Analog-to-Digital Converter (ADC)

Luigi Coppolino, Giovanni Mazzeo

Outline

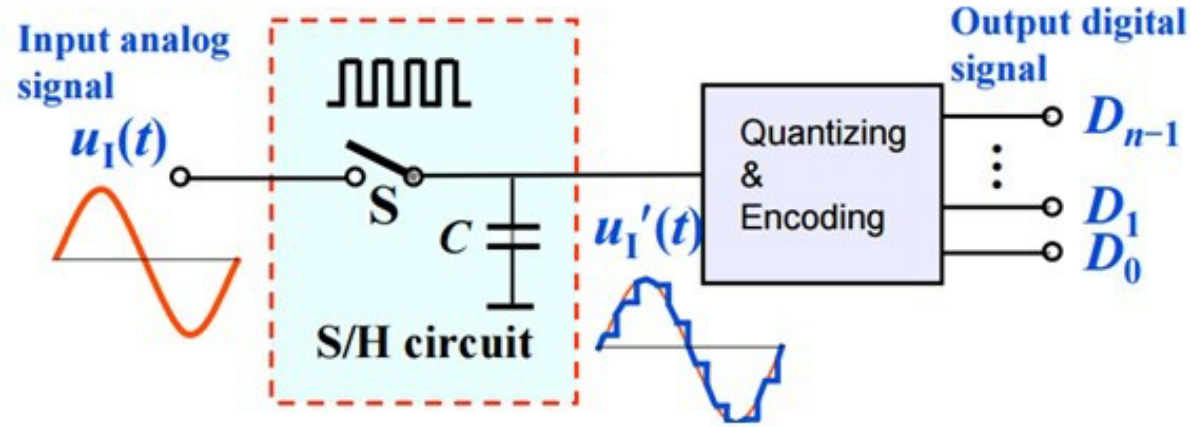
- Cos'è un ADC converter
- L'**ADC** Converter sulla STM32F3-Discovery
- Schema
- I modi di Funzionamento
- Modello di Programmazione
- Esercizio: Lettura Temperatura Interna al SoC

Analog-to-Digital Converter (ADC)

- Un **A**nalog-to-**D**igital **C**onverter (ADC) è una periferica fondamentale per ogni strumento di misura.
- Un ADC converte un segnale analogico con andamento continuo (ad es. una tensione) in una serie di valori discreti.
- Un ADC a partire da un voltaggio compreso tra 0 e Vref di uno specifico input, genera un numero compreso tra 0 e 2N-1 dove N è la sua risoluzione
- Quando si fa uso di un ADC è necessario definire la frequenza alla quale si vuole campionare il segnale di tensione in ingresso.
- Tale frequenza è chiamata **sampling rate** (frequenza di campionamento) del convertitore.

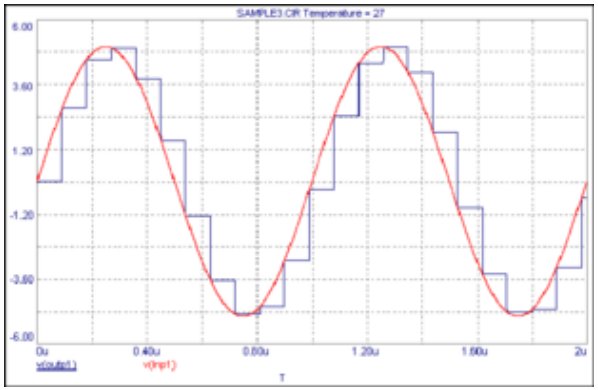
Come funziona un ADC

- Il funzionamento di un ADC si basa su due fasi:
  - S/H: Sampling and holding
  - Q/E: Quantizing and Encoding



S/H: Sampling and Holding 1/2

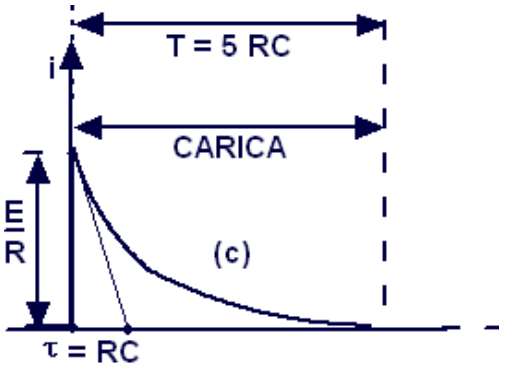
- Un segnale cambia continuamente nel tempo, per effettuare il campionamento è necessario mantenerlo costante per una certa durata di tempo in modo da generare un segnale a gradino che possa essere *quantizzato* dal Q/E
- Il *Sample and Hold circuit* quindi mantiene (hold) il segnale costante per la durata richiesta dal campionatore a digitalizzare il valore.

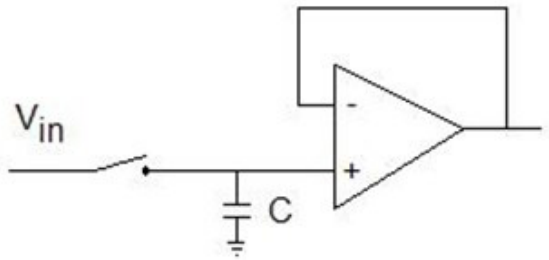


S/H: Sampling and Holding 2/2

- Il circuito che permette il funzionamento del S/H è il seguente:
  - Lo switch è normalmente mantenuto aperto. Quando si vuole effettuare la misurazione lo si chiude. Il condensatore si carica e mantiene il segnale costante per una durata data dalla sua Legge di Scarica.
- Il condensatore si scarica con legge esponenziale:

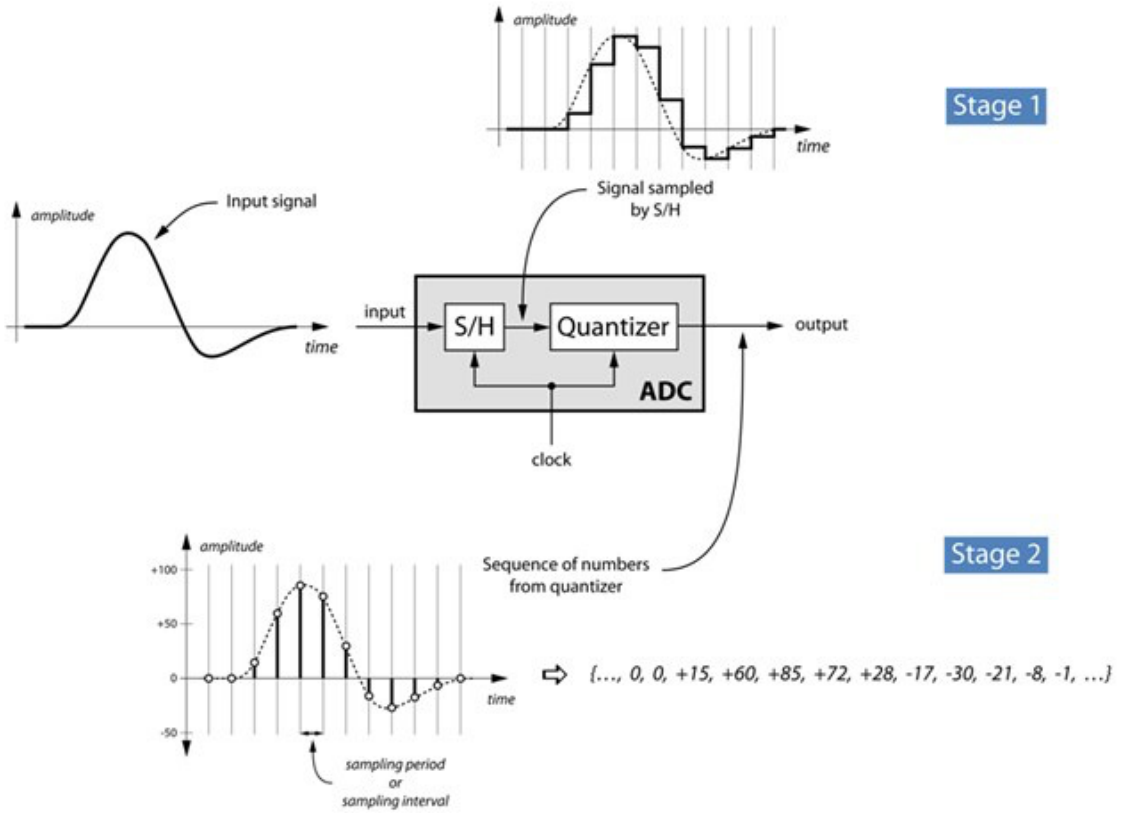
$$i = \frac{V}{R} e^{\frac{-t}{RC}}$$





Q/E: Quantizing and Encoding 1/2

- In uscita dunque dal S/H ci sarà dunque un segnale analogico a gradino che deve essere quantizzato.

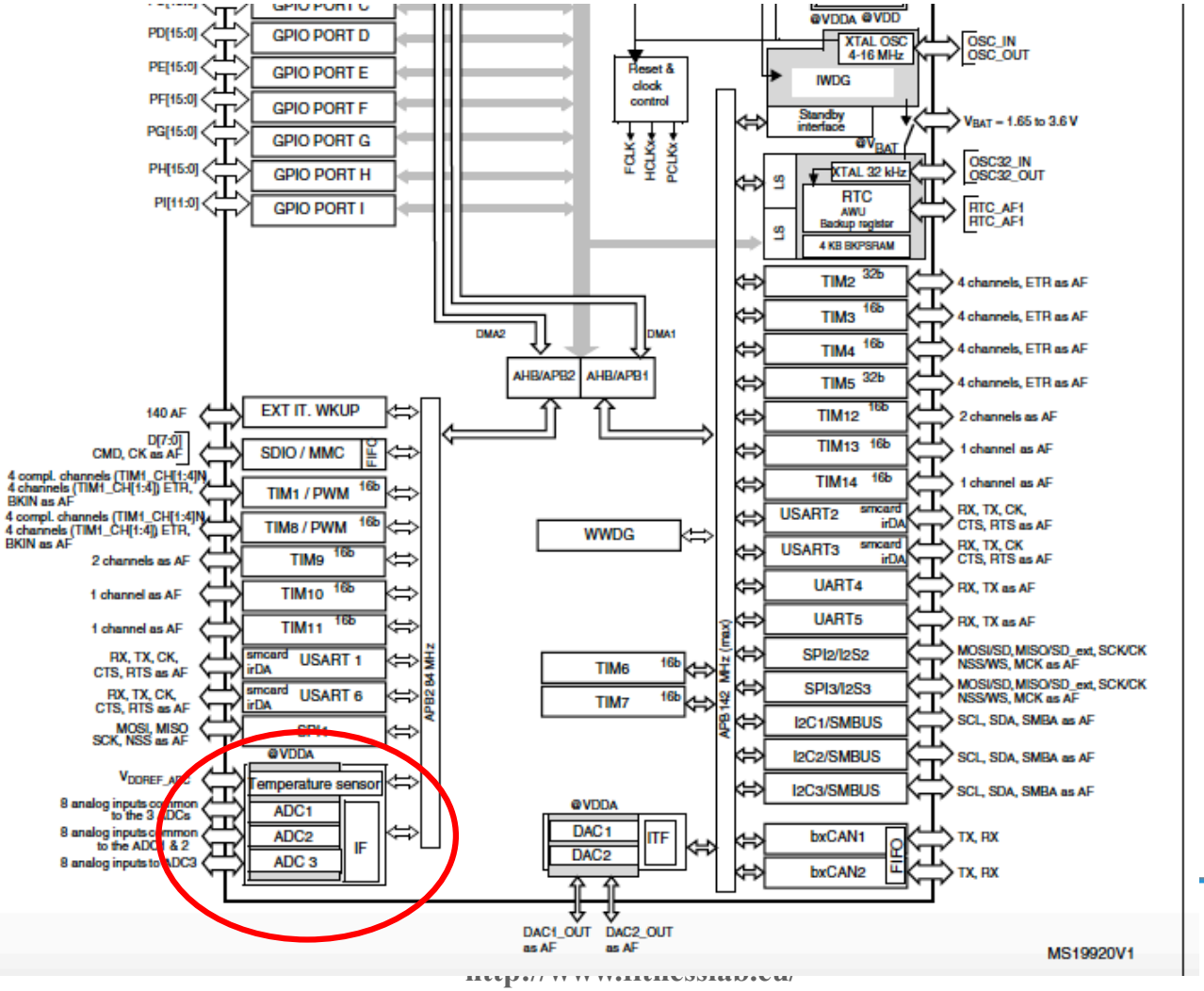


Q/E: Quantizing and Encoding 2/2

- Ad ogni livello di tensione è assegnato un valore numerico che dipende quindi dall'ampiezza del segnale in uscita dal S/H. Tale valore varia in un range dato dalla risoluzione del *Quantizer* (in potenza di 2^n).
- Dopo che tale valore numerico è stato definito sarà convertito dall *Encoder* in formato binario su n bit dove n è la risoluzione del Q/E.
- Il valore ottenuto non sarà mai veramente accurato ma sempre un approssimazione di valori del mondo reale.
- Chiaramente, maggiore è la risoluzione maggiore sarà la qualità.

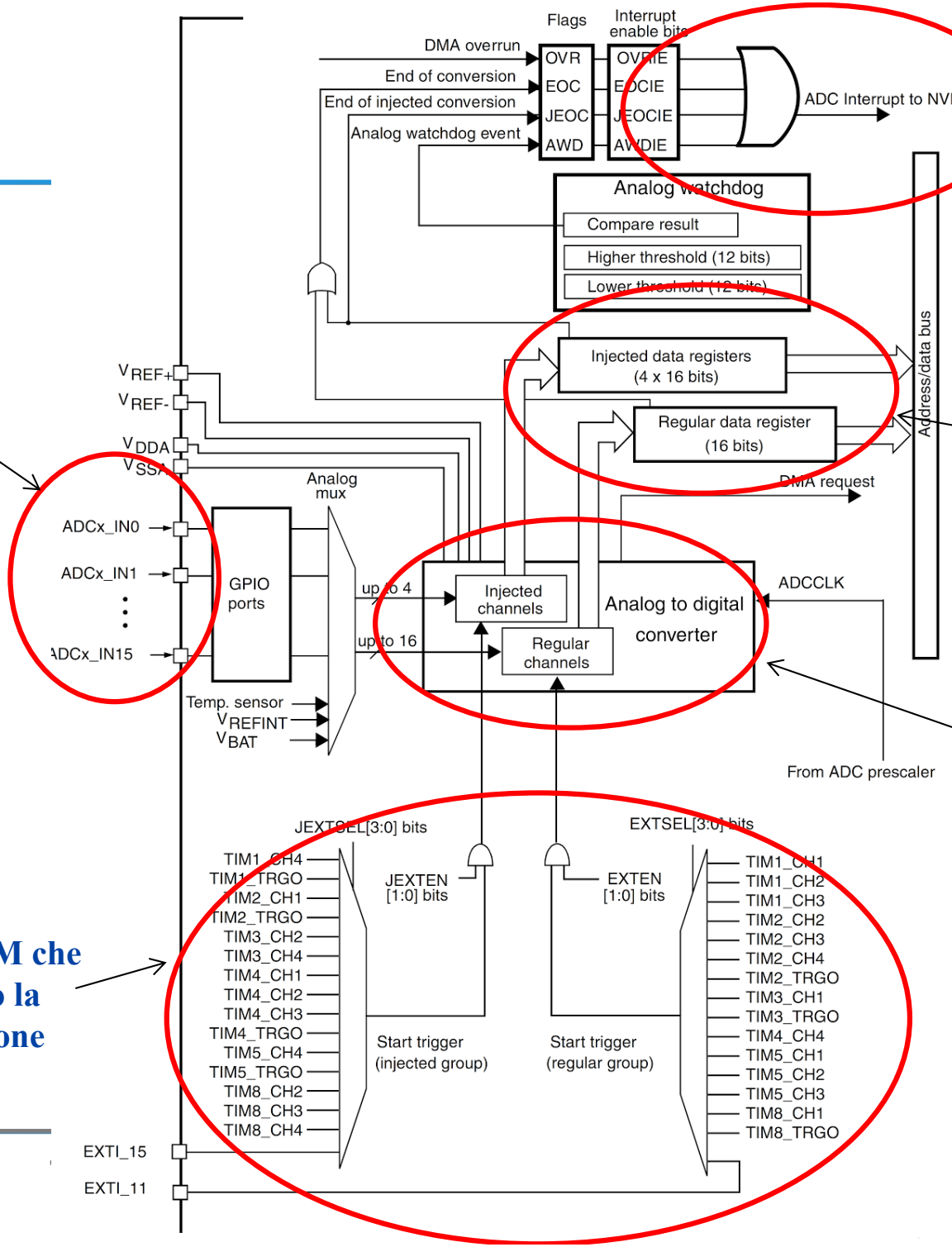
L'ADC sulla STM32F3-Discovery

- Il Microcontrollore STM32F303 installato sulla scheda STM32F3-Discovery presenta al suo interno tre ADC a 12 bit, ognuno



L'ADC sulla STM32F3-Discovery

- L'ADC presenta differenti linee di ingresso per i segnali provenienti dal mondo esterno. Tali linee di ingresso sono connesse a specifici GPIO (vedere lo user manual come fatto per i buttons) che devono essere configurati in **Analog Mode**.
- Inoltre l'ADC può essere configurato a funzionare con interrupt di modo da segnalare il processore quando una conversione è pronta.
- L'ADC può essere triggerato (ovvero abilitato ad eseguire la conversione) da timer TIMs in modo da effettuare conversioni con specifiche cadenze temporali.



La linea di interruzione che permette di segnalare che una conversione è pronta

I segnali analogici in ingresso all'ADC

i registri dove vengono mantenuti i valori delle conversioni

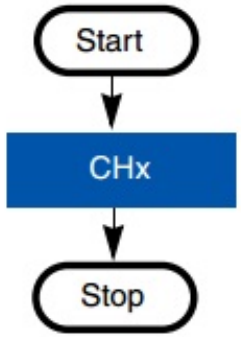
Il core dell'ADC dove ci sarenno le due unità: S/H e Q/E
Injected channels: una tiopologia di canali la cui conversione è prioritaria rispetto ai Regular channels

I Timer TIM che abilitano la conversione

ADC Modi di Funzionamento 1/4

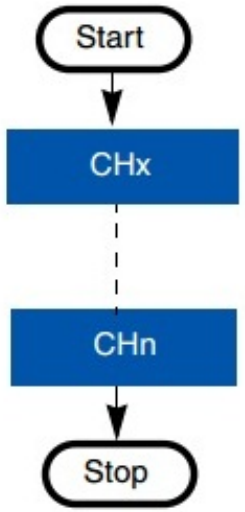
- Single Channel Single Conversion

Es. Lettura del valore di tensione di alimentazione prima dell'avvio del dispositivo



- Multiple Channel Single Conversion

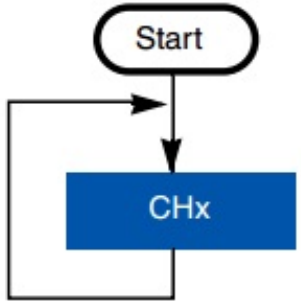
Es. Lettura del valore di posizione di ciascun asse di un braccio meccanico durante la fase di inizializzazione



ADC Modi di Funzionamento 2/4

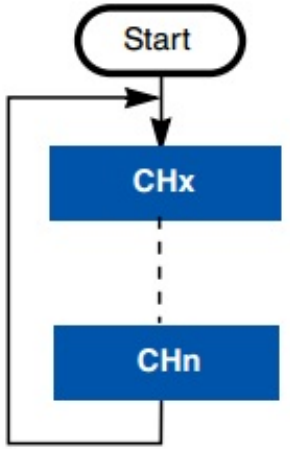
- Single Ch. Continuous Conversion

Es. Misura di una temperatura per la regolazione d
un forno



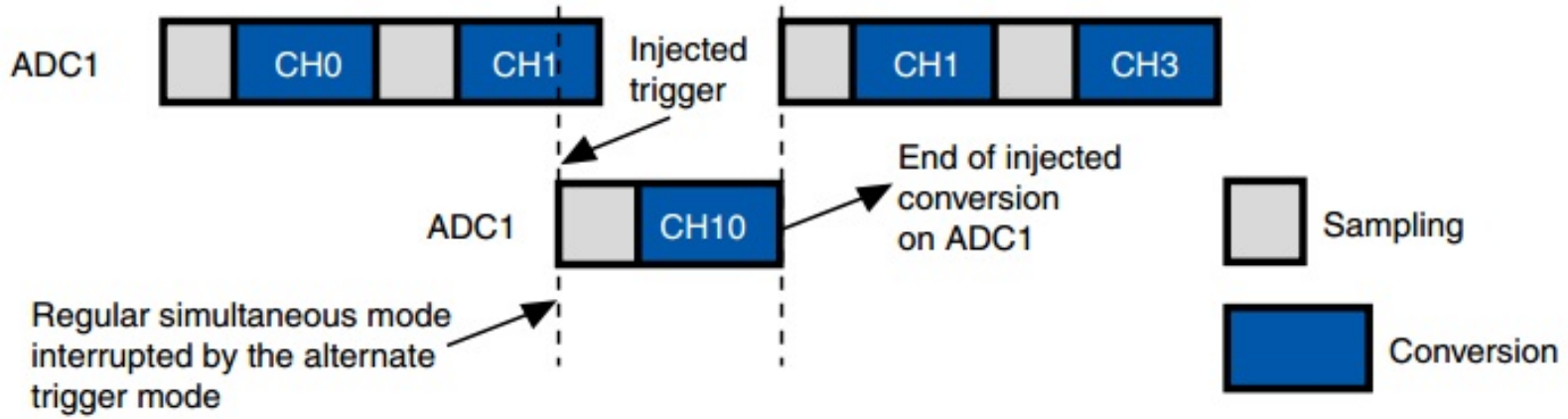
- Multiple Ch. Continuous Conversion

Es. Lettura del valore di carica e/o temperatura di ogni cella in un battery pack



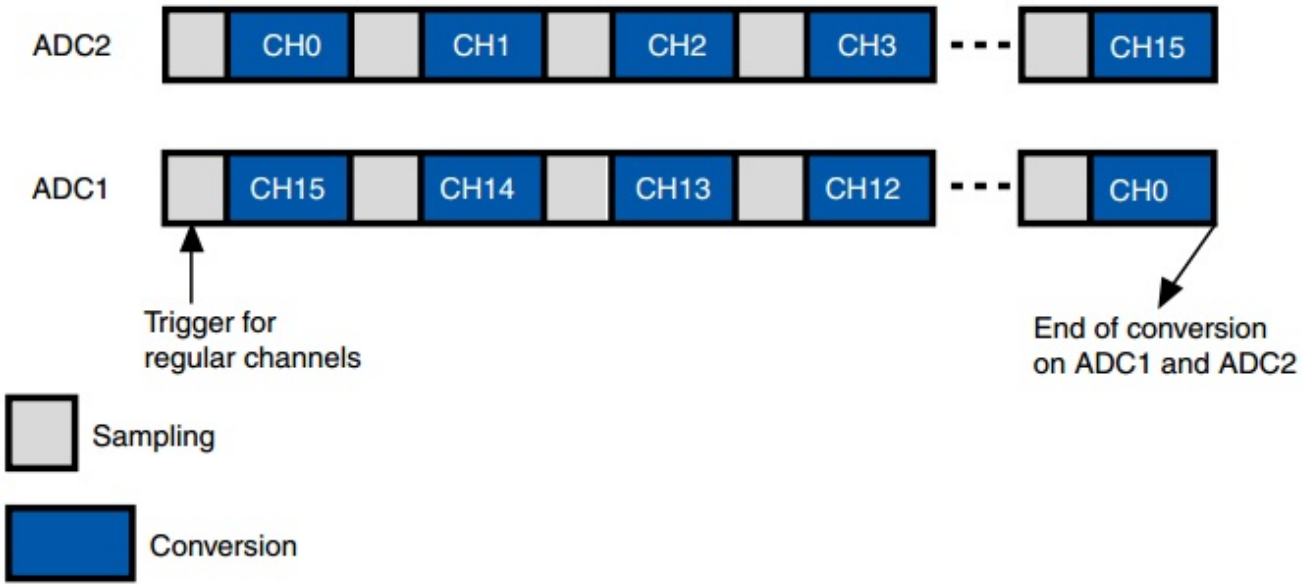
ADC Modi di Funzionamento 3/4

- Injected Conversion
  - Usata se la conversione è attivata da un evento esterno: non coinvolge la CPU
  - Ha priorità sul Regular Group: interrompe la conversione attualmente in atto su tale gruppo



ADC Modi di Funzionamento 4/4

- Dual regular simultaneous mode



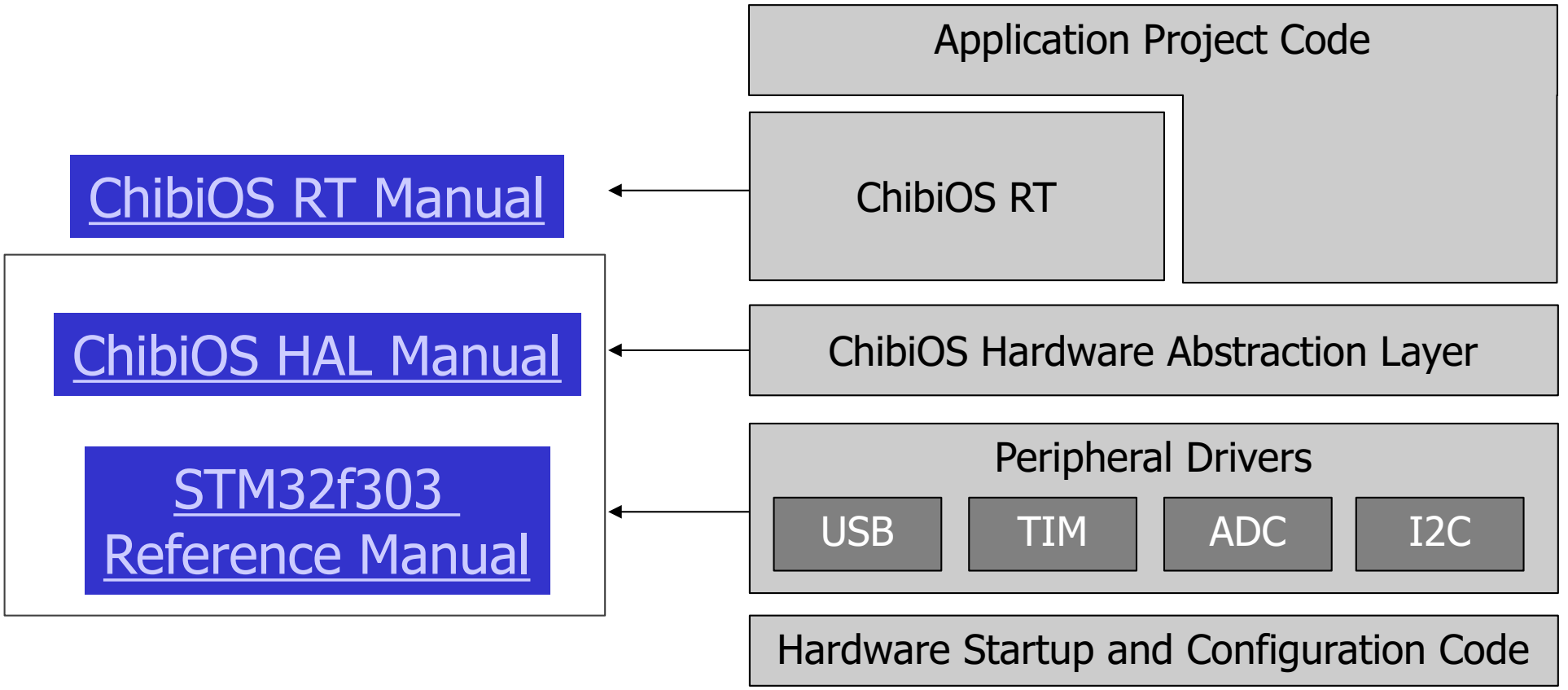
Es. Lettura di tensione e corrente per il calcolo della potenza istantanea

Modello di Programmazione

- Al solito per iniziare a programmare un ADC dobbiamo definirne il suo modello di programmazione
- Oltre a definire una delle modalità di funzionamento viste prima è necessario definire:
  - Quale ADC (ADC1, ADC2,..) e quale channel si intende usare.
  - Il GPIO associato all'ingresso della specifica linea dell'ADC e quindi configurarlo (importante impostare Analog Mode)
  - La risoluzione (12 bit ad esempio)
  - Se fare uso di un trigger esterno che abiliti l'ADC
  - Il numero di conversioni che si desidera effettuare
  - Se fare uso di interrupts per ricevere un evento quando una misura è "pronta"
  - Altro
- Vedere il reference manual della STM32F303 per maggiori info

Manuali da utilizzare

- Per lavorare sull'ADC, nel seguito utilizzeremo:



Esercizio: Leggere La Temperatura Interna della Scheda 1/3

- Vogliamo leggere la temperatura del sensore interno alla scheda facendo uso delle interrupt
- La STM32F3 presenta un sensore di temperatura interna del SoC che è connesso all'ADC. Come ogni sensore di questo tipo, darà in uscita un segnale analogico che deve essere digitalizzato.

Esercizio: Leggere La Temperatura Interna della Scheda 1/3

Reading the temperature

To use the sensor:

3. Select ADC1_IN16 or ADC1_IN18 input channel.
4. Select a sampling time greater than the minimum sampling time specified in the datasheet.
5. Set the TSVREFE bit in the ADC_CCR register to wake up the temperature sensor from power down mode
6. Start the ADC conversion by setting the SWSTART bit (or by external trigger)
7. Read the resulting V_{SENSE} data in the ADC data register
8. Calculate the temperature using the following formula:

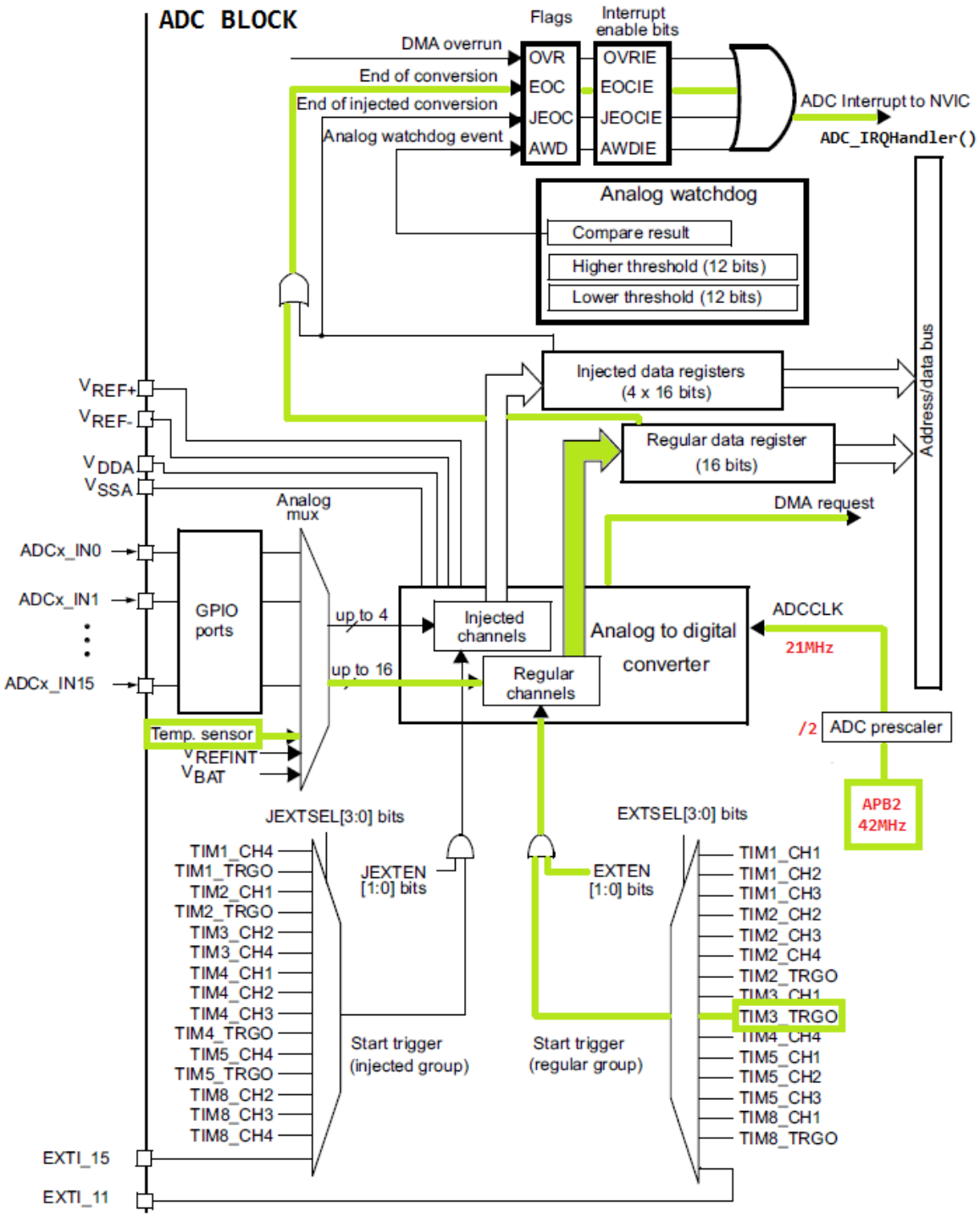
   Temperature (in °C) = {(V_{SENSE} – V_{25}) / Avg_Slope} + 25

   Where:

   – V_{25} = V_{SENSE} value for 25° C
   – Avg_Slope = average slope of the temperature vs. V_{SENSE} curve (given in mV/°C or µV/°C)

   Refer to the datasheet's electrical characteristics section for the actual values of V_{25} and Avg_Slope.

The sensor has a startup time after waking from power down mode before it can output



This is what hardware is doing in the diagram:

- Temp. sensor is ON
- ADC is ON
- TIM3 triggers at 2kHz
- ADC converts the voltage and stores it in the regular data register
- ADC makes a DMA request
- DMA moves data to memory
- ADC raises EOC flag which generates an ADC interrupt
- ADC IRQ handler is called at 2kHz toggling a pin at 1kHz (indirectly testing my calculations)

Software will do the following:

- Calibrating the sensor
- Averaging N ADC samples (VSENSE is the average value)
- Calculating the absolute temperature
- Calculating temperature variations

$$\text{Temp (in °C)} = \frac{(\text{VSENSE} - \text{V25})}{\text{Avg\_Slope}} + 25$$

(The rest is just for fun)

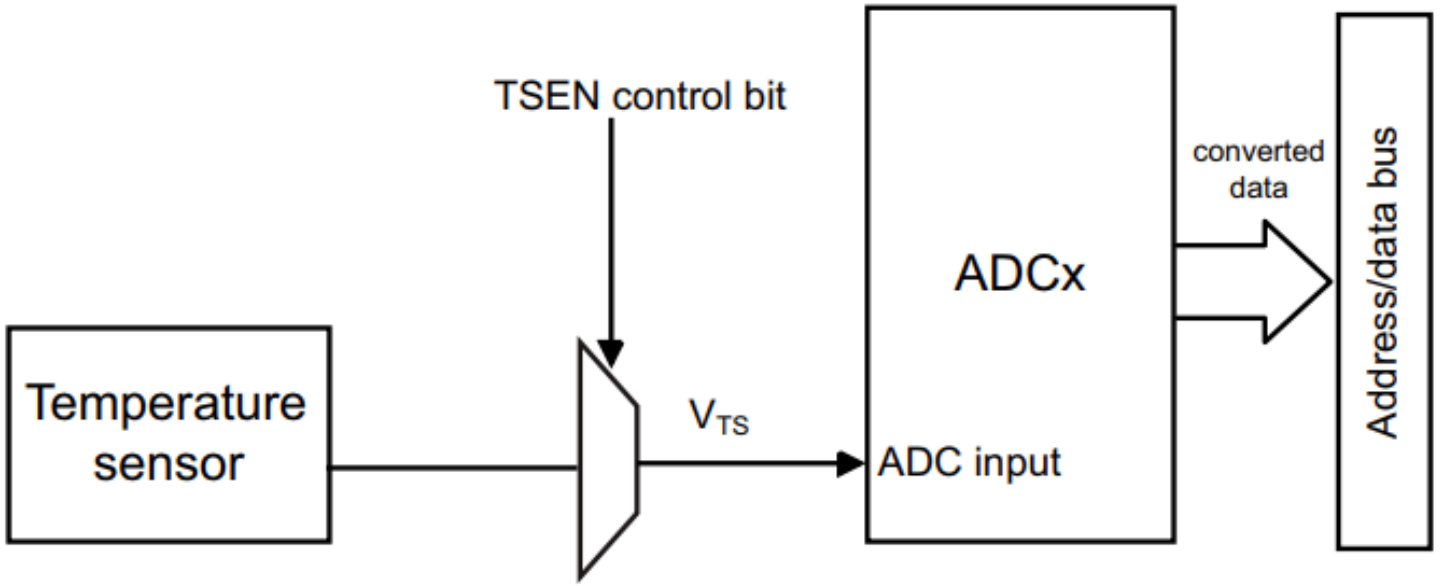
- Setting some threshold temperatures
- If the measured temp. is below/above the threshold values, some LEDs light up :)

Esercizio: Leggere La Temperatura Interna della Scheda 2/3

1. Il primo passo da compiere è quello di abilitare la periferica nel file *mcuconf.h* e *halconf.h*
2. E' necessario poi attivare la periferica attraverso: *adcStart()*
   - Quale ADC dovremo passare alla funzione di start? L'*ADC1* poichè è quello connesso (fisicamente) al sensore di temperatura
3. Fatto ciò sarà necessario abilitare il sensore di temperatura attraverso la funzione dell'hal *adcSTM32EnableTS*()
4. Dunque dovremo avviare la conversione attraverso la funzione *adcStartConversion()*
   - Tra i diversi parametri in ingresso alla funzione è necessario passare la struttura di configurazione *ADCConversionGroup*

Esercizio: Leggere La Temperatura Interna della Scheda 2/3

1. Il primo passo da compiere è quello di abilitare la periferica nel file *mcuconf.h* e *halconf.h*



adcStartConversion()

- Tra i diversi parametri in ingresso alla funzione è necessario passare la struttura di configurazione *ADCConversionGroup*

Esercizio: Leggere La Temperatura Interna della Scheda 3/3

```
 */
typedef struct {
  /**
   * @brief   Enables the circular buffer mode for the group.
   */
  bool_t                    circular;
  /**
   * @brief   Number of the analog channels belonging to the conversion group.
   */
  adc_channels_num_t        num_channels;
  /**
   * @brief   Callback function associated to the group or @p NULL.
   */
  adccallback_t             end_cb;
  /**
   * @brief   Error callback or @p NULL.
   */
  adcerrorcallback_t        error_cb;
  /* End of the mandatory fields.*/
  /**
   * @brief   ADC CFGR register initialization data.
   * @note    The bits DMAEN, DMACFG, OVRMOD, CONT are enforced internally
   *          to the driver, keep them to zero.
   */
  uint32_t                  cfgr;
  /**
```

- Se l'ADC deve operare in circular mode (o **continous**).
- Quanti "**channels**" si vogliono campionare.
- I nomi delle funzioni di **callback**

Esercizio: Leggere La Temperatura Interna della Scheda 3/3

```
   */
  uint32_t                  tr1;
  /**
   * @brief   ADC CCR register initialization data.
   * @note    The bits CKMODE, MDMA, DMACFG are enforced internally to the
   *          driver, keep them to zero.
   */
  uint32_t                  ccr;
  /**
   * @brief   ADC SMPRx registers initialization data.
   */
  uint32_t                  smpr[2];
  /**
   * @brief   ADC SQRx register initialization data.
   */
  uint32_t                  sqr[4];
#if STM32_ADC_DUAL_MODE || defined(__DOXYGEN__)
  /**
   * @brief   Slave ADC SMPRx registers initialization data.
   */
  uint32_t                  ssmpr[2];
  /**
   * @brief   Slave ADC SQRx register initialization data.
   */
  uint32_t                  ssqr[4];
#endif /* STM32_ADC_DUAL_MODE */
} ADCConversionGroup;
```

- La definizione del **Control Register (CR)**
- La definizione **SaMPle time Register (SMPR)**
- La definizione del **SeQuence Register (SQR)**, ovvero dei canali che si desidera campionare

Conversione del Valore letto dall'ADC

- Il dato letto in uscita dall'ADC è proporzionale al riferimento che si sta usando. Come visto prima sarà un valore compreso tra 0 e 2^N-1 dove N è la risoluzione dell'ADC.
- Per ottenere il valore in V bisogna moltiplicare per la tensione di riferimento Vref (nel nostro caso 3.3V) e dividere per la risoluzione.
  - Result_V = (Vref / 2^N-1)) * Result from ADC
- Una volta ottenuto il valore di tensione sarà possibile applicare la caratteristica lineare del sensore per ottenere il valore di temperatura in C°

Conversione del Valore letto dall'ADC

- Il dato l to che si sta us a 0 e 2^N-1 (
- Per otte ne di riferime risoluzi
  - Resu
- Una vo care la caratter tempera

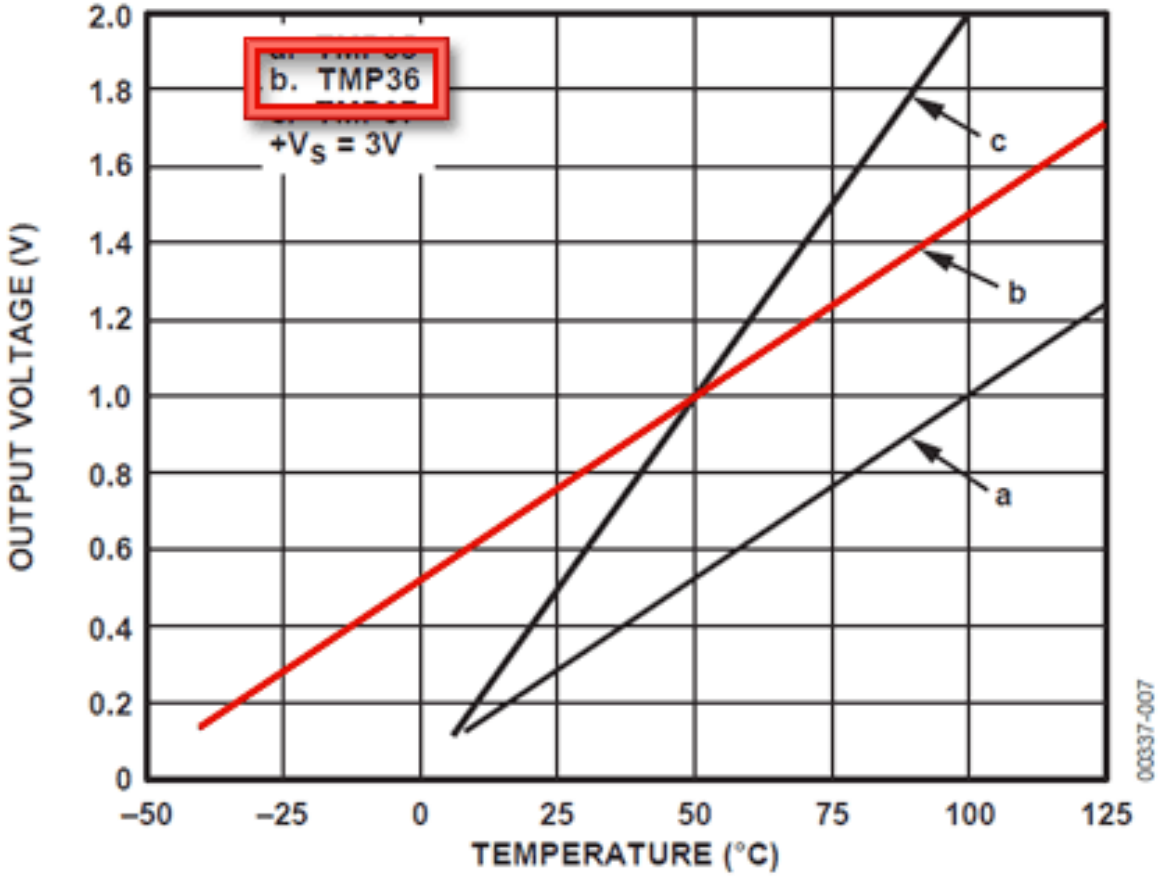


Figure 6. Output Voltage vs. Temperature