arm Education

Embedded Systems Essentials with Arm: Getting Started

Module 3

SV1 (3): Before you begin

Hello and welcome to this lab, which will cover digital input and output.

In this lab, we will explore the capabilities of embedded microcontrollers by implementing two functions in which a board's LEDs are controlled by the state of numerous switches. We will use the DigitalIn, DigitalOut, BusIn, and BusOut Mbed interfaces to assist with this.

By the end of the lab you will have gained insight and practical experience with the Mbed API for digital input/output.

Before starting the exercise, let us have a short review of these topics.

Digital inputs and outputs are the basis for an interaction between peripheral components and the microcontroller system. In contrast to analog inputs and outputs, which are mostly used for reading sensors or generating waveforms, digital inputs and outputs or I/Os can only provide two states: high and low, 0 or 1.

To grasp the basic concept of digital input and output, we will consider the need to have an input switch controlling two output LEDs. In one position, the processor will light the red LED, and in the other position, the processor will light the blue LED.

The circuit diagram shows how we could connect one switch and two LEDs to our Cortex processor. Note the use of additional resistors and the fact that a red LED needs a different resistor to a blue LED.

At this stage, we are not considering the code, we are just designing the interface circuitry. We use one general-purpose I/O pin or GPIO as an input, to read the signal level from the switch, and two further GPIOs as outputs, to drive the red and blue LEDs.

The purpose of pull-up and pull-down resistors is to ensure that there is a default or "known" value on the input pin to prevent an unknown state called "floating". "Floating" is a situation in which there is no input and the program cannot decide whether the pin will be high or low.

In the diagram shown here, if we would like the switch SW1 (IN) to pull the pin to the ground, we use the pull-up resistor layout.

In digital devices, the logical values 0 and 1 represent the logical state of a voltage switched low or high. These states are not necessarily connected to a dedicated voltage, but could range from 0 to 3.3 volts, or any arbitrary different voltage such as 0 to 5 volts for common microcontroller applications.

With general-purpose I/O-pins, a microcontroller usually provides voltages at the dedicated levels 0 and its maximum positive power supply +Vcc. Until recently, +Vcc, the HIGH voltage, was 5 volts; however, today's ultra-low power processors have reduced this voltage to as low as 1.8 volts.

Generally, active components such as LEDs or motors can also be driven directly with GPIOs. But as the ports generally provide quite a low current, usually 10milliampere, and low total power, since 1.8 volts times 10 milliamperes equals 18 milliwatts, the approach of directly

driven components is good for testing purposes but not very common in real-life application. In real-life applications, GPIOs control 'switches,' such as transistors, to disable or enable the power supply for other components.

The most important thing is to remember that GPIOs can only provide two voltage levels. If more than two are required, an analog input or output, or the trick of a pulse width modulation, must be used. This will be touched on in a later module.

The graph shows the range of input signal voltages that will be reliably recognized as legal logic levels. It is plotted against power supply voltage.

Most of the time, we will run our Cortex M processors at 3.3 volts. Taking this as an operating condition, we can use the graph to determine what ranges of voltages will be reliably recognized as valid logic signals:

Logic 0: 0.0–1.1 volts

Logic 1: 2.4–3.3 volts

This means that a signal of 2.0 volts will not be reliably recognized as a valid logic level because it is the "undefined" region. Notice also that input signals that stray outside the power supply voltages may damage the processor.

A simple and fast method for controlling GPIO is using the Mbed API. It provides a number of drivers that provide access to general purpose microcontroller hardware.

The key APIs for digital input/output are:

- DigitalIn
- DigitalOut
- BusIn
- BusOut

We will take a look at some examples and see how they compare to the low-level programming equivalent.

The DigitalIn interface can be used to read the value of a digital input pin, where the logic level is either 1 or 0.

The DigitalOut interface can be used to configure and control a digital output pin by setting the pin to a logic level of 0 or 1.

Any number of Arm Mbed pins can be used as DigitalIn or DigitalOut.

This simple example shows how a digital input pin called "mybutton" and a digital output pin called "Led_out" to turn on an LED if a button is pressed.

There is also a DigitalInOut interface, which is a bidirectional digital pin. It can be used to interface to read the value of a digital pin when set as an input, as well as write the value when set as an output.

The DigitalIn class provides a number of functions. The first two in the list outline how to create a DigitalIn object that is connected to a specified pin. The other functions cover actions such as reading the input, setting the input mode, returning the output setting and providing a shorthand operator for the read function.

The DigitalOut class also provides a number of useful functions. The first two provide constructors for creating DigitalOut objects that are connected to specified pins. Next, there is a function to set the output to a provided value and also read the output setting. Then there are some shorthand operators for the write and read functions.

The BusIn API allows you to combine a number of DigitalIn pins to read them at once. This is useful for checking multiple inputs together as a single interface instead of individual pins.

The BusOut API allows you to combine a number of DigitalOut pins to write to them at once. This is useful for writing to multiple pins together as a single interface instead of individual pins.

Any number of Arm Mbed pins can be used as a BusIn or BusOut. This example showcases how a bus out object is create containing 4 led pins. This can then be used to control the LEDs in a variety of ways.

The BusIn class provides a number of functions. The first two are constructors for creating BusIn objects that are connected to the specified pins. Then there exists functions for reading the value of the input bus, setting the input pin mode and a shorthand for the read function.

The BusOut class provides similar functions. The first two being different constructors for creating BusOut objects that are connected to the specified pins. Then again there are functions for writing to the output bus, reading the value of the output bus, and a couple of shorthand operators for the write and read functions.