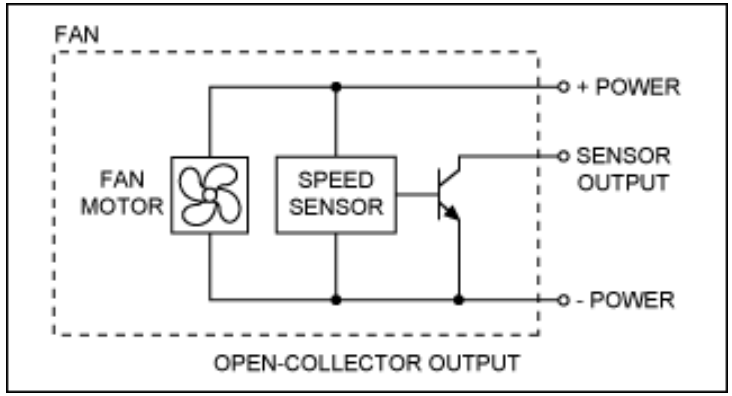
Controllo velocità di rotazione di una ventola con scheda KL25Z

La ventola

- Ventilatore alimentato in DC;
- Cuscinetto a sfera;
- Encoder per misura della velocità di rotazione;
- Velocità di rotazione funzione della tensione di alimentazione.



Model No.	Rated Voltage [V]	Operating Voltage Range [V]	Rated Current [A]	Rated Input [W]	Rated Speed [min⁻¹]	Max. Airflow [m³/min]	Max. Airflow [CFM]	Max. Static Pressure [Pa]	Max. Static Pressure [inchH₂O]	SPL [dB(A)]	Operating Temperature [℃]	Expected Life [h]
9S0912F401	12	5.0 to 13.8	0.14	1.68	2,650	1.26	44.5	30.0	0.120	27	−10 to +70	40,000
9S0912M401			0.11	1.32	2,250	1.07	37.8	21.6	0.087	22		
9S0912L401		6.0 to 13.8	0.07	0.84	1,750	0.83	29.3	13.1	0.053	17		
9S0924F401	24	14 to 26.4	0.09	2.16	2,650	1.26	44.5	30.0	0.120	27		
9S0924M401			0.07	1.68	2,250	1.07	37.8	21.6	0.087	22		
9S0924L401			0.04	0.96	1,750	0.83	29.3	13.1	0.053	17		

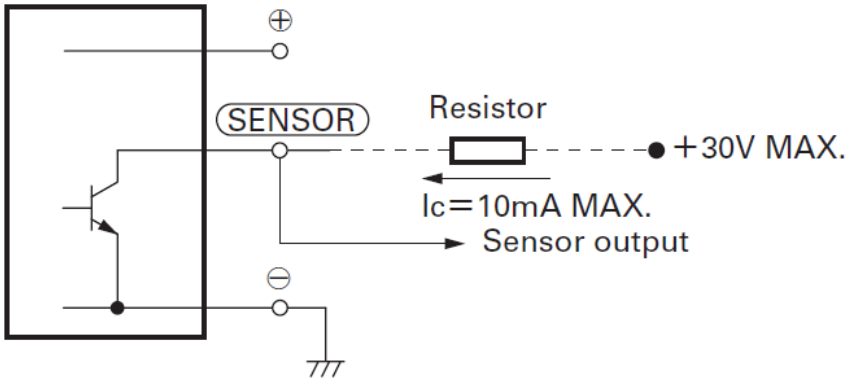
Sensore velocità di rotazione

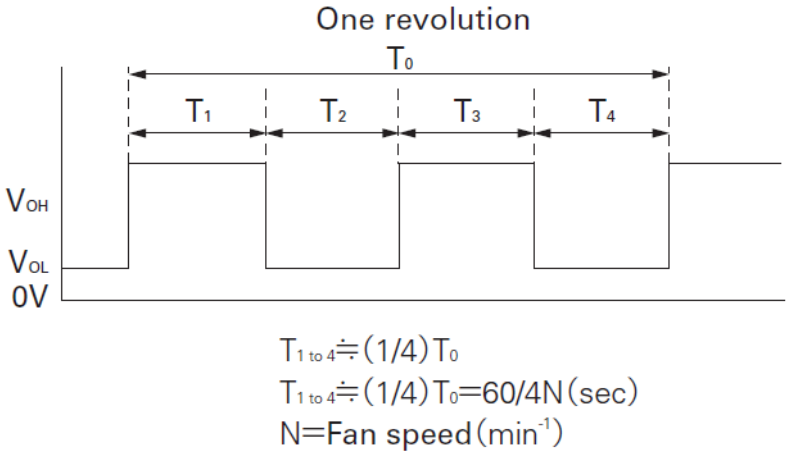
- Il sensore di velocità è basato su encoder. È perciò generato un segnale digitale la cui frequenza è proporzionale alla velocità di rotazione.

- Tipo di uscita: **open collector** ⟶ Necessario un resistore di pull-up esterno

BJT ON → $V_{out} = V_{CEsat}$ (<0.4V)

BJT OFF → I_C=0 → Caduta di tensione su R nulla → $V_{out} = V_{DD}$ (3.3V)





$T_{1\ to\ 4} \fallingdotseq (1/4)T_0$

$T_{1\ to\ 4} \fallingdotseq (1/4)T_0 = 60/4N\,(sec)$

N=Fan speed (min^{-1})

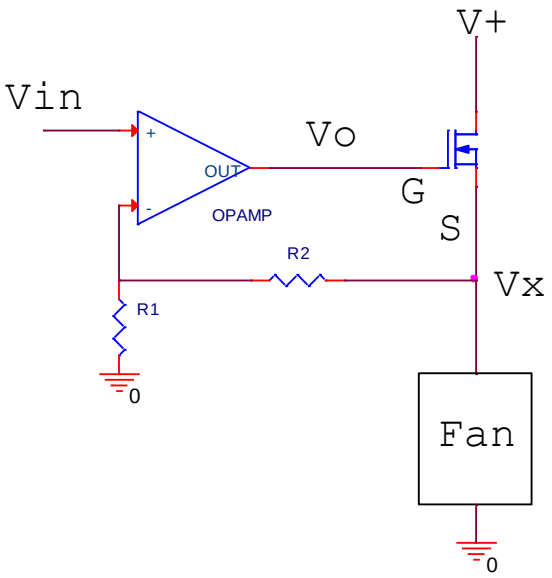
Alimentazione ventola

Tensione massima di uscita KL25Z: 3.3V

Alimentazione ventola: 12V

Corrente richiesta dalla ventola: 70mA

Circuito di amplificazione necessario



$$V_x = \left(1 + \frac{R_2}{R_1}\right) V_{in}$$

3V

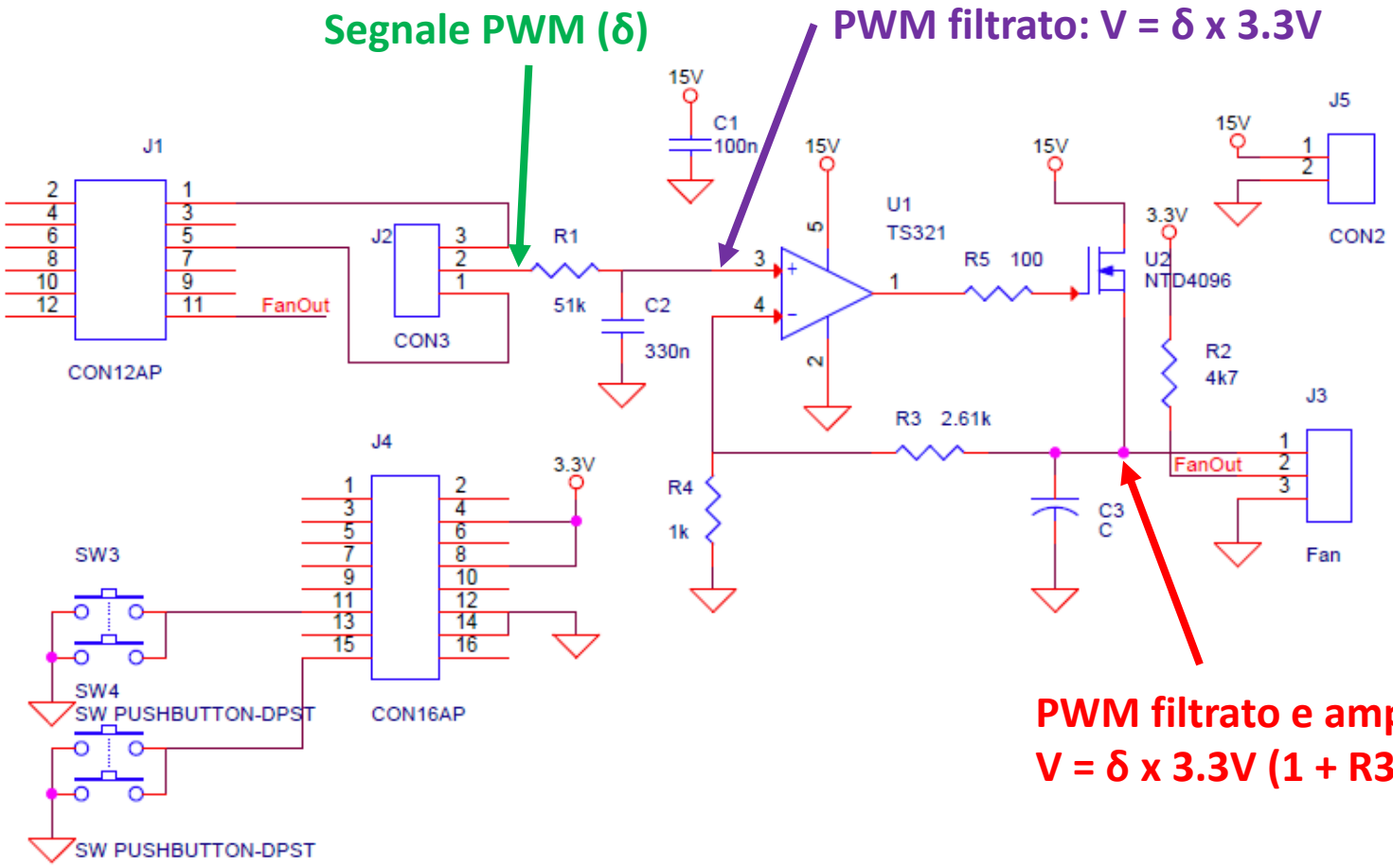
$$V_0 = V_x + V_{GS} > V_x + V_T$$

L'uscita dell'op-amp (V_0) non deve saturare, per cui la tensione di alimentazione dovrà essere:

$$V_+ > V_{x\_max} + V_T = 15V$$

Corrente in uscita fornita dal MOSFET di potenza

Circuito di interfaccia per ventola



FanOut: **PTE20 TPM1_CH0**
Vin: **PTE30 TPM0_CH3**

Materiale fornito per l'esecitazione:

1) Scheda KL25Z

2) Ventola

3) Circuito di interfaccia per la ventola

4) Adattatore USB

Obiettivi del progetto:

1) Generazione di un segnale PWM che permetta di regolare l'alimentazione della ventola e perciò la velocità di rotazione.

2) Implementazione, mediante timer in modalità «input capture», di un sistema di misura della velocità di rotazione della ventola. Sarà necessario sia misurare la durata di un evento esterno (legato al sensore di velocità della ventola), che calcolare internamente la frequenza di rotazione della ventola (in Hz o rpm).

3) Interfacciamento del microcontrollore, mediante periferica seriale, con il PC. Tale collegamento servirà innanzitutto per visualizzare le misure di frequenza eseguite. Inoltre, lo studente è libero di configurare la comunicazione in modo da inviare comandi utili al funzionamento del microcontrollore (ad esempio è possibile modificare dinamicamente il setpoint di velocità mediante seriale).

4) Caratterizzazione statica della ventola. Questo equivale a polarizzare la ventola con tensione (statica) variabile da 0 a Vmax e stimare (mediante la routine implementata su microcontrollore) la frequenza di rotazione. Per valutare la presenza di isteresi è consigliabile variare la tensione sia in salita che in discesa.

5) Analisi della risposta al gradino della ventola. Applicando un gradino di tensione (ovvero di duty clycle) alla ventola dovrà essere memorizzata la frequenza di rotazione in funzione del tempo. Sulla base di questo transitorio la ventola dovrà essere modellata come un sistema del primo ordine.

6) Facoltativo: nel calcolare la frequenza di rotazione (punto 2) si implementi un filtro a media mobile. Il filtro dovrà essere progettato sulla base del sistema da controllare e del rumore eventualmente osservato sperimentalmente.

7) Facoltativo: Implementazione di un controllore PI per regolare la velocità di rotazione.

Svolgimento dell'esercitazione:

1) Si configuri il TPM1 (CH0) come PWM. La frequenza del PWM deve essere impostata considerando il filtro presente in uscita. Dopo aver collegato la scheda e la ventola, si valuti con oscilloscopio la tensione ai capi della ventola stessa. In questa fase è consigliabile eseguire una caratterizzazione statica approssimativa della ventola, visualizzando il segnale di feedback su oscilloscopio. Questo servirà ad individuare il range di frequenza (e tempo), che dovrà essere poi misurato al punto successivo.

2) Mediante il TPM0 (CH3), in modalità "input capture", si misuri il tempo di rotazione della ventola. Il tempo misurato (ovvero il numero di conteggi) dovrà essere convertito in frequenza. Nelle slide successive sono forniti ulteriori dettagli.

3) La comunicazione seriale avviene mediante la periferica UART. Per interfacciare il PC con il microcontrollore si utilizzerà un adattatore USB (TTL-232R). Dovrà essere installato su PC un apposito software (ad esempio Hyper Terminal).

4) La caratterizzazione statica può essere eseguita visualizzando la frequenza di rotazione su PC, oppure lavorando con il microcontrollore in modalità debug (e monitorando una apposita variabile con watchpoint).

5) La risposta al gradino dovrà essere monitorata mediante PC. Si può decidere avviare il test a seguito di un reset, oppure si può implementare via software l'applicazione di una gradino, triggerato mediante UART o mediante viariable watchpoint in debug. In quest'ultimo caso è possibile applicare anche un gradino negativo.

Da riportare nella relazione:

- Codice con descrizione sintetica.

- Motivazioni delle scelte progettuali (configurazione delle periferiche e algoritmo sviluppato). Sarà quindi necessario riportare come la configurazione delle periferiche, la scelta del formato fixed point e l'algoritmo implementato influenzino le prestazioni del sistema (ad esempio accuratezza della frequenza misurata, ripple presente sulla tensione di alimentazione, etc.).

- Per ciascuno degli obiettivi del progetto è necessario riportare le evidenze sperimentali che testimonino il corretto funzionamento del sistema.

Configurazione UART

Al fine di inizializzare la UART è necessario eseguire i seguenti passi (fare riferimento al manuale del microcontrollore per maggiori informazioni):

- Agendo sui registri **SIM_SCGC4** e **SIM_SCGC5** abilitare il clock della periferica **UART1** ed della **porta E**;
- I pin da utilizzare come TX e RX della comunicazione seriale sono **PTE0** e **PTE1**. Selezionare perciò l'alternativa 3 nei registri **PORTE_PCR0** e **PORTE_PCR1**;
- Inserire il divisore (che consente di stabilire il baude rate desiderato) all'interno dei registri **UART*1*_BDH** e **UART*1*_BDL**. Si tratta di due registri a 8 bit. È necessario scrivere prima su BDH e poi su BDL.

  Calcolo del divisore: **BUSCLK/(16×BaudeRate)**

- I registri UART1_C1, UART1_S2, UART1_C3, possono essere lasciati al loro valore di default (0), in modo da ottenere una comunicazione a **8bit**, **senza bit di parità** e con **1 bit di stop**. Qualora si desiderino modificare tali parametri bisognerà agire sui registri di controllo.
- All'interno del registro **UART1_C2** è necessario settare gli appositi bit ***Transmitter Enable*** e ***Receiver Enable***. Lo studente valuti se abilitare l'interrupt in ricezione mediante il bit ***Receiver Interrupt Enable***. Nel caso di attivazione, sarà necessario gestire opportunamente la priorità dell'interrupt, facendo in modo che la sua priorità sia inferiore a quella dei timer.

Trasmissione dati con UART

```
while (!(UART1->S1 & UART_S1_TDRE_MASK));

UART1->D = data[i];
```

Il ciclo while aspetta che il buffer sia vuoto (ovvero che non ci siano precedenti invii in corso).

Il dato da inviare dovrà essere copiato nel registro **UART1_D** (ad 8 bit). Si ricorda che ogni singolo carattere da inviare corrisponde ad 1byte. Per cui *data* dovrà essere un vettore i cui elementi vanno inviati serialmente.

sprintf(data, "%u\r\n", temp);

Per tener conto della codifica ASCII si può utilizzare la routine sprintf. Sarà perciò necessario includere la seguente libreria:

#include <stdio.h>

Ricezione dati con UART (polling)

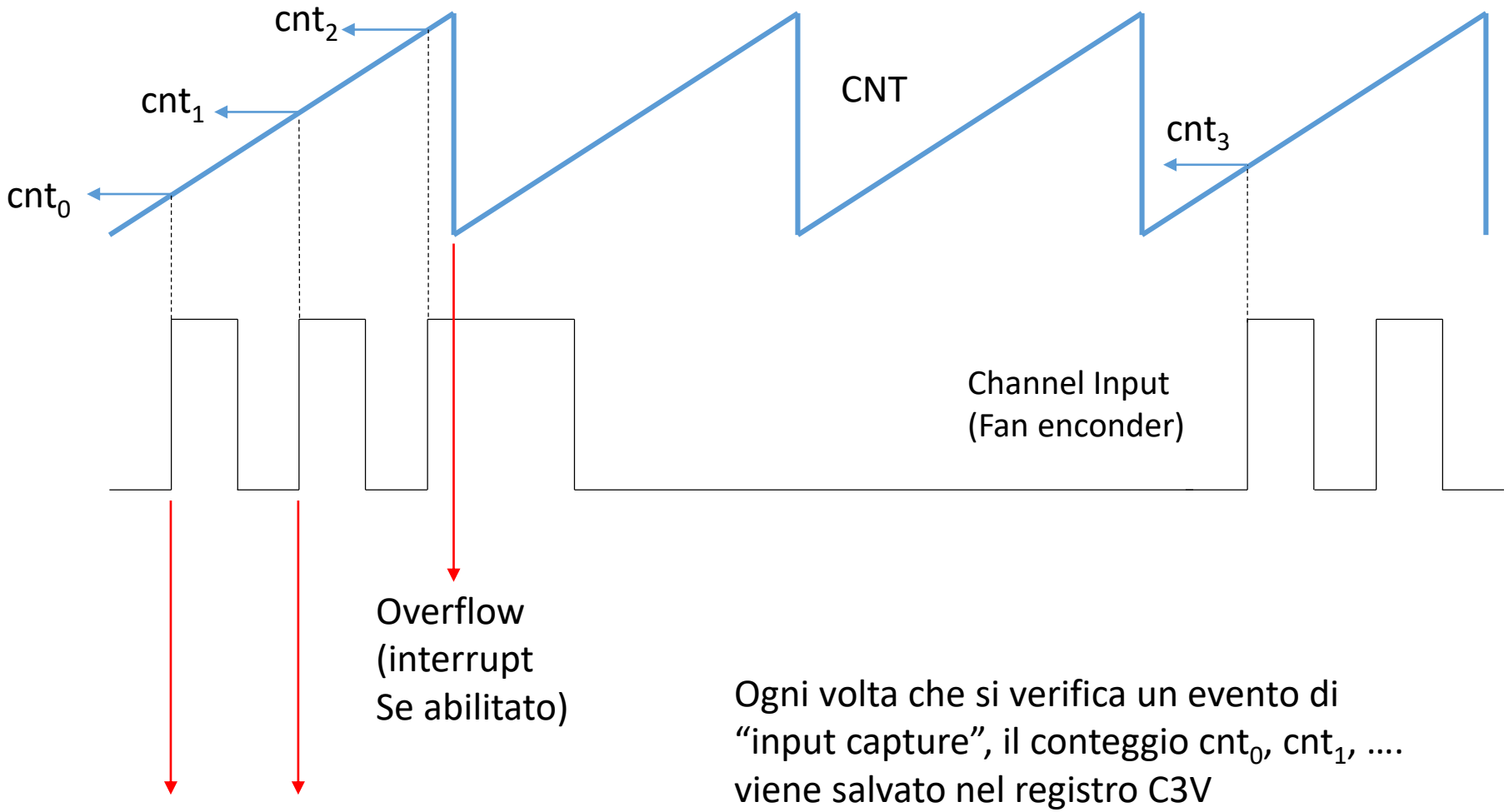
```
if( (UART1->S1 & UART_S1_RDRF_MASK))

data[i]=UART1->D;
```

Bisogna verificare la presenza di dati sul registro UART_D e poi proseguire con la lettura dei dati.

sscanf(data, "%u", &output);

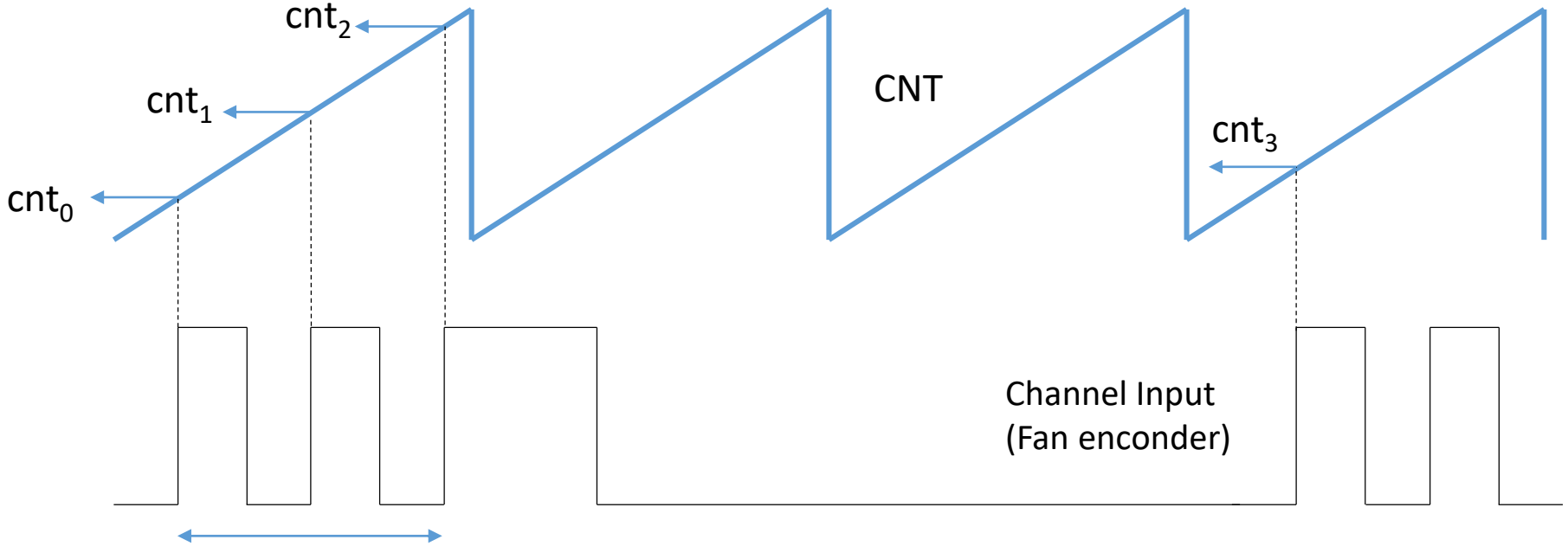
Il contenuto dell'array data (caratteri provenienti dalla seriale) viene memorizzato nella variabile output come un intero (%u)

Misura frequenza di rotazione (1/3)



Ogni volta che si verifica un evento di "input capture", il conteggio cnt_0, cnt_1, …. viene salvato nel registro C3V

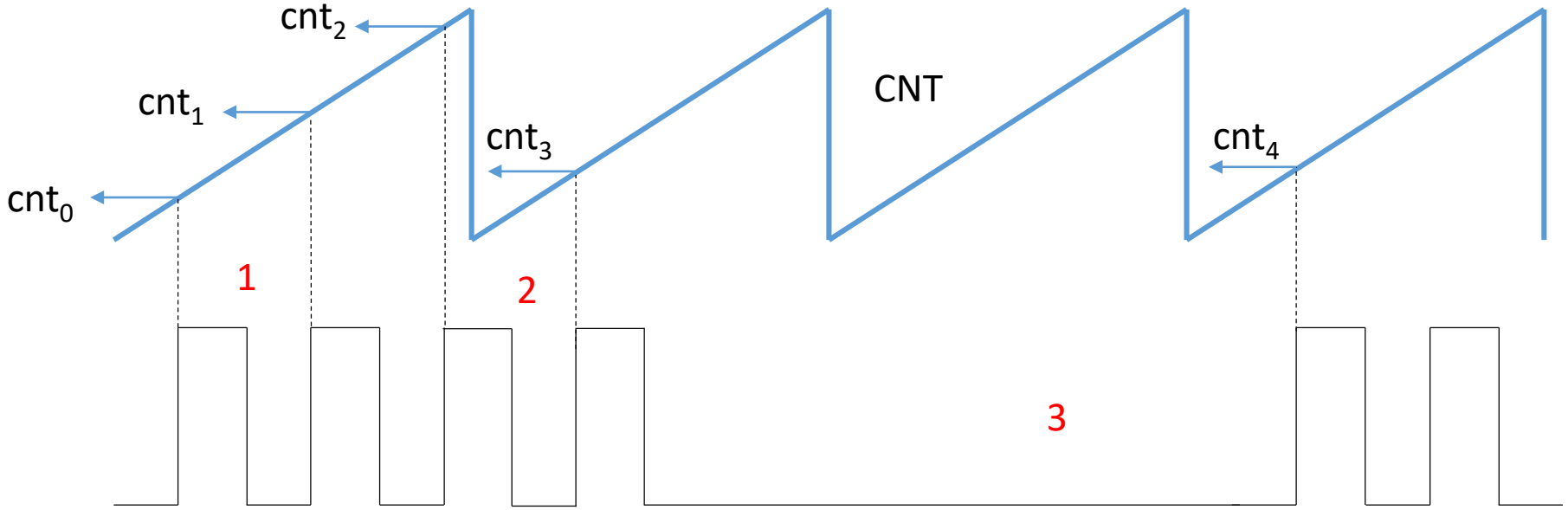
Misura frequenza di rotazione (2/3)



$$t_{FAN} = (cnt_2 - cnt_0)t_0 = \frac{(cnt_2 - cnt_0)}{f_0}$$

$$f_{FAN} = \frac{1}{t_{FAN}} = \frac{f_0}{(cnt_2 - cnt_0)} = \frac{f_{ck}}{(cnt_2 - cnt_0)PS}$$

Misura frequenza di rotazione (3/3)



Tre possibili scenari:

1. All'interno della stessa rampa si verificano due "input capture" (ad esempio cnt_2, cnt_1 e cnt_0). In questo caso è sufficiente calcolare la differenza (cnt_2-cnt_0).
2. Due eventi di "input capture" si verificano a cavallo di uno o più overflow del contatore (ad esempio cnt_3 e cnt_2). E' necessario correggere la differenza (cnt_3-cnt_1) sommando un'opportuna quantità.
3. Se la ventola è ferma, osserveremo diversi overflow senza che sia avvenenuto alcun evento di capture. Bisogna impostare quindi un numero massimo di overflow dopo il quale la ventola si considera ferma (*f* viene fissata a zero e non più calcolata). Il numero massimo di overflow deve essere legato alla frequenza minima della ventola.

Divisione tra interi (1/2)

L'espressione di f_{FAN} individuata in precedenza, prevede la divisione tra due interi. Tale operazione viene eseguita via software dal compilatore, ma il risultato ottenuto sarà anch'esso intero, per cui la parte frazionaria verrà trascurata.

Suggerimento per la divisione tra interi, mediante esempio numerico:

$A = 985_{10} = 1111011001_2$
$B = 231_{10} = \quad 11100111_2$

$C = A / B = 4.264069264$ Risultato atteso

$C = A / B = 4$ Nel caso di risultato intero

Moltiplichiamo e dividiamo arbitrariamente per 2^M

$$C = \frac{A}{B}\frac{2^M}{2^M}$$

A è rappresentato su 10 bit. Disponendo di una aritmetica a 32bit e considerando un bit per il segno, posso utilizzare un valore massimo di M pari a a 21 senza incorrere in saturazione (a valle della moltiplicazione per 2^M).

Scegliamo M = 10

$$C = \frac{A}{B}\frac{2^{10}}{2^{10}} = \frac{985 \cdot 2^{10}}{231}\frac{1}{2^{10}} = \frac{1008640}{231}\frac{1}{2^{10}} \cong 4366\frac{1}{2^{10}}$$

Divisione tra interi (2/2)

$A = 985_{10} = 1111011001_2$
$B = 231_{10} = \quad 11100111_2$

C = A / B = 4.264069264 Risultato atteso

C = A / B = 4 Nel caso di risultato intero

$$C = \frac{A}{B}\frac{2^{10}}{2^{10}} = \frac{985 \cdot 2^{10}}{231}\frac{1}{2^{10}} = \frac{1008640}{231}\frac{1}{2^{10}} \cong 4366\frac{1}{2^{10}} = \frac{1}{2^{10}}C_N$$

$$C_N = 4366_{10} = 00\ldots0001000100001110_2 =$$

parte Intera **= 4**

10 bit – parte frazionaria **=0.263671875**

$$C_N = 0100.0100001110_2$$

Formato di C_N **4.10**

M : coincide con il numero di cifre riservate alla parte frazionaria

La scelta di M deve tener conto di problematiche di overflow e quindi fare in modo che il numero di cifre riservate alla **parte intera** consenta di rappresentare il valore massimo atteso.

Timer in modalità input capture (1/2)

- Nel caso si decida di gestire l'evento di input capture tramite interrupt, oltre al generico interrupt della periferica (registro TPM0_SC, bit TOIE) è necessario abilitare anche quello specifico del canale (registro TPMO_C3SC, bit CHIE).

- La routine di servizio dell'interrupt del TPM0 sarà una sola. Per distinguere quale dei due eventi (overflow o input capture) ha sollevato l'interrupt, è necessario interrogare gli appostiti flag. A titolo di esempio si riporta tale operazione mediante costrutto *if*:

```
if(TPM0->STATUS&TPM_STATUS_TOF_MASK){
…
}

if(TPM0->STATUS&TPM_STATUS_CH3F_MASK){
…
}
```

- Si ricorda che all'interno della routine di servizio sarà necessario azzerare i flag di overflow o del canale:

```
TPM0->STATUS=TPM_STATUS_TOF_MASK

TPM0->STATUS=TPM_STATUS_CH3_MASK
```

Timer in modalità input capture (2/2)

- All'interno della ISR dell'input capture è necessario eseguire il calcolo della velocità di rotazione:
- Sulla base delle slides precedenti, la frequenza di rotazione può essere calcolata come:

$$f_{FAN} == \frac{f_{ck}}{(cnt_2 - cnt_0)PS} 2^M$$

2^M: fattore di scala (M rappresenta in numero di cifre della parte frazionaria)

- È consigliabile calcolare (moltiplicare e dividere) a priori tutte le costanti, in modo che la formula si riduca al rapporto: costante/(cnt_2-cnt_0).

- Qualora si desideri esprimere la velocità di rotazione in giri al minuto è sufficiente ricalcolare la costante moltiplicativa.

- Si ricorda inoltre che la differenza (cnt_2-cnt_0) dovrà essere corretta qualora siano intercorsi uno o più eventi di overflow.