KETTERING UNIVERSITY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CE-426-01

Cortex-M3 General Purpose I/O and Interrupts

Darek Konopka

May 4, 2021

Table of Contents

1. Objectives

- ❖ Create a project for interfacing to Cortex-M3 general purpose I/O
- ❖ Use polling and interrupt methods for interfacing to UART

2. Program Source Code

2.1 uart.h

```
extern void USART1_Init (void);
extern uint8_t SendChar (uint8_t ch);
extern uint8_t GetKey (void);
extern void USART1_IRQHandler (void);
```

2.2 uart.c

```
#include <stm32F10x.h>
#include "GPIO_STM32F10x.h"

volatile extern uint8_t inKey;

/*----------------------------------------------------------------------------
  Initialize UART pins, Baudrate
 *----------------------------------------------------------------------------*/
void USART1_Init (void) {
  int i;

  RCC->APB2ENR |=  (   1UL <<  0);     /* enable clock Alternate Function  */
  AFIO->MAPR   &= ~(   1UL <<  2);     /* clear USART1 remap               */

  RCC->APB2ENR |=  (   1UL <<  2);     /* enable GPIOA clock               */
  GPIOA->CRH   &= ~(0xFFUL <<  4);     /* clear PA9, PA10                  */
  GPIOA->CRH   |=  (0x0BUL <<  4);     /* USART1 Tx (PA9) output push-pull */
  GPIOA->CRH   |=  (0x04UL <<  8);     /* USART1 Rx (PA10) input floating  */

  RCC->APB2ENR |=  (   1UL << 14);     /* enable USART#1 clock             */

  USART1->BRR   = 0x0271;              /* 115200 baud @ PCLK2 72MHz        */
  USART1->CR1   = ((   1UL <<  2) |    /* enable RX                        */
                   (   1UL <<  3) |    /* enable TX                        */
                   (   0UL << 12) );   /* 1 start bit, 8 data bits         */
```

```
  USART1->CR2   = 0x0000;              /* 1 stop bit                    */
  USART1->CR3   = 0x0000;              /* no flow control               */
  for (i = 0; i < 0x1000; i++) __NOP();   /* avoid unwanted output            */

  USART1->CR1  |= ((   1UL << 13) );   /* enable USART                  */
}


/*----------------------------------------------------------------------------
  SendChar
  Write character to Serial Port.
 *----------------------------------------------------------------------------*/
int SendChar (uint8_t ch)  {

  while (!(USART1->SR & USART_SR_TXE));
  USART1->DR = ((uint16_t)ch & 0x1FF);

  return (ch);
}


/*----------------------------------------------------------------------------
  GetKey
  Read character to Serial Port.
 *----------------------------------------------------------------------------*/
uint8_t GetKey (void)  {

  while (!(USART1->SR & USART_SR_RXNE));

  return ((uint8_t)(USART1->DR & 0x1FF));
}

/*----------------------------------------------------------------------------
USART1 IRQ HandlerThe hardware automatically clears the interrupt flag, once the
ISR is entered
*----------------------------------------------------------------------------*/\


void USART1_IRQHandler (void) {

    inKey = (int8_t) (USART1->DR & 0x1FF);

}
```

2.3 main_sample2.c

```
/*
```

```
   Sample program 2 - Hello World for Embedded Systems
   Toggle an LED attached to a GPIO pin
      Use UART inputs to control the operating mode of the program
*/

#include <stm32f10x.h>
#include "GPIO_STM32F10x.h"
#include "Board_LED.h"
#include "uart.h"

#define  LED_0     0                  //this is connected to GPIOB pin 8
(PORTB.8)
#define  LED_1     1                  // Pin 9
#define  LED_2     2                  // Pin 10

// Prototype Functions
void delay (uint32_t count);
void SendText(uint8_t *txt);
void USART1_IRQHandler (void);

// Calls this from uart.c
volatile uint8_t          inKey;

int main()
{

      /* Improved UART code */
      USART1_Init ();//Configure and enable USART1 interrupt
      NVIC->IP[USART1_IRQn] = 0x80;//set priority to 0x80
      //NVIC_SetPriority(USART1_IRQn, 0x80);
      NVIC->ISER[USART1_IRQn/32] = 1UL << (USART1_IRQn%32);//set interrupt enable
bit
      //NVIC_EnableIRQ(USART1_IRQn);
      USART1->CR1 |= USART_CR1_RXNEIE;//enable receiver not empty interrupt


      LED_Initialize();

      //send a character to UART for testing
      SendChar('H');
      SendChar('i');
      SendChar('\n');           //send newline character

      //use your function SendText to send a text string to the UART
      SendText((unsigned  char*) "Welcome!\n");

      while (1) {
            //inKey = GetKey();             //get input character from the UART

            //include a command here to echo the input you received back to the
```

terminal window

```
            // Normal Mode
            if (inKey == '1') {
            // Turn on the GREEN LED (while RED and YELLOW are off)
            LED_Off(LED_0);     // Turn off Red Light
            LED_Off(LED_1);     // Turn off Yellow Light
            LED_On(LED_2);      // Turn on Green Light

            //Wait for 3000 time units
            delay(3000);

            // Turn on the Yellow LED (while RED and Green are off)
            LED_Off(LED_0);     // Turn off Red Light
            LED_Off(LED_2);     // Turn off Green Light
            LED_On(LED_1);      // Turn on Yellow Light

            //Wait for 1000 time units
            delay(1000);

            // Turn on the Red LED (while Green and YELLOW are off)

            LED_Off(LED_1);     // Turn off Yellow Light
                                        LED_On(LED_0);     // Turn on Red
Light
            LED_Off(LED_2);     // Turn off Green Light

            //Wait for 3000 time units
            delay(3000);

            // Override mode
            } else if (inKey == '0') {
            /* While in this mode only the RED LED blinks on and off
            every 1000 time units(while the YELLOW and GREEN are off). */

            LED_On(LED_0);      // Turn on Red Light
            LED_Off(LED_1);     // Turn off Yellow Light
            LED_Off(LED_2);     // Turn off Green Light

            delay(1000);

            LED_Off(LED_0);                   // Turn off Red Light
            LED_Off(LED_1);     // Turn off Yellow Light
            LED_Off(LED_2);     // Turn off Green Light

             delay(1000);

            } else {
            // Made a Reset mode for fun, if the mode is not 0 or 1 then all LED
are off
```

```
                LED_Off(LED_0);     // Turn off Green Light
                LED_Off(LED_1);     // Turn off Yellow Light
                LED_Off(LED_2);     // Turn off Red Light
                }
        }
}


/*---------------------------------------------------------------------------
  Simple delay routine
*---------------------------------------------------------------------------*/
void delay (uint32_t count)
{
        uint32_t index1, index2;

        for(index1 =0; index1 < count; index1++)
        {
                for (index2 = 0; index2 < 1000; index2++);
        }
}

//complete this function for sending a string of characters to the UART
void SendText(uint8_t *text) {

        unsigned int i = 0;

        // Iterate until the terminating null character, '\0', is reached
        while (text[i] != '\0'){
                // print character
                SendChar(text[i]);
                // increment i
                i++;
        }

  return;
}
```

3. Output

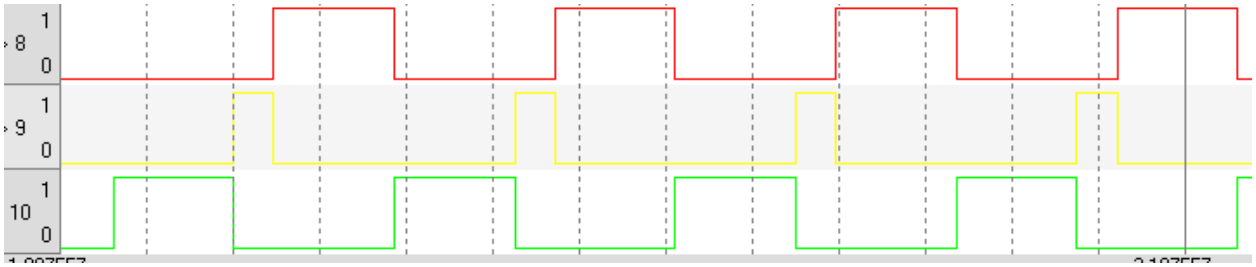

Figure 1: Logic Analyzer in Normal Mode

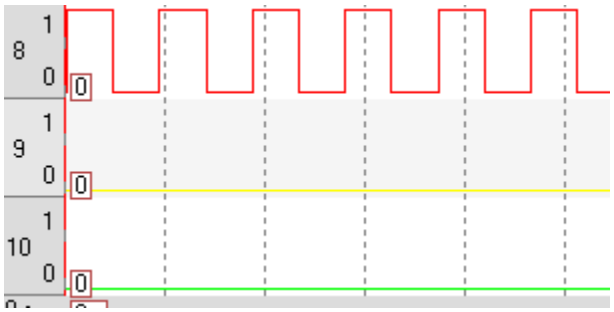

Figure 2: Logic Analyzer in Override Mode

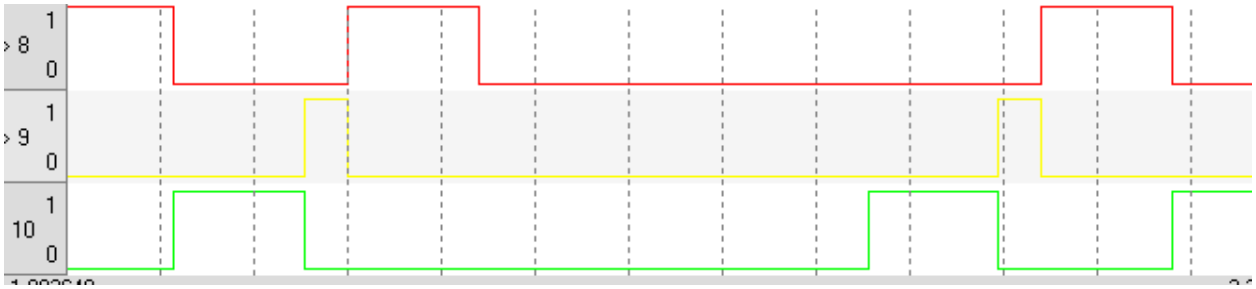

Figure 3: Logic Analyzer going from Normal to Reset back to Normal Mode

4. Questions

4.1 Question 1

What does your program do? How do you interact with it? What are the acceptable input values from the user and how do they relate to what the program does?

> The program uses the UART Terminal to respond to user input. If a user types in 1, then it will turn LED_0 on and off, and if the user types 0 into the terminal, the LED_0 is turned off.

4.2 Question 2

You may have noticed that the following function call in "main_sample2.c" does not do anything:

```
SendText("Welcome!\n");
```

How do you fix this so that the program outputs the specified message on the UART #1 terminal window? Implement it in your program.

> This can be done by iterating over each character in the string sent to the SendChar function until the array of characters (or string) is complete. Each character is printed with the SendChar function until reaching the '\0' character. Since strings are null-terminated and '\0' is the null character, we know that this will always be the last character in the string.

5. Conclusions

This lab has taught us how to use tools to interface with UART, such as the logic analyzer. We also learned how to work with interrupts, and header files as well. The greatest lesson we learned was how to read user input via the UART terminal and as a result change the GPIO pins.