
Experiment 1A: Lighting the Embedded World (2.5%)

Time: 3 hrs

Expectations

- Come to lab ON TIME with all equipment and documents required.
- Ensure you wear enclosed shoes. No student will be admitted to the laboratory without appropriate footwear.
- No food or drink should be brought into the laboratory.
- Be respectful to other students and behave in a courteous and appropriate manner.
- Attendance is compulsory. If you are away sick, you will need a medical certificate and arrange to attend another lab session.
- Keep backups of all programs. Well written subroutines can be reused in future labs saving you time.
- Plan your lab time – you may not finish if you turn up late and unprepared. Before the lab:
 - Read the lab experiments;
 - Think about questions to be answered;
 - Carefully design all programs before programming them!!!!
- **Get all sections marked off as they are completed.**
- You will need to answer questions at the end to demonstrate understanding of what you have done.

Aim

To develop experience in embedded C programming and debugging using the Keil uVision software, including the use of more complex program structures.

Note: In this part, the Tiva will be connected to your PC via a USB cable. Some precautions to avoid damaging the device are as follows.

When the Tiva has power on, you can:

- Press reset button
- Touch / release buttons on the board or connected to the board
- Connect / disconnect a voltmeter to the system
- Plug and unplug the USB cable

When the Tiva has power on, you can NOT:

- Plug / unplug the board into / from a bread-board or peripherals board
- Connect / disconnect cables or wires to external devices
- Insert / remove wires, resistors, or chips from the circuit connected to the board.

Introduction

This experiment aims to familiarize students with use of Keil's uVision5 IDE for implementation on the Tiva Launchpad microcontroller. You have already used Keil's debug mode on a set of smaller tasks, providing some familiarity with the software, as well as the C language and programming skills. This experiment aims to further develop these skills, by developing a program that will use non-destructive bit manipulation to use the switches to count in binary on the LEDs from 0 to 3 (2 bit counter)

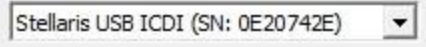
Preliminary

There is no preliminary this week.

Experiment

The template provided on the course website is setup for the Tiva microcontroller and Keil uVision 5. Start by the doing the following:

1. Download lab1A template.
2. Check the configuration of target options:
Ensure Debug and Utilities use Stellaris ICDI,



During this lab, you will be provided with some precompiled functions. The list of the functions is given in the Appendix.

Unzip the Lab1A template and run it. It should turn flash LED D2. Make sure that you understand what the program is doing.

The LEDs can be bright. I suggest that you cover the LED with a piece of paper to diffuse the light.

Part 1: Delay function

3 marks

Open the supplied template. Copy the template to a new folder and call it Part 1. The template will contain code that flashes D3 (PF4) using the supplied functions listed in the Appendix. The supplied template also toggles PF1 so that the output can be seen on an oscilloscope and the duration measured. The Oscilloscope connections are indicated by red arrows in Appendix B

Time delays are useful in many programs, including here. This first part requires you to create a subroutine which can be used to generate a delay of required time.

- Write a delay function to replace msDelay() called milliDelay() which create a 1ms delay by using a **for** loop that just counts to a large number doing nothing whilst it counts. It will require a big number to waste 1 millisecond of time since the Tiva Clock runs at 16MHz. You can fine tune the number of counts by watching PF1 and determining the duration on the Oscilloscope.
- Once you know how many counts are used for 1 millisecond then modify the miilDelay() function so that it waits for a given number of milliseconds. The number of milliseconds to delay must be passed to the function via its argument list.

To test the milliDelay() function, modify the main() function so that the LED flashes at 1HZ with a 50% duty cycle. You can show the operation by turning the LED on, calling the delay, and then turning off the LED. If you turn PF1 on and off at the same time as the LED then you can measure the delay on the oscilloscope.

Show your demonstrator that your delay is working correctly so you can be assessed for this question.

Part 2: This is quite flash

3 marks

Copy the Part1 folder and project and call it Part2.

Modify the program written in Part 1 so that it uses pointers and the C bitwise Operators to turn on, turn off and toggle LED D1 which can be in on Port N. (See the diagram in the Appendix) .

Configure the Tiva board so that Port N is turned on and set to the appropriate mode to operate the LEDs using the PortN functions supplied. See the appendix for the supplied functions.

Using a pointer that is set to the address of the port N Data register (**0x400643FC**), write a 0x1 to this address and see what it does. Change your code so that it clears the LED to check your understanding.

Hint: it will help if you use a #define as shown in the Embedded C lectures for the pointer.

Modify your program so that it does the following using the address of the Port N rather than using the writePortNData() function.

- (i) Repeat 5 times
 - a. Turn on the led on using bitwise operators. (Set the appropriate bit)
 - b. Delay 1 second
 - c. Turn off the LED by using bitwise operators. (Clear the appropriate bit)
 - d. Delay 1 second

Note: The bitwise operators are: | & ^ ~ (OR, AND, XOR, NOT)

At the successful completion of the program, show you demonstrator the code and the debugger results so that you can be assessed.

Part 3: Point to the light

2 marks

Copy the Part2 folder and project and call it Part3. Write a program so that it uses pointers and the C bitwise operators to:

- (i) Turn D1 on and D2 off for 1 second
- (ii) Turn D2 on and D1 off for 1 second
- (iii) Turn D1 and D2 on for 2 seconds
- (iv) Turn D1 and D2 off for 2 seconds
- (v) Repeat from step (i)

PORT N Data register address is **0x400643FC**.

Refer to the Appendix to determine the appropriate pins for the LEDs.

Part 4: It all counts you know

4 marks

In a new project, write a program to count up in binary from 00 to 11 on the LEDs on Port N. There should be a 1 second delay between each change of count.

You must use:

- pointers as per the previous question. (#define)
- bitwise operators to set and clear bits a counter
- variable to be represented on the LEDs.

HINT: try clearing the data register before writing value of the counter.

Part 5: Using the switches

3 marks

Modify the above code to read from SW1 and move to the next count rather in place of the delay loop. The switches can be found on Port J. You will need to write code to turn on port J and to set it to the appropriate mode using the supplied functions. You should access the data on Port J using pointers (#define) and bitwise operators.

Note: When pressed the switch gives a 0 and when not pressed it gives a 1. The address of the data register

HINT: you will need to set the required PORTJ pins to appropriate direction (ie OUTPUT or INPUT). You will need to the read and/or write the PORTJ data register directly. The Port J Data Register is **0x400603FC**.

Experiment Submission

Before leaving the lab you need to submit all code you have written for each part of the experiment as a single zip file containing all the folders for the different parts. This is done by uploading the zip file containing your code to the course website This submission provides some backup of your work and is required for a mark to be given for each lab part. Note, however, that the mark you receive for each part will be based on what has been demonstrated during your lab session.

Ensure you have shown your demonstrator the correct operation of each completed part of this lab and have completed the online submission before leaving the laboratory. You will not be able to complete this later.

Appendix A: Functions provided in lib file

PORT F Functions

void enablePF(void)

Input: Nil

This function turns on the PORTF GPIO subsystem. This is required before PORTF can be accessed/used.

void setPFOutput(uint8_t pins)

Input: pins

This function sets the specified pins of PORTF to Output Mode so that the pins can output voltage when the writePFData() function is used later. The pins will be a number between 0x00 and 0xFF which corresponds to an 8 bit value. The Pin number corresponds to the Bit number in the pins variable.

Port Pin	P7	P6	P5	P4	P3	P2	P1	P0
Bit in the pins argument	B7	B6	B5	B4	B3	B2	B1	B0

void setPFInput(uint8_t pins)

Input: pins

This function sets the specified pins of PORTF to Input Mode so that the an input voltage can be read from the pin when the readPFData() function is used later. The pins will be a number between 0x00 and 0xFF which corresponds to an 8 bit value. The Pin number corresponds to the Bit number in the pins variable.

Port Pin	P7	P6	P5	P4	P3	P2	P1	P0
Bit in the pins argument	B7	B6	B5	B4	B3	B2	B1	B0

void writePFData(uint8_t pins)

Input: value

This function writes the value to the port. Only those pins that have been set to output will output a voltage. The value will be a number between 0x00 and 0xFF which corresponds to an 8 bit value. The Pin number corresponds to the Bit number in the value variable. A bit set to 0 corresponds to a LOW output and a bit set to 1 corresponds to a HIGH output.

Port Pin	P7	P6	P5	P4	P3	P2	P1	P0
Bit in the value argument	B7	B6	B5	B4	B3	B2	B1	B0

uint8_t readPFData(void)

Output: The value read from the Pins.

This function reads the input state (0 or 1) of the pins on PORTF. A bit set to 0 corresponds to a LOW input and a bit set to 1 corresponds to a HIGH input. The value returned will be a number between 0x00 and 0xFF which corresponds to an 8 bit value. The Pin number corresponds to the Bit number in the returned value.

Port Pin	P7	P6	P5	P4	P3	P2	P1	P0
Bit in the return value	B7	B6	B5	B4	B3	B2	B1	B0

PORT N Functions

void enablePN(void)

Input: Nil

This function turns on the PORTN GPIO subsystem. This is required before PORTN can be accessed/used.

void setPNOOutput(uint8_t pins)

Input: pins

This function sets the specified pins of PORTN to Output Mode so that the pins can output voltage when the writePNData() function is used later. The pins will be a number between 0x00 and 0xFF which corresponds to an 8 bit value. The Pin number corresponds to the Bit number in the pins variable.

Port Pin	P7	P6	P5	P4	P3	P2	P1	P0
Bit in the pins argument	B7	B6	B5	B4	B3	B2	B1	B0

void setPNIInput(uint8_t pins)

Input: pins

This function sets the specified pins of PORTN to Input Mode so that an input voltage can be read from the pin when the readPNData() function is used later. The pins will be a number between 0x00 and 0xFF which corresponds to an 8 bit value. The Pin number corresponds to the Bit number in the pins variable.

Port Pin	P7	P6	P5	P4	P3	P2	P1	P0
Bit in the pins argument	B7	B6	B5	B4	B3	B2	B1	B0

void writePNData(uint8_t pins)

Input: value

This function writes the value to the port. Only those pins that have been set to output will output a voltage. The value will be a number between 0x00 and 0xFF which corresponds to an 8 bit value. The Pin number corresponds to the Bit number in the value variable. A bit set to 0 corresponds to a LOW output and a bit set to 1 corresponds to a HIGH output.

Port Pin	P7	P6	P5	P4	P3	P2	P1	P0
Bit in the value argument	B7	B6	B5	B4	B3	B2	B1	B0

PORT J Functions

void enablePJ(void)

Input: Nil

This function turns on the PORTJ GPIO subsystem. This is required before PORTJ can be accessed/used.

void setPJOutput(uint8_t pins)

Input: pins

This function sets the specified pins of PORTJ to Output Mode so that the pins can output voltage when the writePJData() function is used later. The pins will be a number between 0x00 and 0xFF which corresponds to an 8 bit value. The Pin number corresponds to the Bit number in the pins variable.

Port Pin	P7	P6	P5	P4	P3	P2	P1	P0
Bit in the pins argument	B7	B6	B5	B4	B3	B2	B1	B0

void setPJInput(uint8_t pins)

Input: pins

This function sets the specified pins of PORTJ to Input Mode so that the an input voltage can be read from the pin when the readPJData() function is used later. The pins will be a number between 0x00 and 0xFF which corresponds to an 8 bit value. The Pin number corresponds to the Bit number in the pins variable.

Port Pin	P7	P6	P5	P4	P3	P2	P1	P0
Bit in the pins argument	B7	B6	B5	B4	B3	B2	B1	B0

void writePJData(uint8_t pins)

Input: value

This function writes the value to the port. Only those pins that have been set to output will output a voltage. The value will be a number between 0x00 and 0xFF which corresponds to an 8 bit value. The Pin number corresponds to the Bit number in the value variable. A bit set to 0 corresponds to a LOW output and a bit set to 1 corresponds to a HIGH output.

Port Pin	P7	P6	P5	P4	P3	P2	P1	P0
Bit in the value argument	B7	B6	B5	B4	B3	B2	B1	B0

uint8_t readPJData(void)

Output: The value read from the Pins.

This function reads the input state (0 or 1) of the pins on PORTJ. A bit set to 0 corresponds to a LOW input and a bit set to 1 corresponds to a HIGH input. The value returned will be a number between 0x00 and 0xFF which corresponds to an 8 bit value. The Pin number corresponds to the Bit number in the returned value.

Port Pin	P7	P6	P5	P4	P3	P2	P1	P0
Bit in the return value	B7	B6	B5	B4	B3	B2	B1	B0

Uint8_t readPJData(void)

Output: The value read from the Pins.

This function reads the input state (0 or 1) of the pins on PORTF. A bit set to 0 corresponds to a LOW input and a bit set to 1 corresponds to a HIGH input. The value returned will be a number between 0x00 and 0xFF which corresponds to an 8 bit value. The Pin number corresponds to the Bit number in the returned value.

Port Pin	P7	P6	P5	P4	P3	P2	P1	P0
Bit in the return value	B7	B6	B5	B4	B3	B2	B1	B0

Delay Functions

void msDelay(uint32_t ms)

This function creates a delay of ms milliseconds. It will return after waiting for the given time.

Appendix B: Board Connections

