Embedded Systems Essentials with Arm: Getting Started

Module 4

KV3 (4): Exception Processing: Examples

Let’s look at an example of using a switch and interrupt service routine to control an LED, and inspect the code in a bit more detail. This code example is written in C without use of the Mbed API, but serves as an example for the practicality of exception handling.

The first few lines show the including of important header files as well as the definition and initialization of a counter.

The second block includes the code for the interrupt service routine or ISR, which contains the functional code of incrementing the counter. The ISR is activated by the press of the switch connected to the pin, triggering the interrupt. The RGB LED is controlled by the counter; once the counter is incremented by the ISR, the LEDs are altered.

In the main routine, seen in the third box, general setup is carried out and the ports are initialized and connected to the ISR.

The previous example was written in plain C. 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 first two functions, “InterruptIn(PinName pin)” and “InterruptIn(PinName pin, PinMode mode)” are constructors that can be used to create InterruptIn objects that are connected to specified pins.

The API also provides a “read” function which can be used to read the value from the input pin, which will be represented as a 0 or 1.

Also, the two functions “rise” and “fall” can be used to attach a designated function to call when a rising or falling edge occurs, respectively.

The “mode” function allows the input pin mode to be set, for example, PullUp or PullDown.

Finally, the “enable” and “disable irq (interrupt request)” functions enable or disable interrupt requests respectively.

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

First, 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.