# 829H1 Real-Time Embedded Systems Exercise 2

Candidate No: 105936

12th January 2019

# Contents

## 1. Introduction

This report outlines the work completed during the laboratory sessions, what equipment was used and what was learnt. Code listings for some of the created programs can be found in section 4.

## 2. Equipment

### 2.1. Hardware

The experiments carried out in this report were run on the Freedom-K64F prototyping board[1]. To use this board, we needed to connect the board to the computer using the debug port and copy across the program we wish to run and then press the reset button to load the program.

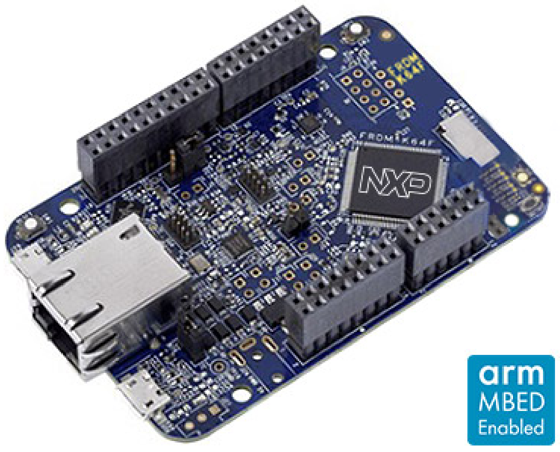

Figure 1: An Image of the Freedom-K64F Board used during experimentation[1]

### 2.2. Software

To develop the experiments, I used Arm's mbed cloud IDE. This allows users to write c++ code and compile it to a device of their choice.

## 3. Experiments

### 3.1. Interrupts

Interrupts seem to work in a similar way to event handlers in other programming languages in that they allow you to run a function out of sequence which can be called from another object. In the example, the red led is being turned on and off when the button SW2 is being released this is because the function ISR1() is being called when the button is released.

| Character Length | String | Time taken to print to screen | Difference to one fewer character length |
|---|---|---|---|
| 0 | | 0.000019 | |
| 1 | a | 0.002075 | 0.002056 |
| 2 | aa | 0.003114 | 0.001039 |
| 3 | aaa | 0.004154 | 0.00104 |
| 4 | aaaa | 0.005194 | 0.00104 |
| 5 | aaaaa | 0.006235 | 0.001041 |
| 6 | aaaaaa | 0.007275 | 0.00104 |
| 7 | aaaaaaa | 0.008314 | 0.001039 |
| 8 | aaaaaaaa | 0.009355 | 0.001041 |
| 9 | aaaaaaaaa | 0.010395 | 0.00104 |
| 10 | aaaaaaaaaa | 0.011435 | 0.00104 |

Table 1: Showing the time taken to print a string to the screen compared to the character length.

### 3.1.1. Interrupt triggered by falling edge

This means that the code should be changed so that the function is called when the button is pressed not when its released therefore all we need to change is the `button.rise(&ISR1);` to `button.fall(&ISR1);` The code for this can be found in section A.1

### 3.1.2. Two Interrupts on a single switch

The interrupts I created worked by switching the green led when it was pressed down and the red led when released. This meant I had to create another function `ISR2()` which would switch the green led. The code for this can be found in section A.2

### 3.1.3. Showing the function runs independently of the while loop

The led is being turned on and off independently of the while loop this can be shown by increasing the wait time in the loop so that the led can be turned on and off without the led flashing. The code for this can be found in section A.3

## 3.2. Timers

This experiment makes use of the timers. Allowing programmers to time things in their programs.

### 3.2.1. Differing times for differing string lengths

For this experiment, I created a program which would create strings of increasing length while timing how long it would take to output them to the terminal. The code for this can be found in section B.1. table 1 shows that adding a character to the string after it is at least 1 character long takes around 0.00104 seconds.

### 3.2.2. Using multiple timers

Unfortunately, I did not have an oscilloscope to hand when running this experiment, therefore, looking at it by eye and as a slow-motion video. I can see that the red led(ledA) flashes every 0.2 seconds and the green led(ledB) flashes every second.

### 3.2.3. Using different repetition rates

For this, I rewrote the task which turned the led on and off to work on the led passed to it meaning I only needed one method for all the leds. Although I still needed a timer for each led, therefore, had to initialise them at the start of the main function. The times I set were 0.2,0.3,0.4 and, 0.5 this lead to a flashy program as the led is changing colour every 0.1 seconds. The code for this can be found in section B.2

## 4. Conclusion

In conclusion, this was a useful quick introduction into how to develop using interrupts and timers also learning how to use tera term to view the console output of the board.

# Appendices

## A. Lab Exercise 1

### A.1. Part 1

```
/* External input causes interrupt, while led flashes*/
#include "mbed.h"
InterruptIn button(SW2); //define and name the interrupt input
DigitalOut led(LED1); //red
DigitalOut flash(LED2); //green

void ISR1() { //this is the response to interrupt, i.e. the ISR
   led = !led;
}
int main() {
    button.fall(&ISR1); // attach the address of the ISR function to the
   interrupt falling edge
    while(1) { // continuous loop, ready to be interrupted
      flash = !flash;
      wait(10);
    }
}
```

### A.2. Part 2

```
/* External input causes interrupt, while led flashes*/
#include "mbed.h"
InterruptIn button(SW2); //define and name the interrupt input
DigitalOut led(LED1); //red
DigitalOut flash(LED2); //green

void ISR1()   //this is the response to interrupt, i.e. the ISR
{
    led = !led;
}

void ISR2()
{
    flash = !flash;
}

int main()
{
    button.rise(&ISR1); // attach the address of the ISR function to the
   interrupt falling edge
    button.fall(&ISR2);
    while(1) { // continuous loop, ready to be interrupted
        wait(10);
    }
}
```

### A.3. Part 3

```
/* External input causes interrupt, while led flashes*/
#include "mbed.h"
```

```
InterruptIn button(SW2); //define and name the interrupt input
DigitalOut led(LED1); //red
DigitalOut flash(LED2); //green

void ISR1() { //this is the response to interrupt, i.e. the ISR
   led = !led;
}
int main() {
    button.rise(&ISR1); // attach the address of the ISR function to the
// interrupt rising edge
    while(1) { // continuous loop, ready to be interrupted
      flash = !flash;
      wait(10);
    }
}
```

## B. Lab Exercise 2

### B.1. Part 1

```
/* Program: A simple Timer Activate Tera Term terminal to test.*/
#include "mbed.h"
#include <string>

Timer t; // define Timer with name "t"
Serial pc(USBTX, USBRX);
int main()
{
    for(int i = 0; i <= 10; i++) { // Test 1 to 10 characters
        string str = "";
        for(int j = 0; j < i; j++) { // Add i characters to string str
            str += "a";
        }
        t.reset();
        t.start(); //start the timer
        pc.printf("%s\n", str.c_str());
        t.stop(); //stop the timer
        pc.printf("The time taken for %i characters was %f seconds\n", i, t
   .read()); //print to pc
    }
}
```

### B.2. Part 2

```
#include "mbed.h"
Timer timer_red; // define Timer with name "timer_red"
Timer timer_green;
Timer timer_blue;
Timer timer_red2;
DigitalOut red(LED1);
DigitalOut green(LED2);
DigitalOut blue(LED3);
DigitalOut red2(LED4);

void task_switch(DigitalOut led); //function prototypes

int main()
```

```
{
    timer_red.start(); //start the Timers
    timer_green.start();
    timer_blue.start();
    timer_red2.start();
    while (1) {
        if (timer_red.read()>0.2) { //test Timer value
            task_switch(red); //call the task if trigger time is reached
            timer_red.reset(); //and reset the Timer
        }
        if (timer_green.read()>0.3) {
            task_switch(green);
            timer_green.reset();
        }
        if (timer_blue.read()>0.4) {
            task_switch(blue);
            timer_blue.reset();
        }
        if (timer_red2.read()>0.5) {
            task_switch(red2);
            timer_red2.reset();
        }

    }
}

void task_switch(DigitalOut led)  //Switches the input led
{
    led = !led;
}
```

## References

[1] N. Products. (Mar. 2014). Frdm-k64f: Freedom development platform for kinetis® k64, k63, and k24 mcus, [Online]. Available: https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/kinetis-cortex-m-mcus/k-seriesperformancem4/k2x-usb/freedom-development-platform-for-kinetis-k64-k63-and-k24-mcus:FRDM-K64F.