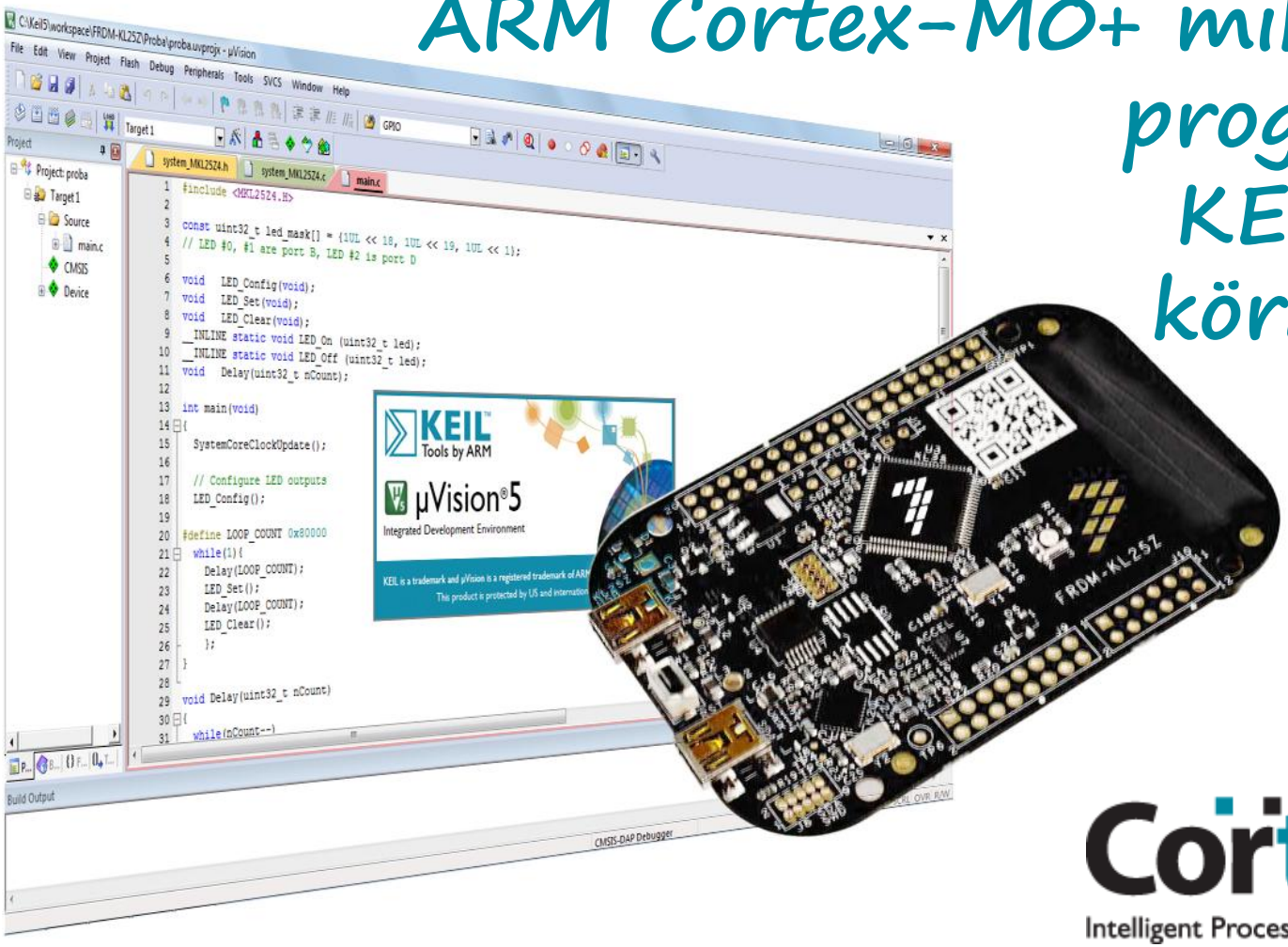
ARM Cortex-M0+ mikrovezérlő programozása KEIL MDK 5 környezetben



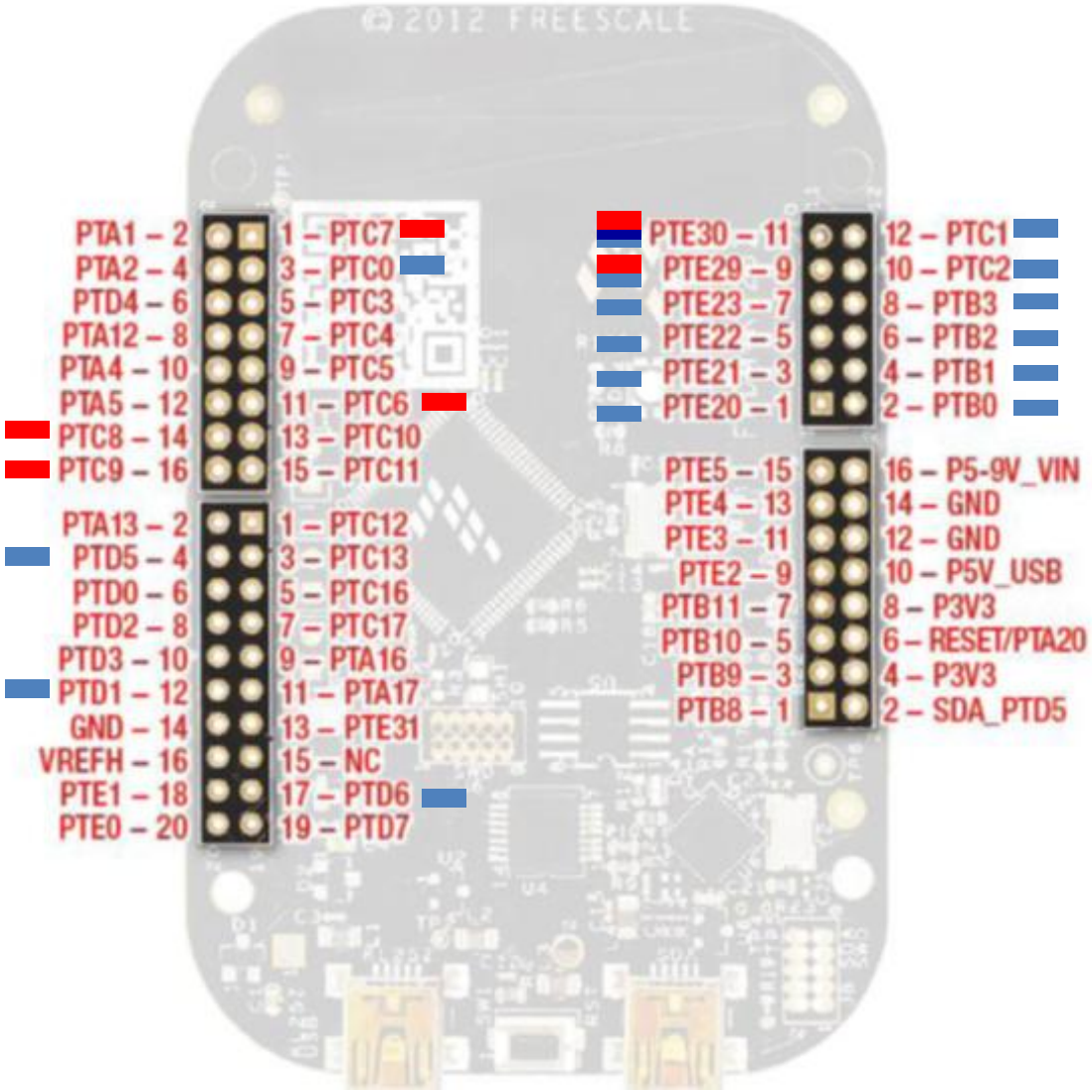
Cortex
Intelligent Processors by ARM

7. Analóg perifériák

Felhasznált anyagok, ajánlott irodalom

- Joseph Yiu: **The Definitive Guide to ARM® Cortex®-M0 and Cortex-M0+ Processors** (2nd Ed.)
- Muhammad Ali Mazidi, Shujen Chen, Sarmad Naimi, Sepehr Naimi: **Freescale ARM Cortex-M Embedded Programming**
- ARM University Program: **Course/Lab Material for Teaching Embedded Systems/MCUs** (for the **FRDM-KL25Z** board)
- ARM Information Center: **Cortex-M0+ Devices Generic User Guide**
- Freescale: **MKL25Z128VLK4 MCU datasheet**
- Freescale: **KL25 Sub-Family Reference Manual**
- Freescale: **FRDM-KL25Z User Manual**
- Freescale: **FRDM-KL25Z Pinouts**

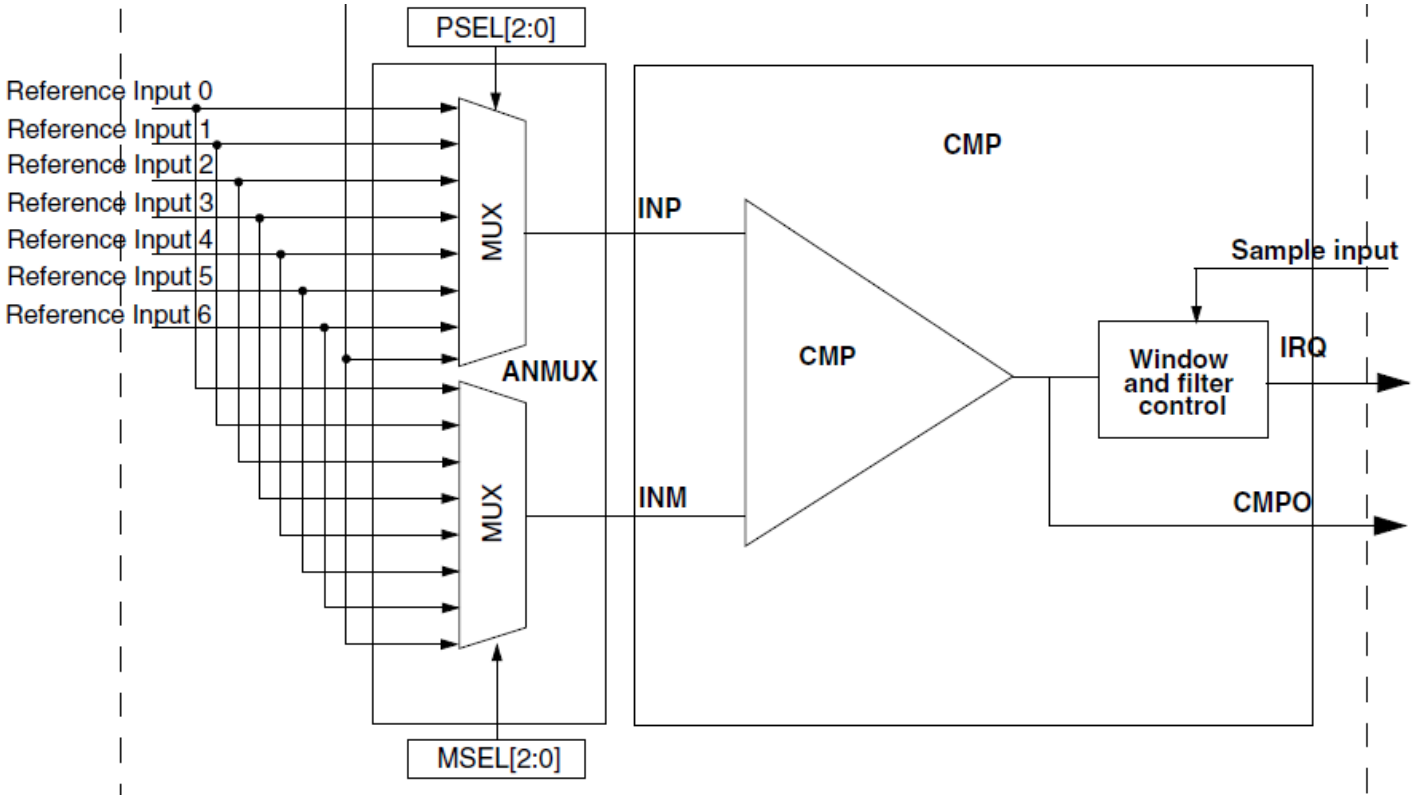
Freedom KL25Z Analóg I/O



- Bemenetek
  - 14 + 2 db. külső ADC (16-bit) csatorna
  - 6 db külső komparátor csatorna
- Kimenet
  1 db DAC (12-bit)

Analóg komparátor

Analóg komparátor



- A komparátor az INP és INM bemenetek jelét hasonlítja össze
- A CMPO kimenet jelzi, hogy INP>INM (1) vagy INP<INM (0)
- Programmegszakítást tud generálni (↑,↓, vagy mindkét élre)
- ANMUX a sok bemenő jel közül választ egyet, PSEL és MSEL bitcsoportok alapján

CMP Control Register 1 CMPO_CR1

Bit	7	6	5	4	3	2	1	0
Read Write	SE	WE	TRIGM	PMODE	INV	COS	OPE	EN
Reset	0	0	0	0	0	0	0	0

- **EN:** Modul engedélyezés (1)
- **OPE:** Kimeneti kivezetés engedélyezése
  - 1: a CMPO kimenőjelet kivezetését engedélyezi
- **PMODE:** Energiatakarékos mód választása
  - 0: Kis fogyasztású, lassú működésű mód
  - 1: Normál fogyasztású gyors üzemmód
- **SE** és **WE** a mi mikrovezérlőnkön nem használható, mindig 0 legyen!

MUX Control Register CMP0_MUXCR

Bit	7	6	5	4	3	2	1	0
Read	0	PSTM	PSEL			MSEL		
Write								
Reset	0	0	0	0	0	0	0	0

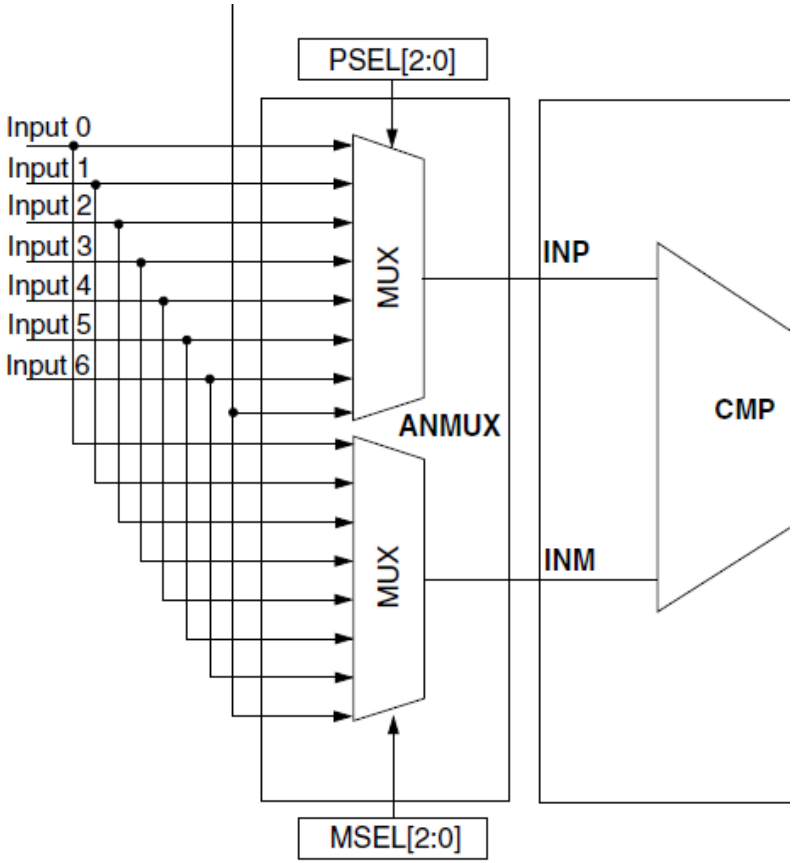
- ~~PSTM: Enable Pass Through Mode~~
  Nincs implementálva!

- **PSEL:** Neminvertáló bemenet választás
  - Kijelöli, hogy melyik bemenőjel (IN0-7) kerüljön az INP neminvertáló bemenetre

- **MSEL:** Invertáló bemenet választás
  - Kijelöli, hogy melyik bemenőjel (IN0-7) kerüljön az INM invertáló bemenetre

  IN6: 1 V-os bandgap referencia

  IN7: A belső 6-bites DAC kimenete



Választható bemenetek

Az INP és INM bemenetekhez rendelhető kivezetések, illetve belső jelforrások.

Kivezetés	Csatorna		Funkció	Megjegyzés
PTC6	000	IN0	CMP0_IN0	
PTC7	001	IN1	CMP0_IN1	
PTC8	010	IN2	CMP0_IN2	
PTC9	011	IN3	CMP0_IN3	
PTE30	100	IN4	CMP0_IN4	12-bit DAC kimenet
PTE29	101	IN5	CMP0_IN5	
	110	IN6		1V Bandgap referencia
	111	IN7		belső 6-bites DAC

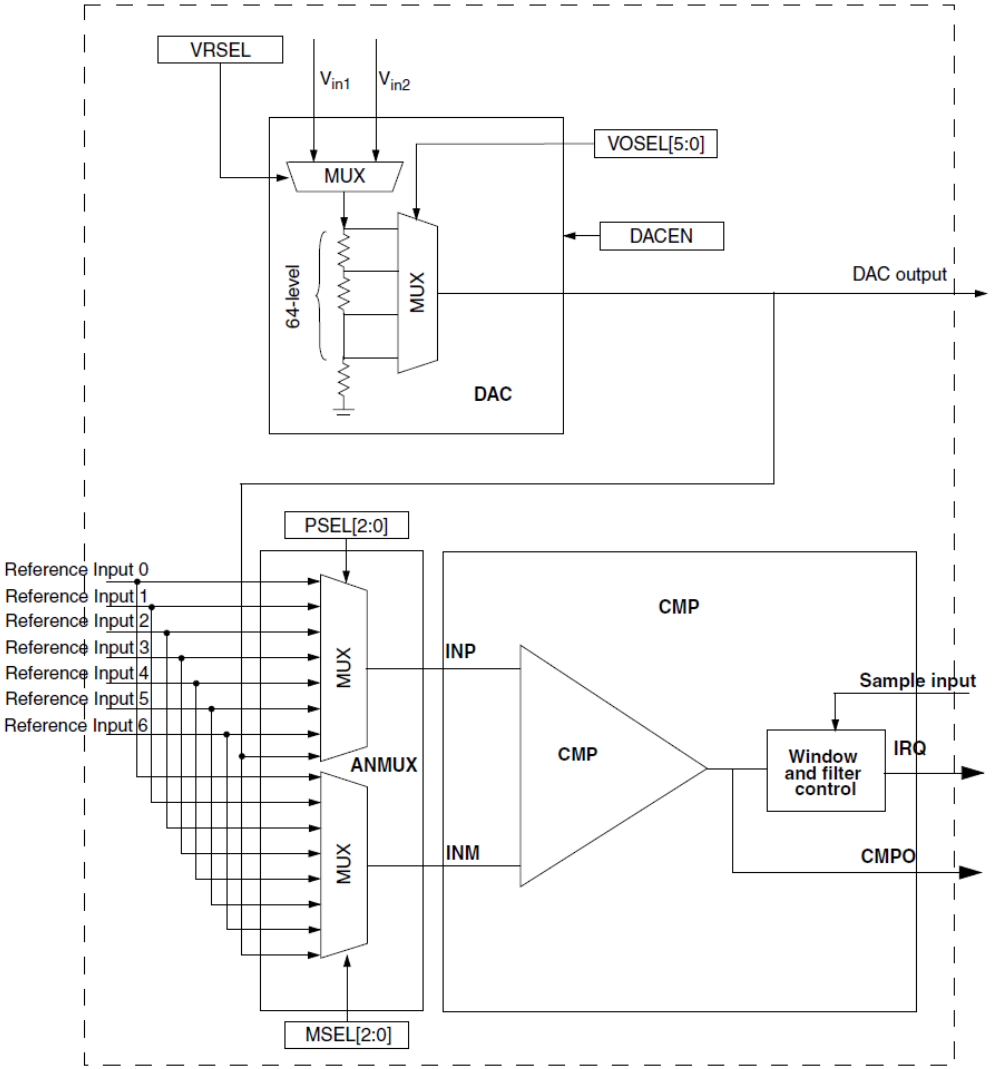
Komparátor megszakítások

Bit	7	6	5	4	3	2	1	0
Read	0	DMAEN	0	IER	IEF	CFR	CFF	COUT
Write						w1c	w1c	
Reset	0	0	0	0	0	0	0	0

- **CMP0_SCR: Állapotjelző és vezérlő regiszter**
  - **COUT:** A komparátor kimenetének pillanatnyi állapota
  - CFR: Felfutó él jelzése. Jelzi, amikor COUT ’1’-be billen. Törlése 1 beírással.
  - CFF: Lefutó él jelzése. Jelzi, amikor COUT ’0’-ba billen. Törlése 1 beírással.
  - **IER:** Programmegszakítás engedélyezése, amikor CFR ’1’-be billen.
  - **IEF:** Programmegszakítás engedélyezése, amikor CFR ’0’-ba billen.

- Megszakítás keletkezik, amikor az engedélyezésnek megfelelő esemény bekövetkezik (fel-, vagy lefutó él, vagy mindkettő)
  - A megszakítást kiszolgáló eljárás CMSIS neve: **CMP0_IRQHandler**

Komparátor DAC



- A 6-bites DAC segítségével beállítható a komparátor referencia szintje.

DAC vezérlő regiszter CMP0_DACCR

Bit	7	6	5	4	3	2	1	0
Read Write	DACEN	VRSEL	VOSEL					
Reset	0	0	0	0	0	0	0	0

- **DACEN:** A 6-bites DAC engedélyezése (1)
- **VRSEL:** DAC referencia feszültség választás
  - 0: VREFH
  - 1: VDD
- **VOSEL:** DAC kimenő feszültség beállítása
  - $V_{DACO} = (VOSEL+1)*(V_{in}/64)$
  - $VOSEL = 64*(V_{DACO} /V_{in}) - 1$

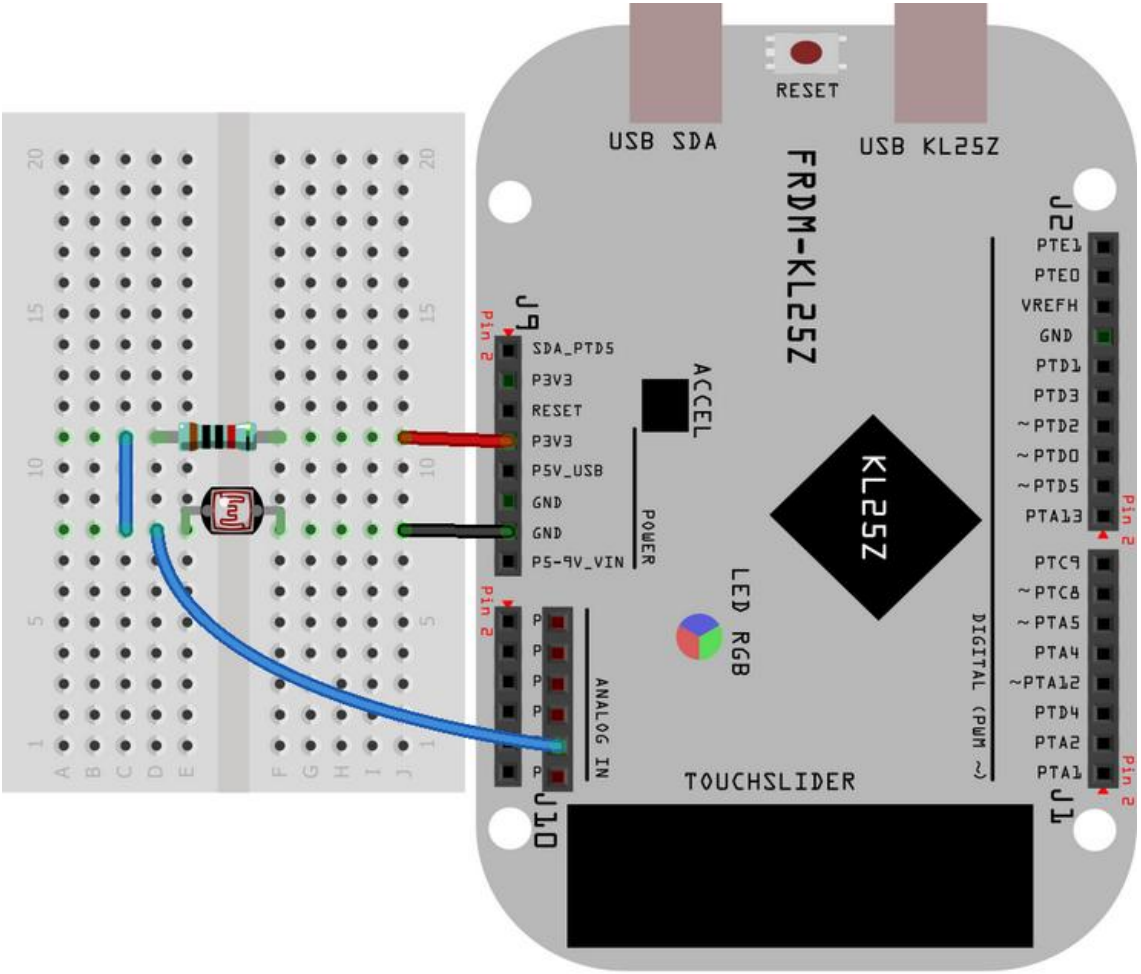
Comparator_1: A komparátor egyszerű használata

A **PTE29** bemeneten figyeljük, hogy a jel nagyobb-e mint 1 V.

A fel-, illetve lefutó élek detektálásakor a zöld LED-et be-, illetve kikapcsoljuk.

A fényérzékeny osztó egy 10 kΩ-os ellenállásból és egy CdS fényérzékeny ellenállásból áll.

Sötétedéskor, vagy árnyékoláskor a zöld LED kigyullad, jó megvilágítás esetén pedig elalszik.



Made with Fritzing.org

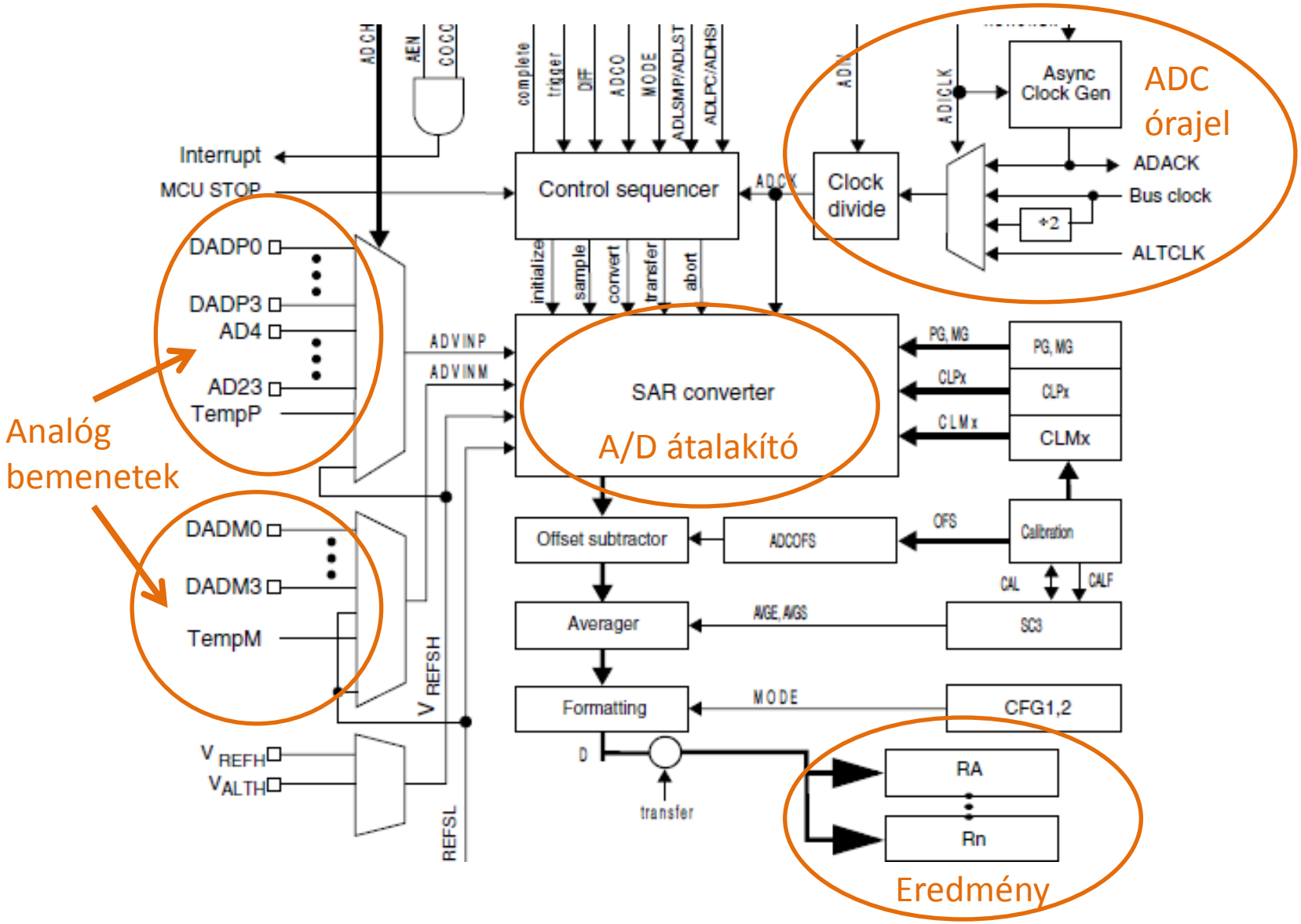
Comparator_1/main.c

```c
#include "MKL25Z4.h"
int main(void) {
    SIM->SCGC5 |= 0x400;          // PORTB engedélyezése (zöld LED)
    PORTB->PCR[19] = 0x100;       // PTB19 legyen GPIO módban
    PTB->PDDR |= 0x80000;         // PTB19 legyen kimenet
    PTB->PSOR = 0x80000;          // LED lekapcsolása
    SIM->SCGC4 |=SIM_SCGC4_CMP_MASK; // A komparátor engedélyezése
    CMP0->CR0   = 0x00;           // Filter és hiszterézis letiltva
    CMP0->CR1   = 0x17;           // Folyamatos mód, gyors mód, nincs kimenet
    CMP0->FPR   = 0x00;           // Szűrés letiltása
    CMP0->SCR   = 0x06;           // Megszakítás letiltása, jelzőbitek törlése
    CMP0->DACCR = 0xD3;           // DAC engedélyezés és 1V-ra állítás
    SIM->SCGC5 |= SIM_SCGC5_PORTE_MASK; // PORTE engedélyezése
    PORTE->PCR[29] = 0x000;       // PTE29 legyen analóg módban
    CMP0->MUXCR = 0x2F;           // INP: IN5 (PTE29), INM: IN7 (DAC)
    while(1) {
    //--- Felfutó él vizsgálata ---
      if((CMP0->SCR & CMP_SCR_CFR_MASK)==CMP_SCR_CFR_MASK) {
        CMP0->SCR |= CMP_SCR_CFR_MASK;                      // Jelzőbit törlése
        PTB->PCOR = 0x80000;                                // LED felkapcsolása
      }
    //--- Lefutó él vizsgálata ---
      if((CMP0->SCR & CMP_SCR_CFF_MASK)==CMP_SCR_CFF_MASK) {
        CMP0->SCR |= CMP_SCR_CFF_MASK;                      // Jelzőbit törlése
        PTB->PSOR = 0x80000;                                // LED lekapcsolása
      }
      delayMs(5);
    }
}
```

Analóg-digitális átalakító

16-bites SAR

ADC blokkvázlat



ADC jellemzők

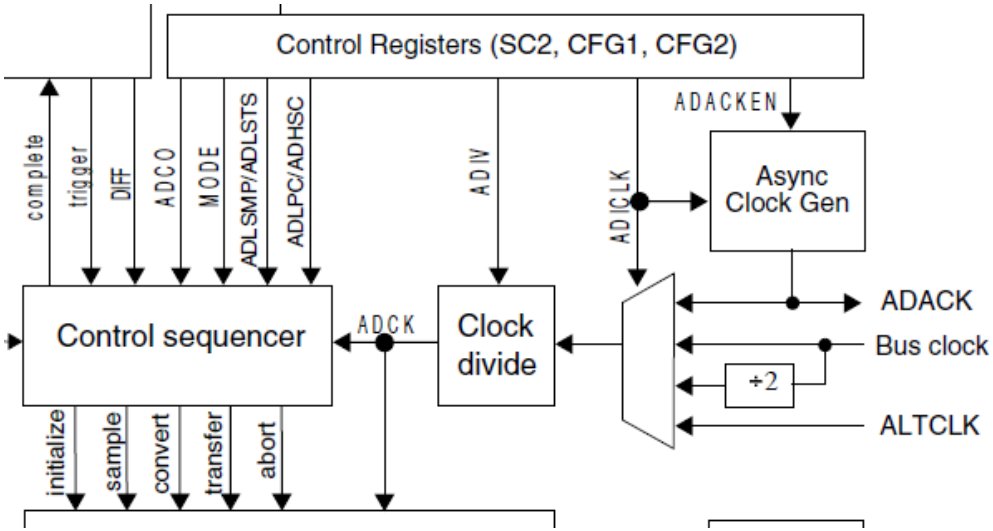
- Fokozatos megközelítést (SAR) használ az átalakításhoz
- Többféle felbontás használható: 16, 13, 12, 11, 10, 9, és 8 bites
- Aszimmetrikus és differenciális bemenetek használata
- Előjeles vagy előjel nélküli eredmény
- Akár 24 analóg bemenet (single-ended), és 4 pár differenciális
- Automatikus összhasonlítás és megszakítás (szint és tartomány)
- Hardveres átlagolás
- Beépített hőmérő
- 1V Bandgap referencia

Az ADC használata

- **ADC konfigurálás**
  - Órajel konfigurálás
  - Referencia feszültség választása
  - Indítójel forrásának kiválasztása
  - Bemeneti csatorna kiválasztása
  - További beállítások (mintavételi idő, hardver átlagolás)
- **A konverzió indítása**
- **Az eredmény kiolvasása**

Az órajel beállítása

- Órajel forrás választása ADICLK
  - Busz órajel, vagy a fele
  - ADACK: helyi órajel, az ADC akkor is működhet, ha a CPU stop módban van
  - ALTCLK: alternatív órajel (MCU-specifikus)



- Az órajel leosztása 2^{ADIV} arányban, ez lesz az ADCK órajel
- ADCK frekvenciája adott határok között legyen a pontosság biztosítására (lásd: KL25 Subfamily datasheet)
  - 1 – 18 MHz (<= 13-bites módban)
  - 2 – 12 MHz (16-bites módban)

Órajel konfiguráló regiszterek

- ADC0_CFG1
  - ADIV: divide clock by 2^{ADIV}
    - 00: 1
    - 01: 2
    - 10: 4
    - 11: 8
  - ADICLK: Input clock select
    - 00: Bus clock
    - 01: Bus clock/2
    - 10: ALTCLK
    - 11: ADACK
- ADC0_CFG2
  - ADACKEN: Enable asynchronous clock

7	6	5	4	3	2	1	0
ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0	0	0	0	0	0	0	0

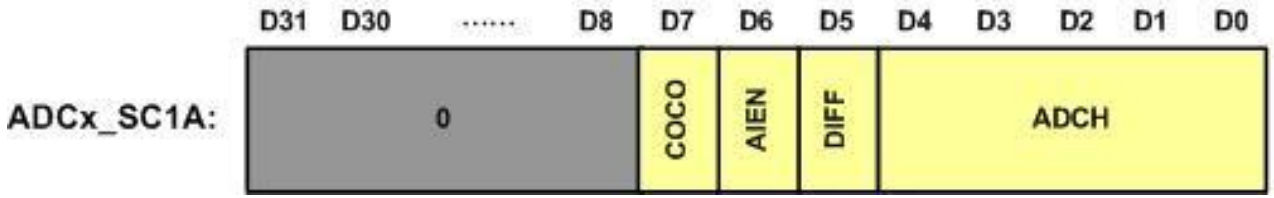
1

Felbontás

Mintavételezési idő nyújtása:
00: 20, **01:** 12, **10:** 6, **11:** 2 ADCK ciklus

4	3	2	1	0
MUXSEL	ADACKEN	ADHSC	ADLSTS	
0	0	0	0	0

ADC0_SC1A állapotjelző és konfigurációs regiszter



- **COCO** - Konverzió vége jelzőbit (csak olvasható, s automatikusan törlődik az eredmény kiolvasásakor, vagy az ADC0_SC1A regiszter újraírásakor)
- **AIEN** - Programmegszakítás engedélyezése
- **DIFF** - Differenciális mód engedélyezése (0: aszimmetrikus mód, 1: differenciális mód)
- **ADCH** - Mérendő csatorna sorszáma

ADC0_SC2 állapotjelző és konfigurációs regiszter

	D31	D30	.....	D8	D7	D6	D5	D4	D3	D2	D1	D0
ADCx_SC2:	0				ADACT	ADTRG	ACFE	ACFGT	ACREN	DMAEN	REFSEL	

ADACT - Adatkonverzió aktív (csak olvasható jelzőbit)

ADTRG - ADC trigger mód (0: szoftveres triggerelés, 1: hardveres triggerelés)

ACFE - Összehasonlítás mód engedélyezés

ACFGT - Összehasonlítási mód: (0: kisebb mint, 1: nagyobb vagy egyenlő)

ACREN - Összehasonlítási tartomány engedélyezés (0: tiltás, csak a CV1 szint számít, 1: engedélyezés: CV1, és CV2 is számít)

DMAEN - DMA átvitel engedélyezés (0: letiltva, 1: engedélyezve), amikor a konverzió befejeződik.

REFSEL - Referencia választása (00: a **VREFH, VREFL** lábakra kötött feszültség, 01: a **VDDA, VSSA** lábakra kötött feszültség a referencia, a többi beállítás érvénytelen)

Program7_1: mérés a 0. csatornában

A program a 0. analóg csatornán (**PTE20** kivezetés) mér, szoftveres indítással. Az eredmény 8-10. bitjeit az RGB LED segítségével. A LED vezérlő kódot a **Program2_7**-ből emeltük át. Ha a **PTE20** lábra kapcsolt feszültség változik, a LED színe is megváltozik.

```
#include "MKL25Z4.h"
int main (void) {
    int result;
    LED_init();                     // RGB LED konfigurálás
    ADC0_init();                    // ADC0 konfigurálás
    while (1) {
        ADC0->SC1[0] = 0;           // ADC konverzió indítása
        while(!(ADC0->SC1[0] & 0x80)); // Konverzió végére várunk
        result = ADC0->R[0];        // Eredmény kiolvasása, COCO bit törlése
        LED_set(result >> 7);       // Az eredmény kijelzése
    }
}

void ADC0_init(void) {
    SIM->SCGC5 |= 0x2000;           // PORTE engedélyezése
    PORTE->PCR[20] = 0;             // PTE20 analóg bemenet legyen
    SIM->SCGC6 |= 0x8000000;        // ADC0 engedélyezése
    ADC0->SC2 &= ~0x40;             // szoftveres triggerelés
    // clock divide by 4, long sample time, single ended 12 bit, bus clock
    ADC0->CFG1 = 0x40 | 0x10 | 0x04 | 0x00;
}
```

LED_init() és LED_set() kódját lásd a korábbi mintaprogramokban!

Program7_2: A beépített hőmérő használata

A **beépített hőmérőt** használjuk, melynek jele a **26. ADC csatorna** kiválasztásával érhető el.

$$\mathbf{Temp = 25 - (V_{Temp} - V_{25})/m}$$

ahol $\mathbf{V_{Temp}}$ a mért feszültség mV-ban, $\mathbf{V_{25}}$ és **m** pedig az adatlapból kiolvashatók:
$\mathbf{V_{25}}$ a 25 ºC-on mért feszültség, kb. 716 mV, **m** pedig a meredekség, kb. 1620 µV/ºC.

Hardveres átlagolás:

Az **ADC0_SC3** regiszterben engedélyezhetjük és konfigurálhatjuk a hardveres átlagolást.
AVGE – átlagolás engedélyezés (0. tilt, 1: engedélyez)
AVGS – Átlagolt minták száma (00: 4 minta , 01: 8 minta , 10: 16 minta, 11: 32 minta)

```
ADC0->SC3 = ADC_SC3_AVGE_MASK    // Hardveres átlagolás engedélyezése
          | ADC_SC3_AVGS(3);     // 32 minta átlagolása
```

Eredmények kiíratása

A mérés eredményét a **Lab04/uart_retarget** mintapéldában bemutatott módon, az UART0 soros porton keresztül íratjuk ki 9600 bps-sel. Rendszer órajel: CPU 48 MHz, busz 24 MHz.

Felbontás beállítása

ADC0_CFG1 regiszter **MODE** bitcsoportja: **00:** 8-bit, **01:** 12-bit, **10:** 10-bit, **11:** 16-bit

Ezt használjuk

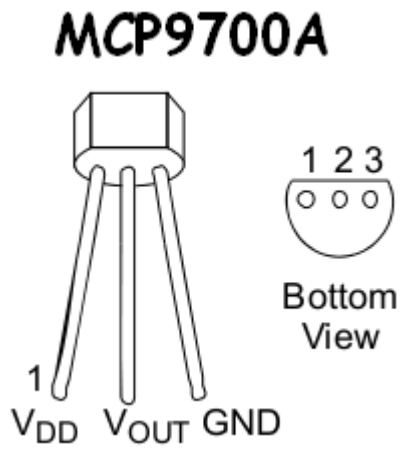
Program7_2 listája

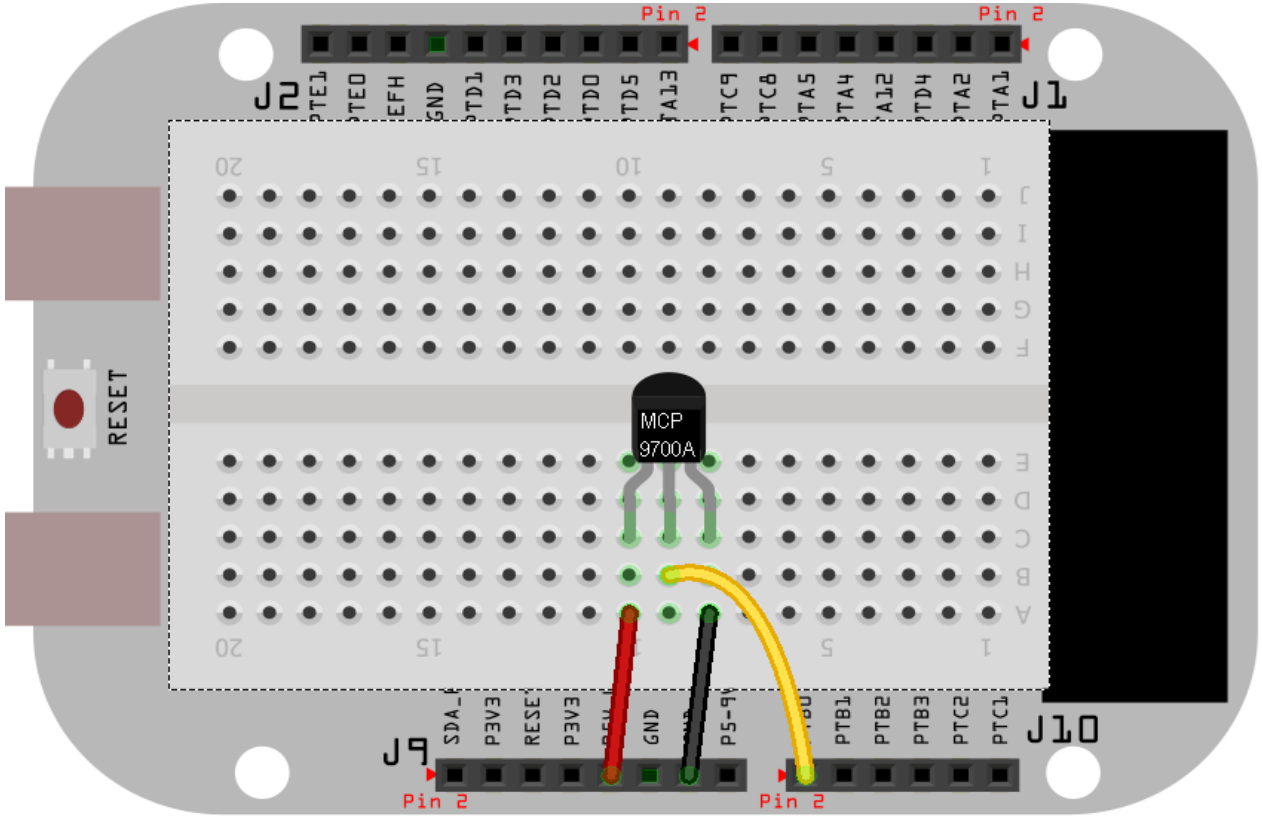
```
#include <MKL25Z4.H>
#include "stdio.h"
void ADC0_init(void) {
    SIM->SCGC6 |= 0x8000000;            // ADC0 engedélyezése
    ADC0->SC2 &= ~0x40;                 // szoftveres triggerelés
    ADC0->SC3 = ADC_SC3_AVGE_MASK       // Hardveres átlagolás engedélyezése
              | ADC_SC3_AVGS(3);        // 32 minta átlagolása
    // Div4, Long sampling, single ended 16-bit, bus clock
    ADC0->CFG1 = 0x40 | 0x10 | 0x0C | 0x00;
}
int main(void) {
  int result,v,temp;
  ADC0_init();                          // ADC0 konfigurálás
  UART_config();                        // UART konfigurálás
  printf("\r\nInternal temperature sensor\r\n");
  while(1){
      ADC0->SC1[0] = 26;                // Mérés a 26. csatornán (belső hőmérő)
      while(!(ADC0->SC1[0] & 0x80));    // Konverzió végére várunk
      result = ADC0->R[0];              //Eredmény kiolvasás, COCO bit törlése

      v = result*3300/65536;
      temp = 25 - (v - 716)*1000/1620;
      printf("volt: %d mV  temp: %d C\r\n",v,temp);
      delayMs(2000);
  }
}
```

```
Internal temperature sensor

volt: 713 mV  temp: 26 C

volt: 713 mV  temp: 26 C

volt: 713 mV  temp: 26 C

volt: 713 mV  temp: 26 C
```

Külső hőmérő használata

Az **MCP9700A** analóg hőmérő, tápfeszültség tartomány: 2,5 V - 5,5 V
Széles hőfoktartományban képes mérni: -40°C - +150°C
Nagy pontosságú : max. ± 2 °C a 0 -70 °C tartományban
Lineáris hőmérsékletfüggés: 0 °C-on 500 mV, meredekség = 10mV/°C





PTB0 bemenet (8. csatorna, Arduino kompatibilis **A0**) 16-bites SE mód, 32 minta átlagolással.

U[mV]= ADC*3300/65535

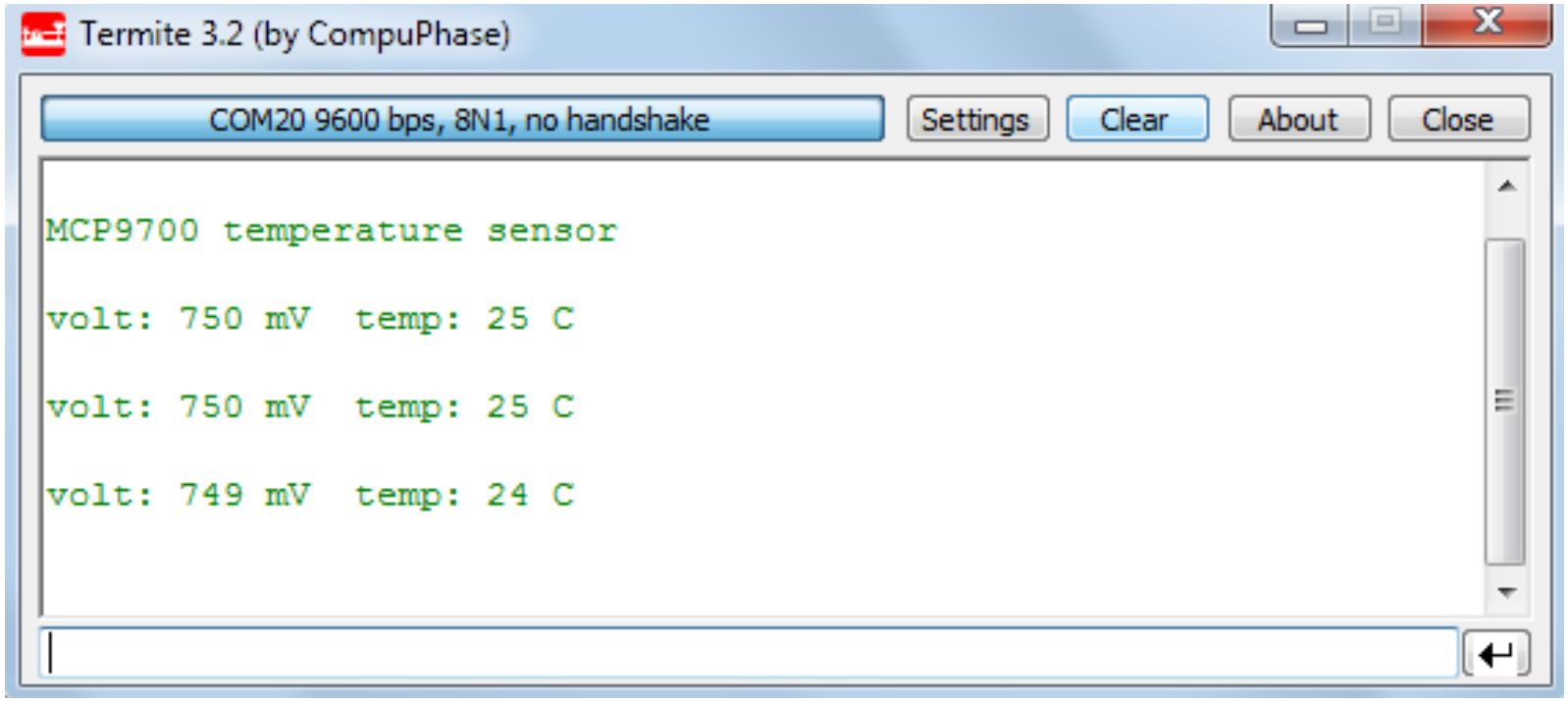
T[°C] = (U[mV] – 500)/10

Az eredményt **UART0**-on írjuk ki.

Program7_3 listája

```
#include <MKL25Z4.H>
#include "stdio.h"
void ADC0_init(void) {
    SIM->SCGC5 |= 0x400;                    // PORTB engedélyezése
    PORTB->PCR[0] = 0;                      // PTB0 legyen analóg módban
    SIM->SCGC6 |= 0x8000000;                // ADC0 engedélyezése
    ADC0->SC2 &= ~0x40;                     // szoftveres triggerelés
    ADC0->SC3 = ADC_SC3_AVGE_MASK           // Hardveres átlagolás engedélyezése
              | ADC_SC3_AVGS(3);            // 32 minta átlagolása
    ADC0->CFG1 = 0x40 | 0x10 | 0x0C | 0x00; // 16-bit SE mód, 6MHz ADCK
}
int main(void) {
  int result,v,temp;
  SystemCoreClockUpdate();
  ADC0_init();                              // ADC0 konfigurálás
  UART_config();                            // UART konfigurálás
  printf("\r\nMCP9700 temperature sensor\r\n");
  while(1){
      ADC0->SC1[0] = 8;                     // Mérés a 8. csatornán (PTB0)
      while(!(ADC0->SC1[0] & 0x80));        // Konverzió végére várunk
      result = ADC0->R[0];                  //Eredmény kiolvasás, COCO bit törlés
      v = result*3300/65536;
      temp = (v - 500)/10;
      printf("volt: %d mV  temp: %d C\r\n",v,temp);
      delayMs(2000);
  }
}
```

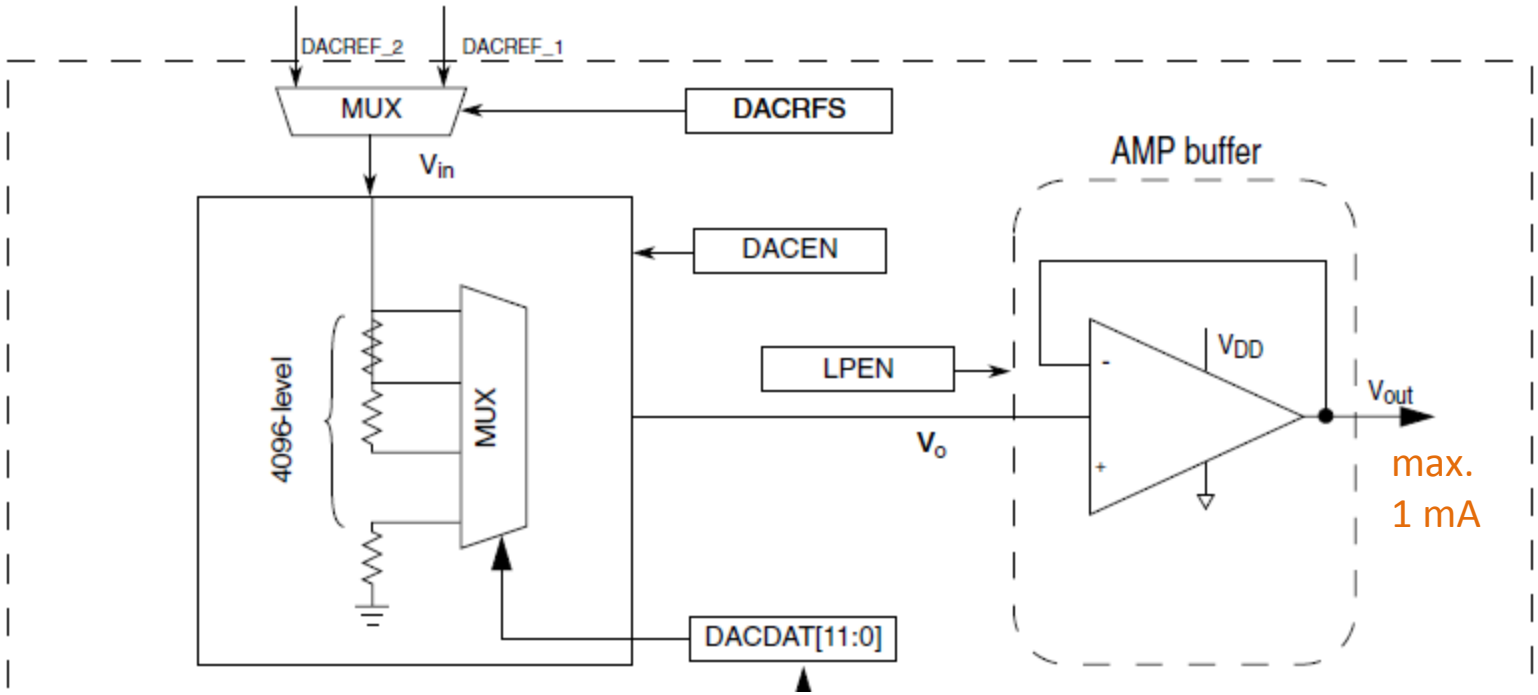
Program7_3 futási eredménye



Digitális – analóg átalakító

12-bit DAC

DAC áttekintés



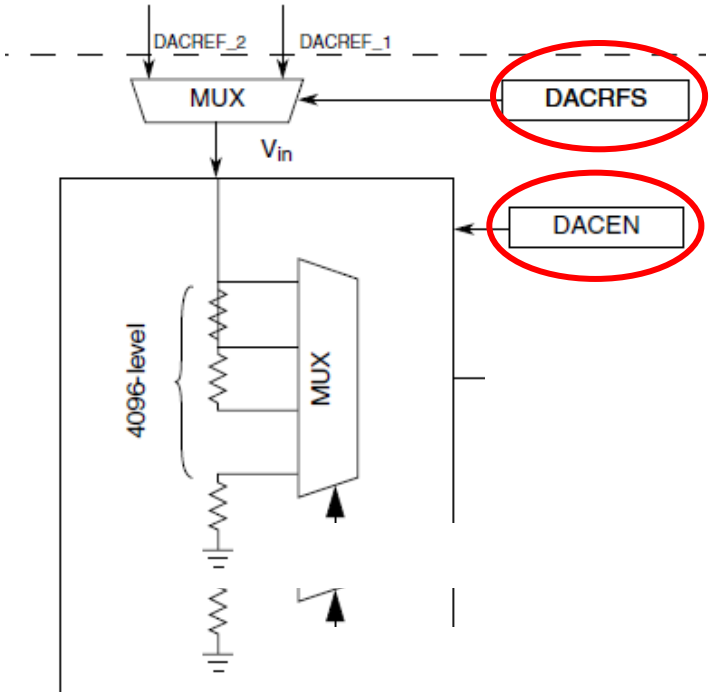
- A DACDAT regiszterbe 12-bites adatot töltünk
- MUX kiválasztja a 4096 fokozatú osztó valamelyik pontját a $V_o = (N+1)*V_{in}/2^{12}$ feszültség előállítására.
- A kimeneti erősítő buffereli a V_o feszültséget, s előállítja a V_{out} kimenőjelet
  - $V_o = V_{out}$ de V_o nem terhelhető, ezért bufferelni kell

DAC üzemmódok

- Normál
  - DAT0 azonnal konvertálódik
- **Bufferelt**
  - Az adat egy FIFO átmeneti tárba kerül
  - A következő adat akkor kerül ki a DAC-ra, amikor a kiválasztott trigger forrás jelzést ad
    - Software Trigger – írás DAC0_C0[DACSWTRG] bitbe
    - Hardware Trigger - a PIT időzítő
  - **Normál Mód**
    - Gyűrűs tárként használja a buffert
  - **Egyszeri pásztázás mód**
    - A mutató tovább lép, míg a buffer végét el nem értük, majd leállás
  - Állapotjelző bitek a **DAC0_SR** regiszterben

DAC vezérlő regiszter 0: DAC0_C0

Bit	7	6	5	4	3	2	1	0
Read	DACEN	DACRFS	DACTRGSEL	0	LPEN	0	DACBTIEN	DACBBIEN
Write				DACSWTRG				
Reset	0	0	0	0	0	0	0	0



- **DACEN** - DAC engedélyezés, ha 1
- **DACRFS** - DAC referencia forrás
  - 0: DACREF_1. VREFH cálasztása
  - 1: DACREF_2. VDDA választása
- **LPEN** – kisfogyasztású mód
  - 0: Gyors mód. Gyors (15 us beállási idő) de nagyobb áramfelvétel (900 uA)
  - 1: Kisfogyasztású mód. Lassabb (100 us beállási idő) de kisebb áramfelvétel (250 uA)
- További regiszterek is vannak a bufferelt módhoz.

DAC vezérlő regiszter 1: DAC0_C1

Bit	7	6	5	4	3	2	1	0
Read	DMAEN	0				DACBFMD	0	DACBFEN
Write								
Reset	0	0	0	0	0	0	0	0

- **DACBFEN**
  - 0: bufferelt üzemmód letiltása
  - 1: bufferelt üzemmód engedélyezése
- **DACBFMD** – Bufferelt mód
  - 0: Normál mód (gyűrűs tár)
  - 1: Egyszeri pásztázás mód
- **DMAEN**
  - DMA átvitel engedélyezése

DAC adatregiszterek

- Ezek a regiszterek (is) csak 8 bitesek

- DATA[11:0] két regiszterben van tárolva
  - DATA0: alsó bájt [7:0 bitek] a DAC0_DAT0L regiszterben
  - DATA1: Falső fél bájt [11:0 bitek] a DAC0_DAT0H regiszterben

Megjegyzések:

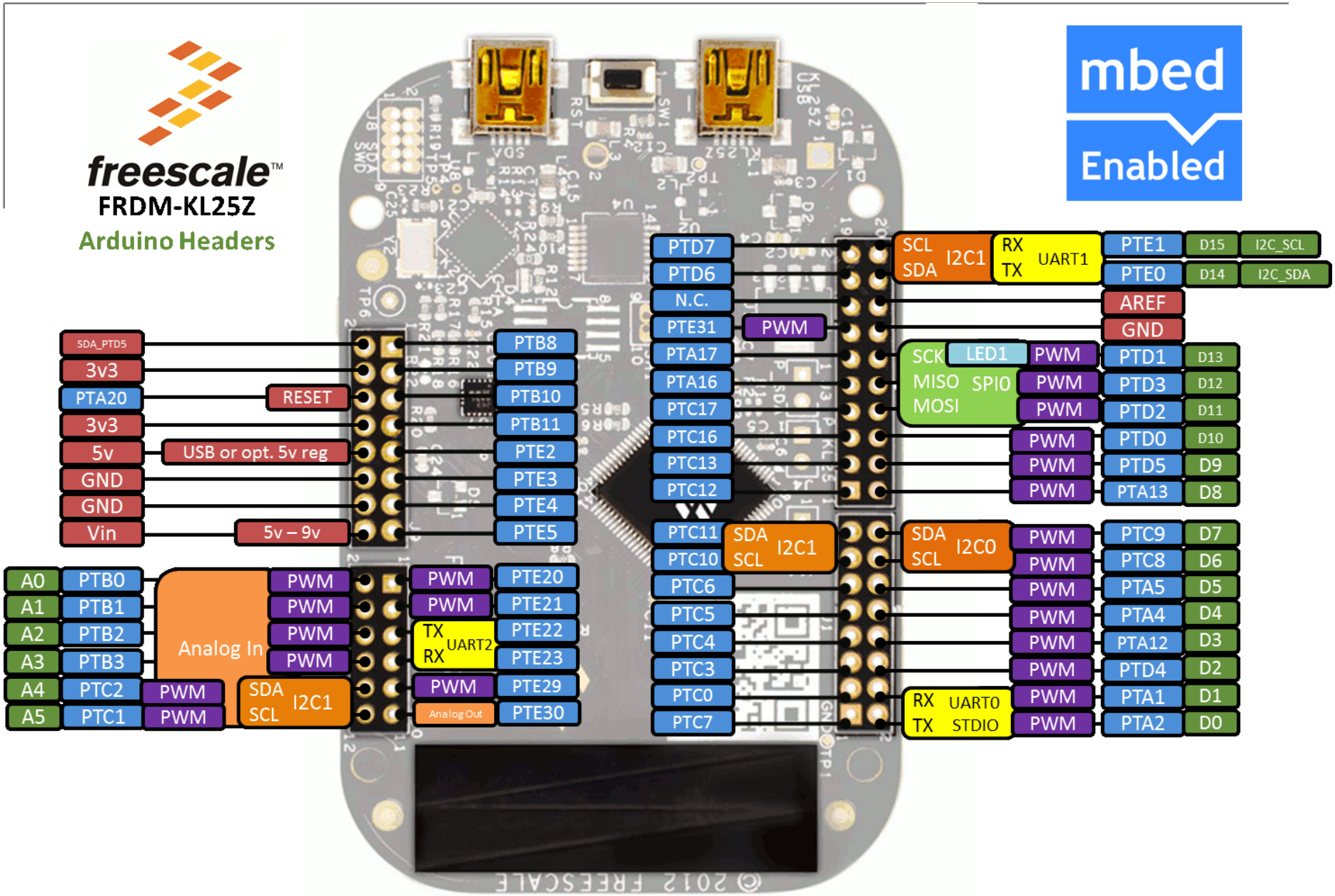
- Az adatregiszterek CMSIS nevei: **DAC0->DAT[0].DATL** és **DAC0->DAT[0].DATH**
- Több adatregiszter is van, de nem bufferelt módban csak **DAT[0]** használható.

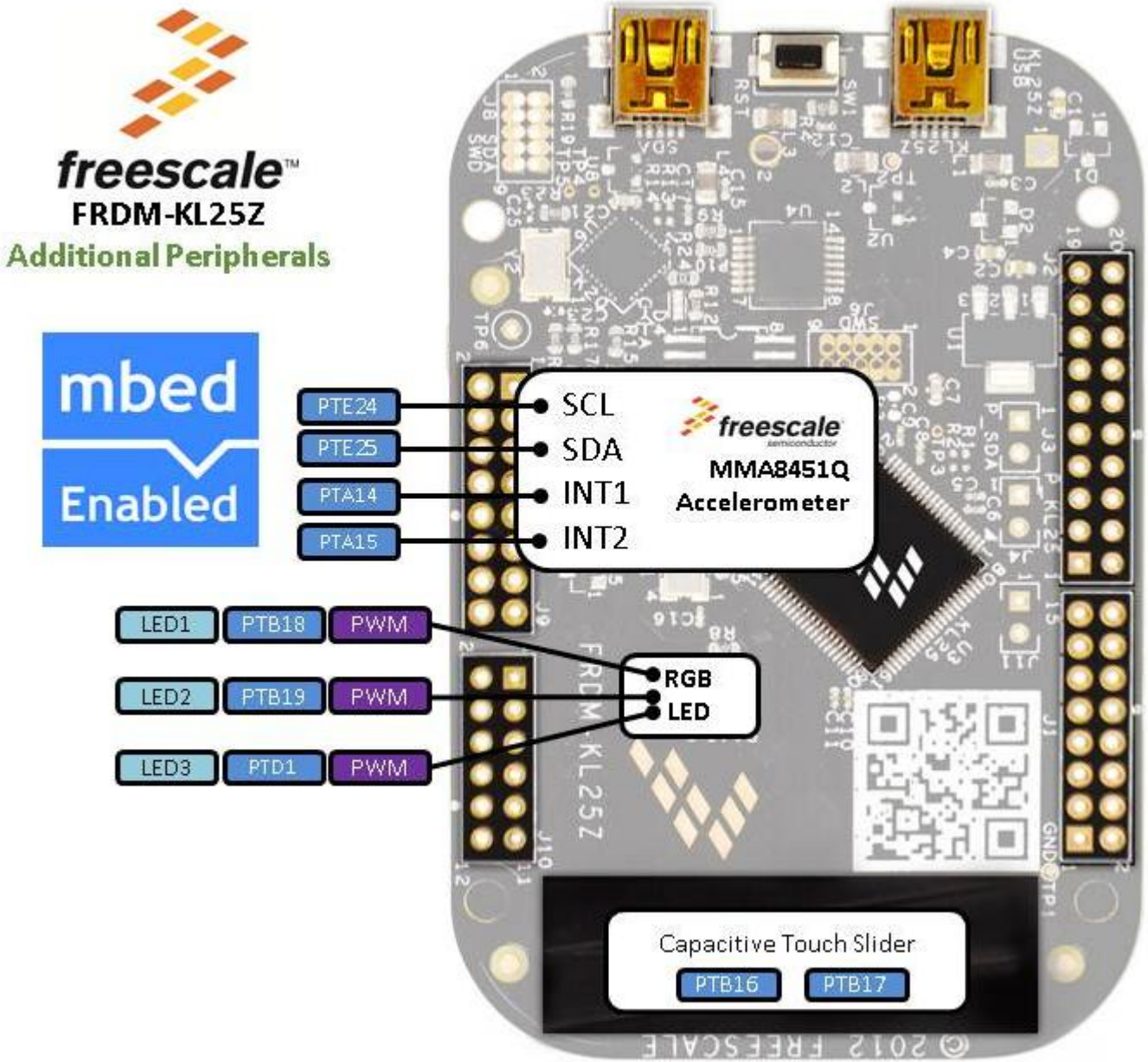
Program7_4: fűrészfog jel generálása

A DAC kimenő jelét 0-tól 4095-ig láptetjük, 16-os lépéssel (összesen 256 ciklus). Minden léptetés után 1 ms-ot várakozunk. A kimenő jel a **PTE30** kivezetésen figyelhető meg.

```
#include "MKL25Z4.h"
void DAC0_init(void);
void delayMs(int n);

int main (void) {
  int i;
  DAC0_init();                              // DAC konfigurálása
  while (1) {
    for (i = 0; i < 0x1000; i += 0x0010) {
      DAC0->DAT[0].DATL = i & 0xff;         // Alsó bájt írása
      DAC0->DAT[0].DATH = (i >> 8) & 0x0f; // Felső bitek írása
      delayMs(1);                           // Késleltetés
    }
  }
}

void DAC0_init(void) {
  SIM->SCGC5 |= 0x2000;        // PORTE engedélyezése
  PORTE->PCR[30] = 0;          // PTE30 analóg módban legyen
  SIM->SCGC6 |= 0x80000000;    // DAC modul engedélyezése
  DAC0->C1 = 0;                // Bufferelt mód letiltása
  DAC0->C0 = 0x80 | 0x20;      // DAC engedélyezés és szoftveres triggerelés
}
```

A már sokszor bemutatott késleltető függvényt itt most nem részletezzük.


freescale
FRDM-KL25Z
Arduino Headers
mbed
Enabled
SDA_PTD5
3v3
PTA20
3v3
5v
GND
GND
Vin
RESET
USB or opt. 5v reg
5v – 9v
PTB8
PTB9
PTB10
PTB11
PTE2
PTE3
PTE4
PTE5
A0 PTB0
A1 PTB1
A2 PTB2
A3 PTB3
A4 PTC2
A5 PTC1
Analog In
PWM
SDA SCL I2C1
PTE20
PTE21
TX RX UART2
PTE22
PTE23
PTE29
PTE30
Analog Out
PTD7
PTD6
N.C.
PTE31
PTA17
PTA16
PTC17
PTC16
PTC13
PTC12
PTC11
PTC10
PTC6
PTC5
PTC4
PTC3
PTC0
PTC7
SCL SDA I2C1
RX TX UART1
PTE1 D15 I2C_SCL
PTE0 D14 I2C_SDA
AREF
GND
SCK MISO MOSI SPI0
LED1
PTD1 D13
PTD3 D12
PTD2 D11
PTD0 D10
PTD5 D9
PTA13 D8
SDA SCL I2C0
PTC9 D7
PTC8 D6
PTA5 D5
PTA4 D4
PTA12 D3
PTD4 D2
PTA1 D1
PTA2 D0
RX TX UART0 STDIO
© 2012 FREESCALE


freescale
FRDM-KL25Z
Additional Peripherals
mbed
Enabled
PTE24
PTE25
PTA14
PTA15
SCL
SDA
INT1
INT2
freescale
MMA8451Q
Accelerometer
LED1
PTB18
PWM
LED2
PTB19
PWM
LED3
PTD1
PWM
RGB
LED
Capacitive Touch Slider
PTB16
PTB17