitfield defines the direction of the channel (input/output) and the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3FP3

10: Reserved

11: CC3 channel is configured as input, IC3 is mapped on TRC. This configuration only works if an internal trigger input is selected through the TS bit in the TIM1\_SMCR register.

*CC3S bits are writable only when the channel is off (CC3E and CC3NE = 0 and updated in TIM\_CCER2).*

### 19.8.13 Capture/compare mode register 4 (TIM1\_CCMR4)

Address offset: 0xC

Reset value: 0x00

Refer to the CCMR1 register description above.

#### Channel configured in output

7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]				OC4PE	Reserved	CC4S[1:0]
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **OC4CE**: Output compare 4 clear enable

Bits 6:4 **OC4M[2:0]**: Output compare 4 mode

Bit 3 **OC4PE**: Output compare 4 preload enable

Bit 2 Reserved

Bits 1:0 **CC4S[1:0]**: Capture/compare 4 selection

This bitfield defines the direction of the channel (input/output) and the used input.

00: CC4 channel is configured as output

01: Reserved

10: CC4 channel is configured as input, IC4 is mapped on TI3FP4

11: CC4 channel is configured as input, IC4 is mapped on TRC. This configuration only works if an internal trigger input is selected through the TS bit in the TIM1\_SMCR register

*Note: CC4S bits are writable only when the channel is off (CC4E and CC4NE = 0 and updated in TIM1\_CCER2).*

#### Channel configured in input

7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:4 **IC4F**: Input capture 4 filter

Bits 3:2 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 1:0 **CC4S[1:0]**: Capture/compare 4 selection

This bitfield defines the direction of the channel (input/output) and the used input.

00: CC4 channel is configured as output

01: Reserved

10: CC4 channel is configured as input, IC4 is mapped on TI3FP4.

11: CC4 channel is configured as input, IC4 is mapped on TRC. This configuration only works if an internal trigger input is selected through the TS bit in the TIM1\_SMCR register

*Note: CC4S bits are writable only when the channel is off (CC4E and CC4NE = 0 and updated in TIM1\_CCER2).*

### 19.8.14 Capture/compare enable register 1 (TIM1\_CCER1)

Address offset: 0x0D

Reset value: 0x00

7	6	5	4	3	2	1	0
CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **CC2NP**: Capture/compare 2 complementary output polarity  
Refer to CC1NP description.

Bit 6 **CC2NE**: Capture/compare 2 complementary output enable  
Refer to CC1NE description.

Bit 5 **CC2P**: Capture/compare 2 output polarity  
Refer to CC1P description.

Bit 4 **CC2E**: Capture/compare 2 output enable  
Refer to CC1E description.

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity  
0: OC1N active high  
1: OC1N active low

*Note: This bit is no longer writable while LOCK level 2 or 3 have been programmed (LOCK bits in TIM1\_BKR register) and CC1S = 00 (the channel is configured in output).*

*On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM1\_CR2 register, the CC1NP active bit takes the new value from the preload bit only when a COM is generated.*

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable  
0: Off - OC1N is not active. OC1N level is then a function of the MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits.  
1: On - OC1N signal is output on the corresponding output pin depending on the MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits.

*Note: On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM1\_CR2 register, the CC1NE active bit takes the new value from the preload bit when a COM is generated.*

Bit 1 **CC1P**: Capture/compare 1 output polarity

**CC1 channel configured as output:**

0: OC1 active high  
1: OC1 active low

**CC1 channel configured as input for trigger function (see Figure 100):**

0: Trigger on a high level or rising edge of TI1F  
1: Trigger on a low level or falling edge of TI1F

**CC1 channel configured as input for capture function (see Figure 100):**

0: Capture on a rising edge of TI1F or TI2F  
1: Capture on a falling edge of TI1F or TI2F

*Note: This bit is no longer writable while LOCK level 2 or 3 have been programmed (LOCK bits in TIM1\_BKR register).*

*On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM1\_CR2 register, the CC1P active bit takes the new value from the preload bit when a COM is generated.*

Bit 0 **CC1E**: Capture/compare 1 output enable

**CC1 channel is configured as output:**

0: Off - OC1 is not active. OC1 level is then a function of the MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits.

1: On - OC1 signal is output on the corresponding output pin depending on the MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits.

**CC1 channel is configured as input:**

This bit determines if a capture of the counter value can be made in the input capture/compare register 1 (TIM1\_CCR1) or not.

0: Capture disabled

1: Capture enabled

*Note: On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM1\_CR2 register, the CC1E active bit takes the new value from the preload bit only when a COM is generated.*

**Table 70. Output control for complementary OC $i$  and OC $i$ N channels with break feature<sup>(1)</sup>**

Control bits					Output states	
MOE	OSSI	OSSR	CC1E	CC1NE	OC $i$	OC $i$ N
1	x <sup>(2)</sup>	0	0	0	Output disabled (not driven by the timer)	Output disabled (not driven by the timer)
		0	0	1	Output disabled (not driven by the timer)	OC $i$ REF + polarity OC $i$ N = OC $i$ REF xor CC $i$ NP
		0	1	0	OC $i$ REF + polarity OC $i$ = OC $i$ REF xor CC $i$ P	Output disabled (not driven by the timer)
		0	1	1	OC $i$ REF + polarity + deadtime	Complementary to OC $i$ REF (not OC $i$ REF) + polarity + deadtime
		1	0	0	Output disabled (not driven by the timer)	Output disabled (not driven by the timer)
		1	0	1	Off state (output enabled with inactive state) OC $i$ = CC $i$ P	OC $i$ REF + polarity OC $i$ N = OC $i$ REF xor CC $i$ NP
		1	1	0	OC $i$ REF + polarity OC $i$ = OC $i$ REF xor CC $i$ P	Off state (output enabled with inactive state) OC $i$ N = CC $i$ NP
		1	1	1	OC $i$ REF + polarity + deadtime	Complementary to OC $i$ REF (not OC $i$ REF) + polarity + deadtime

**Table 70. Output control for complementary OC*i* and OC*iN* channels with break feature<sup>(1)</sup> (continued)**

Control bits					Output states	
MOE	OSSI	OSSR	CCiE	CCiNE	OC <i>i</i>	OC <i>iN</i>
0	0	x <sup>(2)</sup>	x	x	Output disabled (not driven by the timer)	
	0					
	0				Off state (output enabled with inactive state) Asynchronously: OC <i>i</i> = CC <i>iP</i> and OC <i>iN</i> = CC <i>iNP</i> Then if the clock is present: OC <i>i</i> = OIS <i>i</i> and OC <i>iN</i> = OIS <i>N</i> after a deadtime, assuming that OIS <i>i</i> and OIS <i>N</i> do not correspond with OC <i>i</i> and OC <i>N</i> in active state	
	1					
	1					
	1					
	1					

1. Never set CCiE = CCNiE = 0 when the channel is used. When the channel is not used, program CCiP = CCiNP = OISi = OISiN = 0 otherwise.

2. Don't care

**Note:** The state of the external I/O pins connected to the OC*i* channels depends on the OC*i* channel state and the GPIO registers.

### 19.8.15 Capture/compare enable register 2 (TIM1\_CCER2)

Address offset: 0x0E

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved			CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E
	rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:6 Reserved

Bit 5 **CC4P**: Capture/compare 4 output polarity

Refer to CC1P description. As this bit is not connected to the external output, it is not used when the channel 4 is configured in output mode.

Bit 4 **CC4E**: Capture/compare 4 output enable

Refer to CC1E description. As this bit is not connected to the external output, it is not used when the channel 4 is configured in output mode.

Bit 3 **CC3NP**: Capture/compare 3 complementary output polarity

Refer to CC1NP description.

Bit 2 **CC3NE**: Capture/compare 3 complementary output enable

Refer to CC1NE description.

Bit 1 **CC3P**: Capture/compare 3 output polarity

Refer to CC1P description.

Bit 0 **CC3E**: Capture/compare 3 output enable

Refer to CC1E description.

### 19.8.16 Counter high (TIM1\_CNTRH)

Address offset: 0x0F

Reset value: 0x00

	7	6	5	4	3	2	1	0
CNT[15:8]								
	rw							

Bits 7:0 **CNT[15:8]**: Counter value (MSB)

### 19.8.17 Counter low (TIM1\_CNTRL)

Address offset: 0x10

Reset value: 0x00

7	6	5	4	3	2	1	0
CNT[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CNT[7:0]**: Counter value (LSB).

### 19.8.18 Prescaler high (TIM1\_PSCRH)

Address offset: 0x11

Reset value: 0x00

7	6	5	4	3	2	1	0
PSC[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

**PSC[15:8]**: Prescaler value (MSB)

Bits 7:0 The prescaler value divides the CK\_PSC clock frequency. The counter clock frequency  $f_{CK\_CNT}$  is equal to  $f_{CK\_PSC} / (PSCR[15:0]+1)$ . PSCR contain the value which is loaded in the active prescaler register at each UEV (including when the counter is cleared through the UG bit of the TIM\_EGR register or through the trigger controller when configured in trigger reset mode). A UEV must be generated so that a new prescaler value can be taken into account.

### 19.8.19 Prescaler low (TIM1\_PSCRL)

Address offset: 0x12

Reset value: 0x00

7	6	5	4	3	2	1	0
PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **PSC[7:0]**: Prescaler value (LSB)

The prescaler value divides the CK\_PSC clock frequency. The counter clock frequency  $f_{CK\_CNT}$  is equal to  $f_{CK\_PSC} / (PSCR[15:0]+1)$ . PSCR contains the value which is loaded in the active prescaler register at each UEV (including when the counter is cleared through the UG bit of the TIM1\_EGR register or through the trigger controller when configured in trigger reset mode).

A UEV must be generated so that a new prescaler value can be taken into account.

### 19.8.20 Auto-reload register high (TIM1\_ARRH)

Address offset: 0x13

Reset value: 0xFF

7	6	5	4	3	2	1	0
ARR[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **ARR[15:8]**: Auto-reload value (MSB)

ARR is the value to be loaded in the actual auto-reload register. Refer to the [Section 19.3: TIM1 time base unit on page 287](#) for more details about ARR update and behavior. The counter is blocked while the auto-reload value is null.

### 19.8.21 Auto-reload register low (TIM1\_ARRL)

Address offset: 0x14

Reset value: 0xFF

7	6	5	4	3	2	1	0
ARR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **ARR[7:0]**: Auto-reload value (LSB).

### 19.8.22 Repetition counter register (TIM1\_RCR)

Address offset: 0x15

Reset value: 0xFF

7	6	5	4	3	2	1	0
REP[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **REP[7:0]**: Repetition counter value.

When the preload registers are enabled, these bits allow the user to set up the update rate of the compare registers (periodic transfers from preload to shadow registers) as well as the update interrupt generation rate if the update interrupt is enabled (UIE=1).

Each time the REP\_CNT related down-counter reaches zero, a UEV is generated and it restarts counting from the REP value. As REP\_CNT is reloaded with the REP value only at the repetition update event U\_RC, any write to the TIM1\_RCR register is not taken into account until the next repetition update event.

In PWM mode (REP+1) corresponds to:

- The number of PWM periods in edge-aligned mode
- The number of half PWM periods in center-aligned mode

### 19.8.23 Capture/compare register 1 high (TIM1\_CCR1H)

Address offset: 0x16

Reset value: 0x00

7	6	5	4	3	2	1	0
CCR1[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CCR1[15:8]**: Capture/compare 1 value (MSB)

**If the CC1 channel is configured as output (CC1S bits in TIM1\_CCMR1 register):**

The value of CCR1 is loaded permanently into the actual capture/compare 1 register if the preload feature is enabled (OC1PE bit in TIMx\_CCMR1). Otherwise, the preload value is copied in the active capture/compare 1 register when a UEV occurs. The active capture/compare register contains the value which is compared to the counter register, TIMx\_CNT, and signalled on the OC1 output.

**If the CC1 channel is configured as input (CC1S bits in TIM1\_CCMR1 register):**

The value of CCR1 is the counter value transferred by the last input capture 1 event (IC1). In this case, these bits are read only.

### 19.8.24 Capture/compare register 1 low (TIM1\_CCR1L)

Address offset: 0x17

Reset value: 0x00

7	6	5	4	3	2	1	0
CCR1[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CCR1[7:0]**: Capture/compare 1 value (LSB)

### 19.8.25 Capture/compare register 2 high (TIM1\_CCR2H)

Address offset: 0x18

Reset value: 0x00

7	6	5	4	3	2	1	0
CCR2[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CCR2[15:8]**: Capture/compare 2 value (MSB)

**If the CC2 channel is configured as output (CC2S bits in TIM1\_CCMR2 register):**

The value of CCR2 is loaded permanently into the actual capture/compare 2 register if the preload feature is not enabled (OC2PE bit in TIM1\_CCMR2). Otherwise, the preload value is copied in the active capture/compare 2 register when a UEV occurs. The active capture/compare register contains the value which is compared to the counter register, TIM1\_CNT, and signalled on the OC2 output.

**If the CC2 channel is configured as input (CC2S bits in TIM1\_CCMR2 register):**

The value of CCR2 is the counter value transferred by the last input capture 2 event (IC2). In this case, these bits are read only.

### 19.8.26 Capture/compare register 2 low (TIM1\_CCR2L)

Address offset: 0x19

Reset value: 0x00

7	6	5	4	3	2	1	0
CCR2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CCR2[7:0]**: Capture/compare value (LSB)

### 19.8.27 Capture/compare register 3 high (TIM1\_CCR3H)

Address offset: 0x1A

Reset value: 0x00

7	6	5	4	3	2	1	0
CCR3[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CCR3[15:8]**: Capture/compare value (MSB)

**If the CC3 channel is configured as output (CC3S bits in TIM\_CCMR3 register):**

The value of CCR3 is loaded permanently into the actual capture/compare 3 register if the preload feature is not enabled (OC3PE bit in TIM1\_CCMR3). Otherwise, the preload value is copied in the active capture/compare 3 register when a UEV occurs. The active capture/compare register contains the value which is compared to the counter register, TIM1\_CNT, and signalled on the OC3 output.

**If the CC3 channel is configured as input (CC3S bits in TIM\_CCMR3 register):**

The value of CCR3 is the counter value transferred by the last input capture 3 event (IC31).

### 19.8.28 Capture/compare register 3 low (TIM1\_CCR3L)

Address offset: 0x1B

Reset value: 0x00

7	6	5	4	3	2	1	0
CCR3[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CCR3[7:0]**: Capture/compare value (LSB)

### 19.8.29 Capture/compare register 4 high (TIM1\_CCR4H)

Address offset: 0x1C

Reset value: 0x00

7	6	5	4	3	2	1	0
CCR4[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CCR4[15:8]**: Capture/compare value (MSB)

*If the CC4 channel is configured as output (CC4S bits in TIM1\_CCMR4 register):*

The value of CCR4 is loaded permanently into the actual capture/compare 4 register if the preload feature is not enabled (OC4PE bit in TIM1\_CCMR4). Otherwise, the preload value is copied in the active capture/compare 4 register when a UEV occurs. The active capture/compare register contains the value which is compared to the counter register, TIM1\_CNT, and signalled on the OC4 output.

*If the CC4 channel is configured as input (CC4S bits in TIM1\_CCMR4 register):*

The value of CCR4 is the counter value transferred by the last input capture 4 event (IC4).

### 19.8.30 Capture/compare register 4 low (TIM1\_CCR4L)

Address offset: 0x1D

Reset value: 0x00

7	6	5	4	3	2	1	0
CCR4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CCR4[7:0]**: Capture/compare value (LSB)

### 19.8.31 Break register (TIM1\_BKR)

Address offset: 0x1E

Reset value: 0x00

7	6	5	4	3	2	1	0
LOCK							
MOE	AOE	BKP	BKE	OSSR	OSSI		
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It acts only on the channels which are configured in output.

0: OC and OCN outputs are disabled or forced to idle state

1: OC and OCN outputs are enabled if their respective enable bits are set (CC*E* in TIM1\_CCER*i* registers).

See OC/OCN enable description for more details ([Section 19.8.14 on page 353](#)).

Bit 6 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next UEV (if the break input is not active)

*Note:* This bit can no longer be modified while LOCK level 1 has been programmed (LOCK bits in the TIM1\_BKR register).

Bit 5 **BKP**: Break polarity

0: Break input BKIN is active low

1: Break input BKIN is active high

*Note:* This bit can no longer be modified while LOCK level 1 has been programmed (LOCK bits in the TIM1\_BKR register).

Bit 4 **BKE**: Break enable

0: Break input (BKIN) disabled

1: Break input (BKIN) enabled

*Note:* This bit can no longer be modified while LOCK level 1 has been programmed (LOCK bits in the TIM1\_BKR register).

Bit 3 **OSSR**: Off state selection for Run mode

This bit is used when MOE = 1 on channels with a complementary output which are configured as outputs. See OC/OCN enable description for more details ([Section 19.8.14](#)).

0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal = 0)

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CC*E* = 1 or CC*NE* = 1, after which the OC/OCN enable output signal = 1

*Note:* This bit can no longer be modified while LOCK level 2 has been programmed (LOCK bits in the TIM1\_BKR register).

Bit 2 **OSSI**: Off state selection for idle mode

This bit is used when MOE = 0 on channels configured as outputs. See OC enable description for more details ([Section 19.8.14](#)).

0: When inactive, OC*i* outputs are disabled (OC*i* enable output signal = 0)

1: When inactive, OC*i* outputs are forced first with their idle level as soon as CC*E* = 1 (OC enable output signal = 1)

*Note:* This bit can no longer be modified while LOCK level 2 has been programmed (LOCK bits in the TIM1\_BKR register).

### Bits 1:0 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK off - No bits are write protected

01: LOCK level 1 - OIS*i* bit in TIM1\_OISR register and BKE/BKP/AOE bits in TIM1\_BKR register can no longer be written.

10: LOCK level 2 - LOCK level 1 + CC polarity bits (CC*i*P bits in TIM1\_CCER*i* registers, as long as the related channel is configured in output through the CC/S bits) as well as the OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 - LOCK level 2 + CC control bits (OC*i*M and OC*i*PE bits in TIM1\_CCMR*i* registers, as long as the related channel is configured in output through the CC/S bits) can no longer be written.

*Note:* The LOCK bits can be written only once after reset. Once the TIM1\_BKR register has been written, their content is frozen until the next reset.

*Note:* As the bits AOE, BKP, BKE, OSSR, and OSSI can be write-locked depending on the LOCK configuration, it is necessary to configure all of them during the first write access to the TIM1\_BKR register.

### 19.8.32 Deadtime register (TIM1\_DTR)

Address offset: 0x1F

Reset value: 0x00

7	6	5	4	3	2	1	0
DTG7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

### Bits 7:0 **DTG[7:0]**: Deadtime generator set-up

This bitfield defines the duration of the deadtime inserted between the complementary outputs. DT corresponds to this duration. t<sub>CK\_PSC</sub> is the TIM clock pulse.

DTG[7:5] = 0xx => DT= DTG[7:0] x t<sub>dtg</sub> with t<sub>dtg</sub> = t<sub>CK\_PSC</sub> (f1)

DTG[7:5] = 10x => DT= (64 + DTG[5:0]) x t<sub>dtg</sub> with t<sub>dtg</sub>= 2 x t<sub>CK\_PSC</sub> (f2)

DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x t<sub>dtg</sub> with t<sub>dtg</sub>= 8 x t<sub>CK\_PSC</sub> (f3)

DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x t<sub>dtg</sub> with t<sub>dtg</sub> = 16 x t<sub>CK\_PSC</sub> (f4)

#### **Example**

If t<sub>CK\_PSC</sub>= 125 ns (8 MHz), deadtime possible values are:

DTG[7:0] = 0 x 0 to 0 x 7F from 0 to 15875 ns in 125 ns steps (refer to f1)

DTG[7:0] = 0 x 80 to 0 x BF from 16 µs to 31750 ns in 250 ns steps (refer to f2)

DTG[7:0] = 0 x C0 to 0 x DF from 32 µs to 63 µs in 1µs steps (refer to f3)

DTG[7:0] = 0 x E0 to 0 x FF from 64 µs to 126 µs in 2 µs steps (refer to f4)

*Note:* This bitfield can not be modified while LOCK level 1, 2, or 3 have been programmed (LOCK bits in the TIM\_BKR register).

### 19.8.33 Output idle state register (TIM1\_OISR)

Address offset: 0x20

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1		
	rw	rw	rw	rw	rw	rw		

Bits 7:6 Reserved, forced by hardware to 0

Bit 5 **OIS3N**: Output idle state 3 (OC3N output)

Refer to OIS1N bit

Bit 4 **OIS3**: Output idle state 3 (OC3 output)

Refer to OIS1 bit

Bit 3 **OIS2N**: Output idle state 2 (OC2N output)

Refer to OIS1N bit

Bit 2 **OIS2**: Output idle state 2 (OC2 output)

Refer to OIS1 bit

Bit 1 **OIS1N**: Output idle state 1 (OC1N output).

0: OC1N = 0 after a deadtime when MOE = 0

1: OC1N = 1 after a deadtime when MOE = 0

*Note: This bit can no longer be modified while LOCK level 1, 2 or 3 have been programmed (LOCK bits in the TIM1\_BKR register).*

Bit 0 **OIS1**: Output idle state 1 (OC1 output).

0: OC1=0 (after a deadtime if OC1N is implemented) when MOE=0

1: OC1=1 (after a deadtime if OC1N is implemented) when MOE=0

*Note: This bit can no longer be modified while LOCK level 1, 2 or 3 have been programmed (LOCK bits in the TIM1\_BKR register).*

### 19.8.34 DMA control register 1 (TIM1\_DCR1)

Address offset: 0x21

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved	DBA[4:0]							
	rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:5 Reserved, forced by hardware to 0

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit vector defines the base-address for DMA transfers in burst mode (when read/write access is made through the TIM1\_DMAR address). DBA is defined as an offset starting from the address of the TIM1\_CR1 register.

00000: TIM1\_CR1

00001: TIM1\_CR2

00010: TIM1\_SMCR

00011: TIM1\_ETR

...

### 19.8.35 DMA control register 2 (TIM1\_DCR2)

Address offset: 0x22

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved		DBL[4:0]					
			rw	rw	rw	rw	rw

Bits 7:5 Reserved, forced by hardware to 0

Bits 4:0 **DBL[4:0]: DMA burst length**

This 5-bit vector defines the length of DMA transfers in burst mode by setting the number of registers to be accessed.

00000: 1 register

00001: 2 registers

00010: 3 registers

...

11111: 32 registers

*Note:* The timer recognizes a burst transfer when a read or a write access is made to the TIM1\_DMAR register.

### 19.8.36 DMA address for burst mode (TIM1\_DMAR)

Address offset: 0x23

Reset value: 0x00

7	6	5	4	3	2	1	0
DMAB[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **DMAB[15:0]: DMA register for burst accesses**

A read or write access to the TIM1\_DMAR register accesses the register located at the address: "(TIM1\_CR1 address) + DBA + (DMA index)" in which:

- TIM1\_CR1 is the address of control register 1
- DBA is the DMA base address configured in the TIM1\_DCR1 register
- DMA index is the offset automatically controlled by the DMA transfer

### 19.8.37 TIM1 register map and reset values

**Table 71. TIM1 register map**

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	TIM1_CCR1 Reset value	ARPE 0	CMS1 0	CMS0 0	DIR 0	OPM 0	URS 0	UDIS 0	CEN 0
0x01	TIM1_CCR2 Reset value	TI1S 0	MMS2 0	MMS1 0	MMS0 0	CCDS 0	COMS- 0	- 0	CCPC- 0
0x02	TIM1_SMCR Reset value	MSM 0	TS2 0	TS1 0	TS0 0	OCCS 0	SMS2 0	SMS1 0	SMS0 0
0x03	TIM1_ETR Reset value	ETP 0	ECE 0	ETPS1 0	ETPS0 0	EFT3 0	EFT2 0	EFT1 0	EFT0 0
0x04	TIM1_DER Reset value	- 0	- 0	COMDE 0	CC4DE 0	CC3DE 0	CC2DE 0	CC1DE 0	UDE 0
0x05	TIM1_IER Reset value	BIE 0	TIE 0	COMIE 0	CC4IE 0	CC3IE 0	CC2IE 0	CC1IE 0	UIE 0
0x06	TIM1_SR1 Reset value	BIF 0	TIF 0	COMIF 0	CC4IF 0	CC3IF 0	CC2IF 0	CC1IF 0	UIF 0
0x07	TIM1_SR2 Reset value	- 0	- 0	- 0	CC4OF 0	CC3OF 0	CC2OF 0	CC1OF 0	- 0
0x08	TIM1_EGR Reset value	BG 0	TG 0	COMG 0	CC4G 0	CC3G 0	CC2G 0	CC1G 0	UG 0
0x09	TIM1_CCMR1 (output mode) Reset value	CC1CE 0	OC1M2	OC1M1	OC1M0	OC1PE	OC1FE	CC1S1	CC1S0
	TIM1_CCMR1 (input mode) Reset value	IC1F3 0	IC1F2	IC1F1	IC1F0	IC1PSC1	IC1PSC0	CC1S1	CC1S0
0x0A	TIM1_CCMR2 (output mode) Reset value	CC2CE 0	OC2M2	OC2M1	OC2M0	OC2PE	OC2FE	CC2S1	CC2S0
	TIM1_CCMR2 (input mode) Reset value	IC2F3 0	IC2F2	IC2F1	IC2F0	IC2PSC1	IC2PSC0	CC2S1	CC2S0
0x0B	TIM1_CCMR3 (output mode) Reset value	CC3CE 0	OC3M2	OC3M1	OC3M0	OC3PE	OC3FE	CC3S1	CC3S0
	TIM1_CCMR3 (input mode) Reset value	IC3F3 0	IC3F2	IC3F1	IC3F0	IC3PSC1	IC3PSC0	CC3S1	CC3S0
0x0C	TIM1_CCMR4 (output mode) Reset value	OC4CE 0	OC4M2	OC4M1	OC4M0	OC4PE	-	CC4S1	CC4S0
	TIM1_CCMR4 (input mode) Reset value	IC4F3 0	IC4F2	IC4F1	IC4F0	IC4PSC1	IC4PSC0	CC4S1	CC4S0
0x0D	TIM1_CCER1 Reset value	CC2NP 0	CC2NE 0	CC2P 0	CC2E 0	CC1NP 0	CC1NE 0	CC1P 0	CC1E 0
0x0E	TIM1_CCER2 Reset value	- 0	- 0	CC4P 0	CC4E 0	CC3NP 0	CC3NE 0	CC3P 0	CC3E 0
0x0F	TIM1_CNTRH Reset value	CNT15 0	CNT14 0	CNT13 0	CNT12 0	CNT11 0	CNT10 0	CNT9 0	CNT8 0
0x10	TIM1_CNTRL Reset value	CNT7 0	CNT6 0	CNT5 0	CNT4 0	CNT3 0	CNT2 0	CNT1 0	CNT0 0
0x11	TIM1_PSCRH Reset value	PSC15 0	PSC14 0	PSC13 0	PSC12 0	PSC11 0	PSC10 0	PSC9 0	PSC8 0

**Table 71. TIM1 register map (continued)**

Address offset	Register name	7	6	5	4	3	2	1	0
0x12	TIM1_PSCRL Reset value	PSC7 0	PSC6 0	PSC5 0	PSC4 0	PSC3 0	PSC2 0	PSC1 0	PSC0 0
0x13	TIM1_ARRH Reset value	ARR15 1	ARR14 1	ARR13 1	ARR12 1	ARR11 1	ARR10 1	ARR9 1	ARR8 1
0x14	TIM1_ARRL Reset value	ARR7 1	ARR6 1	ARR5 1	ARR4 1	ARR3 1	ARR2 1	ARR1 1	ARR0 1
0x15	TIM1_RCR Reset value	REP7 0	REP6 0	REP5 0	REP4 0	REP3 0	REP2 0	REP1 0	REP0 0
0x16	TIM1_CCR1H Reset value	CCR115 0	CCR114 0	CCR113 0	CCR112 0	CCR111 0	CCR110 0	CCR19 0	CCR18 0
0x17	TIM1_CCR1L Reset value	CCR17 0	CCR16 0	CCR15 0	CCR14 0	CCR13 0	CCR12 0	CCR11 0	CCR10 0
0x18	TIM1_CCR2H Reset value	CCR215 0	CCR214 0	CCR213 0	CCR212 0	CCR211 0	CCR210 0	CCR29 0	CCR28 0
0x19	TIM1_CCR2L Reset value	CCR27 0	CCR26 0	CCR25 0	CCR24 0	CCR23 0	CCR22 0	CCR21 0	CCR20 0
0x1A	TIM1_CCR3H Reset value	CCR315 0	CCR314 0	CCR313 0	CCR312 0	CCR311 0	CCR310 0	CCR39 0	CCR38 0
0x1B	TIM1_CCR3L Reset value	CCR37 0	CCR36 0	CCR35 0	CCR34 0	CCR33 0	CCR32 0	CCR31 0	CCR30 0
0x1C	TIM1_CCR4H Reset value	CCR415 0	CCR414 0	CCR413 0	CCR412 0	CCR411 0	CCR410 0	CCR49 0	CCR48 0
0x1D	TIM1_CCR4L Reset value	CCR47 0	CCR46 0	CCR45 0	CCR44 0	CCR43 0	CCR42 0	CCR41 0	CCR40 0
0x1E	TIM1_BKR Reset value	MOE 0	AOE 0	BKP 0	BKE 0	OSSR 0	OSSI 0	LOCK 0	LOCK 0
0x1F	TIM1_DTR Reset value	DTG7 0	DTG6 0	DTG5 0	DTG4 0	DTG3 0	DTG2 0	DTG1 0	DTG0 0
0x20	TIM1_OISR Reset value	- 0	- 0	OIS3N 0	OIS3 0	OIS2N 0	OIS2 0	OIS1N 0	OIS1 0
0x21	TIM1_DCR1 Reset value	- 0	- 0	- 0	DBA4 0	DBA3 0	DBA2 0	DBA1 0	DBA0 0
0x22	TIM1_DCR2 Reset value	- 0	- 0	- 0	DBL4 0	DBL3 0	DBL2 0	DBL1 0	DBL0 0
0x23	TIM1_DMAR Reset value	DMAB7 0	DMAB6 0	DMAB5 0	DMAB4 0	DMAB3 0	DMAB2 0	DMAB1 0	DMAB0 0

## 20 16-bit general purpose timers (TIM2, TIM3, TIM5)

### 20.1 Introduction

This chapter describes TIM2, TIM3 and TIM5 which are identical timers.

Each timer consists of a 16-bit up-down auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including:

- Time base generation
- Measuring the pulse lengths of input signals (input capture)
- Generating output waveforms (output compare, PWM and One-pulse mode)
- Interrupt capability on various events (capture, compare, overflow)
- Synchronization with other timers or external signals (external clock, reset, trigger and enable)

The timer clock can be sourced from internal clocks or from an external source selectable through a configuration register.

Only the main features of the general purpose timers are given in this chapter. Refer to the corresponding paragraphs of [Section 19: 16-bit advanced control timer \(TIM1\) on page 284](#) for more details on each feature.

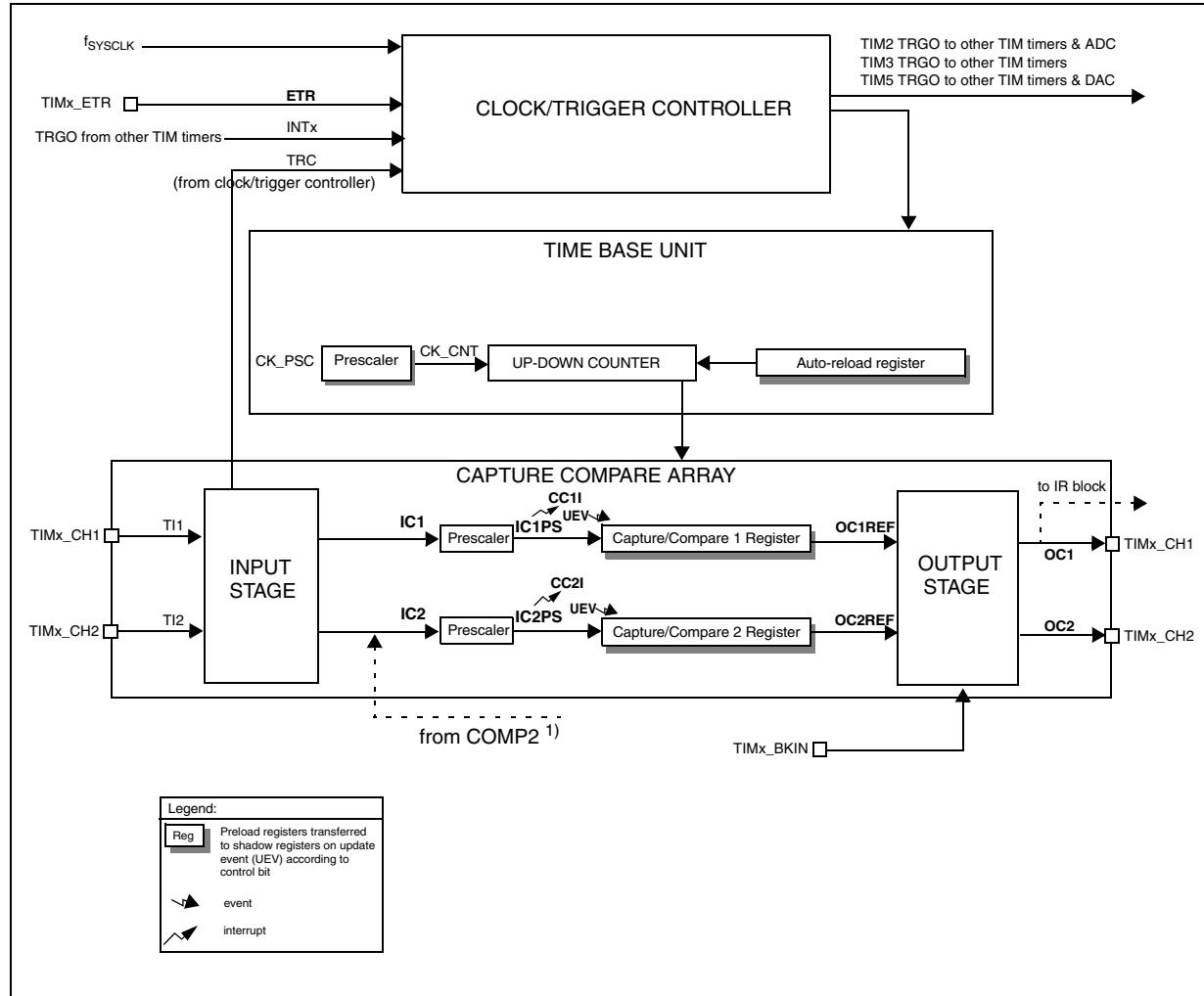
### 20.2 TIMx main features

General purpose TIMx TIM2/TIM3 features include:

- 16-bit up, down, up/downauto-reload counter.
- 3-bit programmable prescaler allowing the counter clock frequency to be divided “on the fly” by any power of 2 from 1 to 128.
- 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Break input to put the timer output signals in reset state or in a known state.
- Input capture 2 can be routed from COMP2 comparator
- Interrupt/ DMA request generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare
  - Break input
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)

## 20.3 TIMx functional description

Figure 120. TIMx block diagram



1. Available only for TIM2 and TIM3

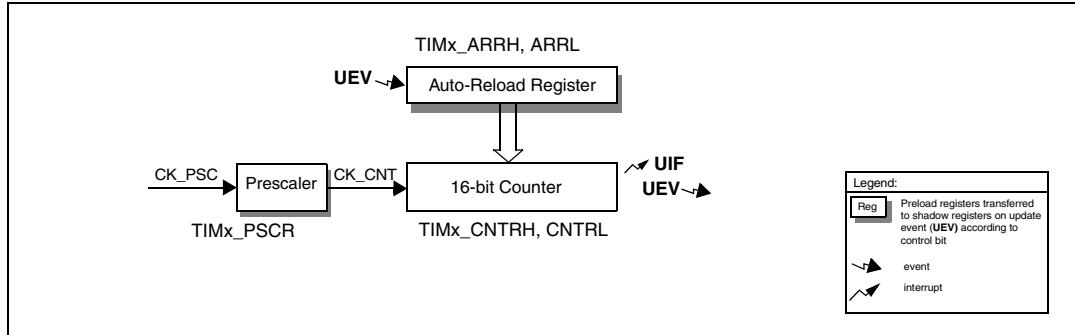
### 20.3.1 Time base unit

The timer has a time base unit that includes:

- 16-bit up-down counter
- 16-bit auto-reload register
- 3-bit programmable prescaler

There is no repetition counter.

The clock source for is the internal clock ( $f_{SYSCLK}$ ). It is connected directly to the CK\_PSC clock that feeds the prescaler driving the counter clock CK\_CNT.

**Figure 121.** Time base unit

For more details refer to [Section 19.3: TIM1 time base unit on page 287](#).

### Prescaler

The prescaler implementation is as follows:

- The prescaler is based on a 7-bit counter controlled through a 3-bit register (in TIMx\_PSCR register). It can be changed on the fly as this control register is buffered. It can divide the counter clock frequency by 1, 2, 4, 8, 16, 32, 64 or 128.

The counter clock frequency is calculated as follows:

$$f_{CK\_CNT} = f_{CK\_PSC}/2^{(PSCR[2:0])}$$

### Counter operation

Refer to [Section 19.3.4: Up-counting mode on page 289](#), [Section 19.3.5: Down-counting mode on page 291](#) and [Section 19.3.6: Center-aligned mode \(up/down counting\) on page 293](#).

## 20.3.2 Clock/trigger controller

Refer to [Section 19.4: TIM1 clock/trigger controller on page 297](#).

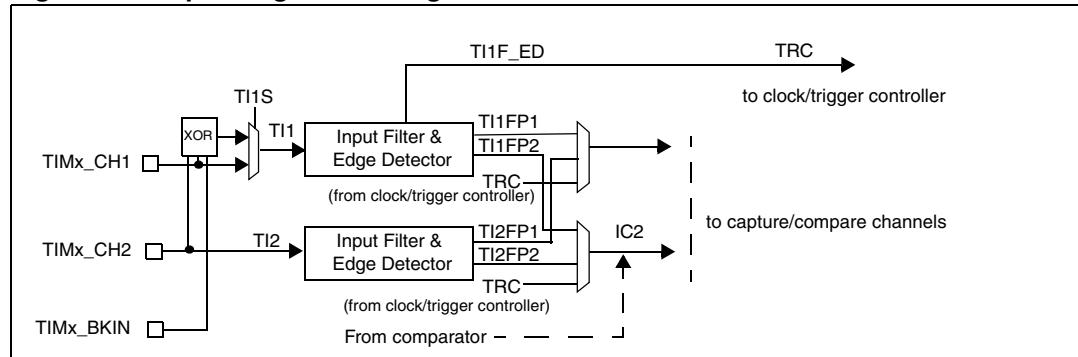
### 20.3.3 Capture/compare channels

#### Input stage

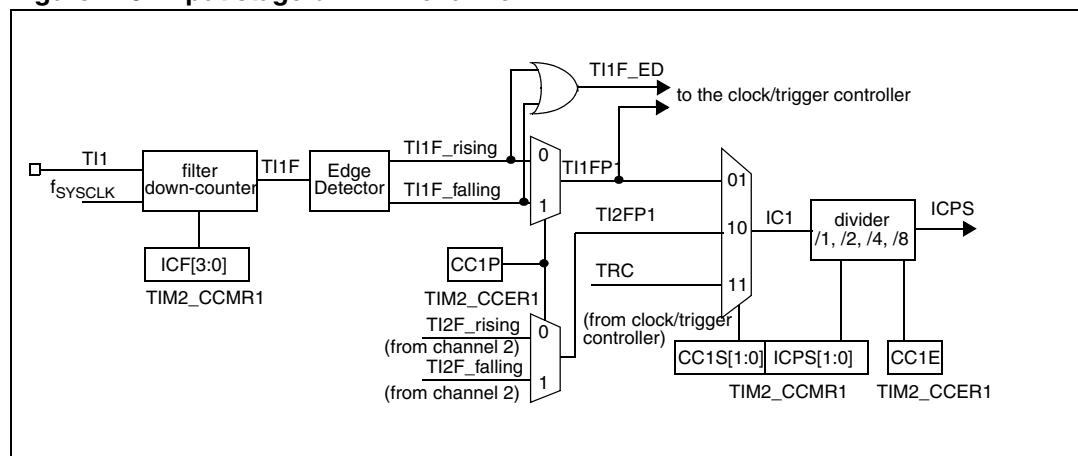
Refer to [Section 19.5: TIM1 capture/compare channels on page 311](#).

There are two input channels, as shown in [Figure 122: Input stage block diagram](#). Channel 2 is internally connected to the comparator.

**Figure 122. Input stage block diagram**



**Figure 123. Input stage of TIM 2 channel 1**

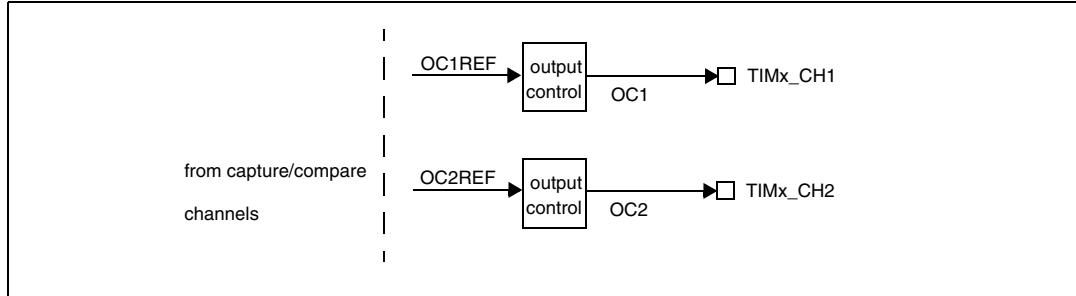


## Output stage

Refer to [Section 19.5.4: Output stage on page 316](#), [Section 19.5.5: Forced output mode on page 317](#), [Section 19.5.7: PWM mode on page 319](#).

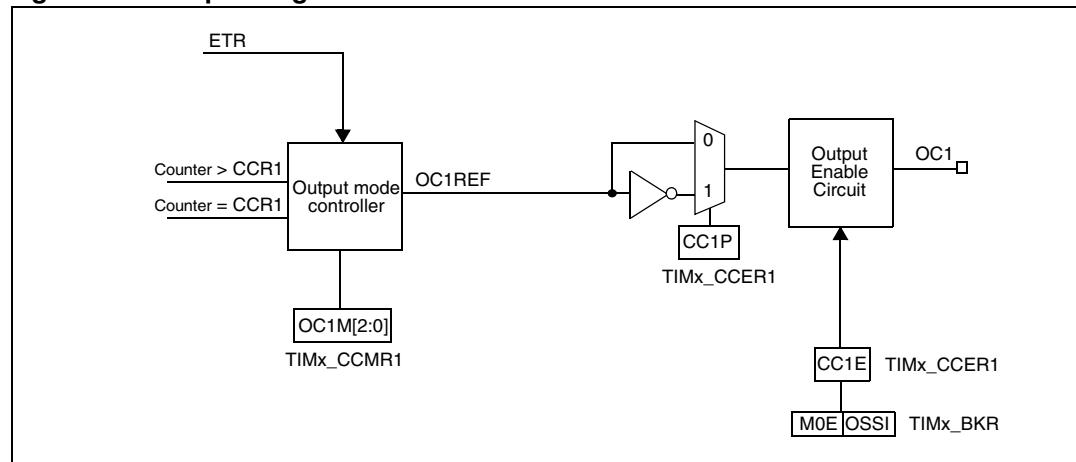
As shown in [Figure 124](#). TIMx outputs have no deadtime or complementary outputs.

**Figure 124. Output stage**



The output stage generates an intermediate waveform which is then used for reference: OC<sub>x</sub>REF (active high). Break functions and polarity act at the end of the chain (see [Figure 125](#)).

**Figure 125. Output stage of channel 1**



## Break

When using the break function, the output enable signals and inactive levels are modified according to additional control bits (MOE and OSS1 bits in the TIMx\_BKR register).

When exiting from reset, the break circuit is disabled and the MOE bit is low. You can enable the break function by setting the BKE bit in the TIMx\_BKR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time.

For more details, refer to [“Using the break function on page 326](#). See also [Table 73: Output control bit for OC<sub>i</sub> channels with break feature \(break feature implemented, complementary output not implemented\) on page 393](#).

#### 20.3.4 Timer input XOR function

The TI1S bit in the TIMx\_CR2 register allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins TIMx\_CH1, TIMx\_CH2 and TIMx\_BKIN.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in [Section 19.5.12: Interfacing with Hall sensors on page 332](#).

### 20.4 TIMx interrupts

The timers have 5 interrupt request sources:

- Capture/compare 2 interrupt
- Capture/compare 1 interrupt
- Update interrupt
- Break input
- Trigger interrupt

To use the interrupt features, for each interrupt channel used, set the desired CC2IE and/or CC1IE bits in the TIMx\_IER register to enable interrupt requests.

The different interrupt sources can be also generated by software using the corresponding bits in the TIMx\_EGR register.

## 20.5 TIMx registers

### 20.5.1 Control register 1 (TIMx\_CR1)

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
ARPE	CMS[1:0]	DIR	OPM	URS	UDIS	CEN	
rw	rw	rw	rw	rw	rw	rw	

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx\_ARR register is not buffered through a preload register. It can be written directly

1: TIMx\_ARR register is buffered through a preload register

Bits 6:5 **CMS(1:0)**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternately. Output compare interrupt flags of channels configured in output (CCIS = 00 in TIMx\_CCMR*i* registers) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternately. Output compare interrupt flags of channels configured in output (CCIS = 00 in TIMx\_CCMR*i* registers) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternately. Output compare interrupt flags of channels configured in output (CCIS = 00 in TIMx\_CCMR*i* registers) are set both when the counter is counting up or down.

*Note:* It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1).

*Encoder mode (SMS = 001, 010, or 011 in GPT\_SMCR register) must be disabled in center-aligned mode.*

Bit 4 **DIR**: Direction

0: Counter used as up-counter

1: Counter used as down-counter

*Note:* This bit is read only when the timer is configured in center-aligned mode or encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the CEN bit)

Bit 2 **URS**: Update request source

0: When enabled by the UDIS bit, the UIF bit is set and an update interrupt request is sent when one of the following events occurs:

- Registers are updated (counter overflow/underflow)
- UG bit is set by software
- Update event is generated through the clock/trigger controller

1: When enabled by the UDIS bit, the UIF bit is set and an update interrupt request is sent only when registers are updated (counter overflow/underflow).

Bit 1 **UDIS**: Update disable

0: A UEV is generated as soon as a counter overflow occurs or a software update is generated or an hardware reset is generated by the clock/trigger mode controller. Buffered registers are then loaded with their preload values.

1: A UEV is not generated, shadow registers keep their value (ARR, PSC, CCR*i*). The counter and the prescaler are re-initialized if the UG bit is set.

Bit 0 **CEN**: Counter enable  
 0: Counter disabled  
 1: Counter enabled

## 20.5.2 Control register 2 (TIMx\_CR2)

Address offset: 0x01

Reset value: 0x00

	7	6	5	4	3	2	1	0
TI1S	MMS[2:0]			CCDS		Reserved		
	rw	rw	rw	rw				

Bit 7 **TI1S**: TI1 (digital filter input) selection  
 0: TI1 is connected to CH1 input pin  
 1: TI1 is connected to the 3 inputs CH1, CH2 and break (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits select the information to be sent in master mode to the slave timers for synchronization (TRGO). The combination is as follows:

000: Reset - the UG bit from the TIMx\_EGR register is used as a trigger output (TRGO). If the reset is generated by the trigger input (clock/trigger mode controller configured in trigger reset mode), the signal on TRGO is delayed compared to the actual reset.

001: Enable - the counter enable signal is used as a trigger output (TRGO). It is used to start several timers at the same time or to control a window in which a slave timer is enabled. The counter enable signal is generated by a logic OR between the CEN control bit and the trigger input when configured in gated mode. When the counter enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx\_SMCR register).

010: Update - The update event is selected as a trigger output (TRGO)

011: Reserved

100: Reserved

101: Reserved

111: Reserved

Bit 3 **CCDS**: Capture/compare DMA selection

0: TIMx\_CC*i* DMA request sent when a CC*i* (capture/compare) event occurs  
 1: TIMx\_CC*i* DMA request sent when an update event occurs

Bits 2:0 Reserved, must be kept cleared

### 20.5.3 Slave mode control register (TIMx\_SMCR)

Address offset: 0x02

Reset value: 0x00

7	6	5	4	3	2	1	0
MSM	TS[2:0]			Reserved	SMS[2:0]		
rw	rw	rw	rw		rw	rw	rw

Bit 7 **MSM** Master/slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between timers (through TRGO).

Bits 6:4 **TS[2:0]** Trigger selection<sup>(1)</sup>

This bitfield selects the trigger input to be used to synchronize the counter.

000: Internal trigger ITR0

001: Internal trigger ITR1

010: Internal trigger ITR2

011: Internal trigger ITR3

100: TI1 edge detector (TI1F\_ED)

101: Filtered timer input 1 (TI1FP1)

110: Filtered timer input 2 (TI2FP2)

111: External trigger input (ETRF) (from TIM1\_ETR pin). Signal filtering and polarity can be controlled by the TIM5\_CCMRi and TIM5\_CCERi registers.

*Note:* These bits must only be changed when they are not used (e.g. when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved.

Bits 2:0 **SMS[2:0]** Clock/trigger/slave mode selection

When external signals are selected, the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description).

000: Clock/trigger controller disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

001, 010 and 011: Reserved

100: Trigger reset mode - Rising edge of the selected trigger signal (TRGI) reinitializes the counter and generates an update of the registers.

101: Gated mode - The counter clock is enabled when the trigger signal (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both the start and stop of the counter are controlled.

110: Trigger mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

111: External clock mode 1 - Rising edges of the selected trigger (TRGI) clock the counter

- Refer to [Table 72: TIMx internal trigger connection on page 377](#) for more details on the ITRx meaning for each timer.

**Table 72. TIMx internal trigger connection**

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM2	TIM4	TIM1	TIM3	
TIM3	TIM4	TIM1		TIM2
	TIM4	TIM1	TIM3	TIM2

*Note:* When a timer is not present in the product, the corresponding trigger (ITRx) is not available.

## 20.5.4 External trigger register (TIMx\_ETR)

Address offset: 0x03

Reset value: 0x00

7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **ETP**: External trigger polarity

This bit selects whether ETR or  $\overline{ETR}$  is used for trigger operations.

0: ETR is non-inverted, active at high level or rising edge.

1: ETR is inverted, active at low level or falling edge.

Bit 6 **ECE**: External clock enable

This bit enables external clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

*Note: Setting the ECE bit has the same effect as selecting the external clock mode 1 with TRGI connected to ETRF (SMS = 111 and TS = 111 in the TIM1\_SMCR register).*

*It is possible to use the external clock mode 2 simultaneously with the following modes: Trigger standard mode, trigger reset mode and trigger gated mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111 in TIMx\_SMCR register).*

*If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input will be ETRF.*

Bits 5:4 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at 1/4 of  $f_{SYSCLK}$  frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler off

01: ETRP frequency divided by 2

10: ETRP frequency divided by 4

11: ETRP frequency divided by 8

Bits 3:0 **ETF[3:0]**: External trigger filter

This bitfield defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at  $f_{SYSCLK}$

1000:  $f_{SAMPLING}=f_{SYSCLK}/8$ , N = 6

0001:  $f_{SAMPLING}=f_{SYSCLK}$ , N = 2

1001:  $f_{SAMPLING}=f_{SYSCLK}/8$ , N = 8

0010:  $f_{SAMPLING}=f_{SYSCLK}$ , N = 4

1010:  $f_{SAMPLING}=f_{SYSCLK}/16$ , N = 5

0011:  $f_{SAMPLING}=f_{SYSCLK}$ , N = 8

1011:  $f_{SAMPLING}=f_{SYSCLK}/16$ , N = 6

0100:  $f_{SAMPLING}=f_{SYSCLK}/2$ , N = 6

1100:  $f_{SAMPLING}=f_{SYSCLK}/16$ , N = 8

0101:  $f_{SAMPLING}=f_{SYSCLK}/2$ , N = 8

1101:  $f_{SAMPLING}=f_{SYSCLK}/32$ , N = 5

0110:  $f_{SAMPLING}=f_{SYSCLK}/4$ , N = 6

1110:  $f_{SAMPLING}=f_{SYSCLK}/32$ , N = 6

0111:  $f_{SAMPLING}=f_{SYSCLK}/4$ , N = 8

1111:  $f_{SAMPLING}=f_{SYSCLK}/32$ , N = 8

### 20.5.5 DMA request enable register (TIMx\_DER)

Address offset: 0x04

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved				CC2DE	CC1DE	UDE	
				rw	rw	rw	

Bits 7:3 Reserved.

Bit 2 **CC2DE**: Capture/compare 2 DMA request enable<sup>(1)</sup>

- 0: CC2 DMA request disabled
- 1: CC2 DMA request enabled

Bit 1 **CC1DE**: Capture/compare 1 DMA request enable<sup>(1)</sup>

- 0: CC1 DMA request disabled
- 1: CC1 DMA request enabled

Bit 0 **UDE**: Update DMA request enable<sup>(1)</sup>

- 0: Update DMA request disabled
- 1: Update DMA request enabled

*Note: The conditions for generating a DMA request on the update event are the same as for setting the UIF bit (in the TIMx\_SR1 register). In particular, the DMA request depends on the URS bit (in the TIMx\_CR1 register).*

1. The DMA request is related to the actual event and not to the status bit (in the TIMx\_SR1 register). Thus no DMA request is sent if the xxDE bit is written to 1 while the corresponding status bit was already set.

## 20.5.6 Interrupt enable register (TIMx\_IER)

Address offset: 0x05

Reset value: 0x00

7	6	5	4	3	2	1	0
BIE	TIE	Reserved			CC2IE	CC1IE	UIE
rw	rw				rw	rw	rw

Bit 7 **BIE**: Break interrupt enable

- 0: Break interrupt disabled
- 1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable

- 0: Trigger interrupt disabled
- 1: Trigger interrupt enabled

Bits 5:3 Reserved, must be kept cleared

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable

- 0: CC2 interrupt disabled
- 1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable

- 0: CC1 interrupt disabled
- 1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

- 0: Update interrupt disabled
- 1: Update interrupt enabled

### 20.5.7 Status register 1 (TIMx\_SR1)

Address offset: 0x06

Reset value: 0x00

7	6	5	4	3	2	1	0
BIF	TIF	Reserved			CC2IF	CC1IF	UIF
rc_w0	rc_w0				rc_w0	rc_w0	rc_w0

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on a trigger event (active edge is detected on TRGI signal and both edges are detected when gated mode is selected). It is cleared by software.

0: No trigger event has occurred

1: Trigger interrupt pending

Bits 5:3 Reserved, must be kept cleared

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag

Refer to CC1IF description.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

**If channel CC1 is configured as output:**

This flag is set by hardware when the counter matches the compare value. It is cleared by software.

0: No match

1: The content of the counter TIMx\_CNT has matched the content of the TIMx\_CCR1 register

**If channel CC1 is configured as input:**

This bit is set by hardware on a capture. It is cleared by software or by reading the TIMx\_CCR1L register.

0: No input capture has occurred

1: The counter value has been captured in TIMx\_CCR1 register (an edge has been detected on IC1 which matches the selected polarity).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update has occurred

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow if UDIS = 0 in the TIMx\_CR1 register
- When CNT is re-initialized by software using the UG bit in TIMx\_EGR register, if URS = 0 and UDIS = 0 in the TIMx\_CR1 register.

### 20.5.8 Status register 2 (TIMx\_SR2)

Address offset: 0x07

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved				CC2OF	CC1OF	Reserved	
rc_w0				rc_w0	Reserved		

Bits 7:3 Reserved

Bit 2 **CC2OF**: Capture/compare 2 overcapture flag

Refer to CC1OF description

Bit 1 **CC1OF**: Capture/compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected

1: The counter value has been captured in TIMx\_CCR1 register while CC1IF flag was already set

Bit 0 Reserved, forced by hardware to 0

### 20.5.9 Event generation register (TIMx\_EGR)

Address offset: 0x08

Reset value: 0x00

7	6	5	4	3	2	1	0
BG	TG	Reserved			CC2G	CC1G	UG
w	w				w	w	w

#### Bit 7 **BG**: Break generation

This bit is set by software to generate an event. It is automatically cleared by hardware.

0: No action

1: A break event is generated. The MOE bit is cleared and the BIF flag is set. An interrupt is generated if enabled by the BIE bit.

#### Bit 6 **TG**: Trigger generation

This bit is set by software to generate an event. It is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx\_SR1 register. An interrupt is generated if enabled by the TIE bit

Bits 5:3 Reserved

#### Bit 2 **CC2G**: Capture/compare 2 generation

Refer to CC1G description

#### Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software to generate an even. It is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

- If the CC1 channel is configured in output mode. In this case, the CC1IF flag is set, and the corresponding interrupt request is sent if enabled.
- If the CC1 channel configured in input mode. In this case, the current value of the counter is captured in the TIMx\_CCR1 register. The CC1IF flag is set, and the corresponding interrupt request is sent if enabled. The CC1OF flag is set if the CC1IF flag is already high.

#### Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Re-initializes the counter and generates an update of the registers. Note that the prescaler counter is also cleared.

### 20.5.10 Capture/compare mode register 1 (TIMx\_CCMR1)

The channel can be used in input (capture mode) or in output (compare mode). The direction of the channel is defined by configuring the CC1S bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OC*i* describes its function when the channel is configured in output and IC*i* describes its function when the channel is configured in input. Therefore, be aware that the same bit can have a different meaning for the input stage and for the output stage.

Address offset: 0x09

Reset value: 0x00

#### Channel configured in output

	7	6	5	4	3	2	1	0
	rw	rw	rw	rw			rw	rw
Reserved	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]		
	rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 Reserved

Bits 6:4 **OC1M[2:0]**: Output compare 1 mode

These bits defines the behavior of the output reference signal OC1REF from which OC1 is derived. OC1REF is active high whereas OC1 active level depends on the CC1P bit.

000: Frozen - The comparison between the output compare register TIMx\_CCR1 and the counter TIMx\_CNT has no effect on the outputs

001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx\_CNT matches the capture/compare register 1 (TIMx\_CCR1).

011: Toggle - OC1REF toggles when TIMx\_CNT=TIMx\_CCR1

100: Force inactive level - OC1REF is forced low

101: Force active level - OC1REF is forced high

110: PWM mode 1 - In up-counting, channel 1 is active as long as TIMx\_CNT< TIMx\_CCR1.

Otherwise, channel 1 is inactive. In down-counting, channel 1 is inactive (OC1REF = 0) as long as TIMx\_CNT> TIMx\_CCR1. Otherwise, channel 1 is active (OC1REF = 1).

111: PWM mode 2 - In up-counting, channel 1 is inactive as long as TIMx\_CNT< TIMx\_CCR1. Otherwise, channel 1 is active.

*Note:* In PWM mode 1 or 2, the OC*i*REF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode. Refer to [Section 19.5.7 on page 319](#) for more details.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx\_CCR1 disabled. TIMx\_CCR1 can be written at anytime. The new value is taken into account immediately.

1: Preload register on TIMx\_CCR1 enabled. Read/write operations access the preload register. TIMx\_CCR1 preload value is loaded in the shadow register at each update event.

*Note:* For correct operation, preload registers must be enabled when the timer is in PWM mode. This is not mandatory in one-pulse mode (OPM bit set in TIMx\_CR1 register).

**Bit 2 OC1FE:** Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger input on the CC output.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is on. The minimum delay to activate CC1 output when an edge occurs on the trigger input, is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. The delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

**Bits 1:0 CC1S[1:0]:** Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1FP1

10: CC1 channel is configured as input, IC1 is mapped on TI2FP1

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIM5\_SMCR register).

*Note: CC1S bits are writable only when the channel is off (CC1E = 0 and is updated in TIMx\_CCER1).*

### Channel configured in input

7	6	5	4	3	2	1	0
IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw

**Bits 7:4 IC1F[3:0]:** Input capture 1 filter

This bitfield defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, sampling is done at  $f_{SYSCLK}$

0001:  $f_{SAMPLING} = f_{SYSCLK}$ , N = 2

0010:  $f_{SAMPLING} = f_{SYSCLK}$ , N = 4

0011:  $f_{SAMPLING} = f_{SYSCLK}$ , N = 8

0100:  $f_{SAMPLING} = f_{SYSCLK}/2$ , N = 6

0101:  $f_{SAMPLING} = f_{SYSCLK}/2$ , N = 8

0110:  $f_{SAMPLING} = f_{SYSCLK}/4$ , N = 6

0111:  $f_{SAMPLING} = f_{SYSCLK}/4$ , N = 8

1000:  $f_{SAMPLING} = f_{SYSCLK}/8$ , N = 6

1001:  $f_{SAMPLING} = f_{SYSCLK}/8$ , N = 8

1010:  $f_{SAMPLING} = f_{SYSCLK}/16$ , N = 5

1011:  $f_{SAMPLING} = f_{SYSCLK}/16$ , N = 6

1100:  $f_{SAMPLING} = f_{SYSCLK}/16$ , N = 8

1101:  $f_{SAMPLING} = f_{SYSCLK}/32$ , N = 5

1110:  $f_{SAMPLING} = f_{SYSCLK}/32$ , N = 6

1111:  $f_{SAMPLING} = f_{SYSCLK}/32$ , N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E = 0 (TIMx\_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

*Note: The internal event counter is not reset when IC1PSC is changed on the fly. In this case the old value is used until the next capture occurs. To force a new value to be taken in account immediately, the CC1E bit can be cleared and set again.*

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1FP1

10: CC1 channel is configured as input, IC1 is mapped on TI2FP1

11: Reserved

*Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx\_CCER1 and updated).*

**20.5.11 Capture/compare mode register 2 (TIMx\_CCMR2)**

**Note:** Refer to [Capture/compare mode register 1 \(TIMx\\_CCMR1\) on page 384](#) for details on using these bits.

Address offset: 0x0A

Reset value: 0x00

**Channel configured in output**

	7	6	5	4	3	2	1	0
Reserved	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		
	rw	rw	rw	rw	rw	rw		

Bit 7 Reserved

Bits 6:4 **OC2M[2:0]**: Output compare 2 mode

Bit 3 **OC2PE**: Output compare 2 preload enable

Bit 2 **OC2FE**: Output compare 2 fast enable

Bits 1:0 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2FP2

10: CC2 channel is configured as input, IC2 is mapped on TI1FP2

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIM5\_SMCR register).

*Note: CC2S bits are writable only when the channel is off (CC2E = 0 in TIMx\_CCER1).*

### Channel configured in input

7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:4 **IC2F[3:0]**: Input capture 2 filter

Bits 3:2 **IC2PCS[1:0]**: Input capture 2 prescaler

Bits 1:0 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2FP2

10: CC2 channel is configured as input, IC2 is mapped on TI1FP2

11: Reserved

*Note: CC2S bits are writable only when the channel is off (CC2E = 0 in TIMx\_CCER1).*

### 20.5.12 Capture/compare enable register 1 (TIMx\_CCER1)

Address offset: 0x0B

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved		CC2P	CC2E	Reserved		CC1P	CC1E
		rw	rw			rw	rw

Bits 6:7 Reserved

Bit 5 **CC2P**: Capture/compare 2 output polarity

Refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable

Refer to CC1E description.

Bits 2:3 Reserved

Bit 1 **CC1P**: Capture/compare 1 output polarity

**CC1 channel configured as output:**

0: OC1 active high

1: OC1 active low

**CC1 channel configured as input for capture function (see Figure 100):**

0: Capture is done on a rising edge of TI1F or TI2F

1: Capture is done on a falling edge of TI1F or TI2F

Bit 0 **CC1E**: Capture/Compare 1 output Enable.

**CC1 channel configured as output:**

0: Off - OC1 is not active.

1: On - OC1 signal is output on the corresponding output pin.

**CC1 channel configured as input:**

In this case, this bit determines if a capture of the counter value can be made in the input capture/compare register 1 (TIMx\_CCR1) or not.

0: Capture disabled

1: Capture enabled

### 20.5.13 Counter high (TIMx\_CNTRH)

Address offset: 0x0C

Reset value: 0x00

7	6	5	4	3	2	1	0
CNT[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CNT[15:8]**: Counter value (MSB)

### 20.5.14 Counter low (TIMx\_CNTRL)

Address offset: 0x0D

Reset value: 0x00

7	6	5	4	3	2	1	0
CNT[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CNT[7:0]**: Counter value (LSB)

### 20.5.15 Prescaler register (TIMx\_PSCR)

Address offset: 0x0E

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved						PSC[2:0]	
						rw	rw

Bits 7:3 Reserved

Bits 2:0 **PSC[2:0]**: Prescaler value

The prescaler value divides the CK\_PSC clock frequency.

The counter clock frequency  $f_{CK\_CNT}$  is equal to  $f_{CK\_PSC} / 2^{(PSC[2:0])}$ . PSC[7:3] are forced to 0 by hardware.

PSCR contains the value which is loaded in the active prescaler register at each update event (including when the counter is cleared through the UG bit of the TIMx\_EGR register).

This means that a UEV must be generated so that a new prescaler value can be taken into account.

### 20.5.16 Auto-reload register high (TIMx\_ARRH)

Address offset: 0x0F

Reset value: 0xFF

7	6	5	4	3	2	1	0
ARR[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **ARR[15:8]**: Auto-reload value (MSB)

ARR is the value to be loaded in the actual auto-reload register.

Refer to the [Section 19.3: TIM1 time base unit on page 287](#) for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is 0.

### 20.5.17 Auto-reload register low (TIMx\_ARRL)

Address offset: 0x10

Reset value: 0xFF

7	6	5	4	3	2	1	0
ARR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **ARR[7:0]**: Auto-reload value (LSB)

### 20.5.18 Capture/compare register 1 high (TIMx\_CCR1H)

Address offset: 0x11

Reset value: 0x00

7	6	5	4	3	2	1	0
CCR1[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CCR1[15:8]**: Capture/compare 1 value (MSB)

**If the CC1 channel is configured as output (CC1S bits in TIMx\_CCMR1 register):**

The value of CCR1 is loaded permanently into the actual capture/compare 1 register if the preload feature is not enabled (OC1PE bit in TIMx\_CCMR1). Otherwise, the preload value is copied in the active capture/compare 1 register when a UEV occurs. The active capture/compare register contains the value which is compared to the counter register, TIMx\_CNT, and signalled on the OC1 output.

**If the CC1 channel is configured as input (CC1S bits in TIMx\_CCMR1 register):**

The value of CCR1 is the counter value transferred by the last input capture 1 event (IC1). In this case, these bits are read only.

### 20.5.19 Capture/compare register 1 low (TIMx\_CCR1L)

Address offset: 0x12

Reset value: 0x00

7	6	5	4	3	2	1	0
CCR1[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CCR1[7:0]**: Capture/compare 1 value (LSB)

### 20.5.20 Capture/compare register 2 high (TIMx\_CCR2H)

Address offset: 0x13

Reset value: 0x00

7	6	5	4	3	2	1	0
CCR2[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CCR2[15:8]**: Capture/compare 2 value (MSB)

**If the CC2 channel is configured as output (CC2S bits in TIMx\_CCMR2 register):**

The value of CCR2 is loaded permanently into the actual capture/compare 2 register if the preload feature is not enabled (OC2PE bit in TIMx\_CCMR2). Otherwise, the preload value is copied in the active capture/compare 2 register when a UEV occurs. The active capture/compare register contains the value which is compared to the counter register, TIMx\_CNT, and signalled on the OC2 output.

**If the CC2 channel is configured as input (CC2S bits in TIMx\_CCMR2 register):**

The value of CCR2 is the counter value transferred by the last input capture 2 event (IC2).

### 20.5.21 Capture/compare register 2 low (TIMx\_CCR2L)

Address offset: 0x14

Reset value: 0x00

7	6	5	4	3	2	1	0
CCR2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CCR2[7:0]**: Capture/compare value (LSB)

## 20.5.22 Break register (TIMx\_BKR)

Address offset: 0x15

Reset value: 0x00

7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	Reserved	OSSI	LOCK	
rw	rw	rw	rw		rw	rw	rw

### Bit 7 MOE: Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It acts only on the channels which are configured in output.

0: OC outputs are disabled or forced to idle state

1: OC outputs are enabled if their respective enable bits are set (CCxE in TIMx\_CCER*i* registers)

See OC enable description for more details ([Section 19.8.14 on page 353](#)).

### Bit 6 AOE: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next UEV (if the break input is not active).

*Note:* This bit can no longer be modified as long as LOCK level 1 has been programmed (LOCK bits in the TIMx\_BKR register).

### Bit 5 BKP: Break polarity

0: Break input BKIN is active low

1: Break input BKIN is active high

*Note:* This bit can no longer be modified as long as LOCK level 1 has been programmed (LOCK bits in the TIMx\_BKR register).

### Bit 4 BKE: Break enable

0: Break input (BKIN) disabled

1: Break input (BKIN) enabled

*Note:* This bit can no longer be modified as long as LOCK level 1 has been programmed (LOCK bits in the TIMx\_BKR register).

### Bit 3 Reserved, must be kept cleared

### Bit 2 OSS1: Off state selection for idle mode

This bit is used when MOE = 0 on channels configured as outputs. See OC enable description for more details ([Section 19.8.14 on page 353](#)).

0: When inactive, OC*i* outputs are disabled (OC*i* enable output signal = 0).

1: When inactive, OC*i* outputs are forced first with their idle level as soon as CC*i* E = 1. OC enable output signal = 1

*Note:* This bit can no longer be modified as soon as the LOCK level 2 has been programmed (LOCK bits in the TIMx\_BKR register).

Bits 1:0 **LOCK[1:0]**: Lock configuration.

These bits offer a write protection against software errors.

00: LOCK off - No bits are write protected

01: LOCK level 1 = OIS<sub>i</sub> bit in TIMx\_OISR register and BKE/BKP/AOE bits in TIMx\_BKR register can no longer be written.

10: LOCK level 2 = LOCK level 1 + CC polarity bits (CC*i*P bits in TIMx\_CCER*i* registers, as long as the related channel is configured in output through the CC/S bits) as well as the OSSR and OSSI bits can no longer be written.

11: LOCK level 3 = LOCK level 2 + CC control bits (OC*i*M and OC*i*PE bits in TIMx\_CCMR*i* registers, as long as the related channel is configured in output through the CC/S bits) can no longer be written.

*Note: The LOCK bits can be written only once after reset. Once the TIMx\_BKR register has been written, their content is frozen until the next reset.*

*Note:* As the bits AOE, BKP, BKE and OSSI can be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx\_BKR register.

**Table 73. Output control bit for OC*i* channels with break feature (break feature implemented, complementary output not implemented)**

Control bits			OC <i>i</i> /OC <i>i</i> _EN output state
MOE bit	OSSI bit	CC <i>i</i> E bit	
1	x	0	Output disabled (not driven by the timer) OC <i>i</i> = CC <i>i</i> P, OC <i>i</i> _EN = 0
		1	OC <i>i</i> = OC <i>i</i> REF + polarity (OC <i>i</i> REF xor CC <i>i</i> P) OC <i>i</i> _EN = 1
0	0	0	Output disabled (not driven by the timer) OC <i>i</i> = OIS <sub>i</sub> , OC <i>i</i> _EN = 0
	0	1	
	1	0	
	1	1	Off State (output enabled with inactive state) OC <i>i</i> = OIS <sub>i</sub> , OCx_EN = 1

*Note:* The state of the external I/O pins connected to the OCx channels depends on the OCx channel state and the GPIO registers.

### 20.5.23 Output idle state register (TIMx\_OISR)

Address offset: 0x16

Reset value: 0x00

7	6	5	4	3	2	1	0
		Reserved		OIS2 rw	Reserved	OIS1 rw	

Bits 7:3 Reserved, must be kept cleared

Bit 2 **OIS2**: Output idle state 2 (OC2 output)

Refer to OIS1 bit

Bit 1 Reserved, must be kept cleared

Bit 0 **OIS1**: Output idle state 1 (OC1 output)

0: OC1 = 0 when MOE = 0

1: OC1 = 1 when MOE = 0

*Note:* This bit can no longer be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIMx\_BKR register).

### 20.5.24 TIMx register map and reset values

Table 74. TIMx register map

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	TIMx_CR1 Reset value	ARPE 0	CMS1 0	CMS0 0	DIR 0	OPM 0	URS 0	UDIS 0	CEN 0
0x01	TIMx_CR2 Reset value	TI1S 0	MMS2 0	MMS1 0	MMS0 0	CCDS 0	- 0	- 0	- 0
0x02	TIMx_SMCR Reset value	MSM 0	TS2 0	TS1 0	TS0 0	- 0	SMS2 0	SMS1 0	SMS0 0
0x03	TIMx_ETR Reset value	ETP 0	ECE 0	ETPS1 0	ETPS0 0	EFT3 0	EFT2 0	EFT1 0	EFT0 0
0x04	TIMx_DER Reset value	- 0	- 0	- 0	- 0	- 0	CC2DE 0	CC1DE 0	UDE 0
0x05	TIMx_IER Reset value	BIE 0	TIE 0	- 0	- 0	- 0	CC2IE 0	CC1IE 0	UIE 0
0x06	TIMx_SR1 Reset value	BIF 0	TIF 0	- 0	- 0	- 0	CC2IF 0	CC1IF 0	UIF 0
0x07	TIMx_SR2 Reset value	- 0	- 0	- 0	- 0	- 0	CC2OF 0	CC1OF 0	- 0
0x08	TIMx_EGR Reset value	BG 0	TG 0	- 0	- 0	- 0	CC2G 0	CC1G 0	UG 0
0x09	TIMx_CCMR1 (output mode) Reset value	- 0	OC1M2 0	OC1M1 0	OC1M0 0	OC1PE 0	OC1FE 0	CC1S1 0	CC1S0 0
	TIMx_CCMR1 (input mode) Reset value	IC1F3 0	IC1F2 0	IC1F1 0	IC1F0 0	IC1PSC1 0	IC1PSC0 0	CC1S1 0	CC1S0 0

**Table 74.** TIMx register map (continued)

<b>Address offset</b>	<b>Register name</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
0x0A	TIMx_CCMR2 (output mode) Reset value	- 0	OC2M2 0	OC2M1 0	OC2M0 0	OC2PE 0	OC2FE 0	CC2S1 0	CC2S0 0
	TIMx_CCMR2 (input mode) Reset value	IC2F3 0	IC2F2 0	IC2F1 0	IC2F0 0	IC2PSC1 0	IC2PSC0 0	CC2S1 0	CC2S0 0
0x0B	TIMx_CCER1 Reset value	- 0	- 0	CC2P 0	CC2E 0	- 0	- 0	CC1P 0	CC1E 0
0x0C	TIMx_CNTRH Reset value	CNT15 0	CNT14 0	CNT13 0	CNT12 0	CNT11 0	CNT10 0	CNT9 0	CNT8 0
0x0D	TIMx_CNTRL Reset value	CNT7 0	CNT6 0	CNT5 0	CNT4 0	CNT3 0	CNT2 0	CNT1 0	CNT0 0
0x0E	TIMx_PSCR Reset value	- 0	- 0	- 0	- 0	- 0	PSC2 0	PSC1 0	PSC0 0
0x0F	TIMx_ARRH Reset value	ARR15 1	ARR14 1	ARR13 1	ARR12 1	ARR11 1	ARR10 1	ARR9 1	ARR8 1
0x10	TIMx_ARRL Reset value	ARR7 1	ARR6 1	ARR5 1	ARR4 1	ARR3 1	ARR2 1	ARR1 1	ARR0 1
0x11	TIMx_CCR1H Reset value	CCR115 0	CCR114 0	CCR113 0	CCR112 0	CCR111 0	CCR110 0	CCR19 0	CCR18 0
0x12	TIMx_CCR1L Reset value	CCR17 0	CCR16 0	CCR15 0	CCR14 0	CCR13 0	CCR12 0	CCR11 0	CCR10 0
0x13	TIMx_CCR2H Reset value	CCR215 0	CCR214 0	CCR213 0	CCR212 0	CCR211 0	CCR210 0	CCR29 0	CCR28 0
0x14	TIMx_CCR2L Reset value	CCR27 0	CCR26 0	CCR25 0	CCR24 0	CCR23 0	CCR22 0	CCR21 0	CCR20 0
0x15	TIMx_BKR Reset value	MOE 0	AOE 0	BKP 0	BKE 0	OSSR 0	OSSI 0	LOCK 0	LOCK 0
0x16	TIMx_OISR Reset value	- 0	OIS4 0	OIS3N 0	OIS3 0	OIS2N 0	OIS2 0	OIS1N 0	OIS1 0

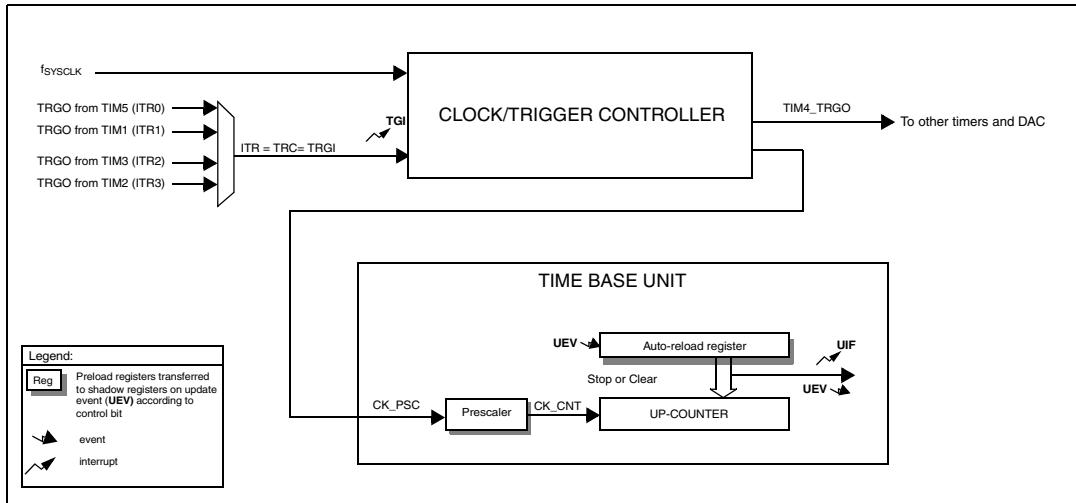
## 21 8-bit basic timer (TIM4)

### 21.1 Introduction

The timer consists of an 8-bit auto-reload up-counter driven by a programmable prescaler. It can be used for time base generation, with interrupt generation on timer overflow.

Refer to [Section 19.3 on page 287](#) for the general description of the timer features.

**Figure 126. TIM4 block diagram**



### 21.2 TIM4 main features

The main features include:

- 8-bit auto-reload up counter
- 4-bit programmable prescaler which allows dividing (also “on the fly”) the counter clock frequency by any power of 2 from 1 to 32768.
- Interrupt generation
  - On counter update: Counter overflow
  - On trigger input
- DMA request generation
  - On counter update: Counter overflow

### 21.3 TIM4 interrupts

The timer has 2 interrupt request sources:

- Update interrupt (overflow, counter initialization)
- Trigger input

## 21.4 TIM4 clock selection

The clock source for the timer is the internal clock ( $f_{SYSCLK}$ ). It is connected directly to the CK\_PSC clock that feeds the prescaler driving the counter clock CK\_CNT.

### Prescaler

The prescaler implementation is as follows:

- The TIM4 prescaler is based on a 16-bit counter controlled through a 4-bit register (in TIM4\_PSCR register). It can be changed on the fly as this control register is buffered. It can divide the counter clock frequency by any power of 2 from 1 to 32768.

The counter clock frequency is calculated as follows:

$$f_{CK\_CNT} = f_{CK\_PSC}/2^{(PSCR[3:0])}$$

The prescaler value is loaded through a preload register. The shadow register, which contains the current value to be used, is loaded as soon as the LS byte has been written.

Read operations to the TIM\_PSCR registers access the preload registers, so no special care needs to be taken to read them.

## 21.5 TIM4 registers

### 21.5.1 Control register 1 (TIM4\_CR1)

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
ARPE	Reserved			OPM	URS	UDIS	CEN
rw				rw	rw	rw	rw

Bit 7 **ARPE**: Auto-reload preload enable

- 0: TIM4\_ARR register is not buffered through a preload register. It can be written directly
- 1: TIM4\_ARR register is buffered through a preload register

Bits 6:4 Reserved, must be kept cleared

Bit 3 **OPM**: One-pulse mode

- 0: Counter is not stopped at update event
- 1: Counter stops counting at the next update event (clearing the CEN bit)

Bit 2 **URS**: Update request source

- 0: When enabled, an update interrupt request is sent as soon as registers are updated (counter overflow).
- 1: When enabled, an update interrupt request is sent only when the counter reaches the overflow/underflow.

Bit 1 **UDIS**: Update disable

- 0: A UEV is generated as soon as a counter overflow occurs or a software update is generated. Buffered registers are then loaded with their preload values.
- 1: A UEV is not generated, shadow registers keep their value (ARR, PSC). The counter and the prescaler are re-initialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

- 0: Counter disable
- 1: Counter enable

### 21.5.2 Control register 2 (TIM4\_CR2)

Address offset: 0x01

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved		MMS[2:0]				Reserved		

Bit 7 Reserved, must be kept cleared

Bits 6:4 MMS[2:0]: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: Reset - the UG bit from the TIM4\_EGR register is used as a trigger output (TRGO). If the reset is generated by the trigger input (clock/trigger mode controller configured in trigger reset mode), the signal on the TRGO is delayed compared to the actual reset.

001: Enable - the counter enable signal is used as a trigger output (TRGO). It is used to start several timers at the same time or to control a window in which a slave timer is enabled. The counter enable signal is generated by a logic OR between the CEN control bit and the trigger input when configured in gated mode. When the counter enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIM4\_SMCR register).

010: Update - The update event is selected as trigger output (TRGO)

011: Reserved

100: Reserved

101: Reserved

111: Reserved

Bits 3:0 Reserved, must be kept cleared

### 21.5.3 Slave mode control register (TIM4\_SMCR)

Address offset: 0x02

Reset value: 0x00

	7	6	5	4	3	2	1	0
MSM		TS[2:0]			Reserved		SMS[2:0]	

Bit 7 MSM: Master/slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between timers (through TRGO).

**Bits 6:4 TS[2:0]: Trigger selection**

This bitfield selects the trigger input to be used to synchronize the counter.

- 000: Internal trigger ITR0 connected to TIM5 TRGO
- 001: Internal trigger ITR1 connected to TIM1 TRGO
- 010: Internal trigger ITR2 connected to TIM3 TRGO
- 011: Internal trigger ITR3 connected to TIM2 TRGO
- 100: Reserved
- 101: Reserved
- 110: Reserved
- 111: Reserved

*Note: These bits must only be changed when they are not used (e.g. when SMS = 000) to avoid wrong edge detections at the transition.*

Bit 3 Reserved.

**Bits 2:0 SMS[2:0]:Clock/trigger/slave mode selection**

When external signals are selected, the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input control register and control register description).

000: Clock/trigger controller disabled - If CEN = 1, the prescaler is clocked directly by the internal clock.

- 001: Reserved
- 010: Reserved
- 011: Reserved

100: Trigger reset mode - The rising edge of the selected trigger signal (TRGI) reinitializes the counter and generates an update of the registers.

101: Gated mode - The counter clock is enabled when the trigger signal (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

110: Trigger mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

111: External clock mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

### 21.5.4 DMA request enable register (TIM4\_DER)

Address offset: 0x03

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved							UDE
							rw

Bits 7:1 Reserved, always read as 0.

Bit 0 **UDE**: Update DMA request enable<sup>(1)</sup>

- 0: Update DMA request disabled
- 1: Update DMA request enabled

*Note: The conditions for generating a DMA request on the UEV are the same as for setting the UIF bit (in the TIM4\_SR1 register). In particular, the DMA request depends on the URS bit (in the TIM4\_CR1 register).*

### 21.5.5 Interrupt enable register (TIM4\_IER)

Address offset: 0x04

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved	TIE	Reserved					
	rw						

Bit 7 Reserved, must be kept cleared

Bit 6 **TIE**: Trigger interrupt enable

- 0: Trigger interrupt disabled
- 1: Trigger interrupt enabled

Bits 5:1 Reserved, must be kept cleared

Bit 0 **UIE**: Update interrupt enable

- 0: Update interrupt disabled
- 1: Update interrupt enabled

### 21.5.6 Status register 1 (TIM4\_SR)

Address offset: 0x05

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved	TIF	Reserved					
	rc_w0						

Bit 7 Reserved, must be kept cleared

**Bit 6 **TIF**: Trigger interrupt flag**

This flag is set by hardware on trigger event (active edge detected on TRGI signal, both edges in case gated mode is selected). It is cleared by software.

- 0: No trigger event has occurred
- 1: Trigger interrupt pending

Bits 5:1 Reserved, must be kept cleared

**Bit 0 **UIF**: Update interrupt flag**

This bit is set by hardware on an update event. It is cleared by software.

- 0: No update has occurred
- 1: Update interrupt pending. This bit is set by hardware when the registers are updated:
  - At overflow if UDIS = 0 in the TIM4\_CR1 register
  - When CNT is re-initialized by software using the UG bit in the TIM4\_EGR register, if URS = 0 and UDIS = 0 in the TIM4\_CR1 register.

### 21.5.7 Event generation register (TIM4\_EGR)

Address offset: 0x06

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved	TG w			Reserved			UG w

Bit 7 Reserved, must be kept cleared

**Bit 6 **TG**: Trigger generation**

This bit is set by software to generate an event. It is automatically cleared by hardware.

- 0: No action
- 1: The TIF flag is set in TIM4\_SR1 register. An interrupt is generated if enabled by the TIE bit

Bits 5:1 Reserved, must be kept cleared

**Bit 0 **UG**: Update generation**

This bit can be set by software, it is automatically cleared by hardware.

- 0: No action
- 1: Re-initializes the counter and generates an update of the registers. Note that the prescaler counter is also cleared.

### 21.5.8 Counter (TIM4\_CNTR)

Address offset: 0x07

Reset value: 0x00

7	6	5	4	3	2	1	0
CNT[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **CNT[7:0]**: Counter value

### 21.5.9 Prescaler register (TIM4\_PSCR)

Address offset: 0x08

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved				PSC[3:0]			
				rw	rw	rw	rw

Bits 7:4 Reserved, must be kept cleared

Bits 3:0 **PSC[3:0]**: Prescaler value

The prescaler value divides the CK\_PSC clock frequency. The counter clock frequency  $f_{CK\_CNT}$  is equal to  $f_{CK\_PSC} / 2(PSC[3:0])$ .

PSC contains the value which is loaded into the active prescaler register at each UEV (including when the counter is cleared through the UG bit of TIM4\_EGR).

Consequently, a UEV must be generated so that a new prescaler value can be taken into account.

### 21.5.10 Auto-reload register (TIM4\_ARR)

Address offset: 0x09

Reset value: 0xFF

7	6	5	4	3	2	1	0
ARR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **ARR[7:0]**: Auto-reload value

### 21.5.11 TIM4 register map and reset values

**Table 75. TIM4 register map**

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	TIM4_CR1 Reset value	ARPE 0	- 0	- 0	- 0	OPM 0	URS 0	UDIS 0	CEN 0
0x01	TIM4_CR2 Reset value	- 0	MMS2 0	MMS1 0	MMS0 0	- 0	- 0	- 0	- 0
0x02	TIM4_SMCR Reset value	MSM 0	TS2 0	TS1 0	TS0 0	- 0	SMS2 0	SMS1 0	SMS0 0
0x03	TIM4_DER Reset value	- 0	UDE 0						
0x04	TIM4_IER Reset value	- 0	TIE 0	- 0	- 0	- 0	- 0	- 0	UIE 0
0x05	TIM4_SR1 Reset value	- 0	TIF 0	- 0	- 0	- 0	- 0	- 0	UIF 0
0x06	TIM4_EGR Reset value	- 0	TG 0	- 0	- 0	- 0	- 0	- 0	UG 0
0x07	TIM4_CNTR Reset value	CNT7 0	CNT6 0	CNT5 0	CNT4 0	CNT3 0	CNT2 0	CNT1 0	CNT0 0
0x08	TIM4_PSCR Reset value	- 0	- 0	- 0	- 0	PSC3 0	PSC2 0	PSC1 0	PSC0 0
0x09	TIM4_ARR Reset value	ARR7 1	ARR6 1	ARR5 1	ARR4 1	ARR3 1	ARR2 1	ARR1 1	ARR0 1

## 22 Infrared (IRTIM) interface

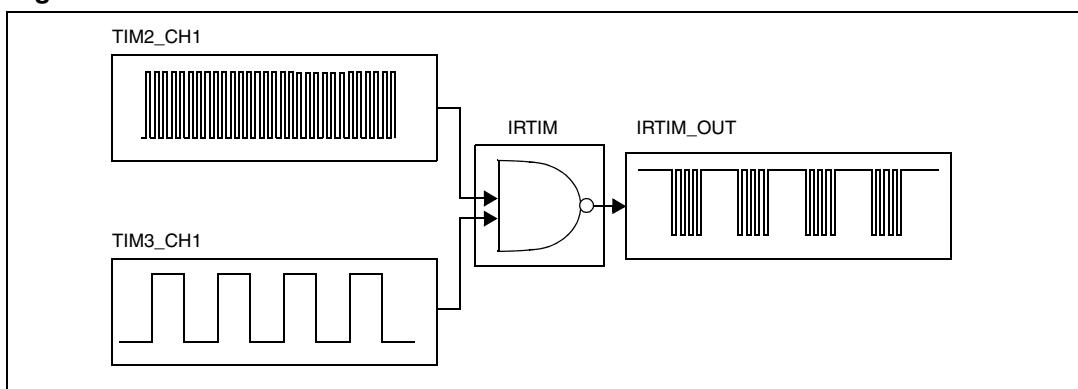
This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

### 22.1 Introduction

An infrared interface (IRTIM) can be used with an IR LED to perform remote control functions.

To generate the infrared remote control signals, the IR interface must be enabled and TIM2 channel 1 (TIM2\_OC1) and TIM3 channel 1 (TIM3\_OC1) must be properly configured to generate correct waveforms.

**Figure 127. IR internal hardware connections with TIM2 and TIM3**



### 22.2 Main features

All standard IR pulse modulation modes can be obtained by programming the two timer output compare channels.

TIM 2 is used to generate the high frequency carrier signal, while TIM3 generates the modulation envelope.

The infrared function is output on the IR\_TIM pin. The activation of this function is done through the IR\_CR register. When the IR function is enabled by setting the IR\_EN bit, the standard TIM2\_CC1 and TIM3\_CC1 become automatically inactive (these pins may be used as general purpose I/O pins or for other alternate functions).

The high sink LED driver capability (only available on the IR\_TIM pin) can be activated through the HS\_EN bit in the IR\_CR register and used to sink the high current needed to directly control an infrared LED. When the pin is driving the LED in this mode, the other pin input/output levels cannot be guaranteed. It is therefore recommended to program all other device I/Os in input mode without interrupt before sending any infrared signal. The previous function can be restored immediately after the infrared communication is completed.

When the high sink capability of the pin is not used (or the current is limited to the standard I/O capabilities) all other pins of the device can be used normally.

## 22.3 IRTIM register

### 22.3.1 Control register (IR\_CR)

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	HS_EN	IR_EN
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:2 Reserved. Must be kept cleared

Bit 1 **HS\_EN**: High Sink LED driver capability enable.

0: High Sink LED driver capability disabled.

1: High Sink LED driver capability enabled.

When activated, this pin can sink 20 mA min. with a power supply down to 2 V.

Bit 0 **IR\_EN**: Infrared output enable.

This bit enables the IR output.

0: IR\_TIM output disabled.

1: IR\_TIM output enabled and provided to PA0 (TIM2 and TIM3 must have been previously configured properly by software)

### 22.3.2 IRTIM register map and reset values

Table 76. IR register map

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	IR_CR Reset value	- 0	- 0	- 0	- 0	- 0	- 0	HS_EN 0	IR_EN 0

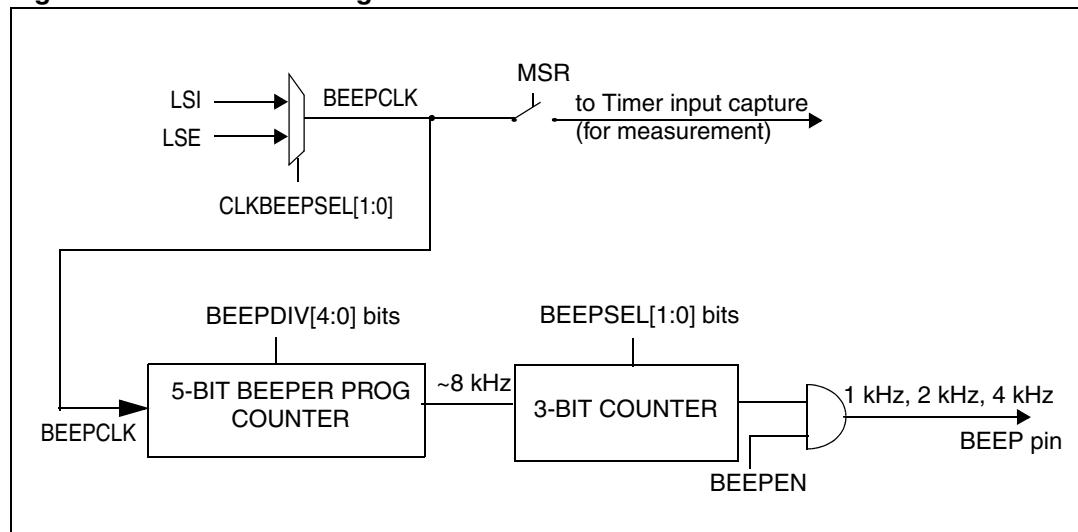
## 23 Beeper (BEEP)

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

### 23.1 Introduction

This Beeper module is used to generate a BEEP signal in the range of 1, 2 or 4 kHz. The Beeper clock is derived from the LSE or LSI clock. The clock is selected by programming the CLKBEEPSEL[1:0] bits in the Clock BEEP register (CLK\_CBEPPR).

**Figure 128. BEEP block diagram**



## 23.2 BEEP functional description

### 23.2.1 Beeper operation

To use the BEEP function, perform the following steps in order:

1. Calibrate the LSI clock frequency as described in [Section 23.2.2: Beeper calibration](#) to define BEEPDIV[4:0] value.
2. Select 1 kHz, 2 kHz or 4 kHz output frequency by writing to the BEEPSEL[1:0] bits in the [BEEP control/status register 2 \(BEEP\\_CSR2\)](#).
3. Set the BEEPEN bit in the [BEEP control/status register 2 \(BEEP\\_CSR2\)](#) to enable the LS clock source.

**Note:** *The prescaler counter starts to count only if BEEPDIV[4:0] value is different from its reset value, 0x1F.*

### 23.2.2 Beeper calibration

This procedure can be used to calibrate the LS clock sources in order to reach the standard frequency output, 1 kHz, 2 kHz or 4 kHz.

Use the following procedure:

1. Disable the BEEP clock by resetting the CLKBEEPSEL bit in the CLK\_CBEPR register.
2. Measure the LSI clock frequency (refer to [Section 23.2.3: LSI clock frequency measurement](#))

**Note:** *This step is performed when the beeper is clocked by the LSI clock source.*

3. Calculate the BEEP<sub>DIV</sub> value as follows, where A and x are the integer and fractional part of BEEPCLK/8 (in kHz):
 
$$\text{BEEP}_{\text{DIV}} = \begin{cases} A-2 & \text{when } x \text{ is less than or equal to } A/(1+2^*A), \text{ else} \\ A-1 & \end{cases}$$
4. Write the resulting BEEP<sub>DIV</sub> value in the BEEPDIV[4:0] bits in the [BEEP control/status register 2 \(BEEP\\_CSR2\)](#).

### 23.2.3 LSI clock frequency measurement

The frequency dispersion of the Low Speed Internal RC (LSI) oscillator after RC factory trimming is 38 kHz on the whole temperature range. To obtain a precise beeper output, the exact LSI frequency has to be measured.

Use the following procedure:

1. Set the MSR bit in the [BEEP control/status register 1 \(BEEP\\_CSR1\)](#) to connect the LSI clock internally to ICAP1 of the TIM2 timer.
2. Measure the frequency of LSI clock using the Timer input capture interrupt.

## 23.3 BEEP registers

### 23.3.1 BEEP control/status register 1 (BEEP\_CSR1)

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved							MSR
							rw

Bits 7:1 Reserved, must be kept cleared.

Bit 0 **MSR**: Measurement enable

This bit connects the BEEPCLK to the TIM2 channel 1 input capture. This allows the timer to be used to measure the LSI frequency ( $f_{LSI}$ ).

0: Measurement disabled

1: Measurement enabled

### 23.3.2 BEEP control/status register 2 (BEEP\_CSR2)

Address offset: 0x03

Reset value: 0x1F

7	6	5	4	3	2	1	0
BEEPSEL[1:0]		BEEPEN	BEEPDIV[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:6 **BEEPSEL[1:0]**: BEEP selection

These bits are set and cleared by software to select 1, 2 or 4 kHz BEEP output when calibration is done.

00: BEEPCLK/(8 x BEEP<sub>DIV</sub>) kHz output

01: BEEPCLK/(4 x BEEP<sub>DIV</sub>) kHz output

1x: BEEPCLK/(2 x BEEP<sub>DIV</sub>) kHz output

Bit 5 **BEEPEN**: BEEP enable

This bit is set and cleared by software to enable the BEEP feature.

0: BEEP disabled

1: BEEP enabled

*Note:* Before enabling or disabling the BEEP feature, the BEEP clock must be disabled by resetting the CLKBEEPSEL[1:0] bits in the CLK\_CBEPPR register.

Bits 4:0 **BEEPDIV[4:0]**: BEEP prescaler divider

These bits are set and cleared by software to define the beeper prescaler dividing factor BEEP<sub>DIV</sub>.

0x00: BEEP<sub>DIV</sub> = 2

0x01: BEEP<sub>DIV</sub> = 3

...

0x0E: BEEP<sub>DIV</sub> = 16

0x0F: BEEP<sub>DIV</sub> = 17

...

0x1E: BEEP<sub>DIV</sub> = 32

*Note:* This register must not be kept at its reset value (0x1F)

### 23.3.3 BEEP register map and reset values

Table 77. BEEP register map

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	BEEP_CSR1	-0	-0	-0	-0	-0	-0	-0	MSR 0
0x01 to 0x02	Reserved								
0x03	BEEP_CSR2	BEEPSEL2 0	BEEPSEL1 0	BEEPEN 0	BEEPDIV4 1	BEEPDIV3 1	BEEPDIV2 1	BEEPDIV1 1	BEEPDIV0 1

## 24 Real-time clock (RTC)

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

### 24.1 Introduction

The real-time clock (RTC) is an independent BCD timer/counter. It provides a time-of-day clock and calendar with an associated programmable alarm. The RTC also includes an auto wakeup unit useful for managing Low power modes.

8-bit registers contain seconds, minutes, hours (12 or 24 hour format), day (day of the week), date (day of the month), month, and year, coded in binary coded decimal format (BCD). The sub-seconds value is also available in binary format on low, medium+ and high density devices only. The adjustment for 28-, 29- (leap year), 30-, and 31-day months is performed automatically.

Additional 8-bit registers contain the programmable alarm subseconds (low, medium+ and high density devices only), seconds, minutes, hours, day, and date.

Low, medium+ and high density devices also allow to calibrate the RTC with a resolution of 0.954 ppm.

After reset, the RTC registers are protected against possible parasitic write accesses.

As long as the supply voltage is maintained in the operating range, the RTC never stops, regardless of the MCU status (Run mode, low power mode or under reset).

## 24.2 RTC main features

The RTC unit main features are the following ones (see [Figure 130: RTC block diagram \(medium density devices\)](#) and [Figure 131: RTC block diagram \(medium+ and high density devices\)](#)):

- A calendar with subseconds (low, medium+ and high density devices only), seconds, minutes, hours (12 or 24 format), day (day of the week), date (day of the month), month, and year.
- Daylight saving time adjustment by software.
- A programmable alarm with interrupt (alarm A). The alarm can be triggered by any combination of the calendar fields.
- An auto wake up unit providing periodic flag triggering an automatic wakeup interrupt.
- Five maskable interrupt/events:
  - Alarm A
  - Wakeup interrupt
  - 3 tamper detections
- Accurate synchronization with an external clock using the subsecond shift feature (on low, medium+ and high-density devices).
- Digital calibration with an accuracy of 0.954 ppm (on low, medium+ and high-density devices).
- Three tamper inputs with configurable filter and internal pull-up to wake up the CPU (available on low, medium+ and high-density devices only).
- Alternate function outputs:
  - RTC\_CALIB output: configurable 512 Hz clock output or 1 Hz clock output on low, medium+ and high-density devices (with RTC clock at 32.768 kHz).
  - RTC\_ALARM output: alarm A or wakeup flag can be routed to this output.

## 24.3 RTC functional description

The RTC unit is controlled by a set of 8-bit registers accessible in read or write mode. For more information, refer to [Section 24.6: RTC registers](#).

In this section:

- RTC\_CRx stands for RTC\_CR1 / RTC\_CR2 / RTC\_CR3,
- RTC\_SSRx for RTC\_SSRH/RTC\_SSRL,
- RTC\_TRx for RTC\_TR1 / RTC\_TR2 / RTC\_TR3,
- RTC\_DRx for RTC\_DR1 / RTC\_DR2 / RTC\_DR3,
- RTC\_ALRMARx for RTC\_ALRMAR1 / RTC\_ALRMAR2 / RTC\_ALRMAR3 / RTC\_ALRMAR4,
- RTC\_ALRMASSRx for RTC\_ALRMASSRH/RTC\_ALRMASSRL,
- RTC\_SHIFTRx for RTC\_SHIFTRH/RTC\_SHIFTRL,
- RTC\_CALRx for RTC\_CALRH/RTC\_CALRL,
- RTC\_TCRx for RTC\_TCR1/RTC\_TCR2

### 24.3.1 Clock and prescalers

The RTC clock source (RTCCLK) used for timer/counter is selected through the clock controller. It can be either the HSE, LSE, HSI or LSI clock. For more information about the RTC clock source configuration, please refer to [Section 9.9: RTC and LCD clock](#).

A clock security system on LSE is implemented on low, medium+ and high density devices to monitor the low speed external clock when it is used as the RTC clock source. For more details about how to use the CSS on LSE feature, please refer to [Section 9.8.2: Clock security system on LSE](#).

In order to access the RTC registers properly, the system clock frequency ( $f_{SYSCLK}$ ) must be equal to or greater than four times the  $f_{RTCCLK}$  RTC clock frequency. This ensures a secure behavior of the synchronization mechanism.

When the system clock (SYSCLK) is equal to a low speed clock (like LSE or LSI), the user must use exactly the same clock as RTCCLK and must set the RATIO bit in the RTC\_CR1 register to disable the synchronization mechanism. In these conditions, the RSF synchronization flag is meaningless.

A programmable prescaler stage generates a 1 Hz clock used to update the calendar. This prescaler stage is split into 2 programmable prescalers ([Figure 130](#) and [Figure 131](#)):

- A 7-bit asynchronous prescaler configured through PREDIV\_A bits of the RTC\_APERR register
- A synchronous prescaler configured through PREDIV\_S bits of the RTC\_SPRERx registers. The prescaler features 13 bits for medium density devices, and 15 bits for low, medium+ and high density devices.

*Note:*

*This division into two blocks (the asynchronous prescaler and the synchronous prescaler) has been defined to reach the lowest possible consumption.*

*It is recommended to keep a high value in the asynchronous prescaler when both prescalers are used to ensure the best consumption level.*

The asynchronous prescaler division factor is consequently set by default to 128, and the synchronous division factor to 256, to get an internal clock frequency of 1 Hz ( $ck_{spre}$ ) with an LSE frequency of 32768 Hz.

The minimum division factor is 2 on medium density devices and 1 on low, medium+ and high density devices. The maximum division factor is  $2^{20}$  on medium density devices and  $2^{22}$  on low, medium+ and high density devices. This corresponds to a maximum input frequency of around 1 MHz and 4 MHz respectively.

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV_S + 1)(PREDIV_A + 1)}$$

The 1 Hz internal clock ( $ck_{spre}$ ) is input to the calendar.

The  $ck_{spre}$  clock can also be used as a time base for the 16-bit wakeup auto-reload timer.

*Note:*

*The 16-bit wakeup auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler so as to reach short time-out periods. See [Section 24.3.4: Periodic auto-wakeup](#) for details.*

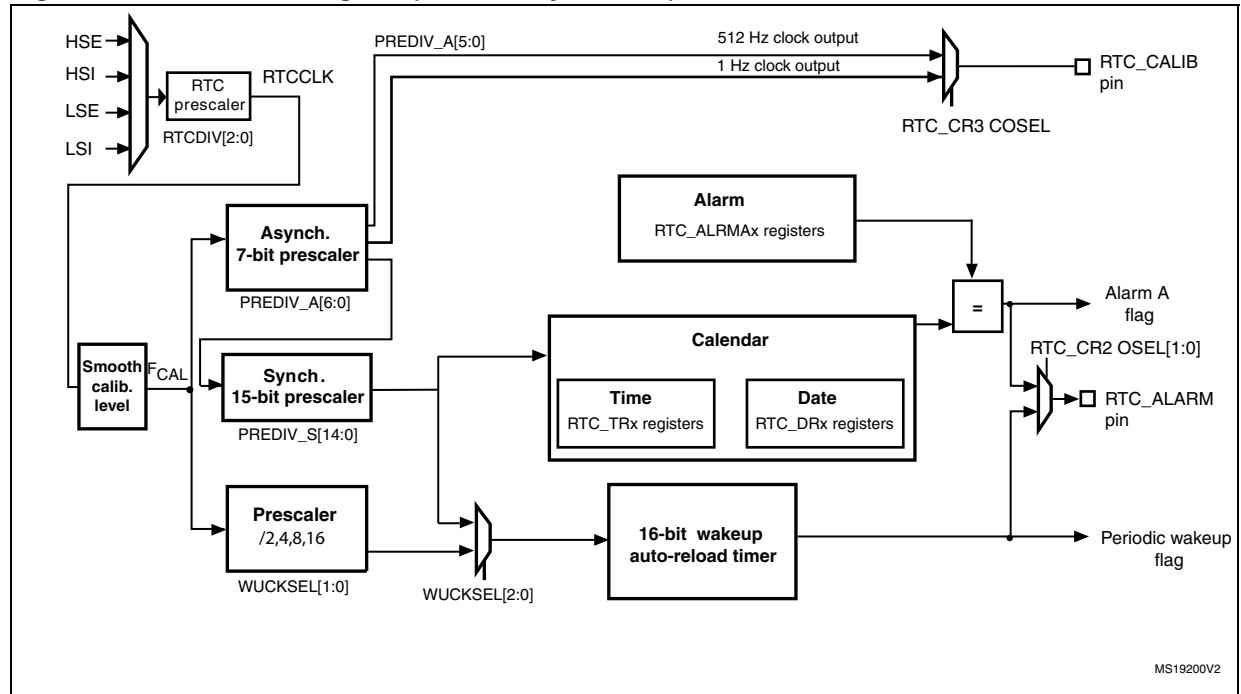
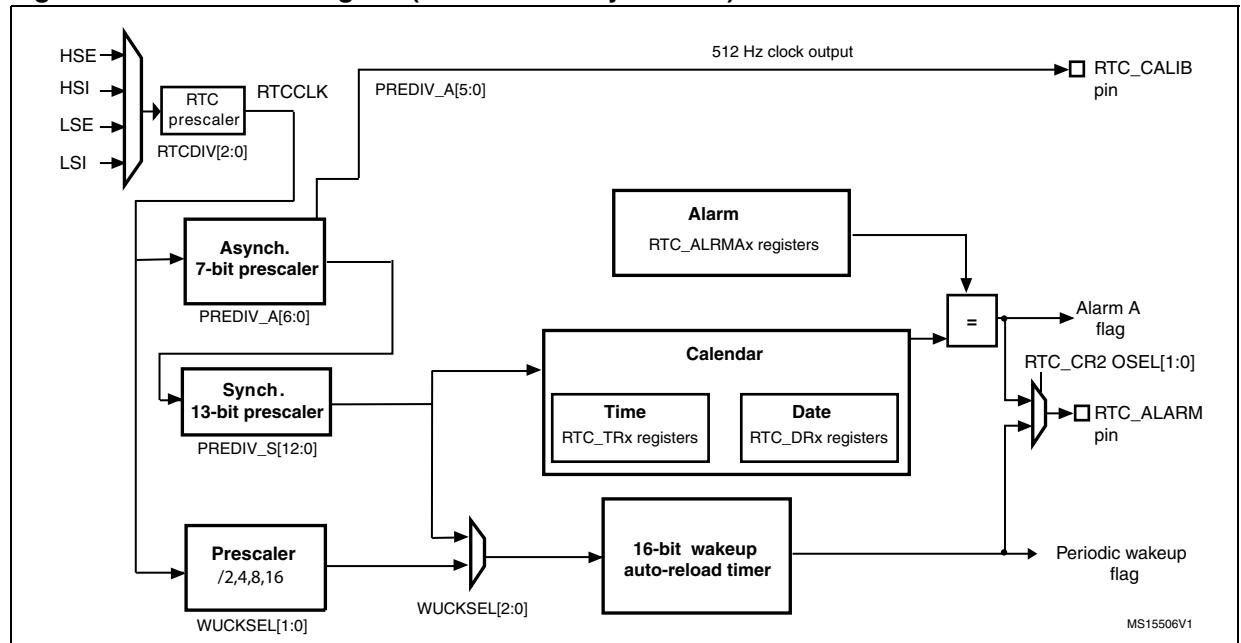
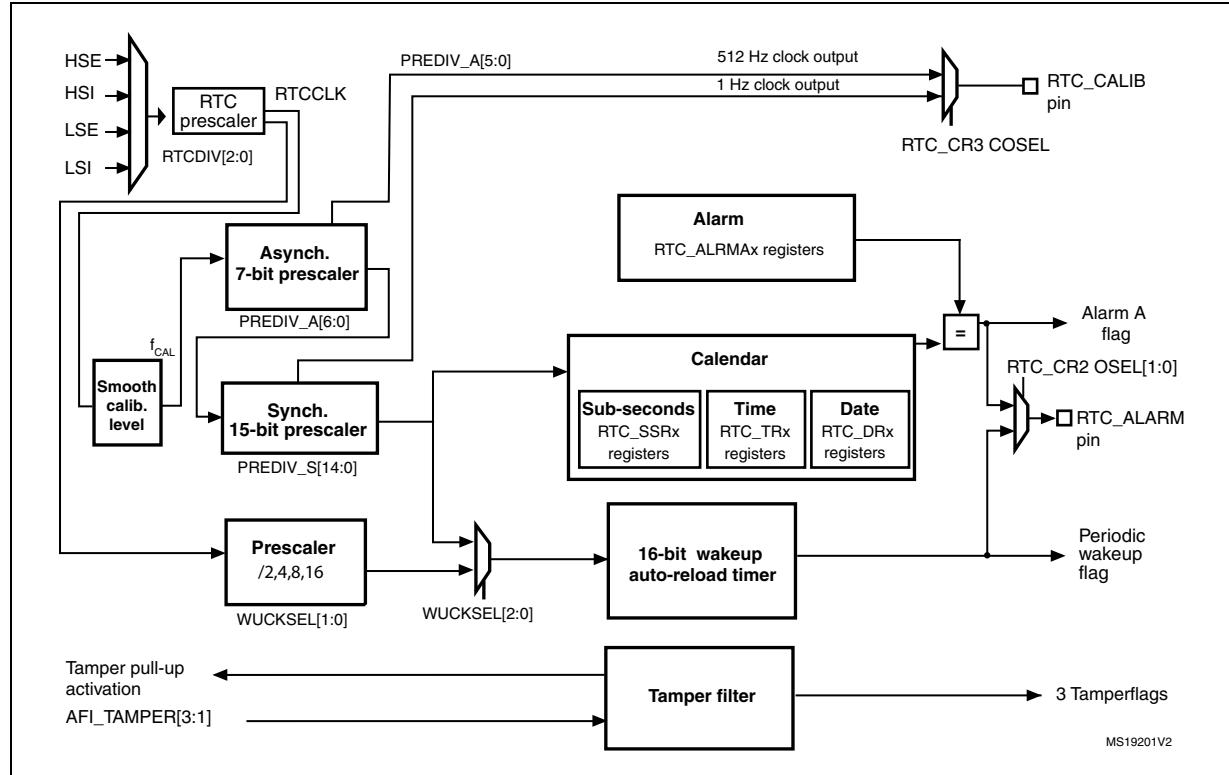
**Figure 129. RTC block diagram (low density devices)****Figure 130. RTC block diagram (medium density devices)**

Figure 131. RTC block diagram (medium+ and high density devices)



### 24.3.2 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers synchronized with SYSCLK (system clock). In low, medium+ and high density devices, they can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC\_SSRx (subseconds) (available on low, medium+ and high density devices only)
- RTC\_TR1 (seconds)
- RTC\_TR2 (minutes)
- RTC\_TR3 (hours)
- RTC\_DR1 (date)
- RTC\_DR2 (day and month)
- RTC\_DR3 (year)

The current calendar value is periodically copied into these shadow registers which are reset by system reset. The copy periodicity is the RTCCLK period. The RSF bit is set in the RTC\_ISR register each time the copy is performed.

*Note:* The copy is not performed during Active-halt mode. When exiting Active-halt mode, the shadow registers are updated after up to one RTCCLK period.

When the user makes a read access to the calendar registers, he reads the content of the shadow registers. In low, medium+ and high density devices, it is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC\_CR1 register. By default, this bit is cleared, and the user accesses the shadow registers.

### 24.3.3 Programmable alarm

The RTC provides a programmable alarm (alarm A). The programmable alarm function is enabled through the ALRAE bit in the RTC\_CR2 register.

The ALRAF flag is set to 1 if the calendar subseconds (low, medium+ and high density devices only), seconds, minutes, hours and/or date match the value programmed in the RTC\_ALRMASSRx and RTC\_ALRMARx alarm registers. The subseconds, seconds, minutes, hours and/or date can be independently selected or masked through the ALSSx and MSKx bits of the RTC\_ALRMASSRx and RTC\_ALRMARx registers.

The alarm interrupt is enabled through the ALRAIE bit in the RTC\_CR2 register. When enabled, the programmable alarm interrupt exits the device from Low power modes.

The alarm A flag signal can be routed to the RTC\_ALARM device output when the OSEL[1:0] bits in the RTC\_CR3 register have the adequate value. The RTC\_ALARM polarity can be configured through the POL bit in the RTC\_CR3 register (see [Section 24.6.11](#)).

A positive shift operation (see [Section 24.3.8: RTC synchronization \(low, medium+ and high density devices only\)](#)) might cause an alarm to be missed if the subsecond field is compared (if MASKSS in RTC\_ALRMASSMSKR is non-zero). Similarly, a negative shift operation might cause an alarm to be activated an extra time if the sub-second field is compared.

**Caution:** If the “seconds” field is masked (MSK1 bit set in RTC\_ALRMAR1), the synchronous prescaler division factor set in the RTC\_SPRER register must be at least 3 to ensure a correct behavior of the RTC.

### 24.3.4 Periodic auto-wakeup

The periodic wakeup flag is generated by a 16-bit programmable binary auto-reload down-counting timer. The wakeup timer range can be extended to 17 bits.

The wakeup function is enabled through the WUTE bit in the RTC\_CR2 register. The wakeup timer clock can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16. When RTCCLK is LSE (32.768 kHz), it is possible to configure the wakeup interrupt period from 122 µs to 32 s, with a resolution down to 61µs,
- ck\_spre (usually 1 Hz internal clock). When ck\_spre is 1 Hz, a wakeup time from 1 s to around 36 hours with one-second resolution can be achieved. This large programmable time range is divided in 2 parts:
  - from 1s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18h to 36h when WUCKSEL[2:1] = 11. In this last case  $2^{16}$  is added to the 16-bit counter current value.

When the initialization sequence is completed (see [Section : Programming the auto-wakeup timer](#)), the wakeup timer starts down-counting. When the wakeup function is enabled, the down-counting remains active in low power modes. In addition, when it reaches 0, the WUTF flag in the RTC\_ISR2 register is set and the wakeup counter is automatically reloaded with its reload value (WUT register value).

The WUTF flag must then be reset by the software.

When the periodic wakeup interrupt is enabled by setting the WUTIE bit in the RTC\_CR2 register, it can exit the device from low power modes.

The periodic wakeup flag can be routed to the RTC\_ALARM device output when the OSEL[1:0] bits in the RTC\_CR3 register have the adequate value. The RTC\_ALARM polarity can be configured through the POL bit in the RTC\_CR3 register (see [Section 24.6.11](#)).

System reset as well as Run and Low power operating modes have no influence on the down-counting timer.

### 24.3.5 RTC initialization and configuration

#### RTC register write protection

By default, all the RTC registers (except the RTC\_ISR2 register which contains the alarm and auto-wakeup timer interrupt source flags) are write-protected. Writing to the RTC registers is enabled by writing a key into the RTC\_WPR write protection register.

The following steps are required to unlock the write protection of the RTC registers.

1. Write ‘**0xCA**’ into the RTC\_WPR register
2. Write ‘**0x53**’ into the RTC\_WPR register

*Note:* *Writing a wrong key reactivates the write protection.*

#### Calendar initialization and configuration

To program the initial time and date calendar values including the time format and potentially the prescaler setup, the following sequence is required:

1. Set INIT bit to 1 in the RTC\_ISR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC\_ISR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles for synchronization purposes.
3. To generate a 1 Hz clock for the calendar counter, program the prescaler register (RTC\_PRER).
4. Load the initial time and date values in the shadow registers (RTC\_TR and RTC\_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC\_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is completed, the subseconds value is also reinitialized, so that the next second increment can occur after a full second.

- Note:*
- 1 *After a system reset, the application can read the INITS flag in the RTC\_ISR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its power-on reset default value (0x00).*
  - 2 *To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC\_ISR register.*

#### Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BCK of the RTC\_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BCK bit to memorize this operation.

### Programming the alarm

Use the following procedure to program or update the programmable alarm (alarm A):

1. Clear ALRAE in RTC\_CR2 to disable alarm A.
2. Poll ALRAWF until it is set in RTC\_ISR1 to make sure the access to alarm registers is allowed. On medium density devices, it takes around 2 RTCCLK clock cycles (due to clock synchronization). On low, medium+ and high density devices, ALRAWF is always set, so this step can be skipped.
3. Program alarm A registers: RTC\_ALRMASSRx and RTC\_ALRMASSMSKR (on low, medium+ and high density devices only) and RTC\_ALRMARx.
4. Set ALRAE in RTC\_CR2 to enable alarm A again.
5. On low, medium+ and high-density devices, after setting ALRAE to '1', the alarm effectively remains deactivated for one additional ck\_apre cycle. In other words, an alarm which is set to occur on the first sub-second update after enabling the alarm will be masked.

*Note:* *Each change in the RTC\_CR2 register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.*

### Programming the auto-wakeup timer

The following sequence is required to configure or change the wakeup timer reload value:

1. Clear WUTE in RTC\_CR2 to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC\_ISR1 to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the value into the wakeup timer (RTC\_WUTRL and RTC\_WUTRH) and select the desired clock (WUCKSEL[2:0] bits in RTC\_CR1).
4. Set WUTE in RTC\_CR2 register to enable the timer again. The wakeup timer restarts down-counting.

## 24.3.6 Reading the calendar

- **In medium density devices, or when BYPSHAD is cleared:**

In order to read the RTC calendar registers (RTC\_SSRx, RTC\_TRx and RTC\_DRx) properly, the system clock frequency ( $f_{SYSCLK}$ ) must be equal to or greater than four times the  $f_{RTCCLK}$  RTC clock frequency. This ensures a secure behavior of the synchronization mechanism.

The RSF bit is set in RTC\_ISR register each time the calendar registers are copied into the RTC\_SSRx, RTC\_TRx and RTC\_DRx shadow registers. The copy is performed every RTCCLK cycle. To ensure the consistency between the values when the software reads the calendar, the update of all shadow registers (RTC\_SSRL, RTC\_SSRH, RTC\_TR1, RTC\_TR2, RTC\_TR3, RTC\_DR1, RTC\_DR2 and RTC\_DR3) is frozen after RTC\_SSRL (low, medium+ and high density devices) or RTC\_TR1 (medium density devices) is read, and until RTC\_DR3 is read. In low, medium+ and high-density devices, if the software does not need to read the sub-second values, it can first read RTC\_TR1 and all the values are locked until RTC\_DR3 is read.

In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK period: RSF must be cleared by software after the first calendar read,

and then the software must wait until RSF is set before reading again the calendar shadow registers.

After a low power mode wakeup, RSF must be cleared by software. The software must then wait until it is set again before reading the RTC\_TR and RTC\_DR registers. The RSF bit must be cleared after wakeup and not before entering low power mode.

- Note:**
- 1 After a system reset, the software must wait until RSF is set before reading the RTC\_SSRx, RTC\_TRx and RTC\_DRx registers. Indeed, a system reset resets the shadow registers to their default values.
  - 2 After an initialization (refer to [Section : Calendar initialization and configuration](#)), or after a shift operation (refer to [Section 24.3.8: RTC synchronization \(low, medium+ and high density devices only\)](#)): the software must wait until RSF is set before reading the RTC\_SSRx, RTC\_TRx and RTC\_DRx registers.

- In low, medium+ and high density devices, when the BYPSHAD control bit in the RTC\_CR1 register is set to 1 (bypass shadow registers):

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for RSF to be set. This is especially useful after exiting from Active-halt since the shadow registers are not updated during Active-halt mode. When BYPSHAD is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct.

- Note:** When BYPSHAD is set to 1, the instructions which read the calendar registers require one extra system cycle to complete.

### 24.3.7 Resetting the RTC

The calendar shadow registers (RTC\_SSRx, RTC\_TRx and RTC\_DRx) and the RTC status registers (RTC\_ISRx) are reset to their default value by all available system reset sources.

On the contrary, the RTC current calendar registers and control registers (RTC\_CRx), the prescaler registers (RTC\_SPREx and RTC\_APRE), the wakeup timer registers (RTC\_WUTRx) and the alarm A registers (RTC\_ALRMASSRx and RTC\_ALRMARx) are reset to their default values by Power-on reset only. They are not affected by a system reset.

In addition, as soon as an RTC clock is enabled in the CLK\_CRTCR clock controller register, the RTC continues to run under reset (if the reset source is other than the Power-on reset).

When power-on reset occurs, the RTC is stopped and all its registers are set to their reset values.

### 24.3.8 RTC synchronization (low, medium+ and high density devices only)

On low, medium+ and high density devices, the RTC can be synchronized to a remote clock with a high degree of precision. After reading the subsecond field (RTC\_SSRx), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be finely adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using the RTC\_SHIFTRx register.

Reading the RTC\_SSRx registers gives the value of the synchronous prescaler counter. This allows to calculate the exact RTC time down to a resolution of  $1/(PREDIV\_S + 1)$

seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV\_S[14:0]. The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV\_S set to 0x7FFF.

However, increasing the synchronous prescaler value means that the asynchronous prescaler value (PREDIV\_A[6:0]) must be decreased to maintain the synchronous prescaler output at 1 Hz. As a result, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption.

The RTC can be finely adjusted using the RTC shift control register (RTC\_SHIFTR). Writing to RTC\_SHIFTR can shift (either delay or advance) the clock by up to a second with a resolution of  $1 / (\text{PREDIV\_S} + 1)$  seconds.

The shift operation consists in adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock. If at the same time ADD1S is set, this adds one second and at the same time subtracts a fraction of second, so this advances the clock.

**Caution:** Before initiating a shift operation, the user must check that SS[15] = 0 in order to ensure that no overflow will occur.

As soon as a shift operation is initiated by a write to the RTC\_SHIFTRL register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

#### 24.3.9 RTC smooth digital calibration (low, medium+ and high density devices only)

On low, medium+ and high density devices, the RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual RTCCLK clock cycles). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short periods of time.

The calibration is performed during a cycle of about  $2^{20}$  RTCCLK clock cycles, or 32 seconds when the input frequency is 32768 Hz.

The calibration registers (RTC\_CALRx) specify the number of RTCCLK clock cycles to be masked during the 32-second cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the 32-second cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting SMC[2] to 1 causes four additional cycles to be masked,
- and so on up to SMC[8] set to 1 which causes 256 clocks to be masked.

While the CALM bits allow to reduce the RTC frequency by up to 487.1 ppm with a fine resolution, the bit CALP of the RTC\_CALRH register can be used to increase the frequency by 488.5 ppm. Setting this bit 1 effectively inserts an additional RTCCLK clock cycle every  $2^{11}$  RTCCLK cycles, which means that 512 clocks are added during every 32-second cycle.

Using the CALM bits in conjunction with CALP, an offset ranging from -511 to 512 RTCCLK cycles can be added during each 32-second cycle. This corresponds to a calibration range of -487.1 to 488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency ( $f_{CAL}$ ) given the input frequency ( $f_{RTCCLK}$ ) is as follows:

$$f_{CAL} = f_{RTCCLK} \times (1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512))$$

### Calibration when PREDIV\_A<3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV\_A bits in RTC\_APRES register) is less than 3. If CALP was already set to 1 and PREDIV\_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PREDIV\_A less than 3, the synchronous prescaler value (PREDIV\_S) should be reduced so that each second is accelerated by 8 RTCCLK clock cycles, which is equivalent to adding 256 clock cycles every 32 seconds. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from -243.3 to 244.1 ppm) can effectively be added during each 32-second cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV\_A equals 1 (division factor of 2), PREDIV\_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV\_A equals 0, PREDIV\_S should be set to 32759 rather than 32767 (8 less).

If PREDIV\_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$f_{CAL} = f_{RTCCLK} \times (1 + (256 - CALM) / (2^{20} + CALM - 256))$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

### Verifying the RTC calibration

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. However, for certain applications, standards require that the RTC precision be measured and verified on each device. An optional 1 Hz output is provided on low, medium+ and high density devices to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period. However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.  
Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).
- Alternatively, the CALW16 bit of the RTC\_CALRH register can be set to 1 to force a 16-second calibration cycle period.  
In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the

calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] and CALW16 can not be set to 1 simultaneously.

- To reduce the calibration cycle to 8 seconds, the CALW8 bit of the RTC\_CALRH register can be set to 1. In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

### Re-calibration on-the-fly

The calibration registers (RTC\_CALRH/RTC\_CALRL) can be updated on-the-fly while RTC\_ISR1/INITF=0, by using the follow process:

1. Poll the RTC\_ISR1/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC\_CALRH, if necessary.
3. Write to the RTC\_CALRL register. RECALPF is then automatically set to 1.
4. Within three ck\_apre cycles after the write operation to RTC\_CALRL, the new calibration settings take effect.

## 24.3.10 Tamper detection (low, medium+ and high density devices only)

There are three tamper detection inputs. Each one is associated with a flag TAMP1F/TAMP2F/ TAMP3F in the RTC\_ISR2 register. Each input can be enabled by setting the corresponding TAMP1E/TAMP2E/TAMP3E bits to 1. The tamper detection control bits are available in the RTC\_TCR1 and RTC\_TCR2 registers.

By enabling tamper detection, the internal GPIO Schmitt trigger of the selected tamper input is disabled in order to avoid additional power consumption.

By setting the TAMPIE bit in the RTC\_TCR1 register, an interrupt is generated when a tamper detection event occurs. A tamper detection event is generated when either 2, 4, or 8 consecutive samples (depending on TAMPFLT bit) are observed at the level configured through the TAMPTRG bit.

The TAMPER inputs are pre-charged through the I/O internal resistance before its state is sampled, unless the pull-up is disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the tamper inputs.

The trade-off between tamper detection latency and any power consumption through the weak pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

*Note:*

*Refer to the datasheets for the electrical characteristics of the pull-up resistors.*

## 24.3.11 Calibration clock output

When the COE bit is set to 1 in the RTC\_CR3 register, a reference clock is provided on the RTC\_CALIB device output. If the COSEL bit (bit 3 in the RTC\_CR3 register) is reset and PREDIV\_A = 0x7F, the RTC\_CALIB frequency is  $f_{RTCCLK}/64$ . This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz.

*Note:*

*The RTC\_CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.*

On low, medium+ and high density devices, if COSEL is set and “PREDIV\_S+1” is a non-zero multiple of 256 (i.e: PREDIV\_S[7:0] = 0xFF), RTC\_CALIB frequency is  $f_{RTCCLK}/(256 * (PREDIV_A+1))$ . This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV\_A = 0x7F, PREDIV\_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

### 24.3.12 Alarm output

Two functions can be selected on Alarm output: ALRAF, and WUTF. They reflect the content of the corresponding flag registers. The OSEL[1:0] control bits in the RTC\_CR3 register are used to activate and select the function which is output on RTC\_ALARM. The polarity of the output is determined by the POL control bit in RTC\_CR3: the opposite of the selected flag bit is output on RTC\_ALARM when the POL bit is set to 1.

## 24.4 RTC low power modes

**Table 78. Effect of low power modes on RTC**

Mode	Description
Low power wait/Wait	No effect RTC interrupts cause the device to exit from Wait mode.
Active halt/Halt	The RTC remains active if its clock source is LSE or LSI. RTC interrupts can exit the device from Active-halt/Halt mode: alarm A interrupt, periodic wakeup interrupt and the 3 tamper interrupts (see <a href="#">Section 24.5: RTC interrupts</a> ).

- Note:
- 1 *The “Active-halt” mode is a low power mode where the CPU and the peripheral clocks are stopped, except the RTC. It is described in [Section 7: Low power modes on page 72](#).*
  - 2 *The RTC is designed to have a reduced consumption when clocked by LSE/LSI (refer to the Electrical characteristics section in the device datasheet for more details).*

## 24.5 RTC interrupts

The alarm A interrupt, the periodic wakeup interrupt and the 3 tamper interrupts share the same interrupt vector.

**Table 79. Interrupt control bits**

Interrupt event	Event flag	Enable control bit	Exit from Wait	Exit from Halt
Alarm A	ALRAF	ALRAIE	Yes	Yes <sup>(1)</sup>
Periodic wakeup (wakeup)	WUTF	WUTIE	Yes	Yes <sup>(1)</sup>
Tamper 1 event	TAMP1F	TAMPIE (RTC_TCR1)	Yes	Yes <sup>(1)</sup>
Tamper 2 event	TAMP2F		Yes	Yes <sup>(1)</sup>
Tamper 3 event	TAMP3F		Yes	Yes <sup>(1)</sup>

1. Wakeup from Halt is possible only when the RTC clock source is LSE or LSI.

## 24.6 RTC registers

Refer to [Section 5.1 on page 63](#) for a list of abbreviations used in register descriptions.

### 24.6.1 Time register 1 (RTC\_TR1)

RTC\_TR1 is a calendar time shadow register. This register can be written in initialization mode. Refer to [Section : Calendar initialization and configuration on page 416](#) and [Section 24.3.6: Reading the calendar on page 417](#).

Address offset: 0x00

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved	ST[2:0]				SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 Reserved, always read as 0.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

**Note:** *This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).*

### 24.6.2 Time register 2 (RTC\_TR2)

RTC\_TR2 is a calendar time shadow register. This register can be written in initialization mode. Refer to [Section : Calendar initialization and configuration on page 416](#) and [Section 24.3.6: Reading the calendar on page 417](#).

Address offset: 0x01

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved	MNT[2:0]				MNU[3:0]			
	rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 Reserved, always read as 0.

Bits 6:4 **MNT[2:0]**: Minute tens in BCD format.

Bits 3:0 **MNU[3:0]**: Minute units in BCD format.

**Note:** *This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).*

### 24.6.3 Time register 3 (RTC\_TR3)

RTC\_TR3 is a calendar time shadow register. This register can be written in initialization mode. Refer to [Section : Calendar initialization and configuration on page 416](#) and [Section 24.3.6: Reading the calendar on page 417](#).

Address offset: 0x02

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved	PM	HT[1:0]		HU[3:0]				
	rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 Reserved, always read as 0.

Bit 6 **PM**: AM/PM notation

0: AM or 24-hour format

1: PM

Bits 5:4 **HT[1:0]**: Hour tens in BCD format.

Bits 3:0 **HU[3:0]**: Hour units in BCD format.

**Note:** *This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).*

#### 24.6.4 Date register 1 (RTC\_DR1)

RTC\_DR1 is a calendar date shadow register. This register can be written in initialization mode. Refer to [Section : Calendar initialization and configuration on page 416](#) and [Section 24.3.6: Reading the calendar on page 417](#).

Address offset: 0x04

Reset value: 0x01

	7	6	5	4	3	2	1	0
Reserved	DT[1:0]			DU[3:0]				
	rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:6 Reserved, always read as 0.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

**Note:** *This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).*

#### 24.6.5 Date register 2 (RTC\_DR2)

RTC\_DR2 is a calendar date shadow register. This register can be written in initialization mode. Refer to [Section : Calendar initialization and configuration on page 416](#) and [Section 24.3.6: Reading the calendar on page 417](#).

Address offset: 0x05

Reset value: 0x21

	7	6	5	4	3	2	1	0	
	WDU[2:0]			MT	MU[3:0]				
	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 7:5 **WDU[2:0]**: Week day units

000: forbidden.

001: Monday

...

111: Sunday

Bit 4 **MT**: Month tens in BCD format

Bits 3:0 **MU[3:0]**: Month units in BCD format

**Note:** *This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).*

#### 24.6.6 Date register 3 (RTC\_DR3)

RTC\_DR3 is a calendar date shadow register. This register can be written in initialization mode. Refer to [Section : Calendar initialization and configuration on page 416](#) and [Section 24.3.6: Reading the calendar on page 417](#).

Address offset: 0x06

Reset value: 0x00

7	6	5	4	3	2	1	0
YT[3:0]				YU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:4 **YT[3:0]**: Year tens in BCD format.

Bits 3:0 **YU[3:0]**: Year units in BCD format.

**Note:** *This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).*

#### 24.6.7 Subsecond register high (RTC\_SSRH)

The RTC\_SSRH is available only on low, medium+ and high density devices. Refer to [Section 24.3.6: Reading the calendar on page 417](#) for read procedures on this register.

Address offset: 0x17

Reset value: 0x00

7	6	5	4	3	2	1	0
SS[15:8]							
r	r	r	r	r	r	r	r

Bits 7:0 **SS[15:8]**: Sub second value upper bits

See RTC\_SSRL for details.

**Note:** *This register is write protected. The write access procedure is described in [Section : RTC register write protection on page 416](#)*

### 24.6.8 Subsecond register low (RTC\_SSRL)

The RTC\_SSRL is available only on low, medium+ and high density devices. Refer to [Section 24.3.6: Reading the calendar on page 417](#) for read procedures on this register.

Address offset: 0x18

Reset value: 0x00

7	6	5	4	3	2	1	0
SS[7:0]							
r	r	r	r	r	r	r	r

Bits 7:0 **SS[7:0]**: Sub second value lower bits

SS[15:0] (in RTC\_SSRRH and RTC\_SSRL registers) is the value of the synchronous prescaler's counter. The fraction of second is given by the formula below:

$$\text{Second fraction} = (\text{RTC_SPRE} - \text{SS}) / (\text{RTC_SPRE} + 1)$$

Note: SS[15:0] can be larger than RTC\_SPRE only after a shift operation. In this case, the correct time/date is one second less than indicated by RTC\_TRx/RTC\_DRx.

**Note:** *This register is write protected. The write access procedure is described in [Section : RTC register write protection on page 416](#)*

### 24.6.9 Control register 1 (RTC\_CR1)

Address offset: 0x08

Power-on reset value: 0x00

Reset value: 0XX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
Reserved	FMT	RATIO	BYPSHAD	Reserved	WUCKSEL[2:0]		
	rw	rw	rw		rw	rw	rw

Bit 7 Reserved, always read as 0.

Bit 6 **FMT**: Hour format

0: 24 hour/day format

1: AM/PM hour format

Bit 5 **RATIO**: System clock (SYSCLK) versus RTCCLK ratio

0:  $f_{SYSCLK} \geq 2 \times f_{RTCCLK}$  must be respected

1:  $f_{SYSCLK} = f_{RTCCLK}$

**Caution:** The case where  $f_{SYSCLK}$  is less than  $2 f_{RTCCLK}$  and not equal to  $f_{RTCCLK}$  is forbidden.

In order to perform a read access to the calendar registers (RTC\_TRx, RTC\_DRx and RTC\_SSRx),  $f_{SYSCLK}$  must be  $\geq 4 \times f_{RTCCLK}$  when RATIO=0.

**Bit 4 BYPShAD:** Bypass the shadow registers

This bit is available on low, medium+ and high density devices only.

0: Calendar values (when reading from RTC\_SSRx, RTC\_TRx, and RTC\_DRx) are taken from the shadow registers, which are updated once every RTCCLK cycle.

1: Calendar values (when reading from RTC\_SSRx, RTC\_TRx, and RTC\_DRx) are taken directly from the calendar counters.

**Bit 3 Reserved, always read as 0.**

**Bits 2:0 WUCKSEL[2:0]:** Wakeup clock selection

000: RTCCLK/16 clock is selected

001: RTCCLK/8 clock is selected

010: RTCCLK/4 clock is selected

011: RTCCLK/2 clock is selected

10x: ck\_spre (usually 1 Hz) clock is selected

11x: ck\_spre (usually 1 Hz) clock is selected and  $2^{16}$  is added to the WUT counter value (refer to the note below).

**Note:** *Wakeup unit counter = WUT counter value (from 0x0000 up to 0xFFFF) + 0x10000 (added value when WUCKSEL[2:1] = "11").*

*Bit 6 of this register can be written in initialization mode only when INITF bit is set to 1 in RTC\_ISR1.*

*Bits 2 to 0 of this register can be written only when WUTE bit is set to 0 in RTC\_CR2 and WUTWF is set to 1 in RTC\_ISR1.*

*This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).*

#### 24.6.10 Control register 2 (RTC\_CR2)

Address offset: 0x09

Power-on reset value: 0x00

Reset value: 0XX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
Reserved	WUTIE	Reserved	ALRAIE	Reserved	WUTE	Reserved	ALRAE
	rw		rw		rw		rw

**Bit 7 Reserved, always read as 0.**

**Bit 6 WUTIE:** Wakeup timer interrupt enable

0: Wakeup timer Interrupt disable

1: Wakeup Timer Interrupt enable

**Bit 5 Reserved, always read as 0.**

**Bit 4 ALRAIE:** Alarm A interrupt enable

0: Alarm A Interrupt disable

1: Alarm A Interrupt enable

**Bit 3 Reserved, always read as 0.**

Bit 2 **WUTE**: Wakeup timer enable

- 0: Wakeup timer disable
- 1: Wakeup timer enable

Bit 1 Reserved, always read as 0.

Bit 0 **ALRAE**: Alarm A enable

- 0: Alarm A disable
- 1: Alarm A enable

*Note:* This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).

#### 24.6.11 Control register 3 (RTC\_CR3)

Address offset: 0x0A

Power-on reset value: 0x00

Reset value: 0XX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
COE	OSEL[1:0]		POL	COSEL	BCK	SUB1H	ADD1H
rw	rw	rw	rw	rw	rw	w	w

Bit 7 **COE** Calibration output enable

This bit enables the RTC\_CALIB device output

- 0: Calibration output disable
- 1: Calibration output enable

Bits 6:5 **OSEL[1:0]**: Output selection

These bits are used to select the flag to be routed to RTC\_ALARM output

- 00: Output disable
- 01: Alarm A output enable
- 10: Reserved
- 11: Wakeup output enable

Bit 4 **POL**: Output polarity

This bit is used to configure the polarity of RTC\_ALARM device output

- 0: The RTC\_ALARM pin is high when ALRAF/WUTF is asserted (depending on OSEL)
- 1: The RTC\_ALARM pin is low when ALRAF/WUTF is asserted (depending on OSEL).

Bit 3 **COSEL**: Calibration output selection

This bit is available on low, medium+ and high density devices only.

When COE=1, this bit selects which signal is output on RTC\_CALIB.

- 0: Calibration output is 512 Hz
- 1: Calibration output is 1 Hz

*Note:* These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV\_A=127 and PREDIV\_S=255). Refer to [Section 24.3.11: Calibration clock output](#) for more details.

Bit 2 **BCK**: Backup

This bit can be written by the software to memorize whether the daylight saving time change has been performed or not.

**Bit 1 SUB1H:** Subtract 1 hour (winter time change)

When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.

Setting this bit has no effect when current hour is 0.

0: No effect

1: 1 hour subtracted to the current time. This can be used for winter time change.

**Bit 0 ADD1H:** Add 1 hour (summer time change)

When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.

0: No effect

1: adds 1 hour to the current time. This can be used for summer time change.

*Note:* ADD1H or SUB1H action must not change the day.

*It is recommended not to change the hour during the last seconds before the calendar hour is incremented, as this could mask the calendar hour increment.*

*This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).*

## 24.6.12 Initialization and status register 1 (RTC\_ISR1)

Address offset: 0x0C

Reset value: 0x00 on medium density devices, 0x01 for low, medium+ and high density devices

7	6	5	4	3	2	1	0
INIT	INITF	RSF	INITS	SHPF	WUTWF	RECALPF	ALRAWF
rw	r	rc_w0	r	r	r	r	r

**Bit 7 INIT:** Initialization mode.

0: Free running mode

1: Initialization mode used to program time and date registers (RTC\_TRx and RTC\_DRx) and the prescaler registers. Counters are stopped and start counting from the new value when the INIT bit is reset.

**Bit 6 INITF:** Initialization flag

When this bit is set to 1, the RTC is in initialization mode and the time, date and prescaler registers can be updated.

0: calendar registers update is not allowed

1: calendar registers update is allowed.

**Bit 5 RSF:** Registers synchronization flag

This bit is used when BYPSHAD=0. It is set by hardware each time the calendar registers are copied into the shadow registers (RTC\_SSRx, RTC\_TRx and RTC\_DRx). It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not synchronized yet

1: Calendar shadow registers synchronized

**Bit 4 INIT5:** Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Power-on reset status).

- 0: Calendar has not been initialized.
- 1: Calendar has been initialized.

**Bit 3 SHPF:** Shift operation pending

This bit is available on low, medium+ and high density devices only.

- 0: No shift operation is pending
- 1: A shift operation is pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC\_SHIFTRL. It is cleared by hardware as soon as the corresponding shift operation has completed.

Writing to SHPF has no effect.

**Bit 2 WUTWF:** Wakeup timer write flag

This bit is set by hardware when the wakeup timer values can be changed, after the WUTE bit has been set to 0 in RTC\_CR2.

- 0: Wakeup timer update not allowed.
- 1: Wakeup timer update allowed.

**Bit 1 RECALPF:** Recalibration pending Flag

This bit is available on low, medium+ and high density devices only.

The status flag RECALPF is automatically set to 1 when the software writes to the register RTC\_CALRL, indicating that the RTC\_CALRH/RTC\_CALRL registers are blocked. When the new calibration settings are taken into account, this bit returns to 0. See [Section : Re-calibration on-the-fly](#).

**Bit 0 ALRAWF:** Alarm A write flag

This bit is set by hardware when the alarm A values can be changed, after the ALRAE bit has been set to 0 in RTC\_CR2. The ALRAWF bit is always set for low, medium+ and high density devices, though the ALRAE must still be cleared before changing alarm A values

- 0: Alarm A update not allowed.
- 1: Alarm A update allowed.

**Note:** *This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).*

### 24.6.13 Initialization and status register 2 (RTC\_ISR2)

Address offset: 0x0D

Power-on reset value: 0x00

Reset value: 0XX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
TAMP3F	TAMP2F	TAMP1F		Reserved	WUTF	Reserved	ALRAF
rc_w0	rc_w0	rc_w0			rc_w0		rc_w0

**Bit 7 TAMP3F:** TAMPER3 detection flag

This bit is available on low, medium+ and high density devices only.

This flag is set by hardware when a tamper detection event is detected on tamper input 3.

It is cleared by software writing 0.

Bit 6 **TAMP2F**: TAMPER2 detection flag

This bit is available on low, medium+ and high density devices only.

This flag is set by hardware when a tamper detection event is detected on tamper input 2.

It is cleared by software writing 0.

Bit 5 **TAMP1F**: TAMPER1 detection flag

This bit is available on low, medium+ and high density devices only.

This flag is set by hardware when a tamper detection event is detected on tamper input 1.

It is cleared by software writing 0.

Bits 4:3 Reserved, always read as 0.

Bit 2 **WUTF**: Periodic wakeup flag

This flag is set by hardware when the wakeup auto-reload down-counter reaches 0.

It is cleared by software by writing 0.

*Note: This flag must be cleared by software at least 1.5 RTCCLK cycles before WUTF is set to 1 again.*

Bit 1 Reserved, always read as 0.

Bit 0 **ALRAF**: Alarm A Flag

This flag is set by hardware when the time and date registers (RTC\_TRx and RTC\_DRx) match the RTC\_ALRMARx registers. On low, medium+ and high density devices, to set this flag, the sub-second registers (RTC\_SSRx) must also match the RTC\_ALRMASSRx registers.

This flag is cleared by software writing 0.

*Note: These flags are cleared around 2 CK clock cycles after writing 0 into them.*

#### 24.6.14 Synchronous prescaler register high (RTC\_SPRERH)

Address offset: 0x10

Power-on reset value: 0x00

Reset value: 0xXX (this register is not impacted by a system reset. It is reset at power-on).

	7	6	5	4	3	2	1	0
Reserved	PREDIV_S[14:8]							
	rw	rw	rw	rw	rw	rw	rw	rw

Bit7 Reserved, always read as 0.

Bits 6:0 **PREDIV\_S[14:8]**: Synchronous prescaler factor MSB

This register contains the MSB of the synchronous division factor:  $\text{ck\_spre frequency} = \text{ck\_apre frequency} / (\text{PREDIV\_S} + 1)$ .

*Note: PREDIV\_S[14:13] are available only on low, medium+ and high density devices.*

*Note: This register can be written in initialization mode only when INITF bit is set to 1 in RTC\_ISR1.*

*This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).*

### 24.6.15 Synchronous prescaler register low (RTC\_SPRERL)

Address offset: 0x11

Power-on reset value: 0xFF

Reset value: 0XX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
PREDIV_S[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

**PREDIV\_S[7:0]: Synchronous prescaler factor LSB**

Bits 7:0 This register contains the LSB of synchronous division factor:  $\text{ck_spre frequency} = \text{ck_apre frequency} / (\text{PREDIV\_S} + 1)$

**Note:** *This register can be written in initialization mode only when INITF bit is set to 1 in RTC\_ISR1.*

*This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).*

### 24.6.16 Asynchronous prescaler register (RTC\_APRLR)

Address offset: 0x12

Power-on reset value: 0x7F

Reset value: 0XX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
PREDIV_A[6:0]							
Reserved	rw						

Bit 7 Reserved, always read as 0.

Bits 6:0 **PREDIV\_A[6:0]: Asynchronous prescaler factor**

This is the asynchronous division factor:

$\text{ck_apre frequency} = f_{\text{RTCCLK}} \text{ frequency} / (\text{PREDIV\_A} + 1)$

**Note:** *PREDIV\_A[6:0] = 000000 is a forbidden value on medium density devices.*

**Note:** *This register can be written in initialization mode only when INITF bit is set to 1 in RTC\_ISR1.*

*This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).*

### 24.6.17 Wakeup timer register high (RTC\_WUTRH)

Address offset: 0x14

Power-on reset value: 0xFF

Reset value: 0xXX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
WUT[15:8]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **WUT[15:8]**: Wakeup MSB

When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every  $(WUT[15:0]+1) \cdot f_{CK_WUT}$  cycle. The period of  $f_{CK_WUT}$  is selected through WUCKSEL[2:0] bits of RTC\_CR1 register.

*Note: The first assertion of the WUTF occurs  $(WUT+1) \cdot f_{CK_WUT}$  cycle  $\pm 1$  after WUTE is set.  
Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0]=011( $f_{RTCCLK}/2$ ) is forbidden.*

*Note:* This register can be written only when WUTE bit is set to 0 in RTC\_CR2, and WUTWF to 1 in RTC\_ISR1.

This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).

### 24.6.18 Wakeup timer register low (RTC\_WUTRL)

Address offset: 0x15

Power-on reset value: 0xFF

Reset value: 0xXX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
WUT[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **WUT[7:0]**: Wakeup LSB

When wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every  $(WUT[15:0]+1) \cdot f_{CK_WUT}$  cycle. The period of  $f_{CK_WUT}$  is selected through WUCKSEL[2:0] bits of RTC\_CR1 register.

*Note: The first assertion of the WUTF occurs  $(WUT+1) \cdot f_{CK_WUT}$  cycle  $\pm 1$  after WUTE is set.  
Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0]=011( $f_{RTCCLK}/2$ ) is forbidden.*

*Note:* This register can be written only when WUTE bit is set to 0 in RTC\_CR2, and WUTWF to 1 in RTC\_ISR1.

This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).

### 24.6.19 Write protection register (RTC\_WPR)

Address offset: 0x19

Reset value: 0x00

7	6	5	4	3	2	1	0
KEY[7:0]							
w	w	w	w	w	w	w	w

Bits 7:0 **KEY[7:0]**: Write Protection Key

This byte is written by software.

Reading this register returns 0x00.

Refer to [RTC register write protection](#) for a description of how to unlock RTC register write protection.

### 24.6.20 RTC shift control register high (RTC\_SHIFTRH)

The RTC\_SHIFTRH register is available only on low, medium+ and high density devices.

Address offset: 0x1A

Reset value: 0x00

7	6	5	4	3	2	1	0	
ADD1S		SUBFS[14:8]						
w	w	w	w	w	w	w	w	

Bit 7 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. The value written to this bit takes effect when RTC\_SHIFTRL is written. Writing to this bit has no effect if there is already a shift operation pending (when SHPF=1, in RTC\_ISR1).

This function is intended to be used with SUBFS (see description in RTC\_SHIFTRL) to effectively add a fraction of a second to the clock in an atomic operation.

Bits 6:0 **SUBFS[14:8]**: MSB to subtract a fraction of a second

This bit is write only and is always read as zero. Writing to these bits has no effect if there is already a shift operation pending (when SHPF=1, in RTC\_ISR1).

The value written to this register takes effect when RTC\_SHIFTRL is written. See description in RTC\_SHIFTRL.

**Note:** *This register is write protected. The write access procedure is described in [Section : RTC register write protection on page 416](#).*

### 24.6.21 RTC shift control register low (RTC\_SHIFTRL)

The RTC\_SHIFTRL register is available only on low, medium+ and high density devices.

Address offset: 0x1B

Reset value: 0x00

7	6	5	4	3	2	1	0
SUBFS[7:0]							
w	w	w	w	w	w	w	w

Bits 7:0 **SUBFS[7:0]**: LSB to subtract a fraction of a second

Before writing to this register, the values for ADD1S and SUBFS[14:8] must first be defined by writing to RTC\_SHIFTRH.

The value which is written to SUBFS[14:0] (included in RTC\_SHIFTRH and RTC\_SHIFTRL) is added to the synchronous prescaler counter. Since this counter counts down, this operation subtracts the delay from the clock, effectively delaying the clock by:

$$\text{Delay (seconds)} = \text{SUBFS} / (\text{PREDIV\_S} + 1)$$

A fraction of a second can be added to the clock when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

$$\text{Advance (seconds)} = (1 - (\text{SUBFS} / (\text{PREDIV\_S} + 1)))$$

See also [Section 24.3.8: RTC synchronization \(low, medium+ and high density devices only\)](#).

**Note:** *This register is write protected. The write access procedure is described in [Section : RTC register write protection on page 416](#).*

### 24.6.22 Alarm A register 1 (RTC\_ALRMAR1)

Address offset: 0x1C

Power-on reset value: 0x00

Reset value: 0xXX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
MSK1	ST[2:0]				SU[3:0]		
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **MSK1**: Alarm A Seconds mask

0: Alarm A set if the seconds match

1: Seconds don't care in alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bit 3:0 **SU[3:0]**: Second units in BCD format

**Note:** *This register can be written either in initialization mode, or when ALRAE bit is set to 0 in RTC\_CR2, and ALRAWF to 1 in RTC\_ISR1.*

*This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).*

### 24.6.23 Alarm A register 2 (RTC\_ALRMAR2)

Address offset: 0x1D

Power-on reset value: 0x00

Reset value: 0xXX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0	
MSK2	MNT[2:0]				MNU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	

Bit 7 **MSK2**: Alarm A minutes mask

0: Alarm A set if the minutes match.

1: Minutes don't care in alarm A comparison.

Bits 6:4 **MNT[2:0]**: Minute tens in BCD format.

Bits 3:0 **MNU[3:0]**: Minute units in BCD format.

**Note:** This register can be written either in initialization mode, or when ALRAE bit is set to 0 in RTC\_CR1, and ALRAWF to 1 in RTC\_ISR1.

This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).

### 24.6.24 Alarm A register 3 (RTC\_ALRMAR3)

Address offset: 0x1E

Power-on reset value: 0x00

Reset value: 0xXX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
MSK3	PM	ALHT[1:0]				HU[3:0]	
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **MSK3**: Alarm A hours mask

0: Alarm A set if the hours match.

1: Hours don't care in alarm A comparison.

Bit 6 **PM**: AM/PM notation

0: AM or 24-hour format.

1: PM

Bits 5:4 **HT[1:0]**: Hour tens in BCD format.

Bits 3:0 **HU[3:0]**: Hour units in BCD format.

**Note:** This register can be written either in initialization mode, or when ALRAE bit is set to 0 in RTC\_CR1, and ALRAWF to 1 in RTC\_ISR1.

This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).

### 24.6.25 Alarm A register 4 (RTC\_ALRMAR4)

Address offset: 0x1F

Power-on reset value: 0x00

Reset value: 0xXX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
MSK4	WDSEL	AL DT[1:0]		DU[3:0]/Day[2:0]			
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **MSK4**: Alarm A Date mask

0: Alarm A set if the Date/Day match.

1: Date/Day don't care in alarm A comparison.

Bit 6 **WDSEL**: Week day selection

0: DU[3:0] represents the date units.

1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 5:4 **DT[1:0]**: Date tens in BCD format.

Bits 3:0 **DU[3:0]**: Date units or Day in BCD format.

**Note:** This register can be written either in initialization mode, or when ALRAE bit is set to 0 in RTC\_CR1, and ALRAWF to 1 in RTC\_ISR1.

This register is write protected. The write access procedure is described in [Section : RTC register write protection](#).

### 24.6.26 Alarm A sub second register high (RTC\_ALRMASSRH)

The RTC\_ALRMASSRH register is available only on low, medium+ and high density devices.

Address offset: 0x24

Power-on reset value: 0x00

Reset value: 0xXX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
Reserved	ALSS[14:8]						
r	rw	rw	rw	rw	rw	rw	rw

Bit 7 Reserved

Bit 6:0 **ALSS[14:8]**: MSB of subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if Alarm A must be activated. Note that only bits 0 up MASKSS-1 (defined in RTC\_ALRMAMSKR register) are compared.

If this value is larger than PREDIV\_S (and if MASKSS is large enough), the alarm will never be activated.

**Note:** This register can be written only when RTC\_CR1/ALRAE is reset, or in initialization mode.

This register is write protected. The write access procedure is described in [Section : RTC register write protection on page 416](#)

### 24.6.27 Alarm A sub second register low (RTC\_ALRMASSRL)

The RTC\_ALRMASSRL register is available only on low, medium+ and high density devices.

Address offset: 0x25

Power-on reset value: 0x00

Reset value: 0XX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
ALSS[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **ALSS[7:0]:** LSB of sub seconds value

This value is compared with the contents of the synchronous prescaler's counter to determine if Alarm A is to be activated. Note that only bits 0 up MASKSS-1 (defined in RTC\_ALRMAMSKR) are compared.

If this value is larger than PREDIV\_S (and if MASKSS is large enough), the alarm will never be activated.

**Note:** This register can be written only when RTC\_CR1/ALRAE is reset, or in initialization mode. This register is write protected. The write access procedure is described in [Section : RTC register write protection on page 416](#)

### 24.6.28 Alarm A subsecond masking register (RTC\_ALRMASSMSKR)

The RTC\_ALRMASSMSKR register is available only on low, medium+ and high density devices.

Address offset: 0x26

Power-on Reset value: 0x00

Reset value: 0XX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
Reserved				MASKSS[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:4 Reserved

Bits 3:0 **MASKSS[3:0]:** Mask the most significant bits starting at this bit

0: no comparison on subseconds for Alarm A. The alarm is activated when seconds unit is incremented (assuming that the rest of the fields match).

1: SS[14:1] are don't care in Alarm A comparison. Only SS[0] is compared.

2: SS[14:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.

3: SS[14:3] are don't care in Alarm A comparison. Only SS[2:0] are compared.

...

12: SS[14:12] are don't care in Alarm A comparison. SS[11:0] are compared.

13: SS[14:13] are don't care in Alarm A comparison. SS[12:0] are compared.

14: SS[14] is don't care in Alarm A comparison. SS[13:0] are compared.

15: SS[14:0] are compared and must match to activate alarm.

**Note:** The overflow bit of the synchronous counter (bit 15) is never compared. This bit can be different from 0 only immediately after a shift operation.

**Note:** This register can be written only when RTC\_CR0/ALRAE is reset, or in initialization mode.  
This register is write protected. The write access procedure is described in [Section : RTC register write protection on page 416](#)

## 24.6.29 Calibration register high (RTC\_CALRH)

The RTC\_CALRH register is available only on low, medium+ and high density devices.

Address offset: 0x2A

Power-on Reset value: 0x00

Reset value: 0XX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
CALP	CALW8	CALW16		Reserved			CALM[8]
rw	rw	rw					rw

Bit 7 **CALP:** Increase of RTC frequency by 488.5 ppm

0: No RTCCLK pulses are added.

1: One RTCCLK pulse is effectively inserted every  $2^{11}$  pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. If the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows:  $(512 * \text{CALP}) - \text{CALM}$ .

Refer to [Section 24.3.9: RTC smooth digital calibration \(low, medium+ and high density devices only\)](#).

Bit 6 **CALW8:** Use an 8-second calibration cycle period

When CALW8 is set to '1', the 8-second calibration cycle period is selected.

*Note:* CALM[1:0] are stucked at "00" when CALW8='1'.

Refer to [Section 24.3.9: RTC smooth digital calibration \(low, medium+ and high density devices only\)](#).

Bit 5 **CALW16:** Use a 16-second calibration cycle period

When CALW16 is set to '1', the 16-second calibration cycle period is selected. This bit must not be set to '1' if CALWU8=1.

*Note:* CALM[0] is stucked at '0' when CALW16='1'.

Refer to [Section 24.3.9: RTC smooth digital calibration \(low, medium+ and high density devices only\)](#).

Bits 4:1 Reserved

Bit 0 **CALM[8]:** MSB of calibration minus

See description in RTC\_CALRL.

**Note:** This register can not be written when RTC\_ISR1/RECALPF is set to 1 (see [Section : Recalibration on-the-fly](#)).

This register is write protected. The write access procedure is described in [Section : RTC register write protection on page 416](#)

### 24.6.30 Calibration register low (RTC\_CALRL)

The RTC\_CALRL register is available only on low, medium+ and high density devices.

Address offset: 0x2B

Power-on Reset value: 0x00

Reset value: 0XX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
CALM[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

**CALM[7:0]:** LSB of calibration minus

The frequency of the calendar is reduced by masking CALM out of  $2^{20}$  RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

Bits 7:0

To increase the frequency of the calendar, this feature should be used in conjunction with CALP.

See [Section 24.3.9: RTC smooth digital calibration \(low, medium+ and high density devices only\)](#).

**Note:** This register can not be written when RTC\_ISR1/RECALPF is 1 (see [Section : Re-calibration on-the-fly](#)).

This register is write protected. The write access procedure is described in [Section : RTC register write protection on page 416](#).

### 24.6.31 Tamper control register 1 (RTC\_TCR1)

The RTC\_TCR1 register is available only on low, medium+ and high density devices.

Address offset: 0x2C

Power-on Reset value: 0x00

Reset value: 0XX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
Reserved	TAMP3TRG	TAMP3E	TAMP2TRG	TAMP2E	TAMP1TRG	TAMP1E	TAMPIE
	rw	rw	rw	rw	rw	rw	rw

Bit 7 Reserved.

Bit 6 **TAMP3TRG:** Active level for tamper input 3

0: TAMPER3 low level triggers a tamper detection event.

1: TAMPER3 high level triggers a tamper detection event.

Bit 5 **TAMP3E:** Tamper detection enable for tamper input 3 (TAMPER3)

When TAMP3E is set, tamper detection is activated for tamper input 3, and a tamper detection event causes a tamper detection interrupt when enabled.

0: Tamper detection disabled on tamper input 3

1: Tamper detection enabled on tamper input 3

- Bit 4 **TAMP2TRG**: Active level for tamper input 2  
 0: TAMPER2 low level triggers a tamper detection event.  
 1: TAMPER2 high level triggers a tamper detection event.
- Bit 3 **TAMP2E**: Tamper detection enable for tamper input 2 (TAMPER2)  
 When TAMP2E is set, tamper detection is activated for tamper input 2, and a tamper detection event causes a tamper detection interrupt when enabled.  
 0: Tamper detection disabled on tamper input 2  
 1: Tamper detection enabled on tamper input 2
- Bit 2 **TAMP1TRG**: Active level for tamper 1  
 0: TAMPER1 low level triggers a tamper detection event.  
 1: TAMPER1 high level triggers a tamper detection event.
- Bit 1 **TAMP1E**: Tamper detection enable for tamper input 1 (TAMPER1)  
 When TAMP1E is set, tamper detection is activated for tamper input 1 and a tamper detection event causes a tamper detection interrupt when enabled.  
 0: Tamper detection disabled on tamper input 1  
 1: Tamper detection enabled on tamper input 1
- Bit 0 **TAMPIE**: Tamper interrupt enable  
 When TAMPIE is set, a tamper detection event generates an interrupt.  
 0: Tamper interrupt disabled  
 1: Tamper interrupt enabled

*Note:* This register is write protected. The write access procedure is described in [Section : RTC register write protection on page 416](#).

#### 24.6.32 Tamper control register 2 (RTC\_TCR2)

The RTC\_TCR2 register is available only on low, medium+ and high density devices.

Address offset: 0x2D

Power-on Reset value: 0x00

Reset value: 0XX (this register is not impacted by a system reset. It is reset at power-on).

7	6	5	4	3	2	1	0
TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw

- Bit 7 **TAMPPUDIS**: TAMPER pull-up disable

This bit determines if each of the tamper pins are precharged before each sample.

0: Tamper pins precharged before sampling (pull-up)

1: Precharge disabled on tamper pins

- Bits 6:5 **TAMPPRCH[1:0]**: Tamper precharge duration

This bit determines the time during which the weak pull-up is activated before each sample.

TAMPPRCH[1:0] are valid for each tamper input. This field has no meaning when TAMPPUDIS=1.

0x0: 1 RTCCLK clock cycle

0x1: 2 RTCCLK clock cycles

0x2: 4 RTCCLK clock cycles

0x3: 8 RTCCLK clock cycles

Bits 4:3 **TAMPFLT[1:0]**: Tamper filter count

This bit determines the number of consecutive samples at the level specified by TAMPxTRG necessary to activate a tamper event. TAMPFLT is valid for each tamper input.

0x0: Tamper is activated after 1 sample at the active level.

0x1: Tamper is activated after 2 consecutive samples at the active level.

0x2: Tamper is activated after 4 consecutive samples at the active level.

0x3: Tamper is activated after 8 consecutive samples at the active level.

Bits 2:0 **TAMPFREQ[2:0]**: Tamper sampling frequency

This bit determines the frequency at which each tamper input is sampled.

0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)

0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)

0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)

0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)

0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)

0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)

0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)

0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

**Note:** This register is write protected. The write access procedure is described in [Section : RTC register write protection on page 416](#).

### 24.6.33 RTC register map and reset values

**Table 80. RTC register map and reset values**

Add. offset	Register name	7	6	5	4	3	2	1	0			
0x00	RTC_TR1 Reset value <sup>(1)</sup>	0	ST[2:0] 000				SU[3:0] 0000					
0x01	RTC_TR2 Reset value <sup>(1)</sup>	0	MNT[2:0] 000				MNU[3:0] 0000					
0x02	RTC_TR3 Reset value <sup>(1)</sup>	0	PM 0	HT[1:0] 00		HU[3:0] 0000						
0x03	Reserved											
0x04	RTC_DR1 Reset value <sup>(1)</sup>	0	0	DT[1:0] 00		DU[3:0] 0001						
0x05	RTC_DR2 Reset value <sup>(1)</sup>	WDU[2:0] 001			MT 0	MU[3:0] 0001						
0x06	RTC_DR3 Reset value <sup>(1)</sup>	YT[3:0] 0000				YU[3:0] 0000						
0x07	Reserved											
0x08	RTC_CR1 Power-on Reset value <sup>(1)</sup>	0	FMT 0	RATIO 0	BYPSHAD <sup>(3)</sup> 0	0	WUCKSEL[2:0] 000					
0x09	RTC_CR2 Power-on Reset value <sup>(1)</sup>	0	WUTIE 0	0	ALRAIE 0	0	WUTE 0	0	ALRAE 0			
0x0A	RTC_CR3 Power-on Reset value <sup>(1)</sup>	COE 0	OSEL[1:0] 00		POL 0	COSEL <sup>(3)</sup> 0	BCK 0	SUB1H 0	ADD1H 0			
0x0B	Reserved											

**Table 80. RTC register map and reset values (continued)**

Add. offset	Register name	7	6	5	4	3	2	1	0												
0x0C	RTC_ISR1 Reset value <sup>(1)</sup>	INIT 0	INITF 0	RSF 0	INITS 0	SHPF <sup>(3)</sup> 0	WUTWF 0	RECALPF <sup>(3)</sup> 0	ALRAWF <sup>(2)</sup> 1												
0x0D	RTC_ISR2 Reset value <sup>(1)</sup>	TAMP3F <sup>(3)</sup> 0	TAMP2F <sup>(3)</sup> 0	TAMP1F <sup>(3)</sup> 0	- 0	- 0	WUTF 0	- 0	ALRAF 0												
0x0E-0x0F	Reserved																				
0x10	RTC_SPRERH Power-on Reset value <sup>(1)</sup>	- 0	PREDIV_S[14:13] <sup>(3)</sup> 00		PREDIV_S[12:8] 00000																
0x11	RTC_SPRERL Power-on Reset value <sup>(1)</sup>	PREDIV_S[7:0] 11111111																			
0x12	RTC_APRLR Power-on Reset value <sup>(1)</sup>	0	PREDIV_A[6:0] 1111111																		
0x13	Reserved																				
0x14	RTC_WUTRH Power-on Reset value <sup>(1)</sup>	WUT[15:8] 11111111																			
0x15	RTC_WUTRL Power-on Reset value <sup>(1)</sup>	WUT[7:0] 11111111																			
0x16	Reserved																				
0x17	RTC_SSRRH <sup>(3)</sup> Reset value <sup>(1)</sup>	SS[15:8] -																			
0x18	RTC_SSRL <sup>(3)</sup> Reset value <sup>(1)</sup>	SS[7:0] -																			
0x19	RTC_WPR Reset value <sup>(1)</sup>	KEY 0x00																			
0x1A	RTC_SHIFTRH <sup>(3)</sup> Reset value <sup>(1)</sup>	ADD1S 0	SUBFS[14:8] 0000000																		
0x1B	RTC_SHIFTRL <sup>(3)</sup> Reset value <sup>(1)</sup>	SUBFS[7:0] 00000000																			
0x1C	RTC_ALRMAR1 Power-on Reset value <sup>(1)</sup>	MSK1 0	ST[2:0] 000			SU[3:0] 0000															
0x1D	RTC_ALRMAR2 Power-on Reset value <sup>(1)</sup>	MSK2 0	MNT[2:0] 000			MNU[3:0] 0000															
0x1E	RTC_ALRMAR3 Power-on Reset value <sup>(1)</sup>	MSK3 0	PM 0	HT[1:0] 00	HU[3:0] 0000																
0x1F	RTC_ALRMAR4 Power-on Reset value <sup>(1)</sup>	MSK4 0	WDSEL 0	DT[1:0] 00	DU[3:0] 0000																
0x20-0x23	Reserved																				
0x24	RTC_ALRMASS RH <sup>(3)</sup> Power-on Reset value <sup>(1)</sup>	ALSS[14:8] 0000000																			
0x25	RTC_ALRMASS RL <sup>(3)</sup> Power-on Reset value <sup>(1)</sup>	ALSS[7:0] 0000000																			

**Table 80. RTC register map and reset values (continued)**

Add. offset	Register name	7	6	5	4	3	2	1	0
0x26	RTC_ALRMASS MSKR <sup>(3)</sup> Power-on Reset value <sup>(1)</sup>	0	0	0	0	MASKSS[3:0] 0000			
0x27-0x29	Reserved								
0x2A	RTC_CALRH <sup>(3)</sup> Power-on Reset value <sup>(1)</sup>	CALP 0	CALW8 0	CALW16 0	- 0000			CALM[8] 0	
0x2B	RTC_CALRL <sup>(3)</sup> Power-on Reset value <sup>(1)</sup>	CALM[7:0] 00000000							
0x2C	RTC_TCR1 <sup>(3)</sup> Power-on Reset value <sup>(1)</sup>	- 0	TAMP3 TRG 0	TAMP3E 0	TAMP2 TRG 0	TAMP2E 0	TAMP1 TRG 0	TAMP1E 0	TAMPIE 0
0x2D	RTC_TCR2 <sup>(3)</sup> Power-on Reset value <sup>(1)</sup>	TAMPPU DIS 0	TAMPPRCH[1:0] 00		TAMPFLT[1:0] 00		TAMPFREQ[2:0] 000		

1. The power-on reset value is given for the registers which are not impacted by a system reset and which are reset at power-on (RTC\_CRx, RTC\_SPREx, RTC\_APRE, RTC\_WUTRx, RTC\_SSRx and RTC\_ALRMARx, RTC\_ALRMASSx, RTC\_CALx, RTC\_TCRx). The system reset value is given for the other registers (RTC\_TRx, RTC\_DRx, RTC\_ISRx).
2. Reset value is 1 on low, medium+ and high density devices and 0 in medium density devices.
3. Available on medium+ and high density devices only.

## 25 Independent watchdog (IWDG)

This section applies to low, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

### 25.1 Introduction

The independent watchdog peripheral can be used to resolve processor malfunctions due to hardware or software failures. It is clocked by the LSI internal RC clock source, and thus stays active even if the main clock fails.

### 25.2 IWDG functional description

*Figure 132* shows the functional blocks of the independent watchdog module.

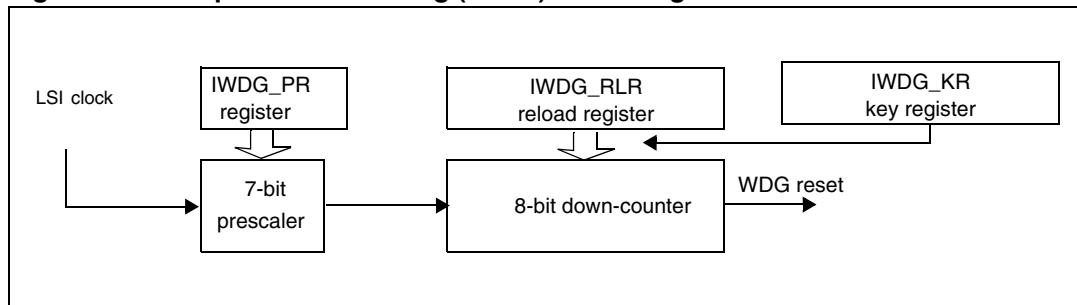
When the independent watchdog is started by writing the value 0xCC in the key register (IWDG\_KR), the counter starts counting down from the reset value of 0xFF. When it reaches the end of count value (0x00) a reset signal is generated (IWDG RESET).

Once enabled, the independent watchdog can be configured through the IWDG\_PR, and IWDG\_RLR registers. The IWDG\_PR register is used to select the prescaler divider feeding the counter clock. Whenever the KEY\_REFRESH value (0xAA) is written in the IWDG\_KR register, the IWDG is refreshed by reloading the IWDG\_RLR value into the counter and the watchdog reset is prevented.

The IWDG\_PR and IWDG\_RLR registers are write protected. To modify them, first write the KEY\_ACCESS code (0x55) in the IWDG\_KR register. The sequence can be aborted by writing 0xAA in the IWDG\_KR register to refresh it.

Refer to [Section 25.3: IWDG registers](#) for details on the IWDG registers.

**Figure 132. Independent watchdog (IWDG) block diagram**



#### Hardware watchdog feature

If the hardware watchdog feature has been enabled through the IWDG\_HW option byte, the watchdog is automatically enabled at power-on, and generates a reset unless the key register is written by the software before the counter reaches end of count. Refer to the option byte description in the datasheet.

### Timeout period

The maximum timeout period can be configured through the IWDG\_PR and IWDG\_RLR registers. It is determined by the following equation:

$$T = T_{LSI} \times P \times R$$

where:

$T$  = Maximum timeout period

$$T_{LSI} = 1/f_{LSI}$$

$$P = 2^{(PR[2:0] + 2)}$$

$$R = RLR[7:0]+1$$

The IWDG counter must be refreshed by software before this timeout period expires. Otherwise, an IWDG reset will be generated after the following delay has elapsed since the last refresh operation:

$$D = T + 3 \times T_{LSI}$$

where  $D$  = delay between the last refresh operation and the IWDG reset.

**Table 81. Min/Max IWDG timeout (LSI clock frequency = 38 kHz)**

Prescaler divider	PR[2:0] bits	Timeout (ms)	
		RL[7:0]= 0x00	RL[7:0]= 0xFF
/4	0	0.11	26.95
/8	1	0.21	53.89
/16	2	0.42	107.79
/32	3	0.84	215.58
/64	4	1.68	431.16
/128	5	3.37	862.32
/256	6	6.74	1724.63

### Using the IWDG in Halt/Active-halt mode

The IWDG can continue to work in Halt or Active-halt mode, depending on the configuration of the IWDG\_HALT option byte. In this case, it can wake up the device from one of these modes. For more details, please refer to the Option Byte description in the datasheet.

**Note:** *The application must configure correctly the IWDG timeout and refresh the IWDG counter before executing the HALT instruction, to avoid unexpected IWDG reset.*

## 25.3 IWDG registers

### 25.3.1 Key register (IWDG\_KR)

Address offset: 0x00

Reset value: 0XX

7	6	5	4	3	2	1	0
KEY[7:0]							
w							

Bits 7:0 **KEY[7:0]**: Key value

The KEY\_REFRESH value must be written by software at regular intervals, otherwise the watchdog generates an MCU reset when the counter reaches 0.

If the IWDG is not enabled by option byte (see datasheet for option byte description), the KEY\_ENABLE value is the first value to be written in this register.

**KEY\_ENABLE** value = 0xCC

Writing the KEY\_ENABLE value starts the IWDG.

**KEY\_REFRESH** value = 0xAA

Writing the KEY\_REFRESH value refreshes the IWDG.

**KEY\_ACCESS** value = 0x55

Writing the KEY\_ACCESS value enables the access to the protected IWDG\_PR and IWDG\_RLR registers (see [Section 25.2](#)).

### 25.3.2 Prescaler register (IWDG\_PR)

Address offset: 0x01

Reset value: 0X00

7	6	5	4	3	2	1	0
Reserved						PR[2:0]	
						rw	

Bits 7:3 Reserved

Bits 2:0 **PR[2:0]: Prescaler divider**

These bits are write access protected (see [Section 25.2](#)). They can be written by software to select the prescaler divider feeding the counter clock.

000: divider /4

001: divider /8

010: divider /16

011: divider /32

100: divider /64

101: divider /128

110: divider /256

111: Reserved

### 25.3.3 Reload register (IWDG\_RLR)

Address offset: 0x02

Reset value: 0xFF

7	6	5	4	3	2	1	0
RL[7:0]							
rw							

**RL[7:0]:** Watchdog counter reload value

Bits 7:0 These bits are write access protected (see [Section 25.2](#)). They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAA is written in the IWDG\_KR register. The watchdog counter counts down from this value. The timeout period is a function of this value and the clock prescaler. Refer to [Table 81](#).

### 25.3.4 IWDG register map and reset values

**Table 82.** IWDG register map

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	IWDG_KR Reset value	KEY7 x	KEY6 x	KEY5 x	KEY4 x	KEY3 x	KEY2 x	KEY1 x	KEY0 x
0x01	IWDG_PR Reset value	- 0	- 0	- 0	- 0	- 0	PR2 0	PR1 0	PR0 0
0x02	IWDG_RLR Reset value	RL7 1	RL6 1	RL5 1	RL4 1	RL3 1	RL2 1	RL1 1	RL0 1

## 26 Window watchdog (WWDG)

This section applies to low, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

### 26.1 Introduction

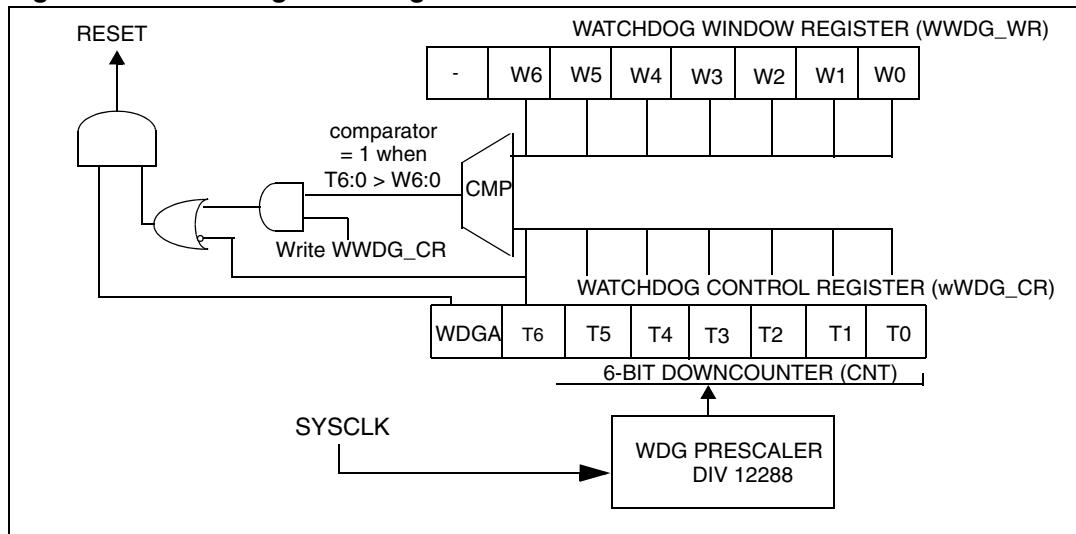
The window watchdog is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence. The watchdog circuit generates an MCU reset on expiry of a programmed time period, 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.

### 26.2 WWDG main features

- Programmable free-running downcounter
- Conditional reset
  - Reset (if watchdog activated) when the downcounter value becomes less than 0x40
  - Reset (if watchdog activated) if the downcounter is reloaded outside the window (see [Figure 134](#))
- Hardware/software watchdog activation (selectable by option byte)
- Optional reset on HALT instruction (configurable by option byte)

### 26.3 WWDG functional description

If the watchdog is activated (the WDGA bit is set) and when the 7-bit downcounter (T[6:0] bits) rolls over from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset cycle pulling low the reset pin. If the software refreshes the counter while the counter is greater than the value stored in the window register, then a reset is generated.

**Figure 133. Watchdog block diagram**

The application program must write in the WWDG\_CR 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 WWDG\_CR register must be between 0xFF and 0xC0:

- 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 WWDG\_CR 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. The timing varies between a minimum and a maximum value due to the unknown status of the prescaler when writing to the WWDG\_CR register (see [Figure 134](#)).

The window register (WWDG\_WR) 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 0x3F. [Figure 134](#) 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.

## 26.4 How to program the watchdog timeout

The formula below can be used to calculate the WWDG timeout,  $t_{WWDG}$ , expressed in ms:

$$t_{WWDG} = T_{SYSCLK} \times 12288 \times (T[5:0] + 1)$$

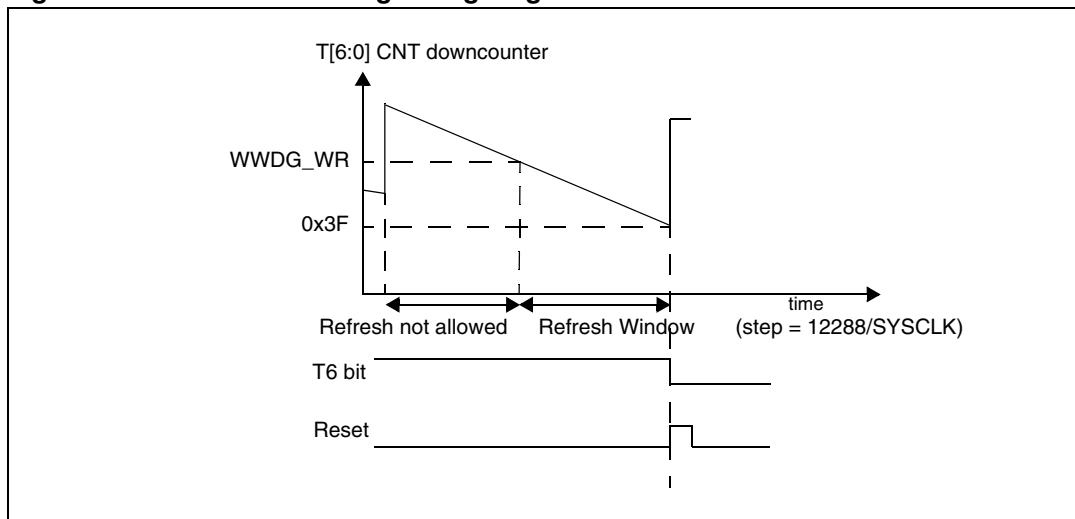
where  $T_{SYSCLK}$  is the peripheral clock period expressed in ms

---

**Warning:** When writing to the WWDG\_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

---

**Figure 134. Window watchdog timing diagram**



**Table 83. Window watchdog timing example**

T[6:0]	$f_{SYSCLK}$ (MHz)	
	2	16
40h	6.144	0.768
7Fh	393.216	49.152

## 26.5 WWDG low power modes

**Table 84. Effect of low power modes on WWDG**

Mode	Description
Wait	No effect on watchdog: The downcounter continues to decrement.

**Table 84. Effect of low power modes on WWDG (continued)**

Mode	Description	
Halt	WWDG_HALT in option byte	
	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.
	1	If an interrupt is received (refer to interrupt table mapping to see interrupts which can occur in Halt mode), the watchdog restarts counting after the stabilization delay. If a reset is generated, the watchdog is disabled (reset state) unless hardware watchdog is selected by option byte. For application recommendations see <a href="#">Section 26.7</a> below.
Active-halt	x	No reset is generated. The MCU enters Active-halt mode. The watchdog counter is not decremented. It stops 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 the stabilization delay.

## 26.6 Hardware watchdog option

If hardware watchdog is selected by option byte, the watchdog is always active and the WDGA bit in the WWDG\_CR register is not used. Refer to the option byte description in the datasheet.

## 26.7 WWDG interrupts

None.

## 26.8 WWDG registers

### 26.8.1 Control register (WWDG\_CR)

Address offset: 0x00

Reset value: 0x7F

7	6	5	4	3	2	1	0
WDGA	T6	T5	T4	T3	T2	T1	T0
rs	rw						

Bit 7 **WDGA**: Activation bit<sup>(1)</sup>

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

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 12288 SYSCLK cycles (approximately). A reset is produced when it rolls over from 0x40 to 0x3F (T6 becomes cleared).

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

### 26.8.2 Window register (WWDG\_WR)

Address offset: 0x01

Reset value: 0x7F

	7	6	5	4	3	2	1	0
Reserved	W6	W5	W4	W3	W2	W1	W0	
	rw							

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.

## 26.9 Window watchdog register map and reset values

Table 85. WWDG register map and reset values

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	WWDG_CR Reset value	WDGA 0	T6 1	T5 1	T4 1	T3 1	T2 1	T1 1	T0 1
0x01	WWDG_WR Reset value	- 0	W6 1	W5 1	W4 1	W3 1	W2 1	W1 1	W0 1

## 27 AES hardware accelerator (AES)

This section applies to high density STM8L16xx devices only.

### 27.1 Introduction

The AES hardware accelerator can be used to both encipher and decipher data using the AES algorithm.

It is a fully compliant implementation of the following standard: the advanced encryption standard (AES) as defined by the Federal information processing standards publication (FIPS PUB 197, 2001 November 26).

The accelerator encrypts and decrypts 128-bit blocks using a 128-bit key length. It can also perform key derivation.

The various chaining algorithms can be supported by software: Electronic codebook (ECB) mode, Cipher block chaining (CBC) mode or Counter (CTR) mode and Galois counter mode (GCM).

It supports DMA transfer for the incoming data and for the outgoing data (2 DMA channels required).

### 27.2 AES main features

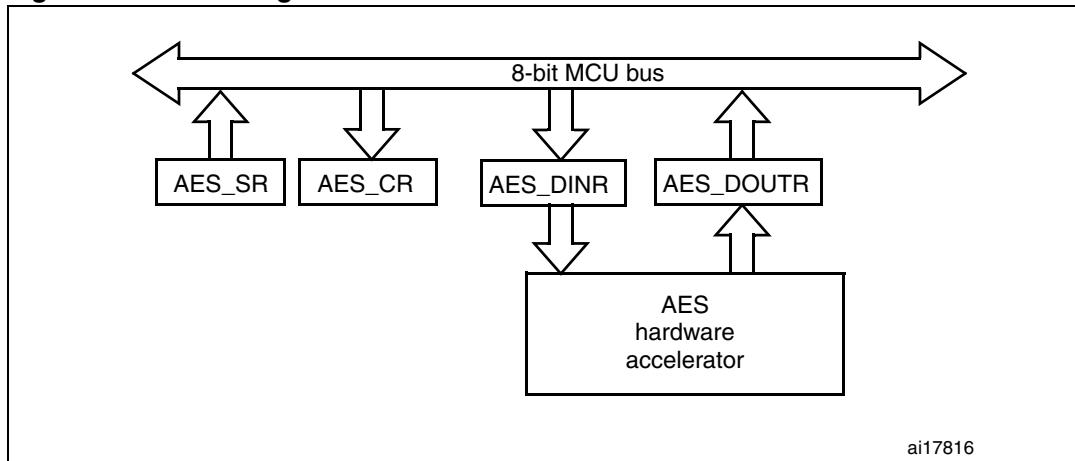
- Encryption/decryption using the AES Rijndael block cipher algorithm
- NIST FIPS 197 compliant implementation of AES encryption/decryption algorithm
- Key scheduler
- Key derivation for decryption
- 128-bit data block processed
- 128-bit key length
- 892 clock cycles to encrypt or decrypt one 128-bit block
- 1x8-bit input buffer and 1x8-bit output buffer
- Automatic data flow control with support of direct memory access (DMA) using 2 channels: one for incoming data, the other for outgoing data.

## 27.3 AES functional description

The cryptographic processor implements a hardware acceleration of AES cryptographic algorithm.

*Figure 135* shows the block diagram of the cryptographic processor.

**Figure 135. Block diagram**



The AES accelerator processes data blocks of 128 bits (16 bytes) using a key with a length of 128 bits and provides four modes of operations:

- Mode 1: encryption
- Mode 2: key derivation
- Mode 3: decryption using a given (pre-computed) decryption key
- Mode 4: key derivation + decryption using an encryption key.

The mode of operation is selected by programming the MODE[1:0] bits in the AES\_CR register. The mode can be changed only when the AES is disabled (EN bit = 0 in the AES\_CR register).

Once enabled (EN bit =1), the AES is in the input phase, waiting for the software to write the input data bytes into the AES\_DINR (32 or 16 bytes depending on the mode of operation).

Once all the write operations are performed, the computation phase takes place. The CCF status flag (computation complete flag) in the AES\_SR register is set once the computation phase is completed. An interrupt can be generated if the CCIE bit =1 in the AES\_CR register.

The operation ends with the output phase where the software reads successively the 16 output data bytes from the AES\_DOUTR register.

During the input and output phases, the software must read or write the data bytes successively but the AES is tolerant to any delay occurring between each read or write operation (example: if servicing another interrupt at this time).

The RDERR and WRERR flags in the AES\_SR register are set when an unexpected read or write operation is detected. An interrupt can be generated if ERRIE=1 into the AES\_CR register. The AES is not disabled after an error detection and continues to work in a normal way.

It is also possible to use the general purpose DMA to write the input bytes and to read the output bytes (refer to [Figure 140](#) and [Figure 141](#)).

**Note:** *The CCF flag has no meaning when DMAEN = 1 in the AES\_CR register, because the write and read operations on the AES\_DINR and AES\_DOUTR registers are automatically managed by the DMA without any software intervention.*

The AES can be re-initialized at any moment by resetting the EN bit in the AES\_CR register. Then the AES can be re-started from the beginning by setting the EN bit, waiting for the first input data byte to be written.

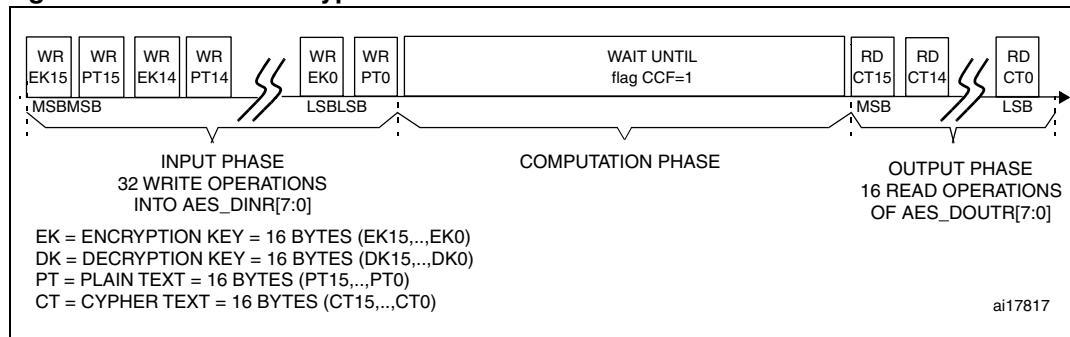
## 27.4 Modes of operation

### 27.4.1 Mode 1: encryption

To select this mode, use the following procedure:

1. Disable the AES by resetting the EN bit in AES\_CR
2. Configure the Mode 1 by programming MODE[1:0]=00 in the AES\_CR register.
3. If required, enable the computation complete and error interrupts through CCIE and ERRIE bits respectively in the AES\_CR register
4. Enable the AES by setting the EN bit in AES\_CR
5. In the AES\_DINR register, write 32 bytes representing the encryption key and plaintext (from MSB to LSB), as shown in [Figure 136: Mode 1: encryption on page 456](#).
6. Wait until the CCF flag is set in the AES\_SR register.
7. Read 16 times the AES\_DOUTR register to get the ciphertext (MSB first) as shown in [Figure 136: Mode 1: encryption on page 456](#).
8. Repeat steps 4, 5 and 6 to process all the blocks.

**Figure 136. Mode 1: encryption**

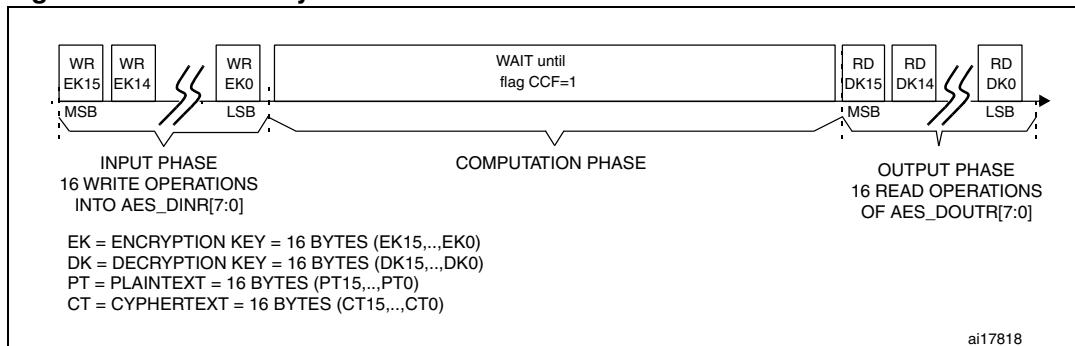


### 27.4.2 Mode 2: Key derivation

To select this mode, use the following procedure:

1. Disable the AES by resetting the EN bit in AES\_CR
2. Configure the Mode 2 by programming MODE[1:0]=01 in the AES\_CR register
3. If required, enable the computation complete and error interrupts through the CCIE and ERRIE bits respectively in AES\_CR register
4. Enable the AES by setting the EN bit in AES\_CR
5. In the AES\_DINR register, write 16 bytes representing the encryption key (from MSB to LSB), as shown in [Figure 137: Mode 2: key derivation on page 457](#).
6. Wait until the CCF flag is set in the AES\_SR register.
7. Read 16 times the AES\_DOUTR register which gives the decryption key (MSB first) as shown in [Figure 137: Mode 2: key derivation on page 457](#).
8. Clear the computation complete flag "CCF" by setting the CCFC bit in the AES\_CR register when the CPU is used for the data input or data output phase.
9. Repeat steps 5, 6, 7 and 8 to process all the keys.

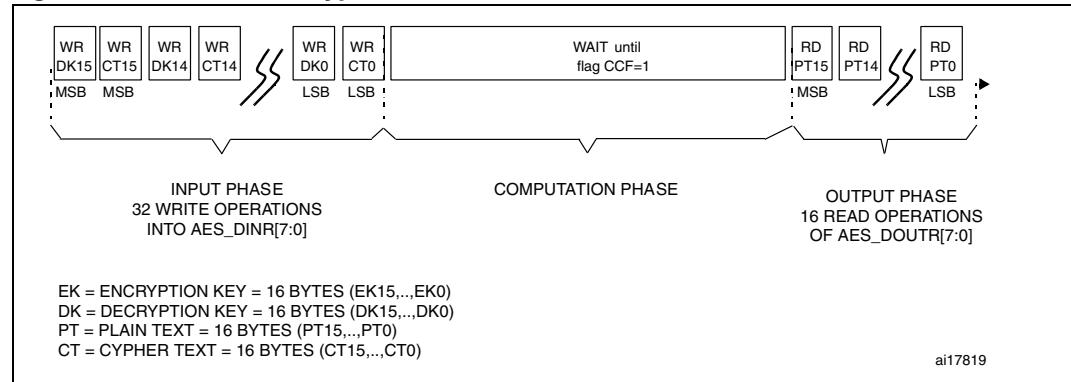
**Figure 137. Mode 2: key derivation**



### 27.4.3 Mode 3: decryption

To select this mode, use the following procedure:

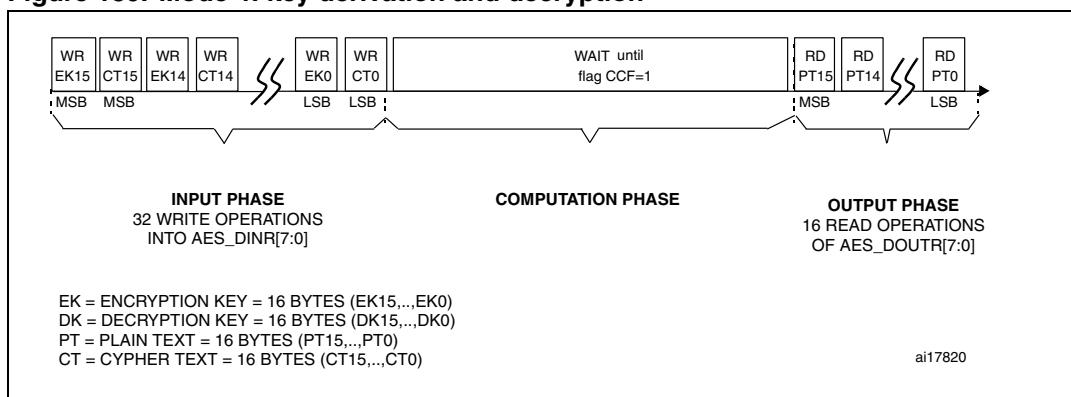
1. Disable the AES by resetting the EN bit in AES\_CR
2. Configure the Mode 3 by programming MODE[1:0]=10 in the AES\_CR register
3. If required, enable the computation complete and error interrupts through the CCIE and ERRIE bits respectively in AES\_CR register
4. Enable the AES by setting the EN bit in AES\_CR
5. In the AES\_DINR register, write 32 bytes representing the decryption key and the ciphertext (from MSB to LSB), as shown in [Figure 138: Mode 3: decryption on page 458](#).
6. Wait until the CCF flag is set into the AES\_SR register.
7. Read 16 times the AES\_DOUTR register which gives the plaintext (MSB first) as shown in [Figure 138: Mode 3: decryption on page 458](#).
8. Clear the computation complete flag "CCF" by setting the CCFC bit in the AES\_CR register when the CPU is used for the data input or data output phase.
9. Repeat steps 5, 6, 7 and 8 to process all the blocks.

**Figure 138. Mode 3: decryption**

#### 27.4.4 Mode 4: key derivation and decryption

To select this mode, use the following procedure:

1. Disable the AES by resetting the EN bit in AES\_CR
2. Configure the Mode 4 by programming MODE[1:0]=11 in the AES\_CR register
3. If required, enable the computation complete and error interrupts through the CCIE and ERRIE bits respectively in AES\_CR register
4. Enable the AES by setting the EN bit in AES\_CR
5. In the AES\_DINR register, write 32 bytes representing the encryption key and the ciphertext (from MSB to LSB), as shown in [Figure 139: Mode 4: key derivation and decryption on page 458](#).
6. Wait until the CCF flag is set in the AES\_SR register
7. Read 16 times to the AES\_DOUTR register which gives the plaintext (MSB first) as shown in [Figure 139: Mode 4: key derivation and decryption on page 458](#).
8. Clear the computation complete flag "CCF" by setting the CCFC bit in the AES\_CR register when the CPU is used for the data input or data output phase.
9. Repeat steps 5, 6, 7 and 8 to process all the blocks

**Figure 139. Mode 4: key derivation and decryption**

## 27.5 AES DMA interface

The AES accelerator provides an interface to connect to the DMA controller. The DMA must be configured to transfer bytes.

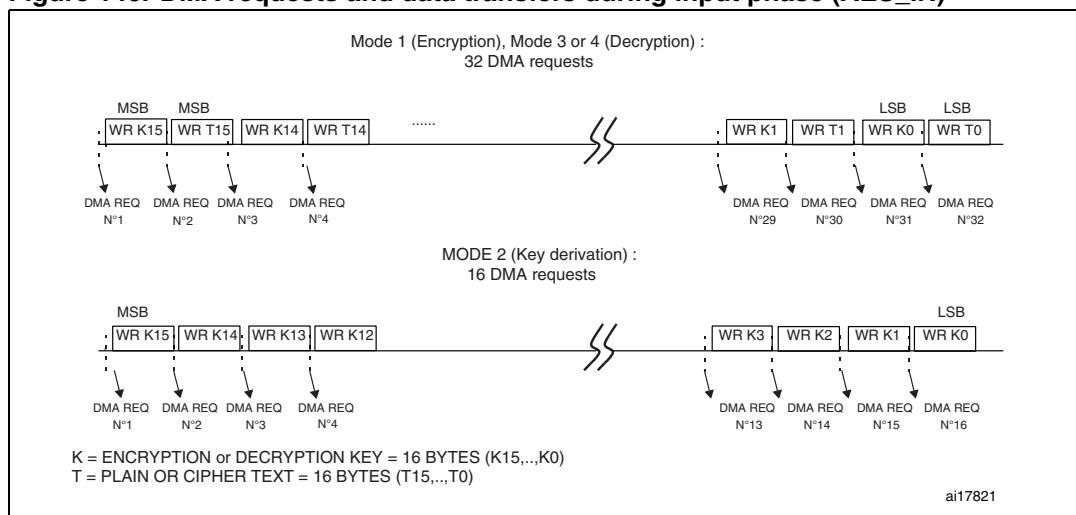
The AES can be associated with two distinct DMA request channels:

- A DMA request channel for the inputs: when the DMAEN bit is set in the AES\_CR register, the AES initiates the DMA request (AES\_IN) during the input phase each time it requires a byte to be written into the AES\_DINR register. The DMA channel must be configured in memory-to-peripheral mode (DIR=1 in the corresponding DMA\_CxCR register)
- A DMA request channel for the outputs: when the DMAEN bit is enabled, the AES initiates a DMA request (AES\_OUT) during the output phase each time it requires a byte to be read into the AES\_DOUTR register. The DMA channel must be configured in peripheral-to-memory mode (DIR=0 in the corresponding DMA\_CxCR register).

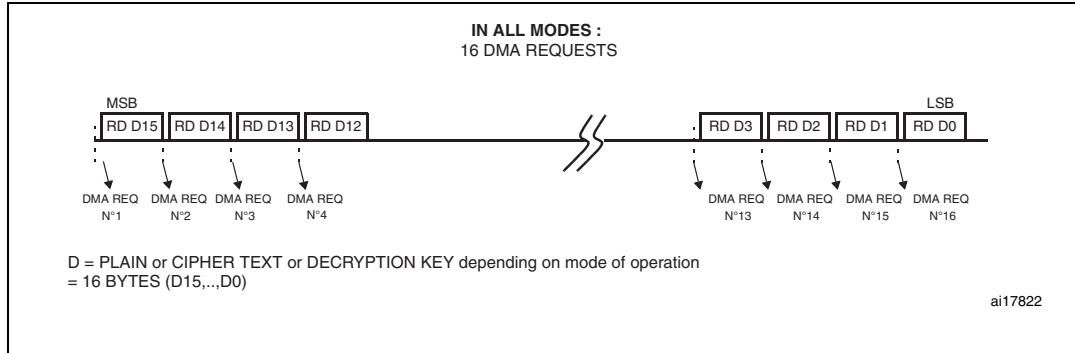
The number of DMA requests asserted for each phase depends on the mode of operation and is described in [Figure 140](#) and [Figure 141](#).

DMA requests are generated until the AES is disabled. Consequently, after the data output phase corresponding to one 128-bit data block processing, the AES switches automatically to a new data input phase for the next data block processing.

**Figure 140. DMA requests and data transfers during input phase (AES\_IN)**



**Figure 141. DMA requests during output phase (AES\_OUT)**



**Note:** When the DMA is enabled (DMAEN = 1), the CCF bit status is not relevant . This bit may remain high and no special care needs to be taken in this case. However, in case the application needs to exit from DMA mode and enable the CPU access for the data input or data output phase, the software must clear this bit.

## 27.6 Error flags

The RDERR flag in the AES\_SR register is set when an unexpected read operation is detected during the computation phase or during the input phase.

The WRERR flag in the AES\_SR register is set when an unexpected write operation is detected during the output phase or during the computation phase.

An interrupt can be generated when one of those two flags are set if the ERRIE bit in the AES\_CR register has been previously set.

If an error is detected, the AES is not disabled by hardware and still continues the on-going processing (if any) in a normal way.

## 27.7 Processing time

The following table summarizes the time required to process a 128-bit block for each mode of operation.

**Table 86.** Processing time (in clock cycles)

Mode of operation	Computation phase
Mode 1: encryption	892
Mode 2: key derivation	320
Mode 3: decryption	892
Mode 4: key derivation + decryption	1228

## 27.8 AES low power modes

**Table 87.** Effect of low power modes on AES

Mode	Description
Low power wait/Wait	No effect on AES. AES interrupt causes the device to exit from Wait or Low power wait mode.
Active-halt/Halt	The AES is stopped until Halt or Active-halt mode is exited.

## 27.9 AES interrupts

**Table 88.** AES interrupt requests

Interrupt event	Event flag	Enable control bit	Exit from Wait	Exit from Halt
AES computation complete flag	CCF	CCIE	yes	No
AES read error flag	RDERR	ERRIE	yes	No
AES write error flag	WRERR	ERRIE	yes	No

## 27.10 AES registers

### 27.10.1 AES control register (AES\_CR)

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
DMAEN	ERRIE	CCIE	ERRFC	CCFC	MODE[1:0]		EN
rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **DMAEN**: DMA enable

- 0: DMA disabled
- 1: DMA enabled

If the DMAEN bit is set, DMA requests are generated for the data input phase as well as for the output data phase.

Bit 6 **ERRIE**: Error interrupt enable

An interrupt is generated if one of the two RDERR or WRERR flags is set.

- 0: ERR interrupt disabled
- 1: ERR interrupt enabled

Bit 5 **CCIE**: Computation complete interrupt enable

An interrupt is generated if the CCF flag is set.

- 0: CCF interrupt disabled
- 1: CCF interrupt enabled

Bit 4 **ERRFC**: Error flags clear

Writing 1 into this bit clears both RDERR and WRERR flags.

This bit is always read as low.

Bit 3 **CCFC**: Computation complete flag clear

Writing 1 into this bit clears the CCF flag.

This bit is always read as low.

Bits 2:1 **MODE[1:0]**: AES mode of operation

- 00: Mode 1: Encryption
- 01: Mode 2: Key derivation
- 10: Mode 3: Decryption
- 11: Mode 4: Key derivation + decryption

*Note: The mode of operation can only be changed if the AES is disabled. Writing to these bits while AES is enabled is forbidden to avoid unpredictable AES behavior.*

Bit 0 **EN**: Enable

- 0: AES disabled
- 1: AES enabled

The AES can be re-initialized at any moment by resetting this bit: the AES is then ready to start processing a new block when EN is set.

### 27.10.2 AES status register (AES\_SR)

Address offset: 0x01

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved, read as 0				WRERR	RDERR	CCF	
				r	r	r	

Bits 7:3 Reserved, read as 0.

Bit 2 **WRERR**: Write error flag

This bit is set by hardware when an unexpected write operation into the AES\_DINR is detected. An interrupt is generated if the ERRIE bit has been previously set into the AES\_CR register.

It is cleared by software by setting the ERRFC bit into the AES\_CR register.

0: No write error detected

1: Write error detected

Bit 1 **RDERR**: Read error flag

This bit is set by hardware when an unexpected read operation into the AES\_DOUTR is detected. An interrupt is generated if the ERRIE bit has been previously set into the AES\_CR register.

It is cleared by software by setting the ERRFC bit in the AES\_CR register.

0: No read error detected

1: Read error detected

Bit 0 **CCF**: Computation complete flag

This bit is set by hardware when the computation is completed. An interrupt is generated if the CCIE bit has been previously set into the AES\_CR register.

It is cleared by software by setting the CCFC bit into the AES\_CR register.

0: Computation is not completed

1: Computation completed

*Note: This bit has a meaning only when DMAEN = 0. It may stay high when DMAEN = 1.*

### 27.10.3 AES data input register (AES\_DINR)

Address offset: 0x02

Reset value: 0x00

7	6	5	4	3	2	1	0
DINR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:0 **DINR[7:0]**: Data input register.

This register must be written 16 or 32 times during the input phase:

- In Mode 1 (encryption), 32 bytes must be written, which represent the plaintext and the encryption key from MSB to LSB, interleaving the key and the plaintext.
- In Mode 2 (key derivation), 16 bytes must be written, which form the encryption key from MSB to LSB.
- In Mode 3 (decryption) and 4 (key derivation+decryption), 32 bytes must be written, which represent the ciphertext and the decryption or encryption key from MSB to LSB, interleaving the key and the ciphertext.

### 27.10.4 AES data output register (AES\_DOUTR)

Address offset: 0x03

Reset value: 0x00

7	6	5	4	3	2	1	0
DOUTR[7:0]							
r	r	r	r	r	r	r	r

Bits 7:0 **DOUTR[7:0]**: Data output register

This register is read only.

Once the CCF flag (computation complete flag) is set, reading 16 times this data register gives access to the 128-bit output results:

- In Mode 1 (encryption), reading 16 bytes gives the ciphertext from MSB to LSB.
- In Mode 2 (key derivation), reading 16 bytes gives the decryption key from MSB to LSB.
- In Mode 3 (decryption) and 4 (key derivation+decryption), reading 16 bytes gives the plaintext from MSB to LSB.

### 27.10.5 AES register map and reset values

**Table 89. AES register map and reset values**

Add. offset	Register name	7	6	5	4	3	2	1	0
0x00	AES_CR Reset value	DMAEN 0	ERRIE 0	CCIE 0	ERRFC 0	CCFC 0	MOD[1:0] 00	EN 0	
0x01	AES_SR Reset value	- 0	- 0	- 0	- 0	- 0	WRERR 0	RDERR 0	CCF 0
0x02	AES_DINR Reset value	DINR[7:0] 0x00							
0x03	AES_DOUTR Reset value	DOUTR[7:0] 0x00							

## 28 Inter-integrated circuit (I<sup>2</sup>C) interface

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

### 28.1 Introduction

I<sup>2</sup>C (inter-integrated circuit) bus interface serves as an interface between the microcontroller and the serial I<sup>2</sup>C bus. It provides multi-master capability, and controls all I<sup>2</sup>C bus-specific sequencing, protocol, arbitration and timing. It supports standard and fast speed modes. It may be used for a variety of purposes, including CRC generation and verification, SMBus (system management bus) and PMBus (power management bus). DMA capability is also available for reduced CPU overload.

## 28.2 I<sup>2</sup>C main features

- Parallel-bus/I<sup>2</sup>C protocol converter
- Multi-master capability: the same interface can act as Master or Slave
- I<sup>2</sup>C Master features:
  - Clock generation
  - Start and Stop generation
- I<sup>2</sup>C Slave features:
  - Programmable I<sup>2</sup>C Address detection
  - Stop bit detection
  - I<sup>2</sup>C dual addressing capability to acknowledge 2 slave addresses (low, medium+ and high density devices only)
- Generation and detection of 7-bit/10-bit addressing and general call
- Supports different communication speeds:
  - Standard speed (up to 100 kHz),
  - Fast speed (up to 400 kHz)
- Status flags:
  - Transmitter/receiver mode flag
  - End-of-byte transmission flag
  - I<sup>2</sup>C busy flag
- Error flags:
  - Arbitration lost condition for master mode
  - Acknowledgement failure after address/ data transmission
  - Detection of misplaced start or stop condition
  - Overrun/underrun if clock stretching is disabled
- 3 types of interrupts:
  - 1 communication interrupt
  - 1 error condition interrupt
  - 1 wakeup from Halt interrupt
- Wakeup capability:
  - MCU wakes up from Low power mode on address detection in slave mode.
- Optional clock stretching
- 1-byte buffer with DMA capability
- Configurable PEC (packet error checking) generation or verification:
  - PEC value can be transmitted as last byte in Tx mode
  - PEC error checking for last received byte
- SMBus 2.0 compatibility:
  - 25 ms clock low timeout delay
  - 10 ms master cumulative clock low extend time
  - 25 ms slave cumulative clock low extend time
  - Hardware PEC generation/verification with ACK control
  - Address resolution protocol (ARP) supported
- PMBus compatibility

## 28.3 $I^2C$ general description

In addition to receiving and transmitting data, this interface converts it from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The interface is connected to the  $I^2C$  bus by a data pin (SDA) and by a clock pin (SCL). It can be connected with a standard (up to 100 kHz), or fast (up to 400 kHz)  $I^2C$  bus.

### Mode selection

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

By default, it operates in slave mode. The interface automatically switches from slave to master, after it generates a START condition and from master to slave, if an arbitration loss or a STOP generation occurs, allowing Multi-Master capability.

### Communication flow

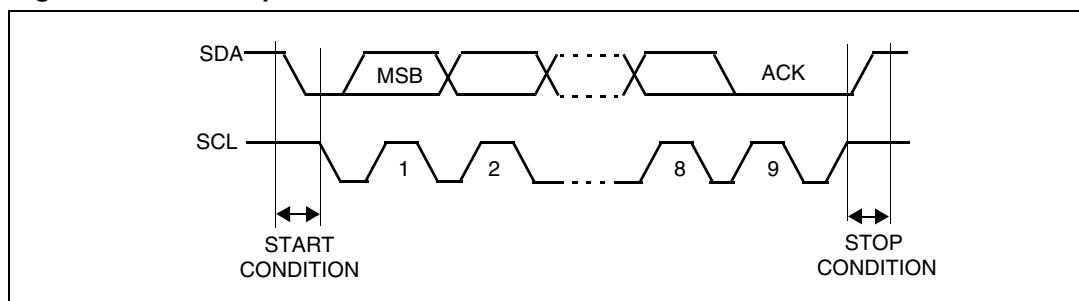
In Master mode, the  $I^2C$  interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a start condition and ends with a stop condition. Both start and stop conditions are generated in master mode by software.

In Slave mode, the interface is capable of recognizing its own addresses (7- or 10-bit), and the General Call address. The General Call address detection may be enabled or disabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the start condition contain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in Master mode.

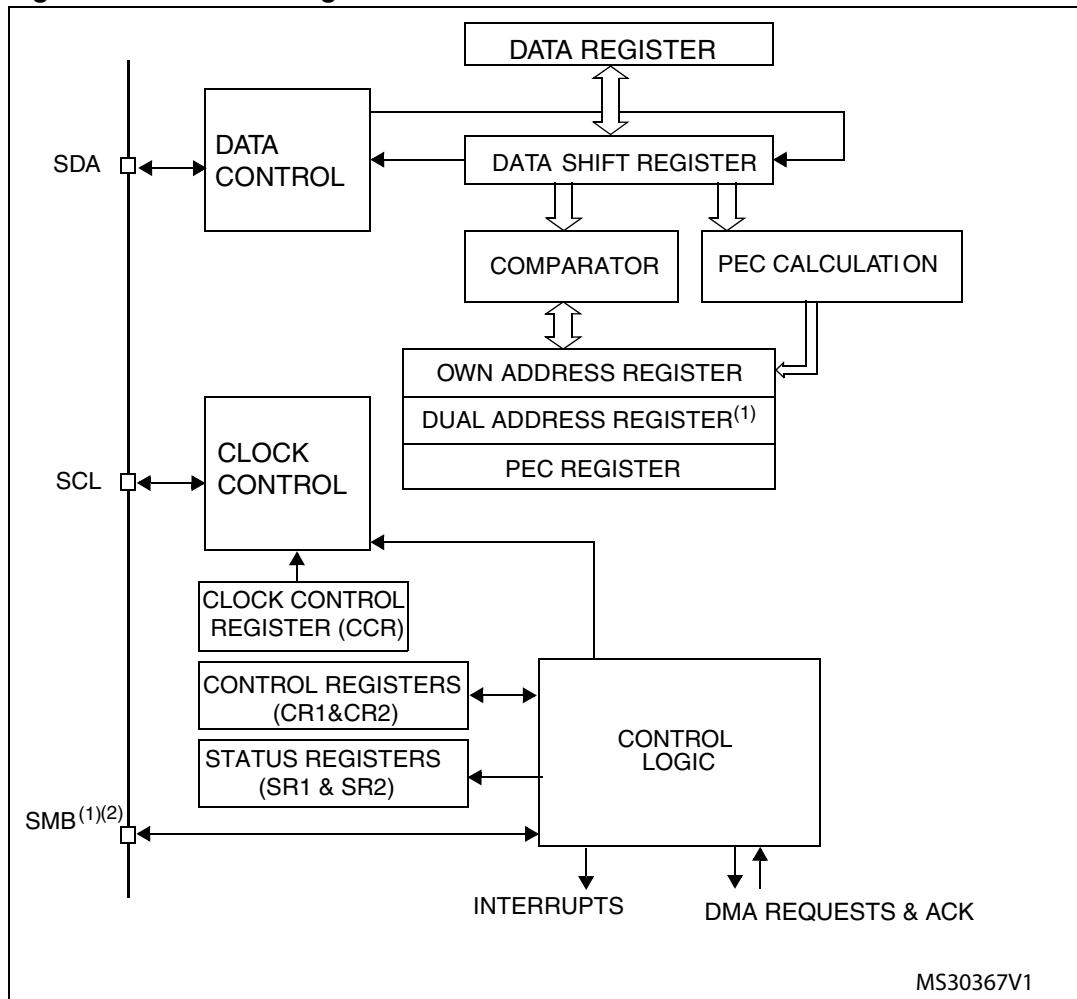
A 9th clock pulse follows the 8 clock cycles of a byte transfer, during which the receiver must send an acknowledge bit to the transmitter. Refer to the following figure.

**Figure 142.  $I^2C$  bus protocol**



Acknowledge may be enabled or disabled by software. The  $I^2C$  interface addresses (dual addressing, 7-/10-bit and/or general call address) can be selected by software.

The block diagram of the  $I^2C$  interface is shown in the following figure.

**Figure 143.  $I^2C$  block diagram**

1. SMB (SMBus ALERT signal) is an optional signal in SMBus mode. This signal is not applicable if SMBus is disabled.
2. Available in medium+ and high density devices only.

## 28.4 I<sup>2</sup>C functional description

By default the I<sup>2</sup>C interface operates in Slave mode. To switch from default Slave mode to Master mode a Start condition generation is needed.

### 28.4.1 I<sup>2</sup>C slave mode

The peripheral input clock must be programmed in the I2C\_FREQR register in order to generate correct timings. The peripheral input clock frequency must be at least:

- 1 MHz in Standard mode
- 4 MHz in Fast mode

As soon as a start condition is detected, the address is received from the SDA line and sent to the shift register. Then it is compared with the address of the interface (I2C\_OAR1L & I2C\_OAR2 if ENDUAL = 1 in medium+ and high density devices and I2C\_OARL in medium density devices) or the General Call address (if ENGC = 1).

*Note:* In 10-bit addressing mode, the comparison includes the header sequence (11110xx0), where xx denotes the two most significant bits of the address.

**Header or address not matched:** the interface ignores it and waits for another Start condition.

**Header matched** (10-bit mode only): the interface generates an acknowledge pulse if the ACK bit is set and waits for the 8-bit slave address.

**Address matched:** the interface generates in sequence:

- An acknowledge pulse if the ACK bit is set
- The ADDR bit is set by hardware and an interrupt is generated if the ITEVTEN bit is set. In medium+ and high density devices, if ENDUAL=1, the software has to read the DUALF bit to check which slave address has been acknowledged.

In 10-bit mode, after receiving the address sequence the slave is always in Receiver mode. It will enter Transmitter mode on receiving a repeated Start condition followed by the header sequence with matching address bits and the least significant bit set (11110xx1).

The TRA bit indicates whether the slave is in Receiver or Transmitter mode.

### Slave transmitter

Following the address reception and after clearing ADDR, the slave sends bytes from the DR register to the SDA line via the internal shift register.

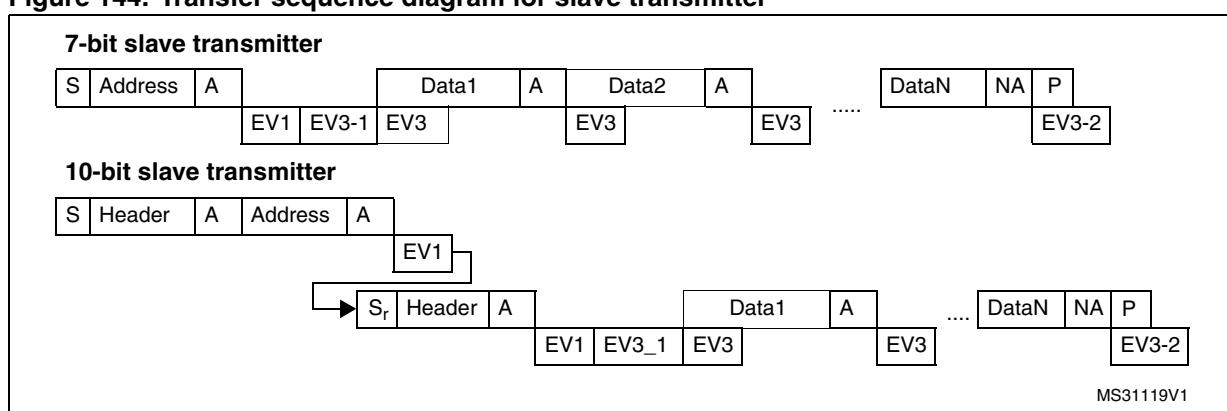
The slave stretches SCL low until ADDR is cleared and DR filled with the data to be sent (see Transfer sequencing EV1 EV3 in the following figure).

When the acknowledge pulse is received:

- The TXE bit is set by hardware with an interrupt if the ITEVTEN and the ITBUFEN bits are set.

If TXE is set and a data was not written in the DR register before the end of the next data transmission, the BTF bit is set and the interface waits until BTF is cleared, by reading the SR1 register and then writing to the DR register, stretching SCL low.

**Figure 144. Transfer sequence diagram for slave transmitter**



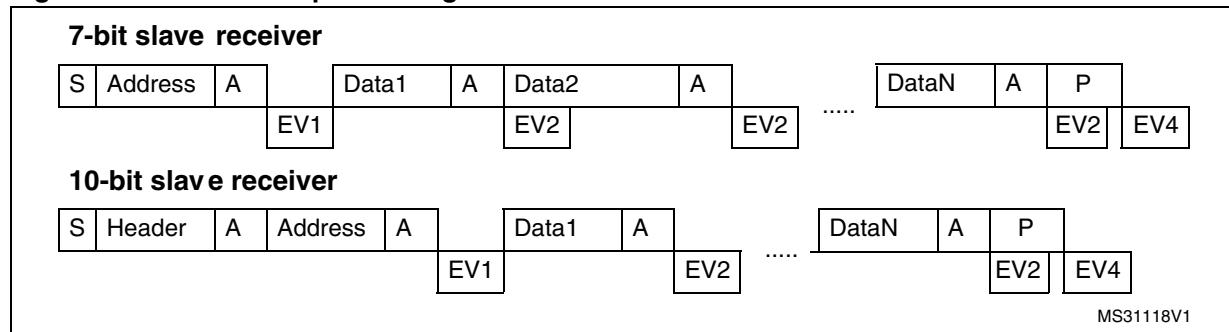
1. Legend:  
**S**= Start, **S<sub>r</sub>** = Repeated Start, **P**= Stop, **A**= Acknowledge, **NA**= Non-acknowledge, **EVx** = Event (with interrupt if ITEVTEN=1)  
**EV1**: ADDR=1, cleared by reading SR1 register followed by reading SR3.  
**EV3-1**: TXE=1, shift register empty, data register empty, write Data1 in DR.  
**EV3**: TXE=1, shift register not empty, data register empty, cleared by writing DR.  
**EV3-2**: AF=1, AF is cleared by writing '0' in AF bit of SR2 register.
2. EV1 and EV3-1 events stretch SCL low until the end of the corresponding software sequence.
3. EV3 software sequence must be performed before the end of the current byte transfer. In case EV3 software sequence can not be managed before the end of the current byte transfer, it is recommended to use BTF instead of TXE with the drawback of slowing the communication.

### Slave receiver

Following the address reception and after clearing ADDR, the slave receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

- An acknowledge pulse if the ACK bit is set
- The RXNE bit is set by hardware and an interrupt is generated if the ITEVTEN and ITBUFEN bit is set.

If RXNE is set and the data in the DR register is not read before the end of the next data reception, the BTF bit is set and the interface waits until BTF is cleared, by reading the SR1 register and then reading the DR register, stretching SCL low (see [Figure 145](#)).

**Figure 145.** Transfer sequence diagram for slave receiver

1. Legend:  
**S**= Start, **S<sub>r</sub>**= Repeated Start, **P**= Stop, **A**= Acknowledge, **NA**= Non-acknowledge, **EVx**= Event (with interrupt if ITEVTEN=1)  
**EV1**: ADDR=1, cleared by reading SR1 register followed by reading SR3.  
**EV2**: RXNE=1, cleared by reading DR register.  
**EV4**: STOPF=1, cleared by reading SR1 register followed by writing CR2 register
2. EV1 event stretches SCL low until the end of the corresponding software sequence.
3. EV2 software sequence must be performed before the end of the current byte transfer.
4. After checking the SR1 register content, the user should perform the complete clearing sequence for each flag found set. Thus, for the ADDR and STOPF flags, the following sequence is recommended inside the I2C interrupt routine:

```

READ SR1
if (ADDR == 1) {READ SR1; READ SR3}
if (STOPF == 1) {READ SR1; WRITE CR2}
  
```

The purpose is to make sure that both ADDR and STOPF flags are cleared if both are found set.

5. See also: [Note 9 on page 494](#).

### Closing slave communication

After the last data byte is transferred, a Stop condition is generated by the master. The interface detects this condition and sets the STOPF bit and generates an interrupt if the ITEVTEN bit is set.

STOPF is cleared by a read of the SR1 register followed by a write to the CR2 register (see [Figure 146](#)).

## 28.4.2 I<sup>2</sup>C master mode

In Master mode, the I<sup>2</sup>C interface initiates a data transfer and generates the clock signal. A serial data transfer always begins with a Start condition and ends with a Stop condition. Master mode is selected as soon as the Start condition is generated on the bus with a START bit.

The following is the required sequence in master mode.

- Program the peripheral input clock in I2C\_FREQR Register in order to generate correct timings.
- Configure the clock control registers
- Configure the rise time register
- Program the I2C\_CR1 register to enable the peripheral
- Set the START bit in the I2C\_CR2 register to generate a Start condition

The peripheral input clock frequency must be at least:

- 1 MHz in Standard mode
- 4 MHz in Fast mode

## Start condition

Setting the START bit causes the interface to generate a Start condition and to switch to Master mode (MSL bit set) when the BUSY bit is cleared.

**Note:** *In master mode, setting the START bit causes the interface to generate a Re-Start condition at the end of the current byte transfer.*

Once the Start condition is sent:

- The SB bit is set by hardware and an interrupt is generated if the ITEVTEN bit is set.

Then the master waits for a read of the SR1 register followed by a write in the DR register with the Slave address () .

## Slave address transmission

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

- In 10-bit addressing mode, sending the header sequence causes the following event:
  - The ADD10 bit is set by hardware and an interrupt is generated if the ITEVTEN bit is set.

Then the master waits for a read of the SR1 register followed by a write in the DR register with the second address byte (see [Figure 146](#) & [Figure 147](#) Transfer sequencing EV9).

The ADDR bit is set by hardware and an interrupt is generated if the ITEVTEN bit is set. Then the master waits for a read of the SR1 register followed by a read in the SR3 register (see [Figure 146](#) & [Figure 147](#) Transfer sequencing EV6).

- In 7-bit addressing mode, one address byte is sent.

As soon as the address byte is sent,

- The ADDR bit is set by hardware and an interrupt is generated if the ITEVTEN bit is set.

Then the master waits for a read of the SR1 register followed by a read in the SR3 register (see [Figure 146](#) & [Figure 147](#) Transfer sequencing EV6).

The master can decide to enter Transmitter or Receiver mode depending on the LSB of the slave address sent.

- In 7-bit addressing mode,
  - To enter Transmitter mode, a master sends the slave address with LSB reset.
  - To enter Receiver mode, a master sends the slave address with LSB set.
- In 10-bit addressing mode,
  - To enter Transmitter mode, a master sends the header (11110xx0) and then the slave address, (where xx denotes the two most significant bits of the address).
  - To enter Receiver mode, a master sends the header (11110xx0) and then the slave address. Then it should send a repeated Start condition followed by the header (11110xx1), (where xx denotes the two most significant bits of the address).

The TRA bit indicates whether the master is in Receiver or Transmitter mode.

## Master transmitter

Following the address transmission and after clearing ADDR, the master sends bytes from the DR register to the SDA line via the internal shift register.

The master waits until the first data byte is written in the DR register, (see [Figure 146 Transfer sequencing EV8\\_1](#)).

When the acknowledge pulse is received:

- The TXE bit is set by hardware and an interrupt is generated if the ITEVTEN and ITBUFEN bits are set.

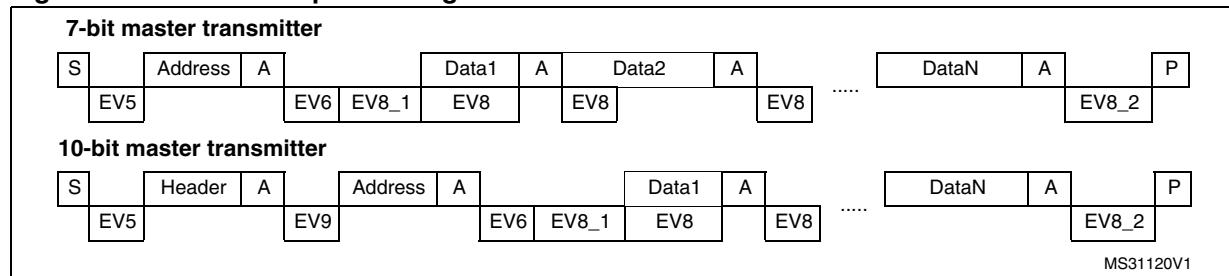
If TXE is set and a data byte was not written in the DR register before the end of the next data transmission, BTF is set and the interface waits until BTF is cleared, by reading the SR1 register and then writing to the DR register, stretching SCL low.

## Closing the communication

After writing the last byte to the DR register, the STOP bit is set by software to generate a Stop condition (see [Figure 146 Transfer sequencing EV8\\_2](#)). The interface goes automatically back to slave mode (MSL bit cleared).

*Note:* Stop condition should be programmed during EV8\_2 event, when either TXE or BTF is set.

**Figure 146. Transfer sequence diagram for master transmitter**



1. Legend:  
**S**= Start, **S<sub>r</sub>** = Repeated Start, **P**= Stop, **A**= Acknowledge, **NA**= Non-acknowledge,  
**EVX**= Event (with interrupt if ITEVTEN=1)  
**EV5**: SB=1, cleared by reading SR1 register followed by writing DR register with Address.  
**EV6**: ADDR=1, cleared by reading SR1 register followed by reading SR3.  
**EV8\_1**: TXE=1, shift register empty, data register empty, write DR register.  
**EV8**: TXE=1, shift register not empty, data register empty, cleared by writing DR register.  
**EV8\_2**: TXE=1, BTF = 1, Program STOP request. TXE and BTF are cleared by HW by stop condition  
**EV9**: ADD10=1, cleared by reading SR1 register followed by writing DR register. See also:[Note 9 on page 494](#)
2. EV8 software sequence must be performed before the end of the current byte transfer. In case EV8 software sequence can not be managed before the end of the current byte transfer, it is recommended to use BTF instead of TXE with the drawback of slowing the communication.

## Master receiver

Following the address transmission and after clearing ADDR, the I<sup>2</sup>C interface enters Master Receiver mode. In this mode the interface receives bytes from the SDA line into the DR register via the internal shift register. After each byte the interface generates in sequence:

- An acknowledge pulse if the ACK bit is set
- The RXNE bit is set and an interrupt is generated if the ITEVTEN and ITBUFEN bits are set () .

If the RXNE bit is set and the data in the DR register was not read before the end of the next data reception, the BTF bit is set by hardware and the interface waits for the BTF bit to be cleared by reading I2C\_SR1 and then I2C\_DR, stretching SCL low.

## Closing the communication

**Method 1:** This method is for the case when the I2C is used with interrupts that have the highest priority in the application.

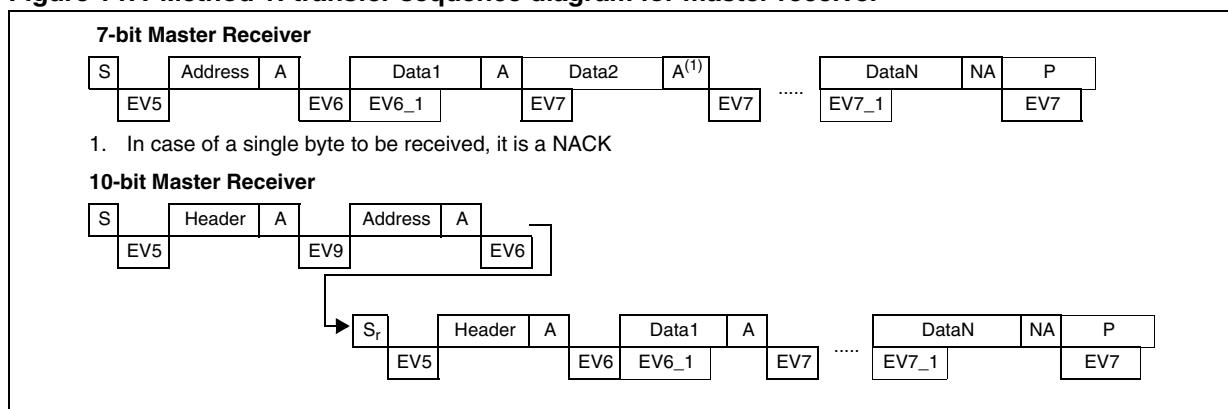
The master sends a NACK for the last byte received from the slave. After receiving this NACK, the slave releases the control of the SCL and SDA lines. Then the master can send a Stop/Re-Start condition.

- In order to generate the non-acknowledge pulse after the last received data byte, the ACK bit must be cleared just after reading the second last data byte (after second last RXNE event).
- In order to generate the Stop/Re-Start condition, software must set the STOP/ START bit just after reading the second last data byte (after the second last RXNE event).
- In case a single byte is to be received, the Acknowledge deactivation and the STOP condition generation are made just after EV6 (in EV6-1 just after ADDR is cleared).

After the Stop condition generation, the interface goes automatically back to slave mode (MSL bit cleared).

**Method 1:** This method is for the case when the I2C is used with interrupts that have the highest priority in the application.

**Figure 147. Method 1: transfer sequence diagram for master receiver**



1. Legend:  
**S**= Start, **S<sub>r</sub>** = Repeated Start, **P**= Stop, **A**= Acknowledge, **NA**= Non-acknowledge,  
**EVx**= Event (with interrupt if ITEVTEN=1)  
**EV5**: SB=1, cleared by reading SR1 register followed by writing DR register.  
**EV6**: ADDR=1, cleared by reading SR1 register followed by reading SR3. In 10-bit master receiver mode, this sequence should be followed by writing CR2 with START = 1.

- EV6\_1:** no associated flag event, used for 1 byte reception only. Program ACK=0 and STOP=1 after clearing ADDR.  
**EV7:** RxNE=1, cleared by reading DR register.  
**EV7\_1:** RxNE=1, cleared by reading DR register, program ACK=0 and STOP request  
**EV9:** ADD10=1, cleared by reading SR1 register followed by writing DR register.
2. If the DR and shift registers are full, the next data reception (I<sup>2</sup>C clock generation for slave) is performed after the EV7 event is cleared. In this case, EV7 does not overlap with data reception.
  3. If a single byte is received, it is NA.
  4. EV5, EV6 and EV9 events stretch SCL low until the end of the corresponding software sequence.
  5. EV7 software sequence must be completed before the end of the current byte transfer. In case EV7 software sequence can not be managed before the current byte end of transfer, it is recommended to use BTF instead of RXNE with the drawback of slowing the communication.
  6. The EV6\_1 or EV7\_1 software sequence must be completed before the ACK pulse of the current byte transfer.
  7. See also: [Note 9 on page 494](#).

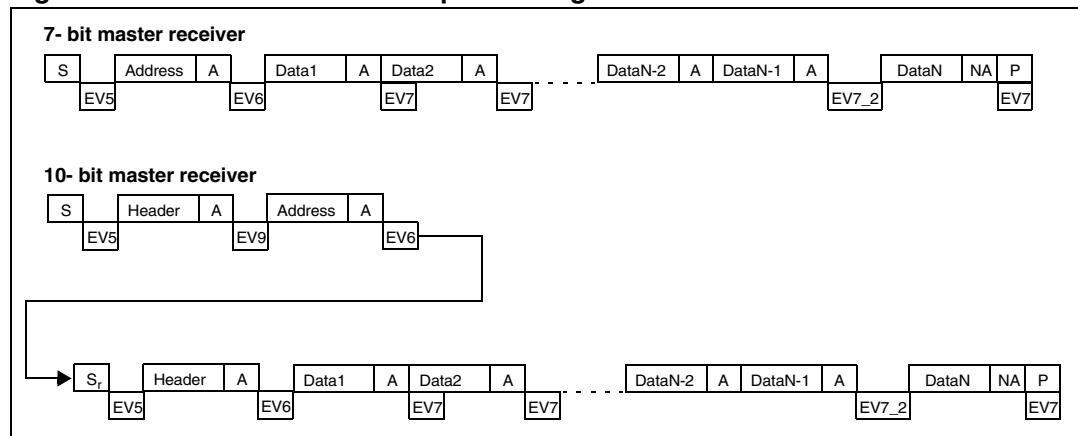
**Method 2:** This method is for the case when the I<sup>2</sup>C is used with interrupts that do not have the highest priority in the application or when the I<sup>2</sup>C is used with polling.

With this method:

- DataN\_2 is not read, so that after DataN\_1, the communication is stretched (both RxNE and BTF are set).
- Then, the ACK bit must be cleared before reading DataN-2 in DR to make sure this bit has been cleared before the DataN Acknowledge pulse.
- After that, just after reading DataN\_2, software must set the STOP/ START bit and read DataN\_1. After RxNE is set, read DataN.

This is illustrated in the following figure:

**Figure 148. Method 2: transfer sequence diagram for master receiver when N >2**



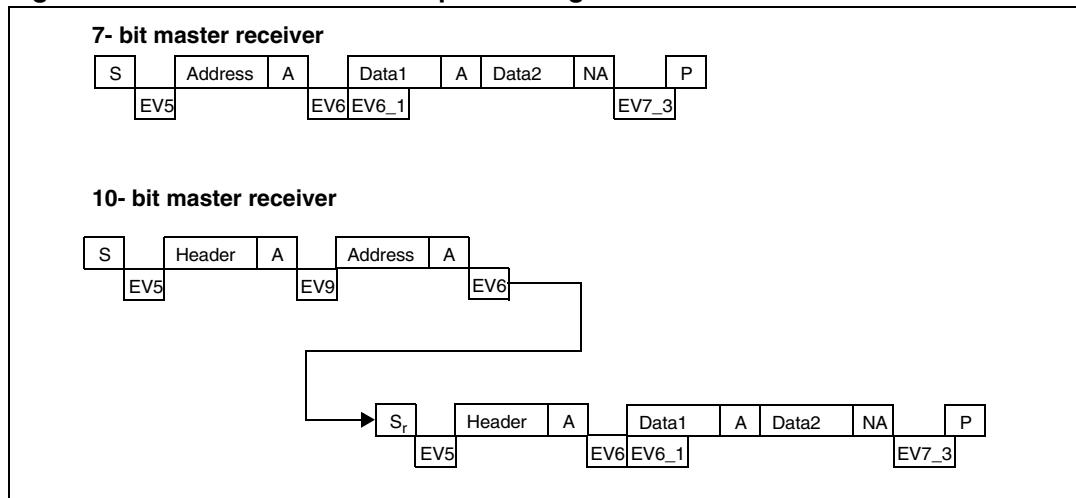
1. Legend:  
**S**= Start, **S<sub>r</sub>** = Repeated Start, **P**= Stop, **A**= Acknowledge, **NA**= Non-acknowledge,  
**EVx**= Event (with interrupt if I<sup>2</sup>EV滕EN=1)  
**EV5:** SB=1, cleared by reading SR1 register followed by writing the DR register.  
**EV6:** ADDR1, cleared by reading SR1 register followed by reading SR3.  
In 10-bit master receiver mode, this sequence should be followed by writing CR2 with START = 1.  
**EV7:** RxNE=1, cleared by reading DR register.  
**EV7\_2:** BTF = 1, DataN-2 in DR and DataN-1 in shift register, program ACK = 0, Read DataN-2 in DR.  
Program STOP = 1, read DataN-1.  
**EV9:** ADD10= 1, cleared by reading SR1 register followed by writing DR register.
2. EV5, EV6 and EV9 events stretch SCL low until the end of the corresponding software sequence.
3. EV7 software sequence must be completed before the end of the current byte transfer. In case EV7 software sequence can not be managed before the current byte end of transfer, it is recommended to use BTF instead of RXNE, with the drawback of slowing the communication.

When 3 bytes remain to be read:

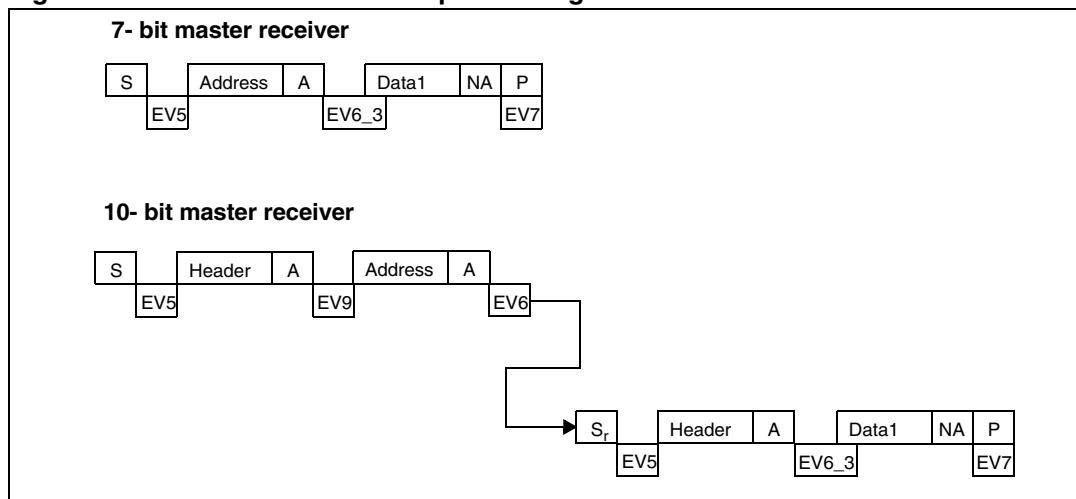
- RxNE = 1 => Nothing (DataN-2 not read).
- DataN-1 received
- BTF = 1 because both shift and data registers are full: DataN-2 in DR and DataN-1 in the shift register => SCL tied low: no other data will be received on the bus.
- Clear ACK bit
- Read DataN-2 in DR => This launches the DataN reception in the shift register
- DataN received (with a NACK)
- Program START/STOP
- Read DataN-1
- RxNE = 1
- Read DataN

The procedure described above is valid for  $N > 2$ . The cases where a single byte or two bytes are to be received should be handled differently, as described below:

- Case of a single byte to be received:
  - In the ADDR event, clear the ACK bit.
  - Clear ADDR
  - Program the STOP/START bit.
  - Read the data after the RxNE flag is set.
- Case of two bytes to be received:
  - Set POS and ACK
  - Wait for the ADDR flag to be set
  - Clear ADDR
  - Clear ACK
  - Wait for BTF to be set
  - Program STOP
  - Read DR twice

**Figure 149. Method 2: transfer sequence diagram for master receiver when N=2**

1. Legend:  
**S**= Start, **S<sub>r</sub>** = Repeated Start, **P**= Stop, **A**= Acknowledge, **NA**= Non-acknowledge, **EVx**= Event (with interrupt if ITEVTEN=1).  
**EV5**: SB=1, cleared by reading SR1 register followed by writing the DR register.  
**EV6**: ADDR1, cleared by reading SR1 register followed by reading SR3.  
In 10-bit master receiver mode, this sequence should be followed by writing CR2 with START = 1.  
**EV6\_1**: No associated flag event. The acknowledge should be disabled just after EV6, that is after ADDR is cleared  
**EV7\_3**: BTF = 1, program STOP = 1, read DR twice (Read Data1 and Data2) just after programming the STOP.  
**EV9**: ADD10= 1, cleared by reading SR1 register followed by writing DR register.
2. EV5, EV6 and EV9 events stretch SCL low until the end of the corresponding software sequence.
3. EV6\_1 software sequence must be completed before the ACK pulse of the current byte transfer.

**Figure 150. Method 2: transfer sequence diagram for master receiver when N=1**

1. Legend:  
**S**= Start, **S<sub>r</sub>** = Repeated Start, **P**= Stop, **A**= Acknowledge, **NA**= Non-acknowledge, **EVx**= Event (with interrupt if ITEVTEN=1).  
**EV5**: SB=1, cleared by reading SR1 register followed by writing the DR register.  
**EV6**: ADDR =1, cleared by reading SR1 resister followed by reading SR3 register.  
**EV6\_3**: ADDR = 1, program ACK = 0, clear ADDR by reading SR1 register followed by reading SR3 register, program STOP =1 just after ADDR is cleared.  
**EV7**: RxNE =1, cleared by reading DR register.

- EV9:** ADD10= 1, cleared by reading SR1 register followed by writing DR register.
2. EV5, EV6 and EV9 events stretch SCL low until the end of the corresponding software sequence.
  3. EV6\_3 software sequence must be completed before the ACK pulse of the current byte transfer.

### 28.4.3 Error conditions

The following are the error conditions which may cause communication to fail.

#### Bus error (BERR)

This error occurs when the I<sup>2</sup>C interface detects an external stop or a start condition during an address or data transfer. In this case:

- The BERR bit is set and an interrupt is generated if the ITERREN bit is set
- In the case of the slave: data are discarded and the lines are released by hardware:
  - In the case of a misplaced start, the slave considers it is a restart and waits for an address or a stop condition.
  - In the case of a misplaced stop, the slave reacts in the same way as for a stop condition and the lines are released by hardware.
- In the case of the master: the lines are not released and there is no effect in the state of the current transmission: software can decide if it wants to abort the current transmission or not.

#### Acknowledge failure (AF)

This error occurs when the interface detects a non-acknowledge bit. In this case,

- The AF bit is set and an interrupt is generated if the ITERREN bit is set
- A transmitter which receives a NACK must reset the communication:
  - If slave: Lines are released by hardware
  - If master: A stop condition or repeated startmust be generated by software

#### Arbitration lost (ARLO)

This error occurs when the I<sup>2</sup>C interface detects an arbitration lost condition. In this case,

- The ARLO bit is set by hardware (and an interrupt is generated if the ITERREN bit is set).
- The I<sup>2</sup>C interface goes automatically back to slave mode (the MSL bit is cleared)
- When the I<sup>2</sup>C loses the arbitration, it is not able to acknowledge its slave address in the same transfer, but it can acknowledge it after a repeated start from the master.
- Lines are released by hardware

#### Overrun/underrun error (OVR)

An Overrun error can occur in slave mode when clock stretching is disabled and the I<sup>2</sup>C interface is receiving data. The interface has received a byte (RXNE = 1) and the data in DR has not been read, before the next byte is received by the interface. In this case,

- The last received byte is lost
- In case of overrun error, software should clear the RXNE bit and the transmitter should re-transmit the last received byte.

Underrun error can occur in slave mode when clock stretching is disabled and the I<sup>2</sup>C interface is transmitting data. The interface has not updated the DR with the next byte (TXE=1), before the clock comes for the next byte. In this case,

- The same byte in the DR register will be sent again
- The user should make sure that data received on the receiver side during an underrun error is discarded and that the next bytes are written within the clock low time specified in the I<sup>2</sup>C bus standard.
- For the first byte to be transmitted, the DR must be written after ADDR is cleared and before the first SCL rising edge. If it is not possible, the receiver must discard the first data.

#### 28.4.4 SDA/SCL line control

- If clock stretching is enabled:
  - Transmitter mode: If TXE = 1 and BTF = 1: the interface holds the clock line low before transmission to wait for the microcontroller to read SR1 and then write the byte in the Data register (both buffer and shift register are empty).
  - Receiver mode: If RXNE = 1 and BTF = 1: the interface holds the clock line low after reception to wait for the microcontroller to read SR1 and then read the byte in the Data Register or write to CR2 (both buffer and shift register are full).
- If clock stretching is disabled in Slave mode:
  - Overrun error in case of RXNE = 1 and no read of DR has been done before the next byte is received. The last received byte is lost.
  - Underrun error in case TXE = 1 and no write into DR has been done before the next byte must be transmitted. The same byte will be sent again.
  - Write Collision not managed.

#### 28.4.5 SMBus

##### Introduction

The System management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I<sup>2</sup>C principles of operation. SMBus provides a control bus for system and power management related tasks. A system may use SMBus to pass messages to and from devices instead of toggling individual control lines.

The system management bus specification refers to three types of devices. A *slave* is a device that is receiving or responding to a command. A *master* is a device that issues commands, generates the clocks, and terminates the transfer. A *host* is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

##### Similarities between SMBus and I<sup>2</sup>C

- 2 wire bus protocol (1 Clk, 1 Data) + SMBus Alert line optional
- Master-slave communication, Master provides clock
- Multi master capability
- SMBus data format similar to I<sup>2</sup>C 7-bit addressing format ([Figure 142](#)).

## Differences between SMBus and I<sup>2</sup>C

The following table describes the differences between SMBus and I<sup>2</sup>C.

**Table 90. SMBus vs. I<sup>2</sup>C**

SMBus	I <sup>2</sup> C
Max. speed 100 kHz	Max. speed 400 kHz
Min. clock speed 10 kHz	No minimum clock speed
35 ms clock low timeout	No timeout
Logic levels are fixed	Logic levels are VDD dependent
Different address types (reserved, dynamic etc.)	7-bit, 10-bit and general call slave address types
Different bus protocols (quick command, process call etc.)	No bus protocols

## SMBus application usage

With System Management Bus, a device can provide manufacturer information, tell the system what its model/part number is, save its state for a suspend event, report different types of errors, accept control parameters, and return its status. SMBus provides a control bus for system and power management related tasks.

## Device identification

Any device that exists on the System Management Bus as a slave has a unique address called the Slave Address. For the list of reserved slave addresses, refer to the SMBus specification ver. 2.0 (<http://smbus.org/specs>).

## Bus protocols

The SMBus specification supports up to 9 bus protocols. For more details of these protocols and SMBus address types, refer to SMBus specification ver. 2.0 (<http://smbus.org/specs>). These protocols should be implemented by the user software.

## Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. The Address Resolution Protocol (ARP) has the following attributes:

- Address assignment uses the standard SMBus physical layer arbitration mechanism
- Assigned addresses remain constant while device power is applied; address retention through device power loss is also allowed
- No additional SMBus packet overhead is incurred after address assignment. (i.e. subsequent accesses to assigned slave addresses have the same overhead as accesses to fixed address devices.)
- Any SMBus master can enumerate the bus

### Unique device identifier (UDID)

In order to provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique device identifier (UDID).

For the details on 128 bit UDID and more information on ARP, refer to SMBus specification ver. 2.0 (<http://smbus.org/specs/>).

### SMBus alert mode

SMBus Alert is an optional signal with an interrupt line for devices that want to trade their ability to master for a pin. SMBALERT is a wired-AND signal just as the SCL and SDA signals are. SMBALERT is used in conjunction with the SMBus general call address. Messages invoked with the SMBus are 2 bytes long.

A slave-only device can signal the host through SMBALERT that it wants to talk by setting ALERT bit in I2C\_CR2 register. The host processes the interrupt and simultaneously accesses all SMBALERT devices through the *Alert response address* (known as ARA having a value 0001 100X). Only the device(s) which pulled SMBALERT low will acknowledge the Alert Response Address. This status is identified using SMBALERT Status flag in I2C\_SR2 register. The host performs a modified Receive Byte operation. The 7 bit device address provided by the slave transmit device is placed in the 7 most significant bits of the byte. The eighth bit can be a zero or one.

If more than one device pulls SMBALERT low, the highest priority (lowest address) device will win communication rights via standard arbitration during the slave address transfer. After acknowledging the slave address the device must disengage its SMBALERT pull-down. If the host still sees SMBALERT low when the message transfer is complete, it knows to read the ARA again.

A host which does not implement the SMBALERT signal may periodically access the ARA.

For more details on SMBus Alert mode, refer to SMBus specification ver. 2.0 (<http://smbus.org/specs/>).

### Timeout error

There are differences in the timing specifications between I<sup>2</sup>C and SMBus. SMBus defines a clock low timeout, TIMEOUT of 35 ms. Also SMBus specifies TLOW: SEXT as the cumulative clock low extend time for a slave device. SMBus specifies TLOW: MEXT as the cumulative clock low extend time for a master device. For more details on these timeouts, refer to SMBus specification ver. 2.0 (<http://smbus.org/specs/>).

The status flag Timeout or Tlow Error in I2C\_SR2 shows the status of this feature.

### How to use the interface in SMBus mode

To switch from I<sup>2</sup>C mode to SMBus mode, the following sequence should be performed.

- Set the SMBus bit in the I2C\_CR1 register
- Configure the SMBTYPE and ENARP bits in the I2C\_CR1 register as required for the application

If you want to configure the device as a master, follow the Start condition generation procedure in [Section 28.4.2: I<sup>2</sup>C master mode](#). Otherwise, follow the sequence in [Section 28.4.1: I<sup>2</sup>C slave mode](#).

The application has to control the various SMBus protocols by software.

- SMB Device default address acknowledged if ENARP=1 and SMBTYPE=0
- SMB Host header acknowledged if ENARP=1 and SMBTYPE=1
- SMB Alert response address acknowledged if SMBALERT=1

#### 28.4.6 DMA requests

DMA requests (when enabled) are generated only for data transfer. DMA requests are generated by Data Register becoming empty in transmission and Data Register becoming full in reception. The DMA must be initialized and enabled before the I<sup>2</sup>C data transfer. DMAEN bit must be set in I<sup>2</sup>C\_ITR before the ADDR event. In master mode or in slave mode when clock stretching is enabled, DMAEN bit can also be set during the ADDR event, before clearing the ADDR flag. The DMA request must be served before the end of the current byte transfer. When the number of data transfers which has been programmed for the corresponding DMA channel is reached, the DMA controller (DMA) sends an End of Transfer EOT signal to the I<sup>2</sup>C interface and generates a Transfer complete interrupt if enabled:

- Master transmitter: In the interrupt routine after the EOT interrupt, disable DMA requests then wait for a BTF event before programming the Stop condition.
- Master receiver: When the number of bytes to be received is equal to or greater than 2, the DMA controller sends a hardware signal EOT\_1 corresponding to the last but one data (number\_of\_bytes - 1). If, in the I<sup>2</sup>C\_CR2 register, the LAST bit is set, the I<sup>2</sup>C automatically sends a NACK after the next byte following EOT\_1. The user can generate a Stop condition in the DMA Transfer complete interrupt routine if enabled.

*Note:* DMA channel 0 is mapped to I<sup>2</sup>C for reception and DMA channel 3 is mapped to I<sup>2</sup>C for transmission.

#### Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAEN bit in the I<sup>2</sup>C\_ITR register. Data will be loaded from a memory area configured using the DMA peripheral (refer to the DMA specification) to the I<sup>2</sup>C\_DR register whenever the TxE bit is set. To map a DMA channel 3 for I<sup>2</sup>C transmission, perform the following sequence.

1. Set the I<sup>2</sup>C\_DR register address in the DMA\_C3PARH/L register. The data will be moved to this address from the memory after each TxE event.
2. Set the memory address in the DMA\_C3M0ARH/L registers. The data will be loaded into I<sup>2</sup>C\_DR from this memory after each TxE event.
3. Configure the total number of bytes to be transferred in the DMA\_C3NDTR register. After each TxE event, this value will be decremented.
4. Configure the channel priority using the PL[0:1] bits in the DMA\_C3SPR register
5. Set the DTD bit and, in the DMA\_C3CR register, configure interrupts after half transfer or full transfer depending on application requirements.
6. Activate the channel by setting the CE bit in the DMA\_C3CR register.

When the number of data transfers which has been programmed in the DMA controller registers is reached, the DMA controller sends an End of transfer EOT/ EOT\_1 signal to the I<sup>2</sup>C interface and the DMA generates an interrupt, if enabled, on the DMA channel interrupt vector.

*Note:* Do not enable the ITBUFEN bit in the I<sup>2</sup>C\_ITR register if DMA is used for transmission.

### Reception using DMA

DMA mode can be enabled for reception by setting the DMAEN bit in the I2C\_ITR register. Data will be loaded from the I2C\_DR register to a memory area configured using the DMA peripheral whenever a data byte is received. To map a DMA channel 0 for I2C reception, perform the following sequence.

1. Set the I2C\_DR register address in DMA\_C0PARH/L registers. The data will be moved from this address to the memory after each RxNE event.
2. Set the memory address in the DMA\_C0M0ARH/L registers. The data will be loaded from the I2C\_DR register to this memory area after each RxNE event.
3. Configure the total number of bytes to be transferred in the DMA\_C0NDTR register. After each RxNE event, this value will be decremented.
4. Configure the channel 0 priority using the PL[0:1] bits in the DMA\_C0SPR register.
5. Reset the DTD bit and configure interrupts in the DMA\_C0CR register after half transfer or full transfer depending on application requirements.
6. Activate the channel 0 by setting the CE bit in the DMA\_C0CR register. When the number of data transfers which has been programmed in the DMA controller registers is reached, the DMA controller sends an End of Transfer EOT/EOT\_1 signal to the I2C interface and the DMA generates an interrupt, if enabled, on the DMA channel interrupt vector.

*Note:* Do not enable the ITBUFEN bit in the I2C\_ITR register if DMA is used for reception.

### 28.4.7 Packet error checking

A PEC calculator has been implemented to improve the reliability of communication. The PEC is calculated by using the  $C(x) = x^8 + x^2 + x + 1$  CRC-8 polynomial serially on each bit.

- PEC calculation is enabled by setting the ENPEC bit in the I2C\_CR1 register. PEC is a CRC-8 calculated on all message bytes including addresses and R/W bits.
  - In transmission: set the PEC transfer bit in the I2C\_CR1 register after the TxE event corresponding to the last byte. The PEC will be transferred after the last transmitted byte.
  - In reception: set the PEC bit in the I2C\_CR2 register after the RxNE event corresponding to the last byte so that the receiver sends a NACK if the next received byte is not equal to the internally calculated PEC. PEC must be set before the ACK pulse of the current byte reception. In case of Master-Receiver, a NACK must follow the PEC whatever the check result.
- A PECERR error flag/interrupt is also available in the I2C\_SR2 register.
- If DMA and PEC calculation are both enabled:
  - In transmission: when the I2C interface receives an EOT signal from the DMA controller, it automatically sends a PEC after the last byte.
  - In reception: when the I2C interface receives an EOT\_1 signal from the DMA controller, it will automatically consider the next byte as a PEC and will check it. A DMA request is generated after PEC reception.
- To allow intermediate PEC transfers, a control bit is available in the I2C\_ITR register (LAST bit) to determine if it is really the last DMA transfer or not. If it is the last DMA request for a master receiver, a NACK is automatically sent after the last received byte.
- PEC calculation is corrupted by an arbitration loss.

## 28.5 I<sup>2</sup>C low power modes

**Table 91.** I<sup>2</sup>C interface behavior in low power modes

Mode	Description
Wait/Low power wait	No effect on I <sup>2</sup> C interface. I <sup>2</sup> C interrupts cause the device to exit from Wait or Low power wait mode.
Halt/Active-halt	<p><b>In slave mode:</b> Communication is reset, except for configuration registers. Device is in slave mode.</p> <p>Wakeup from Halt interrupt is generated if ITEVTEN = 1 and address matched (including allowed headers).</p> <p>The matched address is not acknowledged in Halt mode so the master has to send it again when the CPU is woken up to receive an acknowledge.</p> <p>If NOSTRETCH = 0, SCLH will be stretched after acknowledge pulse in Halt mode until WUFH is cleared by software;</p> <p>None of the flags are set by the address which wakes up the CPU.</p> <p><b>In master mode:</b> Communication is frozen until the CPU is woken up. Wakeup from Halt flag and interrupt are generated if ITEVTEN=1 and there is a HALT instruction.</p> <p><i>Note:</i> <i>It is forbidden to enter Halt mode while a communication is on going.</i></p>

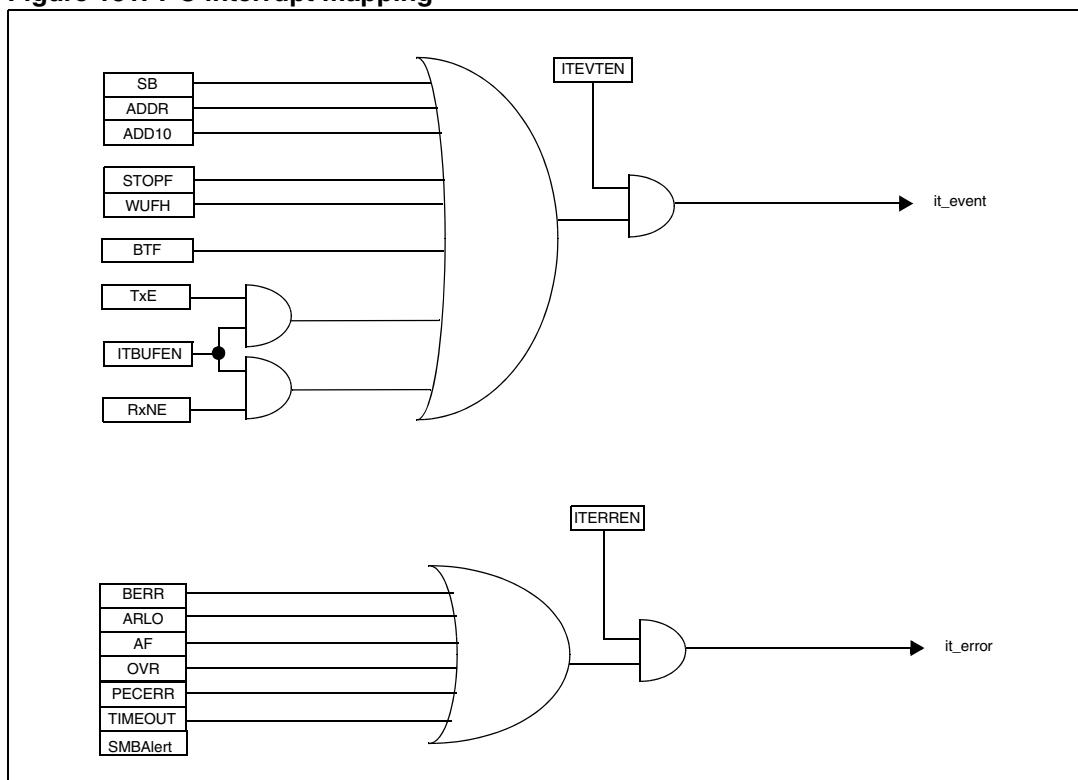
## 28.6 I<sup>2</sup>C interrupts

**Table 92.** I<sup>2</sup>C Interrupt requests

Interrupt event	Event flag	Enable control bit	Exit from Wait/Low power wait	Exit from Halt /Active-halt	
Start bit sent (Master)	SB	ITEVTEN	Yes	No	
Address sent (Master) or Address matched (Slave)	ADDR				
10-bit header sent (Master)	ADD10				
Stop received (Slave)	STOPF				
Data byte transfer finished	BTF				
Wakeup from Halt	WUFH	ITEVTEN	Yes	No	
Receive buffer not empty	RXNE	ITEVTEN and ITBUFEN	Yes		
Transmit buffer empty	TXE				
Bus error	BERR	ITERREN	No		
Arbitration loss (Master)	ARLO				
Acknowledge failure	AF				
Overrun/underrun	OVR				

**Table 92.** I<sup>2</sup>C Interrupt requests (continued)

Interrupt event	Event flag	Enable control bit	Exit from Wait/Low power wait	Exit from Halt /Active-halt
Bus error	BERR	ITERREN	Yes	No
Arbitration loss (Master)	ARLO		Yes	No
Acknowledge failure	AF		Yes	No
Overrun/Underrun	OVR		Yes	No
PEC error	PECERR		Yes	No
Timeout/Tlow error	TIMEOUT		Yes	No
SMBus Alert	SMBALERT		Yes	No

**Figure 151.** I<sup>2</sup>C interrupt mapping

## 28.7 I<sup>2</sup>C registers

### 28.7.1 Control register 1 (I2C\_CR1)

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Reserved	SMBUS	PE
rw	rw	rw	rw	rw		rw	rw

Bit 7 **NOSTRETCH**: Clock stretching disable (Slave mode)

This bit is used to disable clock stretching in slave mode when ADDR or BTF flag is set, until it is reset by software.

- 0: Clock stretching enabled
- 1: Clock stretching disabled

Bit 6 **ENGC**: General call enable

- 0: General call disabled. Address 0x00 is NACKed.
- 1: General call enabled. Address 0x00 is ACKed.

Bit 5 **ENPEC**: PEC enable

- 0: PEC calculation disabled
- 1: PEC calculation enabled

Bit 4 **ENARP**: ARP enable

- 0: ARP disabled
- 1: ARP enabled

SMBus Device default address recognized if SMBTYPE=0

SMBus Host address recognized if SMBTYPE=1

Bit 3 **SMBTYPE**: SMBus type

- 0: SMBus Device
- 1: SMBus Host

Bit 2 Reserved, forced by hardware to 0.

Bit 1 **SMBUS**: SMBus mode

- 0: I<sup>2</sup>C mode
- 1: SMBus mode

Bit 0 **PE**: Peripheral enable

- 0: Peripheral disable
- 1: Peripheral enable: the corresponding I/Os are selected as alternate functions.

*Note: If this bit is reset while a communication is on going, the peripheral is disabled at the end of the current communication, when back to IDLE state.*

*All bit resets due to PE=0 occur at the end of the communication.*

## 28.7.2 Control register 2 (I<sup>2</sup>C\_CR2)

Address offset: 0x01

Reset value: 0x00

7	6	5	4	3	2	1	0
SWRST	Reserved	ALERT	PEC	POS	ACK	STOP	START
rw		rw	rw	rw	rw	rw	rw

### Bit 7 SWRST: Software reset

When set, the I<sup>2</sup>C is at reset state. Before resetting this bit, make sure the I<sup>2</sup>C lines are released and the bus is free.

- 0: I<sup>2</sup>C Peripheral not at reset state
- 1: I<sup>2</sup>C Peripheral at reset state

*Note:* This bit can be used in case the BUSY bit is set to '1' when no stop condition has been detected on the bus.

### Bit 6 Reserved

### Bit 5 ALERT: SMBus alert

This bit is set and cleared by software, and cleared by hardware when PE=0.

- 0: Releases SMBAlert pin high. Alert Response Address Header followed by NACK.
- 1: Drives SMBAlert pin low. Alert Response Address Header followed by ACK.

### Bit 4 PEC: Packet error checking.

This bit is set and cleared by software, and cleared by hardware when PEC is transferred or by a START or Stop condition or when PE=0.

- 0: No PEC transfer
- 1: PEC transfer (in Tx or Rx mode)

*Note:* PEC calculation is corrupted by an arbitration loss.

### Bit 3 POS: Acknowledge position (for data reception).

This bit is set and cleared by software and cleared by hardware when PE=0.

- 0: ACK bit controls the (N)ACK of the current byte being received in the shift register. The PEC bit indicates the current byte in the shift register is PEC.

- 1: ACK bit controls the (N)ACK of the next byte which will be received in the shift register. The PEC bit indicates the next byte in the shift register is PEC.

*Note:* The POS bit is used when the procedure for reception of 2 bytes (see Method 2: transfer sequence diagram for master receiver when N=2) is followed. It must be configured before data reception starts. In this case, to NACK the 2nd byte, the ACK bit must be cleared just after ADDR is cleared.

*Note:*

To check the PEC when it is the second received byte, the PEC bit must be set during an ADDR stretch event after the POS bit has been configured.

### Bit 2 ACK: Acknowledge enable

This bit is set and cleared by software and cleared by hardware when PE=0.

- 0: No acknowledge returned
- 1: Acknowledge returned after a byte is received (matched address or data)

**Bit 1 STOP:** Stop generation

The bit is set and cleared by software, cleared by hardware when a Stop condition is detected, set by hardware when a timeout error is detected.

- In Master mode:

0: No Stop generation.

1: Stop generation after the current byte transfer or after the current Start condition is sent.

- In Slave mode:

0: No Stop generation.

1: Release the SCL and SDA lines after the current byte transfer.

**Bit 0 START:** Start generation

This bit is set and cleared by software and cleared by hardware when start is sent or PE=0.

- In Master mode:

0: No Start generation

1: Repeated start generation

- In Slave mode:

0: No Start generation

1: Start generation when the bus is free

**Note:**

*When STOP, START or PEC is set, the user must not perform any write access to I2C\_CR2 before the control bit is cleared by hardware. Otherwise, a second STOP, START or PEC request may occur.*

### 28.7.3 Frequency register (I2C\_FREQR)

Address offset: 0x02

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved	FREQ[5:0]						
	rw						

Bits 7:6 Reserved

Bits 5:0 **FREQ[5:0]** Peripheral clock frequency. <sup>(1)</sup>

Input clock frequency must be programmed to generate correct timings:

The allowed range is between 1 MHz and 16 MHz

000000: not allowed

000001: 1 MHz

000010: 2 MHz

...

010000: 16 MHz

Higher values: not allowed

1. The minimum peripheral clock frequencies for respecting the I<sup>2</sup>C bus timings are:  
1 MHz for standard mode and 4 MHz for fast mode

### 28.7.4 Own address register LSB (I2C\_OAR1L)

Address offset: 0x03

Reset value: 0x00

7	6	5	4	3	2	1	0
ADD[7:1]							ADD0
rw							rw

Bits 7:1 **ADD1[7:1]** Interface address

bits 7:1 of address

Bit 0 **ADD1[0]** Interface address

7-bit addressing mode: don't care

10-bit addressing mode: bit 0 of address

### 28.7.5 Own address register MSB (I2C\_OAR1H)

Address offset: 0x04

Reset value: 0x00

7	6	5	4	3	2	1	0
ADDMODE	ADDCONF	Reserved			ADD[9:8]	Reserved	
rw	rw				rw		

Bit 7 **ADDMODE** Addressing mode (Slave mode)

0: 7-bit slave address (10-bit address not acknowledged)

1: 10-bit slave address (7-bit address not acknowledged)

Bit 6 **ADDCONF** Address mode configuration

This bit must set by software (must always be written as '1').

Bits 5:3 Reserved

Bits 2:1 **ADD[9:8]** Interface address

10-bit addressing mode: bits 9:8 of address.

Bit 0 Reserved

### 28.7.6 Own address register 2 (I2C\_OAR2)

Address offset: 0x05

Reset value: 0x00

**Note:** This register is available on low, medium+ and high density devices only.

7	6	5	4	3	2	1	0
ADD2[7:1]							ENDUAL
rw							rw

Bits 7:1 **ADD2[7:1]**: Interface address

Bits 7:1 of address in Dual addressing mode.

Bit 0 **ENDUAL**: Dual addressing mode enable

0: Only OAR1 is recognized in 7-bit Addressing mode.

1: Both OAR1 and OAR2 are recognized in 7-bit Addressing mode.

### 28.7.7 Data register (I<sup>2</sup>C\_DR)

Address offset: 0x06

Reset value: 0x00

7	6	5	4	3	2	1	0
DR[7:0]							
rw							

Bits 7:0 **DR[7:0]: Data register** <sup>(1)(2)(3)</sup>

Byte received or to be transmitted to the bus.

- Transmitter mode: Byte transmission starts automatically when a byte is written in the DR register. A continuous transmit stream can be maintained if the next data to be transmitted is put in DR once the transmission is started (TXE=1)
- Receiver mode: Received byte is copied into DR (RXNE=1). A continuous transmit stream can be maintained if DR is read before the next data is received (RXNE=1).

1. In slave mode, the address is not copied into DR.
2. Write collision is not managed (DR can be written if TXE=0).
3. If an ARLO event occurs on ACK pulse, the received byte is not copied into DR and so cannot be read.

### 28.7.8 Status register 1 (I<sup>2</sup>C\_SR1)

Address offset: 0x07

Reset value: 0x00

7	6	5	4	3	2	1	0
TXE	RXNE	Reserved	STOPF	ADD10	BTF	ADDR	SB
r	r		r	r	r	r	r

Bit 7 **TXE: Data register empty (transmitters)** <sup>(1) (2)</sup>

- 0: Data register not empty
- 1: Data register empty

- Set when DR is empty in transmission. TXE is not set during address phase.
- Cleared by software writing to the DR register or by hardware after a start or a stop condition or when PE=0.

*Note:* TXE cannot be cleared by writing the first data in transmission or by writing a data when the BTF bit is set as in both cases, the DR register is still empty.

Bit 6 **RXNE: Data register not empty (receivers)** <sup>(3) (4)</sup>

- 0: Data register empty
- 1: Data register not empty

- Set when data register is not empty in receiver mode. RXNE is not set during address phase.
- Cleared by software reading or writing the DR register or by hardware when PE=0.

*Note:* RXNE cannot be cleared by reading a data when the BTF bit is set as the DR register is still full in this case.

Bit 5 Reserved

Bit 4 **STOPF**: Stop detection (Slave mode) <sup>(5)(6)</sup>

- 0: No Stop condition detected
- 1: Stop condition detected

- Set by hardware when a Stop condition is detected on the bus by the slave after an acknowledge (if ACK=1).
- Cleared by software reading the SR1 register followed by a write in the CR2 register, or by hardware when PE=0

Bit 3 **ADD10**: 10-bit header sent (Master mode) <sup>(7)</sup>

- 0: No ADD10 event occurred.
- 1: Master has sent first address byte (header).

- Set by hardware when the master has sent the first byte in 10-bit address mode.
- Cleared by software reading the SR1 register followed by a write in the DR register of the second address byte, or by hardware when PE=0.

Bit 2 **BTB**: Byte transfer finished <sup>(8)(9) (10)</sup>

- 0: Data byte transfer not done
- 1: Data byte transfer succeeded

- Set by hardware when NOSTRETCH=0 and:
  - In reception when a new byte is received (including ACK pulse) and DR has not been read yet (RXNE=1).
  - In transmission when a new byte should be sent and DR has not been written yet (TXE=1).
- Cleared by software reading SR1 followed by either a read or write in the DR register or by hardware after a start or a stop condition in transmission or when PE=0.

Bit 1 **ADDR**: Address sent (master mode)/matched (slave mode) <sup>(9)(11)</sup>

This bit is cleared by software reading SR1 register followed reading SR3, or by hardware when PE=0.

- Address matched (Slave)
  - 0: Address mismatched or not received.
  - 1: Received address matched.
    - Set by hardware as soon as the received slave address matched with the OAR registers content or a general call or a SMBus is recognized. (when enabled depending on configuration).
- Address sent (Master)
  - 0: No end of address transmission
  - 1: End of address transmission
    - For 10-bit addressing, the bit is set after the ACK of the 2nd byte.
    - For 7-bit addressing, the bit is set after the ACK of the byte.

*Note: ADDR is not set after a NACK reception*

Bit 0 **SB**: Start bit (Master mode) <sup>(9)</sup>

- 0: No Start condition

- 1: Start condition generated.

- Set when a Start condition generated.
- Cleared by software by reading the SR1 register followed by writing the DR register, or by hardware when PE=0

1. The interrupt or DMA request will be generated when DR is copied into shift register after an ACK pulse. If a NACK is received, copy is not done and TXE is not set.
2. TXE is not set if the next byte to be transmitted is the PEC.
3. The interrupt or DMA request will be generated when Shift register is copied into DR after an ACK pulse.
4. RXNE is not set in case of ARLO event.
5. The STOPF bit is not set after a NACK reception.

6. It is recommended to perform the complete clearing sequence (READ SR1 then WRITE CR2) after STOPF is set. Refer to [Figure 145: Transfer sequence diagram for slave receiver on page 472](#)
7. The ADD10 bit is not set after a NACK reception.
8. The BTF bit is not set after a NACK reception, or in case of an ARLO event.
9. Due to timing constraints, when in standard mode if CCR is less than 9 (i.e. with peripheral clock below 2 MHz) with  $f_{SYSCLK} = f_{CPU}$  and the event interrupt disabled, the following procedure must be followed:  
modify the reset sequence in order to insert at least 5 cycles between each operations in the flag clearing sequence. For example, when  $f_{SYSCLK} = f_{CPU} = 1$  MHz, use the following sequence to poll the SB bit:  
`_label_wait: BTJF I2C_SR1,SB,_label_wait  
NOP ;  
NOP;  
NOP ;  
NOP  
NOP  
LD I2C_DR, A ; once executed, the SB bit is then cleared.`
10. The BTF bit is not set if the next byte to be transmitted is the PEC (TRA=1 and PEC=1)
11. In slave mode, it is recommended to perform the complete clearing sequence (READ SR1 then READ SR3) after ADDR is set. Refer to [Figure 145: Transfer sequence diagram for slave receiver on page 472](#).

### 28.7.9 Status register 2 (I2C\_SR2)

Address offset: 0x08

Reset value: 0x00

7	6	5	4	3	2	1	0
SMBALERT	TIMEOUT	WUFH	PECERR	OVR	AF	ARLO	BERR
rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bit 7 **SMBALERT**: SMBus alert

In SMBus host mode:

- 0: no SMBAlert
- 1: SMBAlert event occurred on pin

In SMBus slave mode:

- 0: no SMBAlert response address header
- 1: SMBAlert response address header to SMBAlert LOW received
  - Cleared by software writing 0, or by hardware when PE=0.

Bit 6 **TIMEOUT**: Timeout or Tlow error

- 0: No timeout error
- 1: SCL remained low for 25 ms (timeout)

or

Master cumulative clock low extend time more than 10 ms (Tlow:mext)

or

Slave cumulative clock low extend time more than 25 ms (Tlow:sext)

- When set in slave mode: slave resets the communication and lines are released by hardware
- When set in master mode: Stop condition sent by hardware
- Cleared by software writing 0, or by hardware when PE=0.

Bit 5 **WUFH**: Wakeup from Halt

- 0: no wakeup from Halt mode
- 1: 7-bit address or header match in Halt mode (slave mode) or Halt entered when in master mode.

*Note: This bit is set asynchronously in slave mode (during HALT mode). It is set only if ITEVTEN = 1.*

- Cleared by software writing 0, or by hardware when PE=0.

Bit 4 **PECERR**: PEC Error in reception

- 0: no PEC error: receiver returns ACK after PEC reception (if ACK=1)
- 1: PEC error: receiver returns NACK after PEC reception (whatever ACK)
  - Cleared by software writing 0, or by hardware when PE=0.

Bit 3 **OVR**: Overrun/underrun

- 0: No overrun/underrun
- 1: Overrun or underrun
  - Set by hardware in slave mode when NOSTRETCH=1 and:
  - In reception when a new byte is received (including ACK pulse) and the DR register has not been read yet. New received byte is lost.
  - In transmission when a new byte should be sent and the DR register has not been written yet. The same byte is sent twice.

Cleared by software writing 0, or by hardware when PE=0.

*Note: if the DR write occurs very close to the SCL rising edge, the sent data is unspecified and a hold timing error occurs.*

Bit 2 **AF**: Acknowledge failure.

- 0: No acknowledge failure
- 1: Acknowledge failure
  - Set by hardware when no acknowledge is returned.
  - Cleared by software writing 0, or by hardware when PE=0.

Bit 1 **ARLO**: Arbitration lost (master mode)

- 0: No Arbitration lost detected
- 1: Arbitration lost detected
  - Set by hardware when the interface loses the arbitration of the bus to another master.
  - Cleared by software writing 0, or by hardware when PE=0.

After an ARLO event the interface switches back automatically to Slave mode (MSL=0).

Bit 0 **BERR**: Bus error

- 0: No misplaced Start or Stop condition
- 1: Misplaced Start or Stop condition
  - Set by hardware when the interface detects a SDA rising or falling edge while SCL is high, occurring in a non-valid position during a byte transfer.
  - Cleared by software writing 0, or by hardware when PE=0.

**28.7.10 Status register 3 (I<sup>2</sup>C\_SR3)**

Address offset: 0x09

Reset value: 0x00

7	6	5	4	3	2	1	0
DUALF	SMBHOST	SMBDEFAULT	GENCALL	Reserved	TRA	BUSY	MSL
r	r	r	r		r	r	r

Bit 7<sup>(1)</sup> **DUALF**: Dual flag (Slave mode)

- 0: Received address matched with OAR1
- 1: Received address matched with OAR2
  - Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 6 **SMBHOST**: SMBus Host header (Slave mode)

- 0: No SMBus Host address
- 1: SMBus Host address received when SMBTYPE=1 and ENARP=1.
  - Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 5 **SMBDEFAULT**: SMBus device default address (Slave mode)

- 0: No SMBus Device Default address
- 1: SMBus Device Default address received when ENARP=1
  - Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 4 **GENDCALL**: General call header (Slave mode)

- 0: No general call
- 1: General call header received when ENGC=1
  - Cleared by hardware after a Stop condition or repeated Start condition, or when PE=0.

Bit 3 Reserved

Bit 2 **TRA**: Transmitter/Receiver

- 0: Data bytes received
  - 1: Data bytes transmitted
- This bit is set depending on R/W bit of address byte, at the end of total address phase.  
It is also cleared by hardware after detection of Stop condition (STOPF=1), repeated Start condition, loss of bus arbitration (ARLO=1), or when PE=0.

Bit 1 **BUSY**: Bus busy

- 0: No communication on the bus
- 1: Communication ongoing on the bus
  - Set by hardware on detection of SDA or SCL low
  - cleared by hardware on detection of a Stop condition.

It indicates a communication in progress on the bus. This information is still updated when the interface is disabled (PE=0).

Bit 0 **MSL**: Master/Slave

- 0: Slave mode
- 1: Master mode
  - Set by hardware as soon as the interface is in Master mode (SB=1).
  - Cleared by hardware after detecting a Stop condition on the bus or a loss of arbitration (ARLO=1), or by hardware when PE=0.

1. This bit is available in medium+ and high density devices only (reserved in medium density devices).

**Note:** *Reading I2C\_SR3 after reading I2C\_SR1 clears the ADDR flag, even if the ADDR flag was set after reading I2C\_SR1. Consequently, I2C\_SR3 must be read only when ADDR is found set in I2C\_SR1 or when the STOPF bit is cleared.*

### 28.7.11 Interrupt and DMA register (I2C\_ITR)

Address offset: 0x0A

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN
			rw	rw	rw	rw	rw

Bits 7:5 Reserved.

Bit 4 **LAST**: DMA last transfer

- 0: Next DMA EOT is not the last transfer
- 1: Next DMA EOT is the last transfer

*Note:* This bit is used in master receiver mode to permit the generation of a NACK on the last received data.

Bit 3 **DMAEN**: DMA requests enable

- 0: DMA requests disabled
- 1: DMA request enabled when TxE=1 or RxNE =1

Bit 2 **ITBUFEN**: Buffer interrupt enable

- 0: TXE = 1 or RXNE = 1 does not generate any interrupt.
- 1:TXE = 1 or RXNE = 1 generates Event interrupt.

Bit 1 **ITEVTEN**: Event interrupt enable

- 0: Event interrupt disabled
- 1: Event interrupt enabled

This interrupt is generated when:

- SB = 1 (Master)
- ADDR = 1 (Master/Slave)
- ADD10= 1 (Master)
- STOPF = 1 (Slave)
- BTF = 1 with no TXE or RXNE event
- TXE event to 1 if ITBUFEN = 1
- RXNE event to 1if ITBUFEN = 1
- WUFH = 1 (asynchronous interrupt to wakeup from Halt)

Bit 0 **ITERREN**: Error interrupt enable

- 0: Error interrupt disabled
- 1: Error interrupt enabled

– This interrupt is generated when:

- BERR = 1
- ARLO = 1
- AF = 1
- OVR = 1
- PECERR = 1
- TIMEOUT = 1
- SMBALERT = 1

### 28.7.12 Clock control register low (I2C\_CCRL)

Address offset: 0x02

Reset value: 0x0B

7	6	5	4	3	2	1	0
CCR[7:0]							
rw							

Bits 7:0 **CCR[7:0]** Clock control register (Master mode)

Controls the SCLH clock in Master mode.

- Standard mode or SMBus:

$$\text{Period(I2C)} = 2 * \text{CCR} * t_{MASTER}$$

$$t_{high} = \text{CCR} * t_{MASTER}$$

$$t_{low} = \text{CCR} * t_{MASTER}$$

- Fast mode:

If DUTY = 0:

$$\text{Period(I2C)} = 3 * \text{CCR} * t_{MASTER}$$

$$t_{high} = \text{CCR} * t_{MASTER}$$

$$t_{low} = 2 * \text{CCR} * t_{MASTER}$$

If DUTY = 1: (to reach 400 kHz)

$$\text{Period(I2C)} = 25 * \text{CCR} * t_{MASTER}$$

$$t_{high} = 9 * \text{CCR} * t_{MASTER}$$

$$t_{low} = 16 * \text{CCR} * t_{MASTER}$$

Note:  $t_{CK} = 1/f_{MASTER}$ .  $f_{MASTER}$  is the input clock to the peripheral configured using clock control register.

The minimum allowed value is 04h, except in FAST DUTY mode where the minimum allowed value is 0x01.

$t_{high} = t_{r(SCL)} + t_{w(SCLH)}$ . See device datasheet for the definitions of parameters.

$t_{low} = t_{f(SCL)} + t_{w(SCLL)}$ . See device datasheet for the definitions of parameters.

I<sup>2</sup>C communication speed,  $f_{SCL} = 1/(t_{high} + t_{low})$

These timings are without filters.

### 28.7.13 Clock control register high (I<sup>2</sup>C\_CCRH)

Address offset: 0x0C

Reset value: 0x00

7	6	5	4	3	2	1	0
F/S	DUTY	Reserved		CCR[11:8]			
rw	rw			rw			

Bit 7 **F/S**: I<sup>2</sup>C master mode selection

0: Standard mode I<sup>2</sup>C

1: Fast mode I<sup>2</sup>C

Bit 6 **DUTY**: Fast mode duty cycle

0: Fast mode  $t_{low}/t_{high} = 2$

1: Fast mode  $t_{low}/t_{high} = 16/9$  (see CCR)

Bits 5:4 Reserved

Bits 3:0 **CCR[11:8]**: Clock control register in Fast/Standard mode (Master mode)<sup>(1)</sup>

Controls the SCLH clock in master mode.

– Standard mode:

$$\text{Period(I2C)} = 2 * \text{CCR} * t_{MASTER}$$

$$t_{high} = \text{CCR} * t_{MASTER}$$

$$t_{low} = \text{CCR} * t_{MASTER}$$

– Fast mode:

If DUTY = 0:

$$\text{Period(I2C)} = 3 * \text{CCR} * t_{MASTER}$$

$$t_{high} = \text{CCR} * t_{MASTER}$$

$$t_{low} = 2 * \text{CCR} * t_{MASTER}$$

If DUTY = 1: (to reach 400 kHz)

$$\text{Period(I2C)} = 25 * \text{CCR} * t_{MASTER}$$

$$t_{high} = 9 * \text{CCR} * t_{MASTER}$$

$$t_{low} = 16 * \text{CCR} * t_{MASTER}$$

For instance: in standard mode, to generate a 100 kHz SCL frequency:

If FREQR = 08,  $t_{MASTER} = 125$  ns so CCR must be programmed with 0x28

$$(0x28 \Leftrightarrow 40 \times 125 \text{ ns} = 5000 \text{ ns})$$

*Note:  $t_{high} = t_{f(SCL)} + t_{w(SCLH)}$ . See device datasheet for the definitions of parameters*

*$t_{low} = t_{f(SCL)} + t_{w(SCL)}$ . See device datasheet for the definitions of parameters*

*These timings are without filters.*

- Refer to [Table 93: I<sup>2</sup>C\\_CCR values for SCL frequency table \(fSYSCLK= 10 MHz or 16 MHz\) on page 500](#)

**Note:** The CCR registers must be configured only when the I<sup>2</sup>C is disabled (PE=0).

$f_{MASTER}$  = multiple of 10 MHz is required to generate Fast clock at 400 kHz.

$f_{MASTER} \geq 1$  MHz is required to generate Standard clock at 100 kHz.

**Table 93. I2C\_CCR values for SCL frequency table ( $f_{SYSCLK} = 10$  MHz or 16 MHz<sup>(1)</sup>)**

I2C Speed	I2C frequency ( $f_{SCL}$ ) in Hz	$f_{SYSCLK} = 10$ MHz				$f_{SYSCLK} = 16$ MHz			
		Actual (Hz)	% Error (%)	I2C_CCR (h)	Duty cycle bit	Actual (Hz)	% Error (%)	I2C_CCR (h)	Duty cycle bit
Fast speed	400000	400000	0	1	1	410256.41	2.56	D	0
	370000	370370.37	0.10	9	0	380952.38	2.96	E	0
	350000	370370.37	5.82	9	0	355555.56	1.59	F	0
	320000	333333.33	4.17	A	0	320000	0	2	1
	300000	303030.30	1.01	B	0	313725.49	4.57	11	0
	270000	277777.78	2.88	C	0	280701.75	3.96	13	0
	250000	256410.26	2.56	D	0	253968.25	1.59	15	0
	220000	222222.22	1.01	F	0	222222.22	1.01	18	0
	200000	200000	0	2	1	205128.20	2.56	1A	0
	170000	175438.60	3.20	13	0	172043.01	1.20	1F	0
	150000	151515.15	1.01	16	0	152380.95	1.59	23	0
	120000	123456.79	2.88	1B	0	121212.12	1.01	2C	0
Standard speed	100000	100000	0	32	No impact	100000	0	50	No impact
	50000	50000	0	64		50000	0	A0	
	30000	30120.48	0.40	A6		30075.19	0.25	10A	
	20000	20000	0	FA		20000	0	190	

1. The following table gives the values to be written in the I2C\_CCR register to obtain the required I<sup>2</sup>C SCL line frequency

### 28.7.14 TRISE register (I2C\_TRISER)

Address offset: 0x0D

Reset value: 0x02

7	6	5	4	3	2	1	0
Reserved		TRISE[5:0]					
		rw					

Bits 7:6 Reserved

Bits 5:0 **TRISE[5:0]** Maximum rise time in Fast/Standard mode (Master mode)

These bits must be programmed with the maximum SCL rise time given in the I<sup>2</sup>C bus specification, incremented by 1.

For instance: in standard mode, the maximum allowed SCL rise time is 1000 ns.

If the value in the I<sup>2</sup>C\_FREQR register = 08h, then  $t_{MASTER} = 125$  ns therefore the TRISE[5:0] bits must be programmed with 0x09.

(1000 ns / 125 ns = 8 + 1)

The filter value can also be added to TRISE[5:0].

If the result is not an integer, TRISE[5:0] must be programmed with the integer part, in order to respect the  $t_{HIGH}$  parameter.

*Note: TRISE[5:0] must be configured only when the I<sup>2</sup>C is disabled (PE = 0).*

### 28.7.15 PEC register (I<sup>2</sup>C\_PECR)

Address offset: 0x0E

Reset value: 0x00

7	6	5	4	3	2	1	0
PEC[7:0]							
rw							

Bits 7:0 **PEC[7:0]** Packet error checking register

This register contains the internal PEC when ENPEC=1.

### 28.7.16 I<sup>2</sup>C register map and reset values

**Table 94. I<sup>2</sup>C register map**

Address offset	Register name	7	6	5	4	3	2	1	0		
0x00	I <sup>2</sup> C_CR1 Reset value	NO STRETCH 0	ENGC 0	ENPEC 0	ENARP 0	SMBTYPE 0	I <sup>2</sup> CLOAD 0	SMBUS 0	PE 0		
0x01	I <sup>2</sup> C_CR2 Reset value	SWRST 0	- 0	ALERT 0	PEC 0	POS 0	ACK 0	STOP 0	START 0		
0x02	I <sup>2</sup> C_FREQR Reset value	- 0	- 0	FREQ5 0	FREQ4 0	FREQ3 0	FREQ2 0	FREQ1 0	FREQ0 0		
0x03	I <sup>2</sup> C_OARL Reset value	ADD[7:1] 0							ADD0 0		
0x04	I <sup>2</sup> C_OARH Reset value	ADDMODE 0	ADDCONF 0	- 0	- 0	- 0	ADD1[9:8] 0		- 0		
0x05 <sup>(1)</sup>	I <sup>2</sup> C_OAR2 Reset value	ADD2[7:1] 0							ENDUAL 0		
0x06	I <sup>2</sup> C_DR Reset value	DR[7:0] 0									
0x07	I <sup>2</sup> C_SR1 Reset value	TXE 0	RXNE 0	- 0	STOPF 0	ADD10 0	BTF 0	ADDR 0	SB 0		
0x08	I <sup>2</sup> C_SR2 Reset value	SMBALERT 0	TIMEOUT 0	WUFH 0	PECERR 0	OVR 0	AF 0	ARLO 0	BERR 0		
0x09	I <sup>2</sup> C_SR3 Reset value	DUALF <sup>(1)</sup> 0	SMBHOST 0	SMBDEFAULT 0	GENCALL 0	- 0	TRA 0	BUSY 0	MSL 0		

**Table 94. I<sup>2</sup>C register map (continued)**

<b>Address offset</b>	<b>Register name</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
0x0A	I2C_ITR Reset value	- 0	- 0	- 0	LAST 0	DMAEN 0	ITBUFEN 0	ITEVTEN 0	ITERREN 0
0x0B	I2C_CCRL Reset value				CCR[7:0] 0				
0x0C	I2C_CCRH Reset value	FS 0	DUTY 0	- 0	- 0		CCR[11:8] 0		
0x0D	I2C_TRISER Reset value	- 0	- 0		TRISE[5:0] 0				
0x0E	I2C_PECR Reset value	PEC7 0	PEC6 0	PEC5 0	PEC4 0	PEC3 0	PEC2 0	PEC1 0	PEC0 0

1. Available in medium+ and high density devices only. Reserved in medium density devices.

## 29 Universal synchronous/asynchronous receiver transmitter (USART)

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

### 29.1 USART introduction

The USART (universal synchronous asynchronous receiver transmitter) offers a flexible means of full-duplex data exchange with external equipment requiring an industry standard NRZ asynchronous serial data format. It offers a very wide range of baud rates.

The USART supports synchronous one-way communication and half-duplex single wire communication. Smartcard Protocol and IrDA (infrared data association) SIR ENDEC specifications are also supported. The USART can also be used for multiprocessor communication. High-speed data communication is possible by using the DMA for multibuffer configuration.

### 29.2 USART main features

- Full duplex, asynchronous communications
- NRZ standard format (Mark/Space)
- High-precision baud rate generator system
  - Common programmable transmit and receive baud rates up to  $f_{SYSCLK}/16$
- Programmable data word length (8 or 9 bits)
- Configurable STOP bits - support for 1 or 2 STOP bits
- Transmitter clock output for synchronous communication
- Single wire Half duplex communication
- IrDA SIR Encoder Decoder
  - Support for 3/16 bit duration for normal mode
- Smartcard emulation capability
  - The Smartcard interface supports the asynchronous protocol for Smartcards as defined in ISO 7816-3 standards
  - 1.5 Stop bits for Smartcard operation
- Configurable multibuffer communication using DMA (direct memory access)
  - Buffering of received/transmitted bytes in reserved RAM using centralized DMA
- Separate enable bits for Transmitter and Receiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - End of Transmission flags
- Parity control:

- Transmits parity bit
- Checks parity of received data byte
- 4 error detection flags:
  - Overrun error
  - Noise error
  - Frame error
  - Parity error
- 8 interrupt sources with flags:
  - Transmit data register empty
  - Transmission complete
  - Receive data register full
  - Idle line received
  - Parity error
  - Overrun error
  - Framing error
  - Noise error
- 2 interrupt vectors:
  - Transmitter interrupt
  - Receiver interrupt
- Reduced power consumption mode
- Multi-Processor communication - enter into mute mode if address match does not occur
- Wakeup from mute mode (by idle line detection or address mark detection)
- 2 receiver wakeup modes:
  - Address bit (MSB)
  - Idle line

## 29.3 USART functional description

The interface is externally connected to another device by three pins (see [Figure 152](#)). Any USART bidirectional communication requires a minimum of two pins: USART Receive data input (USART\_RX) and USART transmit data output (USART\_TX):

**USART\_RX** is the serial data input. Over-sampling techniques are used for data recovery by discriminating between valid incoming data and noise.

**USART\_TX** is the serial 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 pin is at high level.

Through these pins, serial data is transmitted and received in normal USART mode as frames including:

- An Idle Line prior to transmission or reception
- A start bit
- A data word (8 or 9 bits) least significant bit first
- 1, 1.5 and 2 STOP bits indicating that the frame is complete
- A status register (USART\_SR)
- Data register (USART\_DR)
- 16-bit baud rate prescaler (USART\_BRR)
- Guard time register for use in Smartcard mode

Refer to the register description for the definitions of each bit.

The following pin is required to interface in synchronous mode:

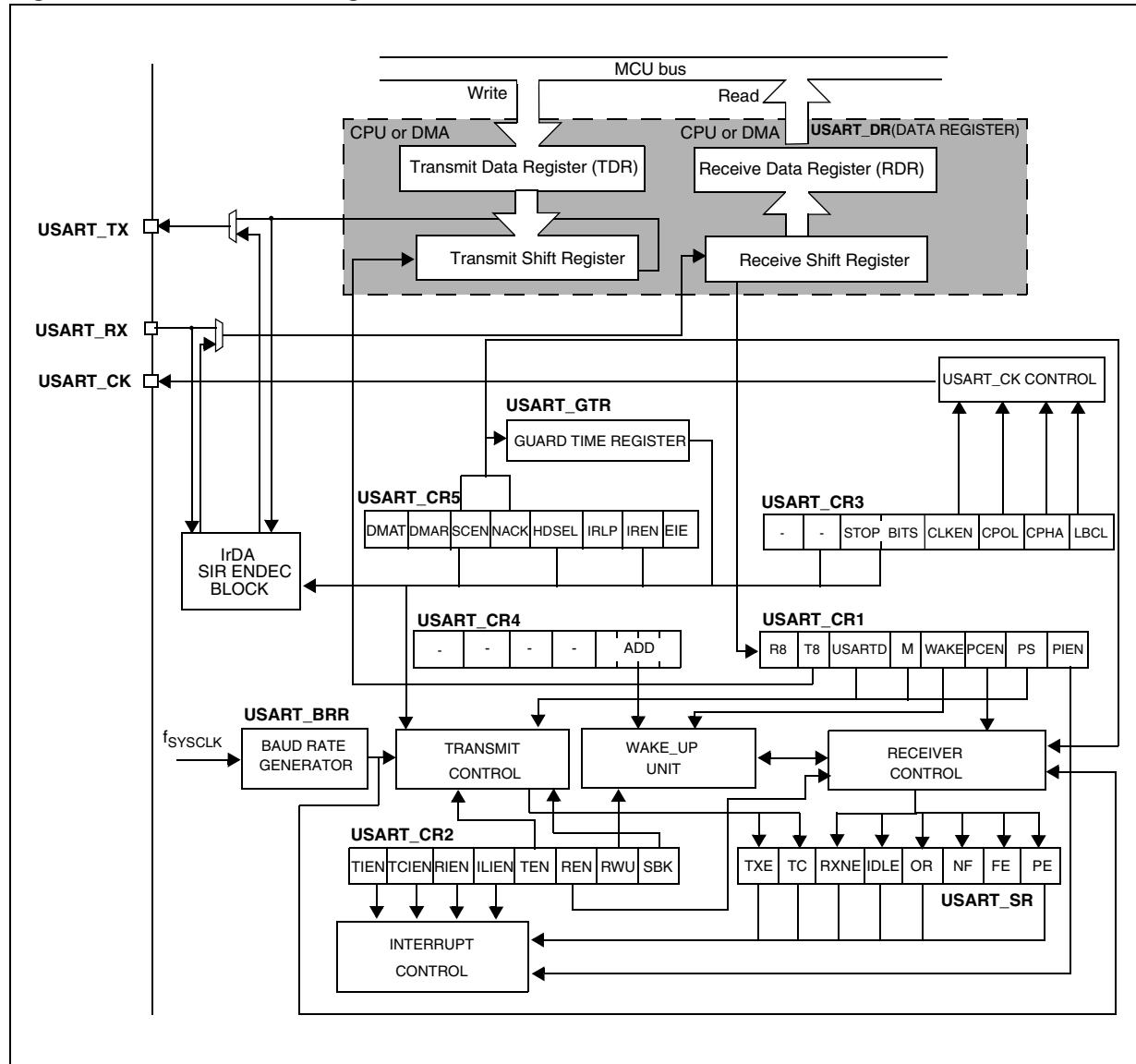
**USART\_CK**: Transmitter clock output. This pin outputs the transmitter data clock for synchronous transmission (no clock pulses on start bit and STOP bit, and a software option to send a clock pulse on the last data bit). This can be used to control peripherals that have shift registers (e.g. LCD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

The USART\_RX and USART\_TX pins are used in IrDA mode as follows:

**USART\_RX** = IrDA\_RDI: Receive Data Input in IrDA mode

**USART\_TX** = IrDA\_TDO: Transmit Data Output in IrDA mode

Figure 152. USART block diagram



### 29.3.1 USART character description

Word length may be selected as being either 8 or 9 bits by programming the M bit in the USART\_CR1 register (see [Figure 153](#)).

The USART\_TX pin is in low state during the start bit. It is in high state during the STOP bit.

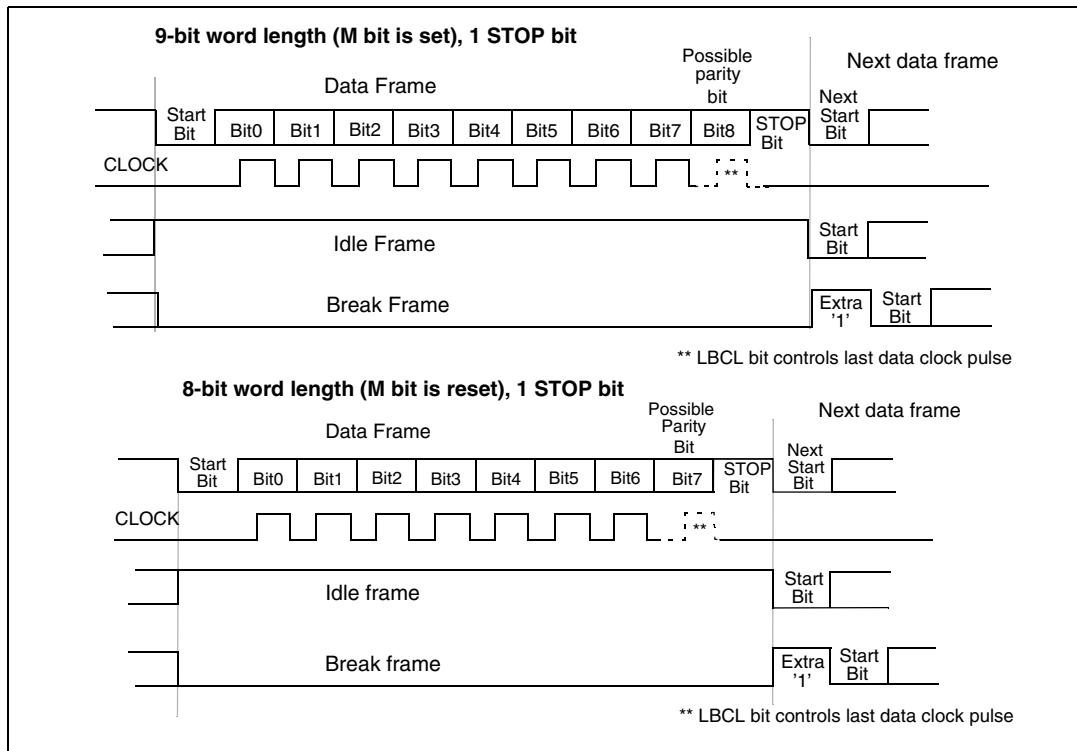
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1” ‘s includes the start bit, the number of data bits and the number of STOP bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame the transmitter inserts either 1 or 2 STOP bits (logic “1” bit) to acknowledge the start bit.

Transmission and reception are driven by a common baud rate generator, the clock for each is generated when the enable bit is set respectively for the transmitter and receiver.

The details of each block is given below.

**Figure 153. Word length programming**



### 29.3.2 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 USART\_CR1 register.

When the transmit enable bit (TEN) is set, the data in the transmit shift register is output on the USART\_TX pin and the corresponding clock pulses are output on the USART\_CK pin.

#### Character transmission

During a USART transmission, data shifts out least significant bit first on the USART\_TX pin. In this mode, the USART\_DR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see [Figure 152](#)).

Every character is preceded by a start bit which is a logic level low for one bit period. The character is terminated by a configurable number of STOP bits.

The following STOP bits are supported by USART.

- Note:*
- 1 *The TEN bit should not be reset during transmission of data. Resetting the TEN bit during the transmission will corrupt the data on the USART\_TX pin as the baud rate counters will get frozen. The current data being transmitted will be lost.*
  - 2 *An idle frame will be sent after the TEN bit is enabled.*

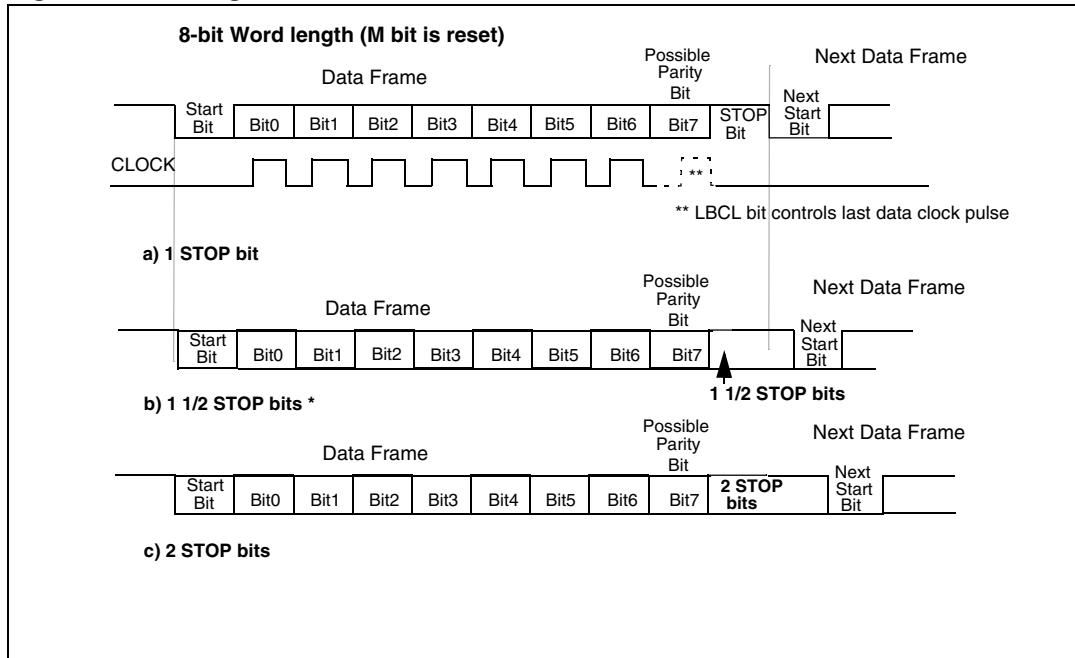
#### Configurable STOP bits during transmission

The number of STOP bits to be transmitted with every character can be programmed in Control register 3, bits 5,4.

- 1 STOP bit: This is the default value of number of STOP bits.
- 2 STOP bits: This will be supported by normal mode USART.
- 1.5 STOP bits: To be used in Smartcard mode.

An idle frame transmission will include the STOP bits.

A break transmission consists of 10 low bits followed by the configured number of STOP bits (when m = 0) and 11 low bits followed by the configured number of STOP bits (when m = 1). It is not possible to transmit long breaks (break of length greater than 10/11 low bits).

**Figure 154. Configurable STOP bits**

### Procedure

1. Program the M bit in USART\_CR1 to define the word length.
2. Program the number of STOP bits in USART\_CR3.
3. Select DMA enable (DMAT) in USART\_CR5 if Multibuffer communication is to take place. Configure the DMA register as explained in [Section 29.3.12: Continuous communication using DMA on page 527](#).
4. Select the desired baud rate by programming the baud rate registers in the following order:
  - a) USART\_BRR2
  - b) USART\_BRR1
5. Set the TEN bit in USART\_CR2 to enable transmitter mode.
6. Write the data to send in the USART\_DR register (this clears the TXE bit). Repeat this for each data to be transmitted in case of single buffer.
7. Once the last data is written to the USART\_DR register, wait until TC is set to '1', which indicates that the last data transmission is complete. This last step is required, for instance, to avoid last data transmission corruption when disabling the USART or entering Halt mode.

### Single byte communication

Clearing the TXE bit is always performed by a write to the data register.

The TXE bit is set by hardware and it indicates:

- The data has been moved from TDR to the shift register and the data transmission has started.
- The TDR register is empty.
- The next data can be written in the USART\_DR register without overwriting the previous data.

This flag generates an interrupt if the TIEN bit is set.

When a transmission is taking place, a write instruction to the USART\_DR register stores the data in the TDR register. The data is copied in the shift register at the end of the current transmission.

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

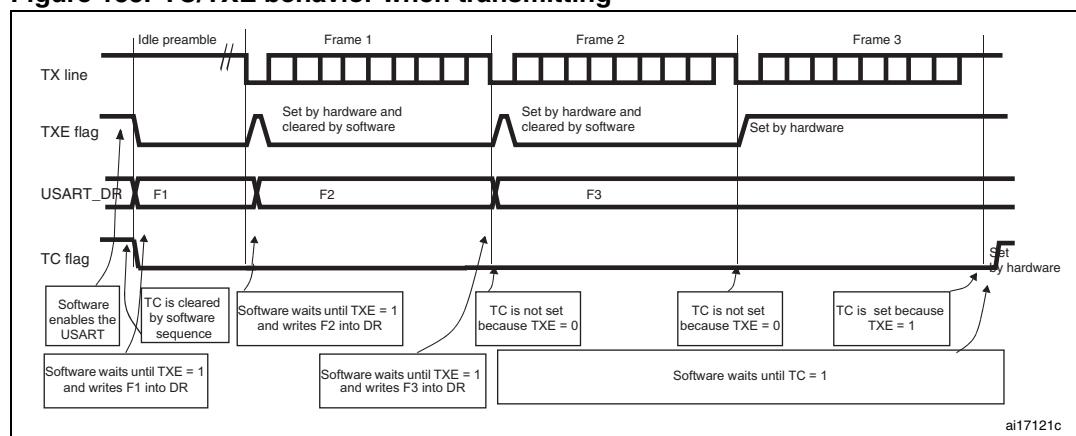
If a frame transmission is complete (after the stop bit) and the TXE bit is set, the TC bit is set. An interrupt is generated if the TCien is set in the USART\_CR2 register. After writing the last data into the USART\_DR register, it is mandatory to wait until TC is set to '1' before entering Halt mode or disabling the USART (see [Figure 155: TC/TXE behavior when transmitting](#)).

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

1. A read to the USART\_SR register
2. A write to the USART\_DR register

*Note:* The TC bit can also be cleared by writing a '0' to it. This clearing sequence is recommended only for multibuffer communication.

**Figure 155. TC/TXE behavior when transmitting**



1. This example assumes that several other transmissions occurred after TE has been set. Otherwise an IDLE preamble would be transmitted first when writing to USART\_DR for the first time.

### Break character

Setting the SBK bit transmits a break character. The break frame length depends on the M bit (see [Figure 153](#)).

If the SBK bit is set to '1' a break character is sent on the USART\_TX line after completing the current character transmission. This bit is reset by hardware when the break character is

completed (during the STOP bit of the break character). The USART inserts a logic 1 bit at the end of the last break frame to guarantee the recognition of the start bit of the next frame.

**Note:** *The break character is sent without taking into account the number of STOP bits. If the USART is programmed with 2 STOP bits, the Tx line is pulled low until the end of the first STOP bit only. Then 2 logic 1 bits are inserted before the next character.*

**Note:** *If the software resets the SBK bit before the start of break transmission, the break character is not transmitted. For two consecutive breaks, the SBK bit should be set after the STOP bit of the previous break.*

### Idle character

Setting the TEN bit drives the USART to send an idle frame before the first data frame.

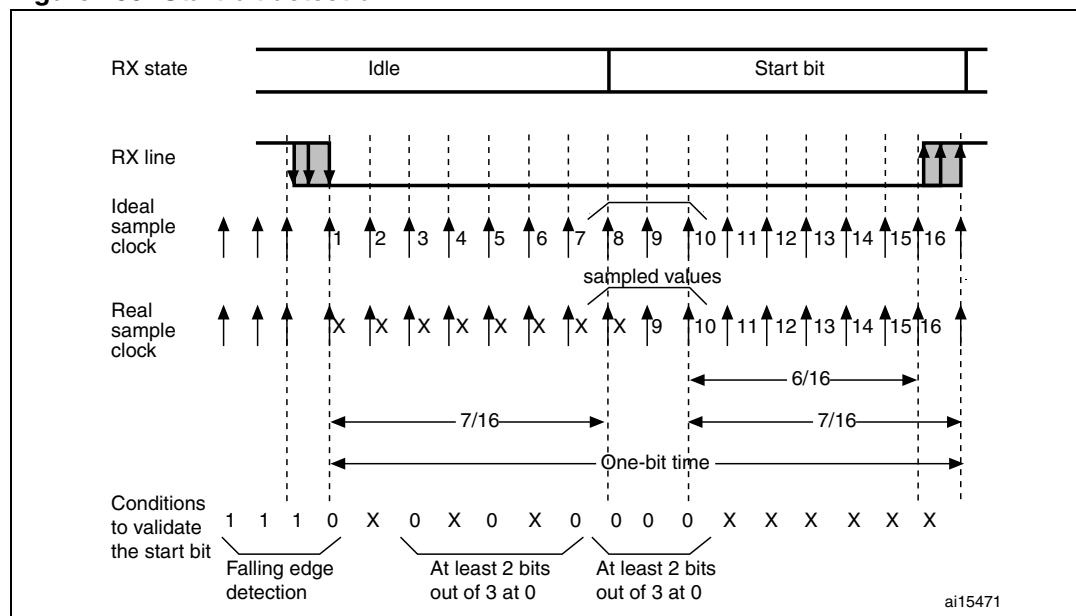
## 29.3.3 Receiver

The USART 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 USART\_CR1 register.

### Start bit detection

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0. The start bit detection sequence shown in [Figure 156](#).

**Figure 156. Start bit detection**



**Note:** *If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.*

If only 2 out of the 3 bits are at 0 (sampling on the 3<sup>rd</sup>, 5<sup>th</sup> and 7<sup>th</sup> bits or sampling on the 8<sup>th</sup>, 9<sup>th</sup> and 10<sup>th</sup> bits), the start bit is validated but the NF noise flag bit is set.

The start bit is confirmed if the last 3 samples are at 0 (sampling on the 8<sup>th</sup>, 9<sup>th</sup>, and 10<sup>th</sup> bits).

## Character reception

During a USART reception, data shifts in least significant bit first through the USART\_RX pin. In this mode, the USART\_DR register consists of a buffer (RDR) between the internal bus and the received shift register (see [Figure 2](#)).

Procedure:

1. Program the M bit in USART\_CR1 to define the word length.
2. Program the number of STOP bits in USART\_CR3.
3. Select DMA enable (DMAR) in USART\_CR5 if multibuffer communication is to take place. Configure the DMA register as explained in [Section 29.3.12: Continuous communication using DMA on page 527](#).
4. Select the desired baud rate by programming the baud rate registers in the following order:
  - a) USART\_BRR2
  - b) USART\_BRR1
5. Set the REN bit USART\_CR2. This enables the receiver which begins searching for a start bit.

When a character is received

- The RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR.
- An interrupt is generated if the RIEN bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.
- In multibuffer, the RXNE bit is set after every byte received and is cleared by the DMA read to the Data Register.
- In single buffer mode, clearing the RXNE bit is performed by a software read to the USART\_DR register. The RXNE flag can also be cleared by writing a zero to it. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.

Note:

*The REN bit should not be reset while receiving data. If the REN bit is disabled during reception, the reception of the current byte will be aborted.*

## Break character

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

## Idle character

When an idle frame is detected, there is the same procedure as a received data character plus an interrupt if the ILIEN bit is set.

## Overrun error

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

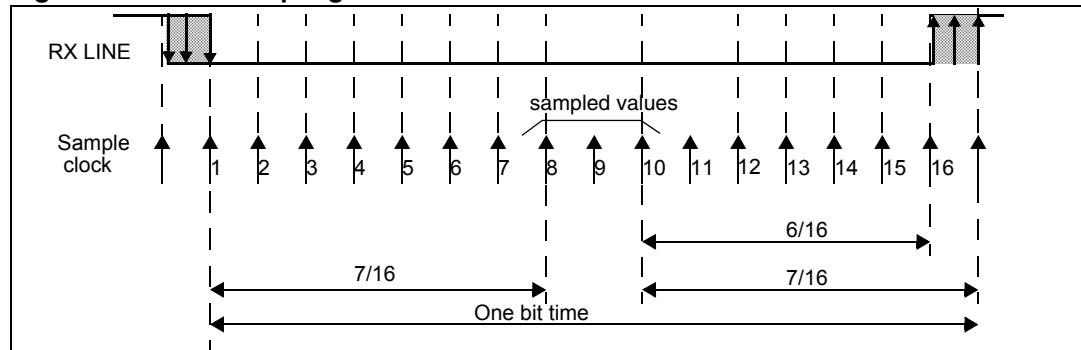
The RXNE flag is set after every byte received. An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

- The OR bit is set.
- The RDR content will not be lost. The previous data is available when a read to USART\_DR is performed.
- The shift register will be overwritten. The second data received during overrun is lost.
- An interrupt is generated if the RIEN bit is set or both the EIE and DMAR bits are set.
- The OR bit is reset by a read to the USART\_SR register followed by a USART\_DR register read operation.

### Noise error

Over-sampling techniques are used for data recovery by discriminating between valid incoming data and noise.

**Figure 157. Data sampling for noise detection**



Note: The sample clock frequency is 16x baud rate.

**Table 95. Noise detection from sampled data**

Sampled value	NF status	Received bit value	Data validity
000	0	0	Valid
001	1	0	Not Valid
010	1	0	Not Valid
011	1	1	Not Valid
100	1	0	Not Valid
101	1	1	Not Valid
110	1	1	Not Valid
111	0	1	Valid

When noise is detected in a frame:

- The NF is set at the rising edge of the RXNE bit.
- The invalid data is transferred from the Shift register to the USART\_DR register.

This bit rises at the same time as the RXNE bit which generates an interrupt. The NF bit is reset by a USART\_SR register read operation followed by a USART\_DR 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.

When the framing error is detected:

- The FE bit is set by hardware
- The invalid data is transferred from the Shift register to the USART\_DR register.
- No interrupt is generated in case of single byte communication. However, this bit rises at the same time as the RXNE bit which itself generates an interrupt.

The FE bit is reset by a USART\_SR register read operation followed by a USART\_DR register read operation.

### Configurable STOP bits during reception

The number of STOP bits to be received can be configured through the control bits of Control Register 3 - it can be either 1 or 2 in normal mode, 1 in IrDA mode and 1.5 in Smartcard mode.

1. 1 STOP bit: Sampling for 1 STOP bit is done on the 8th, 9th and 10th samples.
2. 1.5 STOP bits (Smartcard mode only): Sampling for 1.5 STOP bits is done on the 16th, 17th and 18th samples. An NACK signal received from the Smartcard forces the data signal low during the sampling, flagged as a framing error. Then, the FE flag is set with the RXNE at the end of the 1.5 STOP bit.
3. 2 STOP bits: Sampling for 2 STOP bits is done on the 8th, 9th and 10th samples of the first STOP bit. If a framing error is detected during the first STOP bit the framing error flag will be set. The second STOP bit is not checked for framing error. The RXNE flag will be set at the end of the first STOP bit.

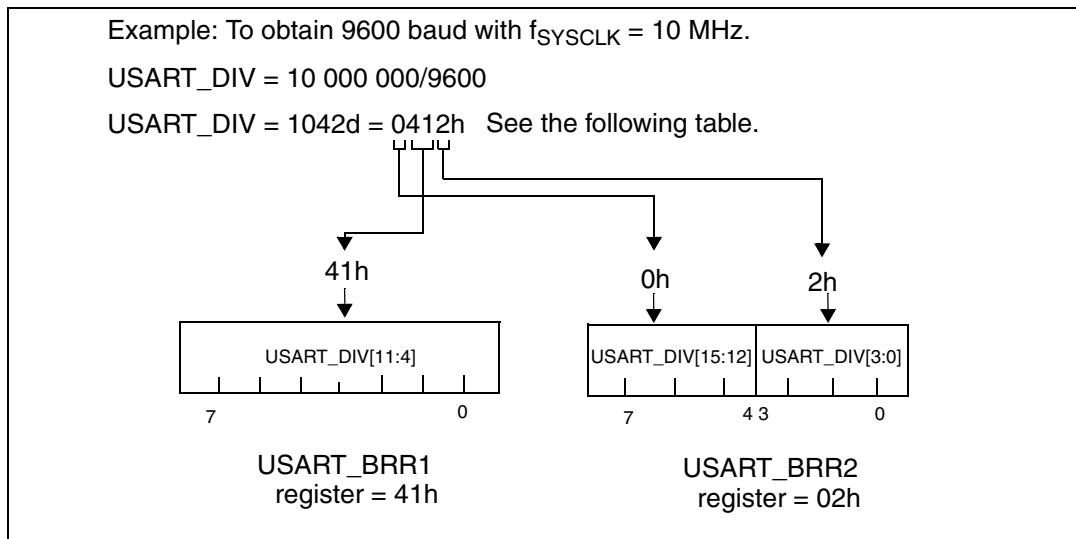
#### 29.3.4 High precision baud rate generator

The receiver and transmitter (Rx and Tx) are both set to the same baud rate programmed by a 16-bit divider USART\_DIV according to the following formula:

$$\text{Tx/ Rx baud rate} = \frac{f_{\text{SYSCLK}}}{\text{USART\_DIV}}$$

The USART\_DIV baud rate divider is an unsigned integer, coded in the BRR1 and BRR2 registers as shown in [Figure 158](#).

Refer to [Table 96](#) for typical baud rate programming examples.

**Figure 158. How to code USART\_DIV in the BRR registers**

**Note:** The Baud Counters will be updated with the new value of the Baud Registers after a write to BRR1. Hence the Baud Register value should not be changed during a transaction. The BRR2 should be programmed before BRR1.

**Note:** USART\_DIV must be greater than or equal to 16d.

**Table 96. Baud rate programming and error calculation**

		$f_{SYSCLK} = 10$ MHz				$f_{SYSCLK} = 16$ MHz				
In bps	Actual (bps)	% Error (%) <sup>(1)</sup>	USART_DIV (h)	BRR1 (h)	BRR2 (h)	Actual	% Error <sup>(1)</sup>	USART_DIV	BRR1	BRR2
2400	2399.81	-0.008	1047	4	17	2399.88	-0.005	1A0B	A0	1B
9600	9596.93	-0.03	412	41	2	9598.08	-0.02	683	68	3
19200	19193.86	-0.03	209	20	9	19207.68	0.04	341	34	1
57600	57471.26	-0.22	AE	A	E	57553.96	-0.08	116	11	6
115200	114942.53	-0.22	57	5	7	115107.91	-0.08	8B	8	B
230400	232558.14	0.94	2B	2	B	231884.06	0.64	45	4	5
460800	454545.45	-1.36	16	1	6	457142.86	-0.79	23	2	3
921600	NA	NA	NA	NA	NA	941176.47	2.12	11	1	1

1. Error % = (Calculated - Desired) Baud Rate / Desired Baud Rate

**Table 97. Baud rate programming and error calculation**

Baud rate	$f_{SYSCLK} = 10 \text{ MHz}$					$f_{SYSCLK} = 16 \text{ MHz}$				
	In bps	Actual (bps)	% Error (%) <sup>(1)</sup>	USART_DIV (h)	BRR1 (h)	BRR2 (h)	Actual	% Error <sup>(1)</sup>	USART_DIV	BRR1
2400	2399.81	-0.008	1047	4	17	2399.88	-0.005	1A0B	A0	1B
9600	9596.93	-0.03	412	41	2	9598.08	-0.02	683	68	3
19200	19193.86	-0.03	209	20	9	19207.68	0.04	341	34	1
57600	57471.26	-0.22	AE	A	E	57553.96	-0.08	116	11	6
115200	114942.53	-0.22	57	5	7	115107.91	-0.08	8B	8	B
230400	232558.14	0.94	2B	2	B	231884.06	0.64	45	4	5
460800	454545.45	-1.36	16	1	6	457142.86	-0.79	23	2	3
921600	NA	NA	NA	NA	NA	941176.47	2.12	11	1	1

1. Error % = (Calculated - Desired) Baud Rate / Desired Baud Rate

**Table 98. Baud rate programming and error calculation**

Baud rate	$f_{MASTER} = 10 \text{ MHz}$					$f_{MASTER} = 16 \text{ MHz}$				
	In bps	Actual	% Error (%) <sup>(1)</sup>	USART_DIV	BRR1	BRR2	Actual	% Error <sup>(1)</sup>	USART_DIV	BRR1
2.4	2.399	-0.04	1047h	04h	17h	2.4	0.0	2710h	71h	20h
9.6	9.596	-0.04	0412h	41h	02h	9.6	0.0	09C4h	9Ch	04h
19.2	19.193	-0.03	0209h	20h	09h	19.2	0.0	04E2h	4Eh	02h
57.6	57.471	-0.22	00AEh	0Ah	0Eh	57.554	-0.08	01A1h	1Ah	01h
115.2	114.942	-0.22	0057h	05h	07h	115.385	0.16	00D0h	0Dh	00h
230.4	232.558	-0.94	002Bh	02h	0Bh	230.769	0.16	0068h	06h	08h
460.8	454.545	-1.36	0016h	01h	06h	461.538	0.16	0034h	03h	04h
921.6	NA	NA	NA			923.077	0.16	001Ah	01h	0Ah

1. Error % = (Calculated - Desired) Baud Rate / Desired Baud Rate

**Note:** The lower the  $f_{SYSCLK}$  frequency, the lower will be the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with this data.

### 29.3.5 USART receiver's tolerance to clock deviation

The USART's asynchronous receiver works correctly only if the total clock system deviation is smaller than the USART receiver's tolerance. The causes which contribute to the total deviation are:

- DTRA: Deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: Error due to the baud rate quantization of the receiver
- DREC: Deviation of the receiver's local oscillator
- DTCL: Deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$\text{DTRA} + \text{DQUANT} + \text{DREC} + \text{DTCL} < \text{USART receiver's tolerance}$$

The USART receiver's tolerance to properly receive data is equal to the maximum tolerated deviation and depends on the following choices:

- 10- or 11-bit character length defined by the M bit in the USART\_CR1 register
- Use of fractional baud rate or not

**Table 99. USART receiver's tolerance when USART\_DIV[3:0] is 0**

M bit	NF is an error	NF is don't care
0	3.75%	4.375%
1	3.41%	3.97%

**Table 100. USART receiver's tolerance when USART\_DIV[3:0] is different from 0**

M bit	NF is an error	NF is don't care
0	3.33%	3.88%
1	3.03%	3.53%

Note:

The figures specified in [Table 99](#) and [Table 100](#) may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M=0 (11-bit times when M=1).

### 29.3.6 Parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCEN bit in the USART\_CR1 register. Depending on the frame length defined by the M bit, the possible USART frame formats are as listed in [Table 101](#).

**Table 101. Frame formats**

M bit	PCEN bit	USART frame
0	0	SB   8 bit data   STB
0	1	SB   7-bit data   PB   STB
1	0	SB   9-bit data   STB
1	1	SB   8-bit data   PB   STB

Legends: SB: Start Bit, STB: STOP bit, PB: Parity bit

**Note:** *In case of wakeup 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.

Ex: data=00110101; 4 bits set => parity bit will be 0 if even parity is selected (PS bit in USART\_CR1 = 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 will be 1 if odd parity is selected (PS bit in USART\_CR1 = 1).

**Transmission:** If the PCEN bit is set in USART\_CR1 then the MSB bit of the data written in the data register is not transmitted but is changed by the parity bit to give an even number of ‘1’s if even parity is selected (PS=0) or an odd number of ‘1’s if odd parity is selected (PS=1).

**Reception:** If the parity check fails, the PE flag is set in the USART\_SR register and an interrupt is generated if the PIEN bit is set in the USART\_CR1 register.

### 29.3.7 Multi-processor communication

It is possible to perform multiprocessor communication with the USART (several USARTs connected in a network). For example, one of the USARTs can be the master, its Tx output is connected to the Rx input of the other USART. The others are slaves, their respective Tx outputs are logically ANDed together and connected to the Rx input of the master.

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

The non addressed devices may be placed in mute mode by means of the muting function.  
In mute mode:

- None of the reception status bits can be set.
- All the receive interrupts are inhibited.
- The RWU bit in USART\_CR2 register is set to 1. RWU can be controlled automatically by hardware or written by the software under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART\_CR1 register:

- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

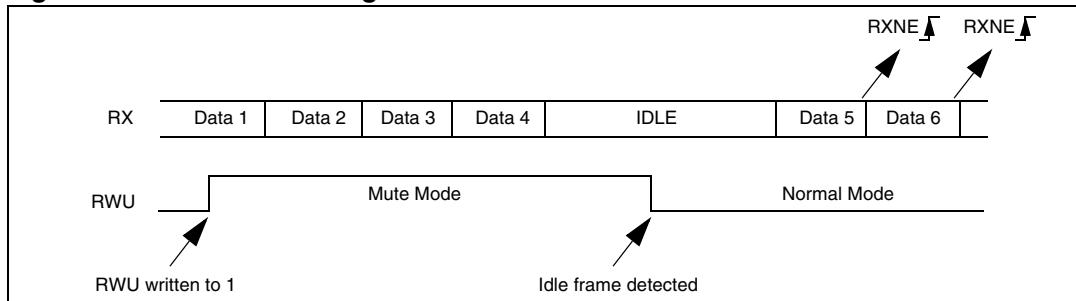
#### Idle line detection (WAKE=0)

The USART enters mute mode when the RWU bit is written to 1.

It wakes up when an Idle frame is detected. Then the RWU bit is cleared by hardware but the IDLE bit is not set in the USART\_SR register. RWU can also be written to 0 by software.

An example of mute mode behavior using idle line detection is given in [Figure 159](#).

**Figure 159. Mute mode using Idle line detection**



#### Address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a '1' else they are considered as data. In an address byte, the address of the targeted receiver is put on the 4 LSB. This 4-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART\_CR4 register.

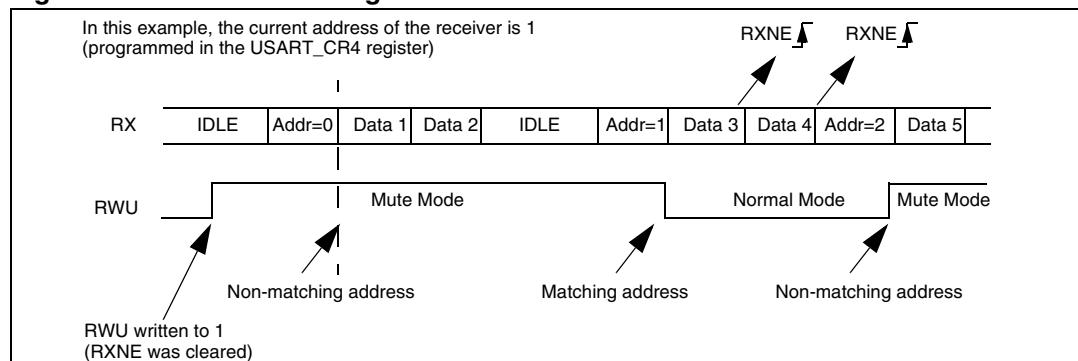
The USART enters mute mode when an address character is received which does not match its programmed address. The RXNE flag is not set for this address byte and no interrupt nor DMA request is issued as the USART would have entered mute mode.

It exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE bit is set for the address character since the RWU bit has been cleared.

The RWU bit can be written to 0 or 1 when the receiver buffer contains no data (RXNE=0 in the USART\_SR register). Otherwise the write attempt is ignored.

An example of mute mode behavior using address mark detection is given in [Figure 160](#).

**Figure 160. Mute mode using address mark detection**



**Note:** If parity control is enabled, the parity bit remains in the MSB and the address bit is put in the "MSB - 1" bit.

For example, with 7-bit data, address mode and parity control:

SB I 7-bit data I ADD I PB I STB

where:

SB = Start Bit

STB = STOP Bit

ADD = Address bit

PB = Parity Bit

### 29.3.8 USART synchronous communication

The USART transmitter allows the user to control bidirectional synchronous serial communications in master mode.

**Note:** This feature is only available for devices with USART\_CK pin. Check the device pinout for availability.

The USART\_CK pin is the output of the USART transmitter clock. No clock pulses are sent to the USART\_CK pin during start bit and STOP bit. Depending on the state of the LBCL bit in the USART\_CR3 register clock pulses will or will not be generated during the last valid data bit (address mark). The CPOL bit in the USART\_CR3 register allows the user to select the clock polarity, and the CPHA bit in the USART\_CR3 register allows the user to select the phase of the external clock (see [Figure 161](#), [Figure 162](#) & [Figure 163](#)). USART\_CKDuring the idle state, preamble phase and break transmission, the external USART\_CK clock is not activated. In synchronous mode, the USART transmitter works exactly like in asynchronous mode. But as USART\_CK is synchronized with USART\_TX (depending on CPOL and CPHA), the data on USART\_TX is synchronous. In this mode, the USART receiver works slightly differently compared to the asynchronous mode: if REN=1, the data is still sampled using the internal oversampling clock and the baud rate clock is output on the USART\_CK pin (rising or falling edge is aligned with the data sampling event depending on CPOL and

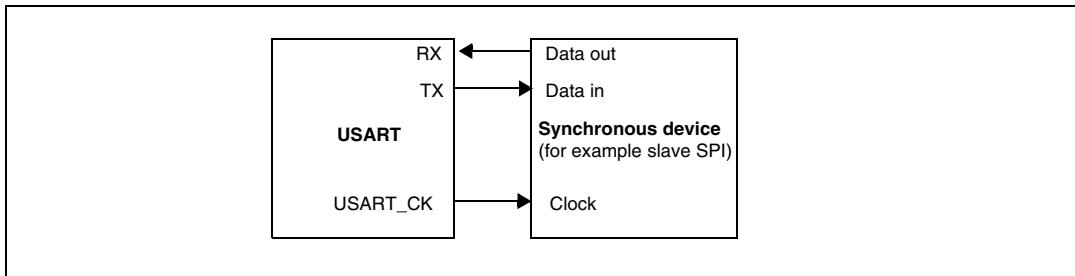
CPHA). But contrary to asynchronous mode, the data is evaluated using one sample and not the majority of 3 samples, meaning that the NF bit will never be set.

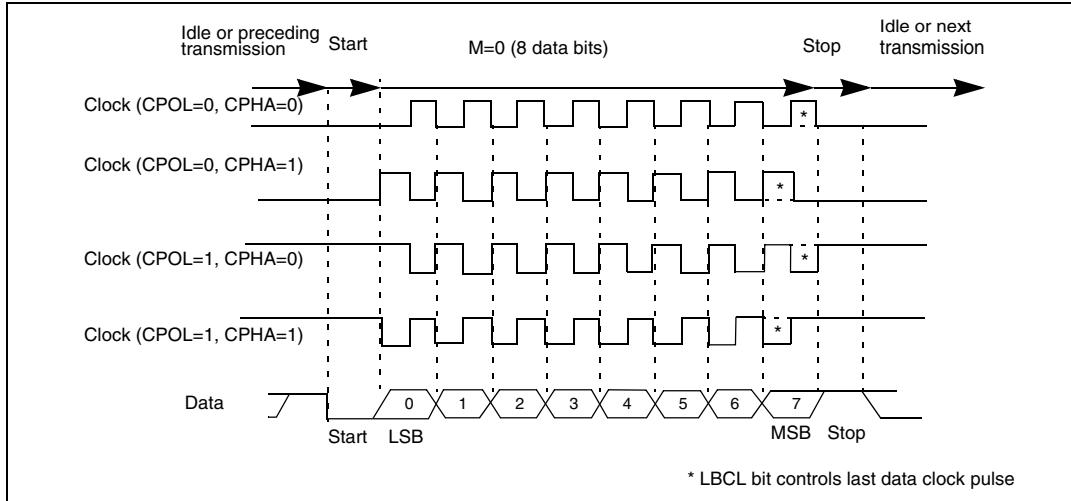
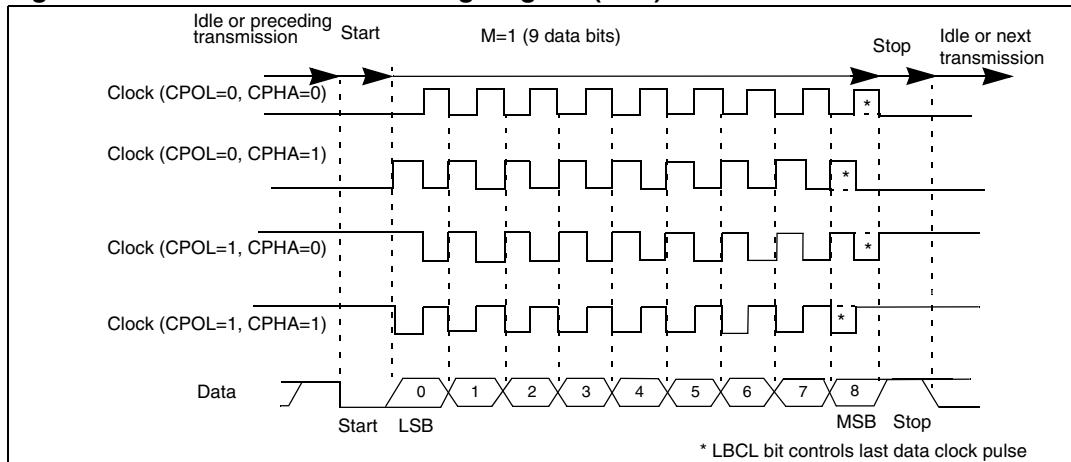
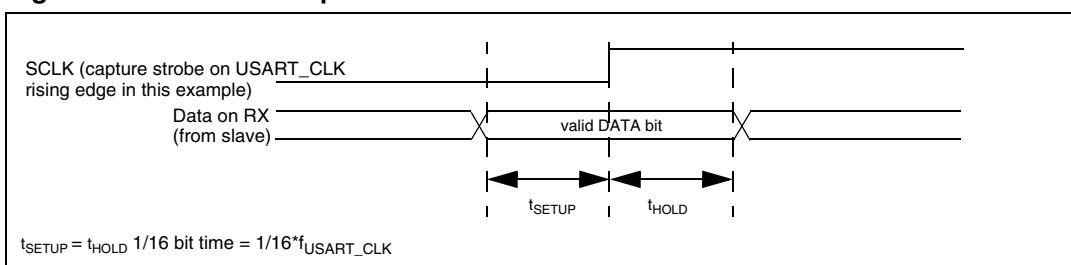
Setup and hold times must be respected (depending on the baud rate: 1/16 bit time for an integer baud rate).

- Note:**
- 1 *The USART\_CK pin works in conjunction with the TX pin. Consequently, the clock is provided only if the transmitter is enabled (TEN=1) and a data is being transmitted (the data register USART\_DR has been written). This means that it is not possible to receive a synchronous data without transmitting data.*
  - 2 *The LBCL, CPOL and CPHA bits in USART\_CR3 have to be selected when both the transmitter and the receiver are disabled (TEN=REN=0) to ensure that the clock pulses function correctly. These bits should not be changed while the transmitter or the receiver is enabled.*
  - 3 *It is recommended to set TEN and REN are set in the same instruction in order to minimize the setup and the hold time of the receiver.*
  - 4 *The USART supports master mode only: it cannot receive or send data related to an input clock (USART\_CK is always an output).*
  - 5 *The data given in this section apply only when the USART\_DIV[3:0] bits in the USART\_BRR2 register are kept at 0. Else the setup and hold times are not 1/16 of a bit time but 4/16 of a bit time.*

This option allows to serially control peripherals which consist of shift registers, without losing any functions of the asynchronous communication which can still talk to other asynchronous transmitters and receivers.

**Figure 161. USART example of synchronous transmission**



**Figure 162. USART data clock timing diagram (M=0)****Figure 163. USART data clock timing diagram (M=1)****Figure 164. RX data setup/hold time****Note:**

The function of USART\_CK is different in Smartcard mode. Refer to [Section 29.3.10: Smartcard](#) for more details.

### 29.3.9 Single wire half duplex communication

The USART can be configured to follow a single wire half duplex protocol. Single-wire half-duplex mode is selected by setting the HDSEL bit in the USART\_CR5 register. In this mode, the following bits must be kept cleared:

- CLKEN bit in the USART\_CR3 register
- SCEN and IREN bits in the USART\_CR5 register

As soon as HDSEL is set:

- USART\_RX is no longer used
- USART\_TX is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. This means that the I/O must be configured so that USART\_TX is configured as floating input (or output high open-drain) when not driven by the USART.

Apart from this, the communications are similar to what is done in normal USART mode. The conflicts on the line must be managed by the software (by the use of a centralized arbiter, for instance). In particular, the transmission is never blocked by hardware and continue to occur as soon as a data is written in the data register while the TEN bit is set.

### 29.3.10 Smartcard

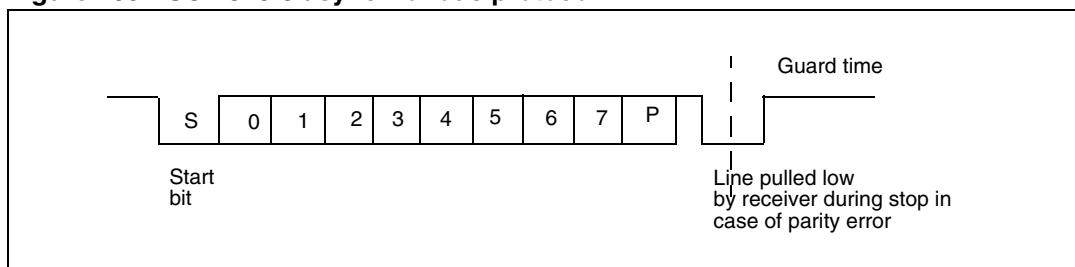
Smartcard mode is selected by setting the SCEN bit in the USART\_CR5 register. In smartcard mode, the following bits must be kept cleared:

- HDSEL and IREN bits in the USART\_CR5 register.

Moreover, the CKEN bit may be set in order to provide a clock to the smartcard.

The Smartcard interface is designed to support asynchronous protocol Smartcards as defined in the ISO7816-3 standard. The USART should be configured as eight bits plus parity and 1.5 STOP bits. With Smartcard mode enabled (which can be done by setting the SCEN bit in the USART\_CR5) the USART can communicate with an asynchronous Smartcard.

**Figure 165. ISO 7816-3 asynchronous protocol**



When connected to a smartcard, the USART\_TX output drives a bidirectional line that is also driven by the smartcard.

Smartcard is a single wire half duplex communication protocol.

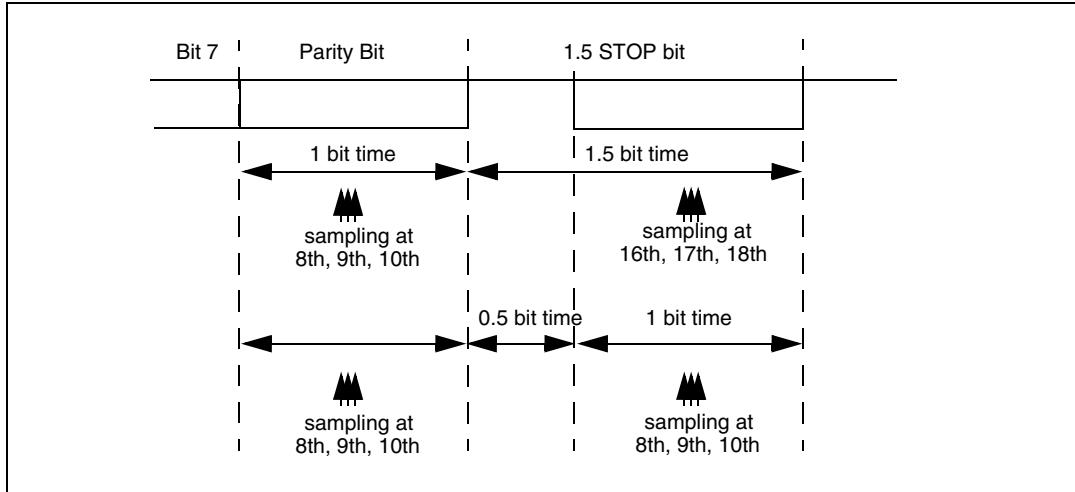
- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register will start shifting on the next baud clock edge. In Smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- If a parity error is detected during reception of a frame programmed with a 1.5 STOP bit period, the transmit line is pulled low for a baud clock period after 1/2 baud clock

period. This is to indicate to the Smartcard that the data transmitted to the USART has not been correctly received. This NACK signal (pulling transmit line low for 1 baud clock) will cause a framing error on the transmitter side (configured with 1.5 STOP bits). The application can handle re-sending of data according to the protocol. A parity error is 'NACK'ed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted.

- The TEN bit must be set to enable:
  - Data transmission
  - Transmission of acknowledgements in case of parity error.  
Software must manage the timing of data transmission to avoid conflicts on the data line when it writes new data in the data register.
- The REN bit must be set to enable:
  - Data reception (sent by the Smartcard as well as by the USART),
  - Detection of acknowledgements in case of parity error.
- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In Smartcard mode an empty transmit shift register triggers the guard time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the guard time counter reaches the programmed value TC is asserted high.
- The de-assertion of TC flag is unaffected by Smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK will not be detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver will not detect the NACK as a start bit.
- The output enable signal for the Smartcard I/O enables driving into a bidirectional line which is also driven by the Smartcard. This signal is active while transmitting the start and data bits and transmitting NACK. While transmitting the STOP bits this signal is disabled, so that the USART weakly drives a '1' on the bidirectional line.

- Note:
- 1 *A break character is not significant in Smartcard mode. A 00h data with a framing error will be treated as data and not as a break.*
  - 2 *No IDLE frame is transmitted when toggling the TEN bit. The IDLE frame (as defined for the other configurations) is not defined by the ISO protocol.*

[Figure 166](#) details how the NACK signal is sampled by the USART. In this example the USART is transmitting a data and is configured with 1.5 STOP bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

**Figure 166. Parity error detection using 1.5 STOP bits**

The USART can provide a clock to the smartcard through the USART\_CK output. In smartcard mode, USART\_CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the prescaler register USART\_PSCR. USART\_CK frequency can be programmed from  $f_{SYSCLK}/2$  to  $f_{SYSCLK}/62$ , where  $f_{SYSCLK}$  is the peripheral input clock.

### 29.3.11 IrDA SIR ENDEC block

IrDA mode is selected by setting the IREN bit in the USART\_CR5 register. The STOP bits in the USART\_CR3 register must be configured to “1 STOP bit”. In IrDA mode, the following bits must be kept cleared:

- STOP and CKEN bits in the USART\_CR3 register,
- SCEN and HDSEL bits in the USART\_CR5 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see [Figure 167](#)).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from the USART. The output pulse stream is transmitted to an external output driver and infrared LED. The USART supports only bit rates up to 115.2 kbps for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to USART. The decoder input is normally HIGH (marking state) in the idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (i.e. the USART is sending data to the IrDA encoder), any data on the IrDA receive line will be ignored by the IrDA decoder and if the Receiver is busy (USART is receiving decoded data from the USART), data on the TX from the USART to IrDA will not be encoded by IrDA.

While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.

- A '0' is transmitted as a high pulse and a '1' is transmitted as a '0'. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see [Figure 168](#)).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41  $\mu$ s. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in USART\_PSCR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than 2 periods will be accepted as a pulse. The IrDA encoder/decoder doesn't work when PSC=0.
- The receiver can communicate with a low power transmitter.
- In IrDA mode, the STOP bits in the USART\_CR2 register must be configured to "1 STOP bit".

### **IrDA low power mode**

The IrDA can be used either in normal mode or in Low Power mode. The Low Power mode is selected by setting the IRLP bit in USART\_CR5 register.

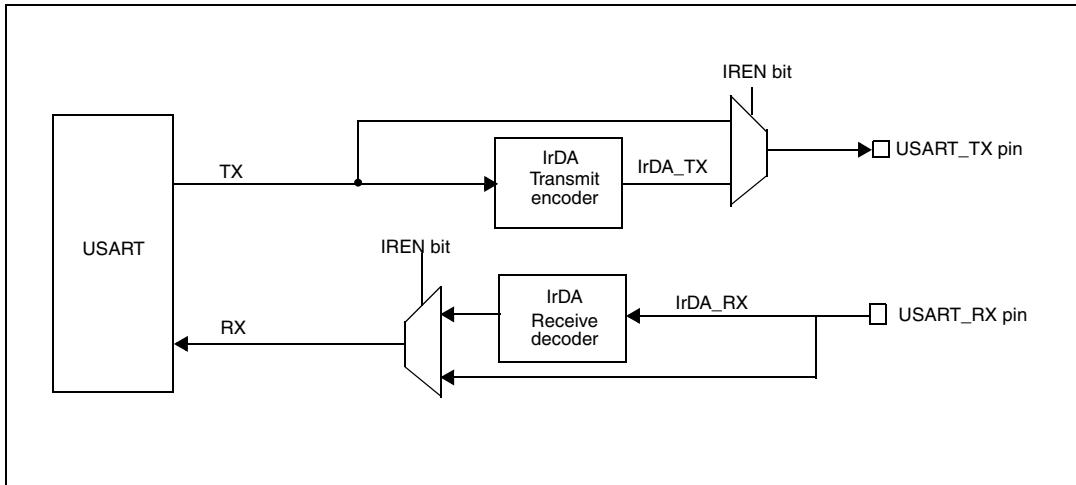
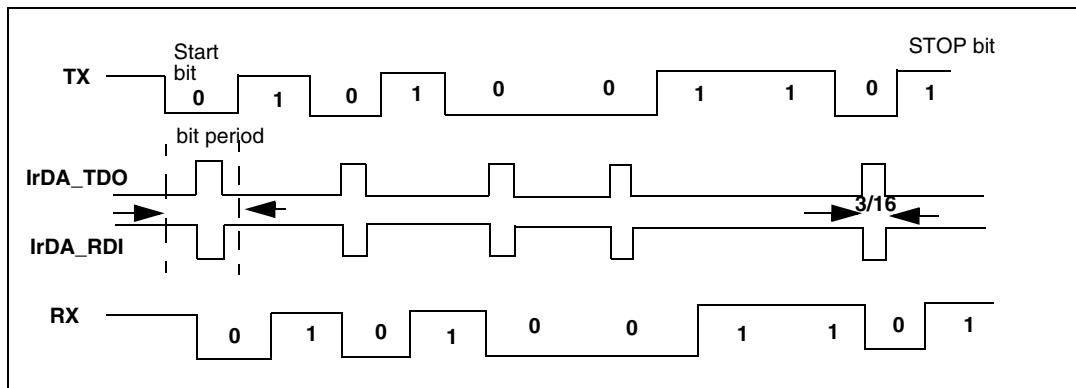
#### ***Transmitter:***

In low power mode the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low power baud rate which can be a minimum of 1.42 MHz. Generally this value is 1.8432 MHz (1.42 MHz < PSC < 2.12 MHz). A low power mode programmable divisor divides the system clock to achieve this value.

#### ***Receiver:***

Receiving in low power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low power Baud clock (PSC value in USART\_PSCR).

- Note:*
- 1 *A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.*
  - 2 *The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).*

**Figure 167. IrDA SIR ENDEC- block diagram****Figure 168. IrDA data modulation (3/16) - normal mode**

### 29.3.12 Continuous communication using DMA

The USART is capable to continue communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

*Note:*

You should refer to product specs for availability of the DMA controller. If DMA is not available in the product, you should use the USART as explained in [Section 29.3.2](#) or [29.3.3](#). In the USART\_SR register, you can clear the TXE/ RXNE flags to achieve continuous communication.

#### Transmission using DMA

DMA mode can be enabled for transmission by setting DMAT bit in the USART\_CR5 register. Data is loaded from a SRAM area configured using the DMA peripheral (refer to the DMA specification) to the USART\_DR register whenever the TXE bit is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART\_DR register address in the DMA control register to configure it as the destination of the transfer. The data will be moved to this address from memory after each TXE event.

2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data will be loaded into the USART\_DR register from this memory area after each TXE event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC bit in the SR register by writing 0 to it.
7. Activate the channel in the DMA register.

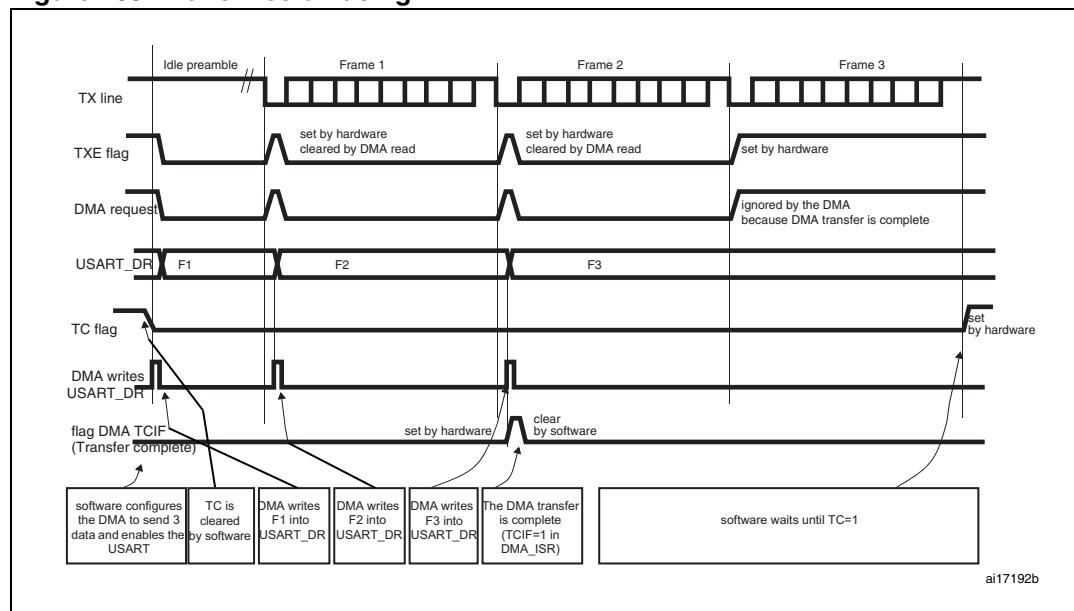
When the DMA has written all the data to be transmitted (flag TCIF is set in the DMA\_ISR register), the TC flag can be monitored to ensure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or entering Halt or Active-halt modes. The software must first wait until TxE is set to 1 and then wait until TC is set to 1.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

*Note:*

*If DMA is used for transmission, do not enable the TIEN bit.*

**Figure 169. Transmission using DMA**



### Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART\_CR5 register. Data is loaded from the USART\_DR register to a SRAM area configured using the DMA peripheral (refer to the DMA specification) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

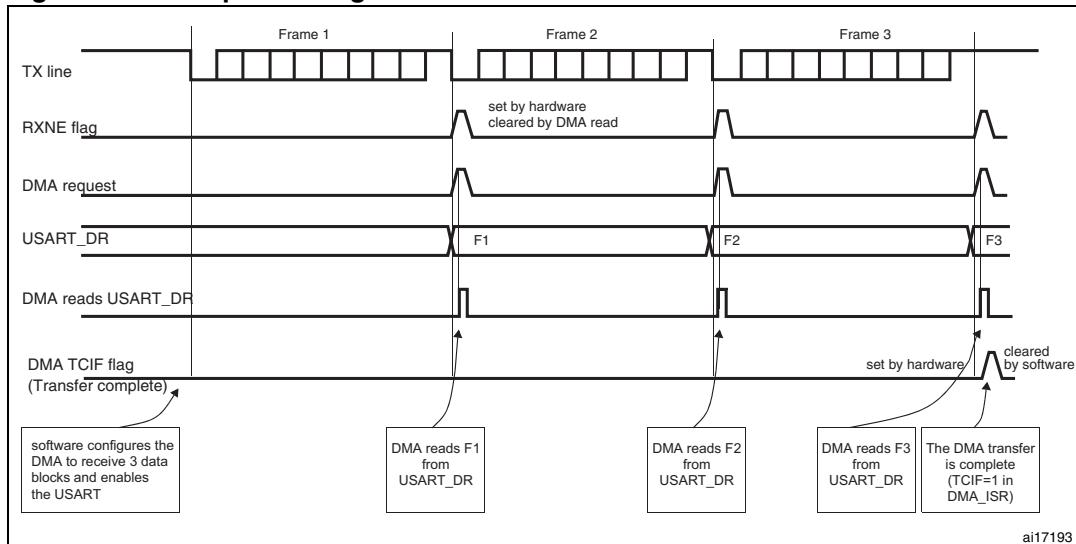
1. Write the USART\_DR register address in the DMA control register to configure it as the source of the transfer. The data will be moved from this address to the memory after each RXNE event.

2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data will be loaded from USART\_DR to this memory area after each RXNE event.
3. Configure the total number of bytes to be transferred in the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector. The DMAR bit should be cleared by software in the USART\_CR5 register during the interrupt subroutine.

*Note:* If DMA is used for reception, do not enable the RIEN bit.

**Figure 170. Reception using DMA**



ai17193

### Error flagging and interrupt generation in multibuffer communication

In case of multibuffer communication if any error occurs during the transaction the error flag will be asserted after the current byte. An interrupt will be generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE in case of single byte reception, there will be separate error flag interrupt enable bit (EIE bit in the USART\_CR5 register), which if set will issue an interrupt after the current byte with either of these errors.

## 29.4 USART low power modes

**Table 102. USART interface behavior in low power modes**

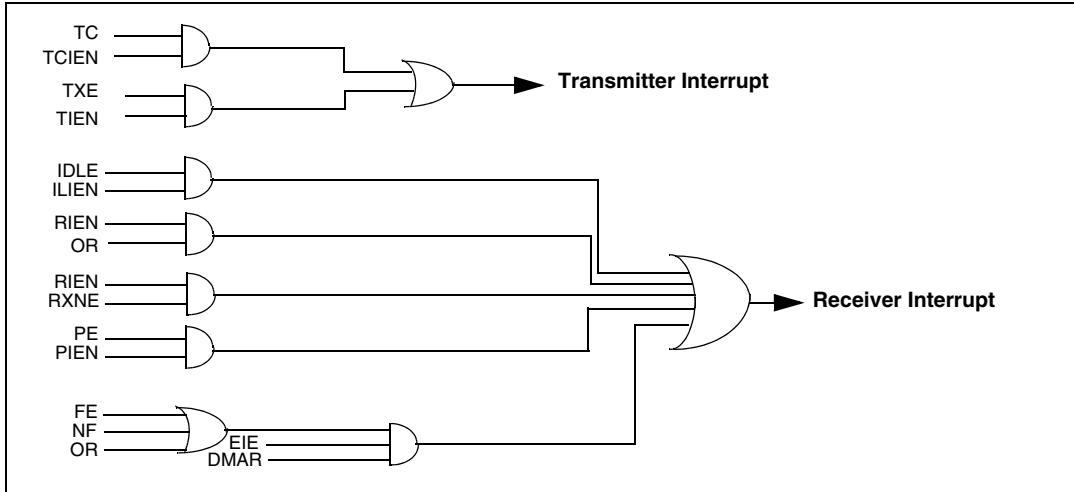
Mode	Description
Wait /Low power wait	No effect on USART. USART interrupts cause the device to exit from Wait or Low power wait mode.
Halt/Active-halt	USART registers are frozen. In Halt mode, the USART stops transmitting/receiving until Halt or Active-halt mode is exited.

## 29.5 USART interrupts

**Table 103. USART interrupt requests**

Interrupt event	Event flag	Enable control bit	Exit from Wait/Low power wait	Exit from Halt/Active-halt
Transmit data register empty	TXE	TIEN	Yes	No
Transmission complete	TC	TCIEN	Yes	No
Received data ready to be read	RXNE	RIEN	Yes	No
Overrun error detected	OR		Yes	No
Idle line detected	IDLE	ILIEN	Yes	No
Parity error	PE	PIEN	Yes	No
Noise flag, overrun error and framing error in multibuffer communication	NF or OR or FE	EIE	Yes	No

- Note:
- 1 *The USART interrupt events are connected to two interrupt vectors (see [Figure 171](#)).*
    - a) Transmission Complete or Transmit Data Register empty interrupt.
    - b) Idle line detection, Overrun error, Receive data register full, Parity error interrupt, and Noise flag (only in multibuffer communication) and Framing error (only in multibuffer communication).
  - 2 *These events generate an interrupt if the corresponding enable control bit is set and the interrupt mask in the CCR register is reset (RIM instruction).*

**Figure 171. USART interrupt mapping diagram**

## 29.6 USART registers

### 29.6.1 Status register (USART\_SR)

Address offset: 0x00

Reset value: 0xC0

7	6	5	4	3	2	1	0
TXE	TC	RXNE	IDLE	OR	NF	FE	PE

#### Bit 7 TXE: 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 TIEN bit =1 in the USART\_CR2 register. It is cleared by a write to the USART\_DR register.

0: Data is not transferred to the shift register

1: Data is transferred to the shift register

#### Bit 6 TC: Transmission complete

TC bit is set by hardware if the transmission of a frame containing data is complete and TXE bit is set. An interrupt is generated if TCIEN=1 in the USART\_CR2 register.

TC bit is cleared either by a software sequence (a read to the USART\_SR register followed by a write to the USART\_DR register), or by programming the bit to '0'. This clear sequence is recommended only for multibuffer communications.

0: Transmission is not complete

1: Transmission is complete

#### Bit 5 RXNE: Read data register not empty

This bit is set by hardware when the content of the RDR shift register has been transferred to the USART\_DR register. An interrupt is generated if RIEN=1 in the USART\_CR2 register. It is cleared by a read to the USART\_DR register.

0: Data is not received

1: Received data is ready to be read.

**Bit 4 IDLE:** IDLE line detected <sup>(1)</sup>

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if the ILIEN=1 in the USART\_CR2 register. It is cleared by a software sequence (a read to the USART\_SR register followed by a read to the USART\_DR register).

0: No Idle Line is detected

1: Idle Line is detected

**Bit 3 OR:** Overrun error<sup>(2)</sup>

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 RXNE=1. An interrupt is generated if RIEN=1 in the USART\_CR2 register. It is cleared by a software sequence (a read to the USART\_SR register followed by a read to the USART\_DR register).

0: No Overrun error

1: Overrun error is detected

**Bit 2 NF:** Noise flag <sup>(3)</sup>

This bit is set by hardware when noise is detected on a received frame. It is cleared by a software sequence (a read to the USART\_SR register followed by a read to the USART\_DR register).

0: No noise is detected

1: Noise is detected

**Bit 1 FE:** Framing error <sup>(4)</sup>

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 (a read to the USART\_SR register followed by a read to the USART\_DR register).

0: No framing error is detected

1: Framing error or break character is detected

**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 a read to the USART\_DR data register). You have to wait for the RXNE flag to be set before clearing it. An interrupt is generated if PIEN=1 in the USART\_CR1 register.

0: No parity error

1: Parity error

1. The IDLE bit is not set again until the RXNE bit has been set itself (i.e. a new idle line occurs)
2. When this bit is set, the RDR register content is not lost but, the shift register is overwritten.
3. This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt.
4. This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. If the word currently being transferred causes both a frame error and an overrun error, it is transferred and only the OR bit is set.

## 29.6.2 Data register (USART\_DR)

Address offset: 0x01

Reset value: 0xXX

7	6	5	4	3	2	1	0
DR[7:0]							
rw							

Bits 7:0 **DR[7:0]**: Data value

Contains the received or transmitted data character, depending on whether it is read from or written to.  
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.  
The RDR register provides the parallel interface between the input shift register and the internal bus.

## 29.6.3 Baud rate register 1 (USART\_BRR1)

The baud rate registers are common to both the transmitter and the receiver. The baud rate is programmed using two registers BRR1 and BRR2. Writing of BRR2 (if required) should precede BRR1, since a write to BRR1 will update the baud counters.

See [Figure 158: How to code USART\\_DIV in the BRR registers on page 515](#) and [Table 96: Baud rate programming and error calculation on page 515](#).

*Note:* The baud counters stop counting if the TEN or REN bits are disabled respectively.

Address offset: 0x02

Reset value: 0x00

7	6	5	4	3	2	1	0
USART_DIV[11:4]							
rw	rw	rw	rw	-	rw	rw	rw

Bits 7:0 **USART\_DIV[11:4]**: USART\_DIV bits <sup>(1)</sup>

These 8 bits define the 2nd and 3rd nibbles of the 16-bit USART divider (USART\_DIV).

1. BRR1 = 0x00 means USART clock is disabled.

## 29.6.4 Baud rate register 2 (USART\_BRR2)

Address offset: 0x03

Reset value: 0x00

7	6	5	4	3	2	1	0
USART_DIV[15:12]				USART_DIV[3:0]			
rw				rw			

Bits 7:4 **USART\_DIV[15:12]**: MSB of USART\_DIV

These 4 bits define the MSB of the USART Divider (USART\_DIV)

Bits 3:0 **USART\_DIV[3:0]**: LSB of USART\_DIV

These 4 bits define the LSB of the USART Divider (USART\_DIV)

### 29.6.5 Control register 1 (USART\_CR1)

Address offset: 0x04

Reset value: 0x00

7	6	5	4	3	2	1	0
R8	T8	USARTD	M	WAKE	PCEN	PS	PIEN

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 **USARTD**: USART disable (for low power consumption)

When this bit is set the USART prescaler and outputs are stopped at the end of the current byte transfer in order to reduce power consumption. This bit is set and cleared by software.

0: USART enabled

1: USART 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, 'n' STOP bit (n depending on STOP[1:0] bits in the USART\_CR3 register)

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**: Wakeup method

This bit determines the USART wakeup method, it is set or cleared by software.

0: Idle line

1: Address mark

Bit 2 **PCEN**: 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, PCEN 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 (PCEN 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 **PIEN**: Parity interrupt enable

This bit is set and cleared by software.

0: Parity interrupt disabled

1: Parity interrupt is generated whenever PE=1 in the USART\_SR register

## 29.6.6 Control register 2 (USART\_CR2)

Address offset: 0x05

Reset value: 0x00

7	6	5	4	3	2	1	0
TIEN	TCIEN	RIEN	ILIEN	TEN	REN	RWU	SBK

Bit 7 **TIEN**: Transmitter interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever TXE=1 in the USART\_SR register

Bit 6 **TCIEN**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever TC=1 in the USART\_SR register

Bit 5 **RIEN**: Receiver interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever OR=1 or RXNE=1 in the USART\_SR register

Bit 4 **ILIEN**: IDLE Line interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An USART interrupt is generated whenever IDLE=1 in the USART\_SR register

Bit 3 **TEN**: Transmitter enable <sup>(1) (2)</sup>

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Bit 2 **REN**: 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 wakeup<sup>(3) (4)</sup>

This bit determines if the USART is in mute mode or not. It is set and cleared by software and can be cleared by hardware when a wakeup sequence is recognized.

0: Receiver in active mode

1: Receiver in mute mode

Bit 0 **SBK**: Send break

This bit set is used to send break characters. It can be set and cleared by software. It should be set by software, and will be reset by hardware during the STOP bit of break.

0: No break character is transmitted

1: Break character will be transmitted

1. During transmission, a “0” pulse on the TEN bit (“0” followed by “1”) sends a preamble (idle line) after the current word.
2. When TEN is set there is a 1 bit-time delay before the transmission starts.
3. Before selecting Mute mode (by setting the RWU bit) the USART must first receive a data byte, otherwise it cannot function in Mute mode with wakeup by Idle line detection.
4. In address mark detection wakeup configuration (WAKE bit=1) the RWU bit cannot be modified by software while the RXNE bit is set.

### 29.6.7 Control register 3 (USART\_CR3)

Address offset: 0x06

Reset value: 0x00

	7	6	5	4	3	2	1	0
Reserved				STOP[1:0]	CLKEN	CPOL	CPHA	LBCL
				rw	rw	rw	rw	rw

Bit 7 Reserved

Bits 5:4 **STOP**: STOP bits

These bits are used for programming the STOP bits.

00: 1 STOP bit

01: Reserved

10: 2 STOP bits

11: 1.5 STOP bits

Bit 3 **CLKEN**: Clock enable

This bit allows the user to enable the USART\_CK pin.

0: USART\_CK pin disabled

1: USART\_CK pin enabled

Bit 2 **CPOL**: Clock polarity<sup>(1)</sup>

This bit allows the user to select the polarity of the clock output on the USART\_CK pin. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: USART\_CK to 0 when idle

1: USART\_CK to 1 when idle.

Bit 1 **CPHA**: Clock phase <sup>(1)</sup>

This bit allows the user to select the phase of the clock output on the USART\_CK pin. It works in conjunction with the CPOL bit to produce the desired clock/data relationship

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

Bit 0 **LBCL**: Last bit clock pulse<sup>(1)(2)</sup>

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 USART\_CK pin.

0: The clock pulse of the last data bit is not output to the USART\_CK pin.

1: The clock pulse of the last data bit is output to the USART\_CK pin.

1. These 3 bits (CPOL, CPHA, LBCL) should not be written while the transmitter is enabled.
2. 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 USART\_CR1 register.

### 29.6.8 Control register 4 (USART\_CR4)

Address offset: 0x07

Reset value: 0x00

7	6	5	4	3	2	1	0
Reserved				ADD[3:0]			
				rw			

Bit 7:4 Reserved

Bits 3:0 **ADD[3:0]**: Address of the USART node

This bit-field gives the address of the USART node.

This is used in multiprocessor communication during mute mode, for wakeup with address mark detection.

### 29.6.9 Control register 5 (USART\_CR5)

Address offset: 0x08

Reset value: 0x00

7	6	5	4	3	2	1	0
DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	r	r	rw	rw	rw	rw

Bit 7 **DMAT**: DMA Enable transmitter.

This bit is set/reset by software

- 1: DMA mode is enabled for transmission.
- 0: DMA mode is disabled for transmission.

Bit 6 **DMAR**: DMA Enable receiver.

This bit is set/reset by software

- 1: DMA mode is enabled for reception.
- 0: DMA mode is disabled for reception.

Bit 5 **SCEN**: Smartcard mode enable.

This bit is used for enabling Smartcard mode.

- 0: Smartcard Mode disabled
- 1: Smartcard Mode enabled

Bit 4 **NACK**: Smartcard NACK enable.

- 0: NACK transmission in case of parity error is disabled
- 1: NACK transmission during parity error is enabled.

Bit 3 **HDSEL**: Half duplex selection.

Selection of Single-wire half duplex mode

- 0: Half duplex mode is not selected
- 1: Half duplex mode is selected

Bit 2 **IRLP**: IrDA Low power.

This bit is used for selected between normal and Low power IrDA mode

- 0: Normal mode
- 1: Low power mode

Bit 1 **IREN**: IrDA mode enable.

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

Bit 0 **EIE**: Error Interrupt enable.

This bit is required to enable interrupt generation in case of a framing error, overrun error or noise error (FE=1 or OR=1 or NF=1 in the USART\_SR register) in case of Multibuffer communication (DMAR=1 in the USART\_CR5 register).

0: Interrupt is inhibited

1: An interrupt is generated whenever DMAR=1 in the USART\_CR5 register and FE=1 or OR=1 or NF=1 in the USART\_SR register.

### 29.6.10 Guard time register (USART\_GTR)

Address offset: 0x09

Reset value: 0x00

7	6	5	4	3	2	1	0
GT[7:0]							
rw							

Bits 7:0 **GT**: Guard time value.

This register gives the Guard time value in terms of number of baud clocks.

This is used in Smartcard mode. The Transmission Complete flag is set after this guard time value.

### 29.6.11 Prescaler register (USART\_PSCR)

Address offset: 0x0A

Reset value: 0x00

**Note:** Care must be taken to program this register with correct value, when both Smartcard and IrDA interfaces are used in the application

7	6	5	4	3	2	1	0
PSC[7:0]							
rw							

Bits 7:0 **PSC[7:0]: Prescaler value.**

– In IrDA Low power mode:

PSC[7:0] = IrDA Low power baud rate <sup>(1)</sup>

Used for programming the prescaler for dividing the system clock to achieve the low power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

– In Smartcard mode:

PSC[4:0]: Prescaler value. <sup>(2) (3)</sup>

Used for programming the prescaler for dividing the system clock to provide the smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

1. This prescaler setting has no effect if IrDA mode is not enabled.
2. This prescaler setting has no effect if Smartcard mode is not enabled.
3. Bits [7:5] have no effect even if Smartcard mode is enabled.

### 29.6.12 USART register map and reset values

**Table 104. USART register map**

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	USART_SR Reset value	TXE 1	TC 1	RXNE 0	IDLE 0	OR 0	NF 0	FE 0	PE 0
0x01	USART_DR Reset value				DR[7:0] xxxxxxxx				
0x02	USART_BRR1 Reset value				USART_DIV[11:4] 00000000				
0x03	USART_BRR2 Reset value			USART_DIV[15:12] 0000		USART_DIV[3:0] 0000			

**Table 104. USART register map (continued)**

<b>Address offset</b>	<b>Register name</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>
0x04	USART_CR1 Reset value	R8 0	T8 0	USARTD 0	M 0	WAKE 0	PCEN 0	PS 0	PIEN 0
0x05	USART_CR2 Reset value	TIEN 0	TCIEN 0	RIEN 0	ILIEN 0	TEN 0	REN 0	RWU 0	SBK 0
0x06	USART_CR3 Reset value	- 0	- 0	STOP 00		CKEN 0	CPOL 0	CPHA 0	LBCL 0
0x07	USART_CR4 Reset value	- 0	- 0	- 0	- 0	ADD[3:0] 0000			
0x08	USART_CR5 Reset value	DMAT 0	DMAR 0	SCEN 0	NACK 0	HDSEL 0	IRLP 0	IREN 0	EIE 0
0x09	USART_GTR Reset value	GT[7:0] 00000000							
0x0A	USART_PSCR Reset value	PSC[7:0] 00000000							

## 30 Serial peripheral interface (SPI)

This section applies to low density STM8L05xx/STM8L15xx devices, medium density STM8L05xx/STM8L15xx devices, medium density STM8AL31xx/STM8AL3Lxx devices, medium+ density STM8L05xx/STM8L15xx devices and high density STM8L05xx/STM8L15xx/STM8L16xx devices, unless otherwise specified.

### 30.1 Introduction

The serial peripheral interface (SPI) allows half/ full duplex, synchronous, serial communication with external devices. The interface can be configured as the master and in this case it provides the communication clock (SCK) to the external slave device. The interface is also capable of operating in multi-master configuration.

It may be used for a variety of purposes, including simplex synchronous transfers on 2 lines with a possible bidirectional data line or reliable communication using CRC checking.

### 30.2 SPI main features

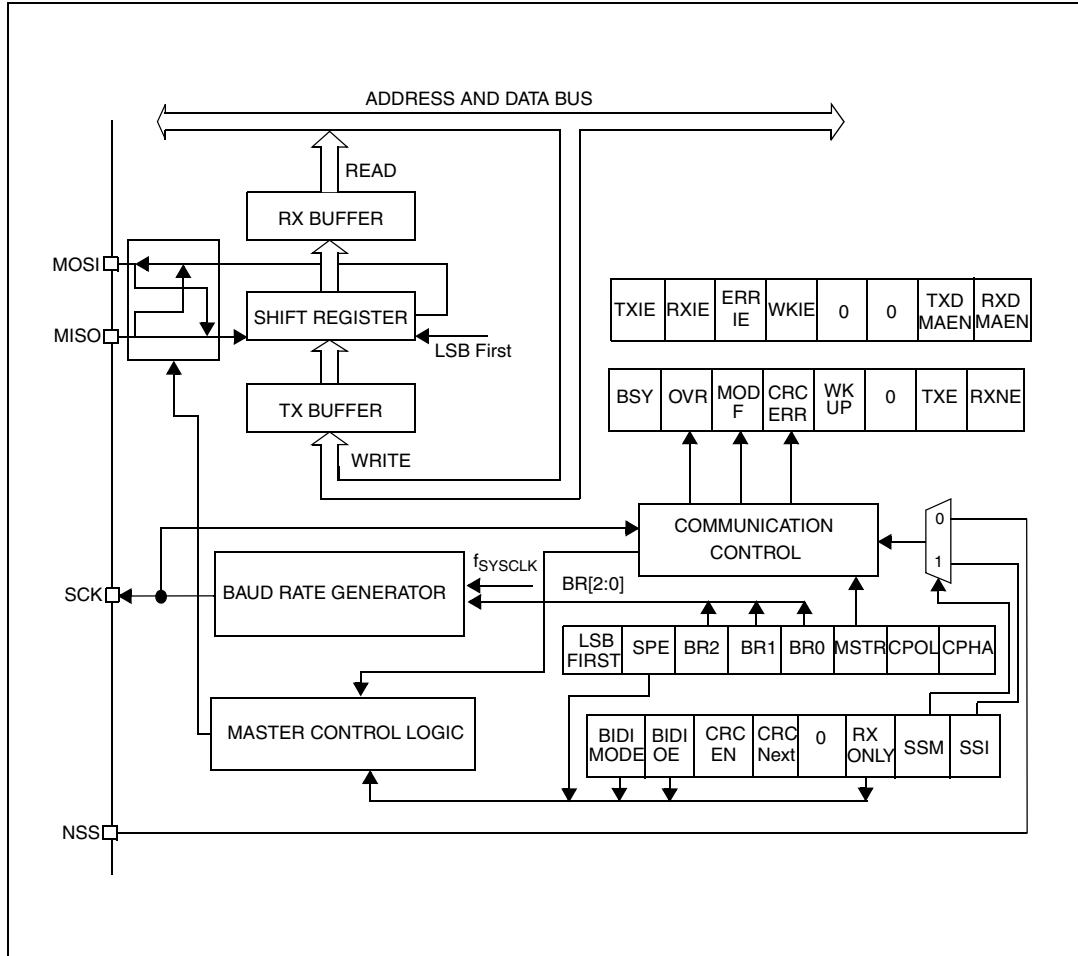
- Full duplex synchronous transfers (on 3 lines)
- Simplex synchronous transfers on 2 lines with or without a bidirectional data line
- Master or slave operation
- 8 Master mode frequencies ( $f_{SYSCLK}/2$  max.)
- Slave mode frequency ( $f_{SYSCLK}/2$  max.)
- Faster communication - Maximum SPI speed: 8 MHz
- NSS management by hardware or software for both master and slave
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- Master mode fault and overrun flags with interrupt capability
- 1-byte transmission and reception buffer with DMA capability: Tx and Rx requests
- Hardware CRC feature for reliable communication:
  - CRC value can be transmitted as last byte in Tx mode
  - CRC error checking for last received byte
- Wakeup capability:  
The MCU wakes up from Low power mode in full or half duplex transmit-only modes

## 30.3 SPI functional description

### 30.3.1 General description

The block diagram of the SPI is shown in [Figure 172](#).

**Figure 172. SPI block diagram**



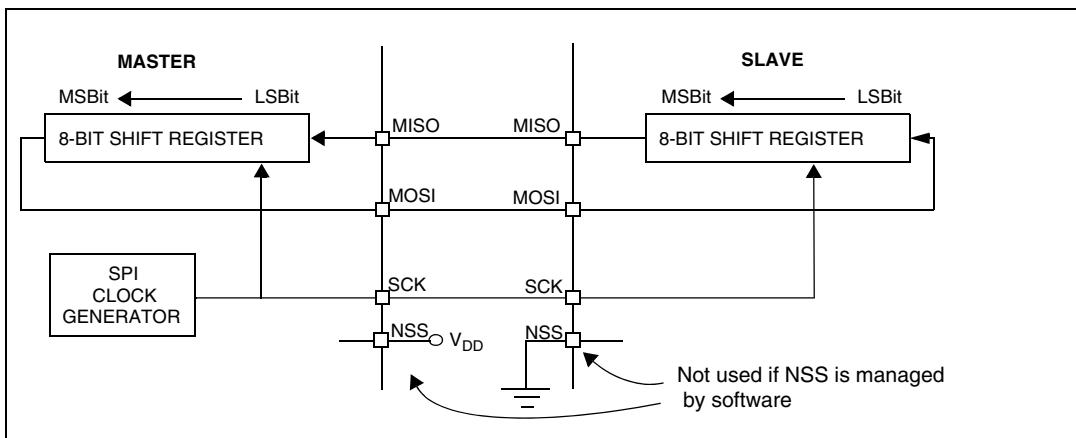
The SPI is connected to external devices through four pins:

- **MISO**: Master In / Slave Out data. This pin can be used to transmit data in slave mode and receive data in master mode.
- **MOSI**: Master Out / Slave In data . This pin can be used to transmit data in master mode and receive data in slave mode.
- **SCK**: Serial Clock output for SPI masters and Serial Clock input for SPI slaves.
- **NSS**: Slave select. This is a optional pin to select a slave device. This pin acts as a 'chip select' to let the SPI master communicate with slaves individually and to avoid contention on the data lines. Slave NSS inputs can be driven by standard I/O ports on the master device. When configured in master mode (**MSTR** bit =1) and if NSS is pulled low, the SPI enters master mode fault state: the **MSTR** bit is automatically reset and the device is configured in slave mode (refer to [Section 30.3.10: Error flags on page 559](#)).

A basic example of interconnections between a single master and a single slave is illustrated in [Figure 173](#).

**Note:** *The I/O slope on which the SPI outputs are connected is controlled by hardware as fast slope output whatever the I/O port configuration.*

**Figure 173. Single master/ single slave application**



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 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).

### Slave select (NSS) pin management

A hardware or software slave select management configuration can be set using the Software slave select management (SSM) bit from the SPI\_CR2 register.

- **Software NSS management (SSM = 1):** with this configuration, slave select information is driven internally by the Internal slave select (SSI) bit value in the SPI\_CR2 register. The external NSS pin remains free for other application uses.
- **Hardware NSS management (SSM = 0):** For devices set as master, this configuration allows multimaster capability. For devices set as slave, the NSS pin works as a classical NSS input. The slave is selected when the NSS line is in low level and is not selected if the NSS line is in high level.

**Note:** *When the master is communicating with SPI slaves which need to be deselected between transmissions, the NSS pin must be configured as a GPIO.*

### Clock phase and clock polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits. The CPOL (clock polarity) bit controls the steady state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, SCK pin has a low level idle state. If CPOL is set, SCK pin has a high level idle state.

**Note:** *Make sure the SPI pin is configured at the idle state level of the SPI in order to avoid generating an edge on the SPI clock pin when enabling or disabling the SPI cell.*

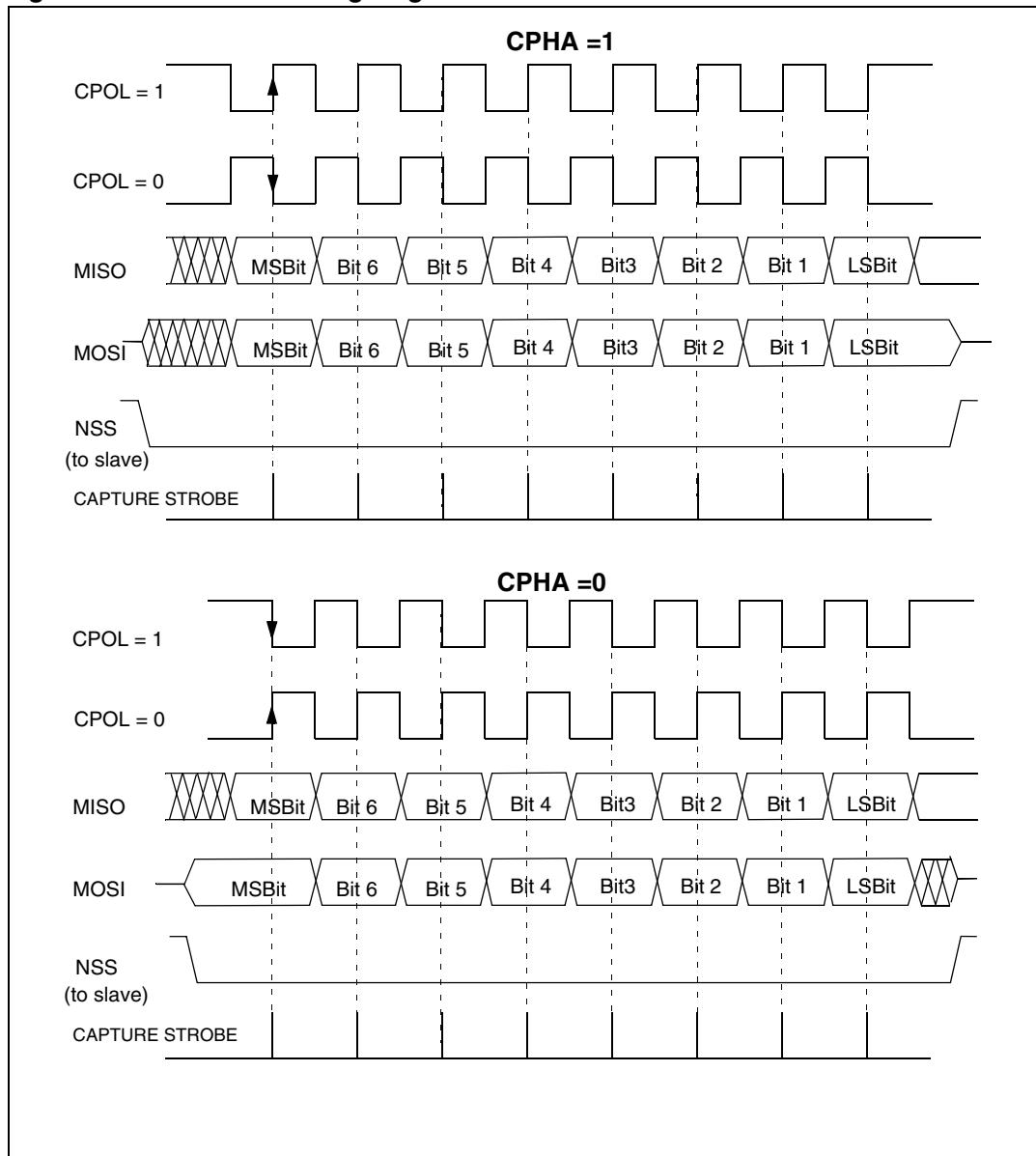
If CPHA (clock phase) bit is set, the second edge on the SCK pin (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set) is the MSBit capture strobe. Data is latched on the occurrence of the first clock transition. If CPHA bit is reset, the first edge on the SCK pin (falling edge if CPOL bit is set, rising edge if CPOL bit is reset) is the MSBit capture strobe. Data is latched on the occurrence of the second clock transition.

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

[Figure 174](#) shows an SPI transfer with the four combinations of the CPHA and CPOL bits. The diagram may be interpreted as a master or slave timing diagram where the SCK pin, the MISO pin, the MOSI pin are directly connected between the master and the slave device.

**Note:**

- 1 *Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.*
- 2 *Master and slave must be programmed with the same timing mode.*
- 3 *The idle state of SCK must correspond to the polarity selected in the SPI\_CR1 register (by pulling up SCK if CPOL=1 or pulling down SCK if CPOL=0).*

**Figure 174. Data clock timing diagram**

1. These timings are shown with the LSBFIRST bit reset in the SPI\_CR1 register.

### Frame format

Data can be shifted out either MSB-first or LSB-first depending on the value of the LSBFIRST bit in the SPI\_CR1 Register.

### 30.3.2 Configuring the SPI in slave mode

In slave configuration, the serial clock is received on the SCK pin from the master device. The value set in the BR[2:0] bits in the SPI\_CR1 register, does not affect the data transfer rate.

Follow the procedure below to configure the SPI in slave mode:

1. Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock (see [Figure 174](#)). For correct data transfer, the CPOL and CPHA bits must be configured in the same way in the slave device and the master device.
2. The frame format (MSB-first or LSB-first depending on the value of the LSBFIRST bit in the SPI\_CR1 register) must be the same as the master device.
3. In Hardware mode (refer to [Slave select \(NSS\) pin management on page 543](#)), the NSS pin must be connected to a low level signal during the complete data transmit sequence. In NSS Software mode, set the SSM bit and clear the SSI bit in the SPI\_CR2 register.
4. Clear the MSTR bit and set the SPE bit to assign the pins to alternate functions.

In this configuration the MOSI pin is a data input and the MISO pin is a data output.

Note:

*In applications with a parallel multi-slave structure, with separate NSS signals and the slave MISO outputs connected together, the corresponding GPIO registers must be configured correctly. The SPI\_MISO pin is controlled by the SPI peripheral only when the NSS signal is active and the device is selected as slave. When the NSS signal is released, the pin is driven by GPIO register settings only. To function correctly, the GPIO has to be configured in input pull-up mode with no interrupt. This configuration is done using the GPIO\_DDR, GPIO\_CR1 and GPIO\_CR2 registers - see [Section 10.8.1: Alternate function output](#).*

### 30.3.3 Configuring the SPI master mode

In a master configuration, the serial clock is generated on the SCK pin.

Follow the procedure below to configure the SPI in master mode:

1. Select the BR[2:0] bits to define the serial clock baud rate (see SPI\_CR1 register).
2. Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock (see [Figure 174](#)).
3. Configure the LSBFIRST bit in the SPI\_CR1 register to define the frame format.
4. In Hardware mode, connect the NSS pin to a high-level signal during the complete data transmit sequence. In software mode, set the SSM and SSI bits in the SPI\_CR2 register.
5. Set the MSTR and SPE bits (they remain set only if the NSS pin is connected to a high-level signal).

In this configuration the MOSI pin is a data output and to the MISO pin is a data input.

### 30.3.4 Configuring the SPI for simplex communications

The SPI is capable of operating in simplex mode in 2 configurations.

- 1 clock and 1 bidirectional data wire
- 1 clock and 1 data wire (Receive-only or Transmit-only)

#### 1 clock and 1 bidirectional data wire

This mode is enabled by setting the BDM bit in the SPI\_CR2 register. In this mode SCK is used for the clock, and MOSI in master or MISO in slave mode is used for data communication. The transfer direction (Input/output) is selected by the BDOE bit in the SPI\_CR2 register. When this bit is set to 1, the data line is output, otherwise it is input.

#### 1 clock and 1 unidirectional data wire (BDM = 0)

In this mode, the application can use the SPI either in transmit-only mode or in receive-only mode:

- Transmit-only mode is similar to full-duplex mode (BDM = 0, RXONLY = 0): the data is transmitted to the transmit pin (MOSI in master mode or MISO in slave mode) and the receive pin (MISO in master mode or MOSI in slave mode) can be used as general purpose I/O. In this case, the application just needs to ignore the Rx buffer (if the data register is read, it does not contain the received value).
- In receive-only mode, the application can disable the SPI output function by setting the RXONLY bit in the SPI\_CR2 register. In this case, it frees the transmit I/O pin (MOSI in master mode or MISO in slave mode) so it can be used for other purposes.

To start the communication in receive-only mode, configure and enable the SPI:

- In master mode, the communication starts immediately and stops when the SPE bit is reset and the current reception stops. There is no need to read the BSY flag in this mode. It is always set when an SPI communication is ongoing.
- In slave mode, the SPI continues to receive as long as the NSS is pulled down (or the SSI bit is reset in NSS software mode) and the SCK is running.

### 30.3.5 Data transmission and reception procedures

#### Rx and Tx buffer

In reception, data are received and then stored into an internal Rx buffer while in transmission, data are first stored into an internal Tx buffer before being transmitted.

A read access of the SPI\_DR register returns the Rx buffered value whereas a write access of the SPI\_DR stores the written data into the Tx buffer.

#### Start sequence in master mode

- In full-duplex (BDM = 0 and RXONLY = 0)
  - The sequence begins when data is written into the SPI\_DR register (Tx buffer).
  - The data is then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MOSI pin.
  - At the same time, the received data on MISO pin is shifted in serially to the 8-bit shift register and then parallel loaded into the SPI\_DR register (Rx Buffer).

- In unidirectional receive-only mode (BDM = 0 and RXONLY = 1)
  - The sequence begins as soon as the bit SPE = 1
  - Only the receiver is activated and the received data on MISO pin is shifted in serially to the 8-bit shift register and then parallel loaded into the SPI\_DR register (Rx Buffer).
- In bidirectional mode, when transmitting (BDM = 1 and BDOE = 1)
  - The sequence begins when a data is written into the SPI\_DR register (Tx buffer).
  - The data is then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MOSI pin.
  - No data is received.
- In bidirectional mode, when receiving (BDM = 1 and BDOE = 0)
  - The sequence begins as soon as SPE = 1 and BDOE = 0.
  - The received data on MOSI pin is shifted in serially to the 8-bit shift register and then parallel loaded into the SPI\_DR register (Rx Buffer).
  - The transmitter is not activated and no data is shifted out serially to the MOSI pin.

### Start sequence in slave mode

- In full-duplex (BDM=0 and RXONLY=0)
  - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MOSI pin. The remaining 7 bits are loaded into the shift register.
  - At the same time, the data is parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MISO pin. The software must have written the data to be sent before the SPI master device initiates the transfer.
- In unidirectional receive-only mode (BDM = 0 and RXONLY = 1)
  - The sequence begins when the slave device receives the clock signal and the first bit of the data on its MOSI pin. The remaining 7 bits are loaded into the shift register.
  - The transmitter is not activated and no data is shifted out serially to the MISO pin.
- In bidirectional mode, when transmitting (BDM = 1 and BDOE = 1)
  - The sequence begins when the slave device receives the clock signal and the first bit of the Tx buffer is transmitted to the MISO pin.
  - The data is then parallel loaded from the Tx buffer into the 8-bit shift register during the first bit transmission and then shifted out serially to the MISO pin. The software must have written the data to be sent before the SPI master device starts the transfer.
  - no data is received.
- In bidirectional mode, when receiving (BDM = 1 and BDOE = 0)
  - The sequence starts when the slave device receives the clock signal and the first bit of the data to its MISO pin.
  - The data received on MISO pin is shifted in serially to the 8-bit shift register and then parallel loaded into the SPI\_DR register (Rx Buffer).
  - The transmitter is not activated and no data is shifted out serially to the MISO pin.

## Handling data transmission and reception

The TXE flag (Tx buffer empty) is set when the data is transferred from the Tx buffer to the shift register. It indicates that the internal Tx buffer is ready to be loaded with the next data. An interrupt can be generated if TXIE bit in the SPI\_ICR register is set.

*Note:* *The software must ensure that TXE flag is set to 1 before attempting to write into the Tx buffer. Otherwise, it will overwrite the data which was previously written in the Tx buffer.*

The RXNE flag (Rx buffer not empty) is set on the last sampling clock edge, when the data is transferred from the shift register to the Rx buffer. It indicates that a data is ready to be read from the SPI\_DR register. An interrupt can be generated if RXIE bit in the SPI\_ICR register is set. Clearing the RXNE bit is performed by reading the SPI\_DR register.

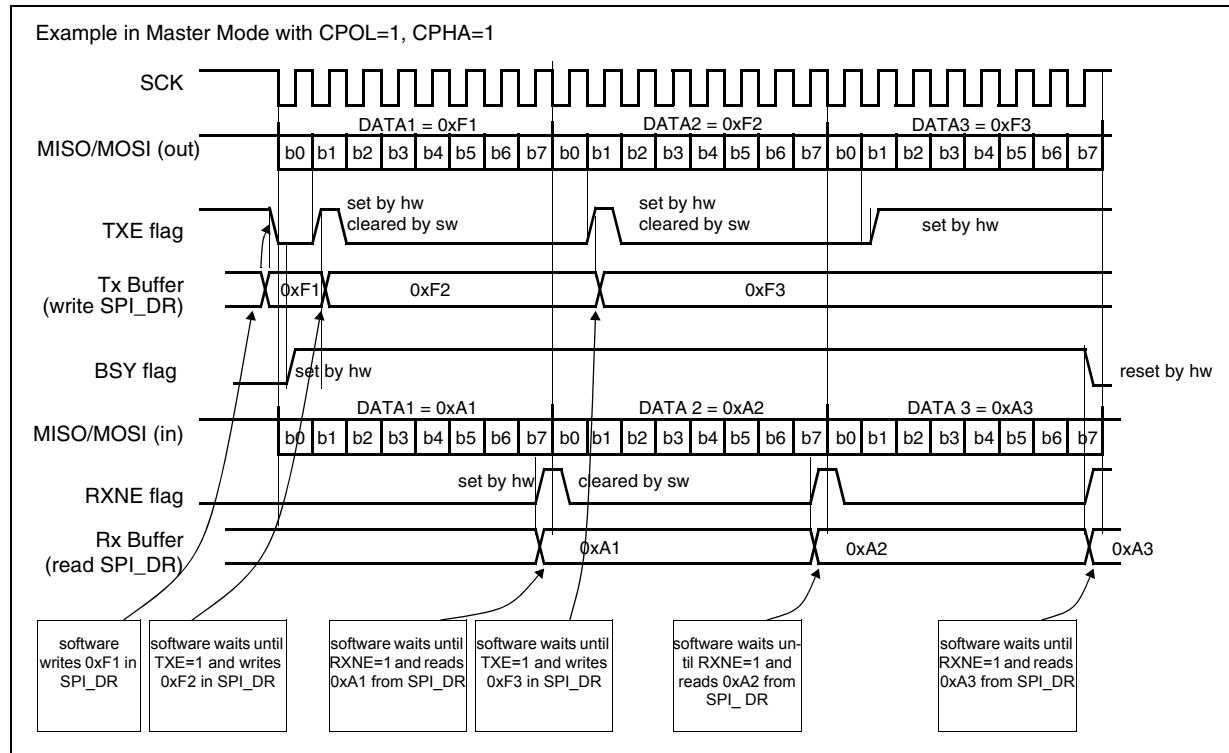
In some configurations, the BSY flag can be used during the last data transfer to wait until the completion of the transfer.

## Full Duplex Transmit and receive procedure in master or slave mode (BDM=0 and RXONLY = 0)

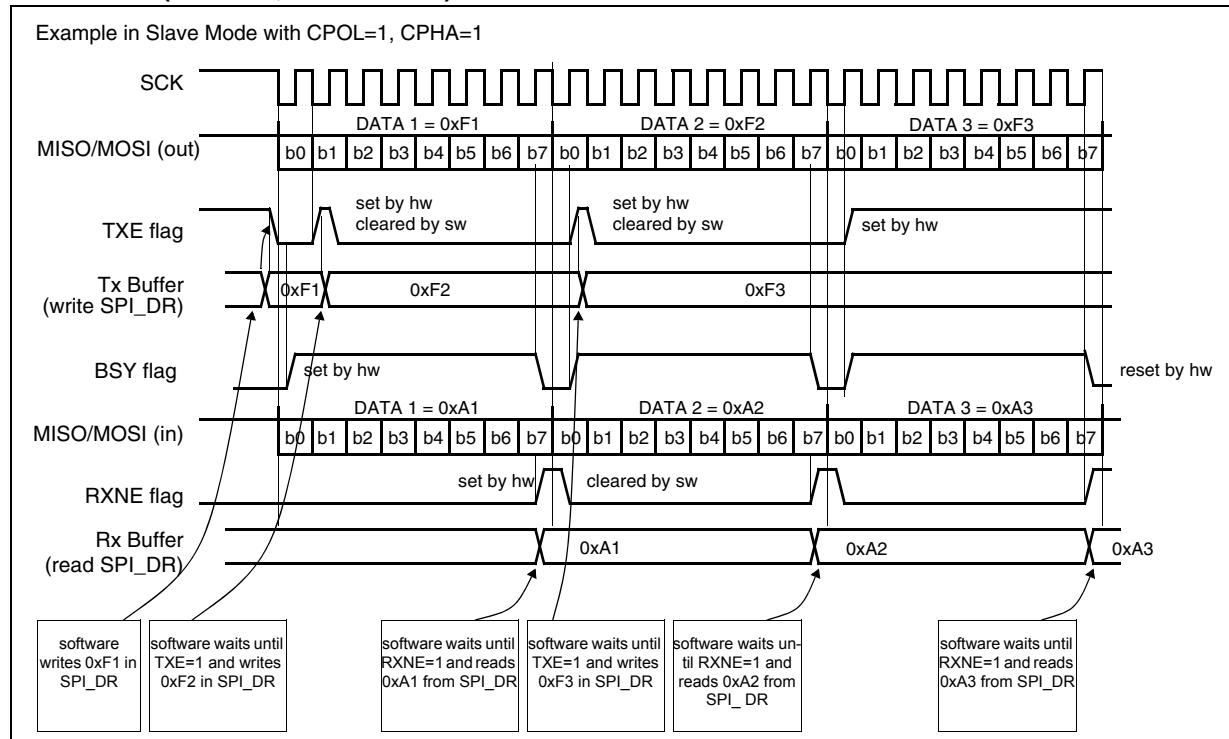
1. Enable the SPI by setting the SPE bit
2. Write the first data to be transmitted in the SPI\_DR register (this clears the TXE flag).
3. Wait until TXE = 1 and write the second data to be transmitted. Then wait until RXNE = 1 and read the SPI\_DR to get the first received data (this clears the RXNE bit). Repeat this operation for each data to be transmitted/received until the n-1 received data.
4. Wait until RXNE = 1 and read the last received data.
5. Wait until TXE = 1 and then wait until BSY = 0 before disabling the SPI.

This procedure can also be implemented using dedicated interrupt subroutines launched at each rising edge of RXNE or TXE flags.

**Figure 175. TXE/RXNE/BSY behavior in full duplex mode (RXONLY = 0). Case of continuous transfers**



**Figure 176. TXE/RXNE/BSY behavior in slave / full duplex mode (BDM = 0, RXONLY = 0). Case of continuous transfers**



### Transmit-only procedure (BDM = 0 RXONLY = 0)

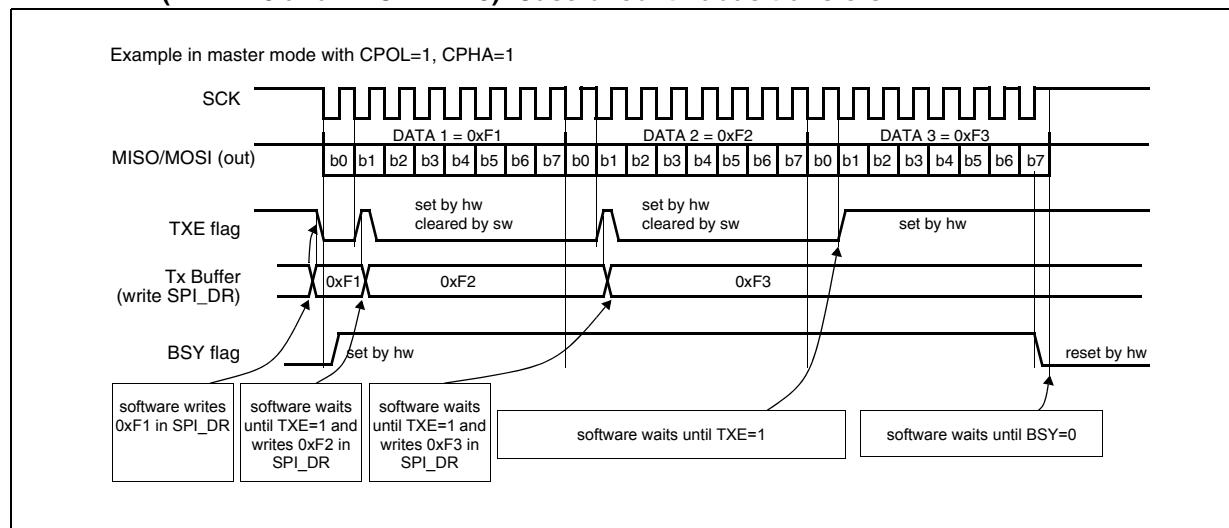
In this mode, the procedure can be reduced as described below and the BSY bit can be used to wait until the effective completion of the transmission (see [Figure 175](#) and [Figure 176](#)):

1. Enable the SPI by setting the SPE bit
2. Write the first data to send in the SPI\_DR register (this clears the TXE bit).
3. Wait until TXE = 1 and write the next data to be transmitted. Repeat this step for each data to be transmitted.
4. After writing the last data in the SPI\_DR register, wait until TXE = 1 and then wait until BSY=0 which indicates that the transmission of the last data is complete.

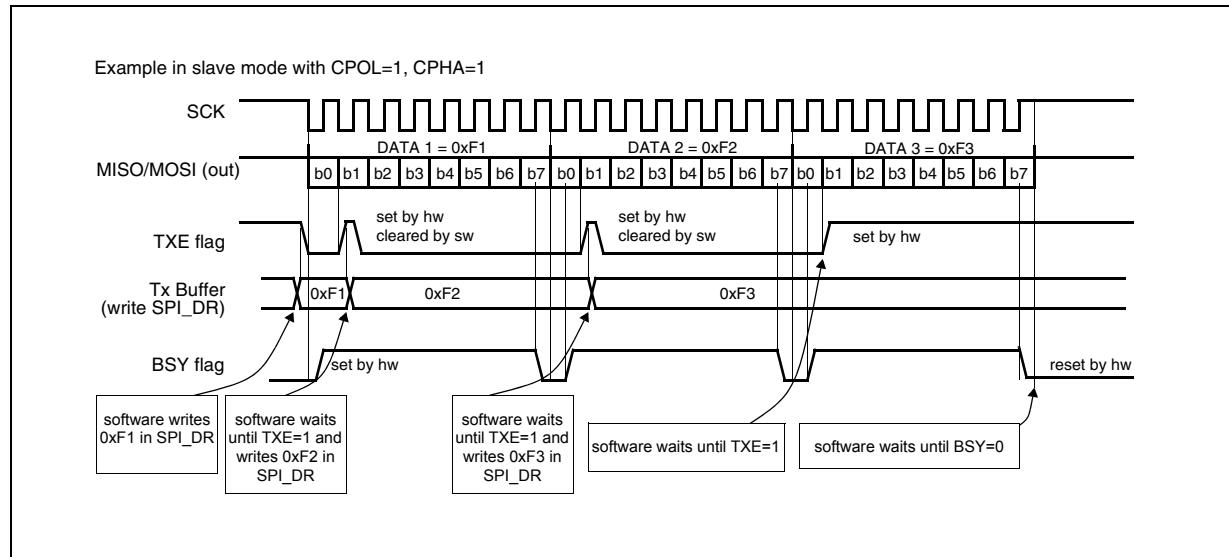
This procedure can be also implemented using dedicated interrupt subroutines launched at each rising edge of TXE flag.

- Note:**
- 1 *In master mode, during discontinuous communications, there is a 2 CPU clock period delay between the write operation to SPI\_DR and the BSY bit setting. As a consequence, in transmit-only mode, it is mandatory to wait first until TXE is set and then until BSY is reset after having written the last data.*
  - 2 *After transmitting two data in transmit-only mode, the OVR flag is set in the SPI\_SR register since the received data are never read.*

**Figure 177. TXE/BSY in master transmit-only mode  
(BDM = 0 and RXONLY = 0). Case of continuous transfers**



**Figure 178. TXE/BSY in slave transmit-only mode (BDM = 0 and RXONLY = 0). Case of continuous transfers**



### Bidirectional transmit procedure (BDM = 1 and BDOE = 1)

In this mode, the procedure is similar to the Transmit-only procedure except that the BDM and BDOE bits must both be set in the SPI\_CR2 register before enabling the SPI.

### Unidirectional receive-only procedure (BDM = 0 and RXONLY = 1)

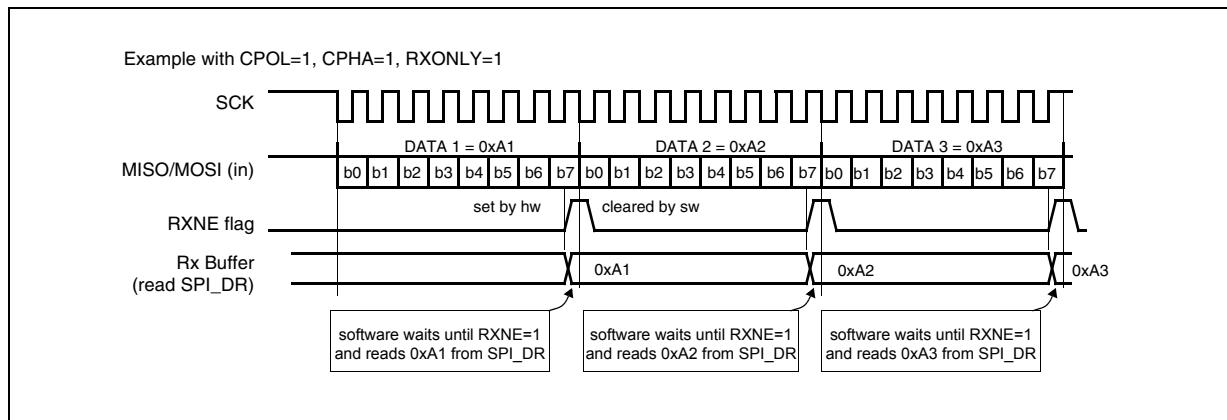
In this mode, the procedure can be reduced as described below (see [Figure 179](#)):

1. Set the RXONLY bit in the SPI\_CR2 register
2. Enable the SPI by setting bit SPE to 1:
  - a) In master mode, this immediately activates the generation of the SCK clock, and data is received serially until the SPI is disabled (SPE = 0).
  - b) In slave mode, data are received when the SPI master device drives NSS low and generates the SCK clock.
3. Wait until RXNE = 1 and read the SPI\_DR register to get the received data (this clears the RXNE bit). Repeat this operation for each data to be received.

This procedure can be also implemented using dedicated interrupt subroutines launched at each rising edge of the RXNE flag.

*Note:* If it is required to disable the SPI after the last transfer, follow the recommendation described in [Section 30.3.8: Disabling the SPI on page 556](#).

**Figure 179. RXNE behavior in receive-only mode (BDM = 0 and RXONLY = 1). Case of continuous transfers**



### Bidirectional receive procedure (BDM = 1 and BDOE = 0)

In this mode, the procedure is similar to the Receive-only procedure except that the BDM bit must be set and the BDOE bit must be reset in the SPI\_CR2 register before enabling the SPI.

### Continuous and discontinuous transfers

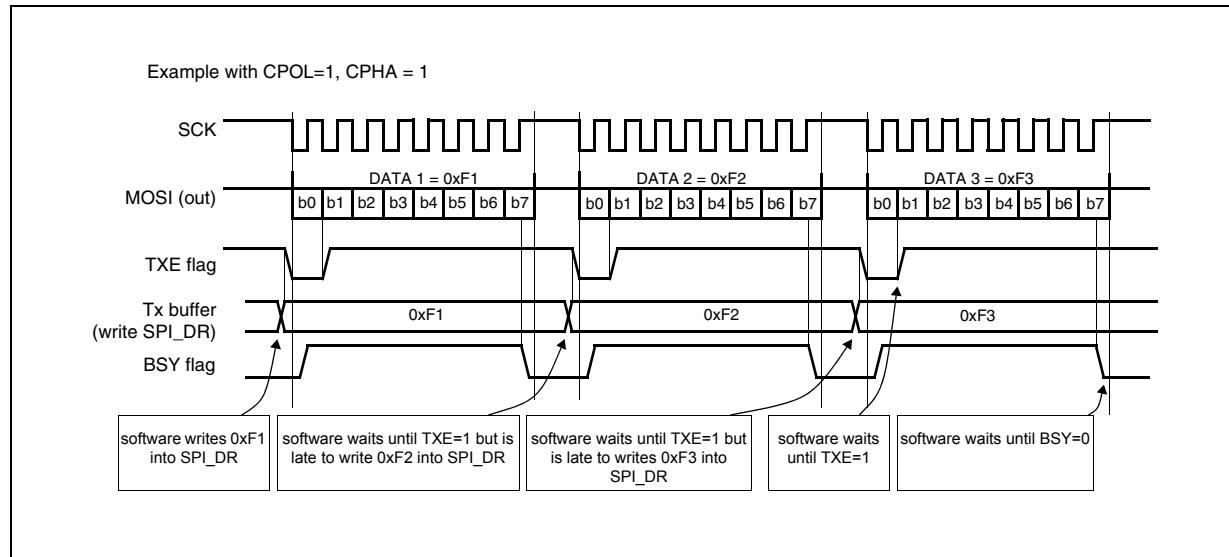
When transmitting data in master mode, if the software is fast enough to detect each TXE rising edge (or TXE interrupt) and to immediately write the SPI\_DR register before the ongoing data transfer is complete, the communication is said to be continuous. In this case, there is no discontinuity in the generation of the SPI clock between each data and the BSY bit will never be reset between each data transfer.

On the contrary, if the software is not fast enough, this can lead to some discontinuities in the communication. In this case, the BSY bit is reset between each data transmission (see [Figure 180](#)).

In master receive-only mode (BDM = 0 and RXONLY = 1) or in bidirectional receive mode (BDM = 1 and BDOE = 0), the communication is always continuous and the BSY flag is always read at 1.

In slave mode, the continuity of the communication is decided by the SPI master device. But even if the communication is continuous, the BSY flag goes low between each transfer for a minimum duration of one SPI clock cycle (see [Figure 176](#)).

**Figure 180. TXE/BSY behavior when transmitting (BDM = 0 and RXLONY = 0). Case of discontinuous transfers**



### 30.3.6 CRC calculation

A CRC calculator has been implemented for communication reliability. Separate CRC calculators are implemented for transmitted data and received data. The CRC is calculated using a programmable polynomial serially on each bit. The CRC is calculated on the sampling clock edge defined by the CPHA and CPOL bits in the SPI\_CR1 register.

CRC calculation is enabled by setting the CRCEN bit in the SPI\_CR1 register. This action resets the CRC registers (SPI\_RXCRCR and SPI\_TXCRCR). When the CRCNEXT bit in SPI\_CR2 is set, the SPI\_TXCRCR value is transmitted at the end of the current byte transmission.

If a byte is present in the Tx buffer, the CRC value is transmitted only after the transmission of this byte. During the transmission of CRC, the CRC calculator is switched off and the register value remains unchanged.

The CRCERR flag in the SPI\_SR register is set if the value received in the shift register during the SPI\_TXCRCR value transmission does not match the SPI\_RXCRCR value.

SPI communication using CRC is possible through the following procedure:

- Program the CPOL, CPHA, LSBfirst, BR, SSM, SSI and MSTR values.
- Program the polynomial in the SPI\_CRCPR register
- Enable the CRC calculation by setting the CRCEN bit in the SPI\_CR1 register. This also clears the SPI\_RXCRCR and SPI\_TXCRCR registers
- Enable the SPI by setting the SPE bit in SPI\_CR1
- Start the communication and sustain the communication until all but one byte has been transmitted or received.
- On writing the last byte to the Txbuffer, set the CRCNext bit in the SPI\_CR2 register to indicate that after transmission of the last byte, the CRC should be transmitted. The CRC calculation will be frozen during the CRC transmission.
- After transmitting the last byte, the SPI transmits the CRC. CRCNext bit is reset. The CRC is also received and compared against the SPI\_RXCRCR value. If the value does not match, the CRCERR flag in SPI\_SR is set and an interrupt can be generated when the ERRIE in the SPI\_ICR register is set.

**Note:**

*With high bit rate frequencies, the user must take care when transmitting CRC. As the number of used CPU cycles has to be as low as possible in the CRC transfer phase, the calling of software functions in the CRC transmission sequence is forbidden to avoid errors in the last data and CRC reception.*

*When the SPI is in slave mode, be careful to enable CRC calculation only when the clock is stable. If not, a wrong CRC calculation may be done.*

*For high bit rate frequencies, it is advised to use the DMA mode to avoid the degradation of the SPI speed performance due to CPU accesses impacting the SPI bandwidth.*

*When the STM8L05xx/15xx/16xx is configured as slave and the NSS hardware mode is used, the NSS pin needs to be kept low between the data phase and the CRC phase.*

When the SPI is configured in slave mode with the CRC feature enabled, CRC calculation takes place even if a high level is applied on the NSS pin. This may happen for example in case of a multislave environment where the communication master addresses slaves alternately.

Between a slave deselection (high level on NSS) and a new slave selection (low level on NSS), the CRC value should be cleared on both master and slave sides in order to resynchronize the master and slave for their respective CRC calculation.

To clear the CRC, follow the procedure below:

1. Disable SPI (SPE = 0)
2. Clear the CRCEN bit
3. Set the CRCEN bit
4. Enable the SPI (SPE = 1)

### 30.3.7 Status flags

There are three status flags to allow the application to completely monitor the state of the SPI bus.

### Tx buffer empty flag (TXE)

When set, this flag indicates that the Tx buffer is empty and that the next data to be transmitted can be loaded into the buffer. The TXE flag is reset when writing the SPI\_DR register.

### Rx buffer not empty (RXNE)

When set, this flag indicates that there is a valid received data in the Rx buffer. This flag is reset when SPI\_DR is read.

### Busy flag (BSY)

This BSY flag is set and reset by hardware (writing to this flag has no effect). The BSY flag indicates the state of the communication layer of the SPI.

When BSY is set, it indicates that the SPI is busy communicating. There is one exception in master mode / bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0) where the BSY flag is kept low during the reception.

The BSY flag is useful to detect the end of a transfer if the software wants to disable the SPI and enters Halt mode (or disable the peripheral clock). This will avoid corrupting the last transfer. For this, the procedure described below must be strictly respected.

The BSY flag is also useful to avoid write collisions in a multimaster system.

The BSY flag is set when a transfer starts with the exception of master mode / bidirectional receive mode (MSTR = 1 and BDM = 1 and BDOE = 0).

It is reset:

- when a transfer is finished (except in master mode if the communication is continuous)
- when the SPI is disabled
- when a master mode fault occurs (MODF = 1)

When communication is not continuous, the BSY flag is low between each communication.

When communication is continuous, in master mode, the BSY flag is kept high during the whole transfers.

When communication is continuous, in slave mode, the BSY flag goes back to low state for one SPI clock cycle between each transfer.

*Note:* *Do not use the BSY flag to handle each data transmission or reception. It is better to use TXE and RXNE flags instead.*

### 30.3.8 Disabling the SPI

When a transfer is terminated, the application can stop the communication by disabling the SPI peripheral. This is done by resetting the SPE bit.

For some configurations, disabling the SPI and entering Halt mode while a transfer is ongoing, can cause the current transfer to be corrupted and/or it can happen that the BSY flag becomes unreliable.

To avoid any of these effects, it is recommended to respect the following procedure when disabling the SPI:

**In master or slave full duplex mode (BDM = 0, RXONLY = 0):**

1. Wait until RXNE = 1 to receive the last data
2. Wait until TXE = 1
3. Then wait until BSY = 0
4. Disable the SPI (SPE = 0) and eventually enter Halt mode (or disable the peripheral clock).

**In master or slave unidirectional transmit-only mode (BDM = 0, RXONLY = 0) or bidirectional transmit mode (BDM = 1, BDOE = 1):**

After the last data is written in the SPI\_DR register:

1. Wait until TXE = 1
2. Then wait until BSY = 0
3. Disable the SPI (SPE = 0) and, if desired, enter Halt mode (or disable the peripheral clock).

**In master unidirectional receive-only mode (MSTR = 1, BDM = 0, RXONLY = 1) or bidirectional receive mode (MSTR = 1, BDM = 1, BDOE = 0):**

This case must be managed in a particular way to ensure that the SPI does not initiate a new transfer:

1. Wait for the second to last occurrence of RXNE = 1 (n-1)
2. Then wait for one SPI clock cycle (using a software loop) before disabling the SPI (SPE = 0)
3. Then wait for the last RXNE=1 before entering Halt mode (or disabling the peripheral clock).

Note:

*In master bidirectional receive mode (MSTR=1 and BDM=1 and BDOE=0), the BSY flag is kept low during a transfer.*

**In slave receive-only mode (MSTR = 0, BDM = 0, RXONLY = 1) or bidirectional receive mode (MSTR = 0, BDM = 1, BDOE = 0):**

1. You can disable the SPI (write SPE = 1) whenever you want: the current transfer will complete before being effectively disabled.
2. Then, if you want to enter Halt mode, you must first wait until BSY = 0 before entering Halt mode (or disabling the peripheral clock).

### 30.3.9 SPI communication using DMA (direct memory addressing)

To operate at its maximum speed, the SPI needs to be fed with the data to be transmitted, and the data received on the Rx buffer should be read to avoid overrun. To facilitate the transfers, the SPI features a DMA capability implementing a simple request/acknowledge protocol.

A DMA access is requested when the enable bit in the SPI\_ICR register is enabled. Separate requests must be issued to the Tx and the Rx buffers (see [Figure 181](#) and [Figure 182](#)):

- In transmission, a DMA request is issued each time TXE is set to 1. The DMA then writes the SPI\_DR (this clears the TXE flag).
- In reception, a DMA request is issued each time RXNE is set to 1. The DMA then reads the SPI\_DR (this clears the RXNE flag).

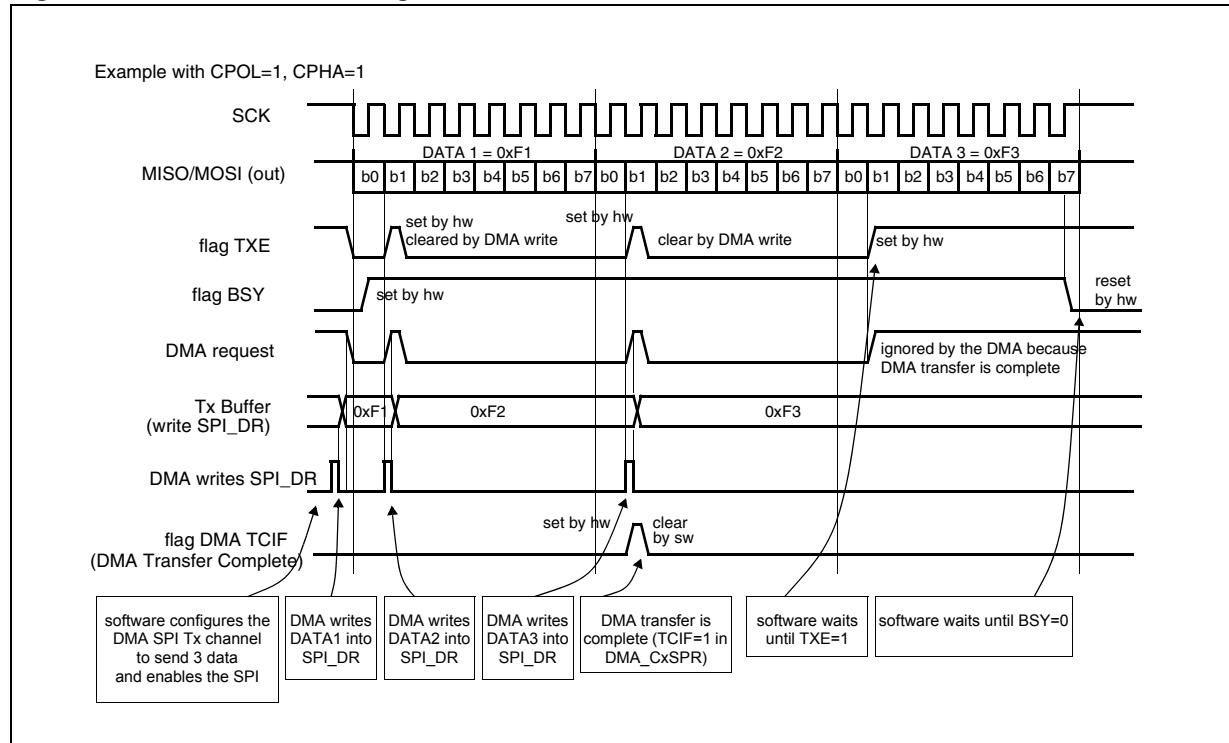
When the SPI is used only to transmit data, it is possible to enable only the SPI Tx DMA channel. In this case, the OVR flag is set because the data received are not read.

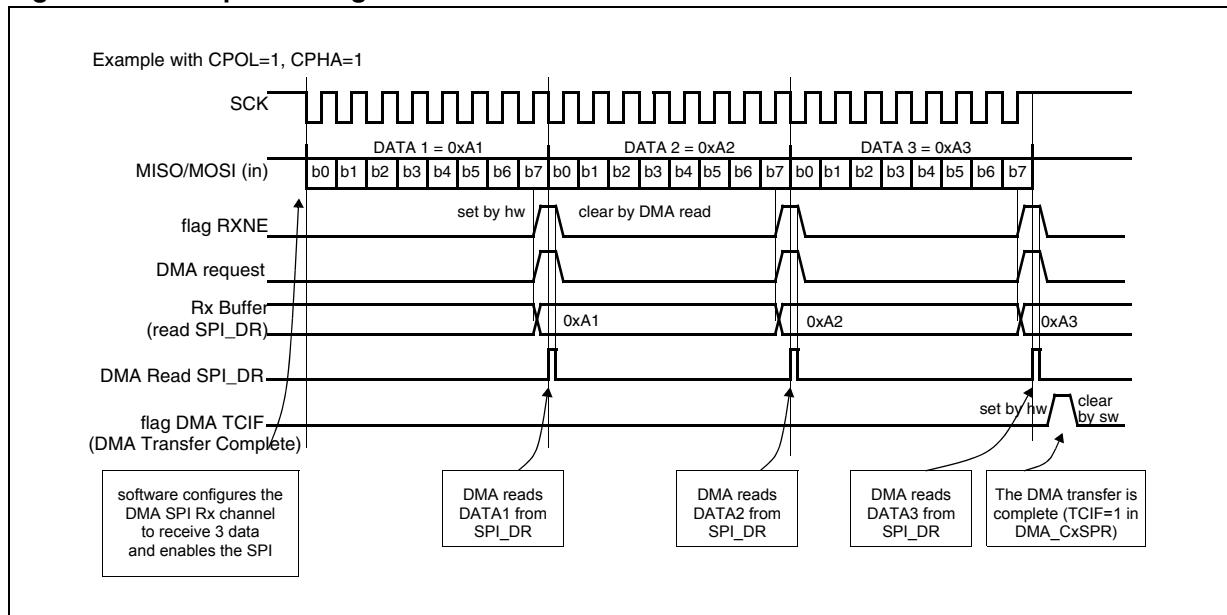
When the SPI is used only to receive data, it is possible to enable only the SPI Rx DMA channel.

In transmission mode, when the DMA has written all the data to be transmitted (flag TCIF is set in the DMA\_CxSPR register), the BSY flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or entering Halt mode. The software must first wait until TXE=1 and then wait until BSY=0.

**Note:** *During discontinuous communications, there is a 2 CPU clock period delay between the write operation to SPI\_DR and the BSY bit setting. As a consequence, it is mandatory to wait first until TXE=1 and then until BSY=0 after having written the last data.*

**Figure 181. Transmission using DMA**



**Figure 182. Reception using DMA**

### 30.3.10 Error flags

#### Master mode fault (MODF)

Master mode fault occurs when the master device has its NSS pin pulled low (in NSS hardware mode) or SSI bit low (in NSS software mode), this automatically sets the MODF bit. Master mode fault affects the SPI peripheral in the following ways:

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

Use the following software sequence to clear the MODF bit:

1. Make a read or write access to the SPI\_SR register while the MODF bit is set.
2. Then write to the SPI\_CR1 register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled high during the MODF bit clearing sequence. The SPE and MSTR bits can be restored to their original state after this clearing sequence.

As a security, hardware does not allow you to set the SPE and MSTR bits while the MODF bit is set.

In a slave device the MODF bit cannot be set. However, in a multi-master configuration, the device can be in slave mode with this MODF bit set. In this case, the MODF bit indicates that there might have been a multimaster conflict for system control. You can use an interrupt routine to recover cleanly from this state by performing a reset or returning to a default state.

### Overrun condition

An overrun condition occurs, when the master device has sent data bytes and the slave device has not cleared the RXNE bit resulting from the previous data byte transmitted.

When an overrun condition occurs:

- OVR bit is set and an interrupt is generated if the ERRIE bit is set.

In this case, the receiver buffer contents will not be updated with the newly received data from the master device. A read to the SPI\_DR register returns this byte. All other subsequently transmitted bytes are lost.

Clearing the OVR bit is done by a read access to the SPI\_DR register followed by a read access to the SPI\_SR register.

### CRC error

This flag is used to verify the correctness of the value received when the CRCEN bit in the SPI\_CR2 register is set. The CRCERR flag in the SPI\_SR register is set if the value received in the shift register after the SPI\_TXCRCR value transmission does not match the SPI\_RXCRCR value. Refer to [Chapter 30.3.6: CRC calculation](#).

## 30.3.11 SPI low power modes

**Table 105. SPI behavior in low power modes**

Mode	Description
Wait/Low power wait	No effect on SPI. SPI interrupt events cause the device to exit from Wait or Low power wait mode.
Halt/Active-halt	SPI registers are frozen. In Halt mode, the SPI is inactive. If the SPI is in master mode, then communication resumes when the device is woken up by an interrupt with "wakeup from Halt mode" capability. If the SPI is in slave mode, then it can wake up the MCU from Halt mode after detecting the first sampling edge of data.

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

When the microcontroller is in Halt mode, the SPI is still capable of responding as a slave provided the NSS pin is tied low or the SSI bit is reset before entering Halt mode.

When the first sampling edge of data (as defined by the CPHA bit) is detected:

- The WKUP bit is set in the SPI\_SR register
- An interrupt is generated if the WKIE bit in the SPI\_ICR register is set.
- This interrupt wakes up the device from Halt mode.
- Due to the time needed to restore the system clock, the SPI slave sends or receives a few data before being able to communicate correctly. It is then mandatory to use the following protocol:
  - A specific value is written into the SPI\_DR before entering Halt mode. This value indicates to the external master that the SPI is in Halt mode
  - The external master sends the same byte continuously until it receives from the SPI slave device a new value other than the unique value indicating the SPI is in Halt mode. This new value indicates the SPI slave has woken-up and can correctly communicate.

### Restrictions in receive-only modes

The wake-up functionality is not guaranteed in receive-only modes (BDM = 0 and RXONLY = 1 or BDM = 1 and BDOE = 0) since the time needed to restore the system clock can be greater than the data reception time. A loss of data in reception would then be induced and the slave device can not indicate to the master which data has been properly received.

### 30.3.12 SPI interrupts

**Table 106. SPI interrupt requests**

Interrupt event	Event flag	Enable control bit	Exit from Wait/Low power wait	Exit from Halt /Active-halt
Transmit buffer empty flag	TXE	TXIE	Yes	No
Receive buffer not empty flag	RXNE	RXIE	Yes	No
Wakeup event flag	WKUP	WKIE	Yes	Yes
Master mode fault event	MODF	ERRIE	Yes	No
Overrun error	OVR		Yes	No
CRC error flag	CRCERR		Yes	No

## 30.4 SPI registers

### 30.4.1 SPI control register 1 (SPI\_CR1)

Address offset: 0x00

Reset value: 0x00

7	6	5	4	3	2	1	0
LSBFIRST	SPE		BR [2:0]		MSTR	CPOL	CPHA
rw	rw		rw		rw	rw	rw

**Bit 7 LSBFIRST:** Frame format <sup>(1)</sup>

- 0: MSB is transmitted first
- 1: LSB is transmitted first

**Bit 6 SPE:** SPI enable <sup>(2)</sup>

- 0: Peripheral disabled
- 1: Peripheral enabled

**Bits 5:3 BR[2:0]:** Baud rate control

- |                              |                               |
|------------------------------|-------------------------------|
| 000: f <sub>SYSCLK</sub> /2  | 100: f <sub>SYSCLK</sub> /32  |
| 001: f <sub>SYSCLK</sub> /4  | 101: f <sub>SYSCLK</sub> /64  |
| 010: f <sub>SYSCLK</sub> /8  | 110: f <sub>SYSCLK</sub> /128 |
| 011: f <sub>SYSCLK</sub> /16 | 111: f <sub>SYSCLK</sub> /256 |

*Note: These bits should not be changed when the communication is ongoing.*

**Bit 2 MSTR:** Master selection <sup>(1)</sup>

- 0: Slave configuration
- 1: Master configuration

Bit1 **CPOL:** Clock polarity (1)

- 0: SCK to 0 when idle
- 1: SCK to 1 when idle

Bit 0 **CPHA:** Clock phase (1)

- 0: The first clock transition is the first data capture edge
- 1: The second clock transition is the first data capture edge

1. This bit should not be changed when the communication is ongoing.
2. When disabling the SPI, follow the procedure described in [Section 30.3.8: Disabling the SPI on page 556](#)

### 30.4.2 SPI control register 2 (SPI\_CR2)

Address offset: 0x01

Reset value: 0x00

7	6	5	4	3	2	1	0
BDM	BDOE	CRCEN	CRCNEXT	Reserved	RXOnly	SSM	SSI

Bit 7 **BDM:** Bidirectional data mode enable

- 0: 2-line unidirectional data mode selected
- 1: 1-line bidirectional data mode selected

Bit 6 **BDOE:** Input/Output enable in bidirectional mode

This bit selects the direction of transfer in bidirectional mode when BDM is set to 1.

- 0: Input enabled (receive-only mode)
- 1: Output enabled (transmit-only mode)

In master mode, the MOSI pin is used and in slave mode, the MISO pin is used.

Bit 5 **CRCEN:** Hardware CRC calculation enable

- 0: CRC calculation disabled
- 1: CRC calculation Enabled

*Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation*

Bit 4 **CRCNEXT:** Transmit CRC next

- 0: Next transmit value is from Tx buffer
- 1: Next transmit value is from Tx CRC register

Bit 3 Reserved

Bit 2 **RXONLY:** Receive only

- 0: Full duplex (Transmit and receive)
- 1: Output disabled (Receive only mode)

This bit combined with BDM bit selects the direction of transfer in 2 line uni-directional mode

This bit is also useful in a multi-slave system in which this particular slave is not accessed, the output from the accessed slave is not corrupted.

Bit 1 **SSM:** Software slave management

- 0: Software slave management disabled
- 1: Software slave management enabled

When the SSM bit is set, the NSS pin input is replaced with the value coming from the SSI bit

Bit 0 **SSI**: Internal slave select

This bit has effect only when SSM bit is set. The value of this bit is forced onto the NSS pin and the I/O value of the NSS pin is ignored.

- 0: Slave mode
- 1: Master mode

### 30.4.3 SPI interrupt control register (SPI\_ICR)

Address offset: 0x02

Reset value: 0x00

7	6	5	4	3	2	1	0
<b>TXIE</b>	<b>RXIE</b>	<b>ERRIE</b>	<b>WKIE</b>	Reserved		<b>TXDMAEN</b>	<b>RXDMAEN</b>
<i>rw</i>	<i>rw</i>	<i>rw</i>	<i>rw</i>			<i>rw</i>	<i>rw</i>

Bit 7 **TXIE**: Tx buffer empty interrupt enable <sup>(1)</sup>

- 0: TXE interrupt masked
- 1: TXE interrupt not masked. This allows an interrupt request to be generated when the TXE flag is set.

Bit 6 **RXIE**: RX buffer not empty interrupt enable <sup>(2)</sup>

- 0: RXNE interrupt masked
- 1: RXNE interrupt not masked. This allows an interrupt request to be generated when the RXNE flag is set.

Bit 5 **ERRIE**: Error interrupt enable

- 0: Error interrupt is masked
- 1: Error interrupt is enabled. This allows an interrupt request to be generated when an error condition occurs (CRCERR, OVR, MODF)

Bit 4 **WKIE**: Wakeup interrupt enable

- 0: wakeup interrupt masked
- 1: wakeup interrupt enabled. This allows an interrupt request to be generated when the WKUP flag is set.

Bits 3:2 Reserved

Bit 1 **TXDMAEN**: *Tx Buffer DMA Enable*

When this bit is set, the DMA request is made whenever the TXE flag is set.

- 0: Tx buffer DMA disabled
- 1: Tx buffer DMA enabled

Bit 0 **RXDMAEN**: *Rx Buffer DMA Enable*

When this bit is set, the DMA request is made whenever the RXNE flag is set.

- 0: Rx buffer DMA disabled
- 1: Rx buffer DMA enabled

1. To function correctly, the TXIE and TXDMAEN bits should not be set at the same time.

2. To function correctly, the RXIE and RXDMAEN bits should not be set at the same time.

### 30.4.4 SPI status register (SPI\_SR)

Address offset: 0x03

Reset value: 0x02

7	6	5	4	3	2	1	0
BSY	OVR	MODF	CRCERR	WKUP	Reserved	TXE	RXNE

Bit 7 **BSY**: Busy flag

0: SPI not busy

1: SPI is busy in communication

This flag is set and reset by hardware.

*Note: BSY flag must be used with caution: refer to Section 30.3.7: Status flags on page 555 and Section 30.3.8: Disabling the SPI on page 556*

Bit 6 **OVR**: Overrun flag

0: No Overrun occurred

1: Overrun occurred

This flag is set by hardware and reset by a software sequence.

Bit 5 **MODF**: Mode fault

0: No Mode fault occurred

1: Mode fault occurred

This flag is set by hardware and reset by a software sequence.

Bit 4 **CRCERR**: CRC error flag

0: CRC value received matches the SPI\_RXCRCR value

1: CRC value received does not match the SPI\_RXCRCR value

This flag is set by hardware and cleared by software writing 0.

Bit 3 **WKUP**: Wakeup flag

0: No wakeup event occurred

1: Wakeup event occurred

This flag is set on the first sampling edge on SCK when the STM8 is in Halt mode and the SPI is configured as slave.

This flag is reset by software writing 0.

Bit 2 Reserved

Bit 1 **TXE**: Transmit buffer empty

0: Tx buffer not empty

1: Tx buffer empty

Bit 0 **RXNE**: Receive buffer not empty

0: Rx buffer empty

1: Rx buffer not empty

### 30.4.5 SPI data register (SPI\_DR)

Address offset: 0x04

Reset value: 0x00

7	6	5	4	3	2	1	0
DR[7:0]							
rw							

Bits 7:0 **DR[7:0]: Data register**

Byte received or to be transmitted.

The data register is split into 2 buffers - one for writing (Transmit buffer) and another one for reading (Receive buffer). A write to the data register will write into the Tx buffer and a read from the data register will return the value held in the Rx buffer.

### 30.4.6 SPI CRC polynomial register (SPI\_CRCPR)

Address offset: 0x05

Reset value: 0x07

7	6	5	4	3	2	1	0
CRCPOLY[7:0]							
rw							

Bits 7:0 **CRCPOLY[7:0]: CRC polynomial register**

This register contains the polynomial for the CRC calculation.

The CRC polynomial (0x07) is the reset value of this register. You can configure an other polynomial as required for your application.

### 30.4.7 SPI Rx CRC register (SPI\_RXCRCR)

Address offset: 0x06Reset value: 0x00

7	6	5	4	3	2	1	0
RxCRC[7:0]							
r							

Bits 7:0 **RxCRC[7:0]: Rx CRC Register**

When CRC calculation is enabled, the RxCRC[7:0] bits contain the computed CRC value of the subsequently received bytes. This register is reset when the CRCEN bit in SPI\_CR2 register is written to 1. The CRC is calculated serially using the polynomial programmed in the SPI\_CRCPR register.

*Note: A read to this register when the BSY Flag is set could return an incorrect value.*

### 30.4.8 SPI Tx CRC register (SPI\_RXCRCR)

Address offset: 0x07 Reset value: 0x00

7	6	5	4	3	2	1	0
TxCRC[7:0]							
r							

Bits 7:0 **TxCRC[7:0]**: Tx CRC register

When CRC calculation is enabled, the TxCRC[7:0] bits contain the computed CRC value of the subsequently transmitted bytes. This register is reset when the CRCEN bit of SPI\_CR2 is written to 1. The CRC is calculated serially using the polynomial programmed in the SPI\_CRCPR register.

*Note: A read to this register when the BSY flag is set could return a incorrect value*

## 30.5 SPI register map and reset values

Table 107. SPI register map and reset values

Address offset	Register name	7	6	5	4	3	2	1	0
0x00	SPI_CR1 reset value	LSB FIRST 0	SPE 0	BR2 0	BR1 0	BR1 0	MSTR 0	CPOL 0	CPHA 0
0x01	SPI_CR2 reset value	BDM 0	BDOE 0	CRCEN 0	CRCNEXT 0	- 0	RXONLY 0	SSM 0	SSI 0
0x02	SPI_ICR reset value	TXIE 0	RXIE 0	ERRIE 0	WKIE 0	- 0	- 0	TXDMAE N 0	RXDMAEN 0
0x03	SPI_SR reset value	BSY 0	OVR 0	MODF 0	CRCERR 0	WKUP 0	- 0	TXE 1	RXNE 0
0x04	SPI_DR reset value	MSB 0	- 0	- 0	- 0	- 0	- 0	- 0	LSB 0
0x05	SPI_CRCPR reset value	MSB 0	- 0	- 0	- 0	- 0	- 1	- 1	LSB 1
0x06	SPI_RXCRCR reset value	MSB 0	- 0	- 0	- 0	- 0	- 0	- 0	LSB 0
0x07	SPI_TXCRCR reset value	MSB 0	- 0	- 0	- 0	- 0	- 0	- 0	LSB 0

## 31 Revision history

**Table 108. Document revision history**

Date	Revision	Changes
05-Aug-2009	1	<p>Initial release.</p>
10-Sep-2009	2	<p>Document main sections reorganized, and ADC, SPI, I2C, USART and DMA peripherals renamed ADC1, SPI1, I2C1, USART1, and DMA1.</p> <p>Updated <a href="#">Figure 9: Reset circuit</a>. Updated <a href="#">Section 7.8.1: Entering Halt mode</a>. Added <a href="#">Note 1</a> below <a href="#">Table 10</a>. Changed BGF to VREFINTF in PWR_CR2 register.</p> <p>Whole Clock control section reorganized. Updated <a href="#">Figure 13 on page 89</a>.</p> <p>Updated <a href="#">Section 12.3: Interrupts and low power modes</a>, and <a href="#">Section 12.9.8: External interrupt port select register (EXTI_CONF1)</a></p> <p>Changed f<sub>MASTER</sub> and SYSCLK/DIV to SYSCLK. <a href="#">Section 9.13.1</a>, <a href="#">Section 9.13.2</a>, <a href="#">Section 9.13.3</a>, <a href="#">Section</a>, and <a href="#">Section 9.13.12</a> updated. Added PCKEN26 in <a href="#">Table 17: Peripheral clock gating bits (PCKEN 20 to PCKEN 25)</a>. Modified <a href="#">Section 11: Routing interface (RI) and system configuration controller (SYSCFG) on page 109</a></p> <p>Updated <a href="#">Figure 15</a> and <a href="#">Section Table 22.: I/O groups and selection on page 111</a>.</p> <p>Updated <a href="#">Transmission using DMA</a>.</p> <p>Added <a href="#">Section 15.3.3: DAC output switch configuration</a>.</p> <p>Updated <a href="#">Figure 32: Comparator block diagram</a>, <a href="#">Figure 33</a>. Modified COMP1 configuration sequence in <a href="#">Section 16.3</a>. Updated procedure step 1 in <a href="#">Section 16.4.1: Using the comparators in window mode</a>. Updated <a href="#">Section 16.7.1: Comparator control and status register 1 (COMP_CSR1)</a>, <a href="#">Section 16.7.2: Comparator control and status register 2 (COMP_CSR2)</a>, Updated <a href="#">Section 16.7.4</a> and <a href="#">Section 16.7.5</a>.</p> <p>Updated <a href="#">Section 17.3.2: Frequency generator</a> in <a href="#">Section 17: LCD controller</a>.</p> <p>Updated <a href="#">Figure 59: Clock/trigger controller block diagram</a>, <a href="#">Figure 69: Timer chaining system implementation example</a>, <a href="#">Figure 70: Trigger/master mode selection blocks</a>, <a href="#">Figure 102: Input stage block diagram</a>, and <a href="#">Figure 106: TIM4 block diagram</a>. Modified TS bit definition in <a href="#">Section 19.8.3</a>, <a href="#">Section 20.5.3</a>, and <a href="#">Section 21.5.3</a>. Added <a href="#">Note 1</a> under <a href="#">Table 61: Output control for complementary OC<sub>i</sub> and OC<sub>iN</sub> channels with break feature</a>.</p> <p>Restored TIMx XOR in <a href="#">Figure 79</a>, <a href="#">Section 19.8.2: Control register 2 (TIM1_CR2)</a>. Updated OC4FE bit in <a href="#">Section 19.8.13: Capture/compare mode register 4 (TIM1_CCMR4)</a>. Updated <a href="#">Section 20.3.3: Capture/compare channels</a>. Updated <a href="#">Section 23.1: Introduction</a> and <a href="#">Figure 108: BEEP block diagram</a> in <a href="#">Section 23: Beeper (BEEP)</a>. Changed f<sub>L<sub>S</sub></sub> to BEEPCLK.</p> <p>Updated RTC <a href="#">Clock and prescalers</a>, <a href="#">Section 24.3.4: Periodic auto-wakeup</a>. Added note in <a href="#">Section 24.9.14: Asynchronous prescaler register (RTC_APERR)</a>.</p>

**Table 108. Document revision history (continued)**

Date	Revision	Changes
10-Sep-2009	2 (cont.)	<p>Updated <a href="#">Figure 110: Independent watchdog (IWDG) block diagram</a>, <a href="#">Table 72: Minimum/maximum values of IWDG timeout period (LSI clock frequency = 38 kHz)</a>, and <a href="#">Section 25.3: Using the IWDG in Halt/Active-halt mode</a>. Updated <a href="#">Figure 111</a> and <a href="#">Figure 112</a>. Replaced fckc_wwdg_ck by SYSCLK in the whole Window watchdog section. Removed section “Using Halt mode with the WWDG” and Figure “Approximate timeout duration”. Update <a href="#">Section 26.4</a>.</p> <p><a href="#">Section 29.3.5: Data transmission and reception procedures</a>: SPI timing diagrams revised and description of receive-only mode expanded. Added <a href="#">Section 29.3.8: Disabling the SPI</a>.</p>
11-Dec-2009	3	<p>Removed proprietary code area (PCODE).</p> <p>Removed TLI interrupt vector.</p> <p>Added <a href="#">Section 2: Boot ROM on page 33</a></p> <p>Modified <a href="#">Figure 3: Medium density STM8L15x Flash program and data EEPROM organization on page 36</a></p> <p>Updated <a href="#">Section 6.3: Brownout reset (BOR) on page 55</a> and <a href="#">Section 6.4: Programmable voltage detector (PVD) on page 56</a></p> <p>Modified <a href="#">Figure 4: UBC area size definition on page 37</a></p> <p>Modified bit 6 name and description in <a href="#">Section 6.6.1: Power control and status register 1 (PWR_CSR1) on page 58</a></p> <p>Modified description of Low power run mode in <a href="#">Section 7: Low power modes on page 60</a></p> <p>Modified <a href="#">Table 10: Low power mode summary on page 60</a></p> <p>Modified <a href="#">Section 7.6: Low power run mode on page 68</a></p> <p>Modified <a href="#">Figure 9: Reset circuit on page 71</a></p> <p>Modified clock frequency (38 kHz instead of 32 kHz in <a href="#">Section 9.5: LSI clock on page 78</a>)</p> <p>Added note to <a href="#">Section : Manual switching on page 79</a></p> <p>Changed position of <a href="#">Section 4: Single wire interface module (SWIM) and debug module (DM) on page 52</a></p> <p>Modified <a href="#">Section 12.9.8: External interrupt port select register (EXTI_CONF1) on page 140</a></p> <p>Modified <a href="#">Table 32: DMA1 channel request mapping on page 150</a> (ADC1 and DAC)</p> <p>Updated <a href="#">Section : DAC DMA requests on page 153</a></p> <p>Replaced t<sub>STAB</sub> with t<sub>WKUP</sub> in <a href="#">Section 14.3: ADC functional description on page 167</a></p> <p>Modified <a href="#">Section 14.3.3: ADC on-off control on page 168</a> (ADC_CR1 instead of ADC_CR2)</p> <p>Modified <a href="#">Section 14.3.4: Single conversion mode on page 168</a> (added one note)</p> <p>Modified <a href="#">Section 14.3.5: Continuous conversion mode on page 170</a></p> <p>Modified <a href="#">Section 14.3.9: Channel selection (Scan mode) on page 171</a></p> <p>Updated <a href="#">Section 14.3.16: Temperature sensor on page 174</a> (added one paragraph)</p> <p>Added <a href="#">Section 14.3.17: Internal reference voltage conversion on page 175</a></p> <p>Added note in <a href="#">Section 16.5: COMP low power modes on page 204</a></p> <p>Modified <a href="#">Section 19.7.1: DMA single mode on page 287</a></p>

**Table 108. Document revision history (continued)**

Date	Revision	Changes
11-Dec-2009	3 (cont.)	<p>Modified Section 19.7.2: DMA burst mode on page 288</p> <p>Modified Section 15.3.8: DAC DMA request (TSEL bit condition)</p> <p>Modified Section 25.1: Introduction on page 392</p> <p>Modified Section 24: Real-time clock (RTC) on page 368</p> <p><i>Figure 146: Data clock timing diagram on page 483:</i> Removed “from master” and “from slave” beneath MISO and MOSI respectively.</p> <p>Modified Section 29.4.5: SPI data register (SPI_DR) on page 504 and Section 29.4.6: SPI CRC polynomial register (SPI_CRCPR) on page 504</p> <p><i>Master mode fault (MODF) on page 497:</i> SPE and MSTR bits can be returned to their original state only after a MODF bit clearing sequence.</p> <p>Updated Bus error (BERR) on page 416</p>
11-Feb-2010	4	<p>EXTI_CONF1 instead of EXTI_CONF</p> <p>Modified Section 9.4: LSE clock on page 78</p> <p>Modified Section 9.11: Configurable clock output capability (CCO) on page 85</p> <p>Modified Section 9.13.5: Peripheral clock gating register 2 (CLK_PCKENR2) on page 91 (PCKEN26 and PCKEN27 removed)</p> <p>and <i>Table 17: Peripheral clock gating bits (PCKEN 20 to PCKEN 25) on page 91</i> (WWDG removed)</p> <p>and Modified Bit 3 and bit 5 description in Section 9.13.7: External clock register (CLK_ECKCR) on page 93</p> <p>Modified Section 11.2.2: I/O groups on page 126,</p> <p>Modified description of Section 9.13.13: HSI calibration register (CLK_HSICALR) on page 97</p> <p>Modified Section 9.13.14: HSI clock calibration trimming register (CLK_HSITRIMR) on page 98</p> <p>Replaced SPI1_REMAP by SPI1_REMAP1 in Section 11.5.1: SYSCFG remap control register 1 (SYSCFG_RMPCR1) on page 123</p> <p>Modified Section 11.5.1: SYSCFG remap control register 1 (SYSCFG_RMPCR1) on page 123</p> <p>Replaced TIM3TRIG_REMAP by TIM3TRIG_REMAP1 in Section 11.5.2 and Section 11.5.3</p> <p>Added Figure 14: GPIO block diagram on page 102</p> <p>Removed references to 64-pin packages in Section 11.3.4: DAC routing on page 113</p> <p>Replaced DMA1 by DMA in Section 19.7.1: DMA single mode on page 287, Section 19.7.2: DMA burst mode on page 288 and Figure 99: DMA burst modes example on page 288</p> <p>Added Section 19.5.11: Timer input XOR function on page 285, Section 19.5.12: Interfacing with Hall sensors on page 285 and Section 20.3.4: Timer input XOR function on page 328</p> <p>Modified Section 24.8: RTC interrupts on page 376</p> <p>Modified note in Section 24.9.18: Alarm A register 1 (RTC_ALRMAR1) on page 387 (RTC_CR2 instead of RTC_CR1)</p> <p>Modified Table 80: I2C_CCR values for SCL frequency table (fSYSCLK = 10 MHz or 16 MHz) on page 437</p> <p>Modified Table 83: Baud rate programming and error calculation on page 451</p> <p>Modified BERR bit description in Section 28.7.9: Status register 2 (I2C_SR2) on page 494</p>

**Table 108. Document revision history (continued)**

Date	Revision	Changes
26-Jul-2010	5	<p>Modified Introduction on 1st page</p> <p>Modified address offset in <i>SYSCFG remap control register 1 (SYSCFG_RMPCR1)</i> and <i>SYSCFG remap control register 2 (SYSCFG_RMPCR2)</i></p> <p>Modified <i>Section 8: Reset (RST) on page 84</i></p> <p>Added <i>Section 8.1: “Reset state” and “under reset” definitions on page 84</i></p> <p>Modified <i>Section 2: Boot ROM</i> (peripheral references)</p> <p>Modified <i>Figure 17: Clock structure on page 88</i></p> <p>Modified <i>Section 9.5: LSI clock on page 92</i></p> <p>Modified <i>Section 9.8: Clock security system (CSS) on page 96</i></p> <p>UBC last page address updated</p> <p>Added <i>Section 9.12: Clock-independent system clock sources for TIM2/TIM3 on page 100</i></p> <p>Modified <i>Section 9.14.5: Peripheral clock gating register 2 (CLK_PCKENR2) on page 105</i></p> <p>Modified reset value in <i>Section 9.14.2: Clock RTC register (CLK_CRTCR) on page 101</i></p> <p>Added note to <i>Section 7.8.1: Entering Halt mode on page 82</i> and <i>Section 7.9: Active-halt mode on page 83</i></p> <p>Modified <i>Section 10.4: Reset configuration on page 119</i></p> <p>Modified reset value in <i>Section 10.9.2: Port x pin input register (Px_IDR) on page 121</i>, in <i>Section 10.9.4: Port x control register 1 (Px_CR1) on page 122</i> and in <i>Table 26: GPIO register map on page 123</i></p> <p>Modified <i>Section 11.2.6: DAC routing on page 130</i></p> <p>Modified <i>Section 14.3.3: ADC on-off control on page 198</i></p> <p>Modified <i>Section 14.3.4: Single conversion mode on page 198</i></p> <p>Modified <i>Section 14.3.16: Temperature sensor on page 204</i> and <i>Section 14.3.17: Internal reference voltage conversion on page 205</i></p> <p>Modified <i>Figure 50: Comparator block diagram on page 239</i></p> <p>Updated <i>Figure 50: Comparator block diagram on page 239</i></p> <p>Modified <i>Figure 51: COMP1 interconnections on page 241</i> and <i>Figure 52: COMP2 interconnections on page 242</i></p> <p>Added note below <i>Figure 51: COMP1 interconnections on page 241</i></p> <p>Updated <i>Figure 54: Configuring the comparators in window mode on page 243</i></p> <p>Added note below <i>Figure 54: Configuring the comparators in window mode on page 243</i></p> <p>Added note in <i>Section 30.3.2: Configuring the SPI in slave mode</i></p> <p>Added note to bit 5 in <i>Section 23.3.2: BEEP control/status register 2 (BEEP_CSR2) on page 408</i></p> <p>Modified RTC section, including: <i>Section 24.2: RTC main features</i>, <i>Section 24.3.1: Clock and prescalers</i>, <i>Section 24.3.4: Periodic auto-wakeup</i>, <i>Section 24.3.7: Resetting the RTC</i>, <i>Section 24.3.9: RTC smooth digital calibration (low, medium+ and high density devices only)</i> and <i>Section 24.6: RTC registers</i></p> <p>Modified <i>Section 17.6.5: Port mask registers (LCD_PM) on page 275</i></p> <p>Modified <i>Figure 107: Center-aligned PWM waveforms (ARR = 8) on page 321</i></p> <p>Modified <i>Section 24.2: RTC main features</i>, <i>Section 24.3.9: RTC smooth digital calibration (low, medium+ and high density devices only)</i> and <i>Section 24.6.11: Control register 3 (RTC_CR3) on page 428</i></p> <p>Modified <i>Figure 143: I2C block diagram on page 464</i>, <i>Figure 135: Transfer sequence diagram for slave transmitter on page 468</i> and <i>Figure 145: Transfer sequence diagram for slave receiver on page 472</i></p> <p>Modified <i>Section 28.4.2: I<sup>2</sup>C master mode on page 472</i></p>

**Table 108. Document revision history (continued)**

Date	Revision	Changes
26-Jul-2010	5 (cont.)	<p>...</p> <p>PO bit description changed in <a href="#">Section 28.7.2: Control register 2 (I2C_CR2) on page 488</a></p> <p>Modified note 8 in <a href="#">Section 28.7.8: Status register 1 (I2C_SR1) on page 492</a>, <a href="#">Section 28.7.12: Clock control register low (I2C_CCRL) on page 498</a> and <a href="#">Section 28.7.13: Clock control register high (I2C_CCRH) on page 499</a></p> <p>Modified <a href="#">Section 30.3.6: CRC calculation</a> (added one paragraph)</p>
10-Sep-2010	6	<p>Added information on STM8L15x medium+ and high density devices:</p> <p>Modified <a href="#">Section 2: Boot ROM</a> (peripheral references)</p> <p>Modified <a href="#">Section 3: Flash program memory and data EEPROM on page 37</a></p> <p><a href="#">Section : WFE control register 4 (WFE_CR4) on page 79</a> added</p> <p><a href="#">Section 13.3.4: DMA1 request mapping on page 182</a> modified</p> <p>Added <a href="#">Section 9.8.2: Clock security system on LSE on page 97</a></p> <p><a href="#">Section 9.14.6: Peripheral clock gating register 3 (CLK_PCKENR3) on page 106</a> added</p> <p>Modified <a href="#">Table 27: I/O groups and selection (medium, medium+ and high density devices) on page 132</a> and <a href="#">Table 28: TIM1 input capture routing on page 129</a></p> <p>Modified <a href="#">Section 11.2.6: DAC routing on page 130</a></p> <p>Modified <a href="#">Section 11.4.14: Analog switch register 2 (RI_ASCR2) on page 141</a></p> <p><a href="#">Section 11.5.1: SYSCFG remap control register 1 (SYSCFG_RMPCR1) on page 150</a>, <a href="#">Section 11.5.2</a> modified and <a href="#">Section 11.5.3: SYSCFG remap control register 3 (SYSCFG_RMPCR3) modified</a></p> <p>Features added in <a href="#">Section 24: Real-time clock (RTC) on page 410</a>: calendar and alarm subseconds, digital calibration, tamper, and AFO_CALIB (1Hz), subsecond Shift.</p> <p>Added <a href="#">Section 12.9.7: External interrupt status register 1 (EXTI_SR1) on page 167</a></p> <p>Modified <a href="#">Section 12.9.7: External interrupt status register 1 (EXTI_SR1) on page 167</a></p> <p>Added <a href="#">Section 12.9.7: External interrupt status register 1 (EXTI_SR1)</a></p> <p>Added <a href="#">Section : AES DMA requests on page 184</a></p> <p>Number of channels changed (28 instead of 25) in <a href="#">Section 14: Analog-to-digital converter (ADC) on page 196</a></p> <p><a href="#">Section 28.7.5: Own address register MSB (I2C_OAR1H) on page 491</a> added DUALF bit added in <a href="#">Section 28.7.10: Status register 3 (I2C_SR3) on page 495</a> register</p> <p>TIM5 added in Timer sections</p> <p>Added <a href="#">Section 9.8.2: Clock security system on LSE on page 97</a></p> <p>Modified RTC section, including <a href="#">Section : Programming the alarm</a></p> <p>Modified <a href="#">Section 30.3.6: CRC calculation</a> (added one paragraph)</p> <p>Added <a href="#">Section 27: AES hardware accelerator (AES) on page 454</a>.</p> <p>Modified <a href="#">Section 6.2: Power-on reset (POR)/power-down reset (PDR) on page 65</a>, <a href="#">Section 6.3: Brownout reset (BOR) on page 66</a></p> <p>Modified reset values (power-on reset values) in <a href="#">Section 24.6: RTC registers on page 423</a></p> <p>Modified <a href="#">Section 7.8: Halt mode on page 82</a>.</p> <p>Modified <a href="#">Section 17.2: LCD controller main features on page 251</a></p> <p>Modified <a href="#">Section 17.4: LCD controller low power modes on page 270</a></p> <p>Modified CC bits description in <a href="#">Section 17.6.2: Control register 2 (LCD_CR2) on page 273</a>.</p>

**Table 108. Document revision history (continued)**

Date	Revision	Changes
10-Sep-2010	6 (cont.)	<p>Modified <a href="#">Section 26.8.1: Control register (WWDG_CR) on page 452</a>  Modified RATIO bit description in <a href="#">Section Note:: This register is write protected. The write access procedure is described in Section : RTC register write protection on page 416</a> on page 426</p>
12-May-2011	7	<p>Modified <a href="#">Figure 7: UBC area size definition for low density devices on page 44</a>, <a href="#">Figure 9: UBC area size definition for medium + STM8L05xx/15xx devices on page 45</a> and <a href="#">Figure 10: UBC area size definition for high density devices on page 46</a></p> <p>Modified <a href="#">Section 6.2: Power-on reset (POR)/power-down reset (PDR) on page 65</a></p> <p>Added <a href="#">Section 6.5: Internal voltage reference (VREFINT) on page 68</a>.</p> <p>Modified note 1 below <a href="#">Table 11</a> and modified <a href="#">Table 11: Internal voltage reference status during Halt/Active-halt mode on page 68</a></p> <p>Modified note concerning the RTC peripheral and LSI/LSE clocks in <a href="#">Section 9.8.2: Clock security system on LSE on page 97</a>.</p> <p>Modified SWITCHF, SWITCHEN and CSSEN bit description in <a href="#">Section 9.8.3: CSS on LSE control and status register (CSSLSE_CSR) on page 98</a>.</p> <p>Modified LSERDY bit description in <a href="#">Section 9.14.8: External clock register (CLK_ECKCR) on page 108</a></p> <p>Modified note below <a href="#">Figure 22: GPIO block diagram on page 117</a> and added note below <a href="#">Table 24: I/O port configuration summary on page 118</a></p> <p>Modified <a href="#">Figure 24: Routing interface (RI) block diagram (low density devices) on page 126</a></p> <p>Modified <a href="#">Section 11.2.2: I/O groups on page 126</a></p> <p>Added note to <a href="#">Section 14.3.15: Schmitt trigger disabling on page 204</a></p> <p>Modified <a href="#">Figure 50: Comparator block diagram on page 239</a></p> <p>Modified STE bit description in <a href="#">Comparator control and status register 1 (COMP_CSR1) on page 245</a></p> <p>Modified INSEL and OUTSEL bit description in <a href="#">Section 16.8.3: Comparator control and status register 3 (COMP_CSR3) on page 247</a></p> <p>Modified <a href="#">Comparator control and status register 4 (COMP_CSR4) on page 248</a> and <a href="#">Comparator control and status register 5 (COMP_CSR5) on page 248</a></p> <p>Modified <a href="#">Figure 64: Resistive network on page 268</a> and <a href="#">Section 17.3.8: Generation of LCD voltage levels on page 267</a>.</p> <p>Modified <a href="#">Section 24: Real-time clock (RTC) on page 410</a>.</p> <p>Modified <a href="#">Section 24.3.1: Clock and prescalers on page 412</a></p> <p>Modified <a href="#">Section 24.3.6: Reading the calendar on page 417</a></p> <p>Added note 2 in <a href="#">Section : Calendar initialization and configuration on page 416</a></p> <p>Modified <a href="#">Section 24.6.29: Calibration register high (RTC_CALRH) on page 439</a> and <a href="#">Section 24.6.30: Calibration register low (RTC_CALRL) on page 440</a></p> <p>Added note below <a href="#">Figure 145: Transfer sequence diagram for slave receiver on page 472</a>.</p> <p>Modified <a href="#">Section : Closing slave communication on page 472</a></p> <p>Modified BERR bit description in <a href="#">Section 28.7.9: Status register 2 (I2C_SR2) on page 494</a></p> <p>Modified <a href="#">Section 29.6.1: Status register (USART_SR) on page 531</a> (TC status is "rc_w0" instead of "r")</p> <p>Modified <a href="#">Figure 155: TC/TXE behavior when transmitting on page 510</a></p> <p>Added Step 6 to <a href="#">Section : Transmission using DMA on page 527</a></p> <p>Modified <a href="#">Figure 169: Transmission using DMA on page 528</a> and <a href="#">Figure 170: Reception using DMA on page 529</a>.</p>

**Table 108. Document revision history (continued)**

Date	Revision	Changes
06-Jun-2011	7 (cont.)	<p>Modified ADDR2 bits (bits 7:1) in <a href="#">28.7.6: Own address register 2 (I2C_OAR2) on page 491</a> and in <a href="#">Table 94: I<sup>2</sup>C register map on page 501</a>.</p> <p>Modified <a href="#">Section 30.3.1: General description on page 542</a></p> <p>Added one note in <a href="#">Section 24.6.12: Initialization and status register 1 (RTC_ISR1) on page 429</a></p> <p>Modified <a href="#">Table 48: Interrupt requests on page 205</a></p> <p>Modified <a href="#">Section 20.5.2: Control register 2 (TIMx_CR2) on page 376</a></p> <p>Modified <a href="#">Section 17.1.1: Definitions on page 250</a>, <a href="#">Section 17.2: LCD controller main features on page 251</a>, <a href="#">Section 17.3.2: Frequency generator on page 254</a>, <a href="#">Section 17.3.3: Common driver on page 257</a>, <a href="#">Section 17.3.4: Segment driver on page 265</a>, <a href="#">Section 17.3.6: Blink on page 266</a>, <a href="#">Section 17.3.7: Multiplexing COM[7:4] and SEG[43:40], SEG[39:36], or SEG[31:28] on page 266</a>, <a href="#">Section 17.3.8: Generation of LCD voltage levels on page 267</a>, and <a href="#">Section 17.4: LCD controller low power modes on page 270</a>.</p> <p>Modified <a href="#">Figure 64: Resistive network on page 268</a>.</p> <p>Modified <a href="#">Section 24.3.9: RTC smooth digital calibration (low, medium+ and high density devices only) on page 419</a> and <a href="#">Table 80: RTC register map and reset values on page 442</a>.</p> <p>Modified <a href="#">Section 24.6.21: RTC shift control register low (RTC_SHIFTRL) on page 435</a>.</p> <p>Modified URS bit description in <a href="#">Section 20.5.1: Control register 1 (TIMx_CR1) on page 375</a>.</p>
14-Jun-2011	8	<p>Added information on low density devices</p> <p>Removed references to TLI</p>
02-Jul-2012	9	<p>Updated for STM8L05x devices</p> <p>Replaced BALERT by SMB in <a href="#">Figure 143: I<sup>2</sup>C block diagram on page 469</a></p> <p>Modified title in <a href="#">Table 99</a> and <a href="#">Table 100</a></p> <p>Corrected note ‘can be 44, 40 or 28’ in <a href="#">Section 17.3.7: Multiplexing COM[7:4] and SEG[43:40], SEG[39:36], or SEG[31:28] on page 266</a></p> <p>Added <a href="#">Section 17.3.9: LCD buffer update on page 270</a></p> <p>Added sentence “If internal VLCD booster...” in <a href="#">Table 62: LCD behavior in low power modes</a></p> <p>Corrected TIM3_CC1 typo in <a href="#">Table 38: DMA1 channel request mapping on page 182</a></p> <p>Modified “When the I/Os are switched to analog mode...” paragraph in <a href="#">Section 11.2.1: RI functional description on page 126</a></p> <p>Modified <a href="#">Section 11.2.4: TIM2 &amp; TIM3 routing on page 129</a></p> <p>Replaced TSSUR with THALT in <a href="#">Figure 25: TIM2 and TIM3 interconnections</a></p> <p>Modified bit values in <a href="#">Section 11.4.23: I/O switch register 4 (RI_IOSR4) on page 147</a></p> <p>Replaced note for bit 1 CC1IF in <a href="#">Section 19.8.7: Status register 1 (TIM1_SR1) on page 344</a></p> <p>Added sentence “The DMA must be initialized...” in <a href="#">Section 28.4.6: DMA requests on page 483</a></p> <p>Removed all “for ordering information..” sentences and added the sentence at the bottom of the page in <a href="#">Introduction</a></p>

**Table 108. Document revision history (continued)**

Date	Revision	Changes
25-Oct-2012	10	<p>Updated for STM8AL31xx and STM8L3Lxx devices.</p> <p>Modified PVDE description in <a href="#">Section 6.7.1: Power control and status register 1 (PWR_CSR1) on page 70</a></p> <p>Added note in <a href="#">Section 15.3.1: DAC channel x enable on page 219</a></p> <p>Modified description in LCD <a href="#">Dead time on page 269</a></p> <p>Added sentence about Schmitt trigger in <a href="#">Section 24.3.10: Tamper detection (low, medium+ and high density devices only) on page 421</a></p> <p>Modified reset value of <a href="#">Section 24.6.13: Initialization and status register 2 (RTC_ISR2) on page 430</a></p>

**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 TWO AUTHORIZED ST REPRESENTATIVES, 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.

© 2012 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 - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

[www.st.com](http://www.st.com)

