Timer, interrupt

MICLAB-03

Név: Stefán Kornél

Dátum: 2024. 09. 30. 18:00

Mérőhely: 7 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.targetoverflowfrequency" 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"/>
  </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 érdektelen.

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 ONBOARD_BTN P0_B2
#define LED_ENABLE 0
#define LED_DISABLE 1
#define MAGIC_WAIT 3795
#define ALLOWED_DELAY 200
#define COUNTER_RESET 0

enum {
        WAIT_FOR_LED, // Waiting for LED to turn on (counting to 3795ms)
        LED_ON, // LED is on, waiting for button press
        LED_OFF // LED turned off after button press
} State;

extern uint16_t globalCounter = 0;
extern uint16_t globalState = WAIT_FOR_LED;

//-------------------------------------------------------------------------
// TIMER0_ISR
//-------------------------------------------------------------------------
//
// TIMER0 ISR Content goes here. Remember to clear flag bits:
// TCON::TF0 (Timer 0 Overflow Flag)
//
//-------------------------------------------------------------------------
SI_INTERRUPT (TIMER0_ISR, TIMER0_IRQn)
{
        switch (globalState)
        {
                // This is the initial state
```

```
                case WAIT_FOR_LED:
                        globalCounter++;

                        if (globalCounter == MAGIC_WAIT)
                        {
                                globalState = LED_ON;
                                globalCounter = COUNTER_RESET;
                                ONBOARD_LED = LED_ENABLE;
                        }
                        else
                        {
                                ONBOARD_LED = LED_DISABLE;
                        }

                        break;
                // This is the speed testing part
                case LED_ON:
                        if (globalCounter > ALLOWED_DELAY) {
                                break;
                        }

                        globalCounter++;

                        if (!ONBOARD_BTN)
                        {
                                globalState = LED_OFF;
                                ONBOARD_LED = LED_DISABLE;
                        }

                        break;
                // This is the win condition
                case LED_OFF:
                default:
                        break;
        }
}
```

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.enabletimer2interrupt"
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="TIMER16_2" propertyId="ABPeripheral.included" value="true"/>
    <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="5"/>
    <property object="TIMER16_2" propertyId="timer16.initandreloadvalue.timerreloadvalue"
value="14494"/>
    <property object="TIMER16_2" propertyId="timer16.reloadhighbyte.reloadhighbyte" value="56"/>
    <property object="TIMER16_2" propertyId="timer16.reloadlowbyte.reloadlowbyte" value="158"/>
    <property object="TIMER16_3" propertyId="ABPeripheral.included" value="true"/>
    <property object="TIMER_SETUP_0" propertyId="ABPeripheral.included" value="true"/>
  </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-ben nincsen semmi érdekes.

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 LED_ON 0
#define LED_OFF 1
#define FIRST_LETTER 0
#define NO_WAIT 0
#define SHORT_WAIT 1
#define LONG_WAIT 3
#define WORD_WAIT 10
#define ONBOARD_LED P1_B4
#define ONBOARD_BTN P0_B2

enum {
        DOT_ON, DOT_OFF,
        DASH_ON, DASH_OFF,
        LETTER_PAUSE, WORD_PAUSE
};

// Current state
extern uint16_t morseState = DOT_ON;
// Timer for longer operations (DASH)
extern uint16_t morseCounter = 0;
// Where we are in SOS
extern uint16_t letterIndex = 0;

const unsigned char morsePattern[] = {
        DOT_ON, DOT_OFF, DOT_ON, DOT_OFF, DOT_ON, DOT_OFF, // S = ...
        LETTER_PAUSE, // Pause between S and O
        DASH_ON, DASH_OFF, DASH_ON, DASH_OFF, DASH_ON, DASH_OFF, // O = ---
        LETTER_PAUSE, // Pause between O and S
        DOT_ON, DOT_OFF, DOT_ON, DOT_OFF, DOT_ON, DOT_OFF, // S = ...
        WORD_PAUSE // Word pause after SOS 2s
};

//-----------------------------------------------------------------------------
// TIMER2_ISR
```

```
//-----------------------------------------------------------------------------
//
// TIMER2 ISR Content goes here. Remember to clear flag bits:
// TMR2CN0::TF2H (Timer # High Byte Overflow Flag)
// TMR2CN0::TF2L (Timer # Low Byte Overflow Flag)
//
//-----------------------------------------------------------------------------
SI_INTERRUPT (TIMER2_ISR, TIMER2_IRQn)
{
        TMR2CN0_TF2H = 0; // Clear the timer interrupt flag

        switch (morseState)
        {
                case DOT_ON:
                        ONBOARD_LED = LED_ON;
                        morseCounter = NO_WAIT;
                        morseState = DOT_OFF;

                        break;

                case DOT_OFF:
                        ONBOARD_LED = LED_OFF;
                        morseCounter = NO_WAIT;
                        letterIndex++;
                        morseState = morsePattern[letterIndex];

                        break;

                case DASH_ON:
                        ONBOARD_LED = LED_ON;

                        if (morseCounter >= LONG_WAIT)
                        {
                                morseState = DASH_OFF;
                                morseCounter = NO_WAIT;
                        }
                        else
                        {
                                morseCounter += SHORT_WAIT;
                        }

                        break;
                case DASH_OFF:
                        ONBOARD_LED = LED_OFF;
                        morseCounter = NO_WAIT;
                        letterIndex++;
                        morseState = morsePattern[letterIndex];

                        break;
                case LETTER_PAUSE:
                        ONBOARD_LED = LED_OFF;

                        if (morseCounter >= LONG_WAIT)
                        {
                                morseCounter = NO_WAIT;
                                letterIndex++;
                                morseState = morsePattern[letterIndex];
                        }
                        else
                        {
                                morseCounter += SHORT_WAIT;
                        }

                        break;
                case WORD_PAUSE:
                        ONBOARD_LED = LED_OFF;

                        if (morseCounter >= WORD_PAUSE)
                        {
                                morseCounter = NO_WAIT;
                                letterIndex = FIRST_LETTER;
                                morseState = morsePattern[letterIndex];
                        }
                        else
```

```
                        {
                                morseCounter += SHORT_WAIT;
                        }

                        break;
                default:
                        break;
        }
}
```

Az elkészült programot be kell mutatni!

A gyakorlatvezető ellenőrizte:

- Igen
- Nem

A program működött:

- Igen
- Nem

Megjegyzések

A laboron megbeszéltük, hogy a második feladat kódja hosszú (129), de ez itt speciális engedéllyel lehet hosszú.