# Obiettivi esercitazione:

- Implementazione filtro IIR passa basso del primo ordine

- Specifiche:
  - Frequenza di taglio $f_t$ = 10kHz
  - Attenuazione @ 100kHz = 20dB
  - Guadagno in banda = 1
  - Clock del processore = 48MHz

# Acquisizione dati:

- Utilizziamo il convertitore ADC in modalità "continuous conversion"

Tempo di campionamento:

Scegliendo ADCK = 12 MHz
Si ottiene ***$f_c$* = 12 MHz / 40 = 300 kHz**

0 1

$$\text{ConversionTime} = \underbrace{\text{SFCAdder}}_{0} + \underbrace{\text{AverageNum}}_{1} \times (\text{BCT} + \text{LSTAdder} + \text{HSCAdder}) \textbf{ = 40 cycles}$$

**Table 28-72. Base conversion time (BCT)**

| Mode | Base conversion time (BCT) |
|---|---|
| 8b single-ended | 17 ADCK cycles |
| 9b differential | 27 ADCK cycles |
| 10b single-ended | 20 ADCK cycles |
| 11b differential | 30 ADCK cycles |
| 12b single-ended | 20 ADCK cycles |
| 13b differential | 30 ADCK cycles |
| 16b single-ended | 25 ADCK cycles |
| 16b differential | 34 ADCK cycles |

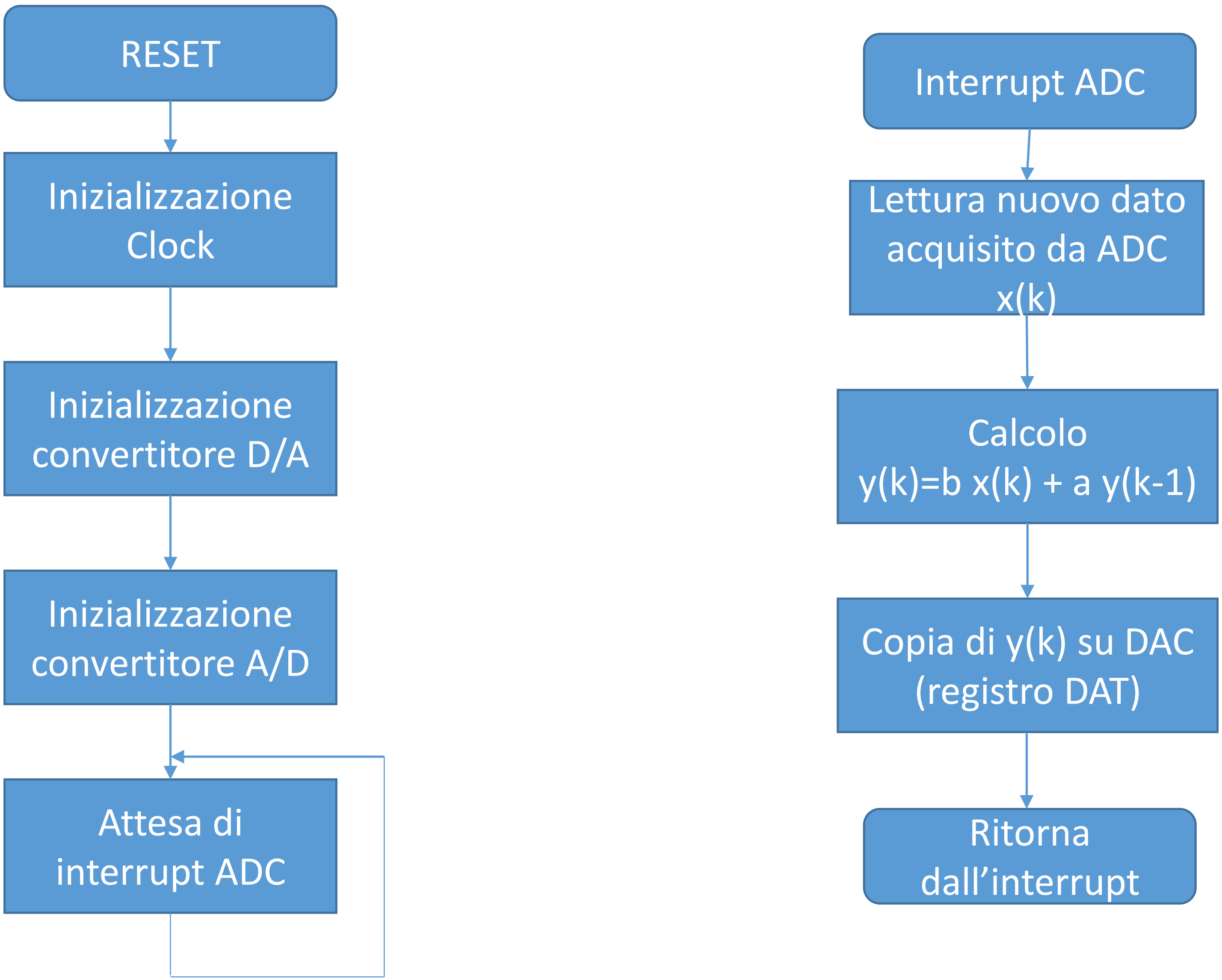
**Table 28-73. Long sample time adder (LSTAdder)**

| CFG1[ADLSMP] | CFG2[ADLSTS] | Long sample time adder (LSTAdder) |
|---|---|---|
| 0 | xx | 0 ADCK cycles |
| 1 | 00 | 20 ADCK cycles |
| 1 | 01 | 12 ADCK cycles |
| 1 | 10 | 6 ADCK cycles |
| 1 | 11 | 2 ADCK cycles |

**Table 28-74. High-speed conversion time adder (HSCAdder)**

| CFG2[ADHSC] | High-speed conversion time adder (HSCAdder) |
|---|---|
| 0 | 0 ADCK cycles |
| 1 | 2 ADCK cycles |

# Diagramma di Flusso:



STEP 1:

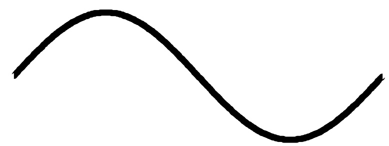
Generazione sinusoide con generatore di forme d'onda

CH1

Frequenza = 10 kHz

Offest = 1.5 V

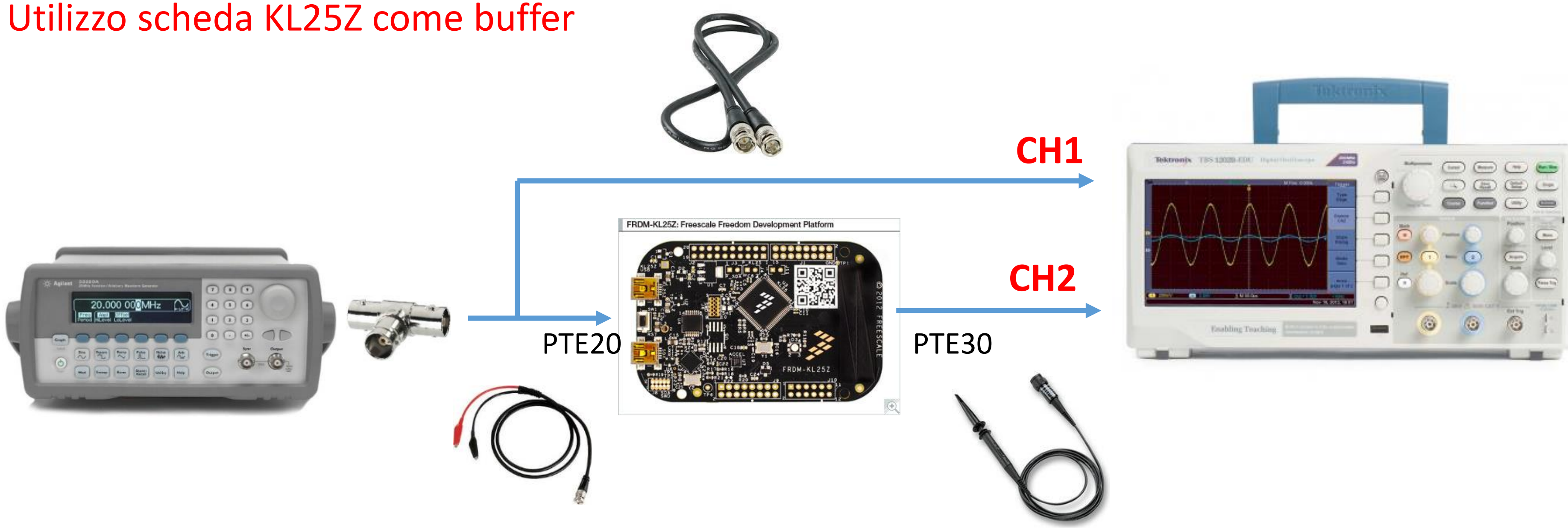
Ampiezza pico-picco = 2V



Verificare con l'oscilloscopio
i livelli di tensione

STEP 2:

Utilizzo scheda KL25Z come buffer



# Filtro IIR:

$$\boldsymbol{y(k) = b \cdot x(k) + a \cdot y(k-1)}$$

essendo il guadagno in banda unitario, si avrà: $a = 1 - b$

In tempo continuo:

$$\frac{y(s)}{x(s)} = \frac{1}{1 + s\tau} \qquad \text{con} \qquad \tau = \frac{1}{2\pi f_t}$$

Considerando l'approssimazione di Eulero si passa nel dominio Z:

$$s = \frac{1-z^{-1}}{T_c} \Rightarrow \frac{y(k)}{x(k)} = \frac{1}{1 + \frac{1-z^{-1}}{T_c}\tau}$$

Ricordando che:

$$y(k)z^{-1} = y(k-1)$$

Ricaviamo i coefficienti:

$$\boxed{\mathrm{b} = \frac{T_c}{T_c + \tau}} \qquad \boxed{\mathrm{a} = 1 - \mathrm{b} = \frac{\tau}{T_c + \tau}}$$

*Per il calcolo dei coefficienti consideriamo $f_c = 1 / T_c = 300$ kHz*

# Elaborazione di numeri frazionari:

I coefficienti *a* e *b* sono dei numeri frazionari e <1. Per la loro rappresentazione utilizzeremo la **normalizzazione all'intero**, del tutto equivalente alla **rappresentazione in virgola fissa.**

NOMALIZZAZIONE ALL'INTERO

Valori input da ADC o da precedente cacolo (possono essere trattati come interi)

$$\boldsymbol{y(k) = b \cdot x(k) + a \cdot y(k-1)}$$

Numeri frazionari

Posso moltiplicare e dividere arbitrariamente per $2^N$

$$y(k) = [b \cdot x(k) + a \cdot y(k-1)\,]\,\frac{2^N}{2^N} = \frac{[2^N b \cdot x(k) + 2^N a \cdot y(k-1)\,]}{2^N}$$

# Elaborazione di numeri frazionari:

$$y(k) = [b \cdot x(k) + a \cdot y(k-1)]\,\frac{2^N}{2^N} = \frac{[2^N b \cdot x(k) + 2^N a \cdot y(k-1)]}{2^N}$$

$\boxed{b' = 2^N b}$ $\boxed{a' = 2^N a}$

$$\boxed{y(k) = \frac{[b' \cdot x(k) + a' \cdot y(k-1)]}{2^N}}$$

- Se a≤1 e b≤1, **il valore massimo di *a'* e *b'* è pari a $2^N$**
- N è perciò scelto in base al numero di bit desiderato (N = numero di bit).

Essendo i numeri *unsigned* uso come fattore di scala $2^{16}$

<u>Esempio:</u>

**b=0.9**
**a=0.1**
**N=16**

$$b' = 2^N b = 0.9 \cdot 2^{16} = 58982.4 \approx \mathbf{58982}$$ **0xE666**

$$a' = 2^N a = 0.1 \cdot 2^{16} = 6553.6 \approx \mathbf{6554}$$ **0x199A**

Una volta eseguito il troncamento (o meglio arrotondamento) all'intero, il numero può essere facilmente convertito in binario (o in esadecimale per compattezza)

# Elaborazione di numeri frazionari:

$$y(k) = \frac{[58982 \cdot x(k) + 6554 \cdot y(k-1)\ ]}{2^N} = \frac{58982 \cdot x(k)}{2^N} + \frac{6554 \cdot y(k-1)}{2^N}$$

Esecuzione moltiplicazione:

Il prodotto tra due numeri a 16 bit (o in generale N bit) da luogo ad un risultato a 32bit (o in generale a 2N bit)

1110011001100110 x 1100000000000000 = 10101100110011001000000000000000

$b' = 58982$ **$x\ (k)$ = 49152** **$b'\ x(k)$ = 2899083264**

$$\frac{58982 \cdot x(k)}{2^N}$$ **Divisione per $2^N$ ($2^{16}$):** shift a destra di N (16) posizioni!

10101100110011001000000000000000 $\xrightarrow{/2^{16}}$ $\underbrace{1010110011001100}_{44236}$1000000000000000

**2899083264**

**$b\ x(k)$ = 0.9 x 49152 = 44236.8**

Errore dovuto alla precedente approssimazione

# Elaborazione di numeri frazionari:

In generale, potremmo avere a che fare con numeri non frazionari puri (>1) oppure molto piccoli (<<1).

*a*>1 $\Rightarrow$ $a' = 2^N a > 2^N$ **quindi non rappresentabili su N bit! Problemi di saturazione.**

*a*<<1 $\Rightarrow$ $a' = 2^N a \ll 2^N$ **errore di approssimazione elevato!**

$$y(k) = \frac{2^N b \cdot x(k)}{2^N} + \frac{2^N 2^M a \cdot y(k-1)}{2^N 2^M}$$

$$b' = 2^N b \qquad a' = 2^N 2^M a$$

*Il coefficiente **a** può essere moltiplicato per un ulteriore fattore $\mathbf{2^M}$, con M sia positivo che negativo, tale da ottenere una frazionario puro vicino all'unità*

**Fondamentale:** la divisione per il fattore di narmalizzazione (*$2^N$ o $2^N 2^M$*) deve essere fatta dopo ogni moltiplicazione.
**La somma può essere eseguita tra numeri con la stessa normalizzazione.**

# Elaborazione di numeri frazionari:

$$y(k) = \frac{b' \cdot x(k)}{2^N} + \frac{a' \cdot y(k-1)}{2^N 2^M}$$

Esempio:

**b=0.9**
**a=0.1**
**N=16**
**M=3**

$$b' = 2^N b = 0.9 \cdot 2^{16} = 58982.4 \approx \mathbf{58982}$$ **0xE666**

$$a' = 2^N 2^M a = 0.1 \cdot 2^3 \cdot 2^{16} = 52428.8 \approx \mathbf{52429}$$ **0xCCCD**

$$y(k) = \frac{\mathbf{58982} \cdot x(k)}{2^{16}} + \frac{\mathbf{52429} \cdot y(k-1)}{2^{16} 2^3}$$

Diversi **divisori** da utilizzare a seguito di ogni prodotto (prima della somma)