APPENDIX A

Parallel port is a simple and inexpensive tool for building computer controlled devices and projects. The simplicity and ease of programming makes parallel port popular in electronics hobbyist world. The parallel port is often used in Computer controlled robots, Atmel/PIC programmers, home automation ..etc.. Everybody knows what is parallel port, where it can be found, and for what it is being used. The primary use of parallel port is to connect printers to computer and is specifically designed for this purpose. Thus it is often called as printer Port or Getronics port (this name came from a popular printer manufacturing company 'Getronics' who devised some standards for parallel port). You can see the parallel port connector in the rear panel of your PC. It is a 25 pin female (DB25) connector (to which printer is connected). On almost all the PCs only one parallel port is present, but you can add more by buying and inserting ISA/PCI parallel port cards.

Parallel port modes

The IEEE 1284 Standard which has been published in 1994 defines five modes of data transfer for parallel port. They are,

1. Compatibility mode
2. Nibble mode
3. Byte mode
4. EPP
5. ECP

The programs, circuits and other information found in this tutorial are compatible to almost all types of parallel ports and can be used without any problems (Not tested, just because of confidence!). More information on parallel port operating modes can be found here.

Hardware

The pin outs of DB25 connector is shown in the picture below

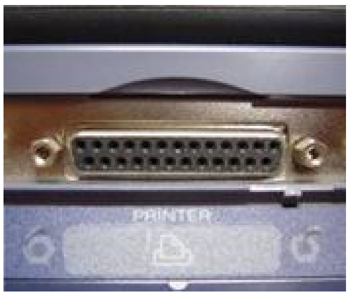


Figure A.1: Parallel Port hardware

The lines in DB25 connector are divided in to three groups:

1. Data lines (data bus)
2. Control lines
3. Status lines

As the name refers, data is transferred over data lines, control lines are used to control the peripheral and of course, the peripheral returns status signals back computer through Status lines. These lines are connected to data, control and status registers internally. The details of parallel port signal lines are given in the next page.

Pin No.	**Signal name**	**Direction**	**Register bit**	**Inverted**
1	nStrobe	Out	Control-0	Yes
2	Data 0	In/Out	Data-0	No
3	Data 1	In/Out	Data-1	No
4	Data 2	In/Out	Data-2	No
5	Data 3	In/Out	Data-3	No
6	Data 4	In/Out	Data-4	No
7	Data 5	In/Out	Data-5	No
8	Data 6	In/Out	Data-6	No
9	Data 7	In/Out	Data-7	No
10	nAck	In	Status-6	No
11	Busy	In	Status-7	Yes
12	Paper out	In	Status-5	No
13	Select	In	Status4	No
14	Linefeed	Out	Control-1	Yes
15	nEror	In	Status-3	No
16	nIntialize	Out	Control-2	No
17	nSelect Printer	Out	Control-3	Yes
18-25	Ground	-	-	-

Parallel port registers

As we know, the Data, Control and status lines are connected to their corresponding registers inside the computer. So by manipulating these registers in program, one can easily read or write to parallel port with programming languages like VC++.

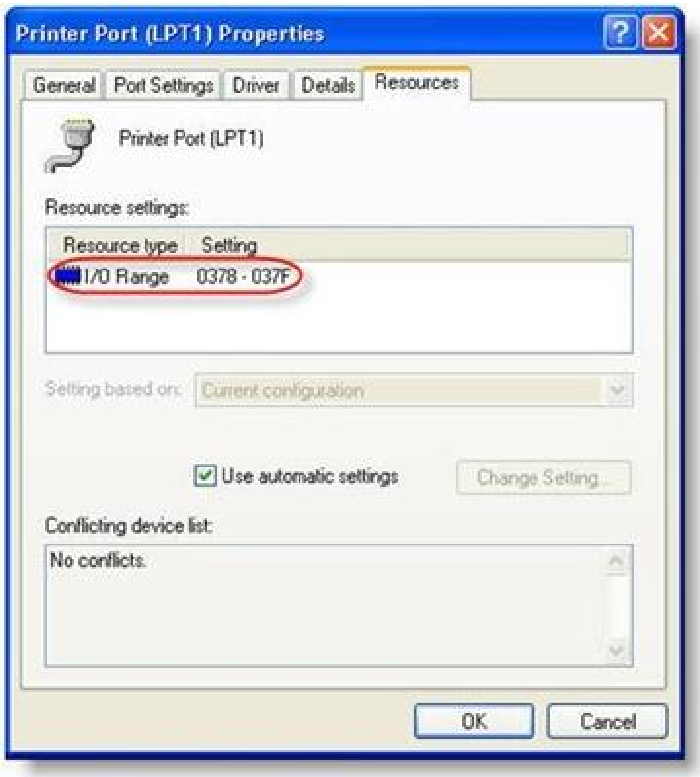
The registers found in standard parallel port are:

1. Data register
2. Status register
3. Control register

As their names specifies, Data register is connected to Data lines, Control register is connected to control lines and Status register is connected to Status lines. (Here the word connection does not mean that there is some physical connection between data/control/status lines. The registers are virtually connected to the corresponding lines.). So whatever you write to these registers, will appear in corresponding lines as voltages, Of course, you can measure it with a multi meter. And whatever you give to Parallel port as voltages can be read from these registers (with some restrictions). For example, if we write '1' to Data register, the line Data0 will be driven to +5v. Just like this, we can programmatically turn on and off any of the data lines and Control lines

Location of registers

Register	LPT1	LPT2
Data register r(Base address+ 0)	0x378	0x278
Status register (Base address+ 1)	0x379	0x279
Control register (Base address + 2)	0x37a	0x27a



Resource settings:

Resource type	Setting
I/O Range	0378 - 037F

Setting based on: Current configuration

☑ Use automatic settings

Conflicting device list:

No conflicts.

Figure A.2: Printer port

Data Register:

Data port has 8 (D0-D7) pins. We can output 8 bit data from here. If we don't send any data to data port, the value is default 00000000.

Status Register

With status register, we can get 5 bit input. (15, 13,12,11,10 pins) The default value of these pins is "1" (+5V). With some buttons we can ground these pins and make them "0".

But as an exception, the pin S7 is default "0" and if we ground it, it will be "1".

INP &h378+1

Control Register

We can use the control register for output and also input. These 4 pins have the default value of “1”. If status and data ports are not enough for us then we can use control port. Usage is same as above.

APPENDIX B

For Information Regarding the Windows NT/2000/XP/Windows 7

Kernel Solutions it is recommended to refer to the website: Http://www.logix4u.net which has plenty of information about this title, which was avoided to not be included in this project for the sake of not rectifying the main subject and to limit the diversity of topics.

-Parallel Port Programming

Visual C++ Parallel Port Programming: Visual C++ Output to the parallel port can be accomplished in Visual C++ with the help of the header library "conio.h". This library contains a function, _outp, which can write data to a given port. The code below takes an input from the user and writes it to the parallel port.

```
#include <conio.h>
#include <stdio.h>
int _outp(unsigned short port, int databyte);
// this program accepts an input from the user
// in decimal and outputs that number as an 8-bit
// binary number to the port at 378 hex, usually
// LPT1
int main ( ) {
int inval = 0;
while ( inval < 256) {
printf("Enter a value in decimal (256 to quit)>");
scanf("%d", &inval);
_outp(0x378, inval);
}
_outp(0x378, 0);
return 0;
}
```