Lab 04

Beverly Yee

u0770041

Prelab Questions

1. What is the difference between a parallel and serial interface? (4.3.1)

**The way they transfer data. Parallel uses multiple wires that represent a single binary bit and receives data all at once. Serial uses a single wire that stacks up the data bits in consecutive order and streams it over a period of time.**

1. What is the difference between a synchronous and asynchronous interface? (4.3.2)

**Synchronous interfaces uses a physical clock and are simpler in design, although they require an extra clock connection. Asynchronous interfaces either use a virtual clock or estimate the time intervals between signals. Asynchronous interfaces are more complex and do not have as good of data rates as Synchronous interfaces.**

1. What is one thing a communication protocol does? (4.4.2)

**It defines the collection of bits received in an input signal into useful data.**

1. What does the baud rate of a signal mean? (4.6.2)

**The frequency in between transmission of bits and is usually determined by the number of bits per second.**

1. What register in the USART would you use to enable the transmitter hardware? (4.7.1)

**USART\_CR1, Control Register 1**

USART\_CR1 – en/den interrupt conditions and portions of USART peripheral

USART\_CR2 – control signal polarity and routing

USART\_CR3 – manages hardware *flow* control, DMA, and Smartcard interface

1. Does the transmit (TX) line of the USB-USART cable connect to the transmit (TX) or the receive (RX) of the STM32F0? (4.8.1)

**The receive (RX) of the STM32F0**

Notes

4.1—Preparing to use the USART

The STM32F072 has four USART peripherals available; you will need to select one of these to use. We suggest using USART3, which has had the best success for students: the connections for USART1 and USART2 use pins that are unavailable on this board (see Discovery Board Manual Section 4.14 for Extension connectors to view these pins).

Each USART has a small selection of GPIO pins that can be used as its transmit (TX) and receive (RX) signals. Because the split connections on the end of the Adafruit USB-USART cable are fairly short, you will want to choose USART output/input pins that are relatively near a GND pin.

1. Using the chip datasheet, locate pins that connect to TX/RX signals on USART peripherals.

Find pins for USART3

1. Choose a set of RX/TX pins that are near enough to a GND connection such that the USB-USART cable’s wire ends can reach.
2. Connect the USB-UART transmit (TX) line to the STM32F0’s receive (RX) pin. Likewise, connect the STM32F0’s transmit (TX) pin to the USB-UART receive (RX) line.
3. Set the selected pins into alternate function mode and program the correct alternate function number into the GPIO AFR registers.
4. Note the USART that the selected pins connect to, this is the peripheral you will be using in the lab exercises.

4.2—Blocking Reception

1. Check and wait on the USART status flag that indicates the receive (read) register is not empty.

* You can use an empty while loop which exits once the flag is set or simply check each iteration of the main infinite loop.
* It may be helpful to carefully read the bit descriptions in the register map.

1. Test the received data and toggle the appropriate LED

* The receive register can be read like an ordinary variable. However, the data isn’t guaranteed to remain in the register after it has been read once.
* Unless you use a switch statement you may want to save the value into a local variable and test against that.

1. Whenever a key is pressed that doesn’t match an LED color, print an error message to the console.
2. You will probably want to comment or remove any old transmit code and delay statements from the infinite loop.

* The possible reception rate at 115200 Baud is faster than the minimum delay the HAL library functions are designed to provide.
* It is possible to lose received data while waiting for a blocking transmit to complete. Where we are only receiving single bytes it is unlikely to cause problems.

4.3—Interrupt-Based Reception

1. Your command parser must now accept two character commands.

* The first character is a letter matching the one of the LED colors.
* The second character is a number between 0 and 2.
  + ‘0’ turns off the LED
  + ‘1’ turns on the LED
  + ‘2’ toggles the LED

1. Print a command prompt such as “CMD?” when waiting for user input.
2. Entering an unknown character prints an error message and restarts back to the beginning.
3. On a successful command, print a message about which command was recognized.