USING THE PC TO DISPLAY THE OPTICAL ENCODER RESULT

Prepared By
ENG: Ahmed Saeed Eladly

2009
The measuring of the motor speed using optical encoder requires decoder to interpret the output of optical encoder. This output is a digital signal. By understanding this signal, it is easy to measure the position, speed and acceleration for the electrical motor. This decoder may be hardware or software. Using the personal computer to be a decoder for optical encoder is good idea, because it is easy to create computer program to display the result of optical encoder(position, speed and acceleration), and the personal computer is available in any lab, so no need to buy much component to create this decoder. This report will explain the shaft encoder types, parallel port layout, parallel port programming in visual basic and parallel port interface design.
ACKNOWLEDGMENT

I would like to thank all engineers and technicians who help me to produce this project. Specially the sensor lab technicians. Thanks are also extended to Eng: Mohammed Arfaa (the manager of mechatronics department) and I wish to work together for another project in the near future.
CONTENTS

Introduction..................................................................................................................... 1

Chapter 1 : Rotary Encoder

- 1.1. Absolute rotary encoder ................................................................. 4
  - 1.1.1 Construction .............................................................................. 4
    - 1.1.1.1 Mechanical Absolute Encoders ................................. 5
    - 1.1.1.2 Optical Absolute Encoders ........................................... 5
  - 1.1.2 Standard binary encoding ....................................................... 5
  - 1.1.3 Gray encoding ........................................................................... 7
- 1.2. Single-track absolute rotary encoder ............................................. 8
  - 1.2.1 Encoder output formats ......................................................... 9
- 1.3. Incremental rotary encoder ............................................................ 9
- 1.4. Sine wave encoder .............................................................. 11
- 1.5. Encoder technologies ............................................................... 11

Chapter 2 : 25 pin parallel port layout

- 2.1. Parallel port ABCs ........................................................................ 13
- 2.2. Types of Parallel ports ................................................................. 13
- 2.3. Parallel port devices ............................................................... 14
- 2.4. Layout ......................................................................................... 14

Chapter 3 : parallel port programming in visual basic

- 3.1. Reference to Address Printer Port .............................................. 17
- 3.2. Start programming ...................................................................... 18
  - 3.2.1 Output Port ............................................................................. 18
  - 3.2.2 Input Port ................................................................................. 18
  - 3.2.3 Output Port - Private Declare Function ................................ 18
  - 3.2.4 Input Port - Private Declare Function ............................... 19
- 3.3. Programming by using Function Out and Inp ......................... 19
- 3.4. Sending output signal ............................................................... 20

ENG.Ahmed Saeed Eladly
Chapter 4: parallel port interface

- 4.1. Introduction ................................................................. 22
- 4.2. Parts List and Potential Vendor Source .......................... 23
- 4.3. Theory of Operation ..................................................... 24
- 4.4. Circuit Construction ..................................................... 26
- 4.5. QBasic Programming .................................................... 31
  - 4.5.1 Using the Control lines as Additional Digital Output 31
  - 4.5.2 Using the Status lines for Digital Input ................. 32
INTRODUCTION

The measuring of the motor speed using optical encoder requires decoder to interpret the output of optical encoder. This output is a digital signal. By understanding this signal, it is easy to measure the position, speed and acceleration for the electrical motor. In sensor lab there is no decoder to read the output and display the result, but the students read the output of the encoder by using the oscilloscope then interpret the signal to measure the position, speed and acceleration. In real case there is decoder to display the result of optical encoder, and since, the lab trains this student to be a technician, so it is necessary to create a sample for decoder.

This decoder may be hardware or software. Using the personal computer to be a decoder for optical encoder is good idea, because it is easy to create computer program to display the result of optical encoder(position, speed and acceleration), and the personal computer is available in any lab, so no need to buy much component to create this decoder.

To display the result of optical encoder by using the personal computer, it is necessary to follow the following steps:

Step (1):
Connecting the optical encoder output signal to personal computer:

It is necessary to make a communication between optical encoder and personal computer by computer ports such as parallel port.

Parallel port can receive the encoder output signal and pass it to computer processor, so it is necessary to know how parallel port receives data from optical encoder and study parallel port layout. Figure (1) shows the parallel port layout.
The parallel port layout

Since the optical encoder have four outputs signal (A, B, C, D)
So output (A) will be connected to pin 10
, output (B) will be connected to pin 11
, output (C) will be connected to pin 12
And output (D) will be connected to pin 13
Figure (2) shows the output signal from encoder and corresponding parallel port pin.
Step (2):

Creating a computer program:

This program will read the optical encoder signal from parallel port and process this data according to the program instructions and algorithm then display the result (position, speed and acceleration) on the computer screen.

Visual basic is good programming language, because it is visual language, that means the program can give visual comment, error massage, advice massage, warning massage and easy to read the result.

Step (3):

Design interface:

It is necessary to design interface between the personal computer and optical encoder to protect the computer from dangerous signal (high volt signal) and to make the optical encoder signal suitable to computer input signal.

Since the output of optical encoder is 5 volt and the computer input signal in 5 volt too, so this interface will be for protect the computer from dangerous signal only.

Now, after doing these 3 steps the personal computer will be able to display the result of the optical encoder.
A rotary encoder, also called a shaft encoder, is an electro-mechanical device that converts the angular position of a shaft or axle to an analog or digital code, making it an angle transducer. Engineers use rotary encoders in many applications that require precise shaft rotation—including industrial controls, robotics, expensive photographic lenses, computer input devices (such as optomechanical mice and trackballs), and rotating radar platforms. There are two main types: absolute and incremental (relative).

1.1. Absolute rotary encoder

1.1.1 Construction

The absolute digital type produces a unique digital code for each distinct angle of the shaft. They come in two basic types: optical and mechanical.

Figure 1.1(rotary encoder)
1.1.1.1 Mechanical Absolute Encoders

A metal disc containing a set of concentric rings of openings is fixed to an insulating disc, which is rigidly fixed to the shaft. A row of sliding contacts is fixed to a stationary object so that each contact wipes against the metal disc at a different distance from the shaft. As the disc rotates with the shaft, some of the contacts touch metal, while others fall in the gaps where the metal has been cut out. The metal sheet is connected to a source of electric current, and each contact is connected to a separate electrical sensor. The metal pattern is designed so that each possible position of the axle creates a unique binary code in which some of the contacts are connected to the current source (i.e. switched on) and others are not (i.e. switched off).

1.1.1.2 Optical Absolute Encoders

The optical encoder's disc is made of glass with transparent and opaque areas. A light source and photo detector array reads the optical pattern those results from the disc's position at any one time.

This code can be read by a controlling device, such as a microprocessor, to determine the angle of the shaft.

The absolute analog type produces a unique dual analog code that can be translated into an absolute angle of the shaft (by using a special algorithm).

1.1.2 Standard binary encoding

Rotary encoder for angle-measuring devices marked in 3-bit binary. The inner ring corresponds to Contact 1 in the table. Black sectors are "on". Zero degrees are on the right-hand side, with angle increasing counterclockwise. An example of a binary code, in an extremely simplified encoder with only three contacts, is shown below.
### Standard Binary Encoding

<table>
<thead>
<tr>
<th>Sector</th>
<th>Contact 1</th>
<th>Contact 2</th>
<th>Contact 3</th>
<th>Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>off</td>
<td>off</td>
<td>off</td>
<td>0° to 45°</td>
</tr>
<tr>
<td>2</td>
<td>off</td>
<td>off</td>
<td>on</td>
<td>45° to 90°</td>
</tr>
<tr>
<td>3</td>
<td>off</td>
<td>on</td>
<td>off</td>
<td>90° to 135°</td>
</tr>
<tr>
<td>4</td>
<td>off</td>
<td>on</td>
<td>on</td>
<td>135° to 180°</td>
</tr>
<tr>
<td>5</td>
<td>on</td>
<td>off</td>
<td>off</td>
<td>180° to 225°</td>
</tr>
<tr>
<td>6</td>
<td>on</td>
<td>off</td>
<td>on</td>
<td>225° to 270°</td>
</tr>
<tr>
<td>7</td>
<td>on</td>
<td>on</td>
<td>off</td>
<td>270° to 315°</td>
</tr>
<tr>
<td>8</td>
<td>on</td>
<td>on</td>
<td>on</td>
<td>315° to 360°</td>
</tr>
</tbody>
</table>

Table 1.1: Standard Binary Encoding

In general, where there are $n$ contacts, the number of distinct positions of the shaft is $2^n$. In this example, $n$ is 3, so there are $2^3$ or 8 positions.

In the above example, the contacts produce a standard binary count as the disc rotates. However, this has the drawback that if the disc stops between two adjacent sectors, or the contacts are not perfectly aligned, it can be impossible to determine the angle of the shaft. To illustrate this problem, consider what happens when the shaft angle changes from 179.9° to 180.1° (from sector 4 to sector 5). At some instant, according to the above table, the contact pattern changes from off-on-on to on-off-off. However, this is not what happens in reality. In a practical device, the
contacts are never perfectly aligned, so each switches at a different moment. If contact 1 switches first, followed by contact 3 and then contact 2, for example, the actual sequence of codes is:

- off-on-on (starting position)
- on-on-on (first, contact 1 switches on)
- on-on-off (next, contact 3 switches off)
- on-off-off (finally, contact 2 switches off)

Now look at the sectors corresponding to these codes in the table. In order, they are 4, 8, 7 and then 5. So, from the sequence of codes produced, the shaft appears to have jumped from sector 4 to sector 8, and then gone backwards to sector 7, then backwards again to sector 5, which is where we expected to find it. In many situations, this behaviour is undesirable and could cause the system to fail. For example, if the encoder were used in a robot arm, the controller would think that the arm was in the wrong position, and try to correct the error by turning it through 180°, perhaps causing damage to the arm.

### 1.1.3 Gray encoding

Rotary encoder for angle-measuring devices marked in 3-bit binary-reflected Gray code (BRGC). The inner ring corresponds to Contact 1 in the table. Black sectors are "on". Zero degrees is on the right-hand side, with angle increasing anticlockwise.

To avoid the above problem, Gray encoding is used. This is a system of binary counting in which adjacent codes differ in only one position. For the three-contact example given above, the Gray-coded version would be as follows.
In this example, the transition from sector 4 to sector 5, like all other transitions, involves only one of the contacts changing its state from on to off or vice versa. This means that the sequence of incorrect codes shown in the previous illustration cannot happen here.

### 1.2. Single-track absolute rotary encoder

If the designer moves a contact to a different angular position (but at the same distance from the center shaft), then the corresponding "ring pattern" needs to be rotated the same angle to give the same output. If the most significant bit (the inner ring in Figure 1) is rotated enough, it exactly matches the next ring out. Since both rings are then identical, the

### Table 1.2: Gray Coding

<table>
<thead>
<tr>
<th>Sector</th>
<th>Contact 1</th>
<th>Contact 2</th>
<th>Contact 3</th>
<th>Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>off</td>
<td>off</td>
<td>off</td>
<td>0° to 45°</td>
</tr>
<tr>
<td>2</td>
<td>off</td>
<td>off</td>
<td>on</td>
<td>45° to 90°</td>
</tr>
<tr>
<td>3</td>
<td>off</td>
<td>on</td>
<td>on</td>
<td>90° to 135°</td>
</tr>
<tr>
<td>4</td>
<td>off</td>
<td>on</td>
<td>off</td>
<td>135° to 180°</td>
</tr>
<tr>
<td>5</td>
<td>on</td>
<td>on</td>
<td>off</td>
<td>180° to 225°</td>
</tr>
<tr>
<td>6</td>
<td>on</td>
<td>on</td>
<td>on</td>
<td>225° to 270°</td>
</tr>
<tr>
<td>7</td>
<td>on</td>
<td>off</td>
<td>on</td>
<td>270° to 315°</td>
</tr>
<tr>
<td>8</td>
<td>on</td>
<td>off</td>
<td>off</td>
<td>315° to 360°</td>
</tr>
</tbody>
</table>

In this example, the transition from sector 4 to sector 5, like all other transitions, involves only one of the contacts changing its state from on to off or vice versa. This means that the sequence of incorrect codes shown in the previous illustration cannot happen here.

### 1.2. Single-track absolute rotary encoder

If the designer moves a contact to a different angular position (but at the same distance from the center shaft), then the corresponding "ring pattern" needs to be rotated the same angle to give the same output. If the most significant bit (the inner ring in Figure 1) is rotated enough, it exactly matches the next ring out. Since both rings are then identical, the
inner ring can be omitted, and the sensor for that ring moved to the
remaining, identical ring (but offset at that angle from the other sensor on
that ring). Those two sensors on a single ring make a quadrature encoder.

For many years, Torsten Sillke and other mathematicians believed
that it was impossible to encode position on a single track so that
consecutive positions differed at only a single sensor, except for the two-
sensor, one-track quadrature encoder. However, in 1996 Hiltgen, Paterson
and Brandestini published a paper showing it was possible, with several
examples. See Gray code for details.

1.2.1 Encoder output formats

In commercial absolute encoders there are several formats for
transmission of absolute encoder data, including parallel binary, SSI, ISI,
Profibus, CAN DeviceNet, CANopen, Endat and Hiperface, depending
on the manufacturer of the device

1.3. Incremental rotary encoder

An incremental rotary encoder, also known as a quadrature encoder
or a relative rotary encoder, has two outputs called quadrature outputs.
They can be either mechanical or optical. In the optical type there are two
gray coded tracks, while the mechanical type has two contacts that are
actuated by cams on the rotating shaft. The mechanical types requires
debouncing and are typically used as digital potentiometers on equipment
including consumer devices. Most modern home and car stereos use
mechanical rotary encoders for volume. Due to the fact the mechanical
switches require debouncing, the mechanical type are limited in the
rotational speeds they can handle. The incremental rotary encoder is the
most widely used of all rotary encoders due to its low cost: only two
sensors are required.

The fact that incremental encoders use only two sensors does not
compromise their accuracy. One can find in the market incremental
encoders with up to 10,000 counts per revolution, or more.

There can be an optional third output: reference, which happens
once every turn. This is used when there is the need of an absolute
reference, such as positioning systems.

The optical type is used when higher RPMs are encountered or a
higher degree of precision is required.
Incremental encoders are used to track motion and can be used to determine position and velocity. This can be either linear or rotary motion. Because the direction can be determined, very accurate measurements can be made.

They employ two outputs called A & B which are called quadrature outputs as they are 90 degrees out of phase.

The state diagram:

<table>
<thead>
<tr>
<th>Phase</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1.3: Gray coding for c.w and c.c.w rotation

Two square waves in quadrature (clockwise rotation).

The two output wave forms are 90 degrees out of phase, which is all that the quadrature term means. These signals are decoded to produce a count up pulse or a count down pulse. For decoding in software, the A & B outputs are read by software, either via an interrupt on any edge or polling, and the above table is used to decode the direction. For example if the last value was 00 and the current value is 01, the device has moved one half step in the clockwise direction. The mechanical types would be
debounced first by requiring that the same (valid) value be read a certain number of times before recognizing a state change.

If the encoder is turning too fast, an invalid transition may occur, such as 00->11. There is no way to know which way the encoder turned; if it was 00->01->11, or 00->10->11.

If the encoder is turning even faster, a backward count may occur. Example: consider the 00->01->11->10 transition (3 steps forward). If the encoder is turning too fast, the system might read only the 00 and then the 10, which yields a 00->10 transition (1 step backward).

This same principle is used in ball mice to track whether the mouse is moving to the right/left or forward/backward.

Optical tachometer (no quadrature output)
Rotary sensors with a single output are not encoders and cannot sense direction, but can sense RPM. They are thus called tachometer sensors.

1.4. Sine wave encoder
A variation on the Incremental encoder is the Sinewave Encoder. Instead of producing two quadrature square waves, the outputs are quadrature sine waves (a Sine and a Cosine). By performing the arctangent function, arbitrary levels of resolution can be achieved.

1.5. Encoder technologies
Hall-effect quadrature encoder, sensing gear teeth on the driveshaft of a robot vehicle.

Encoders may be implemented using a variety of technologies:

- Conductive tracks. A series of copper pads etched onto a PCB is used to encode the information. Contact brushes sense
the conductive areas. This form of encoder is now rarely seen.

- **Optical.** This uses a light shining onto a photodiode through slits in a metal or glass disc. Reflective versions also exist. This is one of the most common technologies.

- **Magnetic.** Strips of magnetised material are placed on the rotating disc and are sensed by a Hall-effect sensor or magnetoresistive sensor. Hall effect sensors are also used to sense gear teeth directly, without the need for a separate encoder disc.
25 PIN PARALLEL PORT LAYOUT

Chapter

Contents

• 2.1. Parallel port ABCs
• 2.2. Types of Parallel ports
• 2.3. Parallel port devices
• 2.4. Layout

2.1. PARALLEL PORT ABCs

DB25 connector with an 8 bit data bus (Pin 2-7) which is more popularly used for computer printers while is still used for other devices.

The standard length of Printer Parallel cables is a maximum of 15 feet; although there are 50 foot cables, it is not recommended that these cables be used as it can create poor connection and data signals.

2.2. TYPES OF PARALLEL PORTS

Unidirectional - 4-bit standard port which by factory default did not have the capability of transferring data both ways.

Bi-directional - 8-bit standard port which was released with the introduction of the PS/2 port in 1987 by IBM and are still found in computers today. The Bi-directional port is capable of sending 8-bits input and output. Today, on multifunction printers, this port can be referred to as a bi-directional, Centronics, PS/2 type or standard port.

EPP - The Enhanced Parallel Port (EPP) was developed in 1991 by Intel, Xircom and Zenith Data Systems and operates close to ISA bus speed and can achieve transfer rates up to 1 to 2MB/sec of data.
EPP version 1.7 was released in 1992 and later adapted into the IEEE 1284 standard. All additional features are adapted into the IEEE standard.

EPP version 1.9 never existed.

**ECP** - The Enhanced Capabilities Port (ECP), developed by Microsoft and Hewlett-Packard and announced in 1992, is an additional enhanced Parallel port. Unfortunately, with ECP, it requires an additional DMA channel which can cause resource conflicts.

### 2.3. PARALLEL PORT DEVICES

**Printer** - The most common use for the Parallel port.

**Scanner** - Another commonly used parallel device is the Parallel scanner. Parallel scanners are a popular alternative to SCSI scanners because of how easy they are to install.

**External Drives** - Another popular use of the Parallel ports are external drives such as the Iomega Zip Drive, which can be easily removed from one computer and placed onto another.

### 2.4. LAYOUT

![Parallel port layout](http://www.computerhope.com)

Figure 2.1: parallel port layout
<table>
<thead>
<tr>
<th>PIN</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 1</td>
<td>-Strobe</td>
</tr>
<tr>
<td>Pin 2</td>
<td>+Data Bit 0</td>
</tr>
<tr>
<td>Pin 3</td>
<td>+Data Bit 1</td>
</tr>
<tr>
<td>Pin 4</td>
<td>+Data Bit 2</td>
</tr>
<tr>
<td>Pin 5</td>
<td>+Data Bit 3</td>
</tr>
<tr>
<td>Pin 6</td>
<td>+Data Bit 4</td>
</tr>
<tr>
<td>Pin 7</td>
<td>+Data Bit 5</td>
</tr>
<tr>
<td>Pin 8</td>
<td>+Data Bit 6</td>
</tr>
<tr>
<td>Pin 9</td>
<td>+Data Bit 7</td>
</tr>
<tr>
<td>Pin 10</td>
<td>-Acknowledge</td>
</tr>
<tr>
<td>Pin 11</td>
<td>+Busy</td>
</tr>
<tr>
<td>Pin 12</td>
<td>+Paper End</td>
</tr>
<tr>
<td>Pin 13</td>
<td>+Select</td>
</tr>
<tr>
<td>Pin 14</td>
<td>-Auto Feed</td>
</tr>
<tr>
<td>Pin 15</td>
<td>-Error</td>
</tr>
<tr>
<td>Pin 16</td>
<td>-Initialize Printer</td>
</tr>
<tr>
<td>Pin 17</td>
<td>-Select Input</td>
</tr>
<tr>
<td>Pin 18</td>
<td>-Data Bit 0 Return (GND)</td>
</tr>
<tr>
<td>Pin 19</td>
<td>-Data Bit 1 Return (GND)</td>
</tr>
<tr>
<td>Pin 20</td>
<td>-Data Bit 2 Return (GND)</td>
</tr>
<tr>
<td>Pin 21</td>
<td>-Data Bit 3 Return (GND)</td>
</tr>
<tr>
<td>Pin 22</td>
<td>-Data Bit 4 Return (GND)</td>
</tr>
<tr>
<td>Pin 23</td>
<td>-Data Bit 5 Return (GND)</td>
</tr>
<tr>
<td>Pin 24</td>
<td>-Data Bit 6 Return (GND)</td>
</tr>
<tr>
<td>Pin 25</td>
<td>-Data Bit 7 Return (GND)</td>
</tr>
</tbody>
</table>

Table 2.1: parallel port layout

Below is an explanation of each of the above purposes.

**Pin 1** = Data acknowledgement when the signal is low.

**Pin 2 - 9** = Data transfer pins.

**Pin 10** = Acknowledge that the data has finished processing and when the signal is high indicates ready for more.
**Pin 11** = When the signal goes high indicate that the printer has accepted the data and is processing it. Once this signal goes low and Pin 10 goes high will accept additional data.

**Pin 12** = Printer paper jam when signal is high or no signal if printer jam.

**Pin 13** = When high signal printer is indicating that it is on-line and ready to print.

**Pin 14** = When low signal PC has indicated that the printer inset a line feed after each line.

**Pin 15** = Printer sends data to the computer telling it that an error has occurred.

**Pin 16** = When low signal PC has requested that the printer initiate an internal reset.

**Pin 17** = When low signal the PC has selected the printer and should in return prepare for data being sent.

**Pin 18 - 25** = Ground.
I will present the first one, interfacing via Printer Port in order to make understanding and can do by yourself. First of all, you must download File DLL at Application topic, named I/O Port Dll File, after that copy fill DllPort.Dll and paste at Directory Windows\System on your computer.

### 3.1. Reference to Address Printer Port

Using function Inp32 and Out32 extremely need to reference to Address Printer Port in order to verify and send value in each bit of Printer Port that will be divided into 3 parts,

1. Data is the part that send data, 8 bit to Printer (also computer) that we will control signal in each bit in order to connect with printer for reference position of Data Port that will equal to base printer port position.

   LPT1 Data=H378
   LPT2 Data=H278
2. Status is the part that verifies the status of printer such as the printer is error, out of paper, sending signal to work, etc. We can use Status Port to control equipment as Input (+5 Vdc) for reference of Data Port Data Port will equal to the position of base printer port+1.

LPT1 Status=H379 (H378+1)
LPT2 Status=H279 (H278+1)

3. Control is the part that control working of printer such as Strobe, Auto linefeed, Select printer, etc. It can receive and sent data, 6 bit for reference position of Data Port will be equal to the position of base printer port+1

LPT1 Status=H37A(H378+2)
LPT2 Status=H27A(H278+2)

3.2. Start programming

Firstly, you create New Project that chooses Standard EXE, VB Program will create a blank form. After that click menu Project ---> Add Module to create a module file that has type, .BAS in order to give API Function of Dll in module of visual basic for programming that is used several form in Project and for convenient in use with other program. If you will use just a form, you must not create module file, you can declare in form. The type that is used in module is below.

3. 2.1 Output Port

Public Declare Sub Out Lib "DllPort.dll" Alias "Out32" (ByVal PortAddress As Integer, ByVal Value As Integer)

3.2.2 Input Port

Public Declare Function Inp Lib "DllPort.dll" Alias "Inp32" (ByVal PortAddress As Integer) As Integer

If you program just a form, you must use module by changing from Public to Private in the following;

3.2.3 Output Port - Private Declare Function

Private Declare Sub Out Lib "DllPort.dll" Alias "Out32" (ByVal PortAddress As Integer, ByVal Value As Integer)
3.2.4 Input Port - Private Declare Function

Private Declare Function Inp Lib "DllPort.dll" Alias "Inp32" (ByVal PortAddress As Integer) As Integer

3.3. Programming by using Function Out and Inp

After declaring function, this step, you must not be interested in module, but you go to form that you save as .FRM type. First of all, we will make understanding about value that reference to address of printer port by going to View Code’s Mode, declaring variable in form like this:

Dim PortData ' variable for data at port
Const AddressLPT1 =&H378 'Constant for position of Printer Port 1(LPT1) and Data Port
    Const Statusport1 =AddressLPT1+1 ' Constant for position of Status Port(LPT1)
    Const Controlport1 =AddressLPT1+2 ' Constant for position of Control Port (LPT1)
    Const AddressLPT2 =&H278 ' Constant for position of Printer Port 2(LPT2)
    Const Statusport2 =AddressLPT2+1 ' Constant for position of Status Port(LPT2)
    Const Controlport2 =AddressLPT2+2 ' Constant for position of Control Port (LPT2)

So now, we will create form of sending and getting from Parallel Port in program Visual Basic, clicking Control from Control Dialog, is a TextBox, a Label and a Command Button to put in form in the following:

Figure 3.1: I/O parallel port test
For sending value via Parallel Port, you can do like this:

*Call Out (PortAddress,Data)*

For getting value at Parallel Port

*Inp (PortAddress)*

### 3.4. Sending output signal

So now, we will write code in control, starting to send value to control Parallel Port. You must put in Textbox and send value out by clicking at Command Button, for getting value at port will show label. All I mention are written in View Code in Control Command Button.

```vbnet
Private Sub Command1_Click()
    PortData = Val("&H" & Text1.Text) 'count value in Textbox to put in variable PortData
    Call Out(AddressLPT1, PortData) "send to Port
    Label1.Caption = Hex(Inp(AddressLPT1)) "read value from I/O Port LPT1 as Hexadenary
End Sub
```

Sending value of PortAddress at Printer Port =&H378 Data that is value that send to Printer Port by sending by use Hexadenary, but port will transform data as binary numeral system that has 8 bit such as Hexadenary &H0F at Port =000011112 of binary numeral system.

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Hex</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&amp;H01</td>
<td>00000001</td>
</tr>
<tr>
<td>2</td>
<td>&amp;H02</td>
<td>00000010</td>
</tr>
<tr>
<td>3</td>
<td>&amp;H04</td>
<td>00000100</td>
</tr>
<tr>
<td>4</td>
<td>&amp;H08</td>
<td>00001000</td>
</tr>
<tr>
<td>5</td>
<td>&amp;H10</td>
<td>00010000</td>
</tr>
<tr>
<td>6</td>
<td>&amp;H20</td>
<td>00100000</td>
</tr>
<tr>
<td>7</td>
<td>&amp;H40</td>
<td>01000000</td>
</tr>
<tr>
<td>8</td>
<td>&amp;H80</td>
<td>10000000</td>
</tr>
<tr>
<td>All On</td>
<td>&amp;HFF</td>
<td>11111111</td>
</tr>
<tr>
<td>All Off</td>
<td>&amp;H00</td>
<td>00000000</td>
</tr>
</tbody>
</table>

Table 3.1: The table of Hexadenary that control bit that is out via port
The first one, will be signal as TTL about 15 volt, it can use as LED, Transistor, Relay, car’s motor of children, etc. that will have value of LED at 1,2,3,4 in the right side, we may experiment by taking 8 of LED to connect as serie by resistor =1 k ohm, then connect to Printer Port, read Connectors in topic Hardware, menu Parallel Port Detail click here will make you clear about I/O Port Connectors, here, we will use connector of 2-9 as I/O Data and choose one in 18-25 as ground, so we can experiment to transfer with outputs device.

On the picture, I use 3 of LED8, 3 row and switch that this experiment I use with Card I/O Parallel Port IC 8255 that has 9 port of Port I/O or 72 bit that has TTL 0-5 volt. For switch, it has been tested for input signal that I separate into another web page.
4 PARALLEL PORT INTERFACE

Content

- 4.1. Introduction
- 4.2. Parts List and Potential Vendor Source
- 4.3. Theory of Operation
- 4.4. Circuit Construction
- 4.5. QBasic Programming
  - 4.5.1 Using the Control lines as Additional Digital Outputs
  - 4.5.2 Using the Status lines for Digital Input

4.1. Introduction

The Parallel Port Interface Box is a simple device that connects to the IBM PC's parallel (aka printer) port. Plugging this box into the printer port immediately interfaces your PC to the outside world. For instance, you can hook up motors to the box, and write computer programs to control them.

You could interface sensors and turn your computer into a home control unit. For example, you can have motion sensors which can detect when to turn on lights. You can interface a smoke alarm to your computer. The computer can call and play a pre-recorded panic message to 911 when the alarm goes off. You can design a burglar alarm system as well. Almost all PCs have a parallel port. The advantages of using this port are (1) unlike the serial RS-232 port, you can control up to 8

ENG. Ahmed Saeed Elady
devices simultaneously. (2) unlike the AT expansion bus slot, you don't have to open up your computer. This Parallel Interface Box can be built for less than $20 in parts, and within an afternoon.

For this card, Microsoft's **QuickBasic** will be used to do some simple programs.

### 4.2. Parts List and Potential Vendor Source

Below is a part I used for my construction. Additionally, I list the source from which I bought it from, along with the vendor part number and cost. I did some shopping around, and found Digikey a possible single-source for parts. However, I don't find Digikey to be the cheapest place for parts. The 34-pin header and connector, and header, housing and crimps are a bit pricey. Jameco, a cheap source, does not have such parts. These are not critical parts, and you can substitute them for whatever part you might have in your junk box.

<table>
<thead>
<tr>
<th>PART DESCRIPTION</th>
<th>VENDOR PART</th>
<th>PRICE (1995)</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>74367 HEX BUFFER</td>
<td>DIGIKEY DM74LS367-AN-ND</td>
<td>0.95</td>
<td>3</td>
</tr>
<tr>
<td>RED T1 3/4 LED</td>
<td>DIGIKEY #MR3050QT-ND</td>
<td>0.45</td>
<td>8</td>
</tr>
<tr>
<td>YELLOW T1 3/4 LED</td>
<td>DIGIKEY #MR3350QT-ND</td>
<td>0.45</td>
<td>8</td>
</tr>
<tr>
<td>DB25 MALE CONNECTOR</td>
<td>DIGIKEY #225M-ND</td>
<td>0.92</td>
<td>1</td>
</tr>
<tr>
<td>DB25 MALE HOUSING</td>
<td>DIGIKEY #925GP-ND</td>
<td>0.55</td>
<td>1</td>
</tr>
<tr>
<td>220 OHM RESISTORS</td>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>0.100 CENTER HEADERS</td>
<td>E.G. DIGIKEY #WM4000-ND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.100 CENTER HOUSINGS</td>
<td>E.G. DIGIKEY #WM2601-ND</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ENG. Ahmed Saeed Eladly**

23
4.3. Theory of Operation

The parallel port on the IBM PC is a 25-pin female port. The figure and table below describes the pin functions:

![Diagram of parallel port](image)

<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>FUNCTION</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>STROBE</td>
<td>CONTROL</td>
</tr>
</tbody>
</table>
The parallel port has 4 function types for a total of 25 pins: data (8 pins), control (4), status (5) and ground (8). To understand the function of the data, control and status types, consider what happens when you print something on your printer. The printer prints out alphanumeric characters onto paper (thus using the data lines). Sometimes it does a carriage return and linefeed (hence using the control lines). Sometimes, the printer doesn't print because you ran out of paper, or you forgot to have the printer on-line (status lines). Thus the printer has a number of input and output related function types. The 8 data lines are used for 8 digital OUTPUT lines. For example, you can turn on 8 different motors. The 5 status lines are used for 5 digital INPUT lines. Thus you can interface 5 different sensors, like pushbuttons. The 4 control lines can be used for 4 additional digital output lines (thus 4 more motors!).

### 4.3.1 Parallel Port Addresses

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>DATA BIT 0 OUTPUT</td>
</tr>
<tr>
<td>3</td>
<td>DATA BIT 1 OUTPUT</td>
</tr>
<tr>
<td>4</td>
<td>DATA BIT 2 OUTPUT</td>
</tr>
<tr>
<td>5</td>
<td>DATA BIT 3 OUTPUT</td>
</tr>
<tr>
<td>6</td>
<td>DATA BIT 4 OUTPUT</td>
</tr>
<tr>
<td>7</td>
<td>DATA BIT 5 OUTPUT</td>
</tr>
<tr>
<td>8</td>
<td>DATA BIT 6 OUTPUT</td>
</tr>
<tr>
<td>9</td>
<td>DATA BIT 7 OUTPUT</td>
</tr>
<tr>
<td>10</td>
<td>ACKNOWLEDGE STATUS</td>
</tr>
<tr>
<td>11</td>
<td>BUSY STATUS</td>
</tr>
<tr>
<td>12</td>
<td>PE: PAPER TRAY EMPTY STATUS</td>
</tr>
<tr>
<td>13</td>
<td>PRINTER ON-LINE STATUS</td>
</tr>
<tr>
<td>14</td>
<td>AUTO LINEFEED AFTER (CR) CARRIAGE RETURN</td>
</tr>
<tr>
<td>15</td>
<td>PRINTER ERROR STATUS</td>
</tr>
<tr>
<td>16</td>
<td>INITIALIZE PRINTER CONTROL</td>
</tr>
<tr>
<td>17</td>
<td>SELECT/DESELECT PRINTER CONTROL</td>
</tr>
<tr>
<td>18-25</td>
<td>UNUSED/GROUND</td>
</tr>
</tbody>
</table>

Table 4.2: PARALLEL PORT PINOUT
Each device in a computer has an assigned memory address. For example, your CD-ROM occupies an address. So does your hard and floppy drives. The parallel port is no exception. The IBM Technical Reference Manual describes 2 possible addresses for the parallel port. If you have an old PC with a monochrome display adapter (the old Hercules Green Monitor), the address is 3BCH (956 decimal). If you have a computer with a CGA, EGA, VGA or Super VGA display card (most common for computers after 1990) the parallel port address is 378H (888 decimal). This address is important because we can then write appropriate software. This will become clearer in the QBasic Programming Section. If you are not sure about the address of your parallel port consider the following: If your computer is a 286, 386, 486 or Pentium and you have a color monitor (VGA or SVGA), your computer is probably using 378H as the parallel port address.

<table>
<thead>
<tr>
<th>ADAPTOR</th>
<th>DATA</th>
<th>STATUS</th>
<th>CONTROL</th>
</tr>
</thead>
<tbody>
<tr>
<td>NON-MONO E.G. VGA AND SVGA</td>
<td>378h (888d)</td>
<td>379h (889d)</td>
<td>37Ah (890d)</td>
</tr>
<tr>
<td>MONO DISPLAY CARD</td>
<td>3BCh (956d)</td>
<td>3BDh (957d)</td>
<td>3BEh (958d)</td>
</tr>
</tbody>
</table>

Table 4.3: PARALLEL PORT ADDRESS

**NOTE:** The suffix "h" will be used to denote hexadecimal numbers. The suffix "d" will be used for decimal numbers. This table information will be useful later in the QBasic Programming section.

### 4.4. Circuit Construction

#### 4.4.1 Schematic

The figure below describes the basic circuit.
You can click below to download the Adobe Acrobat format of the schematic. I've tried to get a more generic format e.g. GIF or PS but with no success (as seen to above fuzzy picture). I tried to provide a Windows Meta File format but have found bad results e.g. doesn't load into Word well. If you don't know what Acrobat is, visit Adobe to download a copy of its FREE PDF-file viewer.

As one can see there are 8 digital output lines D0 to D7. These are symbolized by a black arrow head. There are also 4 additional output lines: Strobe, LF/CR, Initialize, Select/Deselect. These are symbolized by a hollow arrow head. One should use these digital output lines with some caution. These lines may require some additional programming steps to use. If 8 digital output lines are enough for your application, then try to avoid using these lines. For example, the Strobe line is an important line in itself. It assures that the parallel data on lines D0 to D7 are ready to be sent out simultaneously. One can also see 5 digital input lines: Printer Error, Online, Empty, Acknowledge and Busy. These are symbolized by grey arrow heads. The circuit requires an external 5 V source. This is connected to Vcc (pin 16) and GND (pin 8) of the 74367. The DB-25 connector pin numbers are those of the male connector part. This connector will plug into the PC parallel port. This is emphasized in the figure below.
4.4.2 The 74367 and Current

The circuit makes use of the 74367 Hex Driver Buffer chip. This protects the motherboard from sinking or sourcing too much current. The parallel port can sink or source about 5 mA of current. Thus one cannot directly hook up a motor, for example, across D0 and GND. A motor can easily demand an amp of current! If one would attach a motor directly across D0 and GND, you could possibly blow and damage your motherboard! With the 74367 as a protective buffer, if your application draws or sinks too much current, these chips would blow before surging your motherboard. You can think of these chips like electrical "fuses". These buffer chips are commonly used to interface real world devices to a PC. An additional note: There are many types of 74367 chips. There is the plain 74367. Also there are the 74LS367, 74HCT367 and other TTL variants. The difference between them is voltage that defines a HI and LO signal. I recommend using the 74HCT367. For output, a HI is defined by a minimum voltage of 4.9 V. A LO is defined by a maximum of 0.1 V. For input, a HI is defined by a minimum of 2.0 V and a LO is defined by a maximum of 0.8 V. The HCT series also allows immediate interfacing to CMOS and TTL type chips.

4.4.3 Interface Box

The interface box can be seen in the following photo.
I used 8 red colored binding posts connected to D0 to D7. These posts allow me to hook up an external real-world device such as a transistor very easily. I can use alligator clips or banana plugs, clipping them to these binding posts. I drilled out 8 holes on the Radio Shack cabinet box and inserted these binding posts. I then soldered the binding posts' ends to crimped wires (Digikey part WM2200). I inserted the crimped wires to a housing (Digikey part WM2601-ND). The housing them fits snugly across a header (Digikey part WM4000-ND). Lastly the header is soldered across the D0 to D7 lines of the 74367. One can see the header pins in the following photo.

In my prototype, I did not design for digital inputs, but one can easily use the same procedure of binding posts-crimped wire-housing-header to the 5 status lines.
4.4.4 Interfacing Real World Devices

In the schematic, I show another real world device, an LED and a Switch. One can use a 220 Ohm resistor in series with the LED. I soldered 8 red LEDs to the 8 digital output lines D0 to D7. In the QBasic Programming Section, I will show a QBasic program which will sequence through all 8 LEDs, lighting them up. Also in the schematic is a single-pole-single-throw (SPST) switch. This demonstrates how the circuit can be used for digital input detection. As mentioned previously, you cannot hook up a motor directly across a digital output line and ground. A real world device such as a motor draws too much current. Instead, one would hook up a transistor, like a 2N2222. You can connect the transistor base to a digital line. The Source would be hooked up to the motor's power supply. The transistor's emitter is then hooked to the ground of the 74367 circuit, and the ground of the Motor's power supply. The decision on using a transistor like the 2N2222 would depend on the current draw of the motor. The 2N2222 can handle up to an amp or so. This is fine for, say a Radio Shack 99 cents toy motor.

4.4.5 Additional Construction Tips

As mentioned in the Part List section, there are many optional parts. I used the 34 pin straight lead wirewrap header and connector. I wanted to have a "neat" bus of wires to solder to the DB-25. I also wanted to wirewrap for quick circuit troubleshooting. Of course, there is no real need to use these parts. I listed them as a matter of convenience to the reader. Also, I used a yellow LED for the Strobe output. As one will read in the QBasic programming section, the Strobe is an important signal. I wanted to use a yellow LED to distinguish it from the other 8 red LEDs. The LEDs, and 34 pin connector are seen more clearly in the photo of the constructed circuit board:
4.5. QBasic Programming

The section describes how to light the 8 LEDs connected across the 8 digital output lines D0 to D7. Recall, that the 74367 chips require an external 5 V power supply. After plugging the DB-25 to PC's parallel port, turn on the 5 V power supply. You can now launch QBasic to test the board. Try the following program:

100 REM TESTING THE 8 DIGITAL OUTPUT LINES
110 PORTBASEADDR = 888: REM FOR NON-MONO GRAPHICS CARDS
120 FOR X = 0 TO 255
130 OUT PORTBASEADDR, X
140 FOR DELAY = 1 TO 500: NEXT DELAY
150 NEXT X

If everything has been assembled correctly and the software properly compiled, then one should see the 8 LEDs should "count" from 0 to 255 in a binary fashion. Recall from Table 3, the parallel port address is assigned 888d (378h). We thus can access the 8 digital lines D0 to D7 by writing a number between 0 and 255 decimal. In QBasic, we do this using the OUT statement. In Turbo C, I believe the equivalent statement is OUTPORT. One thing to also notice is how the Strobe line flashes each time data is sent along D0 to D7. It is a kind of syncing signal. Strobe ensures that all data along D0 to D7 are sent at the same time, hence the name parallel port. This is important, especially in timed applications. For example, suppose you made an 8 motor driven robot arm. If you want a synergistic, coordinated motion, you should make sure that all data along D0 to D7 hits all 8 motors at the same time.

4.5.1 Using the Control lines as Additional Digital Outputs

Table 3 gives the address of the control lines as 958d. You can use OUT as well, with the following table:

<table>
<thead>
<tr>
<th>CONTROL BIT</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT 0 - STROBE</td>
<td>LO=NORMAL, HI=D0-D7 STROBED</td>
</tr>
<tr>
<td>BIT 1 - LF/CR</td>
<td>LO=NORMAL, HI=DIGITAL OUTPUT</td>
</tr>
</tbody>
</table>
Table 4.4: CONTROL LINE BIT ASSIGNMENTS

Thus, you use the statement OUT 958, 2 to turn on Bit 1, the LF/CR line for digital output. OUT 958, 6 would turn on Bits 1 and 2.

4.5.2 Using the Status lines for Digital Input

Table 3 notes that the address of the status line is 957d. We can use QBASIC's INP statement to read signals into the computer. INP is used in conjunction with the following table:

<table>
<thead>
<tr>
<th>CONTROL BIT FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT 0-2 NOT USED</td>
</tr>
<tr>
<td>BIT 3 - PRINTER ERROR LO=PRINTER ERROR, HI=NO ERROR</td>
</tr>
<tr>
<td>BIT 4 - ON-LINE LO=NOT ONLINE, HI=PRINTER ONLINE</td>
</tr>
<tr>
<td>BIT 5 - PAPER LO=PAPER, HI=NO PAPER</td>
</tr>
<tr>
<td>BIT 6 - ACKNOWLEDGE LO=DATA SENT, HI=NORMAL</td>
</tr>
<tr>
<td>BIT 7 - BUSY LO=BUSY, HI=NOT BUSY</td>
</tr>
</tbody>
</table>

Table 4.5: STATUS LINE BIT ASSIGNMENTS

Suppose one uses the switch in the schematic, soldered to ONLINE. The QBASIC statement: SWITCHSTATUS = INP(957) would return the status of the switch. That is, if the SPST switch is pressed, then SWITCHSTATUS would return a number 16d. If the switch is not pressed, then SWITCHSTATUS should be 0.

Where to go from here?

Congradulations if you got this far! So what can and cannot you do with this Parallel Port Interface Box? Well, 8 digital output lines (plus...
possibly 4 others), and 5 digital input lines affords you a many possibilities. First can you do? Well, this box is a cheap and quickly constructable unit. It immediately plugs into the back of your PC. For an example real world application, suppose you wanted to make a home security device. You can use 5 input sensors: strain gages (for smashed windows), sound sensors (for loud sounds), heat sensors (for fires), another strain gage (for detecting if someone is standing on your front door's "Welcome" mat, and an infrared emitter/detector sensor (to detect if someone has walked into hallway). You can use the 8 digital lines to do many event driven things. For example, if the heat sensor is triggered, you can use 1 digital output to turn on another circuit to dial the fire department. If the Welcome mat detects someone standing on it, you can use yet another digital output to turn on the porch light. The possibilities are limitless... almost. Lastly, you could hook up the 8 digital outputs to a digital-to-analog (DAC) converter. You can then have analog output, which can give you a voltage range, to say, do motor control! What can't you do? Well, 5 digital inputs is 3 shy of a full byte. This is unfortunate. Most digital-to-analog converters require a full 8 bits. Thus, you cannot use a DAC for analog input. Oh well... The alternative is to have a 4-bit DAC. You will just have less resolution. Best of luck, and enjoy!
1. The Robot Builder's Bonanza by Gordon McComb
2. BOSCH company manual of speed sensor.
3. http://pinouts.ru/ParallelPorts/ParallelECP_pinout.shtml