Corso di Microcontrollori e DSP

Laura Magistrale in Ingegneria Meccatronica

A.A. 2018/2019

Implementazione di un controllo PI per termo-resistenza

Obiettivi del progetto.

- Controllare la temperatura associata ad un sistema termo-resistenza.
- Utilizzare un segnale PWM per regolare la potenza dissipata sul resistore e quindi la temperatura.
- Implementare un controllo di tipo PI su microcontrollore.
- Emulare il sistema termo-resistenza con un circuito RC.
- Confrontare la risposta ad anello aperto con quella ad anello chiuso.

Per la parte teorica relativa a questa esercitazione, fare riferimento al libro di testo.

Specifiche.

- Costante di tempo della termo resistenza: 6 secondi.
- Frequenza di attraversamento: 0.4Hz.

Passo 1: Inizializzazione del progetto

Si inizializzi un nuovo progetto, scegliendo come dispositivo *target* MKL25Z128xxx4 e selezionando come componenti software da utilizzare CMSIS-CORE e Device-Startup.

Si modifichino i file ***startup_MKL25Z4*** e ***system_MKL25Z4*** in modo da operare a 48MHz (core clock), disabilitare il Watchdog e catturare l'interrupt di TPM0.

Passo 2: Inizializzazione delle periferiche

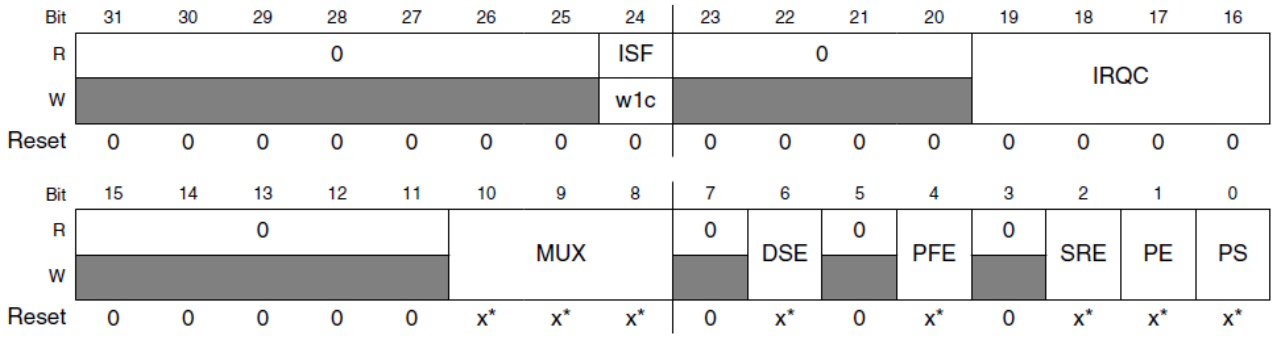
Le periferiche richieste per l'implementazione del progetto sono:

- TPM0 per la generazione del segnale PWM;
- ADC0 per la lettura della temperatura (ovvero della tensione proporzionale al valore di temperatura).
- GPIO B18 e D1 da utilizzare come segnali visivi sincronizzati con la frequenza di acquisizione.

Inizializzazione GPIO

- Abilitare il clock delle porte B e D mediante il registro SIM->SCGC5.

- Nei registri di controllo PORTD->PCR[1] e PORTB->PCR[18] impostare i bit MUX (10-8) al valore 001, in modo da selezione GPIO al pin corrispondente.



- Selezionare i pin D1 e B18 come uscite, settando ad 1 gli opportuni bit dei registri FPTD->PDDR e FPTB->PDDR, rispettivamente.

- Accendere il LED blu, azzerando l'uscita D1 mediante il registro FPTD->PCOR, e spegnere il LED rosso ponendo ad 1 l'uscita B18 mediante il registro FPTB->PSOR.

Toggle LED blu/rosso

Si implementi una routine che, agendo sul registro PTOR delle di FPTD e FPTB, permetta di invertire lo stato dei LED blu e rosso.

Inizializzazione TPM0

- Abilitare il clock della periferica mediante il registro SIM->SCGC6. Si ricorda che lo stesso registro dovrà essere anche modificato per abilitare il clock della periferica ADC0, per cui è opportuno usare l'operatore **|=**

- Configurare il registro TPM0->SC per abilitare l'**interrupt di overflow**, abilitare il conteggio impostando **CMOD** (4-3) a 01 e selezionare un **prescaler pari a 64.** Considerata la risposta nel tempo del sistema (τ = 6), sarà sufficiente utilizzare la seguente frequenza di aggiornamento del PWM (che coincide con quella di acquisizione sull'ADC):

$$f = \frac{f_{clock}}{prescaler \cdot MOD} = \frac{48MHz}{64 \cdot 2^{16}} = 11.444Hz$$

TPMx_SC field descriptions

Field	Description
31–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 DMA	DMA Enable Enables DMA transfers for the overflow flag. 0 Disables DMA transfers. 1 Enables DMA transfers.
7 TOF	Timer Overflow Flag Set by hardware when the LPTPM counter equals the value in the MOD register and increments. The TOF bit is cleared by writing a 1 to TOF bit. Writing a 0 to TOF has no effect. If another LPTPM overflow occurs between the flag setting and the flag clearing, the write operation has no effect; therefore, TOF remains set indicating another overflow has occurred. In this case a TOF interrupt request is not lost due to a delay in clearing the previous TOF.

Field	Description
	0 LPTPM counter has not overflowed. 1 LPTPM counter has overflowed.
6 TOIE	Timer Overflow Interrupt Enable Enables LPTPM overflow interrupts. 0 Disable TOF interrupts. Use software polling or DMA request. 1 Enable TOF interrupts. An interrupt is generated when TOF equals one.
5 CPWMS	Center-aligned PWM Select Selects CPWM mode. This mode configures the LPTPM to operate in up-down counting mode. This field is write protected. It can be written only when the counter is disabled. 0 LPTPM counter operates in up counting mode. 1 LPTPM counter operates in up-down counting mode.
4–3 CMOD	Clock Mode Selection Selects the LPTPM counter clock modes. When disabling the counter, this field remain set until acknolwedged in the LPTPM clock domain. 00 LPTPM counter is disabled 01 LPTPM counter increments on every LPTPM counter clock 10 LPTPM counter increments on rising edge of LPTPM_EXTCLK synchronized to the LPTPM counter clock 11 Reserved
2–0 PS	Prescale Factor Selection Selects one of 8 division factors for the clock mode selected by CMOD. This field is write protected. It can be written only when the counter is disabled. 000 Divide by 1 001 Divide by 2 010 Divide by 4 011 Divide by 8 100 Divide by 16 101 Divide by 32 110 Divide by 64 111 Divide by 128

Table 31-34. Mode, Edge, and Level Selection

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	00	00	None	Channel disabled
X	01/10/11	00	Software compare	Pin not used for LPTPM
0	00	01	Input capture	Capture on Rising Edge Only
		10		Capture on Falling Edge Only
		11		Capture on Rising or Falling Edge
	01	01	Output compare	Toggle Output on match
		10		Clear Output on match
		11		Set Output on match
	10	10	Edge-aligned PWM	High-true pulses (clear Output on match, set Output on reload)
		X1		Low-true pulses (set Output on match, clear Output on reload)
	11	10	Output compare	Pulse Output low on match
		X1		Pulse Output high on match
1	10	10	Center-aligned PWM	High-true pulses (clear Output on match-up, set Output on match-down)
		X1		Low-true pulses (set Output on match-up, clear Output on match-down)

Configurare il registro TPM0_C2SC (accessibile come **TPM0->CONTROLS[2].CnSC**) al fine di utilizzare il canale 2 del modulo come uscita **PWM edge-aligned** con funzione di **output reset on compare**.

TPMx_C*n*SC field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 CHF	Channel Flag Set by hardware when an event occurs on the channel. CHF is cleared by writing a 1 to the CHF bit. Writing a 0 to CHF has no effect. If another event occurs between the CHF sets and the write operation, the write operation has no effect; therefore, CHF remains set indicating another event has occurred. In this case a CHF interrupt request is not lost due to the delay in clearing the previous CHF. 0 No channel event has occurred. 1 A channel event has occurred.
6 CHIE	Channel Interrupt Enable Enables channel interrupts. 0 Disable channel interrupts. 1 Enable channel interrupts.
5 MSB	Channel Mode Select Used for further selections in the channel logic. Its functionality is dependent on the channel mode. When a channel is disabled, this bit will not change state until acknowledged in the LPTPM counter clock domain.
4 MSA	Channel Mode Select Used for further selections in the channel logic. Its functionality is dependent on the channel mode. When a channel is disabled, this bit will not change state until acknowledged in the LPTPM counter clock domain.
3 ELSB	Edge or Level Select The functionality of ELSB and ELSA depends on the channel mode. When a channel is disabled, this bit will not change state until acknowledged in the LPTPM counter clock domain.
2 ELSA	Edge or Level Select The functionality of ELSB and ELSA depends on the channel mode. When a channel is disabled, this bit will not change state until acknowledged in the LPTPM counter clock domain.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
0 DMA	DMA Enable Enables DMA transfers for the channel. 0 Disable DMA transfers. 1 Enable DMA transfers.

- Inizializzare il valore di *match* a 0 mediante il registro TPM0->CONTROLS[2].CnV

- All'interno del registro PORTD->PCR[2] (agendo in particolare sui bit MUX 10-8) selezionare l'**alternativa 4** che consente di collegare il pin D2 all'uscita del canale 2 del TPM0.

- L'inizializzazione del registro SIM->SOPT2 non è richiesta, poiché le impostazioni del file system_MKL25Z4 consentono già si selezionare come sorgente di clock del contatore *MCGPLLCLK* (48MHz).

Inizializzazione ADC

- Abilitare il clock alla periferica ADC0 mediante il registro SIM->SCGC6 (si consiglia di usare l'operatore |=).

- Configurare il modulo in modo da operare con **(Bus clock)/2**, conversione a **16 bit** e in modalità Long **sample time.** Il canale di ingresso e l'avvio della conversione saranno selezionati in seguito (all'interno dell'interrupt di TPM0).

ADCx_CFG1 field descriptions

Field	Description
31–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 ADLPC	Low-Power Configuration Controls the power configuration of the successive approximation converter. This optimizes power consumption when higher sample rates are not required. 0 Normal power configuration. 1 Low-power configuration. The power is reduced at the expense of maximum clock speed.
6–5 ADIV	Clock Divide Select ADIV selects the divide ratio used by the ADC to generate the internal clock ADCK. 00 The divide ratio is 1 and the clock rate is input clock. 01 The divide ratio is 2 and the clock rate is (input clock)/2. 10 The divide ratio is 4 and the clock rate is (input clock)/4. 11 The divide ratio is 8 and the clock rate is (input clock)/8.
4 ADLSMP	Sample time configuration ADLSMP selects between different sample times based on the conversion mode selected. This bit adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption if continuous conversions are enabled and high conversion rates are not required. When ADLSMP=1, the long sample time select bits, (ADLSTS[1:0]), can select the extent of the long sample time. 0 Short sample time. 1 Long sample time.
3–2 MODE	Conversion mode selection Selects the ADC resolution mode. 00 When DIFF=0:It is single-ended 8-bit conversion; when DIFF=1, it is differential 9-bit conversion with 2's complement output. 01 When DIFF=0:It is single-ended 12-bit conversion ; when DIFF=1, it is differential 13-bit conversion with 2's complement output. 10 When DIFF=0:It is single-ended 10-bit conversion ; when DIFF=1, it is differential 11-bit conversion with 2's complement output. 11 When DIFF=0:It is single-ended 16-bit conversion; when DIFF=1, it is differential 16-bit conversion with 2's complement output.
1–0 ADICLK	Input Clock Select Selects the input clock source to generate the internal clock, ADCK. Note that when the ADACK clock source is selected, it is not required to be active prior to conversion start. When it is selected and it is not active prior to a conversion start, when CFG2[ADACKEN]=0, the asynchronous clock is activated at the start of a conversion and deactivated when conversions are terminated. In this case, there is an associated clock startup delay each time the clock source is re-activated. 00 Bus clock 01 (Bus clock)/2 10 Alternate clock (ALTCLK) 11 Asynchronous clock (ADACK)

- Considerata la bassa frequenza con cui sarà richiamato l'interrupt del TPM0, è possibile utilizzare un numero elevato di medie hardware, al fine di migliorare l'accuratezza dell'ADC. Si configuri quindi il registro ADC0_SC3 al fine di abilitare le media hardware (**AVGE**=1) su 32 acquisizioni (**AVGS**=11).

ADCx_SC3 field descriptions (continued)

Field	Description
7 CAL	Calibration Begins the calibration sequence when set. This field stays set while the calibration is in progress and is cleared when the calibration sequence is completed. CALF must be checked to determine the result of the calibration sequence. Once started, the calibration routine cannot be interrupted by writes to the ADC registers or the results will be invalid and CALF will set. Setting CAL will abort any current conversion.
6 CALF	Calibration Failed Flag Displays the result of the calibration sequence. The calibration sequence will fail if SC2[ADTRG] = 1, any ADC register is written, or any stop mode is entered before the calibration sequence completes. Writing 1 to CALF clears it. 0 Calibration completed normally. 1 Calibration failed. ADC accuracy specifications are not guaranteed.
5–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 ADCO	Continuous Conversion Enable Enables continuous conversions. 0 One conversion or one set of conversions if the hardware average function is enabled, that is, AVGE=1, after initiating a conversion. 1 Continuous conversions or sets of conversions if the hardware average function is enabled, that is, AVGE=1, after initiating a conversion.
2 AVGE	Hardware Average Enable Enables the hardware average function of the ADC. 0 Hardware average function disabled. 1 Hardware average function enabled.
1–0 AVGS	Hardware Average Select Determines how many ADC conversions will be averaged to create the ADC average result. 00 4 samples averaged. 01 8 samples averaged. 10 16 samples averaged. 11 32 samples averaged.

Da riportare nella relazione:

- Codice implementato con descrizione sintetica e motivazione delle scelte eseguite nella configurazione delle periferiche

Passo 3: Implementazione con controllo a catena aperta

Per prima cosa si definisca la routine principale (main) eseguita al reset del dispositivo. Essa dovrà sequenzialmente richiamare le routine precedentemente definite per l'inizializzazione delle periferiche, per poi entrare in un loop infinito in attesa di interrupt.

In catena aperta la routine di gestione dell'interrupt può essere lasciata vuota, implementando solo l'azzeramento del flag di overflow:

TPM0->SC|= (1UL << 7)

Sarà poi necessario modificare il valore di match iniziale TPM0->CONTROLS[2].CnV, assegnando un valore pari a 0x8000 (duty cycle = 50%).

A questo punto si avvii una sessione di debug per verificare il corretto funzionamento del codice.

Da riportare nella relazione:

- Codice implementato con descrizione sintetica.

Passo 4: Implementazione emulatore analogico della termo-resistenza

Lo scopo di questa parte dell'esercitazione è quello di implementare un circuito analogico che emuli il comportamento della termo-resistenza, ovvero che abbia la stessa risposta in frequenza.

La dinamica termica di una termo-resistenza può essere espressa come:

$$\Delta T = \frac{R_{TH}}{1 + sR_{TH}C_{TH}} P_D$$

dove P_D è la potenza dissipata sul resistore:

$$P_D = \frac{V_{CC}^2}{R_S} D$$

D è il duty cycle del segnale PWM, R_{TH} = 5 °C/W e C_{TH} = 1.2 J/°C. Di conseguenza la costante di tempo del processo sarà:

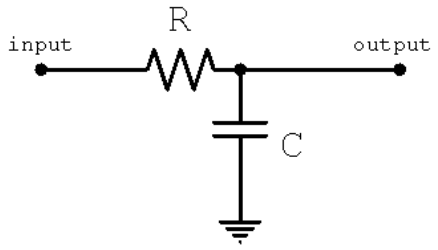
$$\tau_{TH} = R_{TH}C_{TH} = 6\ s$$

Considerando D = 1, V_{CC} = 20 V e R_S = 20 Ω, si avrà un aumento di temperatura massima pari a:

$$\Delta T_{MAX} = R_{TH} \frac{V_{CC}^2}{R_S} = 100°C$$

La temperatura dovrebbe poi essere convertita, mediante trasduttore in una tensione elettrica compatibile con il range di tensione dell'ADC (0 – 3.3V).

La risposta del sistema è perciò approssimabile con una rete RC del primo ordine:



$$V_{out} = \frac{1}{1 + sRC} V_{in}$$

Dove V_{out} rappresenta la temperatura della termo-resistenza tradotta in tensione (e perciò nel range 0 – 3.3V), e V_{in} rappresenta invece il valor medio del segnale PWM. La costante di tempo $\tau = RC$ dovrà perciò essere posta uguale a τ_{TH}.

Si utilizzi:

- R = 270kΩ;

- C = 22µF (condensatore elettrolitico);

ATTENZIONE: il condensatore elettrolitico presenta una polarità da rispettare. Si verifichi perciò che il terminale negativo sia connesso a massa.

Una volta realizzato il circuito su breadbord, si applichi all'ingresso del sistema (V_{in}) l'uscita PWM del microcontrollore (PTD2), mentre l'uscita del sistema (V_{out}) dovrà essere collegata all'ingresso dell'ADC (PTE20).

Si visualizzi sull'oscilloscopio l'uscita PWM e la risposta del sistema (V_{out}) ad anello aperto.

Suggerimenti:

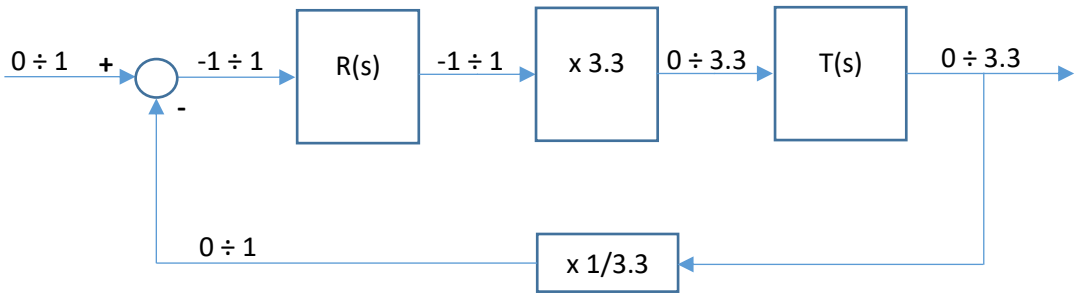
Considerate le dinamiche tipiche, si imposti l'oscilloscopio in modo da visualizzare 2.5 *secondi/divisione.*

Per osservare correttamente l'intero transitorio ci si assicuri che il condensatore sia inizialmente scarico.

Da riportare nella relazione:

- Descrizione del circuito implementato per l'emulazione del sistema.
- Forme d'onda acquisite mediante oscilloscopio (segnale PWM e V_{out}) e valutazione sperimentale di:
    1. Frequenza e Duty cycle PWM;
    2. Tempo necessario al raggiungimento del 90% del valore di V_{out} desiderato.

Passo 5: Progettazione regolatore PI



Si consideri lo schema a blocchi del sistema ad anello chiuso, dove:

$$R(s) = \frac{1}{s}k_i + k_p$$

Al fine di calcolare i coefficienti di si consideri il guadagno di anello espresso come:

$$R(s)T(s) = \frac{k_i}{s}\left(1 + s\frac{k_p}{k_i}\right)\left(\frac{1}{1+s\tau}\right)$$

e si scelgano i coefficienti kp e ki in modo da avere una banda passante pari a 0.4Hz. Nel progetto del regolatore lo studente è libero di scegliere uno dei modi studiati nei corsi precedenti. A puro titolo di esempio, si può far coincidere lo zero del PI con il polo del processo, i.e.

1) $\frac{k_p}{k_i} = \tau$
2) $|R(i2\pi f_c)T(i2\pi f_c)| = 1$ con $f_c = 0.4Hz$

Una volta calcolati i coefficienti, il regolatore numerico sarà così implementato:

$$y(k) = k_p e(k) + y_I(k)$$
$$y_I(k) = k_i' e(k) + y_I(k-1)$$

dove $k_i' = k_i T_C$, $e(k) = x_{REF} - x(k)$, e x_{REF} è il valore di riferimento.

Si utilizzi una notazione fixed point Q15 (1.15) per rappresentare i coefficienti calcolati. Nel caso in cui i coefficienti risultino maggiore di 1, si adotti un opportuno fattore di normalizzazione 2^M.

Si ricorda, che tra un decimale frazionario puro (k) e un numero binario in formato 1.15 (q) si ha la seguente corrispondenza:

$q = b_0 . a_{-1} a_{-2} a_{-3} \ldots a_{-15}$ (con b_0 bit del segno)

$k = a_{-1} 2^{-1} + a_{-2} 2^{-2} + a_{-3} 2^{-3} + \ldots + a_{-15} 2^{-15}$

quindi:

$k = (a_{-1}\,2^{-1} + a_{-2}\,2^{-2} + a_{-3}\,2^{-3} + \ldots + a_{-15}\,2^{-15})\,2^{15} / 2^{15}$

$k\ 2^{15} = (a_{-1}\,2^{-1} + a_{-2}\,2^{-2} + a_{-3}\,2^{-3} + \ldots + a_{-15}\,2^{-15})\,2^{15}$

$k\ 2^{15} = (a_{-1}\,2^{14} + a_{-2}\,2^{13} + a_{-3}\,2^{12} + \ldots + a_{-15}\,2^{0})$

Ovvero, i coefficienti a_{-1} … a_{-n} possono essere facilmente ricavati, moltiplicando k per 2^{15} ed eseguendo poi una trasformazione da decimale a binario considerando il valore decimale come un intero.

Da riportare nella relazione:

- Calcoli eseguiti per la progettazione del regolatore.
- Valori decimali e binari (in format Q15) dei coefficienti k_i' e k_p.

Passo 6: Implementazione del regolatore su microcontrollore

Innanzitutto si assegni un valore iniziale nullo a TPM0->CONTROLS[2].CnV (all'interno della routine di inizializzazione del TPM0).

Nella routine di gestione dell'interrupt (overflow TPM0) si implementi il seguente codice:

- Azzeramento del flag di overflow del TPM0

```
TPM0->SC        |= (1UL << 7);
```

- Inversione della configurazione di accensione dei LED (dal rosso si passi al blu e viceversa).

- Scrittura di una sequenza di zeri nel registro ADC0->SC1[0] al fine di iniziare una nuova conversione sul canale DADP0 in modalità single-ended.

- Attesa di fine conversione monitorando il flag di fine conversione COCO.

```
while((ADC0->SC1[0] & ADC_SC1_COCO_MASK)== 0){};
```

- Lettura del dato convertito sul registro ADC0->R[0] e salvataggio su una variabile del tipo:
  - signed int tempADC // valore misurato da ADC

- Implementazione del regolare numerico utilizzando le seguenti variabili:
  - signed int output=0; // y(k)
  - signed int error=0; // e(k)
  - signed int out_integral=0; // y_I (k)
  - signed int Tref; // x_{REF}

Le variabili così create saranno a 32bit, quindi il formato "fixed point" da considerare sarà 17.15. Tale scelta consentirà di gestire agevolmente eventuali overflow di y(k) e y_I (k).

Il codice relativo all'implementazione del regolatore sarà così composto:

1. Il dato letto dall'ADC è a 16 bit. Poiché decidiamo di lavorare in formato Q15, è necessario shiftare tempADC verso destra (di una posizione) per lasciar spazio al bit del segno (che ovviamente sarà sempre pari a 0 trattandosi di un valore sicuramente positivo).

2. Calcolo dell'errore e(k)=Tref-tempADC. Tref dovrà essere inizializzato al valore desiderato in formato Q15. Qualora si preferisse inserire il valore decimale della temperatura, è necessario introdurre l'opportuna conversione all'interno del codice.

3. Calcolo di $y_I(k)$. Si ricorda che, il risultato del prodotto tra due numeri fixed point (n1.m1 x n2.m2) sarà nel formato (n1+n2).(m1+m2). Ad esempio: 17.15 x 17.15 -> 34.30. Inoltre, l'operazione di moltiplicazione restituisce solo la parte bassa (ovvero 2.30). Per cui, al fine di tornare al formato 17.15 bisogna shiftare il risultato a destra di 15 posizioni.

4. Calcolo della componente proporzionale: K_p e(k). Nel caso in cui k_p fosse stato normalizzato per 2^M, a valle della moltiplicazione bisognerebbe tener conto della normalizzazione prima di sommare il dato con qualsiasi altro valore.
   Ad esempio, se uno dei coefficienti fosse inizialmente moltiplicato per 2^{-4} (quindi M=-4), a valle di una moltiplicazione dovremmo innanzitutto dividere per 2^{15}, shift a destra di 15 posizioni (vedi punto 3), e poi dividere ulteriormente per 2^{-4}, shift a sinistra di 4 posizioni, per rimuovere la normalizzazione iniziale. Lo shift complessivo verso destra sarà:
   15 + M = 15 - 4 = 11

5. Calcolo di y(k). Dopo avere opportunamente normalizzato la componente proporzionale (punto 5) si sommi la componente integrale $y_I(k)$ (punto 3).

6. Controllo ***anti wind-up***. Essendo y(k) limitata al range 0 – 1, è necessario implementare un sistema che tenga conto della saturazione dell'uscita. Lo studente è libero di scegliere qualunque strategia tra quelle già apprese nei corsi riguardanti il controllo dei processi. A titolo di esempio si riporta la seguente soluzione:

   Ad ogni passo si limita il valore dell'uscita y tra ymax e ymin (valori massimi e minimi che valgono in notazione Q15 32767 e 0, rispettivamente). Nel caso di saturazione dell'uscita, il valore della componente integrale viene fissato al valore calcolato al passo precedente, $y_I(k) = y_I(k) - k_i' e(k)$.

- Aggiornamento del duty cycle in base alla nuova uscita y(k).
  ATTENZIONE: il valore da scrivere su TPM0->CONTROLS[2].CnV non è esattamente y(k), ma il suo valore shiftato a sinistra di una posizione (al fine di riallineare il dato su 16 bit).

Da riportare nella relazione:

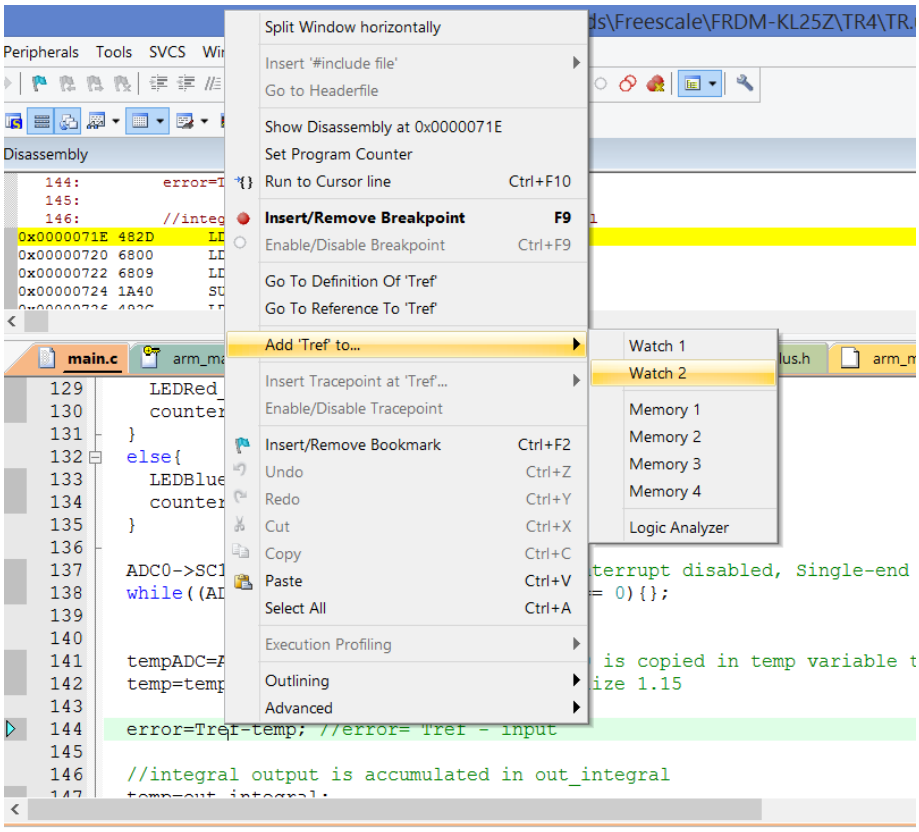
- Codice implementato con descrizione sintetica.

Passo 7: Test sperimentale del sistema regolato mediante microcontrollore

- Si esegua il codice in modalità debug facendo in modo che la temperatura di riferimento sia di 50°C. Mediante oscilloscopio si visualizzi sia l'uscita PWM che l'ingresso dell'ADC (che coincide con la

risposta del sistema controllato). Per osservare correttamente l'intero transitorio ci si assicuri che il condensatore sia inizialmente scarico (tensione in ingresso all'ADC nulla).

- Cliccando con il tasto destro sulla variabile Tref si aggiunga un ***watchpoint*** (come riportato nella figura sottostante). A questo punto, mentre il codice è in esecuzione in modalità *free running*, si modifichi la variabile Tref per passare da 50°C a 52°C e contemporaneamente si visualizzino le forme d'onda sull'oscilloscopio.



Da riportare nella relazione:

- Forme d'onda visualizzate sull'oscilloscopio durante i transitori 0-50°C e 50-52°C.
- In entrambi i casi si stimino i tempi necessari al raggiungimento della temperatura di riferimento (o ad una percentuale di essa) e si confrontino i risultati con il sistema ad anello aperto.
- Si valuti in entrambi i casi l'innesco dell'*anti wind-up* e la coerenza dei risultati con il comportamento atteso.