Embedded Systems Essentials with Arm: Getting Started

Module 4

SV1 (4): Before you begin

Hello and welcome to this lab, which will cover interrupts.

In this lab, we will continue to explore the capabilities of embedded microcontrollers by controlling peripherals through a combined use of digital input/output and interrupts. More specifically we will look at two different ways of controlling peripherals using interrupts: a toggle or a counter.

By the end of the lab you will have gained insight and practical experience with the Mbed API for interrupts.

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

Consider the following example:

Using a switch, we want to change the color of an RGB LED when it is pressed. To do this, we apply an external switch to a GPIO pin on the controller. In order to prevent the GPIO pin from being 'open', we will use a pull up resistor, as shown here.

There are two ways to detect whether a switch is pressed or not: polling and use of interrupts

Polling is the method by which the microcontroller continuously checks the status of a flag. It is slow, inefficient and poorly scalable. The more GPIO or other events have to be polled, the more time is needed. So, there is a natural limit to the number of supervised GPIOs possible at a given processor speed.

However, an interrupt-based routine only runs the code when it is called or initiated by the interrupt-service routine. This routine directly executes addressed code, sometimes without any CPU interaction. Therefore, it is fully scalable, because it is independent from the number of monitored events, so the supervision doesn't consume CPU time.

An interrupt service routine is the name of the code being forced into execution by an interrupt.

The Mbed API provides an interface for utilizing interrupts in an Mbed application. It can be used to trigger events upon digital input changes on a target microcontroller. These interrupts can be triggered on the rising or falling edge of signals.

The InterruptIn class provides a number of functions for interrupts. These include constructors for creating InterruptIn objects that are connected to specified pins, the ability to attach functions to be called upon rising edges or falling edges occurring, and setting the input mode.

This example demonstrates how the InterruptIn API can be used to count rising edges on a pin.

First of all a class named "Counter" is defined. In the public section of this class is a constructor and two functions.

The constructor creates an InterruptIn object on the pin specified to the counter. Also, when the object is created an increment function of this counter instance is attached to the InterruptIn object.

The function "increment()" simply increments a variable called "_count", while the "read()" function returns the value of the variable "_count.

In the private section of the class, two variables are declared, an InterruptIn object called "_interrupt" and a volatile integer called "_count". These are used in the previously described functions/constructor.

After this the line "Counter counter(SW2)" creates a counter object using the constructor and includes "SW2" as the pin parameter.

Then in the main function a while loop is run which prints out the counter value using the read function.