

Kommunikáció - UART

MICLAB-10

Név: Stefán Kornél és Kolman Domonkos

Dátum: 2024. 11. 11 18:00

Mérőhely: 7 bal

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 – Karakter küldése és fogadása polling módban

Kösse össze a két mikrovezérlő kit UART RX és TX vonalait, valamint a GND-t (ügyeljen arra, hogy kimenetet csak bemenettel kössön össze). A baud rate legyen 115200 bit/s (a SYSCLK legyen 24,5 MHz, a Timer 1 Clock Source beállítása a Use SYSCLK legyen). A páros egyik tagja készítsen egy olyan programot, ami a BTNo nyomógomb lenyomására elküld egy „L” karaktert UART-on polling módban. A páros másik tagja pedig folyamatosan fogadja az UART-on beérkező adatot és ha a „L” karakter érkezik be, akkor kapcsolja be az EFM8 kiten lévő LEDo-át.

Küldés programja

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="CLOCK_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="CLOCK_0" propertyId="clock.clockselect.clock sourcedivider" value="SYSCLK / 1"/>
    <property object="CLOCK_0" propertyId="clock.clockselect.sysclk" value="24.500 MHz"/>
    <property object="CROSSBAR0" propertyId="xbar0.uart0.data" value="Enabled"/>
    <property object="DefaultMode" propertyId="mode.diagramLocation" value="100, 100"/>
    <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="PBCFG_0" propertyId="pbcfg.settings.enablecrossbar" value="Enabled"/>
    <property object="TIMER01_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="TIMER01_0" propertyId="timer01.timer1highbyte.timer1highbyte" value="150"/>
    <property object="TIMER01_0"
propertyId="timer01.timer1mode2:8bitcountertimerwithautoreload.targetoverflowfrequency"
value="230400"/>
    <property object="TIMER01_0"
propertyId="timer01.timer1mode2:8bitcountertimerwithautoreload.timerreloadvalue" value="150"/>
```

```

        <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.timer01control.timer1runcontrol"
value="Start"/>
        <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.clocksource" value="Use SYSCCLK"/>
        <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.mode" value="Mode 2, 8-bit
Counter/Timer with Auto-Reload"/>
        <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.timerrunningstate" value="Timer is
Running"/>
        <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.timerswitch1:runcontrol"
value="Start"/>
        <property object="UART_0" propertyId="ABPeripheral.included" value="true"/>
        <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

```

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

#define ONBOARD_BTN P0_B2

enum {
    PRESSED,
    RELEASED
};

static uint8_t button_state = RELEASED;

//-----
// 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 write_to_uart(uint8_t data_to_send)
{
    SBUF0 = data_to_send;

    while (!SCON0_TI);
    SCON0_TI = 0;
}

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

```

```

enter_DefaultMode_from_RESET();
SCON0_TI = 1;

while (1)
{
    // $[Generated Run-time code]
    // [Generated Run-time code]$
    if (!ONBOARD_BTN && button_state == RELEASED) {
        write_to_uart('L');
        button_state = PRESSED;
    }

    if (ONBOARD_BTN && button_state == PRESSED) {
        button_state = RELEASED;
    }
}
}
}

```

Fogadás programja

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-qso24" version="4.0.0" contextId="%DEFAULT%">
  <mode name="DefaultMode">
    <property object="CLOCK_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="CLOCK_0" propertyId="clock.clockselect.clocksourcedivider" value="SYSCLK / 1"/>
    <property object="CLOCK_0" propertyId="clock.clockselect.sysclk" value="24.500 MHz"/>
    <property object="CROSSBAR0" propertyId="xbar0.uart0.data" value="Enabled"/>
    <property object="DefaultMode" propertyId="mode.diagramLocation" value="100, 100"/>
    <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="PBCFG_0" propertyId="pbcfg.settings.enablecrossbar" value="Enabled"/>
    <property object="TIMER01_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="TIMER01_0" propertyId="timer01.timer1highbyte.timer1highbyte" value="150"/>
    <property object="TIMER01_0"
propertyId="timer01.timer1mode2:8bitcountertimerwithautoreload.targetoverflowfrequency"
value="230400"/>
    <property object="TIMER01_0"
propertyId="timer01.timer1mode2:8bitcountertimerwithautoreload.timerreloadvalue" value="150"/>
    <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.timer01control.timer1runcontrol"
value="Start"/>
    <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.clocksource" value="Use SYSCLK"/>
    <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.mode" value="Mode 2, 8-bit
Counter/Timer with Auto-Reload"/>
    <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.timerrunningstate" value="Timer is
Running"/>
    <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.timerswitch1:runcontrol"
value="Start"/>
    <property object="UART_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="UART_0" propertyId="uart.serialportcontrol.enablereceive" value="Enabled"/>
    <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

```
//-----  
// Includes  
//-----  
#include <SI_EFM8BB1_Register_Enums.h>           // SFR declarations  
#include "InitDevice.h"  
// $[Generated Includes]  
// [Generated Includes]$  
  
#define ONBOARD_LED P1_B4  
  
//-----  
// 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 read_uart(void)  
{  
    while(!SCON0_RI);  
    SCON0_RI = 0;  
  
    return SBUF0;  
}  
  
//-----  
// main() Routine  
// -----  
int main (void)  
{  
    // Call hardware initialization routine  
    enter_DefaultMode_from_RESET();  
    ONBOARD_LED = 1;  
  
    while (1)  
    {  
        // $[Generated Run-time code]  
        // [Generated Run-time code]$  
  
        uint8_t input = read_uart();  
  
        if (input == 'L') {  
            ONBOARD_LED = 0;  
        }  
    }  
}
```

Az elkészült programot be kell mutatni!

A gyakorlatvezető ellenőrizte:

- [Igen](#)
- [Nem](#)

A program működött:

- [Igen](#)
- [Nem](#)

2. feladat – 4 jegyű szám küldése PC-re UART-on

A mikrovezérlővel az USB kábelén keresztül UART üzeneteket lehet küldeni a PC-nek (fogadni is lehet PC-től). Ehhez egy segédprogramot kell használni a PC-n, aminek a leírása a feladat végén található.

A SYSCLK legyen a maximális 24,5 MHz. A baud rate legyen közelítőleg 115200 bit/s.

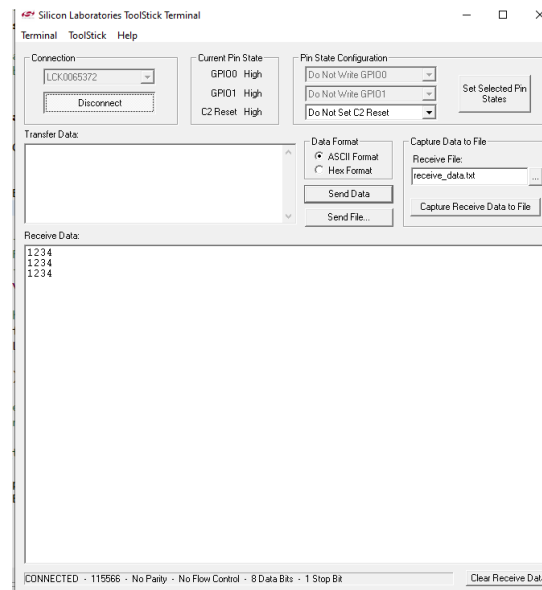
Az UART-ot polling módban használja a kódban, amihez írjon egy saját függvényt. Az „sprintf” függvény segítségével konvertálja át az 1234 négyjegyű számot stringbe. A kapott stringet és egy ‘\n’ karaktert egy megfelelő méretű tömbben tároljon el.

Ezután a karakter tömb elemeit egyesével ki lehet küldeni az UART-on. Másodpercenként egyszeri ismétléssel küldje a négyjegyű számot.

Tipp: a sorvégi ‘\0’ karaktert figyelje az tömb indexelésnél, azt már nem kell kiküldeni.

A bejövő adatokat a PC-n a ToolStick Terminal programmal lehet megjeleníteni. A programot nagyon egyszerű használni: csak a „Connect” gombra kell kattintanunk az előlapon. Fontos, hogy csak akkor tudunk kapcsolódni, ha nem debug módban fut a kódunk, azaz a feltöltés után állítsuk le a debugolást, ekkor újraindul a kódunk, de már nem debug módban. A mikrovezérlőtől fogadott üzenet a „Receive Data:” ablakban jelenik meg.

Készítsen képernyőképet a ToolStick Terminalról működés közben.



1. ábra: Saját képernyőkép a ToolStick Terminalról.

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="CLOCK_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="CLOCK_0" propertyId="clock.clockselect.clock sourcedivider" value="SYSCLK / 1"/>
    <property object="CLOCK_0" propertyId="clock.clockselect.sysclk" value="24.500 MHz"/>
    <property object="CROSSBAR0" propertyId="xbar0.uart0.data" value="Enabled"/>
    <property object="DefaultMode" propertyId="mode.diagramLocation" value="100, 100"/>
    <property object="P0.4" propertyId="ports.settings.iomode" value="Digital Push-Pull Output"/>
    <property object="P0.4" propertyId="ports.settings.outputmode" value="Push-pull"/>
  </mode>
</device:XMLDevice>
```

```

        <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.timer1highbyte.timer1highbyte" value="150"/>
        <property object="TIMER01_0"
propertyId="timer01.timer1mode2:8bitcountertimerwithautoreload.targetoverflowfrequency"
value="230400"/>
        <property object="TIMER01_0"
propertyId="timer01.timer1mode2:8bitcountertimerwithautoreload.timerreloadvalue" value="150"/>
        <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="100"/>
        <property object="TIMER16_2" propertyId="timer16.initandreloadvalue.timerreloadvalue"
value="45119"/>
        <property object="TIMER16_2" propertyId="timer16.reloadhighbyte.reloadhighbyte" value="176"/>
        <property object="TIMER16_2" propertyId="timer16.reloadlowbyte.reloadlowbyte" value="63"/>
        <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.timer01control.timer1runcontrol"
value="Start"/>
        <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.clocksourc" value="Use SYSCLK"/>
        <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.mode" value="Mode 2, 8-bit
Counter/Timer with Auto-Reload"/>
        <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.timerrunningstate" value="Timer is
Running"/>
        <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.timerswitch1:runcontrol"
value="Start"/>
        <property object="UART_0" propertyId="ABPeripheral.included" value="true"/>
        <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

```

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

#define UART_BUFFER_SIZE 6u
#define NUM_COUNT 4u
#define BEST_NUMBERRING_SYSTEM_FOR_HUMANS 10u
#define LAST_NUMBER 9u
#define LAST_NUMBER_CHAR '9'

#define MAGIC_NUMBER 1234u

static char buffer[UART_BUFFER_SIZE];
static uint8_t counter = 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 write_to_uart(uint8_t data_to_send)
{
    SBUF0 = data_to_send;

    while (!SCON0_TI);
    SCON0_TI = 0;
}

void send_string_uart(const char *str)
{
    while (*str != '\0')
    {
        write_to_uart(*str);
        str++;
    }
}

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

    sprintf(buffer, "%d\n", MAGIC_NUMBER);

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

        while(!TMR2CN0_TF2H);
        TMR2CN0_TF2H = 0;
        ++counter;

        if (counter >= 100)
        {
            send_string_uart(buffer);
            counter = 0;
        }
    }
}

```

Az elkészült programot be kell mutatni!

A gyakorlatvezető ellenőrizte:

- Igen
- Nem

A program működött:

- Igen
- Nem

Szorgalmi feladat – Karakter küldése és fogadása interrupt módban

Készítsen egy programot, ami UART-on elküld egy 'L' karaktert, ha a BTNo nyomógomb le van nyomva, ha nincs lenyomva, akkor a 'o' karaktert küldje. A küldés mellett fogadja is a program az UART-on beérkező adatot és az 'L' karakter beérkezése esetén kapcsolja be a LEDO LED-et, a 'o' karakter

beérkezése esetén pedig kapcsolja le a LED-et. A küldés és a fogadás is interrupt módban történjen. A páros mindkét tagjánál ugyanaz a kód fusson.

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="CLOCK_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="CLOCK_0" propertyId="clock.clockselect.clock sourcedivider" value="SYSCLK / 1"/>
    <property object="CLOCK_0" propertyId="clock.clockselect.sysclk" value="24.500 MHz"/>
    <property object="CROSSBAR0" propertyId="xbar0.uart0.data" 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.interruptenable.enableallinterrupts"
value="Enabled"/>
    <property object="INTERRUPT_0" propertyId="interrupt.interruptenable.enableuart0interrupt"
value="Enabled"/>
    <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="P1.4" 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.timer1highbyte.timer1highbyte" value="150"/>
    <property object="TIMER01_0"
propertyId="timer01.timer1mode2:8bitcountertimerwithautoreload.targetoverflowfrequency"
value="230400"/>
    <property object="TIMER01_0"
propertyId="timer01.timer1mode2:8bitcountertimerwithautoreload.timerreloadvalue" value="150"/>
    <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.timer01control.timer1runcontrol"
value="Start"/>
    <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.clocksource" value="Use SYSCLK"/>
    <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.mode" value="Mode 2, 8-bit
Counter/Timer with Auto-Reload"/>
    <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.timerrunningstate" value="Timer is
Running"/>
    <property object="TIMER_SETUP_0" propertyId="timer_setup.timer1.timerswitch1:runcontrol"
value="Start"/>
    <property object="UART_0" propertyId="ABPeripheral.included" value="true"/>
    <property object="UART_0" propertyId="uart.serialportcontrol.enablereceive" value="Enabled"/>
    <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

```
//-----
// Includes
//-----
#include <SI_EFM8BB1_Register_Enums.h> // SFR declarations
#include "InitDevice.h"
```



```

// $[Generated Includes]
// [Generated Includes]$

#define ONBOARD_LED P1_B4
#define ONBOARD_BTN P0_B2

enum {
    UNSENT,
    SENT
};

enum {
    UNREAD,
    READ
};

volatile uint8_t data_to_send = 0;
volatile uint8_t data_to_send_status = SENT;

volatile uint8_t data_to_read = 0;
volatile uint8_t data_to_read_status = READ;

//-----
// 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)
{
    // Call hardware initialization routine
    enter_DefaultMode_from_RESET();

    SCON0_TI = 1;

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

        if (data_to_send_status == SENT)
        {
            data_to_send = ONBOARD_BTN ? 'o' : 'L';
            data_to_send_status = UNSENT;
        }

        if (data_to_read_status == UNREAD)
        {
            ONBOARD_LED = data_to_read == 'o';
            data_to_read_status = READ;
        }
    }
}

```

Interrupts.c

```

// USER INCLUDES
#include <SI_EFM8BB1_Register_Enums.h>

enum {
    UNSENT,

```

```

    SENT
};

enum {
    UNREAD,
    READ
};

extern volatile uint8_t data_to_send;
extern volatile uint8_t data_to_send_status;

extern volatile uint8_t data_to_read;
extern volatile uint8_t data_to_read_status;

//-----
// UART0_ISR
//-----
//
// UART0_ISR Content goes here. Remember to clear flag bits:
// SCON0::RI (Receive Interrupt Flag)
// SCON0::TI (Transmit Interrupt Flag)
//
//-----
SI_INTERRUPT (UART0_ISR, UART0_IRQn)
{
    if (SCON0_RI) {
        data_to_read = SBUF0;
        data_to_read_status = UNREAD;

        SCON0_RI = 0;
    }

    if (SCON0_TI) {
        SBUF0 = data_to_send;
        data_to_send_status = SENT;

        SCON0_TI = 0;
    }
}

```

Az elkészült programot be kell mutatni!

A gyakorlatvezető ellenőrizte:

- Igen
- Nem

A program működött:

- Igen
- Nem

Megjegyzések