California State University, Northridge

Department of Electrical & Computer Engineering

Lab 5

Flashing LEDs

October 13, 2022

ECE 425L

Assigned by: Neda Khavari

Written By: Pathum Addarapathirana & Cristian Robles

Introduction:

Lab 5 introduces us to different addressing modes to better understand using pins as general purpose inputs and outputs on the LPC2148 Education board. This lab also furthers our understanding of the Keil uVision Simulator by introducing the logic analyzer.

Procedure:

Equipment Used

- Keil uVision4
- Keil Debugger
- LPC2148 Education Board
- Oscilloscope

Description of Procedure

1. Set up startup code. This code is needed to startup the lab and it is pretty straightforward. This includes the header below and the mystatup code given in lab 3.

```
1                   GLOBAL Reset_Handler
2                   AREA    mycode, CODE, Readonly
3   Reset_Handler   ;this label is necessary for the first line of your code
```

2. In the next step we simply set up our EQU values as shown below.

```
5   IO0_BASE    EQU 0xE0028000
6   IO0PIN      EQU 0
7   IO0DIR      EQU 0x8
8   IO0SET      EQU 0x4
9   IO0CLR      EQU 0xC
```

3. In the next step we have to set up GPIO. This means we have to set the pins as ‘General Purpose Input and Output’. This is done by storing 0 into PINSEL0 address.

```
5   PINSEL0     EQU     0xE002C000      ;equate symbolic name PINSEL0 as address 0xE002C000
6
7                                       ;Selecting funcion as GPIO by writing all zeros to given address
8               MOV     r0, #0          ;moves #0 into register r0
9               LDR     r1, =PINSEL0    ;puts PINSEL0 in r1 register; CANNOT PUT MOV, outputs error in keil
10              STR     r0,[r1]         ;copies value stored in r0 to memory address of r1
```

4. Then we have to set the signal direction of pins to determine whether they are input pins or output pins. This is done by storing a ‘1’ into each bit that we want to make an output pin of IO0DIR. We only needed to make the first 8 pins an output pin so we used the hex code 0x0000FF00.

```
13  IO0DIR      EQU     0xE0028008      ;IO0DIR assigned to 0xE0028008
14
15                                      ;Selecting signal direction of each port pin, we put '1' in each to bit to make it an output pin
16              MOV     r0, #0x0000FF00
17              LDR     r1, =IO0DIR     ;puts 0xE0028008 or IO0DIR to r1 register
18              STR     r0, [r1]        ;copies value stored in r0 (0x0000FF00) to memory address of r1
```

5. For our first task we are to flash all 8 LED's on and off. We are advised to output signals by writing to IO0SET and IO0CLR. To achieve this we have to first set IO0CLR shown in the screenshot below.

```
38                              ;Clears certain bits in value register IO0PIN, we clear the bit positions with a 1 value
39          ;MOV        r0, #0x0000FF00 ;moves 0x0000FF00 (binary 1111111100000000) into r0 register. (Pins 8-15)
40          LDR     r5, =IO0CLR     ;puts 0XE002800C or IO0CLR to r1 register
41          STR     r2,[r3,r5]      ;copies value stored in r2 to memory address of r3+r5(IO0CLR) (ALL LEDS ON)
```

6. Now we start with the loop of flashing the LED's. For this step we are also asked to practice pre-indexing address modes in our code. First we begin by writing a LOOPALL loop that will loop the entire next process forever to flash the LED lights. Then we must create some kind of delay for at least 1 second or more so it is perceivable to the human eye. The clock of the LPC2148 runs at 12e6 hz. This allows a delay of 434782 cycles per second. So we implement a simple loop as shown in the code below.

```
45                  ;LOOP begin
46                  LDR     r9,=4347820
47  LOOP            SUBS    r9,r9,#1
48                  BNE     LOOP
```

7. Then we simply have to store the desired leds we want to turn on and off, which is LEDs 8-15 which translates to the hex code 0x0000FF00. This value is stored in r2 when we set IO0DIR. So we can use this number to store into a pre indexed value using the IO0BASE and IO0SET to turn the LEDs OFF. This can all be achieved in one step as shown below.

```
53          STR     r2,[r3,r4]          ;copies value stored in r2 to memory address of r3,r4 (ALL LEDS OFF)
```

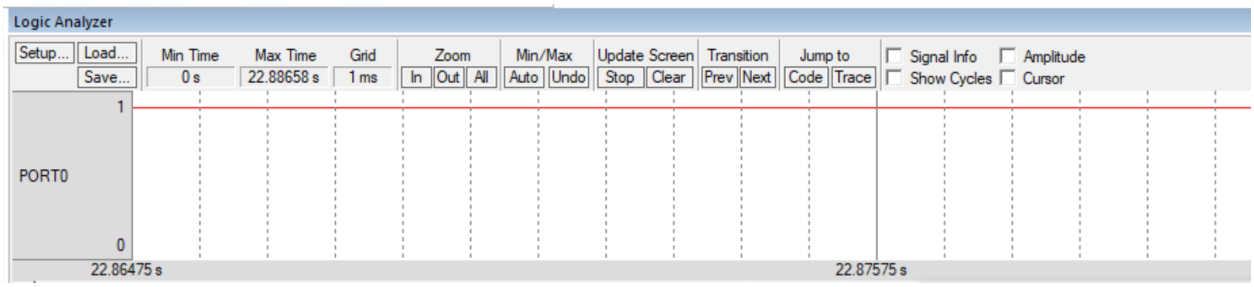
8. Then we introduce another delay for 1 second and then turn the LEDs ON by storing r2 (0x0000FF00) into the pre indexed value of IO0BASE+IO0CLR. This can also be achieved in one step as shown below. The last line is just a loop to go back to the first loop so this flashing continues forever.

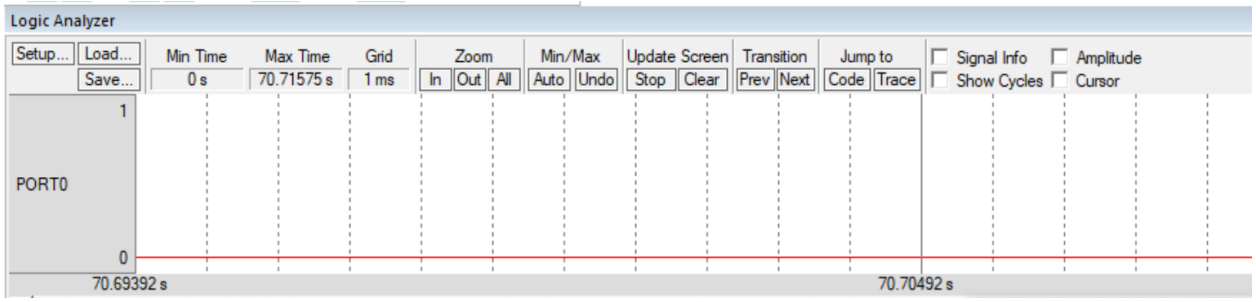
```
55              LDR     r8,=4347820
56  LOOP2       SUBS    r8,r8,#1
57              BNE     LOOP2
58                                  ;turn them ON again
59              STR     r2,[r3,r5]          ;copies value stored in r0 to memory address of r1 (ALL LEDS ON)
60
61              B       LOOPALL
```

9. For the next step of this lab we are to examine all the pseudo instructions for LDR in our first task. The PC offset value is determined when LDR is used. For example in the screenshot above 4347820 is being loaded into r8. But the actual instruction is LDR r8, [PC, #4347820]. This offset indicates how far away it is placed from the current PC value. At this current line it is placed at (*some PC*

value because it's hard to skip through the entire loop to find that exact location) so the number 4347820 is placed in the literal pool at that location.

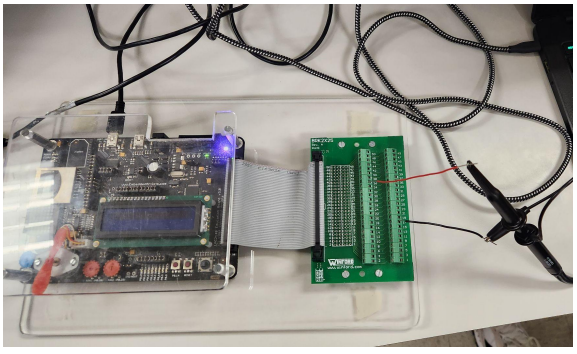
10. For task 3, using the logic analyzer we can see once we set port0 that it jumps from 1 to 0 every second as expected. This is because we set an ANDMASK on port0 to observe its behavior as expected. When we zoom out we can see it jumps from 0 to 1.

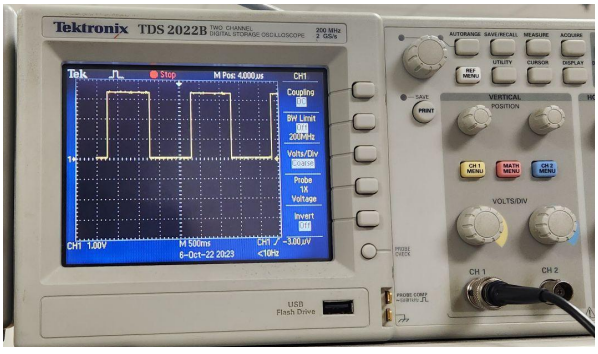




Results:

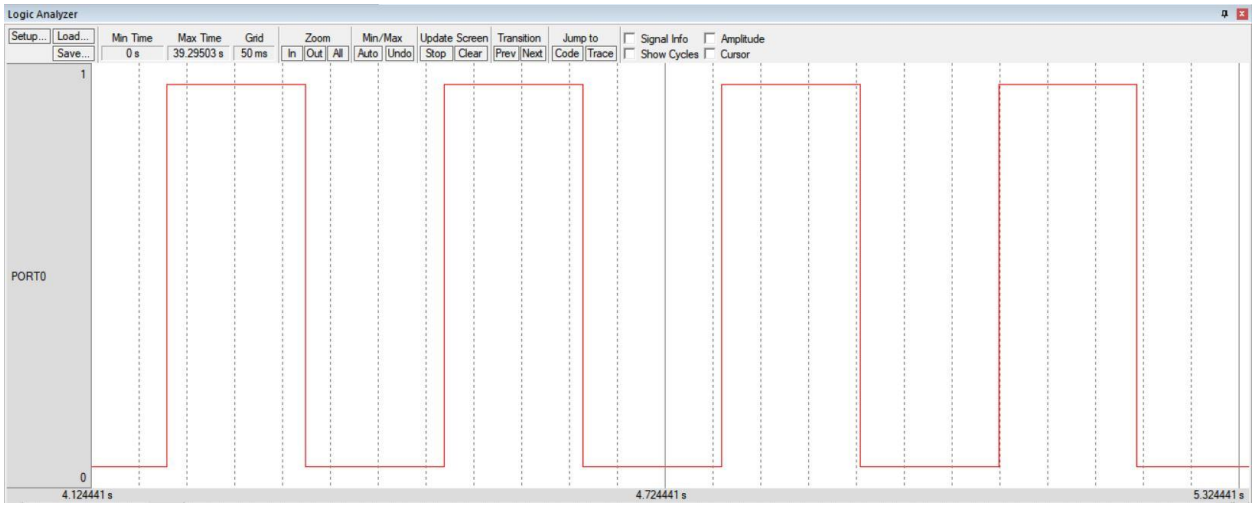
Flashing LEDs and Oscilloscope





For task 1 our program would successfully flash the LEDs on and off every second while the board was on. We used the oscilloscope to see the signal being applied to pin 8 of the LPC2148 and got the square wave seen above

Logic Analyzer



In the logic analyzer we can see the signal change from 0 to 1 multiple times per second. This is because the LPC2148 hardware is different from the board which is being simulated

Questions:

```
IO0CLR EQU 0xE002800C
LDR r0, =0xF0000000
LDR r1, =IO0CLR
STR r0, [r1]
```

Question 1: Is it ok to replace LDR by MOV in the first line of the above code segment? Is it ok to replace instruction LDR by MOV in the second line?

It is not ok to replace the LDR with a MOV in either line as the values 0xF0000000 and 0xE0028000C are greater than 255 in decimal which is the limit for the MOV instruction

```
IO0_BASE EQU 0xE0028000
IO0PIN EQU 0
IO0DIR EQU 0x8
IO0SET EQU 0x4
IO0CLR EQU 0xC
```

Question 2: Any advantage could be gained with the above definitions of control registers along with using pre-indexing address mode?

Placing the value IO0_BASE into a register and using the PIN, DIR, SET, and CLR as offsets to configure the control registers allows for less use of registers and makes the code more readable.

Conclusion:

In conclusion, we were able to successfully set pins as general purpose inputs and outputs using different addressing modes than in lab 4. The different addressing modes allow for more readable code and make our code more efficient. We also learned about how to loop using the branch instruction and were able to observe our signals using both the oscilloscope and the simulator inside the Keil uVision. A way that this lab could be improved is to change the code within the mystartup file so that the signal in the simulator changes once per second like the one we observed on the oscilloscope. This project furthered our knowledge of addressing, loops, and controlling pins on the board.