Sistemi a Microcontrollore

4. IO Analogico

Anno Accademico 2020/2021

Indice

- Uscite Analogiche
  - Pulse Width Modulation
    - PWM negli ARM
  - Digital-to-Analog (D/A) Conversion
- Ingressi Analogici
  - Analog-to-Digital (A/D) Conversion
    - ADC negli ARM

Indice

- Uscite Analogiche
  - Pulse Width Modulation
    - PWM negli ARM
  - Digital-to-Analog (D/A) Conversion
- Ingressi Analogici
  - Analog-to-Digital (A/D) Conversion
    - ADC negli ARM

Pulse Width Modulation

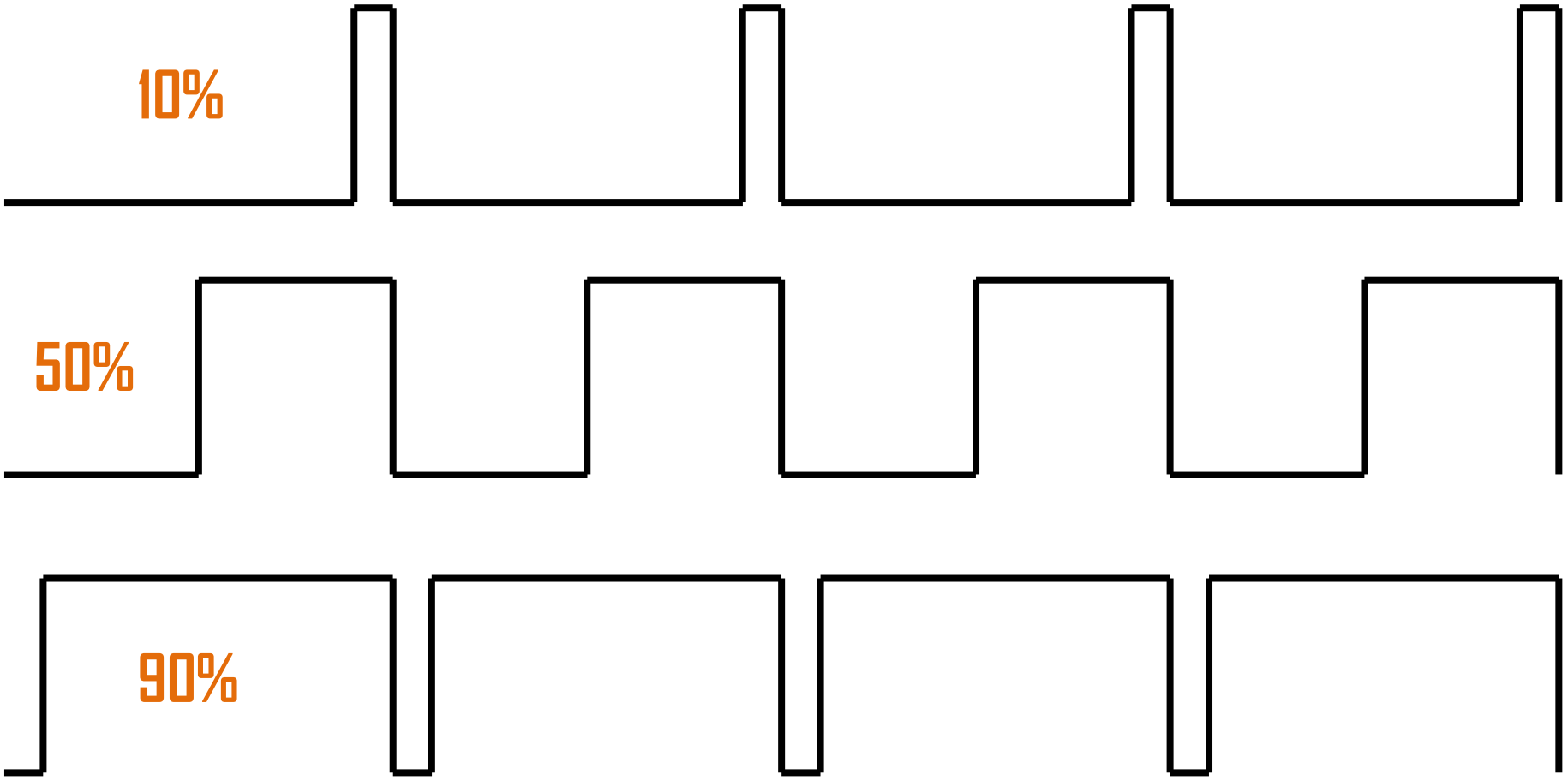
- La **Pulse Width Modulation (PWM)** è un modo di codificare digitalmente livelli di segnale analogici
  - attraverso contatori ad alta risoluzione il **duty cycle** (larghezza/periodo) di un **impulso** viene **modulato** per codificare uno specifico **livello** di un **segnale analogico**
- Il segnale **PWM** è **comunque digitale**
  - il suo valore è sempre o alto (1) o basso (0)
  - se vi è sufficiente larghezza di banda, qualunque segnale analogico può essere codificato con la PWM

Pulse Width Modulation

- È necessario un **filtro passa-basso** (Low-Pass Filter, LPF) per **ripulire il segnale** ed **eliminare le componenti di rumore** della PWM
- Il **voltaggio del segnale di uscita** è direttamente **proporzionale alla larghezza dell'impulso**
  - cambiando la larghezza dell'impulso è possibile controllare il voltaggio (valore analogico) dell'uscita
- I **segnali PWM** sono tipicamente ottenuti **attraverso i timer** in dotazione al **microcontrollore**

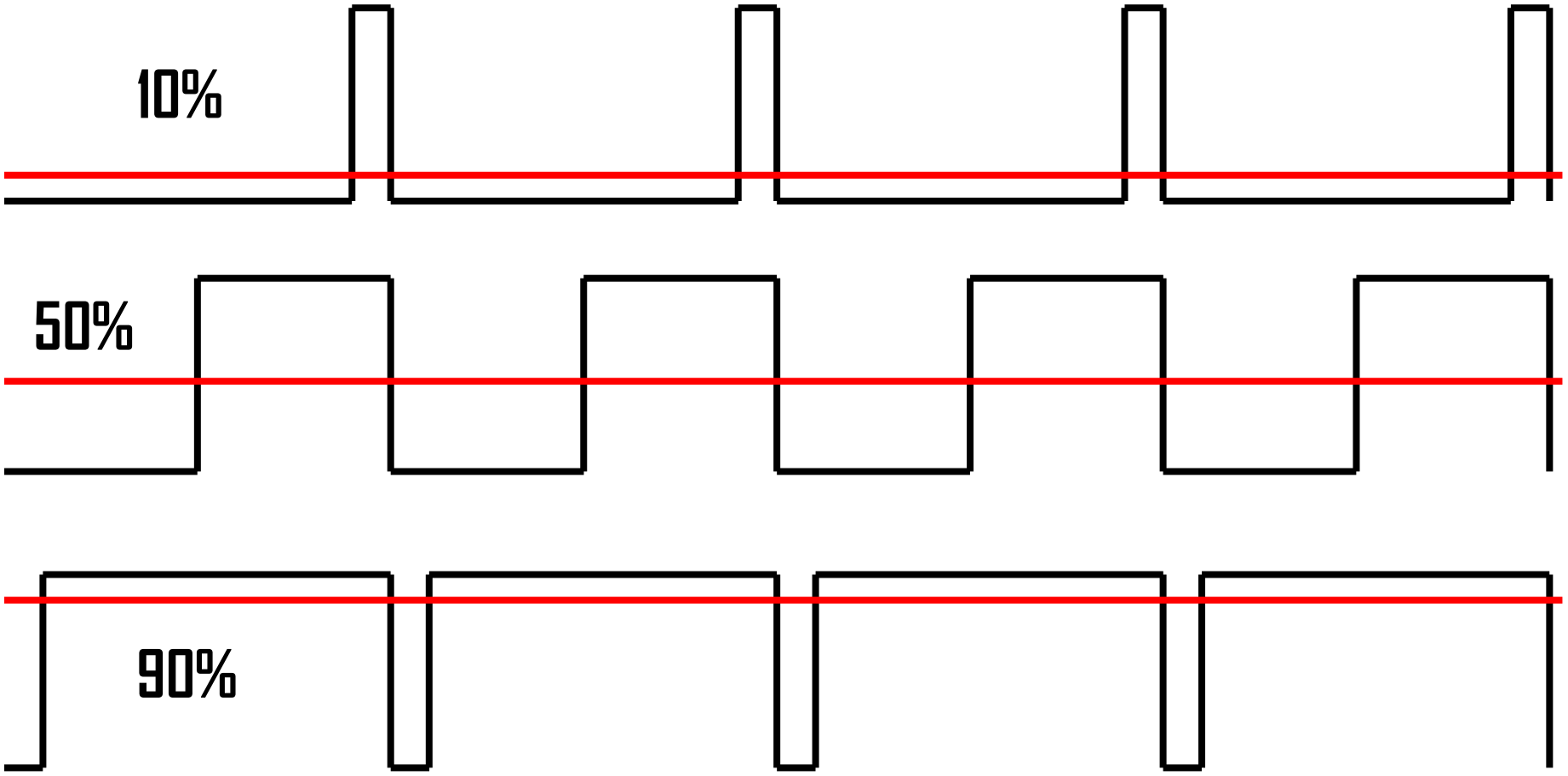
Pulse Width Modulation

duty cycle = t_{HIGH}/t_{TOT}



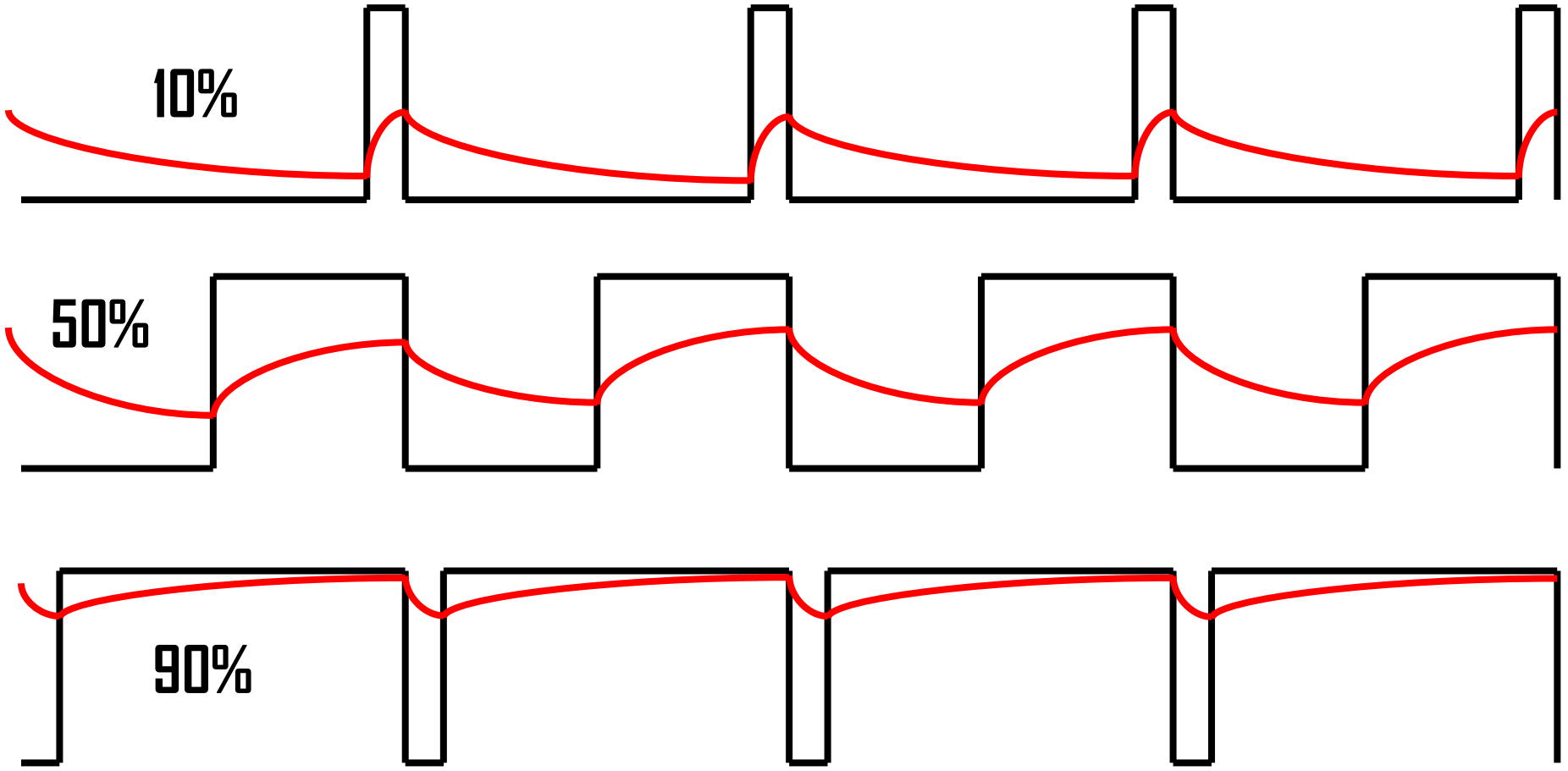
Pulse Width Modulation

output analogico dopo **LPF ideale**



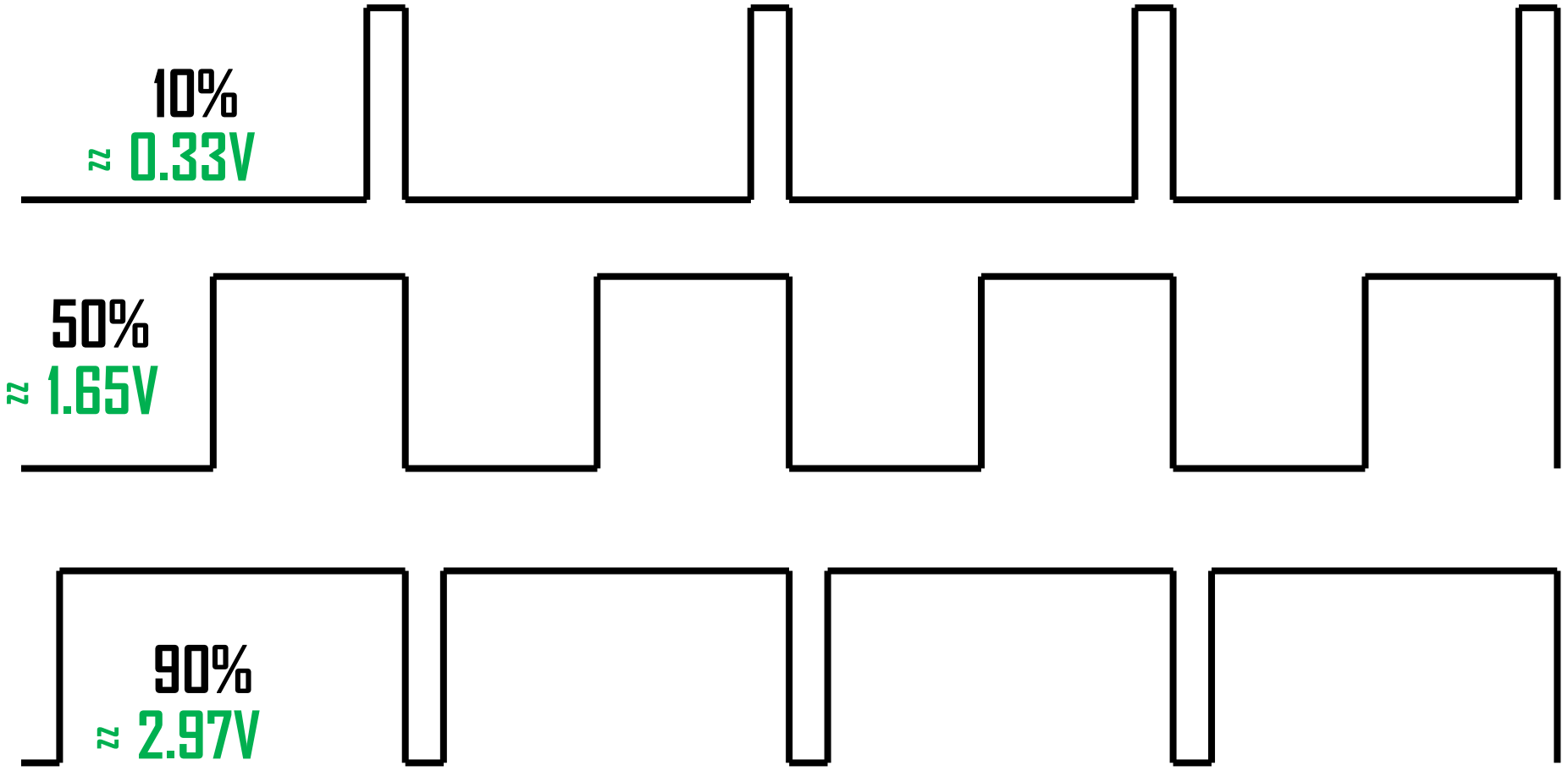
Pulse Width Modulation

output analogico dopo **LPF reale**



Pulse Width Modulation

tensione analogica equivalente (**Vdd=3.3V**) → duty cycle * Vdd



Indice

- Uscite Analogiche
  - Pulse Width Modulation
    - PWM negli ARM
  - Digital-to-Analog (D/A) Conversion
- Ingressi Analogici
  - Analog-to-Digital (A/D) Conversion
    - ADC negli ARM

Pulse Width Modulation nell'ARMv6-M

- La **PWM** nell'ARMv6-M viene implementata **tramite** i **timer** presenti nell'architettura
- In particolare è possibile generare **una diversa uscita PWM** da **ogni canale capture/compare** dei timer
- Sono possibili **due** diverse **modalità** di generazione di uscite analogiche tramite **PWM** nei timer dell'ARMv6-M
  - **edge-aligned** (TIM1, TIM3, TIM14, TIM15, TIM16, TIM17), CMS = 2'b00 nel registro TIMx_CR1
  - **center-aligned** (TIM1, TIM3), CMS != 2'b00 nel registro TIMx_CR1

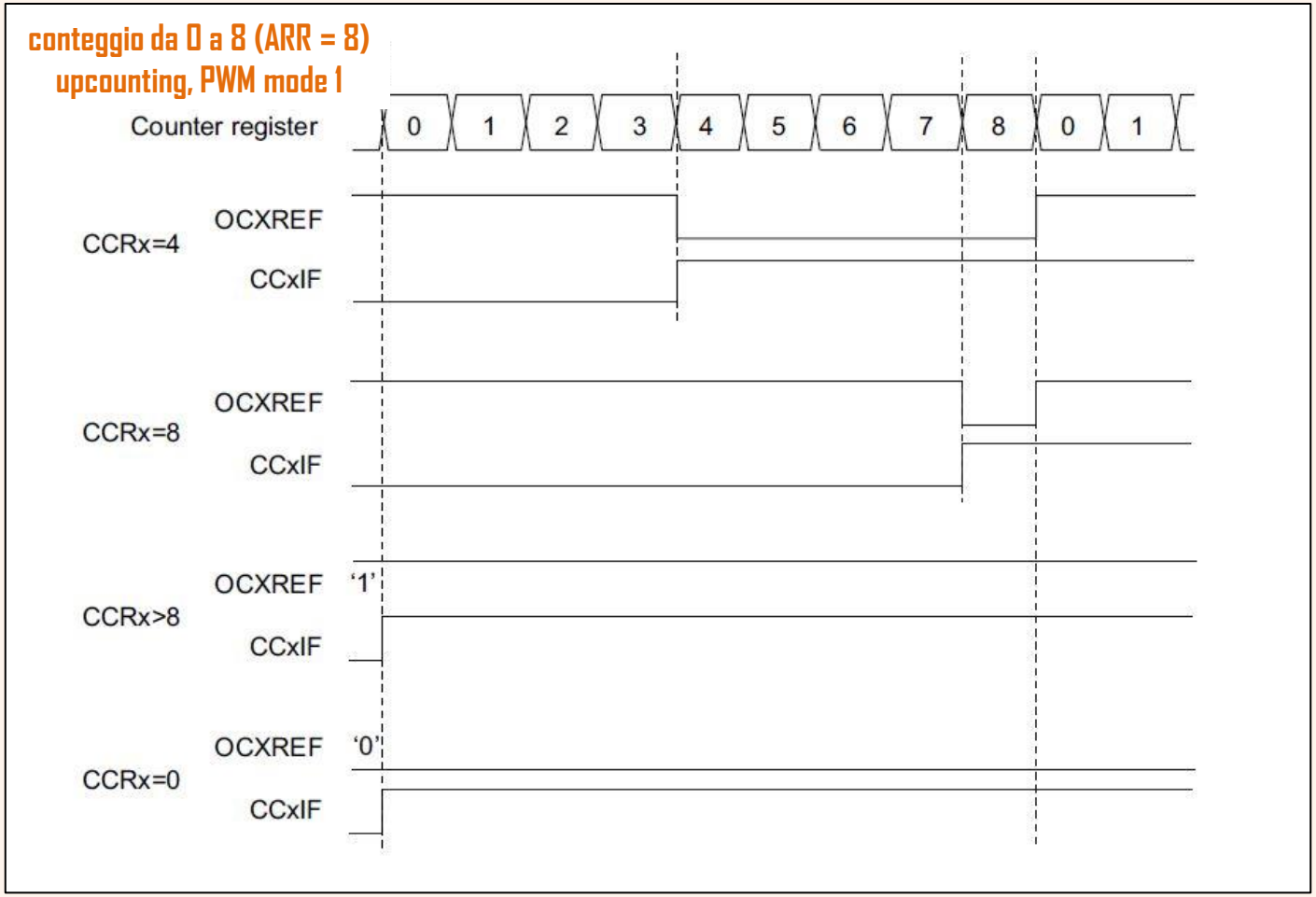
Pulse Width Modulation nell'ARMv6-M

- La PWM nell'ARMv6-M viene **gestita** attraverso due **registri** in particolare (esempio per il timer y)
  - **TIMy_ARR**: determina la **frequenza** della PWM
  - **TIMy_CCRx**: determina il **duty cycle** della PWM per il canale x
- È possibile decidere l'**alternanza dei semi-periodi** positivo e negativo **della PWM** attraverso il mode (OCxM bit del registro TIMy_CCMRx):
  - OCxM = 3'b110 – **PWM mode 1**
  - OCxM = 3'b111 – **PWM mode 2**

Edge-Aligned PWM

- Considerando un **canale** di **capture/compare** con **indice x** nel **timer y**, il segnale di riferimento dell'edge-aligned PWM **OCxREF** se il contatore va in **upcounting** (DIR=0 nel registro TIMy_CR1):
  - resta **alto finché** il conteggio è minore del valore di capture/compare del canale x (**TIMy_CNT < TIMy_CCRx**), **dopodiché va basso** (**PWM mode 1**)
  - resta **basso finché** il conteggio è minore del valore di capture/compare del canale x (**TIMy_CNT < TIMy_CCRx**), **dopodiché va alto** (**PWM mode 2**)

Edge-Aligned PWM



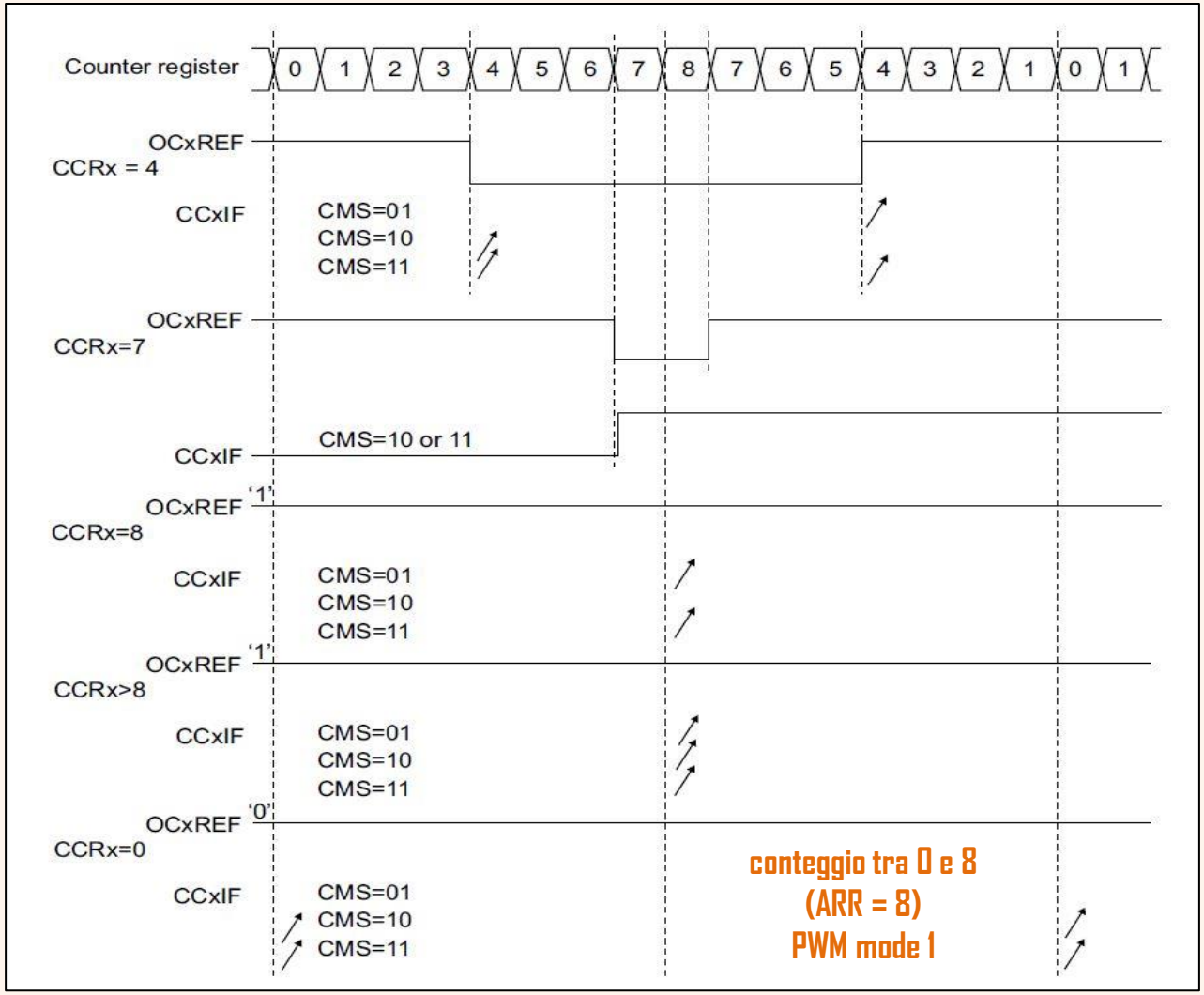
Edge-Aligned PWM

- Considerando un **canale** di **capture/compare** con **indice x** nel **timer y**, il segnale di riferimento dell'edge-aligned PWM **OCxREF** se il contatore va in **downcounting** (DIR=0 nel registro TIMy_CR1):
  - resta **basso finché** il conteggio è maggiore del valore di capture/compare del canale x (**TIMy_CNT > TIMy_CCRx**), **dopodiché va alto** (**PWM mode 1**)
  - resta **alto finché** il conteggio è maggiore del valore di capture/compare del canale x (**TIMy_CNT > TIMy_CCRx**), **dopodiché va basso** (**PWM mode 2**)

Center-Aligned PWM

- Considerando un **canale** di **capture/compare** con **indice x** nel **timer y**, il segnale di riferimento della center-aligned PWM **OCxREF** quando si ha **PWM mode 1**:
  - resta **alto finché** il conteggio è minore del valore di capture/compare del canale x (**TIMy_CNT < TIMy_CCRx**) durante la fase di **upcounting**
  - **va basso finchè** il conteggio è maggiore del valore di capture/compare del canale x (**TIMy_CNT > TIMy_CCRx**) durante la **parte restante** di **upcounting** e in quella di **downcounting**
  - torna **alto** nella **parte restante** del **downcounting** (**TIMy_CNT ≤ TIMy_CCRx**)

Center-Aligned PWM



Center-Aligned PWM

- Considerando un **canale** di **capture/compare** con **indice x** nel **timer y**, il segnale di riferimento della center-aligned PWM **OCxREF** quando si ha **PWM mode 2**:
  - resta **basso finché** il conteggio è minore del valore di capture/compare del canale x (**TIMy_CNT < TIMy_CCRx**) durante la fase di **upcounting**
  - **va alto finchè** il conteggio è maggiore del valore di capture/compare del canale x (**TIMy_CNT > TIMy_CCRx**) durante la **parte restante** di **upcounting** e in quella di **downcounting**
  - torna **basso** nella **parte restante** del **downcounting** (**TIMy_CNT ≤ TIMy_CCRx**)

Center-Aligned PWM

- La particolare **modalità** di **conteggio center-aligned** (CMS!=2'b00 nel registro TIMx_CR1) considerata modifica solamente la generazione dei **flag di interrupt** pendente di capture/compare **CCxIF**
  - Center-aligned **mode 1** (CMS=2'b01): in **downcounting**, quando **TIMy_CNT = TIMy_CCRx**
  - Center-aligned **mode 2** (CMS=2'b10): in **upcounting**, quando **TIMy_CNT = TIMy_CCRx**
  - Center-aligned **mode 3** (CMS=2'b11): in **upcounting** e in **downcounting**, quando **TIMy_CNT = TIMy_CCRx**

PWM

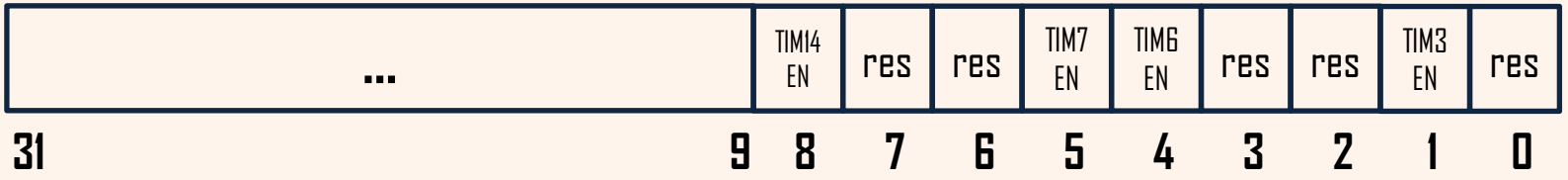
- Per **abilitare la PWM** sul canale capture/compare di indice x del timer y occorre
  - configurare i **bit CCxS** del registro **TIMy_CCMR** in modo che il canale sia usato come output (**2'b00**, valore assunto di default)
  - configurare i **bit OCxM** del registro **TIMy_CCMR** con il **mode desiderato** (3'b110 per PWM mode 1, 3'b111 per PWM mode 2)
  - se si vuole che la **configurazione** sia subito **effettiva**
    - settare il **bit di preload OCxPE** corrispondente nel registro **TIMy_CCMRx**
    - forzare il **caricamento dei registri di preload** settando il **bit UG** nel registro **TIMy_EGR** (simulazione di un evento di update)

PWM

- Per **abilitare la PWM** sul canale capture/compare di indice x del timer y occorre
  - configurare l'**uscita OCx** dal registro **CCER**
    - **abilitarla** settando il **bit CCxE**
    - decidere la **polarità** rispetto a **OCxREF** attraverso il **bit CCxP** (0 → OCx=OCxREF, 1 → OCx=!OCxREF)
  - eventualmente configurare l'**uscita OCxN** dal registro **CCER**
    - **abilitarla** settando il **bit CCxNE**
    - decidere la **polarità** rispetto al **negato di OCxREF** attraverso il **bit CCxNP** (0 → OCx=!OCxREF, 1 → OCx=OCxREF), se abilitato anche OCx

Configurazione della PWM

- **APB Peripheral Clock Enable Register 1 (RCC_APB1ENR)**: registro che abilita il clock interno (8 MHz di default) per i timer dell'ARMv6-M



- **APB Peripheral Clock Enable Register 2 (RCC_APB2ENR)**: registro che abilita il clock interno (8 MHz di default) per i timer dell'ARMv6-M

Configurazione della PWM

- **Timer Control Register 1 (TIMx_CR1)**: registro di **controllo** del timer

  - **CKD, ClocK Division** fattore di divisione della frequenza del clock del timer per funzionalità secondarie
  - **ARPE, Auto-Reload Preload Enable** abilita (ARPE=1) o meno (ARPE=0) il buffering del registro TIMx_ARR
  - **CMS, Center Aligned Mode Selection** seleziona il conteggio center-aligned: non abilitato (CMS=2'b00), abilitato con interrupt output compare attivi solo in upcounting (CMS=2'b01), solo in downcounting (CMS=2'b10), o sia in upcounting che in downcounting (CMS=2'b11)
  - **DIR, DIRection** seleziona la direzione del conteggio: upcounting (DIR=0) o downcounting (DIR=1)
  - **URS, Update Request Source** decide se l'evento di update (UEV) viene generato solo dal counter overflow (URS=0) o anche dal settaggio del bit UG (URS=1)
  - **UDIS, Update Disable** abilita (UDIS=0) o disabilita (UDIS=1) l'evento di update (UEV)
  - **CEN, Counter Enable** abilita (CEN=1) o disabilita (CEN=0) il contatore

Configurazione della PWM

- **Timer Interrupt Enable Register (TIMx_DIER)**: registro di abilitazione degli **interrupt** del timer

  - **CCxIE, Capture/Compare x Interrupt Enable** abilita (CCxIE=1) o disabilita (CCxIE=0) l'interrupt derivante dalle modalità capture input o compare output sul canale x (ve ne possono essere fino a 4 nei timer in relazione al numero di canali di capture/compare)
  - **UIE, Update Interrupt Enable** abilita (UIE=1) o disabilita (UIE=0) l'interrupt derivante dall'evento di update

Configurazione della PWM

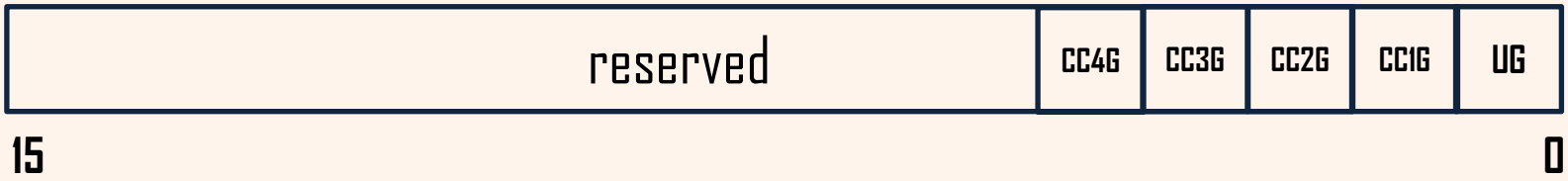
- **Timer Status Register (TIMx_SR)**: registro di **stato** del timer

  - **CCxOF, Capture/Compare x Overcapture Flag** flag che se settato indica la generazione di un nuovo interrupt con modalità input capture o output compare dal canale x prima che il precedente sia stato gestito*
  - **CCxIF, Capture/Compare x Interrupt Flag** flag che se settato indica la generazione di un interrupt con modalità input capture o output compare dal canale x*
  - **UIF, Update Interrupt Flag** flag che se settato indica la generazione di un interrupt tramite evento di update

* ve ne possono essere fino a 4 nei timer general purpose in relazione al numero di canali di capture/compare

Configurazione della PWM

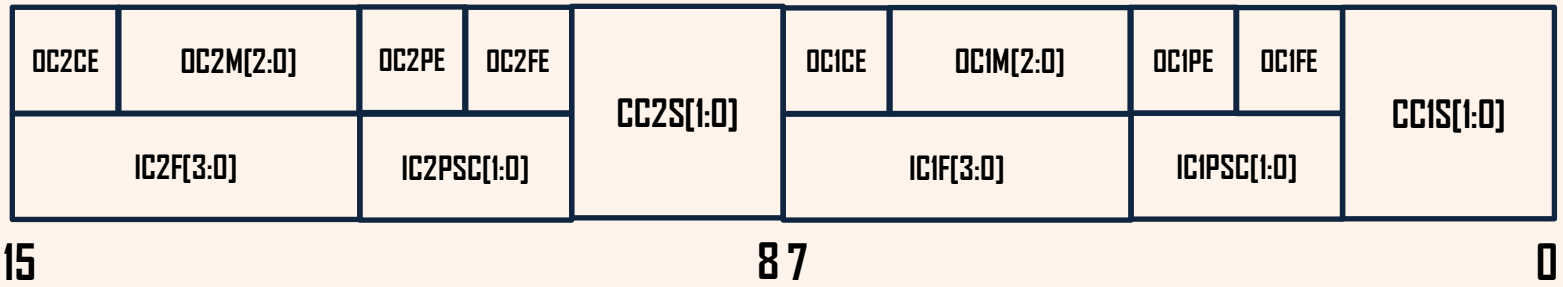
- **Timer Event Generation Register (TIMx_EGR)**: registro di gestione della **generazione di eventi** del timer



  - **CCxG, Capture/Compare 1 Generation** abilita (se CCxG=1) la generazione di un nuovo evento con modalità input capture o output compare dal canale x, ovvero setta il bit CCxIF o CCxOF e, se i relativi interrupt sono abilitati, li solleva (ve ne possono essere fino a 4 nei timer general purpose in relazione al numero di canali di capture/compare)
  - **UG, Update Generation** abilita (se UG=1) la generazione di un nuovo evento di update, che a sua volta porta al reset del contatore e all'update dei registri ad esso relativi

Configurazione della PWM

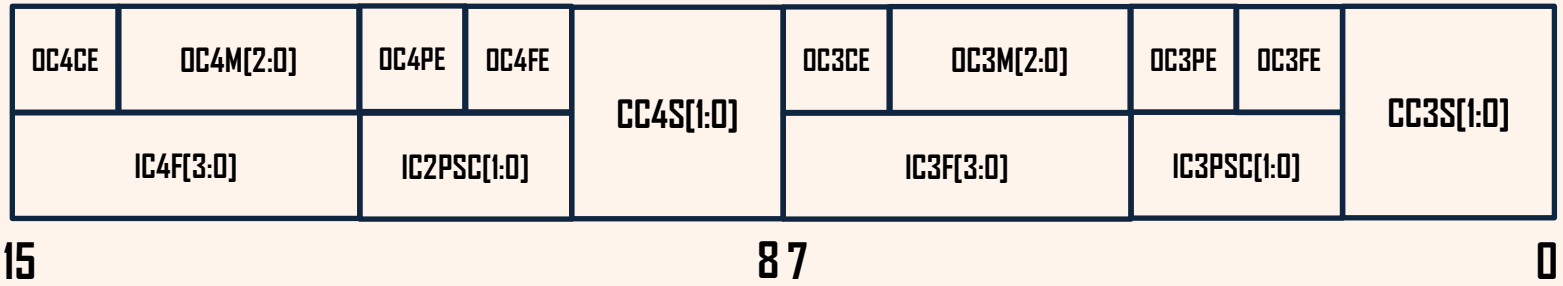
- **Timer Capture/Compare Mode Register 1 (TIMx_CCMR1)**: registro di selezione della modalità **capture/compare** del timer



- **CCxS, Capture/Compare x Selection** definisce la direzione del canale (output se CCxS=2'b00)
- **OCxFE, Output Compare x Fast Enable** se settato abilita la modalità fast per il canale
- **OCxPE Output Compare x Preload Enable** se settato abilita il preload per il canale
- **OCxM Output Compare x Mode** definisce la modalità di output compare (PWM se 110 o 111)
- **OCxCE Output Compare x Clear Enable** se settato abilita la disabilitazione di OCxREF da ETRF

Configurazione della PWM

- **Timer Capture/Compare Mode Register 2 (TIMx_CCMR2)**: registro di selezione della modalità **capture/compare** del timer



- **CCxS, Capture/Compare x Selection** definisce la direzione del canale (output se CCxS=2'b00)
- **OCxFE, Output Compare x Fast Enable** se settato abilita la modalità fast per il canale
- **OCxPE Output Compare x Preload Enable** se settato abilita il preload per il canale
- **OCxM Output Compare x Mode** definisce la modalità di output compare (PWM se 110 o 111)
- **OCxCE Output Compare x Clear Enable** se settato abilita la disabilitazione di OCxREF da ETRF

Configurazione della PWM

- **TIMx_CCMR1/2**: output compare mode (CCxS = 2'b00)

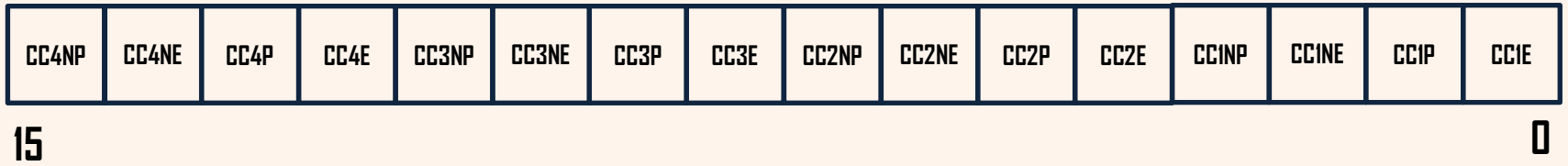
OCxM[2:0]	output compare 1 mode
000	frozen (the comparison between TIMxCCR1 and the TIMxCNT)
001	set the channel 1 to active level on match (OC1REF = 1 if TIMxCCR1==TIMxCNT)
010	set the channel 1 to inactive level on match (OC1REF = 0 if TIMxCCR1==TIMxCNT)
011	channel 1 toggle on match (OC1REF = !OC1REF if TIMxCCR1==TIMxCNT)
100	force inactive level (OC1REF = 0)
101	force active level (OC1REF = 1)
110	PWM mode 1
111	PWM mode 2

OCxPE	output compare x preload enable
0	preload register on TIMx_CCR1 disable (TIMxCCR1 can be written at anytime)
1	preload register on TIMx_CCR1 disable (read/write access preload register, TIMxCCR1 loaded at udate event)

OCxFE	output compare x fast enable
0	minimum delay to activate the output when an edge occurs is 5 clock cycles
1	minimum delay to activate the output when an edge occurs is reduced to 3 clock cycles

Configurazione della PWM

- **Timer Capture/Compare Enable Register (TIMx_CCER)**: abilitazione delle modalità **capture/compare** del canale 1 del timer



  - **CC1NP, Capture/Compare 1 Complementary Output Polarity** definisce il livello di attività del segnale di uscita complementare generato rispetto al negato del segnale OCxREF
  - **CCxNE, Capture/Compare x Complementary Output Enable** se settato abilita la generazione del segnale di uscita complementare (che si riferisce al negato del segnale OCxREF)
  - **CCxP, Capture/Compare x Output Polarity** definisce il livello di attività del segnale di uscita generato rispetto al segnale OCxREF
  - **CCxE, Capture/Compare x Output Enable** se settato abilita la generazione del segnale di uscita

Configurazione della PWM

- **Timer Counter (TIMx_CNT)**: registro che contiene il conteggio attuale del timer

- **Timer Prescaler (TIMx_PSC)**: registro che contiene il fattore di prescaling del clock del timer

Configurazione della PWM

- **Timer Auto-Reload Register (TIMx_ARR)**: registro che contiene il valore da ricaricare nel timer all'update

- **Timer Capture/Compare Register y (TIMx_CCRy)**: registro con il valore di capture/compare del canale y

Configurazione della PWM

MODERy[1:0]	OTYPERy	PUPDRy[1:0]	I/O pin configuration
01	0	00	GP output (PP)
	0	01	GP output (PP+PU)
	0	10	GP output (PP+PD)
		11	reserved
		00	GP output (OD)
	1	01	GP output (OD+PU)
	1	10	GP output (OD+PD)
	1	11	reserved
10	0	00	AF (PP)
	0	01	AF (PP+PU)
	0	10	AF (PP+PD)
	0	11	reserved
	1	00	AF (OD)
	1	01	AF (OD+PU)
	1	10	AF (OD+PD)
	1	11	reserved

output driven by peripherals

Configurazione della PWM

- In ogni porta è necessario configurare almeno **3 registri a** per settare l'**alternate function (AF)**:
  - **Port Mode Register (GPIOx_MODER)**: registro per definire la modalità di utilizzo dei bit, accessibile in **lettura e scrittura**

  - **Alternate Function Registers (GPIOx_AFRL/H)**: per la selezione della periferica sorgente (**lettura e scrittura**)

Configurazione della PWM

pin name	AF0	AF1	AF2	AF3	AF4	AF5	AF6
PA0	-	USART2_CTS	-	-	-	-	-
PA1	EVENTOUT	USART2_RTS	-	-	-	-	-
PA2	TIM15_CH1	USART2_TX	-	-	-	-	-
PA3	TIM15_CH2	USART2_RX	-	-	-	-	-
PA4	SPI1_NSS	USART2_CK	-	-	TIM14_CH1	-	-
PA5	SPI1_SCK	-	-	-	-	-	-
PA6	SPI1_MISO	TIM3_CH1	TIM1_BKIN	-	-	TIM16_CH1	EVENTOUT
PA7	SPI1_MOSI	TIM3_CH2	TIM1_CHIN	-	TIM14_CH1	TIM17_CH1	EVENTOUT
PA8	MCO	USART1_CK	TIM1_CH1	EVENTOUT	-	-	-
PA9	TIM15_BKIN	USART1_TX	TIM1_CH2	-	-	-	-
PA10	TIM15_BKIN	USART1_RX	TIM1_CH3	-	-	-	-
PA11	EVENTOUT	USART1_CTS	TIM1_CH4	-	-	SCL	-
PA12	EVENTOUT	USART1_RTS	TIM1_ETR	-	-	SDA	-
PA13	SWDIO	IR_OUT	-	-	-	-	-
PA14	SWCLK	USART2_TX	-	-	-	-	-
PA15	SPI1_NSS	USART2_RX	-	EVENTOUT	-	-	-

Configurazione della PWM

pin name	AF0	AF1	AF2	AF3	AF4	AF5
PB0	EVENTOUT	TIM3_CH3	TIM1_CH2N	-	-	-
PB1	TIM14_CH1	TIM3_CH4	TIM1_CH3N	-	-	-
PB2	-	-	-	-	-	-
PB3	SPI1_SCK	EVENTOUT	-	-	-	-
PB4	SPI1_MISO	TIM3_CH1	EVENTOUT	-	-	-
PB5	SPI1_MOSI	TIM3_CH2	TIM16_BKIN	I2C1_SMBA	-	-
PB6	USART1_TX	I2C1_SCL	TIM16_CH1N	-	-	-
PB7	USART1_RX	I2C2_SDA	TIM17_CH1N	-	-	-
PB8	-	I2C1_SCL	TIM16_CH1	-	-	-
PB9	IR_OUT	I2C1_SDA	TIM17_CH1	EVENTOUT	-	-
PB10	-	I2C2_SCL	-	-	-	-
PB11	EVENTOUT	I2C2_SDA	-	-	-	-
PB12	SPI2_NSS	EVENTOUT	TIM1_BKIN	-	-	-
PB13	SPI2_SCK	-	TIM1_CH1N	-	-	-
PB14	SPI2_MISO	TIM15_CH1	TIM1_CH2N	-	-	-
PB15	SPI2_MOSI	TIM15_CH2	TIM1_CH3N	TIM15_CH1N	-	-

Configurazione della PWM

pin name	AF0
PC0	EVENTOUT
PC1	EVENTOUT
PC2	EVENTOUT
PC3	EVENTOUT
PC4	EVENTOUT
PC5	-
PC6	TIM3_CH1
PC7	TIM3_CH2
PC8	TIM3_CH3
PC9	TIM3_CH4
PC10	-
PC11	-
PC12	-
PC13	-
PC14	-
PC15	-

pin name	AF0
PD2	TIM3_ETR

pin name	AF0
PF0	.
PF1	-
PF4	-
PF5	-
PF6	I2C2_SCL
PF7	I2C2_SDA

Le porte C, D e F hanno **solo una alternate function possibile** nell'STM32F030R8 per cui **non vi sono registri GPIOC_AFR, GPIOD_AFR, GPIOF_AFR**

Configurazione della PWM

bus	boundary address	size	peripheral
Cortex-M0 Internal Peripherals	0xE000E100 – 0xE000E4FF	1 kB	NVIC
AHB2	0x48001400 – 0x480017FF	1 kB	GPIOF
	0x48000C00 – 0x48000FFF	1 kB	GPIOD
	0x48000800 – 0x48000BFF	1 kB	GPIOC
	0x48000400 – 0x480007FF	1 kB	GPIOB
	0x48000000 – 0x480003FF	1 kB	GPIOA
AHB1	0x40021000 – 0x400213FF	1 kB	RCC
APB	0x40014800 – 0X40014BFF	1 kB	TIM17
	0x40014400 – 0X400147FF	1 kB	TIM16
	0x40014000 – 0X400143FF	1 kB	TIM15
	0x40012C00 – 0x40012FFF	1 kB	TIM1
	0x40010400 – 0x400107FF	1 kB	EXTI
	0x40010000 – 0x400103FF	1 kB	SYSCFG
	0x40002000 – 0x400023FF	1 kB	TIM14
	0x40001400 – 0x400017FF	1 kB	TIM7
	0x40001000 – 0x400013FF	1 kB	TIM6
	0x40000400 – 0x400007FF	1 kB	TIM3

Configurazione della PWM

offset	register	content															
		31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x00	GPIOx_MODER	MODER7[1:0] MODER15[1:0]		MODER6[1:0] MODER14[1:0]		MODER5[1:0] MODER13[1:0]		MODER4[1:0] MODER12[1:0]		MODER3[1:0] MODER11[1:0]		MODER2[1:0] MODER10[1:0]		MODER1[1:0] MODER9[1:0]		MODER0[1:0] MODER8[1:0]	
0x04	GPIOx_OTYPER	OT[15:0] res															
0x08	GPIOx_OSPEEDR	OSPEEDR7[1:0] OSPEEDR15[1:0]		OSPEEDR6[1:0] OSPEEDR14[1:0]		OSPEEDR5[1:0] OSPEEDR13[1:0]		OSPEEDR4[1:0] OSPEEDR12[1:0]		OSPEEDR3[1:0] OSPEEDR11[1:0]		OSPEEDR2[1:0] OSPEEDR10[1:0]		OSPEEDR1[1:0] OSPEEDR9[1:0]		OSPEEDR0[1:0] OSPEEDR8[1:0]	
0x0C	GPIOx_PUPDR	PUPDR7[1:0] PUPDR15[1:0]		PUPDR6[1:0] PUPDR14[1:0]		PUPDR5[1:0] PUPDR13[1:0]		PUPDR4[1:0] PUPDR12[1:0]		PUPDR3[1:0] PUPDR11[1:0]		PUPDR2[1:0] PUPDR10[1:0]		PUPDR1[1:0] PUPDR9[1:0]		PUPDR0[1:0] PUPDR8[1:0]	
0x10	GPIOx_IDR	IDR[15:0] res															
0x14	GPIOx_ODR	ODR[15:0] res															
0x18	GPIOx_BSRR	BS[15:0] BR[15:0]															
0x1C	GPIOx_LCKR	LCK[15:0] res															LCKK
0x20	GPIOx_AFRL	AFSEL3[3:0] AFSEL7[3:0]				AFSEL2[3:0] AFSEL6[3:0]				AFSEL1[3:0] AFSEL5[3:0]				AFSEL0[3:0] AFSEL4[3:0]			
0x24	GPIOx_AFRH	AFSEL11[3:0] AFSEL15[3:0]				AFSEL10[3:0] AFSEL14[3:0]				AFSEL9[3:0] AFSEL13[3:0]				AFSEL8[3:0] AFSEL12[3:0]			
0x28	GPIOx_BRR	BRR[15:0] res															

Configurazione della PWM

offset	register	content															
		31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x00	TIMx_CR1	res						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
		res															
0x0C	TIMx_DIER	...											CC4IE	CC3IE	CC2IE	CC1IE	UIE
		res															
0x10	TIMx_SR	res			CC4OF	CC3OF	CC2OF	CC1OF	...				CC4IF	CC3IF	CC2IF	CC1IF	UIF
		res															
0x14	TIMx_EGR	...											CC4G	CC3G	CC2G	CC1G	UG
		res															
0x18	TIMx_CCMR1	OC2CE/ IC2F[3]	OC2M[2] /IC2F[2]	OC2M[1] /IC2F[1]	OC2M[0] /IC2F[0]	OC2PE/ IC2PSC[1]	OC2FE/ IC2PSC[0]	CC2S[1:0]		OC1CE/ IC1F[3]	OC1M[2] /IC1F[2]	OC1M[1] /IC1F[1]	OC1M[0] /IC1F[0]	OC1PE/ IC1PSC[1]	OC1FE/ IC1PSC[0]	CC1S[1:0]	
		res															
0x1C	TIMx_CCMR2	OC4CE/ IC4F[3]	OC4M[2] /IC4F[2]	OC4M[1] /IC4F[1]	OC4M[0] /IC4F[0]	OC4PE/ IC4PSC[1]	OC4FE/ IC4PSC[0]	CC4S[1:0]		OC3CE/ IC3F[3]	OC3M[2] /IC3F[2]	OC3M[1] /IC3F[1]	OC3M[0] /IC3F[0]	OC3PE/ IC3PSC[1]	OC3FE/ IC3PSC[0]	CC3S[1:0]	
		res															
0x20	TIMx_CCER	CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
		res															
0x24	TIMx_CNT	CNT[15:0]															
		res															
0x28	TIMx_PSC	PSC[15:0]															
		res															
0x2C	TIMx_ARR	ARR[15:0]															
		res															

Configurazione della PWM

offset	register	content 31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x34	TIMx_CCR1	CCR1[15:0]															
		res															
0x38	TIMx_CCR2	CCR2[15:0]															
		res															
0x3C	TIMx_CCR3	CCR3[15:0]															
		res															
0x40	TIMx_CCR4	CCR4[15:0]															
		res															
0x44	TIMx_BDTR	MOE	AOE	BKP	OSSR	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
		res															
0x14	RCC_AHBENR	...															
		...									IOPF_EN	res	IOPD_EN	IOPC_EN	IOPB_EN	IOPA_EN	...
0x18	RCC_APB2ENR	...				TIM1_EN		...									SYSCFG_EN
		...													TIM17_EN	TIM16_EN	TIM15_EN
0x1C	RCC_APB1ENR	...							TIM14_EN	res		TIM7_EN	TIM6_EN	res		TIM3_EN	res
		...															

Configurazione della PWM

bus	boundary address	size	peripheral	description	edge-aligned PWM (E) and/or center-aligned PWM (C)	number of CC channels
APB	0x40014800 – 0X40014BFF	1 kB	TIM17	general purpose timer	E	1
	0x40014400 – 0X400147FF	1 kB	TIM16	general purpose timer	E	1
	0x40014000 – 0X400143FF	1 kB	TIM15	general purpose timer	E	2
	0x40012C00 – 0x40012FFF	1 kB	TIM1	advanced control timer	E, C	4
	0x40002000 – 0x400023FF	1 kB	TIM14	general purpose timer	E	1
	0x40001400 – 0x400017FF	1 kB	TIM7	basic timer	-	0
	0x40001000 – 0x400013FF	1 kB	TIM6	basic timer	-	0
	0x40000400 – 0x400007FF	1 kB	TIM3	general purpose timer	E, C	4
AHB1	0x40021000 – 0x400213FF	1 kB	RCC	reset and clock control	-	-

Esempio: Edge-Aligned PWM

```
/* (0) Enable clock on TIM14 */
/* (1) Set prescaler to 7, so CLK_IN/8 i.e 1MHz */
/* (2) Set ARR=7, the period is 8 us (1 us clock period) */
/* (3) Set CCRx=4, the signal will be high during 4 us */
/* (4) Select PWM mode 1 on OC1 (OC1M = 110),
disable preload register on OC1 (OC1PE = 1) */
/* (5) Select active high polarity on OC1 (CC1P = 0, reset value),
enable the output on OC1 (CC1E = 1)*/
/* (6) Enable counter (CEN = 1) select edge aligned mode (CMS = 00,
reset value) select direction as upcounter (DIR = 0, reset value) */
/* (7) Force update generation (UG = 1) */
RCC->APB1ENR |= RCC_APB1ENR_TIM14EN; /* (0) */
TIM14->PSC = 7; /* (1) */
TIM14->ARR = 7; /* (2) */
TIM14->CCR1 = 4; /* (3) */
TIM14->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1
        | TIM_CCMR1_OC1PE; /* (4) */
TIM14->CCER |= TIM_CCER_CC1E; /* (5) */
TIM14->CR1 |= TIM_CR1_CEN; /* (6) */
TIM14->EGR |= TIM_EGR_UG; /* (7) */
```

Esempio: Edge-Aligned PWM

```
/* (0) Enable clock on GPIOA */
/* (1) Set the mode of pin PA4 to output with source from alternate
function */
/* (2) Set the selector of alternate function for pin PA4 to 0x04
(TIM14_CH1) */
RCC->AHBENR |= RCC_AHBENR_GPIOAEN; /* (0) */
GPIOA->MODER |= 0x200; /* (1)*/
GPIOA->AFR[0] |= 0x04 << GPIO_AFRL_AFRL4_Pos; /* (2) */
```

Esempio: Edge-Aligned PWM

```
/* (0) Enable clock on TIM14 */
/* (1) Set prescaler to 7, so CLK_IN/8 i.e 1MHz */
/* (2) Set ARR=7, the period is 7 us (1 us clock period) */
/* (3) Set CCRx=4, the signal will be high during 4 us */
/* (4) Select PWM mode 1 on OC1 (OC1M = 110),
enable preload register on OC1 (OC1PE = 1) */
/* (5) Select active high polarity on OC1 (CC1P = 0, reset value),
enable the output on OC1 (CC1E = 1)*/
/* (6) Enable counter (CEN = 1) select edge aligned mode (CMS = 00,
reset value) select direction as upcounter (DIR = 0, reset value) */
/* (7) Force update generation (UG = 1) */
*((int*) 0x4002101C) |= 0x100; /* (0) RCC->APB1ENR */
*((int*) 0x40002028) = 7; /* (1) TIM14->PSC */
*((int*) 0x4000202C) = 7; /* (2) TIM14->ARR */
*((int*) 0x40002034) = 0; /* (3) TIM14->CCR1 */
*((int*) 0x40002018) |= 0x40 | 0x20
        | 0x8; /* (4) TIM14->CCMR1 */
*((int*) 0x40002020) |= 0x1; /* (5) TIM14->CCER */
*((int*) 0x40002000) |= 0x1; /* (6) TIM14->CR1 */
*((int*) 0x40002014) |= 0x1; /* (7) TIM14->EGR */
```

Esempio: Edge-Aligned PWM

```
/* (0) Enable clock on GPIOA */
/* (1) Set the mode of pin PA6 to output with source from alternate
function */
/* (2) Set the selector of alternate function for pin PA4 to 0x04
(TIM14_CH1) */
*((int*) 0x40021014) |= 0x20000; /* (0) RCC->AHBENR */
*((int*) 0x48000000) |= 0x200; /* (1) GPIOA->MODER */
*((int*) 0x48000020) |= 0x04 << 16; /* (2) GPIOA->AFR[0] */
```

Esempio: Center-Aligned PWM

```
/* (0) Enable clock on TIM3 */
/* (1) Set prescaler to 7, so CLK_IN/8 i.e 1MHz */
/* (2) Set ARR=8, the period is 16 us (1 us clock period) */
/* (3) Set CCRx = 7, the signal will be high during 14 us */
/* (4) Select PWM mode 1 on OC1 (OC1M = 110),
disable preload register on OC1 (OC1PE = 1, reset value) */
/* (5) Select active high polarity on OC1 (CC1P = 0, reset value),
enable the output on OC1 (CC1E = 1)*/
/* (6) Enable counter (CEN = 1)
select center-aligned mode 1 (CMS = 01) */
/* (7) Force update generation (UG = 1) */
RCC->APB1ENR |= RCC_APB1ENR_TIM3EN; /* (0) */
TIM3->PSC = 7; /* (1) */
TIM3->ARR = 8; /* (2) */
TIM3->CCR1 = 7; /* (3) */
TIM3->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1
        | TIM_CCMR1_OC1PE; /* (4) */
TIM3->CCER |= TIM_CCER_CC1E; /* (5) */
TIM3->CR1 |= TIM_CR1_CMS_0 | TIM_CR1_CEN; /* (6) */
TIM3->EGR |= TIM_EGR_UG; /* (7) */
```

Esempio: Center-Aligned PWM

```
/* (0) Enable clock on GPIOA */
/* (1) Set the mode of pin PA6 to output with source from alternate
function */
/* (2) Set the selector of alternate function for pin PA6 to 0x01
(TIM3_CH1) */
RCC->AHBENR |= RCC_AHBENR_GPIOAEN; /* (0) */
GPIOA->MODER |= 0x2000; /* (1) */
GPIOA->AFR[0] |= 0x01 << GPIO_AFRL_AFRL6_Pos; /* (2) */
```

Esempio: Center-Aligned PWM

```
/* (0) Enable clock on TIM3 */
/* (1) Set prescaler to 7, so CLK_IN/8 i.e 1MHz */
/* (2) Set ARR=8, the period is 16 us (1 us clock period) */
/* (3) Set CCRx = 7, the signal will be high during 14 us */
/* (4) Select PWM mode 1 on OC1 (OC1M = 110),
enable preload register on OC1 (OC1PE = 1, reset value) */
/* (5) Select active high polarity on OC1 (CC1P = 0, reset value),
enable the output on OC1 (CC1E = 1)*/
/* (6) Enable counter (CEN = 1)
select center-aligned mode 1 (CMS = 01) */
/* (7) Force update generation (UG = 1) */
*((int*) 0x4002101C) |=  0x2; /* (0) RCC->APB1ENR */
*((int*) 0x40000428) = 7; /* (1) TIM3->PSC */
*((int*) 0x4000042C) = 8; /* (2) TIM3->ARR */
*((int*) 0x40000434) = 7; /* (3) TIM3->CCR1 */
*((int*) 0x40000418) |= 0x40 | 0x20
        | 0x8; /* (4) TIM3->CCMR1 */
*((int*) 0x40000420) |= 0x1; /* (5) TIM3->CCER */
*((int*) 0x40000400) |= 0x20 | 0x1; /* (6) TIM3->CR1 */
*((int*) 0x40000414) |= 0x1; /* (7) TIM3->EGR */
```

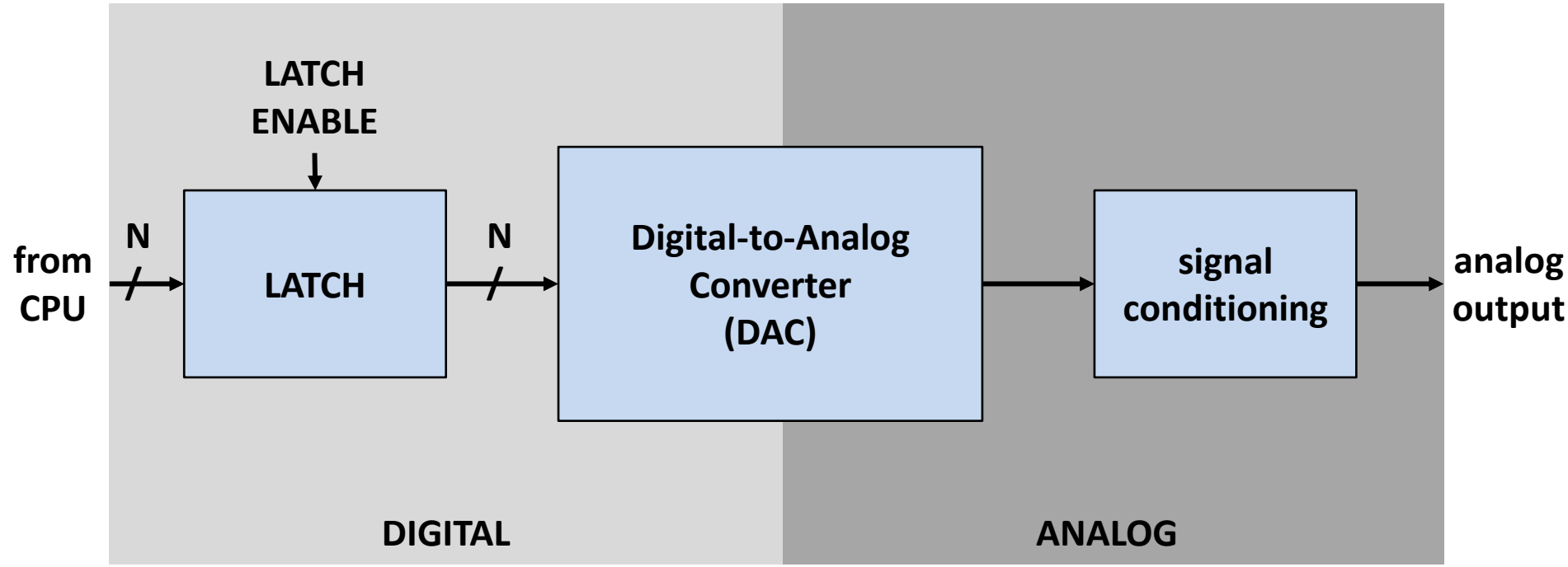
Esempio: Center-Aligned PWM

```
/* (0) Enable clock on GPIOA */
/* (1) Set the mode of pin PA6 to output with source from alternate
function */
/* (2) Set the selector of alternate function for pin PA6 to 0x01
(TIM3_CH1) */
*((int*) 0x40021014) |= 0x20000; /* (0) RCC->AHBENR */
*((int*) 0x48000000) |= 0x2000; /* (1) GPIOA->MODER */
*((int*) 0x48000020) |= 0x01 << 24; /* (2) GPIOA->AFR[0] */
```

Indice

- Uscite Analogiche
  - Pulse Width Modulation
    - PWM negli ARM
  - Digital-to-Analog (D/A) Conversion
- Ingressi Analogici
  - Analog-to-Digital (A/D) Conversion
    - ADC negli ARM

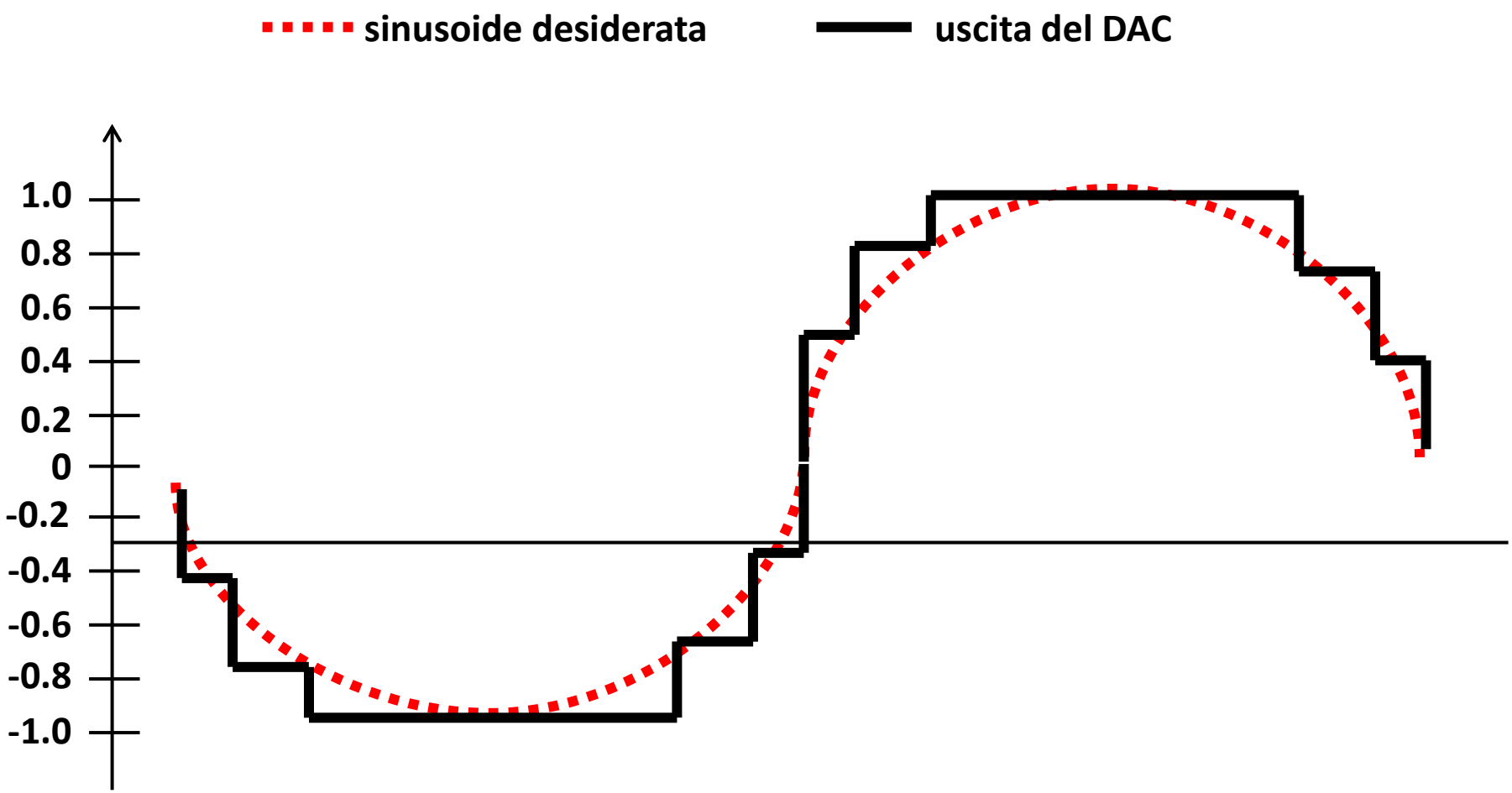
Conversione Digitale-Analogica



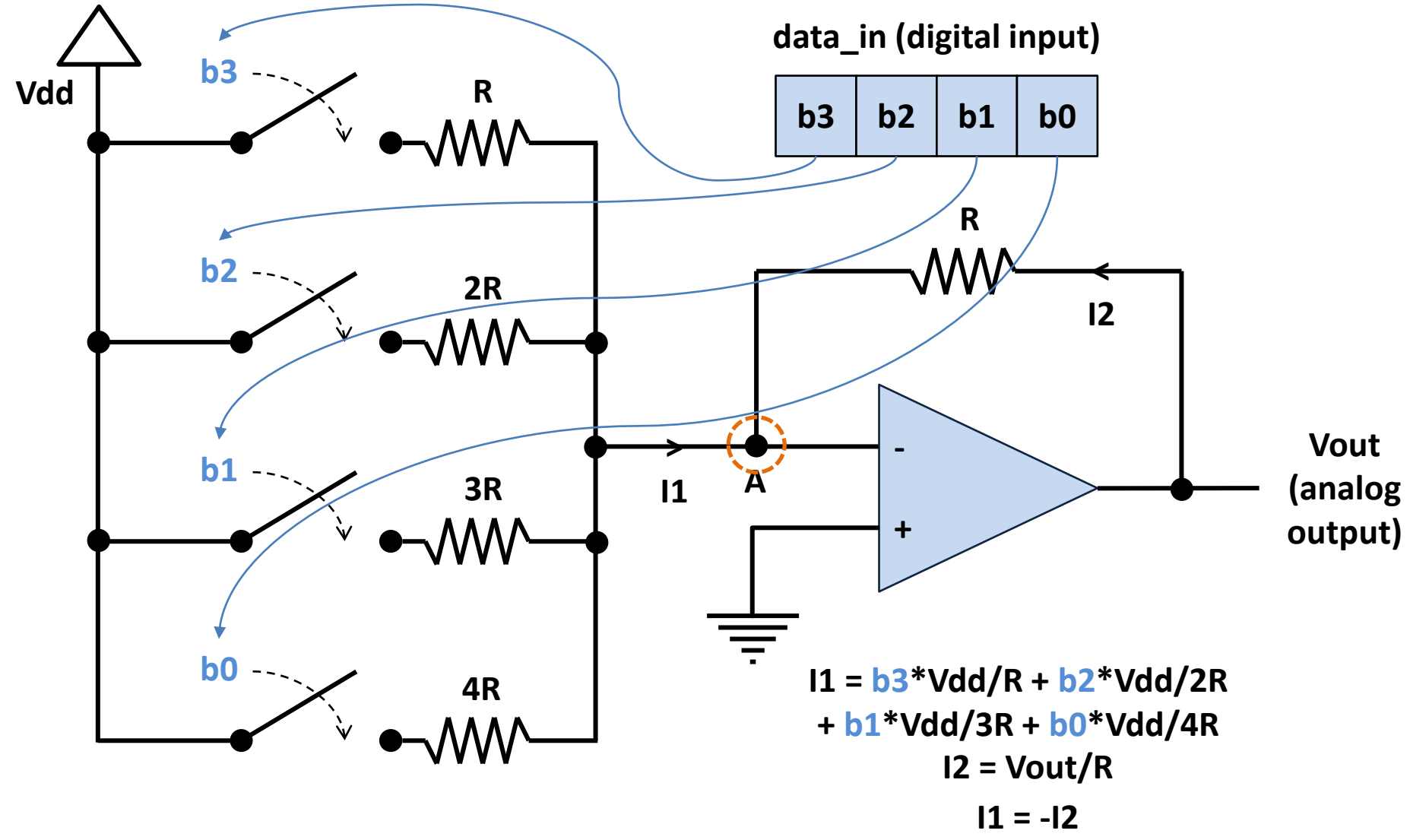
Conversione Digitale-Analogica

- **Un'interfaccia parallela** (N bit) connette il DAC alla **CPU**
- Il latch può essere parte dell'interfaccia o del DAC stesso
- Il **valore digitale** viene **convertito** in un valore **continuo**
- Un blocco di **signal conditioning** può essere utilizzato per **filtrare l'uscita di natura quantizzata**
  - il blocco di signal conditioning fornisce anche isolamento, buffering e amplificazione se necessario

Uscita DA Quantizzata



DAC binary-weighted

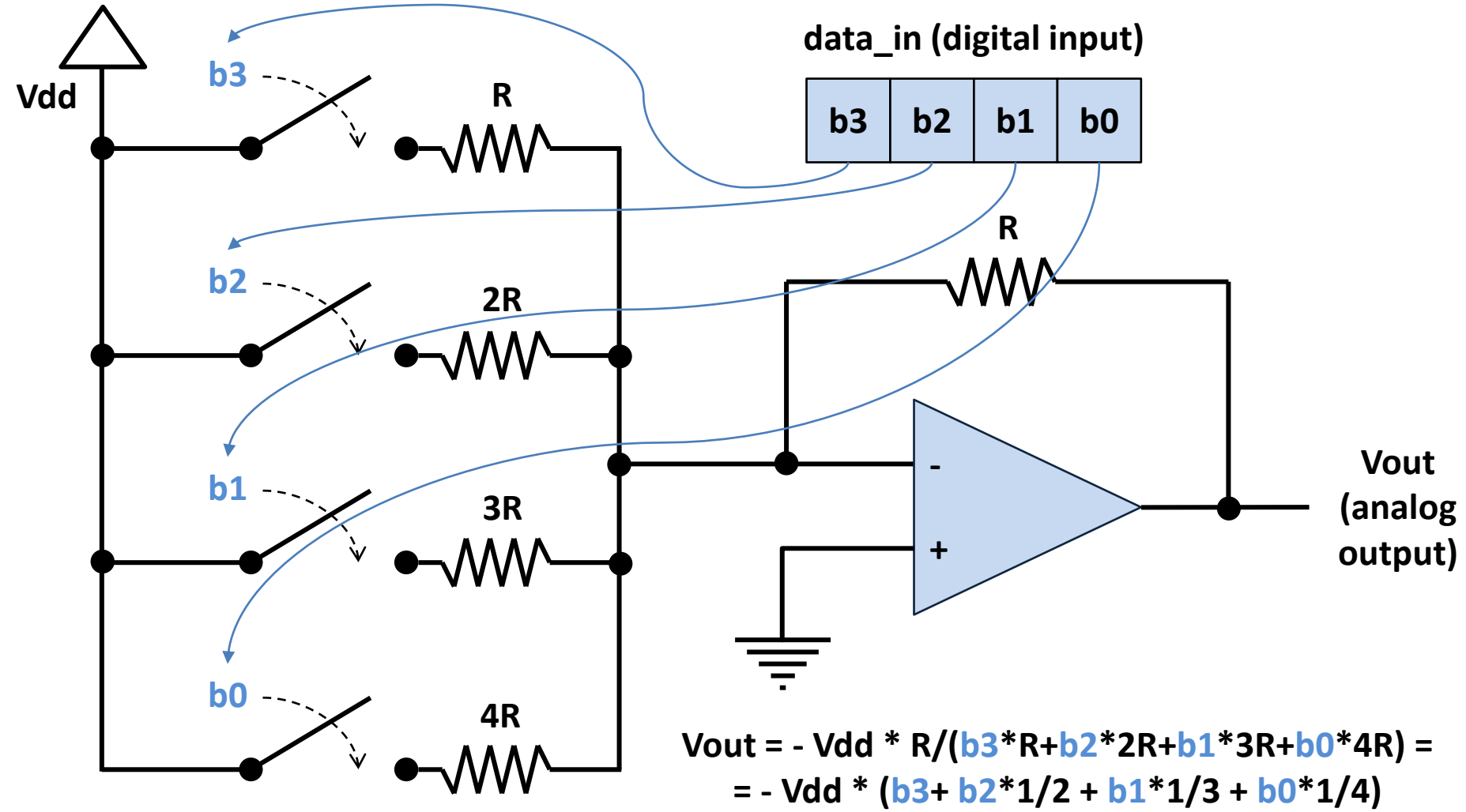


$I1 = b3*Vdd/R + b2*Vdd/2R + b1*Vdd/3R + b0*Vdd/4R$

$I2 = Vout/R$

$I1 = -I2$

DAC binary-weighted

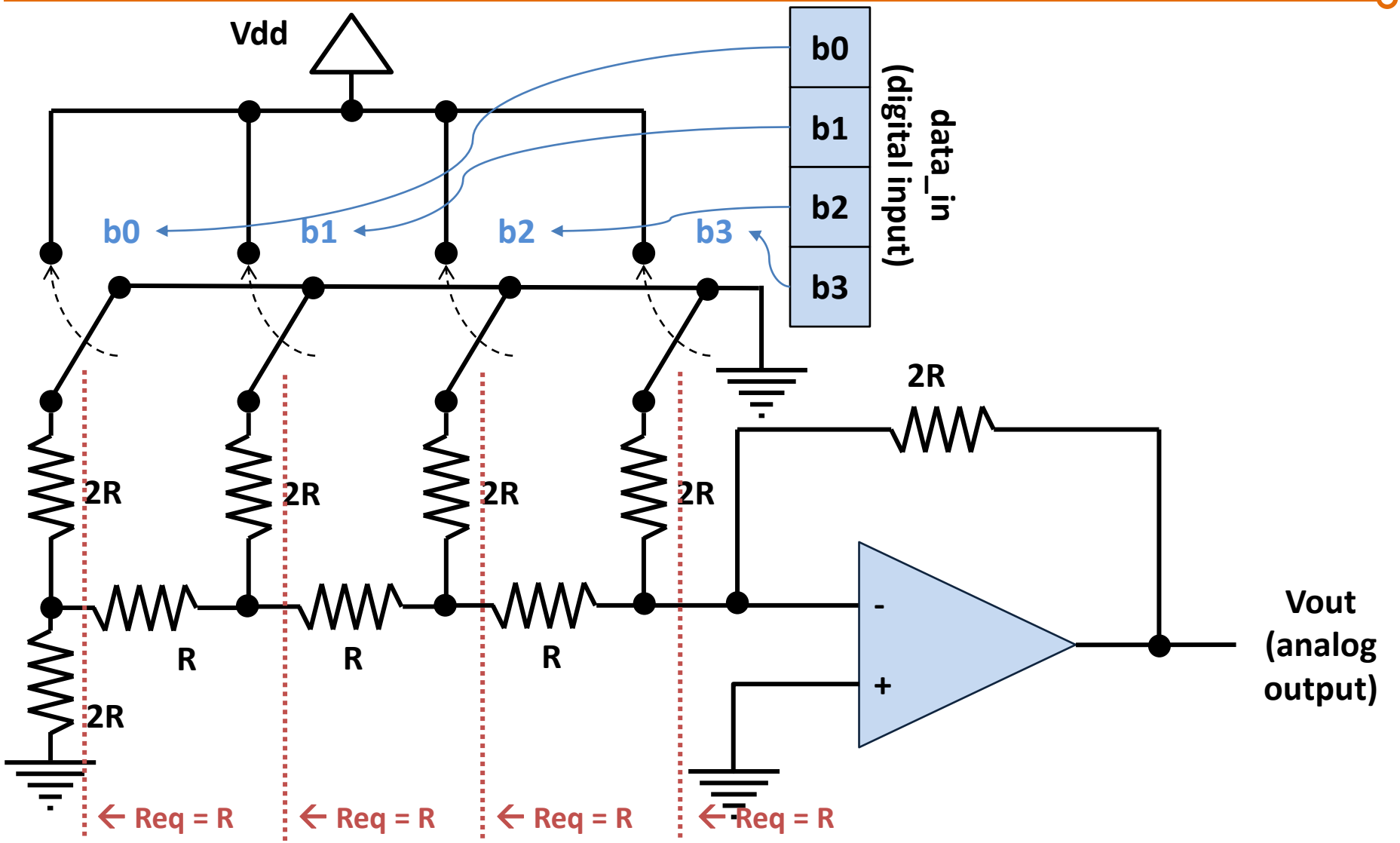


$$Vout = - Vdd * R/(b3*R+b2*2R+b1*3R+b0*4R) =$$
$$= - Vdd * (b3+ b2*1/2 + b1*1/3 + b0*1/4)$$

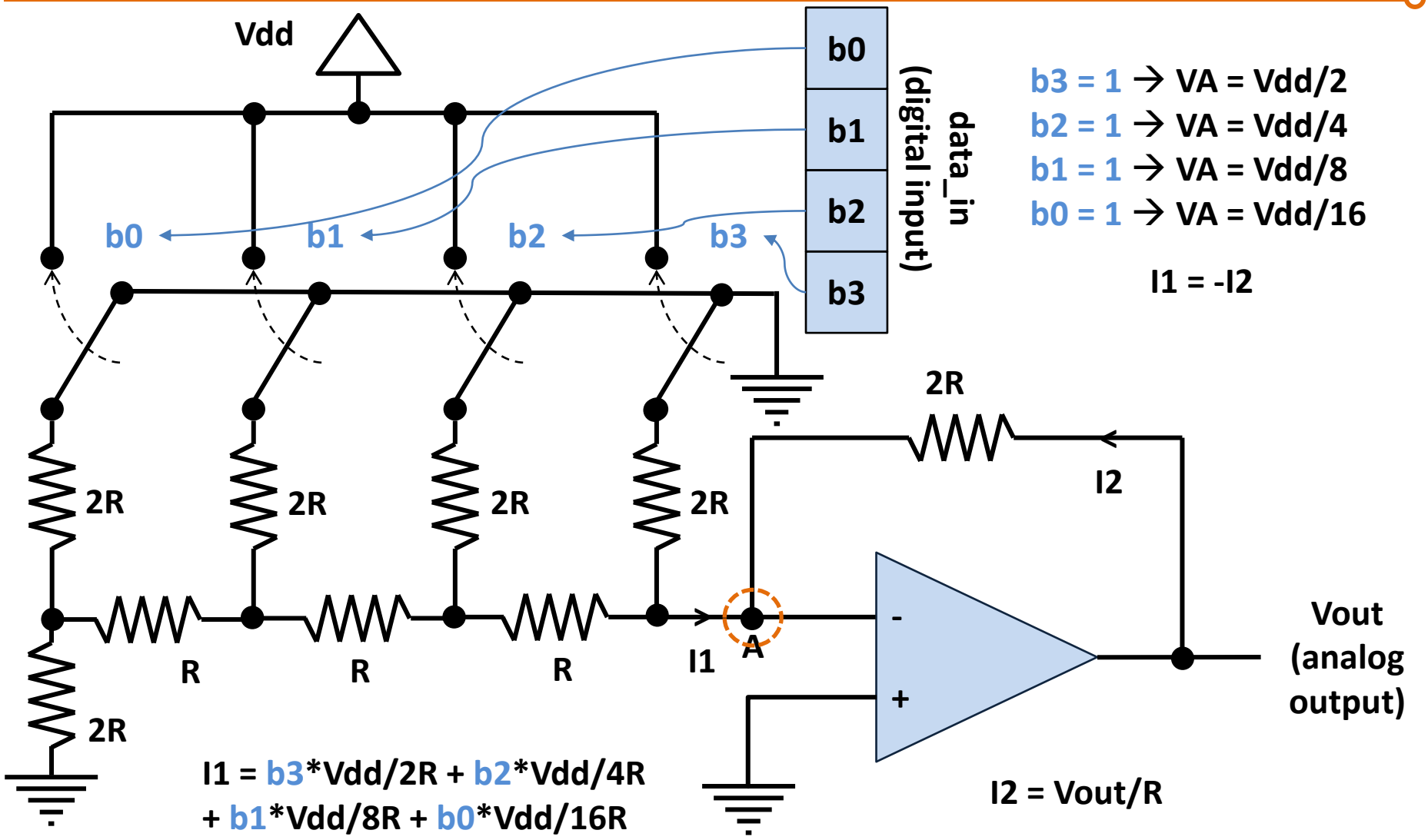
DAC binary-weighted

- Se uno **switch pilotato da un bit viene chiuso** (bit a 1), una **corrente pesata rispetto all'importanza del bit** (correnti maggiori per bit più significativi) viene immessa nell'ingresso invertente dell'amplificatore operazionale
- Per **DAC ad alta risoluzione** (molti bit), occorre un'**ampia gamma di resistori diversi** il che può dare **problemi di stabilità termica e di switching**

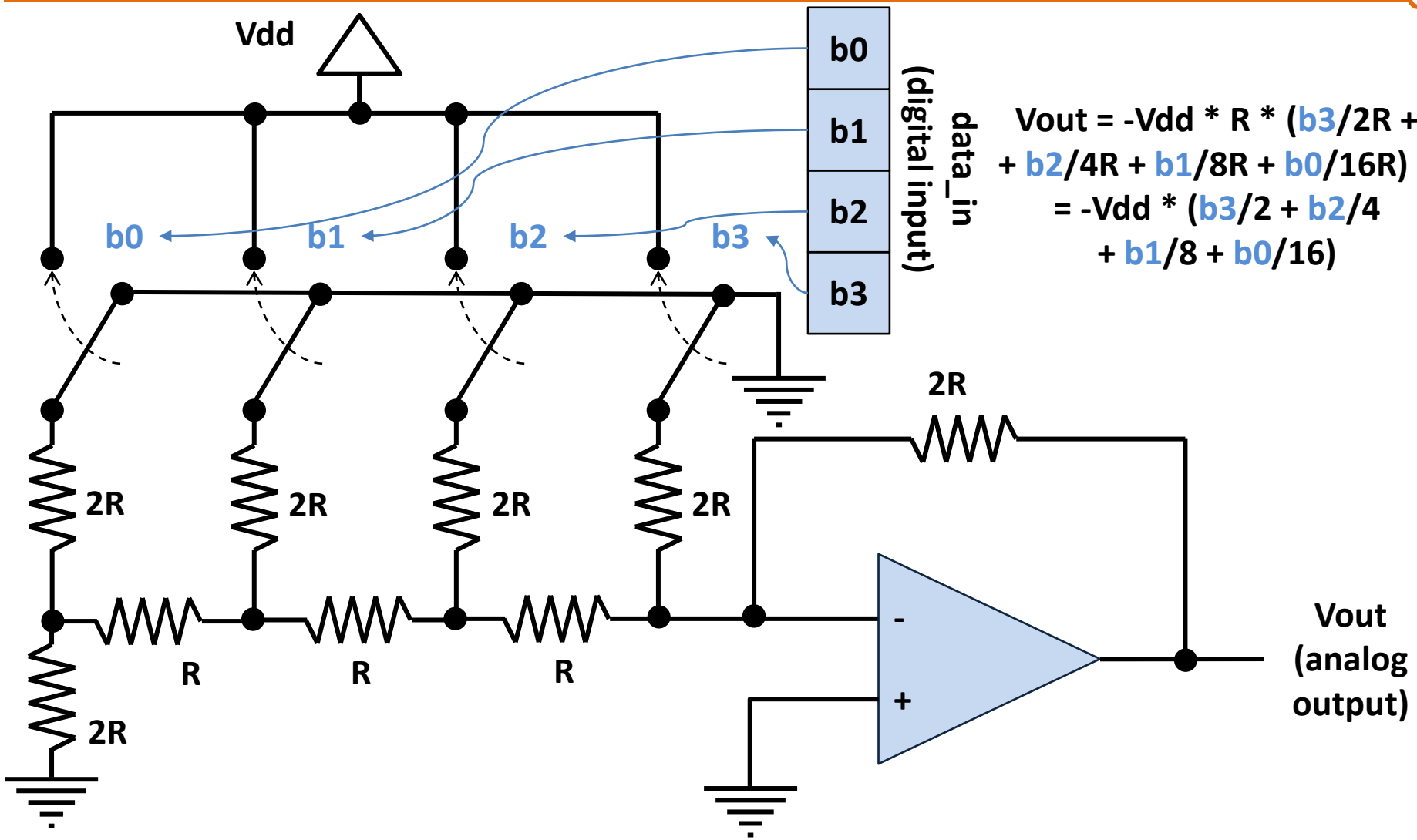
DAC R-2R Ladder



DAC R-2R Ladder



DAC R-2R Ladder



DAC R-2R Ladder

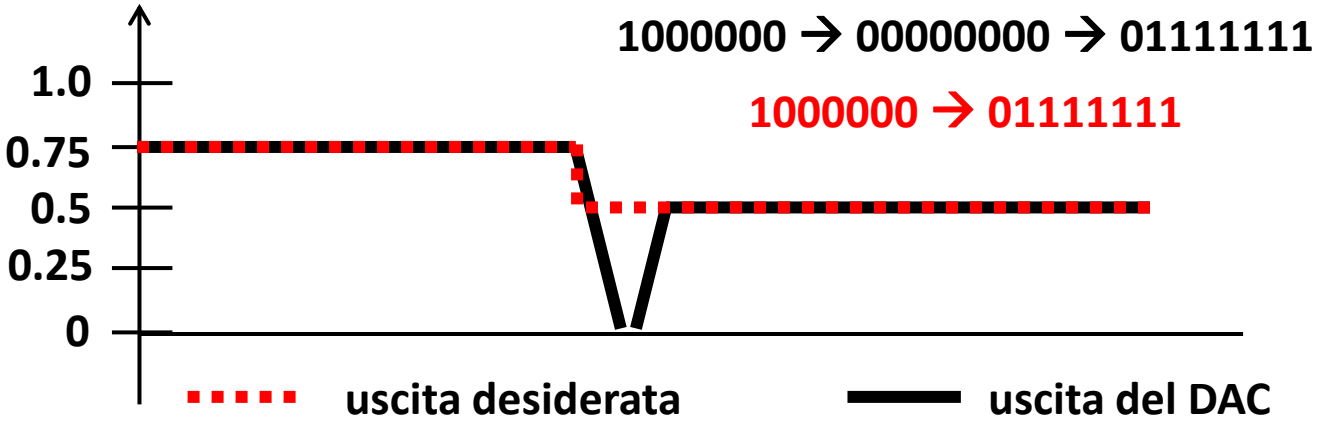
- Se uno **switch pilotato da un bit viene chiuso** (bit a 1), una **corrente pesata rispetto all'importanza del bit** (correnti maggiori per bit più significativi) viene immessa nell'ingresso invertente dell'amplificatore operazionale
- Per **DAC ad alta risoluzione** (molti bit), **non è richiesta un'ampia gamma di resistori**

Specifiche dei DAC

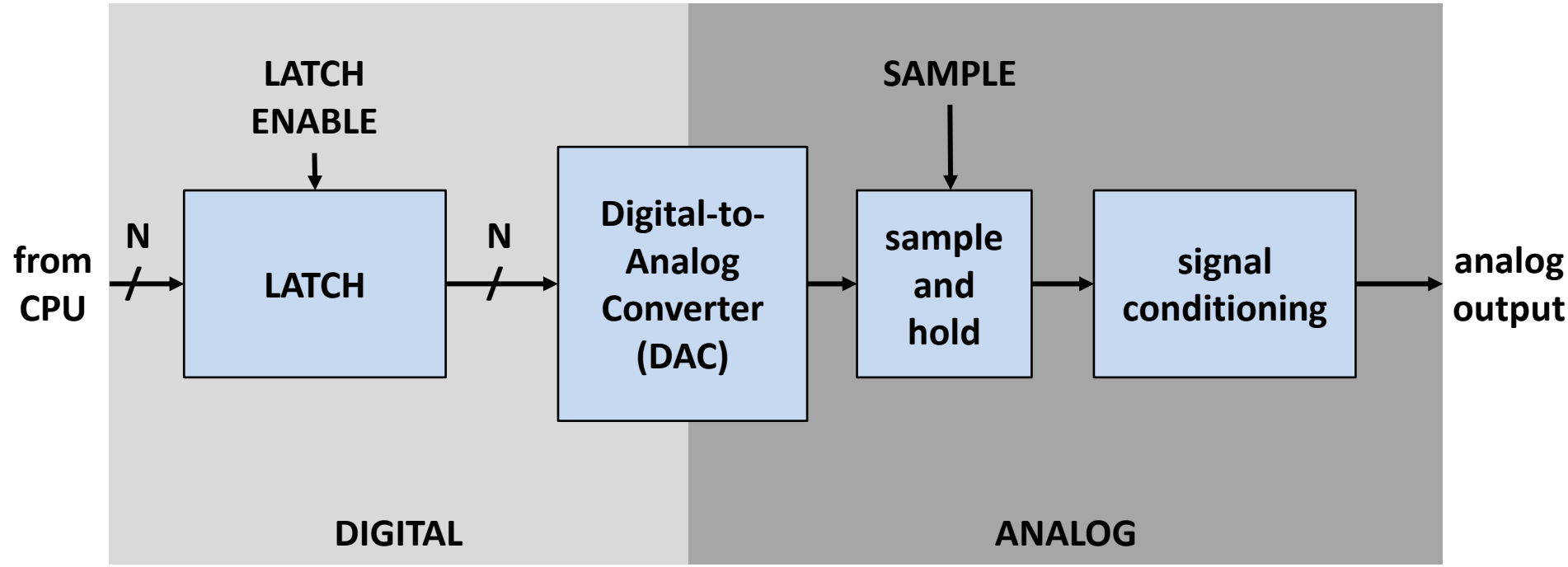
- La **risoluzione** è fissata dal numero di bit dell'input digitale ed è pari al **voltaggio** corrispondente allo **step digitale più piccolo** (1 LSB)
- La **linearità** misura **quanto l'uscita si avvicina ai valori desiderati** (una linea dritta diagonale che va dallo zero al valore massimo)
- Il **tempo di settling** è il **tempo** impiegato dal voltaggio di **uscita** a **stabilizzarsi entro** una specifica **banda di errore**, tipicamente ±1/2 LSB

Glithc nei DAC

- Un **glitch** è un'**uscita indesiderata** causata dalla **commutazione asimmetrica** degli **switch** del DAC
  - se uno **switch commuta** ad 1 a 0 **più velocemente** di un altro, allora può verificarsi un glitch
  - i glitch possono essere **eliminati** utilizzando un circuito di **sample-and-hold**



Conversione Digitale-Analogica Deglitched



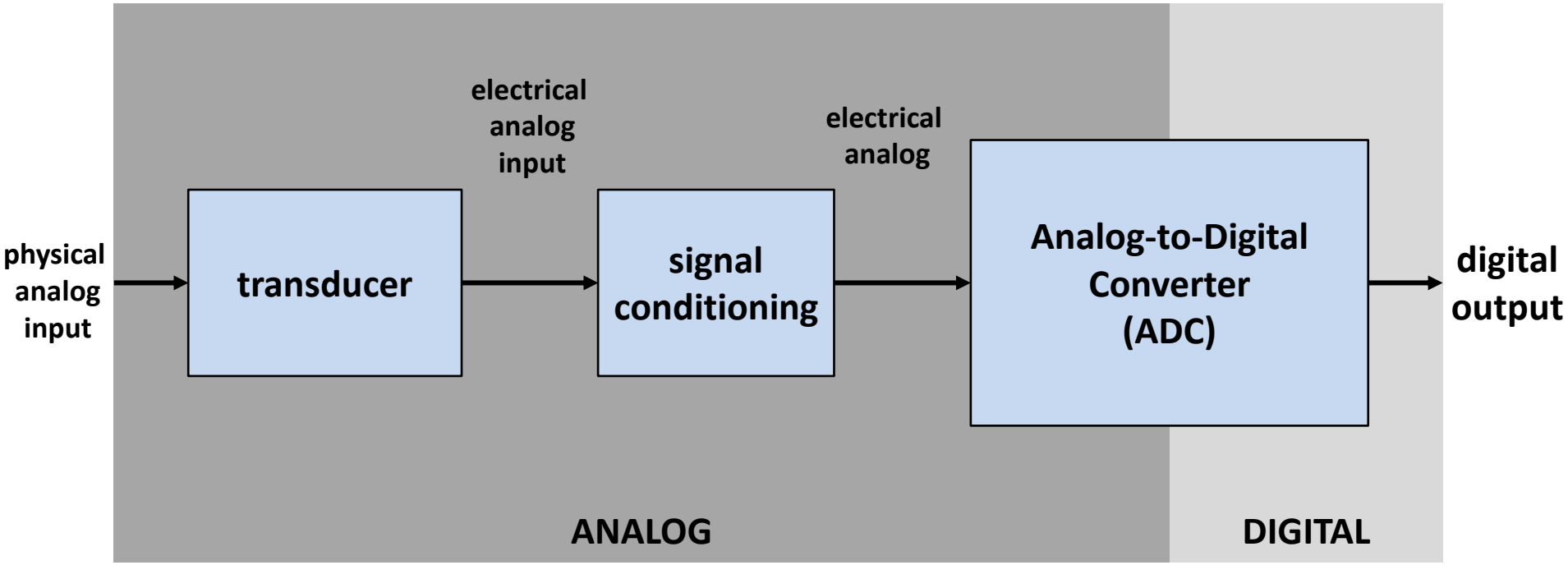
Output Analogici nell'ARMv6-M

- Negli ARM le **uscite analogiche** sono realizzate **esclusivamente** attraverso **segnali PWM** prodotti dai **timer/counter** disponibili nello specifico microcontrollore
- **Non sono presenti convertitori DA** nei microcontrollori di questa famiglia
- Tuttavia i convertitori **DA** vengono spesso **usati** per i convertitori **Analog-to-Digital (ADC)**

Indice

- Uscite Analogiche
  - Pulse Width Modulation
    - PWM negli ARM
  - Digital-to-Analog (D/A) Conversion
- Ingressi Analogici
  - Analog-to-Digital (A/D) Conversion
    - ADC negli ARM

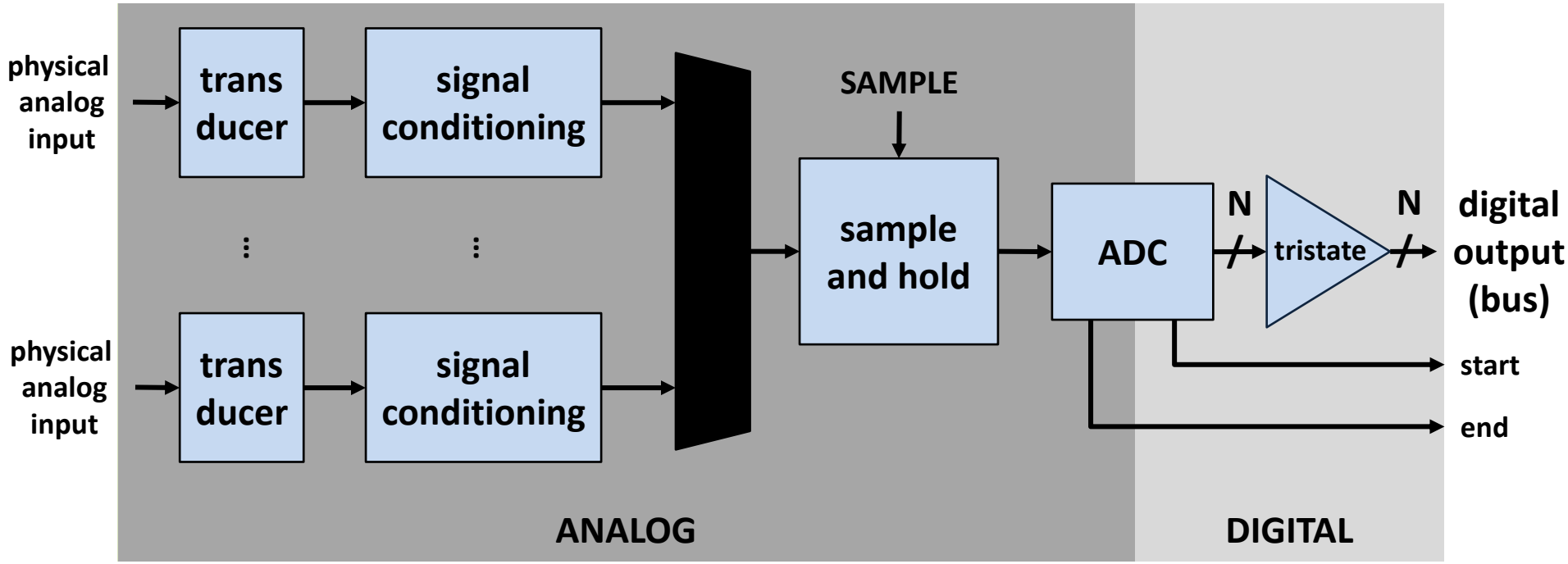
Conversione Analogico-Digitale



Conversione Analogico-Digitale

- Un **trasduttore** converte **valori fisici** in segnali **elettrici** (voltaggi o correnti)
- Il blocco **signal conditioning** ha il compito di
  - **isolare e bufferizzare:** l'input all'ADC può aver bisogno di protezione da voltaggi pericolosi
  - **amplificare:** difficilmente il trasduttore produce il voltaggio o la corrente richiesti dall'ADC, per cui è necessario uno stadio di amplificazione
  - **limitare in banda:** è necessario un filtro passa-basso (Low-Pass Filter, LPF) per limitare le frequenze in input

Conversione Analogico-Digitale



Conversione Analogico-Digitale

- In applicazioni dove vi sono **diversi input analogici** da digitalizzare, un **multiplexer** viene utilizzato per condividere l'ADC tra diverse catene input-trasduttore-signal conditioning
- Prima dell'ADC viene posto un **circuito sample-and-hold** per mantenere l'**ingresso stabile** per tutto il **tempo** necessario alla **conversione** AD
- In uscita all'ADC vi sono tipicamente dei **buffer tristate** per trasmettere le uscite digitali alla CPU

Teorema di Shannon

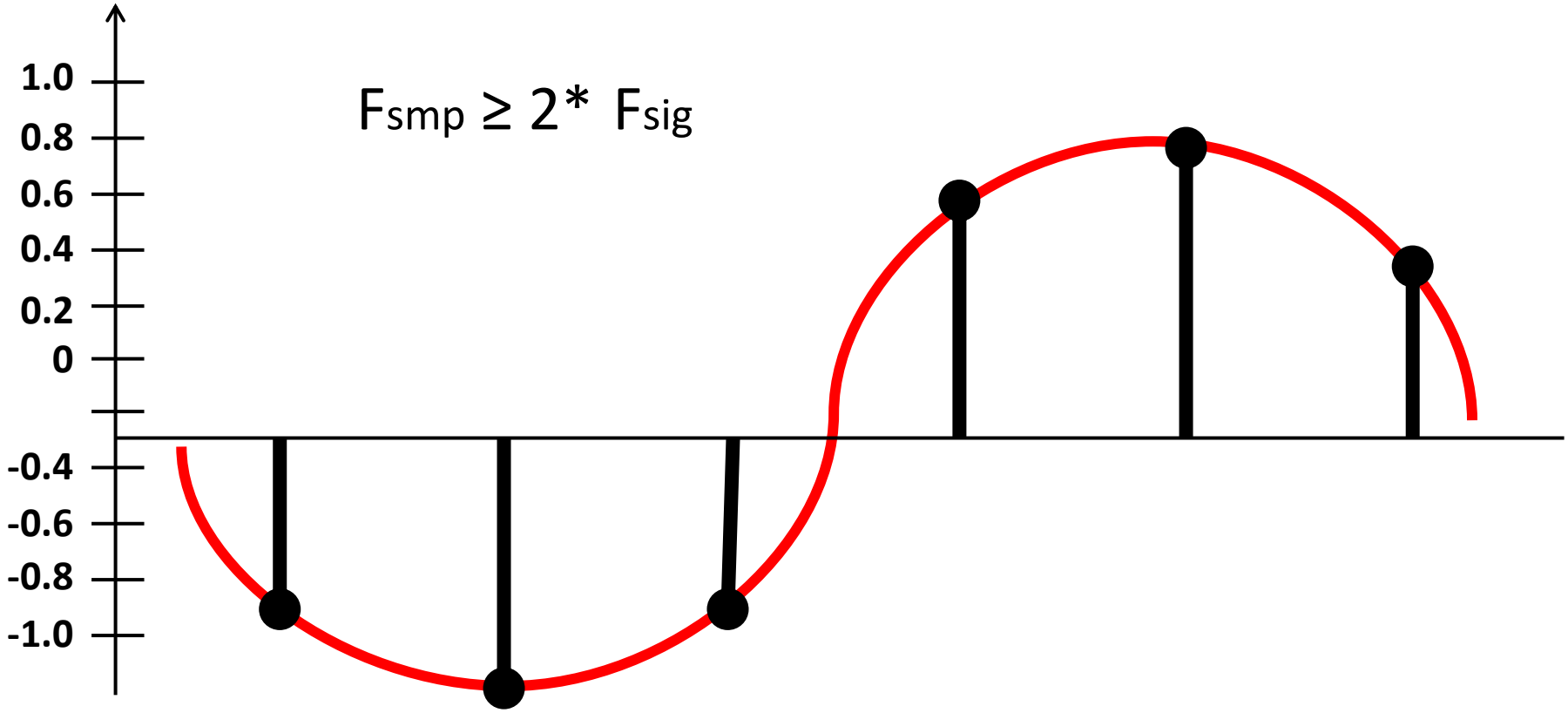
- Se un **segnale periodico** con **frequenza F_{sig}** deve essere campionato per la digitalizzazione, la **frequenza di campionamento F_{smp} minima** è pari al **doppio della frequenza del segnale**

$$f(t) = A*\sin(2\pi F_{sig} t)$$

$$F_{smp} \geq 2* F_{sig}$$

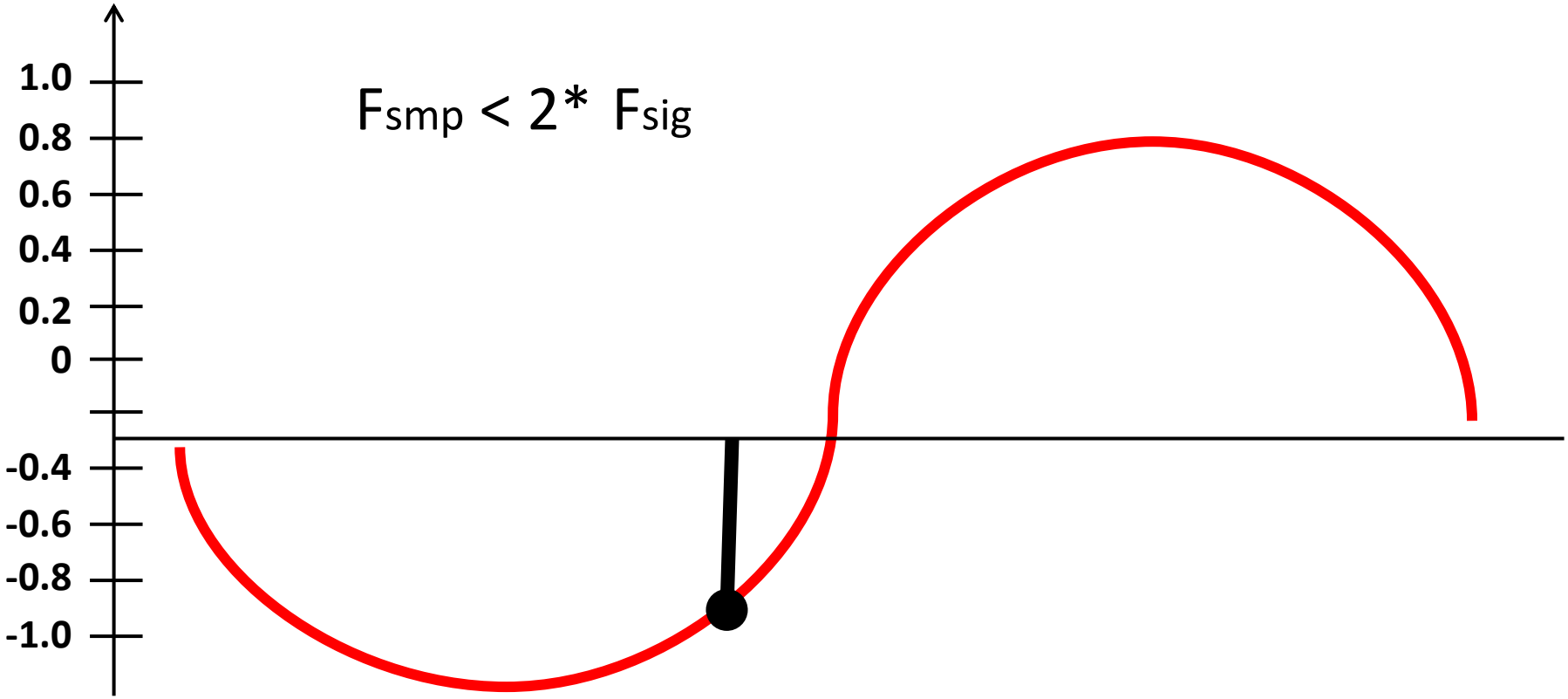
Teorema di Shannon

- Frequenza di campionamento **sufficiente**



Teorema di Shannon

- Frequenza di campionamento **non sufficiente**

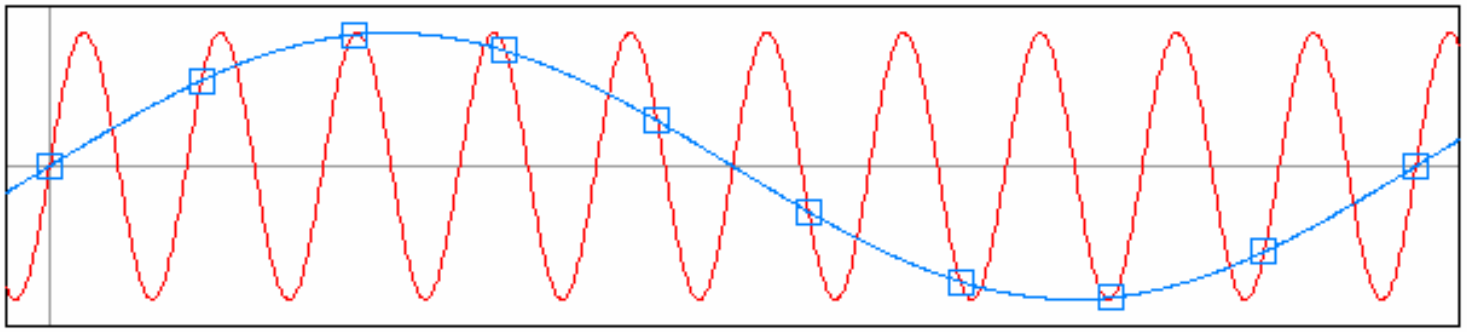


Teorema di Shannon

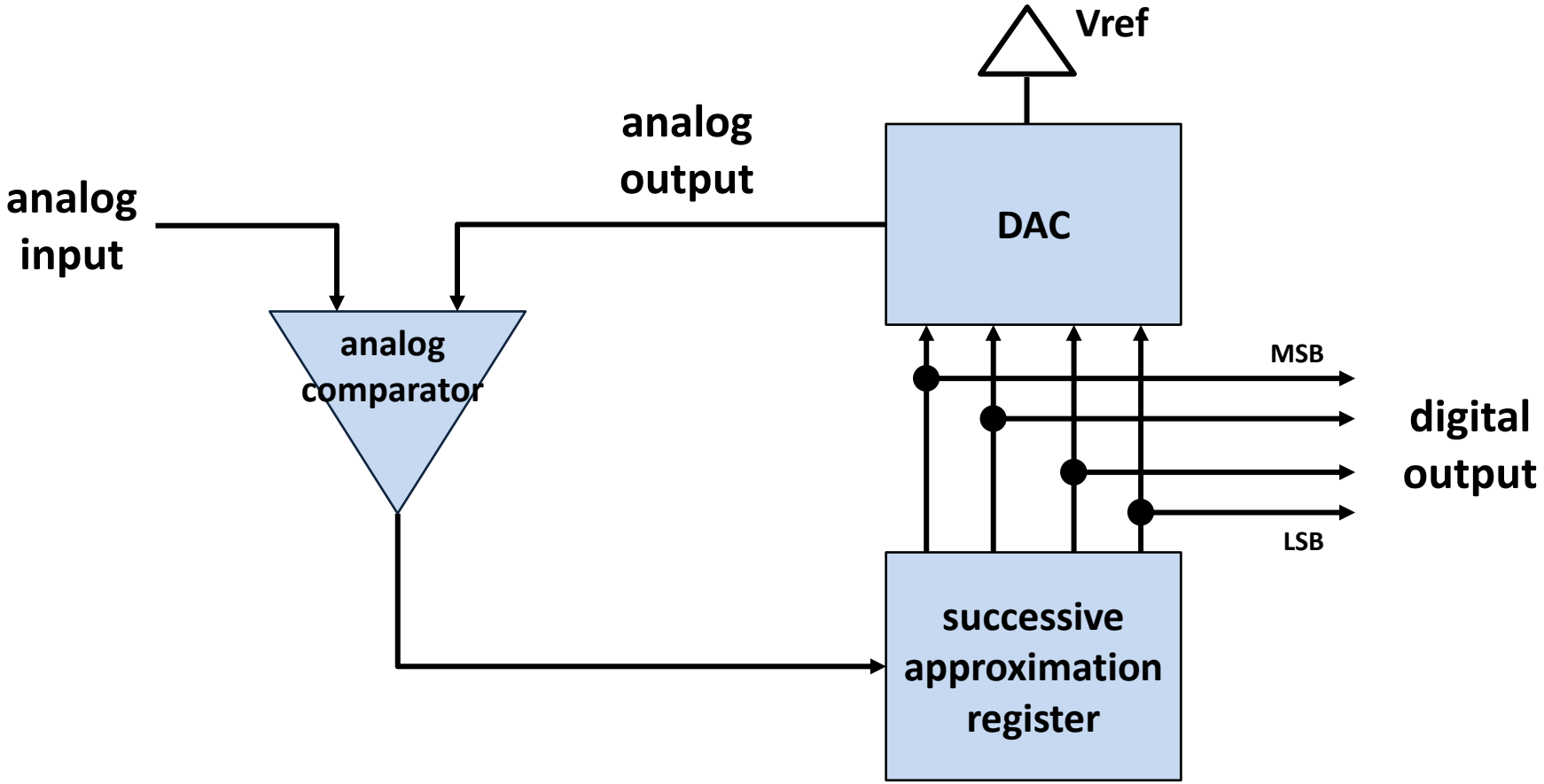
- Il **teorema di Shannon** indica qual è il **limite minimo della frequenza di campionamento** ($2*F_{sig}$ detta anche **frequenza di Nyquist**) per cui l'**informazione** del segnale è comunque **preservata**
- Un **segnale** può essere **riprodotto esattamente** se è campionato ad una **frequenza maggiore** o **uguale** alla frequenza di **Nyquist**
- Se la **frequenza di campionamento** è **minore** di quella di **Nyquist**, il segnale è **sotto-campionato**

Teorema di Shannon

- Un **segnale sotto-campionato**, quando viene riconvertito in un segnale a tempo continuo sarà affetto da un fenomeno detto **aliasing**
  - l'aliasing è la presenza di **componenti non desiderate** nel **segnale ricostruito**: tali componenti non erano presenti nel segnale originale



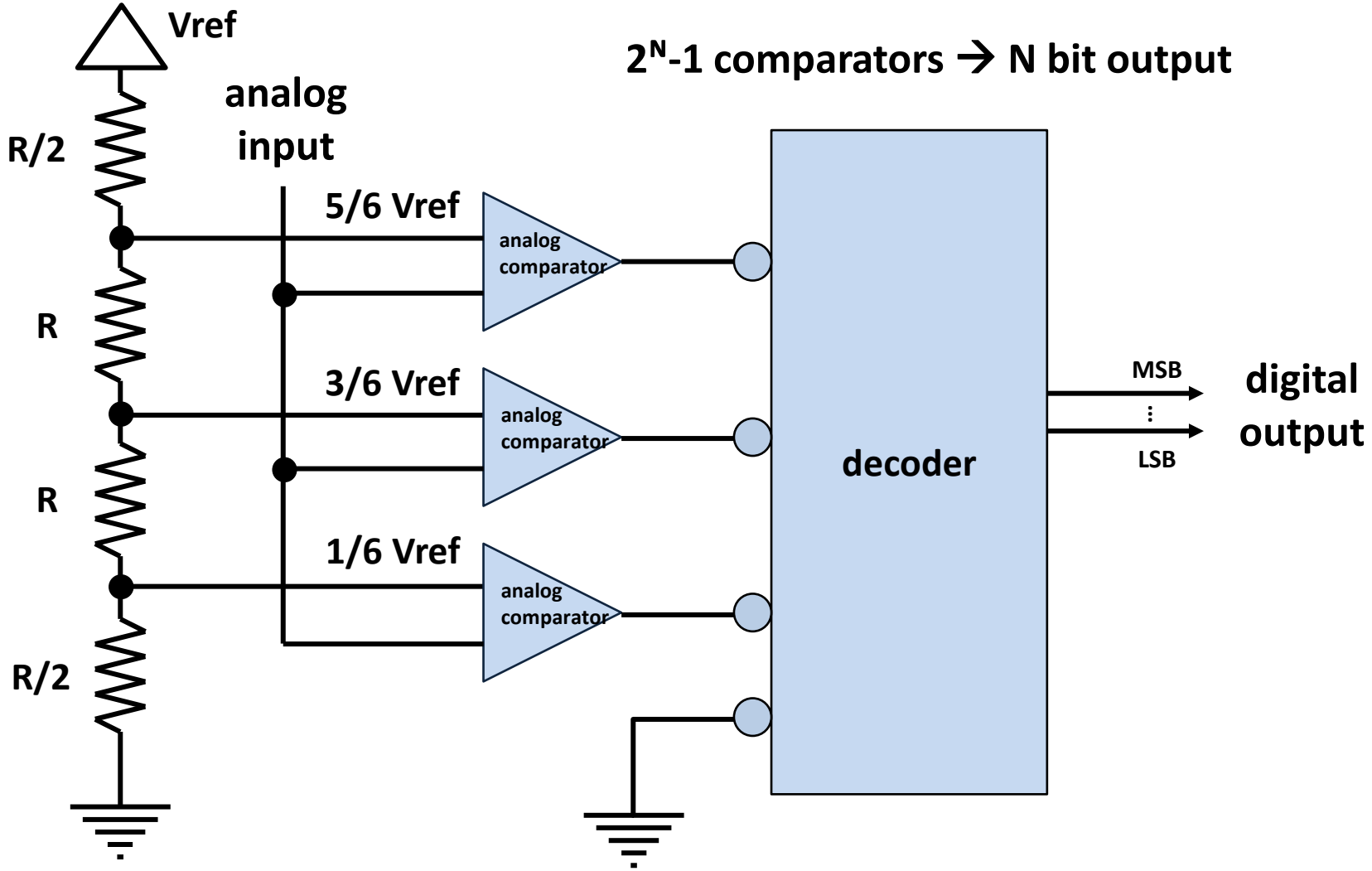
Convertitore AD ad Approssimazioni Successive



Convertitore AD ad Approssimazioni Successive

- Nel **registro ad approssimazioni successive** viene **testato ogni bit**, dal MSB al LSB, inizialmente settandolo
- Per **ogni bit** settato, l'**uscita** (analogica) del **DAC** viene **comparata** con l'**input**
- Se l'uscita del DAC è **minore** dell'input, allora il **bit testato rimane settato**, altrimenti viene resettato
- **N cicli di comparazione** sono necessari per testare tutti i bit del registro ad approssimazioni successive

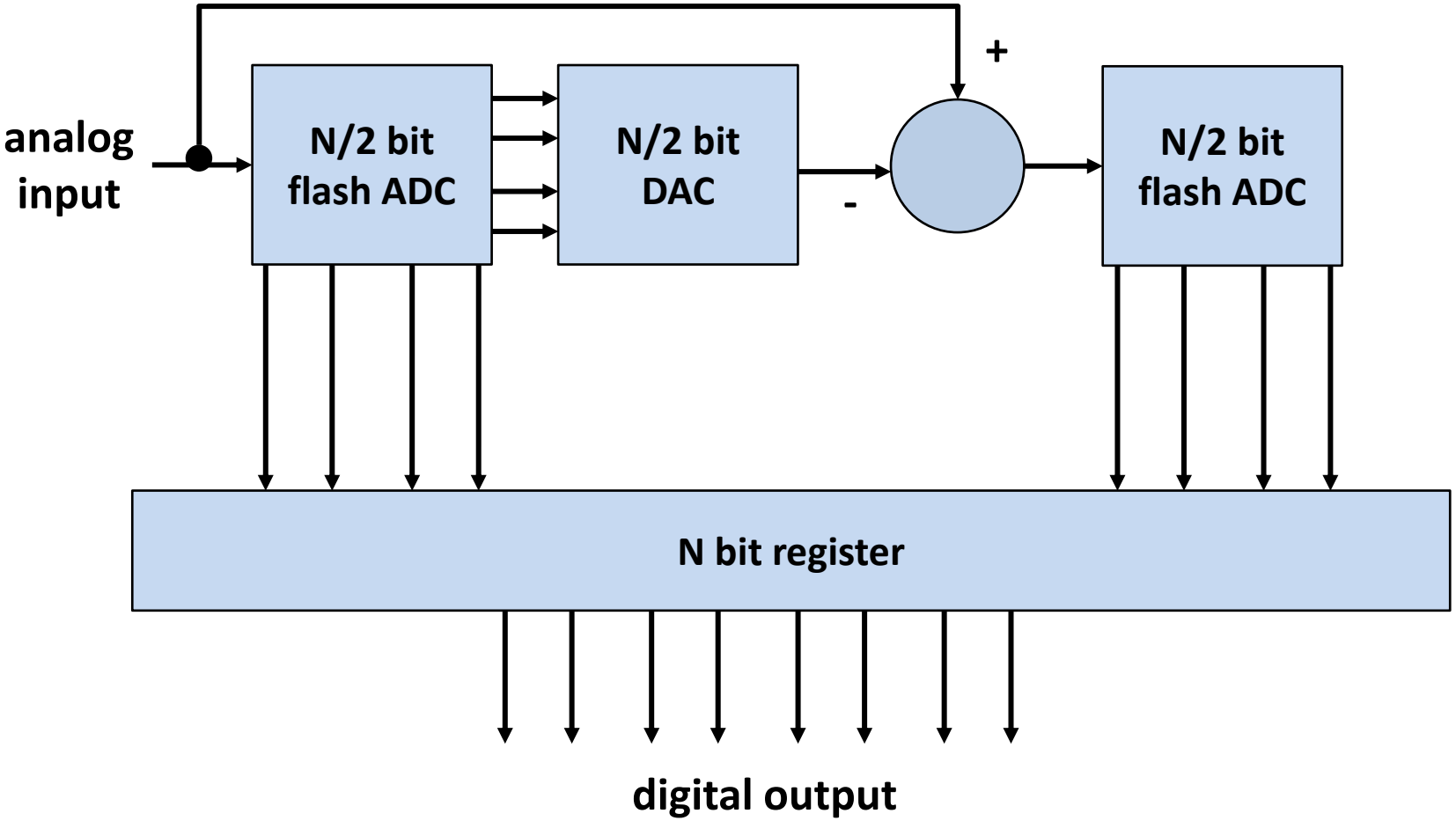
Convertitore AD Parallelo



Convertitore AD Parallelo

- È necessario un insieme di **2^N-1 comparatori** per avere un segnale digitale a **N bit in uscita**
- Il **tempo di conversione** è pari al tempo di propagazione del comparatore analogico e del decoder
- Soluzione molto **veloce** (per questo chiamata anche flash ADC) ma **costosa** rispetto ad altre in **termini di risorse**

Convertitore AD Parallelo a 2 Stadi



Convertitore AD Parallelo a 2 Stadi

- Il segnale analogico in ingresso viene **convertito** in **due parti**
  - dapprima viene fatta una **stima grossolana** tramite il **primo flash ADC**; la sua **uscita** viene mandata al DAC e poi **sottratta all'input analogico**, ottenendone l'errore
  - tale **errore** viene convertito dal **secondo flash ADC** il cui risultato, **combinato** con quello del primo flash ADC, costituisce il **valore digitalizzato**
- Ha **prestazioni simili ad un flash ADC** ma **senza** i **2^N-1 comparatori** (es. N=8 → 30 invece che 255)

Specifiche degli ADC

- Il **tempo di conversione** T_{conv} è il tempo richiesto per completare la conversione del segnale in input e stabilisce il **limite massimo alla frequenza** dello stesso **segnale in ingresso**

$$F_{sig} \leq 1/(2*T_{conv})$$

- La **risoluzione** corrisponde al minor segnale analogico in ingresso per cui l'ADC produce un'uscita digitale, ed è **fissata dal numero di bit** del convertitore

$$resolution = Vref/2^N$$

Specifiche degli ADC

- L'**errore di quantizzazione** deriva dall'**arrotondamento** del segnale analogico con dei valori discreti di tensione (è sempre **≤ 1 LSB**)
- La **linearità** quantifica il **discostamento** dei valori di uscita dal valore **ideale** (diagonale da 0 o -Vref a Vref). La **migliore linearità** che si può ottenere è pari a **±1/2 del LSB**
- Il **tempo di apertura** è il tempo per cui l'ADC "osserva" il segnale analogico in ingresso ed è tipicamente pari al **tempo di conversione**

Indice

- Uscite Analogiche
  - Pulse Width Modulation
    - PWM negli ARM
  - Digital-to-Analog (D/A) Conversion
- Ingressi Analogici
  - Analog-to-Digital (A/D) Conversion
    - ADC negli ARM

ADC nell'ARMv6-M

- L'ARMv6-M è dotato di un **ADC ad approssimazioni successive** con:
  - fino a **12 bit** di risoluzione
  - **18 canali** di input multiplexati (16 sorgenti esterne, 2 interne)
  - conversione con diverse modalità (single, continuous, discontinuous)
  - risultato della conversione memorizzato con **allineamento** a **destra** o a **sinistra** in un registro apposito
  - **funzionalità avanzate** (watchdog, power saving)

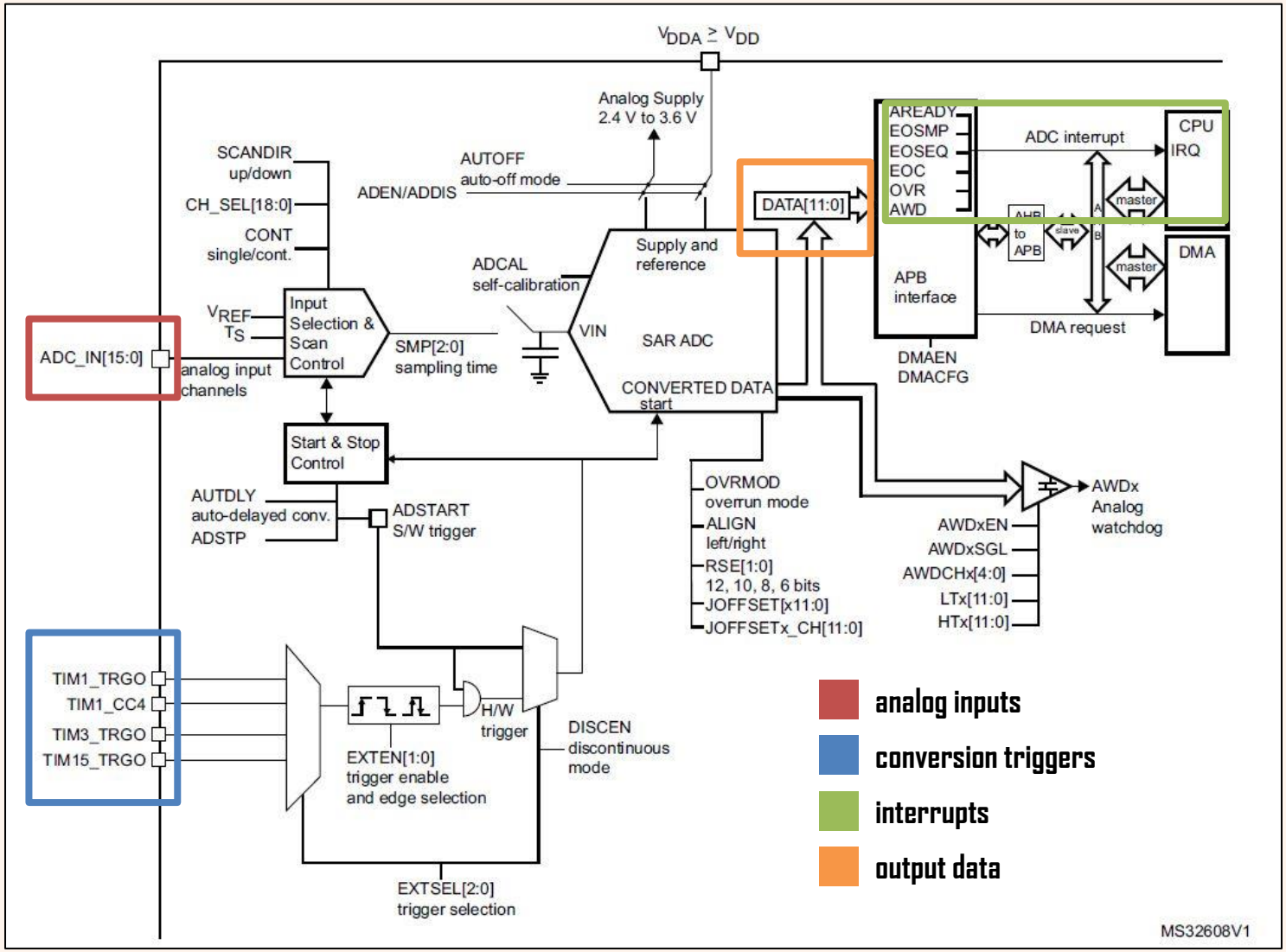
ADC nell'ARMv6-M

- Caratteristiche dell'**ADC** dell'ARMv6-M:
  - **tensioni** permesse
    - alimentazione: da 2.4 V a 3.6 V
    - input: $V_{SSA} \leq V_{IN} \leq V_{DDA}$
  - **risoluzione configurabile** (12, 10, 8 o 6 bit)
  - **tempo di conversione**
    - 12 bit @1MHz → 1 us,
    - 10 bit @1MHz → 0.93 us
  - **canali di input** analogici
    - 16 esterni
    - 2 interni(sensore di temperatura V_{SENS}, riferimento di tensione V_{REF})

ADC nell'ARMv6-M

- Caratteristiche dell'**ADC** dell'ARMv6-M:
  - **inizio conversione**
    - via software
    - via hardware (trigger con polarità configurabile tramite eventi del TIM1, TIM3 e TIM15)
  - **modalità di conversione** (canale singolo o scansione di diversi canali)
    - **single**: una conversione di un'intera sequenza di input in corrispondenza di un evento di trigger
    - **continuous**: conversione continua di un'intera sequenza di input
    - **discontinuous**: un trigger per ogni conversione nella sequenza di input
  - **generazione di interrupt** in base a diversi eventi

ADC negli ARM



Calibrazione dell'ADC nell'ARMv6-M

- L'ADC è in grado di **calibrarsi** ad ogni accensione
- La **calibrazione** dovrebbe essere effettuata **dopo** l'**accensione** e **prima di una conversione** per eliminare l'errore di offset derivante da chip e condizioni operative dell'ADC (soprattutto la tensione di alimentazione V_{DDA})
  - l'**ADC** deve essere **disabilitato** per la calibrazione (**ADEN**=0 nel registro ADC_CR)
  - viene inizializzata via software ponendo il bit **ADCAL** a 1 nel registro ADC_CR
  - il bit ADCAL rimane alto durante tutta la calibrazione e viene resettato via hardware al termine

Abilitazione dell'ADC nell'ARMv6-M

- L'ADC è **disabilitato di default** e posto in modalità di power down (ADEN=0 nel registro ADC_CR)
- All'accensione, l'ADC necessita di un **tempo di stabilizzazione** per **iniziare a convertire** in modo corretto
  - l'ADC si può abilitare ponendo il bit **ADEN** a 1
  - il bit **ADRDY** nel registro ADC_ISR sarà posto a 1 non appena l'ADC è pronto a convertire (tale evento può generare un **interrupt** se il bit ADRDYIE è alto nel registro ADC_IER)
  - l'ADC si può disabilitare ponendo il bit **ADDIS** a 1

Abilitazione dell'ADC nell'ARMv6-M

- **Abilitazione**
  - resettare il bit **ADRDY** nel registro ADC_ISR scrivendoci un 1
  - settare il bit **ADEN** nel registro ADC_CR
  - attendere finché il bit **ADRDY** non va alto nel registro ADC_ISR
- **Disabilitazione**
  - assicurarsi che **ADSTART** sia **0** (ADC fermo) nel registro ADC_CR
  - settare il bit **ADDIS** nel registro ADC_CR
  - attendere finché il bit **ADEN** non va basso per verificare l'operazione
  - resettare il bit **ADRDY** nel registro ADC_ISR scrivendoci un 1

Clock dell'ADC nell'ARMv6-M

- È possibile pilotare l'ADC con **due diversi clock** selezionabili tramite i bit CLKMODE[1:0] del registro ADC_CFGR2
  - **ADC asynchronous clock** (CLKMODE=2'b00): clock dedicato **indipendente** da quello del bus **APB** con cui si può raggiungere la **massima frequenza** di clock ma con **latenza** tra **trigger** e **inizio** conversione **non deterministica**
  - **PCLK** (APB peripheral clock /2 se CLKMODE=2'b01 o /4 se CLKMODE=2'b10): clock derivato da quello del bus APB, dunque già **sincronizzato** con quest'ultimo ed in grado dunque di garantire **latenze deterministiche** tra trigger e inizio conversione (2.75 cicli se /2, 2.625 cicli se /4)

Tempo di Campionamento nell'ADC dell'ARMv6-M

- Prima di iniziare la **conversione** occorre conoscere il **tempo** necessario ad effettuarla
- il **tempo di conversione** dipende dalla resistenza di input e dal voltaggio in ingresso, oggetto della conversione stessa
  - l'ADC **campiona** il **segnale** di ingresso per un numero di periodi di clock specificato dai bit **SMP[2:0]** nel registro ADC_SMPR (comune a tutti i canali)
  - il tempo di conversione si può calcolare come (12 bit risoluzione)

$$t_{CONV} = (SMP + 12.5) * \text{ADC clock period}$$

  - l'ADC indica la fine del campionamento con il flag **EOSMP**

Configurazione dell'ADC nell'ARMv6-M

- Regole di **configurazione**
  - i bit **ADCAL** e **ADEN** possono essere scritti solo se l'ADC è disabilitato (ADEN=0)
  - i bit **ADSTART** e **ADDIS** possono essere scritti solo se l'ADC è abilitato e non ci sono richieste di disabilitazione pendenti (ADEN=1, ADDIS=0)
  - tutti gli altri bit di controllo nei registri **ADC_IER**, **ADC_CFGRi**, **ADC_SMPR**, **ADC_TR**, **ADC_CHSELR** e **ADC_CCR**, possono essere scritti solo se l'ADC è abilitato (ADEN=1) e non c'è una conversione in corso (ADSTART=0)

Configurazione dell'ADC nell'ARMv6-M

- Regole di **configurazione**
  - il bit **ADSTP** può essere scritto solo se l'ADC è abilitato (ADEN=1), possibilmente in conversione (ADSTART=1) e se non vi è una richiesta di disabilitazione pendente (ADDIS=0)
- **Non c'è una protezione hardware** per le operazioni di **scrittura proibite**, ma se tali scritture accadono l'ADC potrebbe entrare in uno **stato indefinito** per cui occorre **resettarlo** ponendo a 0 ADEN e anche tutti i bit del registro ADC_CR

Canali dell'ADC nell'ARMv6-M

- Nell'ADC dell'ARMv6-M ci sono **18 canali**
  - 16 sono connessi alle porte di **GPIO** (ADC_IN0,…., ADC_IN15)
  - 2 sono connessi a pin **interni**: un **sensore di temperatura** (ADC_IN16) e una **tensione di riferimento** (ADC_IN17)
- È possibile convertire **un canale** o **scandirne** più di uno in **sequenza**
  - l'inclusione del canale x nella sequenza da convertire viene specificata dal bit di selezione **CHSELx** nel registro ADC_CHSEL
  - l'ordine con cui sono scanditi i canali è determinato dal bit **SCANDIR** (1'b0 → da 0 a 17, 1'b1 → da 17 a 0) nel registro ADC_CFGR1

Single Conversion Mode

- L'ADC esegue una **singola sequenza di conversione** sui canali
  - configurato ponendo il bit **CONT** a 0 nel registro ADC_CFGR1
  - la conversione inizia settando **ADSTART** nel registro ADC_CR o tramite un evento di trigger hardware
  - dopo il **termine** di **ogni conversione** nella sequenza
    - i dati convertiti sono posti nel registro a 16 bit **ADC_DR**
    - il flag di fine conversione **EOC** viene settato nel registro ADC_ISR
    - se il bit **EOCIE** nel registro ADC_IER è settato, viene generato un interrupt

Single Conversion Mode

- L'ADC esegue una **singola sequenza di conversione** sui canali
  - dopo il **termine** di tutta la **sequenza** di conversione
    - il flag di fine sequenza **EOSEQ** viene settato nel registro ADC_ISR
    - se il bit **EOSEQIE** nel registro ADC_IER è settato, viene generato un interrupt
  - la **lunghezza della sequenza** definisce il **numero di canali** da convertire nella sequenza stessa
    - per avere una **singola conversione** occorre configurare la **lunghezza** della **sequenza** a **1**

Continuous Conversion Mode

- L'ADC **esegue ciclicamente**, una dopo l'altra, **diverse sequenze** di conversione sui canali
  - configurato ponendo il bit **CONT** a 1 nel registro ADC_CFGR1
  - la conversione inizia settando **ADSTART** nel registro ADC_CR o tramite un evento di trigger hardware
  - dopo il **termine** di **ogni conversione** nella sequenza
    - i dati convertiti sono posti nel registro a 16 bit ADC_DR
    - il flag di fine conversione **EOC** viene settato nel registro ADC_ISR
    - se il bit **EOCIE** nel registro ADC_IER è settato, viene generato un interrupt

Continuous Conversion Mode

- L'ADC **esegue ciclicamente**, una dopo l'altra, **diverse sequenze** di conversione sui canali
  - dopo il **termine** di tutta la **sequenza** di conversione
    - il flag di fine sequenza **EOSEQ** viene settato nel registro ADC_ISR
    - se il bit **EOSEQIE** nel registro ADC_IER è settato, viene generato un interrupt
  - dopodiché **inizia immediatamente una nuova sequenza di conversione**
  - la **lunghezza della sequenza** definisce il numero di canali da convertire nella sequenza stessa
    - per avere una **singola conversione per sequenza** occorre configurare la **lunghezza** della **sequenza** a **1**

Discontinuous Conversion Mode

- L'ADC richiede un **trigger software o hardware** (configurabile) **per ogni conversione** nella sequenza
  - configurato ponendo il bit **DISCEN** a 1 nel registro ADC_CFGR1
  - ogni conversione della sequenza inizia settando **ADSTART** nel registro ADC_CR o tramite un evento di trigger hardware
  - non è possibile avere continuous (CONT=1) e discontinuous (DISCEN=1) mode assieme
  - la **lunghezza della sequenza** definisce il numero di canali da convertire nella sequenza stessa
    - per avere una **singola conversione** occorre configurare la **lunghezza** della **sequenza** a **1**

Gestione della Conversione nell'ADC dell'ARMv6-M

- La conversione viene **inizializzata** settando il bit **ADSTART** (EXTEN=2'b00 nel registro ADC_CFGR1) o in occasione di un evento di **trigger hardware** (EXTEN!=2'b00 nel registro ADC_CFGR1)
- **ADSTART** rappresenta anche lo **stato di attività** dell'**ADC**
  - quando ADSTART è **basso**, allora è possibile **configurare** l'ADC per una **nuova conversione**
  - ADSTART è **resettato** via hardware
    - al **termine** di una **sequenza** di conversione in single mode (**DISCEN=0**)
    - al **termine** di una **conversione** in discontinuous mode (**DISCEN=1**)

Gestione della Conversione nell'ADC dell'ARMv6-M

- La **risoluzione** dell'ADC può essere selezionata attraverso i bit **RES[1:0]** del registro ADC_CFGR1 (RES è modificabile solo quando ADEN è basso)
  - risoluzioni minori di quella massima (12 bit) permettono **tempi di conversione** minori

RES	Bits	t_{SAR} [ADC clock periods]	t_{SAR} @14MHz [ns]	t_{SMPL} [ADC clock periods]	t_{CONV} [ADC clock periods]	t_{CONV} @14MHz [ns]
00	12	12.5	893	1.5	15	1000
01	10	11.5	821	1.5	13	928
10	8	9.5	678	1.5	11	785
11	6	7.5	535	1.5	9	643

$$t_{CONV} = t_{SMPL} + t_{SAR} = (SMP+ RES) * \text{ADC clock period}$$

Gestione dei Dati nell'ADC dell'ARMv6-M

- Alla **fine** di ogni **conversione** (quando vi è un evento EOC) il **risultato** viene memorizzato nel **registro** a 16 bit **ADC_DR**
  - il **formato dei dati** nel registro ADC_DR dipende dal **bit di allineamento** (ALIGN nel registro ADC_CFG1) e dalla **risoluzione** (RES nel registro ADC_CFG1)

ALIGN	RES	ADC_DR content															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	00	0	0	0	0	DR[11:0]											
0	01	0	0	0	0	0	0	DR[9:0]									
0	10	0	0	0	0	0	0	0	0	DR[7:0]							
0	11	0	0	0	0	0	0	0	0	0	0	DR[5:0]					
1	00	DR[11:0]												0	0	0	0
1	01	DR[9:0]										0	0	0	0	0	0
1	10	DR[7:0]								0	0	0	0	0	0	0	0
1	11	DR[5:0]						0	0	0	0	0	0	0	0	0	0

Interrupt nell'ADC dell'ARMv6-M

- L'ADC permette la **generazione di interrupt** in corrispondenza di **diversi eventi**
  - quando è **pronto dopo l'abilitazione**
  - quando **termina una conversione** singola
  - quando **termina una sequenza** di conversione
  - quando il **watchdog analogico** rileva un **errore**
  - quando **termina** la fase di **campionamento**
  - quando va in **overrun**, ovvero un dato convertito non viene letto in tempo prima di essere sovrascritto da una nuova conversione

Interrupt nell'ADC dell'ARMv6-M

interrupt event	event flag	enable control bit
ADC ready to convert	ADRDY	ADRDYIE
end of one single conversion	EOC	EOCIE
end of the whole conversion sequence	EOSEQ	EOSEQIE
analog watchdog status bit set	AWD	AWDIE
end of sampling phase	EOSMP	EOSMPIE
overrun	OVR	OVRIE

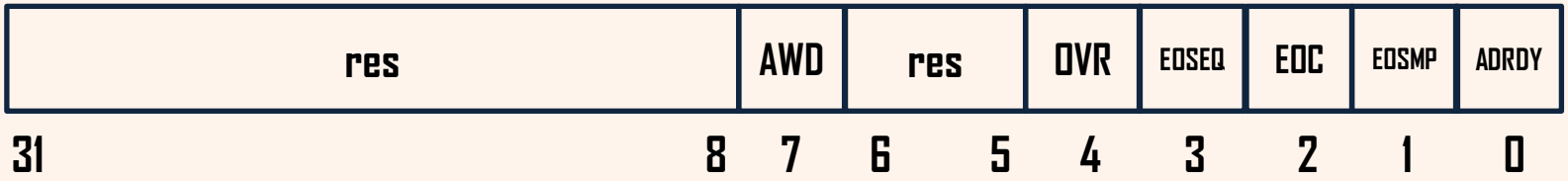
Configurazione dell'ADC

- **APB Peripheral Clock Enable Register 2 (RCC_APB2ENR)**: registro che abilita il clock delle periferiche del bus APB nell'ARMv6-M

  - **ADCEN, ADC Interface Clock Enable** se settato abilita il clock per l'interfaccia (registri di configurazione) dell'ADC

Configurazione dell'ADC

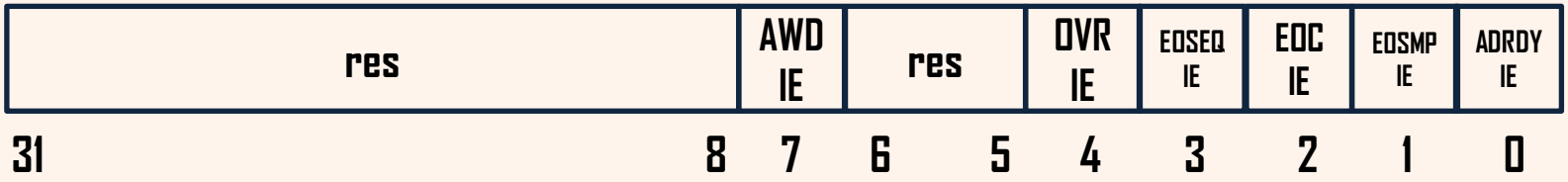
- **ADC Interrupt and Status Register (ADC_ISR)**: registro di interrupt e di stato dell'ADC



  - **ADRDY** ADC ready
  - **EOSMP** end of sampling flag
  - **EOC** end of conversion flag
  - **EOSEQ** end of sequence flag
  - **OVR** ADC overrun
  - **AWD** analog watchdog flag

Configurazione dell'ADC

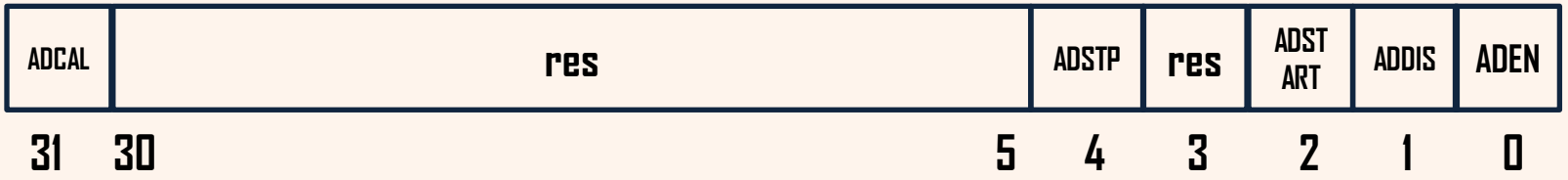
- **ADC Interrupt Enable Register (ADC_IER)**: registro di abilitazione degli interrupt dell'ADC



  - **ADRDYIE** ADC interrupt enable
  - **EOSMPIE** end of sampling interrupt enable
  - **EOCIE** end of conversion interrupt enable
  - **EOSEQIE** end of sequence interrupt enable
  - **OVRIE** ADC overrun interrupt enable
  - **AWDIE** analog watchdog interrupt enable

Configurazione dell'ADC

- **ADC Control Register (ADC_CR)**: registro di controllo dell'ADC



  - **ADEN** ADC enable command (abilita l'ADC, resettato via hardware)
  - **ADDIS** ADC disable command (disabilita l'ADC, resettato via hardware)
  - **ADSTART** ADC start conversion command (inizia una nuova conversione via software, resettato via hardware)
  - **ADSTP** ADC stop conversion command (interrompe ed elimina la conversione corrente via software, resettato via hardware)
  - **ADCAL** ADC calibration (inizializza la calibrazione e sta a 1 fino alla sua fine)

Configurazione dell'ADC

- **ADC Configuration Register 1 (ADC_CFGR1)**: registro di configurazione 1 dell'ADC

  - **RES** Data resolution (risoluzione della conversione: 2'b00=12 bit, 2'b01=10bit, 2'b10=8bit, 2'b11=6bit)
  - **ALIGN** Data alignment (dato in uscita allineato a destra se 0, a sinistra se 1)
  - **EXTSEL** External trigger selection (seleziona l'evento di trigger che dà il via alla conversione tra 8 possibili alternative TRG0:7)

Configurazione dell'ADC

- **ADC Configuration Register 1 (ADC_CFGR1)**: registro di configurazione 1 dell'ADC

  - **EXTEN** External trigger enable and polarity selection (abilita e seleziona la tipologia di trigger per la conversione: 00=software trigger, 01=hardware rising edge, 10=hardware falling edge, 11=hardware rising and falling edge)
  - **OVRMOD** Overrun management mode (determina l'aggiornamento del registro ADC_DR in overrun: se 0 tiene il vecchio dato, se 1 lo sovrascrive)

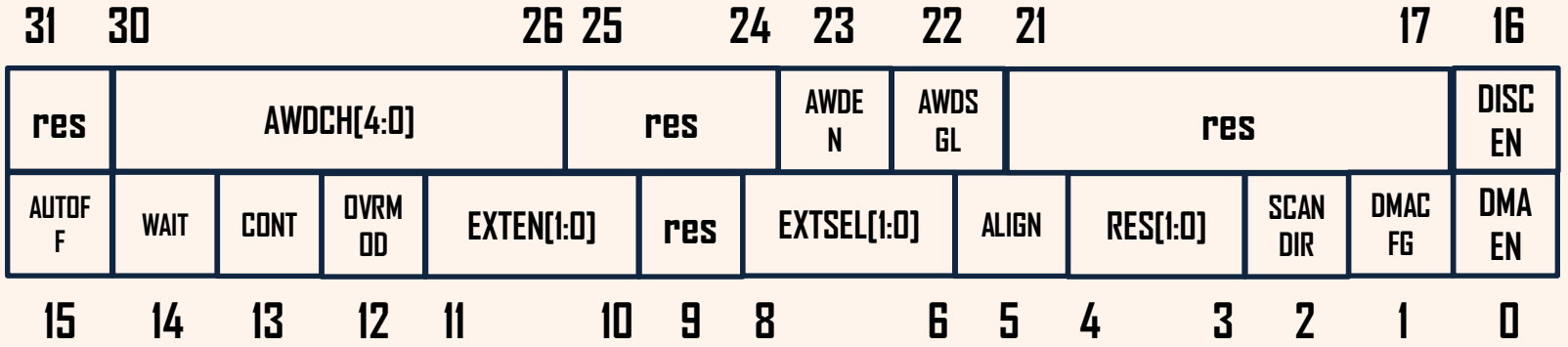
Configurazione dell'ADC

- **ADC Configuration Register 1 (ADC_CFGR1)**: registro di configurazione 1 dell'ADC

  - **CONT** Single/continuous conversion mode (definisce la modalità di conversione: single se 0, continuous se 1)
  - **WAIT** Wait conversion mode (abilita/disabilita la wait conversion mode)
  - **AUTOFF** Auto-off mode (abilita/disabilita la auto-off mode)
  - **DISCEN** Discontinuous mode (abilita o disabilita la discontinuous mode)

Configurazione dell'ADC

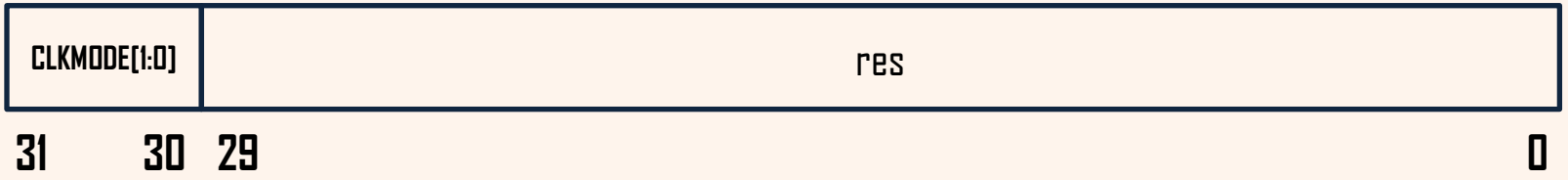
- **ADC Configuration Register 1 (ADC_CFGR1)**: registro di configurazione 1 dell'ADC



- **AWDSGL** Watchdog single/all channels (abilita il watchdog su singolo canale, se a 1, o su tutti i canali, se a 0)
- **AWDEN** Watchdog enable (abilita/disabilita il watchdog)
- **AWDCH** Watchdog channel selection (identifica il canale da monitorare con il watchdog: il valore del segnale specifica l'indice del canale)

Configurazione dell'ADC

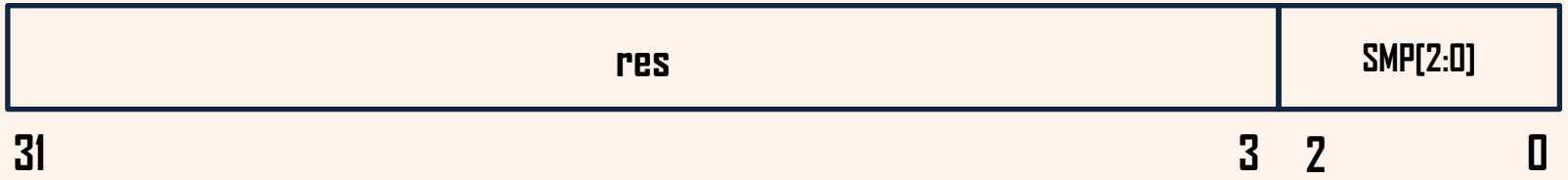
- **ADC Configuration Register 2 (ADC_CFGR2)**: registro di configurazione 1 dell'ADC



  - **CKMODE** ADC clock mode (definisce il clock dell'ADC)
    - 2'b00: ADCCLK (asynchronous clock mode)
    - 2'b01: PCLK/2 (synchronous clock mode)
    - 2'b10: PCLK/4 (synchronous clock mode)
    - 2'b11: reserved

Configurazione dell'ADC

- **ADC Sampling Time Register (ADC_SMPR)**: registro di configurazione del tempo di campionamento dell'ADC



SMP	ADC clock cycles
000	1.5
001	7.5
010	13.5
011	28.5
100	41.5
101	55.5
110	71.5
111	239.5

Configurazione dell'ADC

- **ADC Watchdog Threshold Register (ADC_TR)**: registro di configurazione delle soglie del watchdog dell'ADC

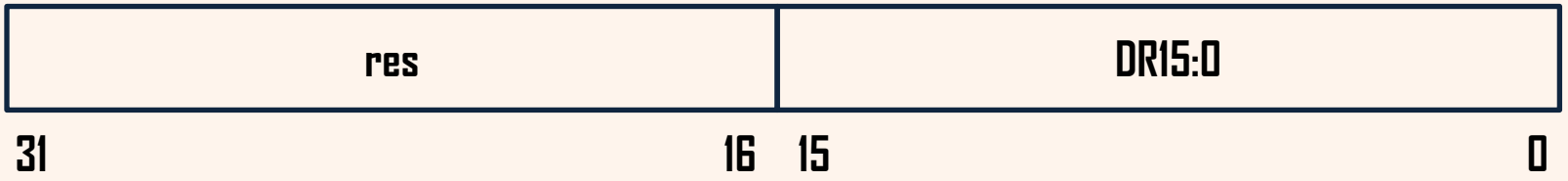
  - **LT** Watchdog lower threshold (definisce la soglia minima di conteggio per il whatchdog dell'ADC sotto la quale vi è un'anomalia)
  - **HT** Watchdog higher threshold (definisce la soglia massima di conteggio per il whatchdog dell'ADC sopra la quale vi è un'anomalia)

Configurazione dell'ADC

- **ADC Channel Selection Register (ADC_CHSELR)**: registro di selezione dei canali dell'ADC

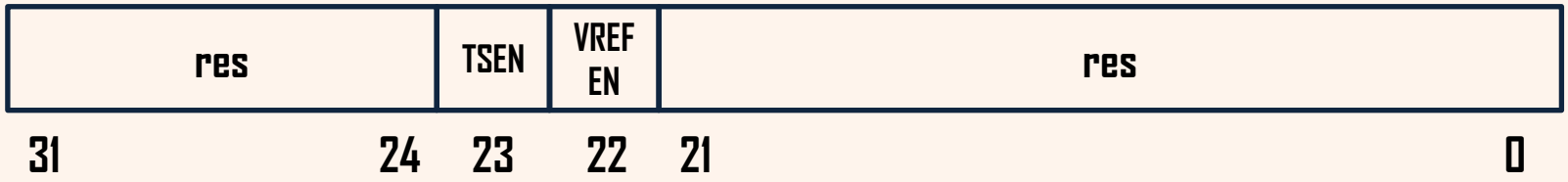
  - **CHSELx** Channel-x selection (se a 1 indica che il canale x verrà selezionato per la conversione, se a 0 il canale x verrà invece ignorato)

- **ADC Data Register (ADC_DR)**: registro contenente il dato convertito dall'ADC



Configurazione dell'ADC

- **ADC Common Configuration Register (ADC_CR)**: registro di configurazione comune dell'ADC



  - **VREFEN** V_{REFINT} enable (abilita/disabilita la tensione di riferimento interna dell'ADC)
  - **TSEN** temperature sensor enable (abilita/disabilita il sensore di temperature dell'ADC)

Configurazione dell'ADC

bus	boundary address	size	peripheral
Cortex-M0 Internal Peripherals	0xE000E100 – 0xE000E4FF	1 kB	NVIC
AHB2	0x48001400 – 0x480017FF	1 kB	GPIOF
	0x48000C00 – 0x48000FFF	1 kB	GPIOD
	0x48000800 – 0x48000BFF	1 kB	GPIOC
	0x48000400 – 0x480007FF	1 kB	GPIOB
	0x48000000 – 0x480003FF	1 kB	GPIOA
AHB1	0x40021000 – 0x400213FF	1 kB	RCC
APB	0x40014800 – 0X40014BFF	1 kB	TIM17
	0x40014400 – 0X400147FF	1 kB	TIM16
	0x40014000 – 0X400143FF	1 kB	TIM15
	0x40012C00 – 0x40012FFF	1 kB	TIM1
	0x40012400 – 0x400127FF	1 kB	ADC
	0x40010400 – 0x400107FF	1 kB	EXTI
	0x40010000 – 0x400103FF	1 kB	SYSCFG
	0x40002000 – 0x400023FF	1 kB	TIM14
	0x40001400 – 0x400017FF	1 kB	TIM7
	0x40001000 – 0x400013FF	1 kB	TIM6
	0x40000400 – 0x400007FF	1 kB	TIM3

Configurazione dell'ADC

offset	register	content															
		31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x00	ADC_ISR	res								AWD	res		OVR	EOSEQ	EOC	EOSMP	ADRDY
		res															
0x04	ADC_IER	res								AWDIE			OVRIE	EOSEQIE	EOCIE	EOSPMIE	ADRDYIE
		res															
0x08	ADC_CR	ADCAL	res										ADSTP	res	ADSTART	ADDIS	ADEN
		res															
0x0C	ADC_CFGR1	AUTOFF	WAT	CONT	OVRMOD	EXTEN[1:0]		res	EXTSEL[2:0]			ALIGN	RES[1:0]		SCANDIR	DMACFG	DMAEN
		res	AWDCH[4:0]					res	AWDEN	AWDSGL		res					DISCEN
0x10	ADC_CFGR2	CKMODE[1:0]	res														
		res															
0x14	ADC_SMPR	res													SMP[1:0]		
		res															
0x20	ADC_TR	res				LT[11:0]											
		res				HT[11:0]											
0x28	ADC_CHSELR	CHSEL15	CHSEL14	CHSEL13	CHSEL12	CHSEL11	CHSEL10	CHSEL9	CHSEL8	CHSEL7	CHSEL6	CHSEL5	CHSEL4	CHSEL3	CHSEL2	CHSEL1	CHSEL0
		res														CHSEL17	CHSEL18
0x40	ADC_DR	DATA[15:0]															
		res															
0x308	ADC_CCR	res															
		res								TSEN	VREFEN	res					
0x18	RCC_APB2ENR	...				TIM1_EN	...	ADC_EN	...								SYSCFG_EN
		...													TIM17_EN	TIM16_EN	TIM15_EN

Esempio: Calibrazione

```
/* (0) Enable clock on ADC interface */
/* (1) Ensure that ADEN = 0 */
/* (2) Clear ADEN by setting ADDIS*/
/* (3) Launch the calibration by setting ADCAL */
/* (4) Wait until ADCAL=0 */
RCC->APB2ENR |= RCC_APB2ENR_ADCEN; /* (0) */
if ((ADC1->CR & ADC_CR_ADEN) != 0){ /* (1) */
        ADC1->CR |= ADC_CR_ADDIS; /* (2) */
}
while ((ADC1->CR & ADC_CR_ADEN) != 0){
        /* For robust implementation, add here time-out management */
}
ADC1->CR |= ADC_CR_ADCAL; /* (3) */
while ((ADC1->CR & ADC_CR_ADCAL) != 0){ /* (4) */
        /* For robust implementation, add here time-out management */
}
```

Esempio: Calibrazione

```
/* (0) Enable clock on ADC interface */
/* (1) Ensure that ADEN = 0 */
/* (2) Clear ADEN by setting ADDIS*/
/* (3) Launch the calibration by setting ADCAL */
/* (4) Wait until ADCAL=0 */
*((int*) 0x40021018) |=  0x200; /* (0) RCC->APB2ENR */
if ((*((int*) 0x40012408) & 0x1) != 0) { /* (1) ADC1->CR */
*((int*) 0x40012408) |= 0x2; /* (2) ADC1->CR */
}
while ((*((int*) 0x40012408) & 0x1) != 0){
/* For robust implementation, add here time-out management */
}
*((int*) 0x40012408) |= 0x80000000; /* (3) ADC1->CR */
while ((*((int*) 0x40012408) & 0x80000000) != 0){ /* (4) ADC1->CR */
/* For robust implementation, add here time-out management */
}
```

Esempio: Abilitazione

```
/* (0) Enable clock on ADC interface */
/* (1) Ensure that ADRDY = 0 */
/* (2) Clear ADRDY */
/* (3) Enable the ADC */
/* (4) Wait until ADC ready */
RCC->APB2ENR |= RCC_APB2ENR_ADCEN; /* (0) */
if ((ADC1->ISR & ADC_ISR_ADRDY) != 0){ /* (1) */
        ADC1->ISR |= ADC_ISR_ADRDY; /* (2) */
}
ADC1->CR |= ADC_CR_ADEN; /* (3) */
while ((ADC1->ISR & ADC_ISR_ADRDY) == 0){ /* (4) */
        /* For robust implementation, add here time-out management */
}
```

Esempio: Abilitazione

```
/* (0) Enable clock on ADC interface */
/* (1) Ensure that ADRDY = 0 */
/* (2) Clear ADRDY */
/* (3) Enable the ADC */
/* (4) Wait until ADC ready */
*((int*) 0x40021018) |=  0x200; /* (0) RCC->APB2ENR */
if ((*((int*) 0x40012400) & 0x1) != 0){  /* (1) ADC1->ISR */
        *((int*) 0x40012400) |= 0x1; /* (2) ADC1->ISR */
}
*((int*) 0x40012408) |= 0x1; /* (3) ADC1->CR */
while ((*((int*) 0x40012400) & 0x1) == 0){ /* (4) ADC1->ISR */
        /* For robust implementation, add here time-out management */
}
```

Esempio: Conversione Single Mode

```
/* (1) Select ADCCLK by writing 00 in CKMODE (reset value) */
/* (2) Select CHSEL0, CHSEL1 corresponding to PA0 and PA1 */
/* (3) Select a sampling mode of 111 i.e. 239.5 ADC clk to be greater
than 17.1us */
int ADC_Result_1, ADC_Result_0;
ADC1->CFGR2 &= ~ADC_CFGR2_CKMODE; /* (1) */
ADC1->CHSELR = ADC_CHSELR_CHSEL1 | ADC_CHSELR_CHSEL0; /* (2) */
ADC1->SMPR |= ADC_SMPR_SMP_0|ADC_SMPR_SMP_1| ADC_SMPR_SMP_2; /* (3) */
ADC1->CR |= ADC_CR_ADSTART; /* Start the ADC conversion */
while ((ADC1->ISR & ADC_ISR_EOC) == 0){ /* Wait end of 1st conv */
        /* For robust implementation, add here time-out management */
}
ADC_Result_0 = ADC1->DR; /* Store ADC conv result (1st channel PA0) */
while ((ADC1->ISR & ADC_ISR_EOC) == 0){ /* Wait end of 2nd conv */
        /* For robust implementation, add here time-out management */
}
ADC_Result_1 = ADC1->DR; /* Store ADC conv result (2nd channel PA1) */
```

Esempio: Conversione Single Mode

```
/* (1) Select ADCCLK by writing 00 in CKMODE (reset value) */
/* (2) Select CHSEL0, CHSEL1 corresponding to PA0 and PA1 */
/* (3) Select a sampling mode of 111 i.e. 239.5 ADC clk to be greater
than 17.1us */
int ADC_Result_1, ADC_Result_0;
*((int*) 0x40012410) &= ~0xC0000000; /* (1) ADC1->CFGR2 */
*((int*) 0x40012428) = 0x2 | 0x1; /* (2) ADC1->CHSELR */
*((int*) 0x40012414) |= 0b111; /* (3) ADC1->SMPR */
*((int*) 0x40012408) |= 0x4; /* Start the ADC conversion */
while ((*((int*) 0x40012400) & 0x4) == 0){ /* Wait end of 1st conv */
        /* For robust implementation, add here time-out management */
}
ADC_Result_0 = *((int*) 0x40012440); /* Store ADC conv result (PA0) */
while ((*((int*) 0x40012400) & 0x4) == 0){ /* Wait end of 2nd conv */
        /* For robust implementation, add here time-out management */
}
ADC_Result_1 = *((int*) 0x40012440); /* Store ADC conv result (PA1)*/
```

Esempio: Conversione Continuous Mode

```
/* (1) Select ADCCLK by writing 00 in CKMODE (reset value) */
/* (2) Select the continuous mode */
/* (3) Select CHSEL0 */
/* (4) Select a sampling mode of 111 i.e. 239.5 ADC clk to be greater
than 17.1us */
/* (5) Enable interrupts on EOC, EOSEQ and overrrun */
ADC1->CFGR2 &= ~ADC_CFGR2_CKMODE; /* (1) */
ADC1->CFGR1 |= ADC_CFGR1_CONT; /* (2) */
ADC1->CHSELR = ADC_CHSELR_CHSEL0; /* (3) */
ADC1->SMPR |= ADC_SMPR_SMP_0|ADC_SMPR_SMP_1|ADC_SMPR_SMP_2; /* (4) */
ADC1->IER = ADC_IER_EOCIE | ADC_IER_EOSEQIE | ADC_IER_OVRIE; /* (5) */
ADC1->CR |= ADC_CR_ADSTART; /* Start the ADC conversion */
/* Configure NVIC for ADC */
/* (7) Enable Interrupt on ADC */
/* (8) Set priority for ADC */
NVIC_EnableIRQ(ADC1_COMP_IRQn); /* (7) */
NVIC_SetPriority(ADC1_COMP_IRQn,0); /* (8) */
```

Esempio: Conversione Continuous Mode

```
/* (0) manage ADC output reading with ISR */
/* (1) copy ADC output into a global variable */
/* (2) it is not necessary to reset EOC interrupt pending flag of the
ADC register ISR, since it is reset by hardware once DR is read */
int ADC_Result[100];
int i = 0;

void ADC_IRQHandler(void){

        ADC_Result[i] = ADC1->DR; /* (1) */
        i = i + 1;

}
```

Esempio: Conversione Continuous Mode

```
/* (1) Select ADCCLK by writing 00 in CKMODE (reset value) */
/* (2) Select the continuous mode */
/* (3) Select CHSEL0 */
/* (4) Select a sampling mode of 111 i.e. 239.5 ADC clk to be greater
than 17.1us */
/* (5) Enable interrupts on EOC, EOSEQ and overrrun */
*((int*) 0x40012410) &= ~ADC_CFGR2_CKMODE; /* (1) ADC1->CFGR2 */
*((int*) 0x4001240C) |= ADC_CFGR1_CONT; /* (2) ADC1->CFGR1 */
*((int*) 0x40012428) = 0x1; /* (3) ADC1->CHSELR */
*((int*) 0x4001240C) |= 0xb111; /* (4) ADC1->SMPR */
*((int*) 0x40012404) = 0xb11100; /* (5) ADC1->IER */
/* Configure NVIC for ADC */
/* (6) Enable Interrupt on ADC */
/* (7) Set priority for ADC */
*((int*) 0xE000E100) |= 0x1000; /* (6) NVIC->ISER */
*((int*) 0xE000E40C) &= 0xFFFFFF00; /* (7) NVIC->IPR5 */
```

Esempio: Conversione Continuous Mode

```
/* (0) manage ADC output reading with ISR */
/* (1) copy ADC output into a global variable */
/* (2) it is not necessary to reset EOC interrupt pending flag of the
ADC register ISR, since it is reset by hardware once DR is read */
int ADC_Result[100];
int i = 0;

void ADC_IRQHandler(void){

        ADC_Result[i] = *((int*) 0x40012440); /* (1) */
        i = i + 1;

}
```

Riassunto

- **Output Analogici**
  - configurazione **tempi PWM** tramite i registri PSC, ARR e CCRx
  - settaggio **PWM mode e abilitazione** con registri CCMR e CCER
  - configurazione **porta** con **alternate function** apposita
- **Input Analogici**
  - **calibrazione** con ADC_CR_CAL nel registro CR
  - **abilitazione** in base a ADC_ISR_ADRDY nel registro ISR
  - configurazione **clock, canali, modalità** tramite registri CFGR2, CHSEL e SMPR
  - **abilitazione** e **gestione** con ADC_CR_ADSTART nel registro CR

Esercizi

- Uscite Analogiche
- Ingressi Analogici

Esercizi

- Uscite Analogiche
- Ingressi Analogici

Uscite Analogiche

- Scrivere un **codice C** per microcontrollore STM32F030R8 per la generazione di un'**uscita analogica** tramite **PWM** generata col **timer 14** in base al valore di comparazione contenuto in **CCR1**. L'uscita analogica va messa sul **pin 7** della **porta A** e deve essere pari a circa **1 V** (3.3 V di escursione massima del segnale).

Configurazione della PWM

pin name	AF0	AF1	AF2	AF3	AF4	AF5	AF6
PA0	-	USART2_CTS	-	-	-	-	-
PA1	EVENTOUT	USART2_RTS	-	-	-	-	-
PA2	TIM15_CH1	USART2_TX	-	-	-	-	-
PA3	TIM15_CH2	USART2_RX	-	-	-	-	-
PA4	SPI1_NSS	USART2_CK	-	-	TIM14_CH1	-	-
PA5	SPI1_SCK	-	-	-	-	-	-
PA6	SPI1_MISO	TIM3_CH1	TIM1_BKIN	-	-	TIM16_CH1	EVENTOUT
PA7	SPI1_MOSI	TIM3_CH2	TIM1_CH1N	-	TIM14_CH1	TIM17_CH1	EVENTOUT
PA8	MCO	USART1_CK	TIM1_CH1	EVENTOUT	-	-	-
PA9	TIM15_BKIN	USART1_TX	TIM1_CH2	-	-	-	-
PA10	TIM15_BKIN	USART1_RX	TIM1_CH3	-	-	-	-
PA11	EVENTOUT	USART1_CTS	TIM1_CH4	-	-	SCL	-
PA12	EVENTOUT	USART1_RTS	TIM1_ETR	-	-	SDA	-
PA13	SWDIO	IR_OUT	-	-	-	-	-
PA14	SWCLK	USART2_TX	-	-	-	-	-
PA15	SPI1_NSS	USART2_RX	-	EVENTOUT	-	-	-

Configurazione della PWM

pin name	AF0	AF1	AF2	AF3	AF4	AF5
PB0	EVENTOUT	TIM3_CH3	TIM1_CH2N	-	-	-
PB1	TIM14_CH1	TIM3_CH4	TIM1_CH3N	-	-	-
PB2	-	-	-	-	-	-
PB3	SPI1_SCK	EVENTOUT	-	-	-	-
PB4	SPI1_MISO	TIM3_CH1	EVENTOUT	-	-	-
PB5	SPI1_MOSI	TIM3_CH2	TIM16_BKIN	I2C1_SMBA	-	-
PB6	USART1_TX	I2C1_SCL	TIM16_CH1N	-	-	-
PB7	USART1_RX	I2C2_SDA	TIM17_CH1N	-	-	-
PB8	-	I2C1_SCL	TIM16_CH1	-	-	-
PB9	IR_OUT	I2C1_SDA	TIM17_CH1	EVENTOUT	-	-
PB10	-	I2C2_SCL	-	-	-	-
PB11	EVENTOUT	I2C2_SDA	-	-	-	-
PB12	SPI2_NSS	EVENTOUT	TIM1_BKIN	-	-	-
PB13	SPI2_SCK	-	TIM1_CH1N	-	-	-
PB14	SPI2_MISO	TIM15_CH1	TIM1_CH2N	-	-	-
PB15	SPI2_MOSI	TIM15_CH2	TIM1_CH3N	TIM15_CH1N	-	-

Configurazione della PWM

pin name	AF0
PC0	EVENTOUT
PC1	EVENTOUT
PC2	EVENTOUT
PC3	EVENTOUT
PC4	EVENTOUT
PC5	-
PC6	TIM3_CH1
PC7	TIM3_CH2
PC8	TIM3_CH3
PC9	TIM3_CH4
PC10	-
PC11	-
PC12	-
PC13	-
PC14	-
PC15	-

pin name	AF0
PD2	TIM3_ETR

pin name	AF0
PF0	.
PF1	-
PF4	-
PF5	-
PF6	I2C2_SCL
PF7	I2C2_SDA

Le porte C, D e F hanno **solo una alternate function possibile** nell'STM32F030R8 per cui **non vi sono registri GPIOC_AFR, GPIOD_AFR, GPIOF_AFR**

Configurazione della PWM

bus	boundary address	size	peripheral
Cortex-M0 Internal Peripherals	0xE000E100 – 0xE000E4FF	1 kB	NVIC
AHB2	0x48001400 – 0x480017FF	1 kB	GPIOF
	0x48000C00 – 0x48000FFF	1 kB	GPIOD
	0x48000800 – 0x48000BFF	1 kB	GPIOC
	0x48000400 – 0x480007FF	1 kB	GPIOB
	0x48000000 – 0x480003FF	1 kB	GPIOA
AHB1	0x40021000 – 0x400213FF	1 kB	RCC
APB	0x40014800 – 0X40014BFF	1 kB	TIM17
	0x40014400 – 0X400147FF	1 kB	TIM16
	0x40014000 – 0X400143FF	1 kB	TIM15
	0x40012C00 – 0x40012FFF	1 kB	TIM1
	0x40010400 – 0x400107FF	1 kB	EXTI
	0x40010000 – 0x400103FF	1 kB	SYSCFG
	0x40002000 – 0x400023FF	1 kB	TIM14
	0x40001400 – 0x400017FF	1 kB	TIM7
	0x40001000 – 0x400013FF	1 kB	TIM6
	0x40000400 – 0x400007FF	1 kB	TIM3

Configurazione della PWM

offset	register	content															
		31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x00	GPIOx_MODER	MODER7[1:0] MODER15[1:0]		MODER6[1:0] MODER14[1:0]		MODER5[1:0] MODER13[1:0]		MODER4[1:0] MODER12[1:0]		MODER3[1:0] MODER11[1:0]		MODER2[1:0] MODER10[1:0]		MODER1[1:0] MODER9[1:0]		MODER0[1:0] MODER8[1:0]	
0x04	GPIOx_OTYPER	OT[15:0] res															
0x08	GPIOx_OSPEEDR	OSPEEDR7[1:0] OSPEEDR15[1:0]		OSPEEDR6[1:0] OSPEEDR14[1:0]		OSPEEDR5[1:0] OSPEEDR13[1:0]		OSPEEDR4[1:0] OSPEEDR12[1:0]		OSPEEDR3[1:0] OSPEEDR11[1:0]		OSPEEDR2[1:0] OSPEEDR10[1:0]		OSPEEDR1[1:0] OSPEEDR9[1:0]		OSPEEDR0[1:0] OSPEEDR8[1:0]	
0x0C	GPIOx_PUPDR	PUPDR7[1:0] PUPDR15[1:0]		PUPDR6[1:0] PUPDR14[1:0]		PUPDR5[1:0] PUPDR13[1:0]		PUPDR4[1:0] PUPDR12[1:0]		PUPDR3[1:0] PUPDR11[1:0]		PUPDR2[1:0] PUPDR10[1:0]		PUPDR1[1:0] PUPDR9[1:0]		PUPDR0[1:0] PUPDR8[1:0]	
0x10	GPIOx_IDR	IDR[15:0] res															
0x14	GPIOx_ODR	ODR[15:0] res															
0x18	GPIOx_BSRR	BS[15:0] BR[15:0]															
0x1C	GPIOx_LCKR	LCK[15:0] res															LCKK
0x20	GPIOx_AFRL	AFSEL3[3:0] AFSEL7[3:0]				AFSEL2[3:0] AFSEL6[3:0]				AFSEL1[3:0] AFSEL5[3:0]				AFSEL0[3:0] AFSEL4[3:0]			
0x24	GPIOx_AFRH	AFSEL11[3:0] AFSEL15[3:0]				AFSEL10[3:0] AFSEL14[3:0]				AFSEL9[3:0] AFSEL13[3:0]				AFSEL8[3:0] AFSEL12[3:0]			
0x28	GPIOx_BRR	BRR[15:0] res															

Configurazione della PWM

offset	register	content															
		31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x00	TIMx_CR1	res						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
		res															
0x0C	TIMx_DIER	...											CC4IE	CC3IE	CC2IE	CC1IE	UIE
		res															
0x10	TIMx_SR	res			CC4OF	CC3OF	CC2OF	CC1OF	...				CC4IF	CC3IF	CC2IF	CC1IF	UIF
		res															
0x14	TIMx_EGR	...											CC4G	CC3G	CC2G	CC1G	UG
		res															
0x18	TIMx_CCMR1	OC2CE/ IC2F[3]	OC2M[2] /IC2F[2]	OC2M[1] /IC2F[1]	OC2M[0] /IC2F[0]	OC2PE/ IC2PSC[1]	OC2FE/ IC2PSC[0]	CC2S[1:0]		OC1CE/ IC1F[3]	OC1M[2] /IC1F[2]	OC1M[1] /IC1F[1]	OC1M[0] /IC1F[0]	OC1PE/ IC1PSC[1]	OC1FE/ IC1PSC[0]	CC1S[1:0]	
		res															
0x1C	TIMx_CCMR2	OC4CE/ IC4F[3]	OC4M[2] /IC4F[2]	OC4M[1] /IC4F[1]	OC4M[0] /IC4F[0]	OC4PE/ IC4PSC[1]	OC4FE/ IC4PSC[0]	CC4S[1:0]		OC3CE/ IC3F[3]	OC3M[2] /IC3F[2]	OC3M[1] /IC3F[1]	OC3M[0] /IC3F[0]	OC3PE/ IC3PSC[1]	OC3FE/ IC3PSC[0]	CC3S[1:0]	
		res															
0x20	TIMx_CCER	CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
		res															
0x24	TIMx_CNT	CNT[15:0]															
		res															
0x28	TIMx_PSC	PSC[15:0]															
		res															
0x2C	TIMx_ARR	ARR[15:0]															
		res															

Configurazione della PWM

offset	register	content															
		31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x34	TIMx_CCR1	CCR1[15:0]															
		res															
0x38	TIMx_CCR2	CCR2[15:0]															
		res															
0x3C	TIMx_CCR3	CCR3[15:0]															
		res															
0x40	TIMx_CCR4	CCR4[15:0]															
		res															
0x44	TIMx_BDTR	MOE	AOE	BKP	OSSR	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
		res															
0x14	RCC_AHBENR	...															
		...									IOPF_EN	res	IOPD_EN	IOPC_EN	IOPB_EN	IOPA_EN	...
0x18	RCC_APB2ENR	...				TIM1_EN		...									SYSCFG_EN
		...													TIM17_EN	TIM16_EN	TIM15_EN
0x1C	RCC_APB1ENR	...							TIM14_EN	res		TIM7_EN	TIM6_EN	res		TIM3_EN	res
		...															

Configurazione della PWM

bus	boundary address	size	peripheral	description	edge-aligned PWM (E) and/or center-aligned PWM (C)	number of CC channels
APB	0x40014800 – 0X40014BFF	1 kB	TIM17	general purpose timer	E	1
	0x40014400 – 0X400147FF	1 kB	TIM16	general purpose timer	E	1
	0x40014000 – 0X400143FF	1 kB	TIM15	general purpose timer	E	2
	0x40012C00 – 0x40012FFF	1 kB	TIM1	advanced control timer	E, C	4
	0x40002000 – 0x400023FF	1 kB	TIM14	general purpose timer	E	1
	0x40001400 – 0x400017FF	1 kB	TIM7	basic timer	-	0
	0x40001000 – 0x400013FF	1 kB	TIM6	basic timer	-	0
	0x40000400 – 0x400007FF	1 kB	TIM3	general purpose timer	E, C	4
AHB1	0x40021000 – 0x400213FF	1 kB	RCC	reset and clock control	-	-

Esercizi

- Uscite Analogiche
- Ingressi Analogici

Ingressi Analogici

- Scrivere un **codice C** per SM32F030R8 che acquisisce un segnale analogico in input ai pin **ADC_IN4** e **ADC_IN5** (corrispondenti ai pin PA4 e PA5) con **risoluzione massima** (pari a **12 bit**) e **tempo di campionamento minimo** (SMP pari a 3'b000) in modalità **discontinuous mode**.

Configurazione dell'ADC

bus	boundary address	size	peripheral
Cortex-M0 Internal Peripherals	0xE000E100 – 0xE000E4FF	1 kB	NVIC
AHB2	0x48001400 – 0x480017FF	1 kB	GPIOF
	0x48000C00 – 0x48000FFF	1 kB	GPIOD
	0x48000800 – 0x48000BFF	1 kB	GPIOC
	0x48000400 – 0x480007FF	1 kB	GPIOB
	0x48000000 – 0x480003FF	1 kB	GPIOA
AHB1	0x40021000 – 0x400213FF	1 kB	RCC
APB	0x40014800 – 0X40014BFF	1 kB	TIM17
	0x40014400 – 0X400147FF	1 kB	TIM16
	0x40014000 – 0X400143FF	1 kB	TIM15
	0x40012C00 – 0x40012FFF	1 kB	TIM1
	0x40012400 – 0x400127FF	1 kB	ADC
	0x40010400 – 0x400107FF	1 kB	EXTI
	0x40010000 – 0x400103FF	1 kB	SYSCFG
	0x40002000 – 0x400023FF	1 kB	TIM14
	0x40001400 – 0x400017FF	1 kB	TIM7
	0x40001000 – 0x400013FF	1 kB	TIM6
	0x40000400 – 0x400007FF	1 kB	TIM3

Configurazione dell'ADC

offset	register	content															
		31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x00	ADC_ISR	res								AWD	res		OVR	EOSEQ	EOC	EOSMP	ADRDY
		res															
0x04	ADC_IER	res								AWDIE			OVRIE	EOSEQIE	EOCIE	EOSPMIE	ADRDYIE
		res															
0x08	ADC_CR	ADCAL	res										ADSTP	res	ADSTART	ADDIS	ADEN
		res															
0x0C	ADC_CFGR1	AUTOFF	WAT	CONT	OVRMOD	EXTEN[1:0]		res	EXTSEL[2:0]			ALIGN	RES[1:0]		SCANDIR	DMACFG	DMAEN
		res	AWDCH[4:0]					res	AWDEN	AWDSGL		res					DISCEN
0x10	ADC_CFGR2	CKMODE[1:0]	res														
		res															
0x14	ADC_SMPR	res													SMP[1:0]		
		res															
0x20	ADC_TR	res				LT[11:0]											
		res				HT[11:0]											
0x28	ADC_CHSELR	CHSEL15	CHSEL14	CHSEL13	CHSEL12	CHSEL11	CHSEL10	CHSEL9	CHSEL8	CHSEL7	CHSEL6	CHSEL5	CHSEL4	CHSEL3	CHSEL2	CHSEL1	CHSEL0
		res														CHSEL17	CHSEL18
0x40	ADC_DR	DATA[15:0]															
		res															
0x308	ADC_CCR	res															
		res								TSEN	VREFEN	res					
0x18	RCC_APB2ENR	...				TIM1_EN	...	ADC_EN	...								SYSCFG_EN
		...													TIM17_EN	TIM16_EN	TIM15_EN