Concordia University
Department of Computer Science and Software Engineering

SOEN422 Fall 2021

—

Lab 5 (5%)

—

UART Serial Communication

Purpose

This lab aims to provide exercises on serial communication involving UART. This communication protocol is used in many embedded systems and device interconnections. **Use base-metal C/C++ for these two tasks, but not using the Arduino Serial library.**

To send character to the Arduino Nano, use: (1) the Arduino IDE in selecting your "Port:" and then select "Serial Monitor" enter your character and "Send" it; or (2) a Serial Console program like PuTTY. Make sure to configure them at 9600 baud, 8 data-bit, No parity, and 1 stop bit (8N1).

Task 1 [2%]: UART Controlling an LED by polling

The first task is to create a program that runs in bare C to control the onboard LED of an Arduino Nano. The user should send the character 'H' via the Serial Monitor in the Arduino IDE to turn on the onboard LED and 'L' to turn off the LED. A BSL USART component must be implemented and used by this task with the following interface:

```
// bsl_Usart.h

#ifndef __bsl_Usart_h
#define __bsl_Usart_h

// Note: These #includes are inappropriate and should be moved/hidden
// in the implementation. HAL or Users should not know if
// the Uart is used by polling or by interrupts.
// #include <avr/io.h>
// #include <avr/interrupt.h>

#include <stdint.h>
#include <stdbool.h>

// Note: All the configuration details are moved/hidden in this version.
// #define BaudRate9600 ((uint16_t)(103))

void bsl_Uart_Init(void);
char bsl_Uart_RxChar(void);
void bsl_Uart_TxChar(char c);

#endif
```

Your BSL USART component implementation **by polling** should be in the following implementation file:

```
// bsl_UsartByPolling.c

#include "bsl_Usart.h" // Share the same interface (polling vs interrupts).

#include <avr/io.h>

#define BaudRate9600 ((uint16_t)(103))

static   bool initialized;

//-----------------------------------------------------------------------
// Receive a character from UART.
//-----------------------------------------------------------------------
char bsl_Uart_RxChar(void) {
    // Your code...
}
//-----------------------------------------------------------------------
// Transmit a character to UART.
//-----------------------------------------------------------------------
void bsl_Uart_TxChar(char c) {
    // Not required for this task.
}
//-----------------------------------------------------------------------
// Initialize UART for polling.
//-----------------------------------------------------------------------
void bsl_Uart_Init(void) {
    if (!initialized) {      // To make sure it will be done only once.
        // Your code...
    }
}
```

Your **task 1** should follow the following bare C program template:

```
// Lab5Task1.c - version with a BSL implementation by polling.

#include <avr/io.h>

#include "bsl_Usart.h" // Share the same interface (polling and interrupts).

#include <stdint.h>
#include <stdbool.h>

volatile char dataReceived;

int main() {
    // ---- Configure the onboard LED.
    // Your code...

    // ---- Configure UART by polling
    // No need to disable/enable interrupts

    bsl_Uart_Init();

    while (true) {
        char dataReceived = bsl_Uart_RxChar();

        // Your looping code...

    }
}
```

Task 2A [1.5%]: UART Controlling an LED by interrupts without a BSL component

Similar to Task 1, this task is to create a program that runs in bare C to control **by interrupts** the onboard LED of an Arduino Nano. The user should send the character '1' (one) via the Serial Monitor in the Arduino IDE to turn on the onboard LED and '0' (zero) to turn off the LED.

Your **task 2A** should follow the following bare C program template:

```
// Lab5Task2WithoutBSL.c - version by interrupts without a BSL interface.

#include <avr/io.h>
#include <avr/interrupt.h>

#include <stdint.h>
#include <stdbool.h>

#define BaudRate9600 ((uint16_t)(103))

volatile char dataReceived;

int main() {
    // Your configuration code...

    while (true) {
        // Your looping code...
    }
}

ISR(USART_RX_vect) {
    // Your ISR code...
}
```

Task 2B [1.5%]: UART Controlling an LED by interrupts with a BSL component

Same behavior as Task 2A, but this time with a BSL component.

You **must used** the same BSL USART component interface `bsl_Usart.h`. Your BSL USART component implementation **by interrupts** should contain your ISR for receiving characters as well as your USART interface configuration (in `bsl_Uart_Init`) in the following implementation file:

```
// bsl_UsartByInterrupts.c

#include "bsl_Usart.h" // Share the same interface (polling vs interrupts).

#include <avr/io.h>
#include <avr/interrupt.h>

#define BaudRate9600 ((uint16_t)(103))

volatile char dataReceived;
static   bool initialized;

//------------------------------------------------------------------------
// Receive a character from UART.
//------------------------------------------------------------------------
char bsl_Uart_RxChar(void) {
    // Your code...
}
//------------------------------------------------------------------------
// Transmit a character to UART.
//------------------------------------------------------------------------
void bsl_Uart_TxChar(char c) {
    // Not required for this task.
}
//------------------------------------------------------------------------
// Initialize UART for received interrupts.
//------------------------------------------------------------------------
void bsl_Uart_Init(void) {
    if (!initialized) {      // To make sure it will be done only once.

    // Your code...

    }
}

ISR(USART_RX_vect) {
    // Your code...
}
```

Your **task 2B** should follow the following bare C program template:

```
// Lab5Task2WithBSL.c - version with BSL using implementation by interrupts.

#include <avr/io.h>
#include <avr/interrupt.h>  // For disable/enable interrupts.

#include "bsl_Usart.h"      // Share the same interface (polling vs interrupts).

#include <stdint.h>
#include <stdbool.h>

int main() {
    // ---- Configure the onboard LED.
    // Your code...

    // ---- Configure UART by interrupts
    cli();
    bsl_Uart_Init();
    sei();

    while (true) {
        char dataReceived = bsl_Uart_RxChar();

        // Your looping code...

    }
}
```

Individual Lab Report Submission

Each student is expected to create a `.zip` file that he will upload to Moodle for grading. The following is expected in the archive:

- A folder containing your source code
- A report in **.PDF** format, no other formats will be accepted

The individual lab report is expected to contain the following for each task:

1. Your code must be commented on by explaining the logic behind it.
   The code for each task should be in separate source files.
2. Discussion and conclusion.

Create a `.zip` file with the following signature:
Soen422_LabSection_Lab5_Lastname_StudentID.zip

Due 11:59 PM - Tuesday, November 9, 2021