Timer, interrupt

MicLab-03

Név:

Dátum:

Mérőhely:

# Bevezetés

Az interrupt használatának megismerése.

## Ajánlott irodalom

<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.

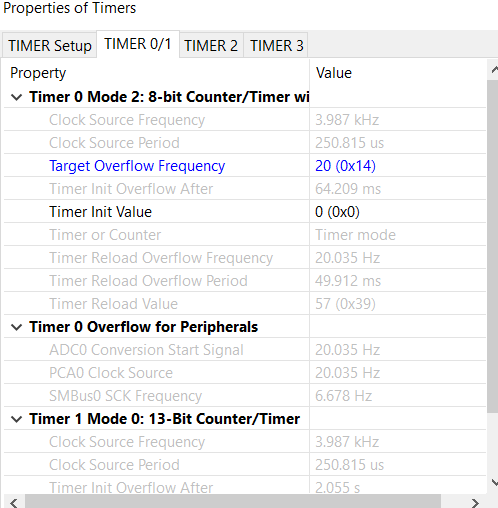
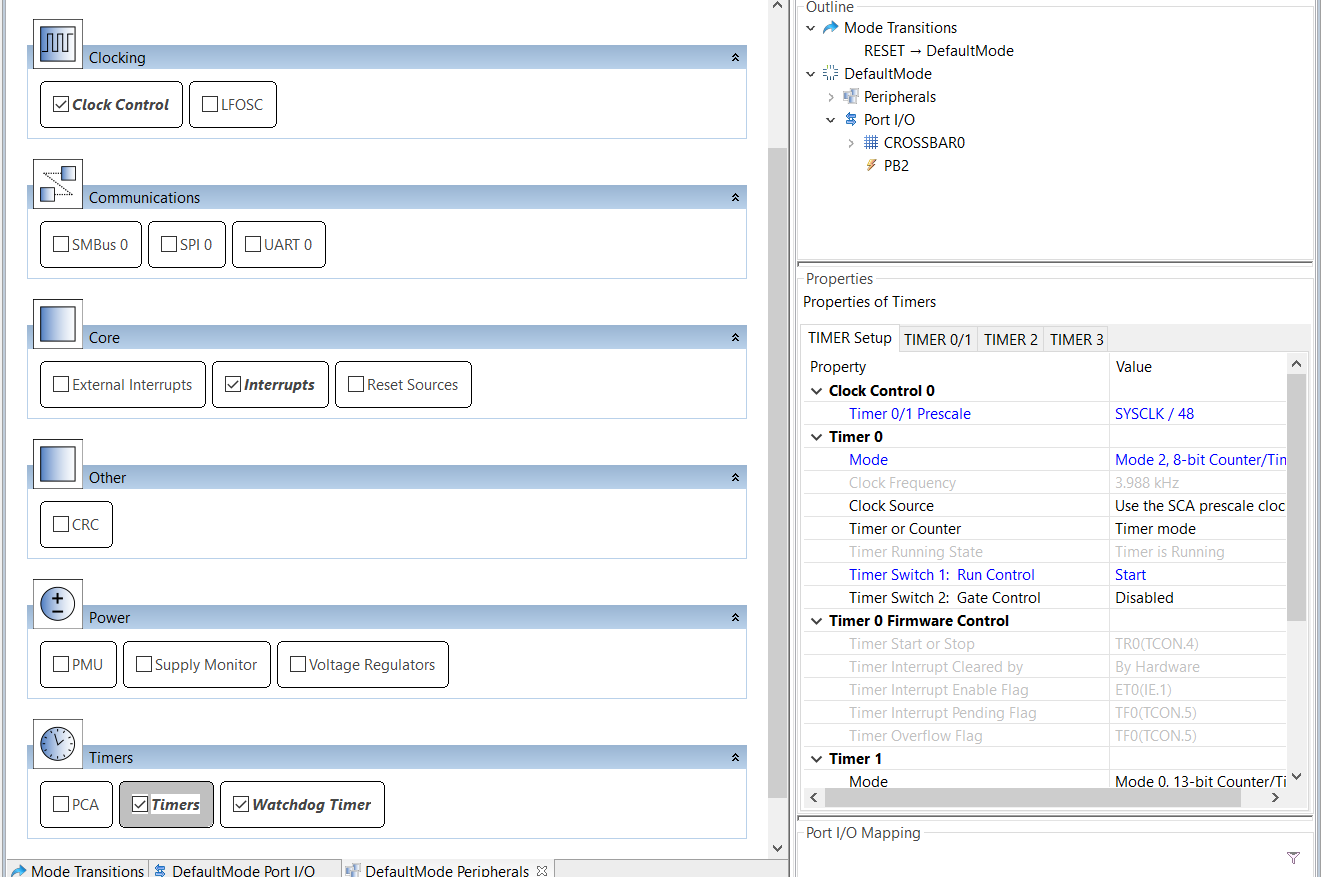
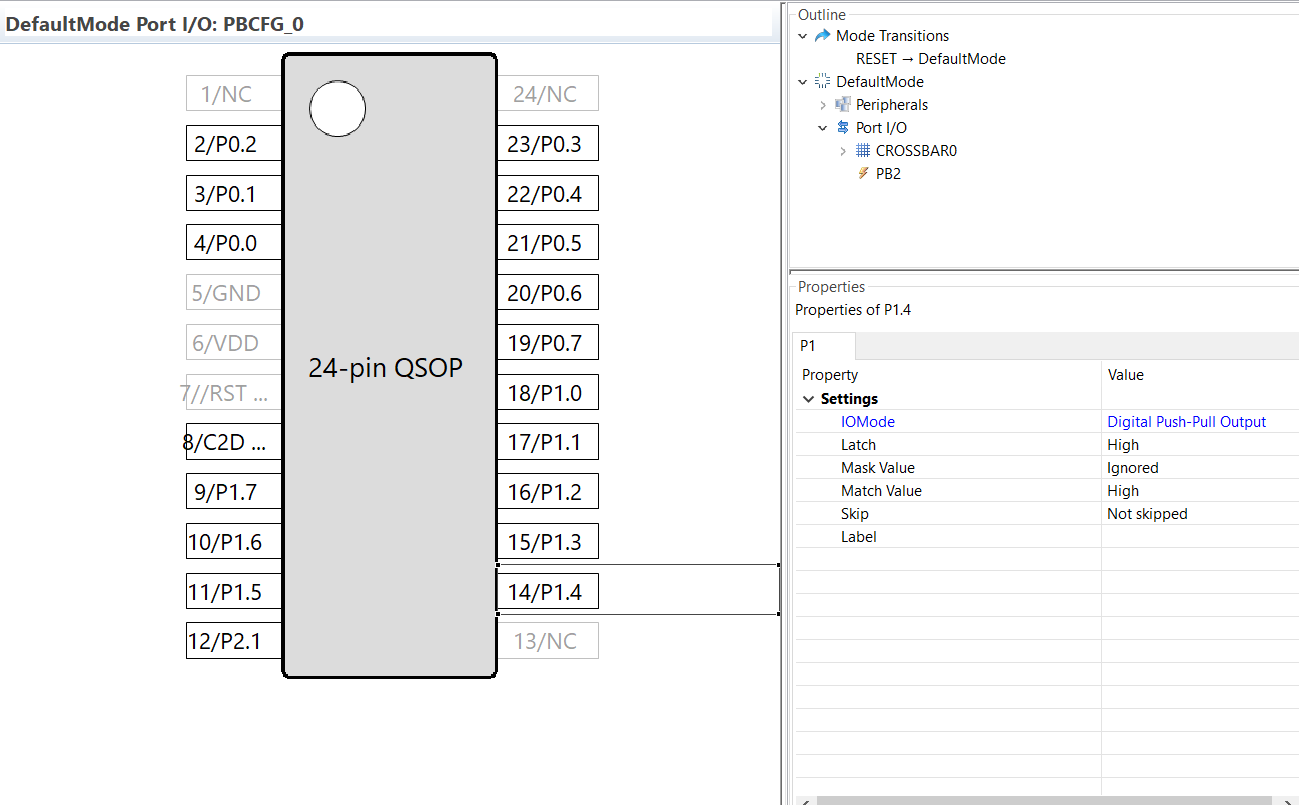
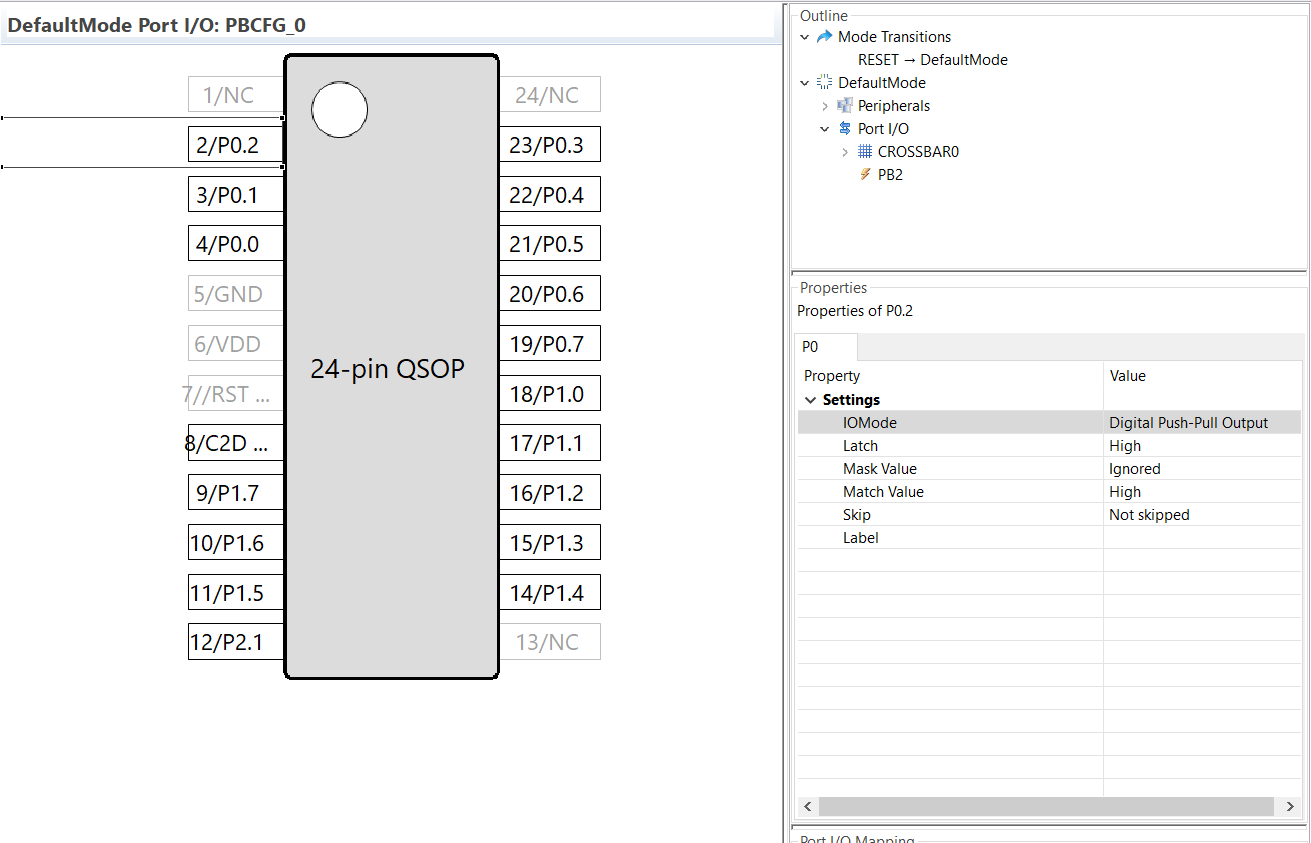
A jegyzőkönyveket a CooSpace-en kell feltölteni, külön pdf formátumban csatolni kell a jegyzőkönyvet (a fájl neve a következő mintát kövesse: NagyJ.KissB.03.pdf), egy külön zip fájlban pedig a kódokat (\*.c, \*.cwg). Amennyiben probléma merül fel a beadás során, az anyagokat az oktató e-mail címére kell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

# 1. feladat – LED vezérlése interrupttal

Írjon egy programot, mely a LED0-ás LED-et 5 Hz, 20 Hz és 100 Hz frekvenciával villogtatja. A vezérlési feladatot a Timer 0 interrupt rutin lássa el, a villogási frekvenciák között a BTN0-ás nyomógomb segítségével lehessen váltani. A nyomógombot pergés mentesíteni is szükséges.

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

Config:

Watchdog timer off, crossbar on, interrupts on, timer 0 interrupt on.

01\_main.c

//=========================================================

// src/01\_main.c: generated by Hardware Configurator

//

// This file will be updated when saving a document.

// leave the sections inside the "$[...]" comment tags alone

// or they will be overwritten!!

//=========================================================

//-----------------------------------------------------------------------------

// Includes

//-----------------------------------------------------------------------------

**#include** <SI\_EFM8BB1\_Register\_Enums.h> // SFR declarations

**#include** "InitDevice.h"

**#include** "ButtonModes.h"

**#include** "IncludesBasics.h"

**extern** uint8\_t rattle\_counter;

**volatile** **char** delayer = 0;

BTN\_MODE BTN0\_mode = BTN0\_MODE\_5HZ;

// $[Generated Includes]

// [Generated Includes]$

//-----------------------------------------------------------------------------

// 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**)

{

// $[SiLabs Startup]

// [SiLabs Startup]$

}

//-----------------------------------------------------------------------------

// main() Routine

// ----------------------------------------------------------------------------

**int** **main** (**void**)

{

uint8\_t rattle\_counter\_ticks = RATTLE\_COUNTER\_5HZ;

// Call hardware initialization routine

enter\_DefaultMode\_from\_RESET();

**while** (1)

{

// $[Generated Run-time code]

// [Generated Run-time code]$

// Perrgésmentesítés és timer reload freq. változtatás.

**if**(BTN0 == ON) {

rattle\_counter = 0;

**while**(rattle\_counter <= rattle\_counter\_ticks);

**if** (BTN0\_mode == BTN0\_MODE\_100HZ) BTN0\_mode = BTN0\_MODE\_5HZ;

**else** BTN0\_mode++;

BTN0\_set\_TIMER01\_0\_TH0\_val(BTN0\_mode);

**while**(BTN0 == ON);

**while**(rattle\_counter < rattle\_counter\_ticks);

}

// Perrgésmentesítési idő megváltoztatása.

**switch** (BTN0\_mode) {

**case** BTN0\_MODE\_5HZ:

rattle\_counter\_ticks = RATTLE\_COUNTER\_5HZ;

**break**;

**case** BTN0\_MODE\_20HZ:

rattle\_counter\_ticks = RATTLE\_COUNTER\_20HZ;

**break**;

**case** BTN0\_MODE\_100HZ:

rattle\_counter\_ticks = RATTLE\_COUNTER\_100HZ;

**break**;

}

}

}

ButtonModes.c

**#include** "ButtonModes.h"

**#include** <SI\_EFM8BB1\_Register\_Enums.h>

**extern** **char** delayer;

**void** **BTN0\_set\_TIMER01\_0\_TH0\_val**(BTN\_MODE mode) {

**switch** (mode) {

**case** BTN0\_MODE\_5HZ :

delayer = 0;

**while** (delayer < 2) delayer++;

TH0 = (TM0\_VAL\_5HZ << TH0\_TH0\_\_SHIFT);

delayer = 0;

**break**;

**case** BTN0\_MODE\_20HZ :

TH0 = (TM0\_VAL\_20HZ << TH0\_TH0\_\_SHIFT);

**break**;

**case** BTN0\_MODE\_100HZ :

TH0 = (TM0\_VAL\_100HZ << TH0\_TH0\_\_SHIFT);

**break**;

}

}

ButtonModes.h

**#ifndef** \_\_BUTTON\_MODES\_H\_\_

**#define** \_\_BUTTON\_MODES\_H\_\_

**#include** "IncludesBasics.h"

**typedef** **char** BTN\_MODE;

**typedef** **unsigned** **char** uchar;

**#define** BTN0\_MODE\_5HZ (BTN\_MODE)0

**#define** BTN0\_MODE\_20HZ (BTN\_MODE)1

**#define** BTN0\_MODE\_100HZ (BTN\_MODE)2

**#define** TM0\_VAL\_5HZ (uchar)0x39

**#define** TM0\_VAL\_20HZ (uchar)0x9C

**#define** TM0\_VAL\_100HZ (uchar)0xEC

**#define** RATTLE\_COUNTER\_5HZ (uint8\_t)5

**#define** RATTLE\_COUNTER\_20HZ (uint8\_t)5\*(TM0\_VAL\_20HZ/TM0\_VAL\_5HZ)

**#define** RATTLE\_COUNTER\_100HZ (uint8\_t)5\*(TM0\_VAL\_100HZ/TM0\_VAL\_20HZ)

**void** **BTN0\_set\_TIMER01\_0\_TH0\_val**(BTN\_MODE mode);

**#endif**

Interrupts.c

//=========================================================

// src/Interrupts.c: generated by Hardware Configurator

//

// This file will be regenerated when saving a document.

// leave the sections inside the "$[...]" comment tags alone

// or they will be overwritten!

//=========================================================

// USER INCLUDES

**#include** <SI\_EFM8BB1\_Register\_Enums.h>

**#include** "IncludesBasics.h"

**volatile** uint8\_t rattle\_counter = 0;

//-----------------------------------------------------------------------------

// TIMER0\_ISR

//-----------------------------------------------------------------------------

//

// TIMER0 ISR Content goes here. Remember to clear flag bits:

// TCON::TF0 (Timer 0 Overflow Flag)

//

//-----------------------------------------------------------------------------

SI\_INTERRUPT (**TIMER0\_ISR**, TIMER0\_IRQn)

{

TCON\_TF0 = 0;

LED0 = !LED0;

rattle\_counter++;

}

InitDevice.c

//=========================================================

// src/InitDevice.c: generated by Hardware Configurator

//

// This file will be regenerated when saving a document.

// leave the sections inside the "$[...]" comment tags alone

// or they will be overwritten!

//=========================================================

// USER INCLUDES

**#include** <SI\_EFM8BB1\_Register\_Enums.h>

**#include** "InitDevice.h"

// USER PROTOTYPES

// USER FUNCTIONS

// $[Library Includes]

// [Library Includes]$

//==============================================================================

// enter\_DefaultMode\_from\_RESET

//==============================================================================

**extern** **void**

**enter\_DefaultMode\_from\_RESET** (**void**)

{

// $[Config Calls]

WDT\_0\_enter\_DefaultMode\_from\_RESET ();

PORTS\_0\_enter\_DefaultMode\_from\_RESET ();

PORTS\_1\_enter\_DefaultMode\_from\_RESET ();

PBCFG\_0\_enter\_DefaultMode\_from\_RESET ();

CLOCK\_0\_enter\_DefaultMode\_from\_RESET ();

TIMER01\_0\_enter\_DefaultMode\_from\_RESET ();

TIMER\_SETUP\_0\_enter\_DefaultMode\_from\_RESET ();

INTERRUPT\_0\_enter\_DefaultMode\_from\_RESET ();

// [Config Calls]$

}

**extern** **void**

**WDT\_0\_enter\_DefaultMode\_from\_RESET** (**void**)

{

// $[Watchdog Timer Init Variable Declarations]

uint32\_t i;

bool ea;

// [Watchdog Timer Init Variable Declarations]$

// $[WDTCN - Watchdog Timer Control]

// Deprecated

// [WDTCN - Watchdog Timer Control]$

// $[WDTCN\_2 - Watchdog Timer Control]

// Feed WDT timer before disabling (Erratum WDT\_E102)

WDTCN = 0xA5;

// Add 2 LFO cycle delay before disabling WDT (Erratum WDT\_E102)

**for** (i = 0; i < (2 \* 3062500UL) / (10000 \* 3); i++)

{

NOP ();

}

// Disable WDT

ea = IE\_EA;

IE\_EA = 0;

WDTCN = 0xDE;

WDTCN = 0xAD;

IE\_EA = ea;

// [WDTCN\_2 - Watchdog Timer Control]$

}

**extern** **void**

**PBCFG\_0\_enter\_DefaultMode\_from\_RESET** (**void**)

{

// $[XBR2 - Port I/O Crossbar 2]

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- Weak Pullups enabled

- Crossbar enabled

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

XBR2 = XBR2\_WEAKPUD\_\_PULL\_UPS\_ENABLED | XBR2\_XBARE\_\_ENABLED;

// [XBR2 - Port I/O Crossbar 2]$

// $[PRTDRV - Port Drive Strength]

// [PRTDRV - Port Drive Strength]$

// $[XBR0 - Port I/O Crossbar 0]

// [XBR0 - Port I/O Crossbar 0]$

// $[XBR1 - Port I/O Crossbar 1]

// [XBR1 - Port I/O Crossbar 1]$

}

**extern** **void**

**CLOCK\_0\_enter\_DefaultMode\_from\_RESET** (**void**)

{

// $[CLKSEL - Clock Select]

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- Clock derived from the Internal High-Frequency Oscillator

- SYSCLK is equal to selected clock source divided by 128

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CLKSEL = CLKSEL\_CLKSL\_\_HFOSC | CLKSEL\_CLKDIV\_\_SYSCLK\_DIV\_128;

// [CLKSEL - Clock Select]$

}

**extern** **void**

**TIMER\_SETUP\_0\_enter\_DefaultMode\_from\_RESET** (**void**)

{

// $[CKCON0 - Clock Control 0]

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- System clock divided by 48

- Counter/Timer 0 uses the clock defined by the prescale field, SCA

- Timer 2 high byte uses the clock defined by T2XCLK in TMR2CN0

- Timer 2 low byte uses the clock defined by T2XCLK in TMR2CN0

- Timer 3 high byte uses the clock defined by T3XCLK in TMR3CN0

- Timer 3 low byte uses the clock defined by T3XCLK in TMR3CN0

- Timer 1 uses the clock defined by the prescale field, SCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CKCON0 = CKCON0\_SCA\_\_SYSCLK\_DIV\_48 | CKCON0\_T0M\_\_PRESCALE

| CKCON0\_T2MH\_\_EXTERNAL\_CLOCK | CKCON0\_T2ML\_\_EXTERNAL\_CLOCK

| CKCON0\_T3MH\_\_EXTERNAL\_CLOCK | CKCON0\_T3ML\_\_EXTERNAL\_CLOCK

| CKCON0\_T1M\_\_PRESCALE;

// [CKCON0 - Clock Control 0]$

// $[TMOD - Timer 0/1 Mode]

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- Mode 2, 8-bit Counter/Timer with Auto-Reload

- Mode 0, 13-bit Counter/Timer

- Timer Mode

- Timer 0 enabled when TR0 = 1 irrespective of INT0 logic level

- Timer Mode

- Timer 1 enabled when TR1 = 1 irrespective of INT1 logic level

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

TMOD = TMOD\_T0M\_\_MODE2 | TMOD\_T1M\_\_MODE0 | TMOD\_CT0\_\_TIMER

| TMOD\_GATE0\_\_DISABLED | TMOD\_CT1\_\_TIMER | TMOD\_GATE1\_\_DISABLED;

// [TMOD - Timer 0/1 Mode]$

// $[TCON - Timer 0/1 Control]

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- Start Timer 0 running

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

TCON |= TCON\_TR0\_\_RUN;

// [TCON - Timer 0/1 Control]$

}

**extern** **void**

**INTERRUPT\_0\_enter\_DefaultMode\_from\_RESET** (**void**)

{

// $[EIE1 - Extended Interrupt Enable 1]

// [EIE1 - Extended Interrupt Enable 1]$

// $[EIP1 - Extended Interrupt Priority 1]

// [EIP1 - Extended Interrupt Priority 1]$

// $[IE - Interrupt Enable]

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- Enable each interrupt according to its individual mask setting

- Disable external interrupt 0

- Disable external interrupt 1

- Disable all SPI0 interrupts

- Enable interrupt requests generated by the TF0 flag

- Disable all Timer 1 interrupt

- Disable Timer 2 interrupt

- Disable UART0 interrupt

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

IE = IE\_EA\_\_ENABLED | IE\_EX0\_\_DISABLED | IE\_EX1\_\_DISABLED | IE\_ESPI0\_\_DISABLED

| IE\_ET0\_\_ENABLED | IE\_ET1\_\_DISABLED | IE\_ET2\_\_DISABLED

| IE\_ES0\_\_DISABLED;

// [IE - Interrupt Enable]$

// $[IP - Interrupt Priority]

// [IP - Interrupt Priority]$

}

**extern** **void**

**TIMER01\_0\_enter\_DefaultMode\_from\_RESET** (**void**)

{

// $[Timer Initialization]

//Save Timer Configuration

uint8\_t TCON\_save;

TCON\_save = TCON;

//Stop Timers

TCON &= ~TCON\_TR0\_\_BMASK & ~TCON\_TR1\_\_BMASK;

// [Timer Initialization]$

// $[TH0 - Timer 0 High Byte]

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- Timer 0 High Byte = 0x39

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

TH0 = (0x39 << TH0\_TH0\_\_SHIFT);

// [TH0 - Timer 0 High Byte]$

// $[TL0 - Timer 0 Low Byte]

// [TL0 - Timer 0 Low Byte]$

// $[TH1 - Timer 1 High Byte]

// [TH1 - Timer 1 High Byte]$

// $[TL1 - Timer 1 Low Byte]

// [TL1 - Timer 1 Low Byte]$

// $[Timer Restoration]

//Restore Timer Configuration

TCON |= (TCON\_save & TCON\_TR0\_\_BMASK) | (TCON\_save & TCON\_TR1\_\_BMASK);

// [Timer Restoration]$

}

**extern** **void**

**PORTS\_0\_enter\_DefaultMode\_from\_RESET** (**void**)

{

// $[P0 - Port 0 Pin Latch]

// [P0 - Port 0 Pin Latch]$

// $[P0MDOUT - Port 0 Output Mode]

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- P0.0 output is open-drain

- P0.1 output is open-drain

- P0.2 output is push-pull

- P0.3 output is open-drain

- P0.4 output is open-drain

- P0.5 output is open-drain

- P0.6 output is open-drain

- P0.7 output is open-drain

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

P0MDOUT = P0MDOUT\_B0\_\_OPEN\_DRAIN | P0MDOUT\_B1\_\_OPEN\_DRAIN

| P0MDOUT\_B2\_\_PUSH\_PULL | P0MDOUT\_B3\_\_OPEN\_DRAIN | P0MDOUT\_B4\_\_OPEN\_DRAIN

| P0MDOUT\_B5\_\_OPEN\_DRAIN | P0MDOUT\_B6\_\_OPEN\_DRAIN

| P0MDOUT\_B7\_\_OPEN\_DRAIN;

// [P0MDOUT - Port 0 Output Mode]$

// $[P0MDIN - Port 0 Input Mode]

// [P0MDIN - Port 0 Input Mode]$

// $[P0SKIP - Port 0 Skip]

// [P0SKIP - Port 0 Skip]$

// $[P0MASK - Port 0 Mask]

// [P0MASK - Port 0 Mask]$

// $[P0MAT - Port 0 Match]

// [P0MAT - Port 0 Match]$

}

**extern** **void**

**PORTS\_1\_enter\_DefaultMode\_from\_RESET** (**void**)

{

// $[P1 - Port 1 Pin Latch]

// [P1 - Port 1 Pin Latch]$

// $[P1MDOUT - Port 1 Output Mode]

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- P1.0 output is open-drain

- P1.1 output is open-drain

- P1.2 output is open-drain

- P1.3 output is open-drain

- P1.4 output is push-pull

- P1.5 output is open-drain

- P1.6 output is open-drain

- P1.7 output is open-drain

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

P1MDOUT = P1MDOUT\_B0\_\_OPEN\_DRAIN | P1MDOUT\_B1\_\_OPEN\_DRAIN

| P1MDOUT\_B2\_\_OPEN\_DRAIN | P1MDOUT\_B3\_\_OPEN\_DRAIN | P1MDOUT\_B4\_\_PUSH\_PULL

| P1MDOUT\_B5\_\_OPEN\_DRAIN | P1MDOUT\_B6\_\_OPEN\_DRAIN

| P1MDOUT\_B7\_\_OPEN\_DRAIN;

// [P1MDOUT - Port 1 Output Mode]$

// $[P1MDIN - Port 1 Input Mode]

// [P1MDIN - Port 1 Input Mode]$

// $[P1SKIP - Port 1 Skip]

// [P1SKIP - Port 1 Skip]$

// $[P1MASK - Port 1 Mask]

// [P1MASK - Port 1 Mask]$

// $[P1MAT - Port 1 Match]

// [P1MAT - Port 1 Match]$

}

Az elkészült programot be kell mutatni!

A gyakorlatvezető ellenőrizte:

* Igen
* Nem

A program működött:

* Igen
* Nem

# 2. feladat – Reakcióidó mérése

Írjon egy programot, ami mérni tudja a felhasználó reakcióidejét. Az idő méréséhez a Timer0-át használja interrupt módban, az időmérés felbontása 1 ms legyen. A működés leírása: az indulás után valamennyi késleltetéssel felkapcsol a LED0, ami után a felhasználónak minél rövidebb időn belül meg kell nyomnia a BTN0-át. Ha a 200 ms-on belül sikerül megnyomnia, akkor a LED0 lekapcsol. (Ügyeljen a megfelelő változóvédelemre és arra, hogy a LED0 ne kapcsoljon fel újra, ha már megtörtént a mérés és az belül volt a 200 ms-on.)

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

Az elkészült programot be kell mutatni!

A gyakorlatvezető ellenőrizte:

* Igen
* Nem

A program működött:

* Igen
* Nem

# Megjegyzések