MTE 241 Lab 2 -- Hardware Interfacing

Bernie Roehl, Fall 2018

The goal of this lab is to familiarize you with interfacing to the various peripherals on the MCB 1700 board.

Learning Objectives

After successfully completing this lab, you should be able to write code to interface to off-chip peripheral devices at a low level, including:

- Digital inputs and outputs such as LEDs, pushbuttons and digital joysticks
- Analog devices such as potentiometers, using analog-to-digital converters

Overview

The Keil MCB 1700 board provides a number of peripherals that you can make use of. You are already familiar with the use of the UART (i.e. serial port) from Lab 1. The additional peripherals you will be using in this lab are as follows:

- LEDs
- Joystick
- Push button
- Potentiometer

Input and Output on the LPC 1768

The LPC 1768 processor is very flexible, and it's possible to configure many of its pins to perform up to four different functions. For example, pin 60 of the processor can be used as any of the following:

- DCD1 (Data Carrier Detect for serial port 1)
- MOSI (Master Out / Slave In for the default SPI interface)
- MOSI0 (Master Out / Slave In for SPI interface 0)
- GPIO (General-Purpose Input/Output)

The function for a particular pin is selected by setting appropriate bits in one of the Pin Select Registers. Fortunately, the processor defaults (on reset) to GPIO for all the pins so you do not need to explicitly configure these registers in order to access most of the peripherals. A notable exception is the analog to digital converter.

In addition to selecting the function of the various pins, each pin's data direction can be set to input or output when it's configured as GPIO. This is done by setting appropriate bits in the Data Direction registers.

The general-purpose digital input/output pins are grouped into "ports", most of which are 32 bits wide. The processor can only read or write the full port at once, so in order to change the value of individual pins you'll have to mask the bits using binary operations. To further complicate things, the bits are set by writing to one register, and cleared by writing to a different register.

All of the ports (both the data ports and the various configuration registers) are memory-mapped. In other words, each register has a particular fixed address in the memory space of the processor. The file `lpc17xx.h` contains definitions for all of these addresses, and should be #included in your source code as follows:

```
#include <lpc17xx.h>
```

The GPIO registers are represented as a pointer to a struct containing a number of fields corresponding to the various registers. For example, we'll be using `LPC_GPIO1` and `LPC_GPIO2`. Within each of those structs are fields named `FIODIR` (to specify the input/output directions), `FIOSET` (to set bits on the port to one), `FIOCLR` (to clear bits on the port to zero) and `FIOPIN` (to read the current value of the pins).

LEDs

There are eight LEDs on the board, which are hooked up to specific pins on GPIO ports 1 and 2. Which LEDs are connected to which pins can be found by examining the schematic for the board.

Joystick

The joystick is actually a four-position switch, with each position corresponding to a particular input pin. In addition, you can press in on the joystick to give you a one-bit input that is also hooked up to a specific input pin. Again, the input pins can be determined from the schematic.

Pushbutton

The pushbutton is handled the same way as the joystick.

Potentiometer

A potentiometer is a variable resistor. In the case of the Keil board, the potentiometer (the blue knob) is connected to an analog-to-digital converter on the LPC 1768 chip.

Your Goal

Your goal in this lab is to write four programs:

Part 1 -- Write a program to continually read the value of the pushbutton, and turn the topmost (or leftmost) LED on when the button is down and off when the button is up.

Part 2 -- Write a program to continually read the joystick and print a message out the serial port that indicates whether the joystick position is North, South, East, West or Centre, and whether or not it's pressed in.

Part 3 -- Write a program to repeatedly read a string that's typed by the user into the serial port, convert that string into a number, and display the number in binary on the LEDs. The number can be in the range of 0 to 255. When entering text through the serial port, you should end the input by pressing ctrl+J which is the ASCII newline character. So for example, if the user enters 75 and presses ctrl+J, the LEDs should display 1001011.

Part 4 -- Write a program to continually read the value of the analog to digital converter, and print the value out the serial port as a decimal number.

You should have a single `main.c`, and select one of four `main()` functions using conditional compilation.

Marking

Each of the four parts of your assignment are worth 20%, and your ability to answer questions about your project will account for the remaining 20% of your mark. Questions will be answered individually by each student in the group.