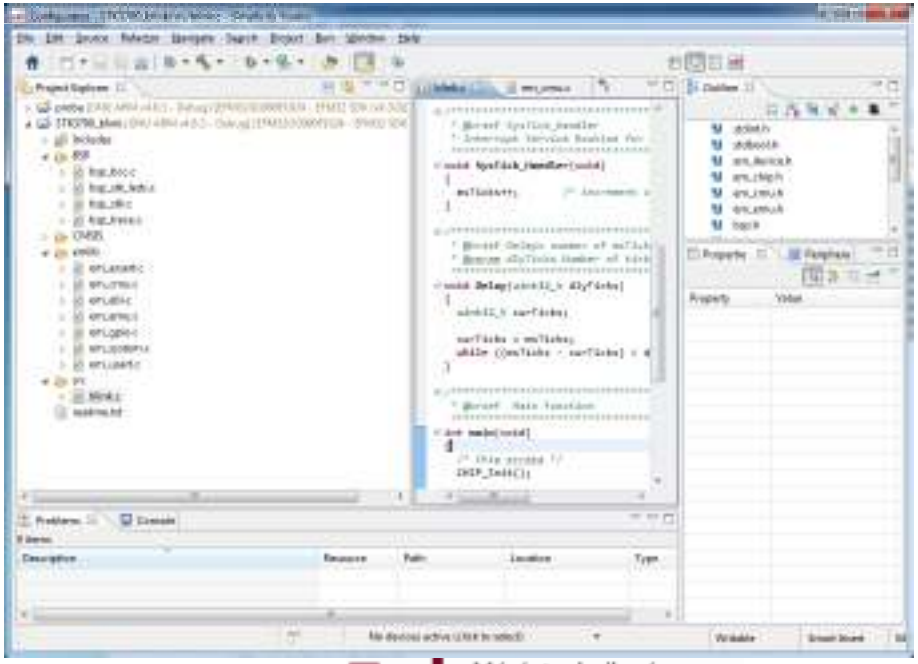
Silabs STK3700, Simplicity Studio laborgyakorlat

Scherer Balázs

Méréstechnika és Információs Rendszerek Tanszék

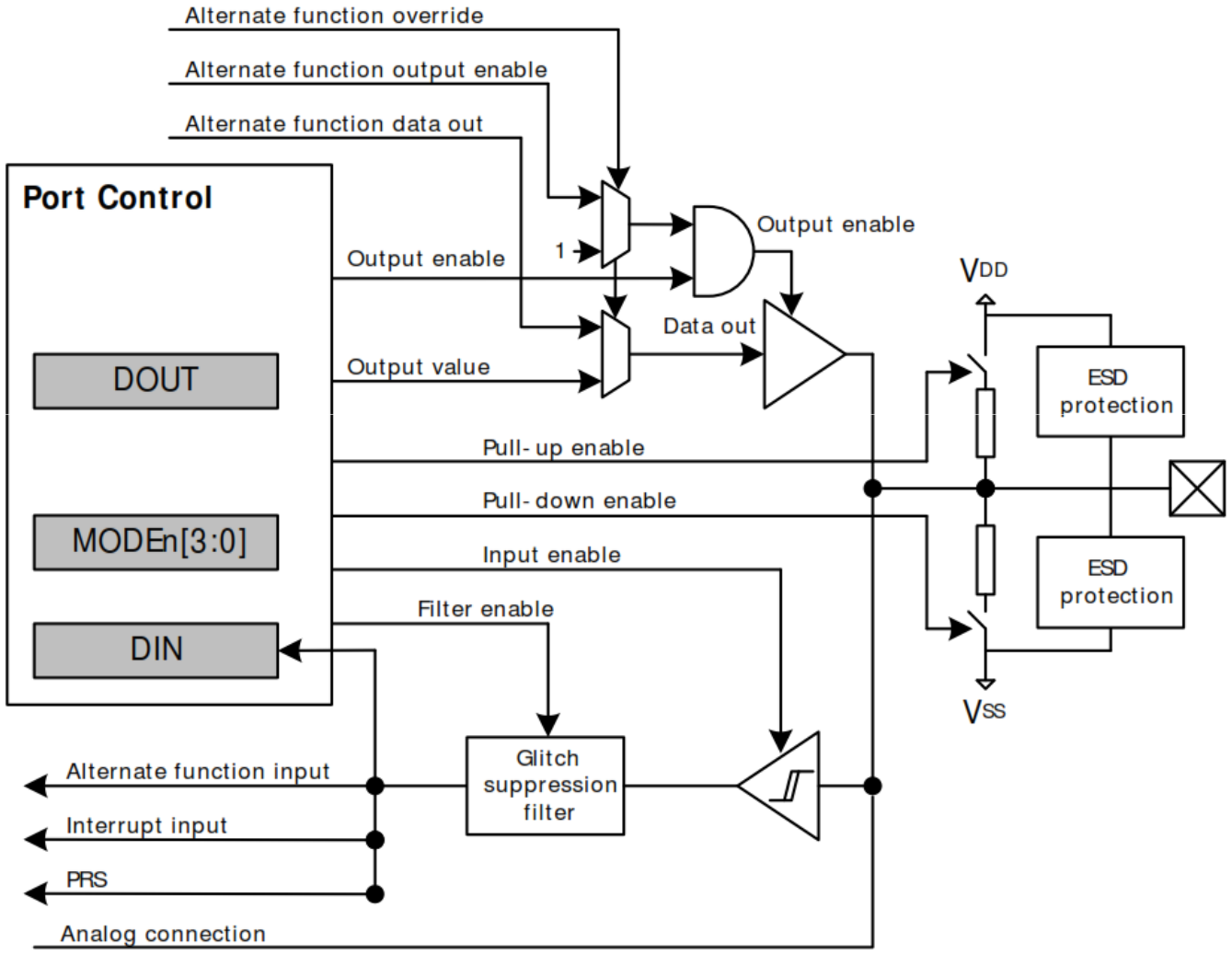
Simplicity Studio

- Saját Firmware library
- Saját ecosystem
- Debug támogatás
- Fogyasztás monitorozás támogatás
- Egyszerű trace feature-ök
- Integrált dokumentáció és példák

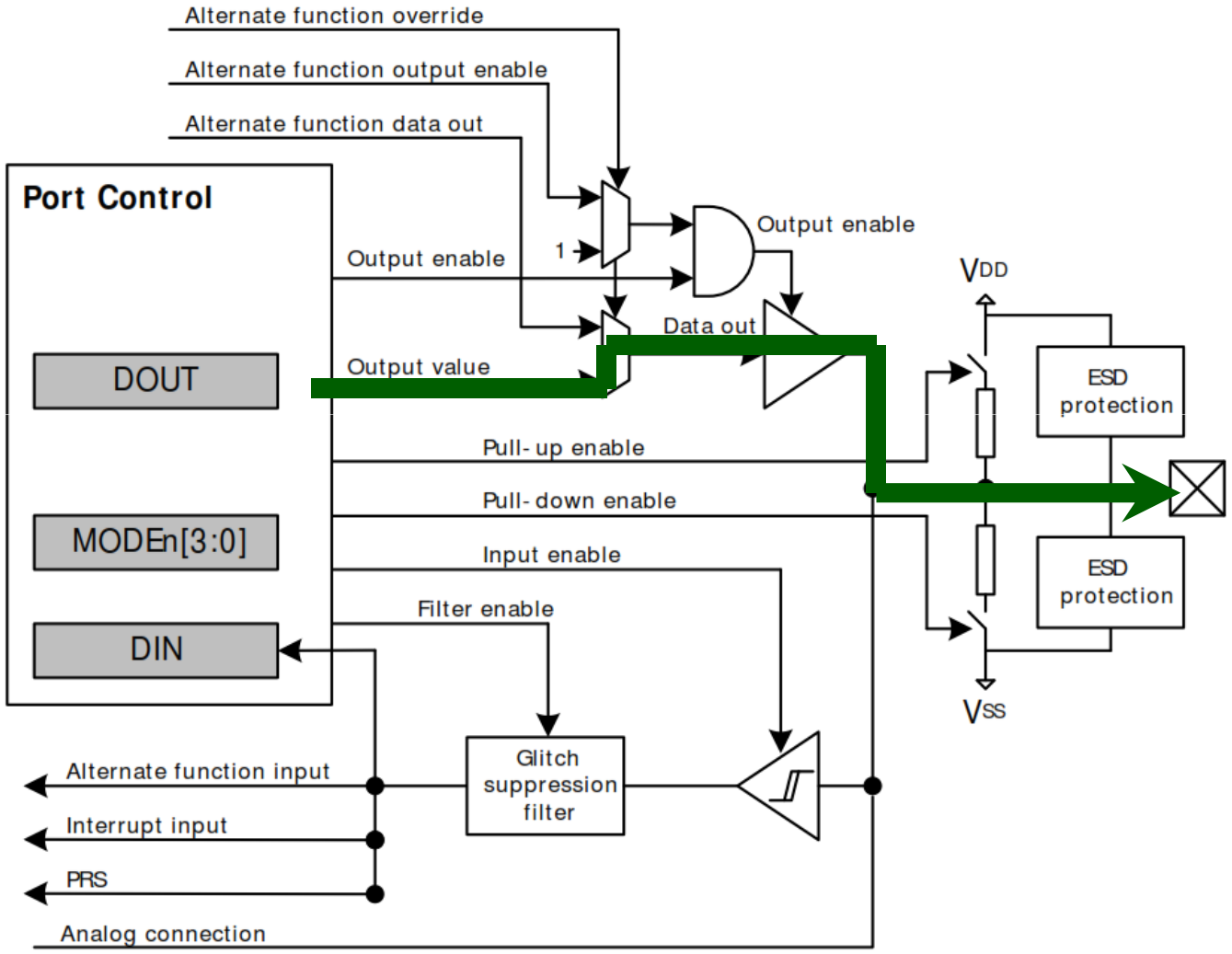


1. Feladat: egyszerű GPIO lábkezelés

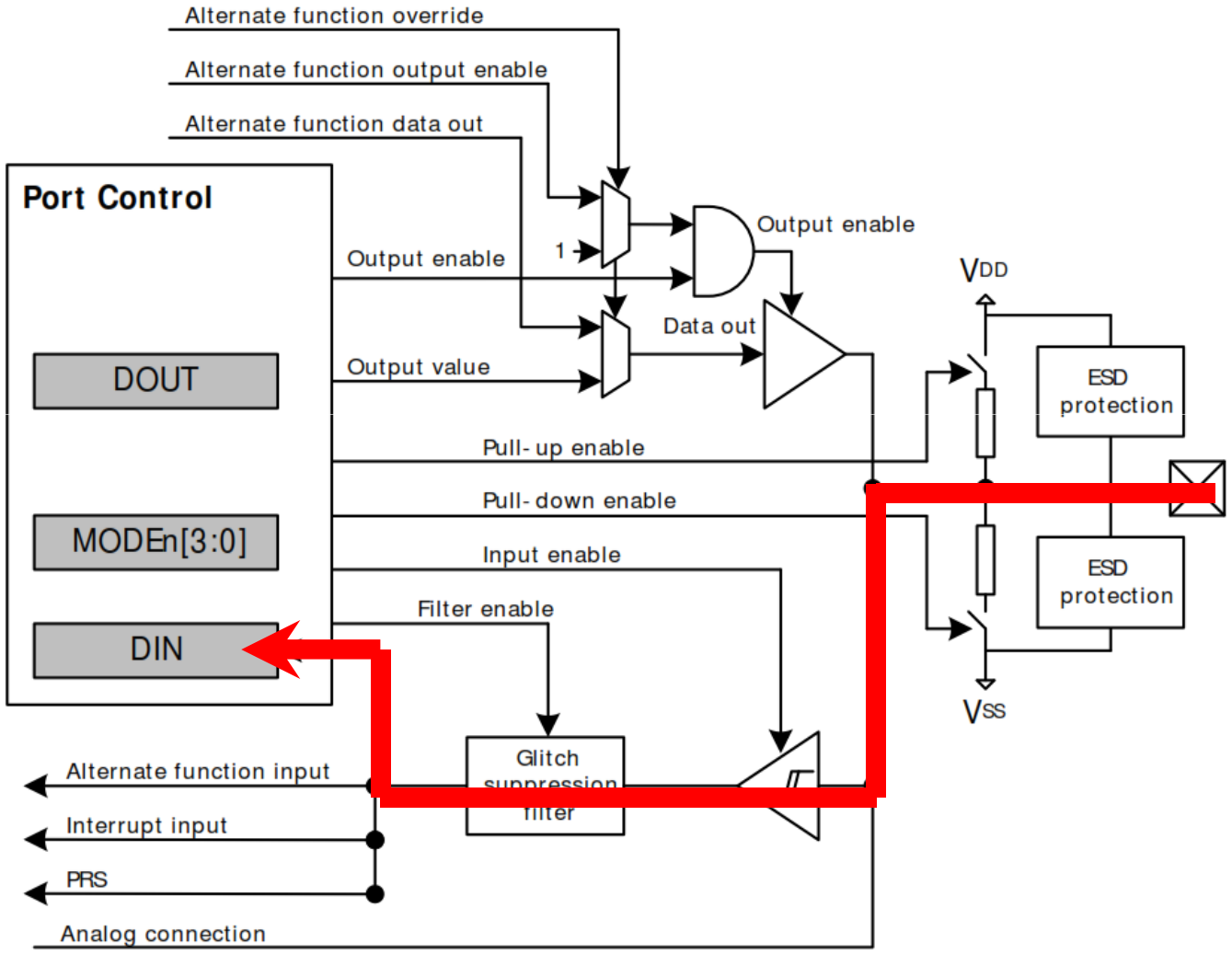
A Giant Gecko GPIO blokkja



A Giant Gecko GPIO blokkja: kimenet

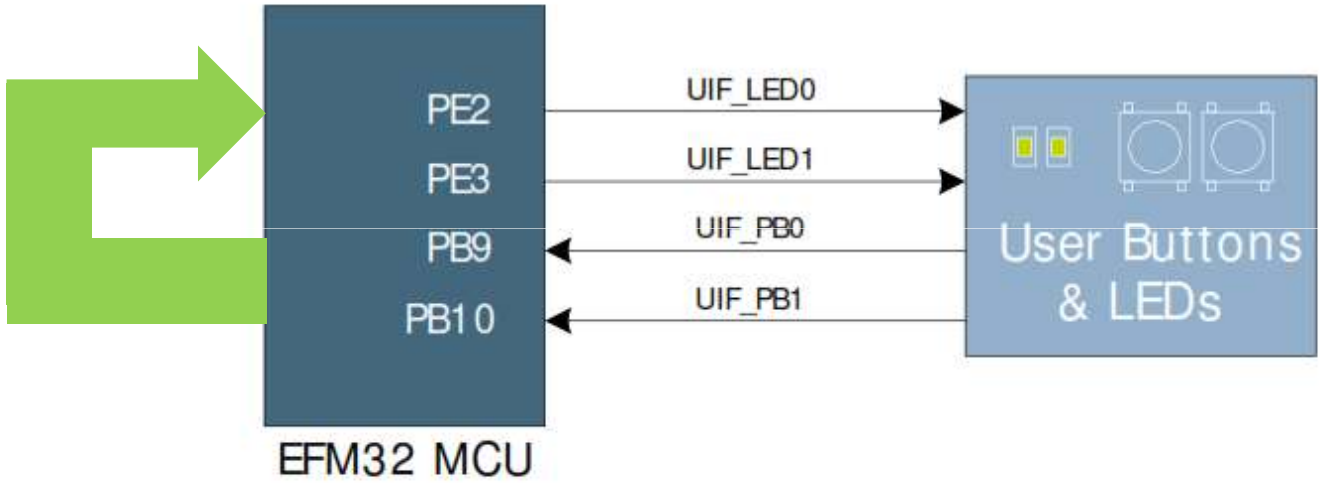


A Giant Gecko GPIO blokkja: bemenet



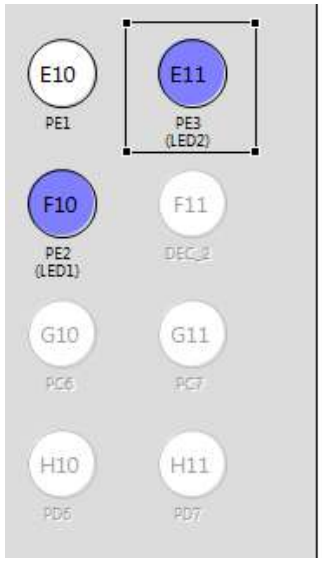
1. Feladat: Nyomógombokról vezérelt LED-ek

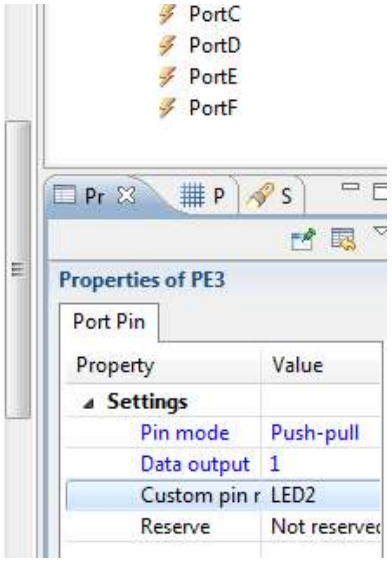
A mi szoftverünk



1. Feladat: Konfigurátoros project létrehozása

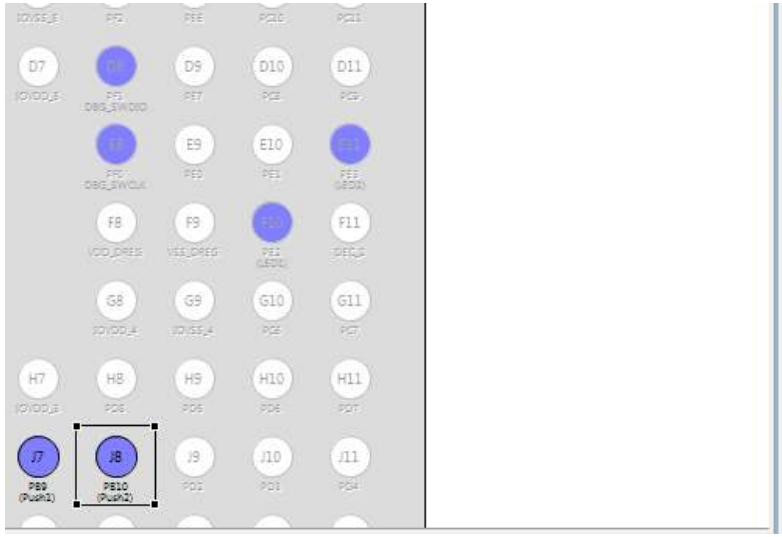
- New Silicon Labs MCU project
- Simplicity Configurator Program
- Konfigurátor beállítások
  - PE2, PE3 output (LED0, LED1)
    - Push-Pull, Data output = 1, Custom pin name

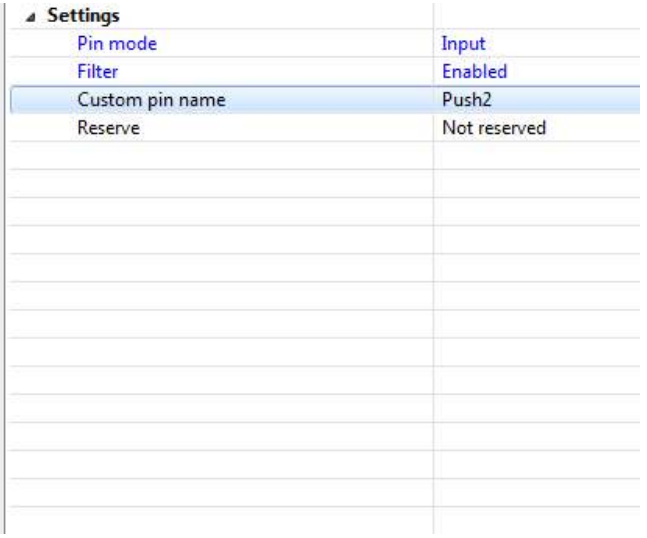




1. Feladat: Konfigurátoros project létrehozása

- New Silicon Labs MCU project
- Simplicity Configurator Program
- Konfigurátor beállítások
  - PE2, PE3 output (LED0, LED1)
  - PB9, PB10 Input, Filer Enabled (Push1, Push2)

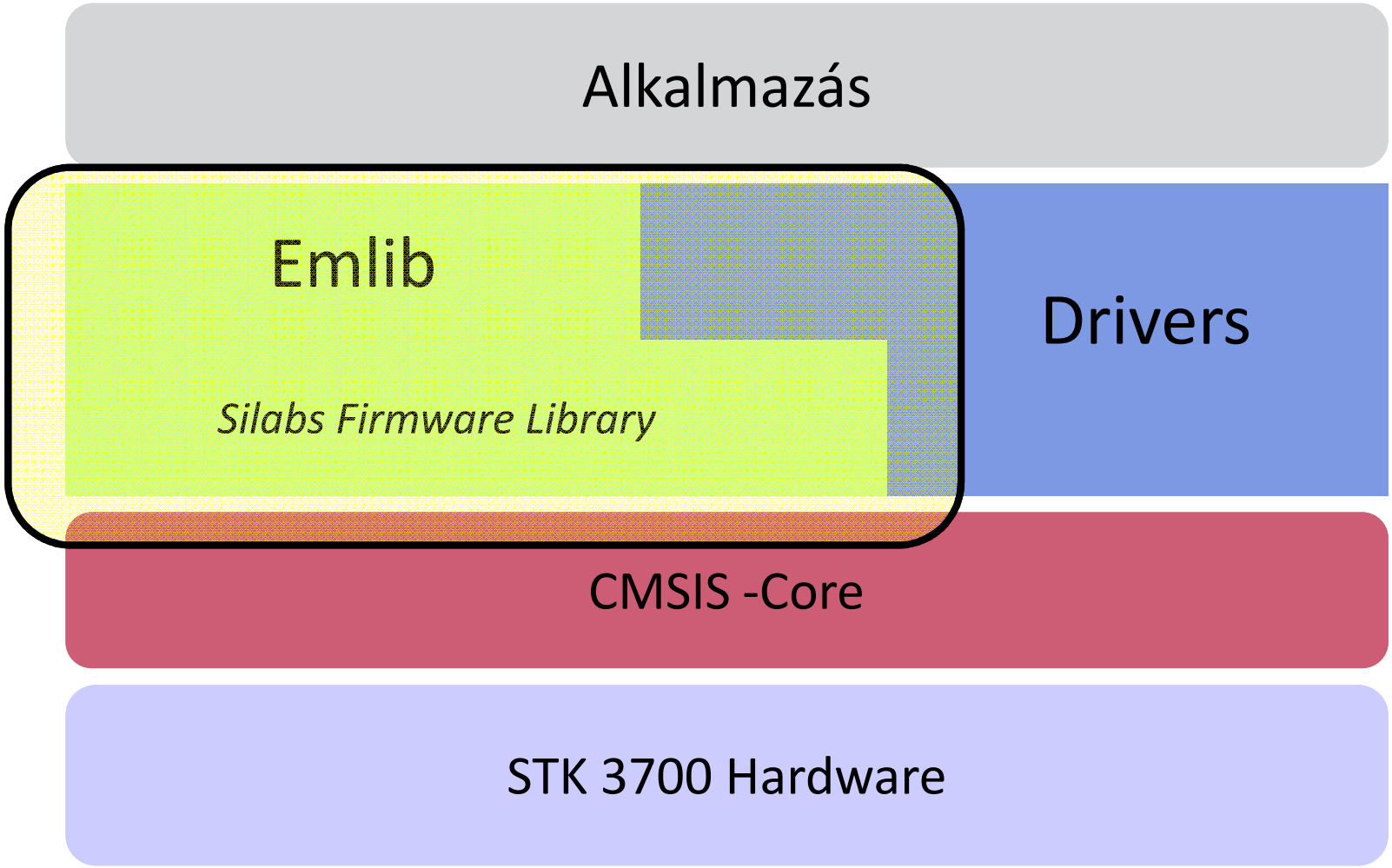




1. Feladat: Project szerkesztése

- InitDevice.c
  - **enter_DefaultMode_from_RESET()**
    - Tartalmazza az összes bekonfigurált inicializációt
    - A default 14 MHz-es belső RC oszcillátor indul
    - Nem kell módosítani
- InitDevice.h
  - **Port és PIN definiciók**
    - Felhasználhatóak a programban
    - Nem kell módosítani
- Main.c
  - Főprogram ide kerül

Silabs SDK alap architektúra



1. Feladat: Firmware Library

EFM32 Giant Gecko Software Documentation

__STATIC_INLINE unsigned int **GPIO_PinInGet** (**GPIO_Port_TypeDef** port, unsigned int pin)
Read the pad value for a single pin in a GPIO port. More...

__STATIC_INLINE void **GPIO_PinOutSet** (**GPIO_Port_TypeDef** port, unsigned int pin)
Set a single pin in GPIO data out register to 1. More...

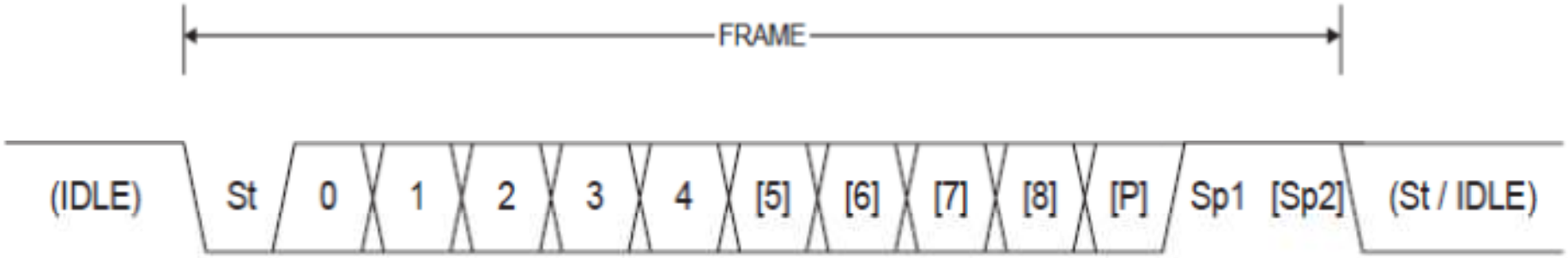
1. Feladat: megoldás

```
int main(void)
{
  /* Chip errata */
  CHIP_Init();

  enter_DefaultMode_from_RESET();

  /* Infinite loop */
  while (1) {
      if ( GPIO_PinInGet (PUSH1_PORT, PUSH1_PIN) == 0)
      {
          GPIO_PinOutClear (LED1_PORT, LED1_PIN);
      }
      else
      {
          GPIO_PinOutSet (LED1_PORT, LED1_PIN);
      }

      if ( GPIO_PinInGet (PUSH2_PORT, PUSH2_PIN) == 0)
      {
          GPIO_PinOutClear (LED2_PORT, LED2_PIN);
      }
      else
      {
          GPIO_PinOutSet (LED2_PORT, LED2_PIN);
      }
  }
}
```

2. Feladat: Kommunikáció a külvilággal: UART

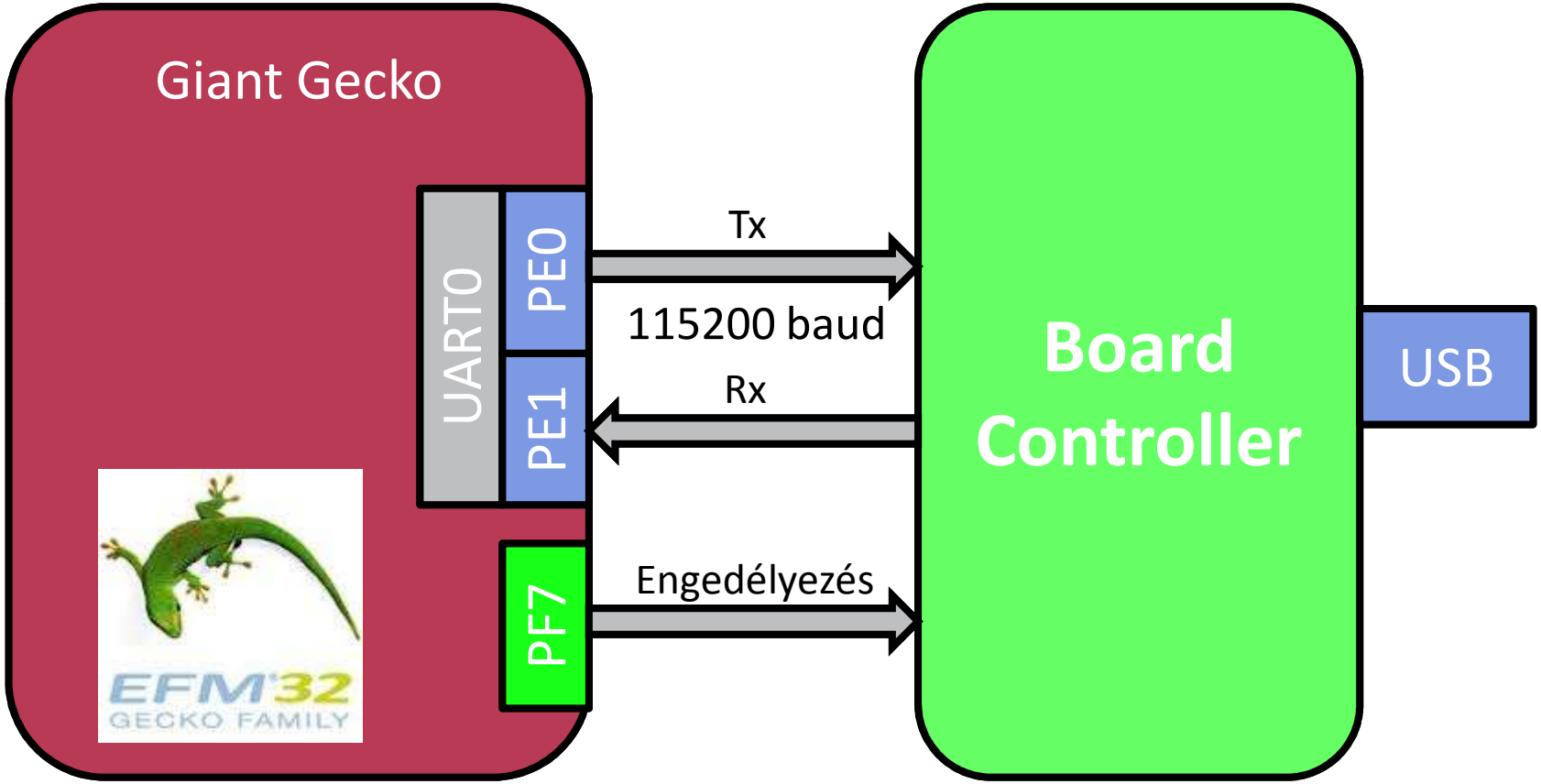
UART *periféria*

- Egy UART keret
  - Start Bit
  - 5, 6, 7, 8 vagy 9 adat bit
  - Paritás Bit
  - 1, 1.5 vagy 2 Stop Bit
- Szabványos adatsebességek
  - 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
- **Fontos a stabil órajel, általában célszerű a kvarc használata**



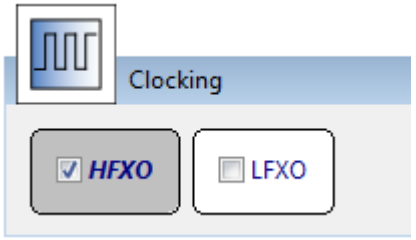
UART *fizikai bekötés*

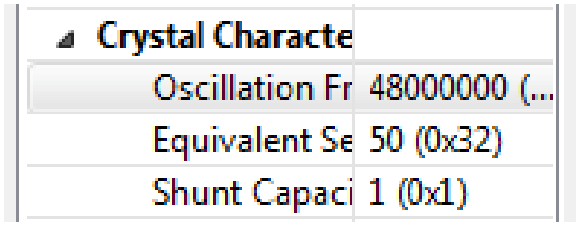
- Demókártya bekötése



Konfigurátoros project bővítése

- Default Mode peripherals fül
  - Órajel forrás konfigurálás: HFXO : 48 MHz. ***Át kell írni !!!***

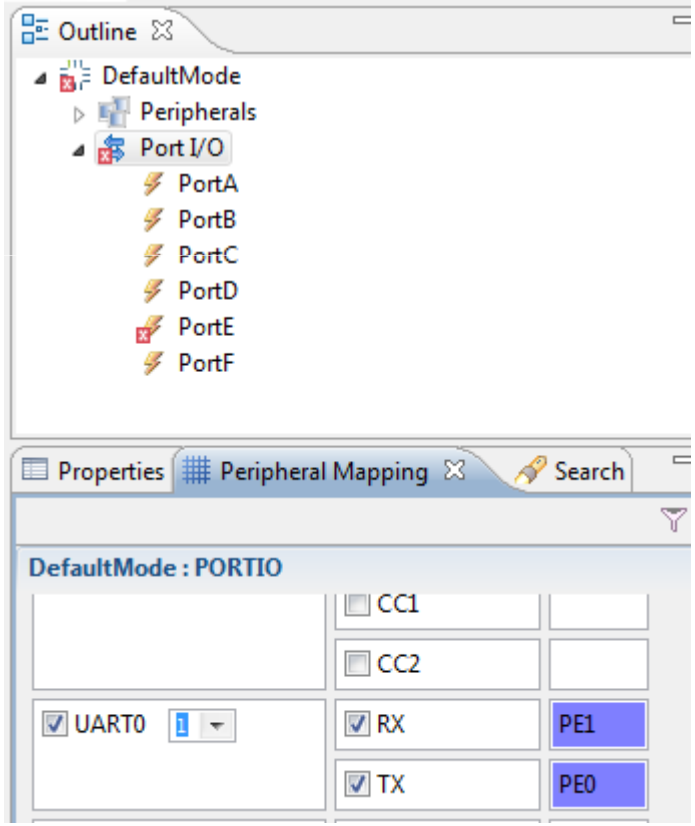




  - UART0 engedélyezés. Paraméterek nem kell változtatni

Lábkonfigurálás 1.

- Port I/O fül jobb felső sarok
  - Engedélyezés. Átállítani az 1. alternatív kivezetésre



Lábkonfigurálás 2.

- Default mode port I/O lábbeállítások
  - UART0:
    - PE0: Push-pull, Data output = 0
    - PE1: Input
  - Board Controller UART enable, GPIO lábként kezelve: PF7
    - Push-Pull, Data output = 1

2.a, Üzenet küldés periodikusan

- Régi program kiegészítés
  - Várakozásra szükség van

```
void   USART_Tx (USART_TypeDef *usart, uint8_t data)
       Transmit one 4-9 bit frame. More...
```

  - Engedélyezést nem elfelejteni

```
GPIO_PinOutSet (UART_ENABLE_PORT, UART_ENABLE_PIN);

USART_Tx (UART0, 'c');
for(volatile int i=0; i<1000000; i++);
```

2.b, Echo

- Karakterre várás és visszaküldés
  - Az Rx Függvény blokkol nincs szükség külön várakozásra

```
uint8_t  USART_Rx (USART_TypeDef *usart)
         Receive one 4-8 bit frame, (or part of 10-16 bit frame). More...
```

```
uint8_t ch = USART_Rx (UART0);
USART_Tx (UART0, ch);
```

2.b, Echo

- Karakterre várás és visszaküldés
  - Az Rx Függvény blokkol nincs szükség külön várakozásra

```
uint8_t  USART_Rx (USART_TypeDef *usart)
         Receive one 4-8 bit frame, (or part of 10-16 bit frame). More...
```

```
uint8_t ch = USART_Rx (UART0);
USART_Tx (UART0, ch);
```

- Nézzük meg mi történt a LED, nyomógomb kezeléssel

3. Feladat: Megszakítás kezelés

3. UART fogadás megszakítással

- Várakozás nélküli echo
- Konfigurátor módosítások nem szükségesek

- UART interrupt engedélyezés

```
__STATIC_INLINE void  USART_IntEnable (USART_TypeDef *usart, uint32_t flags)
                      Enable one or more USART interrupts. More...
```

  - Flag: USART_IF_RXDATAV (efm32gg_usart.h)
    - Data Valid flag

```
#define USART_IF_RXDATAV                (0x1UL << 2)                    /**< RX Data Valid Interrupt Flag */
```

3. UART fogadás megszakítással

- Vektoros Interrupt kezelő engedélyezés: NVIC
  - A periféria kiváltja az IT-t, de ez mondja meg, hogy ki és hogyan kezeli
  - ***NVIC_EnableIRQ(UART0_RX_IRQn)***: engedélyezet IRQ a az emf32gg990f1024.h-ból (em_device.h includeo-on keresztül F3-al eljutva)
    - *UART0_RX_IRQn = 20, /*!< 16+20 EFM32 UART0_RX Interrupt */*
- Interrupt kezelő függvény
  - startup_gcc_efm32gg.s –ból a neve
  - **void *UART0_RX_IRQHandler*(void)**
  - Fogadás, IT nyugtázás: ***USART_IntClear*** (USART_TypeDef *usart, uint32_t flags)

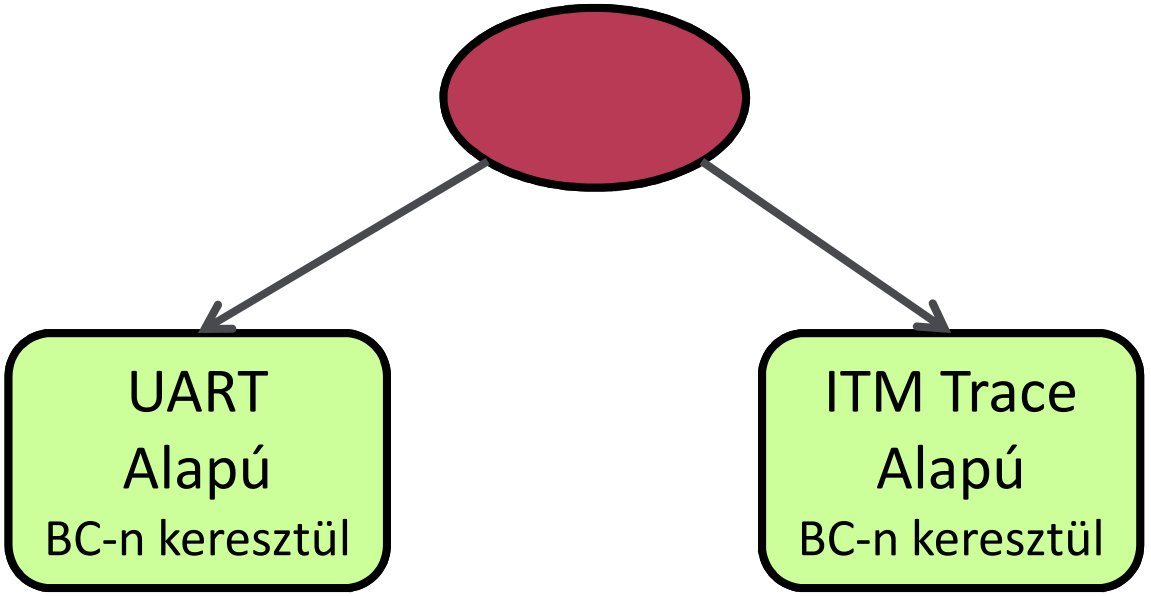
3. UART fogadás megszakítással: megoldás

```c
#include "em_device.h"
#include "em_chip.h"
#include "em_gpio.h"
#include "em_usart.h"
#include "InitDevice.h"

void UART0_RX_IRQHandler(void)
{
      USART_IntClear (UART0, USART_IF_RXDATAV);
      uint8_t ch = USART_Rx (UART0);
      USART_Tx (UART0, ch);
}

/**************************************************************************
 * @brief  Main function
 *************************************************************************/
int main(void)
{
  /* Chip errata */
  CHIP_Init();

  enter_DefaultMode_from_RESET();
  USART_IntEnable (UART0, USART_IF_RXDATAV);
  NVIC_EnableIRQ(UART0_RX_IRQn);

  GPIO_PinOutSet (UART_ENABLE_PORT, UART_ENABLE_PIN);
  /* Infinite loop */
  while (1) {
      if ( GPIO_PinInGet (PUSH1_PORT, PUSH1_PIN) == 0)
      {
          GPIO_PinOutClear (LED1_PORT, LED1_PIN);
      }
      else
      {
          GPIO_PinOutSet (LED1_PORT, LED1_PIN);
      }
```

4. Feladat: Printf-elés

Printf használata

- Legalább 2 lehetőség van az stdout használatára

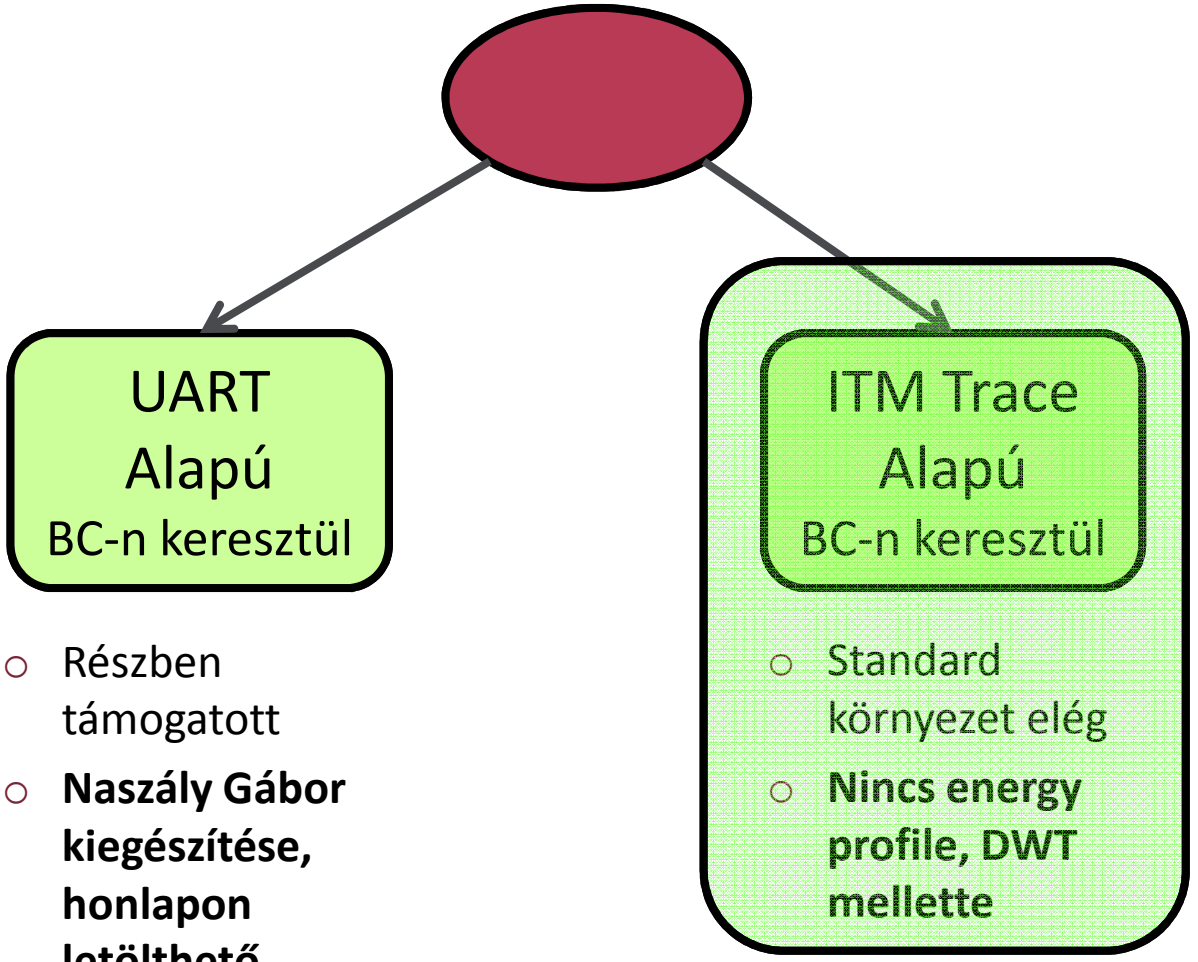


- Részben támogatott
- **Naszály Gábor kiegészítése, honlapon letölthető**

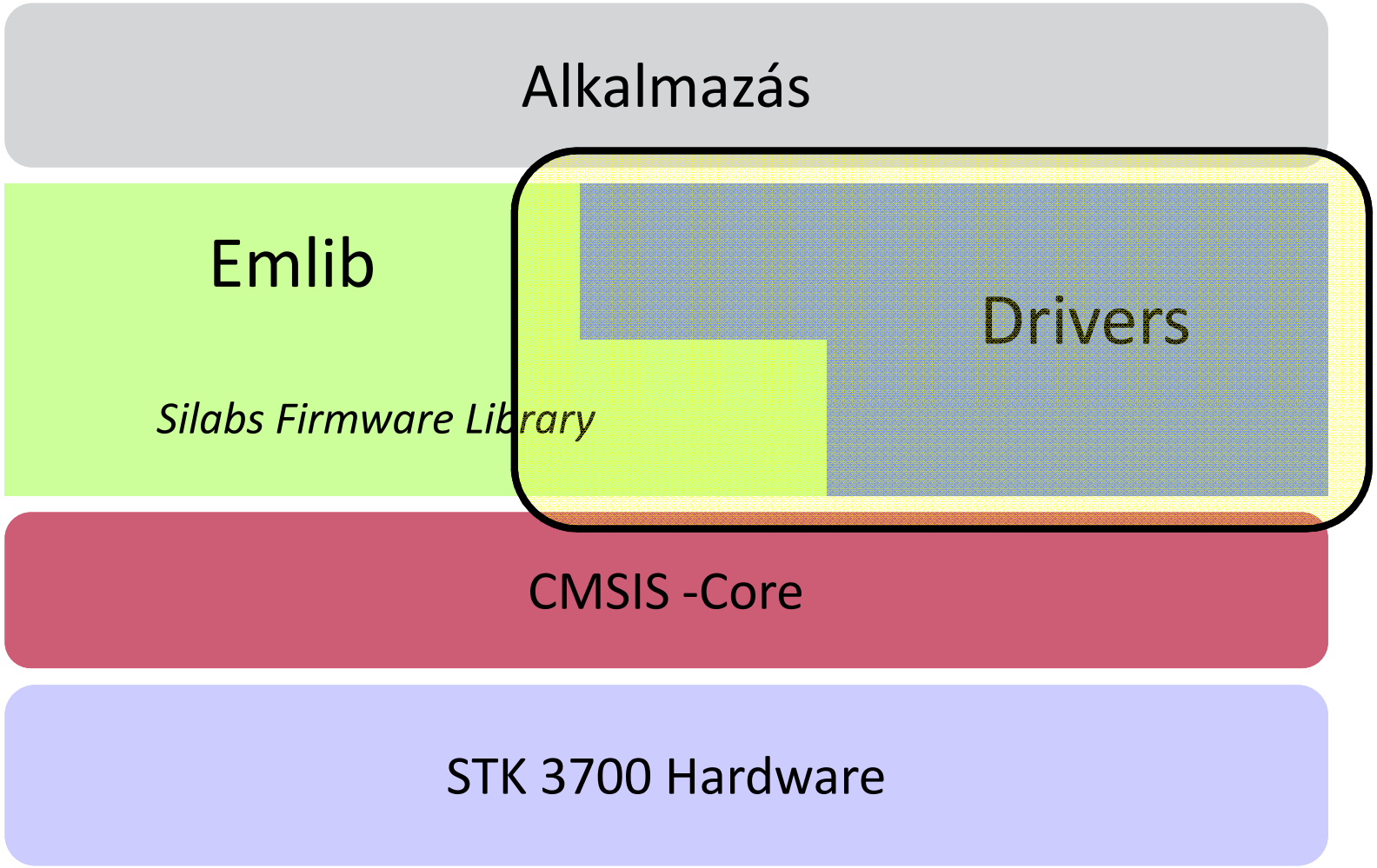
- Standard környezet elég
- **Nincs energy profile, DWT mellette**

Printf használata

- Legalább 2 lehetőség van az stdout használatára



Driver csomag használata



4. feladat: Printf használata: ITM Trace alapon

- Driver könyvtár használja az általunk már kipróbált emf32 könyvtárat
- Driver könyvtárból:
  - A trace csatorna a GPIO Port F, Pin 2-t használja (nem kell külön initelni)
  - ***C:\SiliconLabs\SimplicityStudio\v3\developer\sdks\efm32\v2\kits\common***
    - Vagy bemásolni, vagy Eclipsből beszedni, de akkor törölgetni kell a többi file-t. ***Javasolt a copy.***
  - ***retargetio.c***
  - ***retargetSWO.c***
  - ***retargetSWO.h***
- ***setupSWOForPrint()***
- stdio include kell. **Printf végére \r\n mert nem küldi el !!!**
- Eredmény a Console ablakba jön
- Nincs ilyenkor Energy profile és Interupt trace

4. feladat: megoldás

```
#include <stdio.h>

void UART0_RX_IRQHandler(void)
{
      USART_IntClear (UART0, USART_IF_RXDATAV);
      uint8_t ch = USART_Rx (UART0);
      USART_Tx (UART0, ch);
      printf("IT Uart_Rx:%c \r\n",ch);
}

/**************************************************************
 * @brief  Main function
 **************************************************************/
int main(void)
{
  /* Chip errata */
  CHIP_Init();

  enter_DefaultMode_from_RESET();
  setupSWOForPrint();
  USART_IntEnable (UART0, USART_IF_RXDATAV);
  NVIC_EnableIRQ(UART0_RX_IRQn);
  printf("Hello Silabs \r\n");
```

Console | Memory | Executables

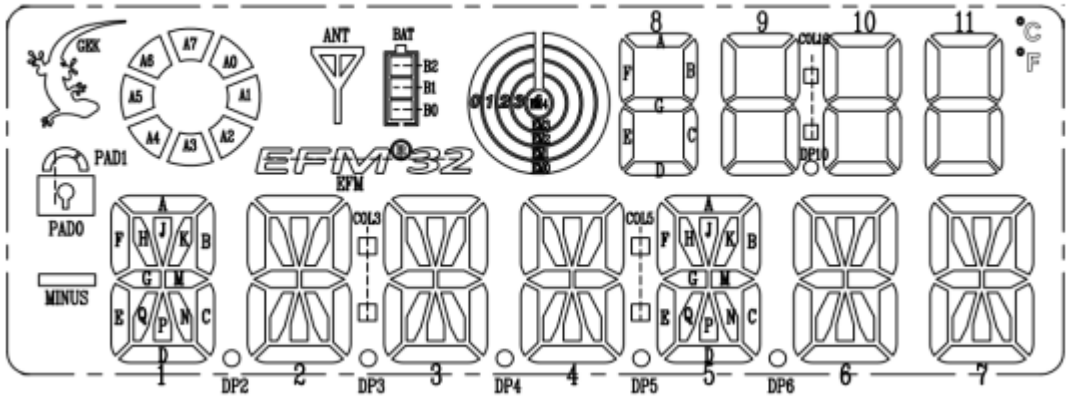
Program Output Console

```
Hello Silabs
IT Uart_Rx:d
IT Uart_Rx:h
```

5. Feladat: LCD kezelés

LCD kijelző kiosztás

Segment placement

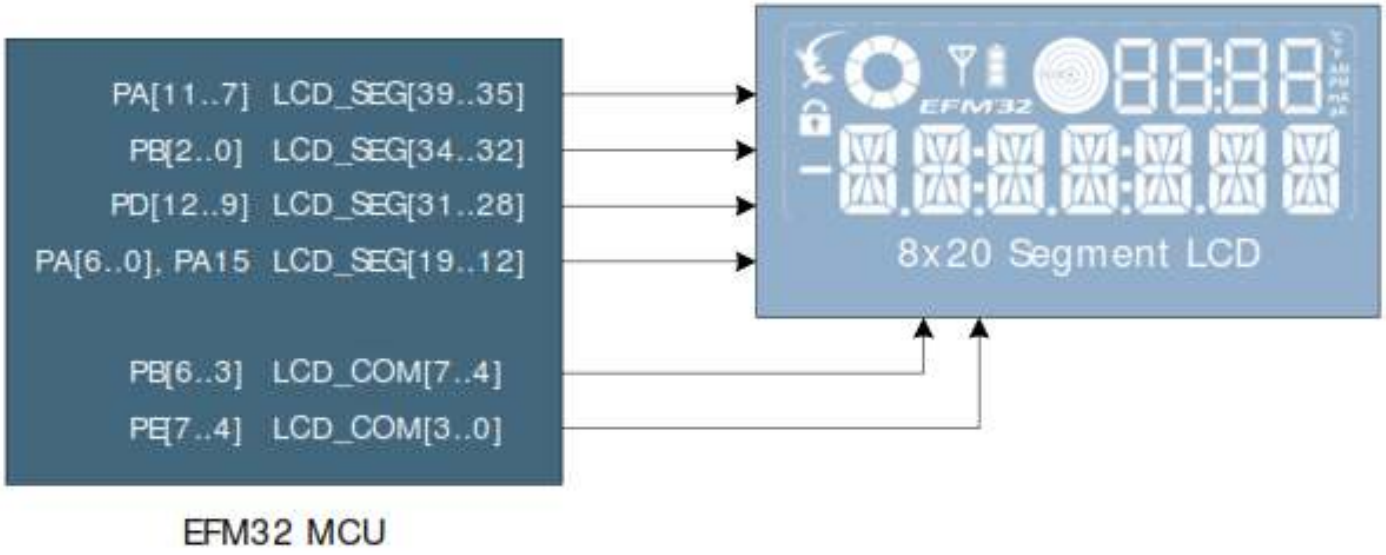


PIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14
——	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
COM0	DP2	1 E	1 D	2 E	2 D	3 E	3 D	4 E	4 D	DP5	5 D	DP6	6 D	7 E
COM1	DP4	1 Q	1 N	2 Q	2 N	3 Q	3 N	4 Q	4 N	5 E	5 N	6 E	6 N	7 Q
COM2	DP3	1 P	1 C	2 P	2 C	3 P	3 C	4 P	4 C	5 Q	5 C	6 Q	6 C	7 P
COM3	COL3	1 G	1 M	2 G	2 M	3 G	3 M	4 G	4 M	5 P	5 M	6 P	6 M	7 G
COM4	MINUS	1 F	1 J	2 F	2 J	3 F	3 J	4 F	4 J	5 G	5 J	6 G	6 J	7 F
COM5	PAD1	1 H	1 K	2 H	2 K	3 H	3 K	4 H	4 K	5 F	5 K	6 F	6 K	7 H
COM6	GEK	1 A	1 B	2 A	2 B	3 A	3 B	4 A	4 B	5 H	5 B	6 H	6 B	7 A
COM7	A7	A6	A5	A4	A3	A2	A1	A0	EFM	5 A	COL5	6 A	ANT	BAT

PIN	15	16	17	18	19	20	21	22	23	24	25	26	27	28
——	S14	S15	S16	S17	S18	S19	COM7	COM6	COM5	COM4	COM3	COM2	COM1	COM0
COM0	7 D	11 A	10 A	9 A	8 A	EM2								COM0
COM1	7 N	11 F	10 F	9 F	8 F	EM4							COM1	
COM2	7 C	11 B	10 B	9 B	8 B	COL10						COM2		
COM3	7 M	11 G	10 G	9 G	8 G	DP10					COM3			
COM4	7 J	11 E	10 E	9 E	8 E	PAD0				COM4				
COM5	7 K	11 C	10 C	9 C	8 C	EM3			COM5					
COM6	7 B	11 D	10 D	9 D	8 D	EM1		COM6						
COM7	°C	°F	B1	B0	B2	EM0	COM7							

5. feladat: LCD kijelző használata

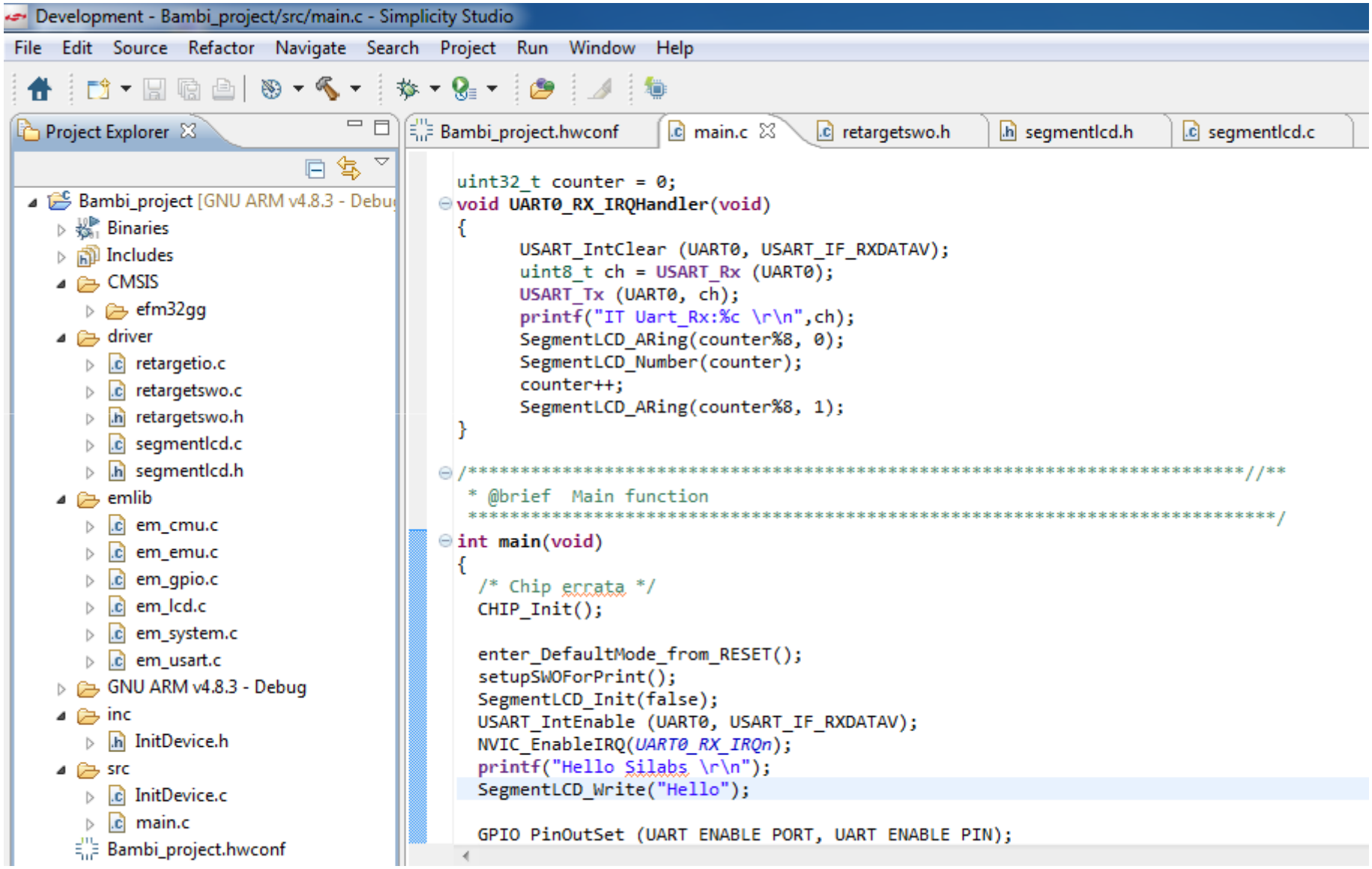
- Beépített energiatakarékos szegmens LCD meghajtó
- Nem kell hozzá konfigurálás, csak file másolás
- Szükséges File-ok (driver és emlib könyvtárak)
  - ***em_lcd.c***
  - ***segmentlcd.c***
  - ***segmentlcd.h***



5. feladat: LCD kijelző használata

- Szegmenses LCD Ininccializáció
  - Include-olni a segmentlcd.h
  - ***SegmentLCD_Init(false)***
- Feladat:
- Statikus üzenet kiírása
  - ***SegmentLCD_Write("Text")***
- Fogadott UART karakterek számolása, kijelzése
  - ***SegmentLCD_Number(int)***
- További kijelző ikonok kipróbálása
  - ***SegmentLCD_Aring***
  - SegmentLCD_Symbol
  - SegmentLCD_Battery

5. feladat: megoldás



Extra feladatok otthoni gyakorlásra

Extra 1.: GPIO IT kezelés

- Configurátorban beállítani a GPIO lábakat: PB9, PB10 (elég az egyiket)
- Megfelelő Firmware library függvények meghívása
  - GPIO_ExtIntConfig
- Vigyázat külön IT engedélyezés függvény az NVIC-ben
  - NVIC_EnableIRQ
- IT kiszolgáló függvény
  - startup_gcc_efm32gg.s –ból a neve
  - IT flag törlés

Extra 2.: Timer 0 IT programozás feladat

- Periódikus IT 1ms-enként ami növel egy számlálót
- Nyomógomb hatására kiírni a számláló értékét, az LCD-re, aki akar csinálhat reflexidő mérőt