Timer, interrupt

MICLAB-03

Név: Pilter Zsófia, Vad Avar

Dátum: 2024.09.30.

Mérőhely: 1 jobb és bal

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 – Reakcióidő 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:

- indulás után valamennyi késleltetéssel felkapcsol a LED0
- 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):

Config:

```
<?xml version="1.0" encoding="ASCII"?>
<device:XMLDevice xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:device="http://www.silabs.com/ss/hwconfig/document/device.ecore"
name="EFM8BB10F8G-A-QSOP24" partId="mcu.8051.efm8.bb1.efm8bb10f8g-a-qsop24"
version="4.0.0" contextId="%DEFAULT%">
  <mode name="DefaultMode">
```

```
    <property object="DefaultMode" propertyId="mode.diagramLocation" value="100,
100"/>
    <property object="INTERRUPT_0" propertyId="ABPeripheral.included"
value="true"/>
    <property object="INTERRUPT_0"
propertyId="interrupt.interruptenable.enableallinterrupts" value="Enabled"/>
    <property object="INTERRUPT_0"
propertyId="interrupt.interruptenable.enabletimer0interrupt" value="Enabled"/>
    <property object="P1.4" propertyId="ports.settings.iomode" value="Digital
Push-Pull Output"/>
    <property object="P1.4" propertyId="ports.settings.outputmode" value="Push-
pull"/>
    <property object="PBCFG_0" propertyId="pbcfg.settings.enablecrossbar"
value="Enabled"/>
    <property object="TIMER01_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="TIMER01_0"
propertyId="timer01.timer0highbyte.timer0highbyte" value="1"/>
    <property object="TIMER01_0"
propertyId="timer01.timer0mode2:8bitcountertimerwithautoreload.targetoverflowfrequ
ency" value="1000"/>
    <property object="TIMER01_0"
propertyId="timer01.timer0mode2:8bitcountertimerwithautoreload.timerreloadvalue"
value="1"/>
    <property object="TIMER16_2" propertyId="ABPeripheral.included" value="true"/>
    <property object="TIMER16_3" propertyId="ABPeripheral.included" value="true"/>
    <property object="TIMER_SETUP_0" propertyId="ABPeripheral.included"
value="true"/>
    <property object="TIMER_SETUP_0" propertyId="timer_setup.timer0.mode"
value="Mode 2, 8-bit Counter/Timer with Auto-Reload"/>
    <property object="TIMER_SETUP_0"
propertyId="timer_setup.timer0.timerrunningstate" value="Timer is Running"/>
    <property object="TIMER_SETUP_0"
propertyId="timer_setup.timer0.timerswitch1:runcontrol" value="Start"/>
    <property object="TIMER_SETUP_0"
propertyId="timer_setup.timer01control.timer0runcontrol" value="Start"/>
    <property object="WDT_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="WDT_0" propertyId="wdt.watchdogcontrol.wdtenable"
value="Disable"/>
    <property object="WDT_0" propertyId="wdt.watchdogcontrol.wdtinitialvalue"
value="5"/>
    <property object="WDT_0" propertyId="wdt.watchdogcontrol.wdtperiodactual"
value="6.554 s"/>
  </mode>
  <modeTransition>
    <property object="RESET &#x2192; DefaultMode"
propertyId="modeTransition.source" value="RESET"/>
    <property object="RESET &#x2192; DefaultMode"
propertyId="modeTransition.target" value="DefaultMode"/>
  </modeTransition>
</device:XMLDevice>
```

Main.c:

```
//=========================================================
// src/feladat03-01_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"
// $[Generated Includes]
// [Generated Includes]$

#define ONBOARD_LED P1_B4
#define ONBOARD_BTN P0_B2
#define LED_OFF 1U

volatile uint16_t counter = 0U;

//-----------------------------------------------------------------------------
// 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)
{
  // Call hardware initialization routine
  enter_DefaultMode_from_RESET();
  ONBOARD_LED = LED_OFF;

  while (1)
  {
    // $[Generated Run-time code]
    // [Generated Run-time code]$
  }
}
```

Interrupts.c:

```
#include <SI_EFM8BB1_Register_Enums.h>

#define ONBOARD_LED P1_B4
#define ONBOARD_BTN P0_B2
#define LEEWAY 200 //ms
#define DELAY 1500 //ms
#define LED_ON 0U
#define LED_OFF 1U
#define BTN_ON 0U
#define BTN_OFF 1U
#define FLAG_OFF 1U
```

```
extern volatile uint16_t counter;
uint8_t off_flag = !FLAG_OFF;



//-----------------------------------------------------------------------------
// 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;
	counter++;

	if(counter > DELAY && !off_flag)
	{

		ONBOARD_LED = LED_ON;

		if(ONBOARD_BTN == BTN_ON && counter < (DELAY + LEEWAY) )
		{
			ONBOARD_LED = LED_OFF;
			off_flag = FLAG_OFF;


		}

	}

}
```

Az elkészült programot be kell mutatni!

A gyakorlatvezető ellenőrizte:

- Igen
- Nem

A program működött:

- Igen
- Nem

2. feladat – LED időzített vezérlése - SOS

Írjon egy programot, mely a LEDo segítségével kiadja az SOS morse kódot. A kód végén várjon 2 másodpercet, majd ismételje meg előről a folyamatot.

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

Config:

```
<?xml version="1.0" encoding="ASCII"?>
```

<device:XMLDevice xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:device="http://www.silabs.com/ss/hwconfig/document/device.ecore"
name="EFM8BB10F8G-A-QSOP24" partId="mcu.8051.efm8.bb1.efm8bb10f8g-a-qsop24"
version="4.0.0" contextId="%DEFAULT%">
  <mode name="DefaultMode">
    <property object="DefaultMode" propertyId="mode.diagramLocation" value="100,
100"/>
    <property object="INTERRUPT_0" propertyId="ABPeripheral.included"
value="true"/>
    <property object="INTERRUPT_0"
propertyId="interrupt.interruptenable.enableallinterrupts" value="Enabled"/>
    <property object="INTERRUPT_0"
propertyId="interrupt.interruptenable.enabletimer0interrupt" value="Enabled"/>
    <property object="P1.4" propertyId="ports.settings.iomode" value="Digital
Push-Pull Output"/>
    <property object="P1.4" propertyId="ports.settings.outputmode" value="Push-
pull"/>
    <property object="PBCFG_0" propertyId="pbcfg.settings.enablecrossbar"
value="Enabled"/>
    <property object="TIMER01_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="TIMER01_0"
propertyId="timer01.timer0highbyte.timer0highbyte" value="1"/>
    <property object="TIMER01_0"
propertyId="timer01.timer0mode2:8bitcountertimerwithautoreload.targetoverflowfrequ
ency" value="1000"/>
    <property object="TIMER01_0"
propertyId="timer01.timer0mode2:8bitcountertimerwithautoreload.timerreloadvalue"
value="1"/>
    <property object="TIMER16_2" propertyId="ABPeripheral.included" value="true"/>
    <property object="TIMER16_2" propertyId="timer16.control.clocksource"
value="SYSCLK"/>
    <property object="TIMER16_2" propertyId="timer16.control.runcontrol"
value="Start"/>
    <property object="TIMER16_2" propertyId="timer16.control.timerrunningstate"
value="Timer is Running"/>
    <property object="TIMER16_2"
propertyId="timer16.initandreloadvalue.targetoverflowfrequency" value="1000"/>
    <property object="TIMER16_2"
propertyId="timer16.initandreloadvalue.timerreloadvalue" value="62474"/>
    <property object="TIMER16_2"
propertyId="timer16.reloadhighbyte.reloadhighbyte" value="244"/>
    <property object="TIMER16_2" propertyId="timer16.reloadlowbyte.reloadlowbyte"
value="10"/>
    <property object="TIMER16_3" propertyId="ABPeripheral.included" value="true"/>
    <property object="TIMER_SETUP_0" propertyId="ABPeripheral.included"
value="true"/>
    <property object="TIMER_SETUP_0"
propertyId="timer_setup.clockcontrol.timer2lowbyteclockselect" value="Use
SYSCLK"/>
    <property object="TIMER_SETUP_0" propertyId="timer_setup.timer0.mode"
value="Mode 2, 8-bit Counter/Timer with Auto-Reload"/>
    <property object="TIMER_SETUP_0"
propertyId="timer_setup.timer0.timerrunningstate" value="Timer is Running"/>
    <property object="TIMER_SETUP_0"
propertyId="timer_setup.timer0.timerswitch1:runcontrol" value="Start"/>
    <property object="TIMER_SETUP_0"
propertyId="timer_setup.timer01control.timer0runcontrol" value="Start"/>
    <property object="WDT_0" propertyId="ABPeripheral.included" value="true"/>

```
    <property object="WDT_0" propertyId="wdt.watchdogcontrol.wdtenable"
value="Disable"/>
    <property object="WDT_0" propertyId="wdt.watchdogcontrol.wdtinitialvalue"
value="5"/>
    <property object="WDT_0" propertyId="wdt.watchdogcontrol.wdtperiodactual"
value="6.554 s"/>
  </mode>
  <modeTransition>
    <property object="RESET &#x2192; DefaultMode"
propertyId="modeTransition.source" value="RESET"/>
    <property object="RESET &#x2192; DefaultMode"
propertyId="modeTransition.target" value="DefaultMode"/>
  </modeTransition>
</device:XMLDevice>
```

Main.c:

```
//=========================================================
// src/feladat03-02_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"
// $[Generated Includes]
// [Generated Includes]$

#define ONBOARD_LED P1_B4
#define LED_ON 0U
#define LED_OFF 1U

volatile uint16_t counter = 0U;
volatile uint8_t state = 0U;

//-------------------------------------------------------------------------------
// 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)
{
  // Call hardware initialization routine
  enter_DefaultMode_from_RESET();
  ONBOARD_LED = LED_OFF;

  while (1)
  {
        switch(state)
        {
              case 0:
              case 2:
              case 4:
              case 6:
              case 7:
              case 8:
              case 10:
              case 11:
              case 12:
              case 14:
              case 15:
              case 16:
              case 18:
              case 20:
              case 22:
                    ONBOARD_LED = LED_ON;
                    break;
              case 1:
              case 3:
              case 5:
              case 9:
              case 13:
              case 17:
              case 19:
              case 21:
              case 23:
              case 24:
              case 25:
              case 26:
              case 27:
              case 28:
              case 29:
              case 30:
                    ONBOARD_LED = LED_OFF;
                    break;



        }

    // $[Generated Run-time code]
    // [Generated Run-time code]$
  }
}
```

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>

#define ONBOARD_LED P1_B4
#define LED_ON 0U
#define LED_OFF 1U
#define DOT 2000/7U
#define MAX_STATE 30

extern volatile uint16_t counter;
extern volatile uint8_t state;


//-----------------------------------------------------------------------------
// 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;

	counter++;
		if(counter > DOT)
		{
			state++;
			counter = 0U;
		}

		if(state>MAX_STATE)
		{
			state=0U;
		}
}
```

Az elkészült programot be kell mutatni!

A gyakorlatvezető ellenőrizte:

- Igen
- Nem

A program működött:

- Igen
- Nem

Megjegyzések