LED-ek vezérlése shift regiszterrel

MICLAB-07

Név: Pilter Zsófia, Vad Avar

Dátum: 2024.10.21.

Mérőhely: 1 bal és jobb

Bevezetés

Az interrupt használatának megismerése.

Ajánlott irodalom

- A házi feladatban találhatók
- Honlap: 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 – LED1 és LED7 bekapcsolása

Kapcsolja be a kiegészítő panelen lévő LED1 és LED7 nevű LED-eket. (Kövesse az órai ppt-t és használja a MicLab-utmutato.pdf-ben lévő kapcsolódó részeket.)

Konfigurálja az SPI-t az órai diasor segítségével. A Clock Phase és a Clock Polarity beállításaival 4 eset lehetséges az adat (MOSI) és az órajel (SCK) viszonyára. Válassza ki a Clock Phase és a Clock Polarity helyes kombinációját a shift regiszter idődiagramja és a Laboratory Practicals pdf 7.7-es ábrája alapján. Az SPI 3 vezetékes módban legyen és az órajele legyen ~1,5 MHz.

Az SPI **polling módon** történő használatával töltse fel a shift regisztert. A shift register írásához készítsen egy saját függvényt. Ne feledje, hogy a shift register csak akkor adja ki a kimeneteire a beleírt byte-ot, ha az OE_RCLK vonalon egy felfutó élt kap.

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="CROSSBAR0" propertyId="xbar0.spi0.clockdata"
value="Enabled"/>
    <property object="DefaultMode" propertyId="mode.diagramLocation" value="100,
100"/>
    <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="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="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:

```
//-------------------------------------------------------------
// src/feladat07-01_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 DispClock P0_B0
#define DispData P0_B2
#define DispOutEnable P0_B3

#define DECODER_A P1_B1
#define DECODER_B P1_B2
#define DECODER_C P0_B4

#define LED_7 ~(1)
#define LED_1 ~(2)
#define LED_2 ~(4)
#define LED_3 ~(8)
#define LED_6 ~(16)
#define LED_5 ~(32)
#define LED_4 ~(64)

#define T1 1
#define T2 2
#define T3 3
#define T4 4
#define T5 5

//-----------------------------------------------------------------------------
// 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)
{
  // $[SiLabs Startup]
  // [SiLabs Startup]$
}

void activate_display(uint8_t led_group)
{
    if (led_group == T1)
    {
        DECODER_A = 0;
        DECODER_B = 0;
        DECODER_C = 0;
    }
    else if (led_group == T2)
    {
        DECODER_A = 1;
        DECODER_B = 0;
        DECODER_C = 0;
    }
    else if (led_group == T3)
    {
        DECODER_A = 0;
        DECODER_B = 1;
        DECODER_C = 0;
    }
    else if (led_group == T4)
    {
```

```
        DECODER_A = 1;
        DECODER_B = 1;
        DECODER_C = 0;
    }
    else if (led_group == T5)
    {
        DECODER_A = 0;
        DECODER_B = 0;
        DECODER_C = 1;
    }

    DispOutEnable = 1;
}

void deactivate_display(void)
{
  DispOutEnable = 0;
}

void transmit_to_spi(uint8_t value)
{
    SPI0CN0_SPIF = 0;
    SPI0DAT = value;

    while (!SPI0CN0_SPIF){}

    SPI0CN0_SPIF = 0;
}

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

  while (1)
  {

    // $[Generated Run-time code]
    // [Generated Run-time code]$
        deactivate_display();

        transmit_to_spi(LED_1 & LED_7);

        activate_display(T5);
  }
}
```

Az elkészült programot be kell mutatni!

A gyakorlatvezető ellenőrizte:

- Igen
- Nem

A program működött:

- Igen
- Nem

2. feladat – Elektronikus dobókocka megvalósítása

Írjon egy olyan programot, ami a 7db LED segítségével meg tudja jeleníteni egy 6 oldalú dobókocka oldalait. Amíg a kiegészítő panelen lévő SW1 nyomógomb le van nyomva, addig léptesse a főprogramban a dobókocka oldalait. Mivel a léptetés nagy sebességgel történik, így teljesíthető a véletlenszerűség, ami egy kockadobásnak tekinthető.

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="CROSSBAR0" propertyId="xbar0.spi0.clockdata"
value="Enabled"/>
    <property object="DefaultMode" propertyId="mode.diagramLocation" value="100,
100"/>
    <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="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="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
//=========================================================
// src/feladat07-02_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"
#define DispClock P0_B0
#define DispData P0_B2
#define DispOutEnable P0_B3

#define SW1 P0_B1

#define DECODER_A P1_B1
#define DECODER_B P1_B2
#define DECODER_C P0_B4

#define LED_7 ~(1)
#define LED_1 ~(2)
#define LED_2 ~(4)
#define LED_3 ~(8)
#define LED_6 ~(16)
#define LED_5 ~(32)
#define LED_4 ~(64)

#define T1 1
#define T2 2
#define T3 3
#define T4 4
#define T5 5

uint8_t actual_number = 0;

//-------------------------------------------------------------------------------
// 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)
{
  // $[SiLabs Startup]
  // [SiLabs Startup]$
}

void activate_display(uint8_t led_group)
{
    if (led_group == T1)
    {
        DECODER_A = 0;
        DECODER_B = 0;
```

```
        DECODER_C = 0;
    }
    else if (led_group == T2)
    {
        DECODER_A = 1;
        DECODER_B = 0;
        DECODER_C = 0;
    }
    else if (led_group == T3)
    {
        DECODER_A = 0;
        DECODER_B = 1;
        DECODER_C = 0;
    }
    else if (led_group == T4)
    {
        DECODER_A = 1;
        DECODER_B = 1;
        DECODER_C = 0;
    }
    else if (led_group == T5)
    {
        DECODER_A = 0;
        DECODER_B = 0;
        DECODER_C = 1;
    }

    DispOutEnable = 1;
}

void deactivate_display(void)
{
    DispOutEnable = 0;
}

void transmit_to_spi(uint8_t value)
{
    SPI0CN0_SPIF = 0;
    SPI0DAT = value;

    while (!SPI0CN0_SPIF);

    SPI0CN0_SPIF = 0;
}


uint8_t get_led_combination(uint8_t number)
{
  if (number == 0)
  {
    return LED_7;
  }
  else if (number == 1)
  {
    return LED_1 & LED_6;
  }
  else if (number == 2)
  {
    return LED_1 & LED_7 & LED_6;
```

```
  }
  else if (number == 3)
  {
    return LED_1 & LED_4 & LED_3 & LED_6;
  }
  else if (number == 4)
  {
    return LED_1 & LED_4 & LED_3 & LED_6 & LED_7;
  }
  else if (number == 5)
  {
    return LED_1 & LED_2 & LED_4 & LED_3 & LED_5 & LED_6;
  }
  else
  {
    return 0;
  }
}


//-------------------------------------------------------------------------
// main() Routine
// ------------------------------------------------------------------------
int main(void)
{
    enter_DefaultMode_from_RESET();

    while (1)
    {
       deactivate_display();
        if (!SW1)
        {
            actual_number++;

                if (actual_number > 5)
                {
                    actual_number = 0;
                }

        }
        transmit_to_spi(get_led_combination(actual_number));

        activate_display(T5);
    }
}
```

Az elkészült programot be kell mutatni!

A gyakorlatvezető ellenőrizte:

- Igen
- Nem

A program működött:

- Igen
- Nem

Szorgalmi feladat – LED-ek léptetése timerrel (futófény)

Valósítson meg futófényt, amely a kiegészítőpanelen található LED-ek sorszáma szerint növekvő sorrendben be-, majd kikapcsolja a LED-eket. A LED-ek léptetését egy timerrel valósítsa meg, aminek a túlcsordulási rátája 35 Hz legyen.

Valósítson meg irányváltási funkciót is, amennyiben a kiegészítő panelen található SW1-es nyomógomb lenyomásra kerül a futófény a LED-ek sorszáma szerinti csökkenő sorrendben folytatódjon. (Újbóli lenyomás esetén természetesen ismét növekvő sorrendben.). Irányváltás csak a nyomógomb felengedésére történjen!

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

Az elkészült programot be kell mutatni!

A gyakorlatvezető ellenőrizte:

- Igen
- Nem

A program működött:

- Igen
- Nem

Megjegyzések