829H1 Real-Time Embedded Systems Exercise 1

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. Also in the last experiment we used an Oscilloscope to measure the

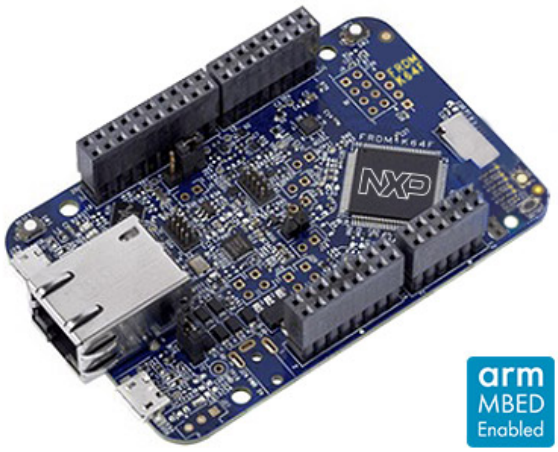

Figure 1: An Image of the Freedom-K64F Board used during experimentation[1]

voltage signals from the board to ensure that the program was working properly.

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. Digital Output

This section will look into programming the digital outputs of the Freedom-K64F board. More specifically I set the LED on the board to turn on and off every second.

3.1.1. Adding Comments

After adding some comments to the program to describe the code better there was no observable change in the program behaviour.

3.1.2. Varying the Wait Parameter

Changing the wait parameter appeared to change the time between the LED turning on and off on the board.

3.1.3. Varying the Wait Function

In addition to initially using the wait() function I also used the wait_ms() which allowed me to specify the wait time in milliseconds. The code for this part can be found in section A.1

3.1.4. Using Different LEDs

I also attempted to use different LEDs rather than just LED2 I discovered that LED1 is red, LED2 is green and LED3 is blue. LED4 is also red for some reason although I was expecting it to be white.

3.1.5. Using Multiple LEDs Together

This worked surprisingly nicely as the LEDs were close enough that setting colours to be on at the same time meant that they would combine for example I was able to combine red and green to make yellow. I was also able to combine all of them to make white. A couple of programs demonstrating this can be found in section A.2 and section A.3.

3.1.6. Why set the value of the LEDs to 1 and 0

This is because the LEDs appear to be connected in a open drain mode which means that if I provide a current to the led it turns off.

3.2. While Loop

The while loop allows for the code inside the loop to be continuously run while the statement for the loop is true.

3.2.1. Using ++ and − operators

This is a shorthand way of writing $i = i + 1$ and $i = i - 1$ this makes sense as incrementing and decrementing are very easy for computers to do which means that having a shorthand way of writing it means it can be translated in to the computer code more efficiently. It is also possible to use += and −= which allow you to add and subtract values from variables.

3.2.2. Replacing myled=1 with myled.write(1)

This has the same function I believe what is happening is that the DigitalOut class overloads the assignment operator so that .write() is called on the value is assigned to the object. The program for this experiment can be found in section B.1.

3.3. Digital Input

This consisted of using everything I had learnt so far and also using the value of the SW2 switch to change the function of the lights. This was quite simple as all you have to do is call the read function on the input you wish and the value is provided. A program

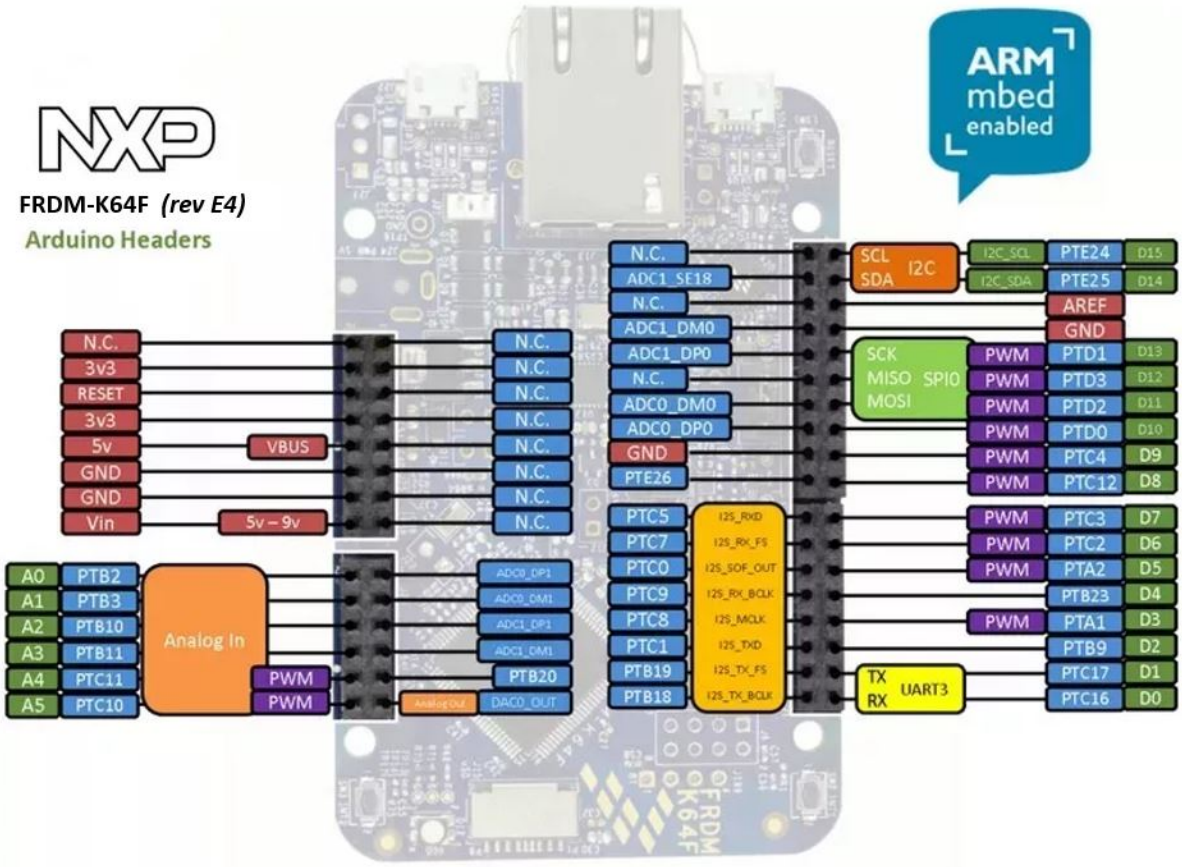

Figure 2: An Image of the Freedom-K64F Board with the codes for the header pins overlaid[2]

which changed the colour of the light if you press the switch can be found at section C.1. The slightly difficult task was creating the oscilloscope program as I had to find out the time period needed for a 100hz and 200hz frequency and then all I had to do was set the value of the pin. I used the PTC16 port on the board to output the current to, this was identified using fig. 2. This program can be found at section C.2

4. Conclusion

In conclusion a lot was learnt and the ability to interface with the different components on the board has grown. Also knowledge about how to develop on embedded systems has been attained and should provide a useful starting ground for future development.

Appendices

A. Lab Exercise 1

A.1. Part 1

```
#include "mbed.h" //include the mbed header file as part of this program
// program variable myled is created, and linked with mbed LED2

DigitalOut myled(LED4); // Creates a DigitalOut obejct for LED4 and allows
    for control over the LED4 Device

int main() { //the function starts here
     while (1) { //a continuous loop is created
        myled = 1; //switch the led off, by setting the output to logic 1
        wait_ms(250); //wait 1/4 seconds
        myled = 0; //switch the led on
        wait_ms(250); //wait 1/4 seconds
     } //end of while loop
} //end of main function
```

A.2. Part 2

```
#include "mbed.h" //include the mbed header file as part of this program
// program variable myled is created, and linked with mbed LED2

DigitalOut myledr(LED1); // Creates a DigitalOut obejct for LED4 and allows
     for control over the LED4 Device
DigitalOut myledg(LED2);
DigitalOut myledb(LED3);

int main() { //the function starts here
     while (1) { //a continuous loop is created
        myledr = 1; //switch the led off, by setting the output to logic 1
        myledg = 1;
        myledb = 1;
        wait_ms(250); //wait 1/4 seconds
        myledr = 0; //switch the led on
        myledg = 0;
        myledb = 0;
        wait_ms(250); //wait 1/4 seconds
     } //end of while loop
} //end of main function
```

A.3. Part 3

```
#include "mbed.h" //include the mbed header file as part of this program
// program variable myled is created, and linked with mbed LED2

DigitalOut myledr(LED1); // Creates a DigitalOut obejct for LED4 and allows
     for control over the LED4 Device
DigitalOut myledg(LED2);
DigitalOut myledb(LED3);

int main() { //the function starts here
     while (1) { //a continuous loop is created
```

```
        //myledr = 1; //switch the led off, by setting the output to logic 1
        //myledg = 1;
        //myledb = 1;
        wait_ms(250); //wait 1/4 seconds
        myledr = 0; //switch the led on
        myledg = 0;
        myledb = 1;
        wait_ms(250); //wait 1/4 seconds
    } //end of while loop
} //end of main function
```

B. Lab Exercise 2

B.1. Part 1

```
#include "mbed.h"

DigitalOut myled(LED1);
DigitalOut yourled(LED2);

int main() {
  char i=0; //declare variable i, and set to 0
  while(1){ //start endless loop
  while(i<5) { //start first conditional while loop
     myled.write(1);
     wait(0.2);
     myled.write(0);
     wait(0.2);
     i++; //increment i
  } //end of first conditional while loop
  while(i>0) { //start second conditional loop
     yourled.write(1);
     wait(0.2);
     yourled.write(0);
     wait(0.2);
     i--;
  }
  break;
     } //end infinite loop block
} //end of main
```

C. Lab Exercise 3

C.1. Part 1

```
#include "mbed.h"
DigitalOut redLED(LED1);
DigitalOut greenLED(LED2);
DigitalIn switchinput(SW2);


int main() {
   int onesec = 1000;
    while(1) {
        if (switchinput==1) { //test value of switchinput
```

```
//execute following block if switchinput is 1
          redLED = 1; // flash red led
          wait_ms(onesec);
          redLED = 0;
          wait_ms(onesec);
     } //end of if
   else { //here if switchinput is 0
      greenLED = 1; // flash green led
      wait_ms(onesec);
      greenLED = 0;
      wait_ms(onesec);
     } //end of else
    } //end of while(1)
   } //end of main
```

C.2. Part 2

```
#include "mbed.h"
DigitalOut d0PowerOut(PTC16);
DigitalIn switchinput(SW2);


int main() {
    int OneHundredHzWait = 10;
    int TwoHundredHzWait = 5;
    while(1) {
        if (switchinput==1) { //test value of switchinput
//execute following block if switchinput is 1
          d0PowerOut = 1; // flash red led
          wait_ms(TwoHundredHzWait);
          d0PowerOut = 0;
          wait_ms(TwoHundredHzWait);
     } //end of if
   else { //here if switchinput is 0
      d0PowerOut = 1; // flash green led
      wait_ms(OneHundredHzWait);
      d0PowerOut = 0;
      wait_ms(OneHundredHzWait);
     } //end of else
    } //end of while(1)
   } //end of main
```

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.

[2] A. Limited. (Dec. 2015). Frdm-k64f, [Online]. Available: os.mbed.com/platforms/FRDM-K64F/.