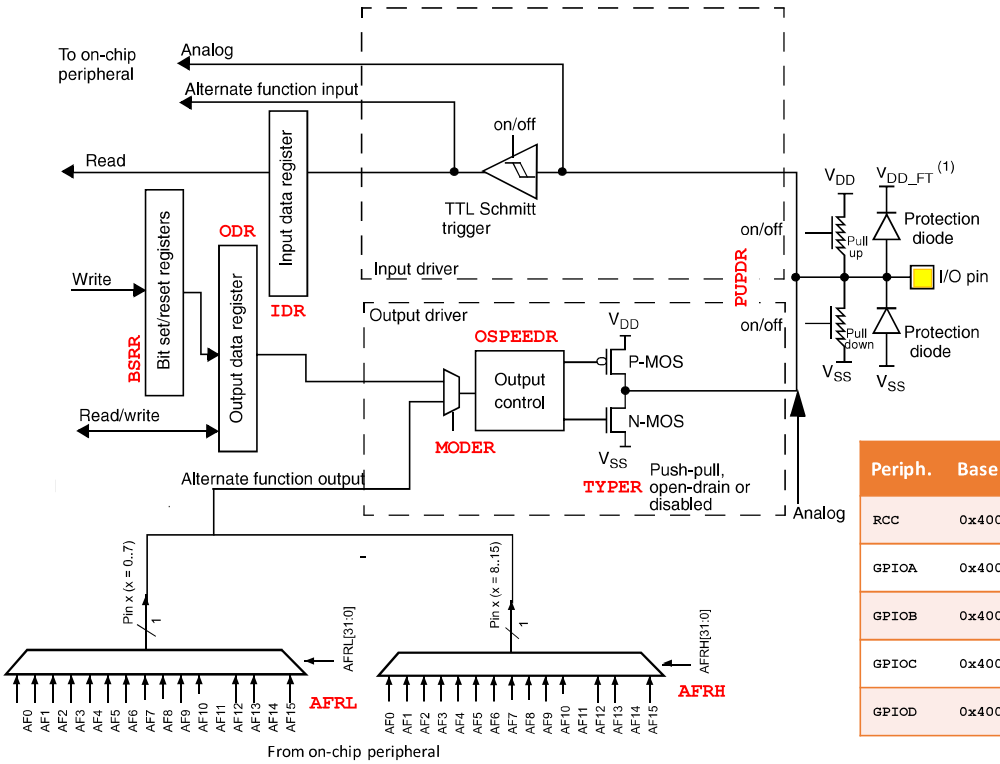
Reset and clock control

Register	Description	Mask
RCC_AHB1ENR	**RCC peripheral clock (bus AHB1) enable register,** RCC->AHB1ENR	
	OTGHSULPIEN: RCC_AHB1ENR[30]: USB OTG HSULPI clock enable	RCC_AHB1ENR_OTGHSULPIEN
	OTGHSEN: RCC_AHB1ENR[29]: USB OTG HS clock enable	RCC_AHB1ENR_OTGHSEN
	DMA2EN: RCC_AHB1ENR[22]: DMA2 clock enable	RCC_AHB1ENR_DMA1EN
	DMA1EN: RCC_AHB1ENR[21]: DMA1 clock enable	RCC_AHB1ENR_DMA2EN
	BKPSRAMEN: RCC_AHB1ENR[18]: Backup SRAM interface clock enable	RCC_AHB1ENR_BKPSRAMEN
	CRCEN: RCC_AHB1ENR[12]: CRC clock enable	RCC_AHB1ENR_CRCEN
	GPIOHEN: RCC_AHB1ENR[6]: IO port H clock enable	RCC_AHB1ENR_GPIOIEN
	GPIOGEN: RCC_AHB1ENR[5]: IO port G clock enable	RCC_AHB1ENR_GPIOHEN
	GPIOFEN: RCC_AHB1ENR[4]: IO port F clock enable	RCC_AHB1ENR_GPIOGEN
	GPIOEEN: RCC_AHB1ENR[4]: IO port E clock enable	RCC_AHB1ENR_GPIOFEN
	GPIODEN: RCC_AHB1ENR[3]: IO port D clock enable	RCC_AHB1ENR_GPIOEEN
	GPIOCEN: RCC_AHB1ENR[2]: IO port C clock enable	RCC_AHB1ENR_GPIODEN
	GPIOBEN: RCC_AHB1ENR[1]: IO port B clock enable	RCC_AHB1ENR_GPIOCEN
	GPIOAEN: RCC_AHB1ENR[0]: IO port A clock enable	RCC_AHB1ENR_GPIOBEN
	• 0: Considered peripheral **Clock Disable**	RCC_AHB1ENR_GPIOAEN
	• 1: Considered peripheral **Clock Enable**	
RCC_AHB2ENR	**RCC peripheral clock (bus AHB2) enable register,** RCC->AHB2ENR	
	OTGFSEN: RCC_AHB2ENR[7]: USB OTG FS clock enable	RCC_AHB2ENR_OTGFDEN
	DCMIEN: RCC_AHB2ENR[0]: Camera interface enable	RCC_AHB2ENR_DCMIEN
	• 0: Considered peripheral **Clock Disable**	
	• 1: Considered peripheral **Clock Enable**	
RCC_AHB3ENR	**RCC peripheral clock (bus AHB3) enable register,** RCC->AHB3ENR	
	QSPIEN: RCC_AHB1ENR[1]: QUADSPI memory controller module clock enable	RCC_AHB3ENR_QSPIEN
	FMCEN: RCC_AHB3ENR[0]: Flexible memory controller module clock enable	RCC_AHB3ENR_FMCEN
	• 0: Considered peripheral **Clock Disable**	
	• 1: Considered peripheral **Clock Enable**	
RCC_APB1ENR	**RCC peripheral clock (bus APB1) enable register,** RCC->AHB3ENR	
	DACEN: RCC_APB1ENR[29]: DAC interface clock enable	RCC_APB1ENR_DACEN
	PWREN: RCC_APB1ENR[28]: Power interface clock enable	RCC_APB1ENR_PWREN
	CECEN: RCC_APB1ENR[27]: CEC interface clock enable	RCC_APB1ENR_CECEN
	CAN2EN: RCC_APB1ENR[26]: CAN 2 clock enable	RCC_APB1ENR_CAN2EN
	CAN1EN: RCC_APB1ENR[25]: CAN 1 clock enable	RCC_APB1ENR_CAN1EN
	FMPI2C1EN: RCC_APB1ENR[24]: FMPI2C1 clock enable	
	I2C3EN: RCC_APB1ENR[23]: I2C3 clock enable	RCC_APB1ENR_FMPI2C1EN
	I2C2EN: RCC_APB1ENR[22]: I2C2 clock enable	RCC_APB1ENR_I2C3EN
	I2C1EN: RCC_APB1ENR[21]: I2C1 clock enable	RCC_APB1ENR_I2C3EN
	UART5EN: RCC_APB1ENR[20]: UART5 clock enable	RCC_APB1ENR_I2C1EN
	UART4EN: RCC_APB1ENR[19]: UART4 clock enable	RCC_APB1ENR_UART5EN
	USART3EN: RCC_APB1ENR[18]: USART3 clock enable	RCC_APB1ENR_UART4EN
	USART2EN: RCC_APB1ENR[17]: USART2 clock enable	RCC_APB1ENR_UART3EN
	SPDIFRXEN: RCC_APB1ENR[16]: SPDIF-Rx clock enable	RCC_APB1ENR_UART2EN
	SPI3EN: RCC_APB1ENR[15]: SPI3 clock enable	RCC_APB1ENR_SPDIFRXEN
	SPI2EN: RCC_APB1ENR[14]: SPI2 clock enable	RCC_APB1ENR_SPI3EN
	WWDGEN: RCC_APB1ENR[11]: Window watchdog clock enable	RCC_APB1ENR_SPI2EN
	TIM14EN: RCC_APB1ENR[8]: TIM14 clock enable	RCC_APB1ENR_WWDGEN
	TIM13EN: RCC_APB1ENR[7]: TIM13 clock enable	RCC_APB1ENR_TIM14EN
	TIM12EN: RCC_APB1ENR[6]: TIM12 clock enable	RCC_APB1ENR_TIM13EN
	TIM7EN: RCC_APB1ENR[5]: TIM7 clock enable	RCC_APB1ENR_TIM12EN
	TIM6EN: RCC_APB1ENR[4]: TIM6 clock enable	RCC_APB1ENR_TIM7EN
	TIM5EN: RCC_APB1ENR[3]: TIM5 clock enable	RCC_APB1ENR_TIM6EN
	TIM4EN: RCC_APB1ENR[2]: TIM4 clock enable	RCC_APB1ENR_TIM5EN
	TIM3EN: RCC_APB1ENR[1]: TIM3 clock enable	RCC_APB1ENR_TIM4EN
	TIM2EN: RCC_APB1ENR[0]: TIM2 clock enable	RCC_APB1ENR_TIM3EN
	• 0: Considered peripheral **Clock Disable**	RCC_APB1ENR_TIM2EN
	• 1: Considered peripheral **Clock Enable**	
RCC_APB2ENR	**RCC peripheral clock (bus APB2) enable register,**	
	SAI2EN: RCC_APB2ENR[23]: SAI2 clock enable	RCC_APB2ENR_SAI2EN
	SAI1EN: RCC_APB2ENR[22]: SAI1 clock enable	RCC_APB2ENR_SAI1EN
	TIM11EN: RCC_APB2ENR[18]: TIM11 clock enable	RCC_APB2ENR_TIM11EN
	TIM10EN: RCC_APB2ENR[17]: TIM10 clock enable	RCC_APB2ENR_TIM10EN
	TIM9EN: RCC_APB2ENR[16]: TIM9 clock enable	RCC_APB2ENR_TIM9EN
	SYSCFGEN: RCC_APB2ENR[14]: System configuration controller clock enable	RCC_APB2ENR_SYSCFGEN
	SPI4EN: RCC_APB2ENR[13]: SPI4 clock enable	RCC_APB2ENR_SPI4EN
	SPI1EN: RCC_APB2ENR[12]: SPI1 clock enable	RCC_APB2ENR_SPI1EN
	SDIOEN: RCC_APB2ENR[11]: SDIO clock enable	RCC_APB2ENR_SDIOEN
	ADC3EN: RCC_APB2ENR[10]: ADC3 clock enable	RCC_APB2ENR_ADC3EN
	ADC2EN: RCC_APB2ENR[9]: ADC2 clock enable	RCC_APB2ENR_ADC2EN
	ADC1EN: RCC_APB2ENR[8]: ADC1 clock enable	RCC_APB2ENR_ADC1EN
	USART6EN: RCC_APB2ENR[5]: USART6 clock enable	RCC_APB2ENR_USART6EN
	USART1EN: RCC_APB2ENR[4]: USART1 clock enable	RCC_APB2ENR_USART1EN
	TIM8EN: RCC_APB2ENR[]1: TIM8 clock enable	RCC_APB2ENR_TIM8EN
	TIM1EN: RCC_APB2ENR[0]: TIM1 clock enable	RCC_APB2ENR_TIM1EN
	• 0: Considered peripheral **Clock Disable**	
	• 1: Considered peripheral **Clock Enable**	

General Purpose Digital I/O



Periph.	Base Address
RCC	0x4002 3800
GPIOA	0x4002 0000
GPIOB	0x4002 0400
GPIOC	0x4002 0800
GPIOD	0x4002 0C00

Register	Offset
MODER	0x00
TYPER	0x04
OSPEED	0x08
PUPDR	0x0C
IDR	0x10
ODR	0x14
BSRR	0x18
LCKR	0x1C
AFRL	0x20
AFRH	0x24

Register	Description	Mask	R/W
GPIOx_MODER	**GPIO port mode register,** GPIOx->MODER		R/W
	MODERy[1:0]: GPIOx_MODER[2y+1:2y]: Configuration of Port x **direction** for bit y • 00: Input mode (reset state) GPIO_MODE_INPUT (0x00) • 01: Output mode GPIO_MODE_OUTPUT_PP (0x01) • 10: Alternate function mode GPIO_MODE_AF_PP (0x02) • 11: Analog mode GPIO_MODE_ANALOG (0x03)	GPIO_MODER_MODERy_Pos = 2y GPIO_MODER_MODERy_Msk = 11b<<2y GPIO_MODER_MODERy_0 = 01b<<2y GPIO_MODER_MODERy_1 = 10b<<2y	
GPIOx_TYPER	**GPIO port output type register,** GPIOx->OTYPER		R/W
	OTy: GPIOx_TYPER[y]: Configuration of Port x **output type** for bit y • 0: Output push-pull (reset state) • 1: Output open-drain	GPIO_OTYPER_OT_y = 01b << y	
GPIOx_PUPDR	**GPIO port pull-up/pull-down register,** GPIOx->PUPDR		R/W
	PUPDRy[1:0]: GPIOx_PUPDR[2y+1:2y]: Configuration of Port x **output pull-up or pull-down** for bit y • 00: No pull-up, pull-down GPIO_NOPULL (0x00) • 01: Pull-up GPIO_PULLUP (0x01) • 10: Pull-down GPIO_PULLDOWN (0x02) • 11: Reserved	GPIO_PUPDR_PUPDRy_Pos = 2y GPIO_PUPDR_PUPDRy_Msk = 11b<<2y GPIO_PUPDR_PUPDRy_0 = 01b<<2y GPIO_PUPDR_PUPDRy_1 = 10b<<2y	
GPIOx_OSPEEDR	**GPIO port output speed register,** GPIOx->OSPEEDR		R/W
	OSPEEDRy[1:0]: GPIOx_OSPEEDR[2y+1:2y]: Configuration of Port x **output speed** for • 00: 2 MHz Low speed GPIO_SPEED_FREQ_LOW (0x00) • 01: 25 MHz Medium speed GPIO_SPEED_FREQ_MEDIUM (0x01) • 10: 50 MHz Fast speed GPIO_SPEED_FREQ_HIGH (0x02) • 11: 100 MHz High speed on 30pF GPIO_SPEED_FREQ_VERY_HIGH (0x03)	GPIO_OSPEEDER_OSPEEDERy_Pos = 2y GPIO_OSPEEDER_OSPEEDERy_Msk=11b<<2y GPIO_OSPEEDER_OSPEEDERy_0 =01b<<2y GPIO_OSPEEDER_OSPEEDERy_1 =10b<<2y	
GPIOx_IDR	**GPIO port input data register,** GPIOx->IDR		R
	IDRy: GPIOx_IDR[y]: **input value** of bit y for Port x	GPIO_IDR_IDRy = 01b << y	
GPIOx_ODR	**GPIO port output data register,** GPIOx->ODR		R/W
	ODRy: GPIOx_ODR[y]: Configuration of **output value** for bit y of Port x	GPIO_ODR_ODRy = 01b << y	
GPIOx_BSRR	**GPIO port bit set/reset register,** GPIOx->BSRR		W
	BRy: GPIOx_BSRR[y+16]: Reset bit y of Port x • 0: no action • 1: **reset** the corresponding ODRx bit, **reset** the output bit y of Port x **BSy**: GPIOx_BSRR[y]: Set bit y of Port x	GPIO_BSRR_BSy = 01b << (y+16)	

	• 0: 0: no action • 1: **set** the corresponding ODRx bit, **set** the output bit y of Port x	GPIO_BSSR_BSy = 01b << y
GPIOx_AFRL	**GPIO function low register,** y ∈[0,7], GPIOx->AFR[0]	R/W
	AFRLy: GPIOx_AFRL[4y+3:4y]= **i** (4 bits), **selection** of function **i** for bit y of Port x	GPIO_AFRL_AFSELy_Pos = 4y GPIO_AFRL_AFSELy_Msk = 0xF<<4y
GPIOx_AFRH	**GPIO function high register** y ∈[8,15], GPIOx->AFR[1]	R/W
	AFRHy: GPIOx_AFRL[4(y-8)+3:4(y-8)]= **i** (4 bits), **selection** of function **i** for bit y of Port x	GPIO_AFRH_AFSELy_Pos = 4(y-8) GPIO_AFRH_AFSELy_Msk = 0xF<<4(y-8)
GPIOx_LCKR	**GPIO port configuration lock register** GPIOx->LCKR	W
	LCKy: GPIOx_LCKR[y]: These bits are R/W but can only be written when the LCKK bit is 0. • 0: Port configuration not locked • 1: Port configuration locked	GPIO_LCKR_LCKy = 01b << y
	LCKK: GPIOx_LCKR[16]: This bit can be read any time. It can only be modified using the lock key write sequence 1. WR LCKR[16] = '1' + LCKR[15:0] 2. WR LCKR[16] = '0' + LCKR[15:0] 3. WR LCKR[16] = '1' + LCKR[15:0] 4. RD LCKR LCKR[16] = '1' (this read operation is optional but it confirms that the lock is active)	GPIO_LCKR_LCKK = 01b << 16

Activation of the clock signal to enable each port GPIOx with register RCC_AHB1ENR

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved			CRCEN	Reserved			GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
Read/Write				rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

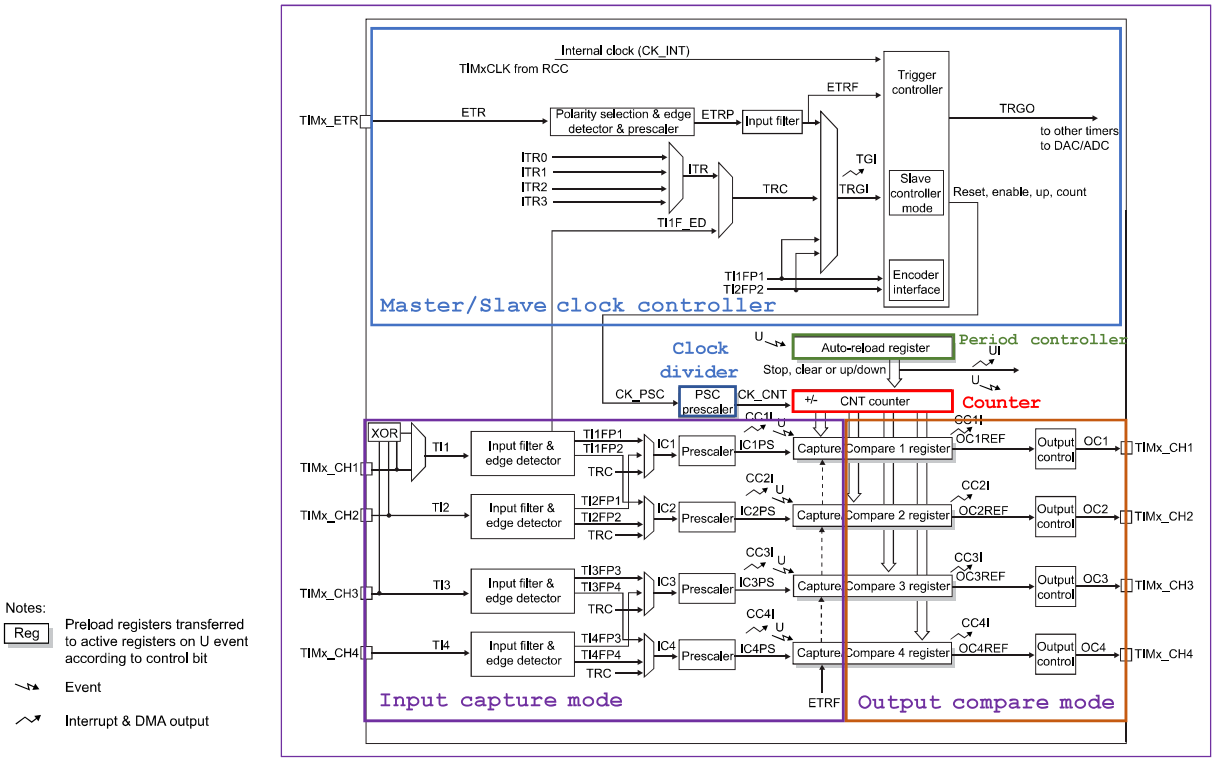
Access to a register in the peripheral by using CMSIS header file:

```
#define GPIOA_BASE       (0x40020000)
#define GPIOB_BASE       (0x40020400)
…
#define GPIOA            (GPIO_Type *)GPIOA_BASE)
#define GPIOB            (GPIO_Type *)GPIOB_BASE)
…
#define GPIO_PIN_0    ((uint16_t)0x0001)  /* Pin 0 selected    */
…
GPIOA->MODER =
```

Definition for Alternate Function Mapping

```
      /**   * @brief   AF 0 selection      */
#define GPIO_AF0_RTC_50Hz  ((uint8_t)0x00U)  /* RTC_50Hz */
#define GPIO_AF0_MCO       ((uint8_t)0x00U)  /* MCO (MCO1 and MCO2) */
#define GPIO_AF0_TAMPER    ((uint8_t)0x00U)  /* TAMPER (TAMPER_1 TAMPER_2)*/
#define GPIO_AF0_SWJ       ((uint8_t)0x00U)  /* SWJ (SWD and JTAG) */
#define GPIO_AF0_TRACE     ((uint8_t)0x00U)  /* TRACE  */

    /**     * @brief   AF 1 selection     */
#define GPIO_AF1_TIM1          ((uint8_t)0x01U)  /* TIM1 */
#define GPIO_AF1_TIM2          ((uint8_t)0x01U)  /* TIM2 */

    /**     * @brief   AF 2 selection     */
#define GPIO_AF2_TIM3          ((uint8_t)0x02U)  /* TIM3 */
#define GPIO_AF2_TIM4          ((uint8_t)0x02U)  /* TIM4 */
#define GPIO_AF2_TIM5          ((uint8_t)0x02U)  /* TIM5 */

    /**     * @brief   AF 3 selection     */
#define GPIO_AF3_TIM8          ((uint8_t)0x03U)  /* TIM8 */
#define GPIO_AF3_TIM9          ((uint8_t)0x03U)  /* TIM9 */
#define GPIO_AF3_TIM10         ((uint8_t)0x03U)  /* TIM10 */
#define GPIO_AF3_TIM11         ((uint8_t)0x03U)  /* TIM11 */
#define GPIO_AF3_CEC           ((uint8_t)0x03U)  /* CEC */

    /**     * @brief   AF 4 selection     */
#define GPIO_AF4_I2C1          ((uint8_t)0x04U)  /* I2C1 */
#define GPIO_AF4_I2C2          ((uint8_t)0x04U)  /* I2C2 */
#define GPIO_AF4_I2C3          ((uint8_t)0x04U)  /* I2C3 */
#define GPIO_AF4_FMPI2C1       ((uint8_t)0x04U)  /* FMPI2C1 */
#define GPIO_AF4_CEC           ((uint8_t)0x04U)  /* CEC */

    /**     * @brief   AF 5 selection     */
#define GPIO_AF5_SPI1          ((uint8_t)0x05U)  /* SPI1/I2S1 */
#define GPIO_AF5_SPI2          ((uint8_t)0x05U)  /* SPI2/I2S2 */
#define GPIO_AF5_SPI3          ((uint8_t)0x05U)  /* SPI3/I2S3 */
#define GPIO_AF5_SPI4          ((uint8_t)0x05U)  /* SPI4 */

    /**     * @brief   AF 6 selection     */
#define GPIO_AF6_SPI2          ((uint8_t)0x06U)  /* SPI2/I2S2 */
#define GPIO_AF6_SPI3          ((uint8_t)0x06U)  /* SPI3/I2S3 */
#define GPIO_AF6_SPI4          ((uint8_t)0x06U)  /* SPI4 */
#define GPIO_AF6_SAI1          ((uint8_t)0x06U)  /* SAI1 */

    /**     * @brief   AF 7 selection     */
#define GPIO_AF7_USART1        ((uint8_t)0x07U)  /* USART1 *
#define GPIO_AF7_USART2        ((uint8_t)0x07U)  /* USART2 */
#define GPIO_AF7_USART3        ((uint8_t)0x07U)  /* USART3 */
#define GPIO_AF7_UART5         ((uint8_t)0x07U)  /* UART5 */
#define GPIO_AF7_SPI2          ((uint8_t)0x07U)  /* SPI2/I2S2 */
#define GPIO_AF7_SPI3          ((uint8_t)0x07U)  /* SPI3/I2S3  */
#define GPIO_AF7_SPDIFRX       ((uint8_t)0x07U)  /* SPDIFRX      */

    /**     * @brief   AF 8 selection     */
#define GPIO_AF8_UART4         ((uint8_t)0x08U)  /* UART4  */
#define GPIO_AF8_UART5         ((uint8_t)0x08U)  /* UART5  */
#define GPIO_AF8_USART6        ((uint8_t)0x08U)  /* USART6     */
#define GPIO_AF8_SPDIFRX       ((uint8_t)0x08U)  /* SPDIFRX  */
#define GPIO_AF8_SAI2          ((uint8_t)0x08U)  /* SAI2       */

    /**     * @brief   AF 9 selection     */
#define GPIO_AF9_CAN1          ((uint8_t)0x09U)  /* CAN1  */
#define GPIO_AF9_CAN2          ((uint8_t)0x09U)  /* CAN2  */
#define GPIO_AF9_TIM12         ((uint8_t)0x09U)  /* TIM12     */
#define GPIO_AF9_TIM13         ((uint8_t)0x09U)  /* TIM13     */
#define GPIO_AF9_TIM14         ((uint8_t)0x09U)  /* TIM14     */
#define GPIO_AF9_QSPI          ((uint8_t)0x09U)  /* QSPI  */

    /**     * @brief   AF 10 selection     */
#define GPIO_AF10_OTG_FS        ((uint8_t)0x0AU)  /* OTG_FS     */
#define GPIO_AF10_OTG_HS        ((uint8_t)0x0AU)  /* OTG_HS     */
#define GPIO_AF10_SAI2          ((uint8_t)0x0AU)  /* SAI2       */
#define GPIO_AF10_QSPI          ((uint8_t)0x0AU)  /* QSPI  */

    /**     * @brief   AF 11 selection     */
#define GPIO_AF11_ETH           ((uint8_t)0x0BU)  /* ETHERNET     */

    /**     * @brief   AF 12 selection     */
#define GPIO_AF12_FMC           ((uint8_t)0x0CU)  /*
FMC                        */
#define GPIO_AF12_OTG_HS_FS     ((uint8_t)0x0CU)  /* OTG HS configured in
FS,     */
#define GPIO_AF12_SDIO          ((uint8_t)0x0CU)  /*
SDIO                       */

    /**     * @brief   AF 13 selection     */
#define GPIO_AF13_DCMI          ((uint8_t)0x0DU)  /* DCMI     */

    /**     * @brief   AF 15 selection     */
#define GPIO_AF15_EVENTOUT      ((uint8_t)0x0FU)  /* EVENTOUT     */
```

Timer TIM2 to TIM5



Description of the different steps to setup the timer TIMx

1. Main configuration

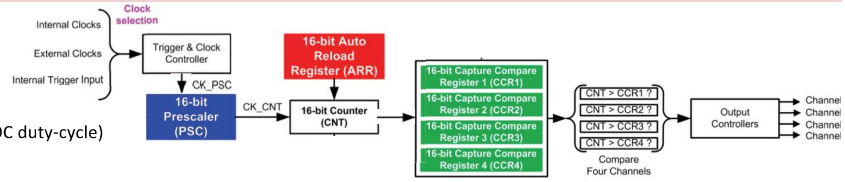
1. Enable peripheral Clocks for TIMx and GPIO via RCC module
2. Configure the GPIO pins (alternative function mode)
3. Configure the timer-counter (see part 2)
4. Configure the timer channels (see part 3 ou 4)
5. Enable the timer `CEN` in register `TIMx_CR1`

2. Timer configuration

1. Set the prescaler value in register `TIMx_PSC` to define the counter clock CK_CNT frequency
2. Set the auto-reload value in register `TIMx_ARR` to define the timer period
3. Configure the counter (mode `CMS`, direction `DIR`, preload `ARPE`) in register `TIMx_CR1`

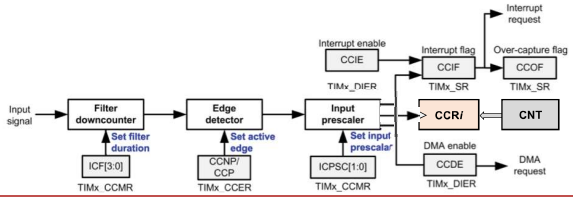
3. Configuration of output compare (OC) mode for channel *i*

1. Set the channel *i* as output: `CCiS[1:0]` in `TIMx_CCMRy`
2. Select the output compare mode: `OCiM[2:0]` – `TIMx_CCMRy`
3. Manage output preload `OCiPE` in register `TIMx_CCMRy`
4. Enable output compare: `CCiE` in register `TIMx_CCER`
5. Set the output compare value in register `TIMx_CCRi` (define OC duty-cycle)
6. Enable interrupt (if required): `CCiIE` in register `TIMx_DIER`
7. Enable DMA (if required): `CCiDE` in register `TIMx_DIER`



4. Configuration of input capture mode for channel *i*

1. Select the active input: `CCiS[1:0]` in register `TIMx_CCMRy`
2. Manage the input filter: `ICiF[3:0]` in register `TIMx_CCMRy`
3. Set the active edge: `CCiP`–`CCiNP` in register `TIMx_CCER`
4. Manage the input prescaler: `ICiPSC[1:0]` in register `TIMx_CCMRy`
5. Enable input capture: `CCiE` in register `TIMx_CCER`
6. Enable interrupt (if required): `CCiIE` in register `TIMx_DIER`
7. Enable DMA (if required): `CCiDE` in register `TIMx_DIER`



Register	Description	
TIMx_CR1	**TIMx control register 1**	
	CKD[1:0]: TIMx_CR1[9:8]: **Clock division**: division ratio between CK_INT frequency and sampling clock used by the digital filters (ETR, TIx) • 00: $t_{DTS} = t_{CK_INT}$ • 01: $t_{DTS} = 2 \times t_{CK_INT}$ • 10: $t_{DTS} = 4 \times t_{CK_INT}$ • 11: Reserved mode	**TIM_CR1_CKD** TIM_CLOCKDIVISION_DIV1 TIM_CLOCKDIVISION_DIV2 TIM_CLOCKDIVISION_DIV4 .
	ARPE: TIMx_CR1[7]: Auto-reload preload enable • 0: TIMx_ARR register is not buffered • 1: TIMx_ARR register is buffered	**TIM_CR1_ARPE**
	CMS[1:0]: TIMx_CR1[6:5]: Center-aligned mode selection • 00: Edge-aligned mode. • 01: Center-aligned mode 1. • 10: Center-aligned mode 2. • 11: Center-aligned mode 3.	**TIM_CR1_CMS** . TIM_COUNTERMODE_CENTERALIGNED1 TIM_COUNTERMODE_CENTERALIGNED2 TIM_COUNTERMODE_CENTERALIGNED3
	DIR: TIMx_CR1[4]: **Direction** • 0: Counter used as upcounter • 1: Counter used as downcounter	**TIM_CR1_DIR** TIM_COUNTERMODE_UP TIM_COUNTERMODE_DOWN
	OPM: TIMx_CR1[3]: One Pulse Mode • 0: Counter is not stopped at update event • 1: Counter stops counting at the next update event	**TIM_CR1_OPM** TIM_OPMODE_REPETITIVE TIM_OPMODE_SINGLE

	URS: TIMx_CR1[2]: Update request source
	• 0: Any of the following events generate an interrupt or DMA request
	  • Counter overflow/underflow (OVF/UNF)
	  • Setting the UG bit
	  • Update generation through the slave mode controller
	• 1: Only counter OVF/UNF generates an interrupt or DMA request.
		TIM_CR1_URS

	UDIS: TIMx_CR1[1]: Update disable • 0: UEV enabled • 1: UEV disabled	**TIM_CR1_UDIS**
	CEN: TIMx_CR1[0]: Counter enable • 0: Counter disabled • 1: Counter enabled	**TIM_CR1_CEN**
TIMx_CR2	**TIMx control register 2**	
	TI1S: TIMx_CR2[7]: TI1selection • 0: The TIMx_CH1 pin is connected to TI1 input • 1: XOR (TIMx_CH1, CH2, CH3) => TI1 input(XOR)	**TIM_CR2_TI1S** TIM_TI1SELECTION_CH1 TIM_TI1SELECTION_XORCOMBINATION
	MMS[2:0]: TIMx_CR2[6:4]: select the information to be sent in master mode to slave timers for synchronization (TRGO). • 000: Reset • 001: Enable - the Counter enable signal, CNT_EN, is used as trigger output (TRGO) • 010: Update - The update event is selected as trigger output • 011: Compare Pulse • 100: Compare - OC1REF signal is used as trigger output (TRGO) • 101: Compare - OC2REF signal is used as trigger output (TRGO) • 110: Compare - OC3REF signal is used as trigger output (TRGO) • 111: Compare - OC4REF signal is used as trigger output (TRGO)	**TIM_CR2_MMS** TIM_TRGO_RESET TIM_TRGO_ENABLE TIM_TRGO_UPDATE TIM_TRGO_OC1 TIM_TRGO_OC1REF TIM_TRGO_OC2REF TIM_TRGO_OC3REF TIM_TRGO_OC4REF
	CCDS: TIMx_CR2[3]: Capture/compare DMA selection • 0: CCx DMA request sent when CCx event occurs • 1: CCx DMA requests sent when update event occurs	**TIM_CR2_CCDS**
TIMx_SMCR	**TIMx slave mode control register**	
	ETP: TIMx_SMCR[15]: External trigger polarity • 0: ETR is noninverted, active at high level or rising edge • 1: ETR is inverted, active at low level or falling edge	**TIM_SMCR_ETP** TIM_ETRPOLARITY_INVERTED TIM_ETRPOLARITY_NONINVERTED
	ECE: TIMx_SMCR[14]: External clock enable • 0: External clock mode 2 disabled • 1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF	**TIM_SMCR_ECE**
	ETPS[2:0]: TIMx_SMCR[13:12]: External trigger prescaler • 00: Prescaler OFF • 01: ETRP frequency divided by 2 • 10: ETRP frequency divided by 4 • 11: ETRP frequency divided by 8	**TIM_SMCR_ETPS** TIM_CLOCKPRESCALER_DIV1 TIM_CLOCKPRESCALER_DIV2 TIM_CLOCKPRESCALER_DIV4 TIM_CLOCKPRESCALER_DIV8
	ETF[3:0]: TIMx_SMCR[11:8]: External trigger filter f_s: sampling frequency N: number of consecutive events to validate a transition	**TIM_SMCR_ETF**

ETF	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
f_s	No filter	f_{CK_INT}		K	f_{DTS}/K											
					2	2	4	4	8	8	16	16	16	32	32	32
N		2	4	8	6	8	6	8	6	8	5	6	8	5	6	8

MSM: TIMx_SMCR[7] Master/Slave mode • 0: No action • 1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO).	**TIM_SMCR_MSM**

TS: TIMx_SMCR[6:4]: Trigger selection to synchronize the counter — **TIM_SMCR_TS**

- 000: Internal Trigger 0 (ITR0) — TIM_CLOCKSOURCE_ITR0
- 001: Internal Trigger 1 (ITR1). — TIM_CLOCKSOURCE_ITR1
- 010: Internal Trigger 2 (ITR2). — TIM_CLOCKSOURCE_ITR2
- 011: Internal Trigger 3 (ITR3). — TIM_CLOCKSOURCE_ITR3
- 100: TI1 Edge Detector (TI1F_ED) — TIM_CLOCKSOURCE_TI1ED
- 101: Filtered Timer Input 1 (TI1FP1) — TIM_CLOCKSOURCE_TI1
- 110: Filtered Timer Input 2 (TI2FP2) — TIM_CLOCKSOURCE_TI2
- 111: External Trigger input (ETRF) — TIM_CLOCKSOURCE_ETRMODE1

SMS[2:0]: TIMx_SMCR[2:0]: Slave mode selection, — **TIM_SMCR_SMS**

- 000: Slave mode disabled, if CEN=1: prescaler clocked by internal clock.
- 001: Encoder mode 1, CNT counts up/down on TI2FP2 edge depending on TI1FP1 — TIM_ENCODERMODE_TI1
- 010: Encoder mode 2, CNT counts up/down on TI1FP1 edge depending on TI2FP2 — TIM_ENCODERMODE_TI2
- 011: Encoder mode 3, CNT counts up/down on both TI1FP1 and TI2FP2 edges — TIM_ENCODERMODE_TI12
- 100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
- 101: Gated Mode, counter clock enabled when the trigger input (TRGI)high.
- 110: Trigger Mode, CNT counter starts at a rising edge of the trigger TRGI
- 111: External Clock Mode 1, Rising edges of the selected trigger (TRGI) clock the counter.

TIMx_DIER — TIMx DMA/Interrupt enable register

TDE: TIMx_DIER[14]: Trigger DMA request enable — **TIM_DIER_TDE**

- 0: Trigger DMA request disabled.
- 1: Trigger DMA request enabled.

CC4DE: TIMx_DIER[12]: Capture/Compare4 DMA request enable — **TIM_DIER_CC4DE**
CC3DE: TIMx_DIER[11]: Capture/Compare4 DMA request enable — **TIM_DIER_CC3DE**
CC2DE: TIMx_DIER[10]: Capture/Compare4 DMA request enable — **TIM_DIER_CC2DE**
CC1DE: TIMx_DIER[9]: Capture/Compare4 DMA request enable — **TIM_DIER_CC1DE**

- 0: CC*i* DMA request disabled.
- 1: CC*i* DMA request enabled.

UDE: TIMx_DIER[8]: Update DMA request enable — **TIM_DIER_UDE**

- 0: Update DMA request disabled.
- 1: Update DMA request enabled.

TIE: TIMx_DIER[6]: Trigger interrupt enable — **TIM_DIER_TIE**

- 0: Trigger interrupt disabled.
- 1: Trigger interrupt enabled.

CC4IE: TIMx_DIER[4]: Capture/Compare 4 interrupt enable — **TIM_DIER_CC4IE**
CC3IE: TIMx_DIER[3]: Capture/Compare 3 interrupt enable — **TIM_DIER_CC3IE**
CC2IE: TIMx_DIER[2]: Capture/Compare 2 interrupt enable — **TIM_DIER_CC2IE**
CC1IE: TIMx_DIER[1]: Capture/Compare 1 interrupt enable — **TIM_DIER_CC1IE**

- 0: CC*i* interrupt disabled.
- 1: CC*i* interrupt enabled.

UIE: TIMx_DIER[0]: Update interrupt enable — **TIM_DIER_UIE**

- 0: Update interrupt disabled
- 1: Update interrupt enabled

TIMx_SR — TIMx status register

CC4OF: TIMx_SR[12]: Capture/Compare 4 overcapture flag — **TIM_SR_CC4OF**
CC3OF: TIMx_SR[11]: Capture/Compare 3 overcapture flag — **TIM_SR_CC3OF**
CC2OF: TIMx_SR[10]: Capture/Compare 2 overcapture flag — **TIM_SR_CC2OF**
CC1OF: TIMx_SR[9]: Capture/Compare 1 overcapture flag — **TIM_SR_CC1OF**

- 0: No overcapture has been detected
- 1: counter captured in TIMx_CCR*i* reg. while CC*i*IF flag was already set

TIF: TIMx_SR[6]: Trigger interrupt flag — **TIM_SR_TIF**

- 0: No trigger event occurred
- 1: Trigger interrupt pending

CC4IF: TIMx_SR[4]: Capture/Compare 4 interrupt flag — **TIM_SR_CC4IF**
CC3IF: TIMx_SR[3]: Capture/Compare 3 interrupt flag — **TIM_SR_CC3IF**
CC2IF: TIMx_SR[2]: Capture/Compare 2 interrupt flag — **TIM_SR_CC2IF**
CC1IF: TIMx_SR[1]: Capture/Compare 1 interrupt flag — **TIM_SR_CC1IF**

- If channel CC1 is configured as **output**:
  - 0: No match
  - 1: Content of cpt TIMx_CNT matches content of TIMx_CCR1 register.
- If channel CC1 is configured as **input**:
  - 0: No input capture occurred
  - 1: Counter value has been captured in TIMx_CCR1 register

UIF: TIMx_SR[6]: Update interrupt flag — **TIM_SR_UIF**

- 0: No update occurred
- 1: Update interrupt pending

TIMx_EGR — TIMx event generation register

These bits are set by software in order to generate an event, it is automatically cleared by hardware.

TG: TIMx_EGR[6]: Trigger generation — **TIM_EGR_TG**

- 0: No action
- 1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled

CC4G: TIMx_EGR[4]: Capture/compare 4 generation — **TIM_EGR_CC4G**
CC3G: TIMx_EGR[3]: Capture/compare 3 generation — **TIM_EGR_CC3G**
CC2G: TIMx_EGR[2]: Capture/compare 2 generation — **TIM_EGR_CC2G**
CC1G: TIMx_EGR[1]: Capture/compare 1 generation — **TIM_EGR_CC1G**

- 0: No action

- 1: A capture/compare event is generated on channel **i**:

UG: TIMx_EGR[0]: Update generation — **TIM_EGR_UG**
- 0: No action
- 1: Re-initialize the counter and generates an update of the registers.

TIMx_CCMR1 — TIMx capture/compare mode register 1

Output compare mode

OC2CE: TIMx_CCMR1[15]: Output compare 2 clear enable — **TIM_CCMR1_OC2CE**
OC1CE: TIMx_CCMR1[7]: Output compare 1 clear enable — **TIM_CCMR1_OC1CE**
- 0: OC**i**Ref is not affected by the ETRF input
- 1: OC**i**Ref is cleared as soon as a High level is detected on ETRF input

OC2M[2:0]: TIMx_ CCMR1[14:12]: Output compare 2 mode — **TIM_CCMR1_OC2M**
OC1M[2:0]: TIMx_ CCMR1[6:4]: Output compare 1 mode — **TIM_CCMR1_OC1M**
- 000: Frozen — TIM_OCMODE_TIMING
- 001: Set channel 1 to active (1) level on match. — TIM_OCMODE_ACTIVE
- 010: Set channel 1 to inactive (0) level on match. — TIM_OCMODE_INACTIVE
- 011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1. — TIM_OCMODE_TOGGLE
- 100: Force inactive level - OC1REF is forced low. — TIM_OCMODE_FORCED_INACTIVE
- 101: Force active level - OC1REF is forced high. — TIM_OCMODE_FORCED_ACTIVE
- 110: PWM mode 1 — TIM_OCMODE_PWM1
- 111: PWM mode 2 — TIM_OCMODE_PWM2

OC2PE: TIMx_CCMR1[11]: Output compare 2 preload enable — **TIM_CCMR1_OC2PE**
OC1PE: TIMx_CCMR1[3]: Output compare 1 preload enable — **TIM_CCMR1_OC1PE**
- 0: Preload reg. on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime
- 1: Preload reg. on TIMx_CCR1 enabled. Preload value is loaded at each update event.

OC2FE: TIMx_CCMR1[10]: Output compare 2 fast enable — **TIM_CCMR1_OC2FE**
OC1FE: TIMx_CCMR1[2]: Output compare 1 fast enable — **TIM_CCMR1_OC1FE**
- 0: fast mode disable: normal behavior — TIM_OCFAST_DISABLE
- 1: fast mode enable — TIM_OCFAST_ENABLE

CC2S[1:0]: TIMx_CCMR1[9:8]: Direction of the channel (input/output)+ used input — **TIM_CCMR1_CC2S**
CC1S[1:0]: TIMx_CCMR1[1:0]: Direction of the channel (input/output)+ used input — **TIM_CCMR1_CC1S**
- 00: CC1 channel is configured as output. — .
- 01: CC1 channel is configured as input, IC1(IC2) mapped on TI1(TI2) - direct — TIM_ICSELECTION_DIRECTTI
- 10: CC1 channel is configured as input, IC1(IC2) mapped on TI2(TI1) - inversion — TIM_ICSELECTION_INDIRECTTI
- 11: CC1 channel is configured as input, IC1 is mapped on TRC. — TIM_ICSELECTION_TRC

Input capture mode

IC2F[3:0]: TIMx_CCMR1[15:12]: Input capture 2 filter — **TIM_CCMR1_IC2F**
IC1F[3:0]: TIMx_CCMR1[15:12]: Input capture 1 filter — **TIM_CCMR1_IC1F**

defines frequency used to sample TI1 input + number N of consecutive events are needed to validate a transition on the output
f_s: sampling frequency
N: number of consecutive events to validate a transition

ETF	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
f_s	No filter	f_{CK_INT}			f_{DTS}/K											
K					2	2	4	4	8	8	16	16	16	32	32	32
N		2	4	8	6	8	6	8	6	8	5	6	8	5	6	8

IC2PSC[1:0]: TIMx_CCMR1[11:10]: Input capture 2 prescaler — **TIM_CCMR1_IC2PSC**
IC1PSC[1:0]: TIMx_CCMR1[11:10]: Input capture 1 prescaler — **TIM_CCMR1_IC1PSC**
- 00: no prescaler, capture each time an edge is detected on capture input — TIM_ICPSC_DIV1
- 01: capture is done once every 2 events — TIM_ICPSC_DIV2
- 10: capture is done once every 4 events — TIM_ICPSC_DIV4
- 11: capture is done once every 8 events — TIM_ICPSC_DIV8

CC2S[1:0]: TIMx_CCMR1[9:8]: Capture/compare 1 selection — **TIM_CCMR1_CC2S**
CC1S[1:0]: TIMx_CCMR1[1:0]: Capture/compare 1 selection — **TIM_CCMR1_CC1S**
defines direction of the channel (I/O) + used input.
- 00: CC**i** channel is configured as output. — .
- 01: CC**i** channel is configured as input, IC1(IC2) mapped on TI1(TI2) - direct — TIM_ICSELECTION_DIRECTTI
- 10: CC**i** channel is configured as input, IC1(IC2) mapped on TI2(TI1) - inverse — TIM_ICSELECTION_INDIRECTTI
- 11: CC**i** channel is configured as input, IC**i** mapped on TRC. — TIM_ICSELECTION_TRC

TIMx_CCMR2 — TIMx capture/compare mode register 2

Similar to TIMx_CCMR1 for channel 3 and 4

TIMx_CCER — TIMx capture/compare enable register

Channel configured as output - compare mode

CC4NP: TIMx_CCER[15]: Capture/Compare 4 output Polarity — **TIM_CCER_CC4NP**
CC3NP: TIMx_CCER[11]: Capture/Compare 3 output Polarity — **TIM_CCER_CC3NP**
CC2NP: TIMx_CCER[7]: Capture/Compare 2 output Polarity — **TIM_CCER_CC2NP**
CC1NP: TIMx_CCER[3]: Capture/Compare 1 output Polarity — **TIM_CCER_CC1NP**
- CC**i**NP must be kept cleared in this case

CC4P: TIMx_CCER[13]: Capture/Compare 4 output Polarity — **TIM_CCER_CC4P**
CC3P: TIMx_CCER[9]: Capture/Compare 3 output Polarity — **TIM_CCER_CC3P**
CC2P: TIMx_CCER[5]: Capture/Compare 2 output Polarity — **TIM_CCER_CC2P**
CC1P: TIMx_CCER[1]: Capture/Compare 1 output Polarity — **TIM_CCER_CC1P**
- 0: OC**i** active high
- 1: OC**i** active low

	CC4E: TIMx_CCER[13]: Capture/Compare 4 output Enable **CC3E**: TIMx_CCER[8]: Capture/Compare 3 output Enable **CC2E**: TIMx_CCER[4]: Capture/Compare 2 output Enable **CC1E**: TIMx_CCER[0]: Capture/Compare 1 output Enable • 0: Off - OC*i* is not active • 1: On - OC*i* signal is output on the corresponding output pin	**TIM_CCER_CC4E** **TIM_CCER_CC3E** **TIM_CCER_CC2E** **TIM_CCER_CC1E**
	Channel configured as input - capture mode	
	CC4NP - CC4P: TIMx_CCER[15, 13]: Capture/Compare 4 output Polarity **CC3NP - CC3P**: TIMx_CCER[11, 9]: Capture/Compare 3 output Polarity **CC2NP - CC2P**: TIMx_CCER[7, 5]: Capture/Compare 2 output Polarity **CC1NP - CC1P**: TIMx_CCER[3, 1]: Capture/Compare 1 output Polarity select TI1FP*i* and TI2FP*i* polarity for trigger or capture operations. • 00: noninverted/rising edge • 01: inverted/falling edge • 10: reserved, do not use this configuration. • 11: noninverted/both edges	**TIM_CCER_CC*i*NP** **TIM_CCER_CC*i*P** TIM_INPUTCHANNELPOLARITY_RISING TIM_INPUTCHANNELPOLARITY_FALLING TIM_INPUTCHANNELPOLARITY_BOTHEDGE
	CC4E: TIMx_CCER[13]: Capture/Compare 4 output Enable **CC3E**: TIMx_CCER[8]: Capture/Compare 3 output Enable **CC2E**: TIMx_CCER[4]: Capture/Compare 2 output Enable **CC1E**: TIMx_CCER[0]: Capture/Compare 1 output Enable Determines if a capture of the counter value can actually be done into the input capture/compare register *i* (TIMx_CCR*i*) or not. • 0: Capture disabled • 1: Capture enabled	**TIM_CCER_CC4E** **TIM_CCER_CC3E** **TIM_CCER_CC2E** **TIM_CCER_CC1E**
TIMx_CNT	**TIMx counter**	
	CNT[15:0]:Counter value	
TIMx_PSC	**TIMx prescaler**	
	PSC[15:0]:Prescaler value Counter clock frequency $f_{CK_CNT} = f_{CK_PSC} / (PSC[15:0] + 1)$.	
TIMx_ARR	**TIMx auto-reload register**	
	ARR[15:0]: Auto-reload value	
TIMx_CCR*i*	**TIMx capture/compare register *i***	
	CCR*i*: value to be loaded in the capture/compare *i* register (preload value). CCR*i*: counter value transferred by the last input capture 1 event (IC1)	

TIMx_DCR	**TIMx DMA control register**	
	DBL[4:0]: TIMx_DCR[12:8] = ***x-1*** (0 to 18): DMA burst length: ***x*** number of DMA transfers **DBA**[4:0]:DMA base address, DBA defined as an offset starting from the address of the TIMx_CR1 register. **DBL** registers from TIMx_CR1 are transferred.	TIM_DMABURSTLENGTH_*x*TRANSFERS TIM_DMABASE_**XXX** **XXX**: name of the register
TIMx_DMAR	**TIMx DMA address for full transfer**	
	DMAB[15:0]:DMA register for burst accesses	
TIM2_OR	**TIM2 option register**	
	ITR1_RMP[1:0]: TIM2_OR[11:10]: Internal trigger 1 remap • 00: TIM8_TRGOUT • 01: Reserved • 10: OTG FS SOF is connected to the TIM2_ITR1 input • 11: OTG HS SOF is connected to the TIM2_ITR1 input	

Analog to Digital Converter

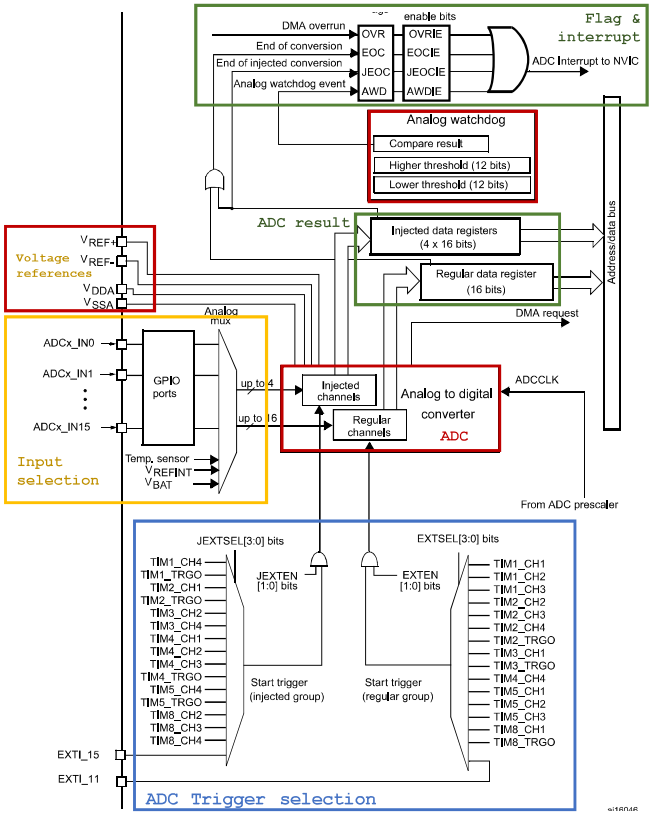
Description of the different steps to setup the Analog to Digital Converter

1. Main configuration

1. Enable peripheral Clocks for ADC and GPIO via RCC module
2. Configure the GPIO pins as analog input
3. Disable the ADC, `ADON` in register `ADC_CR2`
4. Configure the ADC peripheral (see part 2)
5. Enable the ADC, `ADON` in register `ADC_CR2`

2. ADC configuration

1. Set the conversion timing
   - Pre-scaler, `ADCPRE` in register `ADC_CCR`
   - Sampling time, `MPx` in register `ADC_SMPR2/1`
2. Select the conversion mode
   - Single Channel or Scan mode, `SCAN` in register `ADC_CR1`
   - Single conversion or Continuous mode, `CONT` in register `ADC_CR2`
   - Discontinuous mode, `DISCEN` in register `ADC_CR1`
3. Select the HW conversion triggering
   - Edge, `EXTEN` in register `ADC_CR2`
   - Source, `EXTSEL` in register `ADC_CR2`
4. Set the data characteristics
   - Resolution, `L` in register `ADC_CR1`
   - Alignment (right or left), `ALIGN` in register `ADC_CR2`
5. Definition of the AD conversion sequence
   - Length of the sequence, `RES` in register `ADC_SQR1`
   - List of channel(s), `SQ1,…` in register `ADC_SQR3`
6. Enable interrupt (if required): `EOCIE` in register `ADC_CR1`
7. Enable DMA (if required): `DMA` in register `TIMx_DIER`



Register	Description
ADC_CCR	**ADC common control register**

TSVREFE: ADC_CCR[23]: Temperature sensor and VREFINT enable
- 0: Temperature sensor and VREFINT channel disabled
- 1: Temperature sensor and VREFINT channel enabled

ADC_CCR_TSVREFE

VBATE: ADC_CCR[22]: VBAT enable
- 0: VBAT channel disabled
- 1: VBAT channel enabled

ADC_CCR_VBATE

ADCPRE[1:0]: ADC_CCR[17:16]: ADC prescaler
- 00: PCLK2 divided by 2
- 01: PCLK2 divided by 4
- 10: PCLK2 divided by 6
- 11: PCLK2 divided by 8

ADC_CCR_ADCPRE
ADC_CLOCK_SYNC_PCLK_DIV2
ADC_CLOCK_SYNC_PCLK_DIV4
ADC_CLOCK_SYNC_PCLK_DIV6
ADC_CLOCK_SYNC_PCLK_DIV8

DMA[1:0]: ADC_CCR[15:14]: Direct memory access mode for multi ADC mode
- 00: DMA mode disabled
- 01: DMA mode 1 enabled (2 / 3 half-words one by one - 1 then 2 then 3)
- 10: DMA mode 2 enabled (2 / 3 half-words by pairs - 2&1 then 1&3 then 3&2)
- 11: DMA mode 3 enabled (2 / 3 bytes by pairs - 2&1 then 1&3 then 3&2)

ADC_CCR_DMA

DDS: ADC_CCR[13]: DMA disable selection (for multi-ADC mode)
- 0: No new DMA request is issued after the last transfer
- 1: DMA requests are issued as long as data are converted and DMA = 01, 10, 11

ADC_CCR_DDS

DELAY[4:0]: ADC_CCR[11:8]: Delay between 2 sampling phases
- Set and cleared by software. These bits are used in dual or triple interleaved modes.
- ***i***: (5+ ***i***) * TADCCLK

ADC_CCR_DELAY
ADC_TWOSAMPLINGDELAY_*j*CYCLES

MULTI[4:0] : ADC_CCR[4:0]:Multi ADC mode selection. These bits are written by software to select the operating mode.

Independent mode:
- 00000: All the ADCs independent

Dual mode, ADC1 and ADC2 working together, ADC3 is independent
- 00001: Combined regular simultaneous + injected simultaneous mode
- 00010: Combined regular simultaneous + alternate trigger mode
- 00101: Injected simultaneous mode only 00110: Regular simultaneous mode only 00111: interleaved mode only
- 01001: Alternate trigger mode only

Triple mode: ADC1, 2 and 3 working together
- 10001: Combined regular simultaneous + injected simultaneous mode

ADC_CCR_MULTI

- 10010: Combined regular simultaneous + alternate trigger mode
- 10101: Injected simultaneous mode only
- 10110: Regular simultaneous mode only
- 10111: interleaved mode only
- 11001: Alternate trigger mode only

ADC_CR1 TIMx control register 1

OVRIE: ADC_CR1[26]: Overrun interrupt enable — **ADC_CR1_OVRIE**
- 0: Overrun interrupt disable
- 1: Overrun interrupt enable

RES[1:0]: ADC_CR1[25:24]: Resolution — **ADC_CR1_RES**, ADC_RESOLUTION_12B, ADC_RESOLUTION_10B, ADC_RESOLUTION_8B, ADC_RESOLUTION_6B
- 00: 12 bits (minimum 15 ADCCLK cycles).
- 01: 10 bits (minimum 13 ADCCLK cycles).
- 10: 8 bits (minimum 11 ADCCLK cycles).
- 11: 6 bits (minimum 9 ADCCLK cycles).

DISCNUM[2:0]: ADC_CR1[15:13]: Discontinuous mode channel count — **ADC_CR1_DISCNUM**
- i: i+1 channels, with i∈[0;7]

JDISCEN: ADC_CR1[12]: Discontinuous mode on injected channels — **ADC_CR1_JDISCEN**
- 0: Discontinuous mode on injected channels disabled
- 1: Discontinuous mode on injected channels enabled.

DISCEN: ADC_CR1[11]: Discontinuous mode on regular channels — **ADC_CR1_DISCEN**
- 0: Discontinuous mode on regular channels disabled
- 1: Discontinuous mode on regular channels enabled.

JAUTO: ADC_CR1[10]: Automatic injected group conversion — **ADC_CR1_JAUTO**
- 0: Automatic injected group conversion disabled
- 1: Automatic injected group conversion enabled

SCAN: ADC_CR1[8]: Scan mode — **ADC_CR1_SCAN**
- 0: Scan mode disabled
- 1: Scan mode enabled

JEOCIE: ADC_CR1[7]: Interrupt enable for injected channels — **ADC_CR1_JEOCIE**
- 0: JEOC interrupt disabled
- 1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

AWDIE: ADC_CR1[6]: Analog watchdog interrupt enable — **ADC_CR1_AWDIE**
- 0: Analog watchdog interrupt disabled
- 1: Analog watchdog interrupt enabled

EOCIE: ADC_CR1[5]: Interrupt enable for EOC — **ADC_CR1_EOCIE**
- 0: EOC interrupt disabled
- 1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

AWDEN: ADC_CR1[23]: Analog watchdog enable on regular channels — **ADC_CR1_AWDEN**
- 0: Analog watchdog disable on regular channels
- 1: Analog watchdog enable on regular channels

JAWDEN: ADC_CR1[22]: Analog watchdog enable on regular channels — **ADC_CR1_JAWDEN**
- 0: Analog watchdog disable on injected channels
- 1: Analog watchdog enable on injected channels

AWDSGL: ADC_CR1[9]: Enable the watchdog on a single channel in scan mode — **ADC_CR1_AWDSGL**
- 0: Analog watchdog enabled on all channels
- 1: Analog watchdog enabled on a single channel

AWDCH[4:0]: ADC_CR1[4:0] : Analog watchdog channel select bits — ADC_ANALOGWATCHDOG_SINGLE_REG, ADC_ANALOGWATCHDOG_SINGLE_INJEC, ADC_ANALOGWATCHDOG_SINGLE_REGINJEC, ADC_ANALOGWATCHDOG_ALL_REG, ADC_ANALOGWATCHDOG_ALL_INJEC, ADC_ANALOGWATCHDOG_ALL_REGINJEC, ADC_ANALOGWATCHDOG_NONE
- *i* : ADC analog input Channel *i* ∈[0;18] — **ADC_CR1_AWDCH**, ADC_CHANNEL_*i*

ADC_CR2 ADC control register 2

SWSTART: ADC_CR2[30]: Start conversion of regular channels — **ADC_CR2_SWSTART**
- 0: Reset state
- 1: Starts conversion of regular channels

EXTEN[2:0]: ADC_CR2[29:28]: External trigger enable for regular channels — **ADC_CR2_EXTEN**, ADC_EXTERNALTRIGCONVEDGE_NONE, ADC_EXTERNALTRIGCONVEDGE_RISING, ADC_EXTERNALTRIGCONVEDGE_FALLING, ADC_EXTERNALTRIGCONVEDGE_RISINGFALLING
- 00: Trigger detection disabled
- 01: Trigger detection on the rising edge
- 10: Trigger detection on the falling edge
- 11: Trigger detection on both the rising and falling edges

EXTSEL[3:0]: ADC_CR2[27:24]: External event select for regular group — **ADC_CR2_EXTSEL**, ADC_EXTERNALTRIGCONV_T1_CC1, ADC_EXTERNALTRIGCONV_T1_CC2, ADC_EXTERNALTRIGCONV_T1_CC3, ADC_EXTERNALTRIGCONV_T2_CC2, ADC_EXTERNALTRIGCONV_T2_CC3, ADC_EXTERNALTRIGCONV_T2_CC4, ADC_EXTERNALTRIGCONV_T2_TRGO, ADC_EXTERNALTRIGCONV_T3_CC1, ADC_EXTERNALTRIGCONV_T3_TRGO, ADC_EXTERNALTRIGCONV_T4_CC4, ADC_EXTERNALTRIGCONV_T5_CC1, ADC_EXTERNALTRIGCONV_T5_CC2, ADC_EXTERNALTRIGCONV_T5_CC3, ADC_EXTERNALTRIGCONV_T8_CC1, ADC_EXTERNALTRIGCONV_T8_TRGO, ADC_EXTERNALTRIGCONV_Ext_IT11
- 0000: Timer 1 CC1 event
- 0001: Timer 1 CC2 event
- 0010: Timer 1 CC3 event
- 0011: Timer 2 CC2 event
- 0100: Timer 2 CC3 event
- 0101: Timer 2 CC4 event
- 0110: Timer 2 TRGO event
- 0111: Timer 3 CC1 event
- 1000: Timer 3 TRGO event
- 1001: Timer 4 CC4 event
- 1010: Timer 5 CC1 event
- 1011: Timer 5 CC2 event
- 1100: Timer 5 CC3 event
- 1101: Timer 8 CC1 event
- 1110: Timer 8 TRGO event
- 1111: EXTI line 11

JSWSTART: ADC_CR2[22]: Start conversion of injected channels • 0: Reset state • 1: Starts conversion of injected channels	**ADC_CR2_JWSTART**
JEXTEN[2:0]: ADC_CR2[21:20]: External trigger enable for injected channels • 00: Trigger detection disabled • 01: Trigger detection on the rising edge • 10: Trigger detection on the falling edge • 11: Trigger detection on both the rising and falling edges	**ADC_CR2_JEXTSEN**
JEXTSEL[3:0]: ADC_CR2[19:16]:External event select for injected group. These bits select the external event used to trigger the start of conversion of an injected group. • 0000: Timer 1 CC4 event • 0001: Timer 1 TRGO event • 0010: Timer 2 CC1 event • 0011: Timer 2 TRGO event • 0100: Timer 3 CC2 event • 0101: Timer 3 CC4 event • 0110: Timer 4 CC1 event • 0111: Timer 4 CC2 event • 1000: Timer 4 CC3 event • 1001: Timer 4 TRGO event • 1010: Timer 5 CC4 event • 1011: Timer 5 TRGO event • 1100: Timer 8 CC2 event • 1101: Timer 8 CC3 event • 1110: Timer 8 CC4 event • 1111: EXTI line15	**ADC_CR2_JEXTSEL**
ALIGN: ADC_CR2[11]: Data alignment • 0: Right alignment • 1: Left alignment	**ADC_CR2_ALIGN** ADC_DATAALIGN_RIGHT ADC_DATAALIGN_LEFT
EOCS: ADC_CR2[10]: End of conversion selection. • 0: The EOC bit is set at the end of each sequence of regular conversions. Overrun detection is enabled only if DMA=1. • 1: The EOC bit is set at the end of each regular conversion. Overrun detection is enabled.	**ADC_CR2_EOCS**
DDS: ADC_CR2[9]: DMA disable selection (for single ADC mode) • 0: No new DMA request is issued after the last transfer • 1: DMA requests are issued as long as data are converted and DMA=1	**ADC_CR2_DDS**
DMA: ADC_CR2[8]: Direct memory access mode (for single ADC mode) • 0: DMA mode disabled • 1: DMA mode enabled	**ADC_CR2_DMA**
CONT: ADC_CR2[1]: Continuous conversion • 0: Single conversion mode • 1: Continuous conversion mode	**ADC_CR2_CONT**
ADON: ADC_CR2[0]: A/D Converter ON/OFF • 0: Disable ADC conversion and go to power down mode • 1: Enable ADC	**ADC_CR2_ADON**

ADC_SMPR1 **ADC_SMPR2**	**ADC Sample Time Register** : channel 10 to 18 **ADC Sample Time Register** : channel 0 to 9	
	MPx[2:0]:Channel ***i*** ∈[10;18] sampling time selection • 000: 3 cycles • 001: 15 cycles • 010: 28 cycles • 011: 56 cycles • 100: 84 cycles • 101: 112 cycles • 110: 144 cycles • 111: 480 cycles	**ADC_SMPR1_SMP*j*** ***j* 10 to 18** **ADC_SMPR2_SMP*j*** ***j* 0 to 9** ADC_SAMPLETIME_3CYCLES ADC_SAMPLETIME_15CYCLES ADC_SAMPLETIME_28CYCLES ADC_SAMPLETIME_56CYCLES ADC_SAMPLETIME_84CYCLES ADC_SAMPLETIME_112CYCLES ADC_SAMPLETIME_144CYCLES ADC_SAMPLETIME_480CYCLES
ADC_DR	**ADC regular data register**	
	DATA[15:0]: ADC_DR[15:0]: Regular data : conversion result (left or right aligned)	**ADC_DR_DATA**
ADC_SR	**ADC common status register**	
	OVR: ADC_SR[5]: Overrun : set by hardware when data are lost • 0: No overrun occurred • 1: Overrun has occurred	**ADC_SR_OVR**
	STRT: ADC_SR[4]: Regular channel start flag • 0: No regular channel conversion started • 1: Regular channel conversion has started	**ADC_SR_STRT**
	JSTRT: ADC_SR[3]: Injected channel start flag • 0: No injected group conversion started • 1: Injected group conversion has started	**ADC_SR_JSTRT**
	JEOC: ADC_SR[2]: Injected channel end of conversion • 0: Conversion is not complete • 1: Conversion complete	**ADC_SR_JEOC**
	EOC: ADC_SR[1]: Regular channel end of conversion • 0: Conversion (EOCS=0), or sequence of conversions (EOCS=1) not complete • 1: Conversion complete (EOCS=0), or sequence of conversions complete (EOCS=1)	**ADC_SR_EOC**
	AWD: ADC_SR[0]: Analog watchdog flag	

	• 0: No analog watchdog event occurred • 1: Analog watchdog event occurred	ADC_SR_AWD
ADC_CSR	**ADC common status register**	
	Group together the status bits of ADC3_SR, ADC2_SR register and ADC1_SR registers.	
ADC_SQRx	**ADC regular sequence register x**	
	L[3:0]: ADC_SQR1[23:20]: Regular channel sequence length, total number of conversion • *i*: *i+1* conversions • ... **SQ***i*[4:0]: channel number for *i*th conversion in regular sequence • ADC_SQR1: SQ13 to SQ16 • ADC_SQR2: SQ7 to SQ12 • ADC_SQR3: SQ1 to SQ6	ADC_SQR1_L ADC_SQR1_SQj j : 13 to 16 ADC_SQR2_SQj j : 7 to 12 ADC_SQR3_SQj j : 1 to 6
ADC_CDR	**ADC common regular data register for dual and triple modes**	
	•	
ADC_JOFRx	**ADC injected channel data offset register x** ∈ [1;4]	
	•	
ADC_HTR	**ADC watchdog higher threshold register**	
	•	
ADC_LTR	**ADC watchdog lower threshold register**	
	•	
ADC_JSQR	**ADC injected sequence register**	
	•	
ADC_JDRx	**ADC injected data register x** ∈ [1;4]	
	•	

Digital to Analog Converter

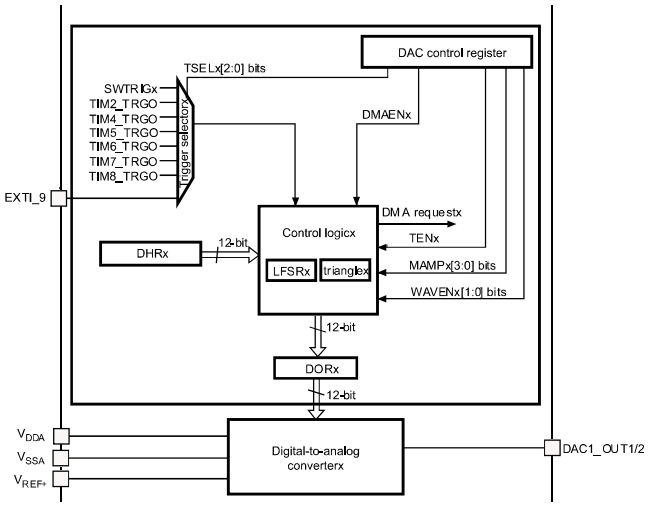
Description of the different steps to setup the Analog to Digital Converter

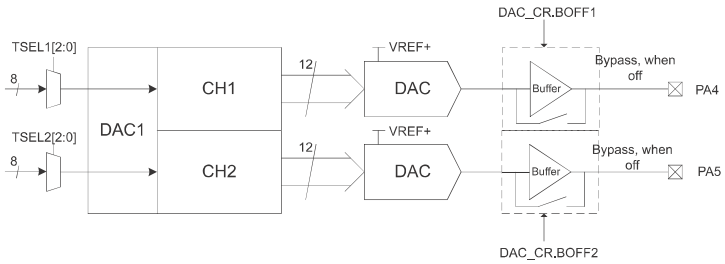
1. Main configuration

1. Enable peripheral Clocks for DAC and GPIO via RCC module
2. Configure the GPIO pins as analog output
3. Disable the DAC, EN in register DAC_CR
4. Configure the DAC peripheral (see part 2)
5. Enable the DAC, EN in register DAC_CR

2. DAC configuration

1. Select the conversion triggering (if required)
   - Enable, TENx in register DAC_CR
   - Source, TSELx in register DAC_CR
2. Enable wave generation (if required): WAVEx in register DAC_CR
3. Enable DMA (if required): DMAENx in register DAC_CR





Register	Description	
DAC_CR	**DAC control register**	
	DMAUDRIE2: DAC_CR[29]: DAC channel2 DMA underrun interrupt enable. • 0: DAC channel2 DMA underrun interrupt disabled • 1: DAC channel2 DMA underrun interrupt enabled	**DAC_CR_DMAUDRIE2**
	DMAEN2: DAC_CR[28]: DAC channel2 DMA enable • This bit is set / cleared by software. • 0: DAC channel2 DMA mode disabled • 1: DAC channel2 DMA mode enabled	**DAC_CR_DMAEN2**
	MAMP2[3:0]: DAC_CR[27:24] DAC ch2 mask/amplitude selector, triangle gen. mode • 0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1 • 0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3 • 0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7 • 0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15 • 0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31 • 0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63 • 0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127 • 0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255 • 1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511 • 1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023 • 1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047 • ≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095	**DAC_CR_MAMP2**
	WAVE2[1:0]: DAC_CR[23:22]: DAC channel2 noise/triangle wave generation enable. Only used if bit TEN2 = 1 (DAC channel2 trigger enabled) • 00: wave generation disabled • 01: Noise wave generation enabled • 1x: Triangle wave generation enabled	**DAC_CR_WAVE2**
	TSEL2[2:0]: DAC_CR[21:19]: DAC channel2 trigger selection. Select the external event used to trigger DAC channel2. Only used if DAC ch2 trigger enabled. • 000: Timer 6 TRGO event • 001: Timer 8 TRGO event • 010: Timer 7 TRGO event • 011: Timer 5 TRGO event • 100: Timer 2 TRGO event • 101: Timer 4 TRGO event • 110: External line9	**DAC_CR_WAVEN2** **DAC_TRIGGER_T6_TRGO** **DAC_TRIGGER_T8_TRGO** **DAC_TRIGGER_T7_TRGO** **DAC_TRIGGER_T5_TRGO** **DAC_TRIGGER_T2_TRGO** **DAC_TRIGGER_T4_TRGO** **DAC_TRIGGER_EXT_IT9**
	• 111: Software trigger	**DAC_TRIGGER_SOFTWARE**
	TEN2: DAC_CR[18]: DAC channel2 trigger enable. • 0: DAC channel2 trigger disabled DAC_DHRx => DAC_DOR2 register : 1 cycle • 1: DAC channel2 trigger enabled DAC_DHRx => DAC_DOR2 register : 3 cycles	**DAC_CR_TEN2**
	BOFF2: DAC_CR[17]: DAC channel2 output buffer disable • 0: DAC channel2 output buffer enabled • 1: DAC channel2 output buffer disabled	**DAC_CR_BOFF2** **DAC_OUTPUTBUFFER_ENABLE** **DAC_OUTPUTBUFFER_DISABLE**
	EN2: DAC_CR[16]: DAC channel2 enable • 0: DAC channel2 disabled • 1: DAC channel2 enabled	**DAC_CR_EN2**
	DMAUDRIE1: DAC_CR[13]: DAC channel1 DMA Underrun Interrupt enable • 0: DAC channel1 DMA Underrun Interrupt disabled • 1: DAC channel1 DMA Underrun Interrupt enabled	**DAC_CR_DMAUDRIE1**
	DMAEN1: DAC_CR[12]: DAC channel1 DMA enable • 0: DAC channel1 DMA mode disabled • 1: DAC channel1 DMA mode enabled	**DAC_CR_DMAEN1**
	MAMP1[3:0]: DAC_CR[11:8]: DAC channel1 mask/amplitude selector. • 0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1 • 0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3 • 0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7 • 0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15 • 0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31 • 0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63 • 0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127 • 0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255 • 1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511 • 1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023 • 1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047 • ≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095	**DAC_CR_MAMP1**
	WAVE1[1:0]: DAC_CR[7:6]: DAC channel1 noise/triangle wave generation enable • 00: wave generation disabled • 01: Noise wave generation enabled • 1x: Triangle wave generation enabled	**DAC_CR_WAVE1**
	TSEL1[2:0]: DAC_CR[5:3]: DAC channel1 trigger selection • 000: Timer 6 TRGO event • 001: Timer 8 TRGO event • 010: Timer 7 TRGO event	**DAC_CR_TSEL1** **DAC_TRIGGER_T6_TRGO** **DAC_TRIGGER_T8_TRGO** **DAC_TRIGGER_T7_TRGO**

- 011: Timer 5 TRGO event — DAC_TRIGGER_T5_TRGO
- 100: Timer 2 TRGO event — DAC_TRIGGER_T2_TRGO
- 101: Timer 4 TRGO event — DAC_TRIGGER_T4_TRGO
- 110: External line9 — DAC_TRIGGER_EXT_IT9
- 111: Software trigger — DAC_TRIGGER_SOFTWARE

TEN1: DAC_CR[2]: DAC channel1 trigger enable — DAC_CR_TEN1
- 0: DAC channel1 trigger disabled. DAC_DHRx => DAC_DOR1 register : 1 cycle
- 1: DAC channel1 trigger enabled DAC_DHRx => DAC_DOR1 register : 3 cycles

BOFF1: DAC_CR[1]: DAC channel1 output buffer disable
- 0: DAC channel1 output buffer enabled — DAC_CR_BOFF1 DAC_OUTPUTBUFFER_ENABLE
- 1: DAC channel1 output buffer disabled — DAC_OUTPUTBUFFER_DISABLE

EN1: DAC_CR[0]: DAC channel1 enable — DAC_CR_EN1
- 0: DAC channel1 disabled
- 1: DAC channel1 enabled

DAC_SWTRIGR DAC software trigger register

SWTRIG2: DAC_SWTRIGR[1]: DAC channel2 software trigger — DAC_SWTRIGR_SWTRIG1
- 0: Software trigger disabled
- 1: Software trigger enabled

SWTRIG1: DAC_SWTRIGR[0]: DAC channel1 software trigger — DAC_SWTRIGR_SWTRIG1
- 0: Software trigger disabled
- 1: Software trigger enabled

Note: These bits are cleared by hardware (one APB1 clock cycle later) once the DAC_DHR*i* register value has been loaded into the DAC_DOR*i* register.

DAC_DHR12R1 DAC channel1 12-bit right-aligned data holding register

DACC1DHR[11:0]: DAC_DHR12R1[11:0]: DAC channel1 12-bit right-aligned data — DAC_DHR12R1_DACC1DHR

DAC_DHR12L1 DAC channel1 12-bit left-aligned data holding register

DACC1DHR[11:0]: DAC_DHR12L1[15:4]: DAC channel1 12-bit left-aligned data — DAC_DHR12L1_DACC1DHR

DAC_DHR8R1 DAC channel1 12-bit right-aligned data holding register

DACC1DHR[8:0]: DAC_DHR12R1[8:0]: DAC channel1 8-bit right-aligned data — DAC_DHR8R1_DACC1DHR

DAC_DHR12R2 DAC channel2 12-bit right-aligned data holding register

DACC2DHR[11:0]: DAC_DHR12R2[11:0]: DAC channel2 12-bit right-aligned data — DAC_DHR12R2_DACC2DHR

DAC_DHR12L2 DAC channel2 12-bit left-aligned data holding register

DACC2DHR[11:0]: DAC_DHR12L2[15:4]: DAC channel2 12-bit left-aligned data — DAC_DHR12L2_DACC2DHR

DAC_DHR8R2 DAC channel2 12-bit right-aligned data holding register

DACC1DHR[8:0]: DAC_DHR12R2[8:0]: DAC channel2 8-bit right-aligned data — DAC_DHR8R2_DACC2DHR

DAC_DHR12RD DAC Dual DAC 12-bit right-aligned data holding register

DACC2DHR[11:0] DAC_DHR12R1[26:16]:: DAC channel2 12-bit right-aligned data — DAC_DHR12RD_DACC1DHR

DACC1DHR[11:0]: DAC_DHR12R1[11:0]: DAC channel1 12-bit right-aligned data — DAC_DHR12RD_DACC2DHR

DAC_DHR12LD DAC channel1 12-bit right-aligned data holding register

DACC2DHR[11:0] DAC_DHR12R1[31:20]:: DAC channel2 12-bit left-aligned data — DAC_DHR12LD_DACC1DHR

DACC1DHR[11:0]: DAC_DHR12R1[15:14]: DAC channel1 12-bit left-aligned data — DAC_DHR12LD_DACC2DHR

DAC_DHR8RD DUAL DAC 8-bit right aligned data holding register

DACC2DHR[7:0] DAC_ DHR8RD[15:8]: DAC channel2 8-bit right-aligned data — DAC_DHR8RD_DACC2DHR

DACC1DHR[7:0]: DAC_ DHR8RD[7:0]: DAC channel1 8-bit right-aligned data — DAC_DHR8RD_DACC1DHR

DAC_DOR1 DAC channel 1 data output register

DACC1DOR [11:0]: DAC_ DOR1[11:0]: DAC channel 1 data output — DAC_DOR1_DACC1DOR

DAC_DOR2 DAC channel 2 data output register

DACC2DOR [11:0]: DAC_ DOR2[11:0]: DAC channel 2 data output — DAC_DOR2_DACC2DOR

DAC_SR DAC status register

DMAUDR2: DAC_ SR[29]: DAC channel 2 DMA underrun flag, set by HW, cleared by SW — DAC_SR_DMAUDR1
- 0: No DMA underrun error condition occurred for DAC channel 2
- 1: DMA underrun error condition occurred for DAC channel 2

DMAUDR1: DAC_ SR[13]: DAC channel 1 DMA underrun flag — DAC_SR_DMAUDR2
- 0: No DMA underrun error condition occurred for DAC channel 1
- 1: DMA underrun error condition occurred for DAC channel 1
  - the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate)