# Practical 3B

# Introduction to Microcontrollers - GPIO inputs

**Before the practical ensure that you complete the following:**

*Read and revise:*

**Chapter 6 - C for microcontrollers** (pp. 134 to 144) and **Chapter 7 - General purpose input output** (pp. 145-161) and in the **EEE2046F/EEE2050F course notes**. Please note that the practicals for this course contain a large self-study component and therefore be prepared before the start of the practical.

*Bring the following to the lab session :*

- 1 x UCT STM32F0 development board
- A PC/laptop running Eclipse with the ARM toolchain setup. All lab PC's in the departmental Red Lab have this preinstalled and configured.
- A USB-A to USB-B cable

## 3B.1 Introduction

The aim of this practical is to introduce you to basic C programming for microcontrollers. In this practical you are introduced to the Eclipse Integrated Development Environment (IDE), which you will use to program and debug your UCT STM32 development board throughout the semester. You will then investigate the general purpose output (GPIO) capability of your STMicroelectronics STM32F0 Microcontroller. You will also learn how to read the microcontroller reference manuals to identify which registers are needed to control the system's operation.

This practical contains both coding and a report submission and may take longer to complete than the 2 hour practical session. Please ensure you copy and paste your code from each question into a document and that it is well formatted so that it can be easily marked after the practical. Submit both your document and **.c** file on VULA by the due date. You are also welcome to complete the coding during free sessions in the Red Laboratory.

Please name your practical report as follows:

**Prac3B-STUDENTNUMBER**

Call a teaching assistant or tutor if you need assistance at any stage during the practical session.

## 3B.2 Eclipse Integrated Development Environment (IDE)

Eclipse is an open source *Integrated Development Environment* (IDE) used for editing, compiling, debugging and programming. It can be used to produce code for a wide variety of platforms and can work with a number of programming languages. We will be using it to produce C code to program our STM32F051 microcontroller. Please see Appendix B.1 for more details.

- Open Eclipse on your computer
- Create a new project and name it as follows:

**Prac3B-STUDENTNUMBER**

- Setup the project environment as described in the instructions in Appendix B.1

**NOTE:** Clear, concise and well commented coding is encouraged. Please ensure that you explain all steps in your code and follow good coding practice.

Connect your UCT development board to the PC using the USB cable. You are now ready to begin your task.

## 3B.3 Task

Design an *8-bit UP/DOWN binary counter* using the STM32F051C6 microcontroller and your development board. All your code must be written in C.[1] The 8-bit binary count will be displayed on the 8 LEDS on the development board. The direction of the count (either UP or DOWN) will be controlled by pressing the push buttons SW1 and SW2. The layout of the devices on the UCT development board can be seen in figure 3B.1.

[1] Note: Although there are peripheral libraries available for the STM32F0 devices, we will not be using these in this course. We will learn how to create our own peripheral libraries instead.

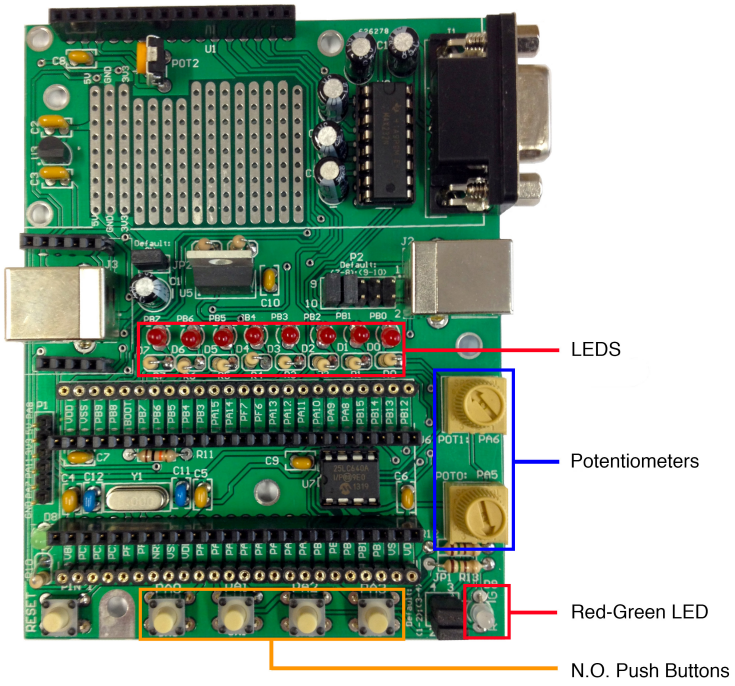


**Figure 3B.1: The simple input/output devices on the UCT STM32F0 development board** The basic I/O devices on the development board have been labelled as follows: **red** - digital outputs which have been connected to LEDS, **orange** - digital inputs which have been connected to normally open push buttons and **blue** - analogue inputs connected to two potentiometers.

Each port's operation is controlled by a number of GPIO peripheral registers as well as a *Reset and Clock Control* peripheral register ( **`AHBENR`**) which define the following:

- The AHB bus clock connections to the GPIO port **`RCC->AHBENR.`**
- The register's mode (Digital input, digital output, alternative function such as SPI, $I^2C$ etc. or analogue mode)
- GPIO port's output type (Output enabled as push/pull or enabled as open-drain)
- GPIO port's output switching speed (low, medium or high)
- GPIO port's internal pull up or pull down resistor enable
- Used to read GPIO port input data register
- GPIO port output data register
- GPIO port set or reset the output data register
- Registers to select an assigned alternative function for a port pin

The internal structure of a single pin in a port can be seen in figure 3B.2.

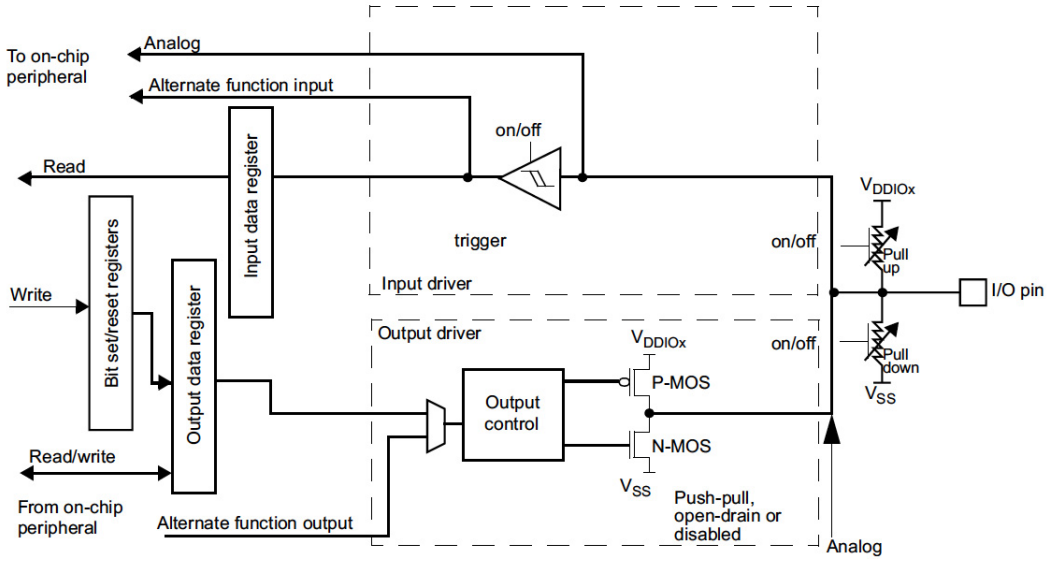


**Figure 3B.2: The internal structure of a GPIO pin in the STM32F0 microcontroller.** (Taken from [16])

## 3B.4 Questions and coding

Please answer the following questions in your practical report and where necessary modify or write code into your **main.c** file. Build and test your code after each code modification step.

(a) Using the schematic for the UCT STM32F051 Dev. Board at the end of the practical sheet (figure 3B.3), identify which port(s) the LEDS are connected to on the UCT STM32 development board and which registers are needed to control their operation. **[5]**

(b) Again using the schematic for the UCT STM32F051 Dev. Board , identify which port(s) the Normally Open (N.O.) pushbuttons are connected to on the UCT STM32 development board and which registers are needed to control their operation. **[5]**

(c) What values must be written to the registers in (a) if you wish to output digital values to the LEDS and turn them off at the start of the program? **[3]**

(d) What values must be written to the registers in (b) if you wish to use push buttons `SW0 (START)`, `SW1 (UP)` and `SW2 (DOWN)` as digital inputs? **[3]**

(e) Write a C function in your `main.c` file called `InitPorts()` which initialises the ports as described in (a) and (b). Copy this function and paste it in your report. **[2]**

(f) In your `main` loop in your `main` function write code which will turn LED `D0` ON and keep it ON when `SW0` is pressed. Copy this function and paste it in your report. **[2]**

We are now going to write a C program to produce a 8-bit UP/DOWN binary counter. The counter must start counting up from `0` when `SW0` is pressed once. If `SW2 (DOWN)` is now pressed momentarily, the counter must start to count DOWN from its current value. If `SW1 (UP)` is now pressed momentarily, it must start counting UP again from its current count value. It must reset itself to `0` when it reaches the end of its UP count and reset itself to `255` when it reaches the end of its DOWN count. It should then KEEP counting UP or DOWN depending on the previous count direction (loop operation) until `SW0 (START)` is pressed, which will RESET the counter to `0`. The counter must count up or down in increments of approximately 1 second. Create a function called `Delay()` which uses two nested `for` loops to create an approximate 1 second delay. Create

two values `DELAY1` and `DELAY2` which are initialised at the start of the program. These will be used as the values for your `for` loops to count up to. You can therefore change the length of your delay easily.

(g) Write a function called `CountUp(char value)` which takes in a character value, displays it on the LEDs, increments it, and then returns the new value. Copy this function into your report. **[5]**

(h) Write a function called `CountDown(char value)` which takes in a character value, displays it on the LEDs, decrements it, and then returns the new value. Copy this function into your report. **[5]**

(i) Complete the program, so that it only starts counting when `SW0` has been pressed and then `SW1` and `SW2` can be used to change the direction of the count. Ensure that your code is well commented. Copy your code into your report. **[5]**

Ensure that your **main.c** file is well commented and **copy the entire contents of the main.c file and paste it onto a new page in your report**.

## 3B.5 Practical Submission

Submit your completed practical report on VULA as a **.pdf**[2] under the correct assignment. Show all your calculations.

Your document must be named as follows so that it is easily identifiable:

**Prac3B-STUDENTNUMBER.pdf**

Additionally you must submit your **main.c** as a separate file on VULA. It must be renamed as follows so that it is easily identifiable:

**Prac3B-STUDENTNUMBER.c**

Do **NOT** zip the files together but upload them individually to VULA. Ensure that your name and student number is clearly written in your practical report. Practical reports must be completed and submitted on VULA by 23h55 by the due date on the VULA calendar.

[2] **Note** if it is submitted as a **.doc/.docx** or equivalent format file, it will **NOT** be marked.

## 3B.6 Marks Breakdown

| **Marks** | |
|---|---|
| **Questions** | |
| (a) | 5 |
| (b) | 5 |
| (c) | 3 |
| (d) | 3 |
| (e) | 2 |
| (f) | 2 |
| (g) | 5 |
| (h) | 5 |
| (i) | 5 |
| **Code** | 5 |
| **Total** | **40 Marks** |

Up to 5 marks will be deducted for untidy reports or uncommented code. **5% will be deducted per day for late hand in's for up to one week, after which you will receive 0 and the practical report will no longer be accepted.** Please ensure that you submit the files in the correct place and read the instructions given on VULA with regards to file naming. **NO EMAILED** pracs will be accepted.

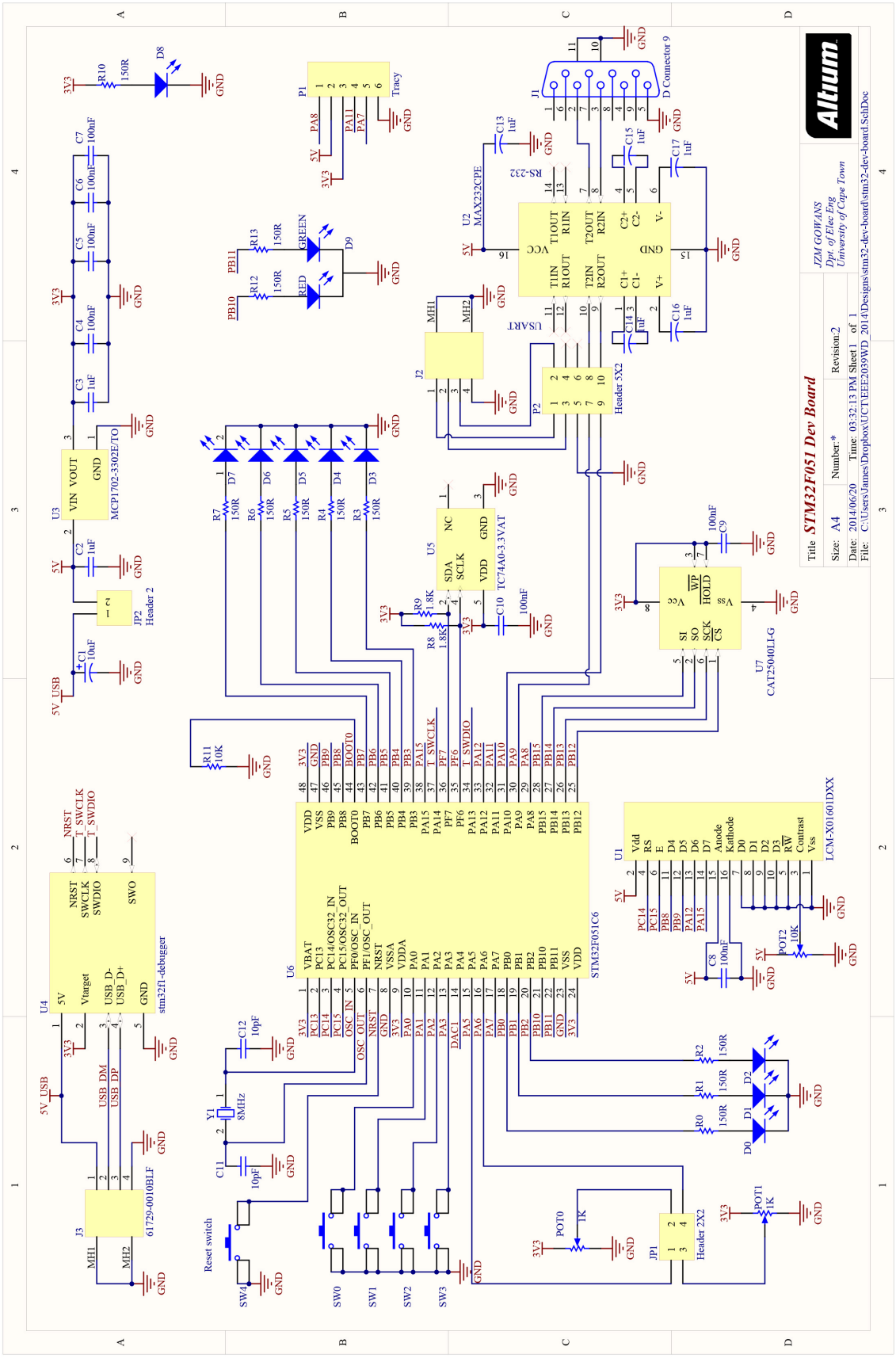


**Figure 3B.3: Circuit diagram for the UCT STM32 Development Board.** (Taken from [6])