

18.4.5	RTC counter register (RTC_CNTH / RTC_CNTL)	491
18.4.6	RTC alarm register high (RTC_ALRH / RTC_ALRL)	492
18.4.7	RTC register map	493
19	Independent watchdog (IWDG)	494
19.1	IWDG introduction	494
19.2	IWDG main features	494
19.3	IWDG functional description	494
19.3.1	Hardware watchdog	495
19.3.2	Register access protection	495
19.3.3	Debug mode	495
19.4	IWDG registers	496
19.4.1	Key register (IWDG_KR)	496
19.4.2	Prescaler register (IWDG_PR)	496
19.4.3	Reload register (IWDG_RLR)	497
19.4.4	Status register (IWDG_SR)	497
19.4.5	IWDG register map	499
20	Window watchdog (WWDG)	500
20.1	WWDG introduction	500
20.2	WWDG main features	500
20.3	WWDG functional description	500
20.4	How to program the watchdog timeout	502
20.5	Debug mode	503
20.6	WWDG registers	504
20.6.1	Control register (WWDG_CR)	504
20.6.2	Configuration register (WWDG_CFR)	505
20.6.3	Status register (WWDG_SR)	505
20.6.4	WWDG register map	506
21	Flexible static memory controller (FSMC)	507
21.1	FSMC main features	507
21.2	Block diagram	508
21.3	AHB interface	509
21.3.1	Supported memories and transactions	510
21.4	External device address mapping	511

21.4.1	NOR/PSRAM address mapping	511
21.4.2	NAND/PC Card address mapping	512
21.5	NOR Flash/PSRAM controller	513
21.5.1	External memory interface signals	514
21.5.2	Supported memories and transactions	515
21.5.3	General timing rules	517
21.5.4	NOR Flash/PSRAM controller asynchronous transactions	517
21.5.5	Synchronous transactions	535
21.5.6	NOR/PSRAM control registers	541
21.6	NAND Flash/PC Card controller	548
21.6.1	External memory interface signals	549
21.6.2	NAND Flash / PC Card supported memories and transactions	551
21.6.3	Timing diagrams for NAND and PC Card	551
21.6.4	NAND Flash operations	552
21.6.5	NAND Flash prewait functionality	553
21.6.6	Computation of the error correction code (ECC) in NAND Flash memory	554
21.6.7	PC Card/CompactFlash operations	555
21.6.8	NAND Flash/PC Card control registers	557
21.6.9	FSMC register map	564
22	Secure digital input/output interface (SDIO)	566
22.1	SDIO main features	566
22.2	SDIO bus topology	567
22.3	SDIO functional description	569
22.3.1	SDIO adapter	570
22.3.2	SDIO AHB interface	580
22.4	Card functional description	581
22.4.1	Card identification mode	581
22.4.2	Card reset	581
22.4.3	Operating voltage range validation	582
22.4.4	Card identification process	582
22.4.5	Block write	583
22.4.6	Block read	584
22.4.7	Stream access, stream write and stream read (MultiMediaCard only)	584
22.4.8	Erase: group erase and sector erase	585
22.4.9	Wide bus selection or deselection	586

22.4.10	Protection management	586
22.4.11	Card status register	589
22.4.12	SD status register	592
22.4.13	SD I/O mode	596
22.4.14	Commands and responses	597
22.5	Response formats	601
22.5.1	R1 (normal response command)	601
22.5.2	R1b	601
22.5.3	R2 (CID, CSD register)	601
22.5.4	R3 (OCR register)	602
22.5.5	R4 (Fast I/O)	602
22.5.6	R4b	602
22.5.7	R5 (interrupt request)	603
22.5.8	R6	604
22.6	SDIO I/O card-specific operations	604
22.6.1	SDIO I/O read wait operation by SDIO_D2 signalling	604
22.6.2	SDIO read wait operation by stopping SDIO_CK	605
22.6.3	SDIO suspend/resume operation	605
22.6.4	SDIO interrupts	605
22.7	CE-ATA specific operations	605
22.7.1	Command completion signal disable	605
22.7.2	Command completion signal enable	606
22.7.3	CE-ATA interrupt	606
22.7.4	Aborting CMD61	606
22.8	HW flow control	606
22.9	SDIO registers	606
22.9.1	SDIO power control register (SDIO_POWER)	607
22.9.2	SDI clock control register (SDIO_CLKCR)	607
22.9.3	SDIO argument register (SDIO_ARG)	608
22.9.4	SDIO command register (SDIO_CMD)	609
22.9.5	SDIO command response register (SDIO_RESPCMD)	610
22.9.6	SDIO response 1..4 register (SDIO_RESPx)	610
22.9.7	SDIO data timer register (SDIO_DTIMER)	611
22.9.8	SDIO data length register (SDIO_DLEN)	611
22.9.9	SDIO data control register (SDIO_DCTRL)	612
22.9.10	SDIO data counter register (SDIO_DCOUNT)	613

22.9.11	SDIO status register (SDIO_STA)	614
22.9.12	SDIO interrupt clear register (SDIO_ICR)	615
22.9.13	SDIO mask register (SDIO_MASK)	617
22.9.14	SDIO FIFO counter register (SDIO_FIFOCNT)	619
22.9.15	SDIO data FIFO register (SDIO_FIFO)	620
22.9.16	SDIO register map	621
23	Universal serial bus full-speed device interface (USB)	622
23.1	USB introduction	622
23.2	USB main features	622
23.3	USB functional description	622
23.3.1	Description of USB blocks	624
23.4	Programming considerations	625
23.4.1	Generic USB device programming	625
23.4.2	System and power-on reset	626
23.4.3	Double-buffered endpoints	631
23.4.4	Isochronous transfers	634
23.4.5	Suspend/Resume events	635
23.5	USB registers	637
23.5.1	Common registers	637
23.5.2	Endpoint-specific registers	644
23.5.3	Buffer descriptor table	648
23.5.4	USB register map	651
24	Controller area network (bxCAN)	653
24.1	bxCAN introduction	653
24.2	bxCAN main features	653
24.3	bxCAN general description	654
24.3.1	CAN 2.0B active core	655
24.3.2	Control, status and configuration registers	655
24.3.3	Tx mailboxes	655
24.3.4	Acceptance filters	655
24.4	bxCAN operating modes	656
24.4.1	Initialization mode	657
24.4.2	Normal mode	657
24.4.3	Sleep mode (low-power)	657

24.5	Test mode	658
24.5.1	Silent mode	658
24.5.2	Loop back mode	659
24.5.3	Loop back combined with silent mode	659
24.6	Debug mode	660
24.7	bxCAN functional description	660
24.7.1	Transmission handling	660
24.7.2	Time triggered communication mode	662
24.7.3	Reception handling	662
24.7.4	Identifier filtering	664
24.7.5	Message storage	668
24.7.6	Error management	670
24.7.7	Bit timing	670
24.8	bxCAN interrupts	672
24.9	CAN registers	674
24.9.1	Register access protection	674
24.9.2	CAN control and status registers	674
24.9.3	CAN mailbox registers	684
24.9.4	CAN filter registers	691
24.9.5	bxCAN register map	695
25	Serial peripheral interface (SPI)	699
25.1	SPI introduction	699
25.2	SPI and I ² S main features	700
25.2.1	SPI features	700
25.2.2	I ² S features	701
25.3	SPI functional description	702
25.3.1	General description	702
25.3.2	Configuring the SPI in slave mode	706
25.3.3	Configuring the SPI in master mode	707
25.3.4	Configuring the SPI for half-duplex communication	707
25.3.5	Data transmission and reception procedures	708
25.3.6	CRC calculation	715
25.3.7	Status flags	717
25.3.8	Disabling the SPI	718
25.3.9	SPI communication using DMA (direct memory addressing)	719

25.3.10	Error flags	721
25.3.11	SPI interrupts	722
25.4	I ² S functional description	723
25.4.1	I ² S general description	723
25.4.2	Supported audio protocols	724
25.4.3	Clock generator	731
25.4.4	I ² S master mode	736
25.4.5	I ² S slave mode	737
25.4.6	Status flags	739
25.4.7	Error flags	740
25.4.8	I ² S interrupts	740
25.4.9	DMA features	741
25.5	SPI and I ² S registers	742
25.5.1	SPI control register 1 (SPI_CR1) (not used in I ² S mode)	742
25.5.2	SPI control register 2 (SPI_CR2)	744
25.5.3	SPI status register (SPI_SR)	745
25.5.4	SPI data register (SPI_DR)	746
25.5.5	SPI CRC polynomial register (SPI_CRCPR) (not used in I ² S mode)	746
25.5.6	SPI RX CRC register (SPI_RXCRCR) (not used in I ² S mode)	747
25.5.7	SPI TX CRC register (SPI_TXCRCR) (not used in I ² S mode)	748
25.5.8	SPI_I ² S configuration register (SPI_I2SCFGR)	748
25.5.9	SPI_I ² S prescaler register (SPI_I2SPR)	750
25.5.10	SPI register map	751
26	Inter-integrated circuit (I2C) interface	752
26.1	I ² C introduction	752
26.2	I ² C main features	752
26.3	I ² C functional description	753
26.3.1	Mode selection	753
26.3.2	I ² C slave mode	755
26.3.3	I ² C master mode	757
26.3.4	Error conditions	764
26.3.5	SDA/SCL line control	765
26.3.6	SMBus	766
26.3.7	DMA requests	768
26.3.8	Packet error checking	770

26.4	I ² C interrupts	770
26.5	I ² C debug mode	772
26.6	I ² C registers	772
26.6.1	I ² C Control register 1 (I2C_CR1)	772
26.6.2	I ² C Control register 2 (I2C_CR2)	774
26.6.3	I ² C Own address register 1 (I2C_OAR1)	776
26.6.4	I ² C Own address register 2 (I2C_OAR2)	776
26.6.5	I ² C Data register (I2C_DR)	777
26.6.6	I ² C Status register 1 (I2C_SR1)	777
26.6.7	I ² C Status register 2 (I2C_SR2)	780
26.6.8	I ² C Clock control register (I2C_CCR)	781
26.6.9	I ² C TRISE register (I2C_TRISE)	782
26.6.10	I ² C register map	784
27	Universal synchronous asynchronous receiver transmitter (USART)	785
27.1	USART introduction	785
27.2	USART main features	786
27.3	USART functional description	787
27.3.1	USART character description	790
27.3.2	Transmitter	791
27.3.3	Receiver	794
27.3.4	Fractional baud rate generation	798
27.3.5	USART receiver's tolerance to clock deviation	800
27.3.6	Multiprocessor communication	800
27.3.7	Parity control	802
27.3.8	LIN (local interconnection network) mode	803
27.3.9	USART synchronous mode	805
27.3.10	Single-wire half-duplex communication	807
27.3.11	Smartcard	808
27.3.12	IrDA SIR ENDEC block	810
27.3.13	Continuous communication using DMA	812
27.3.14	Hardware flow control	815
27.4	USART interrupts	816
27.5	USART mode configuration	817
27.6	USART registers	817

27.6.1	Status register (USART_SR)	818
27.6.2	Data register (USART_DR)	820
27.6.3	Baud rate register (USART_BRR)	820
27.6.4	Control register 1 (USART_CR1)	821
27.6.5	Control register 2 (USART_CR2)	823
27.6.6	Control register 3 (USART_CR3)	824
27.6.7	Guard time and prescaler register (USART_GTPR)	826
27.6.8	USART register map	827
28	USB on-the-go full-speed (OTG_FS)	828
28.1	OTG_FS introduction	828
28.2	OTG_FS main features	829
28.2.1	General features	829
28.2.2	Host-mode features	830
28.2.3	Peripheral-mode features	830
28.3	OTG_FS functional description	831
28.3.1	OTG pins	831
28.3.2	OTG full-speed core	831
28.3.3	Full-speed OTG PHY	832
28.4	OTG dual role device (DRD)	833
28.4.1	ID line detection	833
28.4.2	HNP dual role device	833
28.4.3	SRP dual role device	834
28.5	USB peripheral	834
28.5.1	SRP-capable peripheral	835
28.5.2	Peripheral states	835
28.5.3	Peripheral endpoints	836
28.6	USB host	838
28.6.1	SRP-capable host	839
28.6.2	USB host states	839
28.6.3	Host channels	841
28.6.4	Host scheduler	842
28.7	SOF trigger	843
28.7.1	Host SOFs	843
28.7.2	Peripheral SOFs	843
28.8	OTG low-power modes	844

28.9	Dynamic update of the OTG_FS_HFIR register	845
28.10	USB data FIFOs	846
28.11	Peripheral FIFO architecture	846
28.11.1	Peripheral Rx FIFO	846
28.11.2	Peripheral Tx FIFOs	847
28.12	Host FIFO architecture	847
28.12.1	Host Rx FIFO	847
28.12.2	Host Tx FIFOs	848
28.13	FIFO RAM allocation	848
28.13.1	Device mode	848
28.13.2	Host mode	849
28.14	USB system performance	849
28.15	OTG_FS interrupts	850
28.16	OTG_FS control and status registers	852
28.16.1	CSR memory map	853
28.16.2	OTG_FS global registers	858
28.16.3	Host-mode registers	878
28.16.4	Device-mode registers	889
28.16.5	OTG_FS power and clock gating control register (OTG_FS_PCGCCTL)	912
28.16.6	OTG_FS register map	913
28.17	OTG_FS programming model	922
28.17.1	Core initialization	922
28.17.2	Host initialization	923
28.17.3	Device initialization	923
28.17.4	Host programming model	924
28.17.5	Device programming model	940
28.17.6	Operational model	942
28.17.7	Worst case response time	960
28.17.8	OTG programming model	961
29	Ethernet (ETH): media access control (MAC) with DMA controller	968
29.1	Ethernet introduction	968
29.2	Ethernet main features	968
29.2.1	MAC core features	969

29.2.2	DMA features	970
29.2.3	PTP features	970
29.3	Ethernet pins	971
29.4	Ethernet functional description: SMI, MII and RMII	972
29.4.1	Station management interface: SMI	972
29.4.2	Media-independent interface: MII	975
29.4.3	Reduced media-independent interface: RMII	977
29.4.4	MII/RMII selection	979
29.5	Ethernet functional description: MAC 802.3	979
29.5.1	MAC 802.3 frame format	980
29.5.2	MAC frame transmission	984
29.5.3	MAC frame reception	991
29.5.4	MAC interrupts	996
29.5.5	MAC filtering	997
29.5.6	MAC loopback mode	1000
29.5.7	MAC management counters: MMC	1000
29.5.8	Power management: PMT	1001
29.5.9	Precision time protocol (IEEE1588 PTP)	1004
29.6	Ethernet functional description: DMA controller operation	1010
29.6.1	Initialization of a transfer using DMA	1011
29.6.2	Host bus burst access	1011
29.6.3	Host data buffer alignment	1012
29.6.4	Buffer size calculations	1012
29.6.5	DMA arbiter	1013
29.6.6	Error response to DMA	1013
29.6.7	Tx DMA configuration	1013
29.6.8	Rx DMA configuration	1022
29.6.9	DMA interrupts	1030
29.7	Ethernet interrupts	1031
29.8	Ethernet register descriptions	1032
29.8.1	MAC register description	1032
29.8.2	MMC register description	1049
29.8.3	IEEE 1588 time stamp registers	1054
29.8.4	DMA register description	1058
29.8.5	Ethernet register maps	1071

30	Device electronic signature	1076
30.1	Memory size registers	1076
30.1.1	Flash size register	1076
30.2	Unique device ID register (96 bits)	1077
31	Debug support (DBG)	1079
31.1	Overview	1079
31.2	Reference Arm® documentation	1081
31.3	SWJ debug port (serial wire and JTAG)	1081
31.3.1	Mechanism to select the JTAG-DP or the SW-DP	1082
31.4	Pinout and debug port pins	1082
31.4.1	SWJ debug port pins	1082
31.4.2	Flexible SWJ-DP pin assignment	1082
31.4.3	Internal pull-up and pull-down on JTAG pins	1083
31.4.4	Using serial wire and releasing the unused debug pins as GPIOs ..	1085
31.5	STM32F10xxx JTAG TAP connection	1085
31.6	ID codes and locking mechanism	1087
31.6.1	MCU device ID code	1087
31.6.2	Boundary scan TAP	1088
31.6.3	Cortex®-M3 TAP	1088
31.6.4	Cortex®-M3 JEDEC-106 ID code	1088
31.7	JTAG debug port	1088
31.8	SW debug port	1090
31.8.1	SW protocol introduction	1090
31.8.2	SW protocol sequence	1090
31.8.3	SW-DP state machine (reset, idle states, ID code)	1091
31.8.4	DP and AP read/write accesses	1092
31.8.5	SW-DP registers	1092
31.8.6	SW-AP registers	1093
31.9	AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP	1093
31.10	Core debug	1095
31.11	Capability of the debugger host to connect under system reset	1096
31.12	FPB (Flash patch breakpoint)	1096
31.13	DWT (data watchpoint trigger)	1097

31.14	ITM (instrumentation trace macrocell)	1097
31.14.1	General description	1097
31.14.2	Time stamp packets, synchronization and overflow packets	1097
31.15	ETM (Embedded Trace Macrocell™)	1099
31.15.1	ETM general description	1099
31.15.2	ETM signal protocol and packet types	1099
31.15.3	Main ETM registers	1100
31.15.4	ETM configuration example	1100
31.16	MCU debug component (DBGMCU)	1100
31.16.1	Debug support for low-power modes	1100
31.16.2	Debug support for timers, watchdog, bxCAN and I ² C	1101
31.16.3	Debug MCU configuration register	1101
31.17	TPIU (trace port interface unit)	1103
31.17.1	Introduction	1103
31.17.2	TRACE pin assignment	1105
31.17.3	TPUI formatter	1106
31.17.4	TPUI frame synchronization packets	1107
31.17.5	Transmission of the synchronization frame packet	1107
31.17.6	Synchronous mode	1107
31.17.7	Asynchronous mode	1108
31.17.8	TRACECLKIN connection inside the STM32F10xxx	1108
31.17.9	TPIU registers	1108
31.17.10	Example of configuration	1109
31.18	DBG register map	1110
32	Revision history	1111

List of tables

Table 1.	Sections related to each STM32F10xxx product	40
Table 2.	Sections related to each peripheral	42
Table 3.	Register boundary addresses	50
Table 4.	Flash module organization (low-density devices)	54
Table 5.	Flash module organization (medium-density devices)	55
Table 6.	Flash module organization (high-density devices)	56
Table 7.	Flash module organization (connectivity line devices)	56
Table 8.	XL-density Flash module organization	57
Table 9.	Boot modes	60
Table 10.	CRC calculation unit register map and reset values	65
Table 11.	Low-power mode summary	72
Table 12.	Sleep-now	74
Table 13.	Sleep-on-exit	74
Table 14.	Stop mode	75
Table 15.	Standby mode	76
Table 16.	PWR register map and reset values	80
Table 17.	BKP register map and reset values	85
Table 18.	RCC register map and reset values	121
Table 19.	RCC register map and reset values	156
Table 20.	Port bit configuration table	161
Table 21.	Output MODE bits	161
Table 22.	Advanced timers TIM1 and TIM8	166
Table 23.	General-purpose timers TIM2/3/4/5	166
Table 24.	USARTs	166
Table 25.	SPI	167
Table 26.	I2S	167
Table 27.	I2C	168
Table 28.	bxCAN	168
Table 29.	USB	168
Table 30.	OTG_FS pin configuration	168
Table 31.	SDIO	169
Table 32.	FSMC	169
Table 33.	Other IOs	170
Table 34.	CAN1 alternate function remapping	176
Table 35.	CAN2 alternate function remapping	176
Table 36.	Debug interface signals	176
Table 37.	Debug port mapping	177
Table 38.	ADC1 external trigger injected conversion alternate function remapping	177
Table 39.	ADC1 external trigger regular conversion alternate function remapping	177
Table 40.	ADC2 external trigger injected conversion alternate function remapping	177
Table 41.	ADC2 external trigger regular conversion alternate function remapping	178
Table 42.	TIM5 alternate function remapping	178
Table 43.	TIM4 alternate function remapping	178
Table 44.	TIM3 alternate function remapping	178
Table 45.	TIM2 alternate function remapping	179
Table 46.	TIM1 alternate function remapping	179
Table 47.	TIM9 remapping	179
Table 48.	TIM10 remapping	179

Table 49.	TIM11 remapping	180
Table 50.	TIM13 remapping	180
Table 51.	TIM14 remapping	180
Table 52.	USART3 remapping	180
Table 53.	USART2 remapping	180
Table 54.	USART1 remapping	181
Table 55.	I2C1 remapping	181
Table 56.	SPI1 remapping	181
Table 57.	SPI3/I2S3 remapping	181
Table 58.	ETH remapping	182
Table 59.	GPIO register map and reset values	194
Table 60.	AFIO register map and reset values	195
Table 61.	Vector table for connectivity line devices	198
Table 62.	Vector table for XL-density devices	201
Table 63.	Vector table for other STM32F10xxx devices	204
Table 64.	External interrupt/event controller register map and reset values	214
Table 65.	ADC pins	218
Table 66.	Analog watchdog channel selection	220
Table 67.	External trigger for regular channels for ADC1 and ADC2	225
Table 68.	External trigger for injected channels for ADC1 and ADC2	226
Table 69.	External trigger for regular channels for ADC3	226
Table 70.	External trigger for injected channels for ADC3	226
Table 71.	ADC interrupts	236
Table 72.	ADC register map and reset values	252
Table 73.	DAC pins	255
Table 74.	External triggers	258
Table 75.	DAC register map	273
Table 76.	Programmable data width and endian behavior (when bits PINC = MINC = 1)	279
Table 77.	DMA interrupt requests	280
Table 78.	Summary of DMA1 requests for each channel	282
Table 79.	Summary of DMA2 requests for each channel	283
Table 80.	DMA register map and reset values	289
Table 81.	Counting direction versus encoder signals	330
Table 82.	TIMx Internal trigger connection	344
Table 83.	Output control bits for complementary OC _x and OC _{xN} channels with break feature	355
Table 84.	TIM1 and TIM8 register map and reset values	363
Table 85.	Counting direction versus encoder signals	393
Table 86.	TIMx Internal trigger connection	409
Table 87.	Output control bit for standard OC _x channels	418
Table 88.	TIMx register map and reset values	423
Table 89.	TIMx internal trigger connection	449
Table 90.	Output control bit for standard OC _x channels	456
Table 91.	TIM9/12 register map and reset values	458
Table 92.	Output control bit for standard OC _x channels	465
Table 93.	TIM10/11/13/14 register map and reset values	468
Table 94.	TIM6 and TIM7 register map and reset values	481
Table 95.	RTC register map and reset values	493
Table 96.	Min/max IWDG timeout period (in ms) at 40 kHz (LSI)	495
Table 97.	IWDG register map and reset values	499
Table 98.	Minimum and maximum timeout values @36 MHz (f _{PCLK1})	503
Table 99.	WWDG register map and reset values	506

Table 100.	NOR/PSRAM bank selection	511
Table 101.	External memory address	512
Table 102.	Memory mapping and timing registers	512
Table 103.	NAND bank selections	513
Table 104.	Programmable NOR/PSRAM access parameters	514
Table 105.	Nonmultiplexed I/O NOR Flash	514
Table 106.	Multiplexed I/O NOR Flash	515
Table 107.	Nonmultiplexed I/Os PSRAM/SRAM	515
Table 108.	NOR Flash/PSRAM controller: example of supported memories and transactions	516
Table 109.	FSMC_BCRx bit fields	519
Table 110.	FSMC_BTRx bit fields	519
Table 111.	FSMC_BCRx bit fields	521
Table 112.	FSMC_BTRx bit fields	521
Table 113.	FSMC_BWTRx bit fields	522
Table 114.	FSMC_BCRx bit fields	524
Table 115.	FSMC_BTRx bit fields	524
Table 116.	FSMC_BWTRx bit fields	525
Table 117.	FSMC_BCRx bit fields	526
Table 118.	FSMC_BTRx bit fields	527
Table 119.	FSMC_BWTRx bit fields	527
Table 120.	FSMC_BCRx bit fields	529
Table 121.	FSMC_BTRx bit fields	529
Table 122.	FSMC_BWTRx bit fields	530
Table 123.	FSMC_BCRx bit fields	531
Table 124.	FSMC_BTRx bit fields	532
Table 125.	FSMC_BCRx bit fields	537
Table 126.	FSMC_BTRx bit fields	538
Table 127.	FSMC_BCRx bit fields	539
Table 128.	FSMC_BTRx bit fields	540
Table 129.	Programmable NAND/PC Card access parameters	549
Table 130.	8-bit NAND Flash	549
Table 131.	16-bit NAND Flash	550
Table 132.	16-bit PC Card	550
Table 133.	Supported memories and transactions	551
Table 134.	16-bit PC-Card signals and access type	556
Table 135.	ECC result relevant bits	563
Table 136.	FSMC register map	564
Table 137.	SDIO I/O definitions	570
Table 138.	Command format	574
Table 139.	Short response format	575
Table 140.	Long response format	575
Table 141.	Command path status flags	575
Table 142.	Data token format	578
Table 143.	Transmit FIFO status flags	579
Table 144.	Receive FIFO status flags	580
Table 145.	Card status	590
Table 146.	SD status	593
Table 147.	Speed class code field	594
Table 148.	Performance move field	594
Table 149.	AU_SIZE field	595
Table 150.	Maximum AU size	595
Table 151.	Erase size field	595

Table 152. Erase timeout field	596
Table 153. Erase offset field	596
Table 154. Block-oriented write commands	598
Table 155. Block-oriented write protection commands	599
Table 156. Erase commands	599
Table 157. I/O mode commands	600
Table 158. Lock card	600
Table 159. Application-specific commands	600
Table 160. R1 response	601
Table 161. R2 response	601
Table 162. R3 response	602
Table 163. R4 response	602
Table 164. R4b response	603
Table 165. R5 response	603
Table 166. R6 response	604
Table 167. Response type and SDIO_RESPx registers	610
Table 168. SDIO register map	621
Table 169. Double-buffering buffer flag definition	632
Table 170. Bulk double-buffering memory buffers usage	633
Table 171. Isochronous memory buffers usage	634
Table 172. Resume event detection	636
Table 173. Reception status encoding	647
Table 174. Endpoint type encoding	647
Table 175. Endpoint kind meaning	647
Table 176. Transmission status encoding	648
Table 177. Definition of allocated buffer memory	651
Table 178. USB register map and reset values	651
Table 179. Transmit mailbox mapping	668
Table 180. Receive mailbox mapping	668
Table 181. bxCAN register map and reset values	695
Table 182. SPI interrupt requests	722
Table 183. Audio-frequency precision using standard 8 MHz HSE (high-density and XL-density devices only)	732
Table 184. Audio-frequency precision using standard 25 MHz and PLL3 (connectivity line devices only)	734
Table 185. Audio-frequency precision using standard 14.7456 MHz and PLL3 (connectivity line devices only)	735
Table 186. I ² S interrupt requests	741
Table 187. SPI register map and reset values	751
Table 188. SMBus vs. I ² C	766
Table 189. I ² C Interrupt requests	770
Table 190. I ² C register map and reset values	784
Table 191. Noise detection from sampled data	797
Table 192. Error calculation for programmed baud rates	799
Table 193. USART receiver tolerance when DIV_Fraction is 0	800
Table 194. USART receiver tolerance when DIV_Fraction is different from 0	800
Table 195. Frame formats	802
Table 196. USART interrupt requests	816
Table 197. USART mode configuration	817
Table 198. USART register map and reset values	827
Table 199. OTG_FS input/output pins	831
Table 200. Compatibility of STM32 low power modes with the OTG	844

Table 201. Core global control and status registers (CSRs)	853
Table 202. Host-mode control and status registers (CSRs)	854
Table 203. Device-mode control and status registers	855
Table 204. Data FIFO (DFIFO) access register map	856
Table 205. Power and clock gating control and status registers	857
Table 206. TRDT values	863
Table 207. Minimum duration for soft disconnect	891
Table 208. OTG_FS register map and reset values	913
Table 209. Ethernet pin configuration	971
Table 210. Management frame format	973
Table 211. Clock range	975
Table 212. TX interface signal encoding	976
Table 213. RX interface signal encoding	977
Table 214. Frame statuses	993
Table 215. Destination address filtering	999
Table 216. Source address filtering	1000
Table 217. Receive descriptor 0	1028
Table 218. Ethernet register map and reset values	1072
Table 219. SWJ debug port pins	1082
Table 220. Flexible SWJ-DP pin assignment	1083
Table 221. JTAG debug port data registers	1089
Table 222. 32-bit debug port registers addressed through the shifted value A[3:2]	1090
Table 223. Packet request (8-bits)	1091
Table 224. ACK response (3 bits)	1091
Table 225. DATA transfer (33 bits)	1091
Table 226. SW-DP registers	1092
Table 227. Cortex®-M3 AHB-AP registers	1094
Table 228. Core debug registers	1095
Table 229. Main ITM registers	1098
Table 230. Main ETM registers	1100
Table 231. Asynchronous TRACE pin assignment	1105
Table 232. Synchronous TRACE pin assignment	1105
Table 233. Flexible TRACE pin assignment	1106
Table 234. Important TPIU registers	1108
Table 235. DBG register map and reset values	1110
Table 236. Document revision history	1111

List of figures

Figure 1.	System architecture (low-, medium-, XL-density devices)	47
Figure 2.	System architecture in connectivity line devices	48
Figure 3.	CRC calculation unit block diagram	64
Figure 4.	Power supply overview	68
Figure 5.	Power on reset/power down reset waveform	70
Figure 6.	PVD thresholds	71
Figure 7.	Simplified diagram of the reset circuit	91
Figure 8.	Clock tree	93
Figure 9.	HSE/ LSE clock sources	94
Figure 10.	Simplified diagram of the reset circuit	124
Figure 11.	Clock tree	126
Figure 12.	HSE/ LSE clock sources	128
Figure 13.	Basic structure of a standard I/O port bit	160
Figure 14.	Basic structure of a 5-Volt tolerant I/O port bit	160
Figure 15.	Input floating/pull up/pull down configurations	163
Figure 16.	Output configuration	164
Figure 17.	Alternate function configuration	165
Figure 18.	High impedance-analog configuration	166
Figure 19.	ADC / DAC	169
Figure 20.	External interrupt/event controller block diagram	207
Figure 21.	External interrupt/event GPIO mapping	210
Figure 22.	Single ADC block diagram	217
Figure 23.	Timing diagram	220
Figure 24.	Analog watchdog guarded area	220
Figure 25.	Injected conversion latency	222
Figure 26.	Calibration timing diagram	224
Figure 27.	Right alignment of data	224
Figure 28.	Left alignment of data	224
Figure 29.	Dual ADC block diagram ⁽¹⁾	229
Figure 30.	Injected simultaneous mode on 4 channels	230
Figure 31.	Regular simultaneous mode on 16 channels	230
Figure 32.	Fast interleaved mode on 1 channel in continuous conversion mode	231
Figure 33.	Slow interleaved mode on 1 channel	232
Figure 34.	Alternate trigger: injected channel group of each ADC	232
Figure 35.	Alternate trigger: 4 injected channels (each ADC) in discontinuous mode	233
Figure 36.	Alternate + Regular simultaneous	234
Figure 37.	Case of trigger occurring during injected conversion	234
Figure 38.	Interleaved single channel with injected sequence CH11, CH12	235
Figure 39.	Temperature sensor and VREFINT channel block diagram	235
Figure 40.	DAC channel block diagram	255
Figure 41.	Data registers in single DAC channel mode	257
Figure 42.	Data registers in dual DAC channel mode	257
Figure 43.	Timing diagram for conversion with trigger disabled TEN = 0	258
Figure 44.	DAC LFSR register calculation algorithm	259
Figure 45.	DAC conversion (SW trigger enabled) with LFSR wave generation	260
Figure 46.	DAC triangle wave generation	260
Figure 47.	DAC conversion (SW trigger enabled) with triangle wave generation	261
Figure 48.	DMA block diagram in connectivity line devices	275

Figure 49.	DMA block diagram in low-, medium- high- and XL-density devices	276
Figure 50.	DMA1 request mapping	281
Figure 51.	DMA2 request mapping	283
Figure 52.	Advanced-control timer block diagram	294
Figure 53.	Counter timing diagram with prescaler division change from 1 to 2	296
Figure 54.	Counter timing diagram with prescaler division change from 1 to 4	296
Figure 55.	Counter timing diagram, internal clock divided by 1	297
Figure 56.	Counter timing diagram, internal clock divided by 2	298
Figure 57.	Counter timing diagram, internal clock divided by 4	298
Figure 58.	Counter timing diagram, internal clock divided by N	298
Figure 59.	Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)	299
Figure 60.	Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)	299
Figure 61.	Counter timing diagram, internal clock divided by 1	301
Figure 62.	Counter timing diagram, internal clock divided by 2	301
Figure 63.	Counter timing diagram, internal clock divided by 4	302
Figure 64.	Counter timing diagram, internal clock divided by N	302
Figure 65.	Counter timing diagram, update event when repetition counter is not used	303
Figure 66.	Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6	304
Figure 67.	Counter timing diagram, internal clock divided by 2	304
Figure 68.	Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36	305
Figure 69.	Counter timing diagram, internal clock divided by N	305
Figure 70.	Counter timing diagram, update event with ARPE=1 (counter underflow)	306
Figure 71.	Counter timing diagram, Update event with ARPE=1 (counter overflow)	306
Figure 72.	Update rate examples depending on mode and TIMx_RCR register settings	307
Figure 73.	Control circuit in normal mode, internal clock divided by 1	308
Figure 74.	TI2 external clock connection example	309
Figure 75.	Control circuit in external clock mode 1	310
Figure 76.	External trigger input block	310
Figure 77.	Control circuit in external clock mode 2	311
Figure 78.	Capture/compare channel (example: channel 1 input stage)	312
Figure 79.	Capture/compare channel 1 main circuit	312
Figure 80.	Output stage of capture/compare channel (channel 1 to 3)	313
Figure 81.	Output stage of capture/compare channel (channel 4)	313
Figure 82.	PWM input mode timing	315
Figure 83.	Output compare mode, toggle on OC1	317
Figure 84.	Edge-aligned PWM waveforms (ARR=8)	318
Figure 85.	Center-aligned PWM waveforms (ARR=8)	320
Figure 86.	Complementary output with dead-time insertion	321
Figure 87.	Dead-time waveforms with delay greater than the negative pulse	322
Figure 88.	Dead-time waveforms with delay greater than the positive pulse	322
Figure 89.	Output behavior in response to a break	325
Figure 90.	Clearing TIMx OCxREF	326
Figure 91.	6-step generation, COM example (OSSR=1)	327
Figure 92.	Example of one pulse mode	328
Figure 93.	Example of counter operation in encoder interface mode	331
Figure 94.	Example of encoder interface mode with TI1FP1 polarity inverted	331
Figure 95.	Example of Hall sensor interface	333
Figure 96.	Control circuit in reset mode	334
Figure 97.	Control circuit in gated mode	335
Figure 98.	Control circuit in trigger mode	336
Figure 99.	Control circuit in external clock mode 2 + trigger mode	337
Figure 100.	General-purpose timer block diagram	367

Figure 101. Counter timing diagram with prescaler division change from 1 to 2	369
Figure 102. Counter timing diagram with prescaler division change from 1 to 4	369
Figure 103. Counter timing diagram, internal clock divided by 1	370
Figure 104. Counter timing diagram, internal clock divided by 2	370
Figure 105. Counter timing diagram, internal clock divided by 4	371
Figure 106. Counter timing diagram, internal clock divided by N	371
Figure 107. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)	372
Figure 108. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)	372
Figure 109. Counter timing diagram, internal clock divided by 1	373
Figure 110. Counter timing diagram, internal clock divided by 2	374
Figure 111. Counter timing diagram, internal clock divided by 4	374
Figure 112. Counter timing diagram, internal clock divided by N	374
Figure 113. Counter timing diagram, Update event	375
Figure 114. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6	376
Figure 115. Counter timing diagram, internal clock divided by 2	376
Figure 116. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36	377
Figure 117. Counter timing diagram, internal clock divided by N	377
Figure 118. Counter timing diagram, Update event with ARPE=1 (counter underflow)	378
Figure 119. Counter timing diagram, Update event with ARPE=1 (counter overflow)	378
Figure 120. Control circuit in normal mode, internal clock divided by 1	379
Figure 121. TI2 external clock connection example	380
Figure 122. Control circuit in external clock mode 1	381
Figure 123. External trigger input block	381
Figure 124. Control circuit in external clock mode 2	382
Figure 125. Capture/compare channel (example: channel 1 input stage)	382
Figure 126. Capture/compare channel 1 main circuit	383
Figure 127. Output stage of capture/compare channel (channel 1)	383
Figure 128. PWM input mode timing	385
Figure 129. Output compare mode, toggle on OC1	387
Figure 130. Edge-aligned PWM waveforms (ARR=8)	388
Figure 131. Center-aligned PWM waveforms (ARR=8)	389
Figure 132. Example of one-pulse mode	390
Figure 133. Clearing TIMx OCxREF	392
Figure 134. Example of counter operation in encoder interface mode	394
Figure 135. Example of encoder interface mode with TI1FP1 polarity inverted	394
Figure 136. Control circuit in reset mode	395
Figure 137. Control circuit in gated mode	396
Figure 138. Control circuit in trigger mode	397
Figure 139. Control circuit in external clock mode 2 + trigger mode	398
Figure 140. Master/Slave timer example	398
Figure 141. Gating timer 2 with OC1REF of timer 1	399
Figure 142. Gating timer 2 with Enable of timer 1	400
Figure 143. Triggering timer 2 with update of timer 1	401
Figure 144. Triggering timer 2 with Enable of timer 1	402
Figure 145. Triggering timer 1 and 2 with timer 1 TI1 input	403
Figure 146. General-purpose timer block diagram (TIM9 and TIM12)	426
Figure 147. General-purpose timer block diagram (TIM10/11/13/14)	427
Figure 148. Counter timing diagram with prescaler division change from 1 to 2	429
Figure 149. Counter timing diagram with prescaler division change from 1 to 4	429
Figure 150. Counter timing diagram, internal clock divided by 1	430
Figure 151. Counter timing diagram, internal clock divided by 2	431
Figure 152. Counter timing diagram, internal clock divided by 4	431

3.3 Memory map

See the datasheet corresponding to your device for a comprehensive diagram of the memory map. [Table 3](#) gives the boundary addresses of the peripherals available in all STM32F10xxx devices.

Table 3. Register boundary addresses

Boundary address	Peripheral	Bus	Register map
0xA000 0000 - 0xA000 0FFF	FSMC	AHB	Section 21.6.9 on page 564
0x5000 0000 - 0x5003 FFFF	USB OTG FS		Section 28.16.6 on page 913
0x4003 0000 - 0x4FFF FFFF	Reserved		-
0x4002 8000 - 0x4002 9FFF	Ethernet		Section 29.8.5 on page 1071
0x4002 3400 - 0x4002 7FFF	Reserved		-
0x4002 3000 - 0x4002 33FF	CRC		Section 4.4.4 on page 65
0x4002 2000 - 0x4002 23FF	Flash memory interface		-
0x4002 1400 - 0x4002 1FFF	Reserved		-
0x4002 1000 - 0x4002 13FF	Reset and clock control RCC		Section 7.3.11 on page 121
0x4002 0800 - 0x4002 0FFF	Reserved		-
0x4002 0400 - 0x4002 07FF	DMA2		Section 13.4.7 on page 289
0x4002 0000 - 0x4002 03FF	DMA1		-
0x4001 8400 - 0x4001 FFFF	Reserved		-
0x4001 8000 - 0x4001 83FF	SDIO		Section 22.9.16 on page 621

Table 3. Register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4001 5800 - 0x4001 7FFF	Reserved	APB2	-
0x4001 5400 - 0x4001 57FF	TIM11 timer		Section 16.5.11 on page 468
0x4001 5000 - 0x4001 53FF	TIM10 timer		Section 16.5.11 on page 468
0x4001 4C00 - 0x4001 4FFF	TIM9 timer		Section 16.4.13 on page 458
0x4001 4000 - 0x4001 4BFF	Reserved		-
0x4001 3C00 - 0x4001 3FFF	ADC3		Section 11.12.15 on page 252
0x4001 3800 - 0x4001 3BFF	USART1		Section 27.6.8 on page 827
0x4001 3400 - 0x4001 37FF	TIM8 timer		Section 14.4.21 on page 363
0x4001 3000 - 0x4001 33FF	SPI1		Section 25.5 on page 742
0x4001 2C00 - 0x4001 2FFF	TIM1 timer		Section 14.4.21 on page 363
0x4001 2800 - 0x4001 2BFF	ADC2		Section 11.12.15 on page 252
0x4001 2400 - 0x4001 27FF	ADC1		
0x4001 2000 - 0x4001 23FF	GPIO Port G		
0x4001 1C00 - 0x4001 1FFF	GPIO Port F		
0x4001 1800 - 0x4001 1BFF	GPIO Port E		
0x4001 1400 - 0x4001 17FF	GPIO Port D		Section 9.5 on page 194
0x4001 1000 - 0x4001 13FF	GPIO Port C		
0x4001 0C00 - 0x4001 0FFF	GPIO Port B		
0x4001 0800 - 0x4001 0BFF	GPIO Port A		
0x4001 0400 - 0x4001 07FF	EXTI		Section 10.3.7 on page 214
0x4001 0000 - 0x4001 03FF	AFIO		Section 9.5 on page 194

Table 3. Register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4000 7800 - 0x4000 FFFF	Reserved	APB1	-
0x4000 7400 - 0x4000 77FF	DAC		Section 12.5.14 on page 273
0x4000 7000 - 0x4000 73FF	Power control PWR		Section 5.4.3 on page 80
0x4000 6C00 - 0x4000 6FFF	Backup registers (BKP)		Section 6.4.5 on page 85
0x4000 6400 - 0x4000 67FF	bxCAN1		Section 24.9.5 on page 695
0x4000 6800 - 0x4000 6BFF	bxCAN2		-
0x4000 6000 ⁽¹⁾ - 0x4000 63FF	Shared USB/CAN SRAM 512 bytes		Section 23.5.4 on page 651
0x4000 5C00 - 0x4000 5FFF	USB device FS registers		Section 26.6.10 on page 784
0x4000 5800 - 0x4000 5BFF	I2C2		Section 27.6.8 on page 827
0x4000 5400 - 0x4000 57FF	I2C1		-
0x4000 5000 - 0x4000 53FF	UART5		Section 25.5 on page 742
0x4000 4C00 - 0x4000 4FFF	UART4		Section 25.5 on page 742
0x4000 4800 - 0x4000 4BFF	USART3		-
0x4000 4400 - 0x4000 47FF	USART2		Section 19.4.5 on page 499
0x4000 4000 - 0x4000 43FF	Reserved		Section 20.6.4 on page 506
0x4000 3C00 - 0x4000 3FFF	SPI3/I2S		Section 18.4.7 on page 493
0x4000 3800 - 0x4000 3BFF	SPI2/I2S		-
0x4000 3400 - 0x4000 37FF	Reserved		Section 16.5.11 on page 468
0x4000 3000 - 0x4000 33FF	Independent watchdog (IWDG)		Section 16.4.13 on page 458
0x4000 2C00 - 0x4000 2FFF	Window watchdog (WWDG)		Section 17.4.9 on page 481
0x4000 2800 - 0x4000 2BFF	RTC		Section 15.4.19 on page 423
0x4000 2400 - 0x4000 27FF	Reserved		
0x4000 2000 - 0x4000 23FF	TIM14 timer		
0x4000 1C00 - 0x4000 1FFF	TIM13 timer		
0x4000 1800 - 0x4000 1BFF	TIM12 timer		
0x4000 1400 - 0x4000 17FF	TIM7 timer		
0x4000 1000 - 0x4000 13FF	TIM6 timer		
0x4000 0C00 - 0x4000 0FFF	TIM5 timer		
0x4000 0800 - 0x4000 0BFF	TIM4 timer		
0x4000 0400 - 0x4000 07FF	TIM3 timer		
0x4000 0000 - 0x4000 03FF	TIM2 timer		

1. This shared SRAM can be fully accessed only in low-, medium-, high- and XL-density devices, not in connectivity line devices.

3.3.1 Embedded SRAM

The STM32F10xxx features up to 96 Kbytes of static SRAM. It can be accessed as bytes, half-words (16 bits) or full words (32 bits). The SRAM start address is 0x2000 0000.

3.3.2 Bit banding

The Cortex®-M3 memory map includes two bit-band regions. These regions map each word in an alias region of memory to a bit in a bit-band region of memory. Writing to a word in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.

In the STM32F10xxx both peripheral registers and SRAM are mapped in a bit-band region. This allows single bit-band write and read operations to be performed. The operations are only available for Cortex®-M3 accesses, not from other bus masters (e.g. DMA).

A mapping formula shows how to reference each word in the alias region to a corresponding bit in the bit-band region. The mapping formula is:

$$\text{bit_word_addr} = \text{bit_band_base} + (\text{byte_offset} \times 32) + (\text{bit_number} \times 4)$$

where:

bit_word_addr is the address of the word in the alias memory region that maps to the targeted bit.

bit_band_base is the starting address of the alias region

byte_offset is the number of the byte in the bit-band region that contains the targeted bit

bit_number is the bit position (0-7) of the targeted bit.

Example:

The following example shows how to map bit 2 of the byte located at SRAM address 0x20000300 in the alias region:

$$0x22006008 = 0x22000000 + (0x300*32) + (2*4).$$

Writing to address 0x22006008 has the same effect as a read-modify-write operation on bit 2 of the byte at SRAM address 0x20000300.

Reading address 0x22006008 returns the value (0x01 or 0x00) of bit 2 of the byte at SRAM address 0x20000300 (0x01: bit set; 0x00: bit reset).

For more information on Bit-Banding refer to the *Cortex®-M3 Technical Reference Manual*.

3.3.3 Embedded Flash memory

The high-performance Flash memory module has the following key features:

- For XL-density devices: density of up to 1 Mbyte with dual bank architecture for read-while-write (RWW) capability:
 - bank 1: fixed size of 512 Kbytes
 - bank 2: up to 512 Kbytes
- For other devices: density of up to 512 Kbytes
- Memory organization: the Flash memory is organized as a main block and an information block:
 - Main memory block of size:
 - up to 128 Kbytes × 64 bits divided into 512 pages of 2 Kbytes each (see [Table 8](#)) for XL-density devices
 - up to 4 Kb × 64 bits divided into 32 pages of 1 Kbyte each for low-density devices (see [Table 4](#))
 - up to 16 Kb × 64 bits divided into 128 pages of 1 Kbyte each for medium-density devices (see [Table 5](#))
 - up to 64 Kb × 64 bits divided into 256 pages of 2 Kbytes each (see [Table 6](#)) for high-density devices
 - up to 32 Kbit × 64 bits divided into 128 pages of 2 Kbytes each (see [Table 7](#)) for connectivity line devices
 - Information block of size:
 - 770 × 64 bits for XL-density devices (see [Table 8](#))
 - 2360 × 64 bits for connectivity line devices (see [Table 7](#))
 - 258 × 64 bits for other devices (see [Table 4](#), [Table 5](#) and [Table 6](#))

The Flash memory interface (FLITF) features:

- Read interface with prefetch buffer (2x64-bit words)
- Option byte Loader
- Flash Program / Erase operation
- Read / Write protection

Table 4. Flash module organization (low-density devices)

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 03FF	1 K
	Page 1	0x0800 0400 - 0x0800 07FF	1 K
	Page 2	0x0800 0800 - 0x0800 0BFF	1 K
	Page 3	0x0800 0C00 - 0x0800 0FFF	1 K
	Page 4	0x0800 1000 - 0x0800 13FF	1 K
	.	.	.
	.	.	.
	Page 31	0x0800 7C00 - 0x0800 7FFF	1 K

Table 4. Flash module organization (low-density devices) (continued)

Block	Name	Base addresses	Size (bytes)
Information block	System memory	0x1FFF F000 - 0x1FFF F7FF	2 K
	Option bytes	0x1FFF F800 - 0x1FFF F80F	16
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPTR	0x4002 2020 - 0x4002 2023	4

Table 5. Flash module organization (medium-density devices)

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 03FF	1 K
	Page 1	0x0800 0400 - 0x0800 07FF	1 K
	Page 2	0x0800 0800 - 0x0800 0BFF	1 K
	Page 3	0x0800 0C00 - 0x0800 0FFF	1 K
	Page 4	0x0800 1000 - 0x0800 13FF	1 K
	.	.	.
	.	.	.
	Page 127	0x0801 FC00 - 0x0801 FFFF	1 K
Information block	System memory	0x1FFF F000 - 0x1FFF F7FF	2 K
	Option bytes	0x1FFF F800 - 0x1FFF F80F	16
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPTR	0x4002 2020 - 0x4002 2023	4

Table 6. Flash module organization (high-density devices)

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 07FF	2 K
	Page 1	0x0800 0800 - 0x0800 0FFF	2 K
	Page 2	0x0800 1000 - 0x0800 17FF	2 K
	Page 3	0x0800 1800 - 0x0800 1FFF	2 K
	.	.	.
	Page 255	0x0807 F800 - 0x0807 FFFF	2 K
Information block	System memory	0x1FFF F000 - 0x1FFF F7FF	2 K
	Option bytes	0x1FFF F800 - 0x1FFF F80F	16
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Table 7. Flash module organization (connectivity line devices)

Block	Name	Base addresses	Size (bytes)
Main memory	Page 0	0x0800 0000 - 0x0800 07FF	2 K
	Page 1	0x0800 0800 - 0x0800 0FFF	2 K
	Page 2	0x0800 1000 - 0x0800 17FF	2 K
	Page 3	0x0800 1800 - 0x0800 1FFF	2 K
	.	.	.
	Page 127	0x0803 F800 - 0x0803 FFFF	2 K
Information block	System memory	0x1FFF B000 - 0x1FFF F7FF	18 K
	Option bytes	0x1FFF F800 - 0x1FFF F80F	16

Table 7. Flash module organization (connectivity line devices) (continued)

Block	Name	Base addresses	Size (bytes)
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WRPR	0x4002 2020 - 0x4002 2023	4

Table 8. XL-density Flash module organization

Block	Name	Base addresses	Size (bytes)		
Main memory	Bank 1	Page 0	0x0800 0000 - 0x0800 07FF	2 K	
		Page 1	0x0800 0800 - 0x0800 0FFF	2 K	
		
		Page 255	0x0807 F800 - 0x0807 FFFF	2 K	
	Bank 2	Page 256	0x0808 0000 - 0x0808 07FF	2 K	
		Page 257	0x0808 0800 - 0x0808 0FFF	2 K	
		
		Page 511	0x080F F800 - 0x080F FFFF	2 K	
Information block		System memory	0x1FFF E000 - 0x1FFF F7FF	6 K	
		Option bytes	0x1FFF F800 - 0x1FFF F80F	16	

Table 8. XL-density Flash module organization (continued)

Block	Name	Base addresses	Size (bytes)
Flash memory interface registers	FLASH_ACR	0x4002 2000 - 0x4002 2003	4
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4
	FLASH_SR	0x4002 200C - 0x4002 200F	4
	FLASH_CR	0x4002 2010 - 0x4002 2013	4
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201C - 0x4002 201F	4
	FLASH_WPR	0x4002 2020 - 0x4002 2023	4
	Reserved	0x4002 2024 - 0x4002 2043	32
	FLASH_KEYR2	0x4002 2044 - 0x4002 2047	4
	Reserved	0x4002 2048 - 0x4002 204B	4
	FLASH_SR2	0x4002 204C - 0x4002 204F	4
	FLASH_CR2	0x4002 2050 - 0x4002 2053	4
	FLASH_AR2	0x4002 2054 - 0x4002 2057	4

Note: For further information on the Flash memory interface registers, refer to the “STM32F10xxx XL-density Flash programming manual” (PM0068) for XL-density devices, “STM32F10xxx Flash programming manual” (PM0075) for other devices.

Reading the Flash memory

Flash memory instructions and data access are performed through the AHB bus. The prefetch block is used for instruction fetches through the ICode bus. Arbitration is performed in the Flash memory interface, and priority is given to data access on the DCode bus.

Read accesses can be performed with the following configuration options:

- Latency: number of wait states for a read operation programmed on-the-fly
- Prefetch buffer (2 x 64-bit blocks): it is enabled after reset; a whole block can be replaced with a single read from the Flash memory as the size of the block matches the bandwidth of the Flash memory. Thanks to the prefetch buffer, faster CPU execution is possible as the CPU fetches one word at a time with the next word readily available in the prefetch buffer
- Half cycle: for power optimization

Note: These options have to be used in accordance with the Flash memory access time. The wait states represent the ratio of the SYSCLK (system clock) period to the Flash memory access time:

- 0 wait states, if $0 < \text{SYSCLK} \leq 24 \text{ MHz}$
- 1 wait state, if $24 \text{ MHz} < \text{SYSCLK} \leq 48 \text{ MHz}$
- 2 wait states, if $48 \text{ MHz} < \text{SYSCLK} \leq 72 \text{ MHz}$

Half cycle configuration is not available in combination with a prescaler on the AHB. The system clock (SYSCLK) should be equal to the HCLK clock. This feature can therefore be

used only with a low-frequency clock of 8 MHz or less. It can be generated from the HSI or the HSE but not from the PLL.

The prefetch buffer must be kept on when using a prescaler different from 1 on the AHB clock.

The prefetch buffer must be switched on/off only when SY SCLK is lower than 24 MHz and no prescaler is applied on the AHB clock (SY SCLK must be equal to HCLK). The prefetch buffer is usually switched on/off during the initialization routine, while the microcontroller is running on the internal 8 MHz RC (HSI) oscillator.

Using DMA: DMA accesses Flash memory on the DCode bus and has priority over ICode instructions. The DMA provides one free cycle after each transfer. Some instructions can be performed together with DMA transfer.

Programming and erasing the Flash memory

The Flash memory can be programmed 16 bits (half words) at a time.

For write and erase operations on the Flash memory (write/erase), the internal RC oscillator (HSI) must be ON.

The Flash memory erase operation can be performed at page level or on the whole Flash area (mass-erase). The mass-erase does not affect the information blocks.

To ensure that there is no over-programming, the Flash Programming and Erase Controller blocks are clocked by a fixed clock.

The End of write operation (programming or erasing) can trigger an interrupt. This interrupt can be used to exit from WFI mode, only if the FLITF clock is enabled. Otherwise, the interrupt is served only after an exit from WFI.

The FLASH_ACR register is used to enable/disable prefetch and half cycle access, and to control the Flash memory access time according to the CPU frequency. The tables below provide the bit map and bit descriptions for this register.

For complete information on Flash memory operations and register configurations, refer to the STM32F10xxx Flash programming manual (PM0075) or to the XL STM32F10xxx Flash programming manual (PM0068).

Flash access control register (FLASH_ACR)

Address offset: 0x00

Reset value: 0x0000 0030

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
								PRFTBS	PRFTBE	HLFCYA	LATENCY				
								r	rw	rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **PRFTBS**: Prefetch buffer status

This bit provides the status of the prefetch buffer.

- 0: Prefetch buffer is disabled
- 1: Prefetch buffer is enabled

Bit 4 **PRFTBE**: Prefetch buffer enable

- 0: Prefetch is disabled
- 1: Prefetch is enabled

Bit 3 **HLFCYA**: Flash half cycle access enable

- 0: Half cycle is disabled
- 1: Half cycle is enabled

Bits 2:0 **LATENCY**: Latency

These bits represent the ratio of the SYSCLK (system clock) period to the Flash access time.

000 Zero wait state, if $0 < \text{SYSCLK} \leq 24 \text{ MHz}$

001 One wait state, if $24 \text{ MHz} < \text{SYSCLK} \leq 48 \text{ MHz}$

010 Two wait states, if $48 \text{ MHz} < \text{SYSCLK} \leq 72 \text{ MHz}$

3.4 Boot configuration

In the STM32F10xxx, 3 different boot modes can be selected through BOOT[1:0] pins as shown in [Table 9](#).

Table 9. Boot modes

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as boot space
0	1	System memory	System memory is selected as boot space
1	1	Embedded SRAM	Embedded SRAM is selected as boot space

The values on the BOOT pins are latched on the 4th rising edge of SYSCLK after a reset. It is up to the user to set the BOOT1 and BOOT0 pins after Reset to select the required boot mode.

The BOOT pins are also re-sampled when exiting from Standby mode. Consequently they must be kept in the required Boot mode configuration in Standby mode. After this startup delay has elapsed, the CPU fetches the top-of-stack value from address 0x0000 0000, then starts code execution from the boot memory starting from 0x0000 0004.

Due to its fixed memory map, the code area starts from address 0x0000 0000 (accessed through the ICode/DCode buses) while the data area (SRAM) starts from address 0x2000 0000 (accessed through the system bus). The Cortex®-M3 CPU always fetches the reset vector on the ICode bus, which implies to have the boot space available only in the code area (typically, Flash memory). STM32F10xxx microcontrollers implement a special mechanism to be able to boot also from SRAM and not only from main Flash memory and System memory.

Depending on the selected boot mode, main Flash memory, system memory or SRAM is accessible as follows:

- Boot from main Flash memory: the main Flash memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x800 0000). In other words, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x800 0000.
- Boot from system memory: the system memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x1FFF B000 in connectivity line devices, 0x1FFF F000 in other devices).
- Boot from the embedded SRAM: SRAM is accessible only at address 0x2000 0000.

Note:

When booting from SRAM, in the application initialization code, you have to relocate the vector table in SRAM using the NVIC exception table and offset register.

For XL-density devices, when booting from the main Flash memory, you have an option to boot from any of two memory banks. By default, boot from Flash memory bank 1 is selected. You can choose to boot from Flash memory bank 2 by clearing the BFB2 bit in the user option bytes. When this bit is cleared and the boot pins are in the boot from main Flash memory configuration, the device boots from system memory, and the boot loader jumps to execute the user application programmed in Flash memory bank 2. For further details refer to AN2606.

Note:

When booting from Bank2 in the applications initialization code, relocate the vector table to the Bank2 base address. (0x0808 0000) using the NVIC exception table and offset register.

Embedded boot loader

The embedded boot loader is located in the System memory, programmed by ST during production. It is used to reprogram the Flash memory with one of the available serial interfaces:

- In low-, medium- and high-density devices the bootloader is activated through the USART1 interface.
- In XL-density devices the boot loader is activated through the following interfaces: USART1 or USART2 (remapped).
- In connectivity line devices the boot loader can be activated through one of the following interfaces: USART1, USART2 (remapped), CAN2 (remapped) or USB OTG FS in Device mode (DFU: device firmware upgrade).

The USART peripheral operates with the internal 8 MHz oscillator (HSI). The CAN and USB OTG FS, however, can only function if an external 8 MHz, 14.7456 MHz or 25 MHz clock (HSE) is present.

Note: *For further details refer to AN2606.*

4 CRC calculation unit

Low-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density devices are *STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes*.

XL-density devices are *STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 768 Kbytes and 1 Mbyte*.

Connectivity line devices are STM32F105xx and STM32F107xx microcontrollers.

4.1 CRC introduction

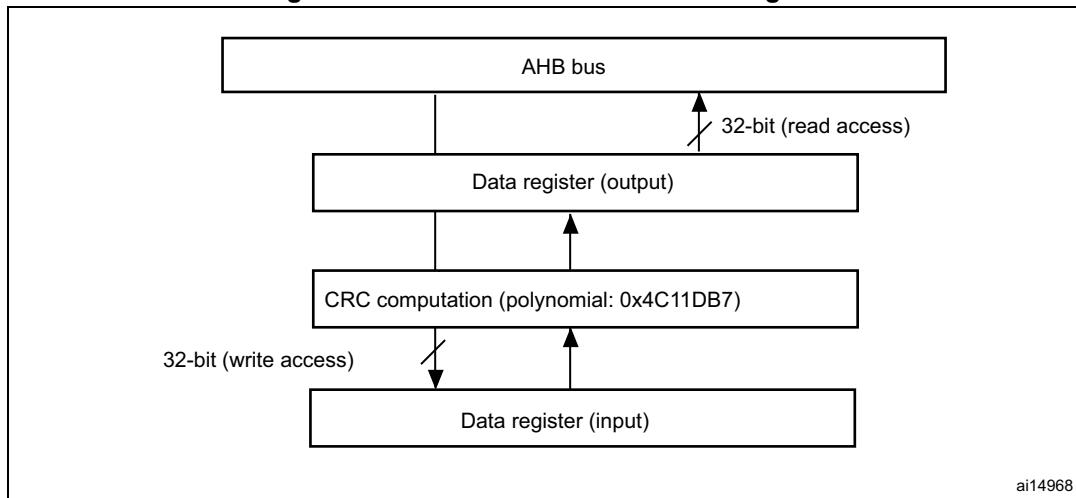
The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from a 32-bit data word and a fixed generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the EN/IEC 60335-1 standard, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link-time and stored at a given memory location.

4.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
 - $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Single input/output 32-bit data register
- CRC computation done in 4 AHB clock cycles (HCLK)
- General-purpose 8-bit register (can be used for temporary storage)

The block diagram is shown in [Figure 3](#).

Figure 3. CRC calculation unit block diagram

4.3 CRC functional description

The CRC calculation unit mainly consists of a single 32-bit data register, which:

- is used as an input register to enter new data in the CRC calculator (when writing into the register)
- holds the result of the previous CRC calculation (when reading the register)

Each write operation into the data register creates a combination of the previous CRC value and the new one (CRC computation is done on the whole 32-bit data word, and not byte per byte).

The write operation is stalled until the end of the CRC computation, thus allowing back-to-back write accesses or consecutive write and read accesses.

The CRC calculator can be reset to 0xFFFF FFFF with the RESET control bit in the CRC_CR register. This operation does not affect the contents of the CRC_IDR register.

4.4 CRC registers

The CRC calculation unit contains two data registers and a control register. The peripheral The CRC registers have to be accessed by words (32 bits).

4.4.1 Data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR [31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 Data register bits

Used as an input register when writing new data into the CRC calculator.
Holds the previous CRC calculation result when it is read.

4.4.2 Independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								IDR[7:0]							
		rw		rw		rw		rw		rw		rw		rw	

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 General-purpose 8-bit data register bits

Can be used as a temporary storage location for one byte.

This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register.

4.4.3 Control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved														RESET		
																w

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 RESET bit

Resets the CRC calculation unit and sets the data register to 0xFFFF FFFF.
This bit can only be set, it is automatically cleared by hardware.

4.4.4 CRC register map

The following table provides the CRC register map and reset values.

Table 10. CRC calculation unit register map and reset values

Offset	Register	31-24	23-16	15-8	7	6	5	4	3	2	1	0
0x00	CRC_DR	Data register										
	Reset value	0xFFFF FFFF										

Table 10. CRC calculation unit register map and reset values (continued)

Offset	Register	31-24	23-16	15-8	7	6	5	4	3	2	1	0		
0x04	CRC_IDR	Reserved				Independent data register								
	Reset value					0x00								
0x08	CRC_CR	Reserved									RESET	0		
	Reset value													

5 Power control (PWR)

Low-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density devices are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

XL-density devices are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 768 Kbytes and 1 Mbyte.

Connectivity line devices are STM32F105xx and STM32F107xx microcontrollers.

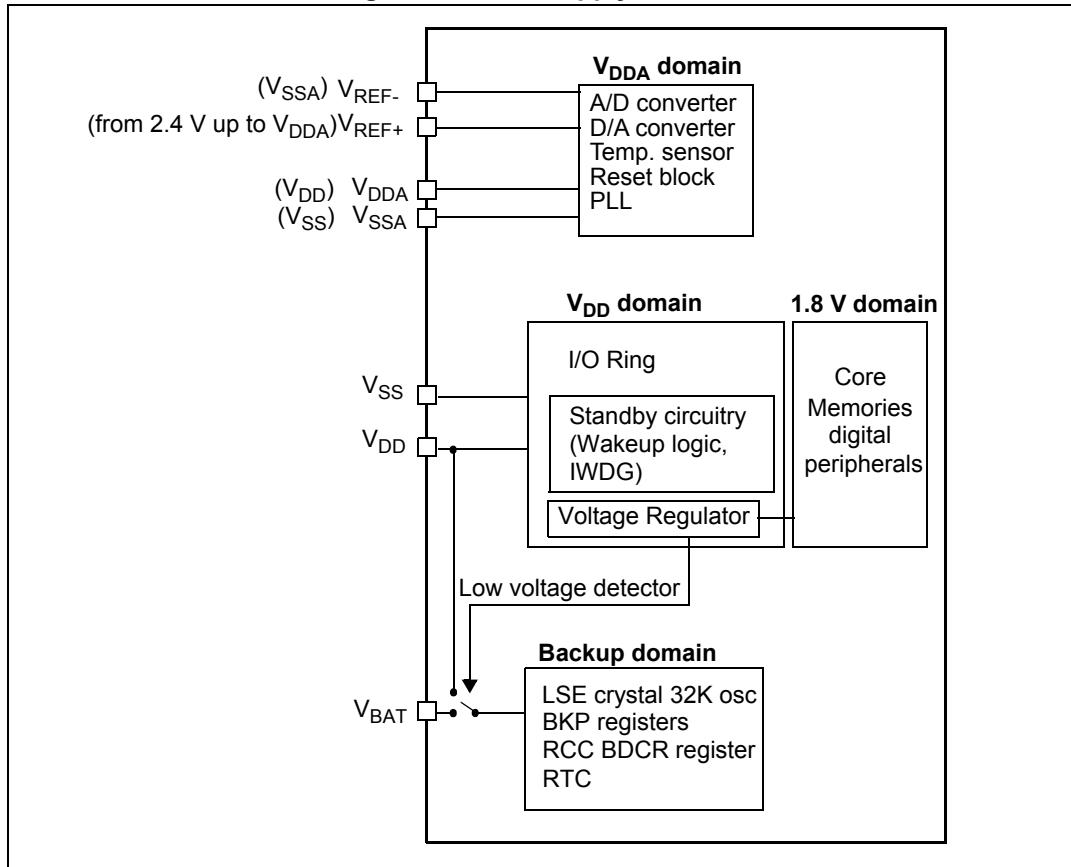
This section applies to the whole STM32F101xx family, unless otherwise specified.

5.1 Power supplies

The device requires a 2.0-to-3.6 V operating voltage supply (V_{DD}). An embedded regulator is used to supply the internal 1.8 V digital power.

The real-time clock (RTC) and backup registers can be powered from the V_{BAT} voltage when the main V_{DD} supply is powered off.

Figure 4. Power supply overview



1. V_{DDA} and V_{SSA} must be connected to V_{DD} and V_{SS}, respectively.

5.1.1 Independent A/D and D/A converter supply and reference voltage

To improve conversion accuracy, the ADC and the DAC have an independent power supply which can be separately filtered and shielded from noise on the PCB.

- The ADC and DAC voltage supply input is available on a separate V_{DDA} pin.
- An isolated supply ground connection is provided on pin V_{SSA}.

When available (according to package), V_{REF-} must be tied to V_{SSA}.

On 100-pin and 144-pin packages

To ensure a better accuracy on low-voltage inputs and outputs, the user can connect a separate external reference voltage on V_{REF+}. V_{REF+} is the highest voltage, represented by the full scale value, for an analog input (ADC) or output (DAC) signal. The voltage on V_{REF+} can range from 2.4 V to V_{DDA}.

On 64-pin packages and packages with less pins

The V_{REF+} and V_{REF-} pins are not available, they are internally connected to the ADC voltage supply (V_{DDA}) and ground (V_{SSA}).

5.1.2 Battery backup domain

To retain the content of the Backup registers and supply the RTC function when V_{DD} is turned off, V_{BAT} pin can be connected to an optional standby voltage supplied by a battery or by another source.

The V_{BAT} pin powers the RTC unit, the LSE oscillator and the PC13 to PC15 IOs, allowing the RTC to operate even when the main digital supply (V_{DD}) is turned off. The switch to the V_{BAT} supply is controlled by the Power Down Reset embedded in the Reset block.

Warning: During $t_{RSTTEMPO}$ (temporization at V_{DD} startup) or after a PDR is detected, the power switch between V_{BAT} and V_{DD} remains connected to V_{BAT} .
 During the startup phase, if V_{DD} is established in less than $t_{RSTTEMPO}$ (Refer to the datasheet for the value of $t_{RSTTEMPO}$) and $V_{DD} > V_{BAT} + 0.6$ V, a current may be injected into V_{BAT} through an internal diode connected between V_{DD} and the power switch (V_{BAT}).
If the power supply/battery connected to the V_{BAT} pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the V_{BAT} pin.

If no external battery is used in the application, it is recommended to connect V_{BAT} externally to V_{DD} with a 100 nF external ceramic decoupling capacitor (for more details refer to AN2586).

When the backup domain is supplied by V_{DD} (analog switch connected to V_{DD}), the following functions are available:

- PC14 and PC15 can be used as either GPIO or LSE pins
- PC13 can be used as GPIO, TAMPER pin, RTC Calibration Clock, RTC Alarm or second output (refer to [Section 6: Backup registers \(BKP\)](#))

Note: Due to the fact that the switch only sinks a limited amount of current (3 mA), the use of GPIOs PC13 to PC15 in output mode is restricted: the speed has to be limited to 2 MHz with a maximum load of 30 pF and these IOs must not be used as a current source (e.g. to drive a LED).

When the backup domain is supplied by V_{BAT} (analog switch connected to V_{BAT} because V_{DD} is not present), the following functions are available:

- PC14 and PC15 can be used as LSE pins only
- PC13 can be used as TAMPER pin, RTC Alarm or Second output (refer to [Section 6.4.2: RTC clock calibration register \(BKP_RTCCR\)](#)).

5.1.3 Voltage regulator

The voltage regulator is always enabled after Reset. It works in three different modes depending on the application modes.

- In Run mode, the regulator supplies full power to the 1.8 V domain (core, memories and digital peripherals).
- In Stop mode the regulator supplies low-power to the 1.8 V domain, preserving contents of registers and SRAM
- In Standby Mode, the regulator is powered off. The contents of the registers and SRAM are lost except for the Standby circuitry and the Backup Domain.

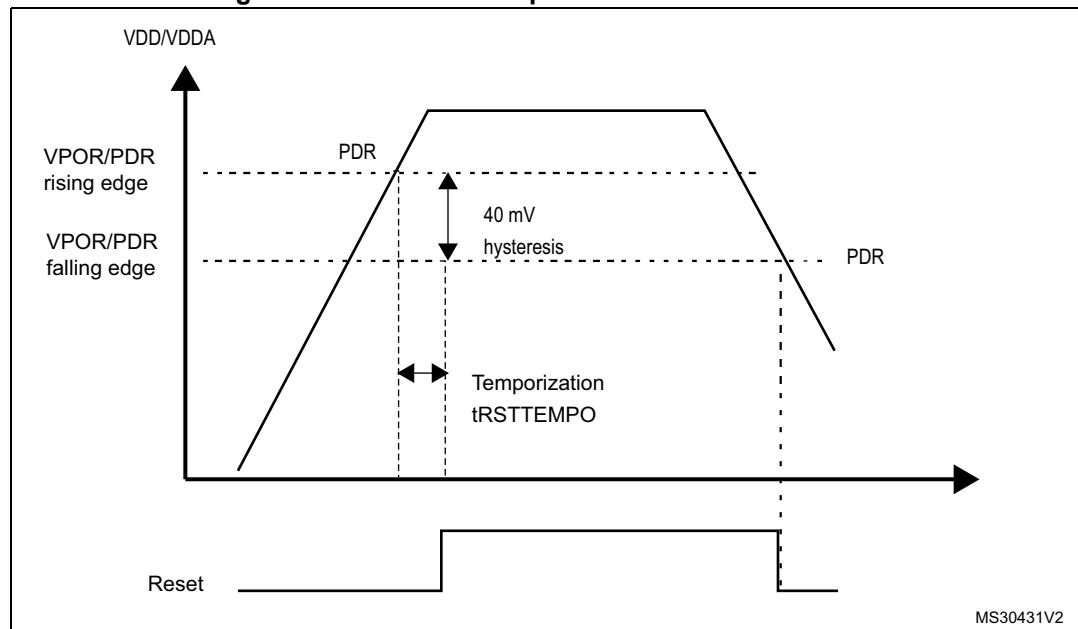
5.2 Power supply supervisor

5.2.1 Power on reset (POR)/power down reset (PDR)

The device has an integrated POR/PDR circuitry that allows proper operation starting from/down to 2 V.

The device remains in Reset mode when V_{DD}/V_{DDA} is below a specified threshold, $V_{POR/PDR}$, without the need for an external reset circuit. For more details concerning the power on/power down reset threshold, refer to the electrical characteristics of the datasheet.

Figure 5. Power on reset/power down reset waveform



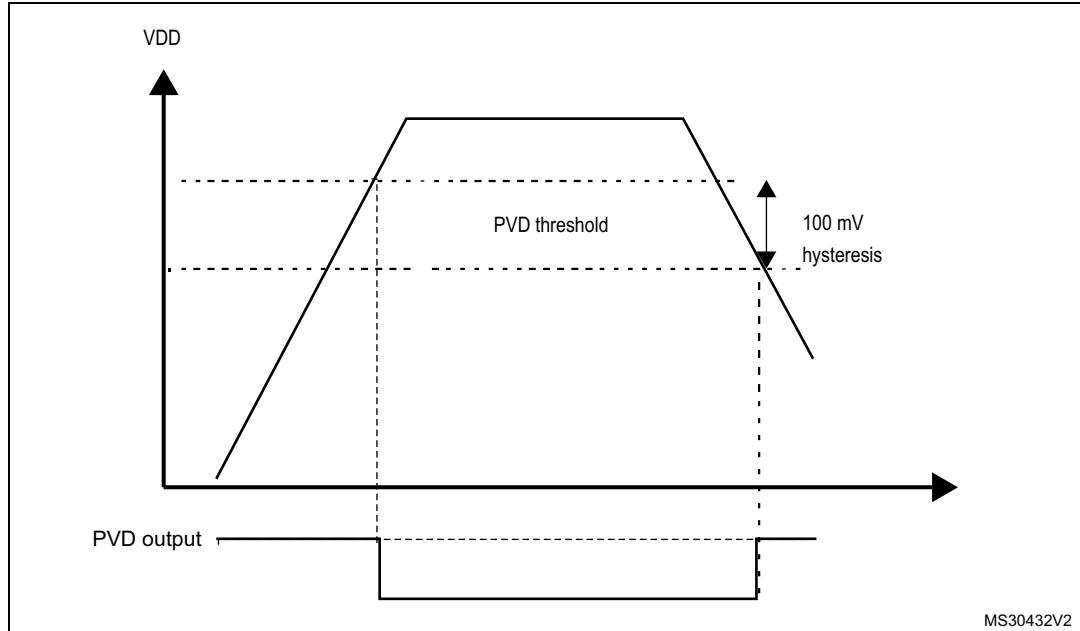
5.2.2 Programmable voltage detector (PVD)

The PVD can be used to monitor the V_{DD}/V_{DDA} power supply by comparing it to a threshold selected by the PLS[2:0] bits in the [Power control register \(PWR_CR\)](#).

The PVD is enabled by setting the PVDE bit.

A PVDO flag is available, in the *Power control/status register (PWR_CSR)*, to indicate if V_{DD}/V_{DDA} is higher or lower than the PVD threshold. This event is internally connected to the EXTI line16 and can generate an interrupt if enabled through the EXTI registers. The PVD output interrupt can be generated when V_{DD}/V_{DDA} drops below the PVD threshold and/or when V_{DD}/V_{DDA} rises above the PVD threshold depending on EXTI line16 rising/falling edge configuration. As an example the service routine could perform emergency shutdown tasks.

Figure 6. PVD thresholds



5.3 Low-power modes

By default, the microcontroller is in Run mode after a system or a power Reset. Several low-power modes are available to save power when the CPU does not need to be kept running, for example when waiting for an external event. It is up to the user to select the mode that gives the best compromise between low-power consumption, short startup time and available wakeup sources.

The STM32F10xxx devices feature three low-power modes:

- Sleep mode (CPU clock off, all peripherals including Cortex®-M3 core peripherals like NVIC, SysTick, etc. are kept running)
- Stop mode (all clocks are stopped)
- Standby mode (1.8V domain powered-off)

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

- Slowing down the system clocks
- Gating the clocks to the APB and AHB peripherals when they are unused.

Table 11. Low-power mode summary

Mode name	Entry	Wakeup	Effect on 1.8V domain clocks	Effect on V _{DD} domain clocks	Voltage regulator
Sleep (Sleep now or Sleep-on -exit)	WFI	Any interrupt	CPU clock OFF no effect on other clocks or analog clock sources	None	ON
	WFE	Wakeup event			
Stop	PDDS and LPDS bits + SLEEPDEEP bit + WFI or WFE	Any EXTI line (configured in the EXTI registers)	All 1.8V domain clocks OFF	HSI and HSE oscillators OFF	ON or in low-power mode (depends on <i>Power control register (PWR_CR)</i>)
Standby	PDDS bit + SLEEPDEEP bit + WFI or WFE	WKUP pin rising edge, RTC alarm, external reset in NRST pin, IWDG reset			OFF

5.3.1 Slowing down system clocks

In Run mode the speed of the system clocks (SYSCLK, HCLK, PCLK1, PCLK2) can be reduced by programming the prescaler registers. These prescalers can also be used to slow down peripherals before entering Sleep mode.

For more details refer to [Section 7.3.2: Clock configuration register \(RCC_CFGR\)](#).

5.3.2 Peripheral clock gating

In Run mode, the HCLK and PCLKx for individual peripherals and memories can be stopped at any time to reduce power consumption.

To further reduce power consumption in Sleep mode the peripheral clocks can be disabled prior to executing the WFI or WFE instructions.

Peripheral clock gating is controlled by the *AHB peripheral clock enable register (RCC_AHBENR)*, *APB1 peripheral clock enable register (RCC_APB1ENR)* and *APB2 peripheral clock enable register (RCC_APB2ENR)*.

5.3.3 Sleep mode

Entering Sleep mode

The Sleep mode is entered by executing the WFI (Wait For Interrupt) or WFE (Wait for Event) instructions. Two options are available to select the Sleep mode entry mechanism, depending on the SLEEPONEXIT bit in the Cortex®-M3 System Control register:

- Sleep-now: if the SLEEPONEXIT bit is cleared, the MCU enters Sleep mode as soon as WFI or WFE instruction is executed.
- Sleep-on-exit: if the SLEEPONEXIT bit is set, the MCU enters Sleep mode as soon as it exits the lowest priority ISR.

In the Sleep mode, all I/O pins keep the same state as in the Run mode.

Refer to [Table 12](#) and [Table 13](#) for details on how to enter Sleep mode.

Exiting Sleep mode

If the WFI instruction is used to enter Sleep mode, any peripheral interrupt acknowledged by the nested vectored interrupt controller (NVIC) can wake up the device from Sleep mode.

If the WFE instruction is used to enter Sleep mode, the MCU exits Sleep mode as soon as an event occurs. The wakeup event can be generated either by:

- enabling an interrupt in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the Cortex®-M3 System Control register. When the MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
- or configuring an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set.

This mode offers the lowest wakeup time as no time is wasted in interrupt entry/exit.

Refer to [Table 12](#) and [Table 13](#) for more details on how to exit Sleep mode.

Table 12. Sleep-now

Sleep-now mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 0 Refer to the Cortex®-M3 System Control register.
Mode exit	If WFI was used for entry: Interrupt: Refer to Section 10.1.2: Interrupt and exception vectors If WFE was used for entry Wakeup event: Refer to Section 10.2.3: Wakeup event management
Wakeup latency	None

Table 13. Sleep-on-exit

Sleep-on-exit	Description
Mode entry	WFI (wait for interrupt) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 Refer to the Cortex®-M3 System Control register.
Mode exit	Interrupt: refer to Section 10.1.2: Interrupt and exception vectors .
Wakeup latency	None

5.3.4 Stop mode

The Stop mode is based on the Cortex®-M3 deepsleep mode combined with peripheral clock gating. The voltage regulator can be configured either in normal or low-power mode. In Stop mode, all clocks in the 1.8 V domain are stopped, the PLL, the HSI and the HSE RC oscillators are disabled. SRAM and register contents are preserved.

In the Stop mode, all I/O pins keep the same state as in the Run mode.

Entering Stop mode

Refer to [Table 14](#) for details on how to enter the Stop mode.

To further reduce power consumption in Stop mode, the internal voltage regulator can be put in low-power mode. This is configured by the LPDS bit of the [Power control register \(PWR_CR\)](#).

If Flash memory programming is ongoing, the Stop mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, The Stop mode entry is delayed until the APB access is finished.

In Stop mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a Reset. See [Section 19.3: IWDG functional description](#).
- Real-time clock (RTC): this is configured by the RTCEN bit in the [Backup domain control register \(RCC_BDCR\)](#)
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the [Control/status register \(RCC_CSR\)](#).
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the [Backup domain control register \(RCC_BDCR\)](#).

The ADC or DAC can also consume power during the Stop mode, unless they are disabled before entering it. To disable them, the ADON bit in the ADC_CR2 register and the ENx bit in the DAC_CR register must both be written to 0.

Note: *If the application needs to disable the external clock before entering Stop mode, the HSEON bit must first be disabled and the system clock switched to HSI. Otherwise, if the HSEON bit remains enabled and the external clock (external oscillator) is removed when entering Stop mode, the clock security system (CSS) feature must be enabled to detect any external oscillator failure and avoid a malfunction behavior when entering stop mode.*

Exiting Stop mode

Refer to [Table 14](#) for more details on how to exit Stop mode.

When exiting Stop mode by issuing an interrupt or a wakeup event, the HSI RC oscillator is selected as system clock.

When the voltage regulator operates in low-power mode, an additional startup delay is incurred when waking up from Stop mode. By keeping the internal regulator ON during Stop mode, the consumption is higher although the startup time is reduced.

Table 14. Stop mode

Stop mode	Description
Mode entry	<p>WFI (Wait for Interrupt) or WFE (Wait for Event) while:</p> <ul style="list-style-type: none"> – Set SLEEPDEEP bit in Cortex®-M3 System Control register – Clear PDDS bit in Power Control register (PWR_CR) – Select the voltage regulator mode by configuring LPDS bit in PWR_CR <p>Note: To enter Stop mode, all EXTI Line pending bits (in Pending register (EXTI_PR)), all peripheral interrupt pending bits, and RTC Alarm flag must be reset. Otherwise, the Stop mode entry procedure is ignored and program execution continues.</p>
Mode exit	<p>If WFI was used for entry:</p> <p>Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). Refer to Section 10.1.2: Interrupt and exception vectors.</p> <p>If WFE was used for entry:</p> <p>Any EXTI Line configured in event mode. Refer to Section 10.2.3: Wakeup event management</p>
Wakeup latency	HSI RC wakeup time + regulator wakeup time from Low-power mode

5.3.5 Standby mode

The Standby mode allows to achieve the lowest power consumption. It is based on the Cortex®-M3 deepsleep mode, with the voltage regulator disabled. The 1.8 V domain is consequently powered off. The PLL, the HSI oscillator and the HSE oscillator are also switched off. SRAM and register contents are lost except for registers in the Backup domain and Standby circuitry (see [Figure 4](#)).

Entering Standby mode

Refer to [Table 15](#) for more details on how to enter Standby mode.

In Standby mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by a reset. See [Section 19.3: IWDG functional description](#).
- Real-time clock (RTC): this is configured by the RTCEN bit in the Backup domain control register (RCC_BDCR)
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the Control/status register (RCC_CSR).
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the Backup domain control register (RCC_BDCR)

Exiting Standby mode

The microcontroller exits the Standby mode when an external reset (NRST pin), an IWDG reset, a rising edge on the WKUP pin or the rising edge of an RTC alarm occurs (see [Figure 179: RTC simplified block diagram](#)). All registers are reset after wakeup from Standby except for [Power control/status register \(PWR_CSR\)](#).

After waking up from Standby mode, program execution restarts in the same way as after a Reset (boot pins sampling, vector reset is fetched, etc.). The SBF status flag in the [Power control/status register \(PWR_CSR\)](#) indicates that the MCU was in Standby mode.

Refer to [Table 15](#) for more details on how to exit Standby mode.

Table 15. Standby mode

Standby mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – Set SLEEPDEEP in Cortex®-M3 System Control register – Set PDDS bit in Power Control register (PWR_CR) – Clear WUF bit in Power Control/Status register (PWR_CSR) – No interrupt (for WFI) or event (for WFI) is pending
Mode exit	WKUP pin rising edge, RTC alarm event's rising edge, external Reset in NRST pin, IWDG Reset.
Wakeup latency	Reset phase

I/O states in Standby mode

In Standby mode, all I/O pins are high impedance except:

- Reset pad (still available)
- TAMPER pin if configured for tamper or calibration out
- WKUP pin, if enabled

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop or Standby mode while the debug features are used. This is due to the fact that the Cortex®-M3 core is no longer clocked.

However, by setting some configuration bits in the DBGMCU_CR register, the software can be debugged even when using the low-power modes extensively. For more details, refer to [Section 31.16.1: Debug support for low-power modes](#).

5.3.6

Auto-wakeup (AWU) from low-power mode

The RTC can be used to wakeup the MCU from low-power mode without depending on an external interrupt (Auto-wakeup mode). The RTC provides a programmable time base for waking up from Stop or Standby mode at regular intervals. For this purpose, two of the three alternative RTC clock sources can be selected by programming the RTCSEL[1:0] bits in the [Backup domain control register \(RCC_BDCR\)](#):

- Low-power 32.768 kHz external crystal oscillator (LSE OSC).
This clock source provides a precise time base with very low-power consumption (less than 1µA added consumption in typical conditions)
- Low-power internal RC Oscillator (LSI RC)
This clock source has the advantage of saving the cost of the 32.768 kHz crystal. This internal RC Oscillator is designed to add minimum power consumption.

To wakeup from Stop mode with an RTC alarm event, it is necessary to:

- Configure the EXTI Line 17 to be sensitive to rising edge
- Configure the RTC to generate the RTC alarm

To wakeup from Standby mode, there is no need to configure the EXTI Line 17.

5.4

Power control registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

5.4.1

Power control register (PWR_CR)

Address offset: 0x000

Reset value: 0x0000 0000 (reset by wakeup from Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
							rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

Bits 31:9 Reserved, must be kept at reset value..

Bit 8 **DBP**: Disable backup domain write protection.

In reset state, the RTC and backup registers are protected against parasitic write access.

This bit must be set to enable write access to these registers.

0: Access to RTC and Backup registers disabled

1: Access to RTC and Backup registers enabled

Note: If the HSE divided by 128 is used as the RTC clock, this bit must remain set to 1.

Bits 7:5 **PLS[2:0]**: PVD level selection.

These bits are written by software to select the voltage threshold detected by the programmable voltage detector

000: 2.2V

001: 2.3V

010: 2.4V

011: 2.5V

100: 2.6V

101: 2.7V

110: 2.8V

111: 2.9V

Note: Refer to the electrical characteristics of the datasheet for more details.

Bit 4 **PVDE**: programmable voltage detector enable.

This bit is set and cleared by software.

0: PVD disabled

1: PVD enabled

Bit 3 **CSBF**: Clear standby flag.

This bit is always read as 0.

0: No effect

1: Clear the SBF Standby Flag (write).

Bit 2 **CWUF**: Clear wakeup flag.

This bit is always read as 0.

0: No effect

1: Clear the WUF Wakeup Flag **after 2 System clock cycles**. (write)

Bit 1 **PDSS**: Power down deepsleep.

This bit is set and cleared by software. It works together with the LPDS bit.

0: Enter Stop mode when the CPU enters Deepsleep. The regulator status depends on the LPDS bit.

1: Enter Standby mode when the CPU enters Deepsleep.

Bit 0 **LPDS**: Low-power deepsleep.

This bit is set and cleared by software. It works together with the PDSS bit.

0: Voltage regulator on during Stop mode

1: Voltage regulator in low-power mode during Stop mode

5.4.2 Power control/status register (PWR_CSR)

Address offset: 0x04

Reset value: 0x0000 0000 (not reset by wakeup from Standby mode)

Additional APB cycles are needed to read this register versus a standard APB read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved								EWUP	Reserved					PVDO	SBF	WUF
								rw						r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **EWUP**: Enable WKUP pin

This bit is set and cleared by software.

0: WKUP pin is used for general purpose I/O. An event on the WKUP pin does not wakeup the device from Standby mode.

1: WKUP pin is used for wakeup from Standby mode and forced in input pull down configuration (rising edge on WKUP pin wakes-up the system from Standby mode).

Note: This bit is reset by a system Reset.

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **PVDO**: PVD output

This bit is set and cleared by hardware. It is valid only if PVD is enabled by the PVDE bit.

0: V_{DD}/V_{DDA} is higher than the PVD threshold selected with the PLS[2:0] bits.

1: V_{DD}/V_{DDA} is lower than the PVD threshold selected with the PLS[2:0] bits.

Note: The PVD is stopped by Standby mode. For this reason, this bit is equal to 0 after Standby or reset until the PVDE bit is set.

Bit 1 **SBF**: Standby flag

This bit is set by hardware and cleared only by a POR/PDR (power on reset/power down reset) or by setting the CSBF bit in the [Power control register \(PWR_CR\)](#)

0: Device has not been in Standby mode

1: Device has been in Standby mode

Bit 0 **WUF**: Wakeup flag

This bit is set by hardware and cleared by hardware, by a system reset or by setting the CWUF bit in the [Power control register \(PWR_CR\)](#)

0: No wakeup event occurred

1: A wakeup event was received from the WKUP pin or from the RTC alarm

Note: An additional wakeup event is detected if the WKUP pin is enabled (by setting the EWUP bit) when the WKUP pin level is already high.

5.4.3 PWR register map

The following table summarizes the PWR registers.

Table 16. PWR register map and reset values

Offset	Register	31	30	29	28	27	26	25	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
0x000	PWR_CR	Reserved																		DBP	PLS [2:0]		PVDE	CSBF	CWUF	0	0	0	0				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x004	PWR_CSR	Reserved																		C_EWUP	Reserved		PVDO	SBF	WUF	0	0	0	0				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Refer to [Table 3 on page 50](#) for the register boundary addresses.

6 Backup registers (BKP)

Low-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density devices are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

XL-density devices are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 768 Kbytes and 1 Mbyte.

Connectivity line devices are STM32F105xx and STM32F107xx microcontrollers.

This section applies to the whole STM32F101xx family, unless otherwise specified.

6.1 BKP introduction

The backup registers are forty two 16-bit registers for storing 84 bytes of user application data.

They are implemented in the backup domain that remains powered on by V_{BAT} when the V_{DD} power is switched off. They are not reset when the device wakes up from Standby mode or by a system reset or power reset.

In addition, the BKP control registers are used to manage the Tamper detection feature and RTC calibration.

After reset, access to the Backup registers and RTC is disabled and the Backup domain (BKP) is protected against possible parasitic write access. To enable access to the Backup registers and the RTC, proceed as follows:

- enable the power and backup interface clocks by setting the PWREN and BKPN bits in the RCC_APB1ENR register
- set the DBP bit in the Power control register (PWR_CR) to enable access to the Backup registers and RTC.

6.2 BKP main features

- 20-byte data registers (in medium-density and low-density devices) or 84-byte data registers (in high-density, XL-density and connectivity line devices)
- Status/control register for managing tamper detection with interrupt capability
- Calibration register for storing the RTC calibration value
- Possibility to output the RTC Calibration Clock, RTC Alarm pulse or Second pulse on TAMPER pin PC13 (when this pin is not used for tamper detection)

6.3 BKP functional description

6.3.1 Tamper detection

The TAMPER pin generates a Tamper detection event when the pin changes from 0 to 1 or from 1 to 0 depending on the TPAL bit in the [Backup control register \(BKP_CR\)](#). A tamper detection event resets all data backup registers.

However to avoid losing Tamper events, the signal used for edge detection is logically ANDed with the Tamper enable in order to detect a Tamper event in case it occurs before the TAMPER pin is enabled.

- **When TPAL=0:** If the TAMPER pin is already high before it is enabled (by setting TPE bit), an extra Tamper event is detected as soon as the TAMPER pin is enabled (while there was no rising edge on the TAMPER pin after TPE was set)
- **When TPAL=1:** If the TAMPER pin is already low before it is enabled (by setting the TPE bit), an extra Tamper event is detected as soon as the TAMPER pin is enabled (while there was no falling edge on the TAMPER pin after TPE was set)

By setting the TPIE bit in the BKP_CSR register, an interrupt is generated when a Tamper detection event occurs.

After a Tamper event has been detected and cleared, the TAMPER pin should be disabled and then re-enabled with TPE before writing to the backup data registers (BKP_DRx) again. This prevents software from writing to the backup data registers (BKP_DRx), while the TAMPER pin value still indicates a Tamper detection. This is equivalent to a level detection on the TAMPER pin.

Note: *Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the data backup registers, the TAMPER pin should be externally tied to the correct level.*

6.3.2 RTC calibration

For measurement purposes, the RTC clock with a frequency divided by 64 can be output on the TAMPER pin. This is enabled by setting the CCO bit in the [RTC clock calibration register \(BKP_RTCCR\)](#).

The clock can be slowed down by up to 121 ppm by configuring CAL[6:0] bits.

For more details about RTC calibration and how to use it to improve timekeeping accuracy, refer to AN2604 "STM32F101xx and STM32F103xx RTC calibration".

6.4 BKP registers

Refer to [Section 2.2 on page 45](#) for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

6.4.1 Backup data register x (BKP_DRx) (x = 1 ..42)

Address offset: 0x04 to 0x28, 0x40 to 0xBC

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **D[15:0]** Backup data

These bits can be written with user data.

Note: The BKP_DRx registers are not reset by a System reset or Power reset or when the device wakes up from Standby mode.

They are reset by a Backup Domain reset or by a TAMPER pin event (if the TAMPER pin function is activated).

6.4.2 RTC clock calibration register (BKP_RTCCR)

Address offset: 0x2C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					ASOS	ASOE	CCO	CAL[6:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **ASOS:** Alarm or second output selection

When the ASOE bit is set, the ASOS bit can be used to select whether the signal output on the TAMPER pin is the RTC Second pulse signal or the Alarm pulse signal:

0: RTC Alarm pulse output selected

1: RTC Second pulse output selected

Note: This bit is reset only by a Backup domain reset.

Bit 8 **ASOE**: Alarm or second output enable

Setting this bit outputs either the RTC Alarm pulse signal or the Second pulse signal on the TAMPER pin depending on the ASOS bit.

The output pulse duration is one RTC clock period. The TAMPER pin must not be enabled while the ASOE bit is set.

Note: This bit is reset only by a Backup domain reset.

Bit 7 **CCO**: Calibration clock output

0: No effect

1: Setting this bit outputs the RTC clock with a frequency divided by 64 on the TAMPER pin. The TAMPER pin must not be enabled while the CCO bit is set in order to avoid unwanted Tamper detection.

Note: This bit is reset when the V_{DD} supply is powered off.

Bit 6:0 **CAL[6:0]**: Calibration value

This value indicates the number of clock pulses that will be ignored every 2²⁰ clock pulses. This allows the calibration of the RTC, slowing down the clock by steps of 1000000/2²⁰ PPM.

The clock of the RTC can be slowed down from 0 to 121PPM.

6.4.3 Backup control register (BKP_CR)

Address offset: 0x30

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														TPAL	TPE
														rw	rw

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **TPAL**: TAMPER pin active level

0: A high level on the TAMPER pin resets all data backup registers (if TPE bit is set).

1: A low level on the TAMPER pin resets all data backup registers (if TPE bit is set).

Bit 0 **TPE**: TAMPER pin enable

0: The TAMPER pin is free for general purpose I/O

1: Tamper alternate I/O function is activated.

Note: Setting the TPAL and TPE bits at the same time is always safe, however resetting both at the same time can generate a spurious Tamper event. For this reason it is recommended to change the TPAL bit only when the TPE bit is reset.

6.4.4 Backup control/status register (BKP_CSR)

Address offset: 0x34

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved						TIF	TEF	Reserved						TPIE	CTI	CTE
						r	r							rw	w	w

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 TIF: Tamper interrupt flag

This bit is set by hardware when a Tamper event is detected and the TPIE bit is set. It is cleared by writing 1 to the CTI bit (also clears the interrupt). It is also cleared if the TPIE bit is reset.

- 0: No Tamper interrupt
- 1: A Tamper interrupt occurred

Note: This bit is reset only by a system reset and wakeup from Standby mode.

Bit 8 TEF: Tamper event flag

This bit is set by hardware when a Tamper event is detected. It is cleared by writing 1 to the CTE bit.

- 0: No Tamper event
- 1: A Tamper event occurred

Note: A Tamper event resets all the BKP_DRx registers. They are held in reset as long as the TEF bit is set. If a write to the BKP_DRx registers is performed while this bit is set, the value will not be stored.

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 TPIE: TAMPER pin interrupt enable

- 0: Tamper interrupt disabled
- 1: Tamper interrupt enabled (the TPE bit must also be set in the BKP_CR register)

Note: A Tamper interrupt does not wake up the core from low-power modes.

This bit is reset only by a system reset and wakeup from Standby mode.

Bit 1 CTI: Clear tamper interrupt

This bit is write only, and is always read as 0.

- 0: No effect
- 1: Clear the Tamper interrupt and the TIF Tamper interrupt flag.

Bit 0 CTE: Clear tamper event

This bit is write only, and is always read as 0.

- 0: No effect
- 1: Reset the TEF Tamper event flag (and the Tamper detector)

6.4.5 BKP register map

BKP registers are mapped as 16-bit addressable registers as described in the table below:

Table 17. BKP register map and reset values

Offset	Register	31	30	29	28	27	26	25	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
0x00	Reserved																																
0x04	BKP_DR1	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Table 17. BKP register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	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				
0x08	BKP_DR2	Reserved												D[15:0]																							
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x0C	BKP_DR3	Reserved												D[15:0]																							
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x10	BKP_DR4	Reserved												D[15:0]																							
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x14	BKP_DR5	Reserved												D[15:0]																							
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x18	BKP_DR6	Reserved												D[15:0]																							
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x1C	BKP_DR7	Reserved												D[15:0]																							
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x20	BKP_DR8	Reserved												D[15:0]																							
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x24	BKP_DR9	Reserved												D[15:0]																							
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x28	BKP_DR10	Reserved												D[15:0]																							
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x2	BKP_RTCCR	Reserved												ASOS	ASOE	CCO	CAL[6:0]																				
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x30	BKP_CR	Reserved																																			
	Reset value																																				
0x34	BKP_CSR	Reserved																																			
	Reset value																																				
0x38		Reserved																																			
0x3C		Reserved																																			
0x40	BKP_DR11	Reserved												D[15:0]																							
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Table 17. BKP register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	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
0x44	BKP_DR12	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x48	BKP_DR13	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x4C	BKP_DR14	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x50	BKP_DR15	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x54	BKP_DR16	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x58	BKP_DR17	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x5C	BKP_DR18	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x60	BKP_DR19	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x64	BKP_DR20	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x68	BKP_DR21	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x6C	BKP_DR22	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x70	BKP_DR23	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x74	BKP_DR24	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x78	BKP_DR25	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Table 17. BKP register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	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
0x7C	BKP_DR26	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x80	BKP_DR27	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x84	BKP_DR28	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x88	BKP_DR29	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x8C	BKP_DR30	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x90	BKP_DR31	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x94	BKP_DR32	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x98	BKP_DR33	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x9C	BKP_DR34	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xA0	BKP_DR35	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xA4	BKP_DR36	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xA8	BKP_DR37	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xAC	BKP_DR38	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xB0	BKP_DR39	Reserved															D[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

18 Real-time clock (RTC)

Low-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density devices are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes.

XL-density devices are STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 768 Kbytes and 1 Mbyte.

Connectivity line devices are STM32F105xx and STM32F107xx microcontrollers.

This section applies to the whole STM32F10xxx family, unless otherwise specified.

18.1 RTC introduction

The real-time clock is an independent timer. The RTC provides a set of continuously running counters which can be used, with suitable software, to provide a clock-calendar function. The counter values can be written to set the current time/date of the system.

The RTC core and clock configuration (RCC_BDCR register) are in the Backup domain, which means that RTC setting and time are kept after reset or wakeup from Standby mode.

After reset, access to the Backup registers and RTC is disabled and the Backup domain (BKP) is protected against possible parasitic write access. To enable access to the Backup registers and the RTC, proceed as follows:

- enable the power and backup interface clocks by setting the PWREN and BKREN bits in the RCC_APB1ENR register
- set the DBP bit the Power Control register (PWR_CR) to enable access to the Backup registers and RTC.

18.2 RTC main features

- Programmable prescaler: division factor up to 2^{20}
- 32-bit programmable counter for long-term measurement
- Two separate clocks: PCLK1 for the APB1 interface and RTC clock (must be at least four times slower than the PCLK1 clock)
- The RTC clock source could be any of the following ones:
 - HSE clock divided by 128
 - LSE oscillator clock
 - LSI oscillator clock (refer to [Section 7.2.8: RTC clock](#) for details)
- Two separate reset types:
 - The APB1 interface is reset by system reset
 - The RTC Core (Prescaler, Alarm, Counter and Divider) is reset only by a Backup domain reset (see [Section 7.1.3: Backup domain reset](#)).
- Three dedicated maskable interrupt lines:
 - Alarm interrupt, for generating a software programmable alarm interrupt.
 - Seconds interrupt, for generating a periodic interrupt signal with a programmable period length (up to 1 second).
 - Overflow interrupt, to detect when the internal programmable counter rolls over to zero.

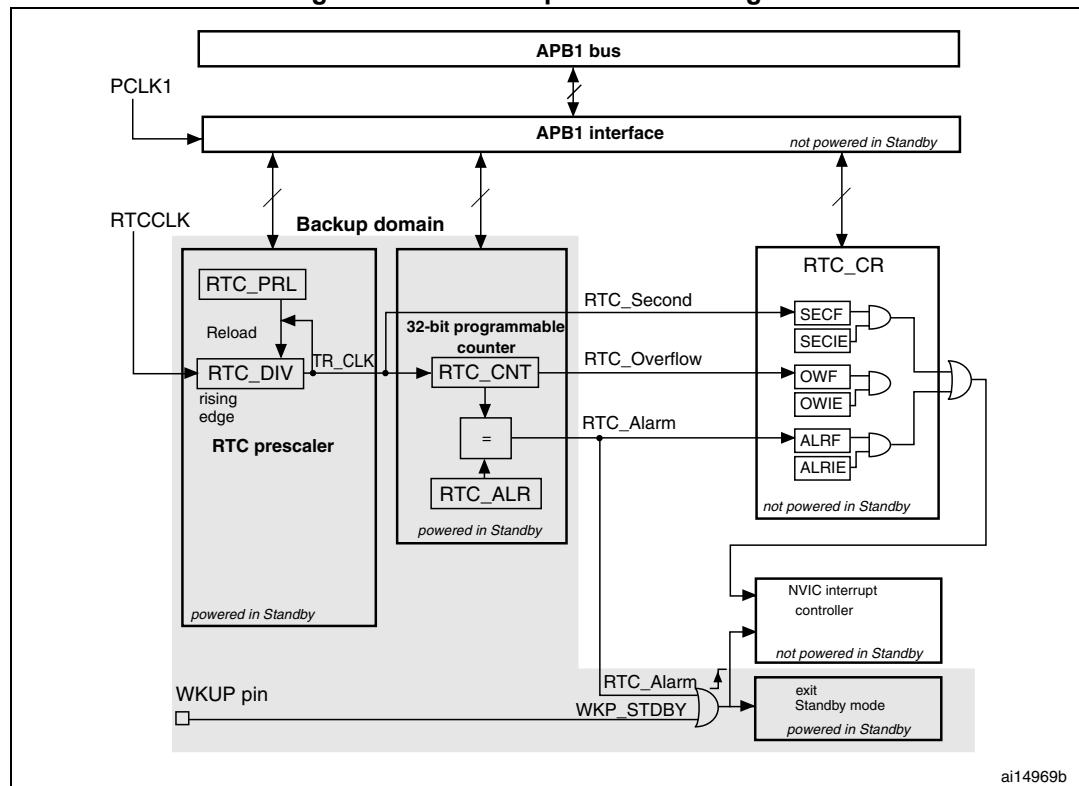
18.3 RTC functional description

18.3.1 Overview

The RTC consists of two main units (see [Figure 179](#)). The first one (APB1 Interface) is used to interface with the APB1 bus. This unit also contains a set of 16-bit registers accessible from the APB1 bus in read or write mode (for more information refer to [Section 18.4: RTC registers](#)). The APB1 interface is clocked by the APB1 bus clock in order to interface with the APB1 bus.

The other unit (RTC Core) consists of a chain of programmable counters made of two main blocks. The first block is the RTC prescaler block, which generates the RTC time base TR_CLK that can be programmed to have a period of up to 1 second. It includes a 20-bit programmable divider (RTC Prescaler). Every TR_CLK period, the RTC generates an interrupt (Second Interrupt) if it is enabled in the RTC_CR register. The second block is a 32-bit programmable counter that can be initialized to the current system time. The system time is incremented at the TR_CLK rate and compared with a programmable date (stored in the RTC_ALR register) in order to generate an alarm interrupt, if enabled in the RTC_CR control register.

Figure 179. RTC simplified block diagram



18.3.2 Resetting RTC registers

All system registers are asynchronously reset by a System Reset or Power Reset, except for RTC_PRL, RTC_ALR, RTC_CNT, and RTC_DIV.

The RTC_PRL, RTC_ALR, RTC_CNT, and RTC_DIV registers are reset only by a Backup Domain reset. Refer to [Section 7.1.3](#).

18.3.3 Reading RTC registers

The RTC core is completely independent from the RTC APB1 interface.

Software accesses the RTC prescaler, counter and alarm values through the APB1 interface but the associated readable registers are internally updated at each rising edge of the RTC clock resynchronized by the RTC APB1 clock. This is also true for the RTC flags.

This means that the first read to the RTC APB1 registers may be corrupted (generally read as 0) if the APB1 interface has previously been disabled and the read occurs immediately after the APB1 interface is enabled but before the first internal update of the registers. This can occur if:

- A system reset or power reset has occurred
- The MCU has just woken up from Standby mode (see [Section 5.3: Low-power modes](#))
- The MCU has just woken up from Stop mode (see [Section 5.3: Low-power modes](#))

In all the above cases, the RTC core has been kept running while the APB1 interface was disabled (reset, not clocked or unpowered).

Consequently when reading the RTC registers, after having disabled the RTC APB1 interface, the software must first wait for the RSF bit (Register Synchronized Flag) in the RTC_CRL register to be set by hardware.

Note that the RTC APB1 interface is not affected by WFI and WFE low-power modes.

18.3.4 Configuring RTC registers

To write in the RTC_PRL, RTC_CNT, RTC_ALR registers, the peripheral must enter Configuration mode. This is done by setting the CNF bit in the RTC_CRL register.

In addition, writing to any RTC register is only enabled if the previous write operation is finished. To enable the software to detect this situation, the RTOFF status bit is provided in the RTC_CR register to indicate that an update of the registers is in progress. A new value can be written to the RTC registers only when the RTOFF status bit value is '1'.

Configuration procedure

1. Poll RTOFF, wait until its value goes to '1'
2. Set the CNF bit to enter configuration mode
3. Write to one or more RTC registers
4. Clear the CNF bit to exit configuration mode
5. Poll RTOFF, wait until its value goes to '1' to check the end of the write operation.

The write operation only executes when the CNF bit is cleared; it takes at least three RTCCLK cycles to complete.

18.3.5 RTC flag assertion

The RTC Second flag (SECF) is asserted on each RTC Core clock cycle before the update of the RTC Counter.

The RTC Overflow flag (OWF) is asserted on the last RTC Core clock cycle before the counter reaches 0x0000.

The RTC_Alarm and RTC Alarm flag (ALRF) (see [Figure 180](#)) are asserted on the last RTC Core clock cycle before the counter reaches the RTC Alarm value stored in the Alarm register increased by one ($\text{RTC_ALR} + 1$). The write operation in the RTC Alarm and RTC Second flag must be synchronized by using one of the following sequences:

- Use the RTC Alarm interrupt and inside the RTC interrupt routine, the RTC Alarm and/or RTC Counter registers are updated.
- Wait for SECF bit to be set in the RTC Control register. Update the RTC Alarm and/or the RTC Counter register.

Figure 180. RTC second and alarm waveform example with PR=0003, ALARM=00004

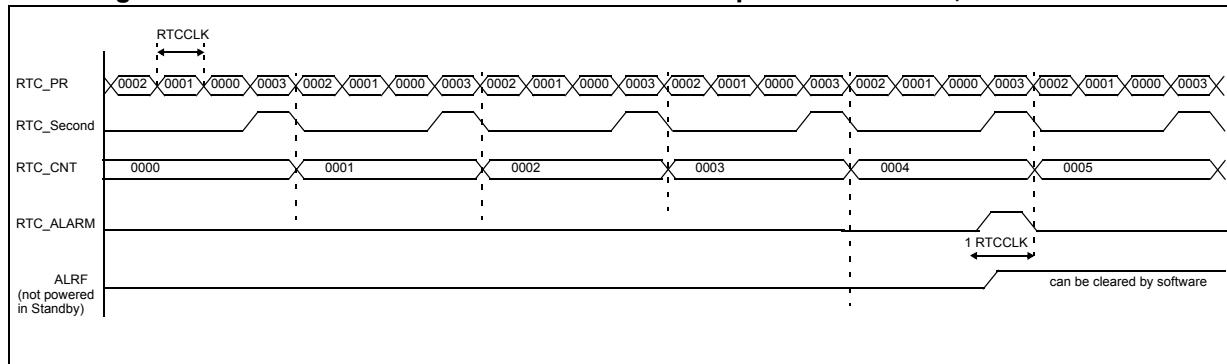
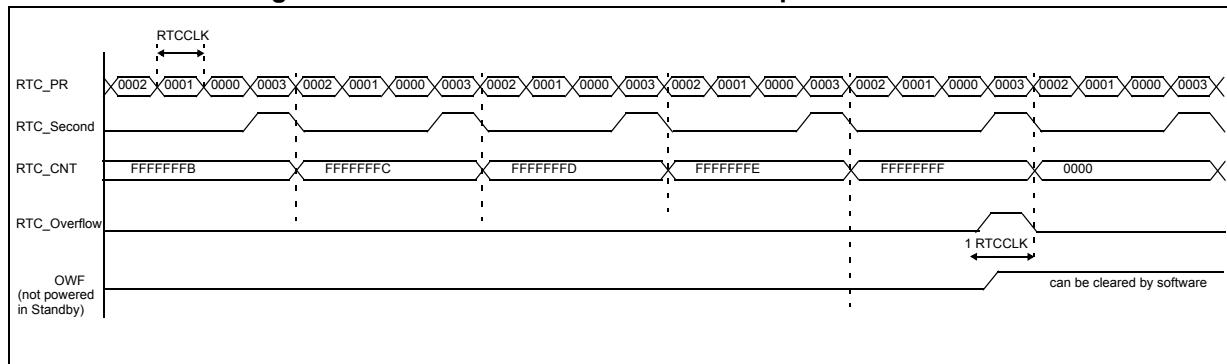


Figure 181. RTC Overflow waveform example with PR=0003



18.4 RTC registers

Refer to [Section 2.2 on page 45](#) for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

18.4.1 RTC control register high (RTC_CRH)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved														OWIE	ALRIE	SECIE
												rw	rw	rw		

Bits 15:3 Reserved, forced by hardware to 0.

Bit 2 **OWIE**: Overflow interrupt enable

- 0: Overflow interrupt is masked.
- 1: Overflow interrupt is enabled.

Bit 1 **ALRIE**: Alarm interrupt enable

- 0: Alarm interrupt is masked.
- 1: Alarm interrupt is enabled.

Bit 0 **SECIE**: Second interrupt enable

- 0: Second interrupt is masked.
- 1: Second interrupt is enabled.

These bits are used to mask interrupt requests. Note that at reset all interrupts are disabled, so it is possible to write to the RTC registers to ensure that no interrupt requests are pending after initialization. It is not possible to write to the RTC_CRH register when the peripheral is completing a previous write operation (flagged by RTOFF=0, see [Section 18.3.4](#)).

The RTC functions are controlled by this control register. Some bits must be written using a specific configuration procedure (see [Configuration procedure](#)).

18.4.2 RTC control register low (RTC_CRL)

Address offset: 0x04

Reset value: 0x0020

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RTOFF	CNF	RSF	OWF	ALRF	SECF		
								r	rw	rc_w0	rc_w0	rc_w0	rc_w0		

Bits 15:6 Reserved, forced by hardware to 0.

Bit 5 **RTOFF**: RTC operation OFF

With this bit the RTC reports the status of the last write operation performed on its registers, indicating if it has been completed or not. If its value is '0' then it is not possible to write to any of the RTC registers. This bit is read only.

- 0: Last write operation on RTC registers is still ongoing.
- 1: Last write operation on RTC registers terminated.

Bit 4 **CNF**: Configuration flag

This bit must be set by software to enter in configuration mode so as to allow new values to be written in the RTC_CNT, RTC_ALR or RTC_PRL registers. The write operation is only executed when the CNF bit is reset by software after has been set.

- 0: Exit configuration mode (start update of RTC registers).
- 1: Enter configuration mode.

Bit 3 **RSF**: Registers synchronized flag

This bit is set by hardware at each time the RTC_CNT and RTC_DIV registers are updated and cleared by software. Before any read operation after an APB1 reset or an APB1 clock stop, this bit must be cleared by software, and the user application must wait until it is set to be sure that the RTC_CNT, RTC_ALR or RTC_PRL registers are synchronized.

- 0: Registers not yet synchronized.
- 1: Registers synchronized.

Bit 2 **OWF**: Overflow flag

This bit is set by hardware when the 32-bit programmable counter overflows. An interrupt is generated if OWIE=1 in the RTC_CRH register. It can be cleared only by software. Writing '1' has no effect.

- 0: Overflow not detected
- 1: 32-bit programmable counter overflow occurred.

Bit 1 **ALRF**: Alarm flag

This bit is set by hardware when the 32-bit programmable counter reaches the threshold set in the RTC_ALR register. An interrupt is generated if ALRIE=1 in the RTC_CRH register. It can be cleared only by software. Writing '1' has no effect.

- 0: Alarm not detected
- 1: Alarm detected

Bit 0 **SECF**: Second flag

This bit is set by hardware when the 32-bit programmable prescaler overflows, thus incrementing the RTC counter. Hence this flag provides a periodic signal with a period corresponding to the resolution programmed for the RTC counter (usually one second). An interrupt is generated if SECIE=1 in the RTC_CRH register. It can be cleared only by software. Writing '1' has no effect.

- 0: Second flag condition not met.
- 1: Second flag condition met.

The functions of the RTC are controlled by this control register. It is not possible to write to the RTC_CR register while the peripheral is completing a previous write operation (flagged by RTOFF=0, see [Section 18.3.4: Configuring RTC registers](#)).

Note: Any flag remains pending until the appropriate RTC_CR request bit is reset by software, indicating that the interrupt request has been granted.

At reset the interrupts are disabled, no interrupt requests are pending and it is possible to write to the RTC registers.

The OWF, ALRF, SECF and RSF bits are not updated when the APB1 clock is not running.

The OWF, ALRF, SECF and RSF bits can only be set by hardware and only cleared by software.

If ALRF = 1 and ALRIE = 1, the RTC global interrupt is enabled. If EXTI Line 17 is also enabled through the EXTI Controller, both the RTC global interrupt and the RTC Alarm interrupt are enabled.

If ALRF = 1, the RTC Alarm interrupt is enabled if EXTI Line 17 is enabled through the EXTI Controller in interrupt mode. When the EXTI Line 17 is enabled in event mode, a pulse is generated on this line (no RTC Alarm interrupt generation).

18.4.3 RTC prescaler load register (RTC_PRLH / RTC_PRL)

The Prescaler Load registers keep the period counting value of the RTC prescaler. They are write-protected by the RTOFF bit in the RTC_CR register, and a write operation is allowed if the RTOFF value is '1'.

RTC prescaler load register high (RTC_PRLH)

Address offset: 0x08

Write only (see [Section 18.3.4: Configuring RTC registers](#))

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										PRL[19:16]					
w	w	w	w												

Bits 15:4 Reserved, forced by hardware to 0.

Bits 3:0 **PRL[19:16]:** RTC prescaler reload value high

These bits are used to define the counter clock frequency according to the following formula:

$$f_{TR_CLK} = f_{RTCCCLK}/(PRL[19:0]+1)$$

RTC prescaler load register low (RTC_PRL)

Address offset: 0x0C

Write only (see [Section 18.3.4: Configuring RTC registers](#))

Reset value: 0x8000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRL[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 15:0 **PRL[15:0]**: RTC prescaler reload value low

These bits are used to define the counter clock frequency according to the following formula:

$$f_{TR_CLK} = f_{RTCCLK}/(PRL[19:0]+1)$$

Caution: The zero value is not recommended. RTC interrupts and flags cannot be asserted correctly.

Note: If the input clock frequency (f_{RTCCLK}) is 32.768 kHz, write 7FFFh in this register to get a signal period of 1 second.

18.4.4 RTC prescaler divider register (RTC_DIVH / RTC_DIVL)

During each period of TR_CLK, the counter inside the RTC prescaler is reloaded with the value stored in the RTC_PRL register. To get an accurate time measurement it is possible to read the current value of the prescaler counter, stored in the RTC_DIV register, without stopping it. This register is read-only and it is reloaded by hardware after any change in the RTC_PRL or RTC_CNT registers.

RTC prescaler divider register high (RTC_DIVH)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												RTC_DIV[19:16]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:4 Reserved

Bits 3:0 **RTC_DIV[19:16]**: RTC clock divider high

RTC prescaler divider register low (RTC_DIVL)

Address offset: 0x14

Reset value: 0x8000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_DIV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **RTC_DIV[15:0]**: RTC clock divider low

18.4.5 RTC counter register (RTC_CNTH / RTC_CNTL)

The RTC core has one 32-bit programmable counter, accessed through two 16-bit registers; the count rate is based on the TR_CLK time reference, generated by the prescaler.

RTC_CNT registers keep the counting value of this counter. They are write-protected by bit RTOFF in the RTC_CR register, and a write operation is allowed if the RTOFF value is '1'. A write operation on the upper (RTC_CNTH) or lower (RTC_CNTL) registers directly loads the corresponding programmable counter and reloads the RTC Prescaler. When reading, the current value in the counter (system date) is returned.

RTC counter register high (RTC_CNTH)

Address offset: 0x18

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_CNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **RTC_CNT[31:16]**: RTC counter high

Reading the RTC_CNTH register, the current value of the high part of the RTC Counter register is returned. To write to this register it is necessary to enter configuration mode (see [Section 18.3.4: Configuring RTC registers](#)).

RTC counter register low (RTC_CNTL)

Address offset: 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **RTC_CNT[15:0]**: RTC counter low

Reading the RTC_CNTL register, the current value of the lower part of the RTC Counter register is returned. To write to this register it is necessary to enter configuration mode (see [Section 18.3.4: Configuring RTC registers](#)).

18.4.6 RTC alarm register high (RTC_ALRH / RTC_ALRL)

When the programmable counter reaches the 32-bit value stored in the RTC_ALR register, an alarm is triggered and the RTC_alarmIT interrupt request is generated. This register is write-protected by the RTOFF bit in the RTC_CR register, and a write operation is allowed if the RTOFF value is '1'.

RTC alarm register high (RTC_ALRH)

Address offset: 0x20

Write only (see [Section 18.3.4: Configuring RTC registers](#))

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_ALR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 15:0 **RTC_ALR[31:16]:** RTC alarm high

The high part of the alarm time is written by software in this register. To write to this register it is necessary to enter configuration mode (see [Section 18.3.4: Configuring RTC registers](#)).

RTC alarm register low (RTC_ALRL)

Address offset: 0x24

Write only (see [Section 18.3.4: Configuring RTC registers](#))

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_ALR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 15:0 **RTC_ALR[15:0]:** RTC alarm low

The low part of the alarm time is written by software in this register. To write to this register it is necessary to enter configuration mode (see [Section 18.3.4: Configuring RTC registers](#)).

18.4.7 RTC register map

RTC registers are mapped as 16-bit addressable registers as described in the table below:

Table 95. RTC register map and reset values

Refer to [Table 3 on page 50](#) for the register boundary addresses.

19 Independent watchdog (IWDG)

Low-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density devices are *STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes*.

XL-density devices are *STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 768 Kbytes and 1 Mbyte*.

Connectivity line devices are STM32F105xx and STM32F107xx microcontrollers.

This section applies to the whole STM32F10xxx family, unless otherwise specified.

19.1 IWDG introduction

The devices have two embedded watchdog peripherals which offer a combination of high safety level, timing accuracy and flexibility of use. Both watchdog peripherals (Independent and Window) serve to detect and resolve malfunctions due to software failure, and to trigger system reset or an interrupt (window watchdog only) when the counter reaches a given timeout value.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails. The window watchdog (WWDG) clock is prescaled from the APB1 clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The IWDG is best suited to applications which require the watchdog to run as a totally independent process outside the main application, but have lower timing accuracy constraints. The WWDG is best suited to applications which require the watchdog to react within an accurate timing window. For further information on the window watchdog, refer to [Section 20 on page 500](#).

19.2 IWDG main features

- Free-running downcounter
- clocked from an independent RC oscillator (can operate in Standby and Stop modes)
- Reset (if watchdog activated) when the downcounter value of 0x000 is reached

19.3 IWDG functional description

[Figure 182](#) shows the functional blocks of the independent watchdog module.

When the independent watchdog is started by writing the value 0xCCCC in the Key register (IWDG_KR), the counter starts counting down from the reset value of 0xFFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 0xAAAA is written in the IWDG_KR register, the IWDG_RLR value is reloaded in the counter and the watchdog reset is prevented.

19.3.1 Hardware watchdog

If the “Hardware watchdog” feature is enabled through the device option bits, the watchdog is automatically enabled at power-on, and will generate a reset unless the Key register is written by the software before the counter reaches end of count.

19.3.2 Register access protection

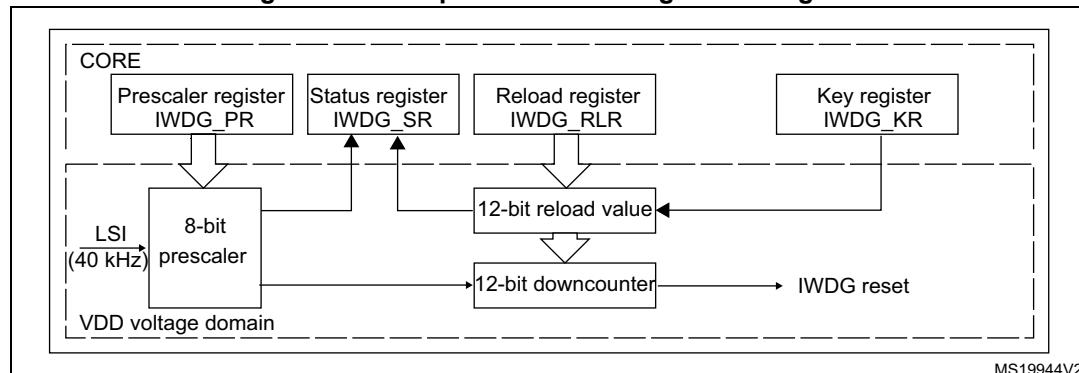
Write access to the IWDG_PR and IWDG_RLR registers is protected. To modify them, first write the code 0x5555 in the IWDG_KR register. A write access to this register with a different value will break the sequence and register access will be protected again. This implies that it is the case of the reload operation (writing 0xAAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value is on going.

19.3.3 Debug mode

When the microcontroller enters debug mode (Cortex®-M3 core halted), the IWDG counter either continues to work normally or stops, depending on DBG_IWDG_STOP configuration bit in DBG module. For more details, refer to [Section 31.16.2: Debug support for timers, watchdog, bxCAN and I²C](#).

Figure 182. Independent watchdog block diagram



Note: The watchdog function is implemented in the V_{DD} voltage domain, still functional in Stop and Standby modes.

Table 96. Min/max IWDG timeout period (in ms) at 40 kHz (LSI)⁽¹⁾

Prescaler divider	PR[2:0] bits	Min timeout RL[11:0]= 0x000	Max timeout RL[11:0]= 0xFFFF
/4	0	0.1	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6

Table 96. Min/max IWDG timeout period (in ms) at 40 kHz (LSI)⁽¹⁾ (continued)

Prescaler divider	PR[2:0] bits	Min timeout RL[11:0]= 0x000	Max timeout RL[11:0]= 0xFFFF
/128	5	3.2	13107.2
/256	6 (or 7)	6.4	26214.4

1. These timings are given for a 40 kHz clock but the microcontroller internal RC frequency can vary. Refer to the LSI oscillator characteristics table in the device datasheet for maximum and minimum values.

The LSI can be calibrated so as to compute the IWDG timeout with an acceptable accuracy. For more details refer to [Section 7.2.5: LSI clock](#).

19.4 IWDG registers

Refer to [Section 2.2 on page 45](#) for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by half-words (16 bits) or words (32 bits).

19.4.1 Key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by Standby mode)

31	30	29	28	27	26	25	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
Reserved															KEY[15:0]																

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]**: Key value (write only, read 0000h)

These bits must be written by software at regular intervals with the key value AAAAh, otherwise the watchdog generates a reset when the counter reaches 0.

Writing the key value 5555h to enable access to the IWDG_PR and IWDG_RLR registers (see [Section 19.3.2](#))

Writing the key value CCCCh starts the watchdog (except if the hardware watchdog option is selected)

19.4.2 Prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	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
Reserved															PR[2:0]																

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **PR[2:0]**: Prescaler divider

These bits are write access protected see [Section 19.3.2](#). They are written by software to select the prescaler divider feeding the counter clock. PVU bit of IWDG_SR must be reset in order to be able to change the prescaler divider.

- 000: divider /4
- 001: divider /8
- 010: divider /16
- 011: divider /32
- 100: divider /64
- 101: divider /128
- 110: divider /256
- 111: divider /256

Note: Reading this register returns the prescaler value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG_SR register is reset.

19.4.3 Reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF (reset by Standby mode)

31	30	29	28	27	26	25	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
Reserved															RL[11:0]																

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RL[11:0]**: Watchdog counter reload value

These bits are write access protected see [Section 19.3.2](#). They are written by software to define the value to be loaded in the watchdog counter each time the value AAAAh 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 96](#).

The RVU bit in the IWDG_SR register must be reset in order to be able to change the reload value.

Note: Reading this register returns the reload value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing on this register. For this reason the value read from this register is valid only when the RVU bit in the IWDG_SR register is reset.

19.4.4 Status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (not reset by Standby mode)

31	30	29	28	27	26	25	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
Reserved															RVU PVU		r	r													

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **RVU**: Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Reload value can be updated only when RVU bit is reset.

Bit 0 **PVU**: Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain (takes up to 5 RC 40 kHz cycles).

Prescaler value can be updated only when PVU bit is reset.

Note: *If several reload values or prescaler values are used by application, it is mandatory to wait until RVU bit is reset before changing the reload value and to wait until PVU bit is reset before changing the prescaler value. However, after updating the prescaler and/or the reload value it is not necessary to wait until RVU or PVU is reset before continuing code execution (even in case of low-power mode entry, the write operation is taken into account and will complete)*

19.4.5 IWDG register map

The following table gives the IWDG register map and reset values.

Table 97. IWDG register map and reset values

Offset	Register	31	30	29	28	27	26	25	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
0x00	IWDG_KR	Reserved										KEY[15:0]										0	0	0	0	0	0	0	0	0	0		
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x04	IWDG_PR	Reserved										PR[2:0]										0	0	0	0	0	0	0	0	0	0		
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x08	IWDG_RLR	Reserved										RL[11:0]										1	1	1	1	1	1	1	1	1	1		
	Reset value											1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
0x0C	IWDG_SR	Reserved										RVU										0	0	0	0	0	0	0	0	0	0		
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Refer to [Section 3.3: Memory map](#) for the register boundary addresses.

20 Window watchdog (WWDG)

Low-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 16 and 32 Kbytes.

Medium-density devices are STM32F101xx, STM32F102xx and STM32F103xx microcontrollers where the Flash memory density ranges between 64 and 128 Kbytes.

High-density devices are *STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 256 and 512 Kbytes*.

XL-density devices are *STM32F101xx and STM32F103xx microcontrollers where the Flash memory density ranges between 768 Kbytes and 1 Mbyte*.

Connectivity line devices are STM32F105xx and STM32F107xx microcontrollers.

This section applies to the whole STM32F10xxx family, unless otherwise specified.

20.1 WWDG 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.

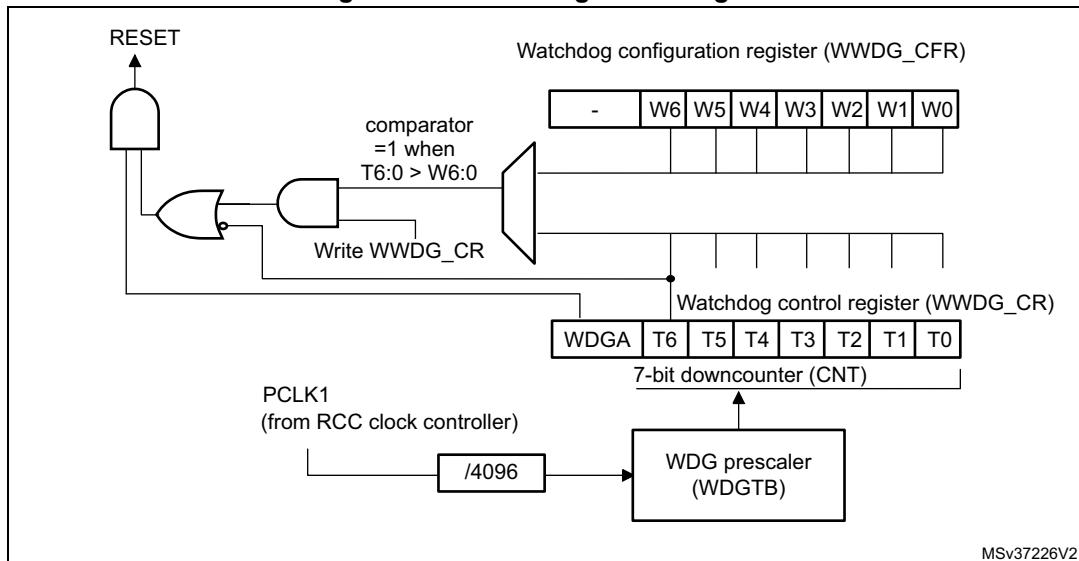
20.2 WWDG main features

- 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 184](#))
- Early wakeup interrupt (EWI): triggered (if enabled and the watchdog activated) when the downcounter is equal to 0x40.

20.3 WWDG functional description

If the watchdog is activated (the WDGA bit is set in the WWDG_CR register) and when the 7-bit downcounter (T[6:0] bits) rolls over from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

Figure 183. Watchdog block diagram



MSv37226V2

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

The watchdog is always 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.

Controlling the downcounter

This downcounter is free-running, counting 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 184](#)). The Configuration register (WWDG_CFR) 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 184](#) 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).

Advanced watchdog interrupt feature

The Early Wakeup Interrupt (EWI) can be used if specific safety operations or data logging must be performed before the actual reset is generated. The EWI interrupt is enabled by setting the EWI bit in the WWDG_CFR register. When the downcounter reaches the value 0x40, an EWI interrupt is generated and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.