ECE353 In-Class Exercise

UART Polling

Problem Objectives

- Configure GPIO Pins PA1 and PA0 to be UART pins
- Configure UART0 to be configured as 8N1, with no interrupts
- Send and Receive messages over the serial debug interface

1. Modify gpioPort.c

A. Do not modify gpioPort.h. It has been updated to include the functions described below and some new bit masks.

B. Copy your version of gpioPort.c into the ./**drivers** folder.

C. Modify your gpioPort.c file to implement the following functions

```
a. bool  gpio_config_alternate_function(uint32_t baseAddr,
   uint8_t pins);
b. bool  gpio_config_port_control(uint32_t baseAddr, uint32_t
   bit_mask);
```

2. Modify uart.c

Implement **`bool uart_init_115K(uint32_t base_addr)`** so that UART0 is configured for a baud rate of 115200, 8N1. **Do not enable the hardware FIFOs or Interrupts.**

3. Modify main.c

A. Complete the function **`uart0_config_gpio`**. It should initialize PA0 and PA1 so they behave as UART pins. When setting the port control mask, make sure to examine **`gpioPort.h`** and use the appropriate bit masks for the PCTL register.

B. Add code to call **`uart0_config_gpio()`** and **`uart_init_115K()`.**

C. Print out **`greeting[]`** using **`uartTxPoll()`.**

D. Receive 4 character from the user. Each character should be transmitted back to the user as soon as it is received. Use **`uartTxPollChar()`.** Each character received should also be placed in **`rx_string`**.

E. Print out **`response []`** using **`uartTxPoll()`.**

F. Print out **`rx_string []`** using **`uartTxPoll()`.**

G. Print out **`exit_msg[]`** using **`uartTxPoll()`.**

4. Observe the Serial Output

Make sure the output strings and input are displayed on the terminal.

5. What to Turn In

Turn in **`uart.c`** to the dropbox on the course website.