

Abstract

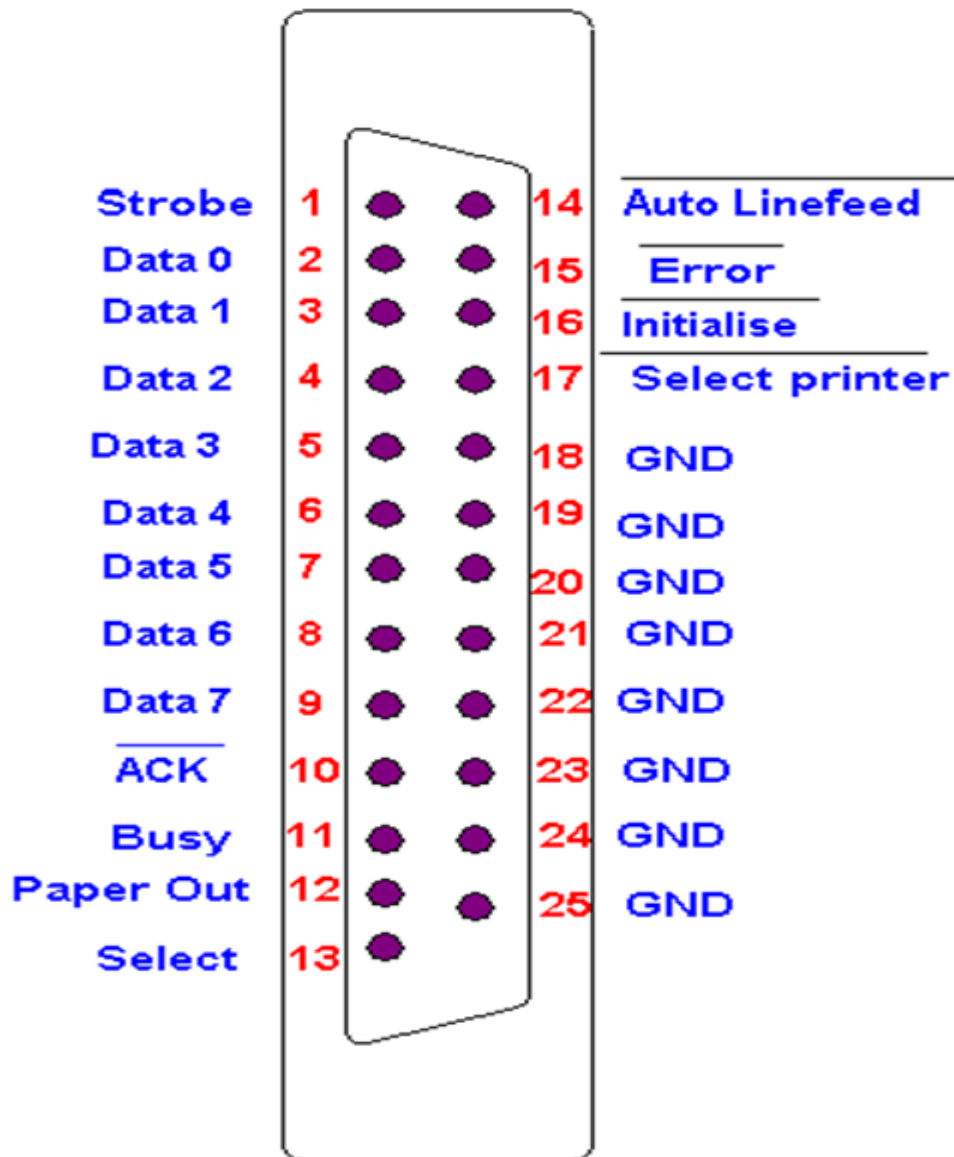
This article presents a simple way to interface electronic hardware with normal PCs. The algorithm stated will allow us to talk with the hardware via LPT port.

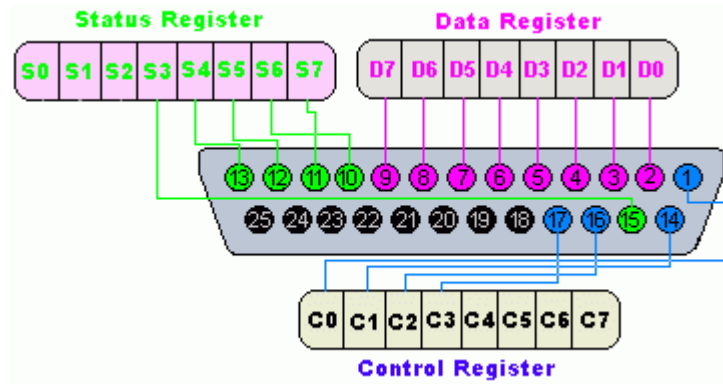
Parallel port programming is easy with C or C++. The keyboard signals (or keystrokes) will be captured and propagated to the parallel port.

1. Introduction

Parallel ports can be very useful in connect our own designed circuit but cautions should be taken as this port can get damaged very easily.

Parallel Port



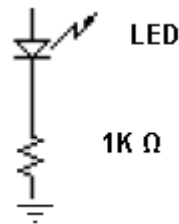


Parallel port is a 25 pin female connector, normally used to connect printer to the PC.. There are 3 sub categories of parallel port pins.

- Data registers : Responsible for data input/output.
- Status registers : Input only port, responsible for notifying the current status
- Control registers : used to send control signals.

Pin No (D-Type 25)	SPP Signal	Direction In/out	Register.bit
1*	nStrobe	In/Out	Control.0
2	Data 0	In/Out	Data.0
3	Data 1	In/Out	Data.1
4	Data 2	In/Out	Data.2
5	Data 3	In/Out	Data.3
6	Data 4	In/Out	Data.4
7	Data 5	In/Out	Data.5
8	Data 6	In/Out	Data.6
9	Data 7	In/Out	Data.7
10	nAck	In	Status.6
11*	Busy	In	Status.7
12	Paper-Out / Paper-End	In	Status.5
13	Select	In	Status.4
14*	nAuto-Linefeed	In/Out	Control.1
15	nError / nFault	In	Status.3
16	nInitialize	In/Out	Control.2
17*	nSelect-Printer/ nSelect-In	In/Out	Control.3
18 - 25	Ground	Gnd	

Circuit Design



The anode needs to be connected to any of the data registers (pin 2 to pin 9) and the cathode pin of LED should be connected to a 1k ohm resistor and can be grounded using any of the pins (18 to 25).

2. Analysis

Windows NT and clones have a strict control over I/O ports. Windows NT/2000/XP raises an exception because they run on protected mode.

Operating Systems on Intel's 80286 onwards starts with real mode and then switched to protected mode or protected virtual address mode. In protected mode there are four rings or privilege modes ranging from 0 to 3 where 0 being most privilege and 3, the least. User mode programs will run in privilege level 3, while device drivers and the kernel will run in privilege level 0 or ring 0. The use of rings monitors the system software to restrict tasks from accessing data or executing privileged instructions. Here, the protected mode will restrict us to talk directly with hardware, so the library is acting as an interface or similar to a device driver.

There are two solutions to solving the problem of I/O access under Windows NT or its clones. The first solution is to write a device driver which runs in ring 0 (I/O privilege level 0) to access your I/O ports but an easier approach will be to use a library(`inpout32.lib`) which can be downloaded free from <http://logix4u.net> with source code.

3. Design



I have used MFC for the purpose. The idea is to capture the arrow keys event both on press and release and accordingly propagate the result to the data registers.

4. Implementation

While using MFC with dialog based, we may require to send request to load the accelerators so that we can capture arrow keys event. Please add the following instruction inside `OnInitDialog()`.

```
m_hAccel = LoadAccelerators(AfxGetResourceHandle(),
MAKEINTRESOURCE(IDR_ACCELERATOR1));
```

Define the parallel port data register as :

```
#define DATA 0x378
```

Now override the virtual function `PreTranslateMessage` for arrow keys event capture.

```
BOOL CKeyMapperDlg::PreTranslateMessage(MSG* pMsg)
{
    //static text feild to update the key press status
    m_lStatus.SetWindowTextW(_T(""));

    int iDataOut = 0;

    if(pMsg->message==WM_KEYDOWN)
    {
        if(pMsg->wParam==VK_UP)
        {
            iDataOut = 1;
            m_lStatus.SetWindowTextW(_T("UP"));
        }
        else if(pMsg->wParam==VK_DOWN)
        {
            iDataOut = 2;
            m_lStatus.SetWindowTextW(_T("DOWN"));
        }

        if(pMsg->wParam==VK_LEFT)
        {
            iDataOut = 4;
            m_lStatus.SetWindowTextW(_T("LEFT"));
        }

        if(pMsg->wParam==VK_RIGHT)
        {
            iDataOut = 8;
            m_lStatus.SetWindowTextW(_T("RIGHT"));
        }
        //Library function to write to the data port
        Out32(DATA, iDataOut);
    }
    else if(pMsg->message==WM_KEYUP)
        Out32(DATA, 0);

    return 0;
}
```

Connect the circuit and pin no. 2 to pin no. 5 will glow on arrow key press and off on release.