Lab Sheet 5: Analog Input

1 Aims

The aim of this lab is to explore the use the Analog to Digital Convertor (ADC). When you have completed this lab, you should:

- Know how to use an ADC in a program.
- Understand the principles of configuring the ADC.

Assessment. If you wish, you can present work on this lab exercise for assessment. Complete any of the following steps (see QM+ for full details and deadlines)

- **Answer the questions on the QMPlus quiz.** *You are recommended to look at the questions about the given code as you go along as they may help you complete the rest of the lab.*
- **Demonstration in the lab.** Ask one of the demonstrators to see your code working during the scheduled lab times.
- **Code assessment.** Complete the QM+ assignment to indicate that your code is to be assessed, provided you have demonstrated it. Do not attach any documents. The only way to submit code is to update your repository on GitHub. You can push code as often as you wish, whether or not you would like it assessed.

1.1 Overview of Activities

Perform the following activities.

1. Activity 1: Download, read, compile and run the sample program. Use the debugger to record the ADC measurements of voltages controlled by VR1 and VR2 (the potentiometers on the shield)
2. Activity 2: Poll the voltage readings periodically.
3. Activity 3: Use the voltage readings to control the flashes of the LEDs.
4. Activity 4 (Bonus): Automatic calibration.

1.2 The Behaviour to Implement

Activities 2 and 3 combine to create the new system. The work is broken down into two activities to make it easier, but you can do both activities together if you wish.

The overall requirements are:

- The shield LEDs flash together (i.e. all on or all off)
- The on time is controlled by VR1
- The off time is controlled by VR2
- The maximum on time or off time is 3 seconds.
- The minimum on time or off time is 0.5 seconds
- Shield button B1 turns the system on and off
- When the system is off, the three colour LED (on the FRDM board) shows red.
- When the system is on, the three colour LED (on the FRDM board) shows green.
- Any design requirements given must be followed.

There are additional requirements for the bonus part.

1.3 Other Information

Additional information is in the following appendices:

- Appendix A: recaps the operation of a voltage divider.
- Appendix B: has notes on the ADC operation.

2 Activity 1: Using the Sample Project

2.1 Download, Read, Compile and Run

Download the sample project. The project:

- Contains code to initialise the ADC and use it in two modes:
  - Single-ended mode
  - Differential mode.
- Pressing button B1 changes between the red and green LEDs on each button press. The red LED is on initially.
- Takes a reading of the voltage for VR1 with the ADC when the button is pressed, using both the single-ended (when red LED lit) and differential modes (when green LED lit).

2.2 Reading the Provided Code

Review the provide code carefully. The code is divided into the following files, plus headers:

1. main.c – code for the cyclic system
2. SysTick.c – functions for using the SysTick counter.
3. rgb.c – contains code for using the RGB LED on the development board
4. led.c – code for the shield LEDs
5. button.c – code for shield buttons
6. adc.c – functions for using the ADC to read the potentiometers

Answer the questions in the QMPlus quiz.

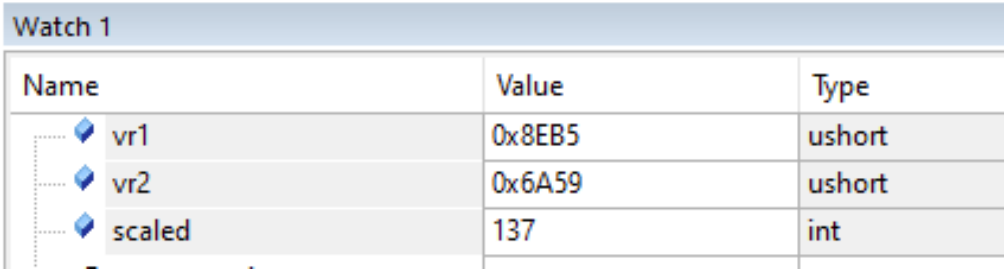
2.3 Use the debugger to record the ADC measurements

The voltages to be measured are those controlled by VR1 and VR2, the two potentiometers on the shield. Use the debugger to record the ADC readings. All the code needed for this section is already provided. The aim is to complete the following table:

Potentio-meter	Min		Max	
	Raw	Scaled	Raw	Scaled
VR1	0x0002	0	0xE71F	298
VR2	0x0002	0	0xE71F	297

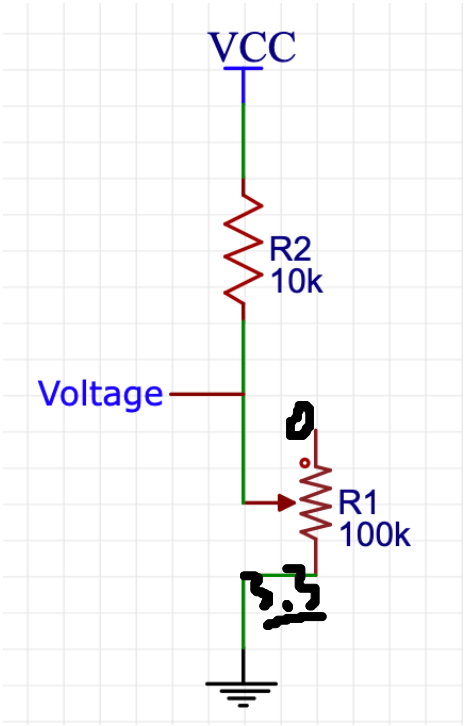
The following variables are used:

- 'vr1', 'vr2' hold the value returned by the ADC. This is in the range 0 to 0xFFFF for 0v and 3.3v respectively. However, the maximum voltage at the measured point is less that 3.3v.
- 'scaled' holds the voltage as a scaled integer, so 2.3 is represented by 230.



The variables can be viewed using a 'watch' in the debugger. Remember to press B1 to update the measurement.

2.4 The Potentiometers



See the voltage divider theory in Appendix A.

The monitored voltage is changed by the potentiometer (variable resister). There is a fixed resistor of 10K Ohms in series with the variable one, whose resistance ranges from 0 to 100K Ohms. However, these values are nominal and some differences may occur.

- When the variable resistor has its minimum resistance, the voltage at the measured point is close to zero
- When the variable resistance has its maximum resistance, the voltage at the measured point is closer to supply voltage (3.3v) but does not reach it because of the 10k Ohm fixed resistor.

We can use PIT as this is the only thing that we have learnt that can make this work.

3 Activity 2: Poll the voltage readings periodically

In the provided code, the ADC measurement is made when B1 is pressed. It is more useful to measure the voltage 'continuously'.

- The measurement frequency should be 50 Hz (i.e. a measurement every 20 ms).
- You can choose to make measurements on the two voltages on alternate cycles or to measure both together every other cycle.
- The two global variables vr1 and vr2 can be used to pass the voltage measured between tasks.

Make the necessary changes to task2MeasureVR and observe the changes using the debugger, without needing to press B1.

4 Activity 3: Use a Variable Resistor to Control the Flashes of an LED

The requirements are given in section 1.2 above.

4.1 Timing Requirements

The two varying voltages control the on and off times of the shield LEDs. The ratio between the current voltage '*v*' and the on (or off) time *t* is shown below:

0 - 1
current voltage/ max voltage

$$\frac{v - v_{min}}{v_{max} - v_{min}} = \frac{t - 0.5}{3.5}$$

t max can be 4 seconds.

Therefore, the state change occurs when the current time *t* satisfies the following relationship.

counter >= 350 * scaled + 50*298 on time.

if voltage is 0.5, then the time is on for, is:
(0.5 *3.5)+0.5 = 2.25

$$t \geq \frac{3.5\ (v - v_{min})}{v_{max} - v_{min}} + 0.5$$

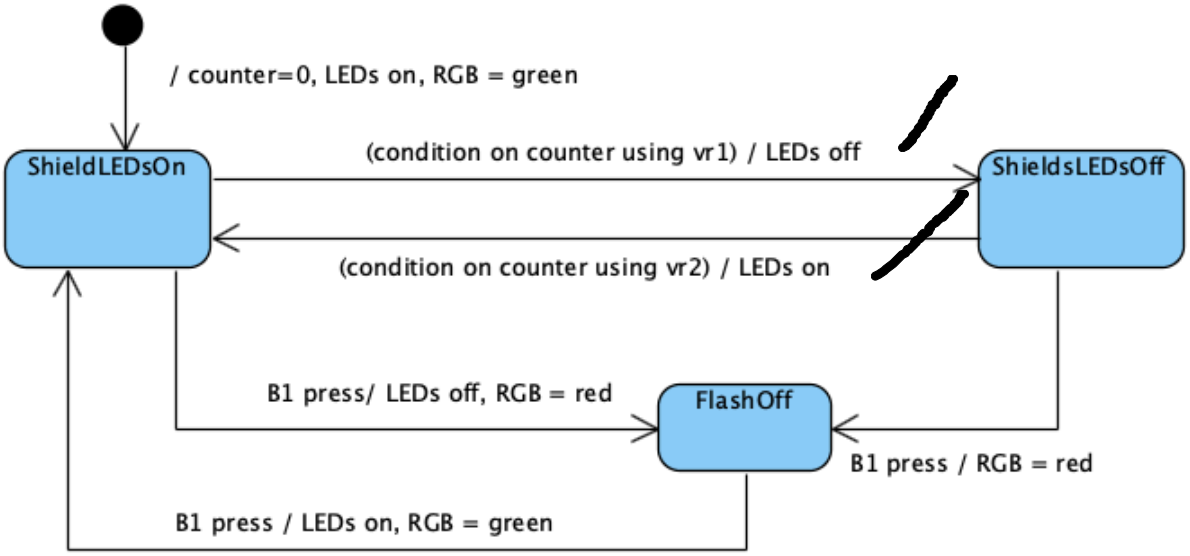
In this formula, the time *t* is in second but it is easier to work in cycles *c*. Recall that *t* is equal to the count cycles (*counter*) times the cycle time (10 ms). **In this application, the counter counts up**. Using this, the transition occurs when the cycle count satisfies the following condition.

$$counter \geq \frac{350\ (v - v_{min})}{v_{max} - v_{min}} + 50$$

You should implement this using integer arithmetic, avoiding division. [*The CPU does not have floating point hardware, so floating point operations are slow.*] Rearrange the formula as necessary.

4.2 State Transition Model

Implement the following state transition model which corresponds to the requirements.



4.3 Software Design Hints

The provided code uses a cyclic system.

- The task in the given code that polls the button B1 is retained.
- The second task regularly measure the voltage (Activity 2). It does not matter that this is unchanged when the LEDs are switched off: the voltages can be ignored.
- Use a third task to interpret the state diagram above, control the LEDs.
- In a cyclic system, you can safely use global variables (vr1 and vr2) to pass the measured voltages between the tasks.

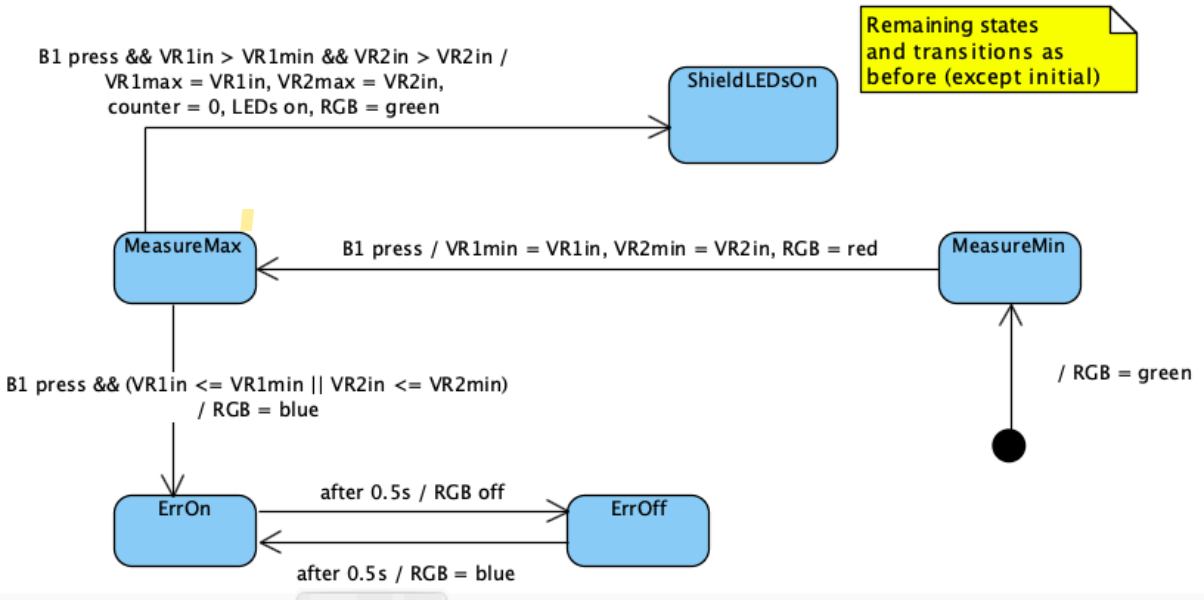
5 Activity 4 (Bonus): Automatic Calibration

A disadvantage of the code as written so far is that the values V_{max} and V_{min} must be entered into the code. It would be more convenient if we could get these automatically; we will call this 'automatic calibration'.

5.1 Additional Requirements for Automatic Calibration

1. After the system starts up the three colour LED shows green.
2. When the button B1 is pressed the measured voltages on VR1 and VR2 are used as the minimum.
3. The three colour LED then shows red.
4. When the button is pressed again the measured voltage is used as the maximum.
5. Providing **both** the maximum values exceed their corresponding minimum values, the system operates as before (initially with a maximum on period).
6. If the maximum does not exceed the minimum, then the system entered an error state: the three colour LED flashes blue on / off with 1 sec period and equal on and off times (i.e. 0.5s on).

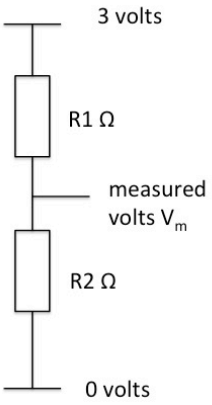
You must follow the diagram below showing the additional states.



Warning: if you choose a value for V_{min} that is above the minimum, then it is possible for the term $v - v_{min}$ to be negative. Ensure that your implementation of the condition does not misbehave and both minimum and maximum times are respected.

6 Appendix A: Voltage Divider

A voltage divider creates a voltage between the supply voltage and zero (ground). The potentiometers create a .



$$i = \frac{3}{R1 + R2}$$

$$V_m = i \times R2 = \frac{R2}{R1 + R2} \times 3 \text{ volts}$$

If the supply voltage is V (shown as 3v in the diagram), the measured voltage should then be R2 / (R1 + R2) x V volts. The total resistance R1 + R2 should ensure that the total current is below 4 mA. Given the supply voltage, at least 1 KΩ should be used.

7 Appendix B: Principles of the Analog to Digital Converter (ADC)

An ADC is a way for the MCU to read (input) a value that varies continuously over some range. This value – for example an angular position, or a temperature – must be converted to a voltage, which is read by the ADC. The ADC works by comparing the voltage with a reference voltage – here 3.3v.

7.1 Accuracy, Time and Configuration

To do the conversion, the ADC must take some time (and draw some current). The conversion time depends on the accuracy in bits and the speed at which the ADC operates. The software provided configures the ADC to operate with 16-bit accuracy.

7.2 Modes of Operation

The ADC has many modes of operation, distinguished by:

- *What triggers the conversion?* We will use a software trigger – i.e. the conversion starts when the program sets the necessary bit; the program then waits for the conversion to complete.
- *What is measured?* The ADC has two modes a) measuring the voltage on a pin relative to the supply voltage (3.3v) b) measure the difference between two voltages. We will use only the first mode.

Differential Conversion: In this mode, the ADC can be used to compare two voltages and give the difference, which can either be positive or negative. The connections are referred to as DPn (plus) and DMn (minus). If the voltage on DPn > voltage on DMn, then the difference is positive. If it is the other way around, the difference is negative. The result is a 16 bit signed number, in the range $\pm$ 2^{15}. The two voltages are in the range 0v to 3.3v (as before) but the difference can be between -3.3v and +3.3v, so that the resolution is halved compared to a single ended conversion.