ZH_1

MICLAB-ZH1

Név: Vad Avar

Dátum: 2024. 11. 04.

Mérőhely: 1 jobb



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"</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"/>
    roperty object="ADC_0" propertyId="adc.configuration.gaincontrol" value="1x
gain"/>
    <property object="ADC_0" propertyId="adc.configuration.sarclockdivider"</pre>
value="0"/>
    cproperty 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)"/>
```

```
<property object="ADC_0" propertyId="adc.view.view" value="Advanced"/>
    roperty object="DefaultMode" propertyId="mode.diagramLocation" value="100,
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"/>
    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"/>
    cproperty object="TIMER16_2" propertyId="ABPeripheral.included" value="true"/>
    roperty 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"/>
    cproperty object="TIMER16_2" propertyId="timer16.reloadlowbyte.reloadlowbyte"
value="16"/>
    cproperty object="VREF_0" propertyId="ABPeripheral.included" value="true"/>
    cproperty object="VREF_0" propertyId="vref.hidden.voltagereferenceselect"
value="VDD pin"/>
    cproperty object="VREF 0"
propertyId="vref.voltagereferencecontrol.selectvoltagereference"
value="Unregulated VDD"/>
    cproperty 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:
//-----
// 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>

Interrupts:

```
// src/feladat01_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 REF_VOLT 3300.0f
#define RES 1024u
//-----
// 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.
//-----
extern volatile uint8_t convCompl = 0u;
extern volatile uint16_t adcData = 0u;
float milivolts = 0.0f;
void SiLabs_Startup (void)
```

```
// $[SiLabs Startup]
// [SiLabs Startup]$
}

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

    while (1)
    {
        if (convCompl) {
            convCompl = 0;

            milivolts = adcData * REF_VOLT / RES;
        }
    }
}
```

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:

```
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>
    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"/>
    <property object="P0.2" propertyId="ports.settings.iomode" value="Digital</pre>
Push-Pull Output"/>
    <property object="P0.2" propertyId="ports.settings.outputmode" value="Push-</pre>
pull"/>
    cproperty object="P0.3" propertyId="ports.settings.iomode" value="Digital"
Push-Pull Output"/>
    cproperty object="P0.3" propertyId="ports.settings.outputmode" value="Push-
pull"/>
    <property object="P0.4" propertyId="ports.settings.iomode" value="Digital</pre>
Push-Pull Output"/>
    cproperty object="P0.4" propertyId="ports.settings.outputmode" value="Push-
pull"/>
    cproperty object="P1.1" propertyId="ports.settings.iomode" value="Digital"
Push-Pull Output"/>
    cproperty object="P1.1" propertyId="ports.settings.outputmode" value="Push-
pull"/>
    <property object="P1.2" propertyId="ports.settings.iomode" value="Digital</pre>
Push-Pull Output"/>
    roperty object="P1.2" propertyId="ports.settings.outputmode" value="Push-
pull"/>
    cproperty object="P1.7" propertyId="ports.settings.inputmode" value="Analog"/>
    <property object="P1.7" propertyId="ports.settings.iomode" value="Analog</pre>
I/0"/>
    <property object="P1.7" propertyId="ports.settings.skip" value="Skipped"/>
    cproperty object="PBCFG_0" propertyId="pbcfg.settings.enablecrossbar"
value="Enabled"/>
    cproperty 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"/>
    cproperty object="SPI_0" propertyId="spi.control.spienable" value="Enabled"/>
    cproperty object="SPI_0" propertyId="spi.view.view" value="Advanced"/>
    cproperty object="TIMER01_0" propertyId="ABPeripheral.included" value="true"/>
```

```
cproperty object="TIMER01 0"
propertyId="timer01.timer1mode2:8bitcountertimerwithautoreload.targetoverflowfrequ
ency" value="1000"/>
   cproperty object="TIMER01 0"
propertyId="timer01.timer1mode2:8bitcountertimerwithautoreload.timerreloadvalue"
value="1"/>
   cproperty object="TIMER16_2" propertyId="ABPeripheral.included" value="true"/>
   cproperty object="TIMER16_2" propertyId="timer16.control.runcontrol"
value="Start"/>
   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"/>
   roperty object="TIMER16_2" propertyId="timer16.reloadlowbyte.reloadlowbyte"
value="1"/>
   cproperty object="TIMER16_3" propertyId="ABPeripheral.included" value="true"/>
   cproperty object="TIMER_SETUP_0" propertyId="ABPeripheral.included"
value="true"/>
   <property object="VREF 0" propertyId="ABPeripheral.included" value="true"/>
   value="VDD pin"/>
   cproperty object="VREF 0"
propertyId="vref.voltagereferencecontrol.selectvoltagereference"
value="Unregulated VDD"/>
   cproperty 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:

```
// $[Generated Includes]
// [Generated Includes]$
#define REF_VOLT 3300.0f
#define RES 1024u
#define NUM_OF_DIGS 4u
//-----
// 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.
//-----
extern volatile uint8_t convCompl = 0u;
extern volatile uint16_t adcData = 0u;
float milivolts = 0.0f;
uint8_t inc = 0;
uint8_t numToDisp = 0;
uint32_t mvInt = 0;
void SiLabs_Startup (void)
 // $[SiLabs Startup]
 // [SiLabs Startup]$
//-----
// main() Routine
// -----
int main (void)
 // Call hardware initialization routine
 enter_DefaultMode_from_RESET();
 DecoderInit();
 while (1)
 {
     if (convCompl) {
        convCompl = 0;
        milivolts = adcData * REF VOLT / RES;
     }
     if (TMR2CN0_TF2H) {
        TMR2CN0\_TF2H = 0;
        TMR2CN0\_TF2L = 0;
        mvInt = 1234u; //float to int?
        for (inc = 0; inc < NUM_OF_DIGS; inc++) {</pre>
           numToDisp = mvInt % (int)pow(10, inc+1);
           WriteDisplayDigit(numToDisp, inc);
           mvInt /= (int)pow(10, 4-inc);
        }
     }
```

```
}
}
```

Interrups:

```
//-----
// 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>
// ADC0EOC_ISR
//-----
//
// ADC0EOC ISR Content goes here. Remember to clear flag bits:
// ADC0CN0::ADINT (Conversion Complete Interrupt Flag)
//-----
extern volatile uint8_t convCompl;
extern volatile uint16_t adcData;
SI_INTERRUPT (ADC0EOC_ISR, ADC0EOC_IRQn)
  ADCOCNO_ADINT = 0;
  adcData = ADC0;
  convCompl = 1;
 }
```

Megjegyzések

Pontozás (tájékoztató jelleggel)

feladat: 200 pont
 feladat: 200 pont