CO326: Industrial Networks

Pre Lab

Lab 01 - Parallel Port I/O (Part I)

Introduction

In computing, the parallel port is an interface used on computers for connecting peripheral devices such as printers (in the early days). In a parallel port, data is sent parallelly; multiple bits of data at once. i.e.: parallel data communication.

Figure 01: Parallel port interface.

Figure 02 shows the pin diagram of the parallel port

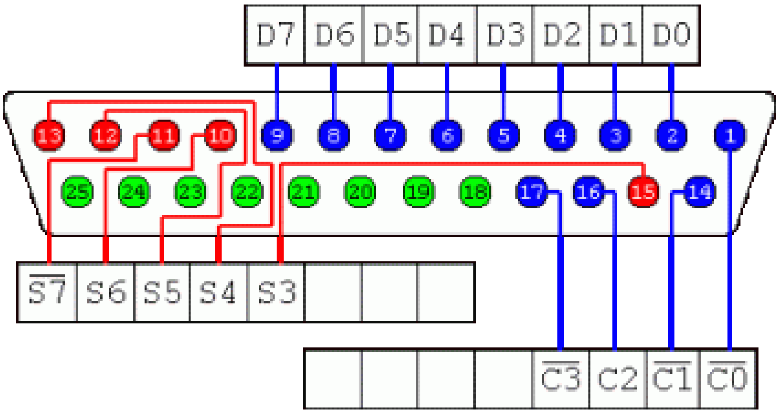

Figure 02: Pin diagram of the parallel port

The D7..D1 Register is the data register and S7..S3 is the status register which we will be using to give outputs and inputs respectively.

As you can see, the pin numbers are not connected to the registers in consecutive order. Also, sometimes they are interchanged, and sometimes they are inverted. You should pay attention carefully to these pins when you program and connect the port into a circuit. You may have to interchange wires and use shift operations/bitwise operations to handle inverted input/output.

Parallel Port Pinout Diagram is as follows (Figure 03)

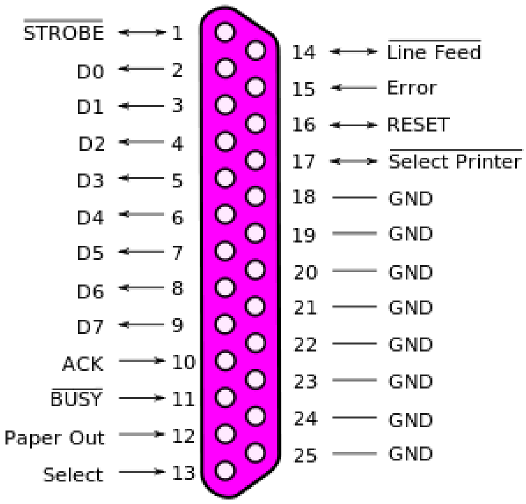

Figure 03: Parallel port pinout

Programming the parallel port

This is a sample program which shows you how to use the parallel port. It reads the value of the status port register and directly writes that value to the data port register, without taking into consideration bit inversions.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/io.h>
#define DATA_PORT 0x378           /* parallel port base address */
#define STATUS_PORT DATA_PORT+1
#define CONTROL_PORT DATA_PORT+2

unsigned char status, data;

void main(){
if (ioperm(DATA_PORT, 1, 1)){
        fprintf(stderr, "Access denied to %x\n", DATA_PORT), exit(1);
}

if (ioperm(STATUS_PORT, 1, 1)){
        fprintf(stderr, "Access denied to %x\n", STATUS_PORT), exit(1);
}

status = inb(STATUS_PORT);
data = status;
outb(data, DATA_PORT);
}
```

ioperm function set port input/output permissions. *inb* does the port input and *outb* does the port output. Please refer to Linux man pages for more information about arguments and return values.

74LS47 IC and 7-Segment display

Figure 2 shows the 74LS47 IC connected to a common anode 7-segment display. By giving binary representation to the input of the IC you can display numbers from 0-9 on the display. Refer to the attached datasheet for the IC for more details.

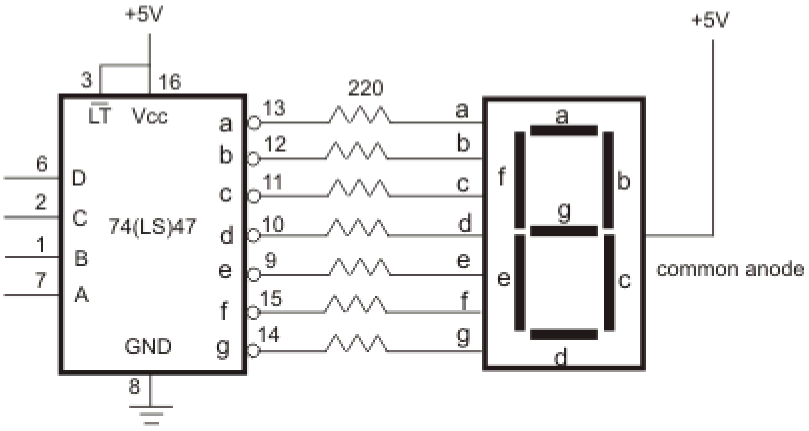

Figure 02: 74LS47 IC connected to common anode 7-segment display

Lab Exercises - Pre Lab

Part 01

1. Draw the circuit diagram that includes a 7-segment display and the data port of the parallel port. **Make sure that you have to connect separate resistors in series for each segment of SSD!**
   Calculate the resistance of the resistors that need to be connected
2. Write a program to light up each segment of SSD one by one.

Part 02: Display 0-9 numbers on a single 7 segment display

1. Differentiate between the common anode and common cathode 7-segment display.
2. Write a program to display characters from 0-9 in an infinite loop with a delay of 1 second between each character.

Part 03: Display 0-9 numbers on a single 7 segment display using 74LS47 IC

3. Draw the circuit diagram that includes a 7-segment display, 74LS47 IC, and the parallel port. Refer to the datasheet of the 74LS47 IC to find the least significant bit of the output. (Use common anode)

4. Write the program to display characters from 0-9 in an infinite loop with a delay of 1 second between each character.

Part 04: Change the numbers displayed in the SSD with a push button

1. Draw the circuit diagram that includes a push button to take inputs, a 7-segment display, and 74LS47 IC to show outputs and the parallel port. Make sure you use proper resistors (pull-up/pull-down) when taking inputs through the push button.

Lab Submission

Create a report including the answers for the Pre Lab exercise. Include all the circuits (take photos), calculations, answers, and code segments in the report in order of the exercise.

Name the report using your group members E numbers. Eg: E20XXX E20XXX.