Lab 01

Input-Output Programming through PC Parallel Port

Prepared By:
Nurul Afiqah binti Zainal
Khairul Fikri bin Muhamad
Ismayuzri Ishak

Fakulti Kejuruteraan Pembuatan, UMP

Lab Location
Sensor & Instrumentations Lab

Lab Outcomes:

By the end of this lab, students should be able to:

1. Write C program to write to parallel port
2. Write C program to read data from parallel port

1.0 Parallel Port Programming

Controlling the "real world" can be accomplished through what we called a "controller". Controller is a programmable electronic device that has input, output, microprocessor and memory. Through controller, we can read real-world signals using input port. The signals can then be processed by the microprocessor and stored in memory. Likewise, we can control the real world through the output port.

Any real-world device can be interfaced to the controller for control purposes. In general, there are three types of controller devices: Personal Computer (PC), microcontroller and programmable logic controller (PLC). In this lab, we are going to use PC as our controller.

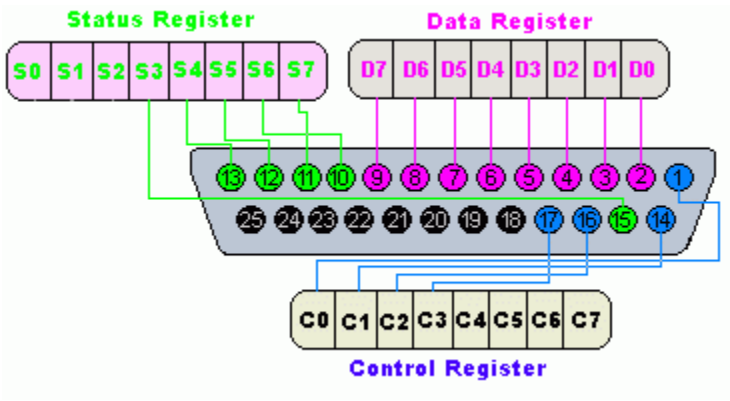
Specifically, we are going to build a simple LED[1] circuit (real-world device) that is interfaced to PC Parallel Port. The PC Parallel Port is either the printer port or a PCI card. The PC parallel port is the 25-pin connector at the back of a PC unit. We will treat the LEDs as our output devices meaning that we are going to use PC to control the LED lighting.

WARNING!!! ALTHOUGH THE KNOWLEDGE IN THIS LAB IS AMUSING, DO NOT TRY THIS AT HOME OR TO YOUR OWN PC. IT MAY DAMAGE YOUR PARALLEL PORT IF NOT CONDUCTED PROFESSIONALLY. I AM NOT RESPONSIBLE FOR ANY LIABILITY AS A RESULT OF USING THE INFORMATION IN THIS LAB. THIS EXPERIMENT IS DESIGNED ONLY FOR LABORATORY SETTING. MORE KNOWLEDGE IS REQUIRED IF YOU WANT TO EXPERIMENT ON YOUR OWN.

[1] LED stands for light emitting diode

2.0 Parallel Port Background

On the right is the schematic diagram of the PC parallel port connector. There are 25 pins all together. The 25-pin represents four main functions:



1. **First Output Port** – the pins from Pin2 to Pin 9. It is called data register. Later, we will control the voltage of each of these eight pins through C program.
2. **Input Port** – the pins are Pin10, Pin11, Pin12, Pin13 and Pin15. It is called status register. One of this pin will be connected to our switch as input device.
3. **Second Output Port** – Pin1, Pin14, Pin16 and Pin17. These pins are also output port and are called control register. This is an extra port in case controlling through our first port is not sufficient.
4. **Ground pins** which are Pin18 to Pin 25. These are PC ground.

Each output pin can have only two values: either 5 volts (greater than 2.8 volts to be more precise) or 0 volt. This is what we call digital value. The digital state of "1" or "ON" represents 5V while the digital value of "0" or "OFF" represents 0 volt.

Let's say that we have a circuit. Each LED is connected to each output pin at data port. Remember that the output pins begin from Pin2 to Pin9 and not from Pin1. To turn on the LED, we need to provide 5 volts to the pin. For instance, if we want to turn on the LED connected to Pin2, we will set the digital value of Pin2 to "1" so that 5 volts will come out from Pin2 and therefore turn on the LED. How to set the digital value of Pin2 to "1"? That is why we use C programming.

Finally, the 1 kOhm resistor is used to limit the current taking out from PC. In this case, by OHM's law, we know that the current is 5 mA. Since we have 8 LED, we have taken out from PC 40 mA of current. That is quite reasonable. For the input circuit from external power to the parallel port, the 47 Ohm resistor should be use in order to limit the input current. You all can calculate the input current that going through the parallel port at 5 voltage input (write in your technical report>circuit analysis).

3.0 Output and Input Programming Using C Language

Below is the table that summarizes Bit to Pin Mapping for standard parallel port:

data port	base	pin 9	pin 8	pin 7	pin 6	pin 5	pin 4	pin 3	pin 2
status port	base+1	pin ~11	pin 10	pin 12	pin 13	pin 15			
control port	base+2					pin ~17	pin 16	pin ~14	pin ~1
bit designation		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
bit value		2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
decimal equivalent value		128	64	32	16	8	4	2	1

Symbol '~' is for inverse function.

Example program that can be programed is:

If we want to turn all LED at data port, we have to set the output port to "11111111" digital value. Each "1" represent each pin. Since there are 8 output pins, we set all pins to "1". Referring to the table, this digital value is equivalent to 255 in decimal. Therefore, to turn all LEDs, using C programming we shall write:

```
(oup32)(0xECD8,255);
```

"oup32" is a C program user-defined function input or output.
"0xECD8" is the base address of the output port.
"255" is the value that we want the output port to have.

As another example, let's say that we want to turn on the third LED only at control port. Digitally, the command should be 00000100. This is equivalent to "4" in decimal value at pin 16. In our C programming, we shall write:

```
(oup32)(0xECD8+2,4);
```

This will cause Pin4 to have 5 volts and the rest of the pins 0 volt.

For the input programming, we want to get the data at data port, we have to set input function. The C programming shall write:

```
(inp32)(0xECD8);
```

This will set all the input data at data port to be read.

Here are example of program to send all data port to low and read the input data at data port:

```
#include <stdio.h>
#include <conio.h>
#include <windows.h>

typedef short _stdcall (*inpfuncPtr)(short portaddr);
typedef void _stdcall (*oupfuncPtr)(short portaddr, short datum);

// Address PCI card ECD8

int main(void)
{
   HINSTANCE hLib;
   inpfuncPtr inp32;
   oupfuncPtr oup32;
   /* Load the library */
   hLib = LoadLibrary("inpout32.dll");
   inp32 = (inpfuncPtr) GetProcAddress(hLib, "Inp32");
   oup32 = (oupfuncPtr) GetProcAddress(hLib, "Out32");

   short x;
   while (!kbhit()) // the program will stop after pressing any keyboard key
   {
```

```
        (oup32)(0xECD8,255); // to send all data port pin high
        x = (inp32)(0xECD8); // read data port
        printf("port read %d\n",x);
    }

    /* Unload the library */
    FreeLibrary(hLib);
    return 0;
}
```

I am not planning to cover 3 credits of C programming but I hope that you can recall whatever you have learned earlier. Happy experimenting!!

4.0 Lab Activities

1. Turn ON LED **1** and **8** and state the digital value.
2. Turn ON LED **3**, **5** and **7** and state the digital value.
3. Turn ON any LED and state the digital value.

References

1. http://www.lvr.com/parport.htm
2. http://www.doc.ic.ac.uk/~ih/doc/par/
3. http://www.epanorama.net/circuits/parallel_output.html
4. http://www.tkk.fi/Misc/Electronics/circuits/lptpower.html