DT228-2 Microprocessor Systems 1

Lab 1.
Introduction to the MSP430 EasyWEB2 development kit.

The EasyWEB2 development kit contains:

- A TIMSP430F149 Microcontroller
- An Ethernet interface
- An LCD display
- Relays
- Buttons
- A buzzer
- Various connectors

This lab will focus on the buttons and the buzzer.

The software development process

Programs are written in C on the PC using the IAR Systems integrated development environment (IDE). To start the IDE go to the Start button, find "IAR Systems" and locate the "Embedded Workbench" program.
When the program is run you will be presented with a set of choices so pick the one that creates a new project.
You will now be asked what type of project you want to create: Click the "+" symbol near the C-program choice and choose "main". This creates a skeleton C program that you can edit.

Parallel I/O Ports

The designers of the TIMSP430 family of microcontrollers implemented the Input/Output ports as "memory mapped ports". The implication of this is that programs write to ports in the same way that they write to program variables in memory which greatly simplifies programming. All that is required of the programmer is that they know the memory addresses of the ports that they wish to interact with. Now, you could open up the manual for the processor and figure all of these out. Happily this is not necessary as Texas Instruments have already prepared a lists of the port addresses for each member of the MSP430 family and placed them in header files.

1 Reading input ports

The buttons on the EASYWEB2 board are connected to port 4 on the MSP430. A section of the electrical schematic of the board is shown in Figure 1 below.

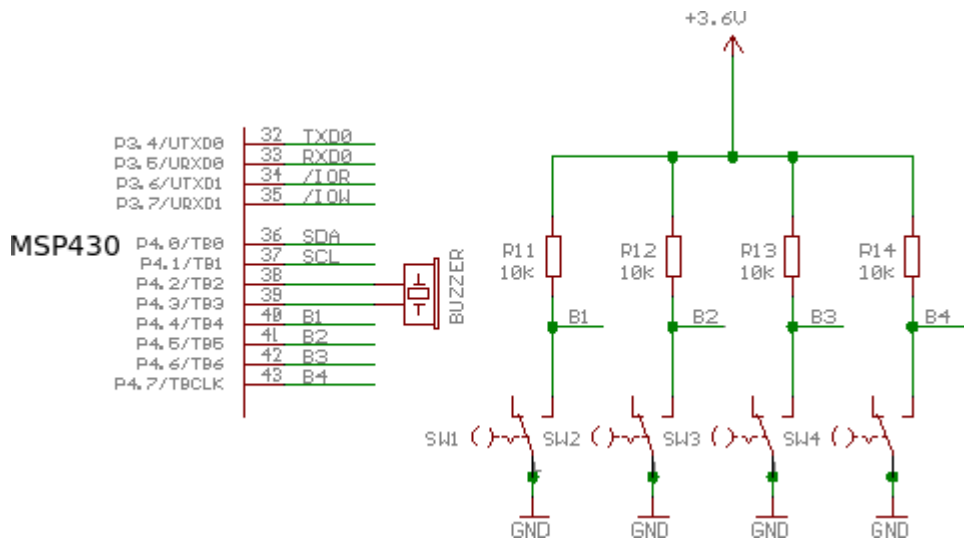

Figure 1

When no button is pressed, bits 4.4 to 4.7 are “pulled high”. When a user presses a button, the corresponding port pin is “pulled low”.
Input the following simple program and using single step debugging, write down how the contents of
port 4's input register (P4IN) change when the buttons are pressed.

```
#include <msp430x14x.h>
int portdata;
void main() {
     WDTCTL = WDTPW + WDTHOLD;
     while(1) {
          portdata = P4IN;
     }
}
```

2 Writing to output ports.
Figure 1 also shows a buzzed connected to port 4. This can be controlled as follows:

```
#include <msp430x14x.h>
void main() {
     WDTCTL = WDTPW + WDTHOLD;
     P4DIR=12;
     while(1) {
          P4OUT = 4;
          P4OUT = 8;
     }
}
```

Single step through this program and note what happens to the buzzer (listen to the buzzer).
Run the program. Do you hear anything? Why not?

3 Sounding the buzzer

The buzzer may be made whistle as follows:

```
#include <msp430x14x.h>

void delay() {
     int count;
     for (count = 0; count < 400; count++);
}
void main() {
     WDTCTL = WDTPW + WDTHOLD;
     P4DIR=12;
     while(1) {
          P4OUT = 4;
          delay();
          P4OUT = 8;
          delay();
     }
}
```

Why does the buzzer now sound when the program is run?

(4) Controlling the buzzer with the buttons.

Investigate a mechanism that will allow you to play different “notes” on the buzzer by pressing different keys (or combination of same).

The lab report:

Lab reports are to be emailed to

frank.duignan@dit.ie

before your next lab!

Include in the report :

- Name and student number (if you do not work alone then all names/numbers)
- Code and explanations of same.
- Explanation of what a 'port' is
- Explanation of what a data direction register is.
- Answers to all questions posed