MICLAB-ZH1

Név: Pilter Zsófia

Dátum: 2024.11.04.

Mérőhely: 1 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:

```
cproperty object="ADC_0"
propertyId="adc.multiplexerselection.positiveinputselection" value="ADC0.15
(P1.7)"/>
       <property object="ADC 0" propertyId="adc.view.view" value="Advanced"/>
       roperty object="DefaultMode" propertyId="mode.diagramLocation" value="100,
       cproperty object="INTERRUPT_0" propertyId="ABPeripheral.included"
value="true"/>
       cproperty object="INTERRUPT 0"
propertyId="interrupt.extendedinterruptenable1.enableadc0conversioncompleteinterru
pt" value="Enabled"/>
       cproperty 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"

**The content of the content of 
I/0"/>
       cproperty object="P1.7" propertyId="ports.settings.skip" value="Skipped"/>
       <property object="PBCFG_0" propertyId="pbcfg.settings.enablecrossbar"</pre>
value="Enabled"/>
       <property object="TIMER01_0" propertyId="ABPeripheral.included" value="true"/>
       cproperty object="TIMER16_2" propertyId="ABPeripheral.included" value="true"/>
       cproperty object="TIMER16_2" propertyId="timer16.control.runcontrol"
value="Start"/>
       <property object="TIMER16_2" propertyId="timer16.control.timerrunningstate"</pre>
value="Timer is Running"/>
       cproperty object="TIMER16_2"
propertyId="timer16.initandreloadvalue.targetoverflowfrequency" value="50"/>
       cproperty object="TIMER16_2"
propertyId="timer16.initandreloadvalue.timerreloadvalue" value="60432"/>
       cproperty object="TIMER16_2"
propertyId="timer16.reloadhighbyte.reloadhighbyte" value="236"/>
       <property object="TIMER16_2" propertyId="timer16.reloadlowbyte.reloadlowbyte"</pre>
value="16"/>
       <property object="TIMER16_3" propertyId="ABPeripheral.included" value="true"/>
       roperty object="TIMER_SETUP_0" propertyId="ABPeripheral.included"
value="true"/>
       cproperty object="VREF_0" propertyId="ABPeripheral.included" value="true"/>
       value="VDD pin"/>
       cproperty 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"</pre>
value="Disable"/>
   </mode>
   <modeTransition>
       cproperty object="RESET → DefaultMode"
propertyId="modeTransition.source" value="RESET"/>
       cproperty object="RESET → DefaultMode"
propertyId="modeTransition.target" value="DefaultMode"/>
   </modeTransition>
</device:XMLDevice>
Main.c:
//-----
```

// src/PilterZsofia.jk9_main.c: generated by Hardware Configurator

```
//
// This file will be updated when saving a document.
// leave the sections inside the "\{[\dots]" comment tags alone
// or they will be overwritten!!
//-----
//-----
// Includes
//-----
                                   // SFR declarations
#include <SI EFM8BB1 Register Enums.h>
#include "InitDevice.h"
#define ADCMEASURE 1
#define VDD 3300.0f
#define MAXADC 1024U
volatile uint16_t g_adc_value =0U;
volatile uint8_t g_adc_flag = !ADCMEASURE;
float adc_data = 0.0f;
uint16_t seged = 0U;
float adc_final = 0.0f;
// $[Generated Includes]
// [Generated Includes]$
//-----
// 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)
 // $[SiLabs Startup]
 // [SiLabs Startup]$
float AdcToMv(uint16_t value)
 adc_data = value * VDD/MAXADC;
 return adc data;
//-----
// main() Routine
// -----
int main (void)
 // Call hardware initialization routine
 enter_DefaultMode_from_RESET();
 while (1)
 {
    if(g_adc_flag)
       g_adc_flag = !ADCMEASURE;
```

```
IE\_EA = 0;
        seged = g_adc_value;
        IE\_EA = 1;
        adc_final = AdcToMv(seged);
       }
   // $[Generated Run-time code]
   // [Generated Run-time code]$
 }
}
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 ADCMEASURE 1
extern volatile uint16_t g_adc_value;
extern volatile uint8_t g_adc_flag;
//----
// ADC0EOC_ISR
//
// ADCOEOC ISR Content goes here. Remember to clear flag bits:
// ADC0CN0::ADINT (Conversion Complete Interrupt Flag)
SI INTERRUPT (ADCOEOC ISR, ADCOEOC IRQn)
 {
   ADCOCNO\_ADINT = 0;
   g_adc_value = ADC0;
   g_adc_flag = 1;
 }
```

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"</pre>
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"/>
    gain"/>
    <property object="ADC_0" propertyId="adc.configuration.sarclockdivider"</pre>
value="0"/>
    <property object="ADC 0" propertyId="adc.control.enableadc" value="Enabled"/>
    cproperty object="ADC 0" propertyId="adc.control.startofconversion"
value="Timer 2 overflow"/>
    cproperty object="ADC 0"
propertyId="adc.multiplexerselection.positiveinputselection" value="ADC0.15
(P1.7)"/>
    cproperty object="ADC_0" propertyId="adc.view.view" value="Advanced"/>
    cproperty object="CROSSBAR0" propertyId="xbar0.spi0.clockdata"
value="Enabled"/>
    <property object="DefaultMode" propertyId="mode.diagramLocation" value="100,</pre>
100"/>
    cproperty object="INTERRUPT_0" propertyId="ABPeripheral.included"
value="true"/>
    cproperty object="INTERRUPT 0"
propertyId="interrupt.extendedinterruptenable1.enableadc0conversioncompleteinterru
pt" value="Enabled"/>
    cproperty object="INTERRUPT 0"
propertyId="interrupt.interruptenable.enableallinterrupts" value="Enabled"/>
    <property object="P0.0" propertyId="ports.settings.iomode" value="Digital</pre>
Push-Pull Output"/>
    <property object="P0.0" propertyId="ports.settings.outputmode" value="Push-</pre>
pull"/>
    cproperty object="P0.2" propertyId="ports.settings.iomode" value="Digital"
Push-Pull Output"/>
    <property object="P0.2" propertyId="ports.settings.outputmode" value="Push-</pre>
pull"/>
    <property object="P0.3" propertyId="ports.settings.iomode" value="Digital</pre>
Push-Pull Output"/>
    <property object="P0.3" propertyId="ports.settings.outputmode" value="Push-</pre>
pull"/>
    Push-Pull Output"/>
    <property object="P0.4" propertyId="ports.settings.outputmode" value="Push-</pre>
pull"/>
    <property object="P1.1" propertyId="ports.settings.iomode" value="Digital</pre>
Push-Pull Output"/>
    <property object="P1.1" propertyId="ports.settings.outputmode" value="Push-</pre>
pull"/>
```

```
<property object="P1.2" propertyId="ports.settings.iomode" value="Digital</pre>
Push-Pull Output"/>
    <property object="P1.2" propertyId="ports.settings.outputmode" value="Push-</pre>
pull"/>
    cproperty object="P1.7" propertyId="ports.settings.inputmode" value="Analog"/>
    cproperty object="P1.7" propertyId="ports.settings.iomode" value="Analog
I/0"/>
    cproperty object="P1.7" propertyId="ports.settings.skip" value="Skipped"/>
    <property object="PBCFG 0" propertyId="pbcfg.settings.enablecrossbar"</pre>
value="Enabled"/>
    <property object="SPI_0" propertyId="ABPeripheral.included" value="true"/>
    cproperty object="SPI_0" propertyId="spi.configuration.clockphase" value="Data"
sample on second edge"/>
    <property object="SPI_0" propertyId="spi.configuration.enablemastermode"</pre>
value="Enable"/>
    cproperty object="SPI_0" propertyId="spi.configuration.spimode"
value="Master"/>
    <property object="SPI_0" propertyId="spi.control.slaveselectmode" value="Slave</pre>
or master 3-wire mode"/>
    <property object="SPI_0" propertyId="spi.control.spienable" value="Enabled"/>
    cproperty object="SPI_0" propertyId="spi.view.view" value="Advanced"/>
    <property object="TIMER01_0" propertyId="ABPeripheral.included" value="true"/>
    cproperty object="TIMER16_2" propertyId="timer16.control.runcontrol"
value="Start"/>
    cproperty object="TIMER16_2" propertyId="timer16.control.timerrunningstate"
value="Timer is Running"/>
    cproperty object="TIMER16_2"
propertyId="timer16.initandreloadvalue.targetoverflowfrequency" value="1000"/>
    cproperty object="TIMER16_2"
propertyId="timer16.initandreloadvalue.timerreloadvalue" value="65281"/>
    cproperty object="TIMER16 2"
propertyId="timer16.reloadhighbyte.reloadhighbyte" value="255"/>
    <property object="TIMER16_2" propertyId="timer16.reloadlowbyte.reloadlowbyte"</pre>
value="1"/>
    cproperty object="TIMER16_3" propertyId="ABPeripheral.included" value="true"/>
    cproperty object="TIMER_SETUP_0" propertyId="ABPeripheral.included"
    cproperty object="VREF_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="VREF_0" propertyId="vref.hidden.voltagereferenceselect"</pre>
value="VDD pin"/>
    cproperty object="VREF_0"
propertyId="vref.voltagereferencecontrol.selectvoltagereference"
value="Unregulated VDD"/>
    <property object="WDT_0" propertyId="ABPeripheral.included" value="true"/>
    cproperty object="WDT_0" propertyId="wdt.watchdogcontrol.wdtenable"
value="Disable"/>
 </mode>
  <modeTransition>
    cproperty object="RESET → DefaultMode"
propertyId="modeTransition.source" value="RESET"/>
    cproperty object="RESET → DefaultMode"
propertyId="modeTransition.target" value="DefaultMode"/>
  </modeTransition>
</device:XMLDevice>
Main.c:
//----
// src/PilterZsofia jk1 2 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"
                                       // SFR declarations
#include <SI_EFM8BB1_Register_Enums.h>
#include "Display-spi.h"
#define ADCMEASURE 1
#define VDD 3300.0f
#define MAXADC 1024U
#define NUMBER_OF_SEGMENTS 3
volatile uint16_t g_adc_value =0U;
volatile uint8_t g_adc_flag = !ADCMEASURE;
float adc_data = 0.0f;
uint16 t seged = 0U;
uint16_t adc_final = 0;
void DecoderInit()
 DECODER_A = 0;
 DECODER_B = 0;
 DECODER_C = 0;
}
uint16_t numbers[NUMBER_OF_SEGMENTS];
void WriteDisplayDigit (uint8_t dispNumber, uint8_t digitSelect)
 /* Clear digit before switch to next to anti-ghosting */
 WriteSPI(CLEAR DISP);
 SelectSegment(digitSelect);
 WriteSPI(dispNumber);
}
void WriteSPI (uint8_t dispNumber)
 /* Code bytes of decimal numbers; bytes are negated;
  * clear MSB to activate decimal point */
 const uint8 t numberCodes[14] =
   { 0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90, 0x7F, 0xFF,
0x9C, 0xC6 };
 SPI0DAT = numberCodes[dispNumber];
 while(!SPI0CN0 SPIF);
 SPIOCNO SPIF = 0;
 DISP_OE_RCLK = 1; //RCLK high: data goes to storage registers
```

```
DISP_OE_RCLK = 0; //Enable outputs, RCLK low
void SelectSegment (uint8_t digitSelect)
 /* Select digit */
 switch(digitSelect)
   {
     case 0:
      DECODER A = 0;
      DECODER_B = 0;
      break;
     case 1:
       DECODER_A = 1;
      DECODER B = 0;
      break;
     case 2:
      DECODER_A = 0;
      DECODER_B = 1;
      break;
     case 3:
      DECODER A = 1;
      DECODER B = 1;
      break;
   }
}
float AdcToMv(uint16_t value)
 adc_data = value * VDD/MAXADC;
 return adc_data;
}
void number_select(uint16_t number)
 numbers[3] = (number/1000)%10;
 numbers[2] = (number/100)%10;
 numbers[1] = (number/10)%10;
 numbers[0] = (number/1)%10;
}
// $[Generated Includes]
// [Generated Includes]$
//-----
// 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)
 // $[SiLabs Startup]
 // [SiLabs Startup]$
```

```
uint8_t inc;
uint16_t number = 0;
// main() Routine
// -----
int main (void)
 // Call hardware initialization routine
 enter_DefaultMode_from_RESET();
 DecoderInit();
 while (1)
 {
     if(g_adc_flag)
       g_adc_flag = !ADCMEASURE;
      IE\_EA = 0;
       seged = g_adc_value;
       IE\_EA = 1;
       adc_final = AdcToMv(seged);
     number_select(adc_final);
     if(!TMR2CN0_TF2H)
        for(inc = 0; inc < NUMBER_OF_SEGMENTS ; inc++)</pre>
          {
            WriteSPI(0xFF);
            WriteDisplayDigit(numbers[inc], inc);
       }
     inc = 0;
   // $[Generated Run-time code]
   // [Generated Run-time code]$
}
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>
extern volatile uint16_t g_adc_value;
extern volatile uint8_t g_adc_flag;
//-----
// ADC0EOC_ISR
                   ______
//
// ADCOEOC ISR Content goes here. Remember to clear flag bits:
// ADC0CN0::ADINT (Conversion Complete Interrupt Flag)
```

```
//
//-----
SI_INTERRUPT (ADC0EOC_ISR, ADC0EOC_IRQn)
{
   ADC0CN0_ADINT = 0;
   g_adc_value = ADC0;
   g_adc_flag = 1;
}
```

Megjegyzések



Pontozás (tájékoztató jelleggel)

1. feladat: 200 pont

2. feladat: 200 pont