ZH1

MICLAB-ZH1

Név: Stefán Kornél

Dátum: 2024. 11. 04 18:00

Mérőhely: 7 bal

A dolgozat megírása során csak a következő eszközök használhatók: toll, ceruza, radír, feladatlap, Simplicity Studio, Excel, Windows számológép, az Asztalon lévő Vizsgaanyag mappa tartalma. Más NEM!

Az Asztalon létre kell hozni egy munkakönyvtárat vezetéknévvel és a keresztnév első betűjével (pl. KissJ). Az elkészült forráskódot és a kitöltött jegyzőkönyvet pdf-ben az Asztalon kell hagyni a munkakönyvtárban.

1. feladat – Potenciométer feszültségének mérése

Az ADC segítségével mérje a kiegészítő panelen lévő potenciométer kimenetét. Az ADC-t interrupt módban használja, 50 Hz mintavételi rátával, a Vref legyen az Unregulated VDD (3,3 V). A system clock maradjon a default 3,062 MHz értéken és az SARCLK is ez az érték legyen. Az ADC-t 10 bites módban használja és az adat legyen jobbra igazítva. A megszakításkezelő függvényben csak az ADC adatának változóba mentése történjen, valamint egy saját változóval jelezze, a főprogramnak, hogy történt egy mérés. A főprogramban, ha történt egy mérés, akkor olvassa ki az ADC adatot és a mért feszültség mV egységben legyen eltárolva egy változóban.

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="ADC_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="ADC_0" propertyId="adc.configuration.gaincontrol" value="1x gain"/>
    <property object="ADC_0" propertyId="adc.configuration.sarclockdivider" value="1"/>
    <property object="ADC_0" propertyId="adc.control.enableadc" value="Enabled"/>
    <property object="ADC_0" propertyId="adc.control.startofconversion" value="Timer 2 overflow"/>
    <property object="ADC_0" propertyId="adc.multiplexerselection.positiveinputselection"
value="ADC0.15 (P1.7)"/>
    <property object="ADC_0" propertyId="adc.view.view" value="Advanced"/>
    <property object="DefaultMode" propertyId="mode.diagramLocation" value="100, 100"/>
    <property object="INTERRUPT_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="INTERRUPT_0"
propertyId="interrupt.extendedinterruptenable1.enableadc0conversioncompleteinterrupt" value="Enabled"/>
    <property object="INTERRUPT_0" propertyId="interrupt.interruptenable.enableallinterrupts"
value="Enabled"/>
    <property object="P1.7" propertyId="ports.settings.inputmode" value="Analog"/>
    <property object="P1.7" propertyId="ports.settings.iomode" value="Analog I/O"/>
```

```
    <property object="P1.7" propertyId="ports.settings.skip" value="Skipped"/>
    <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="50"/>
    <property object="TIMER16_2" propertyId="timer16.initandreloadvalue.timerreloadvalue"
value="60432"/>
    <property object="TIMER16_2" propertyId="timer16.reloadhighbyte.reloadhighbyte" value="236"/>
    <property object="TIMER16_2" propertyId="timer16.reloadlowbyte.reloadlowbyte" value="16"/>
    <property object="TIMER16_3" propertyId="ABPeripheral.included" value="true"/>
    <property object="TIMER_SETUP_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="VREF_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="VREF_0" propertyId="vref.hidden.voltagereferenceselect" value="VDD pin"/>
    <property object="VREF_0" propertyId="vref.voltagereferencecontrol.selectvoltagereference"
value="Unregulated VDD"/>
    <property object="WDT_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="WDT_0" propertyId="wdt.watchdogcontrol.wdtenable" value="Disable"/>
  </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

```c
//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------
#include <SI_EFM8BB1_Register_Enums.h>                  // SFR declarations
#include "InitDevice.h"
// $[Generated Includes]
// [Generated Includes]$

#define ADC_RESOLUTION 1024.0f
#define ADC_VOLTAGE 3300u

enum
{
        Read, Unread,
};

volatile uint16_t adc_value = 0;
volatile uint8_t adc_status = Read;

static uint16_t local_adc_value = 0;
static uint16_t voltage_mv = 0;

void SiLabs_Startup(void)
{
        // $[SiLabs Startup]
        // [SiLabs Startup]$
}

//-----------------------------------------------------------------------------
// main() Routine
// ----------------------------------------------------------------------------
int main(void)
{
        // Call hardware initialization routine
        enter_DefaultMode_from_RESET();

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

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

                if (adc_status == Unread) {
                        IE_EA = 0;
                        local_adc_value = adc_value;
                        IE_EA = 1;

                        voltage_mv = adc_value * (ADC_VOLTAGE / ADC_RESOLUTION);

                        adc_status = Read;
                }
        }
}
```

Interrupts.c

```
// USER INCLUDES
#include <SI_EFM8BB1_Register_Enums.h>

enum
{
        Read, Unread,
};

extern volatile uint16_t adc_value;
extern volatile uint8_t adc_status;

//-----------------------------------------------------------------------------
// ADC0EOC_ISR
//-----------------------------------------------------------------------------
//
// ADC0EOC ISR Content goes here. Remember to clear flag bits:
// ADC0CN0::ADINT (Conversion Complete Interrupt Flag)
//
//-----------------------------------------------------------------------------
SI_INTERRUPT (ADC0EOC_ISR, ADC0EOC_IRQn)
{
        ADC0CN0_ADINT = 0;

        adc_value = ADC0;
        adc_status = Unread;
}
```

2. feladat – Potenciométer feszültségének mérése, megjelenítése kijelzőn

Az eredményt mV egységben, folyamatosan frissülő módon jelenítse meg a kijelzőn.

A Display-spi.c fájlban lévő DecoderInit() és WriteDisplayDigit() függvények használatával a kijelző vezérlése teljes mértékben megoldott. A Display-spi.h header include-olása szükséges a main.c elején (a meglévők include-ok alatt) és be kell másolni a két fájlt az „src” és az „inc” mappákba. Az SPI, a kijelző vezérléséhez szükséges **port kimenetek konfigurálása**, a **számjegyekre bontás**, a **digitek léptetése** és a **léptetés időzítésének megírása viszont a feladat része**. Az időzítés egy timer overflow flagjének polling módban történő figyelésével valósítható meg legegyszerűbben. A ciklusidő, azaz a digitek kapcsolása között eltelt idő, legyen 1 ms. (A Timer 3-at most nem célszerű használni).

Egészítse ki a feladatot egy 5 mérési pontból álló átlagolással is.

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="ADC_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="ADC_0" propertyId="adc.configuration.gaincontrol" value="1x gain"/>
    <property object="ADC_0" propertyId="adc.configuration.sarclockdivider" value="1"/>
    <property object="ADC_0" propertyId="adc.control.enableadc" value="Enabled"/>
    <property object="ADC_0" propertyId="adc.control.startofconversion" value="Timer 2 overflow"/>
    <property object="ADC_0" propertyId="adc.multiplexerselection.positiveinputselection"
value="ADC0.15 (P1.7)"/>
    <property object="ADC_0" propertyId="adc.view.view" value="Advanced"/>
    <property object="CROSSBAR0" propertyId="xbar0.spi0.clockdata" value="Enabled"/>
    <property object="DefaultMode" propertyId="mode.diagramLocation" value="100, 100"/>
    <property object="INTERRUPT_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="INTERRUPT_0"
propertyId="interrupt.extendedinterruptenable1.enableadc0conversioncompleteinterrupt" value="Enabled"/>
    <property object="INTERRUPT_0" propertyId="interrupt.interruptenable.enableallinterrupts"
value="Enabled"/>
    <property object="P0.0" propertyId="ports.settings.iomode" value="Digital Push-Pull Output"/>
    <property object="P0.0" propertyId="ports.settings.outputmode" value="Push-pull"/>
    <property object="P0.2" propertyId="ports.settings.iomode" value="Digital Push-Pull Output"/>
    <property object="P0.2" propertyId="ports.settings.outputmode" value="Push-pull"/>
    <property object="P0.3" propertyId="ports.settings.iomode" value="Digital Push-Pull Output"/>
    <property object="P0.3" propertyId="ports.settings.outputmode" value="Push-pull"/>
    <property object="P0.4" propertyId="ports.settings.iomode" value="Digital Push-Pull Output"/>
    <property object="P0.4" propertyId="ports.settings.outputmode" value="Push-pull"/>
    <property object="P1.1" propertyId="ports.settings.iomode" value="Digital Push-Pull Output"/>
    <property object="P1.1" propertyId="ports.settings.outputmode" value="Push-pull"/>
    <property object="P1.2" propertyId="ports.settings.iomode" value="Digital Push-Pull Output"/>
    <property object="P1.2" propertyId="ports.settings.outputmode" value="Push-pull"/>
    <property object="P1.7" propertyId="ports.settings.inputmode" value="Analog"/>
    <property object="P1.7" propertyId="ports.settings.iomode" value="Analog I/O"/>
    <property object="P1.7" propertyId="ports.settings.skip" value="Skipped"/>
    <property object="PBCFG_0" propertyId="pbcfg.settings.enablecrossbar" value="Enabled"/>
    <property object="SPI_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="SPI_0" propertyId="spi.configuration.clockphase" value="Data sample on second
edge"/>
    <property object="SPI_0" propertyId="spi.configuration.enablemastermode" value="Enable"/>
    <property object="SPI_0" propertyId="spi.configuration.spimode" value="Master"/>
    <property object="SPI_0" propertyId="spi.control.slaveselectmode" value="Slave or master 3-wire
mode"/>
    <property object="SPI_0" propertyId="spi.control.spienable" value="Enabled"/>
    <property object="SPI_0" propertyId="spi.view.view" value="Advanced"/>
    <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_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="50"/>
    <property object="TIMER16_2" propertyId="timer16.initandreloadvalue.timerreloadvalue"
value="60432"/>
    <property object="TIMER16_2" propertyId="timer16.reloadhighbyte.reloadhighbyte" value="236"/>
    <property object="TIMER16_2" propertyId="timer16.reloadlowbyte.reloadlowbyte" value="16"/>
    <property object="TIMER16_3" propertyId="ABPeripheral.included" value="true"/>
    <property object="TIMER16_3" propertyId="timer16.initandreloadvalue.targetoverflowfrequency"
value="0"/>
    <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="VREF_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="VREF_0" propertyId="vref.hidden.voltagereferenceselect" value="VDD pin"/>
    <property object="VREF_0" propertyId="vref.voltagereferencecontrol.selectvoltagereference"
value="Unregulated VDD"/>
    <property object="WDT_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="WDT_0" propertyId="wdt.watchdogcontrol.wdtenable" value="Disable"/>
  </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

```c
//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------
#include <SI_EFM8BB1_Register_Enums.h>                  // SFR declarations
#include "InitDevice.h"
#include "Display-spi.h"
// $[Generated Includes]
// [Generated Includes]$

#define ADC_RESOLUTION 1024.0f
#define ADC_VOLTAGE 3300u
#define ADC_AVERAGE_COUNT 5u

#define LETTER_COUNT 4u
#define DISPLAY_LOOPS 4u

#define HUMAN_UNDERSTANDABLE_NUMBER_BASE 10

enum
{
        Read, Unread,
};

volatile uint16_t adc_value = 0;
volatile uint8_t adc_status = Read;

static uint8_t adc_values_read = 0;
static uint16_t voltage_mv_total = 0;
static uint16_t local_adc_value = 0;
static uint16_t voltage_mv = 0;

enum
{
        ReadingAdc,
        DisplayingVoltage
};

static uint8_t program_status = ReadingAdc;

static uint8_t letters[LETTER_COUNT] = {
        1, 2, 3, 4
};

//-----------------------------------------------------------------------------
// 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 display_loop = 0;
        uint8_t index = 0;
        uint16_t voltage_mv_local;

        // Call hardware initialization routine
        enter_DefaultMode_from_RESET();


        DecoderInit();

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

                if (program_status == ReadingAdc && adc_status == Unread)
                {
                        IE_EA = 0;
                        local_adc_value = adc_value;
                        IE_EA = 1;

                        adc_values_read += 1;

                        voltage_mv = adc_value * (ADC_VOLTAGE / ADC_RESOLUTION);
                        voltage_mv_total += voltage_mv;

                        adc_status = Read;
                        program_status = DisplayingVoltage;

                        if (adc_values_read == ADC_AVERAGE_COUNT)
                        {
                                voltage_mv_local = voltage_mv_total / ADC_AVERAGE_COUNT;

                                for (index = 0; index < LETTER_COUNT; index++)
                                {
                                        letters[(LETTER_COUNT - 1) - index] = voltage_mv_local %
HUMAN_UNDERSTANDABLE_NUMBER_BASE;
                                        voltage_mv_local /= HUMAN_UNDERSTANDABLE_NUMBER_BASE;
                                }

                                adc_values_read = 0;
                                voltage_mv_total = 0;
                        }
                }

                if (program_status == DisplayingVoltage) {
                        for (display_loop = 0; display_loop < DISPLAY_LOOPS; display_loop++)
                        {
                                for (index = 0; index < LETTER_COUNT; index++) {
                                        WriteDisplayDigit(letters[index], index);

                                        TCON_TF0 = 0;
                                        while (!TCON_TF0);
                                }
                        }

                        WriteDisplayDigit(CLEAR_DISP, LETTER_COUNT - 1);

                        program_status = ReadingAdc;
                }
        }
}
```

Interrupts.c

```
// USER INCLUDES
#include <SI_EFM8BB1_Register_Enums.h>

enum
{
        Read, Unread,
};

extern volatile uint16_t adc_value;
extern volatile uint8_t adc_status;

//-----------------------------------------------------------------------------
// ADC0EOC_ISR
//-----------------------------------------------------------------------------
//
// ADC0EOC ISR Content goes here. Remember to clear flag bits:
// ADC0CN0::ADINT (Conversion Complete Interrupt Flag)
//
//-----------------------------------------------------------------------------
SI_INTERRUPT (ADC0EOC_ISR, ADC0EOC_IRQn)
{
        ADC0CN0_ADINT = 0;

        adc_value = ADC0;
        adc_status = Unread;
}
```

Megjegyzések

2. feladat megjelenítési algoritmusa:

ADC olvasás és megjelenítés váltva. Minden 5. ADC olvasásnál elvégezzük az előző 4 méréssel együtt az átlagolást. Ez azt eredményezi, hogy minden 5. megjelenítési ciklus kap új feszültség mutatást, addig az előzőt mutatják. Ezen túl a vibrálás elkerülése érdekében (ADC művelet és osztások ideje) a kijelzés ciklus többször megy végig a kijelzőn, így több ideig jeleníti meg => erősebb fény (PWM). Egy megjelenített teljes sor után törlésre kerül az egész kijelző az anti-ghostingért.

Pontozás (tájékoztató jelleggel)

1. feladat: 200 pont

2. feladat: 200 pont