

# Kód írása regiszterszinten

Név: Pilter Zsófia, Vad Avar

Dátum: 2024.11.25

Mérőhely: 1 jobb és bal

## Ajánlott irodalom

A jegyzőkönyvhöz tartozó PPT (MicLab-Registers.pptx)

EFM8BB1 Reference Manual: <https://www.silabs.com/documents/public/reference-manuals/efm8bb1-rm.pdf>

Laboratory Practical: [http://eta.bibl.u-szeged.hu/901/1/2011\\_0104\\_szte\\_6\\_laboratory\\_practical.pdf](http://eta.bibl.u-szeged.hu/901/1/2011_0104_szte_6_laboratory_practical.pdf)

Honlap: <http://www.inf.u-szeged.hu/noise/Education/MicLab/>

## Jegyzőkönyv készítése

A jegyzőkönyvek az órán végzett munka dokumentálására szolgálnak. A letölthető minta jegyzőkönyvet kell kiegészíteni a megfelelő információkkal: név, dátum, mérőhely (pl. 3. jobb), a feladatokhoz tartozó esetleges kifejtendő válaszokkal, valamint a kódok lényeges részével.

A jegyzőkönyveket a Coospace-en kell feltölteni, külön pdf formátumban csatolni kell a jegyzőkönyvet (a fájl neve a következő mintát kövesse: NagyJ.KissB.03.pdf), egy külön zip fájlban pedig a kódokat (\*.c, \*.cwg). Amennyiben probléma merül fel a beadás során, az anyagokat az oktató e-mail címére kell elküldeni, levél tárgya legyen pl. MicLab 03.

## 1. feladat – LEDo bekapcsolása regiszterszinten

Írjon egy olyan C kódot, ami bekapcsolja az EFM8BB1 fejlesztőkiten lévő LEDo nevű LED-et. Ehhez csak a Reference Manual használható, a konfigurátor nem. A feladathoz egy „Empty C Program” típusú projektet hozzon létre. A feladatra akkor jár teljes a pontszám, ha a regiszter értékei esetén a konstansok helyett a beépített enumok használatával oldjuk meg. Az enumok az SI\_EFM8BB1\_Register\_Enums.h fájlban találhatók.

*Tipp: A jegyzőkönyvhöz tartozó PPT-ben megtalálhatók a szükséges regiszterek.*

A program részekre bontott forráskódja (Main.c, Interrupts.c, ha van):

### Main.c:

```
//-----  
// Includes  
//-----  
#include <SI_EFM8BB1_Register_Enums.h>  
  
#define LED0 P1_B4// SFR declarations  
#define SHIFT_VALUE 1U  
  
//-----  
// SiLabs_Startup() Routine  
// -----  
// This function is called immediately after reset, before the initialization
```

```

// code is run in SILABS_STARTUP.A51 (which runs before main() ). This is a
// useful place to disable the watchdog timer, which is enable by default
// and may trigger before main() in some instances.
//-----
void SiLabs_Startup (void)
{
    // Disable the watchdog here
    WDTCN |= WDTCN_WDTCN_FMASK;
    XBR2 |= XBR2_XBARE__ENABLED;
    P1MDOUT |= P1MDOUT_B4__PUSH_PULL;
}

//-----
// main() Routine
// -----
// Note: the software watchdog timer is not disabled by default in this
// example, so a long-running program will reset periodically unless
// the timer is disabled or your program periodically writes to it.
//
// Review the "Watchdog Timer" section under the part family's datasheet
// for details. To find the datasheet, select your part in the
// Simplicity Launcher and click on "Data Sheet".
//-----
int main (void)
{
    LED0 = P1_B4__LOW;
    LED0 &= ~(SHIFT_VALUE << IE_ET0__SHIFT);
    while (1)
    {
        // Spin forever
    }
}

```

Az elkészült programot be kell mutatni!

A gyakorlatvezető ellenőrizte:

- Igen
- Nem

A program működött:

- Igen
- Nem

## 2. feladat - Timer 3 használata 16 bites módban, interrupt engedélyezéssel

Írjon egy programot regiszterszinten, ami a Timer 3 segítségével villogtatja a LED0-át 10 Hz-es frekvenciával. A Timer 3-at 16 bites módban használja és engedélyezze a hozzá tartozó interruptot. Az interrupt rutint is írja meg (number\_of\_interrupt: TIMER3\_IRQn), amiben a LED0-hoz tartozó bit kapcsolgatása történik. A reload érték képlettel legyen implementálva a kódban, ne csak egy konstansként, melynek „bemenete” legyen a timer kívánt túlsordulási frekvenciája. (A képlethez segítséget talál a Miclab-utmutato.pdf-ben).

*Tipp: a Timer 3 használatához interrupt módban a TMR3CNo (TR3, TF3H bitek) és a TMR3RLH, TMR3RLL reload regiszterek írása szükséges, valamint a Timer 3-hoz tartozó interrupt engedélyezése*

az EIE1 regiszterben és az összes interrupt engedélyezése az IE regiszterben. A TF3H overflow flaget a megfelelő bitművelettel kell törölnünk, úgy hogy ne írjuk felül a többi helyiértéken lévő biteket, mert a Timer 3 control regisztere nem bitcímezhető.

A program részekre bontott forráskódja (Main.c, Interrupts.c, ha van):

## Main.c:

```
//-----  
// Includes  
//-----  
#include <SI_EFM8BB1_Register_Enums.h>           // SFR declarations  
#include <SI_EFM8BB1_Defs.h>  
  
#define LED0 P1_B4  
#define SYSCLK 24500000u  
#define LED0 P1_B4  
#define SYSCLK 24500000u  
#define TIMER_SIZE 0xFFFF  
#define TIMER_PRESCALER 12u  
#define SYSCLOCK_PRESCALER 8u  
#define SHIFT_VALUE 8u  
#define MASK_VALUE 0xFF  
#define TARGET_FREQUENCY 10u  
//-----  
// SiLabs_Startup() Routine  
// -----  
// This function is called immediately after reset, before the initialization  
// code is run in SILABS_STARTUP.A51 (which runs before main() ). This is a  
// useful place to disable the watchdog timer, which is enable by default  
// and may trigger before main() in some instances.  
//-----  
void SiLabs_Startup (void)  
{  
    // Disable the watchdog here  
    WDTCN |= WDTCN_WDTCN_FMASK;  
    XBR2 |= XBR2_XBARE__ENABLED;  
    P1MDOUT |= P1MDOUT_B4__PUSH_PULL;  
}  
  
void timer3_reload(uint16_t overflow_freq)  
{  
    uint16_t reload_value;  
  
    reload_value = TIMER_SIZE - ((SYSCLK / (TIMER_PRESCALER *  
SYSCLOCK_PRESCALER))/overflow_freq);  
  
    TMR3RLL = reload_value & MASK_VALUE;  
    TMR3RLH = (reload_value >> SHIFT_VALUE) & MASK_VALUE;  
  
    TMR3CN0 = TMR3CN0_TR3__RUN;  
    EIE1 |= EIE1_ET3__ENABLED;  
    IE |= IE_EA__ENABLED;  
}  
  
void Timer3_ISR(void) interrupt TIMER3_IRQn  
{  
    TMR3CN0 &= ~TMR3CN0_TF3H_BMASK;
```

```

    LED0 = ~LED0;
}

//-----
// main() Routine
// -----
// Note: the software watchdog timer is not disabled by default in this
// example, so a long-running program will reset periodically unless
// the timer is disabled or your program periodically writes to it.
//
// Review the "Watchdog Timer" section under the part family's datasheet
// for details. To find the datasheet, select your part in the
// Simplicity Launcher and click on "Data Sheet".
//-----
int main (void)
{
    LED0 = P1_B4__LOW;
    timer3_reload(TARGET_FREQUENCY);

    while (1) {}                // Spin forever
}

```

Az elkészült programot be kell mutatni!

A gyakorlatvezető ellenőrizte:

- Igen
- Nem

A program működött:

- Igen
- Nem

## Megjegyzések