**CG2271 Real Time Operating Systems**

**Lab 1 – ARM Cortex M0+ GPIO Programming**

1. Introduction

In this lab we will use what we have learnt about the Arm Cortex M0+ GPIO subsystem to drive the LED that is on the Freedom MCXC444 board.

1. Submission Instructions

Do this lab during the slot allocated to you, and submit the report by **Friday 5 September 2025, 11.59 pm** to your respective bin on Canvas.

You can do the demo at the end of the lab session or the start of the following lab session.

1. Understanding the FRDM-MCXC444 Board

Looking at the Schematics of the board in the SPF-93440\_A.pdf file, we see how the RGB LED is connected to the MCX444:

A computer screen shot of a computer

AI-generated content may be incorrect.

We see the following:

|  |  |
| --- | --- |
| **Color** | **Pin** |
| Red | PTE31 |
| Green | PTD5 |
| Blue | PTE29 |

Thus, we must enable ports D and E in the System Integration Module (SIM). Additionally we look at how the LEDs are connected:

A diagram of a led

AI-generated content may be incorrect.

Question 1 (1 mark)

Are the LEDs active high or active low (i.e. should the GPIO pin be driven high or low to turn on the LED)? Explain your answer. (2 marks)

The FRDM-MCXC444 board has two switches, the WAKEUP button at the top and the NMI button at the bottom. We will use this switch:

A close-up of a circuit board

AI-generated content may be incorrect.

The circuit diagram shows that this switch (SW3) is connected to PTA4:

A diagram of a circuit

AI-generated content may be incorrect.

Question 2 (1 mark)

Is SW3 active high or active low? Explain your answer. (2 marks)

1. Explore Programming the GPIO Ports

A project called “Blinky” has been provided to you in the Blinky directory which was included in CG2271Lab1.zip. Let us explore this project a bit:

1. Select “Import project(s) from file system” from the Quickstart Panel at the bottom left of the MCUXpresso screen:

A screenshot of a computer

AI-generated content may be incorrect.

Click Browse next to “Project directory (unpacked)”:

A screenshot of a software

AI-generated content may be incorrect.

Navigate to the directory where you unzipped the Blinky project. Click Open, then Finish.

If you get a box like:

A white rectangular box with black text

AI-generated content may be incorrect.

Click “Yes To All”.

The Blinky project will now be in the Project Explorer. Click Source then Blinky.c to open the source file.

1. Examine the following code:

A white background with black text

AI-generated content may be incorrect.  
Explain what each of the following lines do (1 mark each – total 4 marks)

Question 3 (4 marks)

|  |  |
| --- | --- |
| Line | Brief Explanation |
| SIM->SCGC5 |=  SIM\_SCGC5\_PORTE\_MASK; |  |
| PORTE->PCR[RED\_PIN]&= ~PORT\_PCR\_MUX\_MASK; |  |
| PORTE->PCR[RED\_PIN]= PORT\_PCR\_MUX(1); |  |
| GPIOE->PDDR |=  (1 << RED\_PIN); |  |

1. Create a New Project

Create a new C/C++ project and call it CG2271Lab2. See Lab 1 if you are not sure how to create a new project in MCUXpresso.

1. Programming the GPIO Ports

In this part of the lab we will program the FRDM-MCXC444 board so that We will now program the GPIO ports for the LEDs and the SW3 switch.

1. Delete the contents of the CG2271Lab2.c source file, and copy over the contents from the lab2.c source file provided in this ZIP file.
2. Fill in the body for initGPIO to initialize the GPIO for the 3 LEDs and SW3. (4 marks):

Question 4 (4 marks)

**void** **initGPIO**() {

}

1. Fill in the code in ledOn to turn on the respective LEDs. Cut and paste your code into the space below:

Question 5 (3 marks)

**void** **ledOn**(TLED led) {

**switch**(led) {

**case** *RED*:

// Code to turn on RED LED

**break**;

**case** *GREEN*:

// Code to turn on GREEN LED

**break**;

**case** *BLUE*:

// Code to turn on BLUE LED

**break**;

}

}

1. Fill in the code in ledOff to switch off the LED indicated. Cut and paste your code here (3 marks)

Question 6 (3 marks)

**void** **ledOff**(TLED led) {

**switch**(led) {

**case** *RED*:

// Turn off RED led here

**break**;

**case** *GREEN*:

// Turn off GREEN led here

**break**;

**case** *BLUE*:

// Turn off BLUE led here

**break**;

}

}

1. We will now amend main to read the switch. Locate these lines:

**int** count = 0;

**while**(1) {

// Switch is active low

**if**(// Code to read switch) {

**switch**(count) {

Change “// Code to read switch” to read SW3. This line should evaluate to TRUE if SW3 is pressed. Cut and paste your code below (1 mark)

Question 7 (1 mark)

**while**(1) {

// Switch is active low

**if**( ) {

**switch**(count) {

**case** 0:

ledOn(*RED*);

**break**;

**case** 1:

ledOn(*GREEN*);

**break**;

…

1. Demo (3 marks)

Demo your code to your TA either by the end of the Lab 2 session, or at the start of Lab 3.