

## Lab #5 Robot Arm

Note: READ this handout carefully, it contains material that will appear on an upcoming test!

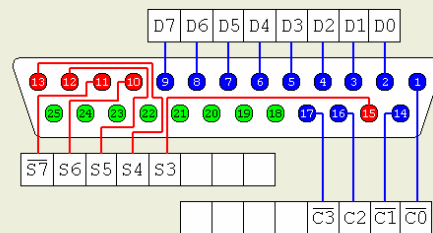
### **Background:**

This lab will use the OWI-007 Robot Arm with Computer Interface. The interface will use the Parallel port of a PC to turn on/off motors using the binary patterns in the table below.

The Original Parallel Port on the IBM PC has the following PINOUT.

The original IBM-PC's Parallel Printer Port had a total of 12 digital outputs and 5 digital inputs accessed via 3 consecutive 8-bit ports in the processor's I/O space.

- 8 output pins accessed via the **DATA Port**
- 5 input pins (one inverted) accessed via the **STATUS Port**
- 4 output pins (three inverted) accessed via the **CONTROL Port**
- The remaining 8 pins are grounded



25-way Female D-Type Connector

**Figure 1**<sup>1</sup>

The Robot Arm uses pins D0..D7 on the parallel port to turn on and off motors for the robot. The bit pattern required to make each individual move is in Table 1 below.

To write a bit pattern to these pins you have to use the OUT op code. The address for the port **MUST** be in register DX. The address for LPT1 Data Port is 378h. The value to write to the port must be in AL. The commands required to write to the port are as follows:

```
Mov dx,378h
Mov al,value_to_write
Out dx,al
```

So if we want to open the gripper we would have to do the following commands:

```
Mov dx,378h           ;Write to LPT1
Mov al,0000_1111b    ;the value to open the gripper
Out dx,al
```

<sup>1</sup> <http://www.doc.ic.ac.uk/~ih/doc/par/>

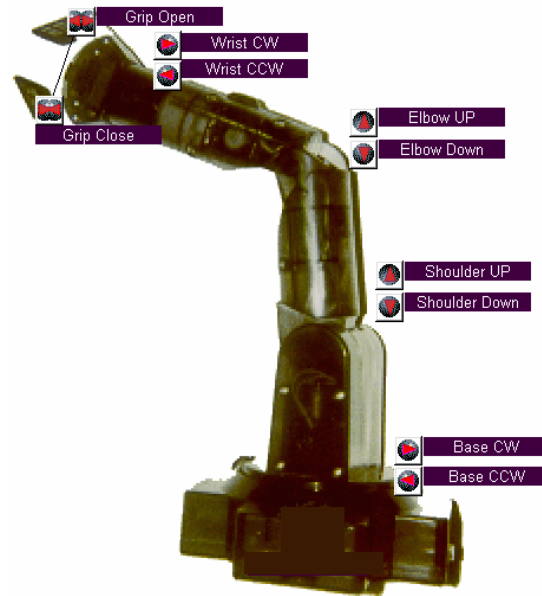


Figure 2<sup>2</sup>

Table 1:

Joint	Motion	Binary Output
Gripper	Open	0000_1111b
	Close	0000_0000b
Wrist	CW	0000_0100b
	CCW	1000_0111b
Elbow	Up	0001_0111b
	Down	0000_0001b
Shoulder	Up	0000_0011b
	Down	0100_0111b
Base	CW	0010_0111b
	CCW	0000_0010b
<b>STOP</b>		<b>0000_0111b</b>

**Time Delay:**

If we just write one value after another to the output port, the robot will only move for a very short period of time before doing the next move (the exact time it will move is the time it takes from the execution of the first OUT instruction to the execution of the next OUT instruction), so we need to add a time delay between each move.

<sup>2</sup> Picture from OWI-007 Robot Arm Win98 software

### **SUB-ROUTINES:**

The best way to accomplish this is by using a SUB-ROUTINE (aka a FUNCTION). A sub-routine is a code segment that can be executed multiple times in the same program and it is called by the following instruction:

*CALL label*

A sub-routine **MUST** be written in such a way as it will **NOT** affect registers required by the code calling it. Usually a sub-routine will place most/all of the registers on the stack before doing anything else and then before it returns to the program that call it, it will restore the registers to their previous values (the exception is when an answer is to be returned to the program, in which case the answer will be returned in a register selected for that function and this will be noted in the header)

For the robot program, a sub-routine to generate a time delay (approximately 5 seconds) has been written for you. This can be downloaded from:

[http://ncatecit.info/spring\\_2008/ect313/labs/time\\_delay.asm](http://ncatecit.info/spring_2008/ect313/labs/time_delay.asm) (or click on link from class webpage)

Just copy and paste this code at the end of your code (header and all) in your program. Then use `CALL TIME_DELAY` everywhere you want the delay to occur (usually after the `OUT` instruction).

### **Assignment:**


Write a program that will do the following sequence of robot arm moves using indirect memory addressing, the `LEA` instruction, a table and a loop (see your notes from this week's class):

Stop  
Wait for a key to be  
pressed\*  
Wrist CW  
Elbow Up  
Gripper Open  
Base CW  
Wrist CCW  
Elbow Down  
Gripper Close  
Base CCW  
Stop  
End the Program<sup>+</sup>

\* see `INT 16H / AH=00` in help / Interrupts (to use this interrupt mov into ah the value of 00 then use `INT 16H` to execute the software interrupt).

<sup>+</sup> to end the program use the `RET` instruction instead of `HLT`. This will return to the DOS prompt.

## NOTES:

- Each student is **required** to do HIS/HER own program. NO COPYING WILL BE TOLERATED!
- Have instructor check your work BEFORE trying it on the hardware (incorrect values being sent to the port can cause harm to the robot circuitry).
- To run the code on the DOS computer with the robot arm, you must use the  button. When asked, save the compiled code to a FLOPPY DISK (supplied by the instructor). Make sure it saves as a .COM file (if it tries to save as a .BIN file you forgot to tell the compiler where to store the program in memory). Once you take it to the DOS computer, boot the computer. Once a c:> prompt appears put in your disk and type A: <enter> to change to the floppy drive. To find your program on the floppy type LDIR <enter> to look at the long directory names (right hand column of output) and their short versions (left hand column). Type the file name in the RIGHT HAND COLUMN. Wait a short period of time, turn on the robot arm power then press a key to start the robot movement.
- Your code should have a proper header and comments (see text chp 8)
- Program is due at the start of next lab via online submission. The program will also be demonstrated to the instructor at that time. **Students who cannot explain how their code works will be given a 0 for the assignment!**